Modellierung und Realisierung eines Informationssystems für wissenschaftliche Literatur mit WWW-Zugang

Diplomarbeit von
Anja Beckmann

August 1996

Universität Leipzig, Institut für Informatik
Betreuer: Prof. Dr. E. Rahm, Dr. D. Sosna

Inhaltsverzeichnis

1. Einführung

Mit der stetig wachsenden Zahl von Publikationen wächst auch das Bedürfnis nach einer Erschließung und effizienten Verwaltung dieser Informationen und nach komfortablen Recherchemöglichkeiten. Die in vielen Bibliotheken benutzten Literaturverwaltungsprogramme sind vor allem durch mangelnde Standardisierung gekennzeichnet, wodurch der Datenaustausch zwischen den Bibliotheken, aber auch die Entwicklung neuer Anwendungen erheblich erschwert werden. Auch die Benutzerfreundlichkeit bestehender Systeme ist oft mangelhaft. Mit der zunehmenden Bedeutung von elektronischen Publikationen müssen sich die wissenschaftlichen Bibliotheken zudem neuen Anforderungen stellen, denen herkömmliche Systeme zur Verwaltung von Literaturdaten nicht mehr gerecht werden.

Relationale Datenbankmanagementsysteme (RDBMS) [Dat95], [Mei95] weisen gegenüber den bisher im Bibliothekswesen verwendeten Dateiverwaltungssystemen viele Vorteile auf: sie erleichtern die Portierung von Anwendungen sowie den Austausch von Daten und unterstützen Interoperabilität, sie gewährleisten aufgrund ihrer hohen Leistungsfähigkeit auch bei vielen gleichzeitigen Nutzern meist schnelle Bearbeitungszeiten und bieten systemseitige Sicherung der Datenintegrität.

Diese Arbeit stellt einen Entwurf für ein Literaturverwaltungssystem vor, der auf dem relationalen Modell basiert und die Verwaltung und Ausleihe von Literaturbeständen modelliert, wobei auch elektronische Publikationen integriert wurden. Eine Implementierung des Entwurfs wurde mit dem RDBMS Sybase [Syb1] realisiert, welches am Institut für Informatik der Universität Leipzig zur Verfügung steht.

Sybase basiert wie die meisten RDBMS auf dem Client-Server-Prinzip, d.h. die Daten werden unabhängig von der Anwendung in einem leistungsfähigen Datenbank-Server gehalten. Die Datenbankanwendung kann dagegen auf einem Client-Rechner, z.B. auf einem PC oder einer Workstation, laufen. Für die Anwendungsentwicklung kommen zum einen die herkömmlichen Datenbank-Frontends, wie z.B. PowerBuilder, in Frage, zum anderen bietet sich mit dem World Wide Web (WWW) und seinen weitverbreiteten Browsern die Möglichkeit, ohne größeren Aufwand die Anwendungen unter einer einheitlichen und komfortablen Oberfläche einem breiten Nutzerkreis verfügbar zu machen. Da die WWW-Clients (Browser) bei den potentiellen Nutzern meist ohnehin bereits installiert sind, entfällt die Distribution der Anwendung an den Endnutzer, auch Änderungen und Erweiterungen der Anwendung können zentral erledigt werden. Die Plattformunabhängigkeit von WWW-Anwendungen ist ein weiterer wichtiger Punkt, da dadurch die Entwicklung und Wartung der Anwendungen erheblich erleichtert wird.

Zudem ergibt sich die Möglichkeit zur Erweiterung der Funktionalität einer Literaturverwaltung, da nicht nur die bibliographischen Angaben bereitgestellt werden, sondern zusätzlich auch Volltextversionen und Verweise auf Homepages von Autoren und Verlagen in das Informationsangebot einbezogen werden können. Insbesondere ergibt sich durch die Integration ins WWW auch die Möglichkeit zur Erschließung von Literaturquellen, die nur in elektronischer Form vorliegen, ein Aspekt, der im Hinblick auf die Entwicklungen in diesem Gebiet und auf Projekte wie MEDOC [TUM96], [Rah96] zunehmend an Bedeutung gewinnt.

In den folgenden 3 Kapiteln wird zunächst ein Katalog von Anforderungen an Datenbanksysteme vorgestellt, bezüglich welcher dann das im Bibliothekswesen weitverbreitete Literaturverwaltungssystem *allegro* und als Alternative dazu relationale Datenbankmanagementsysteme untersucht werden. Bei der Betrachtung von relationalen DBMS wird besonders auf das für die Implementierung verwendete RDBMS Sybase Bezug genommen.

Der Kern der Arbeit besteht in einem auf dem relationalen Modell basierenden Entwurf für ein Literaturverwaltungssystem, welches nicht nur bibliographische Nachweise verwaltet, sondern darüberhinaus auch den Zugriff auf in elektronischer Form vorliegende Volltexte ermöglicht und somit auch die Verwaltung von elektronischen Publikationen gestattet. Dieser Entwurf wird im Kapitel 5 vorgestellt und erläutert. Der Entwurf wurde für die Bibliothekszweigstelle Informatik/Rechenzentrum der Universität Leipzig implementiert. Dabei wurden die von der Zweigstelle übernommenen *allegro*-Daten um Institutsberichte und Artikel erweitert sowie Verweise auf Online-Dokumente wie Homepages und Volltextversionen integriert.

Im Kapitel 6 wird die Eignung des WWW als Frontend für Datenbankanwendungen untersucht. Kapitel 7 beschäftigt sich mit der Realisierung einer Online-Recherche und enthält einführende Informationen zum Common Gateway Interface (CGI), zu HTML-Formularen sowie zu Perl und erläutert die Funktionsweise des für die Web-Anbindung benutzten Sybase-Tools web.sql. Auch auf die Implementierung der Online-Recherche wird eingegangen.

Kapitel 8 enthält eine Zusammenfassung und gibt einen Ausblick auf mögliche Verbesserungen und Erweiterungen.

2. Anforderungen an ein Datenbanksystem (DBS)

Um die Eignung eines DBS für die Erfüllung eines bestimmten Aufgabenbereiches beurteilen zu können, müssen zunächst die von dem System zu erfüllenden Anforderungen festgelegt werden. In Anlehnung an [HW93] lassen sich die folgenden Aufgaben feststellen, die von einem DBS erfüllt werden sollten:

- ❑ dauerhafte Datenhaltung
- ❑ kontrollierter Mehrbenutzerbetrieb
- ❑ Datensicherheit
- ❑ Datenschutz
- ❑ verschiedene Nutzerschnittstellen
- ❑ hohe Datenunabhängigkeit
- ❑ Effizienzbeeinflussungsmöglichkeiten (Tuning)
- ❑ Erweiterungsmöglichkeiten

Die **effiziente, dauerhafte Datenhaltung** ist die Hauptaufgabe eines DBS und umfaßt die Realisierung von Möglichkeiten zur Definition und Änderung von Datenstrukturen und zur Dateneingabe, Datenmanipulation und Datenauswertung. Die dazu bereitgestellten Sprachen sollen leicht erlernbar sein und in Verbindung mit einfachen Datenmodellen eine leichte Handhabbarkeit der Daten ermöglichen. Dabei soll Redundanz eliminiert werden, um Inkonsistenzen zu vermeiden. Auch die Durchsetzung von Standards spielt eine große Rolle.

Ein geregelter **Mehrbenutzerbetrieb** stellt sicher, daß jeder Nutzer unabhängig von allen anderen so arbeiten kann, als wäre er der alleinige Benutzer der Datenbank. Das DBS muß also die Probleme lösen, die durch konkurrierenden Zugriff auf Hardware-Ressourcen und Daten entstehen können.

Bezüglich der **Datensicherheit** muß unterschieden werden zwischen logischer und physischer Datensicherheit. Die physische Sicherheit der Daten ist gefährdet durch Systemabstürze, Gerätefehler, beschädigte Datenträger und weitere technische Fehlerquellen. Das DBS muß Funktionen zur Sicherung des Datenbestandes (logging) und zur Wiederherstellung des korrekten Zustandes nach dem Auftreten von Fehlern (recovery) bieten. Unter logischer Datensicherheit versteht man die Wahrung der semantischen Integrität der Daten durch die Vermeidung von inkonsistenten, d.h. inhaltlich fehlerhaften, Zuständen der Datenbank. Ein Konzept zur Transaktionsverwaltung muß sicherstellen, daß eine Änderungsoperation die Datenbank immer aus einem konsistenten Zustand wieder in einen konsistenten Zustand überführt. Durch Integritätsbedingungen kann die fehlerhafte Dateneingabe eingeschränkt werden.

Datenschutz ist gewährleistet, wenn ein Mißbrauch der oder Zugriff auf die in der Datenbank gehaltenen Informationen durch Unberechtigte verhindert werden kann, Zugriffsschutzkonzepte dienen diesem Zweck.

Den verschiedenen Nutzergruppen einer Datenbank, wie Endanwender, Anwendungsprogrammierer und Datenbankadministrator, müssen ihren Kenntnissen und Bedürfnissen entsprechende **Schnittstellen** zur Verfügung stehen. Dabei spielt die Bereitstellung einer SQL-Schnittstelle eine besonders große Rolle. Diese weitverbreitete Datenbanksprache bietet viele Vorteile, da sie standardisiert ist und auf hohem abstrakten Niveau mächtige Konstrukte zur Datendefinition, -manipulation und -auswertung bietet.

Datenunabhängigkeit entsteht durch die Trennung von Anwendungsprogrammen und Daten, wodurch eine Reorganisation der Daten in der Datenbank ohne Änderung der Anwendungsprogramme ermöglicht wird. Ein anwendungsneutrales Datenmodell unterstützt die Datenunabhängigkeit. Es wird unterschieden zwischen physischer Datenunabhängigkeit, die interne Repräsentation und Speicherung betreffend, und der logischen Datenunabhängigkeit, welche Änderungen des Datenmodells und der logischen Sicht auf die Daten möglich macht. Durch eine starke Isolation der Anwendungsprogramme von den Daten wird der Wartungsaufwand für Anwendungen erheblich reduziert.

Im Zusammenhang mit der Datenunabhängigkeit stehen auch die Möglichkeiten zur Beeinflussung der **Effizienz** eines DBS. Da die Performanz eines Systems in großem Maße abhängig ist vom konkreten Anwendungsfall und von Parametern wie der Größe der Datenbank, der Anzahl der parallel arbeitenden Nutzer und der Häufigkeit bestimmter Anfragen, welche sich im Laufe der Zeit beträchtlich ändern können, muß es dem Datenbankadministrator möglich sein, durch Performance-Tuning den sich ändernden Anforderungen an das System gerecht zu werden. Dabei sind sowohl Möglichkeiten zur Reorganisation der Daten, zur Anpassung von Parametern, Speicherbereichen und Zugriffspfaden oder Umstieg auf andere Datenträger - die ohne Änderung der Anwendungen realisierbar sein sollten - als auch die Portierung auf leistungsfähigere Plattformen wichtig.

Darüberhinaus spielen die **Erweiterungsmöglichkeiten** und die Weiterentwicklung des Systems eine nicht zu vernachlässigende Rolle. Ein wichtiger Punkt ist die Skalierbarkeit des Systems, um steigenden Anforderungen gerecht werden zu können. Die zunehmende Bedeutung des elektronischen Publizierens erfordert zudem Konzepte zur Erschließung von Online-Publi-

kationen in Literaturdatenbanken. Dabei spielt die Unterstützung von Möglichkeiten zur Anbindung von Datenbanken an das WWW eine große Rolle.

Die oben genannten Anforderungen an Datenbanksysteme sollen im folgenden auf ihre Erfüllung durch das in vielen Bibliotheken verwendete System *allegro* und, stellvertretend für relationale DBMS, durch das kommerzielle System Sybase untersucht werden.

3. Das Literaturverwaltungssystem *allegro*

Das seit 1980 an der Universitätsbibliothek (UB) Braunschweig entwickelte *allegro* wird bezeichnet als programmierbares Datenbankensystem für variable Daten, welches von seiner Struktur her besonders für bibliothekarische Anforderungen geeignet ist. Es wird derzeit in über 600 Bibliotheken verwendet und ist seit 1991 auch an der UB Leipzig im Einsatz. Die existierenden Module decken wichtige Bereiche der Bibliotheksarbeit wie Katalogisierung, Sacherschließung, OPAC (Online Public Access Catalog), Volltextsuche und Export, Monographien-Erwerbung und Ausleihe ab, dabei kommen jedoch nicht in jeder Installation alle Module zum Einsatz [UBB1].

Zur Programmierung von *allegro* existieren 3 verschiedene Sprachen, die drei verschiedenen Aufgabenbereichen entsprechen:
Die **Daten-Definitionssprache** dient zur Definition der Datenbank, d.h. zur Beschreibung des Datenformats (Kategorien), der Satztypen und der Eingabereihenfolge, welche in einer Konfigurationsdatei abgespeichert werden. Mit der **Exportsprache** wird die Überführung der Daten in ein Ausgabeformat beschrieben, dies gilt für die Bildschirmanzeige ebenso wie für den Index und die Kurzanzeige, Listenausdrucke und Dateiauszüge. Zur Überführung von Fremddaten in eine *allegro*-Datenbank gibt es eine **Importsprache** [UBB2].
Der Zugriff auf die Daten einer Datenbank erfolgt über einen **Index**, der bis zu 11 zu definierende Register enthalten kann. An der UB Leipzig werden die folgenden Register verwendet:

UB Leipzig:

1. Namen von Personen
2. Körperschaften
3. Titelwörter
4. Buchtitel
5. Systematik (sog. Regensburger Verbundklassifikation)
6. Verlage, ISSN, ISBN

UB Leipzig, Zweigstelle Informatik/Rechenzentrum:

1. Namen von Personen
2. Stichwörter und Schlagwörter
3. Titel

Die Beschreibung der Register erfolgt in einer anpaßbaren Index-Parameterdatei, die mit Hilfe der Exportsprache erstellt wird. Die Exportsprache wird von [Eve95] als “der schwierigste Teil

des Systems" bezeichnet. Datensätze, die nicht über diese Register lokalisierbar sind, können nur über ein Volltext-Suchprogramm gefunden werden [Eve95]. Damit scheinen die Möglichkeiten zur Datenauswertung sehr eingeschränkt. Insbesondere fehlt die Möglichkeit, Ad-Hoc-Anfragen bezüglich beliebiger Attribute (Kategorien) eines Datensatzes zu stellen. So ist es dem Benutzer z.B. nicht möglich, alle Titel, die nach 1994 erschienen sind, auszuwählen.

Die zur Dateneingabe und -auswertung zu verwendenden Sprachen werden von den Entwicklern selbst als kompliziert bezeichnet, damit ist der Einsatz diesen Systems immer mit einem sehr **hohen Lernaufwand** verbunden [Eve94]. Jedoch werden bestimmte Konfigurationsdateien mitgeliefert, so für das *allegro*-Standardschema, welches in der Zweigstelle Informatik/URZ verwendet wird, für MAB (das Austauschformat des deutschen Bibliothekswesens), PICA (das Pica3-System des Niedersächsischen Bibliotheksverbundes) und USMARC (ein international verbreitetes Schema) sowie für AV-Medien und Adressenverwaltung [Eve94]. Diese Konfigurationen können nach Belieben geändert und angepaßt werden, daher sind de facto in jeder Bibliothek andere Konfigurationen im Einsatz, so daß der Datenaustausch zwischen Bibliotheken, die *allegro* verwenden, keinesfalls problemlos ist. So werden z.B. derzeit die in der Zweigstelle Informatik/URZ bereits erfaßten Titel in der Hauptbibliothek erneut erfaßt, da ein Datenaustausch zwischen den beiden Systemen, die verschiedene Kategoriensysteme verwenden, nicht ohne größeren Aufwand möglich ist.

Allegro wurde entwickelt für den Betrieb im PC-Umfeld und ist mehrbenutzerfähig, aber auch im Einzelplatzbetrieb auf Maschinen der unteren Leistungsklasse einsetzbar [Eve95]. Wird *allegro* im **Mehrplatzbetrieb** verwendet, so kann jeweils nur ein Benutzer Änderungen an der Datenbank vornehmen; er muß dann die gesamte Datenbank explizit gegen Änderungen sperren und nach dem Bearbeitungsvorgang diese Sperre wieder aufheben [Eve94]. Ein vom DBMS geregelter Mehrbenutzerbetrieb ist also nicht gegeben, eine Transaktionsverwaltung nicht vorhanden.

Allegro bietet eine recht große Flexibilität, wirkt dadurch jedoch auch sehr unübersichtlich und ist wesentlich fehleranfälliger als ein relationales DBMS, da die **Integritätssicherung** nicht durch das Datenbankprogramm erfolgt, sondern vom *allegro*-Programmierer realisiert werden muß. In vielen Fällen unterbleiben diese Maßnahmen, und es entstehen inkonsistente Datenbanken mit hohem Redundanzanteil. Die unzureichende Erfüllung der Anforderungen bezüglich der **logischen Datensicherheit** (semant. Integrität) ist sicherlich eines der Hauptprobleme beim Einsatz von *allegro*.
Auch bezüglich der **physischen Datensicherheit** gibt es Kritikpunkte: Es existieren eine LOG-Datei, in der alle Änderungen an der Datenbank protokolliert werden, und das Programm

`UPDATE` zur Wiederherstellung des ursprünglichen Zustandes nach einem Fehlerfall (Systemabsturz, Virenbefall, Hardwarefehler etc.). Der Benutzer muß über den Menüpunkt `SICHERN` Sicherungskopien anlegen und im Bedarfsfall das Programm zur Wiederherstellung der Datenbank starten. Für das Sichern werden die MS-DOS-Befehle `BACKUP` und `RESTORE` verwendet. Dabei werden nur die eigentlichen Titeldaten gesichert, der Index muß anschließend neu angelegt werden. Konfigurations- und Parameterdateien wie z.B. die Index-Parameterdatei sind nur bei genauer Kenntnis der Datenbank und ausreichender Dokumentation rekonstruierbar. Der Anwender muß sich selbst um Sicherheitsvorkehrungen Gedanken machen [Eve94].

Das System *allegro* stellt kein ausreichendes Konzept zum **Schutz der Daten** vor unberechtigtem Zugriff bereit. Die Dateien sind über das Betriebssystem frei zugänglich und manipulierbar. Die Sicherheit der Daten hängt also ab von der Zugangsbeschränkung zu dem Rechner, auf dem sie gespeichert sind. Eine differenzierte Rechtevergabe für verschiedene Nutzerkreise ist damit nicht realisierbar.

Da *allegro* nicht auf dem relationalen Datenmodell beruht, gibt es auch keine SQL-**Schnittstelle**. Es existiert jedoch ein API (application programmer's interface) als Programmierschnittstelle zu C, die die Erstellung von zusätzlichen Anwendungsfunktionen ermöglicht. Für den Bibliothekar und die OPAC-Benutzer stehen 2 verschiedene Schnittstellen bereit. Bezüglich der Benutzerschnittstelle für den Bibliothekar stellt [Eve95] folgendes fest: "Die *allegro*-Oberfläche für den Bibliothekar wird von unvorbereiteten Windows- oder gar Mac-verwöhnten Anwendern als sehr schlecht eingestuft und oftmals kaum richtig verstanden." Zwischen Datenbankadministrator und Anwendungsprogrammierer wird dagegen nicht differenziert, da die Datenhaltung nicht von den Anwendungsprogrammen getrennt ist. Alle die Datenbank betreffenden Informationen sind in über das Dateisystem zugänglichen Dateien abgespeichert. Die Parameterdateien können und sollen mit normalen Texteditoren bearbeitet werden.

Auf **Datenunabhängigkeit** wurde bei der Entwicklung von *allegro* betont verzichtet, es existiert kein von der Anwendung getrenntes Datenmodell. Datenverwaltung und Anwendungsprogramme sind eng verwoben, die Anpassung der Anwendungen an spezielle Bedürfnisse ist mit Hilfe von Parameterdateien möglich. Damit hat der Anwender zwar eine recht hohe Flexibilität bei der Anpassung der Datenformate an das in der jeweiligen Bibliothek verwendete Kategorienschema, die generelle Funktionalität der Anwendungsprogramme ist jedoch ohne gründliches Studium der zugrundeliegenden Speicherstrukturen nicht änderbar. Die Speicherung der Daten erfolgt in der Eingabereihenfolge der Datensätze, die Daten werden mit den zugehörigen Kategorien fortlaufend in einer Datei abgespeichert. Steuerzeichen dienen zur Trennung der einzelnen Datensätze und kodieren semantische Zusammenhänge. Der Zugriff

auf die Daten ist damit zunächst nur über die mitgelieferten Programmodule möglich. Da jedoch die Zugriffsregister strukturell von den eigentlichen Daten getrennt sind, ist zumindest eine Neuindexierung und Änderung der Zugriffsregister jederzeit möglich.

Bezüglich der **Erweiterungsmöglichkeiten** läßt sich sagen, daß *allegro* an der UB Braunschweig gepflegt und weiterentwickelt wird und seit 1990 auch vom Niedersächsischen Ministerium für Wissenschaft und Kultur gefördert wird [Eve94]. Eine UNIX-Version des Systems wurde 1994 erstmals freigegeben [Eve95], dabei scheint die Portierung auf verschiedene Plattformen noch Schwierigkeiten zu bereiten. Inzwischen ist auch die Anbindung von *allegro*-Datenbanken an das WWW und die Online-Recherche im OPAC (Online Public Access Catalog) möglich.

Zusammenfassung:

Obwohl *allegro* von seinen Entwicklern als programmierbares Datenbanksystem bezeichnet wird, ist es eher als Dateiverwaltungssystem einzustufen, da die meisten der im vorigen Kapitel aufgeführten Anforderungen an DBS von *allegro* nur unzureichend erfüllt werden.

Zwar bietet *allegro* eine große Flexibilität bezüglich des verwendeten Datenformats und auch die vielen bereits vorhandenen Anwendungsmodule, die kostengünstig erhältlich sind, fallen positiv ins Gewicht. Jedoch wird die hohe Flexibilität erkauft durch mangelnde Integritätssicherung und fehlende Standardisierung der Schnittstellen. Auch den Anforderungen bezüglich Datenunabhängigkeit, Datenschutz und kontrolliertem Mehrnutzerbetrieb wird *allegro* nicht gerecht. Der letztgenannte Aspekt ist zwar im Einzelplatzbetrieb ohne Bedeutung, jedoch wird es mit zunehmender Vernetzung der Arbeitsplätze in Zukunft kaum noch “stand alone” Anwendungen geben. Da auch die Forderung nach Wiederverwendbarkeit und Mehrfachnutzung bereits vorhandener Datenbestände weiter zunehmen wird, sind in langer Sicht Lösungen gefragt, die den Datenaustausch zwischen verschiedenen Systemen beziehungsweise den Zugriff auf die Daten über Systemgrenzen hinweg unterstützen. Dazu sind vor allem standardisierte Schnittstellen sowie zuverlässige und effiziente Konzepte zur Integritätssicherung notwendig.

4. Diskussion der Eignung eines relationalen DBMS

Im Gegensatz zu einem speziell für die Anwendung im Bibliothekswesen entwickelten System wie *allegro* sind kommerzielle relationale Datenbankmanagementsysteme wie z.B. Sybase und Oracle auf Universalität ausgerichtet. Trotzdem bieten sie eine sehr hohe Leistungsfähigkeit, die auch bei vielen gleichzeitigen Benutzern zumindest für einfache Anfragen schnelle Bearbeitungszeiten garantiert. Im folgenden wird des öfteren auf spezifische Merkmale des RDBMS Sybase [Syb1] Bezug genommen, da dieses System am Institut für Informatik der Universität Leipzig zur Verfügung steht und auch für die Realisierung der Literaturverwaltung verwendet wurde.

Die Grundlage von relationalen DBMS (RDBMS) ist das Anfang der 70er Jahre von E. F. Codd formulierte Relationenmodell, welches mit Mitteln der Relationenalgebra formal beschrieben wurde. Die Stärke des relationalen Modells liegt in der Modellierung einfach strukturierter Daten, welche von RDBMS sehr effizient verwaltet werden können. Da das relationale Modell anwendungsneutral ist, unterstützt es die Datenunabhängigkeit. Es ermöglicht zudem eine bedeutende Verringerung der redundanten Datenhaltung [HW93].

Ein Hauptvorteil von RDBMS ist sicherlich die Tatsache, daß mit SQL eine **standardisierte Anfragesprache** auf hohem abstrakten Niveau zur Verfügung steht. SQL ist weit verbreitet, relativ leicht erlernbar und bietet weitreichende Möglichkeiten zur Dateneingabe, Datenmanipulaion und besonders auch zur effizienten Datenauswertung. Insbesondere bietet eine SQL-Schnittstelle die Möglichkeit, Ad-Hoc-Anfragen unabhängig von bestehenden Anwendungen zu stellen. Auch der geringere Entwicklungsaufwand und die erleichterte Wartbarkeit von Anwendungsprogrammen sprechen für SQL.

Die Grundlage des durch ein RDBMS ermöglichten **geregelten Mehrbenutzerbetriebes** ist das Transaktionskonzept, welches die Eigenschaften Atomarität, Konsistenz, Isolation und Dauerhaftigkeit für alle Transaktionen sicherstellt. Jeder der parallelen Benutzer arbeitet im logischen Einnutzerbetrieb, eine möglichst geringe gegenseitige Behinderung wird durch eine Sperrverwaltung mit feinen Sperrgranulaten erreicht.

RDBMS bieten umfangreiche systemseitige Unterstützung bezüglich **logischer und physischer Datensicherheit**. So gestatten RDBMS bereits bei der Definition eines Schemas die Spezifizierung von Bedingungen, die die semantischen Aspekte der **Datenintegrität** betreffen (Primär- und Fremdschlüsselbedingungen, Wertebereichsbeschränkungen, Standardwerte), z.B. in der SQL-Anweisung `CREATE TABLE`. Eine weitere Möglichkeit zur Überprüfung der

Einhaltung von Integritätsbedingungen durch das DBMS stellen Trigger dar, die bei einer Änderung der Daten aktiviert werden und gegebenenfalls weitere Aktionen auslösen.
Da die automatische Überwachung der Einhaltung von Integritätskriterien durch das DBMS erfolgt, wird dem Anwendungsprogrammierer die Sorge um die Einhaltung der Datenintegrität genommen, insbesondere muß diese Funktionalität nicht durch jede einzelne Anwendung erneut bereitgestellt werden. Die physische Datensicherheit, d.h. Ausfallsicherheit und Ausfallbehandlung, wird durch Logging- und Recovery-Mechanismen gewährt. Sicherungsmaßnahmen sind nicht durch den Nutzer durchzuführen, sondern liegen in der Verantwortung des Datenbankadministrators.

Da die zentrale Kontrolle über alle Daten beim Datenbankadministrator liegt, ist dieser auch für den **Datenschutz** verantwortlich. Das System unterstützt ihn dabei mit Möglichkeiten zur Benutzerauthentisierung und differenzierten Rechtevergabe an Datenbankbenutzer. Ein Zugriff auf die Datenbank ist nur mit gültigem Login und Paßwort möglich, die Vergabe von unterschiedlichen Rechten für verschiedene Zugriffsarten separat für jedes Datenobjekt kombiniert mit der automatisierten Zugriffskontrolle stellt sicher, daß kein unberechtigter Zugriff auf die Daten erfolgen kann.

Mit einem RDBMS wie Sybase ist die Realisierung von sehr differenzierten **Nutzerschnittstellen** möglich. Die Anwendungsentwicklung wird unterstützt durch die standardisierte SQL-Schnittstelle und die Programmierschnittstelle zu C und embedded SQL, es existieren diverse Entwicklungstools wie z.B. APT und Powerbuilder, die den Entwicklungsaufwand erheblich verringern und die Erstellung von komfortablen graphischen Benutzerschnittstellen für den Endanwender unterstützen [Syb1].

Das relationale Modell sichert sowohl die logische als auch die physische **Datenunabhängigkeit**. Letztere bezeichnet die Unabhängigkeit der Anwendungsprogramme von der physischen Repräsentation der Daten, den unterliegenden Speicher- und Indexstrukturen. Damit ist es den Herstellern von RDBMS möglich, mit der rasanten Entwicklung der Bereiche Hard- und Software Schritt zu halten und ihre Systeme ständig weiterzuentwickeln. Der Anwender kann problemlos auf ein neueres System umsteigen und alte Anwendungsprogramme weiterverwenden. Logische Datenunabhängigkeit bezeichnet die Tatsache, daß Anwendungsprogramme auf spezifischen Sichten des konzeptionell Schemas arbeiten können und somit eine anwendungseigene Sicht auf die Daten haben können, die sich bei Änderungen oder Erweiterungen des zugrundeliegenden konzeptionellen Schemas nicht ändert. Dies sichert eine hohe Flexibilität bezüglich späterer Erweiterungen eines Schemas, da bestehende Anwendungsprogramme unverändert weiterbenutzt werden können. Eine vollständige logische Datenunabhängig wird jedoch vom relationalen Modell nicht erreicht.

Die Gewährleistung einer höchstmöglichen **Effizienz** ist eine Aufgabe, die vom DBMS, nicht durch die Anwendungsprogramme gelöst werden sollte. Relationale Datenbankenverwaltungssysteme bieten systemseitige Anfrageoptimierung. Der Optimiser wählt den effektivsten Ausführungsplan für eine Anfrage in Abhängigkeit von Systemstatistiken und den zur Verfügung stehenden Indizes. Die Statistiken über Zugriffshäufigkeiten und dergleichen werden vom System selbst geführt, mit dem Befehl `UPDATE STATISTICS` werden die vom Optimiser verwendeten Informationen aktualisiert [Kir93]. Aufgabe des Datenbankadministrators ist die richtige Konfiguration des DBMS und das Datenbank-Tuning: Durch die Anpassung von Indizes, Puffergrößen, Datenbereichen, Log-Bereichen und weiteren Parametern läßt sich das Leistungsverhalten des DBMS verbessern [HW93].

Ein kommerzielles RDBMS wie Sybase wird natürlich ständig weiterentwickelt und bietet umfangreiche zusätzliche Tools wie den Backup-Server, Enterprise CONNECT und web.sql, mit denen sich weitreichende Möglichkeiten für die **Erweiterung** bestehender Anwendungen bieten, so zum Beispiel die Anbindung einer Datenbank an das World Wide Web (WWW) mit Hilfe von web.sql [Syb2]. Auch ein ODBC-Treiber ist für Sybase erhältlich, mit dem Windows-basierte Anwendungen wie Microsoft Access, Lotus 1-2-3, Microsoft Excel u.a. auf in einer Sybase-Datenbank gehaltene Daten zugreifen können [Syb1]. Aufgrund der von RDBMS unterstützten Interoperabilität können verschiedene Systeme dieselben Daten nutzen, ohne zuvor einen physischen Datenaustausch durchführen zu müssen. Im langfristigen Einsatz ist auch die Skalierbarkeit, d.h. die Tatsache, daß bei steigenden Anforderungen ohne Probleme auf ein leistungsfähigeres System umgestiegen werden kann, von großer Bedeutung.

Zusammenfassung:

RDBMS bieten vor allem in den Bereichen Datenunabhängigkeit, geregelter Mehrbenutzerbetrieb, Datenintegrität und Datensicherheit Konzepte, die weit über den Ansatz anderer Systeme hinausgehen. Zu den Vorteilen von RDBMS zählen auch die von diesen gebotene standardisierte SQL-Schnittstelle, die Skalierbarkeit sowie die Interoperabilität. Jedoch wird die von *allegro* gebotene Flexibilität bezüglich der Änderung des Datenbankschemas nicht erreicht. Zwar sind Erweiterungen des Schemas um Relationen und Attribute problemlos möglich, die Änderung oder Streichung bereits vorhandener Attribute ist jedoch nur mit einigem Aufwand zu realisieren. Auch die Kosten für ein RDBMS liegen um einiges höher als die für die Anschaffung von *allegro* notwendigen Beträge.

5. Entwurf eines relationalen Schemas für ein Literaturverwaltungssystem

5.1. Zielstellung

Das Modell für die Literaturverwaltung soll zum einen die effiziente Verwaltung der bibliographischen Nachweise von Literaturbeständen ermöglichen, wobei die Einbeziehung elektronischer Publikationen eine große Rolle spielt. Zum anderen soll auch die Ausleihe von Literatur modelliert werden, dazu gehört die Verwaltung der Exemplare und Bibliotheksbenutzer.

Ein wichtiger Punkt ist die Sicherung der Konsistenz der Daten, welche durch Vermeidung von Redundanz bei der Datenerfassung und durch die Festlegung von Integritätsbedingungen unterstützt werden muß.

5.2. Logischer Entwurf

Am Beginn der Entwurfsphase stand eine eingehende Beschäftigung mit den für die Bibliotheksarbeit wichtigen Begriffen, Handlungsabläufen und Daten sowie eine gründliche Analyse der von *allegro* verwalteten Daten.
Folgende **Entity-Mengen** kristallisierten sich dabei zunächst heraus:

- ❑ Titel (Dokument)
- ❑ Autoren
- ❑ Verlage
- ❑ Schlagwörter
- ❑ Reihen

Bei der bisherigen Erfassung der Literaturdaten mit *allegro* wurden alle Angaben für Autoren, Verlage, Schlagwörter und Reihen für jeden Titel erneut aufgenommen. Die dadurch bedingte Redundanz führte zu vielen Inkonsistenzen aufgrund verschiedener Schreibweisen. Durch die Trennung dieser Angaben von den Titeleinträgen erfolgt nun die Erfassung der betreffenden Daten nur noch einmal, wodurch die durch Eingabefehler entstehenden Inkonsistenzen vermieden werden können. Auch die Verwaltung von Verlags- und Personendaten vereinfacht sich dadurch, da Änderungen der Daten (z.B. neue Verlagsadresse) nur an einer Stelle durchgeführt werden brauchen.

Die in *allegro* alle einheitlich als **Titel** erfaßten Bibliotheksbestände weisen bei genauer Betrachtung beträchtliche Unterschiede bei den je Typ zu erfassenden Angaben auf. Da damit kaum einheitliche Attribute für die verschieden Ausleihobjekte gefunden werden konnten, wurde der Aufteilung in die folgenden Entity-Mengen der Vorzug gegeben: **Buchtitel**, **Tagungsband**, **Nonprintmedien**, **Dissertation** und **Report**. **Tagungsbände** unterscheiden sich von den, hier mit **Buchtitel** bezeichneten, normalen Büchern besonders dadurch, daß sie Beiträge vieler verschiedener Autoren enthalten und einer bestimmten Konferenz zugeordnet werden. Eine weitere Entity-Menge **Nonprintmedien** soll die gerade in einer Informatikbibliothek häufig anfallenden CD-ROMs mit Programmpaketen, Nachschlagewerken und dergleichen aufnehmen. In den Entwurf wurden außerdem die bisher nicht erfaßten Typen **Artikel** und **Zeitschriften** aufgenommen. Die Entity-Menge **Abstract** soll die zu den Literaturstellen gehörenden Abstracts aufnehmen.

Als **Autoren** werden hier allgemein alle am Entstehen eines Werkes beteiligten Personen verstanden. Dies betrifft vor allem die Verfasser und Herausgeber. Aber auch die Einbeziehung von Übersetzern, Illustratoren usw. ist denkbar.

Eine genauere Untersuchung der bisher als **Verlag** erfaßten Daten ergab, daß bei Reports und Dissertationen zumeist Universitäten und Hochschulen als Verlage aufgenommen wurden. Hier erschien eine Trennung in die disjunkten Entity-Mengen **Verlag** und **Institution** sinnvoll. Institutionen können auch an der Ausrichtung von Konferenzen und Tagungen beteiligt sein.

Als **Reihen** gelten neben den Serien und Reihen der Verlage und den Hochschulschriftenreihen auch Konferenzen, die wiederholt unter dem gleichen Namen stattfinden.

Für die Modellierung der recht komplizierten Vorgänge bei der Verwaltung von Zeitschriften wurden die Entity-Mengen **Zeitschrift** (Zeitschriftentitel), **Zeitschriftenausgabe** und **Zeitschriftenband** eingeführt. Eine Zeitschrift hat einen Titel, einen Verlag und eine ISSN. Um die zu einem Fachgebiet erhältlichen Zeitschriften ausfindig machen zu können, muß auch die Zuordnung von Schlagwörtern zu Zeitschriftentiteln möglich sein. Die in periodischen Zeitabständen erscheinenden Ausgaben (Hefte) einer Zeitschrift werden als Zeitschriftenausgaben erfaßt, welche bestimmte Artikel enthalten können. Nach einer gewissen Zeit erfolgt das Binden der einzelnen Ausgaben zu einem Zeitschriftenband, so daß die Ausgaben nun nicht mehr einzeln ausleihbar sind.

Um die Ausleihe von Literaturbeständen zu modellieren, war die Einführung der Entity-Mengen **Exemplar** und **Leser** (Bibliotheksbenutzer) notwendig. Das Exemplar enthält exemplar-

spezifische Attribute wie Signatur und Zugangsnummer sowie Informationen über die Ausleihbarkeit und den Zustand eines konkreten Exemplars. In der Entity-Menge Leser werden die Bibliotheksbenutzerdaten verwaltet, für die Leser muß eine Lesekartennummer vorhanden sein, welche auch als Schlüssel dient, das Studienfach wird erfaßt und ebenso das Geburtsdatum. Der Ausleihvorgang wird durch eine n:m Beziehung zwischen Exemplar und Leser modelliert. Der Ausleihe müssen Angaben über Ausleih- und Rückgabedatum, Verlängerungen u.ä. zugeordnet werden. Auch Vorbestellungen werden über diese Beziehung modelliert. Falls bereits abgeschlossene Ausleihvorgänge zum Zwecke des Nachweises oder für statistische Auswertungen dokumentiert werden müssen, ist die Einführung einer Entity-Menge Ausleih_history in Erwägung zu ziehen.

Um auch die Verwaltung von nicht zu den Bibliotheksbeständen gehörender Literatur zu ermöglichen, wurde die Entity-Menge **Besitzer** hinzugefügt. Damit können zugleich auch weitere Bibliotheken in derselben Datenbank verwaltet werden.

Elektronische Versionen von Büchern, Artikeln, Reports usw. werden bei der Erfassung genauso wie in gedruckter Form vorliegende Titel behandelt und der entsprechenden Relation zugeordnet. Um dem Benutzer den Zugriff auf derartige Online-Dokumente zu ermöglichen, wird jeweils die URL des Dokuments (z.B. WWW- oder FTP-Adresse) als Attribut erfaßt.

Die zur Modellierung der Literaturverwaltung benötigten Entity-Mengen werden im folgenden noch einmal aufgelistet:

Literaturbestand (mit Ausnahme der Zeitschrift sind dies die ausleihbaren Bibliotheksobjekte):

- ❑ Report
- ❑ Dissertation
- ❑ Nonprintmedien
- ❑ Zeitschrift
- ❑ Zeitschriftenausgabe
- ❑ Zeitschriftenband
- ❑ Artikel
- ❑ Buchtitel
- ❑ Tagungsband

zur Beschreibung und Suche im Literaturbestand:

- ❑ Reihe
- ❑ Abstract
- ❑ Schlagwort
- ❑ Autoren
- ❑ Verlag
- ❑ Institution

zur Modellierung von Ausleihvorgängen:

- ❑ Exemplar
- ❑ Leser
- ❑ Besitzer

5.3. Entity-Relationship-Diagramm

Die Zusammenhänge zwischen den im vorhergehenden Abschnitt vorgestellten Entity-Mengen werden im Entity-Relationship-Diagramm der Abbildung 5.1 dargestellt. Auf die Darstellung der Attribute und Kardinalitätsrestriktionen im Diagramm wird aus Platzgründen verzichtet. Die Attribute werden im Kapitel 5.4. näher beschrieben. Auf die Kardinalitätsrestriktionen wird in der folgenden Erläuterung näher eingegangen.

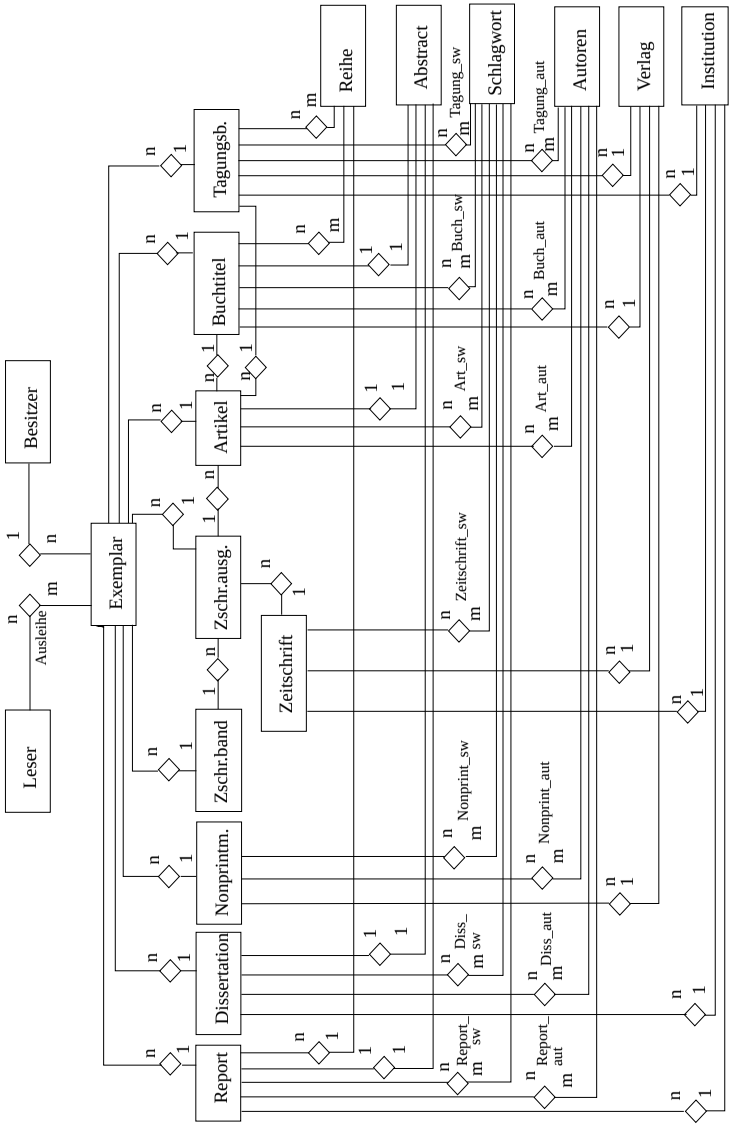

Abbildung 5.1 Entity-Relationship-Diagramm für ein Literaturverwaltungssystem

Erläuterung des Entity-Relationship-Diagramms:

Die Entity-Mengen im oberen Abschnitts des Diagramms sind die zur Modellierung der Ausleihe benötigten, darunter stehen in einer Reihe nebeneinander die als Literaturstellen oder Titel charakterisierbaren Entity-Mengen. Die Literaturstellen können in einer beliebigen Anzahl von Exemplaren vorliegen, wobei ein Exemplar immer zu genau einem Titel gehört. Es ist auch möglich, daß von einem Titel keine gedruckten Exemplare vorliegen, da dieser Titel sich auf eine Online-Publikation bezieht, welche über die in den Titeldaten gespeicherte URL abrufbar ist.

Als Zeitschrift wird der Zeitschriftentitel erfaßt, nicht die einzelnen Ausgaben einer Zeitschrift, welche in die Relation Zeitschriftenausgabe gehören. Da ein Zeitschriftentitel nicht physisch existiert, kann es von ihm auch keine Exemplare geben. Zeitschriftenausgaben können einem Zeitschriftenband zugeordnet sein, wodurch das Binden von mehreren Ausgaben zu einem Band dargestellt wird.

In der rechten vertikalen Reihe sind die Entity-Mengen zu finden, die den Literaturstellen gewissermaßen als Suchbegriffe zugeordnet werden können. Dabei läßt sich auch eine Volltextsuche in den Abstracts realisieren. Ein Abstract muß nicht zu jedem Titel vorliegen.

Zwischen Autoren bzw. Schlagwort und den Literaturstellen bestehen jeweils n:m Beziehungen, da ein Titel beliebig viele Schlagwörter und beteiligte Personen haben kann und umgekehrt jedes Schlagwort und jeder Autor mehreren Titeln zugeordnet werden kann. Ebenso besteht zwischen Leser und Exemplar eine n:m Beziehung, da ein Leser mehrere Exemplare ausleihen oder vorbestellen kann und ein Exemplar zwar nur von einem Leser ausgeliehen, jedoch von weiteren Lesern vorbestellt sein kann. Ob die Anzahl der von einem Leser ausgeliehenen Exemplare begrenzt ist, hängt von den Gegebenheiten der jeweiligen Bibliothek ab, auch die Anzahl der Vorbestellungen je Exemplar kann eventuell eingeschränkt sein.

Die n:m Beziehungen zwischen Tagungsband bzw. Buchtitel und Reihe sind in ihrer Kardinalität begrenzt, da ein Buchtitel nicht zu mehr als zwei Reihen gehören kann, ebenso kann ein Tagungsband nur einer Konferenz und einer Reihe zugeordnet werden.

5.4. Relationales Schema

Im folgenden werden die Relationen für die Entity-Mengen und die n:m Beziehungen aus dem Entwurf beschrieben. Mit den in Abb. 5.1 verwendeten Bezeichnungen werden auch die Relationen benannt, jedoch werden die im relationalen Modell verwendeten Bezeichner grundsätzlich kleingeschrieben. Die zu einer Relation gehörenden Attribute werden nachfolgend jeweils in einer Tabelle aufgelistet und näher erläutert.

Allgemeine Bemerkungen:
Primärschlüssel werden durch Unterstreichung gekennzeichnet, Fremdschlüssel sind *schräggedruckt*. Generell wurde die Einführung interner Identifikatoren als Primärschlüssel, die unabhängig von sonstigen Attributen sind, vorgezogen. Fremdschlüssel haben immer die gleiche Bezeichnung wie in der Tabelle, in der sie Primärschlüssel sind. Da Sybase für Fremdschlüssel, die als `FOREIGN KEY` definiert sind, keine NULL-Werte zuläßt, muß die Sicherung der referentiellen Integrität dieser Fremdschlüssel durch Trigger erfolgen, d.h. es muß sowohl beim Einfügen eines Satzes überprüft werden, ob ein Eintrag zu dem verwendeten Fremdschlüssel existiert, als auch beim Löschen eines Satzes in der referenzierten Tabelle, ob abhängige Einträge existieren.

Bei der Entscheidung darüber, ob eine bestimmte Angabe als ein Attribut oder aber aufgesplittet in mehrere Attribute modelliert werden sollte, wurde der Darstellung als einzelnes Attribut dann der Vorzug gegeben, wenn die Angabe nur informativ ist, jedoch nicht als Suchbedingung verwendet werden soll. Dies gilt zum Beispiel für die Attribute `preis`, `umfang` und `auflage` der Relation `buchtitel` sowie für die Adressenangabe bei den Verlagen und Institutionen.

Das Attribut `url` gibt den Uniform Resource Locator (z.B. WWW- oder FTP-Adresse) für elektronische Versionen von Titeln bzw. für die Homepages von Autoren, Verlagen und Institutionen an.

Das zur Beschreibung der Literaturbestände verwendete Attribut `cr` bezieht sich auf die CR-Klassifikation, das Klassifikationsschema für Informatikliteratur der Zeitschrift Computing Reviews, welches auch an der UB Leipzig in der Zweigstelle Informatik/Rechenzentrum benutzt wird.

In den folgenden Tabellen bedeutet der Kommentar N (NULL), daß der Wert des entsprechenden Attributes NULL sein darf, während NN (NOT NULL) darauf hinweist, daß für dieses Attribut NULL ein nicht erlaubter Wert ist.

Relationen für die Entity-Mengen:

tagungsband

Attribute	Typ	Kommentar
tagungsid	integer	NN, Primärschlüssel
titel	varchar(255)	NN, Titel des Tagungsbands laut Vorlage
tagungsort	varchar(80)	N
tagungsdatum	varchar(20)	N, kann vom Erscheinungsjahr abweichen
kschaft	varchar(100)	N, Körperschaft
isbn	varchar(15)	N
jahr	smallint	N, Erscheinungsjahr
umfang	varchar(50)	N
preis	varchar(20)	N, Preis + Währung
konfid	integer	N, Fremdschlüssel (Trigger), Verweis auf entsprechende Konferenz in Relation Reihe
konfnr	smallint	N, Zählung der Konferenz
reihenid	integer	N, Fremdschlüssel (Trigger), Verweis auf Serie in Relation Reihe
bandnr	varchar(10)	N, Bandnummer der Serie
verlagsid	integer	NN, Fremdschlüssel
instid	integer	N, Fremdschlüssel (Trigger)
url	varchar(200)	N, URL für Online-Dokument
bemerkung	varchar(100)	N

Bemerkungen:

Der Titel eines Tagungsbandes entspricht dem auf der Haupttitelseite des Buches angegebenen Titel, z.B. “Very large data bases : 19th International Conference on Very Large Data Bases, August 24th - 27th, 1993, Dublin, Ireland ; proceedings”. Tagungen werden nach RAK (“Regeln für die alphabetische Katalogisierung”) jedoch auch als Körperschaften behandelt und dementsprechend erfaßt. Für obiges Beispiel sähe dieser Eintrag folgendermaßen aus: “International Conference on Very Large Data Bases <19, 1993, Dublin>”.

Die n:m Beziehung zwischen `tagungsband` und `reihe` wurde nicht in eine eigene Relation umgewandelt, da ein Tagungsband nur zu einer Konferenz gehören kann, auf welche durch den Fremdschlüssel `konfid` verwiesen wird. In der Relation `reihe` werden sowohl Serien als auch Konferenzen verwaltet, wobei die ersteren mit dem Typ `'s'` und die Konferenzen mit dem Typ `'k'` gekennzeichnet sind. Es muß sichergestellt werden, daß die Reihe, auf die `konfid` verweist, den Typ `'k'` hat. Mit `konfnr` kann die Zählung der Konferenz vermerkt werden, für das oben erwähnte Beispiel wäre die `konfnr` = 19. Dabei muß sichergestellt werden, daß die `konfnr` nur dann eingegeben wird, wenn die `konfid` einen Wert ungleich NULL hat. Außerdem kann ein Tagungsband auch zu einer Serie gehören, dafür steht der Fremdschlüssel `reihenid` zur Verfügung, der in der Relation `reihe` auf einen Eintrag mit dem Typ `'s'` verweisen muß. Das Attribut `bandnr` bezieht sich auf diese Serie und darf nur angegeben werden, wenn die `reihenid` ungleich NULL ist. Nicht alle Serien haben numerierte Bände. Das Attribut `instid` kann auf die Institution, die die Tagung ausgerichtet hat, verweisen.

buchtitel

Attribute	Typ	Kommentar
buchid	integer	NN, Primärschlüssel
gesamtid	integer	N, buchid des übergeordneten Titels (Gesamttitels) bei mehrbändigen Werken, (Trigger)
bandnr	smallint	N, Bandnummer bei mehrbändigen Werken
baende	smallint	N, Anzahl der Bände, die zu einem Haupteintrag existieren
hstitel	varchar(255)	N, Hauptsachtitel
estitel	varchar(255)	N, Einheitssachtitel (Titel in Originalfassung)
isbn	varchar(15)	N
isbn_2	varchar(15)	N
auflage	varchar(70)	N, z.B. 1. erweiterte, überarb. etc.
jahr	smallint	N, Erscheinungsjahr
medium	varchar(20)	NN, z.B. Buch, Mikrofiche
umfang	varchar(50)	N
preis	varchar(20)	N, Preis + Währung
bezugsinfo	varchar(30)	N, Informationen zum Bezug des Titels
verf_in_vorlform	varchar(130)	N, Verfasserangabe in Vorlageform
hsv	varchar(100)	N, Hochschulschriftenvermerk
kschaft1	varchar(100)	N, 1. Körperschaft
kschaft2	varchar(100)	N, 2. Körperschaft
reihenid1	integer	N, Fremdschlüssel (Trigger)
bandnr1	varchar(10)	N, Bandnummer bzgl. der 1. Reihe
reihenid2	integer	N, Fremdschlüssel (Trigger)
bandnr2	varchar(10)	N, Bandnummer bzgl. der 2. Reihe
verlagsid	integer	NN, Fremdschlüssel
abstractid	integer	N, Fremdschlüssel (Trigger)

buchtitel

Attribute	Typ	Kommentar
url	varchar(200)	N, URL für Online-Dokument
cr	varchar(80)	N, CR-Klassifikation als String
bemerkung	varchar(100)	N

Bemerkungen:

Der Bibliothekar unterscheidet zwischen Hauptaufnahmen und Bandaufnahmen. Eine Bandaufnahme wird dann notwendig, wenn ein mehrbändiges Werk bzw. eine mehrbändige Ausgabe eines Werkes vorliegen. Die Hauptaufnahme enthält dann die für das Gesamtwerk gültigen Angaben, die Unterbände werden einzeln erfaßt. Dabei läßt sich nicht generell sagen, welche Angaben der Hauptaufnahme und welche den Bänden zugeordnet werden, da dies von Titel zu Titel verschieden sein kann. Haupt- und Bandaufnahmen haben daher prinzipiell die gleichen Attribute und werden auch beide als Buchtitel erfaßt. Um die Verbindung zwischen der Hauptaufnahme und den einzelnen Bänden herzustellen, wurden die Attribute `gesamtid`, `bandnr` und `baende` eingeführt. Bei einem Haupteintrag gibt `baende` die Anzahl der zu diesem Eintrag gehörenden Bandaufnahmen an. Für die Bandaufnahmen sind die anderen beiden Attribute von Bedeutung: `bandnr` ist die Nummer des Bandes und `gesamtid` gibt die `buchid` des zu diesem Band gehörenden Haupteintrages an, dessen Existenz durch einen Trigger geprüft werden muß. Eine hierarchische Schachtelung von Bänden ist auf diese Weise ebenfalls darstellbar.

Die n:m Beziehung zwischen `buchtitel` und `reihe` wurde nicht in eine Tabelle überführt, da ein Buchtitel maximal zu zwei Reihen gehören kann, statt dessen wurden die beiden Fremdschlüssel `reihenid1` und `reihenid2` in die Relation `buchtitel` aufgenommen. Die Bandnummern beziehen sich jeweils auf die Reihe, dabei muß durch einen Trigger gesichert werden, daß eine Bandnummer nur dann eingegeben werden darf, wenn auch das zugehörige Attribut für die Reihe einen Wert hat, es hat jedoch nicht jede Reihe nummerierte Bände. Auch die referentielle Integrität der reihenids muß durch den Trigger gewährleistet werden.

Die zwischen den Relationen `buchtitel` und `abstract` bestehende 1:1 Beziehung könnte sowohl durch einen Fremdschlüssel in der Tabelle `buchtitel` als auch durch einen Fremdschlüssel in `abstract` abgebildet werden. Da jedoch der Benutzer das Abstract eher als Zusatzinformation zu einem Buch abfragen wird, ist das Einfügen des Fremdschlüssels `abstractid` in `buchtitel` sicherlich die bessere Variante. Allerdings muß ein Trigger die referentielle Integrität dieses Fremdschlüssels sichern.

artikel

Attribute	Typ	Kommentar
artikelid	integer	NN, Primärschlüssel
titel	varchar(255)	NN, Titel des Artikels
austitel	varchar(255)	N, Titel der Quelle
jahr	smallint	N
seite_von	smallint	N, Anfangsseite
seite_bis	smallint	N, letzte Seite
url	varchar(200)	N, URL für Online-Dokument
cr	varchar(80)	N, CR-Klassifikation
abstractid	integer	N, Fremdschlüssel (Trigger)
buchid	integer	N, Fremdschlüssel (Trigger)
zeitschrausgid	integer	N, Fremdschlüssel (Trigger)
tagungsid	integer	N, Fremdschlüssel (Trigger)
bemerkung	varchar(100)	N

Bemerkungen:

Die Titelangabe für die Quelle des Artikels sowie die Jahresangabe sind erforderlich, um auch Artikel korrekt erfassen und recherchierbar machen zu können, die nicht aus bereits im Katalog erfaßten Büchern, Tagungsbänden oder Zeitschriften stammen, sondern zum Beispiel als Kopie oder in elektronischer Form vorliegen. Die referentielle Integrität der Fremdschlüssel muß durch Trigger gewährleistet werden. Die Attribute `buchid, zeitschrausgid` bzw. `tagungsid` verweisen auf Einträge zu den entsprechenden Büchern, Tagungsbänden oder Zeitschriften.

zeitschrift

Attribute	Typ	Kommentar
zeitschrid	integer	NN, Primärschlüssel
titel	varchar(100)	NN, Titel der Zeitschrift
issn	varchar(14)	N
verlagsid	integer	N, Fremdschlüssel (Trigger)
instid	integer	N, Fremdschlüssel (Trigger)
seit	datetime	N, seit wann bestellt
bis	datetime	N, bis wann bestellt
url	varchar(200)	N, URL für Online-Dokumente
bemerkung	varchar(100)	N, z.B. Hinweis auf alten Namen

Bemerkungen:

Als `zeitschrift` wird der Zeitschriftentitel erfaßt, nicht die einzelnen Ausgaben (Hefte) einer Zeitschrift, welche in die Relation `zeitschriftenausgabe` gehören. Da ein Zeitschriftentitel nicht physisch existiert, kann es von ihm auch keine Exemplare geben. Da Zeitschriften nicht nur von Verlagen, sondern auch von Hochschulen und anderen Institutionen im Selbstverlag herausgegeben werden, wurden die Attribute `verlagsid` und `instid` eingeführt. Ein Trigger muß die referentielle Integrität dieser Fremdschlüssel sichern, außerdem darf nur maximal einer dieser beiden Fremdschlüssel NULL sein. Wenn sich eine von einem Verlag herausgegebene Zeitschrift trotzdem einer Institution zuordnen läßt, so werden beide erfaßt. Arbeitsberichte und Reports werden jedoch nicht als Zeitschriften erfaßt, sie werden der Relation `report` zugeordnet.

zeitschrausgabe

Attribute	Typ	Kommentar
zeitschrausgid	integer	NN, Primärschlüssel
jahrgang	varchar(10)	NN
ausgabe	varchar(10)	NN
zeitschrid	integer	NN, Fremdschlüssel
zeitschrbandid	integer	N, Fremdschlüssel (Trigger)
bemerkung	varchar(100)	N

Bemerkungen:

Die Relation `zeitschriftenausgabe` nimmt die einzelnen Ausgaben einer Zeitschrift auf. Der Fremdschlüssel `zeitschrid` verweist auf den Zeitschriftentitel, dem die Ausgabe zugehört. Werden nach einer gewissen Zeit die Ausgaben zu einem Zeitschriftenband gebunden, so muß die `zeitschrbandid` des Zeitschriftenbandes nachgetragen werden. Ein Trigger muß die referentielle Integrität sichern.

zeitschrband

Attribute	Typ	Kommentar
zeitschrbandid	integer	NN, Primärschlüssel
bandbezeichnung	varchar(30)	NN
bemerkung	varchar(100)	N

nonprintmedien

Attribute	Typ	Kommentar
nonprintid	integer	NN, Primärschlüssel
titel	varchar(255)	NN
medium	varchar(20)	NN
umfang	varchar(50)	N
verlagsid	integer	N, Fremdschlüssel (Trigger)
bemerkung	varchar(100)	N

Bemerkungen:

Als `nonprintmedien` werden hier Programmpakete, Videos, CD-ROMs usw. verstanden, die nicht in die anderen Relationen für Literaturbestände eingeordnet werden können. Dagegen werden elektronische Versionen von Büchern, Artikeln, Reports und Dissertationen genauso wie gedruckte Titel behandelt und in der entsprechenden Relation erfaßt. Falls in einer Bibliothek Videos, Tonträger o.ä. in größerem Umfang erfaßt werden, ist auch eine Erweiterung des Schemas um eine Relation für diese Ausleihobjekte denkbar, so daß auf spezielle Anforderungen eingegangen werden kann.

dissertation

Attribute	Typ	Kommentar
dissid	integer	NN, Primärschlüssel
titel	varchar(255)	NN
instid	integer	NN, Fremdschlüssel
jahr	smallint	N
isbn	varchar(15)	N
typ	varchar(1)	NN, d = diss, p = projekt, i = diplom
umfang	varchar(50)	N
abstractid	integer	N, Fremdschlüssel (Trigger)
url	varchar(200)	N, URL für Online-Dokument
cr	varchar(80)	N, CR-Klassifikation
bemerkung	varchar(100)	N

Bemerkung:

Die Relation `dissertation` kann bei Bedarf neben Dissertationen auch weitere Hochschulschriften wie Diplom- oder Studienarbeiten aufnehmen, welche anhand des Attributs `typ` unterschieden werden.

report

Attribute	Typ	Kommentar
reportid	integer	intern, NN, Primärschlüssel
titel	varchar(255)	NN
instid	integer	NN, Fremdschlüssel
jahr	smallint	NN
abstractid	integer	N, Fremdschlüssel (Trigger)
url	varchar(200)	N, URL für Online-Dokument
cr	varchar(80)	N, CR-Klassifikation
umfang	varchar(50)	N
reihenid	integer	N, Fremdschlüssel (Trigger)
lfdnr	smallint	N, fortlaufende Nummer
bemerkung	varchar(100)	N

Bemerkungen:

In der Relation `report` werden Reports und technische Berichte erfaßt, die von Universitäten, Hochschulen oder anderen Forschungseinrichtungen herausgegeben werden. Diese Reports können einer Schriftenreihe zugeordnet werden (z.B. “Hildesheimer Informatikberichte”). Das Attribut `lfdnr` bezieht sich auf die Zählung der Ausgaben dieser Reihe, die zumeist nach Jahrgängen getrennt erfolgt.

reihe

Attribute	Typ	Kommentar
reihenid	integer	NN, Primärschlüssel
reihentitel	varchar(100)	NN
kurzform	varchar(20)	N, Abkürzung z.B. bei Konferenzen
typ	varchar(1)	NN, k=Konferenz, s=Serie
bemerkung	varchar(100)	N

Bemerkungen:

Außer den Reihen und Serien wie z.B. “Lecture notes in computer science” werden auch jährlich wiederkehrende Tagungen und Konferenzen als Reihen erfaßt, um so die Suche danach durch eine einheitliche Schreibform zu unterstützen. Die Unterscheidung erfolgt durch das Attribut `typ`. Dabei muß sichergestellt werden, daß die von den Relationen `buchtitel` und `report` verwendeten Fremdschlüssel für die `reihenid` auf eine dem Typ Serie zugeordnete Reihe verweisen. Da es für viele Reihen auch Kurzformen gibt (z.B. VLDB), wurde das Attribut `kurzform` eingeführt, um eine Suche nach diesen Bezeichnern zu ermöglichen.

abstract

Attribute	Typ	Kommentar
abstractid	integer	NN, Primärschlüssel
abstracttext	text	NN, Text des Abstracts

schlagwort

Attribute	Typ	Kommentar
swid	integer	NN, Primärschlüssel
schlagwort	varchar(50)	NN, UNIQUE

autoren

Attribute	Typ	Kommentar
autorid	integer	NN, Primärschlüssel
name	varchar(30)	NN
zusatz	varchar(10)	N, Namenszusatz wie van, van der
initialen	varchar(6)	N, (nicht ausgeschriebene Vornamen)
vornamen	varchar(30)	N
institution	varchar(50)	N
url	varchar(200)	N, URL für Homepage
pseudonym	integer	N, autorid (Trigger)
bemerkung	varchar(100)	N

Bemerkungen:

Als Autoren werden sämtliche an der Entstehung eines Titels beteiligten Personen verstanden, also die Verfasser, Herausgeber, Übersetzer usw.

Die Namenszusätze werden als eigenes Attribut erfaßt, da sie die Sortierreihenfolge beim Sortieren nach dem Nachnamen nicht beeinflussen sollen, jedoch auch nicht als Vorname erfaßt werden sollen. Mit `pseudonym` wird auf einen Eintrag zu derselben Person, jedoch unter einem anderen Namen verwiesen. Dies ist der Erfassung von Zweitnamen und Pseudonymen als Attribut vorzuziehen, da bei der Ausgabe eines Titels der Name in der Vorlageform, d.h. wie auf dem Titelblatt, angegeben werden muß.

verlag

Attribute	Typ	Kommentar
verlagsid	integer	NN, Primärschlüssel
name	varchar(80)	NN
ort	varchar(80)	N
isbnkz	varchar(10)	N, ISBN-Kennzahl des Verlags
adresse	varchar(80)	N
tel	varchar(20)	N
fax	varchar(20)	N
url	varchar(200)	N, URL für Homepage
bemerkung	varchar(100)	N

Bemerkung:

Ein Verlag muß für jeden neuen Erscheinungsort neu aufgenommen werden, sowohl die Adreßangaben als auch die URL und die ISBN-Kennzahl können sich für den gleichen Verlag in Abhängigkeit vom Erscheinungsort ändern. Das Attribut `isbnkz` kann dem Bibliothekar bei der Erfassung von Titeln helfen, indem anhand der eingegebenen ISBN bereits die in Frage kommenden Verlage vorgeschlagen werden.

institution

Attribute	Typ	Kommentar
instid	integer	NN, Primärschlüssel
name	varchar(100)	NN
ort	varchar(80)	NN
adresse	varchar(80)	N
tel	varchar(20)	N
fax	varchar(20)	N
url	varchar(200)	N, URL für Homepage
bemerkung	varchar(100)	N

Bemerkung:

Als Institutionen werden Universitäten, Hochschulen und Forschungseinrichtungen verstanden, welche Reports und Dissertationen veröffentlichen, Zeitschriften im Selbstverlag herausbringen oder Tagungen ausrichten.

exemplar

Attribute	Typ	kommentar
exemplarid	integer	NN, Primärschlüssel
origid	integer	NN, Fremdschlüssel (Trigger), Identifikator der betreffenden Literaturstelle
typ	char	NN, Kennzeichnung des Typs der Literaturstelle, t=Tagungsb., b=Buchtitel, a=Artikel, z=Zeitschriftenausg., s=Zeitschriftenband, n=Nonprintm., d=Diss., r=Report
signatur	varchar(30)	N, Standortsignatur der Bibliothek
zugangsnr	varchar(30)	N, Zugangsnummer (UB)
zustand	char	NN, b=beschädigt, n=nicht vorhanden, u=unbekannt, s=bestellt, v=vorhanden, a=ausgesondert, r=in reparatur
ausleihstatus	char	NN, a=ausleihbar, s=semesterapparat, p=präsenzbestand, e=entliehen, f=fernleihe, n=nicht verleihbar
erfdatum	datetime	NN, Erfassungsdatum
aendatum	datetime	N, Datum der letzten Änderung
erwerbungsdaten	varchar(30)	N
besitzerid	integer	NN, Fremdschlüssel
bemerkung	varchar(100)	N

Bemerkungen:

Entsprechend dem `typ` des Exemplars muß durch einen Trigger geprüft werden, ob der mit `origid` angegebene Fremdschlüssel in der zugehörigen Tabelle als Primärschlüssel existiert, um die referentielle Integrität zu sichern. Ebenso muß beim Löschen von Einträgen in den Relationen der Ausleihobjekte geprüft werden, ob abhängige Einträge existieren. Als `erwerbungsdaten` könnte erfaßt werden, aus welchen Mitteln das betreffende Exemplar beschafft wurde.

Als Default-Wert für `zustand` könnte `v` vereinbart werden, da die anderen Werte eher Ausnahmesituationen repräsentieren. Der `ausleihstatus` kann nur dann die Werte `a, s, p` oder `e` annehmen, wenn das Exemplar vorhanden, also `zustand = v` ist, dies muß ebenfalls durch einen Trigger geprüft werden.

leser

Attribute	Typ	Kommentar
leserid	integer	entspr. Lesekartennr.,NN, Primärschlüssel
name	varchar(30)	NN
zusatz	varchar(10)	N, Namenszusatz wie von, van
vornamen	varchar(30)	NN
adr1	varchar(150)	NN
adr2	varchar(150)	N
tel	varchar(15)	N
fach1	varchar(20)	N, Studienfach bei Studenten
fach2	varchar(20)	N
fach3	varchar(20)	N
status	char	NN, s=Student, a=Angest., g=Gastdozent, p=Prof. etc.
bemerkung	varchar(100)	N

Bemerkungen:

Als erste Adresse (`adr1`) gilt die Anschrift am Studienort, eventuell kann auch die Heimatanschrift von Studenten (`adr2`) erfaßt werden. Bei Bedarf kann der Wertebereich des Attributs `status` erweitert werden, die angegebenen Werte sind nur ein Vorschlag. Auch die Aufnahme der email-Adresse als Attribut ist zu erwägen, Mahnungen und Informationen könnten auf diesem Wege wesentlich schneller und kostengünstiger übermittelt werden. Jedoch ist die Rechtsgültigkeit von per email versendeten Mahnungen noch ungeklärt.

besitzer

Attribute	Typ	Kommentar
besitzerid	integer	NN, Primärschlüssel
name	varchar(100)	NN, Nachname des Besitzers bzw. Bezeichnung der Bibliothek
vornamen	varchar(30)	N
sitz	varchar(30)	N
adr	varchar(150)	N
tel	varchar(15)	N
bemerkung	varchar(100)	N

Relationen für die n:m Beziehungen:

ausleihe

Attribute	Typ	Kommentar
leserid	integer	NN, Fremdschlüssel, Teil des Primärschlüssels
exemplarid	integer	NN, Fremdschlüssel, Teil des Primärschlüssels
ausleihdatum	datetime	NN, Datum der Vorbestellung oder des Beginns der Entleihfrist, Teil des Primärschlüssels
typ	char	NN, Ausleihart: v=Vorbestellung, a=Ausleihe
abgabedatum	datetime	N, Ende der Entleihfrist, bei Vorbestellung NULL
anz_verlaeng	smallint	N, Anzahl der Verlängerungen
bemerkung	varchar(100)	N

Bemerkungen:

Der Wertebereich für `typ` ist nur ein Vorschlag, je nach Bibliothek kann es verschiedene Arten von Ausleihvorgängen mit unterschiedlichen Leihfristen geben. Das `abgabedatum` könnte in Abhängigkeit vom `typ` automatisch berechnet und eingefügt werden. Bei einer Vorbestellung ist der Wert von `abgabedatum` NULL, wird ein vorbestelltes Buch ausgeliehen, so muß der `typ` geändert sowie das `abgabedatum` eingetragen werden. Die Anzahl der pro Leser ausleihbaren Bücher kann, eventuell auch abhängig vom `status` des Lesers, eingeschränkt sein, ebenso die maximale Anzahl der Verlängerungen.

Die Verwendung eines zusammengesetzten Primärschlüssels aus `leserid, exemplarid` und `ausleihdatum` ermöglicht es, bereits abgeschlossene Ausleihvorgänge weiter als Nachweis in der Tabelle `ausleihe` zu führen, auch die Verwaltung der Vorbestellung eines Exemplars durch einen Leser, der dieses Exemplar zu diesem Zeitpunkt bereits entliehen hat, ist damit möglich.

Jedoch bleibt zu bedenken, daß bei einer großen Zahl von Ausleihvorgängen das Volumen der in dieser Relation gespeicherten Datensätze stark anwachsen und nach einer gewissen Zeit auch die Performanz beeinträchtigen würde. Da zudem die Daten vergangener Ausleihen nur recht selten benötigt werden, ist die Einführung einer Relation `ausleih_history` in Betracht zu ziehen. Dies ist wohlgemerkt nur dann notwendig, wenn die Ausleihvorgänge über die Dauer der Ausleihe hinaus aufbewahrt werden sollen. Die Relation `ausleih_history` würde die gleichen Attribute wie für `ausleihe` oben beschrieben aufweisen, sowie zusätzlich ein Attribut für das tatsächliche Abgabedatum besitzen. Ausleihvorgänge werden erst bei Rückgabe der entliehenen Literatur in die `ausleih_history` übernommen, Vorbestellungen werden nicht in diese Relation mit übernommen. Für eine große Bibliothek mit einer hohen Zahl von Ausleihvorgängen wäre auch das Anlegen von nach Jahren getrennten Tabellen für die `ausleih_history` denkbar.

Es besteht weiterhin die Frage, ob Fernleihen ebenfalls über dieses System verwaltet werden sollen, oder ob die betroffenen Exemplare lediglich als fernverliehen gekennzeichnet werden und die Verwaltung der Leihvorgänge an anderer Stelle erfolgt.

Bemerkung zu den m:n Beziehungen zwischen Autoren und Ausleihobjekten:

Die n:m Beziehungen zwischen den Relationen der Ausleihobjekte und der Relation `autoren` wurden jeweils um die Attribute `rolle` und `rang` erweitert, um die Art und Weise der Beteiligung der jeweiligen Personen an der erfaßten Literaturstelle festzuhalten und so die korrekte Titelausgabe zu ermöglichen. Die im folgenden für `rolle` angegebenen Wertebereiche sind nur Anregungen, die Einbeziehung weiterer Beteiligungsarten ist je nach Bedarf möglich.

tagung_aut

Attribute	Typ	Kommentar
tagungsid	integer	NN, Fremdschlüssel
autorid	integer	NN, Fremdschlüssel
rolle	char	NN, h=Herausgeber etc.
rang	smallint	N, 1..9

buch_aut

Attribute	Typ	Kommentar
buchid	integer	NN, Fremdschlüssel
autorid	integer	NN, Fremdschlüssel
rolle	char	NN, v=Verfasser, h=Herausgeber, u=Übersetzer, m=Mitarbeiter etc.
rang	smallint	N, 1..9

art_aut

Attribute	Typ	Kommentar
artikelid	integer	NN, Fremdschlüssel
autorid	integer	NN, Fremdschlüssel
rolle	char	NN, v=Verfasser, h=Herausgeber, u=Übersetzer
rang	smallint	N, 1..9

nonprint_aut

Attribute	Typ	Kommentar
nonprintid	integer	NN, Fremdschlüssel
autorid	integer	NN, Fremdschlüssel
rolle	char	NN, v=Verfasser, h=Herausgeber etc.
rang	smallint	N, 1..9

diss_aut

Attribute	Typ	Kommentar
dissid	integer	NN, Fremdschlüssel
autorid	integer	NN, Fremdschlüssel
rolle	char	NN, v=Verfasser, b=Betreuer
rang	smallint	N, 1..9

report_aut

Attribute	Typ	Kommentar
reportid	integer	NN, Fremdschlüssel
autorid	integer	NN, Fremdschlüssel
rang	smallint	N, 1..9

tagung_sw

Attribute	Typ	Kommentar
tagungsid	integer	NN, Fremdschlüssel
swid	integer	NN, Fremdschlüssel

buch_sw

Attribute	Typ	Kommentar
buchid	integer	NN, Fremdschlüssel
swid	integer	NN, Fremdschlüssel

art_sw

Attribute	Typ	Kommentar
artikelid	integer	NN, Fremdschlüssel
swid	integer	NN, Fremdschlüssel

zeitschrift_sw

Attribute	Typ	Kommentar
zeitschrid	integer	NN, Fremdschlüssel
swid	integer	NN, Fremdschlüssel

nonprint_sw

Attribute	Typ	Kommentar
nonprintid	integer	NN, Fremdschlüssel
swid	integer	NN, Fremdschlüssel

diss_sw

Attribute	Typ	Kommentar
dissid	integer	NN, Fremdschlüssel
swid	integer	NN, Fremdschlüssel

report_sw

Attribute	Typ	Kommentar
reportid	integer	NN, Fremdschlüssel
swid	integer	NN, Fremdschlüssel

5.5. Implementierte Fassung

Bisher implementiert und getestet wurde die Literaturverwaltung für Buchtitel, Reports, Dissertationen und Artikel mit Stammsätzen für Autoren, Verlage, Institutionen und Schlagwörter. Die Daten für Buchtitel und Dissertationen sowie für einige Reports wurden aus der *allegro*-Datenbank der Zweigstelle Informatik/Rechenzentrum der UB Leipzig übernommen. Angaben zu den verwendeten Ladeprogrammen finden sich im Anhang 3. Weitere Reports sowie Artikel wurden zu Testzwecken selbst erfaßt. Die Aufteilung der Bücher in Buchtitel und Tagungsbände aus den vorhandenen Daten war nicht automatisierbar. Auch Zeitschriften wurden bisher nicht erfaßt.

Die Erfassung einzelner Exemplare war anhand der von *allegro* bereitgestellten Daten nicht realisierbar, da in der für die Bestandsangabe vorgesehenen Kategorie keineswegs nur die Anzahl der zu dem jeweiligen Titeleintrag gehörenden Exemplare stand. Vielmehr wurde dort auch teilweise die Signatur älterer Exemplare angegeben, deren Titelangaben noch nicht im elektronischen Katalog erfaßt wurde. Daher wurden die eigentlich für die Relation Exemplar vorgesehenen Attribute `signatur` und `zugangsnr` zu den Tabellen für die Ausleihobjekte hinzugenommen, weiterhin wurden Bestandsangaben aus dem Dump übernommen (siehe Anhang 1 und 2). Um eine computerisierte Ausleihe zu realisieren, müssen zuvor die Exemplarbestände der Bibliothek genauer als bisher erfaßt werden sowie die Leserdaten eingegeben werden. Dazu ist die Erstellung eines Eingabe-Tools notwendig, welches den Bibliothekar bei der Erfassung dieser Daten und bei der Titelerfassung unterstützt.

Um eine Nutzung der bisherigen Implementation zu ermöglichen, wurde eine Benutzeroberfläche für die Literaturrecherche erstellt. Nähere Informationen zu dieser Online-Recherche finden sich im Kapitel 7.3.

Weiterführende Informationen:

Im Anhang 1 findet sich eine Übersicht über die derzeit in der Datenbank befindlichen Tabellen und die Abweichungen zu dem in Kapitel 5.4. beschriebenen Entwurf. Anhang 2 enthält eine Auflistung der vom *allegro*-System der Zweigstelle Informatik/Rechenzentrum verwendeten Kategorien und ihrer Entsprechungen in der Datenbank. Eine Übersicht zu den verwendeten Ladeprogrammen, SQL-Skripten und Triggern ist im Anhang 3 zu finden.

6. Beurteilung des WWW als Frontend für DB-Anwendungen

6.1. Allgemeine Bemerkungen zum World Wide Web

Das World Wide Web (auch WWW, W3 oder Web genannt) begann 1989 als Projekt zur Arbeit mit verteilten Informationen am European Laboratory for Particle Physics (CERN) und entwikkelte sich mit atemberaubender Geschwindigkeit zu einem der modernsten weltweiten Informationssysteme [SWH95]. Das WWW ermöglicht die Bereitstellung von **Hypertext-Dokumenten**, die dadurch gekennzeichnet sind, daß sie Verweise, sogenannte Hyperlinks, auf andere Dokumente besitzen. Web-Dokumente können neben normalem Text und Hypertext auch Grafiken und Bilder enthalten, auch die Bereitstellung von anderen Medien wie Video-Sequenzen und Audio-Tracks wird unterstützt.

Web-Dokumente werden von Web-Servern bereitgestellt und mit sogenannten Browsern (auch Viewer, WWW-Clients) betrachtet. Web-Server und Web-Clients kommunizieren miteinander unter Verwendung des Protokolls **HTTP** (Hypertext Transfer Protocol), welches seit 1990 in Gebrauch ist. Die Adressierung von Dokumenten im WWW erfolgt über sogenannte **URLs** (Uniform Resource Locator), welche aus drei Teilen bestehen: der erste Teil spezifiziert das Übertragungsprotokoll (meist HTTP), der zweite gibt den Host an, der als Server das Dokument bereitstellt, der dritte Teil der URL bezeichnet den Pfadnamen des Files, welches das Dokument enthält [Tan96]. Die meisten Browser unterstützen auch die älteren Übertragungsprotokolle wie z.B. FTP (File Transfer Protocol) und gopher, bei welchen im Gegensatz zu HTTP die Verbindung zwischen Client und Server während der gesamten Sitzung (session) besteht. Das Protokoll HTTP soll im folgenden kurz erläutert werden [nach Tan96].

Beim Aufruf eines Web-Dokuments über dessen URL ermittelt der Browser über DNS (Domain Name Service) zunächst die IP-Adresse (IP = Internet Protocol) des angegebenen Servers und stellt dann automatisch eine TCP-Verbindung (TCP = Transmission Control Protocol) zu dem Web-Server her, auf dem das gesuchte Dokument liegt. Der Client sendet dann einen Request, worauf der Server mit der Übertragung der angeforderten Seite oder aber mit einer Fehlermeldung antwortet. Nach der Übertragung des Dokuments wird die Verbindung zum Web-Server beendet. Dies bedeutet, daß pro Verbindung nur ein File übertragen wird, so daß also für jedes in ein Web-Dokument integriertes Bild (“in-line image”) eine neue Verbindung zum Server hergestellt werden muß. Der WWW-Client entscheidet anhand der Information zum Inhalt des Dokuments (z.B. “`Content-Type: text/html`”) über die Art der Darstellung der Informationen.

Zur Beschreibung von Hypertext-Dokumenten dient **HTML** (Hypertext Markup Language), welches eine Anwendung des ISO-Standards für SGML (Standard Generalized Markup Language) ist. HTML ist eine Sprache zur Beschreibung von Dokumenten; sie besteht aus Anweisungen zur Formatierung, welche in das Dokument eingebettet werden. Diese Einbettung der Formatierungsanweisungen sowie die Standardisierung von HTML gestatten den Browsern das Darstellen und Reformatieren von Web-Dokumenten unabhängig von der Hardware-Plattform und dem Betriebssystem. Weiterführende Informationen zur Arbeit mit HTML finden sich in [SWH95], eine gute Online-Dokumentation bietet [Han96].

6.2. Vorteile

Die Hauptvorteile der Verwendung des World Wide Web als Frontend für Datenbankanwendungen liegen meines Erachtens in der dadurch problemlos zu realisierenden weiten **Verfügbarkeit** der Anwendungen, die damit jederzeit weltweit zugänglich sind. Die als Oberfläche verwendeten Browser sind für alle Plattformen verfügbar, zumeist kostengünstig erhältlich und stellen eine einheitliche, komfortable graphische Benutzerschnittstelle dar.

Das WWW bietet sich besonders als Bedienoberfläche für universitäre Anwendungen wie z.B. die Literaturverwaltung an, da fast alle Arbeitsplätze über einen Internet-Zugang verfügen, Browser meist ohnehin bereits installiert sind und ihre Benutzung den potentiellen Nutzern einer Literaturverwaltung schon vertraut ist. Damit entfällt auch die bei der Verwendung anderer Front-Ends notwendige Distribution der fertigen Anwendung an den Nutzer. Die Anwendung wird zentral entwickelt und gepflegt, dem Nutzer steht immer die aktuellste Version zur Verfügung.

Aufgrund der **Plattformunabhängigkeit** erfolgt die Anwendungsentwicklung nur einmal, die Wartung der Anwendung wird dadurch erheblich erleichtert, Erweiterungen und Änderungen sind jederzeit möglich, ohne daß dem Endnutzer dadurch ein Aufwand entsteht. Der Zugriff auf die Web-Anwendungen ist damit von jedem an das Internet angeschlossenen Arbeitsplatzrechner möglich, unabhängig vom Typ des Rechners (z.B. Workstation oder PC) und vom verwendeten Betriebssystem.

Eine funktionelle Erweiterung der Literaturverwaltung durch die Integrierung von Verweisen auf die Homepages von Verlagen und Autoren sowie auf Online-Dokumente im World Wide Web ist überhaupt nur durch die Verwendung des WWW als Benutzerschnittstelle möglich.

Auch die Aufnahme von Literaturstellen, die nur online und nicht in gedruckter Form vorliegen, ist nur bei einer Anbindung der Literaturverwaltungsdatenbank an das World Wide Web sinnvoll. Dieser Aspekt wird mit der steigenden Zahl elektronischer Publikationen in der Zukunft noch an Bedeutung gewinnen.

6.3. Nachteile

Ein gewisser Nachteil von WWW-Anwendungen besteht in der Tatsache, daß das **Layout** der Benutzeroberfläche nur bedingt beeinflußbar ist, da es abhängig vom verwendeten Browser und den durch den Benutzer einstellbaren Optionen ist. Die Funktionalität wird dadurch jedoch nicht beeinträchtigt, sofern keine browserspezifischen Features (wie z.B. Netscapes Frame-Technologie) verwendet werden. Bei der Weiterentwicklung von HTML wird vermutlich auch die Forderung der Web-Autoren nach besserer Kontrolle über das Layout ihrer Anwendungen einen stärkeren Einfluß haben.

Weiterhin ist zu bedenken, daß das zur Übertragung von WWW-Dokumenten verwendete Protokoll HTTP ein sogenanntes “stateless protocol” ist, d.h. daß die Verbindung zwischen dem HTTP-Server und dem Client nur während der Übertragung eines Dokuments besteht [Grobe]. Die Interaktion mit dem Datenbank-Server kann also nicht direkt durch den WWW-Client erfolgen, sondern muß durch ein zwischen den HTTP-Server und den Datenbank-Server gesetztes Gateway realisiert werden, welches als Datenbank-Client fungiert. Im folgenden Kapitel wird auf diesen Aspekt näher eingegangen.

Auch die Überprüfung der Benutzereingaben ist nicht direkt im WWW-Client realisierbar, sondert erfordert eine zeitaufwendige Server-Interaktion, so daß eine Unterstützung des Nutzers bei Dateneingaben nur mit großem Aufwand realisierbar ist.

Damit ist das WWW besonders gut geeignet für die Realisierung der Benutzeroberfläche eines Bibliothekskataloges. Für bibliothekarische Arbeiten wie die Erfassung und Aktualisierung des Datenbestandes ist jedoch eventuell eine Anwendung, die auf einem leistungsfähigen Datenbank-Frontend wie z.B. Powerbuilder basiert, vorzuziehen.

7. Realisierung einer Online-Recherche

7.1 Grundlagen: Interaktivität durch CGI und HTML-Formulare

Mit der rasanten Entwicklung des World Wide Web in den letzten Jahren entstand der Wunsch nach einer einfachen Möglichkeit zur Erstellung **dynamischer HTML-Dokumente**. Die bereitzustellenden Informationen sollten entsprechend den Angaben des Benutzers erst bei der Anforderung des Dokuments aus einer Datenbank abgefragt werden. Im vergangenen Jahr sind viele Lösungen zur Anbindung von Datenbanken an das WWW entwickelt worden, ein Großteil von ihnen basiert auf dem CGI-Standard [Rowe]. Dieses **Common Gateway Interface** (CGI) ist ein Standard für die Kommunikation zwischen WWW-Servern und Anwendungsprogrammen und zur Zeit wohl die verbreitetste Methode zur Erstellung dynamischer Web-Dokumente. CGI ermöglicht die Ausführung von Skripten und Programmen, welche Nutzereingaben verarbeiten können und dadurch Interaktivität gestatten. Die Programme können in einer beliebigen Skript- oder Programmiersprache geschrieben sein; C/C++, Perl, Visual Basic, Applescript und UNIX Shells sind die gebräuchlichsten [NCSA95]. Bei der Aktivierung eines CGI-Programmes erzeugt der Webserver einen Prozeß für das Anwendungsprogramm. Dieses wird ausgeführt, die Ergebnisse werden über den Webserver dem Browser des Endnutzers übergeben und anschließend wird der Prozeß terminiert.

Um Nutzereingaben zu ermöglichen, wurde HTML um einige Elemente erweitert, welche die Darstellung von **Eingabeformularen** durch den Browser veranlassen und die eingegebenen Werte an einen im Formular spezifizierten HTTP-Server senden. HTTP-Server wurden so modifiziert, daß sie nun das angegebene CGI-Programm aufrufen und ihm die Eingabedaten verfügbar machen [Grobe],[SWH95].

Zur Übergabe der Parameter an den Server existieren zwei verschiedene Methoden: POST und GET [SWH95]. Die Definition eines Formulars erfolgt in folgender Weise:

```
<FORM METHOD=”POST” ACTION=”cgi-bin/myscript”>
```

Mit `METHOD` wird die zu verwendende Übergabemethode angegeben, `ACTION` bezeichnet das beim Absenden des Formulars durch den Server aufzurufende Programm, welches sich im Verzeichnis `cgi-bin` befindet. Das aufgerufene CGI-Programm wertet nun zuerst die Umgebungsvariable `REQUEST_METHOD` aus, um die Übergabemethode festzustellen. Auf die beiden Methoden POST und GET wird im folgenden näher eingegangen.

Die Methode **GET** hängt die zu übertragenden Daten an die URL in folgender Form an:

```
http://www.host.domain/cgi-bin/myscript?var1=wert1&var2=wert2
```

Das aufgerufene Programm findet in der Umgebungsvariable `QUERY_STRING` die Daten aus dem Formular. Diese Methode gestattet die Erstellung von Links, die ohne Involvierung des Nutzers bei ihrer Aktivierung die erforderlichen Daten an das aufgerufene Programm übergeben. Auch ein explizites Abschicken einer Anfrage durch `SUBMIT` ist damit nicht zwingend notwendig. Als Nachteil erweist sich jedoch die begrenzte Länge von URLs (meist 1024 Zeichen), die die Übertragung größerer Datenmengen auf diesem Weg verhindert, da die diese Länge überschreitenden Daten einfach abgeschnitten werden. Sonderzeichen müssen kodiert werden, um Interoperabilität zu gewährleisten [Grobe],[SWH95].

Im Gegensatz dazu erfolgt die Datenübertragung bei der Methode **POST** unabhängig von der URL. Die Umgebungsvariable `CONTENT_LENGTH` enthält die Anzahl der Zeichen, die dem Programm übergeben wurden und nun am *stdin* des Programms in der Form

```
var1=wert1&var2=wert2
```

ohne EOF vorliegen. Diese Methode erlaubt die Übertragung umfangreicher Eingabedaten, auch müssen Sonderzeichen und Leerzeichen nicht kodiert werden. Jedoch ist ein explizites Absenden der Anfrage durch `SUBMIT` notwendig. Ein Nachteil ist auch die fehlende Möglichkeit, auf eine Ergebnisseite einer mit POST gesendeten Anfrage ein Bookmark zu setzen [Grobe],[SWH95].

7.2. Die Skript-Sprache Perl

Perl ist eine Skript-Sprache, die Elemente aus C, awk, sed, grep und der Bourne-Shell kombiniert und damit die Nische zwischen Shell-Skripten und C-Programmen füllt. Perl-Programme (bzw. Perl-Skripte) werden nicht kompiliert, sondern interpretiert. Jedoch ist Perl schneller als die meisten anderen interpretierten Sprachen [Rya93].

“Perl” ist ein Akronym für ***P**ractical **E**xtraction and **R**eport **L**anguage*. Die Sprache wurde von Larry Wall entwickelt und wird auch von ihm gepflegt. Perl ist frei erhältliche Software und arbeitet mit den wichtigsten Plattformen und Betriebssystemen zusammen, damit sind Perl-Skripte auf einer Vielzahl von Systemen lauffähig.

Die Programmflußanweisungen von Perl sind an C angelehnt, Anweisungen enden mit einem Semikolon und Blocks sind in geschweifte Klammern eingeschlossen. Perl besitzt drei **Datentypen**: Skalare, Felder von Skalaren und assoziative Felder (associative arrays) von Skalaren. Variablen brauchen nicht vor der Benutzung definiert werden, Felder können dynamisch erweitert werden.

Perls **Operatoren** sind ebenfalls an C angelehnt, es gibt jedoch eine ganze Reihe darüberhinausgehende String-Operatoren. Die von Perl interpretierten regulären Ausdrücke ähneln den vom vi-Editor verwendeten. Die umfangreiche Möglichkeiten zum Pattern Matching sind größtenteils den UNIX-Tools sed und grep entlehnt. Aufgrund dieser leistungsfähigen und flexiblen String-Verarbeitungsmöglichkeiten ist Perl für den Einsatz bei der Entwicklung von Web-Anwendungen prädestiniert.

Perl bietet zudem viele eingebaute **Routinen** (print, substr, push, pop u.a.), UNIX-ähnliche Routinen wie chmod, chown, mkdir, rename etc. sowie die Möglichkeit, auf Routinen in C-Bibliotheken zuzugreifen. Außerdem gibt es die Möglichkeit, UNIX-Shell-Kommandos auszuführen und Text-Files zu bearbeiten.

Grundlegende Informationen zu Perl finden sich in [Rya93] und in der Manual-Page für Perl, für weiterführende Informationen sind [Schw95] und [Wal92] zu empfehlen.

7.3. Funktionsweise von Sybase web.sql

Um WWW-Clients den Zugriff auf Datenbanken zu ermöglichen, könnte für jede Anwendung ein CGI-Skript geschrieben werden, welches den Zugriff realisiert. Diese Methode ist jedoch ineffektiv und aufwendig, da die Parameterübergabe und die Realisierung der Verbindung zum Datenbank-Server für jede Anwendung erneut programmiert werden müßte. Statt dessen bietet sich der Einsatz eines Web/Datenbank-Gateways an, welches z.B. auf CGI basieren kann. Dieses Gateway-Programm vermittelt zwischen dem Web-Server und dem Datenbank-Server. Es übernimmt die Parameterübermittlung und stellt Funktionen zum Zugriff auf den Datenbankserver bereit, mit denen aus den vom Nutzer über ein Formular eingegebenen Parametern eine SQL-Anfrage erstellt werden kann, die dann an den DB-Server weitergeleitet wird. Nach der Anfragebearbeitung durch den DB-Server werden die von diesem übergebenen Anfrageergebnisse in HTML-Seiten umgewandelt, welche dem Web-Server übermittelt werden. Der WWW-Client erhält nur die HTML-Ausgabe, die Interaktion mit dem DB-Server bleibt vor ihm verborgen. Die Entwicklung von dynamischen Web-Anwendungen mit Datenbankzugriff wird durch derartige Gateway-Programme erheblich vereinfacht.

Das von Sybase entwickelte web.sql ist ein komfortables Gateway-Tool zur Integration von Zu-

griffen auf Daten in Sybase System 10 / System 11 Datenbanken in HTML-Dokumente. Sybase web.sql gibt es als Version für das Netscape Application Program Interface (NSAPI) sowie für das Common Gateway Interface (CGI). Die CGI-Version kooperiert mit allen HTTP-Servern, die den CGI-Standard unterstützen (z.B. Netscape, NCSA, OpenMarket, etc.). Dagegen arbeitet die NSAPI-Version nur mit Netscapes Web-Servern (Commerce Server bzw. Communications Server) zusammen [Syb2]. Auf die daraus resultierenden Performance-Unterschiede wird nach der Erläuterung der Funktionsweise von web.sql näher eingegangen.

Da web.sql auf Sybase Open Client basiert, können damit auch weitere Sybase-Produkte wie SQL Server, Replication Server, Sybase IQ und Enterprise CONNECT genutzt werden. Die letztgenannte Produktfamilie ermöglicht web.sql auch den Zugriff auf nicht in Sybase-Datenbanken gehaltene Daten z.B. der Systeme DB2, Oracle, Informix und Ingres [Syb2].

Der Ansatz von web.sql besteht in der **Erweiterung von HTML** um ein zusätzliches Element, `<SYB>`, mit welchem Datenbankzugriffe direkt in ein HTML-Dokument integriert werden können. Dieses um `<SYB>` erweiterte HTML-Format wird als Hypertext Sybase (HTS) Format bezeichnet, die HTS-Files müssen die Namenserweiterung .hts haben. Das `<SYB>` Element ist ein Container-Element, d.h. es beginnt mit der Anweisung `<SYB>` und endet mit der Anweisung `</SYB>`. Der von diesen Anweisungen umschlossene Teil des HTS-Files wird durch web.sql verarbeitet, die Ergebnisse werden dann dem Web-Server im HTML-Format übergeben. Das heißt, der Endnutzer sieht den mit `<SYB>` markierten Teil des Dokuments nicht, da dieser durch die Ergebnisse ersetzt wird.

Das `<SYB>` Element gibt es in den Formen `<SYB TYPE=SQL>` für Transact SQL Statements und `<SYB TYPE=PERL>` für die Einbindung von Perl-Skripten. Durch die Integration von Perl wird eine weitere Bearbeitung und Formatierung der Anfrageergebnisse möglich, wogegen die Ergebnisse von einfachen Transact SQL Statements nur in Tabellenform ausgegeben werden. Mit dem HTS-File-Format ist nur HTML-Output generierbar. Sybase web.sql kann jedoch auch Perl-Files verarbeiten, womit sich auch andere Web-Dokumente (Sound, Images, Video) erstellen lassen.

Die **Funktionsweise** der CGI-Version von web.sql ist aus der Abbildung 7.1 ersichtlich:

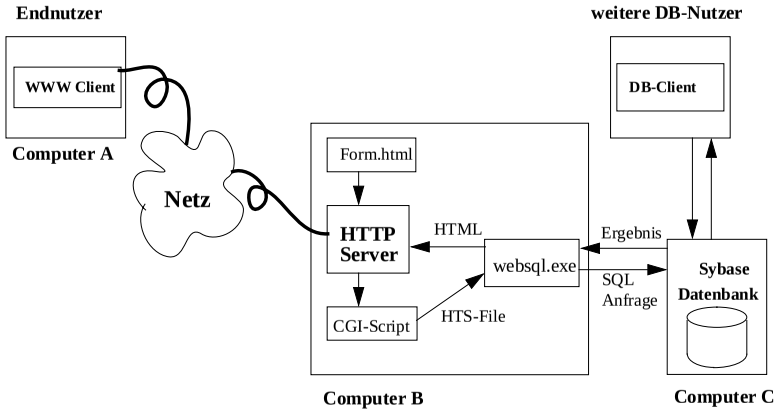

Abbildung 7.1 Funktionsweise des web.sql

Web.sql läuft auf demselben Rechner wie der HTTP-Server. Die URLs für HTS-Files haben die Form:

```
http://www.host.domain/cgi-bin/websql/websql.dir/myfile.hts
```

Beim Aufruf eines solchen Files durch die Aktivierung eines Links oder das Abschicken eines HTML-Formulars wird zunächst das im Verzeichnis `cgi-bin` befindliche CGI-Skript `websql` aufgerufen. Dieses Skript exportiert Umgebungsvariablen und ruft das Programm `websql.exe` auf und übergibt das in der URL angegebene HTS-File an dieses Programm. Durch `websql.exe` wird automatisch die Verbindung zum Sybase-Server erstellt, das übergebene HTS- oder Perl-File wird abgearbeitet und der HTML-Output wird an den Web-Server übertragen, der die Ergebnisseite dann dem Browser übermittelt. Dabei wird der gesamte nicht durch `<SYB>` eingeschlossene HTML-Inhalt dem Web-Server unverändert übergeben. Die dem Programm eventuell über die weiter oben beschriebenen Methoden POST oder GET übermittelten Parameter werden von `websql.exe` als Perl-Variablen bereitgestellt, welche sowohl in Transact SQL Statements als auch in Perl-Skripten verwendet werden können. In einem HTS-Dokument sind beliebig viele `<SYB>` Blöcke erlaubt.

Bei der CGI-Version von web.sql muß das Programm `websql.exe` bei jedem Aufruf eines HTS-Files erneut gestartet werden, was einen erheblichen Overhead verursacht. Dagegen ist die NSAPI-Version von web.sql direkt in den HTTP-Server integriert, zudem kann diese Version

auch eine Anzahl von Verbindungen zum Datenbank-Server im Cache halten, wodurch der durch den Verbindungsaufbau bedingte Overhead reduziert wird. Die NSAPI-Version bietet daher im Vergleich mit der CGI-Version die bessere Performanz. Da am Institut für Informatik der Universität Leipzig jedoch ein HTTP-Server von NCSA verwendet wird, kam für die Implementierung der Online-Recherche die CGI-Version von web.sql zum Einsatz.

Die mit web.sql mitgelieferte Perl-Bibliothek stellt dem Anwendungsprogrammierer zwei verschiedene **Schnittstellen** bereit: zum einen das web.sql Convenience API, welches einige einfache Routinen für die Datenbank-Interaktion bereitstellt und die Anfrageergebnisse in HTML-Tabellen bereitstellt, und zum anderen das komplexere web.sql Client-Library API, das an das Sybase Open Client-Library API angelehnt ist und weiterreichende Möglichkeiten zur zeilenweisen Manipulation der Anfrageergebnisse bietet. Das web.sql Client-Library API unterscheidet sich vom Sybase Open Client-Library API, um die Unterschiede der Programmierung in C und Perl zu reflektieren und die Programmierung mit web.sql zu erleichtern. Einige Routinen wurden zusätzlich in das web.sql Client-Library API aufgenommen, andere wurden nicht übernommen. Insbesondere wird die Verwaltung der Verbindungen (connections) zum DB-Server durch die web.sql Client-Library API automatisch erledigt. Die von den beiden Schnittstellen bereitgestellten Routinen können beliebig kombiniert werden. Weitere Informationen zu den API und zur Arbeit mit web.sql finden sich in [Syb3].

7.4. Realisierung der Online-Recherche mit web.sql

Als Grundlage der Online-Recherche diente die in Kapitel 5.3. beschriebene Implementierung der Literaturverwaltung mit den Daten der Zweigstelle Informatik/Rechenzentrum der Universitätsbibliothek Leipzig. Die verwendeten Relationen und Attribute werden im Anhang 2 näher erläutert.

Im Vordergrund stand die Bereitstellung einer leicht verständlichen, intuitiv bedienbaren Oberfläche mit flexiblen Möglichkeiten zur Suche im Datenbestand und zur Sortierung der Suchergebnisse. Bei der Entwicklung wurde auf browserspezifische HTML-Elemente weitgehend verzichtet, so daß die Online-Recherche mit allen gängigen Browsern zu benutzen sein dürfte. Erfolgreich getestet wurde die Online-Recherche mit den Browsern Netscape, Mosaic, Internet Explorer und Lynx auf verschiedenen Plattformen.

Auf der in Abbildung 7.2 dargestellten Titelseite der Online-Recherche befindet sich ein Formular mit Eingabefeldern für Nachnamen von Autoren, Titel, Verlag, Signatur, Schlagwort, CR-Klassifikation und Erscheinungsjahr, die Sortierung der Anfrageergebnisse ist nach Autor, Titel, Erscheinungsjahr und Signatur möglich.

Mit dem Abschicken des Formulars wird das HTS-File `search.hts` aufgerufen, die Parameter werden mit der Methode POST übergeben. Das File `search.hts` enthält die HTML-Formatierung für die Ergebnisseite sowie ein in web.sqls `<SYB>`-Elemente eingeschlossenes Perl-Skript, welches entsprechend den übergebenen Suchkriterien die SQL-Statements zusammensetzt, die Anfragen abschickt und die Ergebnisse formatiert. Die Suchkriterien werden durch logisches UND verknüpft. Bei der Zusammensetzung der SQL-Statements ist zu beachten, daß nur jene Tabellen in die FROM-Klausel aufgenommen werden, für die auch Suchvorgaben gemacht wurden, da sonst die Titel, für die eine entsprechende Angabe nicht vorliegt, nicht gefunden werden. Beispielsweise sind nicht alle Titel mit Schlagwörtern versehen, so daß ein Join mit der Tabelle `buch_sw` die Zahl der gefundenen Titel stark limitiert.

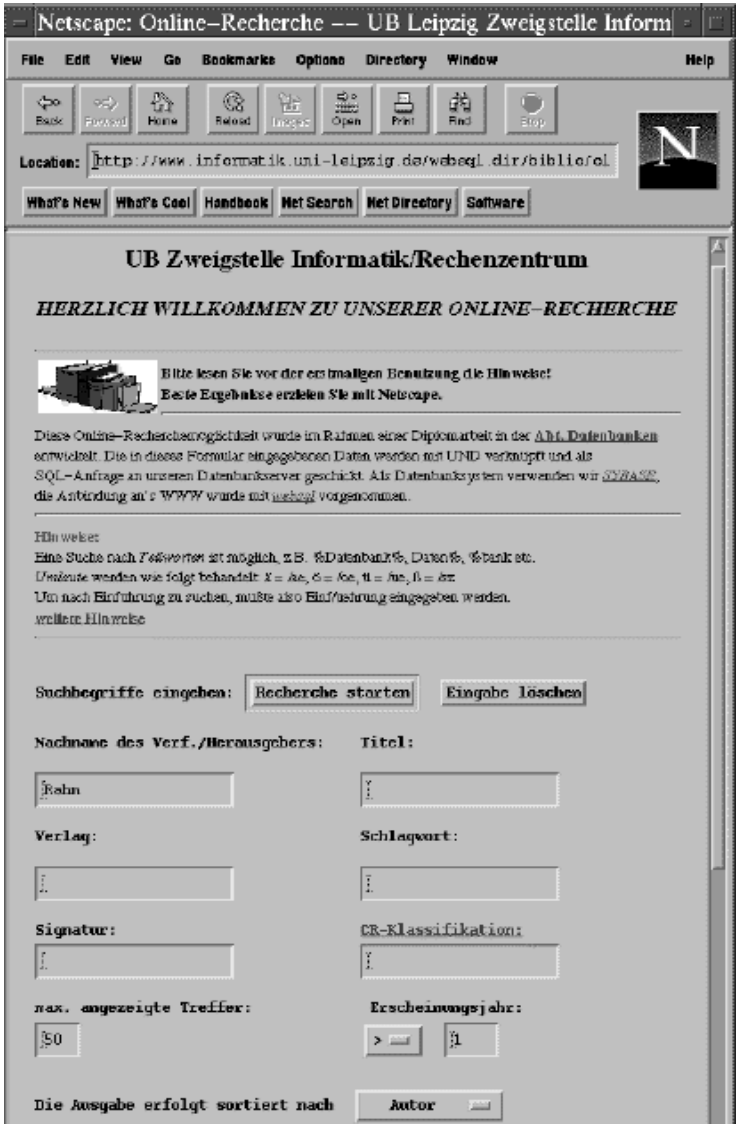

Abbildung 7.2 Titelseite der Online-Recherche

Zusätzlich zum weiter oben beschriebenen Formular wurde auch die Möglichkeit geschaffen, aus der Anzeige einer alphabetischen Liste aller verwendeten Schlagworte über dynamische Links direkt zur Anzeige der zugehörigen Titel zu gelangen. Die Liste der Schlagwörter wird erst beim Aufruf der Seite `sw.hts` erstellt und ist somit immer aktuell. Bei der Aktivierung eines Links aus der Liste wird das HTS-File `searchsw.hts` aufgerufen, welchem die `swid` des gewählten Schlagworts mit der Methode GET übergeben wird.

Um die Ergebnisse einer SQL-Abfrage in übersichtlicher Form anzeigen zu können, sind mehrere Schritte notwendig. Mit einer ersten Suchanfrage werden die Identifikatoren der Titeldatensätze gefunden und entsprechend dem Sortierkriterium geordnet. Duplikate müssen dabei aus der Ergebnismenge entfernt werden, da ein Titel für jede damit in der m:n Beziehung verbundene Person erneut gefunden wird und somit mehrfach in der Ergebnisliste erscheint. In einer Schleife werden dann zu jedem Titel die Personennamen aus der Datenbank geholt und nach der im Attribut `rang` vermerkten Reihenfolge geordnet, wobei Herausgeber durch nachgestelltes (Hrsg.) gekennzeichnet werden, danach werden die restlichen Titeldaten abgefragt.

Die HTML-Formatierung der Ergebnisse erfolgt mit Perl, dabei werden alle deutschen Umlaute durch HTMLs “ISO Latin 1 Character Entities” [Han96] ersetzt. Bei mehrbändigen Ausgaben wird ein dynamischer Link auf die entsprechenden Bände bzw. auf den Haupteintrag eingefügt. Die URLs für die Homepages der Autoren und Verlage und die Verweise auf Online-Dokumente werden in Hyperlinks umgewandelt, so daß der Nutzer sich die Volltextversionen von Reports, Artikeln u.ä. direkt anzeigen lassen oder auf weitere Informationen zugreifen kann. Die Aufbereitung der Anfrageergebnisse geht damit weit über die von den meisten OPACs gebotene Anzeige von passivem Text hinaus. Abbildung 7.3 zeigt den Kopf einer solchen Ergebnisseite.

Die verwendeten HTML- und HTS-Files werden im Anhang 3 aufgeführt.

Abbildung 7.3 Ergebnisseite der Online-Recherche

8. Zusammenfassung und Ausblick

Das Bibliothekswesen ist ein sehr komplexes und kompliziertes Gebiet, eine vollständige Modellierung und Implementierung aller im Bibliothekswesen anzutreffenden Vorgänge sprengt mit Sicherheit den Rahmen einer Diplomarbeit. Der vorliegende Entwurf kann jedoch als Grundlage für weitergehende Implementationen dienen. Er orientiert sich bezüglich der modellierten Vorgänge und Daten an den Gegebenheiten der Zweigstelle Informatik/Rechenzentrum der UB Leipzig, geht jedoch mit der Modellierung der Ausleihvorgänge und der Einbeziehung von Artikeln, Zeitschriften und Online-Publikationen weit über die Funktionalität des derzeit verwendeten Systems hinaus.

Aufgrund der von Bibliothek zu Bibliothek stark abweichenden Vorgaben zur Erfassung von Literaturbeständen erwies es sich als schwierig, einen allgemeingültigen Entwurf zu finden, der ohne Änderungen in jeder Bibliothek angewendet werden kann. Für einen Einsatz in anderen Bibliotheken müssen sicherlich die Wertebereiche einiger Attribute verändert beziehungsweise auch zusätzliche Attribute eingeführt werden. Als Beispiel sei hier nur die Klassifikation der Literatur genannt, die in einer Informatikbibliothek nach der CR-Klassifikation erfolgt, jedoch in anderen Bibliotheken nach anderen Schemata durchgeführt wird. Auch die Ladeprogramme müssen den Kategoriensystemen anderer *allegro*-Systeme angepaßt werden.

Der Entwurf läßt sich durchaus noch um weitere Funktionalitäten erweitern. So könnte eine Verwaltung der Bestellvorgänge realisiert werden, indem Relationen für Lieferanten und Bestellungen angelegt werden. Auch die Bibliothekare könnten in einer Relation verwaltet werden, so daß bei jedem Bestell- und Ausleihvorgang sowie bei jeder Änderung vermerkt wird, wer diese Eintragung vorgenommen hat. Die Einbeziehung der Fernleihe ist ebenfalls denkbar.

Eine Erweiterung der Relation Schlagwort um Verweise auf verwandte oder anderssprachige Schlagwörter und die Einführung einer Schlagworthierarchie wäre sehr wünschenswert, ist jedoch mit einem hohen Arbeitsaufwand verbunden und ohne bibliothekarische Unterstützung kaum zu realisieren. Auch die Einbeziehung der Suche nach der dem Schlagwort entsprechenden CR-Klassifikation wäre realisierbar.

Mit dem `LIKE`-Operator bietet Sybase bereits eine Möglichkeit zur Suche in Texten, damit könnte eine Volltextsuche in den Abstracts realisiert werden. Allerdings wird diese Suche nicht durch einen Index unterstützt. Um eine effektive Volltextsuche in den Abstracts zu ermöglichen, könnte das RDBMS um leistungsfähige Text-Retrieval-Funktionen erweitert werden. In [CBBKR94] wird auf diesen Aspekt näher eingegangen. Mit einem leistungsfähigen Text-Re-

trieval-System ergäbe sich auch die Möglichkeit, die Volltextsuche nicht nur auf den Abstracts, sondern auch auf dem gesamten Text von Publikationen anzubieten. Diese Möglichkeit wäre insbesondere für Artikel, Reports und Dissertationen von Bedeutung.

Mit der Online-Recherche steht zunächst eine komfortable Oberfläche für den Bibliotheksbenutzer zur Verfügung, der in den Literaturbeständen der Zweigstelle Informatik/Rechenzentrum recherchieren will. Zur Zeit muß der Inhalt der Literaturdatenbank jedoch regelmäßig mit Hilfe der Ladeprogramme und einem Dump der *allegro*-Daten aktualisiert werden. Um die Vorteile eines RDBMS besonders bezüglich der Integritätssicherung voll ausschöpfen zu können, ist ein Tool für den Bibliothekar notwendig, welches die direkte Einarbeitung neuer Literaturbestände unterstützt.

Eine Verbesserung der Performanz der Online-Recherche könnte durch den Einsatz von stored procedures erreicht werden, da stored procedures nur einmal compiliert werden und die Anfrageoptimierung bereits zum Zeitpunkt der Compilierung erfolgen kann. Diese Lösung bietet sich besonders für häufig verwendete Anfragen, wie die bei der Aktivierung des HTS-Files `search.hts` in einer Schleife auftretenden Abfragen der Autorendaten und der Titeldaten, an.

9. Literaturverzeichnis

BCN92 Batini, Carlo; Ceri, Stefano; Navathe, Shamkant B.: Conceptual database design: an entity-relationship approach. Benjamin/Cummings, Redwood City, Calif. 1992

CBBKR94 Cremers, A. B.; Balownew, O.; Bode, T.; Kalinski, J.; Rottmann, H.: Ein Ablösesystem für den Bibliotheksverbund Nordrhein-Westfalen : Eine Machbarkeitsstudie. Hochschulbibliothekszentrum des Landes NW (HBZ), Köln 1994

CBBKW95 Cremers, A. B.; Balownew, O.; Bode, T.; Kalinski, J.; Wolff, J.: HBZ-Ablösesystem: Auswertung eines Performanztests. Hochschulbibliothekszentrum des Landes NW (HBZ), Köln 1995

Dat95 Date, C. J.: An Intoduction to Database Systems. Addison-Wesley, Reading, Mass. 1995

Eve95 Eversberg, Bernhard: 14 Jahre allegro: Grundfragen - Kernprobleme - Anwenderkreise. Universitätsbibliothek der TU Braunschweig 1995
http://www.biblio.tu-bs.de/allegro/papiere/mb_fr.htm

Eve94 Eversberg, Bernhard: Allegro-C: Systemhandbuch ; Version 13. Universitätsbibliothek der TU Braunschweig 1994

Grobe Grobe, Michael: An Instantaneous Introduction to CGI Scripts and HTML Forms.
http://kuhttp.cc.ukans.edu/info/forms/forms-intro.html

Han96 Hannah, Michael J.: HTML Reference Manual
Sandia National Laboratories 1996
http://www.sandia.gov/sci_compute/html_ref.html

HW93 Hagen, Manfred; Will, Liane: Relationale Datenbanken in der Praxis. Springer, Berlin 1993

Kir92 Kirkwood, John: High Performance Relational Database Design. Horwood, New York 1992

Mei95 Meier, A.: Relationale Datenbanken - Eine Einführung für die Praxis. Springer, Berlin 1995

NCSA95 National Center for Supercomputing Applications (NCSA): The Common Gateway Interface. 1995
http://hoohoo.ncsa.uiuc.edu/cgi/overview.html

Rah96 Rahm, Erhard: Informatiker bauen elektronische Bibliothek. Journal der Universität Leipzig, Heft 4/1996, S. 19-21

Rowe Rowe, Jeff: Existing Web/Database Gateway Products.
http://cscsun1.larc.nasa.gov/~beowulf/db/existing_products.html

Rya93 Ryan, Patrick M: Introduction to Perl. Hughes STX Corporation, 1993
http://www.khoros.unm.edu/staff/neilb/perl/introduction/home.html

SWH95 Savola, Tom ; Westenbroek, Alan ; Heck, Joseph: Using HTML : special edition. Que Corp., Indianapolis, Ind. 1995

Schw95 Schwartz, Randal L.: Einführung in PERL. O'Reilly, Bonn 1995

Syb1 Sybase, Inc.: Sybase Web Site
http://www.sybase.com/

Syb2 Sybase, Inc.: Sybase web.sql Information Menu.
http://www.sybase.com/products/internet/websql/

Syb3 Sybase, Inc.: Sybase web.sql On-Line Assistance Menu.
http://www.sybase.com/products/internet/websql/assistance.html

Tan96 Tanenbaum, Andrew S.: Computer Networks. Prentice Hall, London 1996

TUM96 Techn. Univ. München: MEDOC - Die Elektronische Informatikbibliothek.
http://medoc.informatik.tu-muenchen.de/deutsch/medoc.html

UBB1 UB Braunschweig: allegro-C.
http://www.tu-bs.de/ub-tubs/allegro/index.html

UBB2 UB Braunschweig: allegro -Ouvertüre
http://www.tu-bs.de/ub-tubs/allegro/ouvert.html

Unl95 Unland, Rainer: Objektorientierte Datenbanken : Konzepte und Modelle. Thomson, Bonn 1995

Vos95 Vossen, Gottfried: Datenbank-Theorie. Thomson, Bonn 1995

Wal92 Wall, Larry; Schwartz, Randal L.: Programming perl. O'Reilly, Berlin 1992

Anhang 1 Dokumentation der implementierten Version

Der in Kapitel 5 beschriebene Entwurf für ein Literaturverwaltungssystem wurde für die Zweigstelle Informatik/Rechenzentrum der Universitätsbibliothek Leipzig implementiert. Im folgenden werden die Relationen der Implementation aufgeführt. In den Bemerkungen werden die Abweichungen der Implementation von dem in Kapitel 5 beschriebenen Entwurf (im folgenden mit Originalentwurf bezeichnet) aufgeführt.

Relationen für die Entity-Mengen

buchtitel

Attribute	Typ	Kommentar
buchid	integer	NN, Primärschlüssel, clustered index buchtitel_pk
gesamtid	integer	N, buchid des übergeordneten Titels (Gesamttitels) bei mehrbändigen Werken, (Trigger)
bandnr	smallint	N, Bandnummer bei mehrbändigen Werken
baende	smallint	N, Anzahl der Bände, die zu einem Haupteintrag existieren
hstitel	varchar(255)	N, Hauptsachtitel, nonclustered index buchtitindex
estitel	varchar(100)	N, Einheitssachtitel (Titel in Originalfassung)
isbn	varchar(15)	N
isbn_2	varchar(15)	N
auflage	varchar(70)	N, Angabe aus Dump übernommen
jahr	smallint	N, Erscheinungsjahr
medium	varchar(20)	NN, z:Z. alles als Buch oder Mikrofiche
umfang	varchar(50)	N, Angabe aus Dump übernommen
preis	varchar(20)	N, Preis + Währung aus Dump
bezugsinfo	varchar(30)	N, Informationen zum Bezug des Titels
verf_in_vorlform	varchar(130)	N, Verfasserangabe in Vorlageform

buchtitel

Attribute	Typ	Kommentar
hsv	varchar(50)	N, Hochschulschriftenvermerk
kschaft1	varchar(100)	N, 1. Körperschaft
kschaft2	varchar(100)	N, 2. Körperschaft
reihe1	varchar(150)	N, 1. Reihenangabe aus Dump (85)
reihe2	varchar(150)	N, 2. Reihenangabe aus Dump (85a)
signatur	varchar(30)	N, Signatur der Bibliothek (eigentl. zum Exemplar)
zugangsnr	varchar(30)	N, Zugangsnummer (UB) (eigentl. zum Exemplar)
bestand	varchar(50)	N, Anzahl der Exemplare, teilweise auch 2. Signatur
weitexemp	varchar(30)	N, weitere Exemplare
erfdatum	varchar(17)	N, aus Dump
aendatum	varchar(17)	N, aus Dump
verlagsid	integer	NN, Fremdschlüssel
abstractid	integer	N, Fremdschlüssel (Trigger)
url	varchar(200)	N, URL für Online-Dokument
cr	varchar(80)	N, CR-Klassifikation, nonclustered index buchcrindex
bemerkung	varchar(100)	N

Bemerkungen:

Der Relation `buchtitel` werden im Moment auch Tagungsbände zugeordnet. Für die Tagungsbände ist im Originalentwurf eine eigene Relation vorgesehen. Die Attribute `signatur` und `zugangsnr` werden im Originalentwurf der Relation `exemplar` zugeordnet. Die Attribute `bestand` und `weitexemp` treten dagegen im Originalentwurf nicht auf, da für jedes einzelne Exemplar ein Eintrag in die Relation `exemplar` erfolgt. Da die Reihen in dieser Implementation noch nicht in einer eigenen Relation verwaltet werden, fehlen die Fremdschlüssel für die reihenid und die Bandnummern. Statt dessen werden die Reihenangaben mit Bandnummern aus dem Dump übernommen und liegen als Strings in den Attributen `reihe1` und `reihe2` vor. Die Angaben für das Erfassungs- und Änderungsdatum sind im Originalentwurf ebenfalls der Relation `exemplar` zugeordnet.

dissertation

Attribute	Typ	Kommentar
dissid	integer	NN, Primärschlüssel, clustered index dissertation_pk
titel	varchar(255)	NN
instid	integer	NN, Fremdschlüssel
jahr	smallint	N
isbn	varchar(15)	N
typ	varchar(1)	NN, d = diss, p = projekt, i = diplom, z.Z. nur d und i
umfang	varchar(50)	N
signatur	varchar(30)	N
zugangsnr	varchar(50)	N
bestand	varchar(30)	N
abstractid	integer	N, Fremdschlüssel (Trigger)
url	varchar(200)	N, URL für Online-Dokument
cr	varchar(80)	N, CR-Klassifikation
bemerkung	varchar(100)	N

Bemerkungen:

Die Attribute `signatur` und `zugangsnr` werden im Originalentwurf der Relation `exemplar` zugeordnet. Das Attribut `bestand` tritt im Originalentwurf nicht auf, da für jedes Exemplar ein eigener Eintrag in die Relation `exemplar` erfolgt.

report

Attribute	Typ	Kommentar
reportid	integer	NN, Primärschlüssel, clustered index report_pk
titel	varchar(255)	NN
instid	integer	NN, Fremdschlüssel
jahr	smallint	NN
lfdnr	smallint	N, fortlaufende Nummer
abstractid	integer	N, Fremdschlüssel (Trigger)
url	varchar(200)	N, URL für Online-Dokument
cr	varchar(80)	N, CR-Klassifikation
umfang	varchar(50)	N
reihe	varchar(150)	N, Reihenangabe aus Dump
signatur	varchar(30)	N, eigentlich zum Exemplar
bemerkung	varchar(100)	N

Bemerkungen:

Das Attribut `signatur` wird im Originalentwurf der Relation `exemplar` zugeordnet. Da die Reihen noch nicht in einer eigenen Relation verwaltet werden, liegt hier auch kein Fremdschlüssel für die Reihe vor. Statt dessen wird die Angabe der Reihe einschließlich Bandnummer aus dem Dump übernommen und liegt als String im Attribut `reihe` vor.

artikel

Attribute	Typ	Kommentar
artikelid	integer	NN, Primärschlüssel, clustered index artikel_pk
titel	varchar(255)	NN, Titel des Artikels
austitel	varchar(255)	NN, Titel der Quelle
jahr	smallint	N
seite_von	smallint	N, Anfangsseite
seite_bis	smallint	N, letzte Seite
url	varchar(200)	N, URL für Online-Dokument
cr	varchar(80)	N, CR-Klassifikation
abstractid	integer	N, Fremdschlüssel (Trigger)
buchid	integer	N, Fremdschlüssel (Trigger)
bemerkung	varchar(100)	N

Bemerkungen:

Die Fremdschlüssel für `zeitschrausgid` und `tagungsid` fehlen hier, da in dieser Implementation die Zeitschriften noch nicht erfaßt werden und die Tagungsbände der Relation `buchtitel` zugeordnet werden.

abstract

Attribute	Typ	Kommentar
abstractid	integer	NN, Primärschlüssel
abstracttext	text	NN, Text des Abstracts

Bemerkung:

Es liegen keine Abweichungen zum Originalentwurf vor.

schlagwort

Attribute	Typ	Kommentar
swid	integer	NN, Primärschlüssel, clustered index schlagwort_pk
schlagwort	varchar(50)	NN, UNIQUE, nonclustered index schlagwort_schlag_2080

Bemerkung:

Es liegen keine Abweichungen zum Originalentwurf vor.

autoren

Attribute	Typ	Kommentar
autorid	integer	NN, Primärschlüssel, clustered index autoren_pk
name	varchar(30)	NN, nonclustered index autindex
zusatz	varchar(10)	N, Namenszusatz wie van, van der
initialen	varchar(6)	N, (nicht ausgeschriebene Vornamen)
vornamen	varchar(30)	N
institution	varchar(50)	N
url	varchar(200)	N, URL für Homepage
pseudonym	integer	N, autorid (Trigger)
bemerkung	varchar(100)	N

Bemerkung:

Es liegen keine Abweichungen zum Originalentwurf vor.

verlag

Attribute	Typ	Kommentar
verlagsid	integer	NN, Primärschlüssel, clustered index verlag_pk
name	varchar(80)	NN, nonclustered index verlagindex
ort	varchar(80)	NN
isbnkz	varchar(10)	N, ISBN-Kennzahl des Verlags
adresse	varchar(80)	N
tel	varchar(20)	N
fax	varchar(20)	N
url	varchar(200)	N, URL für Homepage
bemerkung	varchar(100)	N

Bemerkung:

Es liegen keine Abweichungen zum Originalentwurf vor.

institution

Attribute	Typ	Kommentar
instid	integer	NN, Primärschlüssel, clustered index institution_pk
name	varchar(100)	NN
ort	varchar(80)	NN
adresse	varchar(80)	N
tel	varchar(20)	N
fax	varchar(20)	N
url	varchar(200)	N, URL für Homepage
bemerkung	varchar(100)	N

Bemerkung:

Es liegen keine Abweichungen zum Originalentwurf vor.

Relationen für die n:m Beziehungen

Bemerkungen:

Die Relation `ausleihe` fehlt in der Implementation, da die Exemplare und Leser noch nicht in der Datenbank verwaltet werden. Ebenso fehlen die Relationen `tagung_aut`, `tagung_sw`, `nonprint_aut`, `nonprint_sw` und `zeitschrift_sw`. Alle weiteren Relationen weisen keine Abweichungen vom Originalentwurf auf.

buch_aut

Attribute	Typ	Kommentar
buchid	integer	NN, Fremdschlüssel
autorid	integer	NN, Fremdschlüssel
rolle	char	NN, v = Verfasser, h = Herausgeber, u = Übersetzer, m = Mitarbeiter etc.
rang	smallint	N, 1..9

diss_aut

Attribute	Typ	Kommentar
dissid	integer	NN, Fremdschlüssel
autorid	integer	NN, Fremdschlüssel
rolle	char	NN, v = Verfasser, b = Betreuer
rang	smallint	N, 1..9

art_aut

Attribute	Typ	Kommentar
artikelid	integer	NN, Fremdschlüssel
autorid	integer	NN, Fremdschlüssel
rolle	char	NN, v = Verfasser, h = Herausg., u = Übersetzer
rang	smallint	N, 1..9

report_aut

Attribute	Typ	Kommentar
reportid	integer	NN, Fremdschlüssel
autorid	integer	NN, Fremdschlüssel
rang	smallint	N, 1..9

buch_sw

Attribute	Typ	Kommentar
buchid	integer	NN, Fremdschlüssel
swid	integer	NN, Fremdschlüssel

diss_sw

Attribute	Typ	Kommentar
dissid	integer	NN, Fremdschlüssel
swid	integer	NN, Fremdschlüssel

art_sw

Attribute	Typ	Kommentar
artikelid	integer	NN, Fremdschlüssel
swid	integer	NN, Fremdschlüssel

report_sw

Attribute	Typ	Kommentar
reportid	integer	NN, Fremdschlüssel
swid	integer	NN, Fremdschlüssel

Anhang 2 Erläuterungen zu den *allegro*-Kategorien

Im folgenden werden die in der Zweigstelle Informatik/Rechenzentrum der Universitätsbibliothek Leipzig verwendeten *allegro*-Kategorien und ihre Entsprechung in der Implementation des Literaturverwaltungssystems nach Anhang 1 aufgeführt.

Kennzahl	Bedeutung	Verwendung im Schema
00	Identifikationsnummer	-- nicht benötigt
01	Bandbezeichnung (bei unselbständig gespeicherten Untersätzen)	Attribut bandnr in Tabelle buchtitel
15 u. 15a	Nebeneintragungsvermerk	-- nicht benötigt
20	Hauptsachtitel : Zusatz (lt. Haupttitelseite)	Attribut hstitel in Tabelle buchtitel bzw. Attribut titel in Tabelle dissertation u. report
22	Einheitssachtitel <Sprache> z.B. Originaltitel bei Übersetzungen	Attribut estitel in Tabelle buchtitel
30b	CR-Klassifikation	Attribut cr in Tabelle buchtitel, dissertation bzw. report
31, 31a u. 31b	Schlagwort	Attribut schlagwort in Tabelle schlagwort
39	Verfasserangabe in Vorlageform	Attribut verf_in_vorlform in Tabelle buchtitel
40, 40a u. 40b	1., 2. u. 3. Verfasser in der Form Name, Vorname(n)	Attribute name, vornamen, initialen und zusatz in Tabelle autoren, Attribute rolle='v' und rang=1,2 o. 3 in Relation buch_aut bzw. diss_aut
41, 41a u. 41b	1., 2. u. 3. Herausgeber	Attribute name, vornamen, initialen und zusatz in Tabelle autoren, Attribute rolle='h' und rang=1,2 o. 3 in Relation buch_aut
42	1. Autor bei mehr als 3 Verfassern	Attribute name, vornamen, initialen und zusatz in Tabelle autoren, Attribute rolle='v' und rang=1 in Relation buch_aut
56	Verfasser bei Dissertationen	Attribute name, vornamen, initialen und zusatz in Tabelle autoren, Attribute rolle='v' und rang=1 in Relation diss_aut
60, 60a	Urheber (verantwortl. Körperschaft im Sinne von RAK)	Attribut kschaft1 bzw. kschaft2 in Tabelle buchtitel

Kennzahl	Bedeutung	Verwendung im Schema
61, 61a	beteiligte Körperschaft	Attribut kschaft1 bzw. kschaft2 in Tabelle buchtitel
71	Auflagebezeichnung	Attribut auflage in Tabelle buchtitel
74	Erscheinungsort	Attribut ort in Tabelle verlag bzw. institution (bei Dissertationen u. Reports)
75	Verlag	Attribut name in Tabelle verlag bzw. institution (bei Dissertationen u. Reports)
76	Erscheinungsjahr	Attribut jahr in Tabelle buchtitel, dissertation bzw. report
77	Umfangsangabe : Illustr. u. Begleitmaterial	Attribut umfang in Tabelle buchtitel, dissertation bzw. report
81h	Hochschulschriftenvermerk	Attribut hsv in Tabelle buchtitel
85	Gesamtsachtitel ; Zählung (Reihe)	Attribut reihe1 in Tabelle buchtitel bzw. Attribut reihe in Tabelle report
85a	weitere Reihe	Attribut reihe2 in Tabelle buchtitel
87	ISBN ; 2. ISBN	Attribut isbn (u. bei Bedarf isbn_2) in Tabelle buchtitel bzw. dissertation
90	Standortsignatur	Attribut signatur in Tabelle buchtitel, dissertation bzw. report (eigentlich in Tabelle Exemplar)
91	UB-Signatur	Attribut zugangsnr in Tabelle buchtitel bzw. dissertation (eigentlich in Tabelle Exemplar)
92	Bestandsangaben (Anzahl der Exemplare oder Signatur weiterer Exemplare wenn diese verschieden ist)	Attribut bestand in Tabelle buchtitel bzw. dissertation (bei getrennter Erfassung der Exemplare in Relation exemplar nicht mehr benötigt)
93	weitere Exemplare (Signatur)	Attribut weitexemp in Tabelle buchtitel (bei getrennter Erfassung der Exemplare in Relation exemplar nicht mehr benötigt)
95	Preis	Attribut preis in Tabelle buchtitel
98	Bezugsinformationen	Attribut bezugsinfo in Tabelle buchtitel
99e	Änderungsdatum	aendatum in Tabelle buchtitel (eigentlich in Tabelle Exemplar)
99n	Neuerfassungsdatum	erfdatum in Tabelle buchtitel (eigentlich in Tabelle Exemplar)

Anhang 3 Die verwendeten Programme und Skripte

Bemerkung:
Die in Abschnitt 1 und 2 beschriebenen Programme und Skripte befinden sich derzeit in den jeweiligen Unterverzeichnissen des Verzeichnisses:

`/dargb/local_lips/sybase/work/anjatest`

sowie auf der beigefügten Diskette im Verzeichnis **`LADEN`**
Die im 3. Abschnitt beschriebenen HTML- und HTS-Files befinden sich im Verzeichnis:

`/dargb/httpd/htdocs/websql.dir/biblio`

sowie auf der beigefügten Diskette im Verzeichnis **`HTML`**

1. Die Programme für das Laden der *allegro*-Daten aus der Zweigstelle Informatik/Rechenzentrum in die im Anhang 1 beschriebene Datenbank:

Der Quellcode aller Programme befindet sich im Verzeichnis `sources`, die für die Vorbearbeitung des Dumps benötigten ausführbaren Programme liegen im Verzeichnis `dumpbearbeiten` und die eigentlichen Ladeprogramme sind im Verzeichnis `laden`. Ausführbare Programme sind durch die Endung `.x` gekennzeichnet.

Vor dem Beginn des eigentlichen Ladevorgangs wird der Dump mit den *allegro*-Daten vorbearbeitet, dabei werden die Steuerzeichen interpretiert, Umlaute durch Zeichenkombinationen ersetzt, fehlerhafte Kennzahlen berichtigt, die Schreibweise von Verlagen berichtigt und fehlerhafte Autoreneinträge aufgelistet. Nach einer eventuellen manuellen Nachbearbeitung des Dumps kann dann der Ladevorgang beginnen. Dabei werden zunächst mit dem Programm `insertaut.x` alle Verfasser und Herausgeber in die Tabelle `autoren` geladen. In einem zweiten Schritt werden dann mit dem Programm `inserttitel.x` die restlichen Daten geladen, wobei Verlage, Institutionen und Schlagwörter in die jeweiligen Tabellen eingefügt werden, falls sie dort noch nicht vorhanden waren. Die Titel werden entsprechend der Signatur in Buchtitel, Dissertationen und Reports aufgeteilt und in die entsprechenden Tabellen eingefügt.

Die im Verzeichnis **`dumpbearbeiten`** befindlichen Files und deren Bedeutung:

`kat_1.ald`	eine Kopie des Dumps, wie er vom *allegro*-System gespeichert wird, mit Steuerzeichen, die semantische Zusammenhänge kodieren, diese Datei wird von `kat_1.ald.into.dump.x` benötigt
`kat_1.ald.into.dump.x`	ein C-Programm, das aus dem File `kat_1.ald` das File `dump` erzeugt, in welchem jeder Kategorie eine eigene Zeile zugeordnet ist, die Steuerzeichen herausgefiltert und interpretiert wurden sowie die Umlaute durch /ae, /sz usw. ersetzt wurden, außerdem korrigiert das Programm z. T. die Schreibweise der Verlage
`dump`	von `kat_1.ald.into.dump.x` erzeugtes File mit dem Bibliotheksdump
`cleandump.x`	ein C-Programm, das das File `dump` zeilenweise einliest und fehlerhafte Kategorien berichtigt, die Änderungen werden im File `cdprotokoll` mitgeschrieben, Fehler, die nicht behoben werden konnten, werden im File `cdnochzutun` aufgelistet, der korrigierte Dump wird in das File `neudump` geschrieben
`cdprotokoll`	von `cleandump.x` erzeugtes File, protokolliert erfolgreiche Änderungen
`cdnochzutun`	von `cleandump.x` erzeugtes File mit den noch zu behebenden Fehlern
`neudump`	von `cleandump.x` erzeugtes File mit dem korrigierten Dump, zur weiteren Verwendung durch die Ladeprogramme muß dieses File nach erfolgter Korrektur der restlichen Fehler in das Verzeichnis `laden` kopiert werden
`README`	Hinweise zum Gebrauch der Programme

Die im Verzeichnis **laden** befindlichen Files und deren Bedeutung:

dump	das von `cleandump.x` erzeugte File `neudump` muß (eventuell) manuell berichtigt werden und dann aus dem Verzeichnis `dumpbearbeiten` unter dem Namen `dump` in das Verzeichnis `laden` kopiert werden
insertaut.x	liest die Autorendaten aus dem File `dump` und fügt in der Datenbank noch nicht vorhandene Autoren in die Tabelle `autoren` ein, Fehler werden in die Datei `auterror` geschrieben. Sämtliche autorids werden mit den dazugehörenden Datensatzkennnummern in das File `autid` geschrieben, welches von `inserttitel.x` benötigt wird
auterror	wird von `insertaut.x` erzeugt und enthält Fehlermeldungen
autid	wird von `insertaut.x` erzeugt und enthält autorids, wird von `inserttitel.x` benötigt
inserttitel.x	liest die Titeldaten aus dem File `dump` und fügt sie in die Datenbank ein, dazu wird das File `autid` benötigt, Fehlermeldungen werden in das File `titelerror` geschrieben
titelerror	wird von `inserttitel.x` erzeugt und enthält Fehlermeldungen
inserturl.x	liest aus einem File namens `verlage.html` die URLs und fügt sie in die Hilfstabelle `verlag_urls` ein, nähere Hinweise dazu sind im File `README` zu finden, Fehlermeldungen werden in das File `urlerror` geschrieben
urlerror	wird von `inserturl.x` erzeugt und enthält Fehlermeldungen
verlage.html	wird von `inserturl.x` benötigt, muß von der URL http://www.darmstadt.gmd.de/BV/agef_5.html#Verlage stammen, nähere Hinweise dazu sind im File `README` zu finden
README	Hinweise zum Gebrauch der Programme

Das Ladeprogramm lädt jeweils den gesamten Bibliotheksdump, nicht nur die seit dem letzten Ladevorgang hinzugekommenen Einträge, da auch die älteren Einträge von der Bibliothekarin ständig bearbeitet und korrigiert bzw. erweitert werden. Daher müssen vor jedem Ladevorgang die alten *allegro*-Daten aus der Datenbank gelöscht werden, um immer die aktuellen Daten bereitstellen zu können und doppelte Einträge zu vermeiden. Diesem Zweck dient das Skript `empty.allegrodump-fromdb.sql`, welches die Titeleinträge aus den Tabellen `buchtitel`, `dissertation` und `report` (jedoch nicht die Reports des IfI) löscht und die zugehörigen Einträge aus den Tabellen `buch_sw`, `buch_aut` usw. entfernt. Nicht gelöscht werden die erfaßten Autoren, Verlage, Institutionen und Schlagwörter sowie die selbsterfaßten

Artikel und Reports.

Da aufgrund fehlender Standardisierung die von den Bibliotheken verwendeten Kategorien bei der Erfassung von Literaturbeständen stark voneinander abweichen können, ist das vorliegende Ladeprogramm in dieser Form nur für das Laden der *allegro*-Daten der UB Leipzig Zweigstelle Informatik/URZ verwendbar. Um die Bestände anderer Bibliotheken laden zu können, muß das Ladeprogramm an deren Kategoriensystem angepaßt werden.

2. Die Skripts zum Anlegen, Leeren und Löschen der Datenbank laut Anhang 1 (aktdb) und der gesamten Datenbank laut Entwurf (gesamtdb) sowie der Trigger und Indizes

Die im folgenden aufgelisteten Files befinden sich im Verzeichnis **isqlbatches.**

cr.gesamtdb.sql	erzeugt Tabellen der gesamten Datenbank entsprechend Entwurf
empty.gesamtdb.sql	löscht Inhalt aller Tabellen der gesamten Datenbank
del.gesamtdb.sql	löscht Trigger und Tabellen der gesamten Datenbank
cr.aktdb.sql	erzeugt Tabellen der Datenbank nach Anhang 1
empty.aktdb.sql	löscht Inhalt aller Tabellen der Datenbank nach Anhang 1
del.aktdb.sql	löscht Trigger und Tabellen der Datenbank nach Anhang 1
empty.allegrodumpfromdb.sql	löscht die *allegro*-Daten aus den Tabellen buchtitel, dissertation, report, buch_sw, buch_aut, diss_sw, diss_aut, report_sw, report_aut. Muß vor dem Start des Ladeprogramms aufgerufen werden.
cr.trigger.sql	legt Trigger zur Integritätssicherung an (siehe Übersicht auf der folgenden Seite)
cr.index.sql	legt Indizes auf `autoren`, `buchtitel` und `verlag` an
grant.www.sql	Rechtevergabe für User `www` in Datenbank laut Anhang 1
README	Hinweise zum Gebrauch der Programme

Übersicht über die mit `cr.trigger.sql` angelegten Trigger:

`TRIGGER buchinsert ON buchtitel FOR INSERT, UPDATE`

- prüft, ob `hstitel` = NULL (wird nur für Bandaufnahme erlaubt, d.h. `gesamtid` darf dann nicht ebenfalls NULL sein)
- prüft, ob zu angegebener `gesamtid` ein Eintrag mit der entsprechenden `buchid` in der Tabelle `buchtitel` existiert
- prüft, ob zu angegebener `abstractid` ein Eintrag in der Tabelle `abstract` existiert

`TRIGGER aut_pseu ON autoren FOR INSERT, UPDATE`

- prüft, ob zu angegebenem `pseudonym` ein Eintrag mit der entsprechenden `autorid` in der Tabelle `autoren` existiert

`TRIGGER artinsert ON artikel FOR INSERT, UPDATE`

- prüft, ob zu angegebener `abstractid` ein Eintrag in der Tabelle `abstract` existiert
- prüft, ob zu angegebener `buchid` ein Eintrag in der Tabelle `buchtitel` existiert

`TRIGGER reportinsert ON report FOR INSERT, UPDATE`

- prüft, ob zu angegebener `abstractid` ein Eintrag in der Tabelle `abstract` existiert

`TRIGGER dissinsert ON dissertation FOR INSERT, UPDATE`

- prüft, ob zu angegebener `abstractid` ein Eintrag in der Tabelle `abstract` existiert

3. Die für die Realisierung der Online-Recherche erstellten HTML- und HTS-Files

Die im folgenden aufgeführten Files befinden sich im Verzeichnis:

`/dargb/httpd/htdocs/websql.dir/biblio`

sowie auf der beigefügten Diskette im Verzeichnis **`HTML`**.

`olrecherche.html` Titelseite der Online-Recherche mit einem HTML-Formular zur Eingabe der Suchkriterien, beim Abschicken des Formulars wird `search.hts` aufgerufen, ein weiterer `SUBMIT`-Knopf steht für die alphabetische Anzeige aller Schlagworte bereit, dafür wird `sw.hts` aufgerufen

`search.hts` wird von `olrecherche.html` aus aufgerufen, sucht zu den vom Benutzer eingegebenen Kriterien alle Bücher, Dissertationen, Reports und Artikel aus der Datenbank und gibt diese formatiert und mit Hyperlinks auf weitere Informationen aus, die Suchparameter werden mit der Methode `POST` übergeben

`sw.hts` wird von `olrecherche.html` aus aufgerufen, sucht alle Schlagwörter aus der Datenbank und gibt diese in einer alphabetischen Liste aus, die Schlagwörter werden mit dynamischen Hyperlinks versehen, die bei ihrer Aktivierung das File `searchsw.hts` aufrufen und ihm die `swid` des betreffenden Schlagworts übergeben

`searchsw.hts` wird von `sw.hts` aus aufgerufen, sucht zu der übergebenen `swid` die zugehörigen Bücher, Artikel, Reports und Dissertationen und gibt diese formatiert aus, die `swid` des Schlagworts wird mit der Methode `GET` übergeben

`searchgesamt.hts` dieses File wird benötigt, um das zu einem Artikel gehörende Buch bzw. bei Büchern mit mehreren Bänden die gesamten Bände nebst Hauptaufnahme anzusehen, das File kann von den Ergebnisseiten von `search.hts` bzw. `searchsw.hts` aus über dynamische Hyperlinks aufgerufen werden, dabei wird die entsprechende `buchid` mittels der Methode `GET` übergeben

`bemerkungen.html` enthält Hinweise zur Benutzung der Online-Recherche

`CRKlassifikation.html` dieses File enthält eine Beschreibung der für die Klassifizierung von Informatikliteratur verwendeten CR-Klassifikation, das HTML-File wurde von Frau Dr. Anne Brüggemann-Klein am Institut für Informatik der Universität Freiburg erstellt und von mir übernommen

`README` enthält eine Übersicht über die oben aufgeführten Files

Hinweise zur beigefügten Diskette

Die Diskette enthält die in Anhang 3 beschriebenen Ladeprogramme, SQL-Skripte und HTML- und HTS-Files sowie diese Arbeit als Framemaker-Dokument und als PostScript-File. Es folgt eine kurze Übersicht über die Verzeichnisse und Dateien auf der Diskette.

Verzeichnis **LADEN**

Unterverzeichnis **dumpbearbeiten**

```
kat_1.ald.into.dump.x
cleandump.x
README
```

Unterverzeichnis **laden**

```
insertaut.x
inserttitel.x
inserturl.x
README
```

Unterverzeichnis **isqlbatches**

```
cr.gesamtdb.sql
empty.gesamtdb.sql
del.gesamtdb.sql
cr.aktdb.sql
empty.aktdb.sql
del.aktdb.sql
empty.allegrodumpfromdb.sql
cr.trigger.sql
cr.index.sql
grant.www.sql
README
```

Verzeichnis **HTML**

```
olrecherche.html
search.hts
sw.hts
searchsw.hts
searchgesamt.hts
bemerkungen.html
CRKlassifikation.html
README
```

Verzeichnis **DOK**

```
beckmann.da.ps
beckmann.da.frm
```