Einsatz von *IBM DB2 Digital Library* an Wissenschaftseinrichtungen des Landes Baden-Württemberg

Projektbericht

- Entwurf -

Stand: 31.03.2000

Bibliotheksservice-Zentrum Baden-Württemberg, Konstanz

Lehrstuhl für Organisation und Management von Informationssystemen, Universität Ulm

Rechenzentrum der Universität Karlsruhe

Universitätsbibliothek Karlsruhe

Zentrum für Datenverarbeitung der Universität Tübingen

European Content Management Competency Center, IBM Deutschland Entwicklung GmbH

Redaktion:

Bibliotheksservice-Zentrum Baden-Württemberg

Inhalt

0. Digitale Bibliotheken

(Sebastian Goeser)

Mit den digitalen Bibliotheken ist seit etwa 1994 ein neuer Systemtyp in die Informatik eingeführt worden, der sich mit allen Aspekten der Benutzung schwachstrukturierter Information in elektronischer Form befasst. Das Konzept der digitalen Bibliothek integriert Ansätze aus den Bereichen Datenbankverwaltung, Kooperations- /Workflow-Systeme und Information Retrieval und verfolgt den Anwendungsaspekt, große Mengen von typischerweise unstrukturierten Datenobjekten einer typischerweise großen Benutzergruppe unter Berücksichtigung rechtlicher Beschränkungen zugänglich zu machen. Aus einer funktionalen Sicht unterstützen digitale Bibliotheken - durchaus in Analogie zu realen Bibliotheken- den gesamten Lebenszyklus von Informationsobjekten, der die Phasen (Prä-)Produktion, Aufnahme, Verwaltung, Zugänglichkeit und schließlich Postproduktion und Distribution mit einschließt.

1. Projektbeschreibung

Das Projekt “Einsatz von IBM DB2 Digital Library an wissenschaftlichen Einrichtungen in Baden-Württemberg” - kurz BW/DL Projekt - strebt die Einrichtung einer landesweit operierenden digitalen Bibliothek in Baden-Württemberg an. Auf der Basis des IBM-Produkts IBM DB2 Digital Library werden Methoden und Systeme entwickelt, um universitäre Information in Form von multimedialen Inhaltsobjekten für verschiedenste Anwendungen des akademischen Bereichs verfügbar zu machen. Diese Anwendungen schließen die Speicherung/Archivierung, Suche, Auslieferung und Distribution der Informationsobjekte ein und werden auf Basis der IBM Content Management-Technologie realisiert.

Die Digitale Bibliothek Baden-Württemberg (kurz: BW/DL) ist eine verteilte digitale Bibliothek, die landesweite Suchverfahren mit der lokalen Speicherung von Objekten über das regionale Wissenschaftsnetz integriert.

1.1 Aufbau der Teilprojekte

(Sebastian Goeser)

Das Projekt Baden-Württemberg DL schließt die folgenden Teilprojekte bzw. Projektpartner mit ein:

1. Rechenzentrum der Universität Karlsruhe
2. Lehrstuhl für Organisation und Management von Informationssystemen (LOMI), Universität Ulm
3. Zentrum für Datenverarbeitung, Universität Tübingen
4. Universitätsbibliothek Karlsruhe
5. Bibliotheksservice-Zentrum, Konstanz

Die IBM ist durch das European Content Management Competency Center (ECMCC) der IBM Deutschland Entwicklung GmbH an dem Projekt beteiligt.

Die folgende Tabelle stellt die wesentlichen Charakteristika und Ziele der einzelnen Teilprojekte dar (Stand: Juli 1998):

Zentren	Uni Ulm	TU Karlsruhe UB	TU Karlsruhe RZ	Uni Tübingen	BSZ Konstanz
Beteiligte Institutionen	LOMI	UB	RZ	ZDV	BSZ
Fachbereiche	Informationstechnik, E-Technik			Physik	
Vorgänger-systeme	Mac-basierter Prototype (Vorlesungsserver)	VVV-System, KVK			
Projekte					
Projektziele					
Inhaltliche Projektziele	Unterstützung von Vorlesungsautoren, Einbringung von Vorlesungsmaterialien	Karlsruher Volltext-Veröffentlichungs-verzeichnis (VVV) verfügbar machen	Landesweite Datensicherung, zentraler Library Server	Verfügbarmachung Physik-Dokumente, z.B. Reports	Musiksammlung e-verfügbar machen, Langzeitarchivierung
Methodische Projektziele		Digitalisierung von Mikroformen		1. "Strukturelle" Dokumentformate 2. Aufbau eines Publikationsservers	Erprobung Dublin Core (Konverter DC->SWB, Standardisierung des DC-Formats)
Daten					
Korpus	Vorlesungen, E-Technik Dokumente	1. 600 Objekte des VVV 2. Mikroformen-Bestände (im Besitz der BLB) , 3.NCSTRL-Dokumente		Physik-Dokumente, z.B. Reports, Dissertationen	Teile der Musiksammlung (Theaterzettel) der WLB Stuttgart
Formate	Quicktime (audio track) , PS, PDF	Postscript, PDF, Papier		LaTeX/TeX/ XML/HTML/SGML	Scanning (evtl. Integration mit Audio)
Umfänge	10-20 Vorlesungen	600 (VVV)		50 Dokumente / Jahr	max. 20000 Theaterzettel
Komponenten					
Environments	AIX 4.3 /Oracle 7.3	AIX 4.2+3/Oracle 7.3	AIX 4.2 /Oracle 7.3	NT 4.0 /DB2 5.2	AIX 4.3 /Oracle 7.3
Hardware	RS/6000 F50/2	RS/6000 F50/1	RS/6000 SP/4	NetFinity Workstation	RS/6000 F50/1
DL Komponenten	LS/OS, Search, VideoCharger (?)	LS/OS, Search	LS/OS, Search	LS/OS, Search	LS/OS, Search, Collection Treasury
Andere Komponenten	ViaVoice	Scanner, OCR, TransTool		Servlets / WebSphere, IIS, evtl. Apache	
Standards	ICS	Z39.50		XML/MathML	Dublin Core
Entwicklungs-sprache	xlC, MS VC++ 5.0, Java 1.1.6+	xlC, Java 1.1.6+, Python 5.2	Java 1.1.6+	Java 1.1.6+, MS VC++, Java 1.2	net.data, HTML, Java 1.1.6+

1.2 Beziehung zu anderen Projekten

(Sebastian Goeser)

Das Projekt BW/DL ist im Zusammenhang mit zahlreichen Ansätzen der Forschung an und Entwicklung von digitalen Bibliotheken an deutschen und europäischen Hoch-

schulen zu sehen. Beispielhaft seien das vom Bundesministerium für Wissenschaft und Technologie geförderte Projekt *MeDoc* (Projektdauer 1995-1997) zur Entwicklung volltextbasierter Informationsdienste und das vom Deutschen Forschungsnetz geförderte Projekt *OPUS* zur Online Publikation von Hochschulschriften angeführt. Auf europäischer Ebene ist das Projekt ARIADNE als forschungsnaher Prototyp einer digitalen Bibliothek zu nennen. Gegenüber diesen Ansätzen fokussiert das Projekt BW/DL auf ein hochskalierendes Produktionssystem, das durch den Einsatz eines Softwareprodukts für digitale Bibliotheken gekennzeichnet ist. Ein weiterer Unterschied zu den genannten Ansätzen liegt in dem kollektionsunabhängigen Datenmodell als Basis verschiedener Bibliotheksapplikationen, wobei die aktuellen Standardisierungsansätze berücksichtigt oder ausgenutzt werden.

1.3 Konzept der elektronischen Universität

(Sebastian Goeser)

Digitale Bibliotheken sind ein Schlüsselbaustein für den Aufbau der “Hochschule der Zukunft”, in der Wissen ist nicht mehr an eine bestimmte Lehrsituation gebunden, sondern campusweit - und ggf. weltweit - unter dem "business model" der Hochschule verfügbar ist. Dabei integriert die digitale Bibliothek die gesamte Wertschöpfungskette der universitären Wissensproduktion von der Forschung bis zum Lernen und Anwenden. Dennoch wird die Hochschule mit einer digitalen Bibliothek wie der BW/DL nicht zur Insel: Vielmehr realisiert eine derartige DL eine verteilte und heterogene Architektur, in der fast alle Funktionalität sowohl zentral als auch dezentral durch teilnehmende Institutionen kontrolliert sein kann.

1.4 Elektronische Bibliotheken, Horizon versus Digital Library

(Till Hänisch, Guido Hölting)

(Auszug, der vollständige Text ist unter `http://vts.uni-ulm.de/query/longview.meta.asp?document_id=17` abrufbar.)

1.4.1 Hintergrund

Das Land Baden-Württemberg hat sich für ein landeseinheitliches Lokal- und Verbundsystem in seinen öffentlichen Bibliotheken entschieden, das auf der Basis modernster EDV-Technologie (Datenbank, Netzwerk, Betriebssysteme etc.) die Bibliotheken für die Herausforderungen der Informationsgesellschaft in angemessener Weise vorbereiten soll. Die Wahl ist dabei auf das Bibliothekssystem HORIZON der Firma Dynix gefallen.

In Anbetracht einer Reihe ähnlich titulierter Produkte anderer Anbieter, die sich ebenfalls im Rahmen von Projekten um Fördermittel des Landes bewerben, ist eine begriffliche Abgrenzung und funktionale Analyse erforderlich.

HORIZON ist danach ein **integriertes Bibliothekssystem**, das zur effizienteren Abwicklung der bibliothekarischen Arbeiten von den Möglichkeiten der modernen Informationstechnologie Gebrauch macht und daher zunächst auch nur einen den „klassischen“ Arbeiten innerhalb einer Bibliothek entsprechenden Funktionsumfang aufweist (Katalogisierung, Erwerbung, Ausleihe, OPAC etc.). Es handelt sich also um ein System, das die Bestände lediglich **erwirbt**, **erfasst**, **nachweist** und **verwaltet**.

Die Digital Library der IBM repräsentiert dagegen den Typ einer **elektronischen bzw. digitalen Bibliothek**, deren Aufgabe es vornehmlich ist, **Inhalte** eines multimedialen

Charakters (Text, Bild, Ton, Video etc.) zu speichern und zu distribuieren. Dazu gehört sicher auch ein Katalogsystem, das jedoch in den meisten verfügbaren Produkten dieser Kategorie nur einen minimalen Funktionsumfang aufweist und insbesondere durch das Fehlen von Modulen zur Bewältigung der nach wie vor erforderlichen klassischen Bibliotheksarbeit nur für die digitale Bibliothek selbst brauchbar ist. Die Zielsetzung ist eine andere.

Im Idealfall könnten Systeme beiden Typs sich in einer modernen, zukunftsorientierten Bibliothek bzw. einem Verbund **sinnvoll ergänzen** und von ihren besonderen Leistungsfähigkeiten gegenseitig profitieren.

1.4.2 Dynix HORIZON

HORIZON versteht sich in allererster Linie zunächst als ein System, mit dem die herkömmliche bibliothekarische Arbeit durch den Einsatz moderner Datenverarbeitungstechniken leichter und effizienter durchgeführt werden kann. Es gleicht somit einer besonderen Form der „Lagerverwaltung“, bei der die zu verwaltenden Gegenstände die klassischen Printmedien Bücher und Zeitschriften sind (in jüngerer Zeit auch ergänzt durch audiovisuelle Medien wie Begleit-CDROMs, Videos etc.). HORIZON bildet damit die Arbeitsabläufe einer Bibliothek auf eine Datenbank bzw. Datenbankapplikationen ab. Somit steht es in Konkurrenz zu anderen Bibliothekssystemen, die dasselbe Ziel verfolgen.

Im Gegensatz zu den im Land bereits eingesetzten, heterogenen und größtenteils von der Konzeption her schon veralteten Systemen, ist HORIZON ein zeitgemäßes integriertes Bibliothekssystem, das nicht nur die gegenwärtige bibliothekarische Arbeit vollständig abdeckt, sondern zugleich über eine gute Skalierbarkeit und Offenheit hinsichtlich der Anbindung anderer, Inhalte liefernder Systeme verfügt. Auch hier gibt es vergleichbare Produkte anderer Anbieter. Als Ergebnis einer europaweiten Ausschreibung erhielt jedoch die Firma Dynix den Zuschlag (zunächst für das Verbundsystem, das am BSZ in Konstanz installiert wird, später ebenfalls für die in Zukunft einheitlichen Lokalsysteme). Im Gegensatz zu den Entwicklungen der anderen Bewerber, waren bei HORIZON die erforderlichen systemtechnischen Voraussetzungen erfüllt sowie wesentliche Funktionskomponenten bereits vorhanden. Außerdem stellte es das preislich günstigste Angebot dar.

1.4.3 IBM Digital Library

Konventionelle Bibliotheken dienen im wesentlichen dazu, Dokumente (beispielsweise Bücher) zu archivieren. Besteht seitens eines Kunden Interesse an einem bestimmten Dokument, kann dieses mit Hilfe von Katalogen, die nach speziellen, formalen Kriterien von den Bibliotheken geführt werden, lokalisiert werden. Ein Exemplar dieses Dokuments wird dann dem Anwender ausgehändigt, beziehungsweise reproduziert (üblicherweise fotomechanisch). Ist es dem Kunden nicht möglich, die Bibliothek persönlich aufzusuchen, dann kann er (inzwischen oder jedenfalls in absehbarer Zukunft) mit Hilfe eines in einem Netzwerk verfügbaren Katalogs (OPAC) selbst recherchieren. Die Reproduktion und Distribution des gewünschten Dokuments erfolgen dabei wiederum manuell über Mitarbeiter der Bibliothek. Die Recherche in einem solchen Katalog erfordert jedoch unter Umständen bibliothekarische Erfahrung, so dass auch hierbei im konkreten Einzelfall Hilfe durch Bibliotheksmitarbeiter notwendig sein kann. Dieses Vorgehen kostet (viel) Zeit und Geld, insbesondere, wenn Kunde und Bibliothek räumlich weit voneinander entfernt sind oder es sich um Dokumente handelt, deren Reproduktion aufwendig ist wie beispielsweise bei Bildern und Filmen. Um den Aufwand für dieses Verfahren in einem erträglichen Rahmen zu halten, sorgt man dafür, dass die räumliche Distanz zwischen Kunde und Bibliothek nicht zu groß wird. Dies wird dadurch erreicht, dass an Stellen, an denen sich eine größere Anzahl potentieller Kunden befindet (beispielsweise an Universitäten, in Großstädten), lokale

Bibliotheken eingerichtet werden.

Dieser personal- und zeitintensive Vorgang kann insbesondere bei der Distribution durch den Einsatz moderner Computer- und Netzwerktechnologie vereinfacht und verbilligt werden. Dazu muss erstens eine Recherchemöglichkeit bereitgestellt werden, die auch für den normalen, nicht bibliothekarisch vorgebildeten Kunden verwendbar ist und zweitens ein Distributionsweg geschaffen werden, der keinerlei manuelle Eingriffe benötigt und innerhalb einer kurzen Zeitspanne (Sekunden bis Minuten) die gewünschten Dokumente direkt an den Arbeitsplatz des Kunden liefert. Das erste Ziel kann durch den Einsatz spezieller, medienspezifischer Technologien — für Textdokumente beispielsweise Volltextrecherche — erreicht werden, das zweite durch den Einsatz von moderner WAN Technologie. Voraussetzung hierfür ist allerdings die elektronische Archivierung der Dokumente.

Aufgabe einer digitalen bzw. elektronischen Bibliothek ist deshalb die Archivierung und Distribution von Dokumenten und erst in zweiter Linie eine Bestandsverwaltung im Sinne einer Katalogisierung.

Der Einsatz spezieller Recherchetechniken (möglicherweise in Kombination mit herkömmlichen Katalogisierungsmethoden) ermöglicht allen Kunden die gezielte Recherche in umfangreichen Dokumentbeständen. Die einfache und billige Reproduzierbarkeit elektronischer Dokumente ermöglicht den nahezu gleichzeitigen Zugriff sehr vieler Anwender auf eine Dokumentsammlung beziehungsweise ein Dokument.

1.4.4 Integration

Elektronische Bibliotheks(verwaltungs)systeme (HORIZON) und elektronische Bibliotheken (IBM Digital Library) unterscheiden sich von ihrer Funktionalität und von ihrer Zielsetzung her. Allerdings könnte eine Integration beider Systeme mit dem jeweils spezifischen Know-how zu Synergieeffekten führen und für die künftigen Informationsnachfrager wie -anbieter einen deutlichen Mehrwert darstellen. Zur Integration sind zwei Wege denkbar:

- Eine elektronische Bibliothek kann organisatorisch als einer unter mehreren „konventionellen“ Bibliotheksstandorten angesehen werden, wobei die dort befindlichen Dokumente mit einem Verwaltungssystem wie HORIZON katalogisiert werden. Dabei wird zusätzlich ein Verweis (Link) auf das Dokument im Katalogisat abgespeichert, der den elektronischen Zugriff über einen geeigneten Client (z.B. einen WWW-Browser) ermöglicht. HORIZON wird also für die bibliothekarische Katalogisierung der elektronisch gespeicherten Dokumente verwendet. Um dies zu erreichen, müsste HORIZON um entsprechende Schnittstellen erweitert werden, um Zugriff auf die Daten der elektronischen Bibliothek zu erhalten.

- Die in HORIZON enthaltenen Katalogisierungsdaten könnten aber auch in die elektronische Bibliothek übernommen werden. Dies würde in einer Übergangsphase eine Recherche in den noch nicht elektronisch archivierten Dokumentbeständen ermöglichen, die zwar nicht die Qualität einer sonst möglichen Volltextrecherche bietet, aber wenigstens einen eingeschränkten Zugriff auf den kompletten Bibliotheksbestand erlaubt. Um dies zu erreichen, müsste die Suchmaschine der elektronischen Bibliothek um eine Schnittstelle zum HORIZON-OPAC erweitert werden.

1.5 Erfolgskriterien

(Sebastian Goeser)

Das Projekt BW/DL wird als erfolgreich abgeschlossen betrachtet, wenn zumindest die Gesamtheit der folgenden Kriterien nach Ansicht der Projektverantwortlichen erfüllt sind:

1. Die Kollektionen jedes Teilprojekts sind über einen zentralen Libraryserver suchbar und referenzierbar
2. Die Inhaltsobjekte jeder Kollektion sind unter definierten technischen und administrativen Voraussetzungen jedem Benutzer zugänglich, der diese Voraussetzungen erfüllt
3. Es existiert ein einheitliches Datenmodell, das es ermöglicht, alle Kollektionen des Projekts in einheitlicher Weise zu beschreiben

Das Gesamtsystem wurde zur Projektlaufzeit auf einem Workshop präsentiert und demonstriert (Datum: 1. Juli 1999).

2. Anforderungen

(Sebastian Goeser)

Wie jedes reale Software-Informationssystem ist auch die BW/DL Anforderungen von seiten der intendierten bzw. tatsächlichen Benutzer (Benutzeranforderungen) und Anforderungen aus den technischen Gegebenheiten (Systemanforderungen) unterworfen, wobei die Benutzeranforderungen sich schwerpunktmäßig auf die Funktionalität des Gesamtsystems beziehen. Wenn das Wachstum des Systems hinsichtlich Datenvolumen, Benutzern, Requests, Applikationen und Konfigurationen betrachtet wird, dann können daraus außerdem Anforderungen an die Skalierbarkeit des Gesamtsystems abgeleitet werden.

2.1 Benutzeranforderungen

Benutzer des Systems BW/DL sind bzw. werden sein:

- Menschen, die die BW/DL zur Suche und zum Bezug von Inhaltsobjekten benutzen **(Requestoren)**
- Menschen, die der BW/DL Inhaltsobjekte und deren Beschreibung zur Verfügung stellen **(Provider)**
- Menschen, die auf der Basis der BW/DL Anwendungen und Kollektionen entwickeln **(Entwickler)**
- Menschen, die die BW/DL oder Teile davon administrieren **(Administratoren)**

Die beiden erstgenannten Benutzergruppen werden auch als Endbenutzer bezeichnet. Alle Benutzer treten mit der BW/DL in einer homogenen, angepassten und in ihre Arbeitsumgebung integrierten Weise in Interaktion. Dabei kann natürlich eine Person an einem Arbeitsplatz in mehreren derartigen Benutzerrollen (z.B. als Administrator und Entwickler) tätig sein.

Sortiert nach derartigen Rollen, ergeben sich die folgenden Benutzer- bzw. Benutzungsanforderungen:

1. Requestoren

- Suche in einem kollektionsübergreifenden Katalog der Objekte unter Ausnutzung spezifischer Objekteigenschaften einschließlich, soweit möglich, des physischen Inhalts eines Objekts
- Suche, Anzeigen und Browsen von Metadaten zu einem oder mehreren Objekten aus einer oder mehreren Kollektionen
- Auslieferung von Objekten in digitaler Form an die eigene Arbeitsumgebung
- Visualisierung (Browsing/Viewing) von Objekten in der eigenen Arbeitsumgebung
- Authentifizierung, Einrichtung und Benutzung einer Zugangsberechtigung

2. Provider

- Benutzerschnittstelle zur Aufnahme von Objekten in digitaler Form und ihren Metadaten entsprechend einer Spezifikation
- Berücksichtigung von kollektions- und objektbezogenen Zugriffsrechten auf Benutzerklassenebene
- Typischerweise werden Provider gleichzeitig auch Requestoren sein.

3. Entwickler

- Bereitstellung der Kernfunktionalität der BW/DL in einer homogenen, transparenten und performanten Programmierschnittstelle
- Unterstützung von gängigen Werkzeugen und Standards im Bereich Web-Anwendungen

4. Administratoren

- Verfügbarkeit von Werkzeugen zur Administrierung der Server in der BW/DL hinsichtlich zumindest
  i. Speicherhierarchie
  ii. Definition von Benutzern und Benutzerklassen
  iii. Reorganisierung von Servern
  iv. Erzeugung von Log- und Trace-Information
  v. Performanzverhalten
- Werkzeuge zur Installation und Konfiguration und Aktivierung/Deaktivierung von Komponenten der BW/DL

2.2 Systemanforderungen

Aus technischer Sicht und unter Architekturgesichtspunkten gibt es für das System BW/DL folgende Anforderungen:

- Verwaltung der Metadaten und Indexinformationen auf einem zentralen Library Server (Katalogserver)
- Gemeinsames Multimedia-Datenmodell für alle Kollektionen
- Verwaltung der digitalen Objekte auf lokalen oder zentralen Objektservern
- Unterstützung von heterogenen Systemumgebungen bestehend aus AIX/Oracle und NT/DB2 Installationen
- Schreib-/Leseschnittstelle zum landesweiten Bibliothekssystem
- Optional: Authentifizierung über lokale Benutzerdatenbanken

2.3 Skalierbarkeitsanforderungen

Aus der Evaluation des Produkts IBM DL und der Entwicklung der BW/DL sollen Aussagen zur Skalierbarkeit des Gesamtsystems gewonnen werden, um kritisches Systemverhalten in einer landesweiten BW/DL-Lösung weitestmöglich auszuschließen. Konkret ergeben sich die folgenden Skalierbarkeits-Anforderungen an die BW/DL:

- Unterstützung von n > 10 Objektservern mit Skalierung bei wechsender Serverzahl
- Gemeinsamer Library Server für 10^6 Objekte mit Skalierung bei wachsender Anzahl von Objekten
- Konzept für Skalierung von Datenvolumen und Durchsatz bis ca. 0,5 TB
- N > 100 gleichzeitige Benutzer bzgl. der dritten Systemschicht (Web-Schnittstelle)
- n > 10 gleichzeitige Benutzer bzgl. der mittleren Systemschicht (DL Klienten)
- Performance aus Benutzersicht für Applikationen unter dem BW-Modell

- Store/Retrieve von Dokumenten aus mehreren kleinen Teilen
- Schnelle Suche bis zur Anzeige einer optimalen Resultatliste

3. Architektur und Design

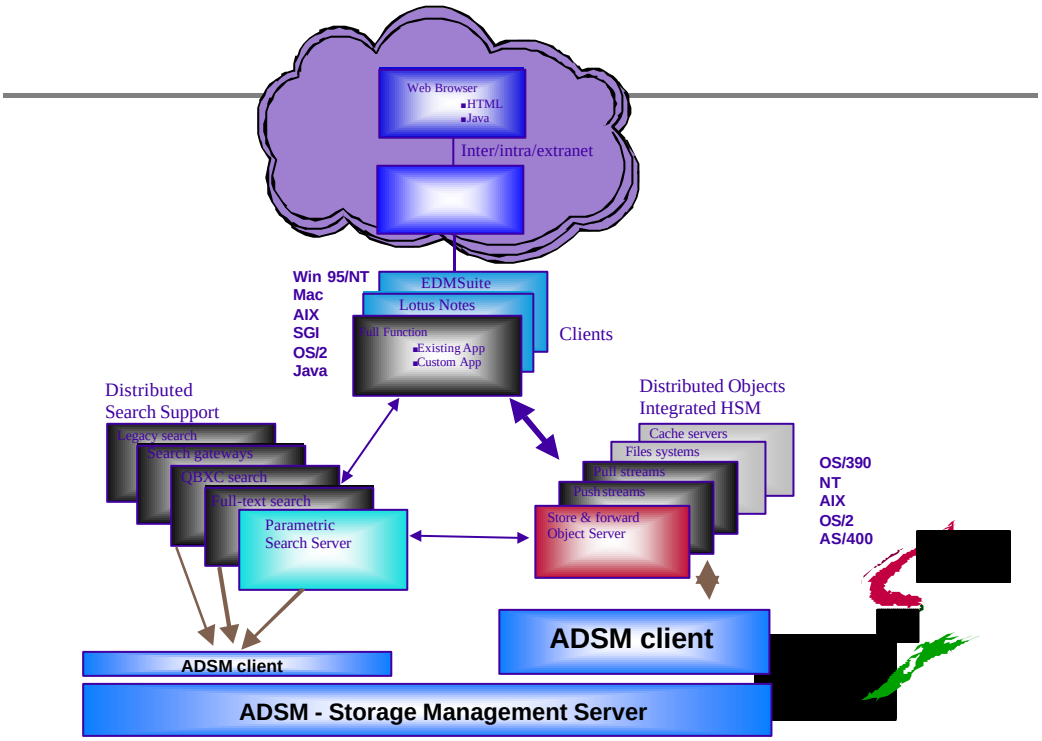
Die BW/DL ist ein verteiltes Informationssystem, das auf der Basis des Produkts IBM DB2 Digital Library eine landesweite Infrastruktur zur Verwaltung von Informationsobjekten unterstützt und dem Endbenutzer verschiedene Anwendungen zur Arbeit mit diesen Objekten anbietet. Aus der Datensicht ist für die BW/DL das multimediale, standardbasierte Datenmodell entscheidend, da durch dieses Modell die einzelnen Anwendungen, die ja sehr stark von den jeweiligen Kollektionen abhängig sind, vertikal integriert werden. Aus der Systemsicht ist für die BW/DL das Zusammenspiel von globalen Funktionen (z.B. Suche, Browsing), lokalen Funktionen (z.B. Delivery) und den externen Schnittstellen zur Bibliothekswelt entscheidend, so dass die BW/DL insgesamt als verteiltes und heterogenes System bezeichnet werden kann.

3.1 Produktarchitektur IBM DB2 Digital Library

(Sebastian Goeser)

Das Produkt IBM DB2 Digital Library V2.4 ist Teil des Lösungsangebots der IBM im Bereich Content Management. Mit diesem Produkt adressiert IBM neben dem klassischen Management strukturierter Daten auch die wachsenden Kundenanforderungen bei der Verarbeitung schwach strukturierter und/oder multimedialer Daten. IBM DB2 Digital Library folgt dabei einem sehr innovativen Architekturansatz, bei dem die Bandbreitenproblematik bei der Verfügbarmachung von großen Inhaltsmengen (z.B. Videos, Bildkollektionen) durch eine Dreiecksarchitektur aufgefangen wird.

Diese Architektur ist in der folgenden Fig. beschrieben:



Sie besteht aus einem Library-Server, der die Metadaten und Indizes verwaltet, einem oder mehreren Objektservern, die die physikalischen Objekte in digitalisierter Form verwalten und ausliefern, und einem Zwischenschicht-Server (client oder middle tier

server), der die Dienstleistungen des Gesamtsystems über transparente und homogene Programmierschnittstellen anbieten. Dabei garantiert das System immer Transaktionssicherheit zwischen Library- und Objektserver. Katalog- und Objektdaten können über Schnittstellen zu entsprechenden IBM-Architekturen in hierarchische Speicherverwaltungen eingebracht werden, neben proprietären Klienten kann die Systemfunktionalität auch über Web-Applikationen dem Benutzerkreis verfügbar gemacht werden.

3.2 Datenmodell

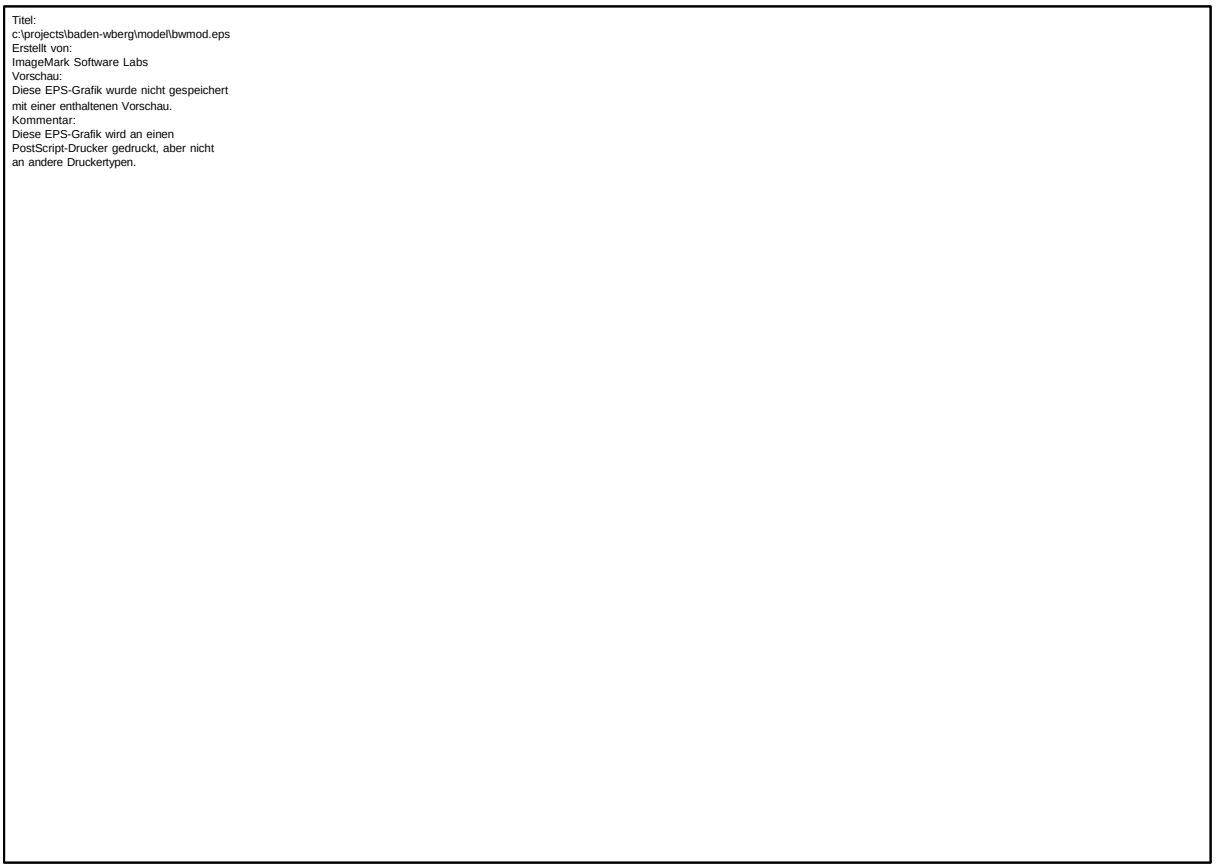
(Till Hänisch)

Ziel bei der Erstellung des Datenmodells war die Definition eines gemeinsamen relationalen/objektorientierten Modells über alle Kollektionen hinweg. Die Schwierigkeiten dabei lagen darin, das Modell einerseits so flexibel zu gestalten, dass unterschiedliche Kollektionen (beispielsweise Lehrmaterialien in Ulm und Theaterzettel in Konstanz) abgebildet werden können, andererseits das Modell so einfach zu halten, dass der Aufwand bei der Anwendungsentwicklung ökonomisch vertretbar bleibt. Dazu war die Entwicklung von Hilfsprogrammen notwendig, die im Abschnitt „Anwendungsentwicklung“ näher erläutert werden.

Basis für das Modell war die Integration des Dublin-Core Metadata Element Set als offenem Standard für den bibliographischen Kern des Modells.

3.2.1 Konzeptuelles Datenmodell

In diesem Modell sind nur die Objekte und einige wesentliche Attribute dargestellt. Die blau gefärbten Objekte stellen den zwischen den Projektpartnern erzielten Konsens über die Abbildung des Dublin-Core, die roten die zur Beschreibung und Archivierung multimedialer Objekte notwendigen Erweiterungen dar.



3.2.2 Abbildung auf die Digital Library

Manche Objekte haben Attribute, die mehrere Werte enthalten können. Beispielsweise kann ein Dokument mehrere Autoren haben.

In einem relationalen Modell würde man diese Attribute als 1:n oder sogar m:n Relationen darstellen und entsprechend diese Relation als Tabelle abbilden. Eine Person kann also an mehreren Creator-Beziehungen beteiligt sein, ein Dokument ebenso.

Somit können einem Dokument beliebig viele Personen als Creator zugeordnet werden. Das Problem bei dieser Art der Abbildung ist, dass erstens eine zusätzliche Tabelle (Creator) benötigt wird und dass zweitens bei der Suche ein Join notwendig wird. Das erste Problem lässt sich bei der Abbildung auf die DL dadurch lösen, dass die Beziehung zwischen Objekt und mehrwertigem Attribut als Folder-Item Beziehung modelliert wird. Handelt es sich um eine 1:n Beziehung, kommt das entsprechende Attribut (Item) in genau einem Folder vor, handelt es sich um eine m:n Beziehung, wird es von mehreren Foldern referenziert. Das Problem, dass bei einer Abfrage über mehrere Attribute ein Join notwendig wird, bleibt naürlich bestehen. Um dies zu umgehen, wird (wo möglich und sinnvoll) eine Denormalisierung verwendet, die verschiedenen Werte des jeweiligen Attributs werden als Attribut=Wert Paare in einem Textpart des Objekts gespeichert. Dies hat nebenbei den Vorteil, dass bei der Suche mit Textminer eine unscharfe Textsuche verwendet werden kann, die z.B. die schreibweisentolerante Suche nach Namen erlaubt.

Zur Untersuchung der programmiertechnischen Aspekte der beiden Modellierungsarten muss zwischen den verschiedenen Zugangsmethoden – Browsing und Query – unterschieden werden.

3.2.3 Foldermodell

Beim Browsing im Foldermodell in der „n“-Richtung (im Beispiel vom Document zum Creator zur Person) kann jedes DL-API verwendet werden, selbst das C++/JAVA-API bietet die Möglichkeit, die Items eines Folders zu erhalten (ohne großen Aufwand bzw. Performanceeinbruch). Die umgekehrte Richtung (vom Creator zum Document) ist nicht ganz so trivial, hier muss zumindest das C-API verwendet werden. Problematisch ist dieser Fall bei der Auflösung einer Query auf einer der „unteren“ Indexklassen: Wird beispielsweise eine Query auf der Indexklasse „BWValue“ durchgeführt, und soll der jeweils zugehörige Dokumenttitel (in der Indexklasse „BWObject“) angezeigt werden, muss für jedes Item aus dem Queryresultat der Weg nach oben durch die Hierarchie verfolgt werden. Dies ist jedoch ein relativ langsamer Vorgang, so dass sich dieses Vorgehen bei einer großen Anzahl von Treffern verbietet. Der einzig sinnvolle Weg ist hier, den dafür notwendigen Join von der Datenbank direkt ausführen zu lassen. Es müssen also Code-Module erstellt werden, die dieses Aufsteigen in der Hierarchie effizient erledigen.

Zwei mögliche Implementierungen dieser Funktion wären: Die Trefferliste einer Subquery wird in einer Datenbanktabelle gespeichert. Mit einer SQL-Query, die einen Join über diese Tabelle und die SBTLINKS-Tabelle enthält, erhält man die ITEMIDs der jeweiligen „BWObject“-Objekte. Diese Query ist natürlich für jede Indexklasse unterschiedlich (PROBLEM: Items, die in mehreren Folder-Indexklassen vorkommen können, erfordern zusätzlich einen Zugriff auf die SBTITEMS-Tabelle).

Eine zweite Möglichkeit wäre, die Folder-Item-Beziehungen beim Start der Applikation zu lesen und im Speicher zu puffern. Damit kann dann ein sehr schneller Zugriff erfolgen.

Der Nachteil der ersten Methode ist, dass die Trefferlisten zuerst in der Datenbank gespeichert werden müssen, bei einer großen Anzahl von Treffern bedeutet dies wieder ein erhebliches Performanceproblem[1]. Dem könnte unter Umständen durch einige „vorgefertigte“ Resultlisten-Tabellen abgeholfen werden (für die verschiedenen Sprachen, Formate,...). Das Problem dabei ist, wie bei der zweiten Lösung, dass diese gepufferten Listen/Tabellen nicht mehr unbedingt mit den eigentlichen DL-Tabellen konsistent sind. Für den ersten Fall ließe sich das zwar unter Umständen mit Triggern,... erreichen, dies bedingt jedoch wieder andere Probleme. Eine einfache Lösung für diese Problematik ist derzeit nicht in Sicht. Vermutlich lassen sich die hier beschriebenen Schwierigkeiten nur durch ein komplexes, heuristisches Verfahren lösen, es müsste ein (für die DL und das Datenmodell spezifischer) Queryoptimizer entwickelt werden.

Alternativ kann bei Abfragen auf Attributen die DL komplett umgangen und die Queries vollständig auf der Datenbank durchgeführt werden. Diese Funktionalität ist bereits prototypisch implementiert, einige Fragen wie die Integration von Volltextrecherchen (Textminer) sind jedoch noch nicht endgültig geklärt.

Im zweiten Fall – einer Query über mehrere Indexklassen – kann wie folgt vorgegangen werden: Die Query wird in Subqueries, die jeweils eine Indexklasse betreffen, aufgelöst. Diese Queries werden nacheinander durchgeführt. Zu den Trefferlisten der einzelnen Queries werden durch das oben beschriebene Heraufwandern in der Hierarchie die zugehörigen „BWObject“-Objekte bestimmt. Aus den erhaltenen Trefferlisten wird dann die Schnittmenge gebildet, diese bildet das Resultat der Query. Das Vorgehen hier entspricht also etwa einem „sort-merge-join“. Eine Query über mehrere Indexklassen liefert also prinzipiell eine Menge von „BWObject“-

[1] Angenommen, die DL enthielte 50000 Dokumente, jeweils in 2 Sprachen (beispielsweise deutsch und englisch). Jede Abfrage, die die Sprache einschließt, führt dann zu einer Subquery mit 50000 Treffern. Wenn das Einfügen in die Resultlistentabelle mit 500 Records/sec erfolgt, bedeutet dies 100 sec je Query.

Objekten zurück. Dieses Verfahren ist genau dann sinnvoll, wenn erstens eine Query ein relativ „teurer“ Vorgang und zweitens das Heraufwandern in der Hierarchie pro Dokument ein relativ „billiger“ Vorgang ist. Diese Voraussetzung ist nur dann erfüllt, wenn der Aufstieg in der Hierarchie direkt auf der Datenbank erfolgt. Dies ist insbesondere entscheidend, wenn die Ergebnisse der einzelnen Subqueries eine hohe Kardinalität haben. Insbesondere bei Abfragen, die das Dokumentformat oder die Sprache einschließen, wird dieser Fall vermutlich relativ häufig auftreten.

Für Abfragen und Browsing wird hier also nur ein Join auf der obersten Indexklasse („BWObject“) durchgeführt. Da dies der einzige auftretende Join ist, erscheint eine Implementierung mit erträglichem Aufwand möglich. Die Zerlegung in Subqueries erlaubt auch die Einbindung von verschiedenen Query-Engines (Textminer,...).

Zusammenfassung

Eine relationale Modellierung mehrwertiger Attribute ist unter Performance-Gesichtspunkten nur dann sinnvoll, wenn (nahezu) alle Abfragen unter Umgehung der DL durchgeführt werden. Natürlich kann die bei der hierarchischen Modellierung verwendete Methode zur Realisierung von Queries über mehrere Indexklassen auch auf die relationale Modellierung übertragen werden, diese bietet dann jedoch keine Vorteile mehr.

In jedem Fall bedeutet die Existenz solcher mehrwertiger Attribute ein erhebliches (Performance-) Problem und einen relativ hohen Aufwand bei der Entwicklung von Anwendungen. Der wesentliche Grund dafür ist, dass die DL nur Operationen auf einer einzigen Indexklasse unterstützt.

3.2.4 Primärschlüssel

Sowohl im Folder- wie auch im relationalen Modell können die DL-ITEMIDs als Primärschlüssel zur Verknüpfung mehrer Indexklassen verwendet werden. Dies funktioniert allerdings nur solange, wie diese bei der Ausführung eines Anwendungsprogramms bekannt sind. Insbesondere bei Loaderprogrammen, also Programmen, die neue Items in der DL anlegen, ist dies nicht der Fall. Wird ein Item von keinen anderen Items referenziert, ist dies kein Problem, da dann zum Zeitpunkt der Erstellung der Primärschlüssel nicht benötigt wird. Wird beispielsweise ein neues Dokument („BWObject“) angelegt, wird dabei dessen Primärschlüssel nicht benötigt. Wenn allerdings ein unter Umständen schon vorhandenes Objekt referenziert werden soll, treten Probleme auf. Soll beispielsweise zu einem Dokument ein Autor eingetragen werden, muss geprüft werden, ob ein entsprechendes „BWCreator“ Objekt bereits existiert. Ist dies nicht der Fall, muss geprüft werden, ob ein entsprechendes „Person“ Objekt existiert. Neue Items dürfen nur dann erzeugt werden, wenn noch keine entsprechenden Objekte existieren. Für diese Prüfungen wird jeweils – unabhängig von der Art der physikalischen Modellierung - ein eindeutiger Schlüssel benötigt.

Dieses Problem tritt bei den folgenden Indexklassen auf:

Indexklasse	Primärschlüssel
OBJECT	ResourceID, ResourceScheme (DL)
LANGUAGE	Language
SUBJECT	SubjectID (oder SubjectText (Part)+ SubjectScheme)
PERSON	CompleteName
FORMAT	MimeFormat

3.2.5 Attribute

3.2.5.1 Formatspezifische Attribute

Zur Beschreibung der einzelnen Dokumente (Parts) werden formatspezifische Attribute benötigt. Beispielsweise könnte ein Bild (mime type image) die Attribute „Größe in Pixeln“, „Auflösung in dpi“ erhalten, ein JPEG-Bild (mime type image/jpeg) zusätzlich das Attribut „Kompression“, ein Video (mime type video/mpeg) die Attribute „Dauer in Sekunden“ und so weiter.

Diese werden in der DL abgebildet, indem die Attribute von den tatsächlichen Parts getrennt werden, das heisst, dass sie nicht als physikalische Attribute im Objekt (den Objekten) Parts gespeichert, sondern in einer eigenen Indexklasse gespeichert werden.

3.2.5.2 Textattribute mit unbeschränkter Länge

Da die IBM-DL für die Feldlänge von Textattributen eine Obergrenze vorsieht (insbesondere müssen, wenn DB2 als Basistechnologie verwendet wird, alle Attribute einer Indexklasse in eine Speicherseite – 4 kByte – passen), können Attribute mit unbeschränkter Länge nicht direkt gespeichert werden, sie müssen in einem Textpart gespeichert werden.

Die folgenden Indexklassen enthalten Parts:

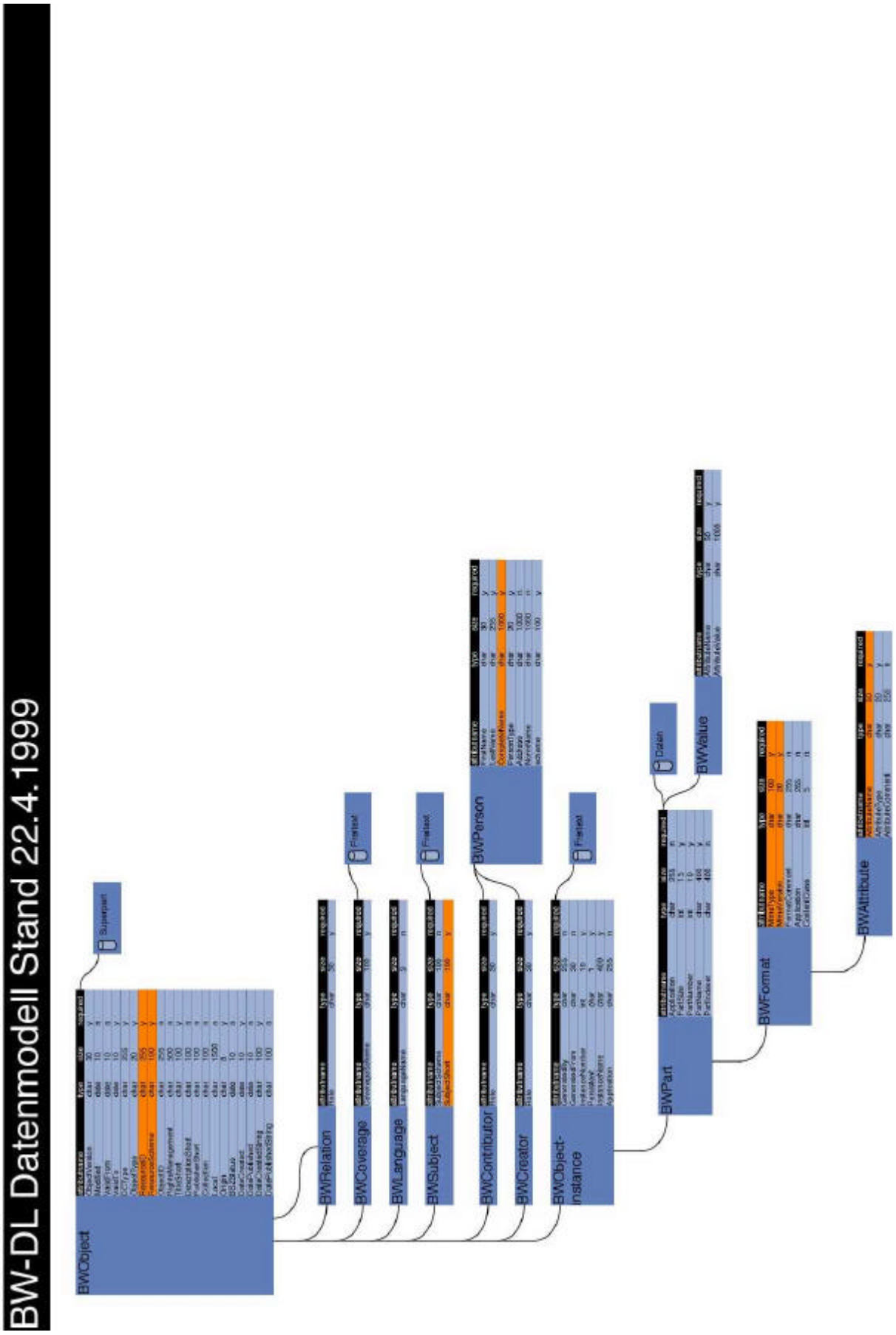
Indexklasse	Inhalt	Short Attribut
BWObject	Siehe unten	J
BWPart	die tatsächlichen Daten	N
BWCoverage	Freitext	N
BWSubject	Text zum Subject	J
BWOobjectinstance	Beschreibung der Instanz, Freitext	N

Ist „Short Attribut“ = „J“, hat die jeweilige Indexklasse ein oder mehrere Attribute, die eine Kurzforn (i.A. die ersten n Zeichen) des Partinhalts enthalten. Diese Attribute dienen der Optimierung bei der Ausgabe von (Treffer-) Listen: Normalerweise wird bei der Ausgabe einer Liste nur der Inhalt des Attributs ausgegeben, es muss nicht der Part vom Objectserver geholt werden.

Die Indexklasse OBJECT enthält einen Textpart, den sogenannten Superpart, der die folgenden Attribut-Wert-Paare enthält:

Attribut	Kardinalität
CREATOR	N
CONTRIBUTOR	N
PUBLISHER	1
DESCRIPTION	1
MODIFIED_BY	N
MODIFIED_AT	N
SUBJECT_VALUE	N
SUBJECT_SCHEME	N
TITLE	1

Dieser Superpart erfüllt mehrere Funktionen: Erstens enthält er Daten, die eigentlich als Attribute gespeichert werden könnten, deren Feldlänge für die DL (bzw. die zugrundeliegende Datenbank) zu groß, insbesondere unbeschränkt ist (Publisher, Description, Titel, Rightsmanagement). Zweitens enthält er mehrwertige Attribute, die ohne weiteres nicht (suchbar) auf die DL abgebildet werden können (Modified_By, Modified_At). Schließlich enthält er Informationen, die mit einer Volltextsuche (insbesondere mit einer inexakten/schreibweisentoleranten) recherchierbar sein sollen (Subject_Value, Subject_Scheme, Creator, Contributor). Diese letzte Redundanz wurde eingeführt, da bei der Suche nach Namen bzw. Schlagworten dem Anwender die korrekte Schreibweise nicht immer bekannt ist.


BW-DL Datenmodell Stand 22.4.1999
BWObject
Superset
BWRelation
BWCoverage
BWLanguage
BWSubject
BWContributor
BWCreator
BWObject-instance
Firstlast
BWPerson
BWPart
BWFormat
BWValue
BWAttribute
Datei

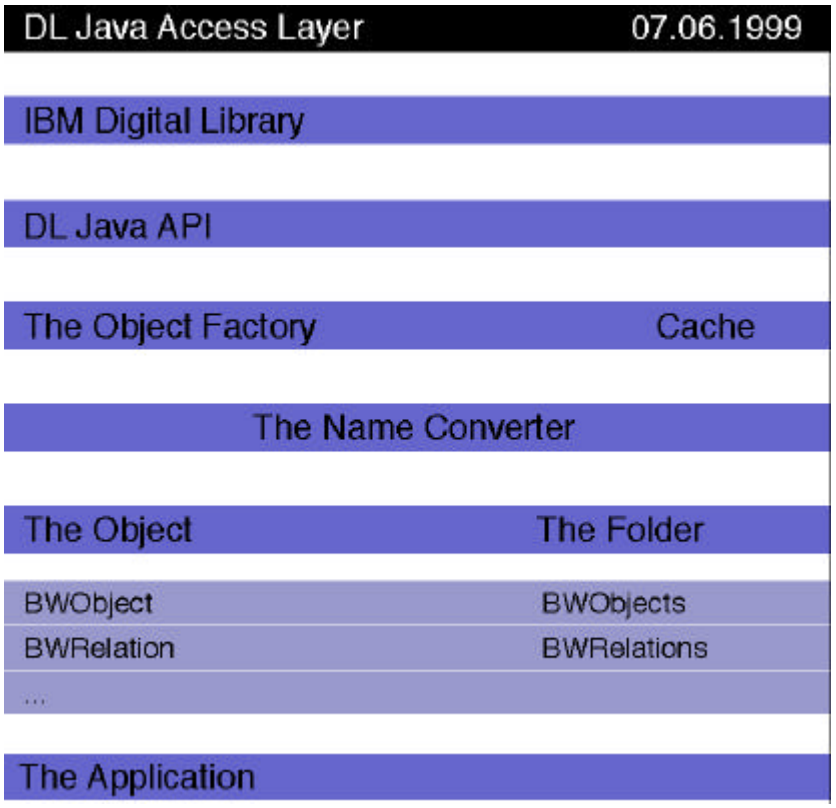
3.2.6 Anwendungsentwicklung

Da für alle Teilprojekte ein gemeinsames Datenmodell verwendet werden sollte, ist dieses zu komplex, um für die Anwendungsentwicklung direkt die Programmierschnittstellen der Digital Library zu verwenden. Um eine einfache Anwendungsentwicklung zu erlauben, wurden deshalb einige Hilfsprogramme sowie ein sogenanntes „Access Layer“ entwickelt.

Um das Datenmodell implementieren und pflegen zu können, mussten zuerst Programme erstellt werden, die Attribute und Indexklassen ausgehend von einer Datenmodellbeschreibung anlegen können; von sich aus bietet die DL nur ein grafisches Interface zur Administration, das zwar gut zur Verwaltung einzelner Indexklassen geeignet ist, bei einem komplexeren Datenmodell jedoch nahezu unbrauchbar ist. Außerdem musste ein Programm zum Laden der Dokumente in die DL entwickelt werden, die mit der DL mitgelieferten Programme sind wiederum für komplexere Datenmodelle kaum einsetzbar.

3.2.6.1 Access Layer

Als Access Layer bezeichnet man eine Softwareschnittstelle zwischen den eigentlichen Applikationen und dem zugrundeliegenden (Datenbank-) System.



Die wesentlichen Gründe für den Einsatz eines Access-Layers sind:

- Kapselung der (technischen) Struktur der Datenbank
  Die tatsächliche Struktur der Daten ist im Allgemeinen für die Datenbank und die Applikation unterschiedlich. Beispielsweise wird die Datenbankstruktur (im Lauf der Zeit) verändert, um verbesserte Performance zu erhalten, Daten werden redundant gespeichert,... Wird ein geeignetes Access Layer verwendet, kann das physikalische Datenmodell (in der Datenbank) geändert werden, ohne dass dies

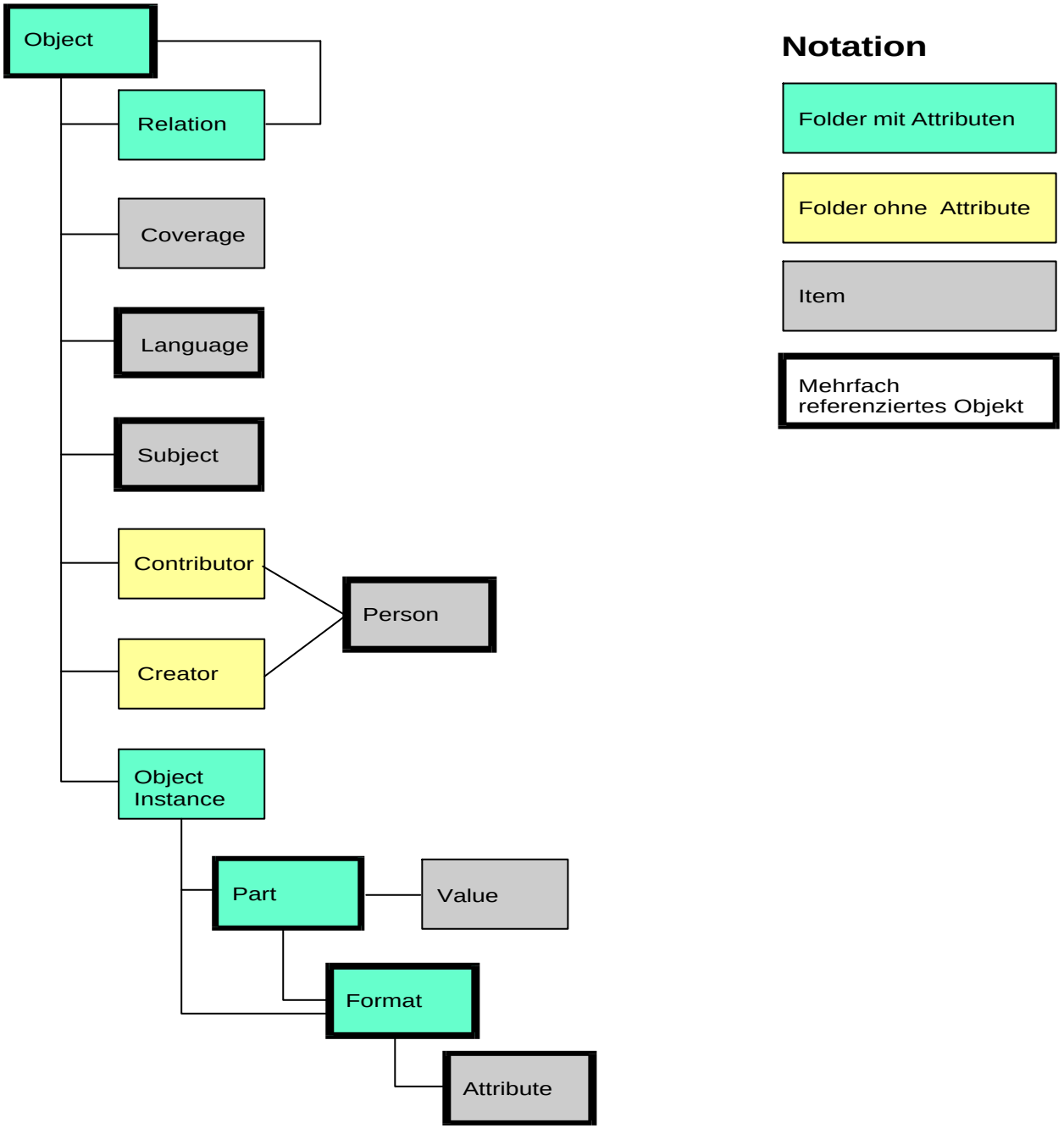
Einfluss auf die Applikationen hat. Nach den bisherigen Erfahrungen mit der IBM-DL werden solche Optimierungen sicherlich notwendig sein (abgesehen davon, dass bereits unser derzeitiges Datenmodell Redundanzen enthält).
→ niedrigerer Wartungsaufwand

- Anforderungen an Applikationsentwickler
  Unabhängig vom verwendeten System erfordert die Programmierung von Datenbanksystemen einen beträchtlichen Umfang an Ausbildung und Erfahrung. ”Naive” Lösungen können zu ernsthaften Performanceproblemen führen. Dies gilt insbesondere für ein System, dessen Schnittstellen nicht standardisiert sind, sondern sich im Lauf der Zeit ändern. Durch ein geeignetes Access Layer wird der datenbankspezifische Code an einer Stelle lokalisiert und nicht über alle Applikationen verteilt.
  → weniger Einarbeitungszeit für Applikationsentwickler

- Fehlerbehandlung
  Jedes (Datenbank-) System verwendet ein spezifisches Modell für die Fehlerbehandlung (spezielle Fehlercodes, Exceptions,...). Ein Access Layer kann die Fehlerbehandlung des zugrundeliegenden Systems in dasjenige der Applikationen transformieren.
  → vereinfachte Fehlerbehandlung in den Applikationen

3.2.6.2 Architektur

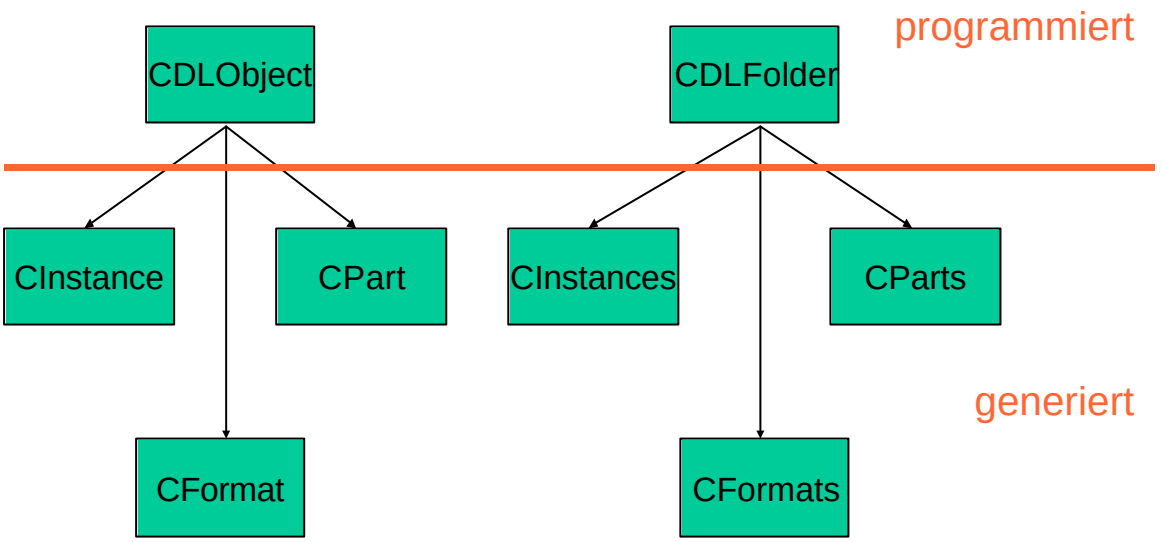
Die Grundidee bei der Architektur war, das DL-Objektmodell auf die Programmiersprache JAVA abzubilden.



Basis für das Access-Layer ist ein JAVA Objekt (theObject), das eine Abstraktion eines DL-Items darstellt und den Zugriff auf einzelne Attribute und Parts der DL erlaubt.

Die jeweiligen JAVA-Klassen, die den tatsächlichen DL-Indexklassen entsprechen, stellen Facade-Objekte dar: Statt der generischen Zugriffsfunktionen (GetAttribute(<name>),...) enthalten diese Klassen für jedes Attribut Methoden zum Zugriff auf Attribute, Parts und Folderinhalte (GetFormat(), SetFormat(), Parts(),...). Diese Klassen werden nicht manuell erstellt, sondern durch ein Generatorprogramm (dlidx) aus der Datenmodellbeschreibung erzeugt.

Access Layer API



LOMI
Universität Ulm

Der Zugriff auf die DL erfolgt über eine zentrale Stelle, ein Singleton Objekt (ObjectFactory). Dieses agiert als Factory für die Erzeugung von Objekten aus der DL. Hier ist der gesamte Zugriffscode gekapselt.

An dieser Stelle sind Optimierungen (z.B. caching,...) implementiert.

Der Zugriff auf die Factory kann auf drei verschiedene Arten erfolgen:

- ItemID
  Die ItemID ist bekannt, die Attributwerte und Parts sollen geladen werden.
- Query auf einer Indexklasse
  Eine Query auf einer bestimmten Indexklasse soll ausgeführt werden. Es werden Objekte der entsprechenden JAVA-Klasse erzeugt (Dabei wird nur die jeweilige ItemID bestimmt, die Attribute,... werden noch nicht geladen).
- Query über mehrere Indexklassen
  Eine Query über beliebige Indexklassen soll ausgeführt werden. Es werden Objekte der Klasse BWObject (oberstes Element der Hierarchie) zurückgeliefert. Dazu wird jeweils eine Query auf jeder betroffenen Indexklasse ausgeführt, danach werden die Ergebnisse über einen Join zusammengeführt. Schließlich werden die zugehörigen BWObject-Elemente gesucht (durch Hinaufwandern in der Hierarchie) und zurückgeliefert.
  Da dieses Vorgehen unter Performance-Gesichtspunkten sehr aufwendig ist, wird es derzeit durch eine direkte (ORACLE/DB-2) Datenbankabfrage ersetzt - der Join wird dann von der Datenbank ausgeführt.

3.2.6.3 Implementierung

In der ersten Version werden die DL-Zugriffe mit dem DL-JAVA API durchgeführt. Dies

wird im Lauf der Zeit durch das C-API bzw. durch direkte Datenbankzugriffe ersetzt, um die Performance zu verbessern.

Es ist relativ einfach, Objekte in der Factory zu cachen, dies ist allerdings nur sinnvoll, wenn alle (schreibenden) Zugriffe über das Access Layer durchgeführt werden, da sonst Konsistenzprobleme auftreten.

3.3 XML-Repräsentation des BWDL Datenmodells

(Heinrich Abele, Annegret Fiebig)

Die XML-Repräsentation des BW-Datenmodells dient der Erfassung und Archivierung sämtlicher Metadaten von Objekten. Ausschlaggebend für die Entscheidung, das Datenmodell in XML-Syntax zu formulieren, war der Wunsch nach einem standardisierten, einheitlichen Format, das auf der einen Seite als Loader-Input für die IBM DB2 DL dienen, darüber hinaus jedoch auch in anderen Funktionszusammenhängen Verwendung finden sollte (Retrieve, Display, Kataloginformation SWB). Da auch andere Projekte das Datenmodell einsetzen wollten (MM-AG), war es notwendig, nicht auf eine bestimmte Applikation bei Erfassung und Speicherung festgelegt zu sein. Eine Metadaten-Datei zu einem Objekt enthält den kompletten Satz an zugehörigen Metadaten, die in XML-Format zum BWDL-Datenmodell konform strukturiert sind.

Im Rahmen der Datenmodell-Entwicklung entstand als eine Repräsentationsform für das Datenmodell eine XML-DTD. Das konzeptuelle Datenmodell bzw. dessen Abbildung auf die DL (s. 3.2.2) wird in XML-Formulierung komplett und ohne Änderungen abgebildet. Die DTD ist nach Funktionen einzelner Attribute modular in mehreren Dateien aufgebaut, so dass bibliothekarische oder technische Vorgaben in gesonderten Listen erweitert oder ggf. geändert werden können. Als weitere Anforderung, die sich aus dem Verhältnis von Konzeption und Einsatz des Datenmodells ergibt, ist es möglich, die DTD zu erweitern (z.B. Rechteverwaltung). Die DTD ist in Tübingen über die Online-Adresse http://tips.zdv.uni-tuebingen.de/BWDL zugänglich.

Metadaten werden entsprechend dieser DTD als XML-validierte Dateien erfasst. Dies ist mit einem XML-Editor, prinzipiell jedoch mit jedem Text-Editor möglich. Derzeit wird eine Eingabemaske als Web-basierte Serverapplikation mit Java-Servlets erarbeitet, die die Eingabe der XML-Daten über HTML-Formulare ermöglicht.

Aus der XML-Datei können diverse Ausgabeformate erzeugt werden: Als derzeit wichtigster Output innerhalb des Landesprojekts müssen das Loader-Eingabeformat für die IBM DB2 Digital Library sowie die Frontdoor-Syntax für den Südwestdeutschen Bibliotheksverbund generiert werden. Für diese Transformationen wurden zunächst DSSSL-Skripte erstellt (Ausführung mit Jade), die demnächst durch serverbasierte Applikationen unter Einbindung von IBM xml4j und der XQL-Engine der GMD ersetzt werden sollen. Als Objektmodelle werden sowohl SAX als auch DOM verwendet. Da die XML-Datei selbst als Part in die DL geladen wurde, kann sie wiederum als Basis für eine Retrieve-Operation genutzt werden (z.B. über SAX-Schnittstelle in Java-Servlets). Ausgabe ist dann der jeweils gesuchte XML-Inhalt, der nach Parsing im HTML-Format erscheint.

Im folgenden wird ein gekürztes Beispiel einer BWDL-Metadatendatei im XML-Format vorgestellt. In der anschließenden, tabellarischen Zuordnung werden die einzelnen Elemente unter Einbeziehung der zugehörigen DTD-Segmente erläutert. Die Reihenfolge der einzelnen Attribute in der Tabelle folgt der Skizzierung des BWDL-Datenmodells in DL-Foldern (s. 3.2.3). Das Datenflussdiagramm unter 3.3.3 zeigt die Metadatendatei im Kontext von Erfassung, Archivierung und Retrieve.

3.3.1 Beispiel für eine BWDL-Metadaten-Datei im XML-Format

Die Zahlenverweise in Klammern beziehen sich auf die Zuordnung zu den BWDL-Attributen im Abschnitt darunter. Anhand der Verweiszahlen kann zwischen der Beispieldatei und den im folgenden Abschnitt beschriebenen Elementen gesprungen werden.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE BWDLmeta SYSTEM
"http://tips.zdv.uni-tuebingen.de/xdtd/bwdlmeta/BWDLMeta.dtd" []>
(9), (10) <BWDLmeta>
<Object ObjectID="UT.Phys-Docs.MSauer19990624" Origin="Tübingen">
(26)   <Creator Role="Author">
(27)     <Person PersonType="person">
           <CompleteName FirstName="Markus" LastName="Sauer" NormName=
           "Sauer.Markus" scheme="noScheme">Sauer, Markus</CompleteName>
           <Address>Fakultät für Physik, Institut für Theoretische Physik
           </Address>
         </Person>
       </Creator>
(12)   <Title>
         <TitleMain>Inklusive Messung von Wirkungsquerschnitten der
         totalen Photoabsorption am Proton und Neutron als Vorbereitung
         für das GDH-Experiment
         </TitleMain>
         <TitleAlternative>Dissertation zur Erlangung des Grades eines
         Doktors der Naturwissenschaften der Fakultät für Physik der
         Eberhard-Karls-Universität zu Tübingen, vorgelegt von Markus
         Sauer aus Reutlingen, 1998
         </TitleAlternative>
       </Title>
(18),(19) <Created scheme="ISO8601">19980219</Created>
(20),(21) <DatePublished scheme="ISO8601">19991031</DatePublished>

(25)   <Subject scheme="noScheme">Photoabsorption</Subject>
       <Subject scheme="noScheme">Gerasimov-Drell-Hearn-
            Summenregel</Subject>
       <Subject scheme="noScheme">MAMI</Subject>
       <Subject scheme="noScheme">Pionphotoproduktion</Subject>
(13)   <Description>
         <DescriptionShort>Experimente zur Untersuchung der Gerasimov--
         Drell--Hearn-- (GDH) Summenregel an den Elektronenbeschleunigern
         MAMI (A2) und ELSA (ELAN--Meßplatz). Pilotexperiment zum Test
         des neu aufgebauten GDH-Detektors am PHOENICS-Meßplatz an
         ELSA.</DescriptionShort>
         <Abstract>Im Rahmen der GDH--Kollaboration finden an den
         Elektronenbeschleunigern MAMI (A2) und ELSA (ELAN--Meßplatz)
         Experimente zur Untersuchung der Gerasimov--Drell--Hearn-- (GDH)
         Summenregel statt. Hierzu werden ...
         </Abstract>
       </Description>
(24)   <Language scheme="NISOZ39.53" language="GER"/>
(5)    <DCType scheme="BSZ">PhdThesis</DCType>
(7)    <ResourceID scheme="noScheme">UT.Phys-Docs.MSauer19990624</ResourceID>
(8)    <Relation Role="dummy">xyz</Relation>
(22)   <Coverage scheme="noScheme">Tübingen</Coverage>
(23)   <Collection>Wissenschaftliche Arbeiten aus Fakultäten der
(15)   Universität Tübingen / 'Graue Literatur'</Collection>
(14)   <Publisher>Universität Tübingen</Publisher>
(11)   <Rightsmanagement>Frei verfügbar</Rightsmanagement>

(16)   <Local>Universität Tübingen / Fakultät für Physik / Institut für
       Theoretische Physik
       </Local>
(2)    <Modified>
         <ModifiedBy>
           <Person PersonType="person">
             <CompleteName FirstName="Clemens" LastName="Harzer"
             NormName="Harzer.Clemens" scheme="noScheme">Harzer,
             Clemens</CompleteName>
             <Address>ZDV der Universität Tübingen</Address>
           </Person>
         </ModifiedBy>
```

```
        <ModifiedAt scheme="ISO8601">19990615</ModifiedAt>
      </Modified>
(3)   <Valid>
(4)     <ValidFrom scheme="ISO8601">19991031</ValidFrom>
        <ValidTo scheme="ISO8601">20100101</ValidTo>
(17)  </Valid>
      <BSZStatus scheme="ISO8601">20000101</BSZStatus>
(1)   <ObjectVersion>1. Version</ObjectVersion>
(6)   <ObjectType>
        <IsDLObject>
(29)      <ObjectInstance InstanceName="UT.Phys-Docs.MSauer19990624_tex"
- (32)    Number="1" Application="application/x-latex" Persistent="Y">

(35)        <Part Number="1" PartName="abstract" Size="2,618">
- (39)       <File name="markus.sauer\TEXT\abstract.tex"
            location="markus.sauer\TEXT\abstract.tex"/>
            <Format Mimetype="application/x-latex" MimeVersion="1.11b">
            </Format>
          </Part>
          <Part Number="17" PartName="auslese" Size="5,467">
          <File name="markus.sauer\Bilder\eps\auslese.eps"
          location="markus.sauer\Bilder\eps\auslese.eps"/>
          <Format Mimetype="application/postscript" MimeVersion="3.0">
          </Format>
          </Part>
          <Part Number="17" PartName="auslese" Size="567">
            <File name="markus.sauer\Bilder\eps\auslese.jpg" location="
            markus.sauer\Bilder\eps\auslese.jpg"/>
(40)        <Format Mimetype="image/jpeg" MimeVersion="5.0">
- (48)       <Attribute Name="Bit-Tiefe für Komponente" Type="int,int,int"
            BWValue="8,8,8">
            </Attribute>
            <Attribute Name="Farbraum" Type="VarChar" BWValue="sRGB">
            </Attribute>
            <Attribute Name="Auflösung/Größe" Type="int,int"
               BWValue="24,36">
              <Comment>Breite, Höhe</Comment>
            </Attribute>
            <Attribute Name="Kompressionsart" Type="VarChar" BWValue="jpeg">
            </Attribute>
            <Attribute Name="Kompressionsrate" Type="VarChar" BWValue="3">
              <Comment>Qualitätsfaktor (1-10; 10 = maximal)</Comment>
            </Attribute>
          </Format>
        </Part>
      </ObjectInstance>
          <ObjectInstance InstanceName="UT.Phys-Docs.MSauer19990624_pdf"
          Number="2" Application="application/pdf" Persistent="Y">
(33),(34)   <Generated Application="Acrobat Distiller 3.01">
            <GeneratedFrom>PS-File</GeneratedFrom>
          </Generated>
            <Part Number="1" PartName="disspdf" Size="1,064,000KB">
              <File name="markus.sauer\pdf\diss.pdf"
          location="markus.sauer\pdf\diss.pdf"/>
              <Format Mimetype="application/pdf" MimeVersion="3.0">
              </Format>
            </Part>
      </ObjectInstance>
      <ObjectInstance InstanceName="UT.Phys-Docs.MSauer19990624_meta"
      Number="4" Application="text/x-xml" Persistent="Y">
        <Part Number="1" PartName="sauer.markus.xml" Size="1,000KB">
          <File name="markus.sauer\meta\sauer.markus.xml"
          location="markus.sauer\meta\sauer.markus.xml"/>
          <Format Mimetype="text/xml" MimeVersion="1.0">
          </Format>
        </Part>
      </ObjectInstance>
    </IsDLObject>
  </ObjectType>
</Object>
</BWDLmeta>
```

3.3.2 Zuordnung zu den BWDL-Attributen

BWDL-Attributname	Abbildung in XML-Syntax

(1) BWObject.ObjectVersion	**<ObjectVersion>**1. Version**</ObjectVersion>**
	
	Zugehöriges DTD-Segment:
	
	`<!ENTITY % ObjectVersion 'INCLUDE' >`
	`<![ %ObjectVersion; [`
	`<!ELEMENT %o.ObjectVersion;`
	`( #PCDATA ) >`
	`<!ATTLIST %o.ObjectVersion;`
	`BWDLForm CDATA 'ObjectVersion'>`
	`]]>`

(2) BWObject.Modified	**<Modified>**
	<ModifiedBy>
	<Person PersonType="person">
	<CompleteName FirstName="Clemens" LastName="Harzer" NormName="Harzer.Clemens" scheme="noScheme">Harzer, Clemens**</CompleteName>**
	<Address>ZDV der Universität Tübingen**</Address>**
	</Person>
	</ModifiedBy>
	<ModifiedAt scheme="ISO8601">19990615**</ModifiedAt>**
	</Modified>
	
	Zugehöriges DTD-Segment:
	
	`<!ENTITY % Modified 'INCLUDE' >`
	`<![ %Modified; [`
	`<!ELEMENT %o.Modified;`
	`( ( %o.ModifiedBy; ) , ( %o.ModifiedAt; )+ )`
	`>`
	`<!ATTLIST %o.Modified;`
	`BWDLForm CDATA 'Modified'`
	`>`
	`]]>`
	
	`<!ENTITY % ModifiedBy 'INCLUDE' >`
	`<![ %ModifiedBy; [`
	`<!ELEMENT %o.ModifiedBy;`
	`( %o.Person; ) >`
	`<!ATTLIST %o.ModifiedBy;`
	`BWDLForm CDATA 'ModifiedBy'>`
	`]]>`
	
	`<!ENTITY % ModifiedAt 'INCLUDE' >`
	`<![ %ModifiedAt; [`
	`<!ELEMENT %o.ModifiedAt;`
	`( #PCDATA ) >`
	`<!ATTLIST %o.ModifiedAt;`
	`%a.date;`
	`BWDLForm CDATA 'ModifiedAt'>`
	`]]>`

(3) BWObject.ValidFrom
(4) BWObject.ValidTo
	<Valid>
	<ValidFrom scheme="ISO8601">19991031**</ValidFrom>**
	<ValidTo scheme="ISO8601">20100101**</ValidTo>**
	</Valid>
	
	Zugehöriges DTD-Segment:
	
	`<!ENTITY % Valid 'INCLUDE' >`
	`<![ %Valid; [`
	`<!ELEMENT %o.Valid;`
	`( ( %o.ValidFrom; )? , ( %o.ValidTo; )? )>`
	`<!ATTLIST %o.Valid;`
	`BWDLForm CDATA 'Valid'>`
	`]]>`
	
	`<!ENTITY % ValidFrom 'INCLUDE' >`
	`<![ %ValidFrom; [`
	`<!ELEMENT %o.ValidFrom;`
	`( #PCDATA ) >`
	`<!ATTLIST %o.ValidFrom;`

	%a.date; BWDLForm CDATA 'ValidFrom'>]]> <!ENTITY % ValidTo 'INCLUDE' > <![%ValidTo; [<!ELEMENT %o.ValidTo; (#PCDATA) > <!ATTLIST %o.ValidTo; %a.date; BWDLForm CDATA 'ValidTo'>]]>
(5) BWObject.DCType	**<DCType scheme="BSZ">**PhdThesis**</DCType>** **Zugehöriges DTD-Segment** <!ENTITY % DCType 'INCLUDE' > <![%DCType; [<!ELEMENT %o.DCType; (#PCDATA) > <!ATTLIST %o.DCType; %a.TypeScheme; BWDLForm CDATA 'DCType' >]]>
(6) BWObject.ObjectType	**<ObjectType>** **<IsDLObject>** **<ObjectInstance …** **</ObjectInstance>** **</IsDLObject>** **</ObjectType>** **Zugehöriges DTD-Segment:** <!ENTITY % ObjectType 'INCLUDE' > <![%ObjectType; [<!ELEMENT %o.ObjectType; (IsDLObject | NoDLObject) > <!ATTLIST %o.ObjectType; BWDLForm CDATA 'ObjectType'>]]> <!ENTITY % IsDLObject 'INCLUDE' > <![%IsDLObject; [<!ELEMENT %o.IsDLObject; (%o.ObjectInstance;)+ > <!ATTLIST %o.IsDLObject; BWDLForm CDATA 'IsDLObject' >]]> <!ENTITY % NoDLObject 'INCLUDE' > <![%NoDLObject; [<!ELEMENT %o.NoDLObject; (%o.ObjectInstance;)* > <!ATTLIST %o.NoDLObject; BWDLForm CDATA 'NoDLObject'>]]>
(7) BWObject.ResourceID **(8)** BWObject.ResourceScheme	**<ResourceID scheme="noScheme">**UT.Phys-Docs.MSauer19990624**</ResourceID>** **Zugehöriges DTD-Segment** <!ENTITY % ResourceID 'INCLUDE' > <![%ResourceID; [<!ELEMENT %o.ResourceID; (#PCDATA) > <!ATTLIST %o.ResourceID; %a.ResourceScheme; BWDLForm CDATA 'ResourceID' >]]>
(9) BWObject.ObjectID **(10)** BWObject.Origin	**<Object ObjectID="UT.Phys-Docs.MSauer19990624"** **Origin="Tübingen">** **…** **</Object>** **Zugehöriges DTD-Segment** <!ENTITY % Object 'INCLUDE' > <![%Object; [<!ELEMENT %o.Object;

	((%o.Creator;)* , (%o.Contributor;)* , (%o.Title;) , (%o.Created;)? , (%o.DatePublished;)? , (%o.Subject;)* , (%o.Description;)? , (%o.Language;)+ , (%o.DCType;) , (%o.ResourceID;) , (%o.Relation;)* , (%o.Coverage;)? , (%o.Collection;)? , (%o.Publisher;) , (%o.Rightsmanagement;)? , (%o.Local;)? , (%o.Modified;)* , (%o.Valid;)? , (%o.BSZStatus;)? , (%o.ObjectVersion;) , (%o.ObjectType;)) > <!ATTLIST %o.Object; %a.origin; ObjectID ID #REQUIRED BWDLForm CDATA 'Object'>]]>
(11) BWObject.Rightsmanagement	**<Rightsmanagement>**Frei verfügbar**</Rightsmanagement>** **Zugehöriges DTD-Segment** <!ENTITY % Rightsmanagement 'INCLUDE' > <![%Rightsmanagement; [<!ELEMENT %o.Rightsmanagement; (#PCDATA) > <!ATTLIST %o.Rightsmanagement; RightsmanagementShort CDATA #IMPLIED BWDLForm CDATA 'Rightsmanagement' >]]>
(12) BWObject.TitleShort	**<Title>** **<TitleMain>**Inklusive Messung von Wirkungsquerschnitten der totalen Photoabsorption am Proton und Neutron als Vorbereitung für das GDH-Experiment **</TitleMain>** **<TitleAlternative>**Dissertation zur Erlangung des Grades eines Doktors der Naturwissenschaften der Fakultät für Physik der Eberhard-Karls-Universität zu Tübingen, vorgelegt von Markus Sauer aus Reutlingen, 1998 **</TitleAlternative>** **</Title>** **Zugehöriges DTD-Segment:** <!ENTITY % Title 'INCLUDE' > <![%Title; [<!ELEMENT %o.Title; ((%o.TitleMain;) , (%o.TitleAlternative;)*)> <!ATTLIST %o.Title; BWDLForm CDATA 'Title'>]]> <!ENTITY % TitleMain 'INCLUDE' > <![%TitleMain; [<!ELEMENT %o.TitleMain; (#PCDATA) > <!ATTLIST %o.TitleMain; BWDLForm CDATA 'TitleMain' >]]> <!ENTITY % TitleAlternative 'INCLUDE' > <![%TitleAlternative; [<!ELEMENT %o.TitleAlternative; (#PCDATA) > <!ATTLIST %o.TitleAlternative; BWDLForm CDATA 'TitleAlternative' >]]>
(13) BWObject.DescriptionShort	**<Description>** **<DescriptionShort>**Experimente zur Untersuchung

	der Gerasimov--Drell--Hearn-- (GDH) Summenregel an den Elektronenbeschleunigern MAMI (A2) und ELSA (ELAN--Meßplatz). Pilotexperiment zum Test des neu aufgebauten GDH-Detektors am PHOENICS-Meßplatz an ELSA. **</DescriptionShort>** **<Abstract>**Im Rahmen der GDH--Kollaboration finden an den Elektronenbeschleunigern MAMI (A2) und ELSA (ELAN--Meßplatz) Experimente zur Untersuchung der Gerasimov--Drell--Hearn-- (GDH) Summenregel statt. Hierzu werden ... **</Abstract>** **</Description>** **Zugehöriges DTD-Segment:** <!ENTITY % Description 'INCLUDE' > <![%Description; [<!ELEMENT %o.Description; ((%o.DescriptionShort;) , (%o.Abstract;))> <!ATTLIST %o.Description; BWDLForm CDATA 'Description' >]]> <!ENTITY % DescriptionShort 'INCLUDE' > <![%DescriptionShort; [<!ELEMENT %o.DescriptionShort; (#PCDATA) > <!ATTLIST %o.DescriptionShort; BWDLForm CDATA 'DescriptionShort' >]]>
(14) BWObject.PublisherShort	**<Publisher>**Universität Tübingen**</Publisher>** **Zugehöriges DTD-Segment:** <!ENTITY % Publisher 'INCLUDE' > <![%Publisher; [<!ELEMENT %o.Publisher; (#PCDATA) > <!ATTLIST %o.Publisher; BWDLForm CDATA 'Publisher' >]]>
(15) BWObject.Collection	**<Collection>**Wissenschaftliche Arbeiten aus Fakultäten der Universität Tübingen / 'Graue Literatur'**</Collection>** **Zugehöriges DTD-Segment:** <!ENTITY % Collection 'INCLUDE' > <![%Collection; [<!ELEMENT %o.Collection; (#PCDATA) > <!ATTLIST %o.Collection; BWDLForm CDATA 'Collection' >]]>
(16) BWObject.Local	**<Local>**Universität Tübingen / Fakultät für Physik / Institut für Theoretische Physik **</Local>** **Zugehöriges DTD-Segment:** <!ENTITY % Local 'INCLUDE' > <![%Local; [<!ELEMENT %o.Local; (#PCDATA) > <!ATTLIST %o.Local; BWDLForm CDATA 'Local' >]]>
(17) BWObject.BSZStatus	**<BSZStatus scheme="ISO8601">**20000101**</BSZStatus>** **Zugehöriges DTD-Segment** <!ENTITY % BSZStatus 'INCLUDE' > <![%BSZStatus; [<!ELEMENT %o.BSZStatus; (#PCDATA) > <!ATTLIST %o.BSZStatus;

	%a.date; BWDLForm CDATA 'BSZStatus'>]]>
(18) BWObject.DateCreated / **(19)** BWObject.DateCreatedString	**<Created scheme="ISO8601">**19980219**</Created>** **Zugehöriges DTD-Segment:** <!ENTITY % Created 'INCLUDE' > <![%Created; [<!ELEMENT %o.Created; (#PCDATA) > <!ATTLIST %o.Created; %a.date; Role CDATA 'original' BWDLForm CDATA 'Created' >]]>
(20) BWObject.DatePublished / **(21)** BWObject.DatePublishedString	**<DatePublished scheme="ISO8601">**19991031**</DatePublished>** **Zugehöriges DTD-Segment:** <!ENTITY % DatePublished 'INCLUDE' > <![%DatePublished; [<!ELEMENT %o.DatePublished; (#PCDATA) > <!ATTLIST %o.DatePublished; %a.date; BWDLForm CDATA 'DatePublished'>]]>
(22) BWRelation	**<Relation Role="dummy">**xyz**</Relation>** **Zugehöriges DTD-Segment:** <!ENTITY % Relation 'INCLUDE' > <![%Relation; [<!ELEMENT %o.Relation; (#PCDATA) > <!ATTLIST %o.Relation; Role CDATA #REQUIRED BWDLForm CDATA 'Relation' >]]>
(23) BWCoverage	**<Coverage scheme="noScheme">**Tübingen**</Coverage>** **Zugehöriges DTD-Segment** <!ENTITY % Coverage 'INCLUDE' > <![%Coverage; [<!ELEMENT %o.Coverage; (#PCDATA) > <!ATTLIST %o.Coverage; %a.CoverageScheme; BWDLForm CDATA 'Coverage' >]]>
(24) BWLanguage	**<Language scheme="NISOZ39.53" language="GER"/>** **Zugehöriges DTD-Segment:** <!ENTITY % Language 'INCLUDE' > <![%Language; [<!ELEMENT %o.Language; EMPTY > <!ATTLIST %o.Language; %a.language; BWDLForm CDATA 'language' >]]>
(25) BWSubject	**<Subject scheme="noScheme">**Photoabsorption**</Subject>** **<Subject scheme="noScheme">**Gerasimov-Drell-Hearn-Summenregel**</Subject>** **<Subject scheme="noScheme">**MAMI**</Subject>** **<Subject scheme="noScheme">**Pionphotoproduktion**</Subject>** **Zugehöriges DTD-Segment:** <!ENTITY % Subject 'INCLUDE' > <![%Subject; [

<table>
<tr><td></td><td>

```
<!ELEMENT %o.Subject;
        ( #PCDATA )                 >
<!ATTLIST %o.Subject;
                        %a.SubjectScheme;
        BWDLForm        CDATA
        'Subject'       >
]]>
```

</td></tr>
<tr><td>(26) BWCreator /
(27) BWPerson</td><td>

```
<Creator Role="Author">
   <Person PersonType="person">
     <CompleteName FirstName="Markus"
     LastName="Sauer" NormName="Sauer.Markus"
     scheme="noScheme">Sauer, Markus
     </CompleteName>
     <Address>Fakultät für Physik, Institut für
     Theoretische Physik
     </Address>
   </Person>
</Creator>
```

Zugehöriges DTD-Segment:

```
<!ENTITY % Creator 'INCLUDE' >
<![ %Creator; [
<!ELEMENT %o.Creator;
        ( %o.Person; )              >
<!ATTLIST %o.Creator;
        Role            CDATA   #REQUIRED
        BWDLForm        CDATA   'Creator'     >
]]>
```

```
<!ENTITY % Person 'INCLUDE' >
<![ %Person; [
<!ELEMENT %o.Person;
        ( ( %o.CompleteName; ) , ( %o.Address; )? )>
<!ATTLIST %o.Person;
        PersonType      ( person | corporation )
        #REQUIRED
        BWDLForm        CDATA           'Person'
        >
]]>
```

```
<!ENTITY % CompleteName 'INCLUDE' >
<![ %CompleteName; [
<!ELEMENT %o.CompleteName;
        ( #PCDATA )                 >
<!ATTLIST %o.CompleteName;
        FirstName       CDATA   #IMPLIED
        LastName        CDATA   #IMPLIED
        NormName        CDATA   #IMPLIED
                %a.PersonScheme;
        BWDLForm        CDATA   'CompleteName' >
]]>
```

```
<!ENTITY % Address 'INCLUDE' >
<![ %Address; [
<!ELEMENT %o.Address;
        ( #PCDATA )                 >
<!ATTLIST %o.Address;
        BWDLForm        CDATA   'Address'     >
]]>
```

</td></tr>
<tr><td>(28) BWContributor</td><td>

Dateibeispiel wäre analog zu <Creator>

Zugehöriges DTD-Segment:

```
<!ENTITY % Contributor 'INCLUDE' >
<![ %Contributor; [
<!ELEMENT %o.Contributor;
        ( %o.Person; )              >
<!ATTLIST %o.Contributor;
        Role            CDATA           #IMPLIED
        BWDLForm        CDATA           'Contributor' >
]]>
```

</td></tr>
<tr><td>(29)BWObjectInstance.InstanceNumber
(30) BWObjectInstance.Persistent
(31)</td><td>

```
<ObjectInstance InstanceName="UT.Phys-
Docs.MSauer19990624_tex" Number="1"
Application="application/x-latex" Persistent="Y">
 …
</ObjectInstance>
```

</td></tr>
</table>

BWObjectInstance.InstanceName **(32)** BWObjectInstance.Application	**Zugehöriges DTD-Segment:** `<!ENTITY % ObjectInstance 'INCLUDE' > <![ %ObjectInstance; [ <!ELEMENT %o.ObjectInstance; ( ( %o.Generated; )? , ( %o.Part; )+ ) > <!ATTLIST %o.ObjectInstance; %a.application; %a.number; InstanceName CDATA #REQUIRED Persistent ( Y | N ) #REQUIRED BWDLForm CDATA 'ObjectInstance' > ]]>`
(33) BWObjectInstance.GeneratedBy **(34)** BWObjectInstance.GeneratedFrom	`<Generated Application="Acrobat Distiller 3.01">   <GeneratedFrom>PS-File</GeneratedFrom> </Generated>` **Zugehöriges DTD-Segment:** `<!ENTITY % Generated 'INCLUDE' > <![ %Generated; [ <!ELEMENT %o.Generated; ( %o.GeneratedFrom; )? > <!ATTLIST %o.Generated; %a.GeneratedBy; BWDLForm CDATA 'Generated' > ]]>` `<!ENTITY % GeneratedFrom 'INCLUDE' > <![ %GeneratedFrom; [ <!ELEMENT %o.GeneratedFrom; ( #PCDATA ) > <!ATTLIST %o.GeneratedFrom; ResourceId CDATA #IMPLIED BWDLForm CDATA 'GeneratedFrom' > ]]>`
(35) BWPart.Application **(36)** BWPart.PartSize **(37)** BWPart.PartNumber **(38)** BWPart.PartName **(39)** BWPart.PartIndexer	`<Part Number="17" PartName="auslese" Size="567">   <File name="markus.sauer\Bilder\eps\auslese.jpg"   location=" markus.sauer\Bilder\eps\auslese.jpg"/>   … </Part>` **Zugehöriges DTD-Segment:** `<!ENTITY % Part 'INCLUDE' > <![ %Part; [ <!ELEMENT %o.Part; ( ( %o.File; ) , ( %o.Format; ) ) > <!ATTLIST %o.Part; %a.number; %a.application; PartName CDATA #REQUIRED PartIndexer CDATA #IMPLIED Size CDATA #REQUIRED BWDLForm CDATA 'Part' > ]]>`

(40) BWFormat.MimeType
(41) BWFormat.MimeVersion
(42) BWFormat.FormatComment
(43) BWFormat.Application
(44) BWValue.AttributeName
(45) BWValue.AttributeValue
(46) BWAttribute.AttributeName
(47) BWValue.AttributeType
(48) BWValue.AttributeComment
	`<Format Mimetype="image/jpeg" MimeVersion="5.0"
	Application="iexplore.exe">
	  <Attribute Name="Bit-Tiefe für Komponente"
	  Type="int,int,int" BWValue="8,8,8">
	  </Attribute>
	  <Attribute Name="Farbraum" Type="VarChar"
	  BWValue="sRGB">
	  </Attribute>
	  <Attribute Name="Auflösung/Größe"
	  Type="int,int" BWValue="24,36">
	  <Comment>Breite, Höhe</Comment>
	  </Attribute>
	  <Attribute Name="Kompressionsart"
	  Type="VarChar" BWValue="jpeg">
	  </Attribute>
	  <Attribute Name="Kompressionsrate"
	  Type="VarChar" BWValue="3">
	  <Comment>Qualitätsfaktor (1-10; 10 =`

```
            maximal)</Comment>
        </Attribute>
</Format>
```

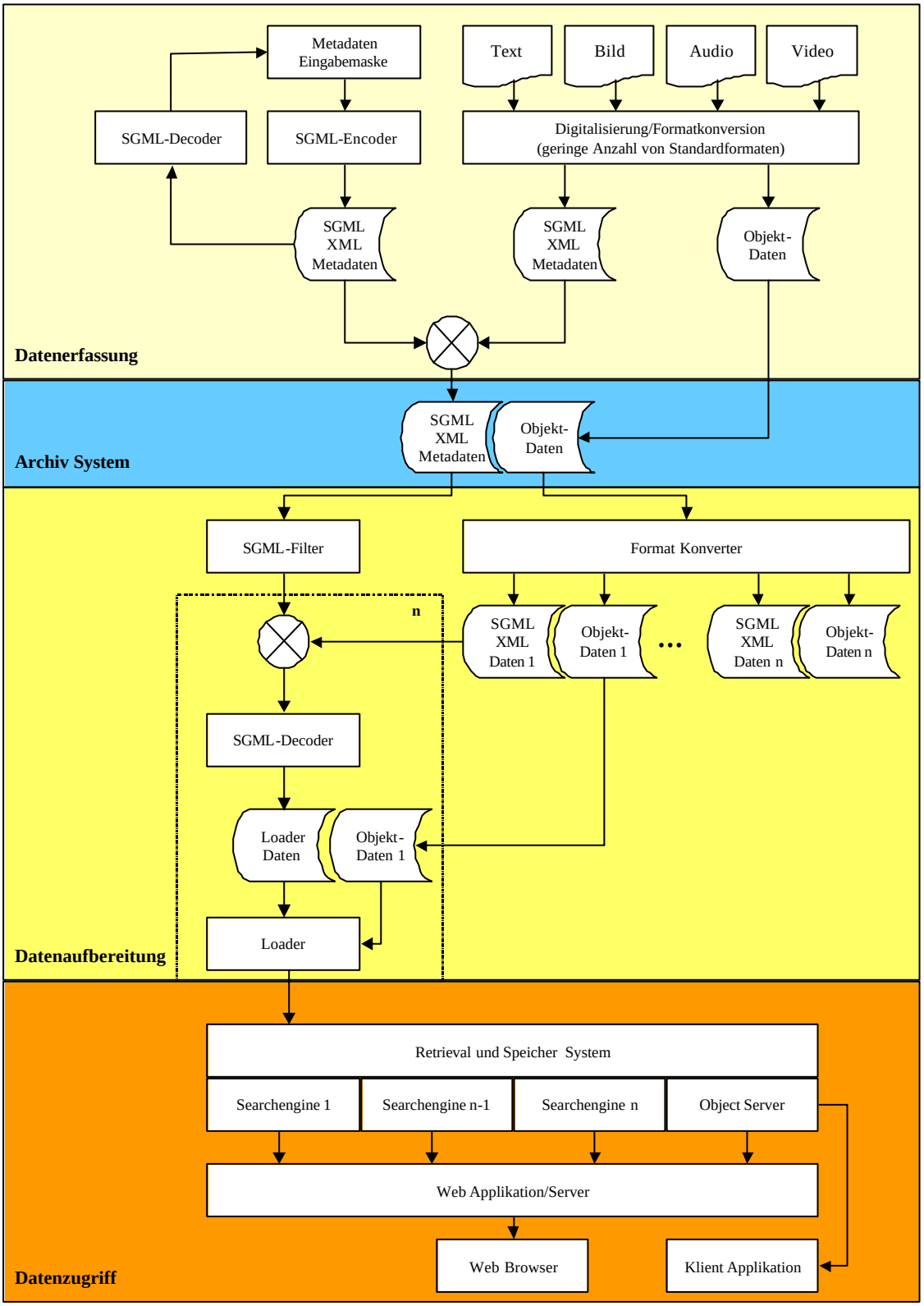
Zugehöriges DTD-Segment

```
<!ENTITY % Format 'INCLUDE' >
<![ %Format;  [
<!ELEMENT %o.Format;
       ( ( %o.Attribute; )* , ( %o.Comment; )? )
       >
<!ATTLIST %o.Format;
                     %a.mime;
                     %a.application;
       ContentClass  CDATA  #IMPLIED
       BWDLForm      CDATA  'format'      >
]]>
<!ENTITY % Attribute 'INCLUDE' >
<![ %Attribute;       [
<!ELEMENT %o.Attribute;
       ( %o.Comment; )?                    >
<!ATTLIST %o.Attribute;
       Name   CDATA  #REQUIRED
       Type   CDATA  #REQUIRED
       BWValueCDATA  #REQUIRED     >
]]>
<!ENTITY % Comment 'INCLUDE' >
<![ %Comment;  [
<!ELEMENT %o.Comment;
       ( #PCDATA )                         >
]]>
```

Vorgegebene Attribute werden als Listen in externen DTD-Dateien geführt. Dies betrifft Schema-Angaben (Datumsschema, Coverage-Schema, ResourceID-Schema), Mimetypes, Applikationen sowie projektspezifische Angaben wie Angabe der einzelnen Sites (KarlsruheRZ, KarlsruheUB, Konstanz, Tübingen, Ulm unter *Origin* des Objekts).

Die Angabe *Filename* und *Filelocation* muss für den Ladevorgang im XML-File enthalten sein; in der DL-Modellierung spielt diese Angabe keine Rolle.

3.3.3 XML-Repräsentation der BWDL-Metadaten im Kontext des Datenflusses zwischen Erfassung, Archivierung und Retrieval



3.4 Loader für die Digital Library

(Günter Radestock)

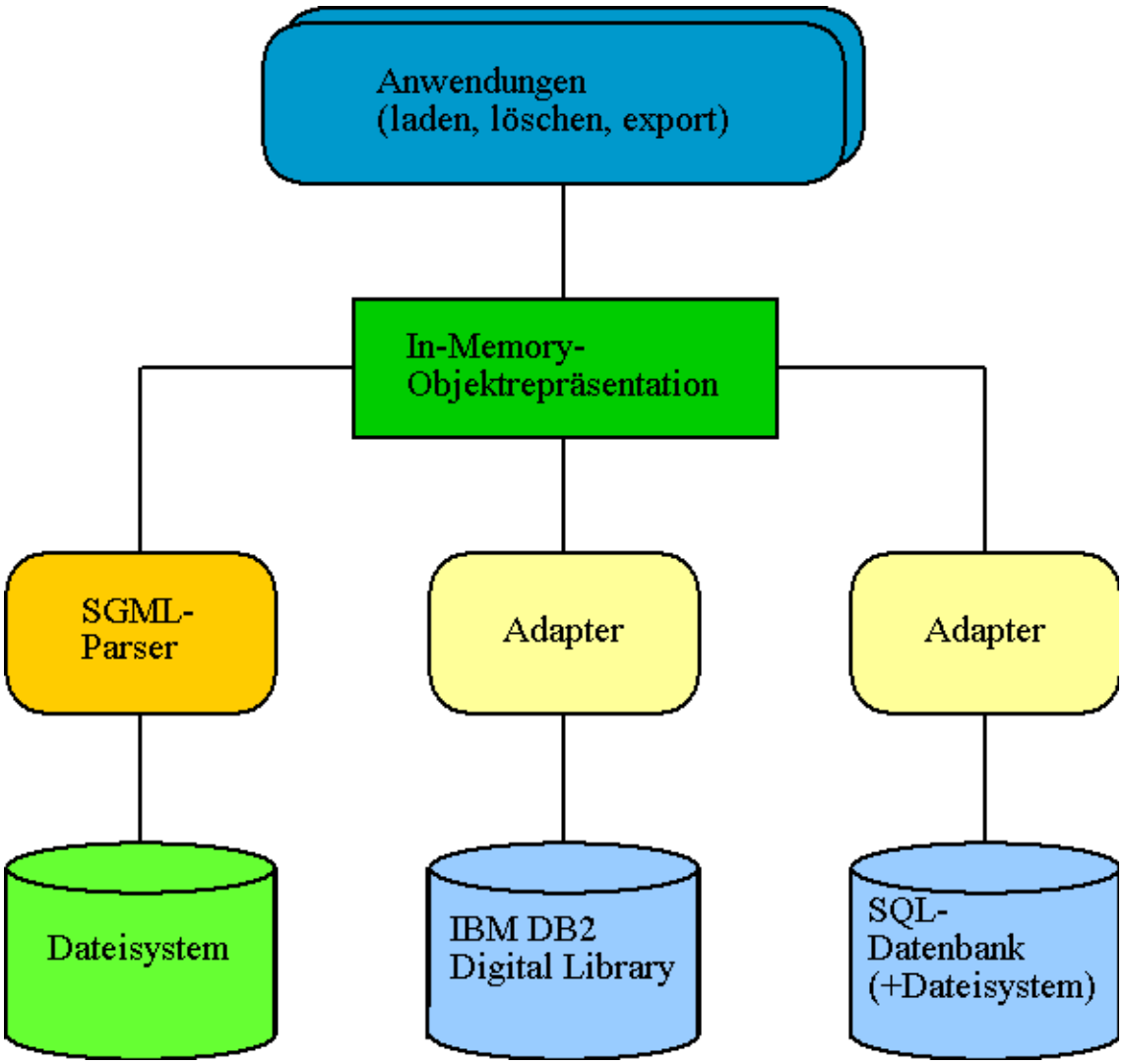
3.4.1 Einleitung

Das Ladeprogramm für die Digital Library (kurz:Loader) ist ein projektweit eingesetztes Software-Tool, mit dem die beschreibenden Metadaten sowie die zugehörigen Multimedia-Dateien eines BW-Objekts in die Digital Library eingebracht werden können. Das Programm interpretiert dabei die in einem SGML/XML-artigen Format vorliegenden Metadaten und bildet sie gemäß dem gemeinsam entwickelten Datenmodell auf die Folderstruktur der Digital Library ab. Zusätzlich wird der zur Volltextsuche benötige Superpart dynamisch erzeugt und zur Indizierung in die Queue des Texminer, der Komponente zur Volltextsuche, gestellt.

Neben dem eigentlichen Ladeprogramm wurden Werkzeuge zum Export von Objekten aus der Digital Library sowie zum Löschen von Objekten aus der Digital Library entwickelt. Mit Hilfe dieser Werkzeuge können Objekte von einer DL-Installation zu einer anderen transportiert werden, durch Export aus der Quell-DL und Import in die Ziel-DL.

3.4.2 Architektur

Der BW-Loader ist als mehrschichtige Architektur entworfen und implementiert (s. Abb.).



Auf unterster Ebene sind die Objekte des gemeinsamen Datenmodells entweder als Beschreibungsdatei zusammen mit Part-Dateien im Dateisystem oder in der Digital Library gespeichert. Ein Adapter oder ein SGML-Parser transformiert das gespeicherte Objekt zunächst in eine in-memory-Repräsentation, um es danach auf ähnliche Weise wieder auszugeben (beim Laden in die DL, beim Export in das Dateisystem).

Neben dem Adapter für die IBM DB2 Digital Library wurde zum Testen des Systems ein Adapter für SQL-Datenbanken prototypisch implementiert.

Zur Implementierung wurde die objektorientierte Skriptsprache Python verwendet. Python steht plattformübergreifend und kostenlos zur Verfügung. Der Loader konnte durch die vorhandene Erfahrung im Umgang mit Python und die bei Python mitgelieferten Klassen (SGML-Parser) zügig fertiggestellt werden und ist durch den objektorientierten Aufbau leicht änderbar. Der Adapter zum Zugriff auf die IBM DB2 Digital Library wurde zum größten Teil in C++ realisiert.

3.4.3 Externe Repräsentation von Objekten

Außerhalb der DL werden die Objekte der Baden-Württemberg-DL in Dateien gespeichert. Zu jedem Objekt existiert eine Beschreibungsdatei mit Metadaten (Titelangabe, Autoren, Bearbeitungsvermerke usw.) und Verweise (Dateinamen) zu den zum Objekt gehörenden Binärdaten. Der Loader verwendet Beschreibungsdateien in einem SGML-ähnlichen Format, das mit einem in Tübingen entwickelten Werkzeug automatisch aus der XML-Notation des BW-Datenmodells generiert werden kann.

Eine Loaderdatei enthält Beschreibungen von einem oder mehreren Objekten, die mit dem Element <bwdoc> definiert werden. Neben Objekten können auch Personen (<person>) und Formate mit ihren formatspezifischen Attributen (<format>) einzeln beschrieben werden.

```
<bwdoc id=1>

<metadata type=dl>

     <Creator persontype=natuerlich>
           <firstname>Martin</firstname>
           <lastname>Br&uuml;hl</lastname>
           <completename>Br&uuml;hl, Martin</completename>
           <address>Institut für Praktische Mathematik der
           Universit&auml;t Karlsruhe</address>
     </creator>

     <publisher>Universit&auml;t Karlsruhe</publisher>

</metadata>

<instance name="Online lesen" application="html">
     <part name="text" file="/extra/vvv/1999/mathematik/1/1.text"
     index=contents>
           <attribute name="for-index-only" value="1">
     <part>
</instance>

</bwdoc>
```

Beispiel für eine Loader-Eingabedatei

Die im Element <metadata> eingebetteten Metadaten können entweder in einer eng an das BW-Datenmodell angelehnten Syntax (siehe Bsp.) oder in der Syntax zur Einbettung von Dublin-Core-Metadaten in HTML-Seiten angegeben werden. Dadurch wird das Übernehmen von Metadaten aus Dublin-Core-Quellen wie dem Karlsruher Volltextarchiv (EVA) ohne aufwendige Konvertierung möglich.

3.4.4 Konsistenz des Datenmodells

Beim Ändern (Einlesen, Löschen) der Objekte in der IBM DB2 Digital Library muss sichergestellt werden, dass die Digital Library in einem konsistenten Zustand bleibt, auch dann, wenn Fehler in den Eingabedaten oder anderswo auftreten. Um die Konsistenzerhaltung zu gewährleisten, wurden verschiedene Maßnahmen ergriffen.

Bevor Objekte in die Digital Library geschrieben werden, werden zunächst die kompletten Metadaten (aus der Beschreibungsdatei) eingelesen und auf syntaktische Korrektheit und Vollständigkeit überprüft. Ebenso wird überprüft, ob alle zum Objekt gehörenden Dateien vorhanden sind und gelesen werden können. Tritt an dieser Stelle ein Fehler auf, so wird eine Fehlermeldung ausgegeben und der Ladevorgang abgebrochen. Der Loader kann auch verwendet werden, um die Integrität von Objekten zu testen, ohne diese zu laden.

Die Änderungen in der Digital Library werden in Transaktionen gekapselt, so dass im Fehlerfall teilweise erfolgte Änderungen rückgängig gemacht werden können. Der Loader stellt beim Laden, Ändern und Löschen von Objekten sicher, dass jedes Objekt dem gemeinsamen Datenmodell entspricht, dass die vereinbarte Folderstruktur auf- bzw. abgebaut wird, dass nicht mehrere Items mit gleicher Kennung existieren usw.

Die in der DL-gespeicherten Objekte werden nach dem Laden mit dem TextMiner

indexiert. Um diese Indexierung zu steuern, wird in der Eingabedatei der Objekte vermerkt, welche Teile in welchen Index aufgenommen werden sollen. Der Loader stellt beim Laden die entsprechenden "Parts" in Indexierungswarteschlangen, die bei einem anschließenden Indexierungslauf zur Erstellung der gewünschten Volltextindizes führen. Neben den Parts werden auch Attribute der Metadaten in einen Volltextindex aufgenommen. Dadurch ist die gleichzeitige Suche nach diesen Attributen und Stichworten im Volltext ohne die manchmal langsame parametrische Suche in mehreren Indexklassen der IBM DB2 Digital Library möglich.

3.4.5 Plattformübergreifende Verfügbarkeit

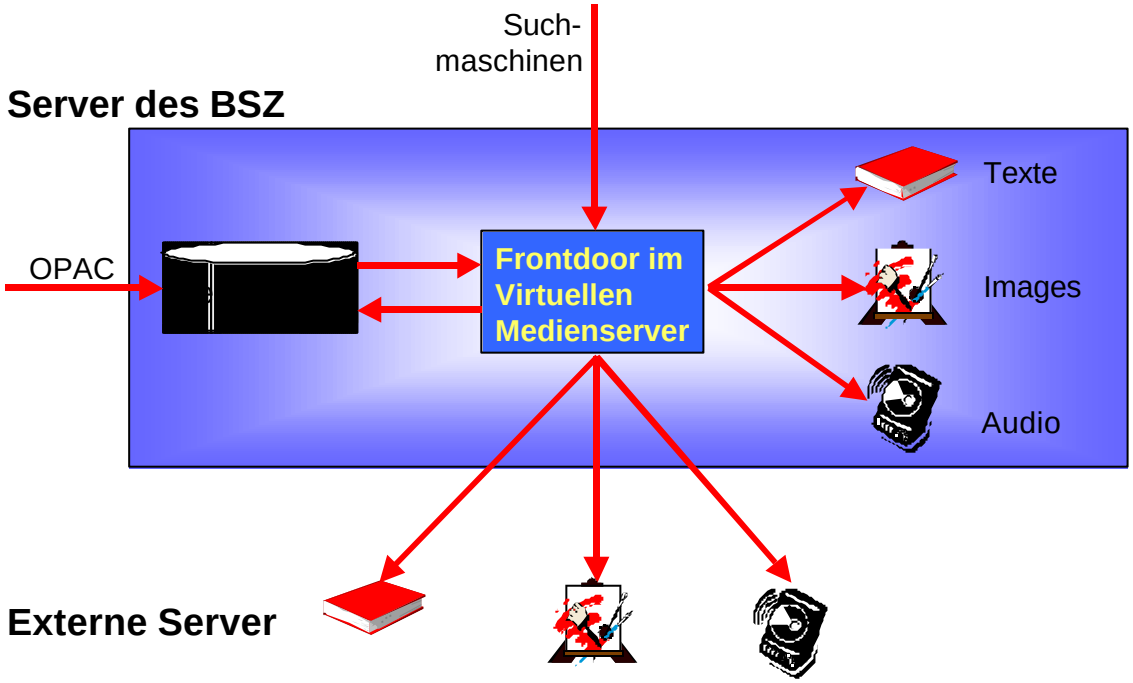
Der Loader steht in Versionen für AIX und Windows NT zur Verfügung; er wurde unter AIX entwickelt. Zur Portierung auf Windows NT musste nur der Adapter portiert werden, der ein Python-API zur Manipulation von IBM DB2 Digital Library Items implementiert. Die zum Loader gehörenden Werkzeuge zum Laden, Löschen und Exportieren von Objekten können als Kommandozeilenprogramme oder als Befehle unter einer grafischen Oberfläche verwendet werden.

3.5 Schnittstelle zum Virtuellen Medienserver des BSZ

(BSZ)

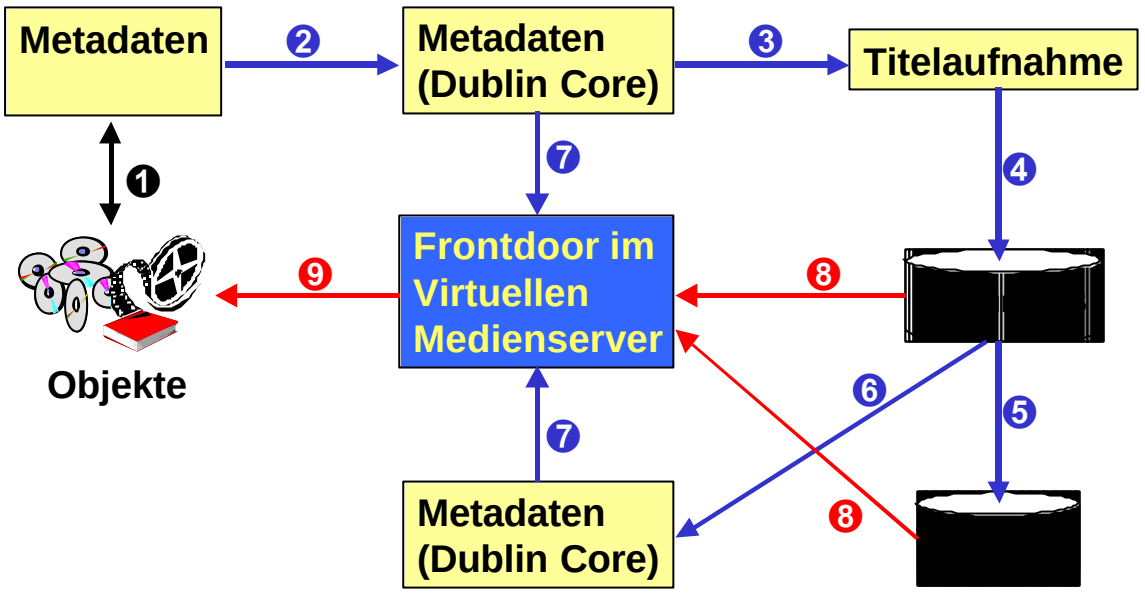
Der Virtuelle Medienserver des BSZ ermöglicht über die Verbunddatenbank den einheitlichen Zugang zu den auf verteilten Servern an verschiedenen Hochschulbibliotheken im Südwestdeutschen Bibliotheksverbund (SWB) gespeicherten Publikationen. Darüberhinaus wird durch Erschließungsdateien ("Frontdoors"), die mit Metadaten versehen sind und auf dem WWW-Server des BSZ vorgehalten werden, gewährleistet, dass die Objekte auch über Suchmaschinen recherchierbar sind.

Der Virtuelle Medienserver des BSZ: Schematische Darstellung



Soll ein Objekt im Virtuellen Medienserver nachgewiesen werden, muss sowohl eine Titelaufnahme in der Verbunddatenbank des SWB angelegt als auch eine Frontdoor für den WWW-Server des BSZ erzeugt werden. Da die Mehrzahl der hierfür notwendigen Informationen (Verfasser, Titel, Herausgeber, Publikationsjahr etc.) bereits als das Objekt beschreibende Metadaten im betreffenden Publikations-System vorliegen, liegt es nahe, diese auch für Titelaufnahmen und Frontdoors weiterzunutzen und so den notwendigen Arbeitsaufwand zu minimieren.

Anhand der folgenden Grafik soll der gesamte Workflow veranschaulicht werden:



1. In der Digital Library sind die Objekte mit den zu diesen gehörenden Metadaten abgelegt.
2. Die Metadaten werden aus dem System exportiert und als HTML-Dateien gemäß Dublin-Core-Spezifikation an das BSZ geliefert.
3. Aus den gelieferten Daten werden Titelaufnahmen im Externformat der Verbunddatenbank generiert.
4. Die Titelaufnahmen werden in die Verbunddatenbank importiert. Durch die Bibliotheken erfolgen eventuell notwendige Korrekturen ("Hochkatalogisieren"), insbesondere im Bereich der Personennamen, Körperschaften und Schlagworte, die an Normdateien abgeglichen werden müssen (Personennamendatei (PND), Gemeinsame Körperschaftsdatei (GKD), Schlagwortnormdatei (SWD)).
5. Nach erfolgter Korrektur werden die Titelaufnahmen von den Bibliotheken für den weiteren Verfahrensablauf freigegeben und können in die lokalen Datenbanken importiert werden.
6. Zudem werden die korrigierten Titelaufnahmen aus der Verbunddatenbank exportiert und in HTML-Dateien (Dublin Core) konvertiert.
7. Aus den in Schritt 2 gelieferten und den in Schritt 6 erzeugten Metadaten werden Frontdoors generiert, die auf dem WWW-Server des BSZ abgelegt werden.
8. Nachdem die URL in der Titelaufnahme eingetragen wurde, kann aus OPACs heraus auf die Frontdoor zugegriffen werden.
9. Von der Frontdoor, auf die der Benutzer über OPACs oder Suchmaschinen gelangte, erfolgt der Zugriff auf das Objekt in der Digital Library.

Dieses Verfahren befindet sich derzeit in Kooperation mit verschiedenen Projekten und Einrichtungen (Volltextserver der Universität Ulm (VTS), Online Publikationsverbund der Universität Stuttgart (OPUS) u.a.) in der Erprobung und soll auch für das BW/DL Projekt eingesetzt werden.

Derzeit müssen alle Verfahren offline durchgeführt werden. Für die Zukunft muss eine Lösung erarbeitet werden, die eine engere Verknüpfung zwischen den in der DL

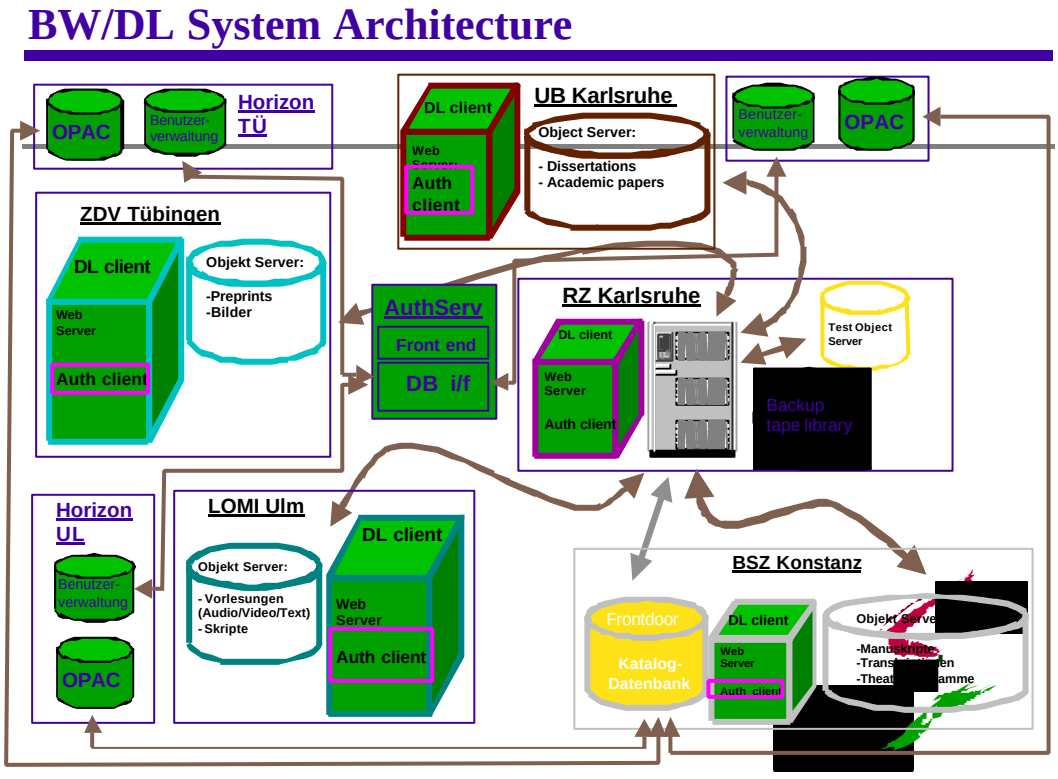
vorgehaltenen Metadaten und den Titelaufnahmen in der Verbunddatenbank vorsieht. So kann erreicht werden, dass Korrekturen in der Datenbank automatisch auch in der DL durchgeführt werden können, was eine erhebliche qualitative Verbesserung der Metadaten mit sich bringen wird und den zeitliche Verzug zwischen der Aufnahme eines Objekts in die DL und dessen Nachweis in der Verbunddatenbank minimiert.

3.6 Netzwerk-Struktur

(Sebastian Goeser)

Die Digitale Bibliothek Baden-Württemberg vereinigt in ihrer Topologie verschiedenartige Anwendungen und Kollektionen auf der Basis einer verteilten Implementierung des BW/DL- Datenmodells. Die Grundbestandteile dieses verteilten Systems, nämlich ein singulärer Library-Server, mehrere Objekt-Server und mehrere Klienten und WebServer ergeben sich aus der oben beschriebenen Dreiecksarchitektur des Produkts IBM DB2 Digital Library (s. 3.1).

Die folgende Figur zeigt die gesamte Topologie des Systems. Abgebildet sind die Basiskomponenten der Digital Library-Installationen in den einzelnen Teilprojekten sowie als übergreifende Systembestandteile erstens die Authentifizierung über einen Authentifizierungsserver und zweitens die Anbindung der BW/DL an den Katalog des Südwestdeutschen Bibliotheksverbundes (SWB.-Katalog) und damit an die OPACs der beteiligten Universitätsbibliotheken.



3.6.1 Der zentrale Library Server am Rechenzentrum der Universität Karlsruhe

(Werner Vogelpohl)

Das Rechenzentrum der Universität Karlsruhe (TH) stellt im Rahmen des Landes-Pilotprojektes "IBM Digital Library" die Ressourcen für die zentrale Verwaltung der Metadaten aller Einzelinstitutionen zur Verfügung.

Den Schwerpunkt der Arbeit am RZ KA bildet der Betrieb des zentralen Library Servers. Angeschlossen an ihn sind zwei lokale Object Server, auf denen digitaliserte Daten aus dem Bereich der Unversität Karlsruhe (im Moment Daten der UB Karlsruhe) gespeichert sind. Die Suche in allen Kollektionen der Projektpartner orientiert sich an den Vorgaben des gemeinsam erarbeiteten Datenmodells, sie wird über einen Webserver (als Client des Library Servers) ausgelöst und eröffnet somit einem großen Benutzerkreis den Zugang.

Verteilten Charakter gewinnt das System durch die plattformunabhängige Anbindung weiterer Object Server und Client Server der Projektpartner über das Forschungsnetz im Lande Baden-Württemberg.

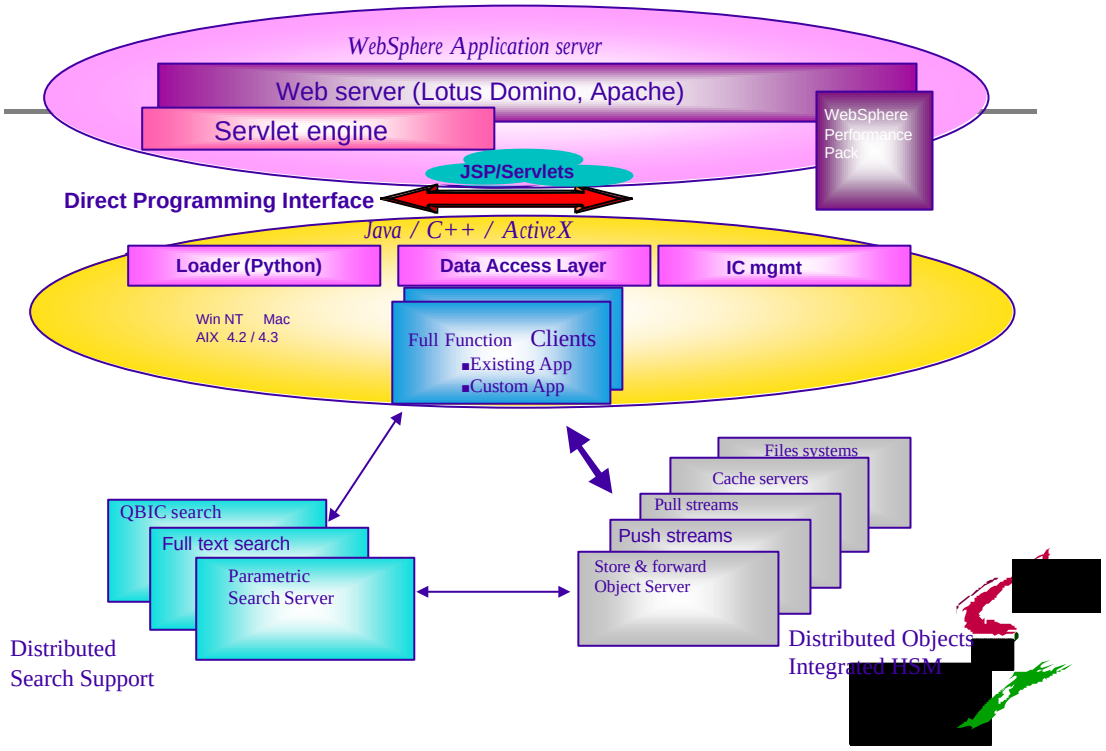
Bedingt durch den verteilten Charakter des BW DL-Systems fließt ein erheblicher Teil der vom RZ KA erbrachten Arbeitsleistung in administrative Aufgaben: Überwachung des Zusammenspiels aller Serverkomponenten, Problemmanagement, Performance-messungen des Gesamtsystems, Datensicherung, Einspielen von Daten, Aktualisierung von Softwarekomponenten.

3.7 Grafische Benutzeroberfläche

3.7.1 Systemumgebung und Werkzeuge

(Sebastian Goeser)

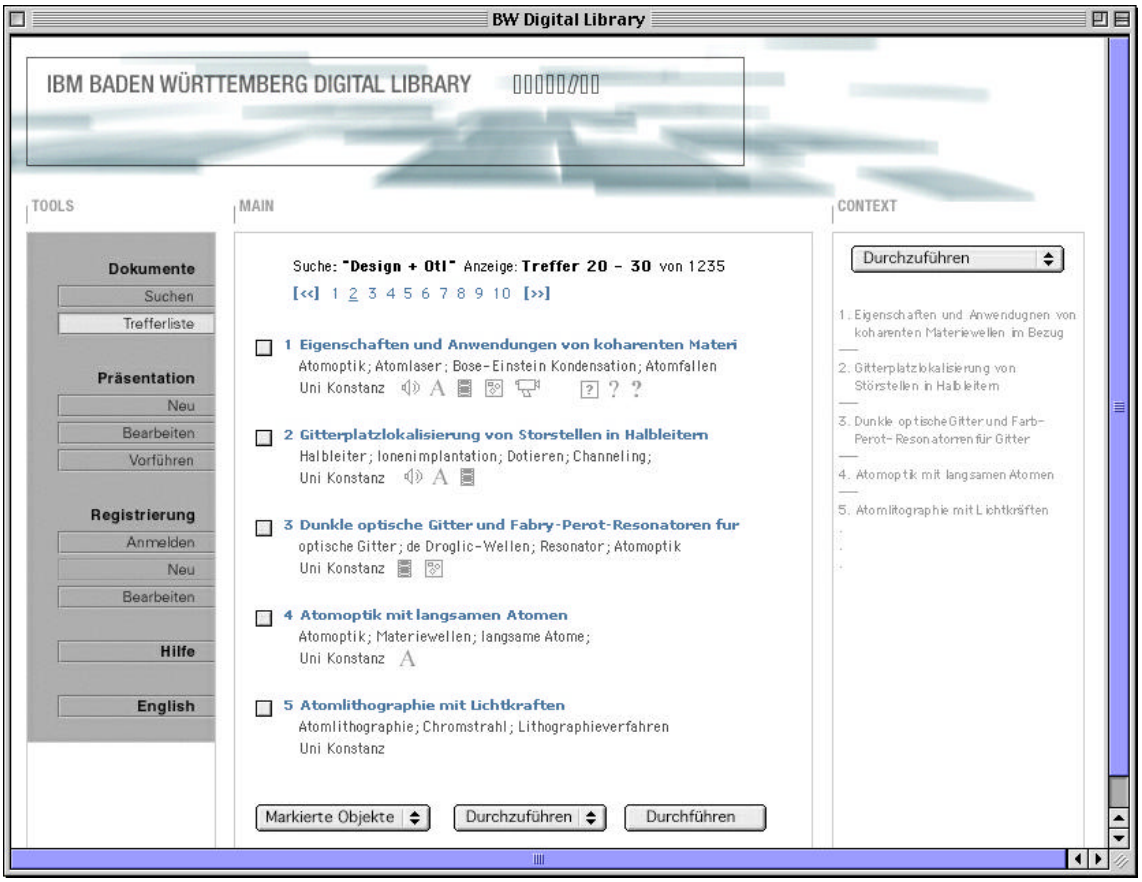
Die Grafischen Benutzeroberflächen (GUI - Graphical User Interface) der Baden-Württemberg Digital Library ergeben sich nach Funktionalität und Architektur aus den lokalen DL-Applikationen der Teilprojekte. Bezüglich der verwendeten Systemumgebung und Werkzeuge wird sich das Projekt BW/DL auf Java-basierte Web-Anwendungen konzentrieren. Hierzu werden unter anderem WebSphere Application Server als server-basiertes Frontend und Java Server Pages als Entwicklungswerkzeug im HTML-Umfeld eingesetzt. Das folgende Diagramm zeigt, wie diese Entwicklungsumgebung auf der Produktarchitektur von IBM DB2 Digital Library aufgesetzt ist:



3.7.2 WWW-Userinterface

(Bernd Aumann)

Für den Zugang zur BW/DL wurde ein HTML-basiertes WWW-Interface entwickelt. (siehe Abbildung).



Das Interface soll einen einfachen und übersichtlichen Zugang zu allen für den Nutzer relevanten Bereichen der BW/DL bieten. Dies wird durch die klare Gliederung in 4 Bereiche erreicht.

1. BW/DL Titelgraphik

Die Titelgraphik gliedert sich in Vordergrund und Hintergrund. Im Vordergrund steht der Name des Systems ergänzt um ein Zeichen, das den Begriff Bibliothek durch die abstrahierte Darstellung einer Buchreihe visualisiert. Der Vordergrund wird durch eine Linie visuell gefasst.

Im Hintergrund befindet sich auf weissem Hintergrund eine 3D-Darstellung im Raum schwebender Flächen. Diese Darstellung erzeugt für die gesamte Seite einen räumlichen Eindruck.

2. Bereich "TOOLS"

Im Bereich "TOOLS" sind sämtliche primären Funktionen der BW/DL zugänglich:

Der Bereich Dokumente bietet Zugang zur Suchmaske und zur Trefferliste. Der Bereich Präsentation erlaubt das Erstellen, Bearbeiten und Vorführen von Präsentationen, die aus Objekten der BW/DL zusammengestellt werden. Der Bereich Registrierung enthält die Funktionen Anmelden, Nutzerdaten bearbeiten und neuen Nutzer anlegen. Zugang zum Hilfesystem und zu einem englischsprachigen Zugang ist vorgesehen.

3. Bereich "MAIN"

In diesem Bereich werden sowohl die zentralen Eingaben getätigt, als auch die Ergebnisse von Anfragen angezeigt. Dies umfasst z.B. Eingabe für Suchanfragen, Ergebnislisten von Suchanfragen, Eingabe von Nutzerdaten bei der Anmeldung, ...

Für am System angemeldete Benutzer werden am unteren Ende dieses Bereiches weitere Funktionen zum Speichern und Bearbeiten von Suchanfragen bzw. einzelnen Suchergebnissen angeboten (Speichern einer Suche, Ablegen einzelner Suchergebnisse in einem "Warenkorb",...)

4. Bereich "CONTEXT"

Die Anzeige in diesem Bereich kann der angemeldete Nutzer selbst wählen. Angezeigt werden können allgemeine Statusinformationen, der Inhalt des Warenkorbes und bereits erstellt Präsentationen.

Durch die klare Gliederung der Funktionsbereiche und die Funktionalität zur Speicherung von Suchanfragen und Präsentationen hat der Nutzer einen einfach zu bedienenden aber trotdem mit hoher Funktionalität ausgestatteten Zugang zur BW/DL.

3.8 Skalierbarkeit

(Sebastian Goeser)

Skalierbarkeitsanforderungen an die BW/DL ergeben sich aus dem Pilotcharakter des Systems mit Bezug auf eine landesweite digitale Bibliothek (s. 2.3). Die Analyse des Gesamtsystems bzgl. Skalierbarkeit erfolgte in drei Schritten, die im folgenden näher erläutert werden:

- Analyse der Performanzcharakteristiken des Basisprodukts IBM DB2 Digital Library in Abhängigkeit von Request-Anzahlen
- Tuning-Maßnahmen, um eine Baseline bzgl. Performanz darzustellen
- Analyse der Performanzcharakteristiken der BW/DL in definierten Benutzungssituationen unter Berücksichtigung der Parameter Volumen und Requests
- Ableitung von Skalierbarkeitsaussagen

3.8.1 Performanz von IBM DB2 Digital Library V2.4

Im folgenden fassen wir Ergebnisse der klassifizierten Studien (1,2) zusammen, die den Projektteilnehmern in schriftlicher Form vorliegt. Die Studien beziehen sich auf die Betriebssysteme AIX (1), das im BW/DL-Kontext für die meisten Installationen verwendet wird und NT (2). Sie können unter dem Gesichtspunkt von Funktionalität und Datenmodellierung nicht direkt auf das BW/DL-System übertragen werden.

Die Studien (1,2) messen die Arbeitslast eines DL-Systems anhand der funktionalen Requests gegen Libraryserver (LSFR) bzw. Objektserver (OSFR). Ein LSFR (OSFR)

ist dabei als elementare Transaktion zwischen der Zwischenschicht, d.h. dem DL Client, und dem Library-Server (Objekt-Server) definiert. Auf Volumenmessungen wurde im Rahmen dieser Studien verzichtet, da für die Objekt-Server Installationen mit sehr großem Datenvolumen in Produktion sind. Vom Testsetting her wurde in insgesamt 8 verschiedenen Konfigurationen, darunter einer SP2-basierten Konfiguration, ein einfaches, typisches Arbeitslast-Szenario durchgefahren, wobei unter der Randbedingung guter Antwortzeiten[1] zwischen 140 und 500 Clients in drei unterschiedlichen Aktivitätsstufen simuliert wurden.

Folgende Ergebnisse aus (1,2) sind im Kontext der BW/DL wesentlich:

- Unter AIX können je nach Konfiguration zwischen 164000 und 387000 LSFRs pro Stunde abgehandelt werden, dies entspricht der Unterstützung von 218 bis 504 Benutzern hoher Aktivität oder 437 bis 1032 Benutzern geringer Aktivität. Unter NT liegen die entsprechenden Zahlen in der gleichen Größenordnung.
- Für eine SP2-Konfiguration mit 3 Objekt-Servern, die der gegenwärtigen BWDL-Konfiguration ähnlich ist, nimmt die Leistungsfähigkeit (LSFRs pro Zeiteinheit) um ca. 15% zu, wenn ein zusätzlicher (vierter) Objektserver eingesetzt wird. Dies ist durch die Abhängigkeit des Arbeitslast-Szenarios von objektbezogenen Operationen zu erklären.
- Hinsichtlich Datendurchsatz sind über AIX und NT sowie allen Konfigurationen Store/Retrieve-Requests für 100KB-Objekte höchstens 50% langsamer als für 1KB-Objekte. D.h., dass der Datendurchsatz der Gesamtapplikation durch entsprechende Packetierung von Objekten deutlich verbessert werden kann.

3.8.2 Tuning-Maßnahmen auf der BW/DL

(noch auszuführen, unter Berücksichtigung von (3))

3.8.3 BW/DL bezogene Performanzmessungen

In Sitzungen des BW/DL-Projektteams am 16.9 und 7.10.1999 wurden die o.a. Skalierbarkeitsanforderungen erarbeitet und ein gemeinsames Vorgehen bei der Messung von Performanzeigenschaften der BW/DL verabredet. Das Ziel dieser Aktivitäten ist, Performanzengpässe mit Bezug auf ein landesweites BW/DL-System rechtzeitig zu erkennen und das applikationsspezifische Tuning zu unterstützen.

Diese projektbezogenen Untersuchungen konzentrieren sich auf das Laden der Karlsruher VVV-Kollektion (ca. 800 BW-Objekte, ca. 23000 physische Objekte) in einem Prozess sowie auf definierte Such- und Retrieveoperationen, die gleichzeitig von mehreren Lokationen durchgeführt werden. Dabei werden Betriebssystem-Parameter wie Prozessorauslastung und IO-Aktivität für die beteiligten Server und Datenbank-bezogene Parameter wie die Transaktionsrate gemessen sowie DL- und applikationsspezifische Logfiles erzeugt. Bei Bereitstellung zusätzlicher Speicher-Hardware durch IBM sollen diese Untersuchungen auf Kollektionen im Gesamtumfang von ca. 32 GB erweitert werden.

Im ersten Schritt wurde die Performanz der BW/DL-Ladeapplikation für die VVV-Kollektion gemessen. Bei der Gesamtzeit der DL-Operationen dieses Ladevorgangs von 5:21 Stunden sind noch mögliche Optimierungsreserven des Loaders zu berücksichtigen. Nach Arbeitslast-Betrachtung ergibt sich folgende Tabelle. Man beachte hier besonders die mittlere Verarbeitungszeit von 0,1 sec für den Store-Request eines Parts, die die in (1) gemessenen Werte sogar noch etwas übertrifft.

---

[1] “Gute Antwortzeiten” wurden durch eine mittlere Dauer des Verbindungsaufbaus (connect request) von höchstens 0.2 sec definiert

Nr	API Request	# Requests	Mittlere V-Zeit	LSFRs	Bemerkungen
1	Query	24.135	0,35 sec	168.945	zur Überprüfung von Duplikaten
2	GetChildren	832	0,24 sec	4.160	Zahlreiche Elemente pro Folder
3	AddChild	48.944	0,15	63.627	Einfügen in die Hierarchie
4	RemoveChild	46	0,1	-	Entfernen aus Hierarchie
5	Store	27.858	0,1	111.432	Abspeichern
6	DeleteItem	693	0,75	-	Löschung eines Items
	Summe	**102.508**			

3.8.4 Ableitung von Skalierbarkeitsaussagen

Die bisherigen Untersuchungen haben keinen Hinweis auf ein Skalierungsproblem erbracht, das einer Erweiterung der BW/DL auf eine größere Anzahl Objekt-Server/Kollektionen prinzipiell entgegensteht. Im Gegenteil scheinen zumindest solche Requests, die physikalische Objekte manipulieren, von einer zunehmenden Anzahl von Objekt-Servern zu profitieren.

Auf der anderen Seite haben wir klare Hinweise auf solche Punkte bekommen, in denen die Performanz der BW/DL durch zusätzliche Programmiermaßnahmen verbessert werden kann. Kritisch ist anzumerken, dass diese zusätzlichen Maßnahmen durch die gegenwärtige Version 2.4 nicht immer optimal unterstützt werden. Zu diesen Maßnahmen zählt insbesondere ein verbesserter Zugriff auf Folder über die DL-Datenbankschnittstelle, aber auch Speicherung und Entspeicherung von BW-Objekten in gepackter Form, um die zugrundeliegenden Caching-Mechanismen besser auszunutzen.

Weitere Aussagen zur Skalierbarkeit einer landesweiten BW/DL werden nach Abschluss der gegenwärtig laufenden Tests zur Suche und Retrieval gemacht werden können.

Bibliographie

(1) Performance Evaluation: Visual Info V. 2.4 / Digital Library V. 2.4 Measurements for AIX Servers, Kiyoshi Shintami, März 1999

(2) Performance Evaluation: Visual Info V. 2.4 / Digital Library V. 2.4 Measurements for NT Servers, Su-jin Chan, März 1999

(3) Image and Workflow Library: Capacity Planning and Performance Tuning for VisualInfo and Digital Library Servers. Mike Ebbers, Gordon Campbell, Peter Carter, Cataldo Mega, Andy Mitchell, Romil Turakhia, IBM RedBook, April 1999

3.9 Untersuchung des Laufzeitverhaltens der IBM DB2 Digital Library

(Udo Willke)

Beim Betrieb der IBM DB2 Digital Library als landesweites System mit Object-Servern an den Standorten der Projektpartner und voraussichtlich vielen gleichzeitigen Benutzern muss auch die Performance des Produktes einer näheren Betrachtung unterzogen werden, um Aussagen zur Dienstgüte im geplanten Produktionsbetrieb zu gewinnen. Wichtige Fragen in diesem Zusammenhang betreffen u.a. die Skalierbarkeit (s. 3.8) des Systems und das Laufzeitverhalten aus Sicht der Benutzer.

Neben Aussagen zu den obengenannten Punkten sollen die Performancetests auch eventuell vorhandene Schwachstellen der beteiligten Systeme identifizieren, die dann möglicherweise durch gezielte Tuningmaßnahmen beseitigt werden können.

Eine voll ausgebaute „Digitale Bibliothek Baden-Württemberg“ könnte typischerweise durch folgende Parameter gekennzeichnet sein:

maximale Objektgröße: **~ 1 GB**
gesamtes Datenvolumen: **~ 500 GB**
Gesamtzahl der Objekte: **~ 10^6 (1 Mio.)**
Anzahl der Object Server: 5 ... 20 (je nach Ausbaustufe des Systems)

Da ein Testsystem mit diesen Kenngrößen aufgrund von fehlender Hardware und digitalen Inhalten nicht ad-hoc realisiert werden konnte, haben sich die Projektpartner auf eine Reihe von kleineren Tests geeinigt, aus denen sich Aussagen zu Skalierbarkeit und Laufzeitverhalten ableiten lassen sollten. Diese sehen wie folgt aus:

1. „Baseline“ - Tests für Load-, Query- und Retrieve-Operationen mit einem einfachen Datenmodell (nur eine Indexklasse). Diese Tests wurden in der Vorbereitungsphase durch die Auswertung von internen Messungen des Entwicklungslabors der Fa. IBM ersetzt. (s. 3.8)
2. Messungen auf dem zentralen Library-Server des Rechenzentrums Karlsruhe mit dem realen BW-Datenmodell (dieser Abschnitt).

Zu 2) sind folgende Messungen geplant bzw. bereits durchgeführt.

a) Laden von BW-Objekten mit dem BW-Loader
b) „Query“ und „Retrieve“ von BW-Objekten über mehrere DL-Clients zur Simulation von mehreren gleichzeitigen Benutzern
c) Skalierungsverhalten bei zunehmendem Datenvolumen; hierzu hat die Fa. IBM die Aufrüstung des zentralen Library Server mit zusätzlichem Plattenplatz zugesichert.

3.9.1 Laden von BW-Objekten mit dem BW-Loader

Die Performance des Loaders scheint auf den ersten Blick wenig relevant für die Gesamtleistung des Systems, da dieser idealerweise nur durch die DL-Systemadministratoren während Zeiten geringer Aktivität auf den Systemen gestartet wird. Erste, zunächst subjektive Erfahrungen haben jedoch gezeigt, dass auch das Laufzeitverhalten des Loaders zum Problem werden kann, wenn eine große Anzahl an BW-Objekten in die DL nachgeladen werden muss (z.B. nach einem Systemcrash oder beim ersten Upload einer neuen großen Kollektion).

Der Pilotbetrieb hat u.a. gezeigt, dass für sehr komplexe BW-Objekte mit z.T. mehreren hundert Parts, wie es für die VVV-Dokumente der Fall ist, die Ladezeiten in der Größenordnung von ~ 1 Minute pro Dokument liegen können. Dies bedeutet jedoch im Umkehrschluss, dass pro Tag (24 Std = 1440 Minuten) maximal 1440

solcher sehr komplexen Objekte geladen werden könnten. Selbst bei einer angenommenen Ladezeit von nur 1 Sekunde pro BW-Objekt wäre auf der anderen Seite die anvisierte Gesamtzahl von 1 Mio. Objekten nur innerhalb von 11,5 Tagen in die DL zu laden.

Laufzeitmessungen des Loaders sind auch deswegen interessant, weil während des Programmablaufs bereits zwei grundlegende Operationen des DL-APIs, nämlich „Query“ und „Store“, ausgeführt werden, und daher die Ergebnisse schon erste Schlüsse auf das Laufzeitverhalten der DL-API-Aufrufe erlauben.

Zur Untersuchung des Ladeprogrammes wurde dieses um Code zur Ausgabe von Zeitmarken vor und nach jedem Aufruf von Programmteilen erweitert, die hauptsächlich die Adapter zum Zugriff auf die DL benutzen (siehe Abschnitt „Loader“). Die Auswertung der Protokolldateien nach dem Ladevorgang erlaubt dann, zwischen DL-spezifischer und nicht-DL-spezifischer Laufzeit zu unterscheiden und so „teure“ von weniger „teuren“ Loaderfunktionen zu trennen.

Ergebnisse der ersten Testläufe auf dem zentralen Library-Server des Rechenzentrums Karlsruhe, bei denen die gesamte Kollektion der UB Karlsruhe geladen wurde, sind bereits im Abschnitt 3.8 im Hinblick auf die Skalierbarkeit des Systems untersucht worden.

Eine genauere Auswertung dieser Daten ergab jedoch einen Bedarf an zusätzlichen Messungen mit einer genauer aufgeschlüsselten Laufzeit der Programmteile. Diese wurden aus organisatorischen Gründen auf dem lokalen DL-System der UB Karlsruhe durchgeführt.

3.9.1.1 Untersuchungen zum Laufzeitverhalten des BW-Loaders

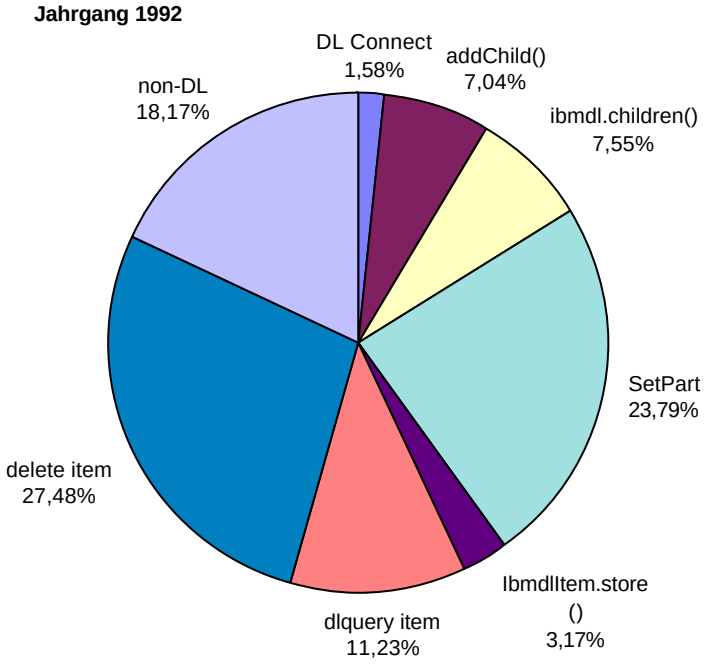
Diese Tests wurden auf dem Entwicklungssystem der UB Karlsruhe durchgeführt, bei dem die IBM DB2 Digital Library, Version 2.4.1 mit Fixpack 3, auf einem IBM RS/6000 F-50 - Server unter AIX 4.3.1 installiert ist. Sowohl lokaler Library- als auch der Object-Server sind auf dem gleichen Rechner vorhanden. Die XML-artigen Loaderdateien und die Multimedia-Dateien waren auf lokalen Dateisystemen verfügbar. Der BW-Loader wurde ebenfalls auf diesem System ausgeführt.

1. Messung: Laden des Jahrgangs 1992

In einer ersten Messung wurden die 24 Dokumente des Jahrganges 1992 aus dem Volltextarchiv in die DL geladen. Von diesen 24 Dokumenten war ein Großteil bereits aus einem vorhergehenden Ladevorgang in der DL vorhanden. In diesem Fall durchläuft der Loader zunächst eine „Delete“-Routine, um das evtl. inkonsistent geladene BW-Objekt zu löschen, um danach mit dem eigentlichen Ladevorgang zu beginnen. Die Gesamtladezeit bei diesem Test betrug ca. 16 Minuten, d.h. im Durchschnitt ca. 41 Sekunden pro Dokument.

Klasse / Funktion	Anzahl Aufrufe	Gesamtlaufzeit [sec]	durchschnittl. Laufzeit pro Aufruf [sec]
DL Connect	24	15.527	0.6470
AddChild()	1350	69.168	0.0512
Ibmdl.children()	93	74.173	0.7976
SetPart	625	233.787	0.3741
IbmdlItem.store()	777	31.112	0.0400
Dlquery item	663	110.319	0.1664
Delete item	690	270.040	0.3914
Gesamtzeit (DL)		804.126	
Gesamtzeit (non-DL)		178.585	

Für die Verteilung der Programmlaufzeit auf die verschiedenen Loaderfunktionen ergibt sich daher folgendes Bild:

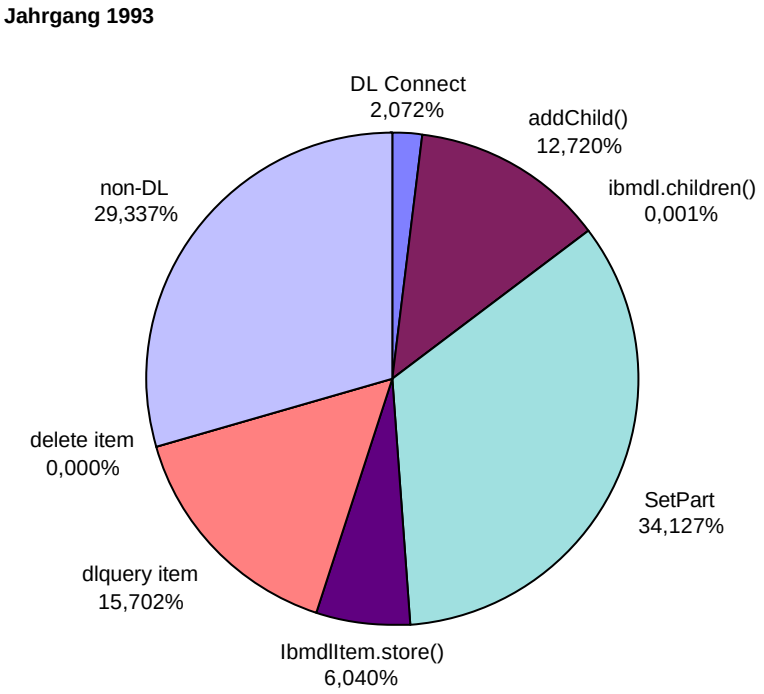


2. Messung: Laden des Jahrganges 1993

Bei dieser Messung war keines der 46 Dokumente bereits in der DL vorhanden, es wurde daher nur der reine Ladevorgang gemessen. Die Gesamtlaufzeit betrug hier insgesamt 18,4 min, d.h. im Durchschnitt 24 Sekunden pro Dokument.

Klasse / Funktion	Anzahl Aufrufe	Gesamtlaufzeit [sec]	durchschnittl. Laufzeit pro Aufruf [sec]
DL Connect	46	22.883	0.4974
addChild()	2278	140.469	0.0616
ibmdl.children()	46	0.016	0.0003
SetPart	1010	376.867	0.3731
IbmdlItem.store()	1362	66.700	0.0489
dlquery item	1116	173.398	0.1553
delete item	0	0.000	0.0000
Gesamtzeit (DL)		780.333	
Gesamtzeit (non-DL)		323.920	

Für den zweiten Test ergibt sich folgendes Bild für Anteile der einzelnen Funktionen an der Laufzeit:



3.9.1.2 Interpretation der Messungen

Unsere Tests zeigen, dass die Kommunikation des Ladeprogramms mit der DL über den Aufruf der DL-APIs zu mehr als 2/3 für die Laufzeit des Programmes verantwortlich ist. Insbesondere die SetPart-Funktion des Loaders, bei der eine Multimedia-Datei gelesen und für die Speicherung in der DL vorbereitet wird, „verbraucht“ bis zu 1/3 der Laufzeit. Sie ist auch mit durchschnittlich 0,37 Sekunden pro Aufruf eine der „teuersten“ und erklärt, warum BW-Objekte mit vielen Parts deutlich längere Ladezeiten haben also solche mit nur wenigen, aber sehr großen Parts.

Eine weitere sehr „teure“ Funktion ist das Löschen ganzer BW-Objekte zur Herstellung der Konsistenz in der Datenbank. Bei bereits vorhandenen BW-Objekten kann sich die Ladezeit um bis zu 50 % verlängern (534 Sekunden ohne „delete“-Operation gegenüber 804 Sekunden mit „delete“-Operation), wie dieser Test zeigt.

Im Verhältnis zur Gesamtlaufzeit wenig aufwendig ist, wie die Diagramme zeigen, der Connect mit der Datenbank, der bei jedem neuen Dokument notwendig ist. Ebenfalls wenig Optimierungspotential vorhanden ist bei den Queries auf der DL, die zur Vermeidung von Redundanzen durchgeführt werden, da diese nur mit weniger als 15 % der Gesamtlaufzeit zu Buche schlagen.

Anhand dieser ersten Ergebnisse lässt sich zeigen, dass eine Optimierung der DL-Performance durch geeignete Tuningmaßnahmen der betroffenen Systeme auch einen Effizienzgewinn des Ladeprogrammes zur Folge hätte. Dies gilt um so mehr, als im geplanten Produktionsbetrieb der Loader auf entfernten DL-Client-Rechnern ablaufen würde, und nur die DL-spezifischen API-Calls auf dem zentralen Library Server und dem betroffenen Object Server Last erzeugen würden.

3.9.2 Messungen zum Laufzeitverhalten bei parametrischen Suchanfragen

Neben den Untersuchungen zum Laufzeitverhalten des Ladeprogrammes wurde das DL-System auch umfangreichen Tests zur Ermittlung von Antwortzeiten und deren Einflußfaktoren unterzogen. Diese Tests sind als Ergänzung der von IBM freundlicherweise zur Verfügung gestellten internen Studie zu verstehen, da sich die Rahmenbedingungen der dort dargestellten Tests von denen der Digital Library Baden-Württemberg[1] in einigen Punkten unterscheiden. Dies betrifft sowohl die Methodik als auch die eingesetzte Programmiersprache.

Während die Testapplikationen der IBM-Studie in der Sprache C++ implementiert sind und vermutlich die Programmierschnittstellen zur DL direkt verwenden, wurde für die Digital Library Baden-Württemberg ein spezieller Access-Layer in der Programmiersprache Java entwickelt, der es erlaubt, Java-Anwendungsprogramme auf der Ebene des BW-Datenmodells zu programmieren. Die IBM-Studie befaßt sich außerdem vorrangig mit Operationen wie „Erzeugen eines Ordners“, „Erzeugen eines Items“ und „Ausliefern eines Items“ sowie mit Funktionen für den Workflow. Die Performance von Suchoperationen auf der DL wird nur am Rande untersucht.

Unsere Untersuchung konzentriert sich auf Laufzeitmessungen und Antwortzeiten, zunächst nur bei Anfragen an den Library Server, da sie sich in unserem Projekt als problematisch herausgestellt haben. Die bereits bekannten Performancedefizite beim „Folder Manager“ werden, wie mit IBM vereinbart, nicht untersucht. Als Grundfunktionen bietet die IBM DB2 Digital Library folgende Anfragemöglichkeiten an:

a) parametrische Anfrage

[1] „Digital Library Baden-Württemberg“ wird in diesem Abschnitt als Bezeichnung für die Implementierung des Produkts „IBM DB2 Digital Library“ im Rahmen des „Digital Library“-Landesprojekts verwendet. „IBM DB2 Digital Library“ wird verwendet, wenn das Produkt als solches gemeint ist.

b) Volltextsuche mit dem Textminer

Parametrische Anfragen können dabei mit einem oder mehreren Kriterien auf der *gleichen* Indexklasse formuliert werden. Die Verknüpfung von Anfragekriterien, die sich über *verschiedene* Indexklassen des BW-Datenmodells erstrecken, z.B. „alle Publikationen einer Person nach dem Jahr 1998“, werden von der IBM DB2 Digital Library nicht direkt unterstützt und müssen daher durch eine eigene Implementierung im Access-Layer realisiert werden. Zwei Lösungsansätze sind im Rahmen dieses Projektes untersucht worden bzw. befinden sich noch in der Untersuchung:

a) Auftrennen der Anfragekriterien in parametrische Abfragen der entsprechenden Indexklassen über die Programmierschnittstellen der DL, Identifikation der zugehörigen „BWObjects“ und nachfolgende UND-Verknüpfung der entstehenden Mengen von Einzelergebnissen.

b) Direkte Abfrage der Oracle-Datenbank über die JDBC-Schnittstelle unter Berücksichtigung der Modellierung der DL-Indexklassen auf der relationalen Datenbank. In diesem Fall werden die Programmierschnittstellen der IBM DB2 Digital Library jedoch weitgehend umgangen.

Textminer-Anfragen dienen in erster Linie der Suche in Volltexten. Bei der Digital Library Baden-Württemberg kommt sowohl die „einfache Suche“ als auch eine kontext-orientierte Suche („proximity search“) zum Einsatz, wobei letztere zur gezielten Suche in speziell ausgezeichneten Feldern (Autor, Titel, Textabsatz, ...) des Superparts verwendet wird.

3.9.2.1 Testbedingungen

Für die Tests wurde das am Rechenzentrum der Universität Karlsruhe vorhandene zentrale Digital Library-Serversystem, zusammen mit einer Reihe von verschiedenen Digital Library-Client-Rechnern eingesetzt, die am Rechenzentrum und der Universitätsbibliothek zur Verfügung standen. Die technischen Daten von Server- und Client-Systemen sind in den Tabellen 6.1 und 6.2 zusammengefaßt.

Knotenname / Funktion	Prozessor	Taktfrequenz	RAM
dl001 Object Server	Power2	135 MHz	256 MB
dl002 Library Server	Power2	77 MHz	256 MB
dl003/dl004 (Oracle DB's)	Power2	77 MHz	256 MB

Tabelle 6.1: Technische Daten der vier SP/2 Konten des zentralen Digital Library-Systems

Das zentrale Digital Library Baden-Württemberg Serversystem besteht aus einem IBM RS/6000 SP/2 - Rechner mit insgesamt vier Konten, die über ein Hochgeschwindigkeits-Backplane miteinander verbunden sind. Zur Verteilung der Serverlast sind der Library-Server, der Object-Server sowie die zugehörigen Datenbanken auf jeweils einem separaten Knoten installiert.

Rechnername / Rechnertyp	Stand-ort	Betriebs-system	Prozessor	Taktfrequenz	RAM
ubkaaixa RS/6000 43P Model 150	UB	AIX 4.3.2	PowerPC	375 MHz	256 MB
ubkaaixg RS/6000 - F 50	UB	AIX 4.3.1	2 x PowerPC	166 MHz	256 MB
ubkaps114 PC - System	UB	Win NT 4.0	Pentium II	300 MHz	128 MB
vogelpohl PC - System	RZ	Win NT 4.0	Pentium II	266 MHz	128 MB
rzhelder PC - System	RZ	Win NT 4.0	Pentium III	500 MHz	256 MB
dl001 SP/2 Knoten	RZ	AIX 4.2.1	Power2	135 MHz	256 MB

Tabelle 6.2: Übersicht über die technischen Daten der eingesetzten DL-Client-Rechner

Bei der Auswahl der DL-Clients wurde dabei darauf geachtet, ein möglichst breites Spektrum an Rechnersystemen in die Tests miteinzubeziehen. Für beide von der IBM DB2 Digital Library unterstützten Betriebssysteme (IBM AIX und MS Windows NT)

kommen daher jeweils drei Rechner verschiedener Leistungsstärke zum Einsatz.

Für den Test wurden insgesamt ca. 4521 Dokumente aus dem Volltextarchiv der Universitätsbibliothek Karlsruhe in den zentralen Library-Server geladen. Um diese Zahl erreichen zu können, wurde der Bestand insgesamt sechsmal geladen, wobei einige Metadaten systematisch verändert wurden, um sicherzustellen, daß diese Dokumente beim Ladevorgang nicht von der Dublettenkontrolle verworfen werden. Man kann daher davon ausgehen, daß die geladenen Dokumente als individuelle Datensätze von der IBM DB2 Digital Library verwaltet werden.

Alle in diesem Abschnitt beschriebenen Testbedingungen wurden während der gesamten Dauer der Tests konstant gehalten.

3.9.2.2 Vorüberlegungen

Beim Laden von Dokumenten in die Digital Library Baden-Württemberg kommt es zu einer heterogenen Verteilung der Metadaten über die verschiedenen Indexklassen. Da für einen Performancetest Anfragen auf Indexklassen mit nur wenigen Einträgen wenig Sinn machen, wurde zunächst die Verteilung der Metadaten geschätzt, bzw. über entsprechende Programme bestimmt. Ausgehend von einer Gesamtzahl von ca. 4521 geladenen Dokumenten kann aufgrund der bekannten Zusammensetzung der Dokumente die in der Tabelle 6.3 zusammengestellte Abschätzung durchgeführt werden:

Indexklasse	Anzahl Items	max. Items
BWObject	ca. 4521	ca. 340
BWPerson	ca. 9000	ca. 730
BWObjectInstance	ca. 13500	ca. 2200
BWPart	ca. 50000+	ca. 1600
BWFormat	19	n/a
BWSubject	54	n/a
BWLanguage	1	n/a
BWCoverage	7	n/a
BWCreator	n/a	n/a
BWContributor BWRelation BWValue BWAttribute	0	n/a

Tabelle 6.3: Verteilung der Metadaten in den verschiedenen Indexklassen

Eine direkte Bestimmung dieser Größen ist aufgrund einer technischen Begrenzung der IBM DB2 Digital Library nur zum Teil möglich. Bei einer parametrischen Abfrage aller „BWObjects“ über einen DL-Client wird immer dann ein Laufzeitfehler ausgelöst, wenn die Anzahl der vom DL-Server zurückgegebenen Ergebnisse einen Grenzwert überschreitet. Dieser Fehler tritt bei allen Anfragen auf jeder beliebigen Indexklasse auf, wenn der in der Tabelle ermittelte Wert in der Spalte „max. Items“ für die betreffende Indexklasse überschritten wird. In der Indexklasse BWObject können also nur maximal ca. 340 Ergebnis-Items verarbeitet werden. Damit ist die genaue Bestimmung der Anzahl aller BWObjects nicht möglich.

Von der Fa. IBM wurde Ende Januar 2000 ein Software-Fix zur Verfügung gestellt, mit dem dieses Problem behoben werden kann. Allerdings haben unsere Untersuchungen gezeigt, daß bei Verwendung dieses Fixes die Laufzeiten für parametrische Anfragen stark ansteigen. Eine genaue Untersuchung der Ursache dieses Effekts konnte bisher jedoch noch nicht durchgeführt werden. Da neben prinzipiellen Ursachen auch Installations- bzw. Konfigurationsfehler in Betracht kommen, wurde der Fix für die weiteren Laufzeitmessungen vom Testrechner wieder entfernt, um die Testbedingungen möglichst konstant zu halten.

Die Tabelle 6.3 zeigt, daß nur vier der insgesamt 13 Indexklassen viele Einträge besitzen. Diese Tatsache wurde bei der Formulierung der Testanfragen berücksichtigt.

Zur Durchführung der Laufzeitmessungen wurde eine Java-Applikation eingesetzt, die eine Reihe von parametrischen Testanfragen aus einer Datei einliest und diese dann sequentiell ausführt. Dabei wurde auf eine möglichst zufällige Abfolge der Testanfragen geachtet, um keine Meßfehler durch evtl. Pufferung von vorhergehenden Anfrageergebnissen zu provozieren. Die Zeiten für die Ausführung jeder einzelnen Testanfrage sowie die Gesamtzeit für alle Anfragen einer Testdatei („Testsuite“) wurden in einer weiteren Datei protokolliert und dann separat ausgewertet. Durch die Definition unterschiedlicher Testanfragen war es möglich, verschiedene Betriebszustände der IBM DB2 Digital Library zu simulieren.

3.9.2.3 Ergebnisse zur Laufzeit und zum Skalierungsverhalten von parametrischen Suchanfragen

Aus verschiedenen Vortests konnten bereits einige interessante Ergebnisse zum prinzipiellen Laufzeitverhalten der IBM DB2 Digital Library ermittelt werden. Es ergaben sich folgende Einflußfaktoren:

a) Datenbank-Login vor jeder Anfrage:

Bei diesem Test hat sich eine deutliche Verschlechterung der Performance gezeigt, wenn für jede Testanfrage ein neuer Datenbank-Login ausgeführt wird. Entgegen den Erwartungen ergibt sich eine Verlängerung der Laufzeit nicht nur durch den DB-Login selbst (3 - 5 Sekunden), sondern auch die für eine einzelne Suchanfrage notwendige Zeit erhöht sich im Durchschitt um 1,4 Sekunden. Digital Library Applikationen sollten daher über eine permanente Verbindung zur Datenbank verfügen, wie es z.B. bei der Implementierung einer Web-Applikation über Java-Servlets der Fall ist.

b) Größe der Indexklasse

In einem weiteren Test konnte gezeigt werden, daß die Laufzeit von parametrischen Anfragen auf den verschiedenen Indexklassen eine Abhängigkeit von der Größe der betreffenden Indexklasse aufweist. Anfragen auf große Indexklassen mit vielen Datenfelder zeigen eine tendenziell längere Laufzeit als Anfragen auf Indexklassen mit wenigen Datenfelder. Typische Meßergebnisse zeigt Diagramm 6.1: Bei 300 zurückgegebenen Ergebnissen dauert eine parametrische Abfrage auf der Indexklasse BWObject ca. 2 Sekunden länger als auf der Indexklasse BWObjectInstance, bei 50 Ergebnissen beträgt der Unterschied jedoch nur noch ca. 0,5 Sekunden. Die Anzahl der zurückgelieferten Ergebnisse kann über den MAX_RESULTS-Parameter der DL Programmierschnittstelle gesteuert werden.

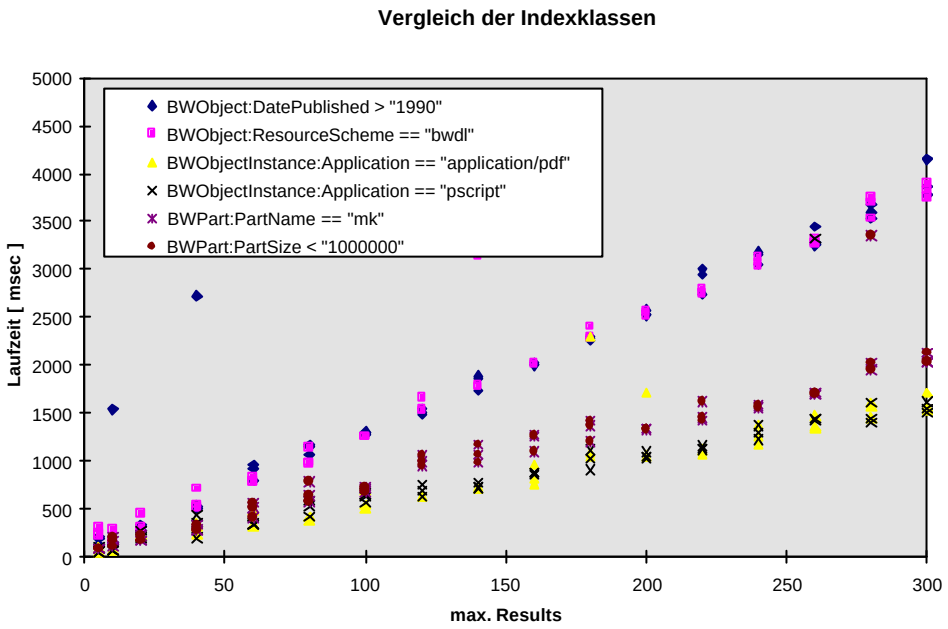


Diagramm 6.1: Vergleich der Laufzeiten von Testanfragen für verschiedene Indexklassen in Abhängigkeit vom MAX_RESULTS - Parameter

c) Leistungsfähigkeit der Netzwerkverbindung und des DL-Client-Rechners

Die Testergebnisse zeigen auch deutliche Unterschiede, wenn die Gesamtlaufzeit einer Serie von Testanfragen für alle eingesetzten Rechner verglichen wird. Eine Zusammenfassung der Ergebnisse zeigt Diagramm 6.2.

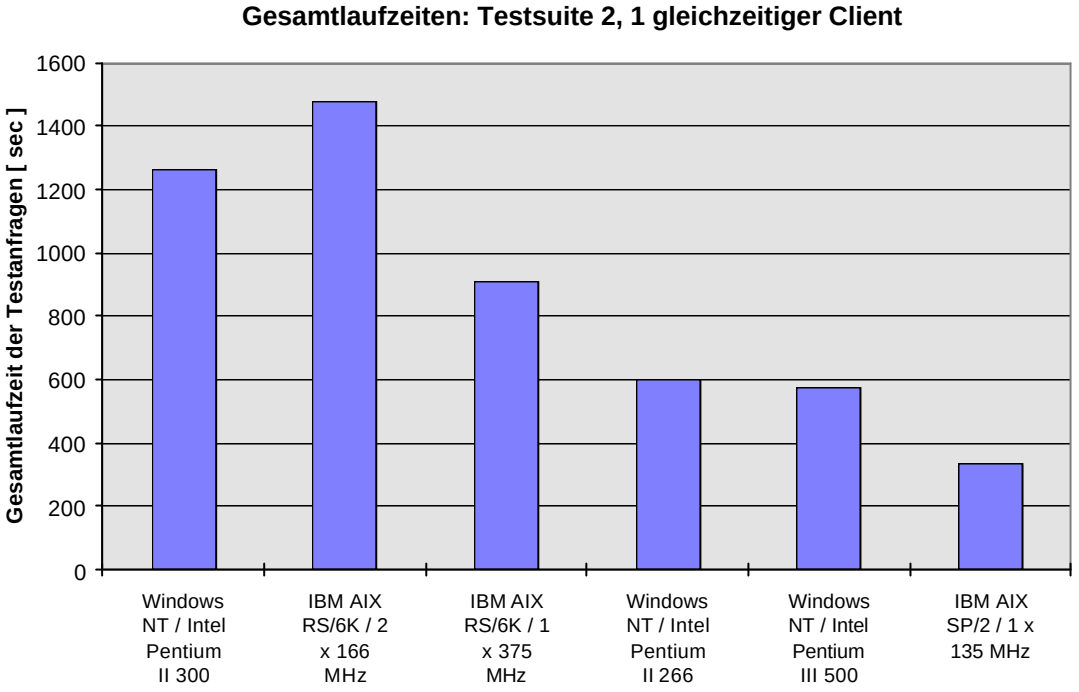


Diagramm 6.2: zeigt deutlich den Einfluß des Standorts des Client-Rechners und dessen Leistungsfähigkeit. Die drei rechten Balken stellen die Laufzeiten der DL-Client-Rechner des Rechenzentrums dar, die drei linken die Rechner der Universitätsbibliothek.

Auf dem Diagramm erkennt man deutlich den Unterschied in der Gesamtlaufzeit zwischen DL-Client-Rechnern des Rechenzentrums und der Universitätsbibliothek (rechte bzw. linke Balkengruppe), deren Rechner über das Campus-Netzwerk mit dem zentralen Library Server kommunizieren müssen. Die geringste Laufzeit für den Knoten des IBM SP/2 - Rechners, der in diesem Test als DL-Client-Rechner eingesetzt wurde, weist auf die große Bedeutung einer leistungsfähigen Netzverbindung hin, da dieses Rechnersystem mit nur wenig Prozessorleistung ausgestattet ist, jedoch über die schnellste Datenverbindung zum Library-Server verfügt. Ein Vergleich von PC- und AIX-Systemen der gleichen Generation zeigt außerdem einen deutlichen Vorsprung für die auf RISC-Prozessoren basierenden IBM AIX-Systeme.

Eine genaue Analyse der bisher vorgestellten Testergebnisse zeigt deutlich den Einfluß der Client-Rechnerleistung und der Leistungsfähigkeit der Netzwerkverbindung auf die Laufzeit von Testanfragen. Es hat sich daher für die nachfolgenden Test als sinnvoll herausgestellt, eine Fallunterscheidung anzustellen. Für die Performace der IBM DB2 Digital Library ist es offenbar entscheidend, wieviele Ergebnisse eine parametrische Anfrage zurückliefert, da diese über das Netzwerk übertragen und vom DL-Client-Rechner verarbeitet werden müssen.

Für die Performance- und Skalierungstests unterscheiden wir daher zwei Fälle.

1) präzise Testanfragen
2) unpräzise Testanfragen

Präzise Testanfragen sind so formuliert, daß sie nur wenige Ergebnisse (n < 80) zurückliefern. Dies kann entweder durch das gewählte Anfragekriterium erreicht werden oder durch Limitieren der Anfrage mit dem MAX_RESULTS - Parameter der DL-Programmierschittstelle. Dieser Fall entspricht den Bedingungen für ein reales Produktionssystem, bei dem nur einfache, d.h. mit einem oder mehreren Kriterien auf einer einzelnen Indexklasse, aber präzise Anfragen durch die Benutzer gestellt werden.

Unpräzise Testanfragen sind so formuliert, daß die Abfrage auf der Indexklasse eine maximale Anzahl an Ergebnissen (340 < n < 2200, je nach Indexklasse) zurückliefert. Dieser Fall würde sich z.B. dann ergeben, wenn eine Anfrage mit mehreren Kriterien auf verschiedenen Indexklassen in einem ersten Schritt in ihre Einzelanfragen aufgelöst würden, um dann die Ergebnisse dieser Einzelanfragen weiterzuverarbeiten. Für ein korrektes Endergebnis müßte jedoch jede Teilanfrage eine vollständige Anzahl an Treffern zurückliefern, was bisher aufgrund der Beschränkung für jede Indexklasse nicht der Fall ist.

Fall 1: Präzise Testanfragen

Das Diagram 6.3 zeigt die Zusammenfassung der Ergebnisse dieser Testläufe. Für jeden Rechner wurden die ausgewählten Testanfragen sowohl einzeln, als auch parallel mit anderen Testrechnern ausgeführt. Drei gleichzeitige Clients beschreibt die Situation, wenn alle drei Testrechner eines Standortes parallel Testanfragen an den Libary Server stellen, bei sechs gleichzeitigen Clients wurden alle Testrechner beider Standorte parallel eingesetzt. Bei 12 gleichzeitigen Clients wurde die Java-Testappliktion jeweils zweimal auf jedem Testrechner gestartet.

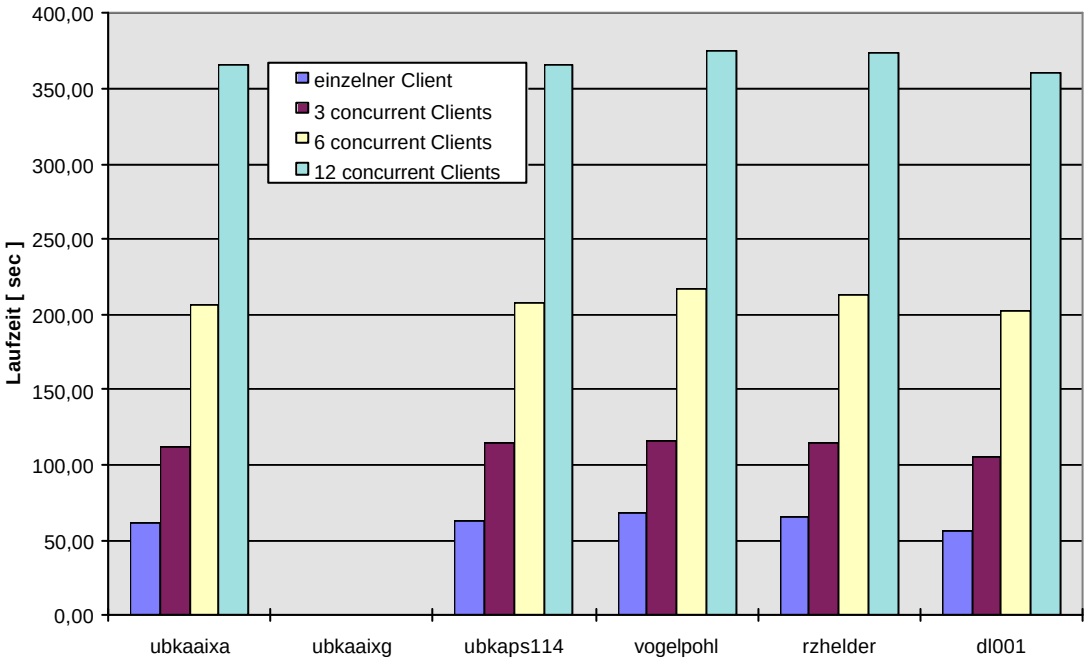


Diagramm 6.3: Gesamtlaufzeiten für die DL-Client-Testrechner bei präzisen Testanfragen

Auf dem Diagramm ist deutlich zu erkennen, wie sich die Gesamtlaufzeiten bei einer zunehmenden Zahl von DL-Clients vergrößern. Es fällt jedoch auch auf, daß sich die Laufzeiten für die unterschiedlichen Rechner in diesem Fall nur wenig unterscheiden.

Auf der Server-Seite weist der Knoten des DL-Systems, auf dem die Oracle-Datenbank für den Library-Server betrieben wird, eine relativ hohe Last auf. Auf dem Library-Server selbst beobachteten wir jedoch nur eine verhältnismäßig geringe Last. Der limitierende Faktor in diesem Test ist also die Oracle-Datenbank, die in dieser Testkonfiguration die Hauptarbeit leisten muß. Der Library-Server nimmt die Anfragen der DL-Clients offenbar nur entgegen und liefert, nach der SQL-Abfrage der Oracle-Datenbank, die Ergebnisse wieder „reihum“ an die DL-Clients zurück. Die Rechnenleistung der Client-Rechner und die Netzwerkverbindung spielen in diesem Fall offenbar eine untergeordnete Rolle.

Um Aussagen zur Skalierbarkeit des Gesamtsystems zu bekommen, sind die Testergebnisse auf dem Diagramm 6.4 nochmals in normierter Form dargestellt. Hier betrachten wir die Veränderung der Gesamtlaufzeit für die Testanfragen bezogen auf die Zeit, die der DL-Client für die Verarbeitung der Testanfragen als einziger aktiver Client gebraucht hat. Der Wert „2“ bedeutet dann eine Verdopplung der Laufzeit. Diese Auswertung ist für die Testkonfigurationen mit drei, sechs und 12 gleichzeitigen zugreifenden DL-Clients im Diagramm 6.4 gezeigt.

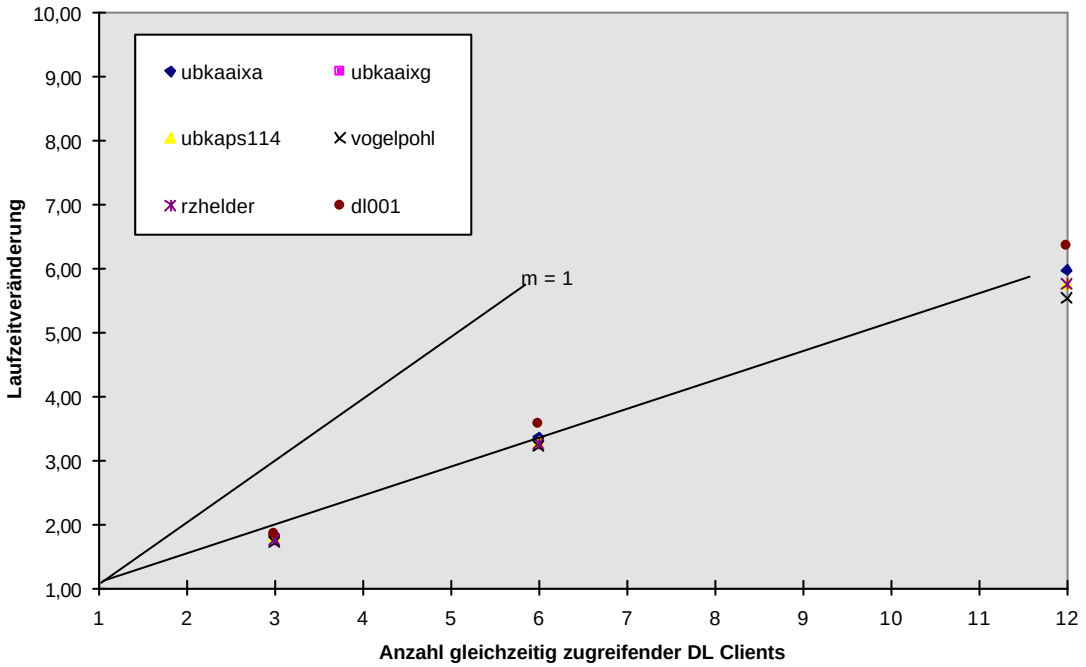


Diagramm 6.4: relative Veränderung der Gesamtlaufzeit der Testanfragen bei zunehmender Anzahl gleichzeitig aktiver DL-Clients für den Fall von präzisen Anfragen

Die nahezu lineare Zunahme der Laufzeit bis auf das etwa Sechsfache bei 12 parallel zugreifenden Clients ist deutlich zu erkennen. Die zusätzlich eingezeichnete Gerade

(Steigung m=1) zeigt das erwartete Verhalten für ein vollständig ausgelastetes Rechnersystem. In diesem Fall würde die Verdoppelung der Arbeitslast auch eine Verdoppelung der Laufzeit für das Testprogramm bedeuten. Unser Test zeigt jedoch, daß die Laufzeiten nur etwa halb so schnell ansteigen wie die Arbeitslast, es ist daher anzunehmen, daß das Gesamtsystem in dieser Testkonfiguration noch nicht vollständig ausgelastet ist.

Eine Schwierigkeit bei der Beurteilung dieser Ergebnisse liegt darin abzuschätzen, wie vielen realen Benutzern einer der gleichzeitig aktiven Clients, der kontinuierlich Testanfragen an den Library Server stellt, entspricht. In der internen IBM-Studie wird das Benutzerverhalten durch den Einbau von Warteschleifen von 15 - 30 Sekunden in die Testapplikationen simuliert. Daher sind die dort ermittelten Benutzerzahlen viel höher als in den hier vorgestellten Tests. Grundsätzlich hängt diese Zuordnung ab vom Verhältnis der Antwortzeit des Systems zur Dauer der Pause, die der Benutzer bis zur nächsten Interaktion mit dem System einlegt. Ist die Pause nur etwa so lange wie die Antwortzeit des Systems, so entspricht ein Testclient etwa zwei realen Benutzern, ist sie zehnmal so lange, entspricht ein Testclient etwa 10 realen Benutzern.

Wenn man für eine einfache Abfrage der Indexklasse BWObject mit 50 Ergebnissen eine Antwortzeit von etwa 0,8 Sekunden für einen einzelnen Benutzer annimmt, so würden bei einer „Denkpause“ von acht Sekunden unseren 12 Testclients etwa 120 reale Benutzer mit gleichem Verhalten entsprechen. Diese müßten dann jedoch aufgrund des Parallelzugriffs 4,8 Sekunden (6 x 0,8 Sekunden) auf die Antwort des Systems warten und dann eine Pause von 48 Sekunden einlegen. Es spricht aber vieles dafür, daß unsere 12 Testclients deutlich weniger realen Benutzern entsprechen dürften.

Fall 2: Unpräzise Anfragen

Für den Fall der unpräzisen Anfragen faßt das Diagramm 6.5 die Ergebnisse zusammen. Die Laufzeiten der einzelnen Client-Rechner zeigen nun deutliche Unterschiede in der Gesamtlaufzeit. Rechner mit hoher Prozessorleistung und guter Netzwerkverbindung zeigen deutlich geringere Gesamtlaufzeiten als weniger leistungsstarke Clients.

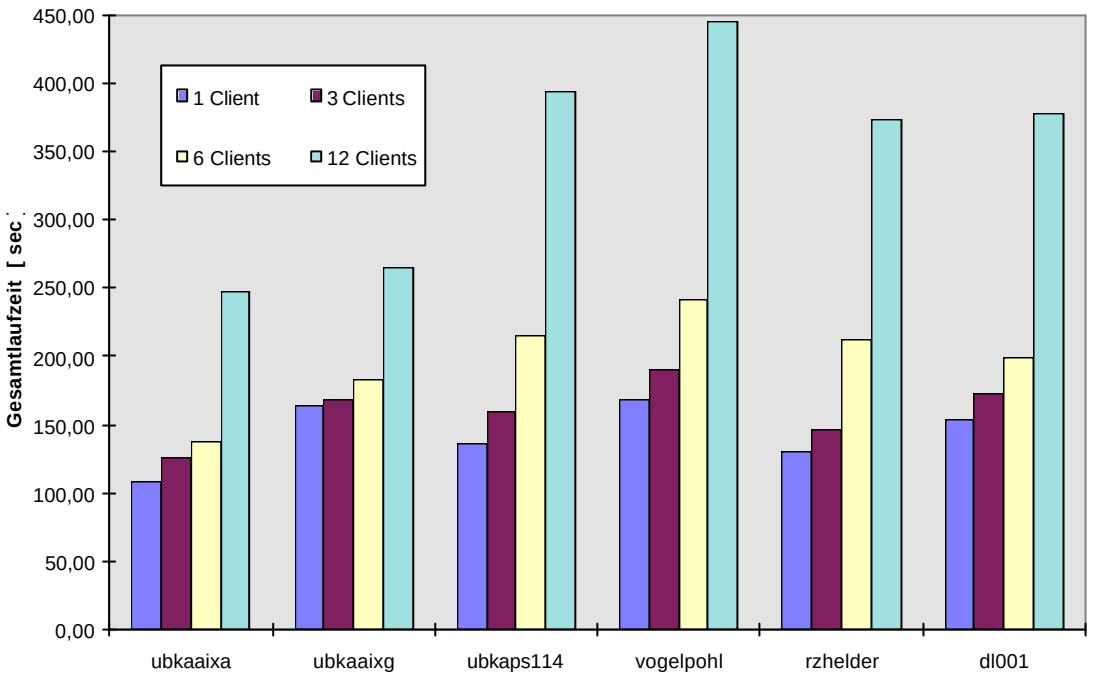


Diagramm 6.5: Gesamtlaufzeiten für die DL-Client-Testrechner bei unpräzisen Testanfragen

Auf der Server-Seite beobachten wir in dieser Testkonfiguration eine hohe Last auf dem Libary-Server, jedoch eine deutliche geringere Last auf dem Konten des SP/2 - Rechners, auf dem die zugehörige Oracle Datenbank betrieben wird. In diesem Test scheint der Library-Server der limitierende Faktor zu sein. Wir gehen aber davon aus, daß sich in dieser Testkonfiguration auch die Netzwerkverbindung kontraproduktiv auf die Gesamtlaufzeiten der Testrechner auswirkt, da die Last auf dem Library-Server während des gesamten Tests moderat bleibt.

Das Skalierungsverhalten des Gesamtsystems wird, analog zum vorhergehenden Fall, im Diagramm 6.6 dargestellt. Die relative Änderung der Gesamtlaufzeit wird wieder in Abhängigkeit der Anzahl von gleichzeitig aktiven DL-Clients untersucht.

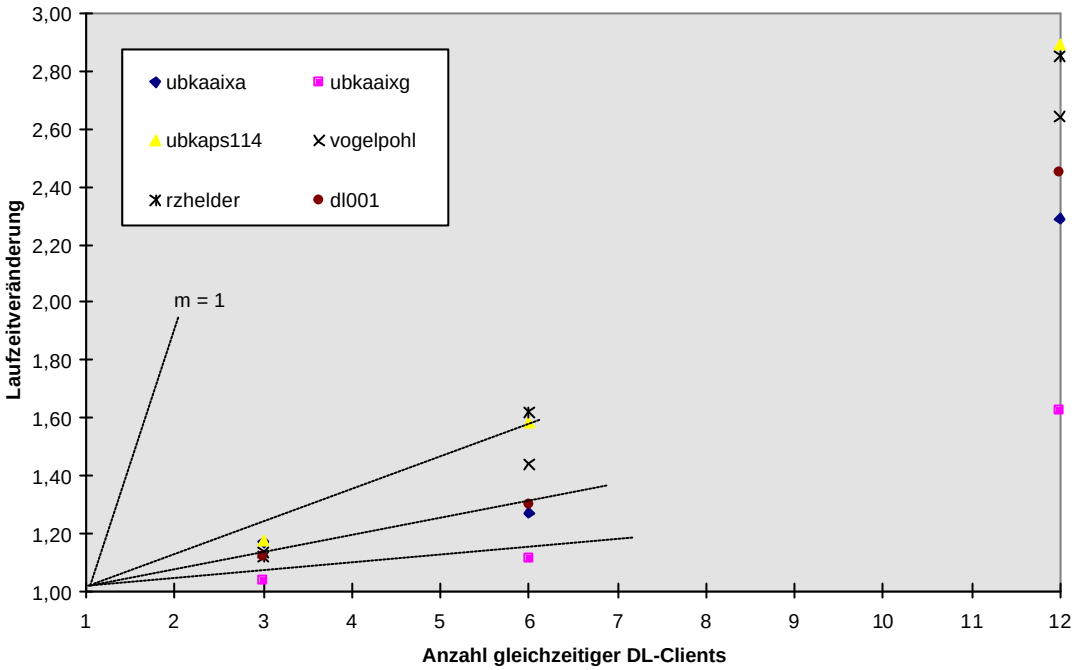


Diagramm 6.6: relative Veränderung der Gesamtlaufzeit der Testanfragen bei zunehmender Anzahl gleichzeitig aktiver DL-Clients für den Fall von unpräzisen Anfragen

Im Gegensatz zur Untersuchung der präzisen Anfragen, zeigt dieses Diagramm keinen gemeinsamen Trend für die relative Veränderung der Laufzeit der verschiedenen Testclients mehr. Die Gesamtlaufzeit wächst für alle Testclients auch nicht mehr so stark an wie im Fall von präzisen Anfragen. Bei 12 gleichzeitig aktiven DL-Clients wächst die Gesamtlaufzeit um höchstens das Zwei- bis Dreifache an, während im vorher untersuchten Fall eine Versechsfachung der Laufzeit gefunden wurde. Für diesen Testfall spielen ganz offensichtlich die Leistungsfähigkeit der eingesetzten Testrechner und die Netzwerkverbindung die wichtigste Rolle. Bei Windows NT - Systemen zeigen die Meßergebnisse ein tendenziell stärkeres Anwachsen der Gesamtlaufzeit als bei den AIX - Systemen. Inwiefern hierfür das verwendete Betriebssystem verantwortlich ist, läßt sich jedoch a priori nicht entscheiden.

Es scheint naheliegend, daß neben der Rechenleistung der eingesetzten Client-Rechner auch die Leistungsfähigkeit der Netzwerkverbindung eine Rolle spielen muß, denn im Fall der unpräzisen Anfragen müssen mit hoher Wahrscheinlichkeit größere Datenmengen über das Netzwerk transportiert werden. Eine Analyse der Netzwerkkommunikation auf der Ebene der Internet-Datenpakete bestätigt diese Annahme und zeigt, daß z.T. Daten in der Größenordnung von mehr als 1 MByte pro Anfrage vom Library-Server zum Digital Library-Client übertragen werden müssen.

Diese Beobachtung wird durch einen weiteren Test bestätigt, der in Zusammenarbeit mit dem Bibliotheks-Servicezentrum in Konstanz durchgeführt wurde. Unter gleichen Testbedingungen braucht hier die gleiche Serie von unpräzisen Anfragen an den zentralen Library-Server in Karlsruhe etwa die dreifache Gesamtlaufzeit, wie für ein vergleichbares IBM RS/6000 - System an der Universitätsbibliothek. Im Fall der präzisen Anfragen war dieser Effekt jedoch kaum sichtbar.

3.9.3 Messungen zum Laufzeitverhalten der Volltextsuche (Textminer)

Neben den parametrischen Suchanfragen bietet die IBM DB2 Digital Library über das Zusatzprodukt „Intelligent Miner for Text“ auch die Möglichkeit einer leistungsfähigen Volltextsuche an. Hierzu wurde im BW-Datenmodell ein mit jedem „BWObject“ assoziierter „Superpart“ definiert, in dem neben dem Volltext auch bibliographisch relevante Informationen wie Autor, Titel, Abstract und Herausgeber gespeichert und zur Volltextindizierung bereitgestellt werden können. Durch die Markierung von Textabsätze mit eindeutigen Bezeichnern (XXXcreator, XXXtitle, XXXpublisher, ...) ist es über die kontext-orientierte Volltextsuche sogar möglich, in diesen „Datenfeldern“ selektiv zu suchen.

In Hinblick auf einen Produktionsbetrieb der Digital Library sind daher folgende zwei Suchmethoden von besonderem Interesse:

a) einfache Suche („simple serach“)
b) kontext-orientierte Suche („proximity search“)

Bei der **einfachen Suche** bilden die vollständigen Superparts die Grundlage der Volltextsuche. Es sind hier ein oder mehrere Suchargumente mit Booleschen Verknüpfungen möglich.

Bei der **kontext-orientieren Suche** erfolgt eine Einschränkung des Suchkontexts auf den gleichen Satz, den gleichen Abschnitt bzw. auf das gleiche Dokument. Bei einer Textminer-Suche, z.B. nach einem Autor eines Dokuments, kann dann über die kontext-orientierte Volltextsuche nach dem Vorkommen von „XXXcreator“ UND dem Autorennamen im gleichen Textabschnitt gesucht.

3.9.3.1 Ergebnisse zur einfachen Textminer-Suche

Die Laufzeit von Textminer-Anfragen setzt sich durch die Implementierung im Access-Layers aus zwei Phasen zusammen

1) Verarbeitung der Textanfrage durch den Textminer
2) Aufbau der DL-spezifischen „Dynamic Data Objects“ und Übertragung der Ergebnisse zum DL Client-Rechner

Das Diagramm 6.7 zeigt die Zusammensetzung der Laufzeiten für eine einfache Suche von Wörtern bei denen ein häufiges Vorkommen in den Volltexten vermutet wird.

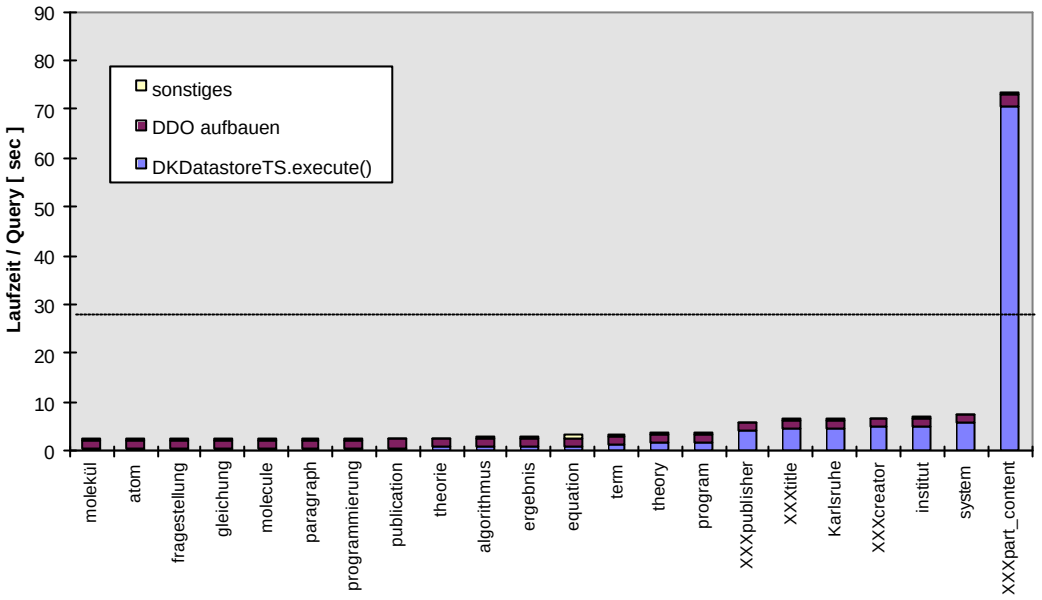


Diagramm 6.7: Laufzeiten für einfache Textminer-Anfragen; Limitierung der Anzahl der Ergebnisse auf 10 über den MAX_RESULTS - Parameter. Der untere Teil der Balken stellt jeweils die verbrauchte Zeit für die erste Phase der Volltextsuche dar, der obere die Zeit, die zum Aufbauen der Datenstrukturen und der Übertragung der Daten zum DL-Client notwendig ist.

Das Diagramm legt die Vermutung nahe, daß die erste Phase der Laufzeit von der Häufigkeit des Vorkommens eines Worts im Volltextindex abhängt. Die für die Auszeichnung der bibliographischen Information verwendeten Bezeichner kommen erwartungsgemäß sehr viel häufiger in den Texten vor, als andere, oft gebrauchte Wörter. Artikel und andere für das Informationsretrieval nicht relevante Wörter wie „und“, „oder“, „mit“, „ohne“, etc. werden, da sie auf der sogenannten „stop-word“-Liste vorkommen, gar nicht in den Volltextindex aufgenommen.

Die lange Laufzeit für die Suche nach XXXpart_content erklärt sich dadurch, daß dieser Bezeichner in jeden Absatz des Volltextes eines Dokuments eingefügt wird. Eine grobe Analyse ergab, daß „XXXpart_content“ ca. 2,1 Mio. Mal in den Volltexten vorkommt, „XXXcreator“ jedoch nur ca. 8.500 Mal. Dies ergibt ein Verhältnis für das Vorkommen dieser beiden „Wörter“ von ca. 1:254.

Wenn die Annahme einer Abhängigkeit der reinen Textminer-Suchzeit von der Worhäufigkeit zutrifft, so könnte man hieraus auch die Schlußfolgerung ziehen, daß die Suche nach „XXXcreator“ in einer Digital Library mit dem 254-fachen Datenbestand dann genauso lange dauern müßte wie für XXXpart_content im aktuellen Datenbestand. In diesem angenommenen Fall würden sich dann ca. 1,15 Mio. vergleichbare Dokumente in der Datenbank befinden. Eine reine Suchzeit von ca. 70 Sekunden für eine Volltextsuche wäre dann sicherlich ein sehr akzeptables Ergebnis.

Die im Diagramm ebenfalls dargestellte Laufzeit zum Aufbau der „Dynamic Data Objects“ nimmt im gezeigten Test mit einer Limitierung der Anzahl auf 10 Ergebnisse einen nur sehr kleinen Anteil ein. Dieser wächst jedoch sehr stark an, wenn mehr Ergebnisse von der Text Search Engine angefordert werden

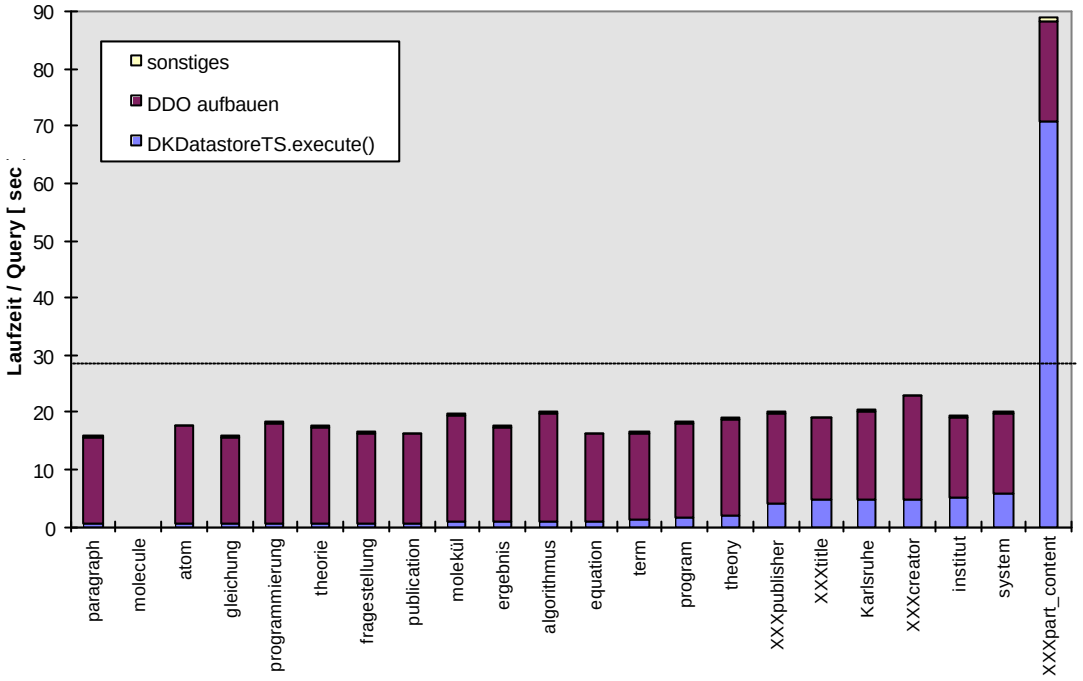


Diagramm 6.8: Laufzeiten für einfache Textminer-Anfragen; Limitierung der Anzahl der Ergebnisse auf 80 über den MAX_RESULTS - Parameter

Diagramm 6.8 zeigt die Situation bei einer Begrenzung der Volltextsuche auf 80 Ergebnisse. Die Suchanfragen dauern nun deutlich länger, da die benötigten Zeiten für den Aufbau und die Übertragung der Datenobjekte die reine Laufzeit für die Suchanfrage deutlich übertreffen. Bei häufig vorkommenden Wörtern nähert sich hier die Antwortzeit des Systems der 30 Sekunden-Grenze, die oft als maximal zumutbare Wartezeit für die Benutzer angenommen wird.

Die reine Laufzeit für die Volltextsuche unterscheidet sich für den Fall der Begrenzung auf 10 bzw. 80 Elemente nur unwesentlich, der MAX_RESULTS - Parameter scheint also auf die Texminer-Suche keine oder nur eine geringe Auswirkung zu haben. Die DL-Programmierschnittstelle zum Textminer Search Engine bietet daher einen zusätzlichen Parameter an, der eine zeitliche Begrenzung der reinen Volltextsuche ermöglicht. Dann werden nur so viele Ergebnisse zurückgeliefert, wie innerhalb der Suchzeit gefunden werden konnten.

3.9.3.2 Ergebnisse zur kontext-orientieren Textminer-Suche

Der häufigste Anwendungsfall der kontext-orientierten Suche bei der Digital Library Baden-Württemberg ist die selektive Suche nach bibliographisch relevanter Information, die im Superpart vorhanden sein kann. Im Diagramm 6.9 ist ein typisches Ergebnis für die Suche nach Autoren zusammengefaßt. Zur Suche wurden die am häufigsten vorkommenden Vornamen der Autoren benutzt, um realistische Testbedingungen zu schaffen.

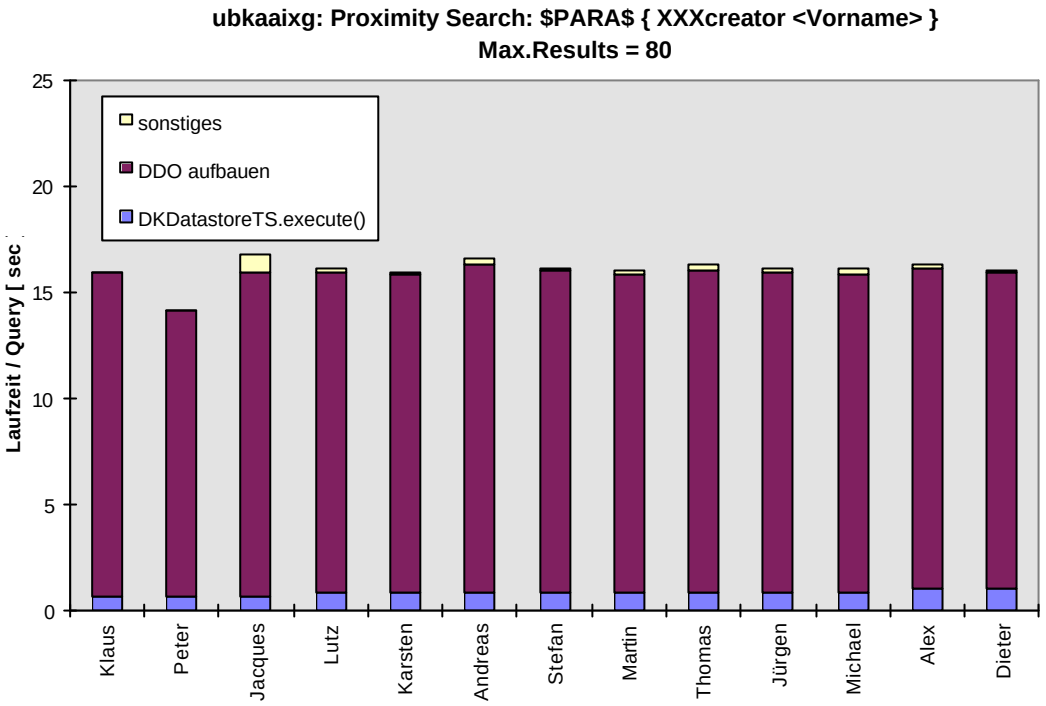


Diagramm 6.9: Laufzeiten für kontext-sensitive Textminer-Anfragen: Suche nach Vornamen der Autoren

Wie im Fall der einfachen Suche sind die Antwortzeiten wieder aufgeschlüsselt in die reine Suchzeit der Text Search Engine (DKDatastore.execute()), und die Zeit, die zum Aufbau und zur Übertragung der Datenobjekte benötigt wird. Generell kann gesagt werden, daß sich die reine Textminer-Suche nach Autoren auf dem vorhandenen Datenbestand als sehr performant herausstellt. Die reinen Suchzeiten liegen in der Regel unter einer Sekunde. Bei 80 Ergebnissen müssen jedoch auch 80 „Dynamic Data Objects“ aufgebaut und zum DL-Client übertragen werden was ein Vielfaches der Antwortzeit der Text Search Engine benötigt.

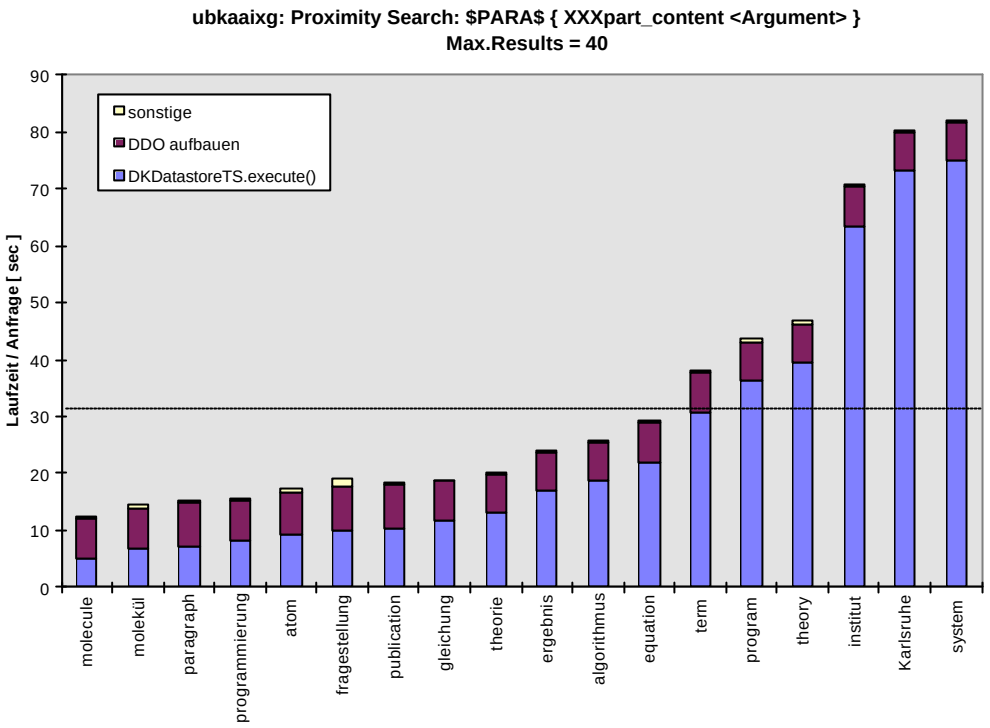


Diagramm 6.10: Laufzeiten für kontext-sensitive Textminer-Anfragen: Suche nach Begriffen im Volltext

Bei der selektiven Suche im Volltext eines Dokuments beobachten wir eine deutliche Zunahme der Suchzeiten. Dies hängt mit der Häufigkeit des Vorkommens des Bezeichners „XXXpart_content“ zusammen, durch den die Suche deutlich aufwendiger wird. Die 30 Sekunden-Marke wird daher für gängige Suchbegriffe häufig überschritten.

Dieses Ergebnis ist jedoch nur teilweise relevant, da die Häufigkeit des Auftretens dieses Bezeichners für den Volltext durch Änderungen des BW-Loader-Codes stark reduziert werden kann, indem der gesamte Volltext in nur einem Textabschitt zusammengefaßt und mit nur einem XXXpart_content-Bezeichner versehen wird. Die Ergebnisse können jedoch einen Anhaltspunkt dafür geben, wie sich die Antwortzeiten bei einem Datenbestand von ca. 1 Mio. Dokumente entwickeln würden. Selbstverständlich bleiben die Antwortzeiten auch bei einem solchen Datenbestand für Volltextsuchen mit sehr spezifischem, d.h. selten eingesetztem Fachvokabular weiterhin gering. Bei einem öffentlich zugänglichem System muß jedoch auch dem Fall von Anfragen mit häufig vorkommenden Suchbegriffen Rechnung getragen werden.

Die Tests zeigen erneut den großen Anteil, den der Aufbau der Datenobjekte an der Gesamtlaufzeit haben können, während die reine Volltextsuche nur wenig von der Anzahl der Ergebnisse beeinflußt wird..

Zum Schluß soll noch darauf hingewiesen werden, daß die vom „Intelligent Miner for Text“ angebotenen Trunkierungsmöglichkeiten das Potential haben, die Laufzeiten für reine Volltextsuchen erheblich zu verlängern. So war es während der Tests sehr leicht möglich „naive“ Anfragen zu formulieren, die selbst beim aktuellen Datenbestand bereits eine reine Suchzeit von bis zu fünf Minuten benötigten. Es müssen daher auf

jedem Fall bei der Applikationsentwicklung für die Digital Library Baden-Württemberg Vorkehrungen getroffen werden, um minutenlange Antwortzeiten zu vermeiden. Dies ist im Fall der Volltextsuche notwendiger als im Fall von einfachen parametrischen Suchanfragen.

3.9.4 Direkte Abfrage der Datenbank über die SQL / JDBC-Schnittstelle

(Werner Vogelpohl)

Aufgrund der Performanceprobleme bei der Benutzung der DL-APIs für die parametrische Suche wurde als Workaround im Access Layer das zusätzliche Java-Package DBQuery entwickelt. Um einzelne Indexklassen direkt in der Datenbank abzufragen, muß zuerst ein Mapping des BW-Datenmodells auf die ‘dynamisch‘ (bei Einspielen des Datenmodells) generierten AVT-Datenbanktabellen des Library Servers durchgeführt werden. Durch Joins mit der ‘statisch‘ vorhandenen Tabelle SBTLINKS erhält man die gesuchte ParentItemID des Dokuments in der Indexklasse **BWObject**. Der folgenden Tabelle liegt der Datenbestand am 31. März 2000 zugrunde.

Indexklasse	DB-Tabellenname	Anzahl der Zeilen	Tiefe in Hierarchie
	SBTNLSKEYWORDS	474	
	SBTATTRDEFS	194	
	SBTCLASSDEFS	29	
	SBTLINKS	297485	
	SBTPARTS	140897	
	SBTITEMS	175389	
BWPart	AVT00016	132736	2
BWValue	AVT00017	0	3
BWFormat	AVT00018	19	3
BWObject	AVT00019	4539	0
BWPerson	AVT00020	3013	2
BWCreator	AVT00021	9421	1
BWSubject	AVT00022	54	1
BWCoverage	AVT00023	7	1
BWLanguage	AVT00024	1	1
BWRelation	AVT00025	0	1
BWAttribute	AVT00026	0	4
BWContributor	AVT00027	0	1
BWObjectInstance	AVT00028	13140	1

Entscheidend für die Antwortzeiten ist (neben der Besetzung der Tabellen) auch hier wieder die Position einer Indexklasse in der Hierarchie des Datenmodells; die einfache Query **BWFormat:Mimetype = 'image/gif'** bedingt einen mehrfachen Join mit der Tabelle SBTLINKS:

```
select distinct L1.PARENTITEMID
from SBTLINKS L1, SBTLINKS L2, SBTLINKS L3, AVT00018 A
where L2.PARENTITEMID=L1.CHILDITEMID
and L1.LINKKIND=1
and L3.PARENTITEMID=L2.CHILDITEMID
and L2.LINKKIND=1
and L3.CHILDITEMID = A.ITEMID
and A.ATTRIBUTE00139 = 'image/gif';
```

während die Query **BWObject:Collection = 'UB Volltexte'** deutlich einfacher umzusetzen ist:

```
select distinct L1.PARENTITEMID
from SBTLINKS L1, AVT00019 A
where L1.PARENTITEMID = A.ITEMID
and A.ATTRIBUTE00149 = 'UB Volltexte'
```

Der Einfachheit halber wurden die Messungen für verschiedene Queries mit einem SQL-Klienten direkt auf dem Datenbank-Knoten durchgeführt. Die jeweilige Trefferliste wird in eine Datei umgeleitet, Caching-Effekte in der SGA sind bei der Zeitmessung nicht berücksichtigt. Indizes liegen auf den Tabellenspalten, soweit der Library Server sie automatisch anlegt: für die AVT-Tabellen auf der Spalte ItemID, bei der SBTLINKS auf vier Spalten gemäß den Angaben in der Application Programming Reference.

In der folgenden Tabelle sind verschieden komplexe Beispielqueries, die Länge der Ergebnislisten und die gemessenen Antwortzeiten festgehalten.

Querystring	Treffer	Antwortzeit [sec]
BWObject: DateCreated > '1990-07-01' AND DateCreated < '1999-07-01' AND BWPerson: CompleteName <> 'Ha%'	4174	8

Querystring	Treffer	Antwortzeit [sec]
BWObject: DateCreated > '1990-07-01' AND DateCreated < '1999-07-01' AND BWPerson: CompleteName like 'Ha%'	42	4
BWObject: DateCreated > '1995-07-01' AND DateCreated < '1996-07-01' AND BWPerson: CompleteName like 'Ha%'	11	< 1
BWObject: Collection = 'UB Volltexte' AND BWPerson: CompleteName like 'Ha%'	43	5
BWFormat: MimeType = 'image/gif'	4270	75
BWFormat: MimeType = 'text/xml'	6	< 1
BWObject: Collection like 'Wissen%' AND BWPerson: LastName like '%er' AND BWObjectInstance: InstanceName like 'UT.Phys%' AND BWFormat: MimeType = 'text/xml'	4	1
BWObject: DescriptionShort like 'Theaterzettel%'	8	< 1

Bei feingranularer (Kombination mehrerer Indexklassen) Suche erkennt man einen deutlichen Performancegewinn, verglichen mit entsprechenden Queries via DL-APIs. Dagegen erweist sich eine Suche nach Attributen mit hoher Kardinalität (Attribute der **BWFomat**) beim hier gezeigten Vorgehen als schwierig. Zu den Antwortzeiten addiert sich die Zeit für den Aufbau eines DDO pro gefundenem Element in **BWObject** hinzu. Bei entsprechend geschickter Suche läßt sich die Länge der zu erwartenden Ergebnisliste und damit die Gesamtwartezeit des Nutzers aber *wahrscheinlich* in erträglichem Rahmen halten.

Systematische Untersuchungen der Datenbankschnittstelle und ihre Einbettung in das Access Layer der Baden-Württemberg Digital Library stehen noch aus. Prinzipiell scheint die Suche mittels JDBC eine lohnende Alternative zur reinen DL darzustellen, solange gegenwärtige Releases der *IBM DB2 Digital Library* keine UND-Verkettung verschiedener parametrischer Suchen mit akzeptabler Performance erlauben.

3.9.5 Fazit

Die in diesem Kapitel vorgestellten, zum Teil sehr umfangreichen Tests haben dazu beigetragen ein detailliertes Verständnis der Funktionsweise der IBM DB2 Digital Library zu gewinnen. Dabei hat sich hinsichtlich der Performance dieser Software-Lösung für das Content Management ein sehr heterogenes Bild ergeben. Das mit seinen Zusatzprodukten sehr umfangreiche System zur Speicherung und Verwaltung multimedialer Inhalte besitzt beispielsweise ein sehr mächtiges Werkzeug zur Volltextrecherche, dessen Performance bei der reinen Suche, bei Beachtung einiger Vorsichtsmaßnahmen bei der Applikationsentwicklung, überzeugen kann.

Die Implemetierung der IBM DB2 Digital Library auf Basis von modernen Software-Konzepten wie den „Dynamic Data Objects“ (DDO’s), definiert durch die „CORBA Persistent Object Service and Object Query Service“ Spezifikation der OMG[1], führt jedoch nach unserer Meinung zu einem System, das sehr hohe Anforderungen an die eingesetzte Hardware stellt. So konnten wir zeigen, daß die reine Volltextsuche über das Zusatzprodukt „Intelligent Miner for Text“ sehr perfomant arbeitet, die Aufbereitung der Ergebnisse in DDO’s jedoch zum Performanceeinbruch führen kann.

Bei den parametrischen Suchanfragen konnten wir einen ähnlichen Effekt zeigen, wenn eine sehr große Anzahl an Ergebnissen auf die Anfragen zurückgeliefert werden. Hier konnten wir auch deutlich den Einfluß des Netzwerks und der Rechenleistung des DL-Client-Rechners zeigen. Es wäre daher wenig überraschend, wenn auch auf der Server-Seite ähnliche Effekte zu erzielen wären. Für den Betrieb einer leistungsfähigen Digital Library Baden-Württemberg erscheint uns daher der Einsatz der neuesten Hardware-Generation für Server und Clients zwingend notwendig.

Die bei den Laufzeittests gewonnen Ergebnisse und Erfahrungen können aber auch helfen, bekannte Performancedefizite durch geschickte Programmierung zu umgehen. So sollten Digital Library-Appliktionen immer über eine permanente Verbindung zur Datenbank verfügen und Suchanfragen sollten nur mit sinnvoll gesetztem MAX_RESULTS - Limit an den Library-Server gestellt werden. Beim Einsatz von Trunkierungen sollten Textminer-Anfragen immer mit einem Zeitlimit versehen werden, um dem Benutzer lange Wartezeiten auf ggf. sogar unerwünschte Ergebnisse zu ersparen.

Es hat sich auch herausgestellt, daß kombinierte Recherchen über mehrere Indexklassen nur über direkte SQL-Datenbankabfragen sinnvoll zu realisieren sind.

Bei Ausnutzung aller Optimierungsmöglichkeiten und Einsatz von leistungsfähiger Hardware ist das Produkt IBM DB2 Digital Library daher durchaus eine geeignete Softwarelösung für die Digital Library Baden-Württemberg.

---

[1] Object Management Group, http://www.omg.org

4. Projektplanung

(BSZ)

4.1 Information und Kommunikation im Projekt

Am BW/DL-Projekt sind fünf Einrichtungen des Landes Baden-Württemberg sowie IBM Deutschland beteiligt. Neben gemeinsamen Projektzielen gibt es, wie in den als Anlagen zum Rahmenvertrag beschriebenen Teilprojekten, divergierende Interessenschwerpunkte - nicht zuletzt bedingt durch die unterschiedliche Provenienz der Einrichtungen: zwei Rechenzentren, zwei bibliothekarische Einrichtungen sowie ein universitärer Lehrstuhl.

Um das Projekt erfolgreich durchführen zu können, war deshalb eine intensive Kommunikation zwischen allen Projektbeteiligten notwendig, nicht zuletzt auch, um diese "verschiedenen Welten" näher zusammenzubringen.

Hierzu fanden zahlreiche Arbeitstreffen statt, bei denen die Beteiligten sich über den Status der Teilprojekte informierten, zentrale Aspekte des Projekts (wie bspw. das gemeinsame Datenmodell) offen - und häufig auch kontrovers - diskutierten sowie das weitere Vorgehen koordinierten. Dreizehn derartige Treffen fanden zwischen Februar 1998 und Februar 2000 statt, hinzu kamen zahlreiche weitere Besprechungen, die der Klärung von Einzelfragen (Datenmodel, Performance-Tests etc.) dienten.

Um die Zahl der - zwar wichtigen, aber aufgrund der verschiedenen Standorte der Beteiligten auch zeitintensiven - Treffen zu beschränken, wurden zusätzlich, organisiert von IBM, Telefonkonferenzen durchgeführt.

Für Programme, Dokumentationen etc. ist auf dem FTP-Server des BSZ ein nur den Projektteilnehmern zugänglicher Bereich eingerichtet, beim ZDV Tübingen wird ein BSCW-Server betrieben. Zusätzlich haben alle anderen Einrichtungen Download-Bereiche auf ihren Servern eingerichtet.

Als wichtigstes und am intensivsten genutztes Kommunikationsmedium erwies sich jedoch die beim BSZ geführte geschlossene Mailing-Liste "ibm-dl", die derzeit 35 Teilnehmer hat und die für alle projektbezogenen Themen genutzt wird: Terminabsprachen, Protokolle, Software-Distribution, Fehlermeldungen u.v.m.

4.2 Öffentlichkeitsarbeit

4.2.1 Präsentationen

Das BW/DL Projekt wurde bei folgenden Veranstaltungen der Fachöffentlichkeit vorgestellt:

1. BSZ-Kolloquium, Universität Stuttgart, 17./18. September 1998
Andreas Lehmann: Das Projekt Digital Library

European IBM DB2 Digital Library Meeting, Stuttgart, 14./15. Dezember 1998
Andreas Lehmann: The IBM Digital Library Project in Baden-Wurttemberg

Workshop "Digitale Bibliotheken", ZDV Tübingen, 5. Februar 1999
Andreas Lehmann: Das IBM Digital Library Projekt in Baden-Württemberg - Vorhaben, Ergebnisse, Perspektiven

89. Deutscher Bibliothekartag, Freiburg, 25.-28. Mai 1999
Andreas Lehmann: Multimediale Objekte im BSZ

Fortbildungsveranstaltung des AKI Stuttgart, 9. November 1999
Sebastian Goeser: Die Digitale Bibliothek Baden-Württemberg

4.2.2 Darstellungen im WWW

BSZ Konstanz:
`http://www.bsz-bw.de/diglib/ibmdl/`

LOMI Ulm:
`http://lomi.e-technik.uni-ulm.de/LOMIWeb/projekte/ibm_dl/ibm_dl.html`

Rechenzentrum der Universität Karlsruhe:
`http://dl001.rz.uni-karlsruhe.de/`

Universitätsbibliothek Karlsruhe:
`http://www.ubka.uni-karlsruhe.de/allg/projekte/dlbw/index.html`

ZDV Tübingen:
`http://www.uni-tuebingen.de/zdv/END/pr/dlprj.htm`

IBM:
`http://www-4.ibm.com/software/data/solutions/customer/baden/baden.htm`

4.3 Terminplanung

Für die koordinierte Zusammenarbeit im Rahmen des Gesamtprojekts war, in Ergänzung zu den Zeitplänen der Einzelprojekte, eine auf diese abgestimmte übergreifende Terminplanung notwendig, die regelmäßig anhand der Projektfortschritte überprüft und fortgeschrieben wurde.

Die weitere Planung hängt wesentlich von der Entscheidung ab, ob und in welcher Form das Projekt nach März 2000 hinaus fortgeführt wird.

Zum jetzigen Zeitpunkt (März 2000) stellt sich die Terminplanung folgendermaßen dar:

Aufgabe	Zuständigkeit	Termin	Status
Installation			
DL 2.4	Alle	03/99	Abgeschl.
"Green Beta"	Alle	Offen	
Oracle 8	BSZ	02/99	Zurück-gestellt *
	LOMI	Offen	
Software-Entwicklung			
Implementierung des Datenmodells (Tool des LOMI)	Alle LOMI (Koordinierung)	03/99	Abgeschl.
Loader-Entwicklung, Parser	UB KA, ZDV	03/99	In Arbeit **
GUI-Entwicklung (Recherche, Präsentation,...)	Alle	Offen	In Arbeit
Entwicklung von DL-Administrationstools	LOMI		Abgeschl.
Schnittstelle zur Verbunddatenbank / Datenfluss	BSZ	04/99	Abgeschl.
Authentifizierung / Benutzerverwaltung	Alle	Offen	In Planung
Bereitstellung der Kollektionen			
Vorbereitung (Sammlung, Verfilmung, Digitalisierung, Bildbearbeitung u.ä.)	Alle	Fortlaufend	In Arbeit
Laden der Objekte in die DL	Alle	Ab 05/99	In Arbeit ***
Anbindung an zentralen LS am RZ KA	Alle	06/99	Abgeschl. ****
Anbindung an Verbund-Datenbank	BSZ	06/99	In Arbeit ***
System-Administration, System-Tests			
Konzeption für Sys-Adm. (Disaster-Recovery etc.)	Alle	Offen	In Planung

* Entgegen früherer Aussagen von IBM unterstützt DL2.4 Oracle 8 nicht.

** Eine lauffähige Version des Loaders wurde rechtzeitig fertiggestellt, weitere Versionen sind in Arbeit.

*** Erste Tests waren erfolgreich, die Arbeiten werden darauf aufbauend fortgeführt.

**** Die Funktionalität wurde sichergestellt, einzelne Objektserver zu Testzwecken aber wieder an die lokalen Libraryserver angehängt.

Aufgabe	Zuständigkeit	Termin	Status
Tests zu Skalierbarkeit/Performance	Alle	03/2000	Abgeschl.
Support			
Schulungen DL 2.4, API	IBM	04/99	Abgeschl.
Neue Releases, CSD's	IBM, RZ KA	Fortlaufend	
Kundennummer	IBM	02/99	Abgeschl.
Zugriff auf Fehler-DB	IBM		Offen
Verbesserung der personellen Unterstützung	IBM		Offen
Öffentlichkeitsarbeit			
Workshop IBM Digital Library	Alle	Offen	
WWW (Prototypen)	Alle	04/99	Abgeschl.
Abschlussbericht	Alle	Offen	