INSTITUT FÜR INFORMATIK
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Systementwicklungsprojekt

Evaluation der Anbindung einer SQL-Datenbank an einen Apache Webserver

Bearbeiter:	Carmen Barth Sandra van Marwick
Aufgabensteller:	Prof. Dr. Heinz-Gerd Hegering
Betreuer:	Gerald Vogt Igor Radisic
Abgabetermin:	11. September 2001

Inhaltsverzeichnis

Kapitel 1

Einleitung

Der Lehrstuhl betreibt auf seinen Rechnern einen öffentlichen und einen internen Server, über die sämtliche Informationen des Lehrstuhls abrufbar sind. Alle Daten werden in Dateien im Textformat gespeichert und die dazugehörigen Rechte werden auf UNIX-Ebene verwaltet. Um unterschiedliche Zugriffsrechte der Server auf die Dateien zu erhalten, laufen diese demzufolge unter verschiedenen Kennungen. Damit Web-Seiten dynamisch generiert werden können, werden awk- und perl- Skripte über Server Side Includes (SSI) und über die CGI-Schnittstelle aufgerufen. Diese proprietäre Methode ist auf Dauer schwierig zu warten und sehr fehleranfällig.

Deshalb entstand die Idee, alle für den Lehrstuhl relevanten personen- und lehrspezifischen Daten in einer geeigneten Datenbank zu hinterlegen, diese an den Apache Web-Server anzubinden und die Informationen passend aufzubereiten. Die Daten werden schließlich je nach Art der Öffentlichkeit über das Web lesend zugänglich gemacht bzw. nur auf den internen Seiten angezeigt. Bei letzterem muss es eine sichere Kommunikation zwischen Browser und Web-Server geben. Zusätzlich bekommen verschiedene Benutzer unterschiedliche Zugriffsrechte auf die Datenbank.

An dieser Stelle sei auch auf ein weiteres Projekt „Implementierung eines datenbankgestützten Workflow-Systems mit Web-Interface“ verwiesen, das auf derselben Datenbank arbeiten soll.

Das gesamte System soll möglichst kostengünstig, flexibel erweiterbar und leicht zu warten sein. Im Rahmen dieses Projektes werden verschiedene freie Datenbanken untersucht, inwieweit sie für den Einsatz am Lehrstuhl in Frage kommen. Für alle geeigneten Datenbanken werden verschiedene Anbindungsmöglichkeiten ans Web evaluiert. Hierbei spielt vor allem die sichere Anmeldung an der Datenbank eine entscheidende Rolle. Eine Auswahl an Datenbanken mit Anbindung wird in einer Installation bereitgestellt.

Ziel des Projektes ist es, darauf ein System zur Verwaltung von Daten, hauptsächlich Terminen, Diplomarbeiten und Fopras, Oberseminaren und Prüfungen zu implementieren und deren Anzeige auf den entsprechenden Seiten der TU und LMU sowie auf den internen Seiten des Lehrstuhls zu realisieren.

Kapitel 2

Anforderungen

2.1 Verwaltungsseiten

Die Datenbank soll alle Informationen, die der Lehrstuhl zur Erstellung der öffentlichen und internen Seiten benötigt, erfassen können. Hierbei handelt es sich sowohl um personenspezifische wie auch um lehrspezifische Daten. Es folgt eine Liste mit zu realisierenden Punkten:

- Allgemeine Terminverwaltung:
  Die Termindatenbank umfasst sowohl öffentlich zugängliche Daten wie Termine für die LMU und/oder TU als auch interne, nur den Mitarbeitern zugängliche. Für diese allgemeinen Termine soll es möglich sein, Zeiträume, Wiederholungen und Anzeigezeiträume anzugeben.

- Verwaltung von Prüfungen:
  Es handelt sich hierbei um die Prüfungsverwaltung für Vor - und Hauptdiplomsprüfungen. Neben der Auswahl von Prüfling und Prüfer soll es eine bequeme Möglichkeit geben, den Prüfungen Beisitzer zuzuordnen.

- Verwaltung von Diplomarbeiten und Fopras:
  Über die internen Verwaltungsseiten können die Arbeiten neu in die Datenbank eingetragen, geändert, vergeben und verlängert werden.

- Verwaltung der Oberseminarseite:
  Es gibt normale Oberseminare, bei denen Studenten ihr abgeschlossenes Fopra präsentieren oder ihr Diplomarbeitsthema vorstellen. Außerdem gibt es noch Doktorrandenkolloquien, bei denen externe Referenten Vorträge halten.

Abgelaufene Termine, die keinen Wiederholungen unterliegen, vergangene Prüfungstermine sowie überholte Ausschreibungen werden nach einer vorzugebenden Zeit durch ein Skript aus der Datenbank entfernt.

2.2 Anzeigeseiten

Allgemeine Termine, aktuelle Ausschreibungen, laufende und abgeschlossene Arbeiten, sowie Oberseminarankündigungen sollen aus der Datenbank übers Web angezeigt werden. Die Anzeige soll, wie bereits vorhanden, parametrisiert werden, sodass die jeweiligen Anzeigen der beiden Universitäten LMU und TU relativ einfach erstellt werden können.

2.3 Sichere Kommunikation

Alle Daten, die nicht öffentlich zugänglich sein sollen, werden über das HTTPS-Protokoll verschlüsselt übertragen. Um auf solche Daten zuzugreifen, erhält jeder Mitarbeiter des Lehrstuhls sowie Befugte ein sog. Client-Zertifikat.

Zur Anmeldung an der Datenbank bei schreibendem Zugriff oder bei bestimmten Lesezugriffen werden gewisse Informationen des persönlichen Client-Zertifikats benötigt. Die Realisierung der Anmeldung wird in Abschnitt 5.4.2 beschrieben.

Kapitel 3

Evaluierung

3.1 Vorauswahl der Datenbank

Für dieses Projekt scheiden alle kostenpflichtigen Datenbanken aus. Es werden daher die drei verfügbaren freien Datenbanken Interbase, PostgreSQL und MySQL nach folgenden Kriterien in Absprache mit dem Lehrstuhl untereinander verglichen:

- Trigger:
  Trigger sind Definitionen einer oder mehrerer Folgeaktionen, die implizit nach Ausführung einer INSERT-, UPDATE- oder DELETE-Anweisung gestartet werden.
- referentielle Integrität:
  Besitzt eine Tabelle einen Fremdschlüssel, der auf einen Primärschlüssel einer referenzierten Tabelle zeigt, muß jeder Wert im Fremdschlüssel einen enstprechenden Eintrag in der referenzierten Tabelle besitzen. Dann besteht referentielle Integrität.[8]
- Transaktionen:
  Abgeschlossene Datenbankaktion, die sich aus mehreren Einzelaktionen zusammensetzen kann.
- Support
- Dokumentation
- ODBC:
  Open Database Connection, eine von Microsoft als Standard etablierte Teilmenge von SQL.
- Subqueries:
  Unterabfragen
- Backup

	Interbase	PostgreSQL	MySQL
Trigger	✓	✓	X
ref. Integrität	✓	✓	X
Transaktionen	✓	✓	X
Support	Newsgroups, Mailinglisten, FAQs	Mailinglisten, FAQs	Mailinglisten
Dokumentation	✓	✓	✓
ODBC	✓	✓	✓
Subqueries	✓	✓	X
Backup-Möglichkeit	✓	SQL-Dump oder Server herunterfahren	SQL-Dump oder Server herunterfahren
Views	✓	✓	X

Tabelle 3.1: Vergleich der Datenbanken

- Views:
  Eine View ist eine spezielle Ansicht von Tabellen, in der selbst keine Daten gespeichert werden. Die Werte werden durch eine Abfrage aus einer oder mehreren Tabellen bezogen.

Tabelle 3.1 ist zu entnehmen, dass Interbase [3] über alle aufgezählten Kriterien verfügt. Da diese Datenbank jedoch nicht auf dem Betriebssystem Solaris für Intel, das auf dem Lehrstuhl läuft, installiert werden konnte, stand sie nicht weiter zur Diskussion. Interbase sei aber an dieser Stelle, für den Fall, dass sich das Betriebssystem ändern sollte, trotzdem erwähnt.

MySQL [5] beherrscht im Gegensatz zu PostgreSQL [7] keine Transaktionen. Um zu verhindern, dass während der Ausführung einer zusammengehörigen Sequenz von Anweisungen andere Benutzer parallel auf die betroffenen Tabellen zugreifen, gibt es die Möglichkeit, einzelne Tabellen zu sperren. Tritt allerdings zwischen den Anweisungen ein Absturz auf, so wird nicht, wie bei PostgreSQL, die ursprüngliche Situation wiederhergestellt. Diese muss dann mittels der Logdatei rekonstruiert werden.

MySQL unterstützt keine referentielle Integrität. Diese ist allerdings im Rahmen des Projektes nicht unbedingt erforderlich, da die SQL-Anweisungen zur Sicherstellung der Konsistenz der Daten in die Skripten aufgenommen werden können. Besonders aufgrund der fehlenden referentiellen Integrität ist MySQL schneller als PostgreSQL.

MySQL kennt jedoch keine Views. Da verschiedene Benutzergruppen mit unterschiedlichen Rechten übers Web auf die Datenbank zugreifen, sind diese zur Ausblendung gewisser Informationen sehr wichtig.

3.2 Anbindung

Für die Anbindung einer Datenbank an den Web-Server kommen drei kostenlose Möglichkeiten in die engere Wahl. Diese werden im folgenden zunächst erläutert und anschliessend untereinander verglichen.

3.2.1 CGI (Common Gateway Interface)

Die vom Anwender im Browser eingegebenen Daten werden an den Web-Server übermittelt. Dieser startet ein auf dem Server laufendes CGI-Programm, wobei die Übergabe der Eingabedaten mittels Umgebungsvariablen geschieht. Während HTTP die Kommunikation zwischen Browser und Web-Server regelt, stellt CGI die Schnittstelle zwischen Web-Server und CGI-Programm dar. Das CGI-Programm wertet nun die Eingabedaten aus und liefert in der Regel eine HTML-Seite zurück, die vom Web-Server an den Client geschickt wird. Abbildung 3.1 zeigt die Kommunikation bezogen auf eine Datenbank-Anwendung.

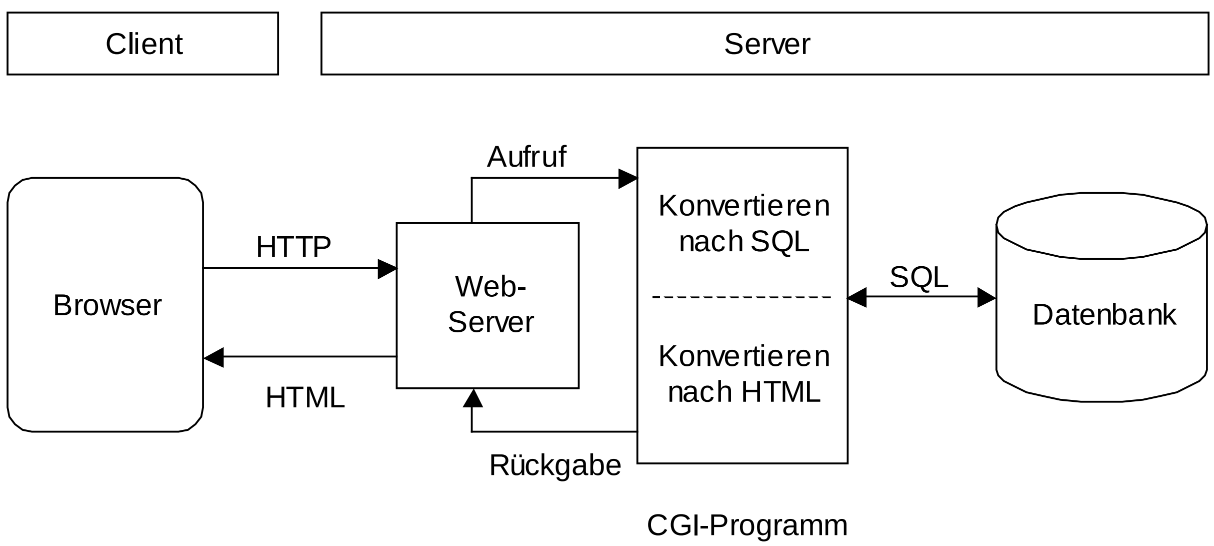

Abbildung 3.1: Kommunikation mittels CGI

CGI-Skripte sind selbständig ablaufende Programme, die auf Anfragen von Benutzern reagieren. Sie realisieren beispielsweise eine Datenbankabfrage. Zur Programmierung können neben Shell-Skripten auch beliebige höhere Sprachen wie beispielsweise C/C++, Java oder Pascal benutzt werden. Am häufigsten ist jedoch die Interpreter-Sprache Perl verbreitet. Perl stellt gute Textmanipulationsfunktionen zur Verfügung und bietet die Möglichkeit, kurze, effiziente Skripten zu schreiben. [1,2]

3.2.2 PHP (PHP Hypertext Preprocessor)

Der Nutzer fordert mit dem Browser ein PHP-Skript beim Web-Server an. Dieser leitet die Anfrage aufgrund der Dateierweiterung .php an das PHP-Modul des Apaches weiter.

Das Modul interpretiert das Skript. Darin enthaltene Datenbankbefehle werden an die Datenbank weitergeleitet. Die von der Datenbank zurückgelieferten Ergebnisse werden anschließend in die Seite als HTML-Code eingebaut. Der Web-Server sendet den HTML-Code an den Client, der aus dem HTML-Code schließlich eine sichtbare Seite generiert. Abbildung 3.2 verdeutlicht dieses Prinzip. [6]

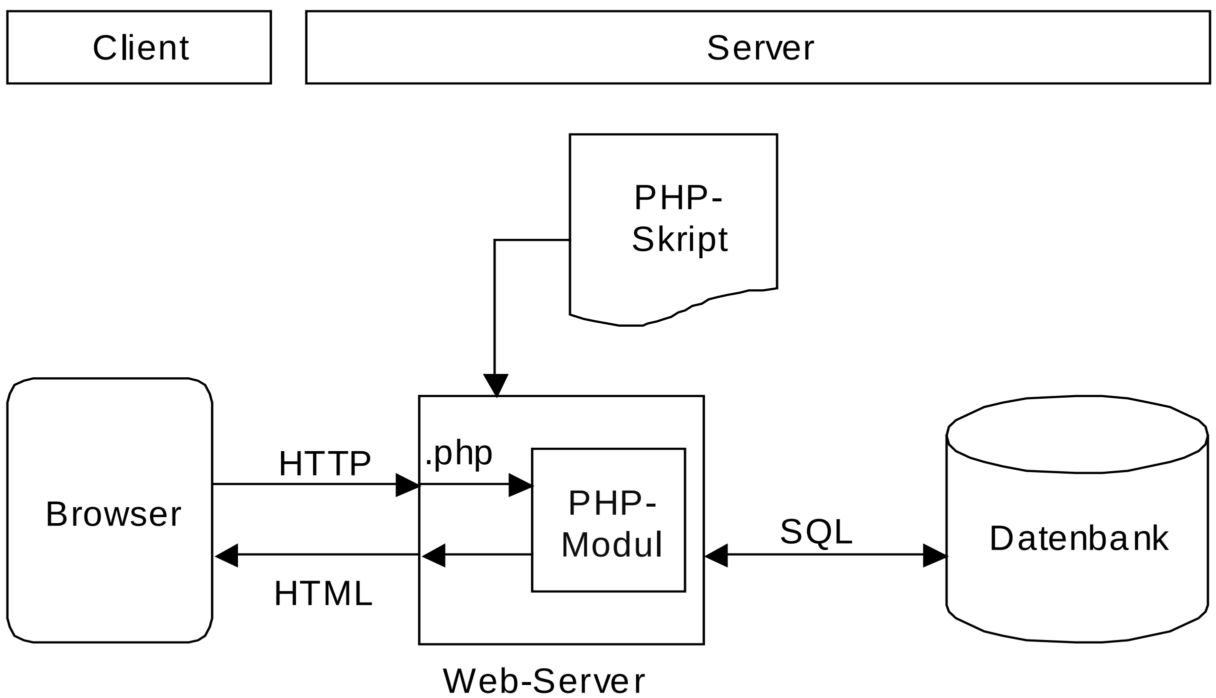

Abbildung 3.2: Kommunikation mittels PHP

3.2.3 JSP (Java Server Pages)

Der Browser schickt einen HTTP-Request an den Server. Der Web-Server gibt die Anfrage der JSP-Seite (aufgrund der Endung .jsp) direkt an die JSP-Engine weiter. Diese besteht im wesentlichen aus zwei Komponenten: dem JSP-Compiler und dem Servlet-Container.

Der JSP-Compiler übersetzt noch nicht compilierte, angefragte JSP-Seiten in Servlets. Servlets sind spezielle Java-Klassen, die HTTP-Requests verarbeiten und HTML-Code generieren. Für die Instantiierung und Ausführung der Servlets ist der Servlet-Container zuständig. Es wird zuerst geprüft, ob das angefragte Servlet bereits läuft und gegebenenfalls gestartet. Das Ergebnis der Servlets (HTML-Code) wird an den Container, die Engine und den Web-Server zurückgegeben, der die HTML-Seite an den Client sendet. Abbildung 3.3 zeigt die Kommunikation der verschiedenen Komponenten.

Eine JSP-Seite ist folgendermassen aufgebaut:
Neben statischem HTML-Code, der das Layout der Seite bestimmt, sind in die Seite JSP-Anweisungen (Tags) eingestreut. Innerhalb dieser Anweisungen können Aufrufe von JavaBeans enthalten sein. [4]

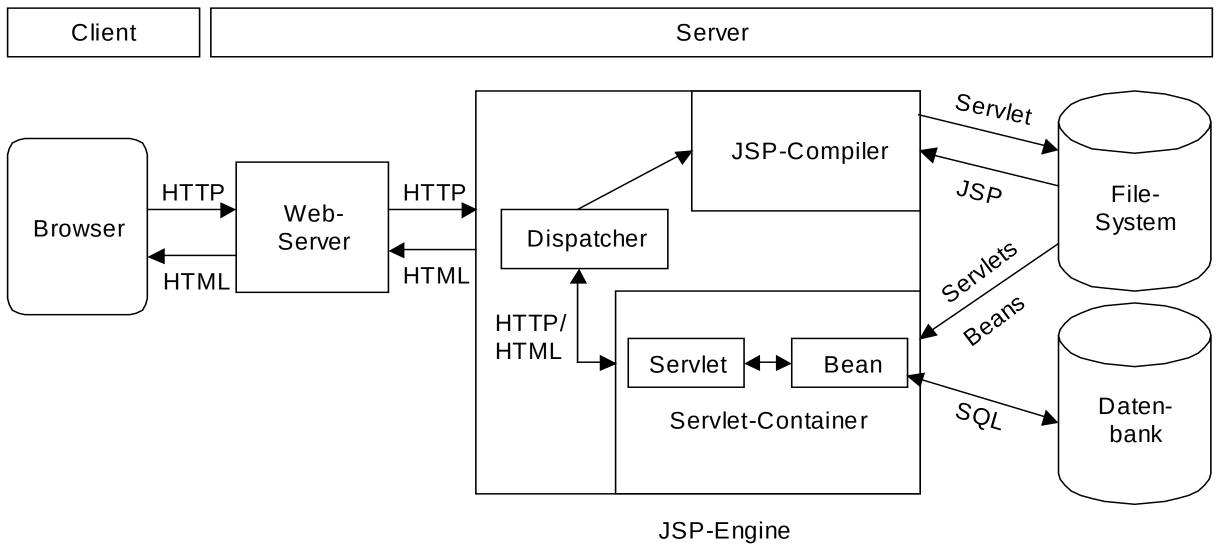

Abbildung 3.3: Kommunikation mittels JSP

3.2.4 Vergleich

Für kleine Anwendungen ist CGI der einfachste Weg und wird durch die standardisierte Schnittstelle von jedem Web-Server und jeder Plattform unterstützt. Große Anwendungen lassen sich mit CGI jedoch nicht effizient realisieren, da es zu Performance-Problemen kommen kann. Für jede Anfrage muss ein neuer Prozess gestartet werden, weshalb diese Technik viele Ressourcen verbraucht und langsam ist.

Im Gegensatz dazu wird bei PHP, wenn es als Apache-Modul verwendet wird, nicht jedesmal ein eigener Prozess gestartet. Diese Lösung ist daher schneller als CGI. Weitere Vorteile sind die geringe Einarbeitungszeit und die gute Datenbankunterstützung für eine Reihe von Datenbanken. Als Nachteile können das Fehlen von nicht-requestgetriebenen Hintergrundprozessen und einem komfortablen Debugger gesehen werden.

Ein Vorteil der Java Server Pages liegt in der Trennung von Repräsentation und Information, d.h., das Layout einer Seite kann getrennt von der eigentlichen Datenverarbeitung (z.B. Durchführung einer Datenbankabfrage) programmiert werden. Ausserdem steht der volle objektorientierte Sprachumfang samt Bibliotheken von Java zur Verfügung. Einen Nachteil stellt jedoch der hohe Ressourcenverbrauch dar, da der Servlet-Container zur Ausführung der Servlets die „Virtual Machine“ benötigt.

Ein Kriterium zur Auswahl der Anbindung für dieses Projekt ist die möglichst geringe Antwortzeit auf die vielen Anfragen an den Lehrstuhl. CGI schneidet hierbei am schlechtesten ab, weshalb es nicht weiter in Frage kommt. PHP stellt die schnellste Technologie dar und bietet ausserdem einen guten, für das Projekt ausreichenden Funktionsumfang. Bei sauberer Programmierung lassen sich auch hier Repräsentation und Information weitgehend voneinander trennen. Aus diesen Gründen fiel die Wahl auf PHP. [1,2,3,4]

Kapitel 4

Systemkomponenten

4.1 Apache 1.3.14

Zusätzlich zu den in der Standardkonfiguration angegebenen Modulen werden folgende zwei Module verwendet: [0]

- mod_ssl
  Dieses Modul stellt die Schnittstelle zwischen OpenSSL und Apache dar. Bei OpenSSL handelt es sich um eine SSL- und Kryptographiebibliothek. Damit wird die Verschlüsselung der Daten gewährleistet. Bevor mod_ssl verwendet werden kann, benötigt der Server zur Authentisierung gegeneinüber den Web-Clients ein Zertifikat. Hierbei handelt es sich um sog. X.509-Zertifikate.

- mod_php4
  Dieses Modul stellt den PHP-Interpreter dar, der als Bestandteil des Apaches läuft. Folgende Zeile der Apache-Konfigurationsdatei bewirkt, dass allen Anfragen, die auf .php enden, der MIME-Typ *application/x-httpd-php* zugewiesen wird.

  ```
  AddType application/x-httpd-php .php
  ```

  Anfragen diesen Typs werden vom Apache zur Interpretation durch den PHP-Interpreter (php-Modul) weitergegeben.

4.2 Aufruf von PHP-Skripten

Ein PHP-Skript kann entweder durch das direkte Eintippen einer URL, oder durch das Abschicken von Formularen aufgerufen werden. Ein Formular ist eine Eingabemaske, die verschiedene Elemente wie beispielsweise Textfelder, Listenfelder und Buttons enthält. Sie werden mittels HTML-Code erzeugt. Abbildung 4.1 zeigt ein Formular mit einem Eingabefeld und einem Button zum Absenden des Formulars.

Abbildung 4.1: Ein Beispielformular.

```
<HTML>
<HEAD>
<TITLE>Testformular</TITLE>
</HEAD>
<BODY>
<H3>Ein kleines Formular</H3>
<FORM ACTION="zielformular.php" METHOD="POST">
<TABLE BORDER="0" CELLSPACING="10">
<TR>
<TD>Textfeld:</TD>
<TD><INPUT TYPE="text" NAME="textfeld" SIZE="8" VALUE="<? echo $textfeld ?>"><TD>
</TR>
<TR>
<TD><INPUT TYPE="submit" NAME="ok" VALUE="OK"></TD>
<TD><INPUT TYPE="hidden" NAME="variablenname" VALUE="variablenwert"></TD>
</TR>
</TABLE>
</FORM>
</BODY>
</HTML>
```

Über das Eingabefeld kann der Benutzer einen Text eingeben. Dem Quellcode ist zu entnehmen, dass der Text als Wert in der Variablen *textfeld* abgelegt wird. Das Formular enthält noch eine weitere Variable *variablenname*, eine sog. *hidden*-Variable. Hidden-Variablen definieren versteckte Daten in einem Formular, die dem Anwender nicht gezeigt werden. In Beispielformular ist der Wert der hidden-Variable *variablenwert*.

Beim Absenden des Formulars werden alle Variableninhalte an das aufgerufene Skript mitübertragen. Im gezeigten Beispiel heißt das aufzurufende Skript *zielformular.php*. Das Schlüsselwort *POST* bestimmt die Methode, wie die Formulardaten vom Client zum Server übertragen werden.

4.3 PostgreSQL-7.0.3 und MySQL-3.22.32

PostgreSQL und MySQL gehören zur Gruppe der relationalen Datenbanken. Für Details sei auf die Literatur [7] und [5] verwiesen.

An dieser Stelle sei jedoch eine Besonderheit erwähnt, die im Projekt häufig zum Einsatz kommt und in den beiden Datenbanken jeweils in unterschiedlicher Form vorhanden ist. Es handelt sich um das automatische Generieren von Primärschlüsseln.

Dazu exisitert bei PostgreSQL das Schlüsselwort SERIAL, das das Anlegen einer einzeiligen Tabelle, einer sogenannten *SEQUENCE*, bewirkt. So können bequem künstliche Schlüssel erzeugt werden.

Bei MySQL heisst das Schlüsselwort zur automatischen Generierung von Primärschlüsseln AUTOINCREMENT.

Kapitel 5

Implementierung

5.1 Datenbankschema

Die Beziehungen und Attribute der Tabellen aus der Datenbank sind den ER-Diagrammen im Anhang zu entnehmen. Im Folgenden wird auf einige Attribute, die sich nicht von selbst erklären, eingegangen.

Alle Primary-Keys, die mit dem Schlüsselwort SERIAL deklariert wurden, erhalten einen, durch die Datenbank automatisch generierten, Integer-Wert.

Tabelle Termine

- anzeigezeitraum, DATE: ein Termin erscheint ab diesem Datum im Netz
- anzeigeseite, INTEGER: Die Anzeigeseite wird durch entsprechende Bitoperation dargestellt: 0=INTERN (weder LMU noch TU), 1=LMU, 2=TU, 3=COMMON (LMU und/oder TU)
- url, VARCHAR: eine URL wird entweder relativ (zum Server) oder absolut angegeben
- wiederholungszeitraum, INTEGER: Es kann jeder beliebige Zeitraum in Tagen, Wochen, Monaten oder Jahren gewählt werden. In der Datenbank ist in der Spalte wiederholungszeitraum ein 32-Bit-Integer angegeben, wobei die oberen 16 Bit für die Zeiteinheit (Tage, Wochen, Monate, Jahre) und die unteren 16 Bit für die entsprechende Zahl reserviert sind.

Tabelle Oberseminartermine

- titel, BOOL: 0=Doktorandenkolloquium, 1=normales Oberseminar
- mitarbeiternummer, INTEGER: speichert den für das OS verantwortlichen Mitarbeiter

- veroeffentlichen, BOOL: 1=die Oberseminarankündigung soll ab jetzt im Netz erscheinen

Tabelle Diplomarbeit/Fopra

- anzeigeseite, INT: wie in Tabelle Termine

Tabelle Mitarbeiter

- typ, INTEGER: 1=Professor, 2=interner Mitarbeiter, 4=externer Mitarbeiter

Tabelle Universitaet

- universitaetsnummer, INTEGER: 1=LMU 2=TU

5.2 Struktur

5.2.1 Verzeichnisstruktur

In Abbildung 5.1 sind die für das Projekt wichtigsten Verzeichnisse dargestellt.

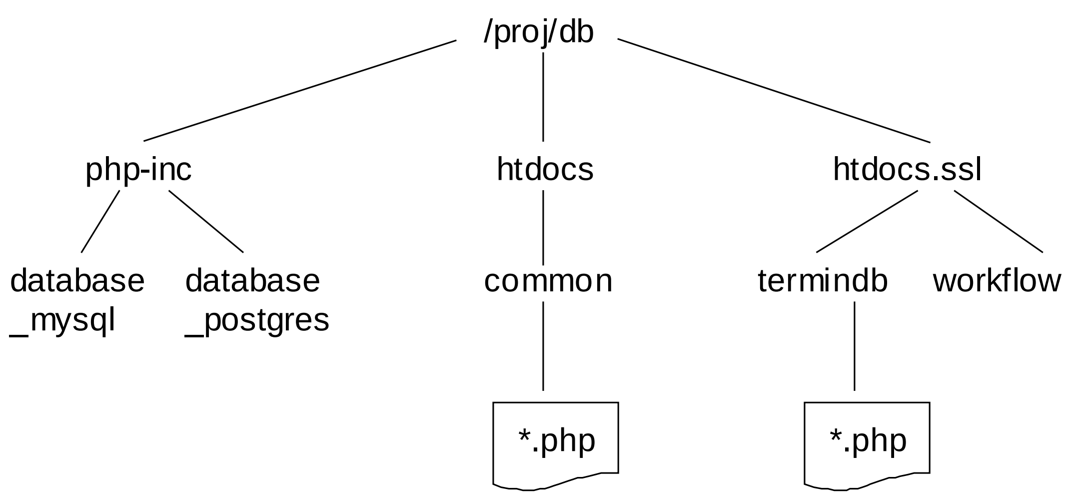

Abbildung 5.1: Verzeichnisse

Der Apache kann nur auf die Verzeichnisse htdocs und htdocs.ssl zugreifen. Diese enthalten folgende Unterverzeichnisse:

- htodcs
  Neben den beiden unispezifischen Verzeichnissen *sunheger5* und *wwwtest* (LMU-Server bzw. TU-Server) extistiert ein Ordner *common*, in dem Skripten für die

öffentlichen Seiten beider Universitäten abgelegt werden. Die Inhalte des *common*-Verzeichnisses können entweder explizit oder implizit verwendet werden, d.h., befindet sich für eine URL eine Datei bzw. ein Verzeichnis nicht im uni-abhängigen Teil (*sunheger5* bzw. *wwwtest*), dafür aber in *common*, dann wird die Datei bzw. das Verzeichnis aus *common* verwendet.

- htdocs.ssl
  Im Ordner *termindb* befinden sich alle PHP-Skripten, die zur Erstellung der internen Verwaltungsseiten dienen.

Im Verzeichnis *php-inc* befinden sich PHP-Skripten, die allgemeine Funktionen enthalten, die in den Skripten zur Erstellung der Verwaltungsseiten und der Anzeige im Web benötigt werden. Hier sind auch die Header-Dateien für TU und LMU sowie die Datenbankfunktionen enthalten.

Damit sowohl die Datenbank PostgreSQL als auch MySQL verwendet und getestet werden kann, werden die datenbankspezifischen Funktionen innerhalb einer Bibliothek zur Verfügung gestellt. Die Auswahl der Datenbank erfolgt über eine Variable (*$db*), sodass je nach Variablenwert (*my* oder *pg*) die spezifischen Funktionen für MySQL bzw. PostgreSQL aufgerufen werden.

5.2.2 Aktualisierungsskript

Die Datenbank soll in regelmäßigen Abständen mittels eines PHP-Skripts aktualisiert werden. Dazu existiert auf der internen Anzeigeseite ein Link, der das sogenannte Aktualisierungssktipt startet. Dieses nimmt folgende Änderungen vor:

- Löschen aus der Datenbank

  Aus den Tabellen *termine* und *pruefungen* werden vergangene Termine bzw. Prüfungen gelöscht.

  Aus den Tabellen *diplomarbeit* und *fopra* werden alle nicht vergebenen Arbeiten gelöscht, die auf den internen Seiten bereits nicht mehr in roter Farbe angezeigt werden. D.h., die Arbeit hing schon seit 90 Tagen im Web und erschien danach für 15 Tage in Rot auf den internen Seiten. Wurde die Arbeit in dieser Zeit nicht verlängert, wird sie durch das Aktualisierungsskript gelöscht.

- Anpassen des Wiederholungszeitraums
  In der Tabelle *termine* besteht die Möglichkeit, einen Wiederholungstermin einzutragen. Dies bedeutet, dass der Termin nach dem angegebenen Zeitraum wieder im Netz erscheint. Das Skript wiederholung.php realisiert diese Wiederholung, indem es das Datum in der Datenbank um den in der Spalte *wiederholungszeitraum* angegebenen Wert erhöht.

5.3 Module

Zum Eintragen der Daten in die Datenbank existieren vier Module:

- Termine
- Prüfungen
- Diplomarbeiten, Fopras
- Oberseminartermine

Diplomarbeiten und Fopras werden als eine Einheit behandelt, da die Tabellen in der Datenbank ausser ihren Namen identisch sind.

Alle Module sind im wesentlichen gleich aufgebaut: Sie enthalten sowohl Formulare zum Eintragen, Bearbeiten, Löschen und Anzeigen von Daten, als auch PHP-Skripten, die für die Datenbankänderungen zuständig sind.

Durch hidden-Variablen, die an ein zentrales Skript (*zentral.php*) weitergereicht werden, wird entschieden, welches Skript als nächstes ausgeführt wird (Variablen *daten* oder *loeschen*) und welches Formular bei erfolgreicher (Variable *erfolg*) Datenbankänderung oder im Fehlerfall (Variable *fehler*) als nächstes angezeigt (Variable *anzeige*) wird. Die Werte der Variablen sind Nummern (siehe Abbildungen), die das jeweilige Skript oder Formular identifizieren. Der Übersichtlichkeit halber haben Formulare eines Moduls Nummern eines Hunderterbereichs.

Das Skript *zentral.php* koordiniert die einzelnen Skripten, wie in Abblidung 5.2 zu sehen ist, folgendermaßen: Sind in einem Formular die Variablen *daten* oder *loeschen* gesetzt und wird der ok-Knopf gedrückt, so wird *zentral.php* aufgerufen und das entsprechende Skript per *include* angefordert. Include bewirkt, dass das entsprechende Skript an diese Stelle im Quelltext eingebunden wird.

Alle Skripten, die mittels *daten* oder *loeschen* aufgerufen werden, enthalten - bis auf eine Ausnahme, auf die später eingegangen wird - die Funktionen *dateneintragen( )* und *datenloeschen( )*. Ist die Variable *loeschen* nicht gesetzt und der Datenbankeintrag erfolgreich durchgeführt, oder die Variable *loeschen* gesetzt und der Löschvorgang erfolgreich beendet, so wird im Zentralskript der Wert der Variablen *erfolg* der Variablen *anzeige* zugewiesen. Andernfalls wird *fehler* zugewiesen. Mittels *anzeige* wird dann das entsprechende Formular angefordert und angezeigt.

Beim Bearbeiten der Formulare werden die Werte aus der Datenbank in das aktuelle Formular übernommen, um dann verändert werden zu können. Die Variablen, die jedes Formular weitersendet, sind den Abbildungen der einzelnen Module zu entnehmen, die im Folgenden detailliert beschrieben werden.

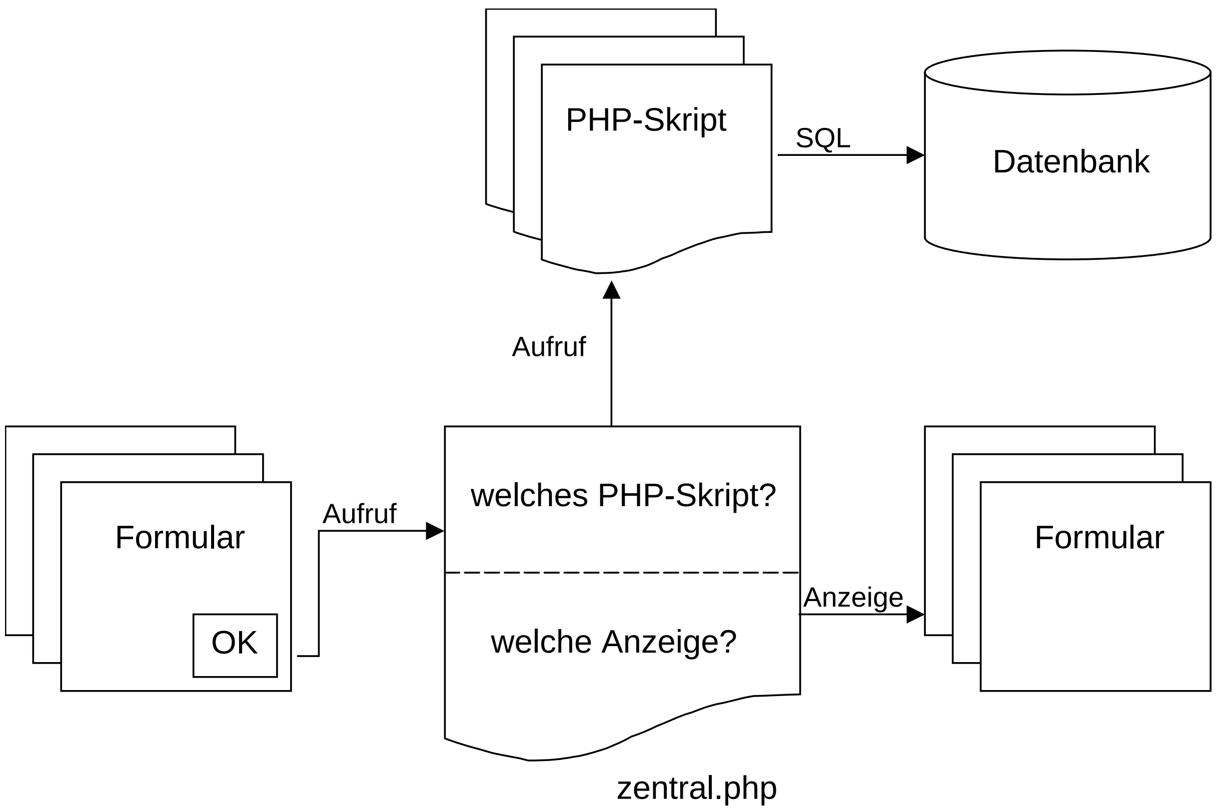

Abbildung 5.2: Verwaltung der Skripten

5.3.1 Termine

Formular 110 beinhaltet die Terminübersicht. Hier werden alle allgemeinen Termine aus der Datenbank angezeigt. Sie können nach verschiedenen Attributen, nämlich nach Text, Anfang, Ende, Url, Anzeigezeitraum und Anzeigeseite sortiert werden.

Über die Links *Neuen Termin eintragen* und *bearbeiten* gelangt man zu Formular 101. Die hidden-Variablen können der Abbildung 5.3 entnommen werden. Beim Bearbeiten wird die Variable *id* mitübergeben, damit erkannt wird, um welchen Termin es sich handelt. Der Wert von *id* wird der Variablen *terminnummer* zugewiesen, so dass die Funktion *dateneintragen( )* weiß, dass es sich um einen Update handelt. Sind weder *id* noch *terminnummer* gesetzt, wird ein Neueintrag durchgeführt. Dieses Konzept erlaubt es, dasselbe Formular für einen Neueintrag und zur Bearbeitung herzunehmen. Es wird in den anderen Modulen - nur mit anderen Variablennamen - genauso gehandhabt.

Über die Textfelder kann ein Termin mit Anfangs - und gegebenenfalls Enddatum sowie Uhrzeit und URL eingegeben werden. Außerdem kann ein beliebiger Wiederholungszeitraum - falls gewünscht - gewählt werden, sodass regelmäßige Termine nur einmal eingetragen werden müssen. Sie werden dann automatisch - je nach Anzeigezeitraum - angezeigt.

Bei Anzeigeseite kann zwischen INTERN, TU, LMU und COMMON ausgewählt werden. Der Termin erscheint auf der Seite des entsprechenden Servers.

Der Link *löschen* in der Terminübersicht führt zu Formular 102. Hier wird ein Termin aus der Datenbank entfernt, indem, wie bereits beschrieben, die Funktion datenloeschen() aufgerufen wird.

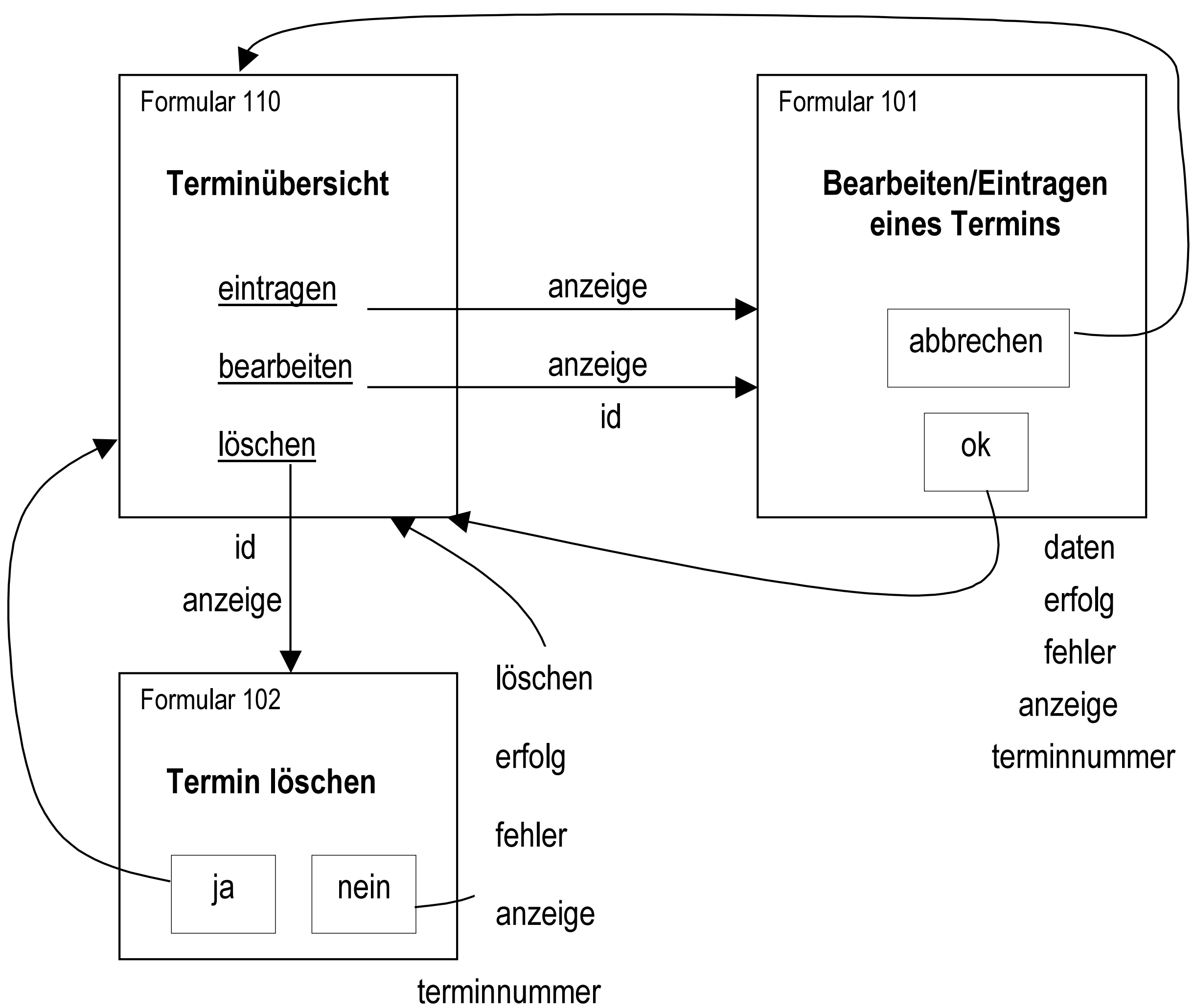

Abbildung 5.3: Das Modul Termine: Flußdiagramm.

5.3.2 Prüfungen

Formular 210 stellt die Prüfungsübersicht dar. Es werden Datum, Anfang, Ende und Kommentar der Prüfung, sowie Prüfling, Prüfer und - falls bereits eingetragen - Beisitzer angezeigt.

Über die Links *Neue Prüfung eintragen* bzw. *bearbeiten* gelangt man zu Formular 201, wobei beim Bearbeiten die *id* der Prüfung als Variable mitübergeben wird. Datum, Anfang, Ende und Kommentar sind Textfelder, Prüfer und Beisitzer Listenfelder, wo jeweils ein Attribut selektiert werden kann. Standardmäßig ist kein Beisitzer ausgewählt.

Wie Abbildung 5.4 zu entnehmen ist, führt der Button *Studentensuche* zu Formular 211. Hierbei werden die Werte der Text - und Listenfelder als hidden-Variablen an alle folgenden Formulare weitergereicht, so dass bereits vorgenommene Eintragungen beim Zurückkehren zu Formular 201 nicht verloren gehen. Um eine lange Liste von tausenden von Studenten zu vermeiden, beinhaltet Formular 211 eine Suchfunktion, mit der gezielt nach Namen und/oder Matrikelnummer eines Studenten gesucht werden kann. Hierbei gilt die übliche Unix-Vereinbarung: "*xx*" im Textfeld Name beispielsweise, sucht nach allen Studenten, in deren Vor - oder Nachnamen "xx" enthalten ist. Es sollte also niemals "*" alleine verwendet werden, da sonst alle in der Datenbank eingegebenen Studenten aufgelistet werden.

Die in der Datenbank gefundenen Studenten werden in Formular 220 in einem Listenfeld aufgeführt. Es kann nun ein Student ausgewählt und mittels Button *Übernehmen* in Formular 201 übernommen werden. Hier können die einzelnen Felder nochmals abgeändert werden, ansonsten bewirkt der ok-Button den Datenbankeintrag. Die Felder Datum, Anfang und Ende müssen ausgefüllt sein und ein Student muss angegeben sein, da andernfalls der Datenbankeintrag nicht erfolgt.
Über den Link *löschen* in Formular 210 lässt sich - wie bei den Terminen - ein Prüfungseintrag aus der Datenbank entfernen.

5.3.3 Diplomarbeiten und Fopras

Da sich die Behandlung von Diplomarbeiten und Fopras kaum unterscheidet, wurde hier eine gemeinsame Anwendung implementiert, wie Abbildung 5.5 zu entnehmen ist.

Formular 510 stellt daher eine Übersicht über alle offenen und vergebenen Ausschreibungen dar. Die Übersicht ist sortiert nach Diplomarbeiten, gefolgt von den Fopras, die jeweils wieder nach verschiedenen Attributen sortiert werden können. Hier sind es Thema, Beschreibung, Anzeigeseite, noch aushängen (im Schaukasten) und vergeben.

Die Links *Neue Arbeit ausschreiben* bzw. *bearbeiten* führen zu Formular 501. Beim Bearbeiten wird, je nach Ausschreibung, die Variable *id_d* für Diplomarbeit oder *id_f* für Fopra übergeben. Um einen Datenbankeintrag zu gewährleisten, müssen zumindest die Textfelder Thema, Beschreibung und Ausschreibungsdatum korrekt ausgefüllt sein. Die Felder Anforderungen und Arbeit verlängern sind optional. Bei Anzeigeseite kann wie im Modul Termine wieder zwischen INTERN, TU, LMU und COMMON ausgewählt werden. Die Checkbox noch aushängen dient als Erinnerung, ob eine Ausschreibung bereits im Schaufenster ausgehängt wurde. Zur Identifikation der Ausschreibung dienen die Radiobuttons Diplomarbeit und Fopra. Diese sind beim Bearbeiten einer Ausschreibung nicht mehr sichtbar. Wurde also beispielsweise aus Versehen Diplomarbeit statt Fopra ausgewählt und in die Datenbank eingetragen, muss die Ausschreibung gelöscht und die Felder neu ausgefüllt werden.

Über den Link *Details* gelangt man zu Formular 610. Hier werden die Informationen bezüglich Bearbeiter einer Arbeit, Betreuer und Bearbeitungszeitraum verwaltet bzw. angezeigt. Hierbei wird an jedes folgende Formular die entsprechende id (*id_d* bzw.*id_f*) zur Identifikation der Arbeit weitergereicht. Eine Arbeit gilt als vergeben, sobald beim Bearbeitungszeitraum ein Anfangsdatum eingetragen ist.

Der Bearbeitungszeitraum wird in Formular 613 verwaltet. Es besteht auch die Möglich-

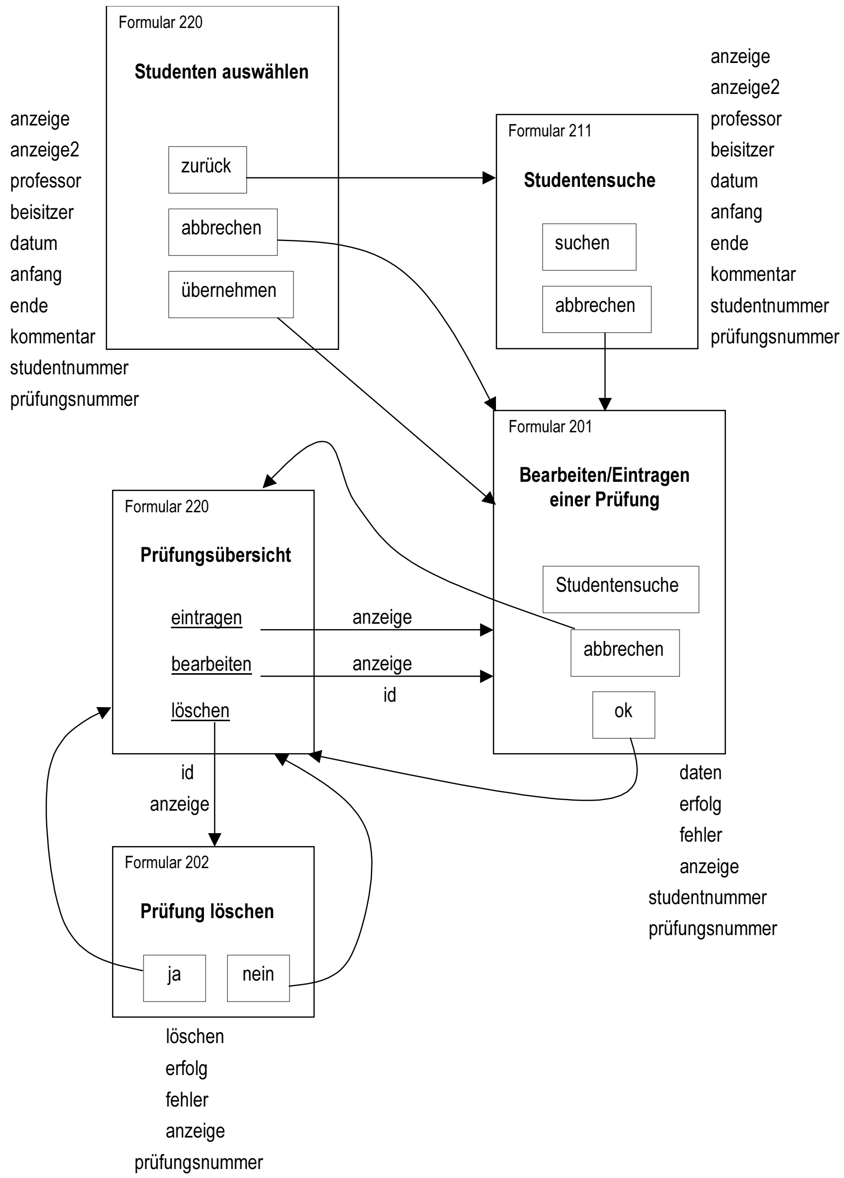

Abbildung 5.4: Das Modul Prüfungen: Flußdiagramm.

keit, eine Arbeit zu verlängern. Die Verlängerung wird in Tagen angegeben und kann jederzeit, wie auch der Bearbeitungszeitraum, abgeändert werden. An dieser Stelle ist anzumerken, dass das zu diesem Formular gehörige Skript, das den Datenbankeintrag bewirkt, als einziges, wie bereits erwähnt, nicht die Funktion *datenloeschen()* enthält. Anders als bei allen anderen Attributen kann der Bearbeitungszeitraum nämlich nicht per Button aus der Datenbank gelöscht werden. Dieser wird bei fehlender Eingabe durch den Benutzer im Skript automatisch aus der Datenbank entfernt.

Einer Ausschreibung können beliebig viele Studenten und Betreuer zugeordnet werden. Dies erlaubt wieder die gemeinsame Behandlung von Fopras und Diplomarbeiten, die in der Regel unterschiedlich viele Bearbeiter haben. Formular 611 stellt eine Suchfunktion für Studenten dar. Diese wurde schon im Zusammenhang mit den Prüfungen beschrieben und funktioniert hier auf gleiche Weise.

Formular 620 enthält alle, den Suchkriterien entsprechenden, Studenten in einem Listenfeld. Der ausgewählte Student wird beim Übernehmen sowohl in die Datenbank eingetragen, als auch in Formular 610 angezeigt.

Mit Formular 614 kann ein Student aus der Zuordnung gelöscht werden.

Ein Betreuer für die entsprechende Diplomarbeit wird in Formular 612 aus einem Listenfeld ausgewählt und ebenfalls mit Übernehmen sowohl in die Datenbank eingetragen als auch in Formular 610 angezeigt.

Das Formular zum Löschen eines Betreuers ist Formular 615.

In Formular 610 gibt es die Möglichkeit, eine Diplomarbeit oder ein Fopra komplett, auch mit Zuordnung der Bearbeiter und Betreuer, aus der Datenbank zu entfernen.

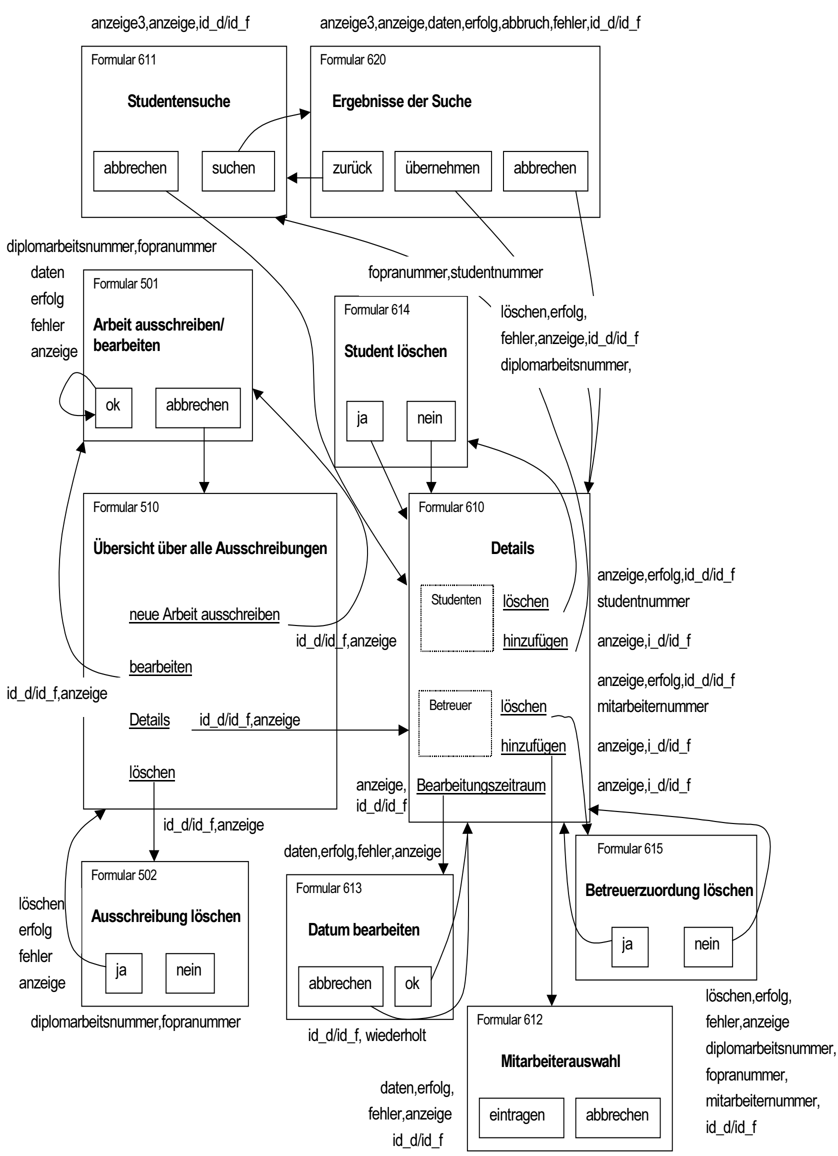

Abbildung 5.5: Das Modul Diplomarbeiten und Fopras: Flußdiagramm.

5.3.4 Oberseminartermine

Formular 310 gibt eine Übersicht über alle Oberseminartermine. Angezeigt werden die Attribute Datum, Uhrzeit, Kommentar, Verzeichnisnummer der TU und LMU, Anzahl der bisher zugeordneten Vorträge, verantwortlicher Mitarbeiter, ob der Termin schon im Web veröffentlicht wird und der Titel. Der Titel dient zur Unterscheidung zwischen normalem Oberseminar, bei dem über Fopras und Diplomarbeiten referiert wird und Doktorandenkolloquium, bei dem ein externer Referent einen Vortrag hält.

Wie Abbildung 5.6 zu entnehmen ist, führen die Links *Neuen Oberseminartermin eintragen* bzw. *bearbeiten* zu Formular 301. Beim Bearbeiten wird wieder die id weitergereicht. Um einen Datenbankeintrag zu erzielen, müssen die Felder Datum, Anfang und Ende korrekt ausgefüllt sein. Das Kommentarfeld sowie die Verzeichnisfelder sind optional. Letztere dienen lediglich der Erstellung der Vorlesungsverzeichnisse. Titel und verantwortlicher Mitarbeiter sind jeweils Listenfelder. Die Standardeinstellungen sind *normal* und *kein verantwortlicher Mitarbeiter ausgewählt*. Beim Bearbeiten können alle Attribute außer dem Titelfeld abgeändert werden.

Der Link *Vorträge anzeigen/hinzufügen* führt zu Formular 320. Hier werden alle zu einem bestimmten Oberseminartermin (Variable *obersemnr*) eingetragenen Vorträge angezeigt. Weiterhin besteht die Möglichkeit, von hier aus über *Vorträge hinzufügen* in Formular 330 einzelne Diplomarbeiten und Fopras mittels Mausklick anzukreuzen und so dem Oberseminartermin zuzuordnen. Es werden nur die vergebenen Diplomarbeiten und Fopras angezeigt, die noch keinem Oberseminartermin zugeteilt wurden.

Ein Oberseminartermin mit allen Vorträgen kann in Formular 302 komplett aus der Datenbank entfernt werden.

5.4 Authentifizierung

5.4.1 Benutzer

Bisher existieren in der Datenbank drei Arten von Benutzern, der superuser *postgres*, die Gruppe *mnmteam* sowie der user *wwwpublic*. Der superuser *postgres* besitzt alle Rechte auf alle Tabellen. Die Gruppe *mnmteam*, die aus den Mitarbeitern des Lehrstuhls besteht, erhält auf alle Tabellen, ausgenommen die Systemtabellen, sämtliche Rechte sowie Leserechte auf die Views (siehe Anhang: grant.mnmteam und views.sql). Der Benutzer *wwwpublic*, der alle nicht zum Lehrstuhl gehörenden Benutzer repräsentiert, erhält ausschliesslich Leserechte auf eine Auswahl an Tabellen (siehe Anhang: grant.allgemein).

5.4.2 Datenbankanmeldung

Da es nicht möglich ist, sich mittels Client-Zertifikat nach der Authentifizierung am Apache auch direkt an der Datenbank anzumelden, wurde folgendes Konzept implementiert:

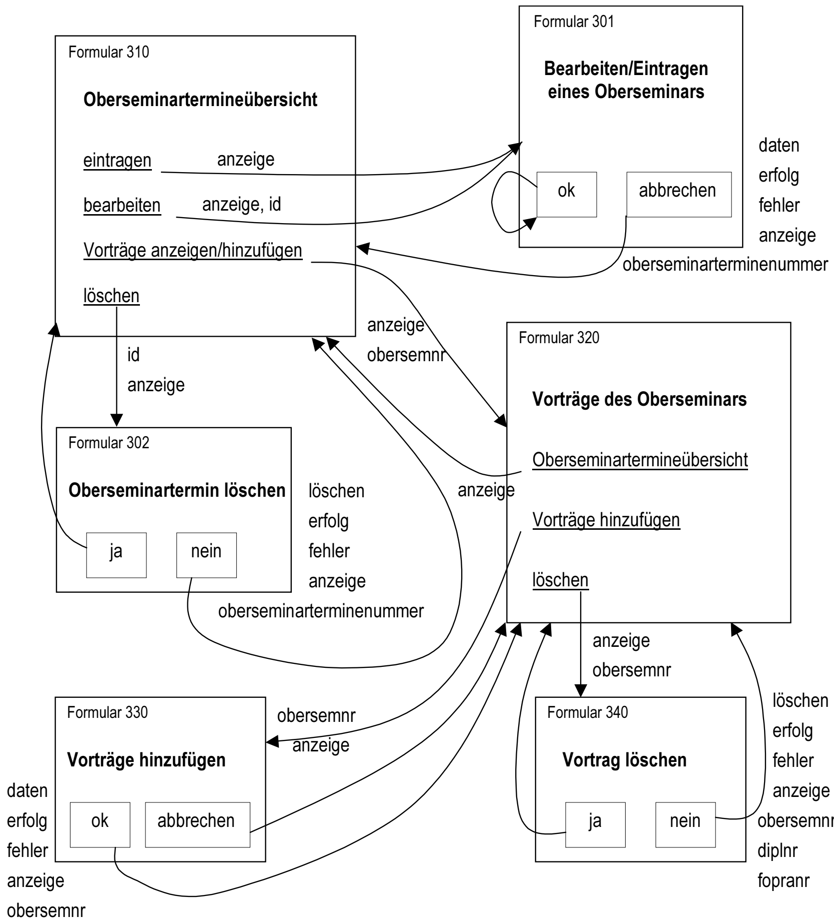

Abbildung 5.6: Das Modul Oberseminartermine: Flußdiagramm.

Die zur Authentifizierung der einzelnen Benutzer benötigten Informationen, nämlich Name, Login sowie das Passwort sind in der Datei authent gespeichert. Für jeden eingetragenen Benutzer ist eine Zeile reserviert. Zum Aufbau der Datenbankverbindung wird die Datei vom Programm zeilenweise eingelesen. Zusätzlich werden die Servervariablen $SSL_CLIENT_VERIFY und $SSL_CLIENT_S_DN_CN dem Programm mitgeliefert. Erfolgt die Datenbankanmeldung durch *wwwpublic*, so werden dessen Name und Passwort an die Datenbank weitergegeben. Andernfalls wird die Servervariable $SSL_CLIENT_VERIFY auf "SUCCESS" überprüft. Dies bedeutet, dass der Benutzer auf eine Seite, zu der er ein Zertifikat benötigt, zugreifen möchte. Die Informationen bzgl. des Zertifikats werden mittels der Servervariablen $SSL_CLIENT_S_DN_CN geprüft. Bei gültigem Zertifikat werden dann wiederum Name und Passwort der Datenbank übergeben. Abbildung 5.7 verdeutlicht nochmals die Schritte zur Datenbankanmeldung.

5.5 Anzeige im Web

Die Anzeige im Web wurde folgendermassen realisiert:
Es existieren zwei Server, der öffentliche Server sowie der Secure Server. Der öffentliche Server kann über zwei Namen angesprochen werden, http://sunheger5.nm.informatik.uni-

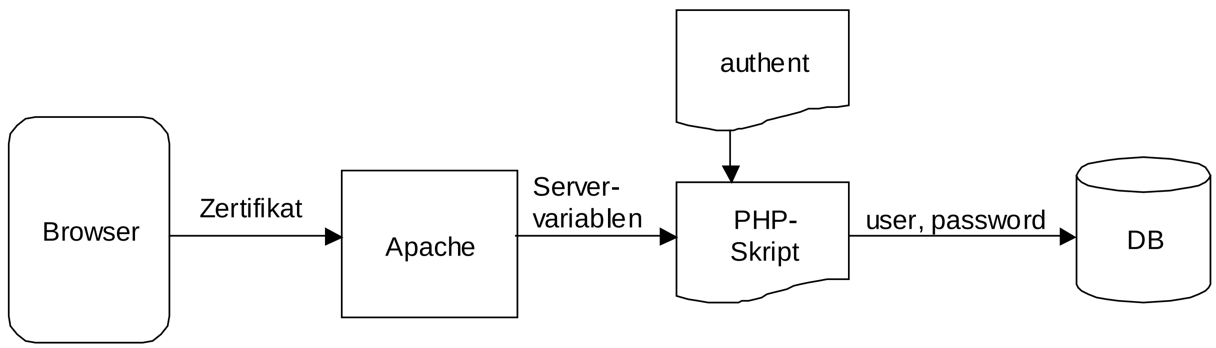

Abbildung 5.7: Authentifizierung gegenüber der Datenbank.

muenchen.de (LMU) und http://wwwtest.nm.informatik.uni-muenchen.de (TU). Der Secure Server wird über die Adresse https://sunheger5.nm.informatik.uni-muenchen.de angesprochen. Mittels der Servervariablen $SERVER_NAME wird entschieden, welche Seite (TU oder LMU) angezeigt werden soll. Dazu wird der entsprechende Header und Trailer der Universität mit *include* eingebunden. Die Servervariable $SSL_CLIENT entscheidet, ob es sich um die Anzeige einer internen oder öffentlichen Seite handelt.

Auf den internen Seiten werden grundsätzlich alle Informationen dargestellt, während für die öffentlichen Seiten nur die Daten aus der Datenbank ausgelesen werden, die für die entsprechende Universität (Spalte *anzeigeseite*: LMU bzw. TU) oder mit COMMON eingetragen wurden (Tabellen: *diplomarbeit, fopra, termine*). Für die Tabelle *oberseminartermine* entscheidet das Attribut *veröffentlichen*, ob ein Oberseminartermin zusätzlich zur internen Anzeige auch auf den öffentlichen Seiten erscheinen soll.

Kapitel 6

Zusammenfassung und Ausblick

Die gesamte Anwendung wurde für beide Datenbanken PostreSQL und MySQL implementiert und getestet. Das ausschlaggebende Kriterium für die endgültige Auswahl der Datenbank ist jedoch die Unterstützung von Views, damit bestimmte Daten dem öffentlichen Web unzugänglich bleiben.
Es ist auch im Nachhinein möglich, jede beliebige Datenbank zu verwenden. Dazu müssen lediglich die Datenbankfunktionen für die verwendete Datenbank geschrieben werden. Eine Bedingung, die eine neue Datenbank erfüllen muss, ist die Unterstützung von Views.

Die Einarbeitung in PHP erwies sich als relativ einfach. Sehr zeitaufwändig war neben der Installation die Entwicklung des Datenbankschemas sowie besonders die Implementierung der Formulare.

Das Datenbankschema wurde so entworfen, dass sich zusätzlich noch ein Konferenzenkalender realisieren lässt. Darin sollte die Zuordnung von Mitarbeitern zu Konferenzen so stattfinden, dass keine Themen mehrfach vorkommen. Ausserdem könnte ein Aufruf (automatische E-Mail-Funktion) zum rechtzeitigen Einreichen von Papieren realisiert werden (Call for Papers).

Ein anderer Punkt, der erweitert werden kann, ist die Benutzerverwaltung und Vergabe der Rechte an der Datenbank. Bislang werden diese Rechte noch manuell in der Datenbank vergeben. Es wäre jedoch sinnvoll, für diese Verwaltungsfunktionen geeignete Formulare zu implementieren, die die Zuordnung von Rechten an die Benutzer und die Umsetzung dieser Zuordnung auf der Ebene der Datenbank im Zusammenspiel mit dem Web-Server einfach und konsistent ermöglichen.

Anhang

Installationen

PostgreSQL

Downloaden und entpacken:

```
ftp://ftp.de.postgresql.org/
gunzip postgresql-7.0.3.tar.gz
tar -xf postgresql-7.0.3.tar
mv postgresql-7.0.3 <BASISPFAD>
```

Anlegen eines Installationsordners:

```
cd <BASISPFAD>
mkdir pgsql
```

Konfiguration des Source-Codes:

```
cd postgresql-7.0.3/src
./configure --prefix=<BASISPFAD>/pgsql
```

Übersetzen des Source-Codes:

```
gmake
```

Installation der ausführbaren Dateien und Bibliotheken:

```
gmake install
```

Umgebungsvariable `PATH` muss <*BASISPFAD*>`/pgsql/bin` enthalten
Umgebungsvariable `LD_LIBRARY_PATH` muss <*BASISPFAD*>`/pgsql` enthalten

Initialisierung:

```
initdb -D <BASISPFAD>/pgsql/data
```

Starten mit:

```
postmaster -i -D <BASISPFAD>/pgsql/data &
```

Beenden mit:

```
pg_ctl -D <BASISPFAD>/pgsql/data -m s stop
```

Apache und PHP

Downloaden und entpacken:

PHP: `http://www.php.net/downloads.php (php-4.0.3pl1)`
Apache: `http://httpd.apache.org/dist (apache_1.3.14)`

```
gunzip apache_1.3.14.tar.gz
tar -xvf apache_1.3.14.tar
gunzip php-4.0.3pl1.tar.gz
tar -xvf php-4.0.3pl1.tar
```

Konfiguration des Apache-Source-Codes:

```
cd apache_1.3.14
./configure --prefix=<BASISPFAD>/apache
```

Konfiguration des PHP-Source-Codes für PostgreSQL und Apache:

```
cd php-4.0.3pl1
./configure --prefix=<BASISPFAD>/php --with-pgsql=<BASISPFAD>/pgsql --with-apache=<BASISPFAD>/apache/apache_1.3.14
--enale-track-vars
```

Übersetzen des Source-Codes:

```
make
```

Installation der ausführbaren Dateien und Bibliotheken:

```
make install
```

Starten mit:

```
bin/apachectl start
```

Beenden mit:

```
bin/apachectl stop
```

Verschiedene Skripte

Die folgenden zwei **grant-Skrpite** bewirken die Rechtevergabe bestimmter Benutzer auf der PostgreSQL-Datenbank.
Das Skript **views.sql** definiert die Views der Datenbank.

grant.mnmteam

Mitglieder der Gruppe **mnmteam** erhalten sowohl alle Rechte (SELECT, INSERT, UPDATE, DELETE, RULE) auf alle Tabellen als auch Leserechte (SELECT) auf alle Views.

```
grant ALL on arbeitet_fuer_praktikum to GROUP mnmteam;
grant ALL on arbeitet_fuer_vorlesung to GROUP mnmteam;
grant ALL on aufgabe to GROUP mnmteam;

grant ALL on calls to GROUP mnmteam;

grant ALL on diplomarbeit to GROUP mnmteam;

grant ALL on doktorandenkolloquium to GROUP mnmteam;
grant ALL on einreichung to GROUP mnmteam;

grant ALL on fopra to GROUP mnmteam;

grant ALL on gruppe_hat_ausarbeitung_abgeg to GROUP mnmteam;
grant ALL on klausur to GROUP mnmteam;
grant ALL on klausur_in_raum to GROUP mnmteam;
grant ALL on konferenz to GROUP mnmteam;

grant ALL on mailadressenmitarbeiter to GROUP mnmteam;
grant ALL on mailadressenstudent to GROUP mnmteam;
grant ALL on mitarbeiter to GROUP mnmteam;
grant ALL on mitarbeiter_betr_diplomarbeit to GROUP mnmteam;
grant ALL on mitarbeiter_betr_fopra to GROUP mnmteam;
grant ALL on mitarbeiter_betr_seminarvortrag to GROUP mnmteam;


grant ALL on oberseminartermine to GROUP mnmteam;
grant ALL on personal_beaufsichtigt_klausur to GROUP mnmteam;
grant ALL on personal_korrigiert_aufgabe to GROUP mnmteam;
grant ALL on praktikum to GROUP mnmteam;

grant ALL on praktikumgruppe to GROUP mnmteam;
```

```
grant ALL on pruefung_muendlich to GROUP mnmteam;
grant ALL on raum to GROUP mnmteam;

grant ALL on schein to GROUP mnmteam;
grant ALL on schein_fuer_fopra to GROUP mnmteam;
grant ALL on schein_fuer_praktikum to GROUP mnmteam;
grant ALL on schein_fuer_seminar to GROUP mnmteam;
grant ALL on schein_fuer_vorlesung to GROUP mnmteam;

grant ALL on seminar to GROUP mnmteam;

grant ALL on seminartyp to GROUP mnmteam;
grant ALL on seminarvortrag to GROUP mnmteam;
grant ALL on student to GROUP mnmteam;
grant ALL on student_anwesend_bei_vortrag to GROUP mnmteam;
grant ALL on student_bearbeitet_uebungsblatt to GROUP mnmteam;
grant ALL on student_besucht_uebung to GROUP mnmteam;
grant ALL on student_besucht_vorlesung to GROUP mnmteam;
grant ALL on student_erhaelt_punkte to GROUP mnmteam;
grant ALL on student_haelt_seminarvortrag to GROUP mnmteam;
grant ALL on student_in_praktikumgruppe to GROUP mnmteam;
grant ALL on student_macht_diplomarbeit to GROUP mnmteam;
grant ALL on student_macht_fopra to GROUP mnmteam;
grant ALL on student_macht_praktikum to GROUP mnmteam;
grant ALL on student_schreibt_klausur to GROUP mnmteam;

grant ALL on termine to GROUP mnmteam;

grant ALL on uebungsblatt to GROUP mnmteam;
grant ALL on uebungsgruppe to GROUP mnmteam;
grant ALL on universitaet to GROUP mnmteam;
grant ALL on verfasser_der_einr to GROUP mnmteam;
grant ALL on vorlesung to GROUP mnmteam;
grant ALL on vorlesung_findet_statt_in to GROUP mnmteam;

grant ALL on wochentag to GROUP mnmteam;

grant select on termine_wwwpublic to GROUP mnmteam;
grant select on fopra_wwwpublic to GROUP mnmteam;
grant select on diplomarbeit_wwwpublic to GROUP mnmteam;
grant select on oberseminarterm_wwwpublic to GROUP mnmteam;
grant select on mitarbeiter_wwwpublic to GROUP mnmteam;
grant select on student_wwwpublic to GROUP mnmteam;

grant ALL on aufgabe_aufgabennummer_seq to GROUP mnmteam;
```

```
grant ALL on calls_callsnummer_seq to GROUP mnmteam;
grant ALL on diplomarbeit_diplomarbeitsn_seq to GROUP mnmteam;
grant ALL on einreichung_einreichnummer_seq to GROUP mnmteam;
grant ALL on fopra_fopranummer_seq to GROUP mnmteam;
grant ALL on konferenz_konferenznummer_seq to GROUP mnmteam;
grant ALL on mitarbeiter_mitarbeiternumm_seq to GROUP mnmteam;
grant ALL on oberseminarte_oberseminarte_seq to GROUP mnmteam;
grant ALL on praktikum_praktikumnummer_seq to GROUP mnmteam;
grant ALL on pruefung_muen_pruefungsnumm_seq to GROUP mnmteam;
grant ALL on raum_raumnummer_seq to GROUP mnmteam;
grant ALL on schein_scheinnummer_seq to GROUP mnmteam;
grant ALL on seminar_seminarnummer_seq to GROUP mnmteam;
grant ALL on student_studentnummer_seq to GROUP mnmteam;
grant ALL on termine_terminnummer_seq to GROUP mnmteam;
grant ALL on vorlesung_vorlesungnummer_seq to GROUP mnmteam;
```

grant.allgemein

Der Benutzer **wwwpublic** erhält Leserechte (SELECT) auf ausgewählte Tabellen sowie auf alle Views.

```
grant select on mailadressenmitarbeiter to wwwpublic;
grant select on mailadressenstudent to wwwpublic;

grant select on mitarbeiter_betr_diplomarbeit to wwwpublic;
grant select on mitarbeiter_betr_fopra to wwwpublic;

grant select on student_macht_diplomarbeit to wwwpublic;
grant select on student_macht_fopra to wwwpublic;

grant select on universitaet to wwwpublic;

grant select on wochentag to wwwpublic;

grant select on termine_wwwpublic to wwwpublic;
grant select on fopra_wwwpublic to wwwpublic;
grant select on diplomarbeit_wwwpublic to wwwpublic;
grant select on oberseminarterm_wwwpublic to wwwpublic;

grant select on mitarbeiter_wwwpublic to wwwpublic;
grant select on student_wwwpublic to wwwpublic;

grant select on doktorandenkolloquium to wwwpublic;
```

views.sql

```
CREATE VIEW termine_wwwpublic AS
SELECT * FROM termine WHERE anzeigeseite <> 0;

CREATE VIEW oberseminarterm_wwwpublic AS
SELECT * FROM oberseminartermine WHERE veroeffentlichen = 't';

CREATE VIEW fopra_wwwpublic AS
SELECT * FROM fopra WHERE anzeigeseite <> 0;

CREATE VIEW diplomarbeit_wwwpublic AS
SELECT * FROM diplomarbeit WHERE anzeigeseite <> 0;

CREATE VIEW mitarbeiter_wwwpublic AS
SELECT titel, name, mitarbeiternummer, vorname, raumnummer, telefonnummer, typ,
firma, homepage FROM mitarbeiter;

CREATE VIEW student_wwwpublic AS
SELECT name, vorname, studentnummer, universitaetsnummer FROM student;
```

ER-Diagramme

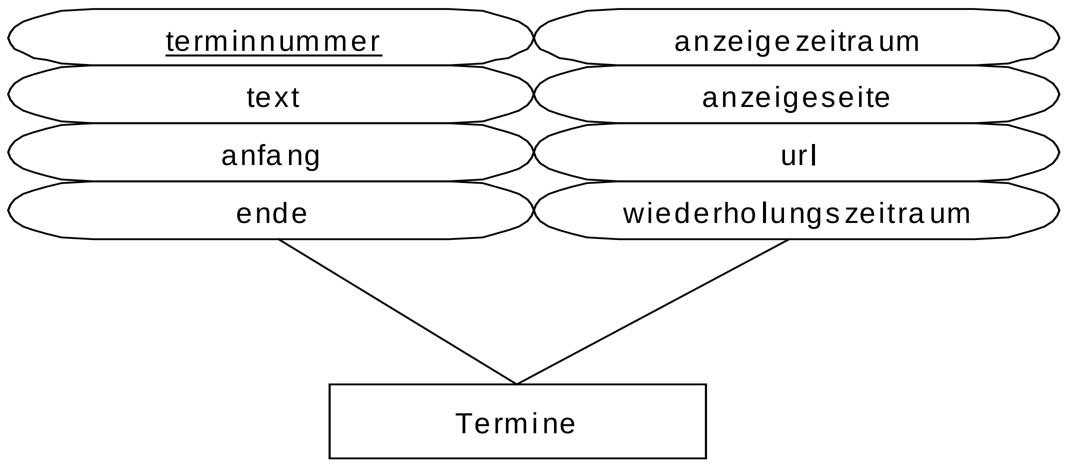

Abbildung 6.1: ER-Diagramm für Termine

Tabelle termine

- terminnummer, SERIAL: Integerwert als Primärschlüssel
- text, TEXT: Terminbeschreibung
- anfang, TIMESTAMP: Anfangsdatum und eventuelle Anfangsuhrzeit
- ende, TIMESTAMP: Enddatum und eventuelle Enduhrzeit
- anzeigezeitraum, DATE: der Termin erscheint ab diesem Datum im Netz
- anzeigeseite, INT: Die Anzeigeseite wird durch entsprechende Bitoperation dargestellt: 0=INTERN (weder LMU noch TU), 1=LMU, 2=TU, 3=COMMON (LMU und/oder TU)
- url, VARCHAR: eine URL wird entweder relativ (zum Server) oder absolut angegeben
- wiederholungszeitraum, INT: Es kann jeder beliebige Zeitraum in Tagen, Wochen, Monaten oder Jahren gewählt werden. In der Datenbank ist in der Spalte wiederholungszeitraum ein 32-Bit-Integer angegeben, wobei die oberen 16 Bit für die Zeiteinheit (Tage, Wochen, Monate, Jahre) und die unteren 16 Bit für die entsprechende Zahl reserviert sind.

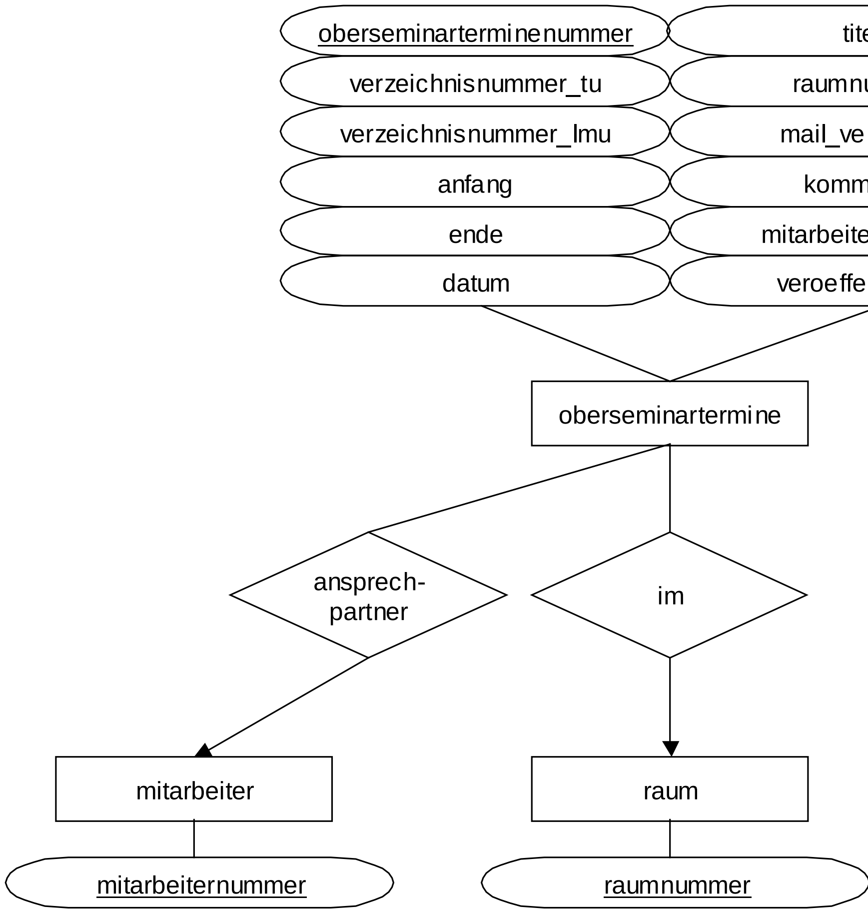

Abbildung 6.2: ER-Diagramm für Oberseminare

Tabelle oberseminartermine

- oberseminarterminenummer, SERIAL: Integerwert als Primärschlüssel
- verzeichnisnummer_tu, VARCHAR: Nummer im Vorlesungsverzeichnis der TU
- verzeichnusnummer_lmu, VARCHAR: Nummer im Vorlesungsverzeichnis der LMU
- anfang, TIME: Anfangsuhrzeit
- ende, TIME: Enduhrzeit
- datum, DATE: Datum
- titel, BOOL: 0=Doktorandenkolloquium, 1=normales Oberseminar
- raumnummer, INT: Fremdschlüssel aus der Raumtabelle, gibt die Raumnummer an
- mail_verschickt, BOOL: 1=es wurde bereits eine Erinnerungsmail verschickt
- kommentar, TEXT: beliebiger Text

- mitarbeiternummer, INT: Fremdschlüssel aus der Mitarbeitertabelle, gibt den für das OS verantwortlichen Mitarbeiter an
- veroeffentlichen, BOOL: 1=die Oberseminarankündigung soll ab jetzt im Netz erscheinen

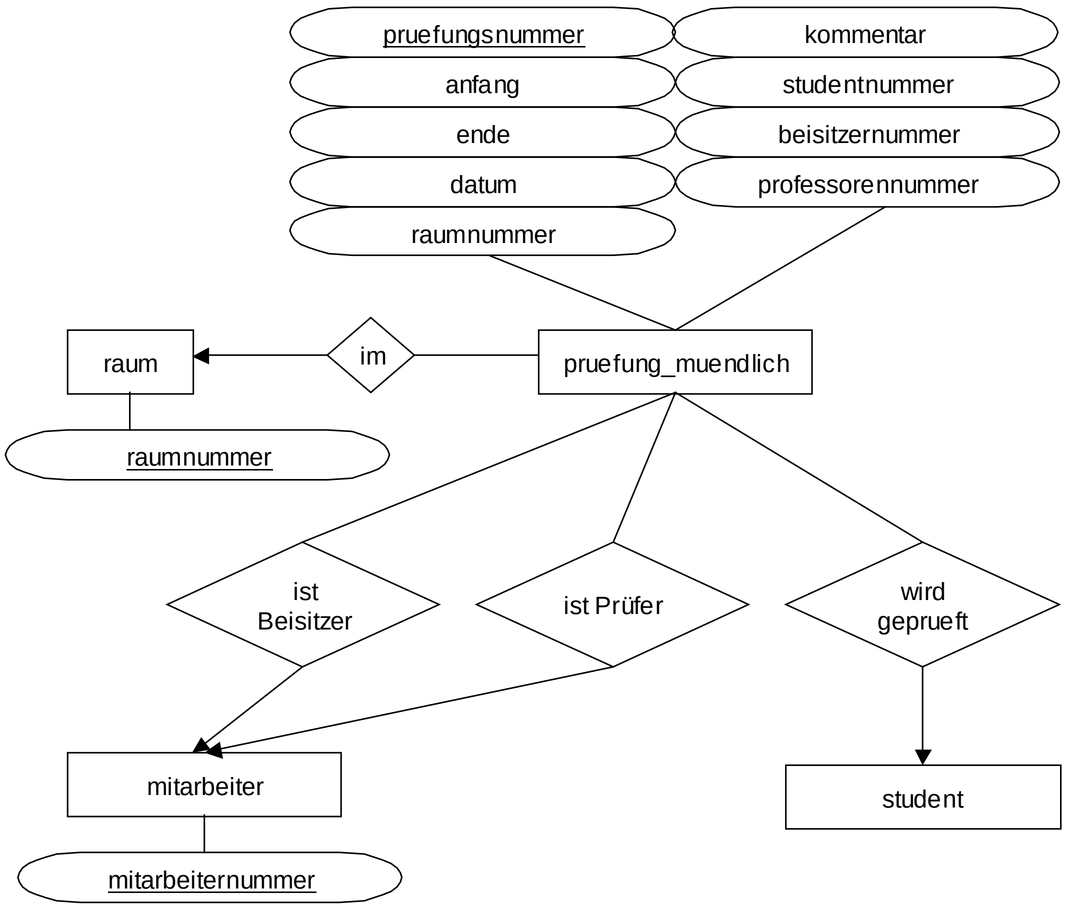

Abbildung 6.3: ER-Diagramm für Prüfungen

Tabelle pruefung_muendlich

- pruefungsnummer, SERIAL: Integerwert als Primärschlüssel
- anfang, TIME: Anfangsuhrzeit
- ende, TIME: Enduhrzeit
- datum, DATE: Datum
- raumnummer, INT: Fremdschlüssel aus der Raumtabelle, gibt die Raumnummer an
- kommentar, TEXT: beliebiger Text zur Beschreibung der Prüfung
- studentnummer, INT: Fremdschlüssel aus der Studenttabelle, gibt den Prüfling an
- beisitzernummer, INT: Fremdschlüssel aus der Mitarbeitertabelle, gibt den Beisitzer an
- professorennummer, INT: Fremdschlüssel aus der Mitarbeitertabelle, gibt den Prüfer an

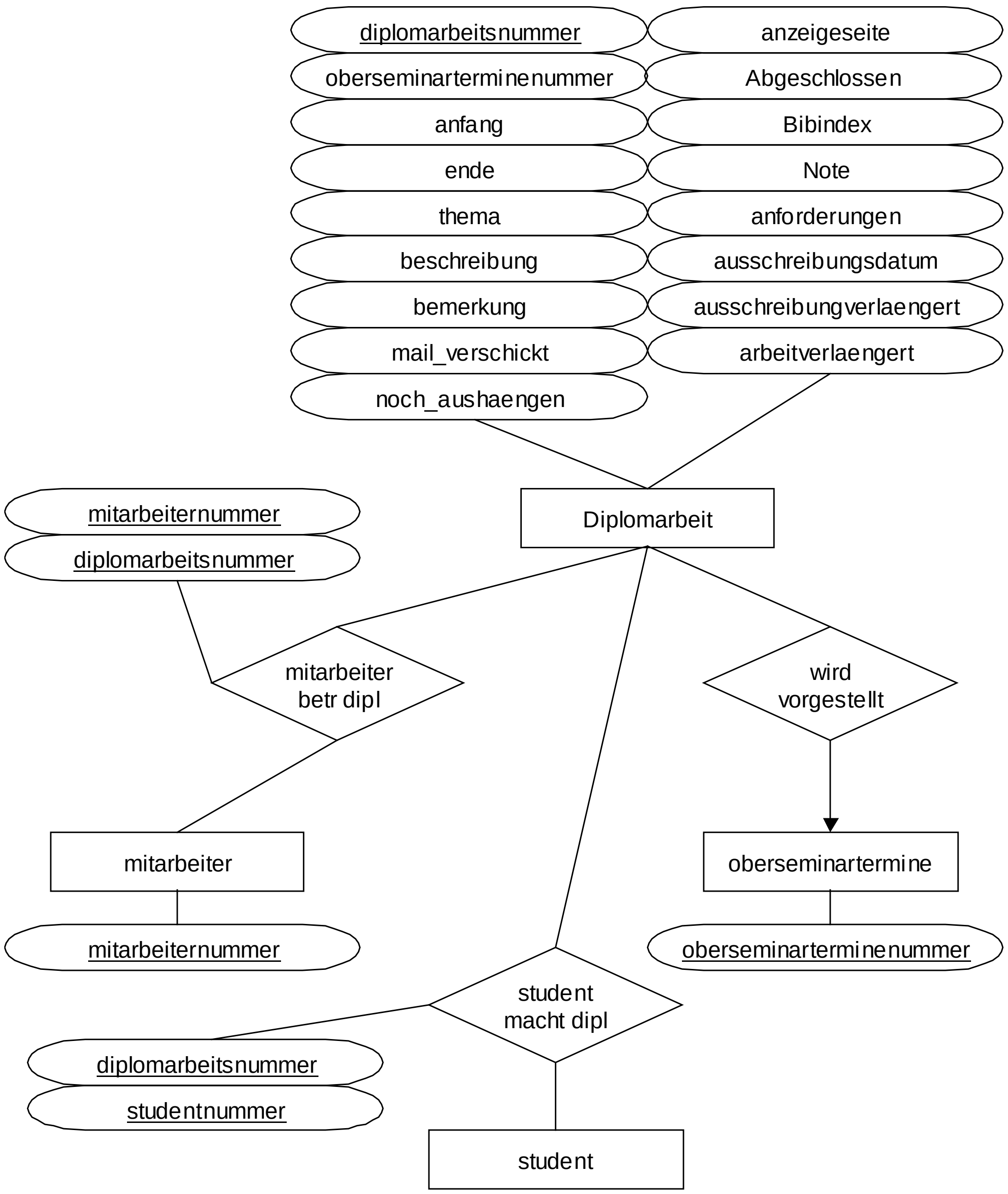

Abbildung 6.4: ER-Diagramm für Diplomarbeiten

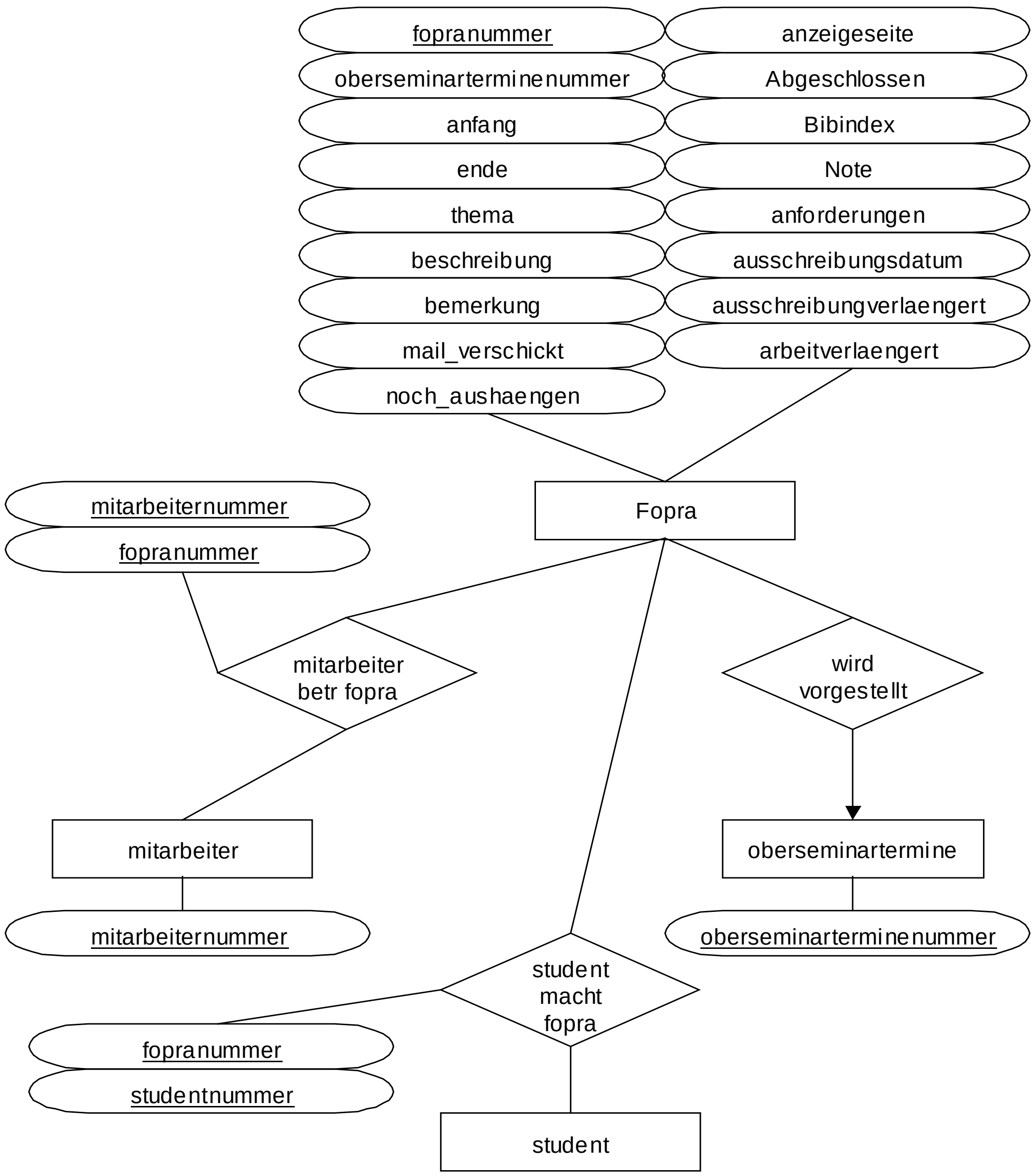

Abbildung 6.5: ER-Diagramm für Fopras

Tabelle Diplomarbeit/Fopra

- diplomarbeitsnummer/fopranummer, SERIAL: Integerwert als Primärschlüssel
- oberseminarterminenummer, INT: Fremdschlüssel aus der Oberseminartermineta-belle, gibt das OS an, an dem der Vortrag gehalten wird
- anfang, DATE: Anfangsdatum
- ende, DATE: Enddatum
- thema, TEXT: Thema der Arbeit
- beschreibung, TEXT: Beschreibung der Arbeit

- bemerkung, TEXT: beliebiger Text
- mail_verschickt, BOOL: 1=es wurde bereits eine Erinnerungsmail zum Verlängern der Arbeit verschickt
- noch_aushaengen, BOOL: 1=die Arbeit soll weiterhin im Netz angezeigt werden
- anzeigeseite, INT: Die Anzeigeseite wird durch entsprechende Bitoperation dargestellt: 0=INTERN (weder LMU noch TU), 1=LMU, 2=TU, 3=COMMON (LMU und/oder TU)
- Abgeschlossen, BOOL: 1=Arbeit ist bereits abgeschlossen
- Bibindex, VARCHAR:
- Note, VARCHAR: Bewertung
- anforderungen, TEXT: Anforderungen an die Arbeit
- ausschreibungsdatum, DATE: Datum der Ausschreibung
- ausschreibungverlaengert, INT: Anzahl der Tage, um die eine offene Ausschreibung verlängert wurde
- arbeitverlaengert, INT: Anzahl der Tage, um die eine vergebene Arbeit verlängert wurde

Tabelle mitarbeiter_betr_diplomarbeit/fopra

- mitarbeiternummer, INT: Fremdschlüssel aus der Mitarbeitertabelle, gibt den Betreuer der Arbeit an
- diplomarbeitsnummer/fopranummer, INT: Fremdschlüssel aus der Diplomarbeits-/Fopratabelle, gibt an, welche Arbeit ein Mitarbeiter betreut

Tabelle student_macht_diplomarbeit/fopra

- diplomarbeitsnummer/fopranummer, INT: Fremdschlüssel aus der Diplomarbeits-/Fopratabelle, gibt an, welche Arbeit ein Student bearbeitet
- studentnummer, INT: Fremdschlüssel aus der Studenttabelle, gibt den Bearbeiter der Arbeit an

Literaturverzeichnis

[0] Apache Web-Server, L. Eilebrecht, mitp-Verlag, Bonn, 2000

[1] Datenbankanbindung für dynamische Web-Seiten, http://www.torsten-horn.de/techdocs/db-web.htm

[2] Datenbank-Anbindung über das World Wide Web - Die CGI-Anwendung, http://www.stud.fernuni-hagen.de/q4167775/sem1908/cgi.htm

[3] Interbase, http://www.borland.com/interbase/

[4] Java Server Pages, http://java.sun.com/products/jsp/jspservlet.html

[5] MySQL, http://www.mysql.com

[6] PHP4: Grundlagen und Profiwissen, J. Krause, Hanser, 2000

[7] PostgreSQL, http://www.de.postgresql.org/

[8] Appendix B - Wörterbuch, http://www.accessarchive.com/FAQs/Modelling/German/Appendix_B_Woerterbuch.htm

CARMEN BARTH, SANDRA VAN MARWICK

SYSTEMENTWICKLUNGSPROJEKT:

Evaluation der Anbindung einer SQL-Datenbank an einen Apache Webserver