Berufsverband Information Bibliothek e.V.

OPL-Kommission

Checklisten

Liste 15

PHP und MySQL

von

Katrin Pieschel und Steffen Paech

Berufsverband Information Bibliothek e.V.

OPL-Kommission

Herausgegeben von der Kommission für One-Person Librarians des Berufsverbands Information Bibliothek BIB.
Erscheint als PDF-Dokument zum Herunterladen aus dem Netz in der 1. Auflage 2006.

Zitiervorschlag: PHP und MySQL / Katrin Pieschel ; Steffen Paech. Hrsg. Berufsverband Information Bibliothek / Kommission für One-Person Librarians. – 1. Aufl. – 2006. (Checklisten ; 15) <http://www.bib-info.de/komm/opl/pub/check15.pdf>

Inhaltsverzeichnis

1. Einleitung

Viele Bibliotheken präsentieren sich heute im Internet. Dazu gehören oftmals folgende Informationen:

- Über die Bibliothek selber: Bestand, Öffnungszeiten, Veranstaltungen usw.
- Der eigene Online-Katalog: mit und ohne Kontofunktionen: wie Vormerkung, Verlängern u.ä.
- Weitere Informationen: Zugang zu Datenbanken, freie u./o. kostenpflichtig, Kooperationspartnern, Veranstaltungen u.v.m.

Dafür reicht es längst nicht mehr aus, eine Homepage mit statischen HTML-Seiten zu erstellen. Denn diese haben den Nachteil, dass mit steigendem Seitenumfang der Pflegeaufwand an einzelnen HTML-Seiten überproportional steigt.

Viele Informationen, die in die Homepage der Bibliothek einfließen können, sind ja bereits in Datenbanken verschiedenster Art vorhanden. Damit Informationen auf der Homepage der Bibliothek immer aktuell sind bzw. der Benutzeranfrage entsprechen, müssen diese dynamisch generiert werden.

Hierfür bieten sich solche Datenbanken und Skriptsprachen an, die frei verfügbar (Open source, Freeware, GNU usw.) sind, offene Schnittstellen bieten und nicht-properitär (an eine Firma und/oder bestimmte Hardwareumgebung gebunden) sind. Weit verbreitet dafür sind MySQL, „die populärste Open-Source-Datenbank der Welt“, und PHP als Skriptsprache, um Datenbankinhalte auf HTML-Seiten anzeigen zu können. PHP ist nicht nur im Zusammenhang mit MySQL von Bedeutung, sondern lässt sich auch gut zur Gestaltung von barrierefreien Internetseiten einsetzen.

In den nachfolgenden Punkten soll darauf näher eingegangen werden. Die vorliegende Checkliste kann und soll aber Handbücher zu PHP und MySQL nicht ersetzen. Ziel ist es, dem interessierten Laien einen Einstieg in die Thematik zu ermöglichen, und dabei auf umfangreiche weiterführende Quellen zu verweisen.

1.1 PHP

PHP steht als Abkürzung für „**PHP: Hypertext Preprocessor“** und ist eine weitverbreitete Open-Source Skriptsprache für Webanwendungen. Das Sprachkonzept ist an C, Perl und Java angelehnt. PHP-Code lässt sich direkt in HTML einbinden, Voraussetzung ist ein Web-Server, der PHP-Befehle interpretieren kann. Mit PHP haben Webentwickler die Möglichkeit,

schnell dynamisch generierte Webseiten zu erzeugen. Neben MySQL lassen sich damit auch andere Datenbanken abfragen.

PHP wurde 1995 von Rasmus Lerdorf geschaffen. Seitdem hat sich PHP beträchtlich weiterentwickelt. Einen ausführlichen Hintergrund dazu kann man hier finden:
http://www.php.net/manual/de/history.php

1.2 MySQL

MySQL ist ein relationales Datenbanksystem. **SQL**[1] steht für **S**tructured **Q**uery **L**anguage und bedeutet, dass es sich um eine strukturierte Abfragsprache handelt. Mit SQL verfügt man über eine einfache Syntaxsprache, die Befehle zur Definition von Datenstrukturen, zur Manipulation von Datenbeständen (Anfügen, Bearbeiten, Löschen) und zur Abfrage von Daten bereitstellt. Heute hat sich SQL als Quasi-Standard etabliert und ist von großer Bedeutung, da hiermit eine weitgehende Unabhängigkeit von der benutzten Software erzielt werden kann. Viele bekannte Datenbanksysteme wie ORACLE, Informix, DB2 (IBM), Microsoft SQL, PostgreSQL, MySQL, ACCESS u.v.m. implementieren Teile des Sprachstandards SQL, verfügen allerdings auch zusätzlich über herstellerspezifische Funktionen, die nicht dem Standard entsprechen.

MySQL ist freie Software und steht unter GPL[2], aber auch unter einer kommerziellen Lizenz[3] zur Verfügung.
MySQL läuft als Datenbanksystem sowohl unter den meisten UNIX-Varianten, MAC-OS, Windows und OS/2 (nur Vers. 3.x). Oftmals findet man MySQL in Kombination mit einem Apache-Webserver und PHP (als Skriptsprache), um dynamische Webseiten generieren zu können.

---

1 s.a. http://de.wikipedia.org/wiki/SQL
2 **GNU General Public License** (*GPL*) ist eine von der Free Software Foundation herausgegebene Lizenz zur Lizenzierung freier Software.
3 s. a. hier: http://de.wikipedia.org/wiki/Duales_Lizenzsystem

2. PHP

2.1 Grundlagen

Wie Eingangs schon erwähnt, handelt es sich bei PHP um eine Scriptsprache die hauptsächlich zur Generierung von dynamischen Webseiten genutzt wird. Syntaktisch hauptsächlich an C angelehnt, ist PHP recht leicht zu erlernen, auch für Leute, die bisher noch nicht viel programmiert haben. Ob nun ein Kontaktformular dessen Daten an eine bestimmte eMail-Adresse geschickt werden sollen oder gleich ein komplett datenbankbasiertes Forum – mit PHP lässt sich einiges an „Leben“ in eine Webseite bringen.

Durch eine Vielzahl an Modulen und Datenbankschnittstellen lässt sich PHP recht einfach für die unterschiedlichsten Webanwendungen nutzen (siehe nächstes Kapitel).

Darüber hinaus existieren mittlerweile eine große Anzahl fertiger Webanwendungen (meist OpenSource), die man nutzen kann.

Das klassische erste Beispiel für Programmiersprachen ist „Hallo Welt“ nachfolgend in PHP. Dabei ist das kleine PHP-Script in HTML eingebettet:

```
---- snip ---
<html>
        <head>
                <title>Testprogramm</title>
        </head>
        <body>
<?
// begin des php-blocks

        echo „Hallo Welt;“

// ende des php-blocks
?>
        </body>
</html>
---- snip ---
```

2.2 Was kann PHP

PHP kann auf unterschiedliche Art und Weise in Webanwendungen zum Einsatz kommen:

- Ausführen serverseitiger Skripte (man benötigt den PHP-Parser, einen Webserver und einen Browser).
- Ausführen von Skripten auf Kommandozeilenebene (man benötigt nur den PHP-Parser, vergleichbar mit BATCH-Dateien unter Win/Taskscheduler bzw. unter UNIX/LINUX sogenannte cron-jobs).
- Schreiben von clientseitigen GUI[4]-Applikationen. (Dafür ist PHP nicht das erste Mittel der Wahl. Lässt sich mit sehr guten Programmierungskenntnissen aber auch dafür einsetzen.)

PHP ist plattformunabhängig und kann sowohl unter UNIX/LINUX und Derivaten als auch unter Windows, MAC , RISC und OS/2 genutzt werden. Außerdem unterstützt PHP die gebräuchlisten Webserver wie APACHE, MS IIS, Personal Web Server, Netscape und iPlanet Server, O'Reilly Website Pro Server, Caudium, Xitami, OmniHTTPd, und viele andere.

PHP unterstützt eine Vielzahl von Datenbanken, wie aus nachfolgender Auflistung deutlich wird:

- Adabas D
- dBASE
- Empress
- FilePro (nur Lesezugriff)
- Hyperwave
- IBM DB2
- Informix
- Ingres
- Interbase
- FrontBase
- MSQL (mini-SQL)
- Direct MS-SQL
- MySQL
- ODBC[5]
- ORACLE
- Ovrimos
- PostgreSQL
- Solid
- Sybase
- Velocis
- Unix dbm

PHP unterstützt u.a. die Kommunikation mit folgenden Services:

---

[4] **GUI** = Graphical User Interface (grafische Benutzerschnittstelle): die einem PC-Benutzer die Interaktion mit dem PC über grafische Elemente, wie Symbole, Schreibtisch (Desktop), Papierkorb und Maus, ermöglicht.

[5] **ODBC=** Open Database Connection Standard: Damit lassen sich alle Datenbanken, die zu diesem Weltstandard gehören ansprechen.

- LDAP: Lightweight Directory Access Protocol, ein Netzwerkprotokoll für Abfrage und Modifikation von Informationen eines Verzeichnisdienstes, z.B. hierarchische Benutzerverwaltrung in großen LANs, Campusnetzen.
  (Weitere Erläuterungen siehe hier: http://de.wikipedia.org/wiki/LDAP)

- IMAP: Internet Message Access Protocol, Zugriff auf und die Verwaltung von E-Mails auf einem (Web)Server, wobei die Mails auf dem Server verbleiben und nur bei Bedarf an den Client übermittelt werden.
  (Weitere Erläuterungen siehe hier: http://de.wikipedia.org/wiki/IMAP)

- POP3: Post Office Protocol Version 3 ist ein Übertragungsprotokoll für einen Client, um Mails von einem E-Mail-Server abzuholen .
  (Weitere Erläuterungen siehe hier: http://de.wikipedia.org/wiki/POP3)

- HTTP: Hypertext Transfer Protocol zum Übertragen von Daten, hauptsächlich um um Webseiten aus dem Inter-/Intranet in einen Browser zu laden.
  (Weitere Erläuterungen siehe hier: http://de.wikipedia.org/wiki/HTTP)

- Netzwerksockets.

Dies sind nur einige Module, die PHP für die Programmierung zur Verfügung stellt.

2.3 Anwendungsmöglichkeiten

Die wohl häufigste Anwendungsform von PHP ist die eines Webfrontends[6] für eine (Internet-) Datenbank.

Nehmen wir den Bibliotheksbereich, so lässt sich mittels PHP und einer SQL-Datenbank z.B. die Recherche für Nutzer in der Bestandsdatenbank realisieren. Der Nutzer bekommt definierte Eingabemasken zur Recherche im Bestand zur Verfügung und greift damit auf denselben Datenbestand zurück, den die Bibliothek zur Erfassung aller Daten benutzt. Durch eine Rechteverwaltung im Hintergrund (PHP) kann der Nutzer nur auf für ihn erlaubte Daten lesend zugreifen.

Ebenso können auf diese Weise Bibliotheksmitarbeiter die Datenbasis pflegen, ohne über Kenntnisse von SQL zu verfügen oder gar direkt auf Datenbankebene arbeiten zu müssen. Es

[6] Ein **Webfrontend** ist der Teil einer Internetanwendung, der für den Benutzer über seinen Internet-Browser sichtbar ist. Dazu sind, im Gegensatz zu klassischen Clientprogrammen, keine Installationen auf dem Benutzer-PC erforderlich. Die Kommunikation mit dem Server erfolgt über das http bzw. HTTPS-Protokoll.

gibt einen definierten Zugriff auf die Datenbank auf der einen Seite und eine einfache Benutzerführung auf der anderen Seite.

Weitere Anwendungsbeispiele wären z.B. ein Terminkalender für die Bibliothekswebseite – damit könn(t)en ebenfalls datenbankgestützt per Web-Frontend die Mitarbeiter ohne Kenntnisse von HTML, PHP und SQL den Terminkalender über ein PHP-basiertes Eingabeformular pflegen und die Nutzer der Bibliothek sehen auf der Webseite immer einen aktuellen Terminkalender und Hinweise zu Veranstaltungen.

Mit klassischem (statischem) HTML müsste **für jede neue** Veranstaltung die Seite des Kalenders bearbeitet werden und der Mitarbeiter, der die Seite pflegen muss, braucht zwingend Kenntnisse in HTML. Mit PHP wird einmal ein entsprechendes Eingabe-Formular geschrieben mit dem die Daten gepflegt werden sollen. Die Pflege der Inhalte über das Formular, kann dann jeder Mitarbeiter vornehmen. Erforderlich dafür ist nur ein Internet-Browser.

Darüber hinaus kann es noch zahlreiche weitere Anwendungsmöglichkeiten geben, die im Arbeitsalltag die Pflege und Aktualisierung von Informationen bzw. Datenbeständen notwendig machen.

2.4 Der Terminkalender – ein Anwendungsszenario

An dieser Stelle soll das zuvor erwähnte Beispiel des Terminkalenders noch einmal aufgegriffen und etwas ausführlicher erläutert werden.

Dazu benötigen Sie zuerst einen Webserver sowie eine Datenbank. Am häufigsten kommen hier sogenannte **XAMP(P)**[7]-Systeme zum Einsatz. Dabei ist es unerheblich, ob es sich um gemieteten Webspace mit PHP und Datenbank bei einem Provider oder um einen eigenen Server handelt. XAMP(P) ist eine freie Distribution zur lokalen Installation und Konfiguration des Webservers *Apache*, der Datenbank *MySQ*L (oder SQLite), den Skritpsprachen *PHP* und *Perl*. Außerdem sind in diesem Paket weitere nützliche Werkzeuge wie der **FTP**[8]**-** Server *File-*

[7] **X** = Platzhalter für Windows oder Linux, **A** = Apache (der Webserver,) **M** = MySQL (der Datenbankserver), **P** = PHP (die Scriptsprache); auch WAMP oder LAMP, wenn das Betriebssystem vorgegeben ist. Gelegentlich sieht man auch XAMPP, WAMPP oder LAMPP in diesen Fällen steht das **letzte P** für Perl (eine weitere Scriptsprache).

[8] **FTP** = File Transfer Protocol ist ein spezifiziertes Netzwerkprotokoll zur Dateiübertragung in TCP/IP-Netzwerken. Es wird benutzt, um Dateien vom Client (lokal) auf den Webserver zu übertragen. (Weitere Erläuterungen siehe hier: http://de.wikipedia.org/wiki/File_Transfer_Protocol)

Zilla Server, der **Mailserver** *Mercury*, *phpMyAdmin*[9], *Webalize*r (ein Tool zum Analysieren der Logfiles des Webservers), *OpenSS*L (OpenSource SSL = Secure Socket Layer für verschlüsselte Verbindungen zum Webserver sowie Verwaltung von Zertifikaten (Weitere Erläuterungen siehe: http://www.openssl.org/) und *Python* (eine Programmiersprache) enthalten.

Wichtig sind die Zugangsdaten für FTP und Datenbank, um die eigenen (oder freien) PHP-Skripte auf dem Webspace zu hinterlegen, ggf. ändern zu können, und um Zugriff auf die Datenbank zu erhalten.

Sie sollten sich als erstes um die Datenbankverwaltung kümmern. Die meisten gemieteten Webspace-Pakete lassen keinen externen Datenbankzugriff zu, daher empfiehlt sich eine PHP-basierte Lösung (z.B. *phpMyAdmin*), die direkt auf dem Server installiert werden kann. Hierüber können Sie dann alle Datenbankfunktionen (Tabellen definieren, ändern, Verknüpfungen zwischen den Tabellen, Daten hinzufügen, ändern, löschen usw.) handhaben.

Zuerst sollten alle benötigten Tabellen (z.B. für den Terminkalender) in der Datenbank eingerichtet werden. Um die Programmierung der benötigten Formulare und aller anderen Funktionen für die Datenbank zu erleichtern, empfiehlt es sich, bereits zu Beginn ein paar Testdatensätze von Hand einzugeben.

Danach sollten Sie das Modul für die Datenpflege (neue Daten erfassen, bestehende Daten ergänzen/ändern, alte Daten löschen) erstellen und zur Verfügung stellen. Wenn Sie damit fertig sind, können Ihre Kollegen/Innen den Datenbestand bereits einpflegen, ohne dass Programmier-, Datenbank und/oder HTML-Kenntnisse benötigt werden.

Zum Schluss erfolgt die Einbindung der Daten in Ihre Homepage selber. Sie können mittels PHP alle benötigten / gewünschten Daten dynamisch aus der Datenbank auslesen und dem Anwender (Mitarbeiter/innen, Leser) auf der Homepage zur Verfügung stellen.

Sind diese Schritte einmal abgearbeitet, getestet und funktionieren, kann jeder Mitarbeiter den Terminkalender über das Datenpflegemodul aktuell halten. Alle benötigten Eingabeformulare sind über den Browser verfügbar.

Die Programmroutine im Hintergrund sorgt dafür, dass neue Daten in die entsprechende(n) Tabelle(n) in die Datenbank geschrieben, geänderte Daten zurückgeschrieben und nicht mehr benötigte Daten gelöscht werden.

---

[9] Webbasierte OpenSource Verwaltungssoftware für MySQL (http://www.phpmyadmin.net/) - in PHP geschrieben

Die angezeigten Daten auf der Webseite werden **nach** Anfrage aus der Datenbank **direkt** bei Aufruf der Seite generiert und sind somit **immer aktuell**.

Zur Erinnerung:

Bei statischen (also feststehenden) HTML-Seiten muss einerseits das Layout für jede Seite des Terminkalenders manuell erstellt werden. Für jeden neuen Termin, den Sie ergänzen wollen, müssten Sie diese manuell in einem HTML-Editor bearbeiten und dann via FTP auf dem Server aktualisieren. Gleichzeitig müssten Sie alle Änderungen an bestehenden Terminen manuell durchführen und veraltete Termine von Hand löschen. Um das alles tun zu können sind **HTML-Kenntnisse zwingend erforderlich.**

Durch die Nutzung von **dynamischen HTML-Seiten**, die auf Anfrage **aktuell direkt** aus Ihrer Datenbank generiert werden, **minimiert** sich der **Aufwand**, da Sie das System nur einmal implementieren müssen. Danach kann jede/r Mitarbeiter/in bzw. auch der Azubi oder Praktikant über einen Internetbrowser den Terminkalender bedienen und pflegen, **ohne** über umfassende **Programmier**-, **Datenbank**- und/oder **HTML-Kenntnisse** verfügen zu müssen.

Im Bereich PHP/MySQL gibt es oftmals die Möglichkeit, auf fertige Module und/oder Skripte zurückgreifen zu können. Nachfolgend sollen dafür einige Beispiele genannt werden, ohne Anspruch auf Vollständigkeit zu erheben.
Auch die Qualität der angebotenen Module kann nicht beurteilt werden. Natürlich müssen diese vorgefertigten Skripte, Templates (Vorlagen) und Module immer noch in die eigene Web-Präsenz eingebunden und ggf. angepasst werden.

Diverse **Webtemplates und Skripte** können Sie u.a. hier finden:

- Inspire-World: http://www.inspire-world.de/
- Jürgens Workshops: http://www.juergens-workshops.de/

PHP-Ressourcen, u.a. auch Kalender, lassen sich hier finden:

- Kostenlose PHP-Scripts: http://php-free.de/
- Diverse PHP-Ressourcen: http://www.php-resource.de/index.php
- Dr. Web - Spickzettel Sammlung: http://www.drweb.de/weblog/weblog/?p=571

3. MySQL

3.1 Relationales Dantenbanksystem / Datenbankmodell (kurze Einführung)

Relationale Datenbanken basieren auf Tabellen die redundanzarm über Primär- und Fremdschlüssel in Beziehung (Relation) zueinander stehen.

Der Primärschlüssel innerhalb einer Tabelle ist ein **eineindeutiger** Wert für **jeden** Datensatz, der nur **einmal** vorkommen darf.

Der Fremdschlüssel stellt die Beziehung zu einem Datensatz einer anderen Tabelle dar, der in dieser Tabelle wiederum einen Primärschlüssel besitzt. Fremdschlüssel müssen nicht eindeutig sein, sondern können mehrmals vorkommen.

Mit dem nachfolgenden, bewusst verkürzten, Beispiel aus dem Bibliothekswesen für eine Tabelle mit redundanten Informationen soll die Problematik noch einmal verdeutlicht werden:

Titel	***ISBN***	***Anschaffungsjahr***	***Bestand***
Das Einsteigerseminar PHP	3-8287-1110-3	2000	*Lesesaal*
Das Einsteigerseminar PHP	3-8287-1110-3	2001	*Freihand*
Das Einsteigerseminar MySQL	3-8266-7021-3	2001	*Lesesaal*
Das Einsteigerseminar MySQL	3-8266-7021-3	2002	*Freihand*

Im oben gezeigten Beispiel werden für **jedes Exemplar eines Buches alle Daten** des Titels komplett erfasst. Man kann hier zwar mit einem Blick in die Tabelle alle Informationen sofort ersehen – aber bei großen Datenbeständen (> 1.000 Titelaufnahmen zzgl. Exemplardatensätzen) ist das mit folgenden Nachteilen verbunden:

- Höherer Speicherbedarf.
- Steigende Fehlerquellen bei Änderungen.
- Unübersichtlichkeit.
- Lange Antwortzeiten bei Suchanfragen (da die gesamte Tabelle sequentiell, d.h. nacheinander von Datensatz 1 bis Datensatz n, durchsucht werden muss).

Ausgehend von diesem Eingangsbeispiel soll nachfolgend versucht werden, die gegebenen Informationen in einem relationalen Datenbankmodell darzustellen.

Es gibt Mediendaten (Bücher, Zeitschriften, CDs, DVDs usw. = Titelaufnahme) mit ihren Eigenschaften (Attributen) wie z.B.: Titel, Autor, ISBN, Erscheinungsjahr und –ort, Umfangsangaben usw. Zu jedem Medium gibt es ein bis mehrere Exemplare (Exemplardatensätze), die ggf. in unterschiedlichen Jahren angeschafft worden sein können.

Zur Vermeidung von Redundanz werden zwei Tabellen **MEDIEN** und **EXEMPLAR** angelegt, die zueinander in Beziehung stehen:

Medien_ID[10]	***Titel***	***Autor***	***ISBN***
1	Das Einsteigerseminar PHP	Wigard, Susanne	3-8287-1110-3
2	Das Einsteigerseminar MySQL	Däßler, Rolf	3-8266-7021-3

In der Tabelle **MEDIEN** bekommt jedes Medium neben seinen Eigenschaften wie Titel, Autor, ISBN, etc. noch eine **eineindeutige *Medien-ID*** (Primärschlüssel) zugewiesen.

Der Primärschlüssel aus der Tabelle **MEDIEN** dient in diesem Beispiel dazu, um in der Tabelle **EXEMPLAR** die Verknüpfung vom Exemplar zum Medium herzustellen.

Exemplar_ID	***Medien_ID***	***Anschaffungsjahr***	***Standort***
1	1	2000	Lesesaal
2	1	2001	Freihand
3	2	2001	Lesesaal
4	2	2002	Freihand

Auch die Tabelle **EXEMPLAR** verfügt mit dem Feld ***Exemplar-ID*** über einen eindeutigen Primärschlüssel. Das Feld ***Medien-ID*** ist hier Fremdschlüssel, welcher auf die Datensätze in der Tabelle **MEDIEN** mit dem angegebenen Primärschlüssel verweist

In diesen beiden Tabellen finden sich nun alle Informationen des zuvor skizzierten Ausgangbeispiels, nur diesmal in einer relationalen Tabellenstruktur mit den entsprechenden Verknüpfungen und demzufolge weniger Redundanz.

Anstatt für **jedes** Exemplar **eines** Buch alle benötigten Informationen (einschließlich der Titelangaben) immer wieder zu erfassen, wird nur auf die entsprechende Datensatznummer der Tabelle **MEDIEN** verwiesen. Die Beziehung der Tabelle **MEDIEN** zu der Tabelle **EXEMPLAR** nennt man eine ***1:n-Beziehung***, d.h. zu einer Titelaufnahme eines Mediums können ein oder mehrere Exemplare existieren.

---

[10] **ID= Ident-Nummer** (eindeutige Nummer zum Identifizieren des Datensatzes und zur Darstellung korrekter Relationen)

Im Bibliotheksalltag sind darüber hinaus aber noch viele weitere Informationen notwendig: Benutzerdaten (Leser der Bibliothek), Ausleihdaten (WER, WAS, WANN, WIE LANGE, Vormerkungen, usw.), Erwerbungsdaten (Bestellvorgänge), Angaben zu Zeitschriftenabos und –umläufen u.v.m.

Es wird deutlich, dass ein EDV-gestütztes Bibliothekssystem, egal ob klassisch als Client-Server-Programm oder als Web-basierte Datenbanklösung, sehr komplex ist und eine Vielzahl von Tabellen zur Informationsdarstellung benötigt, die auch in unterschiedlichen Relationen zueinander stehen können.

3.2 Leistungsfähigkeit von MySQL

Der MySQL-Datenbankserver unterstützt beliebig viele Datenbanken. Jede Datenbank kann beliebig viele Tabellen in beliebiger Größe, nur beschränkt durch das Betriebssystem, enthalten. MySQL verfügt im Bereich der Datenbankmanagementsysteme[11] (DBMS) über die einzigartige Möglichkeit verschiedene Tabellentypen (s. hierzu auch weiterführende Erläuterungen unter: http://de.wikipedia.org/wiki/MySQL) zu verwenden. Gerade diese Funktion macht MySQL zu einem sehr vielseitigen relationalen DBMS, das sich hervorragend an spezielle Anforderungen anpassen lässt. Deshalb wird MySQL vielfach im Bereich von Webservern eingesetzt, i.d.R. unter der freien Lizenz, was für Provider verlockend ist.

Durch das robuste Replikationssystem[12] (seit Vers. 3.22 xx) ist eine extreme Skalierung[13] möglich, die MySQL auch für große Webdienste interessant machen. Diese betreiben oftmals zahlreiche MySQL-Server, um damit beliebige Lasten bedienen zu können.

---

[11] Ein DBMS ist Teil eines Datenbanksystems (DBS), das aus einer Verwaltungssoftware dem Datenbankmanagementsystem (DBMS) und aus dem eigentlichen Datenspeicher, der Datenbank, besteht. Die wesentliche Aufgabe eines DBS besteht darin, große Datenmnengen sicher zu speichen und für Abfragen durch Benutzer bzw. die Anwendungssoftware bereitzustellen. (s.a.: http://de.wikipedia.org/wiki/Datenbanksystem)

[12] Unter einem **Replikationssystem** versteht man in diesem Zusammenhang das Anlegen von Kopien von vorhandenen Datenbanken, Aktualisierung und/oder Synchronisation von großen Datenmengen auch zwischen unterschiedlichen Anwendungen und/oder Plattformen.

[13] In der Informatik bezeichnet **Skalierung** das Verhalten von Programmen und/oder Algorithmen bezüglich ihres Ressourcenbedarfs bei wachsenden Eingabemengen (Datenmengen). Darunter versteht man das Zugriffsverhalten von Datenbanken bei wachsender Größe, Datenmenge und Komplexität und steht für die Anpassungsfähigkeit einer Software bei Änderung der Anforderungen.

3.3 Anwendungsmöglichkeiten

Aus den zuvor genannten Fakten lassen sich MySQL-Datenbanken für ganz unterschiedliche Zwecke einsetzen. Vor allem im Bereich der Verwaltung großer Datenmengen und darauf basierender dynamischer Gestaltung von Internetseiten liegen Hauptanwendungsgebiete für MySQL-Datenbanken.

Das können vor allem solche Anwendungen sein:

- Adressdatenbanken
- Web-basierte Dienstleistungen (freie Email-Accounts, Fotoalben, Linksammlungen (My-Web bei Yahoo) u.ä.)
- Termin- und/oder Veranstaltungskalender
- Bibliothekskataloge (z.B. KVK und XOPAC der Uni Karlsruhe, OPACs der Bibliotheksverbünde, OPACS von Bibliotheken)
- Shop-Anwendungen / Online-Shops (Amazon.de, BOL.de, Versandhandel im Internet u.ä.)

4. Links und Praxisbeispiele

4.1 PHP und MySQL-Beispiele

4.1.1 PHP-Einsatz bei den Hompages der Autoren

Am Beispiel des Mail-Scriptes für das Kontaktformular auf Homepage der Autoren soll überblicksmäßig darauf eingegangen werden, wie man mit PHP dynamische Seiten gestalten kann. Zuerst wird der komplette Quelltext zur Übersicht angezeigt.

```
<?
include  'inc/environment.inc.php';
htmlKopf('Über uns &#187; Kontakt', 'mail');

echo "
<table width='100%' height='100%'>
<tr><td align='center' valign='middle'>
   <h2>Kontakt</h2>
   <br>
   <table><tr><td>\n";

// ---------------------------------------------------------

$sErrorText = "";

$sDatum   = date("Y-m-d H:i:s", time());         // datum
$sIP      = getenv("REMOTE_ADDR");               // ip
$sAgent   = getenv("HTTP_USER_AGENT");           // browser

// per formular (methode='post') uebergebene variabeln
$sEmail   = $_POST["email"];
$sSite    = $_POST["site"];
$sName    = $_POST["name"];
$sFirma   = $_POST["firma"];
$sAnrede  = $_POST["anrede"];
$sTelefon = $_POST["telefon"];
$sText    = $_POST["text"];

if (isset($sEmail)     &&
   trim( $sEmail) != "" &&
   ($sErr = mailcheck($sEmail)) != 1) {
   $sErrorText .= $sErr;
} elseif (trim($sEmail) == "") {
   $sErrorText .= "Bitte tragen Sie Ihre E-Mail-Adresse ein !<br>\n";
}

if (trim($sName) == "") $sErrorText .= "Der Name fehlt !<br>\n";

// sind die uebergebenen daten fehlerhaft ?!
if (trim($sErrorText) != "") {
   msg( $sErrorText, "Fehler");
} else {
```

```
43    // eigentlicher mailtext
44    $sContent = "Datum/Zeit:\t $sDatum\n".
45                "IP:\t\t $sIP\n".
46                "Browser:\t $sAgent\n\n".
47                "Firma:\t $sFirma\n".
48                "Kontakt:\t $sAnrede $sName\n".
49                "Telefon:\t $sTelefon\n".
50                "Mail:\t\? $sEmail\n\n".
60                "------\n\n".
61                "$sText";
62
63    // mail-header
64    $sHeader  = "From: ".$aCFG["sender"]."\n";
65    $sHeader .= "Return-Path: ".$aCFG["sender"]."\n";
66    $sHeader .= "Reply-To: $sEmail";
67
68    // betreff / subject
69    $sSubject = "web-anfrage: $sSite";
70
71    // mailversand
72    $iErg = mail($aCFG["sender"], $sSubject, $sContent, $sHeader);
73
74    // wurde mail erfolgreich verschickt ?!
75    if (!$iErg) {
76          msg("Die Mail konnte nicht versandt werden!", "Fehler");
77    } else {
78          msg("Die Mail wurde erfolgreich versandt.", "Hinweis");
79    }
80  } // sErrorText leer ?
81
82  // ---------------------------------------------------------------
83
84  echo "
85     </td></tr></table>
86  </td></tr></table>\n";
87
88  htmlFuss();
89  ?>
```

Auf einige Passagen des oben abgebildeten Quelltextes soll näher im Detail eingegangen werden.

```
01  <?
02  include  'inc/environment.inc.php';
03  htmlKopf('Über uns &#187; Kontakt', 'mail');
...
88  htmlFuss();
89  ?>
```

In der Zeile 1 wird dem ***Parser*** [14]des Webservers durch die Zeichenfolge **<?** mitgeteilt, dass hier ein PHP-Quelltext beginnt, den er zu verarbeiten (parsen) hat. Das entsprechende Gegenstück **?>**, welches dem Parser das Ende eines PHP-Blockes signalisert, finden wir in Zeile 89.

Derartige PHP-Blöcke können in einer PHP-Datei mehrfach vorkommen. Inhalte außerhalb dieser Blöcke wertet der Webserver als normales HTML, sofern keine anderen Steuersequenzen etwas gegenteiliges veranlassen (z.B. die Interpretation als [Python[15]]-Skript o.ä.).

Über den Befehl ***include*** in Zeile 2 wird eine weitere PHP-Datei eingebunden. In diesem Fall sorgt die eingebundene Datei für die Sitzungsverwaltung[16] (session), das Einbinden der eigenen Funktionsbibliothek sowie zur Definition einiger wichtiger Variabeln.

htmlKopf() und ***htmlFuss()*** (Zeile **3** und **88**) sind z.B. zwei eigene Funktionen aus der in Zeile **2** eingebundenen **Funktionsbibliothek**. Sie sorgen für die einheitliche Darstellung des HTML-Grundlayouts. Zwischen diesen beiden Bereichen wird der eigentliche Inhalt der Seite dargestellt. Mit diesen fünf Zeilen bekommt **jede Seite das gleiche Layout** (inkl. Menü und (Besucher-)zähler).

```
05  echo "
06  <table width='100%' height='100%'>
07  <tr><td align='center' valign='middle'>
08     <h2>Kontakt</h2>
09     <br>
10     <table><tr><td>\n";
```

Mit ***echo*** lassen sich auf einfache Art und Weise HTML-Ausgaben generieren (hier ein Tabellen-Layout).

```
12  // ------------------------------------------------------
```

Mit **//** werden einzeilige Kommentare eingeleitet. Hier soll nur der eigentliche Seiten-Quelltext vom übrigen Code zur Seitengestaltung optisch abgegrenzt werden.

---

[14] Ein **Parser** ist ein Computerprogramm, das entscheidet, ob ein Eingabetext syntaktisch richtig formuliert wurde (im o.g. Beispiel der Quelltext). Wenn das der Fall ist, wird/werden die Anweisung(en) des Quelltexts interpretiert und ausgeführt. Bei fehlerhaften Anweisungen (syntaktisch falsch) wird eine Fehlermeldung ausgegeben und der Programmablauf unterbrochen. (s.a.: http://de.wikipedia.org/wiki/Parser; http://www.computerbase.de/lexikon/Parser

[15] objektorientierte Programmiersprache

[16] **Hier**: zeitweise bestehende Verbindung zwischen Webbrowser (und dessen gestellter Anfrage) und dem Webserver, der die Antwort auf diese Anfrage liefert

Nachfolgend die Quellcodezeilen zur Definition der für das Kontaktformular benötigten Variablen:

```
16  $sDatum    = date("Y-m-d H:i:s", time());          // datum / zeit
17  $sIP       = getenv("REMOTE_ADDR");                // ip
18  $sAgent    = getenv("HTTP_USER_AGENT");            // browser
```

time() liefert die Systemzeit und ***date()*** wandelt sie gemäß den Argumenten in eine speziell formatierte Zeichenkette um (hier: 2006-03-10 10:56:13).

getenv() liefert die angegebenen Umgebungsvariabeln des Webservers wieder.
REMOTE_ADDR enthält die IP-Adresse des Nutzers, der gerade auf die Seite zugreift, und ***HTTP_USER_AGENT*** gibt die Kennung des genutzen Browsers zurück.

```
21  $sEmail    = $_POST["email"];
22  $sSite     = $_POST["site"];
```

Über das globale ***System-Array***[17] ***$_POST*** wird auf die übergebenen Formulardaten zugegriffen. Gleichzeitig wird hiermit ausgeschlossen, dass der Nutzer über manipulierte Skriptaufrufe (z.B. durch zusätzliche GET-Parameter) die Möglichkeit erhält, Variabeleninhalte (absichtlich) zu verändern.

```
29  if (isset($sEmail)      &&
30     trim( $sEmail) != "" &&
31     ($sErr = mailcheck($sEmail)) != 1) {
32     $sErrorText .= $sErr;
33  } elseif (trim($sEmail) == "") {
34     $sErrorText .= "Bitte tragen Sie Ihre E-Mail-Adresse ein !<br>\n";
35  }
36
37  if (trim($sName) == "") $sErrorText .= "Der Name fehlt !<br>\n";
```

In diesem Abschnitt findet eine einfache Plausibilitätsprüfung der eingegebenen Daten statt. Hauptsächlich wird geprüft, ob die übergebenen Variablen nicht leer sind. Zusätzlich wird, wenn der Nutzer im Feld Email eine Emailadresse angegeben hat, geprüft, ob es sich um eine (zumindest syntaktisch name@domainname.TLD [TLD = Top Level Domian, z.B. de]) gültige Email-Adresse handelt. Die Überprüfung der Emailadresse übernimmt die Funktion ***mailcheck()***, welche ebenfalls aus der eingangs erwähnten Funktionsbibliothek stammt.

```
43     // eigentlicher mailtext
44     $sContent = "Datum/Zeit:\t $sDatum\n".
45                 "IP:\t\t $sIP\n".
```

[17] Ein Array dient zur Beschreibung von gleichartigen Datenstrukturen. Damit können Daten gleichen Datentyps so im Computer abgelegt werden, dass der Zugriff darauf über einen Index möglich wird. Das dient zur Beschleunigung des Datenzugriffs. (s.a. http://de.wikipedia.org/wiki/Array)

```
46              "Browser:\t $sAgent\n\n".
47              "Firma:\t $sFirma\n".
48              "Kontakt:\t $sAnrede $sName\n".
49              "Telefon:\t $sTelefon\n".
50              "Mail:\t\? $sEmail\n\n".
60              "------\n\n".
61              "$sText";
```

Der Variabeln ***sContent*** wird der Mailtext zugewiesen. Dieser wird aus den diversen übergebenen und im Skript ermittelten Variablen zusammengesetzt. Die Steuersequenzen **\n** (newline = Zeilen-umbruch) und **\t** (tabstop = Tabulator) dienen der Formatierung des Mailtextes. Der Punkt am jeweiligen Zeilenende verbindet/verkettet die Teilzeichenketten zu einer Zeichenkette.

```
63    // mail-header
64    $sHeader  = "From: ".$aCFG["sender"]."\n";
65    $sHeader .= "Return-Path: ".$aCFG["sender"]."\n";
66    $sHeader .= "Reply-To: $sEmail";
```

An dieser Stelle werden die Header-Daten (Kopfzeilen einer Email) für die zu sendende Email konfiguriert. In den Zeilen 65 und 66 kommt die Kurznotation **.=** zum Einsatz, damit wird an eine bestehende Zeichenkette eine weitere angehägt.

Das Array ***$aCFG*** stammt auch aus unserer Funktionsbibliothek und der Wert ***sender*** aus diesem Array hat die Email-Adresse hinterlegt, an die die Kontaktmails verschickt werden sollen. Das klingt kompliziert, vermeidet aber gleichzeitig, dass das vorhandene PHP-Skript **aktiv** zum Versenden von SPAM benutzt werden kann! Allerdings lässt sich damit nur SPAM gegen andere ausschließen, da die Zieladresse (Empfänger der Email) **nicht** vom Nutzer festgelegt werden kann. SPAM an die eigene Adresse kann damit nicht vermeiden werden.

```
71    // mailversand
72    $iErg = mail($aCFG["sender"], $sSubject, $sContent, $sHeader);
```

Die PHP-Funktion ***mail()*** übernimmt das Versenden der Email und liefert zurück, ob die Aktion erfolgreich war oder nicht. Dazu werden der Funktion der Empfänger, das Subject (Betreff) sowie der Mailtext und optional noch zusätzliche Email-Header-Informationen übergeben.

```
74    // wurde mail erfolgreich verschickt ?!
75    if (!$iErg) {
76          msg("Die Mail konnte nicht versandt werden!", "Fehler");
77    } else {
78          msg("Die Mail wurde erfolgreich versandt.", "Hinweis");
79    }
```

Anschließend wird noch geprüft, ob die Mail erfolgreich verschickt wurde, und dann abhängig vom Ergebnis entweder eine Fehlermeldung oder eine Sendebestätigung ausgegeben. ***msg()*** ist ebenfalls aus unserer Funktionsbibliothek und generiert eine Nachrichtenbox.

Der Nutzer sieht im gewohnten Seiten-Layout nach Benutzung des Kontakformulares (o. erläutertes Skript) lediglich einen Hinweis, ob der Mailversand erfolgreich war oder.

4.1.2 Datenbank-Beispiel: Linksammlung der Autoren

Die Linkverwaltung ist so konzipiert, dass sie (fast) komplett über das Web-Frontend genutzt und gepflegt werden. Jede vorhandene Tabelle verfügt über einen **eineindeutigen Schlüssel** (PK = Primary Key), über den die Relationen (Beziehungen) zu den anderen Tabellen realisiert werden.

Zur Linkverwaltung gehört ebenfalls eine kleine Nutzer- und Rechteverwaltung, um die Berechtigungen (WER darf WAS machen?) für unterschiedliche Anwender der Linksammlung festzulegen. Diese ist bewusst recht einfach, aber flexibel gehalten worden, damit sie einerseits die Anforderungen für die Linkverwaltung erfüllt, aber andererseits auch von anderen Modulen der Seite mitgenutzt werden kann.

Die Beziehungen zwischen den Tabellen werden im Anhang noch einmal an Hand einer Schemazeichnung dargestellt.

Die Nutzer- und Rechteverwaltung besteht aus den Tabellen **RECHT**, **NUTZER** und **NUTZER_RECHT**. In der Tabelle **RECHT** werden alle Rechte, die einem Benutzer zugeordnet werden können definiert. Die Tabelle **NUTZER** erfasst alle Benutzer, mit den erforderlichen persönlichen Angaben, die sich (mit unterschiedlichen Zugriffrechten) auf der Seite anmelden dürfen.

Tab. **RECHT**

PK[18]-Recht_ID	***Name***	***Beschreibung***
1	LINK_Lesen	Nur Lesender Zugriff
2	LINK_SchreibenEigne	Nutzer darf eigene Links verwalten
3	LINK_SchreibenAlle	Nutzer darf alle Links verwalten

[18] **PK =** Primary Key (Primärschlüssel, der **eineindeutig** sein muss)

Tab. **NUTZER**

PK_Nutzer_ID	***Login***	***Passwort***	***Name***	***eMail***
1	Gast	Gast	Gast	test@example.com
2	Nutzer1	geheim	Hans Meier	ich@hansmeier.de
3	Nutzer2	topsecret	Max Müller	max@mueller.com
4	Nutzer3	sagichnicht	Susi Sorglos	susi@sorglos.de

Um dem Benutzer nun sein(e) Recht(e) zuordnen zu können, wird die Verknüpfungstabelle **NUTZER_RECHT** benötigt. Diese legt fest, welcher Benutzer welche Rechte hat.
(Die Werte in den Klammern dienen nur zu Ihrer Orientierung und sind nicht Bestandteil der Tabelle.)

Tab. **NUTZER_RECHT**

PK_Nutzer_Recht_ID	***FK[19]_Nutzer_ID***	***FK_Recht_ID***
1	2 (Hans Meier)	1 (Link lesen)
2	3 (Max Müller)	1 (Link lesen)
3	3 (Max Müller)	2 (Link schreiben eigene)
4	4 (Susi Sorglos)	1 (Link lesen)
5	4 (Susi Sorglos)	3 (Link schreiben alle)

In den Tabellen der Linkverwaltung selbst werden die Nutzerrechte durch Fremdschlüssel auf die **NutzerID** (PK in der Tab. NUZTER, aber **immer Fremdschlüssel** in anderen Tabellen als NUTZER) abgebildet.

Die **Linkverwaltung** besteht aus drei Haupttabellen: **LINK**, **KATEGORIE** und **SUCHBEGRIFF**. Die Einträge der Tabellen Link und Kategorie sollen zusätzlich einen Status (öffentlich, Nutzer und privat) zugeordnet bekommen. Dazu wird noch eine weitere Tabelle **STATUS** angelegt.
Die Tabelle **STATUS** besteht nur aus dem Primärschlüssel sowie einem Feld Bezeichnung. Über diesen Wert wird im Web-Frontend die Verfügbarkeit der jeweiligen Links und Kategorien gesteuert Das Feld ***Erklärung*** dient nur zur Orientierung und ist nicht Bestandteil der Tabelle.

Über die Tabelle **STATUS** wird definiert, welcher Benutzer welche Berechtigungen innerhalb der Linkverwaltung (bzw. auch für die anderen Seitenfunktionen) hat.

[19] **FK =** Foreign Key (Fremdschlüssel, verweist über den Primärschlüssel auf einen Datensatz einer anderen Tabelle)

Die Tabelle **STATUS** sorgt dafür, dass der Benutzer entscheiden kann, welche seiner Links / Kategorien öffentlich (allen zugänglich), nur den angemeldeten Benutzer bzw. nur ihm selbst angezeigt werden. D.h. mit „privat“ gekennzeichnete Kategorien und/oder Links sind nur für diesen Benutzer sichtbar.

Tab. **STATUS**

PK_Status_ID	***Bezeichnung***	***Erklärung***
1	öffentlich	(auch ohne Login sichtbar)
2	Nutzer	(nur für angemeldete Nutzer sichtbar)
3	Privat	(nur für den Eigentümer sichtbar)

Die Kategorie wird mit einem Namen bezeichnet. Außerdem werden Angaben zum Eigentümer, dem (Kategorien-)Status und einer Rückreferenz (auf sich selber) erfasst. Diese Rückreferenz dient dazu, eine Baumstruktur zwischen Haupt – und Unterkategorien abbilden zu können.

Hauptkategorie 1 <NULL>
Unterkategorie 1.1´<ID der Hauptkategorie 1>
UnterUnterkategorie 1.1.1 <ID der Unterkategorie 1.1>
UnterUnterkategorie 1.1.2 <ID der Unterkatergorie 1.1>
... ...
Unterkategorie 1.2 <´<ID der Hauptkategorie 1>
Hauptkategorie 2 <NULL>

Tab. **KATEGORIE**

PK_Kategorie_ID	***Name***	***FK_Status_ID***	***FK_Nutzer_ID***	***FK_Parent_ID***
1	Bücher	1	1	NULL
2	CDs	1	2	NULL
3	Rock/Pop	1	2	2
4	Meine CD's	3	3	2

Das Feld ***FK_Parent_ID*** verweist auf den Primärschlüssel der jeweils übergeordneten Kategorie.

- NULL (= leer, keine übergeordnete Kategorie vorhanden) steht für eine Hauptkategorie (oberste Ebene)
- ('Rock/Pop' sowie 'Meine CD's' verweisen auf CDs (als übergeordnete Kategorie)

Die Tabelle **Link** erfasst alle Angaben zur Beschreibung eines Links: Kategorienebene (FK = Fremdschlüssel / Kategorienangabe aus der Tab. KATEGORIE), Titel, URL, Status (Anzeige-option aus Tab. STATUS), Nutzer (= Eigentümer).

Tab. **Link**

PK_Link_ID	*FK_Kategorie_ID*	*Titel*	*URL*	*FK_Status_ID*	*FK_Nutzer_ID*
1	1	MySQL	http://www.....	1	1
2	2	Hörbuch xyz	http://xyz.de	1	4
3	4	Meine Hits	http://hits.de/...	3	3

Außerdem können zu **jedem** Link beliebig viele Suchbegriffe erfasst werden. **Vorteil:** Links, die in mehrere Kategorien passen würden, brauchen nur einmal erfasst werden und können trotzdem an Hand des Suchbegriffes recherchiert werden. Die Tabelle **SUCHBEGRIFF** besteht aus einem Primärschlüssel, einem Fremdschlüssel auf den Link sowie dem Suchbegriff selber.

Tab. **SUCHBEGRIFF**

PK_Suchbegriff_ID	*FK_Link_ID*	*Suchbegriff*
1	1	EDV
2	1	Datenbank
3	1	SQL
4	3	Musik

Zur Linkverwaltung gehört ebenfalls eine Kategorienverwaltung, auf die hier aber nicht näher eingegangen werden soll. Diese ermöglicht die Pflege und Erfassung von Kategorien für die Linksammlung, inklusive der Zuordnung zu den darüberstehenden Kategorien.

Zusammengefasst wird durch diese (etwas komplexere) Datenbankstruktur Folgendes realisiert:

- Ein Link wird einer Kategorie zugeordnet, gehört einem Nutzer (Eigentümer), hat eine Bezeichnung, eine Link-URL und einen Status.
- Jedem Link können mehrere Suchbegriffe zugeordnet werden.
- Eine Kategorie hat eine Bezeichnung und gehört einem Nutzer.
- Eine untergeordnete Kategorie verweist auf eine übergeordnete Kategorie (Baumstruktur).

Inzwischen umfasst die Linksammlung mehr als 2.600 Links in über 340 Kategorien. Deshalb gibt es die Möglichkeit Links nicht nur an Hand der ausgewählten Kategorie zu recherchieren, sondern über eine Suchfunktion (nur Suchbegriffe, nur URL, nur Bezeichnung **ODER** alle 3 Optionen) relevante Links über **alle** ODER die **ausgewählte Kategorie** zu suchen.

Aus der im Anhang befindlichen Darstellung zum Aufbau der Datenbank für die Linkverwaltung wird deutlich, welche Tabellen verwendet werden und wie sie miteinander in Beziehung stehen.

Die dafür verwendeten Datentypen sollen hier vorab zum besseren Verständnis kurz erläutert werden:

- **integer(11)**
  Integer steht für eine ganze Zahl und die Zahl in der Klammer gibt die maximale Größe der Zahl an, die in diesem Feld verwaltet werden kann. In unserem Fall sind das 11 Byte (88 Bit), sodass wir $2^{88} – 1$ verschiedene Zahlenwerte verwalten können (das eine Bit, dass abgezogen wird dient, zur Unterscheidung von positiven und negativen Werten).
  Möglich sind also Zahlen von 0 bis 2^{87} sowie von -1 bis $-1 * 2^{87}$, das entspricht circa $-1{,}547 * 10^{26}$ bis $1{,}547 * 10^{26}$

- **varchar(n)**
  varchar steht für eine Zeichenkette variabler Länge. n definiert dabei die maximale Länge, möglich sind Werte zwischen 1 und 255 Zeichen.

4.2 Weiterführende Links [20]

4.2.1 PHP

- PHP-Handbuch deutsch: http://www.php.net/manual/de/preface.php
- PHP-Homepage.de!: http://www.php-homepage.de/
  Artikel, Scripte, Handbuch,FAQ, Forum …
- phpbox.de: http://www.phpbox.de/
  Anleitung, Referenz, Tutorial und Forum zu PHP
- artmedic freie PHP-Scripte: http://www.artmedic-phpscripts.de/index.php
  wie Fotoalbum, Diashow, Links, Kalender, Counter, Gästebuch u.v.m.

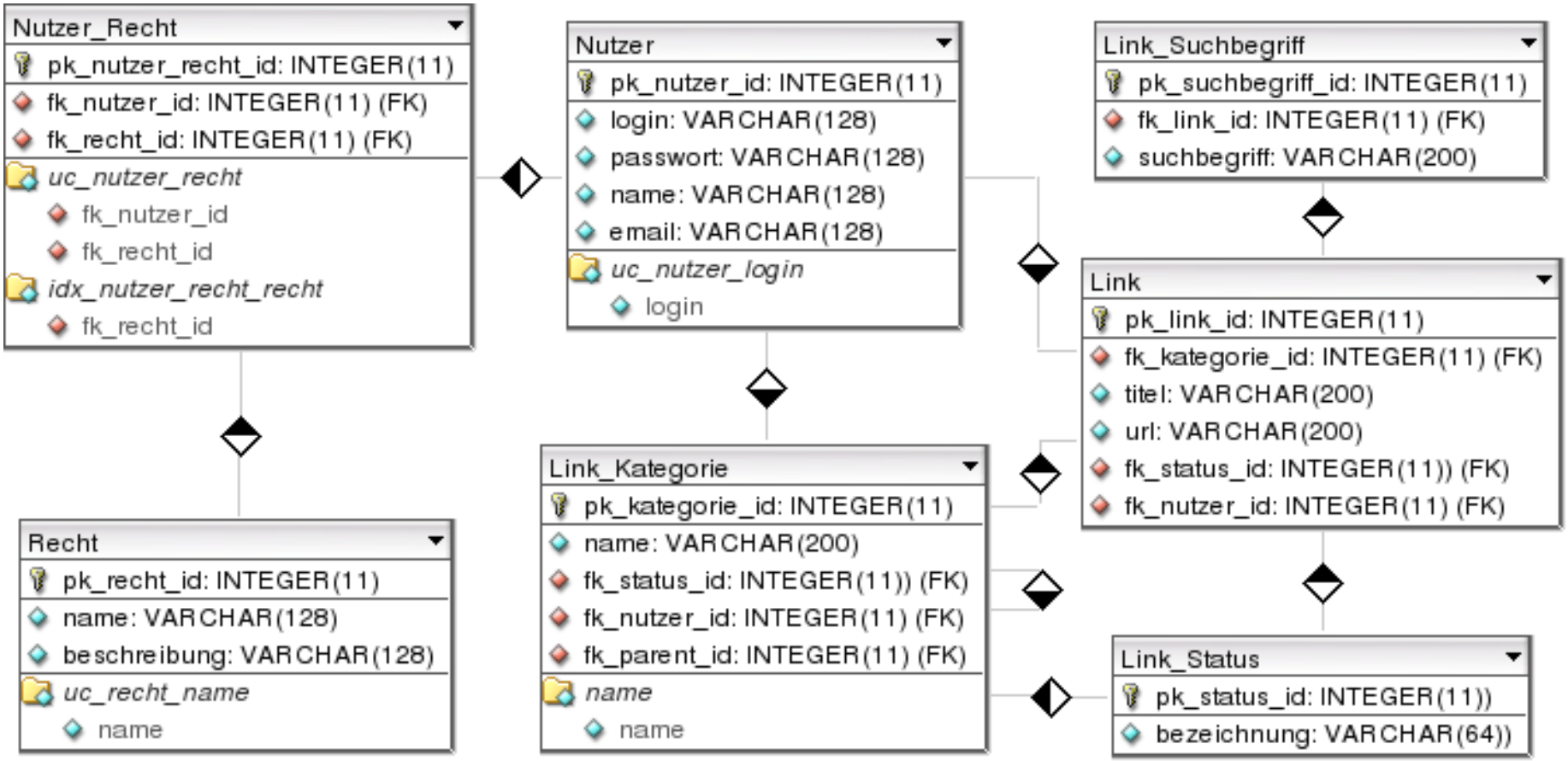
4.2.2 MySQL

- MySQL: Die populärste Open-Source-Datenbank der Welt: http://www.mysql.com/
  Downloads, Dokumentation u.v.m.
- PHP und MySQL kurz erklärt von einer 'Anfängerin': http://www.schattenbaum.net/php/
- Deutsche Mailingliste zu MySQL: http://www.4t2.com/mysql/
  Informations- und Diskussionsforum für Entwickler, Anwender und Interessierte...

4.2.3 Open Source – Bibliotheksprogramme

- Greenstone Digital Library Software:
  http://www.greenstone.org/cgi-bin/library?a=p&p=home
- OpenBiblio: webbasiertes Bibliothekssystem: http://openbiblio.de/
- PhpMylibrary - open source library automation package: http://www.phpmylibrary.org/
- Übersicht über Open Source-Bibliothekssysteme: http://www.oss4lib.org
- Eric Anctil: Open Source Integrated Library Systems (Überblick 2003):
  http://www.anctil.org/users/eric/oss4ils.html

[20] s.a. Linksammlung der Autoren: http://www.pieschel-berlin.de/link/index.php


Nutzer_Recht
pk_nutzer_recht_id: INTEGER(11)
fk_nutzer_id: INTEGER(11) (FK)
fk_recht_id: INTEGER(11) (FK)
uc_nutzer_recht
fk_nutzer_id
fk_recht_id
idx_nutzer_recht_recht
fk_recht_id
Nutzer
pk_nutzer_id: INTEGER(11)
login: VARCHAR(128)
passwort: VARCHAR(128)
name: VARCHAR(128)
email: VARCHAR(128)
uc_nutzer_login
login
Link_Suchbegriff
pk_suchbegriff_id: INTEGER(11)
fk_link_id: INTEGER(11) (FK)
suchbegriff: VARCHAR(200)
Link
pk_link_id: INTEGER(11)
fk_kategorie_id: INTEGER(11) (FK)
titel: VARCHAR(200)
url: VARCHAR(200)
fk_status_id: INTEGER(11)) (FK)
fk_nutzer_id: INTEGER(11) (FK)
Recht
pk_recht_id: INTEGER(11)
name: VARCHAR(128)
beschreibung: VARCHAR(128)
uc_recht_name
name
Link_Kategorie
pk_kategorie_id: INTEGER(11)
name: VARCHAR(200)
fk_status_id: INTEGER(11)) (FK)
fk_nutzer_id: INTEGER(11) (FK)
fk_parent_id: INTEGER(11) (FK)
name
name
Link_Status
pk_status_id: INTEGER(11))
bezeichnung: VARCHAR(64))

Berufsverband Information Bibliothek e.V.

Aufnahmeantrag

Bitte ausdrucken, ausfüllen und unterschrieben an die Geschäftsstelle senden.

☐ Frau ☐ Herr

Name: _____

Vorname(n): _____

Straße: _____

PLZ/Ort: _____

E-Mail: _____

Geburtsdatum: _____

Examen (Art): _____

Examen (Ort/Jahr): _____

Beschäftigungsort: _____

Arbeitsstelle: _____

Beschäftigt als: _____

Einstufung: _____

Abteilung: _____

☐ ganztags ☐ halbtags ☐ Ausbildung ☐ nicht (mehr) berufstätig

Examen voraussichtlich: _____

Mit der Speicherung meiner Adresse und der Verwendung für die satzungsgemäßen Zwecke des Vereins sowie den Vorstand der Zeitschrift BuB bin ich einverstanden.

Ort und Datum: _____

Unterschrift: _____

Ich bevollmächtige der Berufsverband Information Bibliothek e.V. bis auf Widerruf, den jährlichen Mitgliedsbeitrag in der von der Mitgliederversammlung festgesetzten Höhe ab 20____ abzubuchen.

Name: _____

Adresse: _____

Konto-Nr.: _____

Bankleitzahl: _____

Name der Bank, Ort: _____

Ort und Datum: _____

Unterschrift: _____

Geschäftsstelle: Berufsverband Information Bibliothek · Gartenstr. 18 · 72764 Reutlingen
Tel.: 0 71 21 / 34 91-0, Fax: 0 71 21 / 30 04 33 · E-Mail: mail@bib-info.de
Bankverbindung: Volksbank Reutlingen · BLZ 640 901 00 · Konto 159 336 007