Bibliothekssystem der Computerlinguistik

Dainius Ramanauskas

31.03.2002

Inhaltsverzeichnis

Kapitel 1

Einführung

Die Computerlinguistik Bibliothek (Abk. Colibi) umfasst die Bibliotheken der Computerlinguistik und der Phonetik im Computerlinguistikinstitut. Mit Colibi ist sowohl die Bibliothek, als auch die Software, die dort läuft gemeint.

1.1 Motivation

Nach dem Ausfall des bisherigen Datenbanksystems, das zur Speicherung der Bibliotheksdaten genutzt wurde, war eine neue Lösung gefordert. Die alten Daten blieben erhalten (in einer Text Form), da sie regelmäßig vom IBIS (Informatik Bibliotheken in Saarbrücken) gesichert wurden. Somit gab es drei Lösungsmöglichkeiten:

Man kauft eine kommerzielle Datenbank, man kooperiert mit dem DFKI oder der Informatik Bibliothek und übernimmt die Datenbanken, die schon im Betrieb sind, oder man entwirft die Datenbank mit aktuellen Software-Tools vollständig neu.

Die erste Alternative hatte schon am Anfang wenig Chancen. Zum einem sind kommerzielle Datenbanken recht teuer, zum anderen müsste die Software aufwendig an die Colibi Bedürfnisse angepasst, bzw. umprogrammiert werden. Umprogrammieren wäre nur schwer möglich, da normalerweise kein Quellcode von kommerzieller Software verfügbar ist. Die Suche nach der richtigen Datenbank, die Einarbeitungszeit und die möglichen Anpassungen hätten viel Zeit in Anspruch genommen.

Die zweite Wahl kam schon etwas besser an. Die DFKI Bibliothek hatte recht gute Software für Macintosh Computer. Es war aber eine UNIX-

Umgebung gewünscht, da die gesamte Computerlinguistik UNIX einsetzt. Somit entfiel auch diese Alternative.

Die Software der Informatik Bibliothek war zu sehr für die Bedürfnisse deren Bibliothek angepasst und auf den ersten Blick nicht so einfach zu durchschauen. Das Anpassen an die Colibi hätte also auch eine Unmenge an Zeit gekostet.

Die dritte Möglichkeit des Neuaufbaus warf gleich zwei Fragen auf: Womit wird es funktionieren und wird es überhaupt funktionieren? Zu dem Zeitpunkt gab es recht viele Artikel und Bücher darüber, wie man "einfach" mit den aktuellen relationalen Datenbanken wie z.B. MySQL und der Skriptsprache PHP für alle möglichen Bedürfnisse Datenbanken aufbauen könnte.

1.2 Welche Funktionalität braucht die Colibi?

Im Laufe der Planung hat sich folgende zu implementierende Funktionalität herauskristallisiert:

- Alte Daten der Colibi sollten übernommen werden (etwa 8000 Bücher und Artikel) und neue sollten weiter eingegeben werden können (Bücher, Artikel, Benutzer).
- Die Benutzer (Mitarbeiter) benötigten eine GUI, die möglichst deren Anforderungen entspricht (die Daten von Büchern, Artikeln und Benutzern müssten eingegeben, korrigiert, gesucht und gelöscht werden können). Einfache Statistiken sollten zu sehen sein, wie z.B. der Status von ausgeliehenen Büchern. Die Benutzer (Entleiher) bräuchten eine Internet Seite, von der die Bücher abgefragt werden können.
- Die Colibi sollte die alten Funktionen beibehalten und neue dazu bekommen: Tägliches Exportieren von Colibi Daten zum IBIS, Warnungs-Emails sollten automatisch verschickt werden ("Sie haben das Buch "xy" zu lange ausgeliehen.").

1.3 Software

Es bot sich MySQL als Datenbank an, des Weiteren ein beliebiger Browser als GUI und PHP, als eine Skriptsprache, die hervorragende Funktionen zur

Ansprechung der Datenbanken mit sich bringt und auch die angefragten Daten ohne viel Aufwand an den Browser sendet. Dies alles läuft unter einer UNIX-Umgebung, die noch eine Menge an GNU-Tools liefert.

Die Colibi Daten werden in MySQL abgelegt. Mit einem beliebigen Browser kann die Seite der Colibi im Internet erreicht werden, und nach Eingabe des Kennwortes erscheint die Benutzeroberfläche mit Verweisen zu den wichtigsten Funktionen. Das ist die grafische Oberfläche der Colibi. Die Seiten der Colibi sind teilweise in HTML programmiert, teilweise in PHP. Die Abfrage der Daten von MySQL, wie z.B. "Büchersuche", ist in PHP geschrieben.

Somit lassen sich alle Daten (Bücher, Artikel, Entleiher) von der MySQL-Datenbank mit Hilfe von PHP auf den Browser bringen, und sie können dort geändert werden.

Zu den besonderen Eigenschaften zählt das Exportieren von Daten zur IBIS und das Verschicken von Warnungs-Emails. Es wurden PERL- und PHP-Skripte geschrieben, die von Cronjobs (sie starten zum gewünschten Zeitpunkt Programme, Skripte usw.) aufgerufen werden. Die Skripte fragen die Colibi-Daten von Tabellen der MySQL-Datenbank ab und konvertieren sie in ein bestimmtes Format der IBIS. Die Daten werden von der IBIS per FTP abgeholt.

Das Verschicken von Warnungs-Emails wird auch von den PHP-Skripten vollzogen.

1.4 Datenbankdesign

Wenn eine Datenbank entworfen wird, sollte man als erstes die Informationen und Abläufe sammeln, die die Anwender vom Datenbanksystem erwarten. Danach kann man sich überlegen, welche Tabellen benötigt werden und die Typen für die Spalten festlegen. Weitere Details sind in Kapitel 2 und 3 zu finden.

Man sollte sich für die Planung ein wenig Zeit nehmen, weil es nachher sehr schwer ist, ohne großen Aufwand Fehler zu beheben.

Die Colibi Datenbank kann in vier Bereiche eingeteilt werden:

1. Bücher
2. Artikel, Zeitschriften
3. Benutzer

4. Status

Die Gruppierung schafft die nötige Übersicht.

Solch eine Auflistung kann jedoch nicht immer direkt in entsprechende Tabellen übernommen werden, da die Tabelle "Bücher" wegen der Redundanz in zwei Tabellen unterteilt werden muss. Somit wird verhindert, dass z.B. zwei Exemplare eines Buches zwei Mal eingegeben werden und somit unnötig Speicher belegen, oder später Inkonsistenzen hervorrufen. Außerdem wird an Zeit gespart, weil man die Daten eines Buches nur ein Mal eingeben muss. Es gibt also eine Tabelle Bücher, die Stammdaten der Bücher enthält, d.h. Daten die sich nicht ändern, wie Autor, Titel, ISBN, etc. Des Weiteren gibt es eine sogenannte Journaling-Tabelle, die alle Daten vom Exemplar eines Buchs enthält, wie Signatur, Standort, Status (ob das Exemplar ausgeliehen ist oder nicht). Die beiden Tabellen haben einen primären Schlüssel. Die Tabelle "Exemplar" enthält einen primären Schlüssel, der eindeutig das Buch von der Tabelle "Bücher" identifiziert. Des Weiteren enthält die Tabelle "Exemplar" einen Schlüssel "Signatur". Bei der Eintragung wird die Signatur auf die Eindeutigkeit überprüft.

Alle Artikel der Colibi werden in einer Tabelle gespeichert. Es hat sich als nicht lohnend herausgestellt, die Tabelle "Artikel" in weitere Tabellen zu unterteilen, da Artikel im allgemeinen nur ein Mal eingetragen werden. Sie können auch nicht ausgeliehen werden. Zeitschriften eines ganzen Jahres werden zusammengebunden und als Buch erneut eingetragen. Wenn man z.B. nach einem Artikel sucht, findet man auch einen Vermerk, in welcher Zeitschrift dieser zu finden ist.

Benutzerdaten werden, genau wie Artikeldaten, auch in einer Tabelle eingetragen. Die Tabelle beinhaltet solche Informationen wie Name, Adresse, Email, Studienfach. Als Schlüssel sind ein ganzzahliger Wert und die Email-Adresse eingetragen. Die Email-Adresse wird bei der Ausleihe in der Tabelle "Status" eingetragen, was für das Verschicken von Warnungs-Emails wichtig ist.

Die Tabelle "Status" enthält alle Bücher, die ausgeliehen worden sind. Es wird die Email-Adresse des Benutzers, der Verleihzeitraum des Buches, sowie die Signatur des Buches eingetragen. Warnungs-Emails werden verschickt, wenn das Buch zu lange ausgeliehen wurde.

Kapitel 2

Datenbankentwicklung

Wie schon im vorigen Kapitel erwähnt, sind mehrere Schritte zur Entwicklung von Datenbanken nötig. Grundsätzlich muss man dabei folgende Regeln beachten:

2.1 Grundlagen

Bei der Datenmodellierung sollte keine Redundanz vorkommen, da sie grundsätzlich zwei Nachteile mit sich bringt: Zum einen wird mehr Speicherplatz verbraucht, zum anderen sind Änderungen schwieriger durchzuführen. Diese Änderungen sind anfälliger für Fehler, da zwei Attribute geändert werden müssen. Z.B. werden in der Colibi Buch-Daten in zwei Tabellen aufgeteilt: in "Buch"- und "Exemplar"-Tabellen. Die Daten, die sich nicht ändern (z.B. Autor, Titel, etc.), werden in der "Buch"-Tabelle gespeichert. Daten, die sich ändern köennen, sind in der Tabelle "Exemplar" untergebracht.

Ein bestimmtes Tupel in einer Tabelle muss eindeutig identifizierbar sein, weil es keinen definierten Zugriffsweg auf einen bestimmten Datensatz gibt. Daher werden die Tabellen um eine Spalte erweitert, die einen Schlüssel enthält. Mit dem Schlüssel kann dann eindeutig auf eine Zeile einer Tabelle zugegriffen werden. In der Colibi wird für jede Tabelle ("Bücher","Exemplar","Artikel" und "Benutzer") ein eindeutiger Schlüssel generiert, ein ganzzahliger Wert. Jedes Exemplar eines Buches wird eindeutig mit dem primären Schlüssel "Signatur" und jeder Benutzer mit seiner Email-Adresse identifiziert.

Es gibt mehrere Vorgehensweisen zur Entwicklung von Datenmodellen. Die bevorzugte Variante dafür ist, dass man eine Analyse bezüglich der Da-

ten und ihrer optimalen Organisation durchführt. Anschließend kann man sich die entsprechenden Funktionen überlegen. Im nächsten Schritt überprüft man, ob alle Anforderungen realisierbar sind.

Um die benötigten Tabellen zu entwickeln, gibt es für einfache Datenbanken zwei Möglichkeiten: Entweder über die Normalformen oder, etwas intuitiver, über das ER-Modell (Kapitel 3), evtl. anschließend mit Kontrolle durch Normalformen. Erst wenn größere Datenbanken entwickelt werden, muss man mit beiden Möglichkeiten gleichzeitig arbeiten.

Die Colibi wurde entwickelt, indem zunächst die Informationen und Abläufe gesammelt und analysiert wurden. Des Weiteren wurden die Entitäten mit Attributen festgelegt und Schlüssel definiert.

2.2 Normalisierung

Mit der Normalisierung wird das erstellte Relationsmodell (Kapitel 3) noch einmal auf logische Korrektheit überprüft. Das Ziel der Normalisierung ist die Vermeidung redundanter Daten. Die ersten drei Normalformen decken in der Regel die häufigsten Probleme ab, die bei einem Datenmodell auftreten können. Daher wird die Colibi nur mit den ersten drei Normalformen analysiert, um die Tabellen zu überprüfen.

1. **Erste Normalform (vorausgesetzt, dass es nur nur eine Tabelle "Buch" gibt):**

   - **Attribute mit mehreren Werten in atomare Attribute aufspalten.** Das erfüllen nicht alle Tabellen der Colibi Datenbank: Z.B. werden in den Tabellen "Buch" und "Artikel" Autoren oder Editoren in einer Spalte gespeichert. Es ist sinnvoll, die Werte in so wenigen Tabellen wie möglich zu verteilen, da das Abfragen von wenigen Tabellen wesentlich schneller ist und das Entwickeln der GUI viel einfacher und nicht so fehleranfällig ist. Die Werte der Tabelle "Benutzer" werden atomar gespeichert.
   - **Wiederholungsgruppen in mehrere Zeilen aufteilen.** Wenn zwei Exemplare eines Buches vorhanden sind, existieren dann Einträge doppelt. So unterscheiden sich z.B. Autor und Titel nicht, die Signaturen hingegen aber schon.

2. **Zweite Normalform:**

- **Überprüfen der vollen funktionalen Abhängigkeit der Nichtschlüssel-Attribute zum Primärschlüssel.** Die Tabelle "Bücher" von der ersten Normalform wird in zwei Tabellen aufgeteilt: in "Buch" und "Exemplar" Tabellen. Die Tabelle "Buch" enthält solche Attribute wie Autor, Titel, primären Schlüssel (ganzzahliger Wert), etc. Die Tabelle "Exemplar" enthält solche Attribute wie Signatur, Standort, Verfügbarkeit, primären Schlüssel der Tabelle "Buch", die dann alle Attribute enthält, die voll funktional vom primären Schlüssel abhängig sind. Die Tabelle "Exemplar" erhält Attribute, die voll funktional vom primären Schlüssel der Tabelle "Buch" und der Signatur abhängig sind.

3. **Dritte Normalform:**

   - **Überprüfen von transitiven Abhängigkeiten des Primärschlüssels zu den Nichtschlüssel-Attributen.** In der Tabelle "Benutzer" treten solche funktionalen Abghängigkeiten bei der Bestimmung von Ort und Postleitzahl eines Benutzers auf. Der Primärschlüssel der Benutzertabelle ist die Benutzernummer. Über die Benutzernummer kann eindeutig die Postleitzahl und der Ort eines Benutzers bestimmt werden. Andererseits erkennen wir hier aber auch eine funktionale Abhängigkeit zwischen der Postleitzahl und dem Ort: Ist die Postleitzahl eines Kunden bekannt, so kann auch eindeutig auf den Ortsnamen geschlossen werden. Die dritte Normalform ist in der Colibi nicht implementiert.

Generell gilt, dass ein sauber modelliertes ERM bzw. UML-Modell in ein redundanzfreies Relationenmodell überführt werden kann.

Dennoch sollte die Normalisierung in drei Fällen angewandt werden: Zum einen zur nachträglichen Überprüfung von Tabellen, bei denen man Unstimmigkeiten vermutet, wie wir gerade bei der dritten Normalform gesehen haben, zum anderen, wenn nachträglich Attribute zur Datenbank hinzugefügt werden sollen. Schließlich wird Normalisierung angewandt, um eine bereits vorhandene relationelle Datenbankstruktur nachträglich zu verändern.

Kapitel 3

Datenbank Entwurf (ER-Diagramme)

3.1 Erstellung der ER-Diagramme

Die ER-Diagramme (Entity Relationship) dienen dazu, dass der Datenbankplaner sich über die Abhängigkeiten der Datensätze klar wird. Einem ER-Diagramm kann man Zusammenhänge zwischen den Datensätzen und die Aufteilung der Daten in unabhängige Tabellen entnehmen.

3.2 Darstellung der Entitätstypen und Relationen

In den Diagrammen werden die Entitäten, die Beziehungen der Entitäten untereinander, also das **Relationship** und die **Komplexität** der Beziehung aufgezeichnet. Der Übersicht halber gibt es bestimmte Diagrammtypen :

Entitätstypen werden durch ein Rechteck dargestellt, und die zugehörigen Attribute mit einer Ellipse umrahmt.

Das Buch wird mit einem Rechteck umrahmt, die Attribute (der Rest) mit einer Ellipse. Mehrwertige Attribute, z.B. der Autor (weil ein buch Co-Autoren haben kann) werden mit einer doppelten Ellipse umrahmt.

Es gibt neben der Darstellung **einfacher** und **mehrwertiger** Attribute auch noch die **zusammengesetzten** Attribute. Alle Attribute werden mit einer Ellipse umrahmt. **Beziehungen** (relationships) werden durch eine

Raute dargestellt, die Entitäten mit einem Rechteck.

Hier noch einmal die Zusammenfassung der Darstellungen:

Entitäten und **Attribute**:

|Buch|
(InvNr) (Titel) ((Autor)) (Verlag) (ISBN)

Mehrfache Attribute werden so dargestellt:

(Adresse)
(Plz) (Ort) (Straße)

Beziehungen (relationships) werden nun so dargestellt:

|Entität 1|——<Beziehungstyp>——|Entität 2|

Die Art der Beziehungen der einzelnen Entities entscheidet später über die Struktur der SQL Datenbank, also wie die Attribute geordnet werden, wieviele Tabellen angelegt werden müssen, usw. Hier stellt sich die Frage nach der Komplexität der Beziehungen (relationships). Die Komplexität der Beziehungen läßt sich an einem kleinen Beispiel darstellen:

|Person|——<wohnt in>——|Ort|

Dieses Beispiel ist eine 1:1 Beziehung, was bedeutet, dass man in der Datenbank **nicht mehrere Felder** (siehe Kapitel 4) für den Wohnort einer Person reservieren muss. Die Tatsache, dass dieses eine 1:1 Beziehung ist, wird durch eine kleine 1 in dem Diagramm dargestellt:

1 1
|Person|——<wohnt in>——|Ort|

Komplexe Beziehungen stellen eine Entität dar, die zu einer bestimmten Zeit mit mehreren Entitäten in Beziehung steht:

1 N
|Leser|——<leiht aus>——|Buch|

Ein Leser kann zu einem bestimmten Zeitpunkt mehrere Bücher ausleihen. Jedes Buch in einer Bibliothek erhält immer eine eindeutige Inventarnummer (in der Colibi wird die Inventarnummer (Signatur) als Schlüssel für alle Operationen verwendet). Alle Exemplare eines Buches in einer Bibliothek gehören zu einer Entitätsmenge, die aus mehreren gleichen Entities besteht. Jedes "anfassbare" Buch erhält in der Colibi eine eindeutige Inventarnummer, genauso wie der Leser eine eindeutige Mitgliedsnummer erhält. Für die Relation bedeutet dies genaugenommen, dass nur zwischen dem Leser, eindeutig identifiziert durch die Mitgliedsnummer, und dem Buch, identifiziert durch eine eindeutige Inventarnummer, eine 1:N Beziehung bestehen kann. Im grunde genommen muss es also heißen:

1 N
|Mitglied 523|——<leiht aus>——|Inventar 1067|
|Mitglied 523|——<leiht aus>——|Inventar 1068|

3.3 Wertigkeiten der Relationen

Bei der Darstellung der Entity-Typen muss man genauer zwischen den Wertigkeiten der Beziehungsarten unterscheiden. Insgesamt erhält man vier verschiedene Beziehungsarten:

1.1:1 eins zu eins
2.1:N eins zu viele
3.N:1 viele zu eins
4.N:M viele zu viele

Damit dies etwas klarer wird, hier ein Diagramm als Endergebnis der Planungen. Wie man dahin kommt, wird danach detailliert beschrieben.

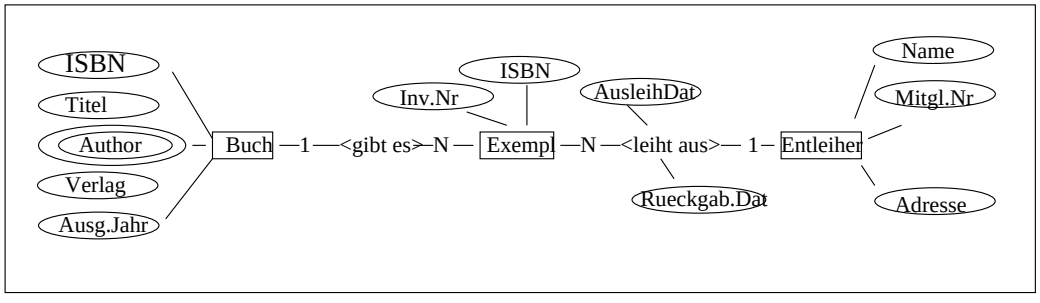

Abb. 4 Diagramm als endergebnis der Planung

Hier kann man nun die verschiedenen Beziehungsarten genau sehen. Betrachten wir die Relation zwischen den Entitäten Buch und Exemplar (gibt es). Hier gibt es eine 1:N Beziehung, weil von einem Buch mehrere Exemplare, also eine Entitätsmenge eines Buches in der Bibliothek vorhanden sein kann. Die Relation (leiht aus) stellt zwischen dem Entleiher und den Exemplaren eine 1:N Beziehung, bzw. zwischen den Exemplaren und dem Entleiher eine N:1 Beziehung dar. In Abbildung 4 ist die Relation N:M nicht eingezeichnet, falls ein Entleiher z.B. mehrere Bücher reservieren lassen möchte. Die Bibliothek hat viele Mitglieder (M), die jeweils mehrere Bücher (N) ausleihen oder reservieren könnten. Hier würde also eine N:M Relation erforderlich sein.

3.4 Wie ER-Diagramme entwickelt werden?

Es gibt eine Möglichkeit die Diagramme systematisch zu entwickeln.

Als Beispiel nehme ich die abgespeckte Version der Colibi und entwickle dann systematisch ein ER-Diagramm.

Zuerst sucht man sich alle Entitäten zusammen (Dinge, Ereignisse, Vorgänge..). In unserem Colibi Fall kommen dann folgende Einträge zusammen:

- Buch
- Person
- Der Vorgang des Ausleihens

Jetzt können wir alle Attribute auflisten, die ein Buch charakterisieren:

- ISBN-Nummer
- Titel
- Die Autoren
- Verlag
- Ercheinungsjahr
- ...

Und jetzt alles auflisten, was eine Person charakterisiert:

- Name
- Adresse
- Anschrift
- ...

Wenn es von Entities, also Buch und Person mehrere gibt, dann müssen wir diesen eindeutige Nummern zuweisen. Dies sind hier die Inverntarnummer für das Buch und eine Mitgliedsnummer für die Person. Dieses zusätzliche Attribut hat einen einfachen Sinn: Man kann dann in der Datenbank mit wenig Platz, also nur zwei Spalten, einer Mitgliedsnummer eine Buchnummer zuordnen. Man erspart sich damit, dass man immer Name, Adresse, Anschrift, usw. in derselben Tabelle mit abspeichern muss. Wir korrigieren also die Attribute für das Buch:

- ISBN-Nummer
- Titel
- Die Autoren
- Verlag
- Erscheinungsjahr
- Inventarnummer

Und für die Person:

- Name
- Adresse
- Anschrift
- Mitgliedsnummer

Nun haben wir noch einen Vorgang (Entität): Das Ausleihen. Der Vorgang des Ausleihens ist an einen Zeitpunkt gebunden. Die Angabe des Datums ist hierfür sinnvoll. Wir ordnen also den beiden Relationen Attribute zu:

Ausleihen: Ausleihdatum, Rückgabedatum.

Wenn man sich nun das Diagramm nochmals anschaut, erkennt man, dass man fast alle Entities schon einzeichnen kann. Wir haben aber zwei Dinge noch nicht berücksichtigt: Zum einen fehlen uns die Wertigkeiten bei den Relationen der Entities untereinander, zum anderen ist die Relation zwischen Buch und Exemplar noch nicht klar. Bei Personen ist es eindeutig: Jeder Mensch ist einmalig, ein Buch nicht.

Die Entitäten müssen also noch in zwei Kategorien eingeordnet werden können:

1. Einmalige Dinge
2. Nicht einmalige Dinge

Danach untersuchen wir nun unsere Entitäten:

- Personen sind einmalig, von keinem Menschen gibt es zwei Exemplare
- Bücher sind nicht einmalig, es gibt eventuell viele Exemplare
- Das Ausleihen ist ebenfalls nicht einmalig, es kann beliebig oft geschehen

Uns interessieren im Moment jedoch nur die Dinge, die nicht einmalig sind. In unserem Beispiel sind das die Bücher. Wir müssen also noch eine weitere Relation und auch eine neue Entität (Exemplar) einführen, damit Bücher mit Exemplaren verknüpft werden können. Auf ein Buch können mehrere Exemplare kommen. Wir haben also hier eine neue 1:N Relation und eine neue Entität. Daher müssen wir die Attribute für die Entitäten Buch und Exemplar noch einmal neu überdenken:

Bisherige Attribute für Buch:

- ISBN-Nummer
- Titel
- Die Autoren

- Verlag
- Erscheinungsjahr
- Inventarnummer

Die Attribute für Buch und Exemplar müssen nach folgenden Kriterien neu vergeben werden:

1. Attribute, die ein Buch eindeutig beschreiben
2. Attribute, die ein Exemplar eindeutig festlegen
3. Gemeinsame Attribute, damit ein Exemplar einem Buch eindeutig zugeordnet werden kann

Daraus ergibt sich folgende Zuordung der Attribute für das Buch:

- ISBN-Nummer
- Titel
- Die Autoren
- Verlag
- Erscheinungsjahr

Und für die Entität Exemplar:

- Inventarnummer

Wir müssen noch die Wertigkeiten bei den Vorgängen zuordnen.

Der Vorgang des Ausleihens bezieht sich auch auf Personen und Exemplare, allerdings kann hier nur eine Person mehrere Exemplare ausleihen. Wir berücksichtigen hier nicht, dass es viele Personen gibt, die viele Bücher ausgeliehen haben können, weil wir jeden Vorgang des Ausleihens für sich allein betrachten, und nicht die Summe der Vorgänge.

Wir sind am Ende der Analyse und können nun die Fakten in ein Diagramm einzeichnen. Der Ort, an dem die Entitäten, also die Person, das Exemplar, und der eine Vorgang eingezeichnet werden, ist zwangsläufig anhand der Relationen gegeben:

Wir haben hier Entleiher und das Buchexemplar, zwischen denen der Vorgang Ausleihen abläuft.

Zwischen Exemplar und Buch gibt es eine eindeutige 1:N Relation, also zeichnen wir die Entität Buch über die Entität Exemplar und dazwischen die Relation.

Wir sind nun am Ende der Entwicklung eines ER - Diagramms. Fassen wir noch einmal die Systematik der Entwicklung zusammen:

1. Man sucht sich zuerst alle Entitäten zusammen, also alle Dinge, Lebewesen, Begriffe, Ereignisse und Vorgänge, also alle Handlungen
2. Es werden die Attribute zu den Entitäten zugeordnet
3. Es werden nun die Entitäten in zwei Kategorien eingeteilt: Einmalig und nicht einmalig, also mehrfach vorhanden
4. Allen einmaligen Entitäten wird ein neues Attribut, also Identifikationsnummer zugeordnet
5. Alle nicht einmalige Entitäten werden in zwei eigene Entitäten aufgespalten und zwar nach folgenden Kriterien:
   (a) Attribute, die die (Ursprungs-) Entität eindeutig beschreiben
   (b) Attribute, die ein Exemplar dieser (Ursprungs-) Entität eindeutig festlegen
   (c) Gemeinsame Attribute, damit ein Exemplar der (Ursprungs-) Entität eindeutig zugeordnet werden kann. Nicht einmalige Entitäten werden somit in zwei Entitäten aufgeteilt, die Entität Exemplar und die ursprüngliche Entität.
6. Zuordnung der Wertigkeiten der Vorgänge oder Ereignisse in Relation zu den Dingen, Lebewesen und Begriffen
7. Festlegung des Ortes beim Einzeichnen in das Diagramm, der sich immer zwangsläufig ergibt

Was kann man aus diesem Diagramm nun entnehmen?

Zuerst dient es dem Datenbankadministrator und den Entwicklern der Interfaces (PHP3) als Übersicht. Dem ER - Diagramm kann man direkt entnehmen, wie viele unabhängige Tabellen unser obiges Beispiel mindestens

benötigt und welche Attribute darin enthalten sind. Zusätzlich müssen alle Tabellen, die miteinander verknüpft werden müssen und können, mindestens ein gemeinsames Attribut enthalten. Man muss sich daher überlegen, welche Informationen für die SELECT-Statements aus den Tabellen entnommen werden müssen, und wie diese eventuell miteinander verknüpft werden können. Hier eine erste Aufspaltung der Daten in einzelne Tabellen:

1. Tabelle "Buch" mit der "Id", den Attributen ISBN, Titel, Autoren, Verlag, Erscheinungsjahr
2. Tabelle "Exemplar" mit der "Id vom Buch", dem Attribut Inventarnummer, (Entliehen)
3. Tabelle "Entleiher" mit den Attributen Name, Adresse, Mitgliedsnummer
4. Tabelle "Ausleihe" mit den Attributen Entleihdatum, Rückgabedatum, Mitgliedsnummer und Inventarnummer

Für eine Verknüpfung der Tabellen "Buch" und "Exemplar" reicht das gemeinsame Attribut, das von MySQL vergeben wird (eindeutiger Schlüssel). Für die Verknüpfung der Tabellen "Exemplar" und "Entleiher" über die Relation "Ausleihe" gibt es die Gemeinsamkeit Inventarnummer und Mitgliedsnummer. Beim Ausleihen ist egal, welches Exemplar eines Buches der Entleiher erhält. Man kann jedoch über eine weitere Tabellenverknüpfung ermitteln, welches Exemplar eines Buches gerade zurückgegeben worden ist, also im Archiv verfügbar ist. Nun stellt sich die Frage, ob es nicht sinnvoll wäre, ein Attribut "Entliehen" der "Tabelle" Exemplar hinzuzufügen. Man kann anhand der Tabelle Ausleihe stets ermitteln, welche Bücher ausgeliehen sind und welche wann spätestens zurückgegeben werden.

Mit all diesen Informationen aus der Erstellung des ER - Diagramms kann man schon recht zügig und treffsicher die Datenbank planen und aufstellen. Man muss bei größeren Datenbanken noch ein paar weitere Prozesse durchführen. Diese werden im Kapitel "Datenbankentwurf" und im letzten Kapitel "Vorschau" kurz behandelt.

Kapitel 4

PHP

4.1 Was ist PHP?

PHP ist eine in HTML eingebettete, serverseitige Skriptsprache und als solche eine Erweiterung für Internetserver, die es ermöglicht, mit verhältnismäßig wenig Aufwand dynamische Webseiten für Multimedia, E-Commerce, usw. im Internet zu entwickeln.

Serverseitig bedeutet in diesem Zusammenhang zunächst nicht mehr und nicht weniger, als dass der Code des Skripts - anders wie es zum Beispiel bei der häufig eingesetzten Skriptsprache Javascript der Fall ist - vom Webserver als das Resultat eines HTTP-Requests ausgeführt wird, das heißt also, durch den Aufruf einer Seite mit der HTTP-Methode GET oder beim Verarbeiten eines Formulars mit der HTTP-Methode POST.

Die Syntax von PHP ist stark an der Sprache C ausgerichtet, wobei auch Java- und Perl-Elemente Einfluss auf die Sprachdefinition hatten. PHP ist weitgehend unabhängig von der verwendeten Systemplattform und ist auf zahlreichen UNIX-Plattformen genau so einsatzfähig wie auf Microsoft-Betriebssystemen.

PHP ist eine junge Programmiersprache, die Anfang 1995 von Rasmus Lerdorf zunächst als PHP/FI speziell für die Realisierung von Webanwendungen geschaffen wurde.

PHP steht derzeit als PHP 3 in Form einer reinen Interpretersprache sowie als PHP 4 als Bytecode-Compiler zur Verfügung. PHP 4 verwendet intern die ZEND Scripting-Engine, die das Skript beim Aufruf kompiliert und auf diese Weise eine im Vergleich zu PHP 3 erheblich verbesserte Verarbeitungs-

geschwindigkeit ermöglicht.

Sowohl PHP 3 als auch PHP 4 können entweder als beliebig portables CGI-Programm oder als integriertes Modul für eine Reihe von HTTP-Servern, hierbei insbesondere auf dem Apache Webserver eingesetzt werden.

PHP stellt über 1.200 Funktionen für ganz unterschiedliche Anwendungen bereit, wobei kaum ein Bereich, der im Zusammenhang mit Webanwendungen genutzt werden kann, nicht abgedeckt ist.

So sind Funktionen zum Zugriff auf das Verzeichnis- und Dateisystem des Webservers genau so vorhanden wie Funktionen zur Verarbeitung von Zeichenketten, Arrays sowie regulären Ausdrücken. Das letztere dürfte für Umsteiger von der Sprache Perl sehr interessant sein.

Es gibt eine Reihe interessanter Funktionen wie z.B. für die Bearbeitung von Bildern, was spaeter auch nicht uninteressant für die Colibi sein könnte, falls man die Bilder von Büchern in die Datenbank aufnehmen wollte.

Mit die interessanteste Eigenschaft von PHP ist jedoch die einfache Anbindung der Sprache an zahlreiche SQL-Datenbanken. Fast alle gängigen Datenbanken werden unterstützt.

Insbesondere unterstützt PHP den Zugriff auf dem verbreiteten, für Web-Anwendungen wegen seiner hohen Verarbeitungsgeschwindigkeit und seinem geringen Speicherverbrauch besonders gut geeigneten Datenbankserver MySQL durch eine Vielzahl von Funktionen.

4.2 Wie arbeitet PHP?

Im Gegensatz zu Perl oder TCL, bei denen der Client (Browser) die Skripte direkt beim Server anfordert, wird der PHP-Code in die HTML-Seite eingebunden. Der Aufrufer einer derartigen Seite bekommt von diesem Code allerdings nichts zu sehen, da dieser bereits serverseitig interpretiert und in HTML-Code umgewandelt wird. Hierzu startet der Webserver den PHP-Interpreter, der das angeforderte Dokument übersetzt und den PHP Sourcecode der Seite ausführt. Die enthaltenen Befehle werden interpretiert und das Resultat findet seinen Platz als HTML-Ausgabe an Stelle des Sourcecodes im gleichen Dokument. Nach der Übersetzung wird die modifizierte Seite zum Client geschickt und dort durch den Browser dargestellt.

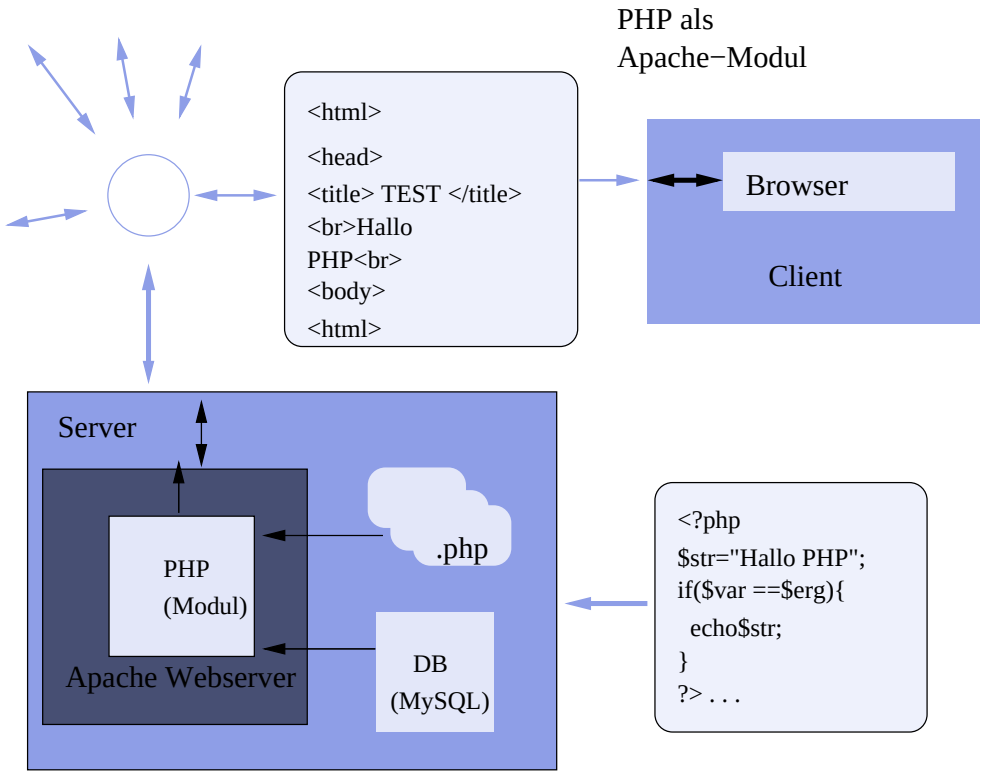

Abbildung 1.

Ob das vom Client aufgerufene Dokument PHP-Programmcode enthält, erkennt der Webserver an den von reinen HTML-Seiten abweichenden Dateiendungen. Welche Dateiendung für PHP-Seiten verwendet wird, kann in der Konfigurationsdatei des Webservers individuell festgelegt werden. Gängige Dateiendungen für PHP-Seiten sind z.B. php3 oder php. Die Webseite wird also auf diese Weise dynamisch, d.h. erst zum Zugriffszeitpunkt durch den Client erstellt und kann daher, in Abhängigkeit von einer Benutzer-Interaktion, noch vor dem Versenden an den Client modifiziert werden.

Aufgrund der für den Vorgang der serverseitigen Interpretation des PHP-Codes erforderlichen Zeit sind PHP-Seiten langsamer als statische HTML-Seiten.

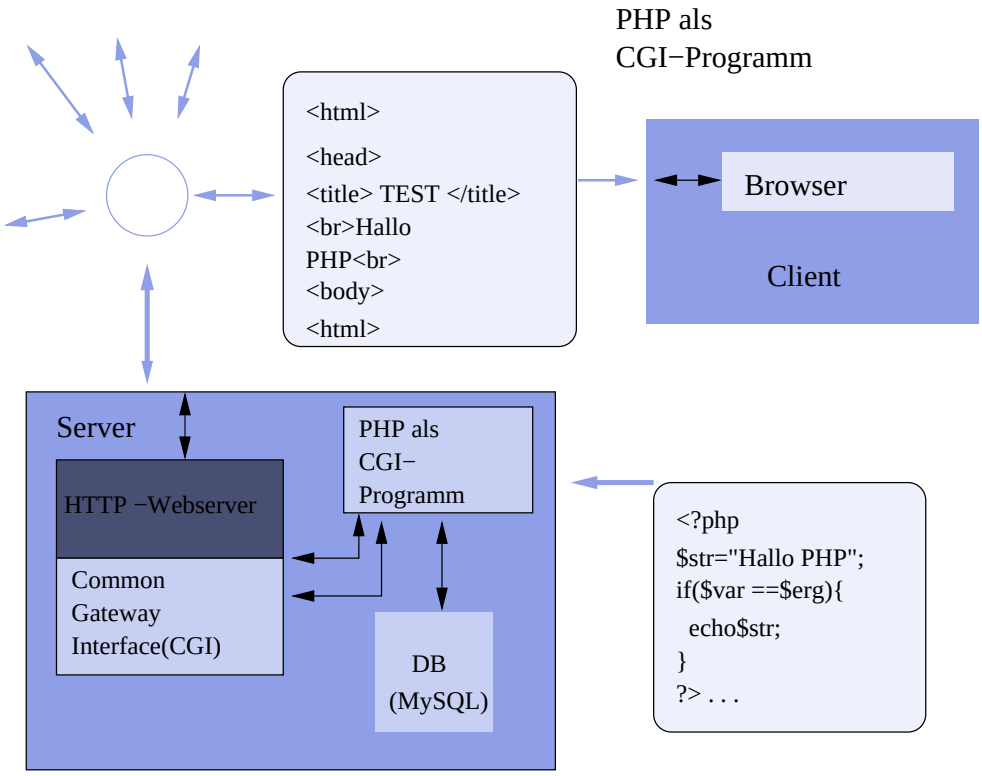

Abbildung 2

4.3 Sicherheit

Sofern die Verwendung von PHP mit Sicherheitsrisiken verbunden ist, liegt das Sicherheitsrisiko immer auf der Seite des Webservers. Z.B. werden JavaScript Programme ähnlich wie PHP-Anweisungen in den HTML-Code einer Seite eingebettet, jedoch im Unterschied zu PHP an den Client (Webbrowser) gesendet und von diesem ausgeführt.

Des weiteren, im Gegensatz zu PHP-Seiten, wo der Programmcode für den Aufrufer (auch im Quelltext-Modus des Browsers) unsichtbar bleibt, da dieser ja - bevor die HTML-Seite ausgeliefert wird - vom Webserver interpretiert, ausgeführt und entfernt wird, kann der JavaScript-Quellcode vom Seitenaufrufer im Webbrowser eingesehen werden.

Anweisungen mit PHP sind auch nicht möglich, im Gegensatz zu Ja-

vaScript. Auf dem Rechner des Clients kann man die Anwendungen weder starten, noch den Rechner abfragen oder ihn steuern, da Benutzer auf dem Client-Computer mit PHP genau so viel oder wenig erreichen können wie mit normalen HTML-Seiten.

Kapitel 5

MySQL

Meistens sind dynamische Internet-Seiten mit Datenbanken verknüpft. Daher spielt die Auswahl einer Datenbank eine wichtige Rolle. Der Aufbau der dynamischen Internet-Seiten hängt direkt von den Daten, die der Datenbankserver liefert, ab.

5.1 Warum gerade MySQL?

Wenn man sich nach einer freien oder preiswerten Datenbank umschaut, gibt es ein paar zur Auswahl, wie z.B. MySQL, mSQL, Postgres oder eine freie aber nicht unterstützte Engine von kommerziellen Verkäufern. Bei der Auswahl muss man beachten, was gerade für das Projekt wichtig ist: Leistung, Support, Zusätzliche Software Tools, strenge SQL-Übereinstimmung oder Erweiterbarkeit.

MySQL bietet viele attraktive Fähigkeiten. Eine der wichtigsten ist Geschwindigkeit. Laut der Internet-Seite[1] ist MySQL die schnellste oder eine der schnellsten verfügbaren Datenbanken.

Die MySQL-Datenbank ist relativ einfach zu benutzen, obwohl sie eine hohe Leistung bietet. Die Wartung ist erheblich einfacher als die von anderen grossen Systemen.

Alle modernen Datenbanksysteme bauen auf SQL (Structured Query Language) auf. SQL versteht auch MySQL. Auf MySQL kann von Applikationen aus, die ODBC (Open Database Connectivity) unterstützen, zugegriffen werden.

[1] http://www.mysql.com/benchmarks.html

Viele Clients können sich mit dem Server verbinden. Es können mehrere Datenbanken gleichzeitig benutzt werden. Man kann interaktiv auf den Server durch mehrere Interfaces zugreifen, was das Absetzen von Queries und das gleichzeitige Einsehen der Ergebnisse erlaubt:Kommandozeile, Browser oder X-Windows Klienten. Zusätzlich gibt es eine Reihe von Programmschnittstellen für C, PERL, Java, PHP und Python. Man kann entweder schon vorhandene Client-Interface Applikationen benutzen oder selbst eine schreiben.

MySQL ist Netzwerkfähig und auf die Daten kann von überall im Internet zugegriffen werden. Es gibt aber auch eine Zugriffskontrolle, damit nur dort zugegriffen wird, wo es erlaubt ist.

Es werden alle gängigen UNIX Derivate unterstützt, aber die Datenbank läuft auch auf nicht UNIX-Systemen, wie Windows. Egal ob PC oder High-End Server, eine Reihe von Hardware wird von MySQL unterstützt.

Kapitel 6

MySQL-Daten mit PHP ins Web bringen

Alle Daten der Colibi werden in der MySQL Datenbank abgelegt. Die Ausgabe der Daten erfolgt dann mit Netscape (GUI). Die Abfrage der MySQL Datenbank erledigen PHP/HTML Skripte.

Was Skriptsprachen im Web angeht, ist Perl nicht alles, obwohl die Sprache weit verbreitet ist. PHP ist eine Skriptsprache, die die Integration von DBMS-Daten in HTML-Dateien erleichtert. Auf vielen Internet-Seiten hat sich neben Perl auch PHP durchgesetzt.

6.1 PHP ganz einfach

Hinter dem Kürzel PHP steckt ein rekursives Akronym 'PHP: Hypertext Preprocessor' - und genau das leistet die Sprache, denn anders als Perl verarbeitet PHP Code, der in 'normales' HTML eingebettet wird. Das folgende Fragment einer HTML-Datei gibt das aktuelle Datum aus:

```
<p> Letzte Veränderung am
<?php
$today = date("d. m. Y");
echo "$today";
?>
</p>
```

Dabei sorgen d und m für die Ausgabe der Tage/Monate mit führender Null (andere Optionen sind in dem doc-Verzeichnis der PHP-Distribution zu finden).

Wie in Kapitel 2 schon erwähnt, handelt es sich bei PHP um eine serverseitig arbeitende Skriptsprache, die wie Perl von einem für sich stehenden Interpreter oder von einem Apache-Modul umgesetzt wird. Interpreter oder Modul erhalten die HTML-Datei mit PHP-Code als Eingabe und liefern reines HTML an den Browser. PHP erkennt 'seinen' Code an den Steuerzeichen <?php ...php>.

6.2 Verbindung zu Datenbanken

Wir schauen uns einen vereinfachten Teil der Colibi an, wo nach den Benutzern gesucht wird. Als ein eindeutiger Schlüssel wird die Email Adresse verwendet. Für die Benutzer gibt es eine Tabelle ("allusers"), die alle Daten der Benutzer der Colibi enthält (wie Adresse, Tel., usw.). Die Datenbank heisst "Bib", wo alle Tabellen der Colibi gespeichert sind. Für uns ist im Moment nur die Tabelle "allusers" interessant.

Um auf eine vorhandene Datenbank zuzugreifen, ist es wie in Perl notwendig, eine Verbindung aufzubauen, ein Select-Statement an ein Ergebnis zu binden und dies in HTML auszugeben. Das folgende Beispiel geht davon aus, dass mehrere PHP-Dateien das Eingangsformular als HTML darstellen und abhängig von den Benutzereingaben weiter vorgeht.

Zur Ausgabe des Formulars in Abbildung 3 kommt es, wenn die Seite aufgerufen wird. Es wird die Variable $action abgefragt (Listing 1). Wenn der Wert leer ist, ruft "case" die entsprechende Funktion auf , in diesem Fall "first_page()".

Der Wert ("Suchen") wird erst in dem Moment gesetzt, in dem ein Benutzer auf den Submit-Knopf klickt, der am Ende der Funktion "first_page()" von Listing 1 auf diesen Namen gesetzt ist:

<input type="submit" name="F" value ="Suchen">.

Dieser Ausdruck weist auch $F den Wert "Suchen" zu, was bewirkt, dass dasselbe Skript durch eine PHP-Variable " $PHP_SELF;" erneut aufgerufen wird. Es wird auch die Variable $action geändert. Das "case"-Statement ruft dann die Funktion "display_results()" auf . Die Variable $F ist in dem Fall

auch gesetzt und Bestandteil der Information, die per "POST" (als Methode) übertragen wird. Formular-werte sind über ihren Namen innerhalb von PHP wie andere Variablen ansprechbar. Ihnen muss man ein Dollarzeichen voranstellen. Im vorliegenden Fall existiert nur eine Variable: das erwähnte $F (für "find").

Die Bearbeitung der Eingabe erfolgt dann in der Funktion "display_results()". Der Code enthält die Anfrage an die Datenbank sowie die Ausgabe in HTML - auch in Listing 1 zu sehen. Für die Verbindung zu MySQL ist erforderlich (siehe Listing 2):

- Ein Aufruf der Funktion mysql_connect() mit den Parametern für Host, Benutzer und Passwort: mysql_connect($host,$user,$password);. Die Funktion bib_connect() wird in Listing 1 aufgerufen, ist selbst aber nicht dort definiert, sondern in einer "Include"-Datei, die in Listing 2 zu sehen ist. Die Datei aus Listing 2 ist der Datei aus Listing 1 bekannt, in dem man sie per "include" einbindet. Die Funktion wird nur ein mal aufgerufen; damit steht die Verbindung mit der MySQL-Datenbank bis die Session beendet wird (d.h. bis man den Internet Browser beendet). PHP verwendet einen so genannten "Link" für die weiteren Operationen, den die Funktion "bib_connect()" zurückgibt.

- Öffnen der gewünschten Datenbank durch mysql_select_db("bib"), siehe Listing 2.

- Formulieren der Anfrage durch $query="select [datenfelder] from [tabellen] where [bedingungen]";. Listing 1.

- Speichern des Ergebnisses in einer Variablen: $result = mysql_query($query);.

- Ausgabe der Daten in einer Schleife : while ($row = mysql_fetch_row($result)){...}.

Die Funktion "display_results" in Listing 1 zeigt ein ausführliches Select-Statement. Im Grunde handelt es sich um eine übliche SQL-Anfrage, die hier als ein String interpretiert wird und damit über ein paar Zeilen geschrieben wird. Einige Formulierungen in Listing 1 sehen recht umständlich aus, etwa:

> wird, ohne \. Ansonsten würde das Einführungszeichen als PHP-Zeichen Interpretiert (damit würde das Select-Statement hier enden), was man nicht haben will. Das Prozentzeichen (%) stellt einen Joker da, vergleichbar mit * unter UNIX-Systemen.

In Listing 1 sind weiter auch andere Funktionen zu sehen, wie html_begin ($titel, $titel, $color). Die Funktion druckt einfach HTML-Code aus. Das erleichtert die Arbeit mit HTML-Code, in dem man die Funktionen definiert, und die dann verwendet.

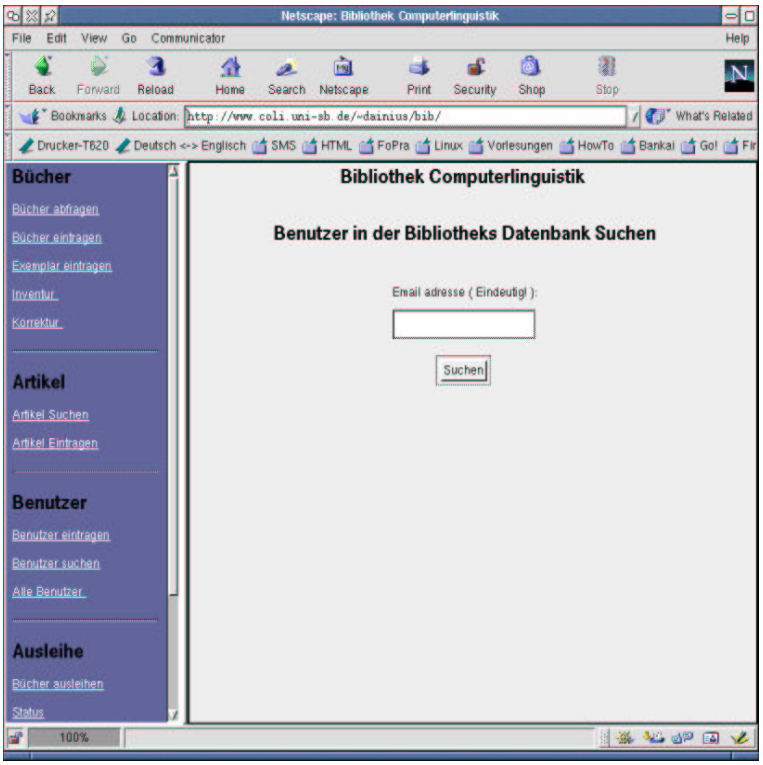

Abbildung 3

Kapitel 7

Colibi

7.1 GUI

Das Hauptmenü der grafischen Oberfläche ist in Abbildung 5 zu sehen:

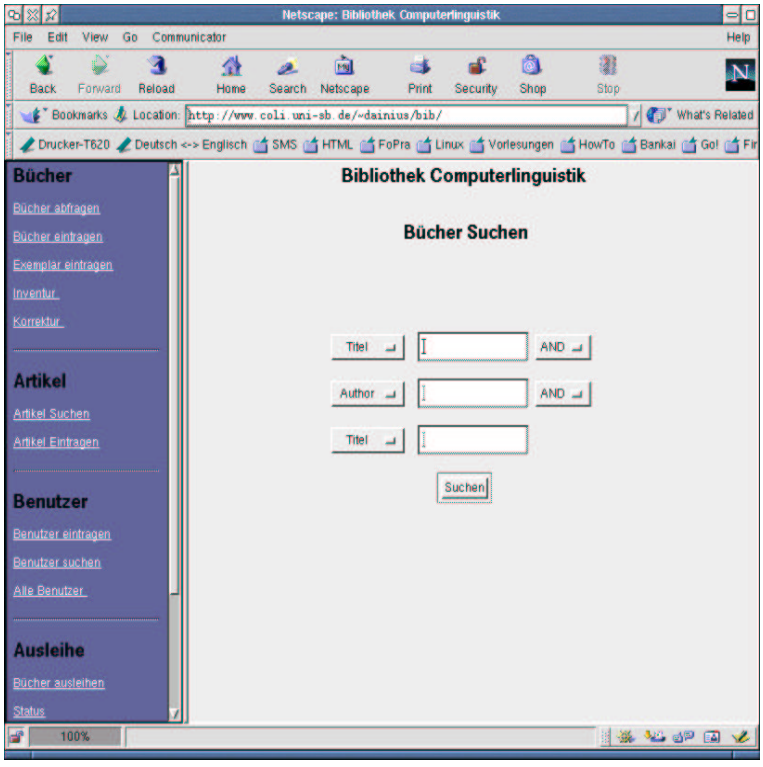

Abbildung 5: GUI der Colibi

7.1.1 Bücher

Bücher können nach Titel, Autor, Signatur, Editor und Standort gesucht werden. Die Abfrage kann mit logischen Verknüpfungen kombiniert werden.

Der Eintrag von Büchern erfolgt in einem Schritt (wenn nur ein Exemplar vorhanden ist) oder in zwei Schritten (wenn mehrere Exemplare einzutragen sind). Wenn nur ein Exemplar vorhanden ist, dann werden einfach die Daten eingegeben. Wenn es mehrere Exemplare gibt, dann sucht man nach der "Id" von dem Buch, die die MySQL-Datenbank vergeben hat und trägt diese zusammen mit Signatur und Standort in die Tabelle ein. Nach dem Eintragen wird die Signatur als eindeutiger Schlüssel in der Colibi verwendet; es gibt eine Warnung, falls eine zweite, gleiche Signatur eingegeben wird.

Die Inventur in der Colibi wird nach Bibliotheksstandorten durchgeführt. Unter dem Link auf der Seite kann man einen vorhandenen Standort eingeben; es werden alle Bücher ausgegeben, die unter dem Standort eingetragen sind.

Eine notwendige Anforderung fuer jede Datenbank, ist die Moeglichkeit, Daten korrigieren zu koennen. Nach der Eingabe der Signatur, werden alle Daten angezeigt und können korrigiert werden.

Bücher können auch gelöscht werden.

7.1.2 Artikel

In der Colibi werden auch wissenschaftliche Artikel von verschiedenen Autoren eingetragen. Es gibt Grundfunktionen wie Suchen, Eintragen, Korrigieren und Löschen.

Manche Artikel (z.B. alle Ausgaben von einem Jahr) werden zusammengebunden und zusätzlich als Buch eingetragen.

7.1.3 Benutzer

Die Daten von Benutzern (Entleiher) können eingetragen, gesucht, korrigiert und gelöscht werden. Als eindeutiger Schlüssel wird die Email-Adresse verwendet.

7.1.4 Ausleihe

Zum Ausleihen muessen sowohl die Email-Adresse, als auch die Dauer der Ausleihe und die Signatur des Buches eingetragen werden. Wenn das Buch zu der gegebenen Zeit nicht zurückgebracht wurde, wird eine Warn-Email an den Entleiher verschickt.

Im Status kann man alle ausgeliehenen Bücher sehen. Hier wird der Benuzer, der Verleihzeitraum und die Signatur des Buches angezeigt.

7.2 MySQL-Tabellen

7.2.1 Bücher-Tabellen

Bücher sind in zwei Tabellen unterteilt: "Books" und "Books_Exemplar". Die "Books"-Tabelle enthält folgende Daten: id, author, editor, title, type, year_a, keyword, series, corporation, publisher, publisher_location, bib, isbn. "Books" Exemplare: id, books_id, coli_id, bib_location, borrowed, typ. Die beide Tabellen sind über die "Id" verknüpft.

7.2.2 Artikel-Tabelle

Alle Artikel der Bibliothek sind in einer Tabelle erftasst.

7.2.3 Benutzer-Tabelle

Alle Entleiher (Benutzer) der Colibi sind in einer Tabelle erfasst.

7.2.4 Status-Tabelle

Die Ausgeliehene Buecher werden in einer Tabelle "status" eingetragen. Jeder Eintrag besteht aus der Email-Adresse des Benutzers, des Zeitraums von wann bis wann ein Buch ausgeliehen ist sowie der signatur des Buches.

7.3 Extras

7.3.1 Daten Export zu IBIS

Die Daten der Colibi werden zu täglich zum IBIS exportiert. Die MySQL Tabellen werden ausgelesen und in IBIS format umgewandelt.

7.3.2 Warnungs-Emails

Jeder Benutzer kann ein Buch bis zu einem bestimmten Datum behalten. Danach kriegt jeder Entleiher eine Warnungs-Email. Wenn das Buch nicht innerhalb von einer Woch zurück gebracht ist, dann wird eine Email an die Mitarbeiter geschickt, die dann weitere Massnahme unternehmen können.

Kapitel 8

Ausblick

8.1 Was kann man besser machen?

Auhoren von der Buch-Tabelle Trennen. Es gibt verschiedene Standarts, wieviel Autoren vom einem Buch in einer Bibliotheks-Datenbank eingetragen werden, manchmal bis zu 25, wie das bei Standart MAB2 ist.

Plannung in so einem Projekt ist sehr wichtig, weil wenn was später geändert wird, muss man was an MySQL, PERL-Skripten (die z.B. die Colibi MySQL-Tabellen auslesen und dann konvertierung in IBIS Format erledigen), sowie PHP anpassungen machen.

Sinnvoll wäre das MAB2[1] verwenden. Dann würde Datenaustausch zwischen mehreren Bibliotheken Problemlos funktionieren.

Sinvoll wäre über PHPLib[2] nachzudenken, falls die Bibliothek wachsen sollte oder ein ähnliches Projekt entstehen würde. PHPLib stellt eine Menge an Klassen zur verfügung, die für Web Applikationen wie Session-Verwaltung, Authentication und etc. benutzt werden.

Inventur funktioniert zwar für einzelne Colibi-Standorte (die Inventur in Colibi wird nach Standorten durchgeführt), erfordert aber noch ein wenig Anpassungen. Das Ziel ist mit einem Mausklick eine Inventur durchführen zu können.

[1] http://www.ddb.de
[2] http://phplib.netuse.de

8.2 Fehler die Vermieden werden sollte

Nur sinvolle Tabellen Spaltungen machen (Normalisierung). Zum einem wird zu viel zugriffe auf verschiedene Tabellen gemacht, zum anderen gibt es viel mehr Arbeitsaufwand, um die PHP-Skripte zu schreiben.

Kapitel 9

Anhang

9.1 Listing 1

<?

//
/*
users_look.php3 - Suche Benutzer
in Computerlinguistik

Autor: Dainius Ramanauskas

Beschreibung:
———
Benutzer werden in der datenbank nach der Email gesucht, spaeter kommt
noch suche nach Name und etc.

Last Modified : 24.12.00
———————-
See the file: bib_standard.inc,

Funktionen:

1. first_page() : - gibt die eingabe seite aus.
   - das gesuchte word wird an '$F' angehaengt.

2. display_results() : gibt die ergebnise raus

```
*/
//////////////////////////////////////////////////////


include ("/home/SG/dainius/php/bib_standard.inc");

define ( FIRST,0);
define ( SECOND,1);
//define ( DEL_BOOKS,2);

//////////////////////////////////////////////////////
//
// Anfang der Funktionen
//
//////////////////////////////////////////////////////

//
// Erste Seite wird aufgerufen
//

function first_page ()
{
  global $PHP_SELF;

  printf ("<FORM method=\"post\" ACTION=\"%s?action=%d\">\n",$PHP_SELF,
SECOND);

  print ("<br><center><h1>Benutzer in der Bibliotheks Datenbank Su-
chen</h1>
          </center>");
```

```
  printf ("<table align=center>");
  printf (" <tr><td align=center>  Email adresse ( Eindeutig! ):</td>
          <tr><td align=center><input type=text name=\"F\"");
  // F wird als globale variable definiert.

  printf ("value=\"\" size=20></td>");
 printf ("<tr><td align=center><input type=\"submit\" value=\"Suchen\"></td>\n");
  printf ("<BR>\n");
  printf ("</table>");

  printf ("</form>\n");

}

//
// Die Resultate werden ausgegeben!!!!!!!!
//

function display_results ()
{
  global $PHP_SELF;
  global $F;


  //-------------------------------------------------
  // -> 1 <- User werden nach der email gesucht
  //-------------------------------------------------


  //
  // hier wird zuerst die eindeutige id von user bestimmt
  //

  $query  = "select
                *
            from
                allusers
            where
```

```
                allusers.u_email
          like
                \"$F%\"";

$result = mysql_query($query)
   or die ("can't execute query , user_find.php : 1");

echo ("<caption><h1><center>Suchergebnis f&#252;r $F :
       </center></h1></caption>");

// Die zahl der treffer:
$menge = mysql_num_rows($result);

// Die zahl der treffer wird ausgegeben:
if ($menge > 0)
  printf ("Die Zahl der gefundenen User : %d\n", $menge);
else
 print ("<font color=red><h1><center>Leider keine treffer<p></font>");

print ("<center><h1> Die gesuchten Benutzer </h1>\n");

echo ("<table align=center border=1 width=70%>");


while ($row = mysql_fetch_row($result))
  {
    print ("<TR>\n");                          // ROWS!!
    for ($i=0; $i < mysql_num_fields($result); $i++)
      {
       printf("<TD> %s</TD>\n", htmlspecialchars ($row[$i]));
      }
    print("</TR>\n");

  }
mysql_free_result($result);

echo "</table>";
```

```
} // ende der funktion



//
// Ende der Funktionen
//

$titel = " Bibliothek Computerlinguistik ";
$color="black";
html_begin($titel,$titel,$color);
link_to_main_page ();

bib_connect()
     or die ("can't connect to server");

     if (empty($action))
     $action = FIRST;

     switch ($action)
{
 case FIRST:
   first_page ();
   break;
 case SECOND:
   display_results ();
   break;

 default:
   die("unknown action code ($action)");
}

link_to_main_page ();
html_end ();

?>
```

9.2 Listing 2

```
<?php
   ////////////////////////////////////////////////////////////////////////////
   //
   // bib_standard.inc : alle colibi funktionen
   //
   /*

    Letzte aenderung: 19.08.01:
    ---------------------
    funktion TextPrint zugefuegt. gibt den text in beliebige farbe aus.

   */
   ////////////////////////////////////////////////////////////////////////////


   function html_begin ($title, $header, $font_color)
   {
    if ($font_color)
      {
     print("<HTML>\n");
     print("<HEAD>\n");
     if ($title)
      print ("<font color=$font_color><TITLE>$title</TITLE></FONT>\n");

     print("</HEAD>\n");
    print("<style type=\"text/css\">
          body, li, td
          {
            font-family: Arial, Helvetica, Meta, sans-serif; color: #000000;
          }
          a
          {
            font-family: Arial, Helvetica, Meta, sans-serif; color: #C00000;
          }
          </style>
           ");
```

```
      print("<BODY bgcolor=#eeeeee>\n");
      if ($header)
          print ("<H2><center><font color=$font_color> $header </center></H2>\n");
       }
    else
      {
       print("<HTML>\n");
       print("<HEAD>\n");
       if ($title)
         print ("<TITLE>$title</TITLE>\n");

       print("</HEAD>\n");
       print("<BODY bgcolor=#eeeeee>\n");
       if ($header)
         print ("<H2><center> $header </center></H2>\n");
      }

    }

    function html_end ()
    {
      print("</BODY></HTML>\n");

    }



     // Connect to mysql server on head
     function bib_connect ()

     {

       $link = @mysql_pconnect ("134.96.68.11", "irgendwas", "irgendwas");
       if ($link && mysql_select_db("bib"))
         return ($link);
       return (FALSE);
```

```
 }

// Connect to mysql server on head
function bib_adm_connect ()
{
  $link = @mysql_pconnect ("134.96.68.11", "irgendwas", "irgendwas");
  if ($link && mysql_select_db("bib"))
    return ($link);
  return (FALSE);
}

// Hier wird der link zur der Haupseite gemacht
function link_to_main_page()
{

  // echo "<center><a href=\"index.html\">Hauptseite</a></center>";

}

// Hier ist der link zur der Hauptseite, wird nur bei der suche von
// books und articles verwendet, die nicht in bib verzeichnis stehen.
// also oeffentlich benutzt werden:
function link_to_main_for_books()
{

  echo "<center><a href=\"books_search.php3\">Hauptseite</a></center>";

}

// Hier ist der link zur der Hauptseite, wird nur bei der suche von
// books und articles verwendet, die nicht in bib verzeichnis stehen.
// also oeffentlich benutzt werden:
function link_to_main_for_articles()
{

  echo "<center><a href=\"articles_search.php3\">Hauptseite</a></center>";

}
```

```
// zeigt zelle
function display_cell ( $tag, $value, $encode )
{

 if ( $encode )
    $value = $value ;
 if ($value == "" )
    $value = "   ";
 print ("<$tag>$value</$tag>\n");

}
//-----------------------------------------------------
// -> 0 <- Funktion zum testen, ob die gleiche signatur schon vorhanden
ist
//          -> return 0 , wenn signatur bereits existiert
//          -> return -1, wenn keine
//
//          Wird bei der eintragung in tabellen "books, books_exemplar"
//          verwendet
//          Auch bei Korrektur wird die signatur ueberprueft.
//-----------------------------------------------------


function Check_Coli_Id($test)
{

   // der query
   $query = " select coli_id from books_exemplar ";

   // query wird an die db geschickt
   $result = mysql_query($query)
     or die ("can't execute query in Check_coli_id");

   // Bearbeitung der '$result'
   while ($row = mysql_fetch_row($result)){
    for ($i = 0; $i < mysql_num_fields($result); $i++){
```

```
      // wenn signatur ex, dann 'true'
      if ($row[$i] == $test)
        return 0;
    }
  }

  // keine gleiche signatur gefunden
  return -1;



}
//-------------------------------------------------
// -> 0 <- Funktion zum testen, ob das Buch schon ausgeliehen ist.
//         Pruefe die status tabelle.
//         -> return 0 , wenn Buch schon ausgeliehen ist
//         -> return -1, wenn nicht ausgeliehen
//
//         Wird in Ausleihe verwendet.
//
//-------------------------------------------------


function Check_If_Book_Bor($test)
{

  // der query
  $query = " select s_coli_id from status ";

  // query wird an die db geschickt
  $result = mysql_query($query)
    or die ("can't execute query in Check_If_Bor_Book");

  // Bearbeitung der '$result'
  while ($row = mysql_fetch_row($result)){
    for ($i = 0; $i < mysql_num_fields($result); $i++){

      // wenn signatur ex, dann 'true'
```

```
      if ($row[$i] == $test)
        return 0;
    }
  }

  // keine gleiche signatur gefunden
  return -1;



}


// bei der Inventur.php3 gibt es leerzeichen vorne. Es wird wegen der
// leerzeichen falsch sortier. Hier ist die funktion, die die ersten
// leerzeichen wegmacht.
function Move_First_Space_Out()
{

  // der query
  $query = " select coli_id from books_exemplar ";

  // query wird an die db geschickt
  $result = mysql_query($query)
    or die ("can't execute query in Check_coli_id");

  // Bearbeitung der '$result'
  while ($row = mysql_fetch_row($result)){
    for ($i = 0; $i < mysql_num_fields($result); $i++){

      // wenn signatur ex, dann 'true'
      if ($row[$i] == $test)
        return 0;
    }
  }

  // keine gleiche signatur gefunden
  return -1;
```

```
}

//-----------------------------------------
// Gibt den text farbig aus!
// 1. Argument: der text
// 2. Argument: dir farbe
//
//-----------------------------------------
function PrintText($text, $color)
{
  echo "<font size=18 color=$color><center>
          <h4>
            $text\n
          </h4>
        </font>";
}

//-----------------------------------------
//
// gibt den Titel vom Buch zurueck:
//  eingabe : signatur
//  ausgabe : titel
//-----------------------------------------
function getBookTitle ( $sig )
{
       $query = " select
                    title
                from
                    books, books_exemplar
                where
                    books_exemplar.coli_id = \"$sig\"
                AND
                    books_exemplar.books_id=books.id";

       $result = mysql_query($query)
          or die ("can't execute query in getBookTitle");

 while ($row = mysql_fetch_row($result))
```

```
  {
        return( $row[0] );
   }
}

//-------------------------------------
//
// gibt den Author vom Buch zurueck:
//  eingabe : signatur
//  ausgabe : author
//-------------------------------------
function getBookAuthor ( $sig )
{
       $query = " select
                    author
                from
                   books, books_exemplar
                where
                   books_exemplar.coli_id = \"$sig\"
                AND
                   books_exemplar.books_id=books.id";

       $result = mysql_query($query)
          or die ("can't execute query in getBookTitle");

 while ($row = mysql_fetch_row($result))
  {
       return( $row[0] );
   }
}

//=====================================================
//
// Definiere globale Variable, das datum von heute ausgibt
//
//=====================================================
 $Datum_Heute = date("Y-m-d");
```

```
    //=============================================================
    //
    // Eintrage von MySQL loeschen
    // eingabe :  tabelle, feld, id
    // ausgabe :  1 = true/  -1 = false
    //=============================================================


    function delMysqlEntry( $table, $field, $id ){

      $query = " delete from
                          $table
                  where
                          $field = $id ";
      // Uberpruefe, ob der query richtig ist:
      //$result = mysql_query($query)
       //    or die ("can't execute bib_standard.inc -> delMysqlEntry.php3
query");

      if (!($result = mysql_query($query)))
           {
            print("errno:" . mysql_errno());
            print("error:" . mysql_error());
               echo "can't execute bib_standard.inc -> delMysqlEntry.php3
query";
           }

      return $result;
    }


    ?>
```