GMD Report 10

GMD –
Forschungszentrum
Informationstechnik
GmbH

Bernd Pörner, Arnd Steinmetz,
Matthias Hemmje

PuMa - Ein webbasiertes Publikations-Management-System

April 1998

GMD –
Forschungszentrum Informationstechnik GmbH
Schloß Birlinghoven
D-53754 Sankt Augustin
Telefon +49 -2241 -14 -0
Telefax +49 -2241 -14 -2618
http://www.gmd.de

Anschriften der Verfasser/Addresses of the authors:
Bernd Pörner
Riegerplatz 7
D-64289 Darmstadt

Arnd Steinmetz
Matthias Hemmje
Institut für Integrierte Publikations- und Informationssysteme
GMD – Forschungszentrum Informationstechnik GmbH
Dolivostraße 15
D-64293 Darmstadt
E-mail: Arnd.Steinmetz@gmd.de
Matthias.Hemmje@gmd.de

ISSN 1435-2702

Referat

»PuMa - Ein webbasiertes Publikations-Management-System«

In der folgenden Studie wird die Konzeption und Implementierung eines Dokumenten-Management-Systems beschrieben, welches auf dem objektrelationalen Datenbank-Management-System »Illustra Server« basiert. Besondere Merkmale des Prototypen sind:

- Unterstützung von komplexen Datentypen innerhalb eines relationalen Datenbankschemas
- Volltextindizierung von PostScript-Dokumenten.
- Kombination von Attribut- und Volltextsuche
- Universelle Zugriffsmöglichkeit über das World Wide Web

In der Studie werden zunächst Basisbegriffe definiert und erläutert, die zum Verständnis des Themengebietes »Dokumenten-Management« von grundsätzlicher Bedeutung sind. Anschließend daran folgt die Beschreibung von Konzeption und Implementierung des eigentlichen Prototypen.

Abstract

»PuMa - A web-based Publication Management System«

The following study describes the conception and implementation of a document management system based on the object-relational database management system »Illustra Server«. Special features of the prototype are:

- Support of complex data types within a relational database scheme
- Fulltext indexing of PostScript documents
- Combination of attribute and fulltext search
- Universal access via World Wide Web

First, base terms are described which have basic importance for the understanding of the working field »document management«. Subsequently, the description of the prototype's conception and implementation follows.

Inhaltsverzeichnis

1 EINLEITUNG UND MOTIVATION 9

2 GRUNDLAGEN 13

2.1 KOMMUNIKATION 13
2.1.1 Direkte Kommunikation 13
2.1.2 Indirekte Kommunikation 13
2.2 DER DOKUMENTENBEGRIFF 14
2.3 DIE VERWALTUNG VON DOKUMENTEN 15
2.3.1 Dokumenttypen 15
2.3.1.1 Analoge Dokumente 16
2.3.1.2 Digitale Dokumente 16
2.4 WAS IST DOKUMENTEN-MANAGEMENT? 17
2.4.1 Manuelles Dokumenten-Management 17
2.4.2 Elektronisches Dokumenten-Management 18
2.5 DATENHALTUNGSTECHNOLOGIEN 20
2.5.1 Dateisysteme 21
2.5.2 Datenbanksysteme 21
2.5.2.1 Relationale Datenbank-Management-Systeme (RDBMS) 22
2.5.2.2 Objektorientierte Datenbank-Management-Systeme (OODBMS) 23
2.5.2.3 Objektrelationale Datenbank-Management-Systeme (ORDBMS) 24
2.5.2.4 Information-Retrieval-Systeme (IRS) 26
2.5.3 Abschließender Vergleich der unterschiedlichen Systeme 27

3 KONZEPTION 29

3.1 DIE PUBLIKATIONEN DER GMD 29
3.1.1 Selbständige Publikationen 29
3.1.1.1 Fortsetzungswerke 29
3.1.1.1.1 GMD-Spiegel 29
3.1.1.1.2 GMD-Studien 29
3.1.1.1.3 GMD-Berichte 29
3.1.1.1.4 Arbeitspapiere der GMD 29
3.1.1.2 Monographien 30
3.1.2 Unselbständige Publikationen 30
3.1.3 Graue Literatur 30
3.2 BISHERIGE BEREITSTELLUNG DER PUBLIKATIONEN DER GMD 30
3.2.1 PUBLICAT-System der GMD 30
3.2.2 GMD-Schriften-Bestellung über das WWW 31
3.2.3 Publikationsreferenzen des IPSI auf dem WWW 31
3.2.4 VISIT-Diplomarbeiten auf dem WWW 32
3.3 DEFINITION EINES PUBLIKATIONS-MANAGEMENT-SYSTEMS FÜR DAS GMD-IPSI 32
3.3.1 Globale Zugriffsmöglichkeiten 32
3.3.2 Umfangreiche Suchmöglichkeiten 32
3.3.2.1 Text-Retrieval 32
3.3.2.2 Inhaltliche Verknüpfung der Dokumente 33
3.3.3 Downloading der Dokumente 33
3.3.4 Einfache Pflege des Dokumentenbestandes 33
3.4 ZUR VERFÜGUNG STEHENDE SOFTWARE-KOMPONENTEN 33
3.4.1 Das ORDBMS »Illustra Server« 33
3.4.1.1 DataBlade-Technologie 34
3.4.2 Das PLS »Text DataBlade« Modul 34
3.4.2.1 Text-Retrieval mit dem PLS Text DataBlade 35
3.4.2.1.1 Suchverfahren 35
3.4.2.1.1.1 Best-Match-Suche 35

3.4.2.1.1.2 Konzeptsuche ... 35
3.4.2.1.2 Wie wird mit dem PLS Text DataBlade gesucht? ... 36
3.4.2.1.2.1 Verwendung von Suchoperatoren ... 36
3.4.3 Das Illustra »Web DataBlade« Modul ... 38
3.4.3.1 Architektur des Web DataBlade Moduls ... 38
3.4.3.2 Web-DataBlade-Tags ... 39
3.4.3.2.1 MISQL ... 39
3.4.3.2.2 MIVAR ... 40
3.4.3.2.3 MIBLOCK ... 41
3.4.3.2.4 Prozeßfunktionen ... 41
3.4.3.2.5 Webdriver-Umgebungsvariablen ... 42
3.4.4 Adobe Acrobat ... 43
3.4.4.1 Das Dateiformat PDF ... 43
3.4.4.1.1 Graphische Merkmale ... 43
3.4.4.1.2 Hypertextfunktionen ... 43
3.4.4.1.2.1 Querverweise oder Verknüpfungen (Links) ... 43
3.4.4.1.2.2 Lesezeichen oder Bookmarks ... 44
3.4.4.1.2.3 Thumbnails ... 44
3.4.4.1.2.4 Anmerkungen oder Notes ... 44
3.4.4.1.2.5 Article Threads ... 44
3.4.4.1.3 Font-Substitution ... 44
3.4.4.1.4 Datenkompression ... 45
3.4.4.1.5 Dateistruktur ... 45
3.4.4.1.6 Datensicherheit ... 45
3.4.4.2 Erstellung und Bearbeitung von PDF-Dateien ... 45
3.5 ENTWURF EINER DATENBANK FÜR DIE IPSI-PUBLIKATIONEN ... 45
3.5.1 Definition des relevanten Realitätsausschnittes ... 46
3.5.2 Datenmodellierung der IPSI-Miniwelt ... 46
3.5.2.1 Entity-Relationship-Modell ... 46
3.5.2.1.1 Objektklassen ... 47
3.5.2.1.2 Beziehungen ... 47
3.5.2.1.2.1 Spezielle Beziehungstypen ... 48
3.5.2.1.3 Kardinalitätsangaben ... 48
3.5.2.2 Erweiterung des ERM ... 49
3.5.2.2.1 Problemfall 1: Objektklassen »Mitarbeiter« und »externe Person« ... 50
3.5.2.2.2 Problemfall 2: Objektklasse »Publikationen« ... 50
3.5.2.2.3 Abstraktionsbeziehung ... 51
3.5.3 Umsetzung in einen relationalen Datenbankentwurf ... 52
3.5.3.1 Umsetzung der Objektklassen ... 52
3.5.3.2 Umsetzung der Beziehungen ... 54
3.5.3.2.1 Einwertige Beziehungen ... 54
3.5.3.2.2 Mehrwertige Beziehungen ... 54
3.5.3.3 Überprüfung der Normalformeneinhaltung ... 54

4 IMPLEMENTIERUNG ... 57

4.1 ERZEUGUNG DER DOKUMENTENDATENBANK ... 57
4.1.1 Erzeugung der Relationen ... 57
4.1.1.1 Umsetzung der aus den Objektklassen resultierenden Relationen ... 57
4.1.1.2 Umsetzung der Verknüpfungsrelationen ... 59
4.1.2 Erzeugung der Textindizes ... 59
4.1.2.1 Index für Relation gmdReportTable: ... 60
4.1.2.2 Index für Relation monographyTable: ... 60
4.1.2.3 Index für Relation diplomaTable: ... 60
4.1.2.4 Index für Relation articleTable: ... 60
4.2 VORBEREITUNG DER DOKUMENTE ... 61
4.3 EINFÜGEN DER DATENSÄTZE ... 61
4.3.1 gmdReportTable: ... 62
4.3.2 newPublicationPersonnelTable: ... 62
4.3.3 newPublicationActivityTable: ... 62
4.3.4 Zur Verfügung stehende Dokumente ... 63
4.4 ERZEUGUNG EINER GRAPHISCHEN BENUTZERSCHNITTSTELLE ... 63

4.4.1 Suche im Dokumentenbestand 64
4.4.1.1 Eingabe der Suchanfrage 64
4.4.1.1.1 Implementierung der Suchmaske (searchForm) 65
4.4.1.2 Suche der Dokumente und Ergebnisanzeige 67
4.4.1.2.1 Implementierung der Such- und Ergebnisseite (fulltextResult) 68
4.4.1.2.1.1 Formulierung der Suchanfrage 70
4.4.1.2.1.2 Anzeige der Suchanfrage 74
4.4.1.3 Anzeige einzelner Publikationen 76
4.4.1.3.1 Implementierung der Dokumenteneinzelanzeige (publicationDetail) 77
4.4.1.3.1.1 Ermittlung und Anzeige der Autorennamen 77
4.4.1.3.1.2 Ermittlung und Anzeige der Publikationsinformationen 77
4.4.1.3.1.3 Ermittlung der durch Zuweisung verknüpften Aktivitäten 80
4.4.1.3.1.4 Ermittlung der durch inhaltliche Relation verknüpften Aktivitäten 81
4.4.1.3.1.5 Schließen des Detailfensters 81
4.4.1.4 Referenzenanzeige verknüpfter Dokumente 82
4.4.1.4.1 Implementierung der aktivitätsbezogenen Referenzenliste (activityResult) 82
4.4.1.4.1.1 Entfernen der Tabelle zur Anzeige der Suchanfrage 83
4.4.1.4.1.2 Entfernen von Bedingungsblöcken (MIBLOCK) 83
4.4.1.4.1.3 Änderung der MISQL-Blöcke zur Dokumentenermittlung und Anzeige 83
4.4.1.4.1.4 Anzeige des Aktivitätsnamens 84
4.4.2 Hinzufügen von Dokumenten 84
4.4.2.1 Benötigte Arbeitsschritte für das Hinzufügen von Dokumenten 85
4.4.2.2 Implementierung der Systempflegefunktionen 86
4.4.2.2.1 Eingabeformulare 86
4.4.2.2.1.1 Einfügen von Publikationen (pubInsert) 87
4.4.2.2.1.2 Einfügen von GMD-Publikationen (gmdReportInsert) 87
4.4.2.2.1.3 Einfügen einer Person in personnelTable bzw. einer Aktivität in activityTable (personalInsert bzw. activityInsert) 91
4.4.2.2.2 Einfügemechanismen für Personen und Aktivitäten 93
4.4.2.2.2.1 Einfügemechanismus für Personen (personalProc) 93
4.4.2.2.2.2 Einfügemechanismus für Aktivitäten (activityProc) 94
4.4.2.2.3 Einfügemechanismen für Dokumente 95
4.4.2.2.3.1 Speicherung einer neuen Publikation in der jeweiligen Publikationstabelle (publicationProc) 96
4.4.2.2.3.2 Speicherung der Aktivitätszugehörigkeit (actInsert) 98
4.4.2.2.3.3 Speicherung der Autorenschaft (authorInsert1/2) 101
4.4.2.2.3.4 Ermittlung und Speicherung der inhaltlichen Verknüpfung zu Aktivitäten (keywordInsert1/2) 102
4.4.2.2.3.5 Anzeige der eingefügten Daten (insertResult) 103

5 FAZIT UND AUSBLICK 107

5.1 LEISTUNGSMERKMALE DES PROTOTYPEN 107
5.1.1 Umfangreiche Suchmöglichkeiten 107
5.1.2 Dynamische Suchergebnisanzeige 108
5.1.3 Einfache Pflege des Dokumentenbestandes 108
5.1.4 Globaler Zugriff 109
5.2 ZUKUNFTSENTWICKLUNGEN 109
5.2.1 Zugriffsbeschränkungen beim Dokumenten-Download 109
5.2.2 Zugriffsbeschränkungen beim Hinzufügen von Dokumenten 110
5.2.3 Neuimplementierung der Pflegefunktionen 110
5.2.4 Namensänderung der Autoren 111
5.2.5 Aktualisierung von Datensätzen 112
5.2.6 Löschen von Dokumenten 112

ANHANG A - LITERATURVERZEICHNIS 113

ANHANG B - BIBLIOGRAPHIE 115

1 Einleitung und Motivation

Eine wachsende Anzahl von Unternehmen, privaten und öffentlichen Institutionen steht mit zunehmender Zeit der immer schwieriger lösbaren Aufgabe gegenüber, einer stetig anwachsenden Menge von Informationen in Form von Schriftdokumenten Herr zu werden.

Wer nicht Gefahr laufen will, (frei nach dem bekannten Sprichwort) die Information vor lauter Produktdokumentationen, Publikationen, internem und externem Schriftverkehr und Bibliotheksbeständen nicht mehr zu sehen, ist gezwungen, Schriftstücke zu sortieren, kategorisieren und so abzulegen, daß diese im Bedarfsfall wieder aufgefunden werden können.

Bedingt durch den Umstand, daß Textdokumente heutzutage größtenteils mit Hilfe von Computern erstellt werden, stehen diese, neben ihrer Erscheinungsform als papierene Schriftstücke, in digitaler Form zur Verfügung. Der Gedanke liegt somit nahe, die Verwaltung von Textdokumenten elektronisch zu bewältigen.

Die Software-Industrie stellt gegenwärtig eine Vielzahl sogenannter elektronischer Dokumenten-Management-Systeme (EDMS) zur Verfügung. Mit EDMS bekommt der Anwender Werkzeuge zur Hand, die es ihm ermöglichen, digitale Dokumente höchst effizient zu verwalten. Heutige EDMS bieten unter anderem folgende Grundfunktionalitäten:

- Ablage von Dokumenten in einem zentralen Speicher
- Orts- und zeitunabhängiger Zugriff auf diesen Speicher
- Simultaner Zugriff mehrerer Nutzer auf ein Dokument
- Problemloses Wiederauffinden gespeicherter Dokumente durch spezielle Suchmechanismen
- Wirkungsvolle Sicherheitsmechanismen, um Dokumente vor unerlaubtem Zugriff zu schützen

Durch die zusätzliche Verfügbarkeit entsprechender Hardware zur Digitalisierung analoger Dokumente stehen Möglichkeiten zur Verfügung, komplette Papierarchive ebenfalls in EDMS einzubinden.

Doch trotz der umfangreichen Möglichkeiten, die EDMS dem Anwender bieten, bergen sie einen nicht zu unterschätzenden Nachteil:

Ein großer Teil der unternehmerischen bzw. institutionellen Informationsverarbeitung findet bereits seit vielen Jahren mit Hilfe relationaler Datenbank-Management-Systeme (RDBMS) statt. Personalverwaltung, Finanzplanung und –kontrolle, Lagerhaltung und Statistik. Alle Informationen, die sich gut strukturieren lassen, können mittels RDBMS einfach und effizient verarbeitet werden.

Die Einführung eines komplett neuen Systems (wie z. B. einem EDMS) bedeutet für Unternehmen oder Institutionen einen nicht unerheblichen Aufwand. Neben dem Preis für den Kauf und die Installation eines EDMS, müssen noch Kosten eingerechnet werden, die für die Schulung der Anwender und des Entwicklungspersonals auflaufen. Es kann sogar vorkommen, daß speziell wegen der Einführung eines EDMS entsprechend ausgebildete Fachkräfte eingestellt werden müssen. Bevor sich ein EDMS im Arbeitsablauf einer Organisation positiv bemerkbar macht, müssen somit hohe Kosten in Kauf genommen werden.

Nachdem jahrelang keine echte Bewegung im Markt der RDBMS auszumachen war, präsentieren nun die führenden Datenbankhersteller den Nachfolger der RDBMS: die objektrelationalen Datenbank-Management-Systeme (ORDBMS).

ORDBMS bieten einerseits die Vorzüge, die man von RDBMS her kennt (effizienter und einfacher Datenzugriff durch SQL, Transaktions-Management, Anpassung an bestehende Hardware-Umgebungen durch freie Skalierbarkeit).

Andererseits durchbrechen ORDBMS die Grenzen, die RDBMS auferlegt waren: Konnten RDBMS nur strukturierte Daten (Zahlen, Währungsdaten, Datum- bzw. Zeitangaben, kurze alphanumerische Zeichenketten, Boole'sche Werte) verwalten, bieten ORDBMS eine effiziente Verwaltung auch für unstrukturierte, komplexe Daten wie Textdokumente, Bilder, Videostreams, Klangdateien usw.

Wenn ORDBMS mit Texten umgehen können, so bringt dies den Gedanken näher, auch eine komplette Dokumenten-Management-Lösung mit Hilfe von ORDBMS zu realisieren. Die Vorteile einer solchen Lösung liegen auf der Hand:

- Sowohl strukturierte (z. B. Personal- und Finanzdaten) als auch unstrukturierte Daten (z. B. Textdokumente) können mit einem einzigen DBM-System verwaltet werden.
- Da ORDBMS nach wie vor eine Standard-SQL-Schnittstelle vorweisen, muß vorhandenes Datenbank-Entwicklungspersonal in nur relativ geringem Maße umgeschult werden.
- Auf den Kauf und die Einführung einer speziellen Dokumenten-Management-Software könnte verzichtet werden.

Die vorliegende Ausarbeitung beschreibt die Konzeption und die prototypische Implementierung einer solchen, auf einem objektrelationalen DBMS basierenden Dokumenten-Management-Lösung am Beispiel eines Publikations-Management-Systems für das in Darmstadt ansässige Institut für Integrierte Publikations- und Informationssysteme (IPSI) der GMD – Forschungszentrum Informationstechnik GmbH.

Dabei findet unter anderem eine Problematik besondere Beachtung:

Mit Hilfe herkömmlicher String-Match-Retrieval-Verfahren kann der Anwender eines Informationssystems zwar Dokumente finden, die möglichst gut zu seiner eingegebenen Suchanfrage passen. Ob er jedoch wirklich alle für seinen Bedarf interessanten Dokumente gefunden hat, sei dahingestellt. Dazu ein kleines Beispiel:

Ein Anwender sucht in einer (imaginären) Textdatenbank des IPSI nach Publikationen über graphische Information-Retrieval-Schnittstellen für Volltextdatenbanksysteme. Das System liefert ihm Dokumente über das IPSI-Projekt »LyberWorld« zurück. Bei LyberWorld handelt es sich um eine dreidimensionale Benutzungsschnittstelle für Information-Retrieval-Systeme. In einem Dokument ist ein Hinweis enthalten, daß LyberWorld schon zusammen mit dem objektorientierten DBMS VODAK getestet wurde. VODAK ist ebenfalls ein IPSI-Projekt, über das ebenfalls innerhalb der IPSI-Textdatenbank Publikationen enthalten sind. Man könnte also sagen, daß durch die Dokumentation des gemeinsame Tests beider Projekte innerhalb einer LyberWorld-Publikation ein inhaltlicher Zusammenhang zwischen dieser Publikation und dem Projekt VODAK besteht. Für den Anwender könnte es unter Umständen interessant sein, auf diesen Sachverhalt hingewiesen zu werden und eine Möglichkeit geboten zu bekommen, die Publikationen des VODAK-Projektes einzusehen. Mittels String-Match-Retrieval ist es jedoch nicht möglich, solche inhaltlichen Verknüpfungen aufzufinden.

Der in dieser Ausarbeitung beschriebene Prototyp zeigt eine Möglichkeit auf, wie derartige thematische Verknüpfungen zwischen Publikationen und Forschungsaktivitäten auf zwar einfache jedoch wirkungsvolle Weise zu realisieren sind.

Die Ausarbeitung gliedert sich in folgende Abschnitte:

In *Kapitel 2* werden Begriffe erläutert, die für das grundlegende Verständnis von Dokumenten-Management notwendig sind. Es findet ein Vergleich zwischen verschiedenen Ausprägungen des Dokumenten-Management statt und es wird definiert, welche Funktionalitäten ein elektronisches Dokumenten-Management-System vorweisen sollte. Außerdem werden unterschiedliche Datenhaltungstechnologien beschrieben.

Den Anfang von *Kapitel 3* bildet eine Untersuchung darüber, welche Publikationsarten innerhalb der GMD produziert werden und wie diese zum gegenwärtigen Zeitpunkt der Öffentlichkeit zugänglich gemacht werden. Darauf aufbauend wird erörtert, welche Funktionalitäten das zu konzipierende Publika-

tions-Management-System vorweisen sollte. Im Anschluß daran erfolgt die Konzeption der dem Publikations-Management-System zugrundeliegenden Datenbank. Den Abschluß des Kapitels bildet die Beschreibung der zur Implementierung zur Verfügung stehenden Software-Komponenten.

Kapitel 4 beschreibt die Implementierung des Prototypen. Diese teilt sich in mehrere Abschnitte auf: Den Anfang bildet die Erstellung der dem System zugrundeliegenden Datenbank und die Speicherung eines Testdatenbestandes. Daran anschließend erfolgt die Realisierung der graphischen Benutzungsoberfläche, die es dem Anwender ermöglicht, innerhalb des Dokumentenbestandes zu suchen und einzelne Dokumente, inklusive inhaltlichen Verknüpfungen, detailliert anzeigen zu lassen. Das Schlußlicht des Kapitels wird durch die Beschreibung der Systempflegefunktionen gebildet, die größtenteils vollautomatisch vom System übernommen werden.

In *Kapitel 5* wird eine Betrachtung des Prototypen vorgenommen und auf seine Funktionalitäten eingegangen. Außerdem erfolgt ein Ausblick auf die mögliche Zukunft des Prototypen. Es wird beschrieben, welche Komponenten noch implementiert werden müssen, um aus dem prototypischen System ein in der Praxis einsetzbares Publikations- bzw. Dokumenten-Management-System erwachsen zu lassen.

2 Grundlagen

2.1 Kommunikation

Wollen Menschen untereinander Informationen austauschen, so kommunizieren sie miteinander. Die grundlegenden Komponenten der Kommunikation sind der Sender, die zu übertragende Information und der Empfänger. Damit die vom Sender abgesetzte Information beim Empfänger ankommt bzw. von diesem verstanden wird, müssen drei grundlegende Voraussetzungen erfüllt sein:

- Die zu vermittelnde Information muß vom Sender in ein kommunizierbares Zeichensystem umgewandelt werden (z. B. Sprache oder Schrift).
- Die Zeichen müssen in physikalische Signale transformiert werden (z. B. Schallwellen) und über ein bestimmtes Medium übertragen werden (z. B. Luft oder Telefonleitung).
- Der Empfänger muß die übermittelten Zeichen deuten und durch Interpretation die ihm vermittelte Bedeutung erschließen.

Grundsätzlich unterscheidet man zwei verschiedene Kommunikationsarten:

2.1.1 Direkte Kommunikation

Der Prozeß der direkten Kommunikation umfaßt die zwischenmenschliche Kommunikation von Angesicht zu Angesicht mittels Sprache, Mimik, Ausdruck usw. als Verständigungsmittel. Der Sender kommt hierbei in direkten Kontakt zum Empfänger; somit ist dem Sender der Empfänger bekannt. Einziges Übertragungsmedium ist die Luft, über die Schall- bzw. Lichtwellen transportiert werden.

2.1.2 Indirekte Kommunikation

Jegliche Kommunikation, die, aus welchen Gründen auch immer, nicht von Angesicht zu Angesicht abgewickelt wird, kann als indirekte Kommunikation bezeichnet werden. Die indirekte Kommunikation kann in zwei unterschiedlichen Ausprägungen erfolgen:

- **Ohne Zeitaufschub.** Hierbei werden die Informationen vom Sender über technische Übertragungssysteme (Telefon, CB-Funk, Fernsehen, Radio) zum Empfänger geleitet. Der durch Signalumwandlung und –übertragung verursachte Zeitverlust bewegt sich meist im Sekundenbereich und ist daher eher vernachlässigbar.
- **Mit Zeitaufschub.** Informationen können aber auch zeitunabhängig übertragen werden. Dies geschieht, indem der Sender die Informationen auf einem Medium abspeichert (Erstellung eines Dokumentes) und dieses Dokument an den Empfänger weiterleitet. Der Empfänger kann dem Dokument dann die übermittelten Informationen entnehmen.

Charakteristisch für die indirekte Kommunikation ist, daß der Sender nicht unbedingt den Empfänger kennt. Am Beispiel eines Nachrichtensprechers, der möglicherweise viele Millionen Zuschauer bzw. Zuhörer hat, wird dieser Sachverhalt besonders deutlich.

2.2 Der Dokumentenbegriff

»**Dokument**, das; -[e]s, -e [mlat. documentum = beweisende Urkunde < lat. documentum = das zur Belehrung über etw. bzw. zur Erhellung von etw. Dienliche, Beweis, zu: docere = lehren ...«[1]

»**Dokumentation** ... Als Dokumente können alle Unterlagen betrachtet werden, die Informationen enthalten, also nicht nur publiziertes Wissen, sondern auch Briefe, Akten, Urkunden, Bildsammlungen, Filme u.v.a. ...«[2]

Ein Dokument ist eine auf einem materiellen Träger fixierte Information, in Schrift- oder Bildform, in akustischer oder einer anderen körperlich manifesten Form. Das Material, aus dem der Träger besteht, kann vielfältig sein: Papier, magnetisierbare Stoffe (Ton- bzw. Videoband, Disketten, Festplatten), fotografische Stoffe (Diapositiv- bzw. Negativfilm) u.v.a.

Die Kommunikation über Dokumente bietet im Vergleich zum direkten Informationsaustausch erhebliche Vorteile:

- **Dokumente lassen sich archivieren**. Mittels Dokumenten ist es möglich, Wissen aufzuzeichnen und somit dauerhaft zu erhalten. Ohne die Verwendung von Dokumenen wäre Wissen lediglich in den Gehirnen bestimmter Personen gespeichert. Beim Tod dieser Menschen wäre diese Wissen unwiederbringlich verloren, es sei denn, sie hätten ihr Wissen an andere Menschen weitergegeben.

- **Dokumente sind personenunabhängig.** Wieder angenommen, Wissen wäre lediglich in den Gehirnen von Menschen gespeichert, so wäre eine Wissensübermittlung nur in Form direkter Kommunikation (mündliche Überlieferung, Gestensprache usw.) möglich. Bei dieser Art der Informationsübermittlung besteht jedoch die Gefahr, daß bestimmte Teile der ursprünglichen Nachricht oder Information weggelassen oder neue (womöglich falsche bzw. irreführende) Teile hinzugefügt werden. Die Folge könnte sein, daß der ursprüngliche Informationsgehalt verfälscht oder gar völlig unbrauchbar geworden ist. Sehr gut deutlich (wenn auch etwas überspitzt) wird die Unzuverlässigkeit mündlicher Überlieferung bei dem Spiel »Stille Post«:
  Eine Person flüstert einer anderen einen Satz ins Ohr. Diese gibt die empfangene Nachricht wieder flüsternd an eine andere Person weiter. Ist die Nachricht bei der letzten Person angekommen, so spricht diese die Nachricht laut aus. Die erste Person der Kommunikationskette spricht die ursprüngliche Nachricht laut aus. Ergebnis ist meistens, daß die Nachricht umso verfälschter ist, je mehr Übermittlungsinstanzen zwischen Anfang und Ende der Kommunikationskette standen.
  Dokumente hingegen bieten die Möglichkeit, Wissen wortgetreu aufzuzeichnen und weiterzugeben.

- **Dokumente sind mengenunabhängig.** Auf dem direkten Kommunikationsweg lassen sich nur begrenzte Mengen an Information übermitteln. Dies liegt einerseits daran, daß sich Gespräche von Angesicht zu Angesicht nicht endlos ausdehnen lassen. Andererseits sind der Aufnahmefähigkeit des menschlichen Gehirns natürliche Grenzen gesetzt. Komplexe und umfangreiche Informationen sind somit für die direkte Kommunikation so gut wie ungeeignet. In Dokumenten hingegen lassen sich beliebig viele beliebig komplexe Informationen unterbringen.

- **Dokumente sind standortunabhängig.** Der Sender kann ein Dokument verfassen, welches später an einen viele Kilometer entfernten Empfänger weitergeleitet wird. Zwar ist es durch moderne Telekommunikationssysteme (z. B. Telefon, Video-Conferencing) möglich, große Entfernungen zu überbrücken, doch besteht dabei immer noch das weiter oben dargestellte Mengen- bzw. Komplexitätsproblem.

Auch wenn die Kommunikation über Dokumente viele Vorteile gegenüber der direkten Kommunikation bietet, birgt sie auch ein nicht zu unterschätzendes Manko:

---

[1] Duden, Das große Wörterbuch der deutschen Sprache. Mannheim: Bibliographisches Institut (1976)

[2] Brockhaus Enzyklopädie in zwanzig Bänden, Wiesbaden: F. A. Brockhaus (1968)

Wenn Menschen direkt miteinander kommunizieren, so wird dabei wesentlich mehr ausgetauscht als bloße Worte. Wichtige Komponenten des Gespräches von Angesicht zu Angesicht sind neben den gesprochenen Worten auch Gesichtsausdrücke, Stimmlagen, Gesten, Berührungen usw. Bislang ist es noch nicht gelungen, diese Kommunikationskomponenten in all ihrer Vielfalt und ihrem Zusammenspiel zufriedenstellend zu dokumentieren.

2.3 Die Verwaltung von Dokumenten

Je mehr Wissen bzw. Information in Dokumentenform fixiert wird, desto wichtiger wird es, Dokumente so zu verwalten, daß problemlos und schnell auf sie zurückgegriffen werden kann. Diese Dokumentenverwaltung muß folgende Grundmerkmale aufweisen:

- **Sammlung** – Um bestimmte Wissensaspekte möglichst vollständig abzudecken und möglichst vielen Benutzern zur Verfügung zu stellen, ist es wichtig, verstreute Dokumentenbestände zusammenzuführen und möglichst zentral abzulegen.
- **Sichtung** – Mehrfach vorhandene bzw. irrelevante Dokumente müssen ausgesondert werden.
- **Inhaltliche Erschließung** – Damit Benutzer gezielt auf bestimmte Dokumente zurückgreifen können, ist es notwendig, den Inhalt der einzelnen Dokumente zu erschließen und diesen so vollständig wie möglich in Katalogen und Indizes zur Verfügung zu stellen.
- **Verfügbarmachung** – Benutzer müssen problemlosen und freien Zugriff auf die Originaldokumente oder Kopien dieser bekommen.

Bevor mit der Verwaltung von Dokumenten begonnen werden kann, ist es jedoch notwendig, zwei grundlegende Fragen zu beantworten:

- Mit welcher Art von Dokumenten hat man es zu tun?
- Welche Art von Dokumentenverwaltung ist für den jeweiligen Dokumenttyp am effizientesten?

2.3.1 Dokumenttypen

In [Henzler 1992] werden Dokumente nach fünf grundlegenden Aspekten kategorisiert:

- **Materialaspekt der Publikationsform** (Papier, Objekte, AV-Medien, Neue Medien)
- **Buchbinderischer Aspekt des Zusammenhalts** (Loseblattform, Gefaltete Form, Geheftete Form, Buchform)
- **Inhaltlicher Aspekt** (Belletristik, Sachliteratur, Studien- und Forschungsliteratur, Auskunftsliteratur - Nachschlagewerke, Nachrichten - Mitteilungen)
- **Erscheinungsweise** (Selbständige bzw. unselbständige Werke)
- **Anlaß der Veröffentlichung** (Amtliche Druckschrift, Halbamtliche Druckschrift, Kongreßschrift - Tagungsschrift, Gelegenheitsschrift, Hochschulschrift, Gesellschaftsschrift, Firmenschrift, Report - Forschungsbericht, Norm - Standard, Patentschrift)

Im Hinblick auf die Sammlung bzw. Archivierung von Dokumenten dürfte der Materialaspekt der interessanteste sein. Das vorliegende Material wird maßgebend darüber entscheiden, welche Verwaltung dem Dokumentenbestand zuteil wird.

Im Laufe der Menschheitsgeschichte wurden die unterschiedlichsten Medien als Dokumentenbasis genutzt: gegerbte Tierhäute, Stein- und Holztafeln, Baumrinde, Papyrus usw. Seit einigen Jahrhunderten hat sich i. A. Papier als Dokumentengrundlage durchgesetzt. Die rapide Entwicklung der Technik in den

letzten hundert Jahren brachte jedoch eine Reihe neuartiger Medien hervor. Um sich einen geordneten Überblick über die unterschiedlichen Dokumenttypen zu verschaffen, ist es nötig, diese zu kategorisieren:

2.3.1.1 Analoge Dokumente

Analoge Dokumente zeichnen sich dadurch aus, daß die in ihnen gespeicherte Information in einer Form vorliegt, die dem Empfänger eine direkte Decodierung ermöglicht.

Analoge Dokumente können in zwei Kategorien unterteilt werden:

- **Dokumente, die ohne technische Hilfsmittel gedeutet werden können.**
  Zu dieser Kategorie gehören Papierdokumente und Objekte wie Gemälde, Kunstplastiken usw. Solange der Empfänger den Zeichensatz versteht, mit dem die Information codiert wurde, kann er das Dokument ohne weitere Hilfsmittel durch Lesen, Betrachten, Befühlen deuten.
- **Dokumente, zu deren Deutung technische Hilfsmittel benötigt werden.**
  Zu dieser Kategorie gehören vor allem die sog. AV-Medien (AV = Audio/Video): Tonband, Videoband, Schallplatte, fotografischer Film usw. Informationen werden direkt in elektromagnetische oder optische Signale umgewandelt und auf diesen Medien gespeichert. Zur Darstellung der Informationen ist die Unterstützung durch ein spezielles technisches Gerät nötig.

2.3.1.2 Digitale Dokumente

Digitale Dokumente zeichnen sich dadurch aus, daß die in ihnen gespeicherten Informationen vor der Speicherung in einen Zahlencode umgewandelt werden. Dieser Zahlencode wird dann wiederum durch optische oder elektromagnetische Signale repräsentiert, die auf entsprechenden Medien gespeichert werden können. Während ein fremdsprachliches Papierdokument für einen Menschen wenigstens syntaktisch interpretierbar ist (d. h. er kann die darin enthaltenen Buchstaben entziffern), ist der binäre Code, in dem ein digitales Dokument verfaßt ist, für einen Menschen weder semantisch noch syntaktisch interpretierbar. Die in digitalen Dokumenten enthaltenen Informationen liegen somit in diskreter Form vor. Zur Erstellung und Interpretation digitaler Dokumente sind daher spezielle Schreib- bzw. Lesesysteme notwendig. Digitale Dokumente können wiederum in zwei verschiedenen Formen vorliegen:

- **elektronische Medien**
  Zur Darstellung von Dokumenten auf Compact Disc (Audio), Digital Audio Tape oder Laserdisc (Video) werden Geräte mit einem sog. D/A-Wandler (D/A = Digital/Analog) benötigt. Der D/A-Wandler konvertiert die Information, die in Form eines binären Zahlencodes gespeichert ist, in analoge Signale um, die als Bild- oder Tonströme durch entsprechende technische Einrichtungen (Lautsprecher, Bildschirm) wiedergegeben werden können.
- **EDV-gestützte Medien**
  Zur Darstellung von Dokumenten, die auf Diskette, Festplatte, CD-ROM, WORM oder magnetooptischen Platten gespeichert sind, wird spezielle Computer-Hardware und -Software benötigt. Im Vergleich zu elektronischen Dokumenten, die lediglich dargestellt werden können, lassen sich EDV-gestützte Dokumente mittels spezieller Software-Programme weiterverarbeiten. So ist es beispielsweise möglich, komplette Kollektionen von Textdokumenten, die in digitaler Form vorliegen, auf bestimmte Wörter hin zu durchsuchen.
  Eine derartige Verarbeitung elektronischer Dokumente ist nur über Umwege möglich: Die in dem elektronischen Dokument enthaltene Information muß erst auf ein EDV-gestütztes Medium übertragen werden. Das nun vorliegende EDV-gestützte Dokument kann danach beliebig weiterverarbeitet werden.
  Ein weiterer Pluspunkt dieser EDV-Dokumente liegt in der problemlosen Reproduzierbarkeit.

Analoge Medien				Digitale Medien	
Direkte Deutung		Deutung mit technischen Hilfsmitteln		Elektro-nische Medien	EDV-gestützte Medien
Papierform	Objekte	AV-Medien			
		Tonträger	*Bildträger*		
Druckwerke Handschriften Musiknoten Bilddokumente	Spiele Gemälde Museums-gegenstände	Tonband Compact Cassette Schallplatte	Videoband Cinefilm Negativfilm Diapositivfilm Mikrofilm Mikrofiche	Laserdisc Compact Disc Digital Audio Tape	Diskette Festplatte CD-ROM WORM Magneto-optische Medien

Tabelle 1: Übersicht Dokumentträgerarten

2.4 Was ist Dokumenten-Management?

Dokumenten-Management ist ein sehr generischer Begriff. Im Grunde genommen kann jeglicher Arbeitsgang, der sich auf ein Dokument bezieht, als Dokumenten-Management (oder zumindest als ein Teilschritt dessen) bezeichnet werden.

2.4.1 Manuelles Dokumenten-Management

Manuelles Dokumenten-Management wird in unserer heutigen Gesellschaft praktisch von jedem betrieben. Fast jeder hat ein Regal, auf dem er seine Bücher sortiert; viele Leute ordnen alle ihre Rechnungen und Kaufbelege ein, um Überblick über ihre Finanzen zu bekommen; kein Student oder Schüler ist ohne geordnete Heft- und Buchsammlung vorstellbar. Natürlich handelt es sich bei den genannten Beispielen um sehr rudimentäres Dokumenten-Management.

Doch auch manuelles Dokumenten-Management kann professionell und mit großem Aufwand betrieben werden. Die am weitest fortgeschrittenen manuellen Dokumenten-Management-Systeme sind wahrscheinlich die Bibliotheken. Gerade an Ihnen wird deutlich, daß Dokumenten-Management eine lange Tradition besitzt. Die ersten Bibliotheken des Abendlandes gab es schon um 600 v. Chr.[3] Ohne Bibliotheken, deren Hauptzweck es ist, Wissen zu sammeln, zu archivieren und zur Verfügung zu stellen, wäre der heutige Wissens- und Entwicklungsstand der Menschheit undenkbar. Ein weiteres Beispiel für manuelles Dokumenten-Management in großem Umfang sind Archive.

Manuelles Dokumenten-Management stammt aus einer Zeit, in der es nur analoge Dokumente gab. Auch heute wird manuelles Dokumenten-Management fast ausschließlich für analoge Dokumentenbestände genutzt. Zwar wird der heimische PC-Anwender seine Diskettensammlung mit großer Wahrscheinlichkeit manuell verwalten, doch ab einer bestimmten Größenordnung macht eine manuelle Verwaltung digitaler Dokumentenbestände keinen Sinn, da hierbei neben den digitalen Dokumenten analoge Verwaltungsdokumente (Kataloge, Indizes, Listen) geführt werden müssen. Bibliotheken und Archive bedienen sich

[3] Brockhaus Enzyklopädie in zwanzig Bänden, Wiesbaden: F. A. Brockhaus (1968)

heutzutage zwar elektronischen Hilfsmitteln - so wird es eine immer häufiger angewandte Praxis, daß beispielsweise Bibliotheken ihre Kataloge und Indizes auf elektronischen Datenbanksystemen führen - doch der Großteil des Dokumenten-Managements, nämlich das Sichten, Archivieren und die Verfügbarmachung der Dokumente, geschieht immer noch auf manuellem Wege.

Manuelles Dokumenten-Management birgt einige schwerwiegende Nachteile:

- **Großer Platzbedarf**
  Früher riesig erscheinende Bibliotheks- und Archivgebäude platzen mit zunehmender Dokumentenmenge schier aus allen Nähten. In manchen Bibliotheken geht man teilweise schon zu einer Verlagerungsstrategie über. Durch diese Verlagerungen wird das Hauptgebäude entlastet; dies aber nur für einen begrenzten Zeitraum - nämlich nur bis zu dem Tag, an dem das Hauptgebäude wieder voll ist und eine Auslagerung weiterer Dokumente bevorsteht. Eine weitere Schwierigkeit dieser Auslagerungspraxis besteht darin, daß auf extern aufbewahrte Dokumente nicht direkt zugegriffen werden kann, sondern diese erst aus den externen Lagern in das Hauptgebäude gebracht werden müssen.

- **Schwierige Lagerung**
  Organische Materialien wie Papier unterliegen einem natürlichen Zerfallsprozeß. Um diesen Prozeß aufzuhalten bzw. zu verlangsamen müssen bestimmte Lagerungsbedingungen (niedrige Temperaturen, geringe Luftfeuchtigkeit, möglichst wenig Licht usw.) vorherrschen.

- **Aufwendige Verwaltung**
  Jedes Dokument innerhalb eines manuellen Dokumenten-Management-Systems muß manuell indiziert und katalogisiert werden. Sollen Dokumente außerdem verliehen werden (wie bei Bibliotheken), so müssen zusätzlich Verleihlisten geführt werden. Es leuchtet ein, daß die manuelle Verwaltung riesiger und kontinuierlich wachsender Dokumentenbestände im Laufe der Zeit immer schwieriger und aufwendiger wird.

- **Eingeschränkte Suchmöglichkeiten**
  Auf der Suche nach relevanten Dokumenten für eine bestimmte Problemstellung kann der Anwender lediglich auf manuell erstellte Kataloge (z. B. Titel- und Schlagwortkataloge) zurückgreifen. Die Dokumente selbst können nicht oder nur sehr mühevoll (Dokument für Dokument) durchsucht werden.

2.4.2 Elektronisches Dokumenten-Management

Mit der starken Verbreitung von Personal-Computern verändert sich die Art und Weise, wie Dokumente produziert werden. Dokumente nehmen vielerlei Gestaltungsformen an: Sie enthalten Text, Tabellen, Grafiken usw. Die Vernetzung von Computern erlaubt es, daß diese Dokumente innerhalb eines betriebsinternen oder sogar weltweiten Netzwerkverbundes ge- bzw. verteilt werden. Über viele Jahre hinweg war die Bedeutung der elektronischen (digitalen) Version eines Dokumentes zweitrangig gegenüber der gedruckten Version. In jüngster Zeit scheint jedoch ein Umdenken stattzufinden. Nach unseren Erfahrungen wird immer öfter das elektronische Dokument als das Original angesehen. Mit der verstärkten Nutzung von elektronischen Informationssystemen wie z. B. dem Internet (mit seinen Diensten *eMail*, *World Wide Web* und *Gopher*) verlieren gedruckte Dokumente an Bedeutung.

Zur effektiven Verwaltung elektronischer Dokumente werden elektronische Dokumenten-Management-Systeme (EDMS) benötigt.

Die Landes- und Hochschulbibliothek in Darmstadt beispielsweise verlegt weniger gefragte Dokumente in die Räume des benachbarten Staatsarchives Hessen.

Ein vollwertiges EDMS verfügt über folgende Grundfunktionalitäten:

- Eine Speicherungsumgebung, in der die Dokumente gehalten werden
- Methoden, um Dokumente dieser Speicherungsumgebung hinzuzufügen
- Methoden, um gespeicherte Dokumente über Suchanfragen wiederzufinden

Je nach Anwendungsbereich bieten EDMS zusätzlich eine oder mehrere der folgenden Leistungsmerkmale:

- **Sperrung von Dokumenten**, so daß die Bearbeitung eines Dokumentes zu einem bestimmten Zeitpunkt immer nur von einem Anwender vorgenommen werden kann (Check-in, Check-out).
- **Versionskontrolle** soll gewährleisten, daß nachvollzogen werden kann, welchen Veränderungen ein Dokument unterlegen war bzw. wer diese Veränderungen vorgenommen hat (Version control, audit trail).
- **Sicherheitsmechanismen** sollen dafür sorgen, daß nur dazu berechtigte Personen auf bestimmte Dokumente zugreifen können.
- **Gruppierung von Dokumenten** in logische Einheiten (z. B. Verzeichnisse oder Bücher)
- **Freitextsuche**, um Dokumente hinsichtlich des Textes, den sie enthalten, wiederzufinden.
- **Vergabe von Dokumentattributen** (Titel, Autor, Datum der letzten Änderung usw.)
- **Unterstützung von Workflow-Management** – Wenn bestimmte Vorgänge in mehreren Instanzen durch verschiedene Personen bearbeitet werden müssen, so muß es möglich sein, daß die vorgangsbezogenen Dokumente von einer Person zu anderen geleitet werden können.
- **Imaging** – Die Möglichkeit, analoge Dokumente zu digitalen Dokumenten zu konvertieren, die dann dem Dokumentenspeicher hinzugefügt werden können.

Elektronische Dokumenten-Management-Systeme bieten im direkten Vergleich zur manuellen Verwaltung analoger Dokumente erhebliche Vorteile:

- **Platzersparnis**
  Die Aufzeichnungsdichte elektronischer Speichermedien ist so hoch, daß beispielsweise der Inhalt einer zwanzigbändigen Enzyklopädie auf einer einzigen Compact Disc abgelegt werden kann.
- **Schneller Zugriff**
  Mußte sich ein Sachbearbeiter früher in ein Archiv begeben, um an ein bestimmtes Dokument zu gelangen, kann er mit Hilfe eines EDMS von seinem Arbeitsplatz aus direkt auf den kompletten Dokumentenbestand zugreifen.
- **Simultaner Zugriff mehrerer Anwender**
  Mehrere Anwender können zeitgleich lesend auf ein Dokument zugreifen. Dies ist beispielsweise mit Papierdokumenten nicht möglich.
- **Komfortable Suchmöglichkeiten**
  Durch spezielle Indexierungs- und Suchmechanismen wird es möglich, auf Dokumente über Aspekte wie Titel, Autorenname und Volltextsuche zuzugreifen. Der Anwender braucht sich nicht länger Gedanken darüber zu machen, wie ein bestimmtes Dokument heißt und wo dieses abgelegt wurde.

Trotz der Vorzüge, die elektronisches Dokumenten-Management gegenüber der traditionellen Dokumentenverwaltung zu bieten hat, eröffnen sich durch den Einsatz von EDMS Schwierigkeiten, die nicht übersehen werden sollten:

- **Technik-Abhängigkeit**
  Ein Unternehmen, das seine Dokumentenverwaltung nur noch über ein EDMS erledigt, ist von diesem abhängig. Fällt der EDMS-Server aus (z. B. durch Funktionsstörung oder Stromausfall), besteht keinerlei Möglichkeit, auf die in ihm gespeicherten Dokumente zuzugreifen. Eine längerfristige Funktionsstörung innerhalb eines EDMS kann somit einen ganzen Betrieb (oder zumindest seine Administration) komplett lahmlegen. Dies kann im Extremfall zu Produktionsausfällen führen, die hohe Kosten nach sich ziehen können.

- **Begrenzte Haltbarkeit elektronischer Dokumente**
  Physische Haltbarkeit: Es ist heute noch weitgehend ungeklärt, wie lange digitale Medien wie CD-ROM oder WORM aufgehoben werden können, bevor die auf ihnen gespeicherten Informationen nicht mehr gelesen werden können.
  Technische Haltbarkeit: Bei den relativ kurzen Innovationszyklen im Bereich der Computertechnik ist es kaum vorstellbar, daß ein Unternehmen mehr als zehn Jahre mit dem selben EDMS arbeitet. Wenn es auf ein anderes System oder eine neuere Version des gleichen EDMS umsteigt, müssen u. U. alle Dokumente in ein neues Format umgewandelt werden. Dies hat wiederum hohe Kosten zur Folge.□

- **Datenschutz**
  Elektronische Dokumente lassen sich relativ einfach manipulieren. Einem Papierdokument sieht man an, ob auf ihm herumradiert oder etwas überschrieben wurde. Eine Manipulation von elektronischen Daten hinterläßt keinerlei Spuren. Aus diesem Grunde werden immer größere Anstrengungen unternommen, Methoden zu entwickeln, die in Dokumenten enthaltenen Informationen mittels spezieller mathematischer Algorithmen zu verschlüsseln, um einen unerlaubten Dokumentenzugriff zu verhindern. Obwohl diese Verschlüsselungsmethoden immer ausgeklügelter werden, ist es bislang noch nicht gelungen, eine Algorithmus zu entwickeln, der absolut sicher ist.
  Daraus ergibt sich folglich eine rechtliche Problematik. Die meisten deutschen Gerichte erkennen elektronische Dokumente nicht als beweisfähige Urkunden an. Will ein Unternehmen rechtlichen Problemen aus dem Wege gehen, ist es somit gezwungen, neben dem EDMS nach wie vor ein Papierarchiv zu führen.

2.5 Datenhaltungstechnologien

Wie im Abschnitt 2.4.2. beschrieben, müssen einige grundlegende Anforderungen erfüllt sein, um eine effiziente Verwaltung elektronischer Dokumente zu gewährleisten:

- Eine Speicherungsumgebung, in der die Dokumente gehalten werden
- Methoden, um Dokumente dieser Speicherungsumgebung hinzuzufügen
- Methoden, um gespeicherte Dokumente über Suchanfragen wiederzufinden

Dem Anwender stehen heutzutage unterschiedliche Datenhaltungstechnologien zur Verfügung:

- Dateisysteme
- Datenbanksysteme
- Datenbank-Management-Systeme
- Information-Retrieval-Systeme

Die genannten Technologien werden im nun folgenden Abschnitt näher beschrieben.

2.5.1 Dateisysteme

Dateisysteme sind einfache Datenverwaltungssysteme, die vom Betriebssystem zur Verfügung gestellt werden. Die Datenverwaltung über Dateisysteme umfaßt alle Arbeiten, die außer der Verwendung von Dateien durch Programme zu erledigen sind, um die eigentliche Dateiarbeit auszuführen bzw. abzusichern. Zu den Aufgaben eines Dateisystems zählen:

- **Führen von Dateikatalogen** (auch Verzeichnisse, Directories genannt), aus denen die Merkmale einer Datei, also Art, Aufbau und Speicherort der Datei ersichtlich werden.
  Somit wird eine Speicherungsumgebung für Dokumente zur Verfügung gestellt.
- **Erzeugung von Dateien** (dies schließt auch das Einfügen von Dateien ein, die von außerhalb des lokalen Dateisystems, also z. B. aus externen Datennetzen, stammen)
  Es stehen also Methoden zur Verfügung, der Speicherungsumgebung neue Dokumente hinzuzufügen.
- **Kopieren, Verschieben und Löschen von Dateien** innerhalb der Verzeichnisstruktur
  Innerhalb der Speicherungsumgebung können Dokumente bewegt oder aus ihr entfernt werden.

Dokumente werden von Dateisystemen in Form einzelner Dateien geführt. Dateisysteme erfüllen ihren Zweck am besten, wenn eine Datei mit einem bestimmten Namen gefragt ist. Typische Anwendungen sind hier das Laden oder Speichern eines Dokumentes in einem Textverarbeitungs- oder Grafikprogramm.

Bei der Oberfläche von Dateisystemen handelt es sich nicht um eine Abfrageschnittstelle. Dateisysteme gehen davon aus, daß die Anwendung oder der Endanwender den Namen der gewünschten Datei kennt. Den Inhalt, der von ihm verwalteten Dokumente, kennt ein Dateisystem nicht. Außer sehr rudimentär ausgeprägten Suchwerkzeugen, die von einigen Betriebssystemen zur Verfügung gestellt werden (z. B. grep oder find unter UNIX), stellen Dateisysteme außer der Dokumentensuche nach Dateinamen keine weiteren Suchmechanismen zur Verfügung, mit welchen man Dokumente wiederfinden kann.

Gerade aufgrund fehlender Suchmechanismen zeigen sich Dateisysteme als ungeeignet, eine effiziente Verwaltung großer Dokumentenmengen zu gewährleisten.

Bekannte Vertreter der Dateisysteme sind :

- **FAT** - File Allocation Table (MS-DOS, MS-Windows)
- **HPFS** - High Performance File System (OS/2)
- **NTFS** - New Technology File System (Windows NT)
- **NFS** - Network File System (UNIX)

2.5.2 Datenbanksysteme

Eine Datenbank ist nach [Henzler 1992] ein organisierter Informationsspeicher, in dem Daten so abgespeichert werden, daß später die Rückgewinnung nach verschiedensten Gesichtspunkten möglich ist. Der Kern einer jeden Datenbank ist die sog. Datenbasis, in der die eigentlichen Nutzdaten gespeichert sind. Für die Speicherung gelten drei Grundregeln:

- Die Daten müssen **nicht-redundant** sein. Dies bedeutet, daß jedes gespeicherte Datum nur einmal in der Datenbasis vorhanden sein darf.
- Die Daten werden völlig **unabhängig voneinander** gespeichert.
- Die gespeicherten Daten sind **mehrfach benutzbar**. Dies bedeutet, daß die Daten beliebig oft abgefragt werden können.

Die Datenbasis wird in Form einer (oder mehrerer) Datei(en) (der sog. *Primärdatei*) im Dateisystem eines Computers vorgehalten.

Ein anderer grundlegender Bestandteil einer Datenbank sind die sog. *Indexdateien*. In ihnen wird für jedes in der Datenbasis gespeicherte Datum der genaue Standort innerhalb der Datenbasis eingetragen.

Damit die gespeicherten Daten innerhalb einer Datenbank sinnvoll genutzt werden können, müssen sie in Beziehung zueinander gebracht werden.

Dies kann beispielsweise durch die Anwendungen geschehen, welche die Daten nutzen sollen. Ein Problem dieser Vorgehensweise ist, daß jede Anwendung die Beziehungen zwischen den Daten neu definieren muß. Es leuchtet ein, daß dies mit steigender Anwendungszahl zu einem relativ großen Aufwand anwachsen kann.

Aus diesem Grunde wurden *Datenbank-Management-Systeme (DBMS)* entwickelt. DBMS sind Software-Programme, die eine anwendungsunabhängige Verwaltung von Datenbanken ermöglichen, also von sich aus die Daten einer Datenbank in Beziehung bringen. Der Vorteil dieser Vorgehensweise ist, daß die Organisation der Daten nur noch einmal erfolgen muß. Der Datenzugriff von Anwendungen aus erfolgt dann stets über das DBMS.

Im Laufe der Zeit wurden mehrere unterschiedliche DBMS-Technologien entwickelt, deren wichtigste Vertreter im Folgenden näher erläutert werden.

2.5.2.1 Relationale Datenbank-Management-Systeme (RDBMS)

Die von einem RDBMS verwalteten Daten werden in Form zweidimensionaler Tabellen (sog. Relationen) organisiert. Eine Tabelle besteht aus einzelnen Attributwerten (Spalten) und Datensätze oder Tupel (Zeilen). Zur Veranschaulichung folgt ein Beispiel:

Matrikelnummer	Vorname	Nachname
940473	Bernd	Pörner
141269	Johnny	Lampe

Tabelle 2: Einfache Relation

Anmerkung zu obigem Beispiel:
Die Relation Student wird durch die Attribute Matrikel-Nr, Vorname und Nachname beschrieben. Jedes Tupel repräsentiert somit einen Studenten.

Der Vorteil von RDBMS liegt darin, daß sich zwischen den an sich voneinander unabhängigen Relationen Verknüpfungen ziehen lassen. Auch hierzu ein Beispiel:

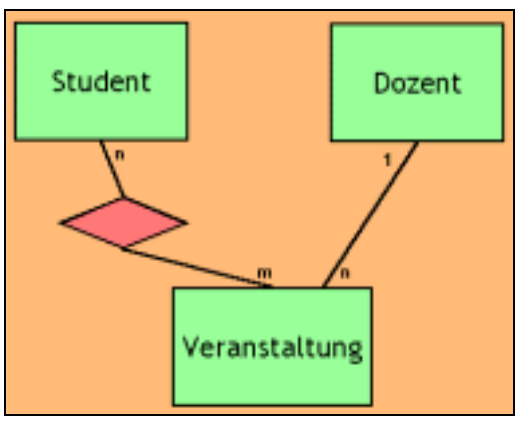

Abbildung 1: Beispiel für die Verknüpfung dreier Objekte

Anmerkung zu obigem Beispiel:
In einem vereinfachten Realitätsausschnitt werden die Beziehungen zwischen Dozenten, Veranstaltungen und Studenten gezeigt. Dabei kann ein Student n Veranstaltungen belegen. Umgekehrt kann eine Veranstaltung von m Studenten belegt werden. (n:m-Beziehung) Ein Dozent kann n Veranstaltungen halten. Umgekehrt kann eine Veranstaltung nur von 1 Dozenten gehalten werden. (1:n-Beziehung)

Durch diese Verknüpfung lassen sich aus mehreren Tabellen (evtl. verschiedener Datenbanken) immer wieder neue Kombinationstabellen zur Datenanzeige erstellen.

Zur Datenabfrage innerhalb von RDBMS wird üblicherweise die Structured Query Language (SQL) genutzt. Die Beliebtheit der SQL rührt vor allem von ihrer

- **Mächtigkeit** – SQL wird nicht nur zur Datenabfrage benutzt. Mit ihr werden Datenbanken und deren Indexdateien generiert, Relationen und Datensätze eingefügt, verändert und gelöscht.
- **Nähe zur natürlichen Sprache**

Üblicherweise werden von RDBMS nur einfache Datentypen unterstützt:

- Zahlen
- Währungsdaten
- Datums- bzw. Zeitangaben
- kurze alphanumerische Zeichenketten
- Boole'sche Werte

Komplexe Datentypen, wie z. B. Grafiken, Video- und Audiostreams, dreidimensionale Objekte oder Textdokumente können von herkömmlichen RDBMS nicht indexiert, durchsucht oder manipuliert werden. Sie werden als sog. *Binary Large Objects (BLOBs)* gespeichert und können lediglich angezeigt werden. Das Wiederfinden von BLOBs ist nur über zusätzliche Attribute (z. B. textliche Beschreibungen), die dem jeweiligen Datensatz angefügt wurden, möglich. Nähere Erläuterungen zu dieser Problematik finden sich in [Stonebraker 1996].

Als Grundlage für ein effizientes Dokumenten-Management eignen sich RDBMS daher kaum. Der Anwendungsbereich von RDBMS ist überall dort, wo Faktendaten (z. B. Personal- oder Finanzdaten) verwaltet werden sollen.

2.5.2.2 Objektorientierte Datenbank-Management-Systeme (OODBMS)

OODBMS zeichnen sich dadurch aus, daß sie eine Speichermöglichkeit für komplexe Objekte bieten, die durch objektorientierte Programmiersprachen wie *C++* oder *SmallTalk* generiert und manipuliert werden. (siehe [Stonebraker 1996])

Der Vorteil der Objektorientierung liegt unter anderem darin, daß Objekte, die untereinander gewisse Ähnlichkeiten (gemeinsame Eigenschaften) aufweisen, zu sog. Klassen zusammengefaßt werden können. So könnte man beispielsweise die Objekte `hausmeister`, `sekretärin`, `sachbearbeiter` und `direktor` zu der Klasse `PERSONAL` zusammenfassen.

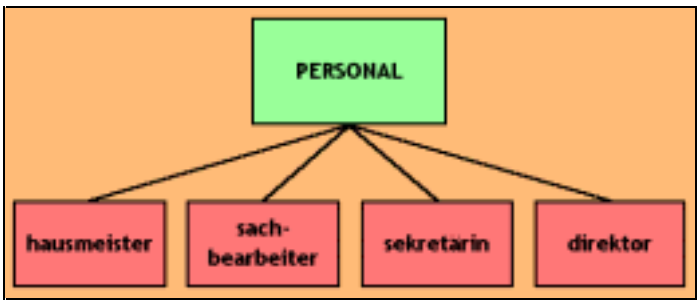

Abbildung 2: Beispiel einer Klassenbildung

Anmerkung zu obigem Beispiel:
Alle Objekte haben gemeinsame Eigenschaften, welche sie zu Angestellten eines Betriebes machen. Diese gemeinsamen Eigenschaften brauchen dann nur einmal, nämlich in der Oberklasse **PERSONAL** *definiert zu werden. Jedes Objekt erhält diese Eigenschaften dann durch die sog. Vererbung. In den Objekten selbst müssen dann nur noch die speziellen Eigenschaften, welche sie von den anderen Objekten unterscheiden, definiert werden.*

Zusätzlich kann man einer Klasse und/oder einzelnen Objekten noch spezielle Methoden zuweisen. Eine vorstellbare Methode bei o. g. Beispiel wäre: »Wie kontaktiere ich `Objekt X`?« Einen `sachbearbeiter` wird man unter einer bestimmten Durchwahl direkt erreichen können, während man den `direktor` normalerweise nur dann an die Strippe bekommen wird, wenn man seine `sekretärin` anruft und diese einen dann an den `direktor` durchstellt (oder auch nicht ...;-).

OODBMS besitzen noch viele weitere Funktionalitäten, deren Erklärung jedoch den Rahmen dieser Ausarbeitung sprengen würde.

Durch die Fähigkeit, komplexe Datentypen (also auch Textdokumente) effizient verwalten zu können, zeigen sich OODBMS als mögliche Grundlage für ein elektronisches Dokumenten-Management-System.

Ein entscheidender Nachteil von OODBMS liegt jedoch darin, daß im Vergleich zu RDBMS meist keine einheitliche Abfrage- oder Datenmanipulationssprache (ähnlich der SQL) zur Verfügung steht. Datenabfragen oder –manipulationen werden bei vielen OODBMS mittels objektorientierter Programmiersprachen (C++, Eiffel usw.) durchgeführt. Während Datenbankapplikationen für RDBMS, bedingt durch die Einfachheit der SQL, auch von weniger erfahrenen Entwicklern erstellt werden können, erfordert die OODBMS-Applikationsentwicklung, aufgrund der Komplexität objektorientierter Programmiersprachen, sehr erfahrene Entwickler/Programmierer.

Ein weiterer Nachteil von OODBMS besteht darin, daß Multiuser-Eigenschaften und –Mechanismen (wie beispielsweise das Transaktions-Management), die bei professionellen RDBMS zur Standardausstattung gehören, bei den meisten OODBMS noch relativ rudimentär ausgeprägt sind.

Diese Mäkel schränken den Anwendungsbereich von OODBMS relativ stark ein. Typische Anwendungsbereiche von OODBMS sind kleine Anwendergruppen, die komplexe Manipulationen an Objektsammlungen in der Größenordnung von zehn Megabyte durchführen müssen. Als Beispiel seien hier die Bearbeitung von CAD-Zeichnungen und anderer komplexer, unstrukturierter Daten genannt.

2.5.2.3 Objektrelationale Datenbank-Management-Systeme (ORDBMS)

ORDBMS sind eine Weiterentwicklung der RDBMS. Grundsätzlich läßt sich sagen, daß mit ORDBMS versucht wird, Eigenschaften von OODBMS in RDBMS zu integrieren. In Stonebraker 1996] werden drei grundsätzliche Voraussetzungen für ein ORDBMS definiert:

Voraussetzung 1: Erweiterung der Basisdatentypen

Wie im Abschnitt 2.5.2.1 dargestellt, können RDBMS lediglich einfache Datentypen vollständig unterstützen:

Komplexe Datentypen werden als BLOBs gespeichert, die weder indexiert, durchsucht noch manipuliert werden können. So ist es in einem RDBMS beispielsweise unmöglich, gespeicherte PostScript-Dokumente nach bestimmten Wortketten zu durchsuchen. Um dies zu realisieren, müßte der Anwender eine spezielle Anwendung auf seinem Client-Rechner vorhalten. Die einzelnen PostScript-Dokumente würden nacheinander vom DBMS-Server auf den Client-Rechner geladen und dort durchsucht werden. Es leuchtet ein, daß dies schon allein wegen dem riesigen Datentransfervolumen einen nicht praktikablen Lösungsweg darstellt.

Ein ORDBMS muß dem Anwender/Entwickler die Möglichkeit bieten, Datentypen und die dafür benötigten Zugriffs-, Indexierungs- und Speichermethoden selbst zu definieren. Somit ist es dann möglich, die o. g. PostScript-Dokumente im DBMS selbst zu durchsuchen. Der Anwender würde dann nur noch die Suchanfrage an das DBMS schicken. (Wesentlich geringeres Datenaufkommen)

Ein weiterer Vorteil dieser Vorgehensweise besteht darin, daß im Falle einer Datenbearbeitung (mit Veränderung der Daten) das DBMS volle Kontrolle über alle Bearbeitungsvorgänge hat. Dateioperationen, die nicht vollständig ausgeführt werden konnten (z. B. aufgrund eines Stromausfalls während der Veränderung einer Datei) werden zurückgesetzt, so daß kein Datenverlust entstehen kann. Bei einem herkömmlichen RDBMS müßten derartige Sicherheitsvorkehrungen beim Bearbeiten von BLOBs von der jeweiligen Client-Anwendung übernommen werden.

Voraussetzung 2: Komplexe Objekte

Die von einem ORDBMS genutzte Datenmanipulationssprache (typischerweise SQL) muß die Arbeit mit komplexen Daten vollständig unterstützen. Dazu ist es notwendig, neue Typkonstruktoren einzuführen. Innerhalb von RDBMS kennt man beispielsweise die Typkonstruktoren *table* oder *view* (virtuelle Ansichtstabelle). Neben *sets* und *references* ist *composite type* der mit Abstand wichtigste Typkonstruktor bei ORDBMS. Der Einfachheit wegen wird nur auf ihn näher eingegangen. Eine detailliertere Darstellung der Typkonstruktoren wird in [Stonebraker 1996] vorgenommen.

Zusammengesetze Datentypen (composite types)

Um in einem ORDBMS Tabellen definieren zu können, müssen vorher sog. Composite Types festgelegt werden. Die Definition eines Composite Type (im Folgenden der Einfachheit wegen CP genannt) enthält den Namen des CP und dessen Attribute mit deren Datentypen. Hierzu ein Beispiel:

```
CREATE TYPE person_t
(
vorname       text,
nachname      text,
geburtsdatum  date
);
```

Aus diesem Typ heraus können dann Tabellen erstellt werden:

```
CREATE TABLE hochschulangehoeriger OF TYPE person_t
```

Man kann sich eine Tabelle als eine Art Container vorstellen, in dem die Instanzen eines CP gehalten werden. Ein Vorteil dieser Vorgehensweise ist die Möglichkeit, mehrere Tabellen desselben Typs zu erstellen. Dies eröffnet die Möglichkeit der *Vererbung*, auf die im nächsten Punkt eingegangen wird.

Ein weiterer Vorteil der Definition von CP liegt darin, daß ein definierter Typ in einer anderen Tabelle als Attribut eingesetzt werden kann:

```
CREATE TABLE gremiumsmitglied
(
position   text,
mitglied   person_t
);
```

In der Tabelle `gremiumsmitglied` müssen die Attribute `vorname`, `nachname` und `geburtsdatum` nicht neu definiert werden; sie werden einfach aus dem Typ `person_t` übernommen.

Voraussetzung 3: Vererbung

Durch die Vererbung ist es möglich, Eigenschaften eines Objektes an ein von ihm abhängiges Unterobjekt weiterzugeben. Hierzu ein Beispiel:

Aus der im vorhergehenden Punkt definierten Tabelle `hochschulangehoeriger` können nun weitere, spezialisierte Typen und Tabellen generiert werden:

```
CREATE TABLE student OF        E        TYPE student_t
(
matri       el_nr    integer,
)
                    ER hochschulangehoeriger;

CREATE TABLE do       ent OF        E        TYPE do       ent_t
(
personal_nr    integer,
)
                    ER hochschulangehoeriger;
```

Die Tabellen `student` und `dozent` übernehmen die Attribute aus der Tabelle `hochschulangehoeriger`, d. h. sie brauchen nicht für jede Tabelle neu definiert zu werden. Es werden nur noch die spezifischen Attribute neu definiert.

Derzeit sind die traditionellen RDBMS die am meisten verbreiteten DBMS. Doch schon aufgrund der Tatsache, daß multimediale Inhalte, die durch ihre Komplexität von RDBMS nicht zufriedenstellend unterstützt werden können, in nahezu allen Bereichen der Informationsverarbeitung Einzug halten, kann davon ausgegangen werden, daß die herkömmlichen RDBMS mit der Zeit immer mehr durch ORDBMS verdrängt werden.

Schon jetzt stehen einige ernstzunehmende ORDBMS zur Verfügung: »DB/2 Universal Database« von IBM (http://www.software.ibm.com), »Dynamic Server« von Informix (http://www.informix.com) und »Oracle8« von Oracle (http://www.oracle.com). Es kann davon ausgegangen werden, daß andere namhafte DBMS-Hersteller wie Oracle und Sybase in nächster Zeit entsprechende Konkurrenzprodukte auf den Markt bringen werden.

2.5.2.4 Information-Retrieval-Systeme (IRS)

IRS sind spezielle Datenbanksysteme für meist unformatierte Daten, mit deren Hilfe umfangreiche Texte sinnvoll verwaltet und wiedergefunden werden können.

Ein Datensatz in einem IRS enthält neben dem Volltext eines Dokumentes auch zusätzliche Metainformationen, wie z. B. Autor, Titel, Stichwörter usw.

Der Zugriff auf die Datensätze in einem IRS erfolgt über spezielle Indexdateien. Der Vorgang der Indexierung kann grundsätzlich folgendermaßen dargestellt werden:

- **Schritt 1:**
  Nach dem Einfügen in das IRS wird ein Dokument in seine Bestandteile, also die einzelnen Wörter, zerlegt. Diesen Vorgang nennt man *Parsing*.
- **Schritt 2 (optional):**
  Die einzelnen Wörter werden auf ihre Stammform zurückgeführt (z. B.: »Studenten« nach »Student«, »gingen« nach »gehen« usw.). Dieser Vorgang wird *Stemming* genannt.
- **Schritt 3:**
  Die Wörter werden mit einer *Stoppwortliste* abgeglichen. In einer Stoppwortliste sind unter- und übersignifikante Wörter enthalten (Bsp: »der« = untersignifikant; in einer medizinischen Dokumentation wird das Wort »Medizin« übersignifikant sein). Die Stoppwörter werden aus dem zu indizierenden Wortbestand gestrichen.

- **Schritt 4:**
  Die übriggebliebenen Wörter werden mit Angabe des genauen Standortes innerhalb der Datenbank und innerhalb des Dokumentes in eine Indexdatei geschrieben.

Leistungsfähige IRS können für jede Kategorie (Autor, Titel, Volltext usw.) einen eigenen Index erstellen.

Bei der Suche nach Begriffen durch einen Anwender greift das IRS auf diese Indizes zurück, greift sich die zutreffenden Datensätze heraus und gibt sie dem Anwender als Antwortmenge zurück. Bei der Vorhaltung getrennter Indizes besteht für den Anwender die Möglichkeit, seiner Suchanfrage eine größere Genauigkeit zu verleihen, indem er die Suche auf bestimmte Kategorien begrenzt.

Typischerweise bieten IRS auch die Möglichkeit, mehrere Begriffe zu einer Suchanfrage zu kombinieren. Dazu werden verschiedene Mengen- und Abstandsoperatoren (siehe dazu Abschnitt 3.4.2.1) zur Verfügung gestellt.

IRS sind sehr spezielle Datenbanksysteme, die nur für die Verwaltung von Textinformationen benutzt werden können und damit auch ein klar umgrenztes Einsatzgebiet haben, nämlich überall dort, wo ausschließlich mit Textinformationen gearbeitet wird (z. B. in wissenschaftlichen Dokumentationsstellen).

2.5.3 Abschließender Vergleich der unterschiedlichen Systeme

Dateisysteme und RDBMS scheiden als mögliche Kandidaten zur Realisierung eines Dokumenten-Management-Systems von vornherein aus:

- **Dateisysteme** bieten keinerlei Möglichkeiten, die Inhalte von Dokumenten (Dateien) zu durchsuchen.
- Mit **RDBMS** gibt es keine Möglichkeit, effiziente Suchmethoden für komplexe Datentypen (also auch nicht für Texte) zur Verfügung zu stellen.

Als mögliche Alternativen bleiben somit nur noch OODBMS, ORDBMS und IRS.

OODBMS bieten die Möglichkeit, komplexe Datentypen effizient zu verwalten. Aufgrund dieser Tatsache werden OODBMS schon innerhalb verschiedener elektronischer Dokumenten-Management-Systeme (wie z. B. *HyperWave*, vormals *Hyper-G*) eingesetzt. Die Anwendungsentwicklung mit OODBMS gestaltet sich jedoch aufgrund dem Fehlen einer standardisierten Abfragesprache verhältnismäßig schwierig. Während bei RDBMS die Anwendungsentwicklung oftmals vom Anwender selbst vorgenommen wird (beispielsweise in großen Unternehmen) ist dies bei OODBMS kaum möglich. Der Anwender muß daher meist auf fertige Systeme zurückgreifen, die ihm von der Software-Industrie angeboten werden. Ein Serienprodukt läßt sich jedoch oftmals nicht optimal oder nur mit erheblichem Aufwand in schon bestehende Software-Umgebungen integrieren.

ORDBMS bieten einerseits durch die Bereitstellung von SQL die Einfachheit eines RDBMS und andererseits durch die Einführung von Merkmalen aus der objektorientierten Welt (Unterstützung komplexer Datentypen, Vererbung, Klassenbildung) die Leistungsfähigkeit eines OODBMS. Während dem Datenbankkern eines OODBMS ein neuer Datentyp buchstäblich hinzuprogrammiert werden muß, gestaltet sich die Erweiterung eines ORDBMS verhältnismäßig einfach:

Ein ORDBMS unterstützt grundsätzlich alle Datentypen, die auch von RDBMS unterstützt werden. Wird die Unterstützung für einen weiteren Datentyp benötigt (beispielsweise für Textdokumente), so wird diese durch die Installation eines Zusatzmoduls (sog. *DataBlades* oder *Extender*) realisiert. Die in diesen Zusatzmodulen enthaltenen Speicher-, Indexierungs- und Zugriffsmethoden setzen sich in den Datenbankkern. Es findet also eine echte Erweiterung der Fähigkeiten des DBMS statt. Zusatzmodule gibt es heute schon für die populärsten komplexen Datentypen (div. Textformate, 2D-Grafiken, 3D-Grafiken, MPEG-Videos usw.)

Vorteil dieser Vorgehensweise: Da die Programmierungsschnittstellen *(API = Application Programming Interface)* zur Entwicklung von Zusatzmodulen vom DBMS-Hersteller offengelegt werden, kann diese durch andere Hersteller erfolgen. Somit ergibt sich eine klare Arbeitsteilung: Der DBMS-Hersteller kann sich auf die Weiterentwicklung der Datenbanktechnologie konzentrieren, während die Zusatzmodule von spezialisierten Unternehmen entwickelt werden. Aufgrund der Spezialisierung eines Software-Unternehmens auf einem bestimmten Gebiet (z. B. Text- oder Grafikanwendungen) werden die von ihm entwickelten Zusatzmodule wahrscheinlich eine wesentlich höhere Funktionalität und Qualität besitzen, als wenn diese durch den DBMS-Hersteller selbst entwickelt würden, der unmöglich über ein solches Spezialwissen in Richtung Text oder Grafik verfügen kann.

Vorteil für den Anwender: Er kann letztendlich selbst darüber entscheiden, wie leistungsfähig sein DBMS ist, indem er nur die für seinen Zweck benötigte Datentypen-Unterstützung hinzukauft. Zusätzlich ist es auch für den Anwender möglich, die Funktionalität seines DBMS selbst zu erweitern, indem er neue Datentypen selbst definiert.

DB-Anwendungsentwickler werden bei einem Umstieg von RDBMS nach ORDBMS nicht viel umlernen müssen, da sie es nach wie vor mit einer SQL-Umgebung zu tun haben werden. Je nachdem, welche Zusatzmodule dem System hinzugefügt werden, wird dieses Standard-SQL um neue Befehlssätze erweitert, um der Behandlung komplexer Datentypen gerecht zu werden. Diese sind jedoch relativ leicht und schnell erlernbar.

IRS sind eigentlich prädestiniert dafür, große Mengen an Textdokumenten zu verwalten. Aufgrund der Vorhaltung spezieller Textindizes ist eine schnelle und effiziente Suche über große Textmengen möglich. IRS sind jedoch sehr spezialisierte Systeme und daher auch relativ unflexibel. Sie werden dort ihren Einsatzbereich finden, wo ausschließlich Textdokumente verwaltet werden müssen. In Bereichen, wo neben Text auch andere Informationen verwaltet werden sollen, müssen neben IRS dann noch andere Systeme (z. B. RDBMS) unterhalten werden. Da ein ORDBMS mittels spezieller Zusatzmodule (siehe Abschnitt 3.4.2: »PLS Text DataBlade Modul«) die Funktionalitäten eines IRS annehmen kann, dürfte gerade in Arbeitsbereichen, wo mehr als nur Text-Retrieval gefragt ist, die Bedeutung reiner IRS ab und die von ORDBMS zunehmen.

3 Konzeption

3.1 Die Publikationen der GMD

Wie praktisch jede andere öffentliche Forschungsinstitution veröffentlicht das GMD-IPSI die Ergebnisse seiner Forschungsaktivitäten in Form schriftlicher Publikationen. Um einerseits verschiedenen Interessengruppen und andererseits unterschiedlichen Veröffentlichungsaspekten gerecht zu werden, stellt das IPSI verschiedene Publikationsarten zur Verfügung:

3.1.1 Selbständige Publikationen

Bei den selbständigen Publikationen handelt es sich um Veröffentlichungen, die von der GMD selbst oder ihren Mitarbeitern herausgegeben werden. Dabei kann zwischen Fortsetzungswerken oder Monographien unterschieden werden.

3.1.1.1 Fortsetzungswerke

3.1.1.1.1 GMD-Spiegel

Der GMD-Spiegel ist eine Zeitschrift, in der in Form von Kurzartikeln über die Forschungsarbeit der GMD berichtet wird. Die Artikel werden von Forschern der GMD verfaßt. Der GMD-Spiegel richtet sich im Vergleich zu praktisch allen anderen GMD-Publikationen an die breite Öffentlichkeit. Er erscheint viermal jährlich.

Der GMD-Spiegel enthält nicht nur Artikel aus dem IPSI, sondern auch aus den anderen Instituten der GMD. Deswegen ist er als zu verwaltende IPSI-Publikation eher uninteressant. Nur einzelne Artikel, die von IPSI-Forschern stammen, sind in unserem Falle von Interesse. Diese werden jedoch weiter unten behandelt.

3.1.1.1.2 GMD-Studien

Mittels der GMD-Studien veröffentlicht die GMD einerseits Proceedings von Tagungen, die von ihr bzw. ihren Instituten ausgerichtet wurden und andererseits Diplomarbeiten, die innerhalb der GMD verfaßt wurden und die thematisch von so großer Bedeutung sind, daß sie der Öffentlichkeit präsentiert werden sollten. Die GMD-Studien erscheinen in unregelmäßiger Folge.

3.1.1.1.3 GMD-Berichte

Mit den GMD-Berichten werden Dissertationen und Habilitationen veröffentlicht, die innerhalb der GMD verfaßt wurden. Auch sie erscheinen in unregelmäßiger Folge.

3.1.1.1.4 Arbeitspapiere der GMD

In den Arbeitspapieren der GMD erscheinen einerseits Preprints von Artikeln, die in anderen Publikationen (Zeitschriften, Proceedings usw.) erscheinen sollen. Andererseits bietet sich hier die Möglichkeit, spezialisierte Einzelergebnisse bestimmter Forschungsaktivitäten zu veröffentlichen. Auch die Arbeitspapiere der GMD erscheinen in unregelmäßiger Folge.

3.1.1.2 Monographien

Monographien innerhalb der GMD sind vorwiegend Bücher und Proceedings, die von GMD-Forschern herausgegeben werden. Es handelt sich hierbei um einen Grenzfall, da die Verbindung zur GMD über die Mitarbeiter hergestellt ist, jedoch der Inhalt der Monographien nicht unbedingt direkt mit den Forschungsaktivitäten der GMD zu tun haben muß.

3.1.2 Unselbständige Publikationen

Bei den unselbständigen Publikationen der GMD handelt es sich um von GMD-Forschern verfaßte Teile selbständiger Publikationen. Dabei gibt es zwei Möglichkeiten:

- Artikel in einer periodisch erscheinenden Publikation (Fachzeitschrift, Zeitung, Newletter usw.)
- Kapitel einer Monographie (Buch, Proceeding usw.)

Diese Form der Publikation macht einen Großteil des gesamten Publikationsaufkommens der GMD aus und kann daher als wichtigste Art der Veröffentlichung betrachtet werden.

3.1.3 Graue Literatur

Als graue Literatur können alle Informationsschriften angesehen werden, die nicht über offizielle Kanäle verteilt werden. Als hervorstechendes Beispiel können hier diejenigen Diplomarbeiten gesehen werden, die zwar in der GMD verfaßt, jedoch nicht in Form einer GMD-Studie veröffentlicht wurden. Trotzdem, daß diese Diplomarbeiten offiziell als nicht veröffentlichungswürdig angesehen werden, reflektieren sie einen Teil der Forschungsarbeit der GMD und haben somit auch eine Berechtigung, wenn auch nur über inoffizielle Kanäle, wie z. B. dem Internet, veröffentlicht zu werden.

3.2 Bisherige Bereitstellung der Publikationen der GMD

Grundsätzlich werden alle offiziellen Publikationen der GMD in Form von Papierdokumenten vorgehalten. Zur Recherche in den Publikationsbeständen sind bislang unterschiedliche Ansätze verfolgt worden:

3.2.1 PUBLICAT-System der GMD

Das PUBLICAT-System ist ein elektronisches Referenz-Datenbank-System, in dem die Gesamtbestände aller Institutsbibliotheken der GMD gespeichert sind. Dieses System ist GMD-intern über das GMD-Netzwerk und extern über das World Wide Web abfragbar.

Zur einfachen Suche wird dem Nutzer eine Suchmaske mit den Feldern AUTOR, TITEL, INSTITUTION und JAHR zur Verfügung gestellt. Nach Abschicken der Suchanfrage werden dem Nutzer die gefundenen Dokumente in Form kurzer Referenzen (AUTOR, TITEL, JAHR) dargestellt. Nun hat der Nutzer die Möglichkeit, sich für jedes einzelne Dokument eine detailliertere Referenz anzeigen zu lassen, die zusätzlich zu den vorgenannten Informationen auch noch die Signatur und den Standort des Schriftstückes in den verschiedenen GMD-Bibliotheken enthält. Im Falle einer GMD-Publikation kann der Nutzer sich von hier aus darüber informieren, in welcher Schriftenreihe (GMD-Studien, GMD-Berichte usw.) die Publikation erschienen ist, von welchem Verlag sie herausgegeben wird und welche ISB- bzw. ISS-Nummer sie besitzt.

Umfangreichere Suchmöglichkeiten bietet der Zugriff auf das PUBLICAT-System über die Retrievalsprache CCL (Common Command Language). Von einer einzelnen Befehlszeile aus, lassen sich CCL-Suchanfragen zu folgenden Dokumentinformationen formulieren:

- TI - Titel
- AU - Personen ohne Stammdatei
- CR - Körperschaft mit Stammdatei
- CO - Körperschaft ohne Stammdatei
- SE - Serie
- VO - Bandangaben
- PU - Verlag
- PY - Erscheinungsjahr
- SB - ISBN
- SS - ISSN
- UT - Kombisuche [OR-Verknüpfung von TI, AU, CO, PU]

Nachteil des PUBLICAT-Systems:

Da im PUBLICAT-System nur Referenzinformationen, nicht aber die Dokumente selbst gehalten werden, besteht weder bei der einfachen Suche über Formular noch bei der detaillierten Suche über CCL die Möglichkeit, die Volltexte der Dokumente zu durchsuchen. Es versteht sich, daß damit auch die Möglichkeit der Volltextausgabe der Dokumente auf dem Bildschirm wegfällt.

Das PUBLICAT-System ist vom WWW aus über den URL http://delphi.gmd.de erreichbar.

3.2.2 GMD-Schriften-Bestellung über das WWW

Unter dem URL http://wsv.gmd.de/aiw/schrift.htm wird die Möglichkeit angeboten, GMD-Schriften zu bestellen. Zu jeder Schriftenreihe (GMD-Studien, GMD-Berichte und Arbeitspapiere der GMD) kann jeweils eine Seite aufgerufen werden, die Referenzinformationen (Autor, Titel, Veröffentlichungsjahr) aller bisher erschienenen Ausgaben der jeweiligen Reihe in chronologischer Reihenfolge auflistet. Ist eine Publikation noch erhältlich, wird dies durch einen Hyperlink-Verweis angezeigt. Verfolgt man diesen Hyperlink, so springt man zu einem Formular, über welches man die gewählte Publikation über eMail bei der GMD bestellen kann.

Außer dem manuellen Absuchen der Listen besteht hier keinerlei Möglichkeit, im Publikationsbestand der GMD zu suchen.

3.2.3 Publikationsreferenzen des IPSI auf dem WWW

Auf dem WWW-Server des GMD-IPSI können mehrere Seiten eingesehen werden, die Referenzlisten der Publikationen der einzelnen Arbeitsbereiche enthalten. Manche Publikationen lassen sich als elektronische Version auf den lokalen Rechner herunterladen. Doch auch hier fehlt jegliche Möglichkeit der gezielten Dokumentensuche. Außerdem sind die von den Arbeitsbereichen zur Verfügung gestellten Referenzlisten unvollständig.

Auf die Referenzlisten des IPSI kann unter http://www.darmstadt.gmd.de/IPSI/pub.html zugegriffen werden.

3.2.4 VISIT-Diplomarbeiten auf dem WWW

Unter dem URL http://www-cui.darmstadt.gmd.de/visit/People/FormerMembers läßt sich eine Seite einsehen, auf dem die ehemaligen Diplomanden der Abteilung VISIT aufgelistet werden. Der Großteil der betreffenden Diplomarbeiten kann von hier aus im PostScript- bzw. Portable-Document-Format auf den lokalen Rechner heruntergeladen werden. Eine Suchmöglichkeit wird jedoch auch hier nicht angeboten.

Fazit:

In der GMD gibt es zur Zeit keine Möglichkeit, Publikationen (und damit sollen die Dokumente selbst gemeint sein) elektronisch zu verwalten und einem breiten Publikum über Volltextrecherche zur Verfügung zu stellen. Es werden lediglich Referenzinformationen angeboten. Diese können einerseits in Form von HTML-Seiten eingesehen werden. Andererseits werden Referenzen auf die Bibliotheksbestände auf einem Information-Retrieval-System (PUBLICAT) vorgehalten, welches jedoch auch keine Möglichkeit der Volltext-Recherche bietet.

3.3 Definition eines Publikations-Management-Systems für das GMD-IPSI

Wie aus dem vorigen Abschnitt zu ersehen ist, gibt die GMD bislang weder seinen Mitarbeitern noch externen Nutzern ein System zur Hand, mit dessen Hilfe eine effektive Suche innerhalb des GMD-Publikationsbestandes möglich ist.

Unser Ziel ist es, ein elektronisches Publikations-Management-System zu konzipieren und implementieren, das folgende Eigenschaften aufweist:

3.3.1 Globale Zugriffsmöglichkeiten

Um zu gewährleisten, daß ein möglichst breites Publikum den Publikationsbestand der GMD nutzen kann, sollte das System über ein allgemein anerkanntes und für die breite Öffentlichkeit zugängliches Medium zur Verfügung gestellt werden. Die in den letzten Jahren stetig ansteigende Beliebtheit des Internet resultiert aus dem Anfang der 90er Jahre eingeführten Informationsdienst World Wide Web. Das WWW macht aufgrund seiner einfachen Bedienbarkeit das Internet für viele Nutzer, die sich normalerweise an keinen Computer herantrauen, zugänglich. Seit der Einführung des World Wide Web nimmt die Zahl der Internet-Anwender rapide zu. Einige Statistiken gehen davon aus, daß mittlerweile über das WWW ca. 50 Millionen Nutzer erreicht werden können. Es wäre von daher ein sinnvoller Schritt, das geplante Publikations-Management-System über das WWW zugänglich zu machen.

3.3.2 Umfangreiche Suchmöglichkeiten

3.3.2.1 Text-Retrieval

Über eine vorgefertigte Suchmaske sollte der Anwender Textsuchen im Dokumentenbestand vornehmen können. Hierbei sollte er die Möglichkeit bekommen, mittels Worttrunkierung und durch den Einsatz von Abstandsoperatoren (z. B. NEAR oder ADJ) und Boole'schen Mengenoperatoren (AND, OR, NOT) seine Suchanfrage zu erweitern bzw. einzugrenzen.

Zusätzlich sollte der Anwender Aspekte wie »Veröffentlichungsjahr« und »Autor« in seine Suche einfließen lassen können.

Letztlich sollte der Anwender bestimmen können, von welcher Publikationsart (Artikel, Monographie, Diplomarbeit usw.) die zurückgegebenen Dokumente sein sollen.

3.3.2.2 Inhaltliche Verknüpfung der Dokumente

Eine herkömmliche Textsuche findet oftmals nicht alle Dokumente, die für den Anwender relevant sein könnten. So kann es beispielsweise vorkommen, daß bestimmte, für den Anwender relevante Dokumente nicht die Wörter enthalten, nach denen gesucht wurde. Hierzu ein Beispiel:

Nehmen wir an, ein Nutzer interessiert sich für das IPSI-Projekt LyberWorld. Ziel des LyberWorld-Systems ist die Konzeption einer graphischen 3D-Benutzungsschnittstelle für Information-Retrieval-Systeme. Nehmen wir an, der Anwender gibt als Suchanfrage die Wörter »lyberworld«, »3d« und »user interface« ein. In einem der vom System zurückgegebenen Dokumente steht ein Hinweis darauf, daß das LyberWorld-System auf die Zusammenarbeit mit dem objektorientierten Datenbanksystem VODAK, ebenfalls ein IPSI-Projekt, getestet wurde. Zu VODAK gibt es innerhalb des Publikations-Management-Systems auch verschiedene Dokumente. Es ist immerhin möglich, daß sich der Anwender nach Lektüre des genannten LyberWorld-Dokumentes auch für VODAK-Dokumente interessiert. Bei einem herkömmlichen Volltext-Suchsystem müßte der Anwender eine zweite Suchanfrage stellen, welche die VODAK-Dokumente zurückgibt.

Es wäre also sinnvoll, dem Anwender gleich bei der Rückgabe der LyberWorld-Dokumente einen Hinweis darauf zu geben, daß innerhalb des Systems Dokumente vorhanden sind, die zwar nicht direkt auf seine Suchanfrage zutreffen, jedoch inhaltlich mit den gefundenen Dokumenten verknüpft sind.

Diese inhaltlichen Verknüpfungen sollten vom System selbständig hergestellt werden, und zwar einerseits bezogen auf die schon gespeicherten Dokumente und andererseits bezogen auf Dokumente, die dem System neu hinzugefügt werden.

3.3.3 Downloading der Dokumente

Dem Anwender soll die Möglichkeit gegeben werden, gefundene Dokumente in Form elektronischer Dateien auf seinen lokalen Rechner herunterzuladen. Hierzu sollten die Dokumente in einem weitestgehend plattformunabhängigen Format (z. B. Acrobat-PDF) zur Verfügung stehen.

3.3.4 Einfache Pflege des Dokumentenbestandes

Das Hinzufügen neuer Dokumente zum Bestand sollte so einfach und unkompliziert sein, daß dies auch von nicht mit dem System vertrauten Personen übernommen werden kann. Dies setzt voraus, daß ein Großteil der Verwaltungsarbeiten (Einfügen der Datensätze in die Datenbank, Aktualisierung der Indizes und Herstellen der inhaltlichen Verknüpfungen zwischen den Dokumenten) vom System selbsttätig durchgeführt werden.

3.4 Zur Verfügung stehende Software-Komponenten

3.4.1 Das ORDBMS »Illustra Server«

Illustra Server von der Firma Illustra ist eines der wenigen auf dem Markt erhältlichen objektrelationalen Datenbank-Management-Systeme. Wie schon im Abschnitt 2.5.2.3 beschrieben, handelt es sich bei objektrelationalen DBMS um relationale DBMS, die dahingehend erweitert wurden, daß Aspekte, die aus der Welt der objektorientierten DBMS bekannt sind, hinzugefügt wurden. Somit kann Illustra Server Funktionen beider Welten, relational und objektorientiert, anbieten:

Illustra Server ist ein relationales DBMS

- Mit Standard-SQL steht ein mächtiges Werkzeug zur Datenabfrage und –manipulation zur Verfügung
- Transaktions- und Recovery-Mechanismen sorgen dafür, daß auch bei simultanem Zugriff durch mehrere Nutzer die Datenkonsistenz ständig gewährleistet ist.

- Durch Skalierbarkeit läßt sich Illustra Server problemlos an bestehende Hardwareumgebungen und die Bedürfnisse des Nutzers anpassen.

Illustra Server ist ein »objektorientiertes« DBMS

- Durch die Möglichkeit, Datentypen vom Datenbank-Entwickler selbst definieren zu lassen, kann Illustra Server praktisch alle Arten von Datentypen unterstützen.
- Die Möglichkeit der Entwicklung spezieller Speicherungs- und Zugriffsmethoden durch den DB-Entwickler hat zur Folge, daß schnell und effizient auf komplexe Datentypen zugegriffen werden kann.
- Funktionen wie die Einkapselung von Objekten oder Vererbung von Objekteigenschaften ermöglichen eine einfachere und realitätsnähere Datenbankmodellierung.
- Durch die Vergabe von Objekt-IDs bei der Speicherung kann direkt auf jedes komplexe Datum zugegriffen werden.

3.4.1.1 DataBlade-Technologie

Die Unterstützung komplexer Datentypen wird bei Illustra Server durch die sog. DataBlade-Technologie ermöglicht. Bei DataBlade-Modulen handelt es sich um Software-Module, mit deren Hilfe die Funktionalität von Illustra Server erweitert wird. Sie enthalten die Definition eines neuen Datentyps und für ihn optimierte Speicherungs- und Zugriffsmethoden. Im direkten Vergleich zu Objektaufsätzen, die zu herkömmlichen RDBMS angeboten werden, stellt sich die DataBlade-Technologie folgendermaßen dar:

- **Objektaufsätze für RDBMS** täuschen dem Anwender Objektfähigkeiten nur vor. In Wirklichkeit werden komplexe Daten nach wie vor als BLOBs (Binary Large Objects) gespeichert. Bei einer inhaltsbasierten Anfrage an diese »angeblichen Objekte« müssen die BLOBs dann Stück für Stück vom Objektaufsatz in Objekte umgewandelt werden, bevor sie durchsucht werden können. Es leuchtet ein, daß diese Umwandlung während der Laufzeit ein sehr zeit- und ressourcenaufwendiger Prozeß ist. Da das DBMS nach wie vor lediglich über seine fixierten Zugriffsmethoden verfügt, muß auch der Suchprozeß vom Objektaufsatz übernommen werden. Somit können die speziellen Anfrageoptimierungsmechanismen, die in einem RDBMS vorhanden sind, nicht genutzt werden. Dies führt wiederum zu einer Verlangsamung des Suchprozesses. Für inhaltsbasierte Anfragen an große Mengen komplexer Daten sind durch Objektaufsätze erweiterte RDBMS daher nicht geeignet.
- Bei der Installation eines **DataBlade-Moduls** werden die in ihm enthaltenen Speicherungs-, Indizierungs- und Zugriffsmethoden in den Kern des DBMS integriert. Somit stehen diese Mechanismen dem DBMS selbst zur Verfügung. Komplexe Daten werden dann als Strukturen gespeichert, deren interne Details für Funktionen, die auf dem DBMS laufen, direkt zugänglich sind. Dazu gehört neben Speicherung, Indizierung und Zugriff auch die Optimierung von Anfragen hinsichtlich des Eingabe/Ausgabe- und CPU-Aufwands. Die Funktionen werden direkt auf dem Server ausgeführt – in unmittelbarer Datennähe. Die Effizienz läßt sich dadurch in hohem Maße steigern, denn die Daten müssen nicht mehr, so wie bei einem Objektaufsatz, für eine Bearbeitung erst aus dem Server herausgeladen werden.

In den beiden folgenden Abschnitten werden zwei DataBlade-Module beschrieben, die für die Implementierung des in dieser Ausarbeitung konzipierten Publikations-Management-Systems genutzt werden sollen:

3.4.2 Das PLS »Text DataBlade« Modul

Durch die Erweiterung des Illustra Server mit dem PLS Text DataBlade Modul der Firma Personal Library Solutions können den Relationen, in denen Textdokumente gespeichert werden, spezielle Textindizes hinzugefügt werden. Diese Indizes werden folgendermaßen erstellt:

1. Der in den Dokumenten enthaltene Text wird in einzelnen Wörter aufgesplittet (Parsing).
2. Dort werden die einzelnen Wörter auf ihre Stammform zurückgeführt (Stemming).
3. Die zurückgeführten Wörter werden mit einer Stoppwortliste abgeglichen, d. h. daß unter- und übersignifikanten Wörter aus dem Wortbestand gelöscht werden.
4. Die übriggebliebenen Wörter werden mit Angabe des genauen Standorts in einen Index geschrieben, welcher innerhalb der jeweiligen Relation abgespeichert wird.

Optional läßt sich die Stemming-Funktion auch deaktivieren, so daß keine Stammwort-Reduktion erfolgt.

Derzeit werden vom PLS Text DataBlade Modul die Dokumentformate ASCII und HTML unterstützt. Für zukünftige Versionen ist die Unterstützung anderer populärer Dokumentformate, wie z. B. PostScript, Portable Document Format, Microsoft Word DOC, Adobe Framemaker MIF usw. geplant.

3.4.2.1 Text-Retrieval mit dem PLS Text DataBlade

Die mit dem PLS Text DataBlade (nachfolgend kurz PLS genannt) erstellten Textindizes können auf vielfältige Art und Weise durchsucht werden. Da es den Rahmen dieser Ausarbeitung sprengen würde, alle Suchmöglichkeiten des PLS darzustellen, soll im Folgenden nur auf die Aspekte eingegangen werden, die in die Implementierung des Publikations-Management-Systems eingeflossen sind.

3.4.2.1.1 Suchverfahren

Das PLS stellt dem Anwender zwei unterschiedliche Suchverfahren zur Verfügung:

3.4.2.1.1.1 Best-Match-Suche

Mit der Best-Match-Suche werden Dokumente gefunden, deren Textinhalt eine hohe Relevanz zur Suchanfrage aufweist. Es ist das im Bereich des Information-Retrieval wohl grundsätzlichste Suchverfahren.

Bsp.: Mit der Suchanfrage »`computer AND information`*« werden Dokumente gefunden, welche die Wörter* **computer** *und* ***information*** *enthalten.*

3.4.2.1.1.2 Konzeptsuche

Die vom PLS angebotene Konzeptsuche läuft in zwei Arbeitsschritten ab:

- Schritt 1: Das PLS sucht nach Dokumenten, deren Textinhalt eine hohe Relevanz zur Suchanfrage aufweist und ermittelt, welche Wörter in diesen Dokumenten möglichst häufig und möglichst regelmäßig vorkommen.
- Schritt 2: Das PLS sucht nach Dokumenten, in denen möglichst viele der ermittelten Wörter vorkommen und gibt diese als Suchergebnis zurück.

Bsp.: Der Anwender gibt als Suchanfrage »`3d AND information ADJ retrieval`*« ein. Das PLS ermittelt alle Dokumente, die auf diese Suchanfrage zutreffen. In diesen kommt der Begriff »LyberWorld« (ein IPSI-Projekt zum Thema »Dreidimensionale graphische Benutzerschnittstellen für Information-Retrieval-Systeme«) oft und mit hoher Regelmäßigkeit vor. Das PLS sucht nun nach Dokumenten, welche den Begriff »LyberWorld« enthalten und gibt diese als Suchergebniszurück.*

Die Konzeptsuche bietet somit Anwendern, die den Wortschatz einer Dokumentendatenbank nicht kennen, eine Möglichkeit, trotz ihrer fachlichen Unkenntnis Suchergebnisse mit u. U. hoher Relevanz zu erhalten.

3.4.2.1.2 Wie wird mit dem PLS Text DataBlade gesucht?

Mit der Funktion `PlsOidRankQuery` läßt sich die Objekt-ID jedes Dokumentes innerhalb einer mit dem PLS vorbereiteten Illustra-Datenbank ausgeben, das auf eine bestimmte Suchanfrage zutrifft.

Anmerkung:
Jeder Datensatz einer Illustra-Datenbank erhält bei der Speicherung eine eindeutige Objekt-ID (oid), mittels derer er jederzeit identifiziert werden kann.

Außerdem wird zu jedem gefundenen Dokument ein Ranking-Wert zurückgegeben, der Auskunft darüber geben soll, wie gut das Dokument auf die Suchanfrage zutrifft.

Anmerkung:
Da in der Dokumentation des PLS keine Angaben darüber gemacht wurden, wie der stets dreistellige Ranking-Wert interpretiert werden kann (ist es die Anzahl der vorkommenden Suchbegriffe, sind es prozentuale Angaben, ... ???) wurde bei der Implementierung darauf verzichtet, sich diesem zu bedienen.

Eine Suchanfrage mit `PlsOidRankQuery` wird in folgendem Format gestellt:

```
PlsOidRan          uer    (    indexname    ,    (suchan    rage)    )
```

Als Beispiel soll ein Index namens `articleIndex` benutzt werden, in dem Dokumente innerhalb einer Relation namens `articleTable` invertiert wurden. Die Suchanfrage lautet »`computer AND 3d`«:

```
PlsOidRan          uer    (    article    ndex    ,    (computer A                d)    )
```

Die angezeigte Suchanfrage würde das Best-Match-Verfahren nutzen. Wollte man eine Konzeptsuche durchführen, so müßte man ein Ausrufungszeichen direkt hinter die in Klammern gesetzten Suchanfrage stellen:

```
PlsOidRan          uer    (    article    ndex    ,    (computer A                d)
```

Nun genügt diese Syntax aber noch nicht, um aus einer SQL-basierten Datenbank Ergebnisse zu erhalten. Der `PlsOidRankQuery`-Block muß in ein `SELECT`-Statement integriert werden. Dies geschieht folgendermaßen:

Da `PlsOidRankQuery` Objekt-IDs (oid) zurückgibt, wird sie mit der `FROM`-Klausel als Datenquelle in ein `SELECT`-Statement integriert. Nehmen wir an, wir möchten den Titel (`title`) und das Veröffentlichungsjahr (`year`) jedes Dokumentes zurückgeliefert bekommen, das auf die Suchanfrage »`computer AND 3d`« zutrifft. Relation ist wieder `articleTable`, der Index wieder `articleIndex`:

```
      ELECT a      title,
       a            ear
FRO        articleTable a,
       PlsOidRan           uer    (    article    ndex    ,    (computer A                d)
            ERE  a           pls_oid    a    oid;
```

Es wird also der Titel und das Veröffentlichungsjahr aller Datensätze zurückgegeben, welche dieselbe Objekt-ID aufweisen, wie die von `PlsOidRankQuery` gefundenen Dokumente (die ja innerhalb der Relation `articleTable` gespeichert sind).

3.4.2.1.2.1 Verwendung von Suchoperatoren

Durch die Verwendung von Suchoperatoren kann einer Retrieval-Anfrage höhere Genauigkeit verliehen werden. Das PLS stellt eine reichhaltige Palette an Suchoperatoren zur Verfügung, die im Folgenden kurz erläutert werden:

3.4.2.1.2.1.1 Boole'sche Mengenoperatoren

- **AND**
  »`computer AND 3d`« findet alle Dokumente, welche beide Wörter, **computer** *und* **3d** vorweisen können.

- **OR**
  »`computer OR 3d`« findet alle Dokumente, welche entweder **computer** *oder* **3d** oder **computer** *und* **3d** vorweisen können.
- **NOT**
  »`computer NOT 3d`« findet alle Dokumente, welche **computer** vorweisen, jedoch *nicht* **3d** enthalten.

3.4.2.1.2.1.2 Abstandsoperatoren

- **Adjacency (`ADJ`)**
  Wird der Abstandoperator `ADJ` zwischen zwei Suchbegriffe gesetzt, so werden Dokumente gefunden, in denen die beiden Suchbegriffe direkt nebeneinander vorkommen. Stoppwörter zwischen den beiden Suchbegriffen werden dabei ignoriert. `ADJ` ist unidirektional, von links nach rechts. Dies bedeutet, das mit einer Suchanfrage »`blue ADJ red`«
  - »The car is blue and red« gefunden wird
  - »The car is red and blue« nicht gefunden wird.
- **Within (W/n)**
  Within ist eine erweiterte Version von `ADJ`. Mit n kann ein Zahlenwert angegeben werden, der bestimmt, wie weit die durch Within miteinander verbundenen Suchbegriffe voneinander entfernt sein dürfen, d. h. wieviele Nicht-Stoppwörter zwischen diesen vorkommen dürfen. Auch Within ist unidirektional, von links nach rechts.

  Bsp.: Mit einer Suchanfrage »`mercedes w/3 world`« wird
  - »Mercedes are the best cars of the world« gefunden
  - »The best cars of the world are build by Mercedes« nicht gefunden
- **Near (`NEAR/n`)**
  Near ist eine bidirektionale Version des Within-Operators. Mit ihr würden bei einer Suchanfrage »`mercedes near/3 world`«
  - sowohl »Mercedes are the best cars of the world«
  - als auch »The best cars of the world are build by Mercedes«
  - gefunden werden.

3.4.2.1.2.1.3 Wildcard-Operatoren

Wildcard-Operatoren können in einer Suchanfrage einzelne Zeichen oder ganze Zeichenketten ersetzen.

- `?` – ersetzt einzelne Zeichen (»`?ate`« findet **hate** und **late**)
- `*` - ersetzt ganze Zeichenketten (»`*late`« findet **relate** und **translate**)

3.4.2.1.2.1.4 Feldoperator /f

Nehmen wir an, in dem PLS-Index `articleIndex` sind die Attribute `title`, `abstract` und `fulltext` der Relation `articleTable` invertiert. Mit dem Feldoperator `/f` könnten nun diese Attribute einzeln durchsucht werden. Dazu ein Beispiel:

Die Suchanfrage »`(computer AND 3d) /f:abstract`« würde alle Dokumente finden, in deren Abstract die Begriffe **computer** und **3d** auftauchen.

Die `/f`-Option läßt die Angabe mehrerer, durch Komma voneinander getrennte, Attibutangaben zu (`/f:abstract,fulltext`).

3.4.2.1.2.1.5 Kombination der Operatoren

Alle genannten Suchoperatoren können innerhalb einer Suchanfrage beliebig miteinander kombiniert werden. Dabei haben die einzelnen Operatoren unterschiedliche Gewichtungen. Diese zeigt sich in folgender Reihenfolge (absteigende Gewichtung nach unten):

1. Feldoperator
2. Abstandsoperatoren
3. NOT
4. AND
5. OR

Dies bedeutet beispielsweise, daß die Suchanfrage »`computer AND 3d OR data`« Dokumente findet, die entweder **computer** *und* **3d** *oder* **data** oder **computer** *und* **3d** *und* **data** vorweisen.

3.4.3 Das Illustra »Web DataBlade« Modul

Das Web DataBlade Modul von Illustra ist eine Entwicklungs- und Verwaltungsumgebung für sog. Web Applications. Mittels Web Applications lassen sich Daten, die innerhalb einer Illustra-Datenbank gespeichert sind, auf dem World Wide Web präsentieren.

Die Web Applications, HTML-Dokumente, die mit speziellen Tags zur Datenbankabfrage ausgestattet sind, werden in eigenen Relationen gespeichert und können alle Inhalte besitzen, die man von herkömmlichen HTML-Seiten her kennt (Grafiken, Java-Applets, JavaScript-Routinen usw.).

Durch die direkte Verbindung mit dem Illustra-System ist es möglich, Sicherheitsvorkehrungen zu treffen (z. B. Paßwortabfragen), um bestimmte Teile einer Web-Präsentation nur bestimmten Nutzern zugänglich zu machen.

Das Web DataBlade Modul arbeitet nahtlos mit anderen DataBlade-Modulen zusammen, so daß deren Funktionalitäten auch für eine Web-Präsentation genutzt werden können.

Dadurch, daß das Web DataBlade Modul fest in das Illustra-System integriert ist, stehen auch Web Applications DBMS-Merkmale wie Transaktions-Management oder Data-Recovery zur Verfügung.

3.4.3.1 Architektur des Web DataBlade Moduls

Das Web DataBlade Modul besteht aus fünf Komponenten:

- **Webdriver:** Webdriver ist ein CGI-Programm, daß die Verbindung zwischen dem Web-Server und der Illustra-Datenbank herstellt.
- **Webdaemon:** Webdaemon ist eine Anwendung, die im Hintergrund darauf wartet, vom Webdriver die Anfrage nach einer Application Page zu erhalten. Webdaemon gibt diese Anfrage an die WebExplode-Funktion weiter. Ergebnisdaten, die von der WebExplode-Funktion im HTML-Format an den Webdaemon zurückgegeben werden, leitet dieser an den Webdriver weiter.
- **WebExplode-Funktion:** Die WebExplode-Funktion durchsucht eine vom Webdaemon angefragte Application Page nach SQL-Kommandos, führt diese in der Illustra-Datenbank aus und formatiert die Ergebnisdaten nach den in der Application Page enthaltenen Anweisungen und gibt diese an den Webdaemon zurück.

- **Web-DataBlade-Tags und -Attribute:** Das Web DataBlade Modul stellt eigene HTML-Tags zur Verfügung, die dazu genutzt werden, in Application Pages SQL-basierte Datenbankanfragen durchzuführen. Diese speziellen Tags werden weiter unten näher beschrieben.
- **Web-DataBlade-Funktionen:** Mittels spezieller Prozeßfunktionen ist es möglich, Ergebnisdaten ansprechend zu formatieren und verschiedene Operationen an diesen durchzuführen.

Das folgende Diagramm erläutert die Funktionsweise des Web DataBlade Moduls:

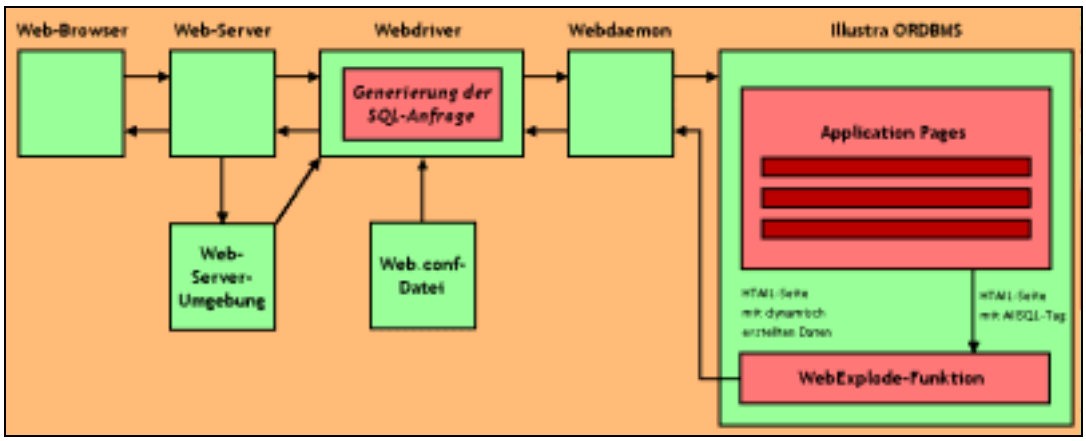

Abbildung 3: Funktionsweise des Web DataBlade

Ein Web-Browser macht eine Anfrage nach einer Application-Page an den Web-Server. Dieser gibt die Anfrage an den Webdriver weiter. Der Webdriver formuliert aus dieser Anfrage ein SQL-Kommando, welches er an den Webdaemon weitergibt. Der Webdaemon eröffnet eine Verbindung zur Illustra-Datenbank, startet die WebExplode-Funktion und übergibt das SQL-Statement zum Auffinden der gewünschten Application Page an diese weiter. Die WebExplode-Funktion greift sich die angefragte Application Page aus der Datenbank, führt die darin enthaltenen SQL-Kommandos in der Illustra-Datenbank aus, formatiert die Ergebnisdaten entsprechend den Vorgaben aus der Application-Page und gibt sie im HTML-Format an den Webdaemon zurück. Der Webdaemon leitet die HTML-Daten an den Webdriver weiter, welcher sie wiederum an den Web-Server weitergibt. Der Web-Server transferiert die Daten an den Web-Browser, der sie entsprechend interpretiert und anzeigt.

3.4.3.2 Web-DataBlade-Tags

Das Web-DataBlade-Modul stellt eine Reihe spezieller HTML-Tags zur Verfügung, die es ermöglichen, innerhalb von Application-Pages SQL-Statements an eine Illustra-Datenbank abzusetzen, Rückgabewerte zu verarbeiten und HTML-konform zu formatieren, um sie dem Anwender innerhalb seines Web-Browsers anzuzeigen.

Im folgenden werden die einzelnen Tags näher erläutert.

3.4.3.2.1 MISQL

Mit dem MISQL-Tag können beliebige SQL-Statements an eine Illustra-Datenbank abgesetzt werden. Dabei spielt es keine Rolle, ob dies SQL-Statements sind, die grundsätzlich von Illustra zur Verfügung gestellt werden, oder solche, die erst durch die Installation eines speziellen DataBlade-Moduls (z. B. PLS Text DataBlade) hinzugekommen sind. Somit stellt das MISQL-Tag eine vollwertige SQL-Kommunikationsschnittstelle zum Illustra-Server dar.

Zwischen dem MISQL-Anfangs-Tag `<?MISQL>` und dem MISQL–End-Tag `<?/MISQL>` können die vom Illustra-Server zurückgegebenen Werte formatiert werden. Es folgt ein einfaches Beispiel zur Veranschaulichung:

Aus der Tabelle `personnelTable` soll der Vor- und Nachname aller Personen zurückgegeben werden, die den Status `student` besitzen. Die Werte sollen, nach Nachname sortiert, in Form einer HTML-Tabelle dargestellt werden:

```
TABLE BOR      ER
T                achname      T                orname      TR
                            L           L                ELECT      irstname, lastname
      FRO       personnelTable
                ERE  status           student      OR      ER BY lastname;
T                          T                              TR
                                 L
      TABLE
```

Das Grundgerüst für die HTML-Tabelle wird außerhalb der MISQL-Tags aufgebaut. Nur die Tabellenelemente, die für die eigentliche Darstellung der Tabellenwerte zuständig sind, werden zwischen die MISQL-Tags gesetzt. Welcher Wert wohin gesetzt wird, wird durch einfach durchnumerierte Variablen (in der Reihenfolge, in der die entsprechenden Attribute in der SQL-Anfrage gesetzt wurden) bestimmt.

Das Ergebnis zeigt sich folgendermaßen:

Nachname	Vorname
Schmidt	Rudolf
Sauerbier	Marianne

Tabelle 3: Beispiel für eine durch MISQL formatierte Ausgabe

3.4.3.2.2 MIVAR

Mit dem MIVAR-Tag können Variablen neu definiert und Werte zugewiesen werden. Außerdem wird das MIVAR-Tag dazu benutzt, Variablenwerte außerhalb von MISQL-Blöcken anzuzeigen. Zur Veranschaulichung auch hier ein Beispiel:

Die erste SQL-Anfrage in einer Application-Page (im Folgenden kurz AppPage genannt) ermittelt das höchste Jahreseinkommen, welches für einen Mitarbeiter in der Tabelle personnelTable gespeichert ist.

```
                          L           L                ELECT      A      (income) FRO
                                L
```

Der zurückgegebebene Wert soll in einer später folgenden SQL-Abfrage als Abfragewert weiterverwendet werden. Dazu ist es nötig, den Rückgabewert einer Variablen `$INCOME` zuzuweisen:

```
                          AR       A      E                       CO      E
```

In der folgenden SQL-Anfrage (Ermittlung aller Angestellten, die mindestens dieses Jahreseinkommen vorweisen können) kann der in der Variable `$INCOME` enthaltene Wert eingesetzt werden:

```
      L
                          L           L                ELECT      irstname, lastname
      FRO       personnelTable
                ERE  income                          CO      E;
L                       ,
                                L
            L
```

Anmerkung:
Der SQL-Profi wird hier vielleicht entgegnen, daß die zwei vorstehenden Abfragen auch in einer einzigen SQL-Routine abgearbeitet werden könnten. Dem muß zwar zugestimmt werden. Das Beispiel sollte jedoch nur zur Veranschaulichung des MIVAR-Tags dienen. Auf einen ausgefeilten SQL-Code wurde deswegen dabei kein großer Wert gelegt.

Wollte man am Ende der AppPage die Variable `$INCOME` ausgeben, so wäre Folgendes vorstellbar:

```
                          as h      chste      ahresein      ommen ist
```

3.4.3.2.3 MIBLOCK

Mit dem MIBLOCK-Tag ist es möglich, die Ausführung bestimmter Teile einer AppPage an bestimmte Bedingungen zu knüpfen. Auch hierzu ein Beispiel:

Nehmen wir an, ein Anwender kann in einem Suchformular einen Wert für ein Jahreseinkommen angeben, um auf einer Ergebnisseite die Namen aller Personen, die dieses Jahreseinkommen vorweisen können, angezeigt zu bekommen. Der im Suchformular eingegebene Wert wird an die für die Suche bzw. Ergebnisanzeige zuständige AppPage durch die Variable `$INCOME` übergeben.

Zuerst würde ein `SELECT COUNT` durchgeführt, um festzustellen, ob es überhaupt Personen mit dem angegebenen Einkommen gibt:

```
L         L         ELECT CO         T (       ) FRO
      L
```

Wenn es keine entsprechende Person gibt, so wird ein Rückgabewert von 0 geliefert. In diesem Fall wird nur der erste der beiden folgenden Anwendungsblöcke ausgeführt. Ist der Rückgabewert größer als 0 (d. h. es gibt Personen mit dem angegebenen Einkommen), wird nur der zweite Anwendungsblock ausgeführt (Suche und Anzeige des Namens jeder Person, die ein entsprechendes Einkommen vorweisen kann).

```
BLOC       CO                    (E     ,      ,        )
Es       urden      eine Personen ge     unden, die ein      ahresein      ommen
AR                 CO      E                    AR      vor      eisen
           BLOC

BLOC       CO                    (       ,      ,       )
Folgende Personen      eisen ein      ahresein      ommen von mind CO      E
L
             L         L         ELECT      irstname, lastname
FRO       personnelTable
ERE  income                    CO      E;
L         ,
                  L
L
           BLOC
```

3.4.3.2.4 Prozeßfunktionen

Für die Definition von MIBLOCK-Bedingungen und für die Bearbeitung von Rückgabewerten innerhalb von MISQL-Blöcken stellt das Web-DataBlade-Modul eine ganze Reihe von Prozeßfunktionen zur Verfügung. Da die Erklärung aller möglichen Prozeßfunktionen den Rahmen dieser Ausarbeitung sprengen würde, beschränkt sich die folgende Aufstellung auf die Prozeßfunktionen, die in der später dargestellten Implementierung genutzt wurden:

Prozeßfunktion	Funktionserklärung
`$(<,val1,val2)`	Ist der Zahlenwert `val1` kleiner als der Zahlenwert `val2`, wird `1` zurückgegeben; andernfalls wird `0` zurückgegeben.
`$(+,val1,val2,...,valn)`	Die Summe der Zahlenwerte `val1, val2, ...`, `valn` wird zurückgegeben.
`$(!=,val1,val2)`	Wenn der Zahlenwert `val1` nicht gleich `val2` ist, wird `1` zurückgegeben; andernfalls wird `0` zurückgegeben.
`$(EQ,string1,string2)`	Wenn `string1` und `string2` identisch sind, wird `1` zurückgegeben; andernfalls wird `0` zurückgegeben.

`$(IF,expr,dotrue)`	Wenn `expr` nicht `0` ist, wird `dotrue` evaluiert und zurückgegeben.
`$(IF,expr,dotrue,dofalse)`	Wenn `expr`, nicht `0` ist, wird `dotrue` evaluiert und zurückgegeben; andernfalls wird `dofalse` evaluiert und zurückgegeben.
`$(NOT,value)`	Die logische Negation von `value`
`$(NXST,varname)`	Trifft zu, wenn die Variable `varname` nicht existiert (d. h. wenn ihr kein Wert zugewiesen wurde)
`$(POSITION,string1,string2)`	Gibt die Anfangsposition von `string2` innerhalb von `string1` an. Wird `string2` nicht gefunden, wird `0` zurückgegeben)
`$(SETVAR,varname,value)`	Setzt die Variable `varname` auf den Wert `value`.

Tabelle 4: Die wichtigsten Prozeßfunktionen des Web DataBlade Moduls

3.4.3.2.5 Webdriver-Umgebungsvariablen

Wie herkömmliche Web-Pages, werden auch AppPages über einen URL aufgerufen. Da die AppPages jedoch in einer Illustra-Datenbank gehalten werden, muß ein URL, der eine AppPage aufruft, spezielle Informationen mit sich führen:

- **URL des Webdriver**
  Da alle AppPages über den Webdriver aufgerufen werden, muß die URL zu diesem den Anfang jeden Aufrufes einer AppPage sein. (Bsp.: »`http://www.test.com/Webdriver.cgi`«)
- **MItab**
  Als zweiter Wert muß der Name der Datenbank-Tabelle angegeben werden, in der die gewünschte AppPage gespeichert ist. (Bsp.: »`MItab=webPages`«)
- **MIcol**
  Als dritter Wert muß der Name des Attributes angegeben werden, in welchem der HTML-Code der AppPage steht. (Bsp.: »`MIcol=source`«)
- **MInam**
  Als vierter Wert muß der Name des Schlüsselattributes der Tabelle, in der die AppPage gespeichert ist, angegeben werden. (Bsp: »`MInam=id`«)
- **MIval**
  Der letzte Wert ist der Schlüsselattributwert der AppPage (also ihr Name).
  (Bsp.: »`MIval=welcome`«)

```
Der komplette URL der Beispiel-AppPage wäre also:
http                                    test      com            ebdriver      cgi
```

Der Webdriver formt aus den MI-Variablen eine SQL-SELECT-Anweisung, mit der die gewünschte AppPage von der WebExplode-Funktion aus der Illustra-Datenbank herausgelesen wird:

```
      ELECT source
FRO             ebPages
          ERE                                elcome      ;
```

Innerhalb der Webdriver-Konfigurationsdatei »Web.conf« können für `MItab`, `MIcol` und `MInam` Standardwerte vorgegeben werden. Dies empfiehlt sich, wenn alle AppPages (oder ein großer Teil davon) in derselben Datenbanktabelle gehalten werden. Somit erspart man dem Anwender die Eingabe eines wirklich sehr langen URL. Auf unser Beispiel bezogen müßte die URL dann nur noch folgendermaßen aussehen:

```
http                                    test      com          ebdriver     cgi          ival
```

Zusätzlich kann in der Datei »Web.conf« die URL des Webdriver als Standardvariable `$WEB_HOME` festgelegt werden. Soll innerhalb einer AppPage ein Hyperlink-Verweis auf eine andere AppPage der gleichen Datenbank gesetzt werden, muß nur noch folgendes angegeben werden:

```
                          AR
    A     REF                          EB_    O    E              val          elcome
                              AR
```

Für Large-Objects (Grafiken, Klangdateien, Videoclips usw.) wird eine ähnliches Verfahren angewandt. Hier heißen die Variablen `MItabObj`, `MIcolObj`, `MInamObj` und `MIvalObj`. Zusätzlich gibt es eine Variable `MItypeObj`, welche den MIME-Type des gewünschten Objektes (z. B. image/gif) spezifiziert. Auch für die Obj-Variablen können in der »Web.conf«-Datei Standardwerte festgelegt werden.

Neben einigen anderen Umgebungsvariablen wird in der »Web.conf«-Datei auch der Name der Illustra-Datenbank festgelegt, in der die AppPages gehalten werden.

3.4.4 Adobe Acrobat

3.4.4.1 Das Dateiformat PDF

Die Basis der Acrobat-Technologie ist das Portable Document Format (PDF). Es handelt sich dabei um ein offenes, plattform-übergreifendes Dateiformat zur Repräsentation von Dokumenten, unabhängig davon, welche Soft- und Hardware und welches Betriebssystem zur Dokumentenerstellung genutzt wurde. Somit wird es möglich, elektronische Dokumente über das Internet und andere Medien zu verteilen, ohne dafür zu unterschiedlichen Dateiformaten greifen zu müssen. Dies bedeutet, daß ein PDF-Dokument, welches auf einem Macintosh-Computer erstellt wurde, auf einem Windows-95-System betrachtet werden kann.

3.4.4.1.1 Graphische Merkmale

PDF arbeitet mit dem Graphikmodell der Programmiersprache PostScript. Von PostScript hat es beispielsweise die Farb- und Auflösungsunabhängigkeit geerbt, was scharfe und farbgetreue Ausdrucke auf nahezu jedem modernen Drucker ermöglicht. Zur Bildschirmdarstellung werden geräteunabhängige Farbmodelle genutzt, die eine farbpräzise Ausgabe auf praktisch jedem Monitor ermöglichen. Die Auflösungsunabhängigkeit führt dazu, daß Dokumente auf dem Bildschirm um bis zu 800% vergrößert werden können, ohne an Darstellungsqualität zu verlieren (z. B. kein Treppeneffekt bei Rundungen). Dies trifft auf Bitmap-Grafiken, die in PDF-Dokumenten enthalten sind, natürlich nicht zu. Sie werden in ihrer ursprünglichen Auflösung gezeigt (gewöhnlicherweise 72-75 dpi), was bei einer starken Vergrößerung zu Qualitätsverlusten der Darstellung führt.

3.4.4.1.2 Hypertextfunktionen

Mittels verschiedener Hypertext-Elemente kann der Benutzer durch ein PDF-Dokument navigieren und wird somit von der linearen Leseart (»von vorn nach hinten«) befreit, die ihm durch Papierdokumente auferlegt wird.

3.4.4.1.2.1 Querverweise oder Verknüpfungen (Links)

Ein Verweis besteht aus einer Quelle (Anker) und einem Ziel. Der Anker wird üblicherweise farbig hervorgehoben. Klickt der Benutzer mit dem Mauszeiger auf diesen Anker, so wird er zu einem vordefinierten Sprungziel geleitet. Mögliche Ziele sind

- Andere Seiten im Dokument
- Andere Dokumente

- WWW-Adressen (URLs)
- Aufruf von externen Anwendungsprogrammen

3.4.4.1.2.2 Lesezeichen oder Bookmarks

Lesezeichen markieren feste Stellen im Dokument durch inhaltliche Beschreibungen und werden am Bildschirm in einem eigenen Rahmen neben dem Dokument angezeigt. Klickt der Benutzer auf ein Lesezeichen, so springt er an die zugehörige Stelle im Dokument. Zusätzlich können die Lesezeichen hierarchisch angeordnet werden und sind somit für die Erstellung von Inhaltsverzeichnissen prädestiniert.

3.4.4.1.2.3 Thumbnails

Thumbnails sind daumennagelgroße Darstellungen der einzelnen Seiten eines Dokuments. Sie werden, wie Lesezeichen, in einem eigenen Rahmen neben dem Dokument dargestellt und erleichtern, gerade bei vielen grafischen Elementen, den Überblick über das Dokument.

3.4.4.1.2.4 Anmerkungen oder Notes

Mittels Anmerkungen kann man kleine Texte in ein Dokument einbauen. Diese werden dann in Form eines kleinen Vierecks auf dem Dokument angezeigt. Klickt man auf dieses Viereck, wird ein Fenster geöffnet, in dem der Anmerkungstext eingeblendet wird.

3.4.4.1.2.5 Article Threads

Article Threads erleichtern das Lesen mehrspaltiger Texte, indem der Benutzer dem Fließtext folgen kann und nicht pausenlos hoch- und herunterscrollen muß.

3.4.4.1.3 Font-Substitution

Fast jeder Computer-Anwender kennt das Problem: Man bekommt von einem Freund ein Text-Dokument (beispielsweise im Word-Format) und möchte sich dieses Dokument an seinem Computer betrachten. Leider hat der Freund eine Schriftart verwendet, die man selbst nicht auf dem Computer installiert hat. Die fehlende Schriftart wird vom System durch einen ganz anderen Font ersetzt, was unangenehme Folgen hat:

- Das Schriftbild stimmt nicht mehr, d. h. eine Serifenschrift wird möglicherweise durch einen serifenlosen Font ersetzt.
- Die Laufweiten stimmen nicht mehr. Der ganze Text wird buchstäblich auseinandergerissen, Einrükkungen werden falsch angezeigt, kein Tabulator stimmt mehr.

PDF löst dieses Problem durch die sogenannte Multiple-Master-Technik. Beim Erstellen einer PDF-Datei werden verschiedene Parameter der verwendeten Fonts (Laufweite, Vorhandensein von Serifen usw.) ermittelt und gespeichert. Fehlen auf dem Anzeigesystem die entsprechenden Fonts, so werden diese von der Acrobat-Software anhand der gespeicherten Font-Parameter nachgebildet. Diese Nachbildungen entsprechen zwar nicht exakt den Originalfonts; es wird jedoch gewährleistet, daß der Grundcharakter eines Dokumentes erhalten bleibt und dieses gut lesbar ist.

3.4.4.1.4 Datenkompression

Textdateien mit hundert oder mehr Seiten können in proprietären Formaten (Word, FrameMaker usw.) leicht zu einer Größe von einigen Megabytes anwachsen, gerade wenn sie viele Grafiken enthalten.

Durch spezielle Kompressionsalgorithmen (CCITT, LZW, RunLength und JPEG) reduziert sich der Platzbedarf von Dokumenten im Portable Document Format erheblich. Es werden nicht nur Rastergrafiken, sondern auch Vektorgraphiken und Text komprimiert.

3.4.4.1.5 Dateistruktur

Eine PDF-Datei ist nach den in ihr enthaltenen Objekten (Seiten, Grafiken, Hyperlinks usw.) organisiert. Eine Tabelle am Ende der PDF-Datei beschreibt die Position jedes im Dokument Objekts. Wird eine PDF-Datei aufgerufen, so wird statt des kompletten Dokuments zunächst diese Tabelle geladen, was den Vorteil hat, daß so in kürzester Zeit auf eine beliebige Seite zugegriffen werden kann. Diese Technik gewährleistet es, daß die Ladezeit eines Dokumentes nicht von dessen Dateigröße abhängt.

PDF erlaubt inkrementele Änderungen. Dies bedeutet, daß bei Änderungen (z. b. Entfernen oder Hinzufügen einer Seite) die Datei nicht komplett neu geschrieben werden muß. Die Änderungen werden einfach hinten an die Datei angehängt. Ersetzte Seiten werden nicht gelöscht, sondern als ungültig markiert. Somit eignet sich PDF auch für WORM-Datenspeicher.

3.4.4.1.6 Datensicherheit

Außer der Möglichkeit, PDF-Dokumente zu verschlüsseln und mit einem Paßwort zu versehen, um unbefugtem Lesen vorzubeugen, können verschiedene Teilfunktionen gesperrt werden:

- Drucken
- Kopieren von Text und Grafik in die Zwischenablage
- Hinzufügen von Anmerkungen

Diese Funktionssperrung macht es möglich, urheberrechtlich geschützte Dokumente frei zu veröffentlichen, ohne Sorgen haben zu müssen, daß diese Dokumente mißbräuchlich genutzt werden könnten.

3.4.4.2 Erstellung und Bearbeitung von PDF-Dateien

Die Erstellung und nachträgliche Bearbeitung von Dateien im Portable Document Format wird über die Acrobat-Produktlinie der Firma Adobe realisiert. Eine nähere Erläuterung der einzelnen Acrobat-Komponenten wird in [Pörner 1997] vorgenommen.

3.5 Entwurf einer Datenbank für die IPSI-Publikationen

Mit Hilfe von Datenbank-Management-Systemen lassen sich große Informationsmengen sehr effizient verwalten. Dabei stellen DBMS jedoch keine echten Software-Lösungen dar. Vielmehr handelt es sich bei DBMS um Werkzeuge, die dem Anwender/Entwickler große Freiräume bei der Gestaltung der Datenverwaltungsumgebung lassen. Um Informationen in einem DBMS verwalten zu können ist es daher notwendig, vorher eine Datenbankstruktur zu konstruieren. Dieser Konstruktionsvorgang wird im Fachjargon konzeptioneller Datenbankentwurf genannt.

Vorbemerkung

Das GMD-IPSI ist schon seit einiger Zeit mit einem breiten WWW-Informationsangebot im Internet vertreten. Die Verwaltung der HTML-Dokumente erfolgte bislang auf herkömmlichem Wege über das Dateisystem. Es stellte sich mittlerweile jedoch heraus, daß diese Art der Dokumentenvorhaltung, gerade bei einem so breiten Dokumentenangebot wie dem des IPSI, sehr ineffektiv ist. Neben der Tatsa-

che, daß alle Querverweise in den HTML-Dokumenten manuell verwaltet werden müssen, besteht die Gefahr, daß die Systemverwalter schnell den Überblick darüber verlieren können, welche Dokumente des Bestandes relevant bzw. irrelevant sind.

Aus diesen Überlegungen heraus wurde beschlossen, im Forschungsbereich CUI (Cognitive User Interfaces) eine datenbankgestützte Verwaltung der HTML-Dokumente mit dem ORDBMS »Illustra Server« zu testen.

Die für diesen Zweck erstellte Datenbank kann andererseits (nach entsprechender Erweiterung des Datenmodells) als Grundlage für das zu konzipierende Publikations-Management-System verwendet werden. Der folgend beschriebene Datenbankentwurf bezieht sich somit auf die Modellierung einer Datenbank für beide Zwecke, Web-Präsentation und Publikations-Management. Im Implementierungsteil dieser Ausarbeitung wird nur noch der Teil der Datenbank angesprochen, der für das Publikations-Management-System von Nutzen ist. Der folgende Datenbankentwurf bezieht aber auch die Teile der Datenbank ein, die für die Web-Präsentation genutzt werden, um dem Leser einen Gesamtüberblick über die komplette Datenbank zu ermöglichen und die Beschreibung des Entwurfs einfacher zu gestalten.

3.5.1 Definition des relevanten Realitätsausschnittes

Nach [Knorz 1997] liegt der erste Schritt eines konzeptionellen Datenbankentwurfes darin, zu klären, welche Daten in der Datenbank benötigt werden und in welchen strukturellen Zusammenhängen diese Daten stehen. Es handelt sich hier um einen Ausschnitt aus der realen Welt; man könnte auch sagen, daß es sich um die Definition einer »Miniwelt« handelt.

Im Folgenden wird die Miniwelt des IPSI definiert:

- **Das IPSI ist in mehrere Arbeitsbereiche unterteilt.**
  Jeder Arbeitsbereich wird durch folgende Merkmale (Attribute) beschrieben: *Bereichsname* und *Bereichskürzel*
- **Die Arbeitsbereiche sind ihrerseits wiederum in Abteilungen unterteilt.**
  Die Abteilungen besitzen folgende Attribute: *Abteilungsname* und *Abteilungskürzel*
- **Innerhalb des Institutes gibt es Forschungsaktivitäten** (Arbeitsgruppen und Projekte). Mit einer Aktivität können eine oder mehrere Abteilungen beschäftigt sein. Eine Abteilung kann mehrere Aktivitäten unterhalten. Attribute der Aktivitäten: *Aktivitätsname*, *Aktivitätskürzel*, *Aktivitätstyp* (Projekt/Arbeitsgruppe)
- **Das Institut beschäftigt Mitarbeiter** (Wissenschaftliche Mitarbeiter, Verwaltungsmitarbeiter, Praktikanten, Diplomanden, Doktoranden, studentische Hilfskräfte). Außer den Verwaltungsmitarbeitern nehmen alle Mitarbeiter direkt an den Forschungsaktivitäten des Institutes teil.
  Attribute der Mitarbeiter: *Anrede*, *Titel*, *Vorname*, *Nachname*, *Position* (Wiss. Mitarbeiter, Praktikant, Diplomand usw.)
- **Im Institut werden Publikationen verfaßt.** Die Publikationen sind Ergebnisse einzelner oder mehrerer Aktivitäten. Zu einer Aktivität können mehrere Publikationen verfaßt werden. Eine Publikation hat einen oder mehrere Autoren. Ein Autor kann mehrere Publikationen verfassen. Ein Autor kann sowohl ein Mitarbeiter als auch eine externe Person (mit den Attributen *Anrede*, *Titel*, *Vorname*, *Nachname*) sein, wobei an einer Publikation mindestens ein Mitarbeiter beteiligt sein muß.
  Attribute der Publikationen: *Titel*, *Erscheinungsjahr*, *Typ* (Artikel/Monographie usw.)

3.5.2 Datenmodellierung der IPSI-Miniwelt

Der nächste Schritt des konzeptionellen Datenbankentwurfes besteht nach [Knorz 1997] in der Umsetzung der Miniwelt in ein Datenbankschema.

3.5.2.1 Entity-Relationship-Modell

Das ursprünglich von Chen [Chen 1976] entwickelte Entity-Relationship-Modell (ERM) stellt ein leistungsstarkes Werkzeug zum Entwurf eines Datenbankschemas dar. Bei der Entwicklung des ERM wurde

von der Tatsache ausgegangen, daß es für den Menschen natürlich ist, seine Welt in Form von Objekten (Entitäten, Entities) wahrzunehmen. Die Aufgabe des ERM besteht nun darin, auf einer rein formalen, konzeptionellen Ebene zu visualisieren, welche Entitäten in einem Realitätsausschnitt vorhanden sind und welche Beziehungen (Relationships) diese Entitäten untereinander haben. Ein ERM besteht aus drei Basiselementen :

3.5.2.1.1 Objektklassen

Objekte der realen Welt werden in einem ERM zu Objektklassen generalisiert. Die Objektklassen (Entities) werden in Form von Rechtecken dargestellt. Die Rechtecke werden einerseits mit dem Objektnamen und andererseits mit dem sie identifizierenden Merkmal (Schlüsselattribut) beschriftet. Steht bei einer Objektklasse kein natürliches Schlüsselattribut zur Verfügung, so wird ein künstliches Schlüsselattribut definiert. (Beispielsweise gibt es bei der Objektklasse Mitarbeiter kein natürliches Schlüsselattribut. Sowohl Vorname als auch Nachname zweier verschiedener Mitarbeiter können identisch sein. Es wird somit das künstliche Schlüsselattribut Personalnummer geschaffen.)

Die Objekte der IPSI-Miniwelt werden zu folgenden Objektklassen (mit ihren jeweiligen Schlüssel-Attributen) generalisiert:

- **Arbeitsbereich (Bereichsnummer)**
  Der Schlüssel »Bereichsnummer« wurde deshalb eingeführt, da im IPSI derzeit an einer Änderung der Institutsstruktur gearbeitet wird, die unter anderem auch eine Umbenennung der Arbeitsbereiche nach sich ziehen kann. Der Schlüssel *Bereichsnummer* ist unabhängig vom *Bereichsname*. Daher kann der *Bereichsname* nachträglich schadlos geändert werden.
- **Abteilung (Abteilungsnummer)**
  Auch hier wurde wegen der bevorstehenden Umstrukturierung ein künstlicher Schlüssel gewählt.
- **Aktivität (Aktivitätsnummer)**
  Im Laufe der Zeit können sich Aktivitätsnamen ändern. Es scheint daher auch hier sinnvoll, einen künstlichen Schlüssel einzuführen.
- **Publikation (Publikationsnummer)**
  Da der Titel einer Publikation i. d. R. aus mehreren Wörtern besteht und der Titel zweier Publikationen gleich sein kann, eignet er sich wenig als Schlüsselattribut. Deshalb auch hier die Einführung eines künstlichen Schlüssels.
- **Mitarbeiter (Personalnummer)**
  (siehe obiges Beispiel)
- **Externe Person (Personalnummer)**
  Durch die Bedingung, daß an Publikationen auch Personen außerhalb des Institutes beteiligt sein können, ergibt sich die Notwendigkeit, für diese Personen eine eigene Objektklasse einzuführen

3.5.2.1.2 Beziehungen

Die zwischen den Objektklassen bestehenden Beziehungen werden ebenfalls im ERM visualisiert. Dabei muß zwischen zwei verschiedenen Beziehungstypen unterschieden werden:

- **Einwertige Beziehungen**
  Wenn durch eine Beziehung zwischen den zwei Objektklassen O1 und O2, einer Instanz der Objekt-

---

Durch eine Entity werden alle diejenigen Merkmale (Attribute) unter einem Oberbegriff zusammengefaßt, die ein spezielles Objekt der realen Welt formal beschreiben.

Die für das IPSI-ERM verwendete Notation wird während der ERM-Definition erläutert.

klasse O1 *genau eine Instanz* der Objektklasse O2 (*oder umgekehrt*) zugeordnet werden kann, hat man es mit einer einwertigen Beziehung zu tun.
Bsp.: Eine Fachbereich kann zwar mehrere Vorlesungen anbieten, eine Vorlesung kann jedoch nur von jeweils einem Fachbereich angeboten werden.
Einwertige Beziehungen werden durch eine schlichte Verbindungslinie dargestellt.

- **Mehrwertige Beziehungen**
  Wenn durch eine Beziehung zwischen den zwei Objektklassen O1 und O2, einer Instanz der Objektklasse O1 *mehrere Instanzen* der Objektklasse O2 (*und umgekehrt*) zugeordnet werden können, so handelt es sich hierbei um eine mehrwertige Beziehung.
  Bsp.: Ein Student kann mehrere Vorlesungen belegen und eine Vorlesung kann von mehreren Studenten belegt werden.
  Mehrwertige Beziehungen werden durch eine Raute dargestellt.

Jede der beiden Beziehungstypen erfährt eine spezielle Behandlungsweise. Dies soll jedoch erst im Abschnitt 3.5.3.2 näher erläutert werden.

3.5.2.1.2.1 Spezielle Beziehungstypen

Normalerweise werden durch eine Beziehung lediglich zwei Objektklassen miteinander verbunden. Es kann jedoch notwendig sein, Beziehungen zwischen drei oder mehr Objektklassen zu schaffen. Umgekehrt ist es gut möglich, eine Objektklasse mit sich selbst in Beziehung zu bringen.

3.5.2.1.3 Kardinalitätsangaben

Mittels Kardinalitätsangaben werden Beziehungen dahingehend spezifiziert, welche Anzahl von Instanzen den durch sie in Verbindung gebrachten Objektklassen gegenseitig zugeordnet werden können.

- **n**
  Einer Instanz der Objektklasse O1 können **genau n** Instanzen der Objektklasse O2 zugeordnet werden.
  Bsp.: Ein Mitarbeiter hat nur ein Büro (n=1). In einem Büro können immer nur zwei Mitarbeiter sitzen (n=2)

- **[min,max]**
  Einer Instanz der Objektklasse O1 können **minimal n** und **maximal m** Instanzen der Objektklasse O2 zugeordnet werden.
  Bsp.: Das Seminar XY wird nur dann abgehalten, wenn mindestens fünf Studenten teilnehmen; es ist jedoch nicht möglich, mehr als 20 Studenten teilnehmen zu lassen [5,20]

- **+**
  Einer Instanz der Objektklasse O1 können **eine oder beliebig mehr** Instanzen der Objektklasse O2 zugeordnet werden. Das + kann mit einer [min,max]-Angabe von [1,(] gleichgesetzt werden.
  Bsp.: Ein Unternehmen muß mindestens einen Gesellschafter haben, kann aber auch beliebig viele Gesellschafter vorweisen.

- *****
  Einer Instanz der Objektklasse O1 können **0, eine oder beliebig mehr** Instanzen der Objektklasse O2 zugeordnet werden. Das * kann mit einer [min,max]-Angabe von [0,(] gleichgesetzt werden.
  Bsp: Ein Ehepaar kann keine Kinder, ein Kind, oder beliebig viele Kinder haben.

Wendet man die obigen Beziehungsregeln auf die IPSI-Miniwelt an, so ergeben sich folgende Beziehungen (mit ihren jeweiligen Kardinalitäten):

- **Arbeitsbereich - Abteilung:**
  Ein Arbeitsbereich kann eine oder mehrere Abteilungen besitzen (+). Eine Abteilung kann jedoch nur einem Arbeitsbereich angehören (1).
  Einwertige Beziehung.

- **Aktivität - Abteilung:**
  An einer Aktivität muß sich mindestens eine Abteilung beteiligen, denkbar ist jedoch auch eine Beteiligung mehrerer Abteilungen (+). Eine Abteilung hat mindestens eine Aktivität, kann sich jedoch auch mit mehreren Aktivitäten befassen (+).
  Mehrwertige Beziehung.

- **Aktivität - Mitarbeiter:**
  Eine Aktivität wird von einem oder mehreren Mitarbeitern durchgeführt (+). Ein Mitarbeiter kann sich mit keiner (Verwaltungsmitarbeiter), einer oder mehreren Aktivitäten beschäftigen (*).
  Mehrwertige Beziehung.

- **Publikation - Aktivität**
  Eine Publikation ist das Ergebnis keiner (z. B. bei einer Monographie eines Mitarbeiters), einer oder mehrerer Aktivitäten (*). Zu einer Aktivität können keine, eine oder mehrere Publikationen verfaßt werden (*).
  Mehrwertige Beziehung.

- **Publikation - Mitarbeiter - externe Person**
  Eine Publikation wird von einem oder mehreren Mitarbeitern (+) mit der Beteiligung keiner, einer oder mehrerer externer Personen (*) verfaßt. Ein Mitarbeiter kann keine, eine oder mehrere Publikationen verfassen (*).
  Mehrwertige Beziehung.

Nachdem die Objektklassen, Beziehungen und Kardinalitäten festgelegt sind, kann das ERM gezeichnet werden. Es stellt sich folgendermaßen dar:

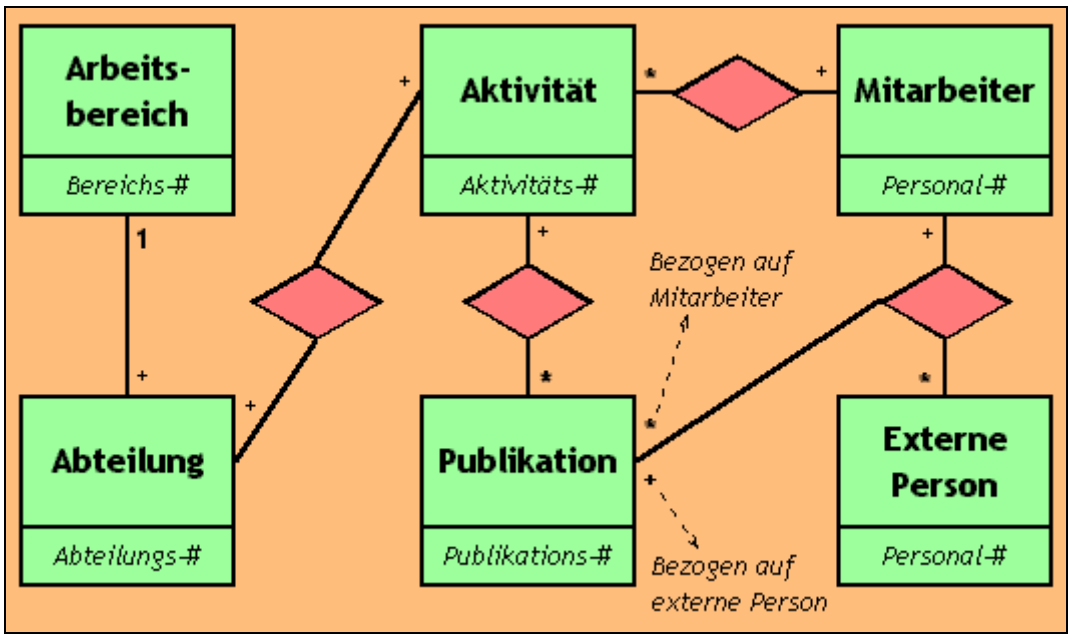

Abbildung 4: Erstes ERM der IPSI-Miniwelt

3.5.2.2 Erweiterung des ERM

Grundsätzlich gesehen ist das obige ERM der IPSI-Miniwelt komplett. Bei näherer Betrachtung fallen jedoch einige Umstände auf, welche die Umsetzung des ERM in einen relationalen Datenbankentwurf komplizieren würden bzw. bei der späteren Verwaltung der Daten zu ernsthaften Schwierigkeiten führen könnten.

3.5.2.2.1 Problemfall 1: Objektklassen »Mitarbeiter« und »externe Person«

Nehmen wir an, eine externe Person X (also der Mitautor einer Publikation) würde zu einem späteren Zeitpunkt zu einem Mitarbeiter. In diesem Fall würde ein neuer »Mitarbeiter«-Datensatz X angelegt. Würde jedoch vergessen werden, den neuen Mitarbeiter als externe Person X (die er ja nun nicht mehr ist) aus der »externe Person«-Tabelle zu löschen, dann wäre ein und dieselbe Person zweimal in der Datenbank gespeichert, was, außer der Tatsache, daß dies eine Verschwendung von Speicherplatz darstellt, früher oder später zu Problemen führen könnte. Wenn nämlich eine Abfrage gestartet würde, die sich auf X als Autor der Publikation Y bezieht, dann würde als Ergebnis der Datensatz X gefunden, obwohl dieser jetzt gar nicht mehr relevant wäre. Die Datenbank wäre somit inkonsistent.

3.5.2.2.2 Problemfall 2: Objektklasse »Publikationen«

Bei der obigen Darstellung handelt es sich um eine vereinfachte Version der IPSI-Miniwelt, die nicht der Tatsache Rechnung trägt, daß die unterschiedlichen Publikationsarten eigene Objektklassen darstellen, da jede Publikationsart ihre eigenen, für sich speziellen Attribute besitzt. Betrachten wir uns dies im Einzelnen:

- **GMD-Studie**, **GMD-Bericht** und **Arbeitspapiere der GMD** können zu einem Publikationstyp zusammengefaßt werden. Diese Objektklasse wird »**GMD-Publikationen**« genannt.
  Sie besitzt folgende Attribute: *Titel*, *Erscheinungsjahr*, *Abstract*, *Volltext*, *Typ* (GMD-Studie, GMD-Bericht, Arbeitspapier), *lfd. Nummer*, *Verlag mit Standortangabe* (GMD-Studien und Arbeitspapiere werden durch die GMD verlegt, die GMD-Berichte hingegen von dem Verlag Oldenbourg).

- **Proceedings**, die nicht als GMD-Studie herausgegeben werden, und **Bücher** können zur Objektklasse »**Monographien**« zusammengefaßt werden.
  Sie besitzen folgende Attribute: *Titel*, *Erscheinungsjahr*, *Abstract*, *Volltext*, *Typ* (Buch oder Proceeding), *Nummer* (bei mehrbändigen Werken), *Verlag mit Standortangabe*

- **Beiträge in Fachzeitschriften** und **Kapitel in Monographien**, die von IPSI-Mitarbeitern geschrieben werden, können zur Objektklasse »**Artikel**« zusammengefaßt werden.
  Sie werden durch folgende Attribute beschrieben: *Titel*, *Erscheinungsjahr*, *Abstract*, *Volltext*, *Typ* (Zeitschriftenartikel oder Monographiekapitel), *Kapitel* bzw. *Seiten* (die der Artikel im übergeordneten Werk in Anspruch nimmt).
  Für das **übergeordnete Werk** ergeben sich noch zusätzliche Attribute: *Titel*, *Ausgabe, lfd. Nummer*, *Verlag mit Standortangabe*. Theoretisch könnte daraus eine eigene Objektklasse definiert werden. Da diese Attribute jedoch nur dazu dienen, Referenzangaben für eine Darstellung im WWW zu generieren, macht es keinen Sinn, die übergeordneten Publikationen über eine eigene Objektklasse suchbar zu machen. Sie können daher in die »**Artikel**«-Klasse aufgenommen werden. Die zu erwartenden Redundanzen (z. B. wenn zwei in der Datenbank gespeicherte Artikel im selben übergeordneten Werk erscheinen) werden sich mit größter Wahrscheinlichkeit in einem kleinen Rahmen bewegen und sind daher zu verschmerzen.

- **Diplomarbeiten** stellen eine eigene, gleichnamige Objektklasse mit folgenden Attributen dar: *Titel*, *Erscheinungsjahr*, *Abstract*, *Volltext*, *Hochschule* (auf der das Diplom erworben wurde), *Diplombetreuer* (GMD-intern).

Jede Publikations-Objektklasse würde eigene Beziehungen zu Aktivitäten, externen Personen und Mitarbeitern besitzen:

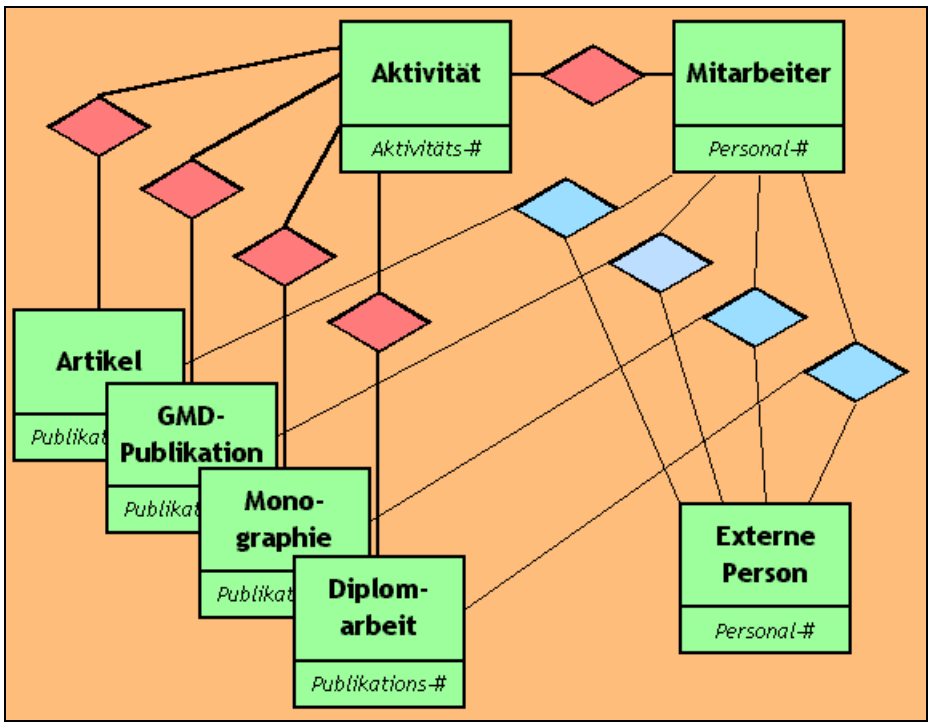

Abbildung 5: Beziehungen der einzenen Publikationsrelationen zu Aktivitäten und Personen

Aus den obigen Beschreibungen wird ersichtlich, daß es sich zwar um unterschiedliche Objektklassen handelt; sie besitzen jedoch einige Gemeinsamkeiten. Jede der Objektklassen hat folgende Merkmale: *Publikationsnummer*, *Titel*, *Erscheinungsjahr*, *Abstract*, *Volltext*. Es erscheint unsinnig, diese Attribute für jede Objektklasse neu zu definieren.

3.5.2.2.3 Abstraktionsbeziehung

Die Lösung für beide o. g. Probleme besteht in der Schaffung von Abstraktionsbeziehungen. Mittels Abstraktionsbeziehungen werden verwandte Objektklassen zusammengefaßt. Dies geschieht (bezogen auf die beiden o. g. Beispiele) folgendermaßen:

- Die verwandten Objektklassen werden weitestgehend generalisiert, so daß die gemeinsamen Attribute die gleichen Namen erhalten. Dies ist bei o. g. Beispielen schon erfolgt.
- Für alle Objektklassen wird ein gemeinsamer, künstlicher Schlüssel definiert. Im Problemfall 1 wäre das beispielsweise eine *Personalnummer*; beim Problemfall 2 würde man eine *Publikationsnummer* wählen.
- Es wird eine zusätzliche Objektklasse geschaffen, die alle gemeinsamen Attribute erhält.
  Beim Problemfall 1 würde man eine Objektklasse »**Personen**« mit den Attributen *Personalnummer*, *Anrede*, *Titel*, *Vorname*, *Nachname* schaffen.
  Beim Problemfall 2 wird eine Objektklasse namens »**Publikationen**« definiert, die folgende Attribute enthält: *Publikationsnummer*, *Titel*, *Erscheinungsjahr*, *Abstract*, *Volltext*.
- Die verwandten Objektklassen werden nun dieser neuen Objektklasse (der Oberklasse) als Unterklassen untergeordnet. Die Unterklassen erben die Attribute der Oberklasse. Somit müssen in jeder Unterklasse nur noch die für sie spezifischen Attribute definiert werden.

Für den Problemfall 2 gibt es nun eine Oberklasse »**Publikationen**« mit den Unterklassen »**GMD-Publikationen**«, »**Monographien**«, »**Artikel**« und »**Diplomarbeit**«.

Für den Problemfall 1 ergibt sich eine noch einfachere Lösung. In der Oberklasse »**Personen**« sind alle für eine externe Person notwendigen Attribute vorhanden. Externe Personen können somit direkt in der

Oberklasse gespeichert werden. Somit muß nur eine Unterklasse »**Mitarbeiter**« mit spezifischen Merkmalen (z. B. Position und Status) geschaffen werden.

Das erweiterte IPSI-ERM zeigt sich jetzt folgendermaßen:

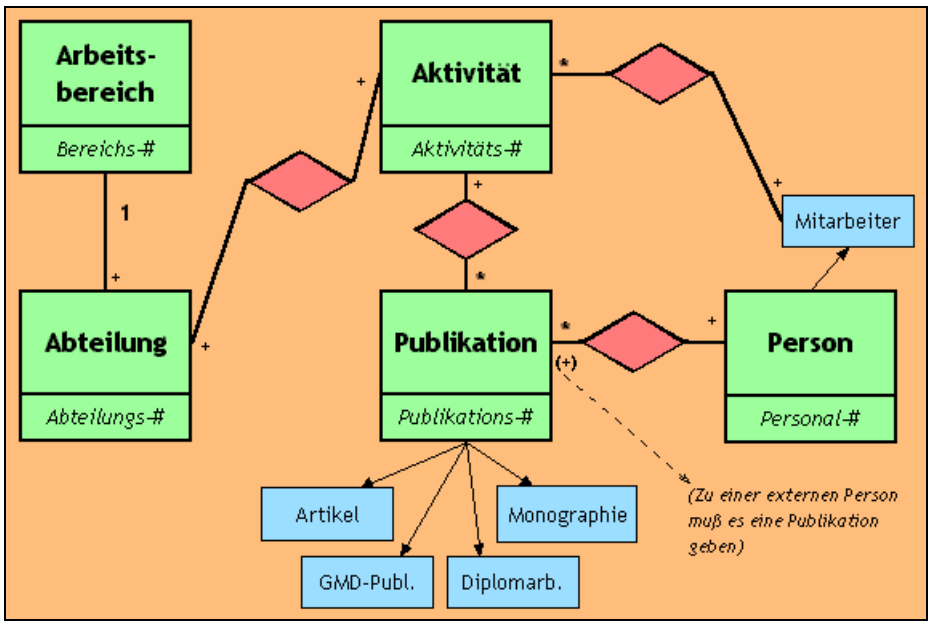

Abbildung 6: Überarbeitetes ERM der IPSI-Miniwelt

3.5.3 Umsetzung in einen relationalen Datenbankentwurf

Als letzter Schritt werden nach [Knorz 1997] die Konstrukte des ERM in Relationen umgesetzt, aus denen später direkt die Datenbanktabellen generiert werden können. Dabei bilden die Attribute die Spalten der Tabelle. Die Tabellenzeilen nehmen die Instanzen der jeweiligen Objektklasse auf. In der Fachsprache werden die Tabellenzeilen *Tupel* genannt.

3.5.3.1 Umsetzung der Objektklassen

Begonnen wird mit den Objektklassen, die direkt in Relationen umgesetzt werden können:

Jede Relation erhält einen Namen (üblicherweise der Name der Objektklasse), geschrieben in Großbuchstaben. Die Attributbezeichnungen werden in Standardschrift, die Schlüsselattribute in Fettschrift und unterstrichen geschrieben. Zusätzlich wird definiert, von welchem Datentyp die einzelnen Attribute sind:

Anmerkung:
Die nun gezeigten Relationen besitzen mehr Attribute als im vorangegangenen Text definiert wurden. Diese zusätzlichen Attribute dienen der besseren Präsentation im WWW. Auf sie wurde im obigen Text verzichtet, da sie die Erklärungen umständlicher gestaltet hätten. Es handelt sich jedoch nicht um Schlüsselattribute, so daß die nachträgliche Ergänzung schadlos vorgenommen werden kann.

ARBE	T	BERE	C
Bereichsnummer		small_integer	
Bereichskürzel		text	
Bereichsname		text	
Bereichsbeschreibung		large_text	
Bereichslogo		large_object	

```
ABTE     L
      bteilungsnummer             small_integer
      bteilungskürzel             text
      bteilungsname               text
      bteilungsbeschreibung       large_text
      bteilungslogo               large_object

A     T                T     T
      kti      it      tsnummer             small_integer
      kti      it      tskürzel             text
      kti      it      tsname               text
      kti      it      tsbeschreibung       large_text
      kti      it      tslogo               large_object

PER      O
      ersonalnummer        small_integer
      nre      e                text
      itel                 text
      orname               text
      achname              text
      tatus                text
      ele      onnummer         text
      ax_      ummer            text
e      ail_          resse          text
                 _      ome      age          text

            TARBE       TER
                                   ir             aus                                             über
      osition      text
      tatus        text                ehemalig             aktuell      o      er      zukün      tig
      ortrait           large_object

P      BL              AT      O
      ublikationsnummer    small_integer
      itel                 text
      ntertitel            text
      rscheinungjahr       small_integer
                                     text
      bstract              large_text
      olltext              large_text

                  _P       BL              AT       O
                                   ir             aus                   B
l            _       ummer          integer
      erlag                text
      erlagsort            text

      O      O      RAP             E
                                   ir             aus                   B
l                               ummer          integer
      erlag                text
      erlagsort            text

ART            EL
                                   ir             aus                   B
      a      itel               text
      eiten                text
      ber_      utoren          text
      ber_      itel            text                     ber                  übergeor      nete
      ber_      ntertitel       text
      ber_                                text                     erio      ikum             onogra
      usgabe               text                  anuar                                      ebruar
l                               ummer          text
      erlag                text
      erlagsort            text
```

```
            PLO       ARBE       T
                                      ir              aus                    B
      ochschule             text
Betreuer                integer                    e       erenz       au
```

3.5.3.2 Umsetzung der Beziehungen

Als nächstes werden die Beziehungen zwischen den Entitäten umgesetzt. Hierbei muß zwischen den einwertigen und mehrwertigen Beziehungen unterschieden werden:

3.5.3.2.1 Einwertige Beziehungen

Kann bei einer einwertigen Beziehung jeder der beiden Relationen nur jeweils eine Instanz der anderen Relation zugeordnet werden (1:1), so werden keine Veränderungen an den Relationen vorgenommen.

Können jedoch einer der beiden Relationen mehrere Instanzen der anderen Relation zugeordnet werden (1:+, 1:* oder 1:n [wobei n>1]), so wird der Relation, von welcher mehrere Instanzen zugeordnet werden können, das Schlüsselattribut der anderen Relation als Fremdschlüssel hinzugefügt. Zum besseren Verständnis folgt die Umsetzung der einzigen einfachen Beziehung innerhalb des IPSI-ERM:

Der Relation ARBEITSBEREICH können eine oder mehrere Instanzen der Relation ABTEILUNG zugeordnet werden. Der Relation ABTEILUNG wird das Schlüsselattribut Bereichsnummer aus der Relation ARBEITSBEREICH als Fremdschlüssel hinzugefügt. Die Relation sieht nach der Veränderung folgendermaßen aus:

```
ABTE      L
      bteilungsnummer             small_integer
      bteilungskürzel             text
      bteilungsname               text
      bteilungsbeschreibung       large_text
      bteilungslogo               large_object
Bereichsnummer              small_integer             e       erenz       au                    B
```

3.5.3.2.2 Mehrwertige Beziehungen

Aus mehrwertigen Beziehungen werden neue Relationen, sog. Verbindungsrelationen, generiert. Diese enthalten grundsätzlich die Schlüsselattribute der Relationen, die über sie verbunden werden; diese stellen dann für die Verbindungsrelation den zusammengesetzten Schlüssel dar. Je nach Bedarf können dann noch eigene Attribute hinzugefügt werden.

- **Beziehung zwischen PUBLIKATION und PERSON:**

```
      R       EBER      C       AFT
      ublikationsnummer         small_integer             e       erenz       au                    B
      ersonalnummer             small_integer             e       erenz       au
```

- **Beziehung zwischen PUBLIKATION und AKTIVITÄT:**

```
ER        EB
      ublikationsnummer         small_integer             e       erenz       au                    B
      kti       it      tsnummer          small_integer             e       erenz       au
```

- **Beziehung zwischen AKTIVITÄT und MITARBEITER:**

```
BE        C             FT
      kti       it      tsnummer          small_integer                   e       erenz       au
      ersonalnummer             small_integer             e       erenz       au
```

- **Beziehung zwischen AKTIVITÄT und ABTEILUNG:**

```
A       F       ABE
      kti       it      tsnummer          small_integer             e       erenz       au
      bteilungsnummer           small_integer             e       erenz       au                    B
```

3.5.3.3 Überprüfung der Normalformeneinhaltung

Durch Fehler, die bei der Relationendefinitionen gemacht wurden, können beim späteren Datenbankbetrieb Inkonsistenzen auftauchen. Um solche Inkonsistenzen zu vermeiden, wurde von Codd die sog. Normalformenlehre eingeführt [Knorz 1997], mittels derer es möglich ist, Datenbankentwürfe auf ihre

Richtigkeit hin zu überprüfen. Im Allgemeinen kennt man drei Normalformen, die im Folgenden näher beschrieben werden.

- **Erste Normalform:**
  Eine Relation ist dann in der ersten Normalform, wenn alle Attributwerte atomaren Charakter haben.

Dies bedeutet, daß ein Attribut innerhalb eines Tupels lediglich einen Wert beherbergen darf. Würde die Relation PERSON im IPSI-Entwurf ein Attribut Name aufweisen, welches den Vor- und Nachnamen einer Person enthält, so würde sie gegen die erste Normalform verstoßen. Das Attribut Name müßte in zwei Attribute Vorname und Nachname aufgeteilt werden.

- **Zweite Normalform:**
  Eine Relation ist dann in der zweiten Normalform, wenn sie die erste Normalform erfüllt und wenn alle Nicht-Schlüsselattribute voll funktional von dem/den Schlüsselattribut/en abhängig sind.

Bsp.: Nehmen wir an, die Arbeitsbereiche und Abteilungen würden (irrtümlicherweise) innerhalb einer Relation verwaltet. Schlüsselattribute sind Bereichsnummer und Abteilungsnummer :

Bereichsnummer	Bereichsname	Abteilungsnummer	Abteilungsname
1	CUI	1	VISIT
1	CUI	2	MIND
2	DIMSYS	3	MIPP

Tabelle 5: Relation, die gegen die zweite Normalform verstößt

Diese Relation wäre nicht in der zweiten Normalform. Würde man die Abteilung MIPP löschen (in diesem Fall die einzige Abteilung des Arbeitsbereiches DIMSYS) würde man unbeabsichtigt auch den Arbeitsbereich DIMSYS löschen (Löschanomalie). Die Relation muß somit in zwei Tabellen (ARBEITSBEREICH und ABTEILUNG) aufgespaltet werden. ABTEILUNG erhält als Schlüssel das Attribut *Abteilungsnummer*, ARBEITSBEREICH die *Bereichsnummer*. Zusätzlich wird, um die Beziehung zum Arbeitsbereich zu erhalten, der Schlüssel *Bereichsnummer* aus der Relation ARBEITSBEREICH als Fremdschlüssel in die Relation ABTEILUNG aufgenommen.

Eine Relation, die der ersten Normalform entspricht und nur ein Schlüsselattribut besitzt, ist automatisch auch in der zweiten Normalform.

- **Dritte Normalform:**
  Eine Relation ist dann in der dritten Normalform, wenn sie die zweite Normalform erfüllt und kein Nicht-Schlüsselattribut transitiv vom Schlüssel abhängig ist.

Als transitiv wird eine indirekte Abhängigkeit bezeichnet. Hierzu ein Beispiel:

Nehmen wir an, die IPSI-Relation ABTEILUNG würde neben *Bereichsnummer* noch das Attribut *Bereichsname* enthalten, so wäre eine transitive Abhängigkeit gegeben. Der *Bereichsname* ist zwar von der *Bereichsnummer* abhängig, jedoch nicht von der *Abteilungsnummer*. Die Abhängigkeit *Bereichsname* zu *Abteilungsnummer* besteht also nur indirekt (transitiv) über *Bereichsnummer*. Wenn der *Bereichsname* in der Relation ARBEITSBEREICH geändert würde, so würde sich diese Änderung nicht auf den *Bereichsname* in der Relation ABTEILUNG auswirken (Änderungsanomalie). Somit müßte das Attribut *Bereichsname* aus der Relation ABTEILUNG gelöscht werden, damit sie der dritten Normalform genügt.

Der obige IPSI-Entwurf verstößt gegen keine der drei Normalformen und kann somit als abgeschlossen angesehen werden.

4 Implementierung

Vorbemerkung:
Für die Arbeit mit Illustra Server hat im Rahmen dieser Diplomarbeit keine grafische Benutzeroberfläche zur Verfügung gestanden. Die einzig verfügbare Schnittstelle zum Illustra-System war das Befehlszeilenprogramm msql. Über die von msql zur Verfügung gestellte Befehlszeile können SQL-Anweisungen an das Illustra-System abgesetzt werden. Alle nachfolgend dargestellten Arbeiten (Erstellen der Relationen und Indizes und das Einfügen der Datensätze in die Datenbank) wurden komplett über die msql-Befehlszeile durchgeführt. Zur Arbeitserleichterung wurden alle SQL-Anweisungen mittels eines Text-Editors in mehreren Textdateien vorformuliert. Diese Textdateien konnten von der msql-Befehlszeile aus, mit dem Kommando \i <dateiname> ins DBMS eingebracht werden.

4.1 Erzeugung der Dokumentendatenbank

Die in Abschnitt 3.5 konzipierte Datenbank soll nicht nur die Publikationen des IPSI aufnehmen, sondern (wie schon erwähnt) auch dazu dienen, eine datenbankgestützte WWW-Präsentation des IPSI zu ermöglichen. Das Publikations-Management-System nutzt somit eine schon vorhandene Datenbank und deren Relationen. Es handelt sich um folgende Relationen:

- Arbeitsbereich (`divisionTable`)
- Abteilung (`departmentTable`)
- Aktivität (`activityTable`)
- Person (`personnelTable`)
- Mitarbeiter (`personnelGmdTable`)

Diese Relationen wurden im Konzeptionsteil dieser Ausarbeitung nur deswegen mit einbezogen, um dem Leser einen Gesamtüberblick über die ganze Datenbank zu verschaffen. Da die Erstellung der o. g. Relationen einerseits nicht im Rahmen dieser Ausarbeitung erfolgte und andererseits diese nur von zweitrangiger Bedeutung für das Publikations-Management-System sind, soll auf deren Erstellungsprozeß nicht näher eingegangen werden. Der interessierte Leser möge sich in den Anhang begeben, wo er die einzelnen Relationsdefinitionen ausführlich beschrieben findet.

Im folgenden Abschnitt findet also keine Neudefinierung einer Datenbank statt. Es handelt sich vielmehr um die Erweiterung einer schon bestehenden Datenbank.

4.1.1 Erzeugung der Relationen

Die in Abschnitt 3.5 modellierten Relationen werden nun in SQL-Definitionen umgesetzt. Dabei ist zu beachten, daß zur Definition von Relationen innerhalb einer objektrelationalen Datenbank zu jeder Relation (Table) ein »Composite Type« definiert werden muß. Praktischerweise kann der Composite Type mit zugehöriger Relation in einer SQL-Anweisung definiert werden.

4.1.1.1 Umsetzung der aus den Objektklassen resultierenden Relationen

```
P      BL            AT      O
       ublikationsnummer    small_integer
       itel                 text
       ntertitel            text
       rscheinungjahr       small_integer
                                      text
       bstract              large_text
       olltext              large_text
```

wird zu:

```
CREATE TABLE ne      PublicationTable OF       E       TYPE       e      PublicationT      pe
(
publication      d               smallint not null primar            e      ,
publicationYear             smallint,
publicationTitle            text,
publicationAbstract         large_text,
publicationFullText         large_text,
publicationP      F             large_ob      ect,
publicationT      pe            text
);
```

Anmerkung zur obigen Definition:
Das Attribut `publicationPDF` *wurde deswegen eingeführt, weil das PLS Text DataBlade derzeit nur Textdokumente im ASCII-Format indexieren kann. Es soll jedoch auch eine PDF-Version des jeweiligen Dokumentes vorgehalten werden, die vom Anwender heruntergeladen werden kann. Somit muß sowohl eine PDF- als auch eine ASCII-Version des Dokumentes abgespeichert werden.*

Die nächsten vier Relationen (die einzelnen Publikationsarten) werden als Unterklassen der Relation `publicationTable` definiert. Sie übernehmen die Attribute dieser Relation. Daher müssen nur die jeweils speziellen Attribute jeder Unterklasse definiert werden:

```
                  _PER      O                   A
                                    ir            aus                   B
l            _      ummer             text
      erlag               text
      erlagsort           text
```

wird zu:

```
CREATE TABLE gmdReportTable OF       E       TYPE       mdReportT      pe
(
gmdReport      olume             text,
gmdReportPublisher          text,
gmdReportPubLocation        text
)
                  ER ne      PublicationTable;

      O      O      RAP            E
                       ir            aus                   B
l                          ummer          text
      erlag               text
      erlagsort           text
```

wird zu:

```
CREATE TABLE monograph      Table OF       E       TYPE       onograph      T      pe
(
monograph            olume            text,
monograph      Publisher          text,
monograph      PubLocation        text
)
                  ER ne      PublicationTable;

ART            EL
                                  ir            aus                   B
      a      itel             text
      eiten               text
      ber_      utoren         text
      ber_      itel           text                        ber                 übergeor      nete
      ber_      ntertitel      text
      ber_                                text                        erio      ikum            onogra
      usgabe              text                        anuar                                       ebruar
l                          ummer          text
      erlag               text
      erlagsort           text
```

wird zu:

```
CREATE TABLE articleTable OF       E       TYPE ArticleT      pe
(
articlePAuthors              text,
articlePTitle                text,
articleP      olume               text,
articleP      ssue                text,
articlePChapter              text,
articlePPages                text,
articlePPublisher            text,
articlePPubLocation          text
)
                  ER ne      PublicationTable;


            PLO      ARBE      T
                                    ir             aus                   B
      ochschule            text
Betreuer              integer                    e      erenz      au
```

wird zu:

```
CREATE TABLE diplomaTable OF       E       TYPE       iplomaT      pe
(
diploma      niversit                 text,
personnel      d                  int re      erences personnel      mdTable
)
under ne      PublicationTable;
```

Anmerkung zur Relation diplomaTable:
Für die Speicherung des Diplombetreuers kann man sich der (außerhalb der Diplomarbeit definierten) Relation personnelGmdTable bedienen. In ihr sind alle Mitarbeiter des IPSI gespeichert. Es wird über die Personalnummer (personnelId) eine Referenz zu dieser Relation gezogen.

4.1.1.2 Umsetzung der Verknüpfungsrelationen

```
      R      EBER      C      AFT
      ublikationsnummer    small_integer              e      erenz      au                          B
      ersonalnummer        small_integer              e      erenz      au
```

wird zu:

```
CREATE TABLE ne      PublicationPersonnelTable OF       E       TYPE    onnelT     PublicationPers
(
publication      d         smallint REFERE      CE       ne      PublicationTable,
personnel      d           int REFERE      CE       personnelTable,
);

ER      EB
      ublikationsnummer    small_integer              e      erenz      au                          B
      kti       it      tsnummer      small_integer              e      erenz      au
```

wird zu:

```
CREATE TABLE ne      PublicationActivit      Table OF       E       TYPE it T    e pe   PublicationActi
(
publication      d         smallint REFERE      CE       ne      PublicationTable,
activit            d            int REFERE      CE       activit      Table
);
```

Anmerkung zu den Beziehungsrelationen:
Da die einzelnen Publikationsarten Unterklassen der Relation newPublicationTable sind, kann die jeweilige publicationId auch über die Relation newPublicationTable abgefragt werden. Somit ist es nicht notwendig, die Verknüpfungsrelationen für jede einzelne Unterklasse zu definieren. Die Definition über die Relation newPublicationTable genügt.

4.1.2 Erzeugung der Textindizes

Damit das PLS Text DataBlade Textdokumente durchsuchen kann, die innerhalb von Datenbankrelationen gespeichert sind, ist es notwendig, spezielle Textindizes anzulegen. Die nach dem Invertierungsprozeß (siehe Abschnitt 3.4.2) übriggebliebenen Wörter werden in diesen Indizes abgespeichert.

Für jede in Frage kommende Relation muß ein Index erstellt werden. Dabei bietet PLS die Möglichkeit, mehrere Attribute in einem Index zu halten. Diese Attribute können beim Suchprozeß zusammen oder getrennt (ähnlich einer Feldsuche in einem Information-Retrieval-System) abgesucht werden.

4.1.2.1 Index für Relation gmdReportTable:

```
CREATE                    E       gmdReport       ndex
O       gmdReportTable
                                PL
(
publicationTitle,
publicationAbstract          large_text_ops,
publicationFullText          large_text_ops
);
```

Anmerkung:
Attributwerte des Datentyps text werden von PLS ohne jede weitere Typangabe invertiert. Für Attributwerte des Datentyps large_text muß eine spezielle Typangabe (large_text_ops) gemacht werden.

4.1.2.2 Index für Relation monographyTable:

```
CREATE                    E       monograph             ndex
O       monograph      Table
                                PL
(
publicationTitle,
publicationAbstract          large_text_ops,
publicationFullText          large_text_ops
);
```

4.1.2.3 Index für Relation diplomaTable:

```
CREATE                    E       diploma       ndex
O       diplomaTable
                                PL
(
publicationTitle,
publicationAbstract          large_text_ops,
publicationFullText          large_text_ops
);
```

4.1.2.4 Index für Relation articleTable:

```
CREATE                    E       article       ndex
O       articleTable
                                PL
(
publicationTitle,
publicationAbstract          large_text_ops,
publicationFullText          large_text_ops,
articlePTitle,
articlePAuthors
);
```

Anmerkung zum articleIndex:
Da auch der Titel und die Autoren der übergeordneten Publikation eines Artikels für eine Suche relevante Aspekte sein können, werden diese auch invertiert.

Die Stemming-Funktion, mit der die zu indizierenden Wörter auf ihre Stammform zurückgeführt werden, wurde deaktiviert. Dies erfolgte deswegen, weil der zu verwaltende Dokumentenbestand sowohl aus englisch- als auch deutschsprachigen Publikationen besteht. Eine Unterstützung deutschsprachiger Dokumente wurde von PLS zwar angekündigt, war aber zum Zeitpunkt der Implementierung nicht verfügbar. Wenn die Stemming-Funktion aktiviert worden wäre, hätte es beim Indizieren zu Rückführungsfehlern bei den deutschsprachigen Dokumenten kommen können. Da die Indizierung aus funktionstechnischen Gründen nicht auf die englischsprachigen Dokumente fixiert werden konnte (»entweder alle oder keine«), wurde die Stemming-Funktion deaktiviert.

Ob die Textindizes vor oder nach der Einfügung der Datensätze in die Datenbank erstmals erstellt werden, spielt keine Rolle:

- Werden die Indizes erst nach der Einfügung der Datensätze erstellt, so werden alle vorhandenen Dokumente invertiert. Dieses Leistungsmerkmal von PLS stellt sicher, daß Indizes im Notfall gelöscht und wieder neu aufgebaut werden können.
- Werden die Indizes vor der Einfügung der Datensätze erstellt, wird jedes neu hinzugefügte Dokument einzeln invertiert. Dieses Leistungsmerkmal von PLS stellt sicher, daß jedes in Zukunft neu zur Datenbank hinzugefügte Dokument automatisch invertiert wird. Wird ein Datensatz gelöscht, so werden alle Indexeinträge des betroffenen Dokumentes automatisch aus dem Index gelöscht.

Die Erweiterung der Datenbank ist damit abgeschlossen und die Datensätze können eingegeben werden.

4.2 Vorbereitung der Dokumente

Alle Dokumente, die in elektronischer Form vorhanden waren, lagen im PostScript-Format vor. Da von den Publikationen einerseits eine ASCII-Version (zur Invertierung durch das PLS Text DataBlade) und andererseits eine PDF-Version (als Download-Datei für Anwender) benötigt wird, mußten alle PostScript-Dateien folgendermaßen bearbeitet werden:

- Mit Acrobat Distiller von Adobe werden alle PostScript-Dateien in das Portable-Document-Format konvertiert.
- Mit GSview, einem PostScript-Interpreter für MS Windows, der auch PDF-Dateien anzeigen kann, wird die Textinformation aus jeder PDF-Datei extrahiert und in jeweils eine ASCII-Datei geschrieben.
- Die nun nicht mehr benötigten PostScript-Dateien werden gelöscht.

Zu nahezu allen Dokumenten (außer den Diplomarbeiten und Monographien) konnten im HTML-Dokumentenbestand des CUI-Web-Servers Abstracts gefunden werden. Sie wurden aus einem Web-Browser mittels »Copy-and-Paste« in einen Texteditor kopiert und dort einzeln als ASCII-Dateien abgespeichert.

4.3 Einfügen der Datensätze

Das Einfügen von Datensätzen erfolgt über die SQL-Anweisung INSERT. Nachstehend findet der Leser beispielhafte INSERT-Anweisungen zum Einfügen eines Dokumentes in eine Publikationsrelation (inklusive der zugehörigen Verknüpfungsrelationen:

4.3.1 gmdReportTable:

```
                ERT             TO gmdReportTable       AL      E
(
      ,
                        ,
            on      eption und      mplementierung eigenen Publ     ation     stems      ,
FileToLO(            home      poerner      pumaAbstract      txt      ),
FileToLO(            home      poerner      pumaFulltext      txt      ),
FileToLO(            home      poerner      puma      pd           ),
                                    tudien      ,
                                ,
                                ,
            an      t Augustin,       erman
);
```

Anmerkung:
Nach der INSERT INTO-Anweisung folgt die Angabe der Zielrelation. Wenn die einzelnen Attributwerte in der Reihenfolge angegeben werden, wie sie auch in der Relation definiert wurden, muß nur noch ein Hinweis VALUES erfolgen.
Die Angabe der Attributwerte wird dann in Klammern gesetzt. Die einzelnen Werte werden durch Kommas voneinander getrennt.
Werte des Datentyps smallint können ohne jede weitere Formatierung angegeben werden. Werte des Datentyps text werden zwischen Hochkommas gesetzt.
Werte des Datentyps large_text oder large_object werden folgendermaßen angegeben:
Mit FileToLO (File to large_object) wird dem System mitgeteilt das es sich um ein large_object (oder large_text) handelt. Danach wird in Klammern der Pfad der Originaldatei angegeben. Wichtig dabei ist, daß die Datei sich auf einem Dateisystem befindet, auf welches das Illustra-System zugreifen kann.
Die SQL-Anweisung wird durch ein Semikolon abgeschlossen, welches dem DBMS das Ende der SQL-Anweisung signalisiert.

4.3.2 newPublicationPersonnelTable:

```
                ERT             TO ne       PublicationPersonnelTable       AL      E
(
      ,
);
```

Anmerkung:
Zuerst erfolgt die Angabe der Publikationsnummer, dann die Angabe der Personalnummer des Autors. Voraussetzung ist also, daß jeder Autor schon einen Eintrag in der Relation personnelTable hat (dies gilt auch für IPSI-Mitarbeiter, da personnelGmdTable eine Unterklasse der Relation personnelTable ist). Hätte die Publikation mehrere Autoren, so müßte für jeden Autor eine eigene INSERT-Anweisung erfolgen.

4.3.3 newPublicationActivityTable:

```
                ERT             TO ne       PublicationActivit      Table       AL      E
(
      ,
);
```

Anmerkung:
Hier gelten die gleichen Regeln wie für die Autoren. Wäre die Publikation mit mehreren Aktivitäten verknüpft, so müßte für jede Aktivität eine eigene INSERT-Anweisung erfolgen. Jede dieser Aktivitäten muß schon innerhalb der Relation activityTable vorhanden sein.

4.3.4 Zur Verfügung stehende Dokumente

In die Datenbank wurden alle verfügbaren Publikationen des Arbeitsbereiches CUI der Jahre 1992-1996 eingefügt. Die Publikationen präsentieren sich folgendermaßen:

Publikationsart	Anzahl	mit Abstract	ohne Abstract
Artikel	59	55	33
GMD-Publikationen	10	5	5
Monographien	4	0	0
Diplomarbeiten	9	0	8
Gesamt	82	60	46

Tabelle 6: Zur Verfügung stehende Dokumente

Die Publikationen wurden von insgesamt 64 verschiedenen Autoren verfaßt und sind insgesamt 12 verschiedenen Aktivitäten zugeordnet.

4.4 Erzeugung einer graphischen Benutzerschnittstelle

Um eine Interaktion zwischen der Datenbank und dem Anwender zu ermöglichen, ist es notwendig, eine Benutzerschnittstelle zu schaffen, die dem Anwender folgende Möglichkeiten bietet:

- Suche von Dokumenten nach verschiedenen Kriterien
- Anzeige der Suchergebnisse
- Hinzufügen neuer Datensätze zur Datenbank

Da nicht davon ausgegangen werden kann, daß der Anwender über ein fundiertes Wissen darüber verfügt, wie eine Datenbank unter Illustra abgefragt bzw. gepflegt werden kann, ist es angebracht, diese Benutzerschnittstelle möglichst intuitiv und einfach zu gestalten. Andererseits sollte sie dem Anwender einen möglichst breiten Funktionsumfang bieten. Da das System außerdem vom WWW aus erreichbar sein sollte, scheint es sinnvoll, die Schnittstelle als graphisches HTML-User-Interface zu gestalten.

Für die Schaffung der graphischen Benutzerschnittstelle wird das Illustra Web DataBlade Modul verwendet, das schon in Abschnitt 3.4.3 eingehend beschrieben wurde. Das Illustra Web-DataBlade stellt verschiedene Funktionalitäten zur Verfügung, die es ermöglichen, vom WWW aus auf eine Illustra-Datenbank zuzugreifen.

Für das Text-Retrieval im Dokumentenbestand wird das PLS Text DataBlade Modul verwendet. Das PLS Text DataBlade Modul wurde schon in Abschnitt 3.4.2 eingehend erläutert.

Im folgenden Text wird beschrieben, wie die einzelnen Systemfunktionalitäten mit der Hilfe der beiden zur Verfügung stehenden DataBlade-Komponenten realisiert werden.

4.4.1 Suche im Dokumentenbestand

4.4.1.1 Eingabe der Suchanfrage

Um dem Anwender einen möglichst einfachen und intuitiven Zugriff auf die mächtigen Suchmöglichkeiten des Illustra-Systems in Verbindung mit dem PLS Text DataBlade zu bieten, soll die Eingabe der Suchanfrage für das Dokumenten-Retrieval über ein vorgefertigtes Suchformular erfolgen.

Vor der Erstellung dieser Suchmaske ist es nötig, festzustellen, welche Suchmöglichkeiten das Publikations-Management-System zuläßt und welche dieser Kriterien für den Anwender von Bedeutung sind.

- **Suche mit Boole'schen Mengenoperatoren und Abstandsoperatoren.**
  Diese Suchmöglichkeit ist ein absolutes Mindestmaß für eine Text-Retrieval-Schnittstelle. Ohne Suchoperatoren besteht keine Möglichkeit, sinnvoll innerhalb großer Textmengen zu suchen.

- **Best-Match-Suche und Konzeptsuche**
  Obwohl die Best-Match-Suche (die in Abschnitt 3.4.2.1.1.1 angesprochen wurde) das wohl am häufigsten angewandte Suchverfahren innerhalb von Text-Retrieval-Systemen darstellt , scheint es sinnvoll, eine Konzeptsuche (auch Abschnitt 3.4.2.1.1.2) anzubieten. Gerade für Anwender, die aufgrund fehlender Hintergrundinformationen keine adäquate Suchanfrage formulieren können, kann eine Konzeptsuche unter Umständen Suchergebnisse von hoher Relevanz zurückliefern.

- **Begrenzung der Textsuche auf bestimmte Dokumentbereiche**
  Oftmals möchte ein Anwender nur nach bestimmten Publikationstiteln suchen. In diesem Falle ist es daher eher hinderlich, wenn das Retrieval-System eingegebene Suchanfragen statt nur über die Dokumententitel auch über die Volltexte und Abstracts der Publikationen durchführt. Die Folge wäre wahrscheinlich, daß das System ein große Menge für den Anwender irrelevanter Dokumente zurückgibt. Für eine Feldsuche spricht noch ein weiterer Grund:
  Abstracts von Publikationen stellen (vorausgesetzt, sie wurden adäquat verfaßt) die Quintessenz einer Publikation dar. Mit verhältnismäßig wenigen Wörtern wird der Inhalt eines unter Umständen sehr umfangreichen Dokumentes ziemlich präzise dargestellt. Es handelt sich bei Abstracts daher um für Text-Retrieval hochinteressante Textbereiche. Es erscheint daher nur als folgerichtig, ein getrenntes Retrieval über die Abstracts durchführen zu können.

- **Eingrenzung der suchbaren Dokumentenmenge nach bestimmten Kriterien**
  Wenn einem Anwender schon von vornherein klar ist, von welchem Autoren die gesuchten Publikationen stammen sollen oder in welchem Zeitraum diese veröffentlicht wurden, so sollte man ihm eine Möglichkeit geben, derartige Suchkriterien in seine Suchanfrage einfließen zu lassen. Die Autoren einer Publikation lassen sich in unserem Fall über die Relation `newPublicationPersonnelTable` abfragen. Das Veröffentlichungsjahr jeder Publikation wird im Attribut publicationYear der Publikationsrelationen (`articleTable`, `gmdReportTable`, `monographyTable`, `diplomaTable` oder deren Oberklasse `newPublicationTable`) festgehalten. Über die Relation `newPublicationActivityTable` kann abgefragt werden, welcher IPSI-Aktivität die Publikation angehört. Es ist jedoch fraglich, ob es sinnvoll wäre, diesen Suchaspekt in das Suchformular aufzunehmen: Erstens sollte die Suchmaske nicht überfrachtet und unübersichtlich sein. Zweitens kann davon ausgegangen werden, daß ein großer Teil der potentiellen Anwender des IPSI-Publikations-Management-Systems nicht unbedingt darüber Bescheid weiß, wie dieses oder jene Projekt genau heißt. Der Aktivitätenaspekt soll

---

Es scheint sich hierbei um eine derart triviale Feststellung zu handeln, daß es nicht gelang, eine Literaturreferenz zur Stützung dieser Behautpung zu finden. Doch schon allein die Tatsache, daß die meisten Online-Datenbanken bei Hostanbietern wie beispielsweise Genios oder Web-Search-Engines wie WebCrwaler oder Lycos lediglich mittels des Best-Match-Verfahrens durchsucht werden können, genügt eigentlich schon, das Best-Match-Verfahren als das am häufigsten angewandte Retrieval-Verfahren zu bezeichnen.

jedoch nicht komplett verloren gehen. Er wird daher zu einem späteren Zeitpunkt wieder aufgegriffen.

- **Eingrenzung der Dokumentenmenge nach Publikationsart**
  Für den Fall, daß sich der Anwender nur für eine bestimmte Publikationsart interessiert, ist es für ihn nützlich, angeben zu können, daß die zurückgegebenen Dokumente nur dieser Publikationsart entstammen.

4.4.1.1.1 Implementierung der Suchmaske (searchForm)

HTML stellt einen Satz spezieller Elemente zur Verfügung, mit deren Hilfe sich Formulare erstellen lassen, in die der Anwender entsprechende Eingaben machen kann. Diese Daten können dann an eine Zieladresse geschickt werden. Folgende Form-Elemente können mit HTML realisiert werden:

- **Texteingabefelder** - Die Größe eines Texteingabefeld ist frei konfigurierbar. So kann ein Texteingabefeld ein einziges Wort aufnehmen. Denkbar ist aber auch die Eingabe kompletter Textdokumente über ein solches Eingabefeld.
- **Auswahllisten** - Über eine Auswahlliste kann der Anwender eine oder mehrere Optionen aus einer vordefinierten Optionsliste auswählen.
- **Auswahlfelder** - Aus mehreren Auswahlfeldern kann der Anwender beliebig viele Optionen auswählen.
- **Radio-Buttons** - Vergleichbar mit den Stationstasten eines Radios, kann der Anwender hier eine von mehreren Optionen auswählen.
- **Versteckte Eingabefelder** - Mit versteckten Eingabefelder ist es möglich, Daten mit einem Formular zu übertragen, ohne daß der Anwender dies registriert.
- **SUBMIT- und RESET-Knopf** - Der SUBMIT-Knopf dient der Übertragung der in das Formular eingegebenen Daten. Der RESET-Knopf wird dazu benutzt, die Eingabefelder eines Formulars im Bedarfsfall wieder in ihren Urzustand zurückzuversetzen.

Die einzelnen Suchkriterien wurden folgendermaßen realisiert:

- Da eine Suchanfrage immer nur ein Suchverfahren anwenden kann (Best-Match oder Konzept) wird die Eingabe des Suchverfahrens über Radio Buttons geregelt.

  Anmerkung:
  Wird vom Anwender die Konzeptsuche ausgewählt, so wird dem im folgenden Texteingabefeld eingegebenen Suchausdruck, der dann intern in Klammern gesetzt wird, ein Ausrufungszeichen hinzugefügt. Dieses Ausrufungszeichen ist die Anweisung für das PLS Text DataBlade, eine Konzeptsuche durchzuführen. Wählt der Anwender die Best-Match-Suche, so wird dem Suchausdruck ein Leerzeichen hinzugefügt. Das Fehlen des Konzept-! führt dazu, daß PLS eine Best-Match-Suche durchführt.

- Die Eingabe der eigentlichen Suchanfrage (Suchbegriffe mit Suchoperatoren) erfolgt über ein Texteingabefeld.
- Die Auswahl der für eine Feldsuche in Frage kommenden Felder erfolgt über Radio Buttons. Folgende Wahlmöglichkeiten werden angeboten: »Alle Felder«, »nur Volltext«, »nur Abstract«, »nur Titel«
- Wird vom Anwender ein bestimmtes Feld zur Suche bestimmt, so wird ein Textstring folgenden Formats eingesetzt:
  `/f: attributname` (wobei `attributname` in unserem Fall `publicationTitle`, `publicationAbstract` oder `publicationFullText` sein kann)

Anmerkung:
Die /f-Option weist das PLS TextDataBlade an, nur nach Begriffen zu suchen, die innerhalb des angegebenen Feldes (oder besser: Attributes) vorkommen. Wird vom Anwender die Option »All indexes« gewählt wird statt der /f-Option ein Leerzeichen eingesetzt.

- Über eine Auswahlliste kann der Anwender einen Autoren für seine Suche bestimmen.
  Um eine vollständige Liste aller im System verfügbaren Autoren zu erhalten, wird über MISQL eine Datenbankanfrage an die Tabelle `newPublicationPersonnelTable` gestellt, welche die Personalnummer (`personnelId`) aller in ihr gespeicherten Autorenschaften zurückgibt. Mittels dieser `personnelId` wird dann aus der Tabelle `personnelTable` der Vor- und Nachname der jeweiligen Personen ausgelesen. Die `personnelId` wird als Weitergabewert für das Suchformular genutzt. Vor- und Nachname dienen zur Bildschirmanzeige in der Auswahlliste:

```
ELECT      A     E          A     T     OR
                    L           L                ELECT                T
                p     personnel     ur     ame,
                p     personnelFirst     ame
 FRO              ne     PublicationPersonnelTable n,
                personnelTable p
              ERE          n     personnel     d     p     personnel     d
 OR     ER BY        p     personnel     ur     ame;
OPT     O          AL     E                                          ,
                              L
OPT     O          ELECTE          AL     E
          ELECT
```

Anmerkung:
Nach den durch MISQL generierten Optionsfelder wird noch ein einzelnes Optionsfeld »No specific« angefügt. Diese Option teilt dem Illustra-System mit, daß die Suche für alle verfügbaren Autoren durchgeführt wird.

- Um den Veröffentlichungszeitraum der gesuchten Publikationen einzugrenzen, steht eine Auswahlliste aller verfügbaren Veröffentlichungsjahre zur Verfügung. Aus dieser Liste können beliebig viele Jahreszahlen ausgewählt werden.
  Auch diese Auswahlliste wird über MISQL generiert:

```
ELECT      A     E          YEAR               LT     PLE                E
                    L           L                ELECT                T
          publicationYear
 FRO           ne     PublicationTable
 OR      ER BY
          publicationYear;
OPT     O          ELECTE
                              L
          ELECT
```

Anmerkung:
Standardmäßig werden alle generierten Jahre als ausgewählt gekennzeichnet; somit wird im Normalfall die vom Anwender eingegebene Suchanfrage auf die Publikationen aller verfügbaren Veröffentlichungsjahre angewendet.

Der Vorteil der durch MISQL generierten Auswahllisten liegt auf der Hand:

Die Auswahlfelder werden bei jedem Aufruf des Suchformulars neu generiert und sind daher stets aktuell. Der Anwender bekommt immer alle verfügbaren Autoren bzw. Veröffentlichungsjahre angezeigt.

- Die Auswahl der Publikationsarten, die zurückgegeben werden sollen, erfolgt über Auswahlfelder. Dabei wird für jede Publikationsart ein Auswahlfeld zur Verfügung gestellt.
  In einem versteckten Eingabefeld wird der Name der Application-Page angegeben, zu der die vom Anwender eingegebenen Daten geschickt werden sollen:

```
          P     T TYPE                    E                A     E
```

Anmerkung:
Die Application-Page »fulltextResult« übernimmt die Suche der Dokumente und die Anzeige der Ergebnisdaten.

- Als letzte Eingabefelder zeigen sich ein SUBMIT- und ein RESET-Button:

```
P       T TYPE                          B       T       AL      E
P       T TYPE              RE      ET
```

Das Suchformular »searchForm« zeigt sich im Web-Browser folgendermaßen:

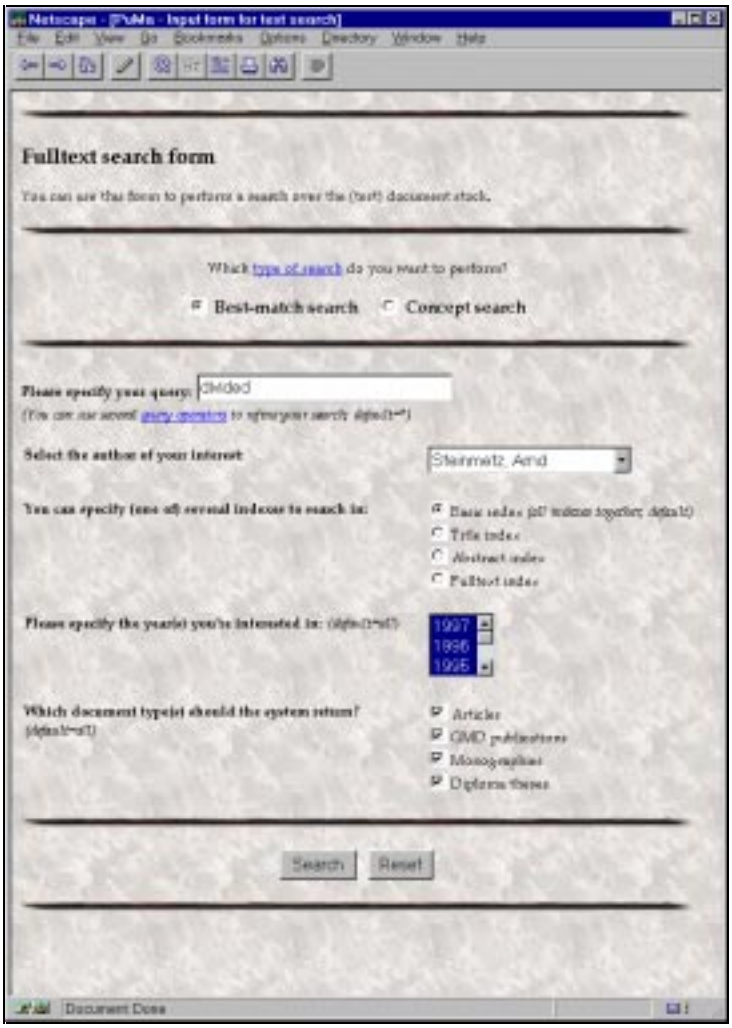

Abbildung 7: Suchformular »searchForm«

4.4.1.2 Suche der Dokumente und Ergebnisanzeige

Die vom Anwender eingegebenen Suchdaten werden vom Suchformular »searchForm« in Form von Prozeßvariablen übergeben:

Name der Variable	Inhalt	Format
$STYPE	Suchverfahren (Best-Match- oder Konzeptsuche)	»!« (für Konzeptsuche) Leerzeichen (für Best-Match-Suche)
$SEARCH	Suchanfrage mit Suchbegriffen und -operatoren	Bsp.: »`computer and 3d`«

$AUTHOR	Autor der gesuchten Dokumente	die personnelId einer Person in personnelTable »=zahlenwert« oder »>0« für alle Autoren
$FIELD	Spezifizierung einer Feldsuche	»/f: publicationTitle« »/f: publicationAbstract« »/f: publicationFullText« oder : Leerzeichen für alle Felder
$YEAR	Angabe der Veröffentlichungsjahre	»zahlenwert« Bsp: »1997«
$TYPE	Angabe der Publikationsarten, die zurückgegeben werden sollen	»Articles« »GMD_publications« »Monographies« »Diploma_theses«

Tabelle 7: Für eine Dokumentensuche benötigte Prozeßvariablen

Für die Variablen $YEAR und $TYPE können u. U. mehrere Werte übergeben werden. Um diese für den weiteren Gebrauch zu erhalten, ist es notwendig, sie in jeweils einen String zu schreiben; um diesen String später wieder in seine einzelnen Werte aufteilen zu können, ist es angebracht, ein Trennzeichen zwischen den einzelnen Werten einzufügen. Dieser Vorgang wird mit dem MIVAR-Tag erledigt:

```
AR      A     E                    E          EPARATE      ,
AR      A     E                    AT     EPARATE          ,
```

Die Werte in $YEAR werden, durch Kommas getrennt, als ein String der Variablen $WHEN übergeben. Das gleiche geschieht mit den Werten in $TYPE (nach $WHAT).

Die in Variablenform zur Verfügung stehenden Suchdaten sollen nun in eine SQL-Suchanfrage eingearbeitet werden, die an die Dokumentendatenbank gestellt wird. Die Suchergebnisse sollen dem Anwender auf einer Ergebnisseite angezeigt werden. Bei der Ergebnisanzeige scheint folgende Vorgehensweise am sinnvollsten:

- Eine Übersichtsseite, auf der eine Zusammenfassung der Suchergebnisse in Form von Literaturreferenzen angezeigt wird.
- Auf der Übersichtsseite soll der Anwender jede einzelne Referenz anwählen können und um das jeweilige Dokument in einer detaillierten Form anzeigen zu lassen.
- Die Suche der Dokumente und Anzeige der Suchergebnisse kann innerhalb einer Application-Page realisiert werden.

4.4.1.2.1 Implementierung der Such- und Ergebnisseite (fulltextResult)

Der Umstand, daß die unterschiedlichen Publikationsarten in verschiedenen Relationen gehalten werden, führte dazu, daß für jede Publikationsart ein eigener PLS-Textindex erstellt wurde:

Publikationsart	Datenbankrelation	PLS-Textindex
Artikel	articleTable	articleIndex
GMD-Publikationen	gmdReportTable	gmdReportIndex
Monographien	monographyTable	monographyIndex
Diplomarbeiten	diplomaTable	diplomaIndex

Tabelle 8: Relationen und Indizes für die einzelnen Publikationsarten

Innerhalb der PLS-Indizes werden folgende Attribute invertiert:

- `publicationTitle`
- `publicationAbstract`
- `publicationFullText`

Anmerkung:
Bei den Artikeln werden zusätzlich noch die Attribute articlePTitle und articlePAuthors invertiert. Sie können nicht getrennt durchsucht werden, sondern werden bei einer alle Felder umfassenden Suche mit einbezogen. Dies Einschränkung wurde deswegen gemacht, weil einerseits zwei zusätzliche Suchoptionen das Suchformular unübersichtlicher hätten erscheinen lassen. Andererseits kann den genannten Attributen kein so hoher Suchwert wie den anderen zugesprochen werden.

Bis auf die Eigenheiten jeder Publikationsart besitzen die Publikationstabellen dieselbe Attributsstruktur:

- `publicationId`
- `publicationYear`
- `publicationTitle`
- `publicationType`
- `publicationAbstract`
- `publicationFullText`
- `publicationPDF`

Der analoge Aufbau der Textindizes und die fast identische Struktur der Publikationstabellen legt den Gedanken nahe, die Dokumentensuche mit einer, alle Tabellen und Indizes umfassenden, SQL-Suchanfrage zu realisieren. Eine solche (stark vereinfachte) SQL-Anfrage würde folgendermaßen aussehen:

```
      ELECT                  T           CT
       p      publication      d,
       p      publicationYear,
       p      publicationTitle
FRO         publication p,
       PlsOidRan            uer      (      article      ndex      ,      (            EARC      )
       PlsOidRan            uer      (      gmdReport      ndex      ,      (            EARC      )
       PlsOidRan            uer      (      monograph           ndex      ,      (            EARC      )
       PlsOidRan            uer      (      diploma      ndex      ,      (            EARC      )
            ERE  a      pls_oid      p      oid
OR     g      pls_oid      p      oid
OR     m      pls_oid      p      oid
OR     d      pls_oid      p      oid;
```

Leider läßt das PLS Text DataBlade eine derartige SQL-Anfrage nicht zu. Das logische OR zum Abgleich der ObjektIDs am Ende des SQL-Blocks wird vom PLS TextDataBlade wie ein logisches AND interpretiert. Vom Hersteller Personal Library Software war diesbezüglich nicht zu erfahren, ob es eine Abhilfe für diesen offensichtlichen Anwendungsfehler des Text DataBlade gibt.

Doch auch ein anderer Aspekt spricht gegen eine einzige SQL-Anfrage:

Für eine adäquate Referenzlisten werden mehr Informationen als nur der Titel und das Veröffentlichungsjahr einer Publikation benötigt. Beispielsweise ist es gerade bei unselbständigen Publikationen wie Artikeln wichtig, auf einen Blick erkennen zu können, wie der Titel der übergeordneten Publikation lautet. Dies bedeutet, daß auch auf die speziellen Attribute jeder Publikationsrelation zugegriffen werden muß. Bei der späteren Formatierung der Ergebnisdaten innerhalb des MISQL-Blocks, die zeilenweise erfolgt, würden sich bei einer einzigen Abfrage aller Attribute zwangsläufig Schwierigkeiten ergeben.

Somit entsteht die Notwendigkeit, die Textsuche und Formatierung der Ergebnisse für jede Publikationsart getrennt durchzuführen. Es ergeben sich also letztendlich vier einzelne SQL-Abfragen.

Anmerkung:
Um die folgende Darstellung der Implementierung der Application-Page »fulltextResult« nicht zu voluminös werden zu lassen, werden die Implementierungsschritte, die sich auf die einzelnen Publikationsarten beziehen, beispielhaft durch die Publikationsart »GMD-Publikationen« dargestellt. Der komplette Quellcode dieser (und jeder anderen folgenden) Application-Page findet sich im Anhang.

4.4.1.2.1.1 Formulierung der Suchanfrage

Für die Anzeige der Referenz einer GMD-Publikation in der Ergebnisseite sind folgende Informationen (Attribute) der Tabelle gmdReportTable von Relevanz:

- Publikationsnummer (`publicationId`)
- Publikationstitel (`publicationTitle`)
- Veröffentlichungsjahr (`publicationYear`)
- Publikationstyp (`publicationType`)
- lfd. Nummer der Publikation (`gmdReportVolume`)
- Verlag (`gmdReportPublisher`)
- Verlagsort (`gmdReportPubLocation`)

Außerdem ist es wichtig, den Autoren in der Referenz anzuzeigen. Daher werden noch Informationen (Attribute) aus der Tabelle personnelTable notwendig:

- Vorname (`personnelFirstName`)
- Nachname (`personnelSurName`)

Um die Namen der Autoren zu erfahren, muß zunächst die `publicationId` der Autoren aus der Tabelle `newPublicationPersonnelTable` ermittelt werden. Mit dieser `publicationId` können dann Vor- und Nachname der Autoren aus `personnelTable` ermittelt werden.

Als Sucheinschränkungen sollen noch die Werte aus $YEAR, $STYPE und $FIELD in die Suchanfrage mit einfließen.

Das MISQL-Statement präsentiert sich folgendermaßen:

```
                                        L              L
      ELECT                        T             CT
       g      publicationYear,
       g      publicationTitle,
       g      publicationT      pe,
       g      gmdReport      olume,
       g      gmdReportPublisher,
       g      gmdReportPubLocation,
       n      publication      d,
       p      personnelFirst      ame,
       p      personnel      ur      ame
FRO
       personnelTable p,
       ne      PublicationPersonnelTable n,
```

```
        gmdReportTable g,
        PlsOidRan            uer      (      gmdReport      ndex      ,      (             EARC      )
             ERE
        (g             pls_oid      g      oid)
A                (n      personnel      d      p      personnel      d)
A                (n      publication      d      g      publication      d)
A                (g      publicationYear in (                  E      ))
A                (g      publication      d in
                          (      ELECT                  T            CT publication      d
                            FRO       ne      PublicationTable
                                        ERE personnel                 A      T      OR)
OR       ER BY
        g      publicationYear desc;
```

Bevor das MISQL-End-Tag erfolgt, müssen die Formatierungsanweisungen für die Suchergebnisse definiert werden.

Anmerkung:
Hat eine Publikation mehrere Autoren, so gibt das oben angeführte SQL-Statement diese Publikation einmal für jeden einzelnen Autoren zurück. Dies bedeutet, daß eine Publikation mit mehreren Autoren mehrfach in der Ergebnisliste angezeigt würde. Da die meisten wissenschaftlichen Publikation vom mehreren Autoren verfaßt werden, würde dieser Umstand zu einer sehr unübersichtlichen Ergebnisanzeige führen.

Um dies zu verhindern, werden in der folgenden Formatierungsanweisung einige der vom Web DataBlade angebotenen Prozeßfunktionen genutzt:

```
(      F,
     (E      ,      ,      T      TLE),      ,      FRO      T      RE      T      (
     (      ET      AR,      RE      T,      (      )      BR
(      ET      AR,      FRO      T,      FRO      T      ,      ;
(      ET      AR,      T      TLE,      )
                                  L
                        AR      FRO      T      RE      T
```

Der Funktionsablauf dieser Formatierungsanweisung wird nun folgend detailliert erläutert:

- Zuerst prüft die Routine, ob der in `$TITLE` enthaltene Wert dem Wert entspricht, der in der Rückgabevariablen $2 (`publicationTitle`) enthalten ist.

  ```
  (E      ,      ,      T      TLE)
  ```

- Taucht ein `publicationTitle` also zum ersten Male auf, kann `$TITLE` nicht `$2` entsprechen. Es wird `dofalse` (die Anweisung nach dem zweiten Komma der `IF`-Funktion) angewendet. Diese gibt die in `$FRONT` und `$REST` enthaltenen Werte aus.

  ```
  ,      FRO      T      RE      T
  ```

- `$FRONT` und `$REST` sind im ersten Lauf noch leer. Es wird somit nichts ausgegeben. Dann wird `$FRONT` mit der danach folgenden `SETVAR`-Anweisung wieder geleert.

  ```
  (      ET      AR,      FRO      T,      )
  ```

- Danach bekommt `$REST` die Werte aus `$1`, `$2`, `$3`, `$4` und `$5` zugewiesen.

  ```
  (      ET      AR,      RE      T,      (      )      BR
  ```

- Ist die erste `IF`-Schleife ausgeführt , wird `$FRONT` zu dem schon in ihm enthaltenen Wert (der jetzt noch nicht existiert) der Wert aus `$9` (`personnelSurName`) und `$8` (`personnelFirstName`) zugewiesen.

  ```
  (      ET      AR,      FRO      T,      FRO      T      ,
  ```

- Der erste Autor wird somit in `$FRONT` geschrieben. Dann wird `$TITLE` auf den Wert in `$2` gesetzt.

  ```
  (      ET      AR,      T      TLE,      )
  ```

- In der nächsten Runde entspricht `$TITLE $2` und es wird nichts ausgegeben (`dotrue`). Nun kommt zu dem schon vorhandenen Wert in `$FRONT` der zweite Autorenname und `$REST` wird wieder auf den Wert in `$2` gesetzt. Dies wird solange durchgeführt, bis die Routine auf einen neuen Wert in `$2` (also einen neuen `publicationTitle`) stößt. Dann werden die Inhalte von `$FRONT` (alle Autorennamen) und `$REST` (alle restlichen Referenzinformationen zu der Publikation) ausgegeben. Für die nachfolgenden Titel wird die ganze Prozedur nochmals durchgeführt; solange, bis keine Rückgabewerte mehr vorhanden sind. Die letzte Publikation würde normalerweise von der Routine verschluckt.

Daher werden nach dem MISQL-End-Tag die Inhalte von `$FRONT` und `$REST` noch einmal ausgegeben.

```
                                                L
                                  AR              FRO       T       RE       T
```

Damit ist die Ausgabe einer Referenz abgeschlossen.

Dieses Verfahren ermöglichst es

1. die Autoren über der Publikationsreferenz
2. jede Publikationsreferenz auch wirklich nur einmal

auszugeben.

Ein Problem, daß bei dem vorstehenden MISQL-Block noch bleibt, ist, daß der Anwender in dem Fall, daß keine Dokumente für seine Suchanfrage gefunden wurden, keine Rückmeldung vom System bekommt. Der Ergebnisbildschirm zeigt sich einfach leer. Es ergeben sich somit Interpretationsschwierigkeiten für den Anwender, da es theoretisch mehrere Möglichkeiten für einen leeren Ergebnisbildschirm gibt:

- Es wurden keine Dokumente gefunden.
- Das DB-System hat die Bearbeitung der Suchanfrage (aus welchen Gründen auch immer) abgebrochen.
- Die Internet-Verbindung ist zusammengebrochen
- usw.

Die Erkenntnis aus dieser Schwierigkeit ist, daß der Anwender für den Fall, daß keine Dokumente für seine Suchanfrage gefunden wurden, eine entsprechende Rückmeldung vom System erhalten muß.

Diese Rückmeldung läßt sich über eine zweite, der oben angeführten `SELECT`-Anweisung vorgelagerte SQL-Anfrage regeln.

Über ein `SELECT COUNT`, daß in den Abfragebedingungen dem oben stehenden `SELECT` entspricht, läßt sich feststellen, wieviel Dokumente zu der vom Anwender formulierten Suchanfrage vom System zurückgegeben werden:

```
                                        L             L                    ELECT CO            T (      )
FRO
       gmdReportTable g,
       PlsOidRan            uer      (      gmdReport      ndex      ,      (            EARC      )      T
            ERE
       (g            pls_oid      g      oid)
A                (g      publicationYear in (                E      )
A                (g      publication      d in
                          (      ELECT                  T           CT publication      d
                           FRO                    ne      PublicationPersonnelTable
                                          ERE          personnel      d      A      T      OR));
```

Findet die Suchanfrage keine Dokumente (Rückgabewert=0), dann wird eine Überschrift »No GMD publications retrieved« ausgegeben. Sind Dokumente vorhanden (Rückgabewert>0), wird eine Überschrift »Retrieved GMD publications« ausgegeben.

```
      (      F,      (E      ,           ,      ),                                    o
tions                                          R           )
                                        L
```

Wenn das `SELECT COUNT` keine Dokumente zurückgibt, dann muß die obere `SELECT`-Anweisung, welche die Dokumentenreferenzen zurückgibt, auch nicht mehr ausgeführt werden. Es wäre zwar an sich nicht falsch, die Suchanweisung auszuführen. Es erscheint jedoch unsinnig, eine SQL-Anweisung auszuführen, welche

- dem Anwender keine Ergebnisse zurückgibt
- die Antwortzeit unnötig verlängert

- das Illustra-System unnötig belastet.

Es ist somit ein sinnvoller Schritt, die obere SQL-Anweisung in einen `MIBLOCK` einzubetten, der erst dann ausgeführt wird, wenn die `SELECT COUNT`-Anweisung einen Wert größer als 0 zurückgibt.

```
BLOC    CO                    (    ,    ,          )
```

Wie schon weiter oben beschrieben, hat der Anwender die Möglichkeit, auszuwählen, von welcher Publikationsart die Dokumente sein sollen, die er zurückgeliefert haben möchte. Daher stellt es einen folgerichtigen Schritt dar, das System nur dann nach GMD-Publikationen suchen zu lassen, wenn der Anwender diese als Rückgabe-Dokumente bestimmt hat. Die gewünschte(n) Publikationsart(en) wird/werden vom Suchformular »searchForm« mittels der Variable `$TYPE` an »fulltextResult« übermittelt. Es ist nun also möglich, sowohl das `SELECT COUNT`, mit dem die Anzahl der gefundenen Dokumente ermittelt wird und das schon in ein `MIBLOCK` eingebettete `SELECT`, mit dem die Referenzinformationen der entsprechenden Dokumente ermittelt werden, wiederum in ein `MIBLOCK` einzubetten, welches nur dann ausgeführt wird, wenn der Anwender die jeweilige Publikationsart (in diesem Fall »GMD-Publikationen«) im Suchformular angegeben hat:

```
BLOC    CO                    (    ,    ,    (PO          T    O
```

Mit der `POSITION`-Funktion wird die Anfangsposition des Strings »GMD_publications« innerhalb des in der Variable `$WHAT` enthaltenen Strings ermittelt. Ist »GMD_publications« in `$WHAT` erhalten, wird ein Wert größer `1` zurückgegeben und die Ausführungsbedingung trifft zu.

Der fertige Block zur Suche und Formatierung der »GMD-Publikationen« präsentiert sich komplett folgendermaßen:

```
                    BLOC    CO                    (    ,    ,    (PO          T    O
                                     L          L              ELECT CO          T (    )
          FRO        gmdReportTable g,
                PlsOidRan            uer     (     gmdReport     ndex     ,     (          EA
                      ERE
                (g            pls_oid     g     oid)
          A            (g     publicationYear in (               E     )
          A            (g     publication     d in
                                     (     ELECT               T          CT publication
                                      FRO               ne     onPublicationPersonnelTable
                                                    ERE     nel     person A     T     OR));

     (     F,     (E     ,          ,     ),                              o
retrieved                                R          ,                              Retrieved

                                           L
                    BLOC    CO                    (    ,    ,          )
                                        L          L
        ELECT                    T          CT
         g     publicationYear,
         g     publicationTitle,
         g     publicationT     pe,
         g     gmdReport     olume,
         g     gmdReportPublisher,
         g     gmdReportPubLocation,
         n     publication     d,
         p     personnelFirst     ame,
         p     personnel     ur     ame
    FRO
         personnelTable p,
         ne     PublicationPersonnelTable n,
         gmdReportTable g,
         PlsOidRan            uer     (     gmdReport     ndex     ,     (          EARC     )
              ERE
         (g          pls_oid     g     oid)
    A            (n     personnel     d     p     personnel     d)
    A            (n     publication     d     g     publication     d)
    A            (g     publicationYear in (               E     ))
    A            (g     publication     d in
                         (     ELECT               T          CT publication     d
                          FRO     ne     PublicationPersonnelTable
                                    ERE personnel               A     T     OR)
    OR     ER BY
         g     publicationYear desc;

         (     F,     (E     ,          ,     T     TLE),          ,     FRO     T     RE
         (     ET     AR,     RE     T,     (                    )          BR          A
```

```
TAR      ET      detail           B                       B            A
                       R     ))
        (      ET     AR,     FRO     T,      FRO     T             ,
        (      ET     AR,     T      TLE,            )

                                  L
                           AR            FRO     T     RE     T           RE     T
                         BLOC
                       BLOC
```

Zu beachten ist, daß der Publikationstitel (`$2`) als Hyperlink ausgelegt wird. Der Hyperlink ruft bei Aktivierung die Application-Page »publicationDetail« auf, die eine detaillierte Anzeige einer Publikation in einem neuen Browser-Fenster ermöglicht. Zur Bestimmung der Publikation, die in »publicationDetail« angezeigt werden soll, wird die Variable `$publication` übergeben. `$publication` enthält die `publicationId` der jeweiligen GMD-Publikation.

4.4.1.2.1.2 Anzeige der Suchanfrage

Da der Anwender auf dem Suchformular »searchForm« eine relativ komplexe und detaillierte Suchanfrage formulieren kann, scheint es angebracht zu sein, ihm diese Suchanfrage auf der Ergebnisseite, über den Suchergebnissen, nochmals zu präsentieren.

Um die Anzeige der Suchanfrage so übersichtlich wie möglich zu halten, wird sie in Form einer Tabelle gestaltet. Dabei stehen in der linken Spalte der Tabelle die Bezeichnungen der Suchoptionen. In der rechten Spalte werden die vom Anwender gewählten Werte gezeigt.

Der Wert für `$SEARCH` (die Suchbegriffe mit -operatoren) wird schon als ein Textstring übergeben und kann daher für die Anzeige unverändert übernommen werden.

Die Werte für `$YEAR` (die Veröffentlichungsjahre) bzw. `$TYPE` (die Publikationsarten) sind schon für die Suche als jeweils ein Textstring den Variablen `$WHEN` bzw. `$WHAT` zugewiesen worden. Auch sie können für die Anzeige unverändert übernommen werden.

Mit dem in `$AUTHOR` enthaltenen Wert verhält es sich anders. Er wird zwar auch in Form eines Textstrings geliefert (ein Vergleichsoperator mit einem Zahlenwert). Da dieser String jedoch für eine Anzeige ungeeignet ist - der Anwender hat den Autoren immerhin über eine Liste von Namen ausgewählt und wird mit einer Personalnummer nicht viel anfangen können - muß erst eine Datenbankabfrage stattfinden, die den Namen des ausgewählten Autors ermittelt.

```
                        L              L                   ELECT personnel      ur       ame,
          personnelFirst      ame
     FRO        personnelTable
                 ERE  personnel      d      A      T      OR;
                                L
```

Das Ergebnis wird als ein Textstring in der neu definierten Variablen `$NAME` zwischengespeichert, um es später in der Anzeigetabelle wiederverwenden zu können.

```
                       AR      A      E                A      E                ,
```

Für den Fall, daß der Anwender im Autorenfeld »No specific« ausgewählt hat, würde die oben stehende Datenbankanfrage die Namen aller in der Tabelle `personnelTable` gespeicherten Personen zurückgeben. Daß die Anzeige aller Personen aus `personnelTable` die Anzeigetabelle restlos überfrachten würde, wird wohl auch einem in den Kognitionswissenschaften weniger erfahrenen Leser einleuchten; ganz abgesehen davon, daß es falsch wäre, da innerhalb von `personnelTable` auch Personen gespeichert sind, die keine Publikationsautoren sind.

Um für diesen Fall vorzusorgen, wird dem oben gezeigten `SELECT`-Statement ein `SELECT COUNT` vorgeschaltet:

```
                        L              L                   ELECT CO            T (       )
     FRO        personnelTable
                 ERE  personnel      d      A      T      OR;
                                L
```

Gibt diese Anfrage einen Rückgabewert größer 1 zurück (in dem Fall, daß »No specific« gewählt wurde) wird `$NAME` der Textstring »No specific« zugewiesen.

```
BLOC      CO                    (     ,     ,          )
     AR      A     E                A     E          o speci     ic
   BLOC
```

Gibt die Anfrage hingegen einen Rückgabewert von `1` zurück, wird die vorher beschriebene Suchanfrage aktiviert und deren Ergebnis der Variablen `$NAME` zugewiesen:

```
BLOC      CO                    (E     ,     ,          )
               L          L              ELECT personnel     ur     ame,
     personnelFirst     ame
FRO       personnelTable
          ERE  personnel     d     A     T     OR;
                    L
               AR      A     E                A     E               ,
          BLOC
```

Die für die Textsuche aktivierten Suchfelder `publicationTitle`, `publicationAbstract` und `publicationFullText` kommen in der Variable `$TYPE` zwar auch in Form eines Textstrings an, jedoch enthält dieser Textstring zusätzlich noch die vorgeschaltete `/f`-Option. Wäre es beispielsweise noch im zumutbaren Rahmen, einen String in der Form /f: `publicationAbstract` anzuzeigen, ist das Leerzeichen, das in dem Fall, wenn der Anwender alle Felder durchsuchen möchte, in `$TYPE` enthalten ist, für die Anzeige wohl kaum zu gebrauchen. Das gleiche Problem ergibt sich bei dem in der Variablen `$STYPE` enthaltenen Wert für das Suchverfahren. Der Anwender wird weder mit der Angabe `Search type = !` noch `Search type=`*`leerzeichen`* viel anfangen können. Für die Werte in `$TYPE` und `$STYPE` muß somit auch eine Umsetzung der teilweise kryptischen Suchvariablen in für den Anwender interpretierbare Informationen stattfinden. Beginnen wir mit der Umsetzung von `$TYPE`:

- In den meisten Fällen wird der Anwender alle Publikationsarten zurückgeliefert bekommen wollen. Somit wird eine neue Variable `$PUB` definiert, die den Textstring »All indexes« zugewiesen bekommt, der in der Anzeigetabelle verwendet werden soll:

```
AR      A     E          P     B     All indexes
```

- In dem Fall, daß der Anwender ein spezielles Feld zur Suche ausgewählt hat, muß der Variable `$PUB` ein anderer Wert zugewiesen werden.

```
BLOC      CO                    (     ,     ,(PO          T     O
     AR      A     E          P     B     Title index
   BLOC

BLOC      CO                    (     ,     ,(PO          T     O
     AR      A     E          P     B     Abstract index
   BLOC

BLOC      CO                    (     ,     ,(PO          T     O
     AR      A     E          P     B     Fulltext index
   BLOC
```

- Trifft die Bedingung eines `MIBLOCK` zu (es wurde also ein bestimmtes Feld ausgewählt), so wird der vorher in `$PUB` festgelegte Wert »All indexes« einfach durch den jeweiligen neuen Wert überschrieben.
  Eine ähnliche Lösung wird auch für `$STYPE` (das Suchverfahren) realisiert:

```
AR      A     E                ERY     Best     match search

BLOC      CO                    (E     ,     ,          TYPE)
     AR      A     E                ERY     Concept search
   BLOC
```

- Nachdem alle Variablenwerte ordnungsgemäß verarbeitet wurden, können sie in Form einer HTML-Tabelle angezeigt werden:

```
                AR
TABLE BOR     ER
T               uer     (     uer     t     pe)     T                EARC
T          Author     T                A     E     TR
T          Years     T                     E          TR
T          Publication t     pe(s)     T                AT     TR
     TABLE
                          AR
```

Die Suchergebnisanzeige zeigt sich im Web-Browser des Anwenders folgendermaßen:

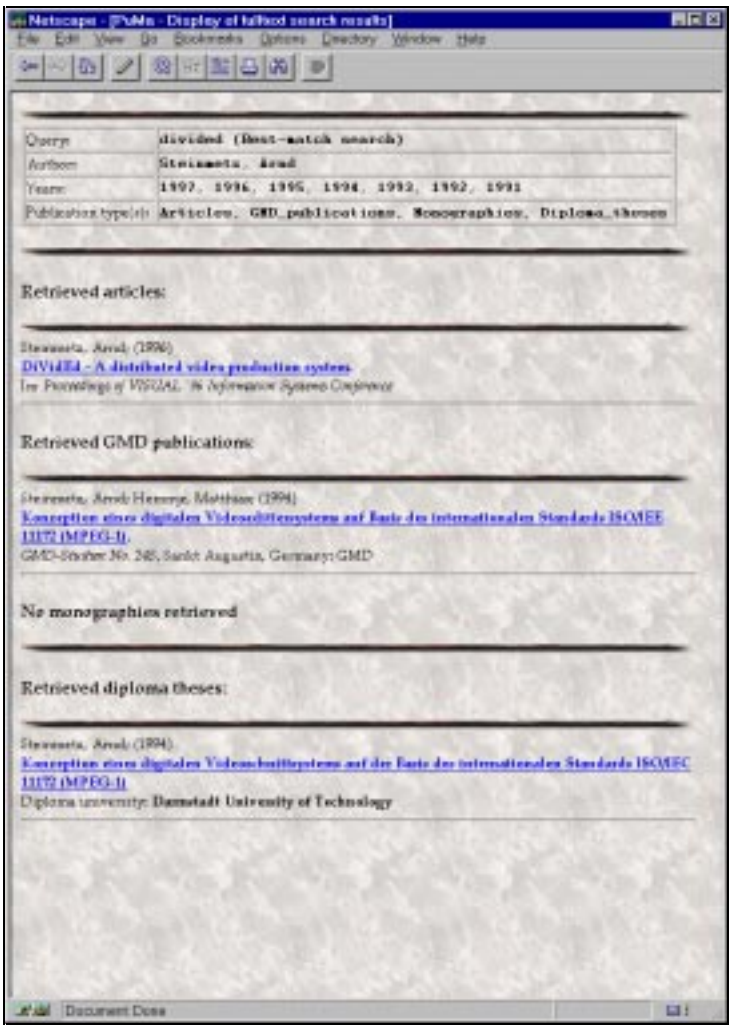

Abbildung 8: Suchergebnisanzeige »fulltextResult«

4.4.1.3 Anzeige einzelner Publikationen

Wie schon im vorhergehenden Abschnitt angesprochen, sollte der Anwender die Möglichkeit geboten bekommen, einzelne Dokumente aus der Ergebnisreferenzliste auszuwählen, um sich diese in einer detaillierten Anzeige betrachten zu können. In dieser Einzelanzeige sollten folgende Informationen vorgehalten werden:

- Alle bibliographischen Referenzinformationen. (Titel, Autor, Veröffentlichungsjahr, Verlag usw.)
- Der Abstract der Publikation (soweit vorhanden).
- Wenn eine elektronische Version des Volltextes der Publikation vorliegt, sollte diese zum Download angeboten werden.
- Zugehörigkeit zu einer oder mehreren IPSI-Aktivität(en).
- Die Zugehörigkeit einer Publikation zu einer IPSI-Aktivität wird über die Tabelle `newPublicationActivityTable` geregelt.
- Semantische Verknüpfung zu einer oder mehreren IPSI-Aktivität(en).
  Bei der semantischen Verknüpfung zwischen Publikationen und Aktivitäten wird von einer, zugegebenermaßen einfachen Retrievaltheorie ausgegangen:
  Taucht im Dokumententext (Titel, Abstract, Volltext) einer Publikation der Name einer IPSI-Aktivität

auf, und die Publikation ist dieser Aktivität in der Relation newPublicationActivityTable nicht zugeordnet, so kann davon ausgegangen werden, daß es einen inhaltlichen Zusammenhang zwischen der Publikation und der jeweiligen Aktivität gibt.

Anmerkung:
Die inhaltliche Verknüpfung eines Dokumentes zu einer bestimmten Aktivität wird bei der Dokumenteneinfügung (Abschnitt 4.4.2.1) durch eine PLS-gestützte Textsuche festgestellt. Der gefundene Kurzname einer Aktivität wird in der Tabelle newPublicationKeywordTable *gespeichert. Diese Tabelle wurde dem Datenbankschema nachträglich hinzugefügt:*

```
CREATE newPublicationKeywordTable OF NEW TYPE NewPublicationKeywordType
(
publicationId  smallint REFERENCES newPublicationTable,
keyword        text
);
```

4.4.1.3.1 Implementierung der Dokumenteneinzelanzeige (publicationDetail)

Möchte der Anwender detaillierte Informationen zu einer einzelnen Publikation ansehen, so klickt er in der Suchergebnisanzeige »fulltextResult« auf den Titel der jeweiligen Publikationsreferenz. Dieser ist als Hyperlink ausgelegt, der

1. die Dokumenteneinzelanzeige aufruft und
2. dieser die publicationId des gewünschten Dokumentes mittels der Variablen $publication übergibt.

In der Dokumenteneinzelanzeige soll, wie in der Suchergebnisanzeige »fulltextResult«, die Autoreninformation zu der jeweiligen Publikation dargestellt werden. Da hier nur eine Publikation angezeigt wird, ist es nicht nötig, den in »fulltextResult« angewandten $FRONT/$REST-Trick auch hier anzuwenden. Hier genügt es, zwei voneinander getrennte MISQL-Blöcke einzurichten; einen für die Ermittlung und Anzeige der Autoren und einen für die Ermittlung und Anzeige der restlichen Informationen. Zuerst erfolgt der MISQL-Block für die Autorenermittlung:

4.4.1.3.1.1 Ermittlung und Anzeige der Autorennamen

Die in der Variablen $publicationId enthaltene publicationId kann direkt in die SQL-Abfrage eingearbeitet werden:

```
                    L             L                  ELECT p      personnelFirst      ame,
        p      personnel      ur      ame
FRO         personnelTable p,
        ne      PublicationPersonnelTable n
              ERE  n      publication      d           publication
A                  p      personnel      d      n      personnel      d;
,             ;
                         L
```

Die Autoren werden nun im Format »Nachname, Vorname; Nachname, Vorname; usw.« dargestellt.

4.4.1.3.1.2 Ermittlung und Anzeige der Publikationsinformationen

Da in der Dokumenteneinzelanzeige nahezu alle der möglichen Informationen zu einer Publikation angezeigt werden sollen, ist es wiederum notwendig, auf die jeweilige Publikations-Unterklasse (articleTable, gmdReportTable, monographyTable oder diplomaTable) zuzugreifen. Somit muß die Ermittlung und Anzeige jeder Publikationsart in jeweils einem eigenen, durch MIBLOCK separierten Ausführungsblock ablaufen. Um die Implementierung nicht für jede Publikationsart erklären zu müssen, wird eine Publikationsart, die Artikel, als Beispiel angeführt.

Da der Publikationstyp (publicationType) auch Bestandteil der Publikations-Oberklasse newPublicationTable ist, kann die Abfrage zur Feststellung der Publikationsart des anzuzeigenden Dokumentes auch hierüber laufen. Das Suchergebnis (publicationType) wird in der Variablen $PTYPE zwischengespeichert.

```
                          L             L                  ELECT publicationT      pe
```

```
FRO        ne      PublicationTable
           ERE  publication      d          publication;
                              L

                AR       A      E                  PTYPE
```

Die unterschiedlichen MISQL-Blöcke, mit denen die detaillierten Informationen zu dem jeweiligen Dokument aus den Publikations-Unterklassen ermittelt werden, können nun in MIBLOCK-Ausführungsblöcke, die mit entsprechenden Bedingungen versehen sind, integriert werden:

```
BLOC       CO                    (E      ,      PTYPE,article)

BLOC       CO                    (E      ,      PTYPE,monograph      )

BLOC       CO                    (E      ,      PTYPE,diploma)

BLOC       CO                    (E      ,      PTYPE,

BLOC       CO                    (E      ,      PTYPE,

BLOC       CO                    (E      ,      PTYPE,Arbeitspapiere der
```

Für die Ermittlung und Anzeige der Publikationsinformationen der Artikel ergibt sich folgende SELECT-Abfrage:

```
                  L            L                ELECT publicationYear,
publicationTitle,
publicationAbstract,
publicationP      F,
articlePTitle,
articlePAuthors,
articlePT      pe,
articleP      olume,
articleP      ssue,
articlePChapter,
articlePPages,
```

```
        articlePPublisher,
        articlePPubLocation
FRO          articleTable
             ERE  publication      d             publication;
```

Die Anzeige der bibliographischen Informationen soll angelehnt an die Zitierregeln der American Psychological Association [APA 1987] erfolgen, die weltweit als einer unter vielen Standards anerkannt sind. Eine der wichtigsten APA-Zitierregeln ist folgende:

Anmerkung:
Wenn die übergeordnete Publikation, in welcher der Artikel erschienen ist, eine Monographie ist, so wird die Seitenangabe im Format »(pp. seitenangabe)« dargestellt. Ist die übergeordnete Publikation ein Periodikum, so wird die Seitenangabe ohne Klammern und ohne »pp.« angezeigt.

Zu beachten ist außerdem, daß nicht zu jedem Artikel alle in `articleTable` enthaltenen Attribute mit Werten gefüllt sind. So ist es beispielsweise möglich, daß zu einem Artikel keine Seitenangaben oder Verlagsinformationen abgespeichert wurden. Sind zu einem Artikel bestimmte Informationen nicht verfügbar, würden diese in der HTML-Anzeige als »NULL« dargestellt. Um dies zu verhindern, muß wiederum mit den vom Web DataBlade zur Verfügung gestellten Prozeßfunktionen gearbeitet werden.

Um eventuelle NULL-Rückgaben besser verarbeiten zu können, bekommen sie den Wert »NIX« zugewiesen:

```
                                AR        A       E                          _              LL
```

Zuerst wird das Veröffentlichungsjahr in Klammern angezeigt, danach der Titel (zur Hervorhebung in Fettdruck):

```
(           )            BR
     B                            B                  BR
```

Danach folgt der Titel der übergeordneten Publikation. Die Autoren der übergeordneten Publikation werden (soweit vorhanden) dem Titel der übergeordneten Publikation vorangestellt. Beiden Information wird ein »In:« vorangestellt.

```
     n            (      F,      (E      ,           ,                 ),,                ,      )
```

Danach werden (soweit vorhanden) die Volume-, Issue- und Kapitelangaben ausgegeben; die Issue-Angabe in Klammern:

```
     (      F,      (E      ,           ,                 ),,                ,      )
     (      F,      (E      ,           ,                 ),,    (                ),     )
     (      F,      (E      ,                 ,               ),,                ,      )
```

Als nächstes erfolgt die Angabe der Seitennummern. Sie muß, wie schon weiter oben beschrieben, je nach Publikationstyp der übergeordneten Publikation speziell formatiert werden. Zur Bestimmung, von welchem Typ die übergeordnete Publikation stammt, wird der Inhalt der Rückgabevariable `$7` (`articlePType`) genutzt.

```
     (      F,      (E      ,           ,monograph      ),      (      F,      (E      ,
```

Jede Eventualität kann aufgefangen werden:

- `publicationPType` = »monography«
- `publicationPType` = »periodical«
- `publicationPPages` = nicht vorhanden

Danach wird der Verlag mit vorangestelltem Verlagsort angezeigt (natürlich nur, soweit vorhanden):

```
     (      F,      (E      ,           ,                 ),,
     (      F,      (E      ,           ,                 ),      (to appear)          ,
          R
```

Als nächstes soll in dem Falle, daß ein Abstract vorhanden ist, dieser angezeigt werden. Zur besseren Übersichtlichkeit soll über dem Abstract eine Überschrift »Abstract:« erscheinen:

```
     (      F,      (E      ,           ,                 ),,                                Abstract
```

Ist eine PDF-Version des Artikel verfügbar, sollte diese (soweit vorhanden) dem Anwender zum Download angeboten werden. Die obige `SELECT`-Anfrage gibt in der Rückgabevariable `$4` (`publicationPDF`) die Large-Object-ID des gespeicherten PDF-Files zurück. Die LO-ID identifiziert jedes in

einer Illustra-Datenbank gespeicherte Large-Object. Mit dessen Hilfe kann das Dokument durch einen Hyperlink zum Download angeboten werden. Dabei ist zu beachten, daß der Hyperlink auch eine Information über den MIME-Type der Datei (`application/pdf`) enthält, da ansonsten der Web-Browser des Anwenders nicht weiß, wie er die ankommende Datei verarbeiten soll. Verfügt der Anwender nicht über den Acrobat Reader oder das Acrobat Plug-In für Web-Browser, die zum Anzeigen und Drucken eines PDF-Dokumentes nötig sind, soll er auch diese auf seinen Rechner herunterladen können:

```
(      F,      (E      ,            ,                  ),,             e serve a P      F version o
REF                        EB_      O      E      LO                                       t      peOb
          ou dont      t have Acrobat Reader on      our machine,      ou can do      nload it
http                                         adobe      com      prodindex      acrobat      readste

                                    L
```

Der für jede Publikationsart spezielle Anzeigeteil ist hiermit abgeschlossen. Die folgenden Schritte (Ermittlung und Anzeige der Aktivitäten, denen die Publikation zugewiesen wurde und die Ermittlung und Anzeige der Aktivitäten, zu denen eine inhaltliche Verknüpfung besteht) müssen nur einmal implementiert werden und können für alle Publikationsarten genutzt werden.

4.4.1.3.1.3 Ermittlung der durch Zuweisung verknüpften Aktivitäten

Der Anwender soll in dem Fall, daß die Publikation einer Aktivität zugeordnet wurde, den Kurznamen dieser Aktivität angezeigt bekommen. Dieser Name soll als Hyperlink ausgelegt werden, der bei Aktivierung ein neues Browser-Fenster öffnet, welches alle Publikationen anzeigt, die dieser Aktivität zugeordnet wurden. Diese neue Referenzanzeige geschieht über die Application-Page »articleResult«, die in einem späteren Abschnitt beschrieben wird.

Es ist somit notwendig, den Aktivitätskurznamen (`activityShortName`) der jeweiligen Aktivität zu ermitteln. Dabei ist zu beachten, daß in dem Fall, wenn die Publikation keiner Aktivität zugeordnet wurde, kein leerer Hinweis ausgegeben wird. Es handelt sich somit wiederum um zwei SQL-Routinen: Eine zur Ermittlung, ob die Publikation überhaupt einer Aktivität zugeordnet wurde, und eine, um den Namen der jeweiligen Aktivität zu ermitteln.

```
                              L           L                ELECT CO            T (      )
        FRO        ne      PublicationActivit      Table
                         ERE  publication      d            publication;
                                    L
```

Gibt diese Anfrage einen Wert größer `0` zurück, ist eine entsprechende Zuordnung vorhanden und der nächste `MISQL`-Block kann ausgeführt werden:

```
                    BLOC      CO                         (      ,      ,            )
  This publication belongs to the      ollo      ing activities
      TABLE
                                 L           L                ELECT a      activit            ho
          FRO        acitivit      Table a,
                ne      PublicationActivit      Table n
                      ERE  n      publication      d            publication
          A              a      activit            d      n      activit            d;

      T                A      REF                        EB_      O      E                  val

                                    L
          TABLE
      ith a mouse clic      on the activit      name      ou receilvicatbinstaossigned atol pub
  this activit
                           BLOC
```

4.4.1.3.1.4 Ermittlung der durch inhaltliche Relation verknüpften Aktivitäten

Analog zur vorstehend beschriebenen Anzeige soll eine Darstellung aller Aktivitäten erfolgen, zu denen eine inhaltliche Verknüpfung besteht. Dabei muß zusätzlich beachtet werden, daß nur solche Aktivitäten angezeigt werden, denen mindestens eine Publikation zugewiesen wurde, da ansonsten die geöffnete Referenzanzeige »articleResult« leer bleibt.

Zuerst erfolgt wieder die Prüfung, ob es überhaupt inhaltliche Verknüpfungen zu einer Aktivität gibt, der mindestens eine Publikation zugeordnet wurde:

```
                           L             L                 ELECT CO             T (      )
      FRO         ne      Publication      e           ordTable
                  ERE  publication      d           publication
      A                   e           ord           (      ELECT                     T
                     FRO          activit      Table a,
                            ne      PublicationActivit      Table n
                                 ERE  a      activit           d      n      activit            d)
                                   L
```

Sind entsprechende Einträge vorhanden (Rückgabewert>0), so kann die nächste Anfrage ausgeführt werden:

```
                 BLOC       CO                           (      ,      ,         )
  There are clues, that this publication has semantic relations to the      ollo      ing activities
        TABLE
                           L             L                 ELECT                     T
         FRO         ne      Publication      e           ordTable
                     ERE  publication      d           publication
         A                   e           ord           (      ELECT                     T
                        FRO          activit      Table a,
                               ne      PublicationActivit      Table n
                                    ERE  a      activit           d      n      activit

        T                   A       REF                     EB_      O      E                val

                                      L
              TABLE
        ith a mouse clic       on the activit       name       ou receive a list of publications assigned to
  this activit
                          BLOC
```

4.4.1.3.1.5 Schließen des Detailfensters

Da zur Dokumenteneinzelanzeige ein neues Browser-Fenster geöffnet wurde, soll dieses Fenster, wenn es nicht mehr benötigt wird, geschlossen werden können. Dies kann zwar auch über das Icon am linken oberen Bildrand erledigt werden. Trotzdem soll der Anwender eine klare Meldung erhalten, daß er dieses Fenster im Bedarfsfall wieder schließen kann.

Das Schließen des Fensters wird über einen HTML-Form-Button realisiert, der eine JavaScript-Funktion auslöst:

```
     FOR
     CE       TER            P                        onClic    T TYPE            button            close();
           FOR
```

Die Dokumenteneinzelanzeige zeigt sich im Web-Browser des Anwenders folgendermaßen:

Abbildung 9: Detailanzeige »publicationDetail«

4.4.1.4 Referenzenanzeige verknüpfter Dokumente

Wenn der Anwender in der Dokumenteneinzelanzeige »publicationDetail« Hinweise darauf bekommt, daß die ausgewählte Publikation bestimmten IPSI-Aktivitäten zugeordnet wurde und/oder inhaltliche Verknüpfungen zu bestimmten IPSI-Aktivitäten bestehen, so sollte es eine Möglichkeit geben, alle Dokumente, die der jeweiligen Aktivität fest zugeordnet wurden, in Form einer Referenzenliste, ähnlich der Darstellung durch »fulltextResult«, anzeigen zu lassen. Von dieser Referenzenliste sollte der Anwender wieder auf die Dokumenteneinzelanzeige »publicationDetail« springen können, um sich die in der Referenzenliste dargestellten Dokumente im Detail betrachten zu können. Um dem Anwender einen möglichst breiten Überblick über eine bestimmte Aktivität zu geben, sollten in der Referenzenliste wirklich alle ihr zugeordneten Publikationen erscheinen; auch wenn der Anwender in seiner ursprünglichen Suche angegeben hat, nur bestimmte Publikationsarten sehen zu wollen.

4.4.1.4.1 Implementierung der aktivitätsbezogenen Referenzenliste (activityResult)

Damit der Anwender sich bei der aktivitätsbezogenen Referenzenliste nicht an eine neue Formatierung der Referenzen gewöhnen muß, soll sie vom Format her der Suchergebnisanzeige »fulltextResult« entsprechen. Dies birgt außerdem den Vorteil, daß für die Implementierung von »activityResult« der Quellcode von »fulltextResult« nur geringfügig verändert werden muß. Aus diesem Grund werden im Folgenden nur die Änderungen dargestellt, die nötig sind, um aus einer Kopie von »fulltextResult« die aktivitätsbezogene Referenzenliste »activityResult« zu implementieren.

4.4.1.4.1.1 Entfernen der Tabelle zur Anzeige der Suchanfrage

Da in »activityResult« keine vom Anwender formulierte Suchanfrage erfolgt, kann folglicherweise auf die Anzeige einer solchen verzichtet werden. Somit fallen auch alle Suchanfragen weg, die für die Anzeige der Suchdaten zuständig sind.

4.4.1.4.1.2 Entfernen von Bedingungsblöcken (MIBLOCK)

In »activityResult« sollen alle Publikationen zu einer Aktivität angezeigt werden. Somit können auch die MIBLOCK-Rahmen wegfallen, die in »fulltextResult« dafür sorgen, daß nur die Suchanfragen der Publikationsarten ausgeführt werden, die vom Anwender zur Rückgabe bestimmt wurden.

4.4.1.4.1.3 Änderung der MISQL-Blöcke zur Dokumentenermittlung und Anzeige

In »activityResult« findet kein Text-Retrieval mehr statt. Es wird nur noch nach Dokumenten einer bestimmten Aktivität gefragt. Deswegen kann der Teil der beiden `SELECT`-Anweisungen (`SELECT COUNT` zur Prüfung, ob zur Anfrage überhaupt Dokumente vorhanden sind und `SELECT` zur Ermittlung der Referenzdaten), der das PLS Text DataBlade anweist, eine Textsuche durchzuführen, wegfallen.

Auch die Sucheinschränkungen bezogen auf Veröffentlichungsjahre und Autoren findet nicht mehr statt und die entsprechenden Anweisungen müssen aus den `SELECT`-Statements entfernt werden.

Hinzugefügt werden muß eine Sucheinschränkung, die dazu führt, daß nur Dokumente einer bestimmten Aktivität gefunden werden. Der entsprechende Aktivätskurzname ist in der Variable `$keyword` enthalten, die von der Application-Page »publicationDetail« übermittelt wurde.

Die `SELECT COUNT`-Anweisung präsentiert sich (auf den `MISQL`-Block für die GMD-Publikationen bezogen) nach der Änderung folgendermaßen:

```
                        L              L                  ELECT CO          T (      )
    FRO        gmdReportTable g,
          ne       PublicationActivit      Table n,
          activit      Table a
               ERE  (g      publication      d      n      publication      d)
    A              (a      activit           d      n      activit           d)
    A              (a      activit           hort      ame                          e
```

In der folgenden Formatierungsanweisung kann der Teil weggelassen werden, der für die Anzeige der Überschrift »No GMD publications retrieved« zuständig war, da der Anwender nicht mehr gezielt nach bestimmten Publikationsarten sucht und daher keine Rückmeldung mehr benötigt, wenn zu einer bestimmten Publikationsart keine Dokumente gefunden wurden. Die Formatierungsanweisung zeigt sich wie folgt:

```
  (      F,      (E      ,           ,      ),           ,

                                     L
```

Wenn Dokumente gefunden wurden (Rückgabewert>0) kann der folgende MISQL-Block ausgeführt werden, der auch entsprechend der neuen Anforderungen verändert wurde:

```
              BLOC       CO                      (      ,      ,           )
                      L              L                  ELECT                T          CT
            g      publicationYear,
            g      publicationTitle,
            g      publicationT      pe,
            g      gmdReport      olume,
            g      gmdReportPublisher,
            g      gmdReportPubLocation,
            n      publication      d,
            p      personnelFirst      ame,
            p      personnel      ur      ame
      FRO        gmdReportTable g,
            ne       PublicationActivit      Table n,
            ne       PublicationPersonnelTable nP,
            activit      Table a,
            personnelTable p
                 ERE  (g      publication      d      n      publication      d)
```

```
A              (a      activit          d      n      activit          d)
A              (p      personnel     d      nP      personnel     d)
A              (nP      publication      d      n      publication      d)
A              (a      activit          hort      ame                      e
OR      ER BY
      g      publicationYear desc;
```

Die Formatierungsanweisungen können unverändert stehen bleiben, da die gesuchten Attribute die gleichen wie aus »fulltextResult« sind.

Um dem Anwender wieder die Möglichkeit zu geben, die referenzierten Dokumente im Detail zu betrachten, ist der Publikationstitel wieder als Hyperlink ausgelegt, der die Application-Page »publicationDetail« wiederholt aufruft. Theoretisch könnte man zur Detailanzeige wieder ein neues Fenster öffnen. Um den Bildschirm des Anwenders jedoch nicht mit unzähligen Browser-Fenstern zu überfüllen und die Übersichtlichkeit, die durch den Gebrauch mehrerer Fenster erreicht werden soll, nicht zu zerstören, wird für den erneuten Aufruf von »publicationDetail« das schon vorher für die Detailanzeige geöffnete Fenster benutzt. Sollte der Anwender das dort vorher angezeigte Dokument wieder sehen wollen, braucht er entweder nur den »Back«-Button seines Browsers zu drücken oder in der Suchergebnisanzeige »fulltextResult« das gewünschte Dokument erneut anzuwählen.

4.4.1.4.1.4 Anzeige des Aktivitätsnamens

Um dem Anwender noch einen Hinweis darauf zu geben, zu welcher Aktivität die angezeigten Dokumente gehören, wird über den Publikationsreferenzen eine Überschrift angezeigt, die den Aktivitätskurznamen enthält:

```
All publications o       the activit                                    AR
```

Die aktivitätsbezogene Referenzenliste »activityResult« präsentiert sich im Web-Browser des Anwenders folgendermaßen:

Abbildung 10: Aktivitätsbezogene Referenzenliste »activityResult«

4.4.2 Hinzufügen von Dokumenten

Wie schon in der Konzeption angesprochen, sollte das Hinzufügen von neuen Dokumenten zum Bestand so unkompliziert sein, daß dies auch von Personen durchgeführt werden kann, die nicht mit dem Illustra-

DBMS vertraut sind. Diese Vorgabe führt letztendlich dazu, dem Anwender für diese Verrichtungen ebenfalls eine graphische Benutzeroberfläche zur Verfügung zu stellen. Da die Suche im Dokumentenbestand schon über eine HTML-basierte Oberfläche geregelt wird, liegt der Schluß nahe, die Oberfläche zur Pflege des Dokumentenbestandes ebenfalls HTML-basiert zu gestalten. Dies hat außerdem den Vorteil, daß die Bestandspflege standort- und plattformunabhängig erfolgen kann, was sich folgendermaßen auswirkt:

- Wenn die Bestandspflege nur einigen bestimmten Mitarbeitern anvertraut wird, können diese die Pflege des Systems von jedem Rechner aus vornehmen, der an das GMD-Netzwerk angeschlossen ist und über einen Web-Browser verfügt.
- In dem Falle, daß die Pflege des Dokumentenbestandes nicht von speziell damit beauftragten Mitarbeitern geregelt wird, sondern jeder Autor seine Publikation selbst zum Bestand hinzufügt, besteht für diesen ebenfalls weitestgehende Standort- und Plattformunabhängigkeit. So ist es beispielsweise denkbar, daß ein Autor einen großen Teil der Dokumentenverfassung an seinem Heimrechner erledigt. Da das Publikations-Management-System speziell für den Einsatz im World Wide Web vorgesehen ist, kann der Autor das Hinzufügen seines Dokumentes zum System auch von seinem Heimrechner aus vornehmen (solange er einen Zugang zum Internet und einen Web-Browser besitzt).

4.4.2.1 Benötigte Arbeitsschritte für das Hinzufügen von Dokumenten

Bevor mit der Implementierung der Hinzufügungsfunktionen begonnen werden kann, muß festgestellt werden, welche Arbeitsschritte notwendig sind, bis ein Dokument vollständig in das Publikations-Management-System integriert ist.

- **Speicherung der eigentlichen Publikationsdaten**
  Der wohl wichtigste Schritt ist die Speicherung der eigentlichen Publikationsdaten in den Publikationstabellen (`articleTable`, `gmdReportTable`, `monographyTable`, `diplomaTable`). Damit der Anwender bei der Suche korrekte Suchergebnisse bekommt, ist es von essentieller Wichtigkeit, daß alle Dokumente in den richtigen Tabellen gespeichert sind. Beispielsweise muß eine »Diplomarbeit«, die im System als »Arbeitspapier der GMD« geführt (und dem Anwender als Suchergebnis auch so angezeigt wird), als ein absolut irrelevantes Suchergebnis gewertet werden.
- **Speicherung der Autorenschaft**
  Da trotz der Fortschritte in Richtung Textverarbeitung ein Computer Texte nicht selbständig schreiben kann, muß es zu jeder Publikation zumindest einen Autoren geben. Diese Autorenschaft muß in der Tabelle `newPublicationPersonnelTable` gespeichert werden. Da diese Tabelle auf die Relation `personnelTable` referenziert, ist es eine weitere Voraussetzung, daß jeder Autor in der Tabelle `personnelTable` verfügbar ist. Ist ein Autor noch nicht in `personnelTable` gespeichert, so muß er dort eingefügt werden, bevor die Autorenschaft in `newPublicationPersonnelTable` festgelegt werden kann.
- **Speicherung der Aktivitätszugehörigkeit**
  Wurde eine Publikation im Rahmen einer bestimmten IPSI-Aktivität verfaßt, so muß diese Aktivitätszugehörigkeit in der Tabelle `newPublicationActivityTable` festgehalten werden. Auch hier gilt: Da `newPublicationActivityTable` auf die Relation `activityTable` referenziert, ist es notwendig, daß jede Aktivität, mit der die Publikation verknüpft werden soll, dort vorhanden ist. Ist dies nicht der Fall, muß diese (wahrscheinlich neue) Aktivität erst zu `activityTable` hinzugefügt werden.
- **Feststellung und Speicherung von inhaltlichen Verknüpfungen**
  Damit das System den Anwender auf eventuelle inhaltliche Verknüpfung zwischen Publikationen und Aktivitäten hinweisen kann, ist es notwendig, daß jedes Dokument bei der Einfügung ins System nach

bestimmten Schlüsselbegriffen durchsucht wird, und daß eventuelle Schlüsselwortvorkommen in der Tabelle `newPublicationKeywordTable` festgehalten werden.

In einem Flußdiagramm stellt sich die Einfügung eines Dokumentes somit folgendermaßen dar:

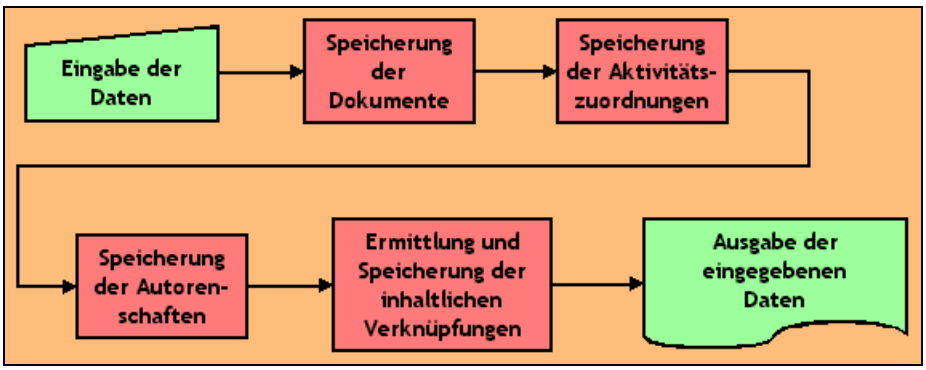

Abbildung 11: Ablauf der Dokumenteneinfügung

- **Automatisierung von Arbeitsschritten**
  Das Einfügen von Datensätzen in Datenbanktabellen in einem RDBMS oder ORDBMS erfolgt über die Datenmanipulationssprache SQL. Da von einem nicht mit RDBMS bzw. ORDBMS vertrauten Benutzer nicht verlangt werden kann, diese Sprache zu beherrschen, müssen die eigentlichen Einfügearbeiten durch das System selbsttätig ausgeführt werden. Die Voraussetzung dafür, daß das System diese Arbeitsschritte von sich aus erledigen kann, ist, daß ihm die einzufügenden Daten vom Anwender zur Verfügung gestellt werden.
  Vorstellbar wäre folgende Lösung:

  1. Der Anwender gibt die einzufügenden Daten in ihm dafür zur Verfügung stehende Formulare ein.
  2. Nach der Übertragung dieser Daten führt das System alle o. g. Arbeiten selbständig aus.

Es müssen somit zwei unterschiedliche Anwendungskomponenten implementiert werden:

1. Eingabeformulare für den Nutzer
2. Mechanismen, um die Daten in die entsprechenden Datenbanktabellen einzufügen.

4.4.2.2 Implementierung der Systempflegefunktionen

4.4.2.2.1 Eingabeformulare

Jede der Publikationstabellen (`articleTable`, `gmdReportTable`, `monographyTable`, `diploma-Table`) hat, abgesehen von den Attributen, die eine Klassenbildung in `newPublicationTable` möglich machen, eigene, für sie spezielle Attribute. Es scheint somit angebracht, dem Anwender für jede Publikationsart ein spezielles Eingabeformular zur Verfügung zu stellen. Folgende Formulare werden benötigt (die Namen der dafür vorgesehenen Application-Pages sind in Klammern gesetzt):

- Einfügen von Artikeln (articleInsert)
- Einfügen von GMD-Publikationen (gmdReportInsert)
- Einfügen von Monographien (monographyInsert)
- Einfügen von Diplomarbeiten (diplomaInsert)

Für den Fall, daß ein Publikationsautor noch nicht in der Tabelle `personnelTable` vorhanden ist, oder, daß eine Aktivität, der die Publikation zugeordnet werden soll, in der Tabelle `activityTable` noch

nicht vorhanden ist, müssen dem Anwender ebenfalls entsprechende Eingabeformulare zur Verfügung stehen:

- Einfügen von Personen (personalInsert)
- Einfügen von Aktivitäten (activityInsert)

4.4.2.2.1.1 Einfügen von Publikationen (pubInsert)

Möchte der Anwender eine neue Publikation zum Bestand hinzufügen, so gelangt er zuerst auf die Application-Page »pubInsert«. Dort kann er auswählen, von welchem Publikationstyp das neue Dokument stammt. Die Auswahl wird dabei über SUBMIT-Buttons vorgenommen, die das entsprechende Eingabeformular aufrufen.

```
                              hich publication t      pe do       ou       ant to insert
TABLE
T                     FOR             ET      O                   PO      T       ACT      O
            P        T TYPE                                     E                     A       E
            P        T TYPE                        B             T                AL      E          Art

T                     FOR             ET      O                   PO      T       ACT      O
            P        T TYPE                                     E                     A       E
            P        T TYPE                        B             T                AL      E

T                     FOR             ET      O                   PO      T       ACT      O
            P        T TYPE                                     E                     A       E
            P        T TYPE                        B             T                AL      E

T                     FOR             ET      O                   PO      T       ACT      O
            P        T TYPE                                     E                     A       E
            P        T TYPE                        B             T                AL      E
TR
      TABLE
```

Die Auswahlseite »pubInsert« wird im Web-Browser des Anwenders folgendermaßen angezeigt:

Abbildung 12: Publikationseingabe »pubInsert«

4.4.2.2.1.2 Einfügen von GMD-Publikationen (gmdReportInsert)

Da die vier benötigten Formulare zur Eingabe der Publikationsarten vom Aufbau her fast analog sind, wird im folgenden Abschnitt (stellvertretend für alle anderen) nur das Formular zum Einfügen einer GMD-Publikation (gmdReportInsert) explizit beschrieben.

4.4.2.2.1.2.1 Formulartyp

Über das Eingabeformular müssen nicht nur herkömmliche Formulardaten (Textstrings und Zahlenwerte), sondern auch Dateien folgenden Formates übertragen werden:

- **Textdateien**
  Abstract und Volltext der Publikation werden im ASCII-Format benötigt, damit sie vom PLS Text DataBlade invertiert werden können.

- **Binärdateien**
  Die spätere Download-Datei muß im PDF-Format gespeichert werden.

Zur Übertragung dieser Daten wird ein spezieller Formulartyp benötigt. Der HTML-Standard sieht solche Funktionalitäten noch nicht vor. Die Firma Netscape hat jedoch einen neuen Formulartyp entwickelt. Er nennt sich »multipart/form-data«.

```
FOR E CTYPE multipartACT O orm data AR ET O EB_
```

Mit diesem neuen Formulartyp ist es möglich,

- ASCII- und Binärdateien über die POST-Methode an CGI-Programme
- ASCII-Dateien über die MAILTO-Methode an Internet-Mail-Teilnehmer

zu versenden.

Voraussetzung für die Nutzung dieser Möglichkeiten ist allerdings, daß der Anwender über den Web-Browser *Netscape Navigator* (ab Version 2.0) verfügt. Netscape hat den Formulartyp »multipart/form-data« dem W3-Konsortium als neuen Bestandteil der HTML-Spezifikation vorgeschlagen. Sollte dieser Vorschlag angenommen werden, kann man davon ausgehen, daß in Zukunft auch andere Web-Browser diesen Formulartyp nutzen können.

4.4.2.2.1.2.2 Aufbau des Eingabeformulars

Für das Einfügen einer GMD-Publikation in die Tabelle `gmdReportTable` werden folgende Daten benötigt:

- Publikationsnummer (`publicationId`)
- Veröffentlichungsjahr (`publicationYear`)
- Titel der Publikation (`publicationTitle`)
- Abstract (`publicationAbstract`)
- Volltext (`publicationFullText`)
- Publikation im PDF-Format (`publicationPDF`)
- Publikationstyp (`publicationType`)
- lfd. Nummer (`gmdReportVolume`)
- Verlag (`gmdReportPublisher`)
- Verlagsort (`gmdReportPubLocation`)

Daraus ergeben sich folgende HTML-Eingabefelder:

- Die Publikationsnummer ist dem Anwender nicht bekannt. Sie muß durch das System ermittelt werden. Dies soll im Hintergrund, also für den Anwender unsichtbar, geschehen.

  Um die richtige `publicationId` für die neue GMD-Publikation zu vergeben, wird zuerst eine `SELECT MAX`-Anfrage an die Tabelle `newPublicationTable` gestellt, die den derzeit höchsten Wert des Attributes `publicationId` ermittelt. Zu diesem Wert wird dann `1` hinzuaddiert. Das Ergebnis (die neue `publicationId`) wird in ein verstecktes Eingabefeld geschrieben:

```
L L ELECT A (publication
FRO ne PublicationTable;
```

```
                P       T TYPE                                          E                       A       E

                                                L
```

- Die Werte für `publicationYear`, `publicationTitle` und `gmdReportVolume` werden über Texteingabefelder eingegeben:

```
B       Publication Year                        B
                        AR      A       E               YEAR
                        AR                      P    AL T TYPE   E    YEAR      TE       T

B       Title                   B
                        AR      A       E               T       TLE
                        AR                      P       T TYPE              TE       T

B             olume                     B
                        AR      A       E                       OL
                        AR                      P   AL  T TYPE   E          OL  TE       T
```

- Die Autoren werden über eine durch `MISQL` generierte `SELECT LIST` ausgewählt. Dabei wird die `personnelId` als Weitergabewert für den Einfügemechanismus, der Vor- und Nachname zur Bildschirmdarstellung genutzt. Zusätzlich wird noch ein Hinweis eingefügt, daß, falls ein Autor noch nicht in der Liste angezeigt wird, dieser eingespeichert werden kann. Dieser Hinweis ist als Hyperlink ausgelegt, der bei Aktivierung das Eingabeformular »personalInsert« aufruft, mittels dessen ein neuer Autor zur Tabelle `personnelTable` hinzugefügt werden kann.

```
B       Author(s)                       B
       ELECT        A       E               A       T       OR                      LT       PLE
                                    L               L                   ELECT                       T
              personnel       d,
              personnel       ur      ame,
              personnelFirst          ame
       FRO          personnelTable
       OR       ER BY
              personnel       ur      ame;
OPT       O             AL      E                                                           ,
                                    L
             ELECT              BR
nsert a       A       REF                                             AR                      EB_       (
```

- Auch für die Auswahl der IPSI-Aktivität, mit der die Publikation verknüpft werden soll, wird über eine durch `MISQL` generierte `SELECT LIST` geregelt. Dabei wird die `activityId` als Weitergabewert und der Kurzname der Aktivität zur Bildschirmdarstellung genutzt. Zu beachten dabei ist, daß nur die Aktivitäten angezeigt werden sollen, die den Status »current« besitzen, da wohl keine Publikationen zu einem schon lange abgeschlossenen Projekt verfaßt werden. Zusätzlich wird noch ein Hinweis eingefügt, daß, falls eine gewünschte Aktivität nicht in der Liste angezeigt wird, ein neue Aktivität eingefügt werden kann. Dieser Hinweis ist ebenfalls als Hyperlink ausgelegt, der bei Aktivierung das Eingabeformular »activityInsert« aufruft, mittels dessen eine neue Aktivität zur Tabelle `activityTable` hinzugefügt werden kann.

```
B       Related activities                      B
       ELECT        A       E               ACT                     LT       PLE                    E
                                    L               L                   ELECT                       T
              activit             d,
              activit             hort      ame
       FRO          activit       Table
                    ERE  activit            tatus           current
       OR       ER BY
              activit             hort      ame;
OPT       O             AL      E
                                    L
             ELECT              BR
nsert a       A       REF                                             activit                 EB_       (
```

- Der Publikationstyp wird auch über eine `SELECT LIST` ausgewählt. Diese kann jedoch nicht über `MISQL` generiert werden, da man nicht sicher sein kann, daß schon GMD-Publikationen aller Ausprägungen (GMD-Studie, GMD-Bericht, Arbeitspapiere der GMD) im System vorhanden sind. Außerdem sind in dem Textstring »Arbeitspapiere der GMD« Leerzeichen enthalten, was bei der Verarbeitung der Daten durch den Einfügemechanismus zu Problemen führen würde. Die Leerzeichen werden daher durch Unterstriche ersetzt:

```
B      Publication t      pe                B
      ELECT       A      E            TYPE
OPT     O             AL      E          Arbeitspapiere_der_                    Arbe
OPT     O             AL      E                                    tudien
OPT     O             AL      E                                Berichte
          ELECT
```

- Verlag und Verlagsort werden erst in der Einfügeroutine bestimmt, da es hier nur zwei Möglichkeiten gibt: Arbeitspapiere und GMD-Studien werden von der GMD in Sankt Augustin, GMD-Berichte vom Verlag Oldenbourg in München verlegt.

  Bei den Einfügeformularen »articleInsert« und »monographyInsert« werden Verlag und Verlagsort vom Anwender in Textfelder eingegeben, da es hier unzählige Möglichkeiten gibt.

- Die drei Dateien (Abstract und Volltext im ASCII-Format und die Publikation im PDF-Format) werden über speziell dafür vorgesehene `FILE`-Felder eingegeben. Der Anwender sieht dann neben jedem Eingabefeld einen Button »Browse«. Drückt er diesen, dann öffnet sich ein Auswahlfenster, in dem er die entsprechende Datei auf seinem Dateisystem auswählen kann.

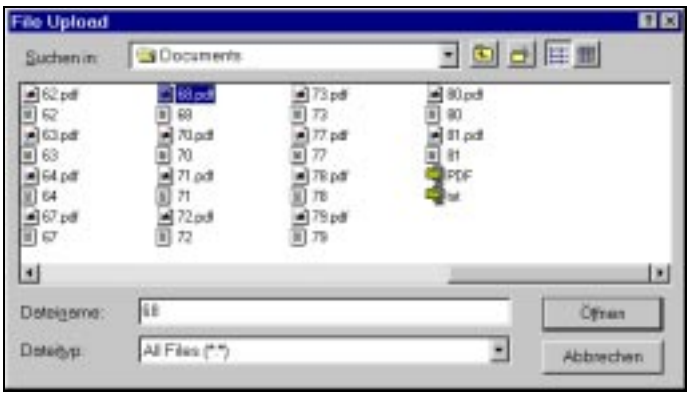

Abbildung 13: Dateiauswahlfenster

Hat er die Datei mit »Öffnen« ausgewählt, schließt sich das Fenster und der Pfadname der lokalen Datei erscheint im Eingabefeld.

```
B      Publication in P      F       ormat            B
           P      T       A      E            P      F                          E

B      Publication in A      C                  ormat            B
           P      T       A      E            A      C                                  E

B      Abstract o       publication                  B
           P      T       A      E            AB      A      C                        E
```

- Zum Schluß folgen noch das versteckte Eingabefeld, mit der Angabe der Application-Page, an welche die Daten übergeben werden, der SUBMIT- und der RESET-Button:

```
           P      T TYPE                                    E                  A      E
           P      T TYPE                          B          T          AL      E
           P      T TYPE             RE      ET
FOR
```

Die vom Anwender eingegebenen Daten werden nun an die Application-Page »publicationProc« übermittelt, die einen Teil des Einfügemechanismus darstellt.

Das Einfügeformular für die GMD-Publikationen zeigt sich im Web-Browser des Anwenders folgendermaßen:

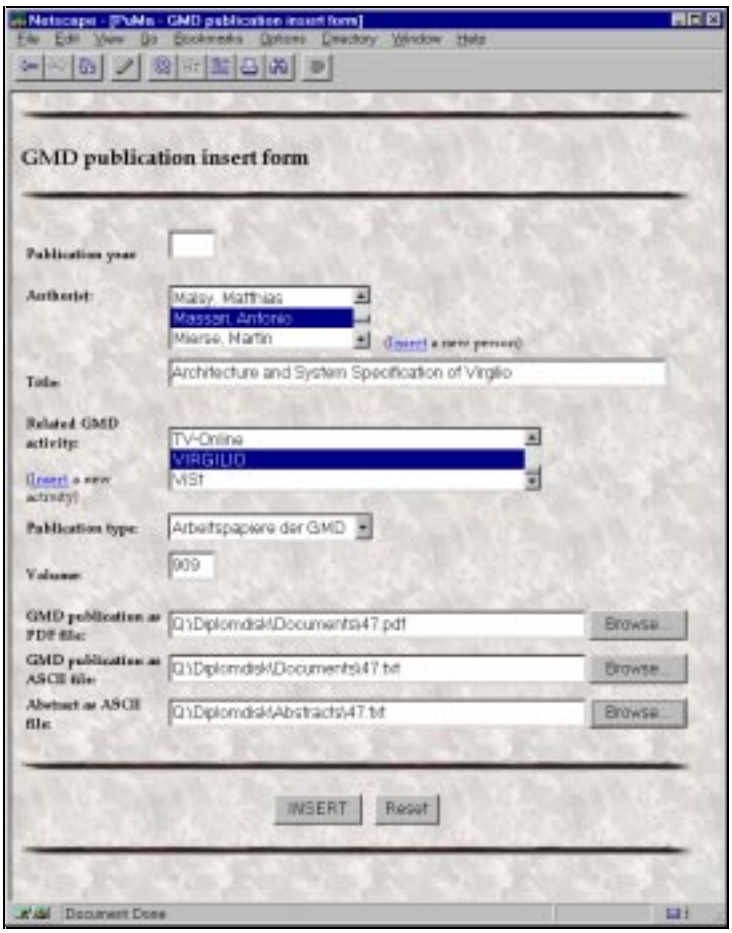

Abbildung 14: Eingabeformular für GMD-Publikationen »gmdReportInsert«

4.4.2.2.1.3 Einfügen einer Person in personnelTable bzw. einer Aktivität in activityTable (personalInsert bzw. activityInsert)

Da sich die Eingabeformulare »personalInsert« und »activityInsert« eigentlich nur durch die Art der einzugebenden Daten von dem Eingabeformular »gmdReportInsert« unterscheiden, wird auf eine detaillierte Erläuterung dieser Application-Pages verzichtet. Erwähnenswert sind folgende Sachverhalte:

- Über das Formular »personalInsert« können lediglich externe Personen in die Tabelle personnelTable eingegeben werden. Auf die Implementierung eines spezielles Formulars für die Eingabe eines neuen GMD-Mitarbeiters wurde aus zwei Gründen verzichtet:
- Es handelt sich hier nur um eine prototypische Implementierung. Daher genügt es, die Einfügefunktionalität an einem einfachen Fall darzustellen.
- Der Bestand der GMD-internen Personen ist im Normalfall ständig aktuell, da das Publikations-Management-System in dieselbe Datenbank integriert ist, die für die WWW-Darstellung des IPSI zuständig ist. Für die WWW-Präsentation werden immer die aktuellsten Daten benötigt. Es kann daher davon ausgegangen werden, daß die Tabelle personnelGmdTable immer auf dem neuesten Stand ist.
- In das Eingabeformular »activityInsert« werden nur die für die Tabelle `activityTable` nötigsten Informationen (`activityId`, `activityLongName`, `activityShortName`, `activityStatus` und `activityType`) eingegeben. Für die anderen Attribute (`activityDescr`, `activityPartner`, `activityLogo` und `activitySmallLogo`) werden `NULL`-Werte eingesetzt. Auch hierfür zählt die Begründung, daß es sich nur um eine prototypische Implementierung handelt, bei der auf die

Eingabe der für das Publikations-Management-System nicht benötigten Attribute verzichtet werden kann.

Die Eingabedaten aus »personelInsert« bzw. »activityInsert« werden an die Application-Pages »personalProc« bzw. »activityProc« übermittelt.

Wie die beiden Eingabeformulare im Web-Browser dargestellt werden, zeigen die zwei folgende Abbildungen:

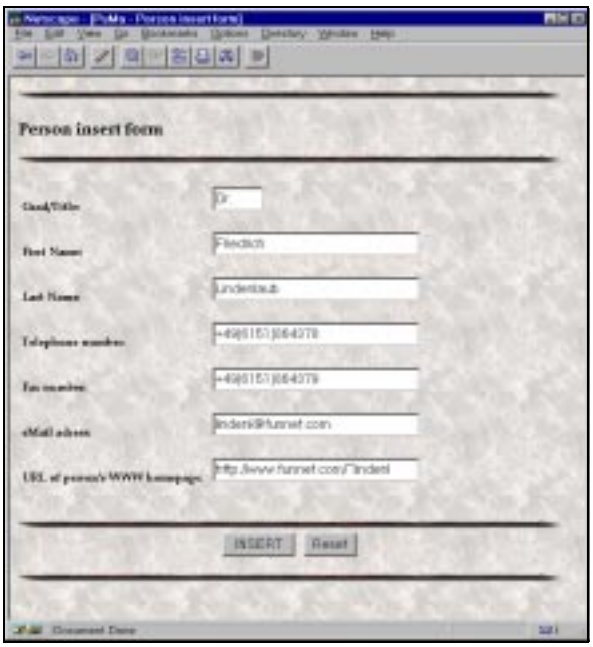

Abbildung 15: Eingabeformular für Personen »personalInsert«

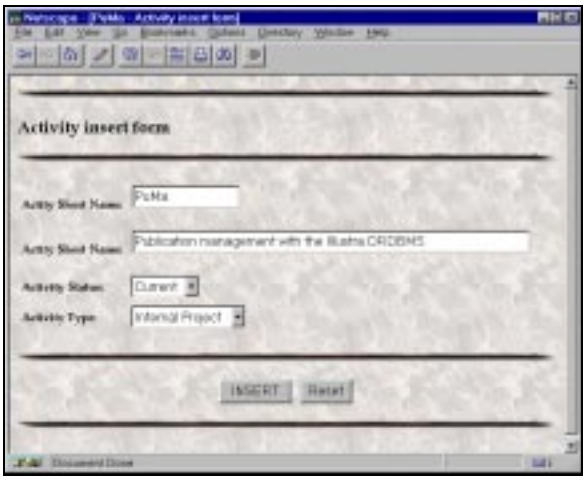

Abbildung 16: Eingabeformular für Aktivitäten »activityInsert«

4.4.2.2.2 Einfügemechanismen für Personen und Aktivitäten

4.4.2.2.2.1 Einfügemechanismus für Personen (personalProc)

Die vom Eingabeformular »personalInsert« übergebenen Variablenwerte werden in ein `INSERT`-Statement eingefügt:

```
                         L         L                     ERT         TO person
        (
                                ,
                               RA         ,
                           F    A    E    ,
                           L    A    E    ,
                           TEL      ,
                           FA          ,
                           E    A    L    ,
                               RL
        );
                                       L
```

Damit der Anwender eine Rückmeldung darüber bekommt, ob die neue Person tatsächlich ins System integriert wurde, wird gleich darauf ein `SELECT` ausgeführt, um den eben eingegebenen Datensatz wieder auszulesen:

```
A ne      person     ith the     ollo     ing pro     ile     as inserted
                         L         L                  ELECT
        FRO      personnelTable
                   ERE personnel     d                 ;

    TABLE
    T        Personal                     T               B                 TR
    T            rad      Title      T               B                 TR
    T        First     ame      T               B                 TR
    T        Last     ame      T               B                 TR
    T        Tel          T               B                 TR
    T        Fax      T               B                 TR
    T        e     ail      T               B                 TR
    T            RL      T               B                 TR

                                       L
        TABLE
```

Als letztes wird dem Anwender ein kleiner Button gezeigt, der ihm bei Knopfdruck wieder auf die Seite »pubInsert« zurückbringt. Da er von dort aus wieder auf das richtige Publikationseingabeformular springen muß, ist gewährleistet, daß die darin enthaltene Autorenauswahlliste (die ja durch ein `SELECT` generiert wird) die neu eingefügte Person enthält.

```
    CE     TER
                              AR
    FOR          ET     O            PO     T     ACT     O               E
               P     T TYPE                    E          A     E
               P     T TYPE            B          T          AL     E          Bac
        FOR
                                  AR
        CE     TER
```

Der Ergebnisbildschirm nach der Einfügung einer Person zeigt sich dem Anwender folgenermaßen:

Abbildung 17: Eingaberückmeldung »personalProc«

4.4.2.2.2.2 Einfügemechanismus für Aktivitäten (activityProc)

Zunächst erfolgt, analog zu »personalProc«, ein `INSERT`-Statement, welches die von »activityInsert« übergebenen Daten in die Datenbank einfügt.

Danach ist folgendes zu beachten:

Auch wenn die Wahrscheinlichkeit nicht sehr groß ist, daß in einer im System gespeicherten Publikation schon der Name einer damals noch nicht vorhandenen Aktivität auftaucht, muß dieser Fall bedacht werden. Es könnte beispielsweise sein, daß in einer Publikation auf ein in der Zukunft neu initiiertes Projekt hingewiesen wird. Somit besteht zwischen dieser Publikation und der nun neu eingefügten Aktivität eine inhaltliche Verknüpfung.

Dies bedeutet im Endeffekt, daß nach dem Einfügen einer neuen Aktivität geprüft werden muß, ob es im bisherigen Publikationsbestand Dokumente gibt, die eine inhaltliche Verknüpfung zu dieser aufweisen. Für jeden Textindex muß eine getrennte PLS-Suche vorgenommen werden. Das dafür benötigte `SELECT`-Statement kann dabei direkt in das `INSERT`-Statement integriert werden, welches ein `keyword`-Vorkommen (`keyword`=Kurzname der neuen Aktivität) in `newPublicationKeywordTable` einträgt. Als Beispiel folgt die Keyword-Suche für den `diplomaIndex`:

```
              L           L                         ERT          TO ne
          ELECT a     publication     d,          ACT     A     E
     FRO      articleTable a,
          PlsOidRan          uer     (     article     ndex     ,        ACT
              ERE  a           pls_oid     a     oid;
                   L
```

Nach abgeschlossener Einfügung aller Daten soll auch hier eine Rückmeldung für den Anwender erfolgen. Zunächst wieder, analog zu »personalProc«, das `SELECT`-Statement für die Tabellenanzeige der neu eingefügten Aktivität.

Danach soll, für den Fall, daß inhaltliche Verknüpfungen gefunden wurden, diese dem Anwender ebenfalls angezeigt werden. Die Publikationen sollen wieder, wie von »searchResult« und »activityResult« bereits gewöhnt, in Form von Literaturreferenzen angezeigt werden:

```
              L           L                  ELECT CO          T (      )
 FRO     ne     Publication     e           ordTable
          ERE     e           ord                  ACT     A     E     ;
                   L

          BLOC     CO                       (     ,     ,          )
```

L

Die dazu benötigten `SELECT`-Statements können, mit Ausnahme einiger kleiner Änderungen, fast unverändert aus »activityResult« übernommen werden:

- `FROM`-Klausel (`newPublicationActivityTable` zu `newPublicationKeywordTable`)
- `WHERE`-Klausel (`activity=$keyword` zu `keyword='$ACTNAME'`)

Auch hier bekommt der Anwender am Ende der Seite einen Button gezeigt, der ihn wieder nach »pubInsert« bringt.

Die folgende Abbildung zeigt den Ergebnisbildschirm nach der Einfügung einer neuen Aktivität:

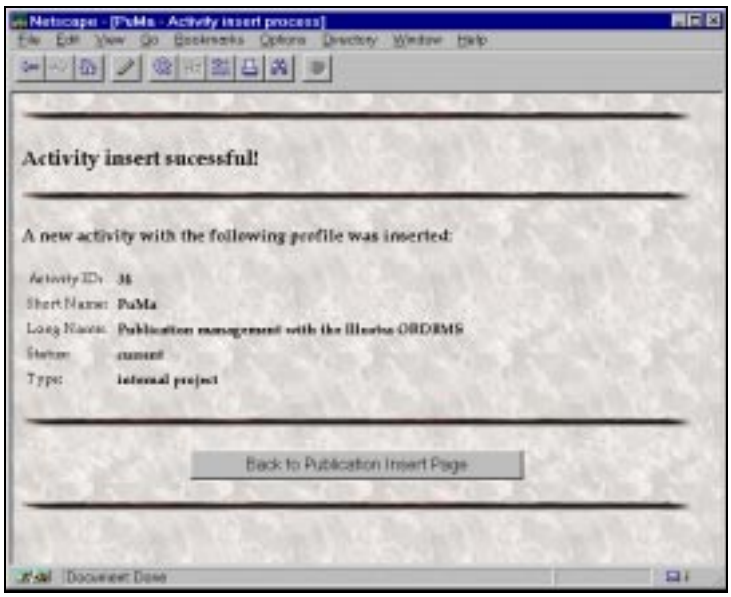

Abbildung 18: Eingaberückmeldung »activityProc«

4.4.2.2.3 Einfügemechanismen für Dokumente

Nachdem der Anwender die Daten des neuen Dokumentes eingegeben hat, müssen diese vom System in entsprechender Weise verarbeitet werden. Konkret bedeutet dies, daß folgende Arbeitsschritte erfolgen müssen:

1. Eintrag des Dokumentes in die entsprechende Publikationstabelle.
2. Eintrag der Autorenschaft in die Tabelle `newPublicationPersonnelTable`.
3. Eintrag der Zuordnung zu einer oder mehrerer Aktivitäten in die Tabelle `newPublicationActivityTable`.
4. Ermittlung, ob es außer der festen thematischen Zuordnung auch inhaltliche Verknüpfungen zu bestimmten Aktivitäten gibt.
5. Eintrag dieser inhaltlichen Verknüpfungen in `newPublicationKeywordTable`.

Es scheint daher angebracht, die einzelnen `INSERT`-Vorgänge in voneinander getrennten Application-Pages abzuhandeln, die automatisch nacheinander ablaufen.

4.4.2.2.3.1 Speicherung einer neuen Publikation in der jeweiligen Publikationstabelle (publicationProc)

Da der eigentliche INSERT-Vorgang nicht auf dem Bildschirm des Anwenders angezeigt werden muß, kann das Einfügen der Publikationsdaten in die jeweilige Publikationstabelle (articleTable, gmdReportTable, monographyTable oder diplomaTable) innerhalb einer einzigen Application-Page realisiert werden. Die für jede Tabelle benötigte INSERT-Anweisung kann in jeweils einem eigenen, mit einer entsprechenden Bedingung versehenen MIBLOCK ablaufen. Da die einzelnen INSERT-Anweisungen im großen und ganzen identisch sind, besteht auch hier keine Notwendigkeit, jeden einzelnen MIBLOCK zu erläutern. Die folgende Erläuterung bezieht sich somit lediglich auf die INSERT-Anweisung für die GMD-Publikationen.

4.4.2.2.3.1.1 Verarbeitung der übergebenen Variablenwerte

Mit den Variablen $ACT (die zugeordneten Aktivitäten) und $AUTHOR (die Autoren der Publikation) können unter Umständen jeweils mehrere Werte übergeben werden. Um diese später korrekt verarbeiten zu können, müssen sie erst, durch Kommas voneinander getrennt, in jeweils einen Textstring geschrieben werden, der dann jeweils einer neuen Variable zugewiesen wird:

```
AR      A     E          A     T     EPARATE          ,

AR      A     E          ACT             EPARATE          ,
```

4.4.2.2.3.1.2 INSERT-Anweisung für die Publikationstabelle

Um für jede Publikation die richtige INSERT-Anweisung auszuwählen, sind diese in jeweils einen, mit einer entsprechenden Bedingung versehenen MIBLOCK einzubetten. Diese sehen folgendermaßen aus:

- für diplomaTable:

```
BLOC      CO                (E      ,      TYPE,diploma)
```

- für articleTable:

```
BLOC      CO                (E      ,      TYPE,article)
```

- für monographyTable:

```
BLOC      CO                (E      ,      TYPE,boo      )
BLOC      CO                (E      ,      TYPE,proceeding)
```

- für gmdReportTable:

```
BLOC      CO                (E      ,      TYPE,
BLOC      CO                (E      ,      TYPE,
BLOC      CO                (E      ,      TYPE,Arbeitspapiere_der_
```

Danach folgt die INSERT-Anweisung. Bei der INSERT-Anweisung für gmdReportTable muß dabei noch ein Spezialfall beachtet werden: Aus verarbeitungstechnischen Gründen mußte, wie schon weiter oben beim Eingabeformular »gmdReportInsert« beschrieben, der Wert von $TYPE bei einem Arbeitspapier von »Arbeitspapiere der GMD« nach »Arbeitspapiere_der_GMD« umgeändert werden. Da in der Tabelle gmdReportTable jedoch wieder der String mit Leerzeichen gespeichert werden soll (weil der Wert in publicationType zur Bildschirmanzeige verwendet wird), muß ein zweiter MIBLOCK eingeführt werden:

```
AR      A     E          TYP          TYPE

BLOC      CO                (E      ,      TYPE,Arbeitspapiere_der_
AR      A     E          TYP      Arbeitspapiere der
BLOC
```

Ist der Publikationstyp »GMD-Studie« oder »GMD-Bericht«, so wird der Wert in $TYPE direkt nach $TYP übernommen. Nur wenn »Arbeitspapiere_der_GMD« auftaucht, trifft die Bedingung für den darunterstehenden MIBLOCK zu und $TYP erhält den Wert »Arbeitspapiere der GMD«. In die INSERT-Anweisung wird jetzt statt $TYPE die Variable $TYP eingesetzt. Bei allen anderen Publikationstypen kann $TYPE direkt in das INSERT-Statement übernommen werden.

Eine weitere Spezialität bei den GMD-Publikationen ist der Verlag und Verlagsort. Wie schon in der Erläuterung zum Eingabeformular »gmdReportInsert« geschildert, gibt es für GMD-Publikationen nur zwei mögliche Verlage bzw. Verlagsorte:

- GMD-Studien und Arbeitspapiere der GMD verlegt die GMD in Sankt Augustin.
- GMD-Berichte verlegt der Verlag Oldenbourg in München.

Da von »gmdReportInsert« keine entsprechenden Werte übermittelt wurden, müssen diese nun definiert werden. Auch dies geschieht mit einer `MIBLOCK`-Anweisung. Erst wird die Variable `$PUBLISH` bzw. `$PUBLOC` mit den Werten »GMD« bzw. »Sankt Augustin, Germany« belegt. Trifft die Bedingung für den danach folgenden `MIBLOCK` zu (es handelt sich um einen GMD-Bericht), dann werden die in `$PUBLISH` bzw. `$PUBLOC` enthaltenen Werte durch »Verlag Oldenbourg« bzw. »München, Germany« überschrieben.

```
AR      A      E          P      BL

AR      A      E          P      BLOC          an      t Augustin,

BLOC      CO                    (E      ,      TYPE,                    B
        AR      A      E        P      BL                          erlag
        AR      A      E        P      BLOC                 nchen,      er
        BLOC
```

4.4.2.2.3.1.3 Verarbeitung der übertragenen ASCII- bzw. PDF-Dateien

Damit die vom Anwender übertragenen Dateien in das nachfolgende `INSERT`-Statement eingefügt und somit in der Datenbank abgespeichert werden können, muß in der Webdriver-Konfigurationsdatei »Web.conf« ein spezieller Eintrag vorgenommen worden sein. Dieser Eintrag spezifiziert ein vom Illustra-DBMS erreichbares Verzeichnis als Upload-Directory. In diesem Upload-Directory werden die übertragenen Dateien vom Webdriver zwischengespeichert. Sie erhalten als neuen Dateinamen eine Prozeßnummer. Der Pfad des Upload-Directory und der jeweilige Dateiname werden vom Webdriver an die AppPage »publicationInsert« übermittelt (über die Variablen `$ASCII`, `$ABSASC` und `$PDF`). Dort werden Pfad und Dateiname in das `INSERT`-Statement eigefügt (`FileToLO('...')`).

Nun kann das `INSERT`-Statement erfolgen:

```
                        L            L                          ERT          TO gmdRep
(
                ,
      YEAR,
            T      TLE      ,
FileToLO(           AB      A      C      ),
FileToLO(           A      C                  ),
FileToLO(           P      F      ),
            TYP      ,
                  OL      ,
            P      BL                  ,
            P      BLOC
);
                              L
```

Nach erfolgtem `INSERT` werden die ASCII-Dateien automatisch vom PLS Text DataBlade invertiert.

Nachdem die Publikation in der richtigen Datenbanktabelle abgespeichert wurde, müssen noch die anderen Arbeitsschritte erfolgen. Die dazu notwendigen Werte müssen wieder in Form von Variablen an die nächste Application-Page gesendet werden. Es handelt sich um »actInsert1«, welche das Einfügen der Datensätze in die Tabelle `newPublicationActivityTable` übernimmt. Es wird also eine unsichtbare HTML-Form aufgebaut, welche die Werte weitergibt.

```
                                 AR
       FOR               A      E                   ext               ET      O                  PO      T
                      P      T TYPE                                     E                  A      E
                      P      T TYPE                                     E                  A      E
                      P      T TYPE                                     E                  A      E
                      P      T TYPE                                     E                  A      E
                      P      T TYPE                                     E                  A      E
             FOR
```

Der Anwender sollte von dem Page-Wechsel nichts bemerken. Dies setzt voraus, daß die Form-Inhalte automatisch übertragen werden. Dies läßt sich durch eine kleine JavaScript-Routine bewältigen:

```
             CR       PT LA                A      E                  ava      cript
document            ext      submit();
document            ext      reset;
                  CR      PT
```

Die Werte werden an »actInsert1« übertragen. Diese AppPage wird im Folgenden ausführlich erläutert.

4.4.2.2.3.2 Speicherung der Aktivitätszugehörigkeit (actInsert)

Bei der Speicherung der Aktivitätszugehörigkeit in der Tabelle `newPublicationActivityTable` ergibt sich eine spezielle Problematik:

In dem Fall, daß die neu eingefügte Publikation mehreren Aktivitäten zugeordnet werden soll, muß für jede Zugehörigkeit ein Datensatz eingefügt werden. Da nie von vornherein festgelegt werden kann, wie vielen Aktivitäten eine Publikation zugeordnet werden kann, muß das dafür benötigte `INSERT`-Statement dynamisch ausgelegt sein. Dies bedeutet, daß es rekursiv (für jeden benötigten Datensatz) einmal ablaufen muß.

Weder SQL noch das Web DataBlade verfügen über Funktionalitäten, eine derartige Routine zu realisieren. Dies bedeutet, daß für die Speicherung der Aktivitätszugehörigkeit eine Programmiersprache benutzt werden muß, welche die Implementierung rekursiver Routinen zuläßt. Für die konkrete Realisierung der Einfügefunktion gibt es grundsätzlich zwei Möglichkeiten:

- **Programmierung einer externen Datenbankfunktion**
  Das Illustra-DBMS gibt dem Datenbankentwickler die Möglichkeit, Funktionen, die mittels SQL nicht zu implementieren sind, in einer anderen Programmiersprache (C, Perl, Tcl/Tk usw.) zu entwikkeln. Diese Funktionen können dann von Illustra aus aufgerufen werden. Sie laufen dann in der Betriebssystem-Umgebung des DBMS-Servers ab und geben ihre Ergebnisse an das Illustra-DBMS zurück. Es wäre also möglich, eine Funktion zu implementieren, die aus den gelieferten Aktivitätsnummern (`activityId`) und der Publikationsnummer (`publicationId`) die benötigte Anzahl an `INSERT`-Statements generiert und diese an das Illustra-DBMS zurückgibt, welches die Statements dann verarbeiten kann.

- **Programmierung innerhalb einer Application-Page**
  Mit Hilfe der Programmiersprache JavaScript läßt sich eine Programmierung auch innerhalb von HTML-Pages ermöglichen. Voraussetzung dafür ist ein JavaScript-fähiger Web-Browser: Netscape Navigator (ab Version 2.0) oder Microsoft Internet Explorer (ab Version 3.0).

Da die Implementierung einer externen Datenbankfunktion für einen in der Programmierung unerfahrenen Entwickler eine relativ komplizierte Angelegenheit darstellt, wurde letztendlich die Entscheidung getroffen, auf die zweite Variante, die JavaScript-Programmierung innerhalb einer Application-Page, zurückzugreifen.

Bei der Anwendung von JavaScript innerhalb von Illustra-AppPages ist jedoch eine Einschränkung zu beachten:

Nachdem eine AppPage aus dem Illustra-DBMS abgerufen wurde, wird darin enthaltener SQL-Code sofort von der WebExplode-Funktion ausgeführt, die Ergebnisse dieser SQL-Statements gemäß der im `MISQL` formulierten Anweisungen formatiert und über den Webdaemon und den Webdriver an den Client weitergeleitet.

Die Möglichkeit, innerhalb einer AppPage mit Hilfe einer JavaScript-Routine alle benötigten `INSERT`-Statements zu generieren, diese in die AppPage schreiben zu lassen und dann die `INSERT`-Statements im Illustra-System ausführen zu lassen, schlägt somit fehl, weil die in der AppPage enthaltene JavaScript-Routine erst beim Client ausgeführt wird.

Die Lösung für diese Problematik besteht darin, eine AppPage vorzuhalten, welche die Vorbereitung der Variablenwerte vornimmt, eine zweite AppPage aufruft und die vorbereiteten Werte an diese übermittelt. In der zweiten AppPage kann dann die `INSERT`-Anweisung ablaufen.

Im folgenden Abschnitt wird dieser Mechanismus ausführlich beschrieben:

Beide in newPublicationActivityTable benötigten Werte werden von der AppPage »publicationProc« in Form der Variablen $ACTIV und $PUBID übergeben. In $PUBID ist nur ein Wert enthalten: die publicationId des neu eingefügten Dokumentes. In $ACTIV sind u. U. mehrere Werte enthalten: die activityId jeder zugehörigen Aktivität. Diese werden in Form eines Textstrings geliefert. Die einzelnen Werte sind durch Kommas voneinander getrennt. Es gilt nun, zwei Operationen durchzuführen:

- Herauslösung der einzelnen Werte aus $ACTIV und
- schrittweise Integration dieser Werte (zusammen mit dem Wert aus $PUBID) in jeweils ein INSERT-Statement.

4.4.2.2.3.2.1 AppPage »actInsert1«

Zunächst wird eine HTML-Form definiert, welche die Werte, die durch die anschließende JavaScript-Routine aus `$ACT` herausgelöst werden, zu der eigentlichen `INSERT`-AppPage (actInsert2) schickt. Da nach abgeschlossener Speicherung der Aktivitätszugehörigkeit auch noch die Autorenschaft und die inhaltliche Verknüpfung gespeichert werden müssen, werden die dafür benötigten Werte auch an »actInsert2« übermittelt, welche sie dann wieder weitergibt. Dabei ist zu beachten, daß die Variablen, deren Werte erst von der JavaScript-Routine eingefügt werden sollen, zunächst auf den Wert `0` gesetzt werden:

```
                              AR
FOR                   A       E           insval           ET      O                PO      T       AC
                P       T TYPE                                   E                A       E
                P       T TYPE                                   E                A       E
                P       T TYPE                                   E                A       E
                P       T TYPE                                   E                A       E
                P       T TYPE                                   E                A       E
                P       T TYPE                                   E                A       E
         FOR
                                     AR
```

Dann wird in der AppPage ein HTML-Tag gesetzt, daß ab der nächsten Zeile JavaScript-Anweisungen folgen:

```
         CR       PT LA                  A       E                   ava       cript
```

Um den in `$ACTIV` enthaltenen Textstring innerhalb der JavaScript-Routine bearbeiten zu können, muß er einer für JavaScript verständlichen Variable zugewiesen werden:

```
                              AR       ACT                    ACT
```

Als nächstes wird eine zweite Hilfsvariable definiert, die zur Ausführung der Routine benötigt wird. Sie soll anfangs keinen Wert enthalten:

```
    ame                  ;
```

Die danach folgende `if`-Anweisung sorgt dafür, daß die Routine nur dann ausgeführt wird, wenn der in `ACT` enthaltene Textstring länger als `0` Zeichen ist. Aus diesem Grund wurde in der Aktivitätstabelle `activityTable` eine Aktivität »No specific« mit der `activityId=33` eingerichtet. Wählt der Anwender in einem Publikationseingabeformular die Aktivitätsoption »No specific« aus, so enthält die Variable `ACT` den Textstring `33`.

```
i      (ACT      length            )
```

Die in `ACT` enthaltenen Werte werden, wie weiter oben beschrieben, durch Kommas voneinander getrennt. Am Ende des Strings wird jedoch kein Komma angefügt. Da dies von der Routine jedoch benötigt wird, fügt die folgende `if`-Anweisung dieses Komma an das String-Ende an:

```
i     (ACT     charAt(ACT     length          )               ,     )
  ACT     ACT            ,     ;
```

Die darauf folgende `while`-Schleife kopiert die erste `activityId` aus `ACT` in die Hilfsvariable Name. Da dies zeichenweise geschieht, muß ein Zähler eingerichtet werden, der anfangs mit `0` initialisiert wird:

```
counter
    hile((counter     ACT     length)          (ACT     charAt(counter)               ,     ))
      ame            ame       ACT     charAt(counter);
  counter            ;
```

Die nun in Name enthaltene `activityId` wird unter dem Namen `NameSQL` in die oben definierte HTML-Form »insval« eingefügt. Das gleiche geschieht mit dem in `$PUBID` enthaltenen Wert.

```
document     insval            ame          L     value          ame;
document     insval     P     B                 value                           AR
```

Der nach der Herauslösung der ersten `activityId` übriggebliebene String-Rest in `ACT` wird der Variablen `newact` zugewiesen.

```
ne     act     ACT     substring(counter          ,ACT     length);
```

Der in `newact` enthaltene String wird nun wieder als `ACT` in die HTML-Form »insval« geschrieben. Dabei ist folgendes zu beachten: Wenn die Routine an die letzte `activityId` angekommen ist, ist keine String-Rest mehr vorhanden, der nach `newact` geschrieben werden könnte. Dies würde dazu führen, daß die Routine beendet würde, bevor die letzte `activityId` durch die HTML-Form »insval« an die `INSERT`-Page »actInsert2« übermittelt wurde. Die letzte Aktivitätszugehörigkeit könnte somit nicht abgespeichert werden. Aus diesem Grund wird der Dummy »`ENDE,`« an `newact` angehängt.

```
document     insval     ACT     value     ne     act          E          E,     ;
```

Letztlich müssen die in der HTML-Form »insval« enthaltenen Variablen an »actInsert2« übermittelt werden:

```
document     insval     submit();
document     insval     reset();
```

4.4.2.2.3.2.2 AppPage »actInsert2«

In »actInsert2« laufen folgende Arbeitsschritte ab:

- Integration der durch »actInsert1« übermittelten Werte für `publicationId` (in `$PUBID`) und `activityId` (in `$NameSQL`) in ein `INSERT`-Statement.
- Herauslösung der nächsten in `$ACT` enthaltenen `activityId`
- Übermittlung der herausgelösten `activityId` nach »actInsert2«.

Dies bedeutet, daß »actInsert2« sich nach Abarbeitung der JavaScript-Routine wieder selbst aufruft, um ein `INSERT`-Statement auszuführen, die nächste `activityId` aus `$ACT` herauslöst usw. Dieser Vorgang wird solange ausgeführt, bis in `$ACT` keine `activityId` mehr übrig geblieben ist.

Der Vorgang läuft folgendermaßen ab:

- Als erstes wird die `INSERT`-Anweisung über `MISQL` an das Illustra-DBMS weitergeleitet.

```
                          L          L                          ERT          TO ne
          (     P     B            ,
                  ame          L);
                                L
```

- Nun müssen zwei HTML-Forms definiert werden:

- Eine für den Fall, daß aus `$ACT` eine weitere `activityId` herausgelöst wird. Diese entspricht der HTML-Form »insval« in »actInsert1«.
- Und eine für den Fall, daß keine weitere `activityId` mehr in `$ACT` vorhanden ist. In diesem Fall müssen die Werte, welche in den AppPages zur Speicherung der Autorenschaft und der Ermittlung und Speicherung der inhaltlichen Verknüpfungen benötigt werden, an die nächste AppPage »authorInsert1« übermittelt werden.

```
                                 AR
  FOR               A       E                 ext            ET     O                PO     T
                 P       T TYPE                                 E                 A      E
                 P       T TYPE                                 E                 A      E
                 P       T TYPE                                 E                 A      E
                 P       T TYPE                                 E                 A      E
        FOR
                                      AR
```

- Dann folgt wiederum eine JavaScript-Routine. Diese unterscheidet sich von der in »actInsert1« enthaltenen Routine nur in einigen wenigen Punkten:
  - Es erfolgt keine Hinzufügung eines Kommas am Ende des in `ACT` enthaltenen Strings, da dieses in dem angehängten »`ENDE,`« schon enthalten ist.
  - An `newact` wird kein »`ENDE,`« mehr angehängt, da dies schon vorhanden ist.
  - Am Ende der Routine wird `newact` daraufhin überprüft, ob der in ihr enthaltene Wert länger als `0` ist. `newact` ist dann gleich `0`, wenn die Routine bei »`ENDE,`« angelangt ist und kein Wert für `newact` übrigbleibt. In diesem Fall werden die in der HTML-Form »Next« enthaltenen Variablen an »authorInsert1« übermittelt.

Die Routine präsentiert sich somit folgendermaßen:

```
i       (ACT       length            )

   counter

         hile((counter      ACT      length)            (ACT      charAt(counter)                ,

            ame              ame         ACT      charAt(counter);


   document      insval           ame           L      value           ame;
   document      insval      P      B                 value                                   AR

   ne      act      ACT      substring(counter           ,ACT      length);

   i      (ne      act      length            )

      document      insval      ACT      value      ne      act;
      document      insval      submit();
      document      insval      reset();

   else

      document            ext      submit();
      document            ext      reset();
```

Nachdem alle `activityId`-Werte abgearbeitet wurden, wird die AppPage »authorInsert1« aufgerufen.

4.4.2.2.3.3 Speicherung der Autorenschaft (authorInsert1/2)

Die Speicherung der Autorenschaft läuft nach genau demselben Schema ab, wie die Speicherung der Aktivitätszugehörigkeit. Dies bedeutet, daß die AppPages »authorInsert1« und »authorInsert2« im Programmablauf genau den AppPages »actInsert1« und »actInsert2« entsprechen. Auf eine detaillierte Erklärung dieser AppPages kann daher verzichtet werden. Erwähnenswert ist nur die Tatsache, daß nach Abarbeitung der Autorenschaft die Variablen `$TYPE` und `$PUBID` an die AppPage »keywordInsert1« übergeben werden.

4.4.2.2.3.4 Ermittlung und Speicherung der inhaltlichen Verknüpfung zu Aktivitäten (keywordInsert1/2)

Die »keywordInsert«-Pages entsprechen in ihrer Grundfunktionalität den »actInsert«- und »authorInsert«-Pages. In einigen Punkten unterscheiden sie sich jedoch von diesen:

- Für die Speicherung in der Datenbanktabelle `newPublicationKeywordTable` sind außer der `publicationId` keine Werte vorhanden. Die zu speichernden Werte für das Attribut `keyword` müssen somit erst ermittelt werden.
- Je nachdem, von welchem Publikationstyp das neu eingefügte Dokument stammt, muß die Keyword-Suche in dem entsprechenden Textindex (`articleIndex`, `gmdReportIndex`, `monographyIndex` oder `diplomaIndex`) durchgeführt werden.

4.4.2.2.3.4.1 AppPage »keywordInsert1«

Zunächst wird auch hier die HTML-Form »insval« (zur Übermittlung der Verarbeitungswerte an »keywordInsert2«) definiert. Sie entspricht der gleichnamigen in »actInsert1« und »authorInsert1« enthaltenen HTML-Form.

Dem Leser soll nun ins Gedächtnis zurückgerufen werden, wie die inhaltliche Verknüpfung einer Publikation zu einer IPSI-Aktivität definiert ist:

»Kommt im Titel, Abstract oder Volltext einer Publikation der Kurzname einer IPSI-Aktivität vor, und die Publikation ist nicht schon dieser Aktivität über `newPublicationActivityTable` *fest zugeordnet, so kann davon ausgegangen werden, daß eine inhaltliche Verknüpfung zwischen der Publikation und der Aktivität besteht.«*

Eine solche inhaltliche Verknüpfung könnte theoretisch noch bewertet werden, indem die Vorkommenshäufigkeit des jeweiligen Aktivitätsnames festgestellt wird. Leider stellt das PLS Text DataBlade keine entsprechende Funktionalität zur Verfügung, so daß nur ermittelt werden kann, ob ein Begriff überhaupt vorkommt.

Um zu ermitteln, nach welchen Keywords innerhalb des neu hinzugefügten Dokumentes gesucht werden soll, wird die folgende `MISQL`-Anfrage definiert. Die zurückgegebenen Namen werden, durch Kommas voneinander getrennt, als ein String in die Variable `$KEYCAN` geschrieben:

```
                        AR        A      E                    EYCA
                        L                L                 ELECT                  T            CT
          activit             hort        ame
    FRO        activit      Table a
               ERE  activit              d        OT in (      ELECT activit            d
                            FRO         ne      PublicationActivit      Table
                                         ERE  publication      d             P      B            )
 (     ET      AR,           EYCA       ,            EYCA                       ,
                                L
```

Die Variable `$KEYCAN` wird nun an die schon aus »actInsert1« und »authorInsert1« bekannte JavaScript-Routine übergeben, die das erste `keyword` herauslöst und mittels der HTML-Form »insval« alle benötigten Werte an »keywordInsert2« übermittelt.

4.4.2.2.3.4.2 AppPage »keywordInsert2«

Um bei der Keyword-Suche im richtigen Textindex zu suchen, ist es notwendig, unterschiedliche, durch `MIBLOCK` voneinander getrennte `MISQL`-Anfragen zu definieren. Durch die von »keywordInsert1« übermittelte Variable `$TYPE` (welche ihr wiederum von allen ihr vorangestellten AppPages übermittelt wurde) kann dann entschieden werden, welcher `MIBLOCK` ausgeführt wird. Für den Fall einer Diplomarbeit, würde dies folgendermaßen aussehen:

```
               BLOC      CO                      (E      ,diploma,      TYPE)
                            L            L               ELECT CO            T (       )
      FRO         diplomaTable d,
             PlsOidRan          uer      (      diploma      ndex      ,               ame
```

```
                    ERE  (d            pls_oid      d      oid)
        A                (publication      d            P      B            );
                                       L
                      BLOC
```

Ist das vorgegebene Keyword gefunden worden, so gibt das `SELECT COUNT` einen Wert von `1` zurück. Dies führt dazu, daß diese »inhaltliche Verknüpfung« in `newPublicationKeywordTable` gespeichert wird:

```
                 BLOC      CO                   (E     ,     ,         )
                                L          L                       ERT          TO ne
        (     P     B              ,
                          ame           L      );
                                       L
                      BLOC
```

Was nun folgt, ist aus »actInsert2« und »authorInsert2« bekannt:

- Herauslösung des nächsten Keywords aus `$KEYCAN`
- Erneuter Aufruf von »keywordInsert2« und Übermittlung der Werte an diese
- usw.

Sind die Keywords abgearbeitet, ist der Einfügungsprozeß eines neuen Dokumentes komplett abgeschlossen. Als letzter Arbeitsschritt wird die `publicationId` und der `publicationType` des neu hinzugefügten Dokumentes an die AppPage »insertResult« übermittelt.

4.4.2.2.3.5 Anzeige der eingefügten Daten (insertResult)

Damit der Anwender sicher sein kann, daß das von ihm eingefügte Dokument einwandfrei in den Publikationsbestand integriert wurde, sollte er vom System eine entsprechende Rückmeldung bekommen. Diese Rückmeldung sollte den Anwender über alle Einfügeaktionen informieren, die aufgrund seiner Eingaben vom System durchgeführt wurden.

Da durch »keywordInsert2« die `publicationId` und der `publicationType` des neu eingefügten Dokumentes übermittelt wurden, können alle anzuzeigenden Daten problemlos aus den entsprechenden Tabellen abgefragt werden.

Begonnen wird mit den bibliographischen Daten. Die bibliographischen Daten sollten in Form einer Literaturreferenz angezeigt werden. Diese Vorgehensweise ist aus mehreren Gründen von Vorteil:

- Der Anwender hat sich durch die Suchergebnisanzeige »searchResult« und die Anzeige verknüpfter Aktivitäten »activityResult« mittlerweile an die Publikationsdarstellung in Form von Literaturreferenzen gewöhnt. Wenn er das neue Dokument gleich nach dem Einfügen als Literaturreferenz angezeigt bekommt, so ist dies für ihn eine buchstäbliche Bestätigung dafür, daß das Dokument »wirklich« ins System integriert wurde.
- Während in einer Tabellendarstellung eventuelle (vom Anwender eingegebene) `NULL`-Werte immer angezeigt würden, ist die Referenzdarstellung dynamisch. Das bedeutet, daß Attribute, die mit `NULL`-Werten belegt wurden, in der Referenz einfach nicht angezeigt werden.

Für die Ermittlung der Referenzdaten können, mit kleinen Abänderungen, die `SELECT`-Statements aus »searchForm« übernommen werden.

Um für jede Publikationsart die richtige `SELECT`-Anfrage zu formulieren, werden diese in jeweils einen eigenen `MIBLOCK` integriert. Als Beispiel folgt der `MIBLOCK` für die Monographien (Bücher):

```
                 BLOC      CO                (E      ,      TYPE,boo      )

                      AR      A     E            T     TLE
                      AR      A     E            FRO     T
                      AR      A     E            RE     T
                      AR      A     E                          _          LL

                                L          L                  ELECT                T
             m      publicationYear,
```

```
                m      publicationTitle,
                m      monograph            olume,
                m      monograph      Publisher,
                m      monograph      PubLocation,
                p      personnelFirst      ame,
                p      personnel      ur      ame
          FRO         personnelTable p,
                ne      PublicationPersonnelTable n,
                monograph      Table m
                     ERE  m      publication      d            P      B
          A               n      publication      d      m      publication      d
          A               n      personnel      d      p      personnel      d;

     (      F,
          (E      ,            ,      T      TLE),            ,      FRO      T      RE      T
          (      ET      AR,      FRO      T,            )      (      ET      AR,      RE      T,
REF                          EB_      O      E          AR    detail      publication      etail      pu
          (      F,      (E      ,            ,                 ),,            )
          (      F,      (E      ,            ,                 ),,                        )
          (      F,      (E      ,            ,                 ),      (to appear)            ,
               ET      AR,      FRO      T,      FRO      T                    ,
               ET      AR,      T      TLE,

                                   L
                              AR
     FRO      T      RE      T
                                   AR
                         BLOC
```

Nachdem die bibliographischen Informationen angezeigt wurden, sollte auch eine Anzeige der Aktivitäten erfolgen, denen die neue Publikation zugeordnet wurde. Diese kann wieder für alle Publikationsarten gemeinsam stattfinden.

```
                              L            L                  ELECT CO            T (      )
          FRO         ne      PublicationActivit      Table
                      ERE  publication      d            P      B            ;
                                   L

               BLOC       CO                        (      ,      ,            )
The inserted publication       as assigned to the       ollo     ing activities
     TABLE
                                   L            L                  ELECT a      activit            ho
          FRO         activit      Table a,
                ne      PublicationActivit      Table n
                     ERE  a      activit          d      n      activit          d
          A               n      publication      d           P      B           ;

     T                            TR

                                   L
          TABLE
             BLOC
```

Das gleiche soll für die Aktivitäten geschehen, zu denen eine inhaltliche Verknüpfung besteht:

```
                              L            L                  ELECT CO            T (      )
          FRO         ne      Publication      e           ordTable
                      ERE  publication      d           P      B            ;
                                   L

               BLOC       CO                        (      ,      ,            )
The inserted publication has semantic relations to the       ollo    vitingsacti
     TABLE
                                   L            L                  ELECT       e           ord
          FRO         ne      Publication      e           ordTable
                      ERE  publication      d           P      B            ;

     T                            TR

                                   L
          TABLE
             BLOC
```

Die AppPage »insertResult« zeigt sich in einem Web-Browser folgendermaßen:

Abbildung 19: Eingaberückmeldung für Dokumenteneinfügung »insertResult«

5 Fazit und Ausblick

5.1 Leistungsmerkmale des Prototypen

In Abschnitt 2.4.2. wurden drei grundlegende Anforderungen definiert, die ein elektronisches Dokumenten-Management-System erfüllen muß, um als ein solches zu gelten:

- Eine Speicherungsumgebung, in der die Dokumente gehalten werden
- Methoden, um Dokumente dieser Speicherungsumgebung hinzuzufügen
- Methoden, um gespeicherte Dokumente über Suchanfragen wiederzufinden

Der aus der Konzeption und Implementierung hervorgegangene Prototyp soll nun in Hinsicht auf diese Grundanforderungen überprüft werden.

Anmerkung:
Während der Implementierungsphase erhielt der Prototyp den Namen »PuMa« (für Publikations-Management). In den folgenden Zeilen wird somit statt von »dem Prototyp« schlicht von »PuMa« die Rede sein.

- **Voraussetzung 1: Vorhandensein einer Speicherungsumgebung, in der die Dokumente gehalten werden.**
  Als Speicherungsumgebung für die Dokumente steht eine Datenbank zur Verfügung. Diese basiert auf einem objektrelationalen Datenbank-Management-System.
- **Voraussetzung 2: Methoden, um Dokumente dieser Speicherungsumgebung hinzuzufügen.**
  Über HTML-basierte Eingabeformulare können der Dokumentendatenbank Dokumente hinzugefügt werden.
- **Voraussetzung 3: Methoden, um gespeicherte Dokumente über Suchanfragen wiederzufinden**
  Über eine HTML-basierte Eingabemaske können Suchanfragen an das Datenbanksystem abgesetzt und somit gespeicherte Dokumente wiedergefunden werden.

Der Prototyp erfüllt alle drei Grundanforderungen und kann somit als echtes Dokumenten-Management-System angesehen werden.

Im folgenden Abschnitt sollen die Leistungsmerkmale des Systems näher beleuchtet werden.

5.1.1 Umfangreiche Suchmöglichkeiten

Die Dokumentensuche wird über eine vordefinierte Suchmaske realisiert. Die Vorhaltung eines solchen Formulars bringt einige Vorteile:

- Der Anwender sieht auf einen Blick, welche Suchmöglichkeiten ihm zur Verfügung stehen.
- Auch weniger versierte Anwender haben einfachen Zugang zum Dokumentenbestand.

Über eine Befehlszeile kann der Anwender auf den vollen Suchfunktionsumfang des PLS Text DataBlade zugreifen. Mit der Möglichkeit, Mengen- und Abstandsoperatoren, Wildcards und Feldoperatoren zur Verfeinerung der Suchanfrage zu verwenden, stehen dem Anwender Möglichkeiten zur Verfügung, die ihm ansonsten nur von einem ausgewachsenen Information-Retrieval-System geboten werden.

Die relationale Speicherstruktur der Dokumentendatenbank ermöglicht zusätzlich eine Art der Suche, die mit einem Information-Retrieval-System nicht (oder wenn überhaupt nur sehr schwierig) möglich ist. Besonders deutlich wird dies an dem dynamischen »Autoren«-Feld des Suchformulars. Bei jedem Aufruf

des Formulars wird durch eine Datenbankanfrage der Bestand an Autoren aktualisiert. Während der Anwender eines IRS erst einen Blick in den Autorenindex werfen muß, um festzustellen, Dokumente welcher Autoren innerhalb des Systems vorhanden sind, sieht der Anwender von PuMa dies auf einen Blick. Bei den Veröffentlichungsjahren wiederholt sich dieses Konzept.

Die strukturierte Informationsvorhaltung läßt es auch zu, daß der Anwender sich einen groben Überblick über den Dokumentenbestand verschaffen kann. Dazu braucht er in der Befehlszeile nur das Wildcardzeichen * einzutragen (*=alles). Er kann nun über die Formularfelder eine grobe Selektion nach Autoren, Veröffentlichungsjahren und Dokumenttypen vornehmen.

5.1.2 Dynamische Suchergebnisanzeige

Die aufgrund einer Suche gefundenen Dokumenten werden im ersten Schritt in Form von Literaturreferenzen angezeigt. Die Generierung der einzelnen Referenzen geschieht dabei angelehnt an den APA-Standard für Referenzierung von Literatur, was die Übersichtlichkeit der Ergebnisanzeige gewährleistet. Mit einem Mausklick auf den Titel einer Referenz öffnet sich ein zweites Anzeigefenster und dem Anwender wird eine detaillierte Ansicht auf das jeweilige Dokument ermöglicht. Diese Detailanzeige beinhaltet folgende Informationen:

- **Komplette Referenz**
  Veröffentlichungsjahr, Autor, Titel, Informationen zu übergeordnetem Werk (falls das Dokument ein Artikel ist) usw.
- **Abstract** (soweit vorhanden)
- **Dokumenten-Download**
  Wird das Dokument in Form einer Acrobat-PDF-Datei vorgehalten, so kann der Anwender sich dieses auf seinen Rechner herunterladen.
- **Aktivitätszuordnung**
  Ist das Dokument einer oder mehreren IPSI-Aktivitäten zugeordnet, so wird dies dem Anwender angezeigt. Mit Mausklick auf den jeweiligen Aktivitätenname erhält der Anwender eine Liste aller im System enthaltenen Publikationen, die dieser Aktivität zugeordnet sind. Von dieser Liste aus kann dann wiederum auf die Detailanzeige gesprungen werden.
- **Inhaltliche Verknüpfung**
  Ist eine inhaltliche Verknüpfung zu Aktivitäten vorhanden, der das Dokument nicht fest zugeordnet ist, so werden auch diese Aktivitäten angezeigt. Auch hier kann der Anwender auf Mausklick einen Überblick über alle dieser Aktivität zugeordneten Publikationen erhalten.

Außer der Referenz, die es auf jeden Fall für jedes Dokument gibt, werden alle Komponenten der Detailanzeige dynamisch vorgehalten. Der Anwender bekommt somit nur Hinweise auf Informationen, die auch tatsächlich vorhanden sind. Selbst die Literaturreferenz ist dynamisch. Fehlen bestimmte Referenzinformationen (z. B. Volume oder Issue), so werden die entsprechenden Anzeigefelder einfach ausgeblendet.

5.1.3 Einfache Pflege des Dokumentenbestandes

Die Pflege eines relational strukturierten Datenbestandes ist erfahrungsgemäß eine relativ aufwendige Angelegenheit. Das Hinzufügen eines Dokumentes zieht in unserem Beispiel eine ganze Reihe von Tabelleneinfügungen nach sich.

Das PuMa-System stellt dem Anwender spezielle Eingabemasken zur Verfügung, mit Hilfe derer er neue Dokumente zum Bestand hinzufügen kann. Der Anwender muß lediglich die von ihm geforderten Informationen in das Formular eintragen und dieses abschicken. Alle Datenbankoperationen werden dann von

PuMa selbständig erledigt. Diese Automatisierung hat einerseits den Vorteil, daß das Hinzufügen von Dokumenten auch durch unbedarfte Anwender erfolgen kann. Andererseits verhindert der Hinzufügemechanismus, daß bestimmte Operationen (wie z. B. das Suchen von inhaltlichen Verknüpfungen in dem neuen Dokument) schlichtweg vergessen werden.

5.1.4 Globaler Zugriff

Durch die Vorhaltung auf dem World Wide Web ist das PuMa-System eben auch »world wide« erreichbar. Abgesehen davon, daß der Kreis der Nutzer dadurch praktisch uneingeschränkt ist, kann die Pflege des Systems absolut plattform- und standortunabhängig erfolgen. Ob der Administrator des Systems im IPSI vor seiner Unix-Workstation oder zuhause vor seinem Linux- oder Windows-PC sitzt spielt keine Rolle.

5.2 Zukunftsentwicklungen

Die verhältnismäßig kurze Zeitspanne, die für Konzeption und Implementierung zur Verfügung stand, hatte zur Folge, daß in PuMa nur sehr grundlegende Dokumenten-Management-Funktionalitäten realisiert werden konnten. Um aus dem Prototypen eine vollständige »real world«-Anwendung erwachsen zu lassen müßten noch einige Funktionskomponenten hinzugefügt werden, die im folgenden Abschnitt kurz beschrieben werden sollen.

5.2.1 Zugriffsbeschränkungen beim Dokumenten-Download

Die Vorhaltung von Publikationen ist rechtlich gesehen eine sehr sensible Angelegenheit. Zwar sind die Autoren der Publikationen IPSI-Mitarbeiter, doch von ihnen verfaßte Artikel sind in Fachzeitschriften und Proceedings erschienen. Die Urheberrechte haben die Autoren somit an die Herausgeber der übergeordneten Publikationen abgegeben. Die Möglichkeit, Artikel kostenlos von einem öffentlich frei zugänglichen Server herunterladen zu können, ist ohne das Einverständnis der Herausgeber eine illegale Handlung. Auch die GMD-Publikationen (GMD-Studien, GMD-Berichte, Arbeitspapiere der GMD) sind normalerweise nur gegen eine (wenn auch geringe) Schutzgebühr erhältlich.

Es wäre somit mehr als nur wünschenswert, wenn eine Möglichkeit bestünde, den Download von Dokumenten einzuschränken. Die Artikel sollten nur GMD-intern zum Download zur Verfügung stehen. Bei den GMD-Publikationen wäre vorstellbar, daß der Download erst dann möglich ist, wenn der Anwender über ein bestimmtes Paßwort verfügt.

Wie jedes andere multiuser-fähige Datenbank-Management-System verfügt auch Illustra Server über die Möglichkeit, bestimmte Teile einer Datenbank nur bestimmten Anwender zugänglich zu machen. Somit würde sich folgende Lösungsmöglichkeit eröffnen:

Statt die Download-Dateien in der jeweiligen Publikationsrelation (`articleTable`, `gmdReportTable`) zu speichern, werden sie in speziellen Relationen abgelegt (`articlePDF`, `gmdReportPDF`). Über einen Fremdschlüssel (`publicationId`) werden diese Relationen mit der jeweiligen Publikationstabelle verknüpft. Die PDF-Relationen werden dann nur einem speziell dafür zu schaffenden Download-Account zugänglich gemacht. Möchte ein Anwender dann von der Dokumentenanzeige aus die PDF-Datei eines Artikels oder einer GMD-Publikation herunterladen, so öffnet sich ein Fenster, das ihn auffordert, einen Anwendernamen und ein Paßwort einzugeben. Kann er diese nicht korrekt angeben, so wird ihm das Downloading verwehrt. Bei der Paßwort-Abfrage für die GMD-Publikationen könnte man einen Hinweis einbauen, daß dem Anwender nach Zahlung einer Schutzgebühr die entsprechenden Sicherheitsinformationen per eMail zugesandt werden.

Dadurch, daß die PDF-Dateien der Artikel und der GMD-Publikationen in voneinander getrennten Tabellen gehalten würden, könnten für diese jeweils eigene Zugriffs-Accounts mit unterschiedlichen Paßwörtern geschaffen werden.

5.2.2 Zugriffsbeschränkungen beim Hinzufügen von Dokumenten

Eine der wichtigsten Leistungsmerkmale von PuMa ist auch zugleich seine größte Sicherheitslücke: die globale Zugriffsmöglichkeit. Mit schönster Regelmäßigkeit ist in den Massenmedien von Menschen zu hören, die es als besonderen Spaß ansehen, in Web-Server-Systeme einzudringen, den dortigen Dokumentenbestand zu manipulieren oder sogar zu zerstören. Als eines der harmloseren Beispiele sei hier die Aktion einiger Hacker genannt, die es schafften, die WWW-Homepage der Central Intelligence Agency (CIA) so zu verändern, daß von dort aus Hyperlinks auf pornographische Web-Angebote führten.

Solche Beispiele machen deutlich, daß es nicht gerade wünschenswert ist, wenn JEDER von ÜBERALL aus Zugriff auf die Bestandspflegefunktionen von PuMa hat. Jedoch auch hierfür gibt es entsprechende Schutzmöglichkeiten:

1. Die Anzeige von PuMa ist HTML-Frame-basiert. Neben dem Hauptfenster, in dem z. B. das Suchformular und die Suchergebnisse angezeigt werden, gibt es noch ein Navigationsfenster. In diesem Navigationsfenster gibt es für jede PuMa-Funktion einen Button, mit Hilfe dessen der Anwender dann beispielsweise zum Suchformular gelangt. Die erste Sicherheitsstufe wäre somit, externen PuMa-Anwendern ein Frame-Set zu präsentierten, auf dem die Dokumentenhinzufügung überhaupt nicht angezeigt wird, und dieser ergo überhaupt nicht ahnt, daß es eine solche gibt.
2. Die AppPages, die für die Systempflege zuständig sind, werden wiederum in einer speziellen Datenbanktabelle gehalten, die wiederum nur mit einem speziellen Paßwort zugänglich ist. Möchte GMD-intern nun jemand auf eine Einfügeformular zugreifen, so muß er erst ein entsprechendes Paßwort eingeben, um eine Freigabe zu erhalten.
3. In dem Fall, daß ein Mitarbeiter viele Dokumente einzugeben hat, würde es natürlich zu einer nervenaufreibenden Angelegenheit ausarten, wenn er bei jedem Aufruf eines Dokumenteneingabeformulars das entsprechende Paßwort wiederholt eingeben muß. Für solch einen Fall kann das Illustra Web DataBlade so konfiguriert werden, daß es nahtlos mit HTTP-Server-Produkten der Firma Netscape zusammenarbeitet. Der Netscape-Server schickt nach einer korrekten Paßworteingabe einen sog. Cookie an den entsprechenden Client-Browser. Ein Cookie ist eine Art Passierschein, der im Cache-Speicher des Anwender gehalten wird. Jedesmal, wenn er wieder die gleiche paßwortgeschützte App-Page aufruft, schickt der Browser dann den Cookie an den Web-Server und zeigt ihm somit immer wieder den Passierschein. Ist dieser korrekt, kann auf eine wiederholte Paßwort-Eingabe verzichtet werden.

5.2.3 Neuimplementierung der Pflegefunktionen

Die Mechanismen, die für die automatische Verarbeitung eines neuen Dokumentes zuständig sind, basieren derzeit noch auf AppPages mit integrierten JavaScript-Routinen. Dies Art der Implementierung wurde jedoch nur deswegen gewählt, weil so innerhalb kürzester Zeit demonstriert werden konnte, wie solch eine automatische Bestandspflege grundsätzlich funktioniert. Sie bringt einige Nachteile mit sich:

- **Relativ langsame Verarbeitung**
  Für das Einfügen eines Dokuments werden stets mehrere AppPages benötigt. Wenn ein Dokument mehrere Autoren hat oder mehreren Aktivitäten zugeordnet wurde, so müssen manche AppPages (actInsert2, authorInsert2) mehrfach aufgerufen werden. Bei der Ermittlung und Speicherung der inhaltlichen Verknüpfungen (keywordInsert2) verhält es sich ebenso. Dies bedeutet, daß während des Einfügungsvorgangs ein ständiger Datentransfer zwischen Client und Server stattfindet. Wird das Einfügen eines neuen Dokumentes IPSI-intern vorgenommen, mag dies (aufgrund der hohen Datendurchsatzrate innerhalb des IPSI) nicht derartig viel Zeit in Anspruch nehmen, so kann dieser Vorgang IPSI-extern mitunter eine oder mehrere Minuten in Anspruch nehmen.

- **Unsichere Verarbeitung**
  Die einzufügenden Daten werden durch den vorgenannten Umstand immer wieder zwischen Client und Server hin und her jongliert. Würde es (aus welchen Gründen auch immer) zu einem Zusammenbruch der Client-Server-Verbindung während des Einfügevorgangs kommen, so wäre es u. U. möglich, daß die Informationen zu einem Dokument unvollständig abgespeichert werden. Da ein Teil der Datenverarbeitung außerhalb des Datenbank-Servers stattfindet hat dieser auch keine Kontrolle über den ganzen Vorgang.

Die Lösung dieses Problems liegt in der Neuimplementierung der Pflegemechanismen als Datenbankfunktion. Der Verarbeitungsvorgang wäre dann folgendermaßen:

1. Über das Eingabeformular werden die Daten an das Illustra-DBMS übermittelt.
2. Illustra startet eine externe Datenbankfunktion (beispielsweise in der Sprache C implementiert) und übergibt dieser Routine die Anwenderdaten.
3. Die Routine generiert aus den Anwenderdaten die entsprechenden SQL-Statements und führt diese im Illustra-DBMS aus.

Ab dem Zeitpunkt, an dem Illustra die Anwenderdaten erhält, hat es volle Kontrolle über den weiteren Verarbeitungsvorgang. Wenn die externe Datenbankfunktion unvorhergesehen abbricht, kann Illustra alle bis dahin ausgeführten SQL-Aktionen mit einem Rollback wieder zurücksetzen. Somit wäre die Konsistenz der Datenbank stets gewährleistet.

5.2.4 Namensänderung der Autoren

Nehmen wir an, der Autor »Bernd Pörner«, von dem schon Publikationen innerhalb des System vorhanden sind, würde heiraten und den Namen seiner Frau (Cassagne) annehmen, dann würde das für den Dokumentenbestand (und alles was davon abhängt) nicht zu unterschätzende Folgen haben:

Ein Anwender, der nach Publikationen von »Bernd Pörner« sucht, bekommt im Suchformular nun unter den Autoren diesen Namen nicht mehr angezeigt. Da er unter Umständen überhaupt nicht weiß, daß »Bernd Pörner« jetzt »Bernd Cassagne« heißt, ahnt er auch nicht, wer sich hinter diesem Namen verbirgt. Für ihn ist »Bernd Pörner« im System einfach nicht mehr existent (obwohl es ihn faktisch noch gibt).

Auch hierfür würde sich eine Lösung anbieten:

Zusätzlich zu der Relation `personnelTable` könnte man eine Relation `nameChangeTable` einführen. Diese bestünde aus dem alten Namen und der jeweiligen `personnelId`. Wenn nun das Suchformular auf `personnelTable` zugreift, muß bei jeder Person eine Prüfung erfolgen, ob es für sie auch einen Eintrag in `nameChangeTable` vorhanden ist. Ist dies der Fall wird der Name ausgelesen. In der Autorenanzeige des Suchformulars würden sich zwei Möglichkeiten realisieren lassen:

- **Anzeige beider Namen**
  Sowohl »Bernd Pörner« als auch »Bernd Cassagne« werden angezeigt. Die übersichtlichere Lösung.
- **Anzeige von »Bernd Cassagne, geb. Pörner«**
  Die formelle Lösung.

In der Referenzenanzeige könnte man den alten Namen hinter dem neuen Namen in Klammern anzeigen. Dies hätte noch den praktischen Nebeneffekt, daß auch der unwissende Anwender erfährt, daß »Bernd Pörner« jetzt endlich unter die Haube gekommen ist.

5.2.5 Aktualisierung von Datensätzen

Es kann vorkommen, daß ein Artikel ins System eingefügt wird, bei dem zwar bekannt ist, in welcher Zeitschrift er erscheinen wird; die Ausgabe, Seitenangabe usw. fehlen jedoch noch. Es müßte also eine Möglichkeit geben, Datensätze im nachhinein zu aktualisieren.

Dies könnte man folgendermaßen realisieren:

Aus einer speziellen Referenzenanzeige wählt man das zu aktualisierende Dokument aus. Die einzelnen Daten werden zu einem Eingabeformular übermittelt, das den Eingabeformularen zur Dokumenteneinfügung ähnelt. In diesem Aktualisierungsformular werden dann in den einzelnen Eingabefeldern die bisherigen Werte angezeigt. Die Änderungen können vorgenommen werden. Zum Abschluß werden die Daten wieder ans System übermittelt, welche es verarbeitet (mittels SQL-UPDATE). Ein Update sollte jedoch nur für bestimmte Informationen vorgenommen werden können, wie z. B. Informationen zur übergeordneten Publikation oder Nummer einer GMD-Publikation. Grundlegende Informationen wie Titel, Abstract, Volltext, Autoren, Aktivitäten berühren die Datenbankkonsistenz. Wenn derartige Änderungen vorgenommen werden sollen, scheint es angebrachter, das Dokument komplett zu löschen (was im nächsten Punkt erläutert wird) und neu einzufügen.

5.2.6 Löschen von Dokumenten

Die Notwendigkeit, ein Dokument aus einem Publikations-Management-System zu löschen wird wahrscheinlich nicht oftmals auftreten. Publikationen sind feststehende Dokumente, die eigentlich nicht gelöscht werden sollten. Es kann jedoch vorkommen, daß ein Dokument irrtümlicherweise von zwei Mitarbeitern zweimal eingegeben wurde, oder daß grundlegende Dokumentinformationen wie Titel oder Volltext falsch eingegeben wurden.

Das zu löschende Dokument sollte man wieder aus einer Referenzenliste auswählen können. Die Löschung des Dokumentes sollte durch eine Sicherheitsabfrage vom Anwender bestätigt werden.

Beim Löschungsvorgang sind dann sowohl der Eintrag aus der jeweiligen Publikationstabelle (`articleTable` usw.) als auch die Einträge aus den Verknüpfungsrelationen (`newPublicationActivityTable`, `newPublicationPersonnelTable`, `newPublicationKeywordTable`), die mit der jeweiligen Publikation zu tun haben, zu löschen. Dabei ist besonders bei den Autoren darauf zu achten, daß in dem Falle, wenn ein Autor durch dieses Dokument neu zum Autorenbestand hinzugekommen ist, dieser aus der Tabelle `personnelTable` gelöscht wird.

Anmerkung der Autoren:
Wahrscheinlich könnte man diese Verbesserungswunschliste noch endlos fortführen. Dies ist wohl eine der grundlegenden Eigenschaften aller Software-Produkte. Verbessern kann man sie immer wieder. Es bleibt jedoch zu hoffen, daß die Hauptpunkte durch die vorstehende Aufzählung abgedeckt sind.

Anhang A - Literaturverzeichnis

[APA 1987] American Psychological Association (1987).
Publication Manual of the American Psychological Association, Third Edition.
Hyattsville, MD: Autor

[Chen 1976] Chen, P.-S. (1976).
The Entity-Relationship Model - Towards an Unified View of Data.
In: ACM Transaction on Data Base Systems 1 (1976) 1, 9-36.

[Henzler 1992] Henzler, R. G. (1992).
Information und Dokumentation.
Berlin et al: Springer-Verlag

[Knorz 1997] Knorz, G. (1997).
Datenbank-Entwurfsmethoden.
In: M. Buder, W. Rehfeld, T. Seeger & D. Strauch (Hrsg.), Grundlagen der praktischen Information und Dokumentation (Band 2, pp. 664-687). München, New Providence, London, Paris: K: G. Sauer

[Pörner 1997] Pörner, Bernd (1997).
Ausarbeitung zur Diplomvorbereitung.
Darmstadt: Fachhochschule Darmstadt, Fachbereich Information und Dokumenation.

[Stonebraker 1996] Stonebraker, M. (1996).
Object-Relational DBMSs - The next great wave.
San Francisco, CA: Morgan Kaufmann Publishers Inc.

Anhang B - Bibliographie

Eco, U. (1993).
Wie man eine wissenschaftliche Abschlußarbeit schreibt
(W. Schick, Übers.). Heidelberg, C. F. Müller (Originalveröffentlichung 1977)

Kuhlen, R. (1992).
Experimentelles und praktisches Information Retrieval.
Konstanz: Universitätsverlag

Lemay, L. (1995).
Web Publishing mit HTML.
München: Markt & Technik

Merz, T. (1996).
Die PostScript- & Acrobat-Bibel.
München: Thomas Merz Verlag

Netscape Communications Corp. (1996**).**
JavaScript Authoring Guide.
http://home.netscape.com/comprod/products/navigator/version_2.0/script/index.html (28. Mai 1997).

Blueridge Technologies (1996).
Document management terms and definitions.
http://www.optix.com:80/definitions.html (28. Mai 1997).

Association for Information and Image Management International.
AIIM Homepage.
http://www.aiim.org (28. Mai 1997).

Parapadakis, G. (1996).
32 Document Management Avenue.
http://www.grp-home.demon.co.uk:80/toc.htm (28. Mai 1997).

LANTIMES online (1995).
Document Management: Managing the Paper Storm.
http://www.wcmh.com/lantimes/usetech/compare/pcdocmng.html (28. Mai 1997).

Beyond the MLA Handbook: Documenting Electronic Sources on the Internet.
http://falcon.eke.edu/honors/beyond-mla/ (28. Mai 1997)