Datenbanken in vernetzten Systemen mit PHP3 und mySQL

geschrieben von Uwe Debacher 2000

Im folgenden Text soll es darum gehen, wie man mit der Programmierumgebung PHP/FI und dem Datenbanksystem mySQL Datenbankanwendungen innerhalb eines Intranetsystems erstellen kann.

Es müssen folgende Bedingungen erfüllt sein:

1. Es läuft der Webserver Apache Version 1.3 (http://www.apache.de)
2. In den Apache wurden die PHP/FI 3.0 Module eingebunden (http://www.php3.de)
3. Das Datenbanksystem mySQL 3 wurde installiert und gestartet (http://www.mysql.com)

Ein Problem besteht darin, dass hier mit drei verschiedenen Programmen gearbeitet wird, die jeweils ihre eigene Programmiersprache besitzen. Die Befehle müssen aber alle in den gleichen Programmtext eingebunden sein (ein vernetztes System).

Grundlage ist immer eine HTML-Seite. Es ist deshalb wichtig zu beachten, an wen sich der jeweilige Programmierbefehl jeweils richtet.

1. Formulare in HTML

Im Text wollen wir nicht auf die Grundlagen von HTML eingehen. Lediglich auf die Arbeit mit Formularen.

Alle Eingaben, die wir in den folgenden Abschnitten programmieren, erfolgen in ein HTML-Formular. Ein Formular besteht mindestens aus den folgenden Tags:

(Kursiv gesetzte Wörter sind Platzhalter für eigene Bezeichner)

```
<FORM ACTION="auswert.php3" METHOD="get">
 <INPUT TYPE="text" NAME="name" SIZE="Size" MAXLENGTH="Länge">
 <INPUT TYPE="submit" VALUE="Absenden">
 <INPUT TYPE="reset" VALUE="Verwerfen">
</FORM>
```

Mit dem Einleitungstag muss ein Programm angegeben werden, das die Eingabedaten auswertet, in diesem Fall ein Programm namens „auswert.php3". Zusätzlich muss angegeben werden, wie dieses Programm die Daten erhält. Dazu gibt es die Möglichkeiten „get" und „post". Bei der Methode „get" werden die Daten einfach an die URL des Auswertprogrammes angehängt:

```
 auswert.php3?name=Meier&vorname=Klaus
Bei der Methode „post" sieht man diese Daten nicht, da eine Art Dialog
erfolgt. Bei der Programmentwicklung ist die Methode „get" praktischer,
im endgültigen Programm ist „post" vorteilhafter.
```

Die zweite Zeile definiert ein Eingabefeld mit dem Namen „name" und der Größe „Size". Diese Angabe betrifft nur die Darstellung auf dem Bildschirm. Die Maximalzahl der Zeichen die eingegeben werden kann wird auf „Länge" festgelegt.

Normalerweise hat ein Formular natürlich mehr als ein Eingabefeld. Die einzelnen Felder werden dabei durch ihre Namen unterschieden.

Wichtig für jedes Formular ist auch ein „Knopf" zum Abschicken. Dazu dient der Typ „submit". Die Daten im Formular werden ausgelesen und an das Auswertprogramm übergeben. Üblich in Formularen ist auch ein „Resetknopf". Klickt man auf diesen Knopf, so werden alle Eingabefelder gelöscht.

1.1 Ergänzungen zu Formularen

Als „action" kann anstelle eines eigentlichen Auswertprogrammes auch eine Mail verschickt werden:

```
 <FORM ACTION="mailto:debacher@gyloh.hh.schule.de" METHOD="post">
```

Das Eingabefeld vom Typ „text" erlaubt auch die Angabe eines Wertes, der vorab in das Feld eingetragen wird.

```
 <INPUT TYPE="text" NAME="name" VALUE="Klausi">
```

Diese Möglichkeit ist besonders dann wichtig, wenn man mittels Formular vorhandene Datensätze ändern möchte.

Es stehen innerhalb eines Formulars nicht nur Eingabezeilen zu Verfügung.

1.2 Hidden

Eine Angabe der folgenden Art:

```
<INPUT TYPE="hidden" NAME="name" VALUE="Klausi">
```

bewirkt keinerlei Darstellung auf dem Bildschirm. Dieses Feld wird mit seinem Wert einfach nur an das Auswertprogramm übergeben, ohne dass der Benutzer Eingabe machen kann oder überhaupt etwas von diesem Feld bemerkt. Dieser Tag macht nur Sinn, wenn ein VALUE mit angegeben wird.

1.3 Password

Auch dies ist nur eine Variation des Types Text.

```
<INPUT TYPE="password" NAME="name">
```

Hierbei wird der Eingabetext nicht auf dem Bildschirm dargestellt.

1.4 Textarea

Mit diesem Tag wird ein Eingabefeld definiert, dessen Höhe und Breite festgelegt werden müssen

```
<TEXTAREA NAME="name" ROWS=Höhe
 COLS=Breite>
 Vorgabetext
</TEXTAREA>
```

Dem Eingabefeld kann man leider keine Maximalzahl von Zeichen übergeben. Höhe und Breite beziehen sich auf die Bildschirmdarstellung. Innerhalb des Bereiches kann gescrollt werden.

Vorgaben werden hier nicht mit dem Value-Tag eingetragen, sondern zwischen Anfangs- und Endtag gesetzt.

1.5 Auswahlfelder

Will man keine freie Eingabe zulassen, sondern dem Benutzer nur die Auswahl zwischen vorgegebenen Werten ermöglichen, dann bietet sich die folgende Kombination an:

```
<SELECT NAME="name" SIZE=Zeilen>
 <OPTION VALUE="wert">Beschreibungstext
 <OPTION SELECTED VALUE="wert">Beschreibungstext
  ...
</SELECT>
```

Hiermit stelle ich dem Benutzer ein Feld zur Verfügung, das mit einem Mausklick geöffnet wird und die angegebenen Einträge zur Auswahl stellt. Nachdem ein Eintrag angeklickt wurde schließt sich das Auswahlfenster wieder.

Ein Wert darf durch Zusatz von SELECTED als voreingestellt gekennzeichnet werden. Dieser Wert erscheint dann auch im geschlossenen Eingabefeld.

1.6 Checkboxen

Die Verwendung von Auswahlfeldern ist sehr platzsparend. Manchmal möchte man aber alle Optionen immer auf dem Bildschirm sehen, dann arbeitet man besser mit Checkboxen

```
<INPUT TYPE="checkbox" NAME="name1" VALUE="wert1">Text1<br>
<INPUT TYPE="checkbox" NAME="name2" VALUE="wert2">Text2<br>
```

Für jeden der Einträge taucht auf dem Bildschirm ein ankreuzbares Kästchen auf. Diese Kästchen sin voneinander unabhängig. Es können also z.B. alle Kästchen oder kein Kästchen angekreuzt sein. Lässt man den VALUE-Eintrag weg, so wird im Zweifelsfall der Wert „on" genommen.

1.7 Radiobuttons

Will man erreichen, dass immer nur eine Möglichkeit aus einer Auswahl markiert werden kann, dann arbeitet man besser mit Radiobuttons:

```
<INPUT TYPE="radio" NAME="name" VALUE="eintrag1">Text1<br>
<INPUT TYPE="radio" NAME="name" VALUE="eintrag2">Text2<br>
```

Hier haben alle Felder den gleichen Namen aber unterschiedliche Werte.

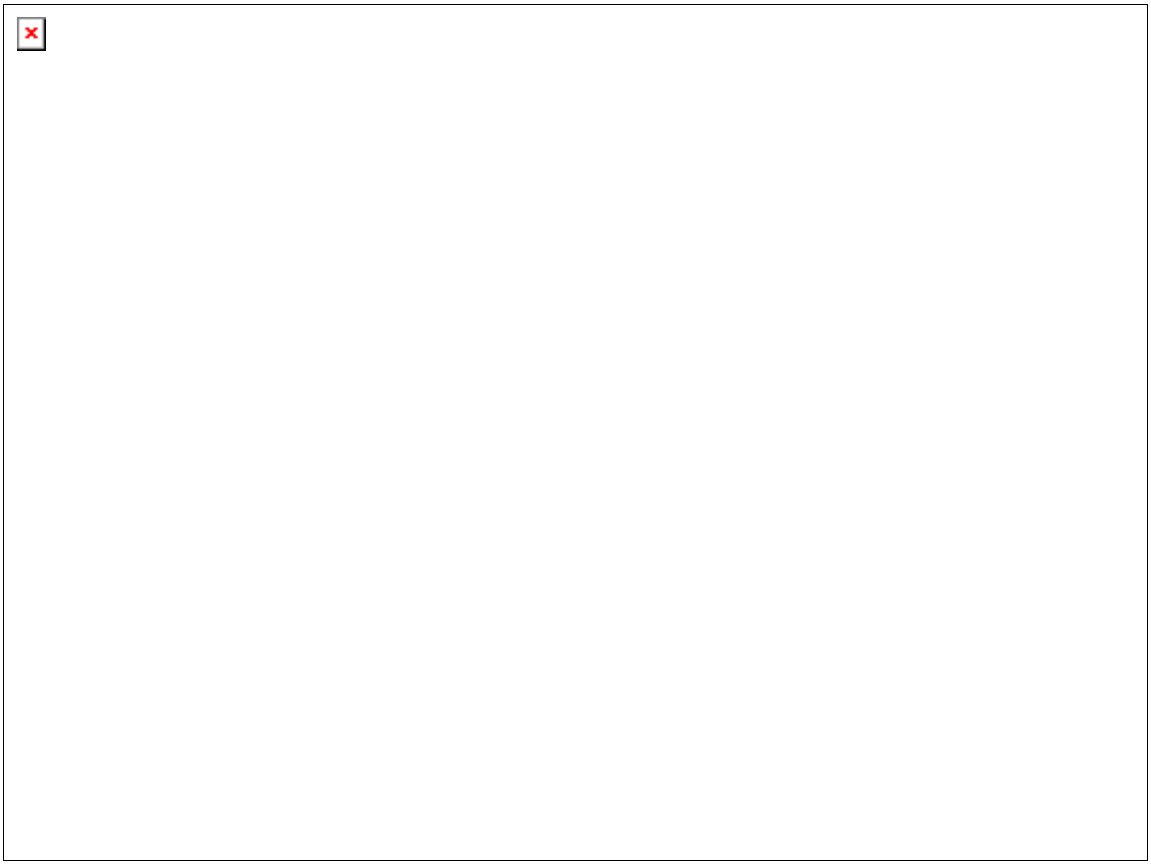
1.8 Ein kleines Beispiel

Das folgende Beilspiel kann als Einstieg in ein eigenes Gästebuch dienen. Konkret handelt es sich hier um das Eingabeformular:

telefonliste.htm

```
!-- Formular für eine Telefonliste von Uwe Debacher 2000-->
<HTML><HEAD><TITLE>Telefonliste</TITLE></HEAD><BODY>
<CENTER><H1>Telefonliste-Eingabeformular</H1></CENTER>
<FORM ACTION="telefon.php3" METHOD="get">
<!-- Ein normales Eingabefeld -->
Name<br>
<INPUT TYPE="text" NAME="name" SIZE=20 MAXLENGTH=50>
<p>Vorname<br>
<INPUT TYPE="text" NAME="vorname" SIZE=20 MAXLENGTH=50>
<p>Telefonnummer<br>
<INPUT TYPE="text" NAME="telefon" SIZE=20 MAXLENGTH=50>
<p><INPUT TYPE="submit" VALUE="Absenden">
<INPUT TYPE="reset"  VALUE="Verwerfen">
</FORM>
</BODY></HTML>
```

Lädt man die Seite in einen Browser, so ergibt sich folgendes Bild:



1.9 Ein umfangreiches Beispiel

Im folgenden Seitenquelltext sind fast alle Möglichkeiten berücksichtigt:

beispiel_gr.htm

```
<!-- Bespielformular von Uwe Debacher,letzte Aenderung am 18.03.2000 -->
<HTML><HEAD><TITLE>Beispielformular</TITLE></HEAD><BODY>
<CENTER><H1>Beispielformular</H1></CENTER>
<FORM ACTION="auswert.php3" METHOD="get">

<!-- Ein normales Eingabefeld -->
Wie hei&szlig;t du?<br>
<INPUT TYPE="text" NAME="name" SIZE=20 MAXLENGTH=50>

 <!-- Ein Textfeld -->
<p>Sage mir deine Meinung:<br>
<TEXTAREA NAME="meinung" ROWS=3 COLS=60>Ich habe keine eigene Meinung</TEXTAREA>

<!-- Ein Auswahlfeld -->
<p>Welches Verkehrsmittel benutzt du zur Schule?
<SELECT NAME="verkehrsm" SIZE=1>
 <OPTION VALUE="b">BUS
 <OPTION VALUE="a">Auto
 <OPTION SELECTED VALUE="f">Fahrrad
</SELECT>

<!-- Checkboxen -->
<p>Besuchst du einen oder mehrere der folgenden Leistungskurse?:<br>
<INPUT TYPE="checkbox" NAME="mathe">Mathematik
<INPUT TYPE="checkbox" CHECKED NAME="info">Informatik

<!-- Radiobuttons -->
```

```
<p>Wie gef&auml;llt dir diese Schule?:<br>
<INPUT TYPE="radio" NAME="urteil" VALUE="s">Sehr gut
<INPUT TYPE="radio" NAME="urteil" CHECKED VALUE="g">Gut
<INPUT TYPE="radio" NAME="urteil" VALUE="m">Mittel

<P><CENTER>
<INPUT TYPE="submit" VALUE="Absenden">
<INPUT TYPE="reset" VALUE="Verwerfen">
</CENTER>
</FORM>
</BODY></HTML>
```

Netscape macht daraus die folgende Darstellung:

Klickt man einfach nur auf „Absenden" ohne weitere Eingaben zu machen, so erscheint folgende URL in der Eingabezeile:

```
http://server/lk/auswert.php3?name=&meinung=Ich+habe+keine+eigene+Meinung&verkehr
```

Damit sieht man genau, welche Daten an das Auswertprogramm übergeben werden.

2. PHP/FI Grundlagen

Bei PHP handelt es sich im Prinzip um eine Erweiterung von HTML. Alle PHP-Dokumente werden

vom Apache-Webserver nach speziellen Programmier-Befehlen durchsucht. Gefundene Befehle werden ausgeführt. Der Benutzer bekommt immer eine normale HTML-Seite geliefert, da alle Befehle schon vom Webserver ausgeführt wurden.

Es ist also wichtig, dass jedes PHP-Programm immer eine korrekte HTML-Seite zurückliefert.

Eine PHP-Seite erkennt der Apache an der speziellen Endung .php3, man kann ihn aber auch so konfigurieren, dass er jede .htm Seite nach entsprechenden Befehlen durchsucht.

PHP-Befehle werden durch Fragezeichen kenntlich gemacht:

```
<? $a=5; ?>
```

Innerhalb der spitzen Klammern können mehrere (beliebig viele) Befehle stehen, die dann aber immer mit einem Semikolon beendet sein müssen:

```
<? $a="Hallo Welt";
echo $a; ?>
```

Der Echo-Befehl bewirkt eine Ausgabe an den Browser.

Dieses kurze Beispiel würde den Text „Hallo Welt" auf dem Bildschirm ausgeben. Dabei wird vorausgesetzt, dass um die beiden Zeilen herum der normale HTML-Rahmen vorhanden ist.

Die komplette Seite hat also folgenden Inhalt:

```
<HTML><HEAD><TITLE>Testprogramm</TITLE></HEAD><BODY>
<CENTER><H1>Testprogramm</H1></CENTER>
<?
$a="Hallo Welt";
echo $a;
?>
</BODY></HTML>
```

2.1 Variablen in PHP

Variablen werden durch ein vorangestelltes $ Zeichen gekennzeichnet. Es gibt drei Variablentypen:

- String $a="5"
- Real $a=5.0
- Integer $a=5

Die Typzuordnung erfolgt automatisch. Die Typzuordnung spielt aber keine sehr große Rolle.

Es gibt aber eine ganze Menge Besonderheiten im Zusammenhang mit Variablen.

```
$a="Hallo";
$$a="Welt";
```

Das doppelte $ hat die Bedeutung, dass der Wert von $a als Variablenbezeichner benutzt wird. Die zweite Zeile hat also den gleichen Effekt wie:

```
$Hallo="Welt";
```

Noch ungewöhnlicher ist der Umgang mit Arrays. Diese werden dynamisch und sehr flexibel

angelegt.

```
$a[0]="Hallo";
$a[1]="Welt";
```

legt ein Feld mit zwei Komponenten an, das aber jederzeit erweitert werden kann. Die Definition hätte man auch abkürzen können mittels:

```
$a[]="Hallo";
$a[]="Welt";
```

Den Rest denkt sich das System hinzu, wobei der Index jeweils hochgezählt wird.

Natürlich dürfen auch Strings als Index auftauchen:

```
$a["Hallo"]="Welt;
```

Zur Arbeit mit Arrays kennt php die Funktionen: Next(), Prev(), Reset(), End() und Key().

2.2 Bedingte Anweisungen

Für bedingte Anweisungen besitzt das System folgendes Konstrukt:

```
if (Bedingung):
 Befehle;
endif;
```

Folgende Variation hat den gleichen Effekt, ist aber weniger lesbar

```
if (Bedingung) {
 Befehle;
}
```

Auch eine zweiseitige Auswahl ist möglich:

```
if (Bedingung):
 Befehle;
else:
 Befehle
endif;
```

Will man noch mehr Fälle unterscheiden, so ist auch folgende Erweiterung vorhanden:

```
elseif (Bedingung):
 Befehle;
```

Wichtig ist, dass die Bedingung immer in Klammern gesetzt werden muss. In der Bedingung können folgende Vergleichsoperatoren auftauche:

- == Gleich
- != Ungleich
- < Kleiner

- <= Kleiner oder gleich
- > Größer
- >= Größer oder gleich

Innerhalb der Bedingung können mehrere Teilbedingungen verknüpft werden. Dafür gibt es u.a. die folgenden Operatoren:

- && AND logisches Und
- || OR logisches Oder

Ein kleines Beispiel (in HTML-Rahmen einbinden):

```
<?
 $a=5;
 $b=-3;
 if ($a>0 && $b>0):
  echo "Beide Zahlen sind positiv";
 elseif ($a<0 && $b<0):
  echo "Beide Zahlen sind negativ";
 else:
  echo "Ein dritter Fall";
 endif;
?>
```

2.3 Switch

Für mehrseitige Auswahlen gibt es in Pascal das CASE. In PHP erreicht man den gleichen Effekt mit SWITCH.

```
switch(Ausdruck) {
  case wert1:
    Befehle1;
    break;
  case wert2:
    Befehle2;
    break;
  default;
    Befehle3;
    break;
endswitch;
```

Ein Beispiel findet sich unter 2.5.

2.4 Wiederholungen

Schleifen werden realisiert mittels:

```
while (Bedingung):
  Befehle;
endwhile;
```

Für jede Wiederholstruktur ist es wichtig, dass die ausgeführten Befehle die Bedingung verändert, sonst läuft die Schleife endlos.

Ein einfaches Beispiel:

```
$i=0;
while ($i<10):
  echo "$i<p>";
  $i++;
endwhile;
```

Hiermit werden die Ziffern von 0 bis 9 ausgegeben.

2.5 Auswertung des Telefonlisten-Formulares

Nach diesen Einführungen nun das Programm zur Auswertung unseres Formulars. Das Programm wertet die Daten aus und gibt eine Rückmeldung an den Benutzer:

telefon.php3

```
<!-- Auswertungsprogramm fuer Telefonliste  letzte Aenderung am 18.4.00 -->
<HTML><HEAD>
<TITLE>Telefonliste Auswertung</TITLE>
</HEAD><BODY>
<H1 align=center>Telefonliste Auswertung</H1>
<?
   if (!$name):
       $name="Namenlos";
   endif;

   echo "Name:    $name<p>";
   echo "Vorame: $vorname<p>";
   echo "Telefon: $telefon<p>";
?>
</BODY></HTML>
```

2.6 Ein komplexeres Programm zur Formularauswertung

Nach diesen Einführungen nun das Programm zur Auswertung unseres Formulars. Das Programm wertet die Daten aus und gibt eine Rückmeldung an den Benutzer:

auwert.php3

```
<!-- Auswertungsprogramm fuer das Bespielformular von Uwe Debacher,
     erstellt am 18.09.97
     letzte Aenderung am 18.09.97 -->
<HTML><HEAD><TITLE>Auswertung</TITLE></HEAD><BODY>
<CENTER><H1>Auswertung</H1></CENTER>
Hallo <b>
<? if (!$name):
    $name="Namenlos";
   endif;
   echo $name ?>
</b><p>
<? switch($verkehrsm):
   case "b":
     echo "Hoffentlich bekommst du im Bus immer einen Sitzplatz";
     break;
   case "a":
     echo "Wo bleibt das Umweltbewu&szlig;tsein? Fahr doch mit dem Bus";
     break;
   default:
```

```
     echo "Das Fahrrad ist doch ein umweltfreundliches Verkehrsmittel";
     break;
   endswitch; ?>

<p>die Meinung:<br><i>
<?echo $meinung ?>
</i><br>zeigt deutlich, da&szlig; es dir auf dieser Schule <b>
<? if ($urteil=="s"):
     echo "sehr gut";
   elseif ($urteil=="g"):
     echo "gut";
   else:
     echo "nicht so gut";
   endif; ?>
</b> gef&auml;llt.<p>Als Leistungskurse hast du angegeben:<br>
<? if ($mathe):
    echo "Mathematik<br>";
   endif;
   if ($info):
    echo "Informatik<br>";
   endif; ?>
</BODY></HTML>
```

Es ergibt sich die folgende Ausgabe:

In dem zugehörigen Listing sind normale HTML-Befehle und PHP-Befehle sauber getrennt. Man darf sogar innerhalb von Strukturen die PHP-Sequenz beenden um normale HTML-Befehle einzubinden. Das spart jeweils den Echo-Befehl. Die Trennung hat noch einen weiteren Vorteil. Da im Echo-Befehl Textkonstanten in Anführungsstriche gesetzt werden müssen, machen viele HTML-Tags Probleme, da dort ebenfalls Anführungsstriche auftauchen. Diese „inneren" Anführungsstriche müssen dann als \" dargestellt werden, was die Übersichtlichkeit verringert.

Aus:

```
<A HREF="test.htm">Nur ein Test</A>
```

würde dann

```
echo "<A HREF=\"test.htm\">Nur ein Test</A>";
```

3. mySQL

SQL steht für Structured Query Language, was so viel wie strukturierte Abfragesprache bedeutet. Diese Sprache ist recht weit verbreitet und wird z.B. auch von Ms-Access unterstützt.

Bevor wir uns mit der Einbindung von mySQL in das PHP-System kümmern, erst einmal eine Zusammenstellung der möglichen mySQL Befehle oder wie es in der Dokumentation heißt „Clauses".

Alle Befehle lassen sich auch direkt an mySQL absenden. Dazu muss man in einer Telnet-Sitzung folgendes Programm aufrufen

/usr/bin/mysql *datenbankname*

In unserem Fall heißt die Datenbank „kurs". Datenbanken dürfen nur von einem Benutzer mit entsprechenden Rechten eingerichtet werden (/usr/bin/mysqladmin create kurs).

3.1 Create

Hiermit wird eine neue Datentabelle in einer vorhandenen Datenbank eingerichtet. Der Name der Datenbank wurde schon vorher festgelegt.

```
CREATE TABLE tabellen_name (
feld_name feld_typ [not null]
[,feld_name feld_typ [not null]]
[,feld_name feld_typ [not null]] );
```

Wird die optionale Angabe „not null" gemacht, so darf dieses Feld nicht leer bleiben. Weitere otionale Angaben sind u.a.:

- `auto_increment` erhöht automatisch den Wert
- `primary key` Primärschlüssel

Als Feld Typen stehen u.a. zur Verfügung:

- `CHAR(`*`Länge`*`)` letztendlich Strings (max. 255 Zeichen)
- `INT` ganze Zahlen
- `REAL` Real Zahlen
- `DATE` Datum in der Art „2000-04-03"
- `TEXT` Text, maximale Länge 65535 (2^16 - 1) Zeichen

Die Tabelle darf auch über mehrere Indexfelder verfügen, die folgendermaßen definiert werden:

```
CREATE [UNIQUE] INDEX index_name ON tabellen_name (
feld_name
[,feld_name] );
```

Die optionale Angabe „unique" bewirkt, dass das entsprechende Feld nicht doppelt vorkommen darf.

Beispiel:

```
CREATE TABLE telefonliste (
name CHAR(20)
,vorname CHAR(20)
,telefon CHAR(15)
) ;
```

3.2 Drop

Drop löscht Informationen. Eine Tabelle wird einfach gelöscht mit:

```
DROP TABLE tabellen_name
```

Ein Index wird entsprechend gelöscht mittels:

```
DROP INDEX index_name FROM tabellen_name
```

3.3 Insert

Insert dient zum Einfügen von Datensätzen in die Datenbank:

```
INSERT INTO tabellen_name
[(feld_name, feld_name, ..)]
VALUES (wert1, wert2,..);
```

Werden die optionalen Spaltennamen weggelassen, so müssen alle Werte in der richtigen Reihenfolge angegeben werden. Will man nur einige Werte eingeben, so müssen die zugehörigen Spaltennamen angegeben sein.

Beispiel:

```
INSERT INTO telfonliste VALUES(
"Meier",
"Klaus",
"4711"
);
```

3.4 Select

Dies ist sicherlich die meistbenutzte clause. Hiermit wird eine vorhandene Datenbank abgefragt. Die (vereinfachte) Syntax lautet:

```
SELECT feld_name [,feld_name] FROM tabellen_name
[WHERE feld_name vergleichs_operator wert]
[ORDER BY feld_name [DESC]]
```

Als feld_name nach SELECT ist auch * als Jokerzeichen für alle Felder zulässig.
Vergleichsoperatoren können sein:

- < kleiner als

- > größer als
- = gleich
- <= kleiner oder gleich
- >= größer oder gleich
- <> ungleich
- LIKE Operator für reguläre Ausdrücke im SQL-Stil
- CLIKE wie LIKE aber nicht case-sensitiv
- RLIKE Operator für reguläre Ausdrücke im Unix-Stil
- SLIKE Soundex-Operator für phonetische Vergleiche

Beispiele:

```
SELECT * FROM telefonliste
```

liefert als Ergebnis die vollständige Tabelle

```
SELECT * FROM telefonliste ORDER BY telefon
```

liefert die vollständige Telefonliste sortiert nach Telefonnummern

```
SELECT name FROM telefonliste
```

liefert alle Namen aus der Tabelle

```
SELECT name FROM telefonliste WHERE name SLIKE 'Meier'
```

liefert die Namen alle Einträge, die wir Meier klingen.

3.5 Delete

Hiermit werden Datensätze einer Tabelle gelöscht:

```
DELETE FROM tabellen_name
WHERE spalten_name vergleichs_operator wert
```

Beispiel:

```
DELETE FROM telefonliste WHERE telefon=4711
```

3.6 Update

Aktualisiert einen vorhandenen Datensatz:

```
UPDATE tabellen_name SET spalten_name=wert
WHERE spalten_name vergleichs_operator wert
```

Beispiel

```
UPDATE telefonliste SET telefon='4711' WHERE telefon='0815'
```

4. Interaktive Nutzung von mySQL

Alle mySQL-Funktionen lassen sich von der Linux-Konsole aus ansprechen. Benutzt haben wir bisher

```
/usr/bin/mysql datenbankname
```

4.1. Das Frontend mysql

Der Befehl kennt noch den Parameter -u, über den ein Benutzername übermittelt werden kann. Nach der Installation ist nur der Benutzer root eingetragen. Ist man unter Linux auch als root angemeldet, so wird der Benutzername automatisch übernommen, ansonsten muss man das Programm folgendermaßen aufrufen:

```
/usr/bin/mysql datenbankname -u root
```

Nach der Installation ist für den Benutzer root kein Passwort gesetzt, sollte man danach gefragt werden, so langt eine leere Eingabe.

Mit diesem Programm kann man alle Funktionen der Datenbank nutzen, die im Abschnitt 3. beschrieben sind.

4.2. Das Administrationsprogramm mysqladmin

Dieses Programm dient dazu die Datenbanken zu steuern. Die meistgenutzten Funktionen hierbei sind die zum Anlegen und Löschen von Datenbanken.

Eine neue Datenbank wird angelegt mittels:

```
/usr/bin/mysqladmin create datenbankname
```

Als normaler Benutzer hat man auf diesen Befehl keinen Zugriff, insofern wird man anfangs den Benutzernamen root mit angeben müssen:

```
/usr/bin/mysqladmin create datenbankname -u root
```

Löschen kann man eine Datenbank mit:

```
/usr/bin/mysqladmin drop datenbankname
```

bzw. der vollständigeren Version

```
/usr/bin/mysqladmin create datenbankname -u root
```

Nach einer Sicherheitsabfrage wird dann die komplette Datenbank mit allen enthaltenen Tabellen gelöscht.

In einem der nächsten Abschnitte wollen wir die Zugriffs-Rechte bearbeiten. Nach einer Änderung an den entsprechenden Tabellen (grant tables) muss man die Datenbank entwerder neu starten mit:

```
/sbin/init.d/mysql restart
```

oder über das Administrationsprogramm mit

```
/usr/bin/mysqladmin relaod -u root
```

die entsprechenden Tabellen neu laden.

4.3 Ausgaben mit mysqlshow

Dieses Programm verhält sich je nach Aufruf recht unterschiedlich. Ruft man es ohne weitere Parameter auf:

```
/usr/bin/mysqlshow
```

so zeigt es eine Liste aller angelegten Datenbanken.

```
+-----------+
| Databases |
+-----------+
| mysql     |
| test      |
+-----------+
```

Gibt man beim Aufruf einen Datenbanknamen mit an, so erhält man eine Liste aller Tabellen dieser Datenbank.

```
Database: mysql
+--------------+
| Tables       |
+--------------+
| columns_priv |
| db           |
| func         |
| host         |
| tables_priv  |
| user         |
+--------------+
```

Sollte der Zugriff abgelehnt werden, nach der Installation darf diese Abfrage nur root stellen, so muss wieder der Parameter -u angegeben werden

```
/usr/bin/mysqlshow mysql -u root
```

Gibt man zusätzlich zum Datenbanknamen auch eine Tabelle mit an

```
/usr/bin/mysqlshow mysql db -u root
```

so erhält man die Definition dieser Tabelle als Ausgabe.

```
Database: mysql Table: db Rows: 7
+-----------------+--------------+------+-----+---------+-------+
| Field           | Type         | Null | Key | Default | Extra |
+-----------------+--------------+------+-----+---------+-------+
| Host            | char(60)     |      | PRI |         |       |
| Db              | char(32)     |      | PRI |         |       |
| User            | char(16)     |      | PRI |         |       |
| Select_priv     | enum('N','Y') |      |     | N       |       |
| Insert_priv     | enum('N','Y') |      |     | N       |       |
```

```
| Update_priv     | enum('N','Y') |      |     | N       |       |
| Delete_priv     | enum('N','Y') |      |     | N       |       |
| Create_priv     | enum('N','Y') |      |     | N       |       |
| Drop_priv       | enum('N','Y') |      |     | N       |       |
| Grant_priv      | enum('N','Y') |      |     | N       |       |
| References_priv | enum('N','Y') |      |     | N       |       |
| Index_priv      | enum('N','Y') |      |     | N       |       |
| Alter_priv      | enum('N','Y') |      |     | N       |       |
+-----------------+---------------+------+-----+---------+-------+
```

4.4. Ausgabe von Tabelleninhalten mit mysqldump

Mit dem zuletzt angesprochenen Befehl kann man abfragen, welche Tabellen in einer Datenbank vorhanden sind und wie sie definiert sind.

Will man auch die Inhalte der Tabellen sehen, so greift man zu

```
/usr/bin/mysqldump mysql db -u root
```

und erhält als Ausgabe die Definition der Tabelle und die Inhalte. Gibt man keine Tabelle an, so erhält man die Ausgabe für alle Tabellen.

```
# MySQL dump 5.13
#
# Host: localhost    Database: mysql
#--------------------------------------------------------
# Server version        3.22.21

#
# Table structure for table 'db'
#
CREATE TABLE db (
  Host char(60) DEFAULT '' NOT NULL,
  Db char(32) DEFAULT '' NOT NULL,
  User char(16) DEFAULT '' NOT NULL,
  Select_priv enum('N','Y') DEFAULT 'N' NOT NULL,
  Insert_priv enum('N','Y') DEFAULT 'N' NOT NULL,
  Update_priv enum('N','Y') DEFAULT 'N' NOT NULL,
  Delete_priv enum('N','Y') DEFAULT 'N' NOT NULL,
  Create_priv enum('N','Y') DEFAULT 'N' NOT NULL,
  Drop_priv enum('N','Y') DEFAULT 'N' NOT NULL,
  Grant_priv enum('N','Y') DEFAULT 'N' NOT NULL,
  References_priv enum('N','Y') DEFAULT 'N' NOT NULL,
  Index_priv enum('N','Y') DEFAULT 'N' NOT NULL,
  Alter_priv enum('N','Y') DEFAULT 'N' NOT NULL,
  PRIMARY KEY (Host,Db,User),
  KEY User (User)
);

#
# Dumping data for table 'db'
#

INSERT INTO db VALUES ('%','test','','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y');
INSERT INTO db VALUES ('%','test\\_%','','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y')
```

Leitet man die Ausgabe in eine Datei um, so kann man diese als Sicherungskopie verwenden oder damit die Datenbank auf einen anderen Rechner kopieren, da alle Zeilen wie bei der Eingabe aufgebaut sind.

4.5. Die Rechteverwaltung in mySQL

Die Datenbank mySQL verfügt über ein recht ausgefeiltes Berechtigungssystem, das am Anfang oft Probleme bereitet. Seine Benutzerverwaltung ist nahezu unabhängig von der des Linux-Systems.

Nach der Installation hat nur der Benutzer „root" Zugriff auf die Datenbank, ein Passwort ist nicht gesetzt.

Diese Rechteverwaltung arbeitet mit der Datenbank *mysql*, die bei der Installation angelegt wird. Innerhalb dieser Datenbank sind vor allem die Tabellen wichtig:

- - db die Datenbanken im System
- - host die Rechner mit Rechten
- - user die Benutzer

Für eine genaue Beschreibung dieser Tabellen sollte man in die mitgelieferte Dokumentation schauen. Für den normalen Betrieb dürfte es langen die Rechte mit dem Tool mysql_setpermissions einzurichten.

/usr/bin/mysql_setpermission -u root

Zunächst fragt das Programm nach dem Passwort, danach erscheint ein kleines Textmenü:

```
Password for user root to connect to MySQL:
#########################################################################
## Welcome to the permission setter 1.2 for MySQL.
## made by Luuk de Boer
#########################################################################
What would you like to do:
1. Set password for a user.
2. Add a database + user privilege for that database.
- user can do all except all admin functions
3. Add user privilege for an existing database.
- user can do all except all admin functions
4. Add user privilege for an existing database.
- user can do all except all admin functions + no create/drop
5. Add user privilege for an existing database.
- user can do only selects (no update/delete/insert etc.)
0. exit this program

Make your choice [1,2,3,4,5,0]:
```

Für das Anlegen einer neuen Datenbank mit entsprechenden Privilegien bietet sich der Menüpunkt 2 an. In dem Dialog sind Benutzereingaben fett hervorgehoben.

```
Which database would you like to add: kurs
The new database kurs will be created

What username is to be created: wwwrun
Username = wwwrun
Would you like to set a password for wwwrun [y/n]: n
We won't set a password so the user doesn't have to use it
We now need to know from what host(s) the user will connect.
Keep in mind that % means 'from any host' ...
The host please: localhost
Would you like to add another host [yes/no]: yes
Okay, give us the host please: %
```

```
Would you like to add another host [yes/no]: no
Okay we keep it with this ...
The following host(s) will be used: localhost,%.
#########################################################################

That was it ... here is an overview of what you gave to me:
The database name       : kurs
The username            : wwwrun
The host(s)             : localhost,%
#########################################################################

Are you pretty sure you would like to implement this [yes/no]: yes
Okay ... let's go then ..

WARNING WARNING SKIPPING CREATE FOR USER 'wwwrun' AND HOST localhost
Reason: entry already exists in the user table.
WARNING WARNING SKIPPING CREATE FOR USER 'wwwrun' AND HOST %
Reason: entry already exists in the user table.
Everything is inserted and mysql privileges have been reloaded.
```

Falls eine Fehlermeldungen wie im unteren Teil auftaucht, kann man sie getrost ignorieren. Sie tauchen nur dann auch, wenn der Benutzer schon mit Rechten für eine andere Datenbank eingetragen ist.

Als erlaubter Host sollte immer *localhost* angegeben werden, sonst kann man nicht über den Webserver zugreifen. Soll auch ein Zugriff über einzelne Klienten möglich sein, eventuell über einen ODBC-Treiber und Access, so müssen alle Rechner angegeben werden, im einfachsten Fall über das Jokerzeichen %. Falls der Rechner eine ständige Internetverbindung besitzt, ist das Jokerzeichen riskant, dann sollte man die Rechner konkret angeben.

5. Einbindung von mySQL in PHP

Will man eine Datenbank ansprechen, so muss zuerst eine Verbindung zum mySQL-Server aufgebaut werden. Dazu dient der Befehl:

```
mysql_connect(hostname, username, password)
```

Solange die Datenbank auf dem gleichen Rechner liegt wie der Webserver, kann man hier einfach „localhost" angeben. Also

```
mysql_connect("localhost");
```

Da kein Benutzername angegeben wurde, wird der aufrufende Benutzer genommen, hier „wwwrun", die Benutzerkennung des Apache. Ein Passwort hatten wir nicht gesetzt.

Nach Abschluss der Datenbankbenutzung sollte das Programm die Verbindung auch wieder korrekt beenden.

```
mysql_close(hostname)
```

Gibt man hier keinen Hostnamen an, so werden alle Verbindungen beendet:

```
mysql_close()
```

5.1 Datenbankbefehle mittels PHP

PHP stellt eine Funktion zur Verfügung, mit deren Hilfe clauses direkt an mySQL weitergegeben werden können.

Dazu dient:

```
ergebnisvariable=mysql(Datenbankname, Abfrage)
```

Konkret könnte das folgendermaßen aussehen:

```
$result=mysql("kurs","SELECT * FROM telefonliste");
```

Die Ergebnisvariable hat den Wert -1, wenn ein Fehler auftrat. Der Fehler kann dann mittels $phperrormsg abgefragt werden.

Falls die Datenbankabfrage erfolgreich war und Ergebnisse zurückliefert, ist die Ergebnisvariable ein wichtiger Schlüssel zu Abfrage der Ergebnisse.

Die Zahl der gefundenen Datensätze kann ermittlt werden mit

```
zahl=mysql_NumRows(ergebnisvariable)
```

Beispiel:

```
$anzahl=mysql_NumRows($result)
```

Das folgende Beispiel gibt die Anzahl der Datensätze in unserer Tabelle aus:

telefon_aus.php3

```
<HTML><HEAD>
<TITLE>Datenbank-Beispiel 1</TITLE>
</HEAD><BODY>
<?
mysql_connect("localhost");
$result=mysql_db_query("kurs", "select * from telefonliste");
$anzahl=mysql_numrows($result);
echo "Anzahl $anzahl Datensaetze";
?>
</BODY></HTML>
```

Etwas aufwendiger wird es, wenn man an das eigentlich Ergebnis der Abfrage heran will. Das Ergebnis wird als Tabelle dargestellt, deren Reihen die Datensätze sind und deren Spalten die Attribute darstellen.

Man kann man die Datenfelder einzeln abfragen:

```
daten=mysql_result(ergebnisvariable, datensatznummer, feld_name)
```

Beispiel:

```
$wert=mysql_result($result, 0, "telefon");
```

Hiermit wird die Telefonnummer des ersten Datensatzes zurückgeliefert.

Ein vollständiges Beispiel hierzu:

telefon_aus2.php3

```
<HTML><HEAD>
<TITLE>Datenbank-Beispiel
2</TITLE>
</HEAD><BODY>
<?
mysql_connect("localhost");
$result=mysql_db_query("kurs", "select * from telefonliste");
echo $phperrormsg;
$anzahl=mysql_numrows($result);
echo "Anzahl $anzahl Datensaetze<p> ";
$tel=mysql_result($result, 0, "telefon");
echo "Datensatz 1:<br>Telefon $tel<p>";
?>
</BODY></HTML>
```

Will man alle Datensätze und alle Felder ausgeben, so muss das Beispiel um ein Schleifenkonstrukt erweitert werden.

telefon_aus3.php3

```
<HTML><HEAD>
<TITLE>Datenbank-Beispiel
3</TITLE>
</HEAD><BODY>
<?
mysql_connect("localhost");
$result=mysql_db_query("kurs", "select * from telefonliste");
echo $phperrormsg;
$anzahl=mysql_numrows($result);
echo "Anzahl $anzahl Datensaetze<p> ";
for ($i=0;$i<$anzahl;$i++) {
  $name=mysql_result($result, $i, "name");
  $vorn=mysql_result($result, $i, "vorname");
  $tel=mysql_result($result, $i, "telefon");
  echo "Datensatz $i:<br>Name: $name<br>Vorname:
  $vorn<br>Telefon $tel<p>";
}
?>
</BODY></HTML>
```

6. Zugriff auf MySQL von Access aus über ODBC-Treiber

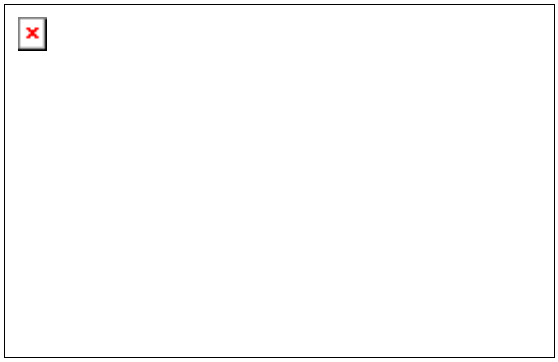
Seit geraumer Zeit existiert der Standard Open Database Connectivity (ODBC), der den Austausch zwischen verschiedenen Datenbankformaten erlaubt. Für MySQL gibt es einen entsprechenden Treiber, der unter Windows installiert werden kann. Dadurch kann jede Windows-Anwendung, die ODBC unterstützt auf MySQL-Datenbanken zugreifen.

6.1. Installation des ODBC-Treibers

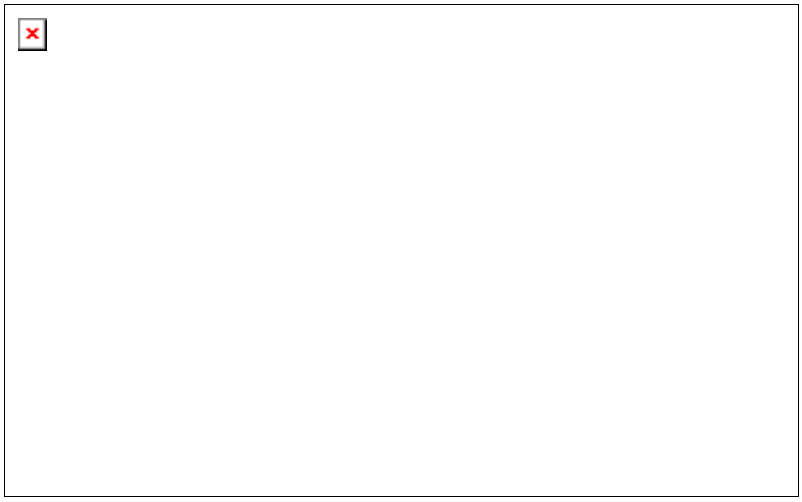
Unter der URL
http://mysql.staufen.de/download_myodbc.html wird
der jeweils aktuelle Treiber zum Download angeboten.
Das etwa 1,5MB grosse Zip-File wird geladen und in
einen beliebigen Ordner entpackt. Von dort aus wird

anschließend das Setup-Programm gestartet.

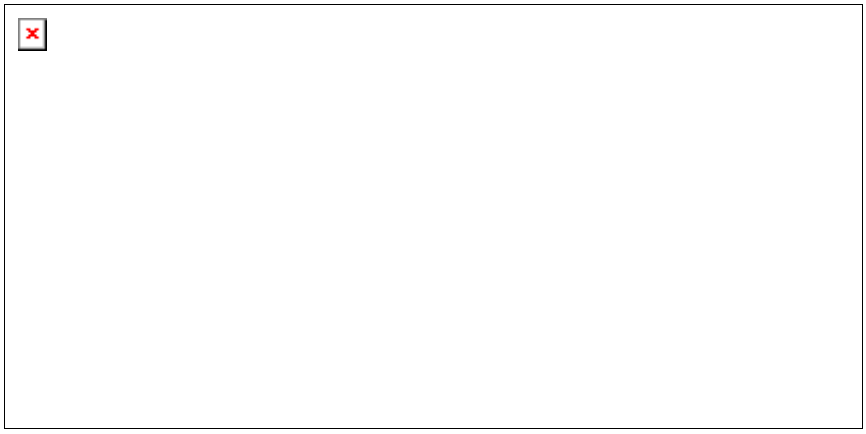
An dieser Stelle ist erst einmal nur ein Klick auf „continue" notwendig.



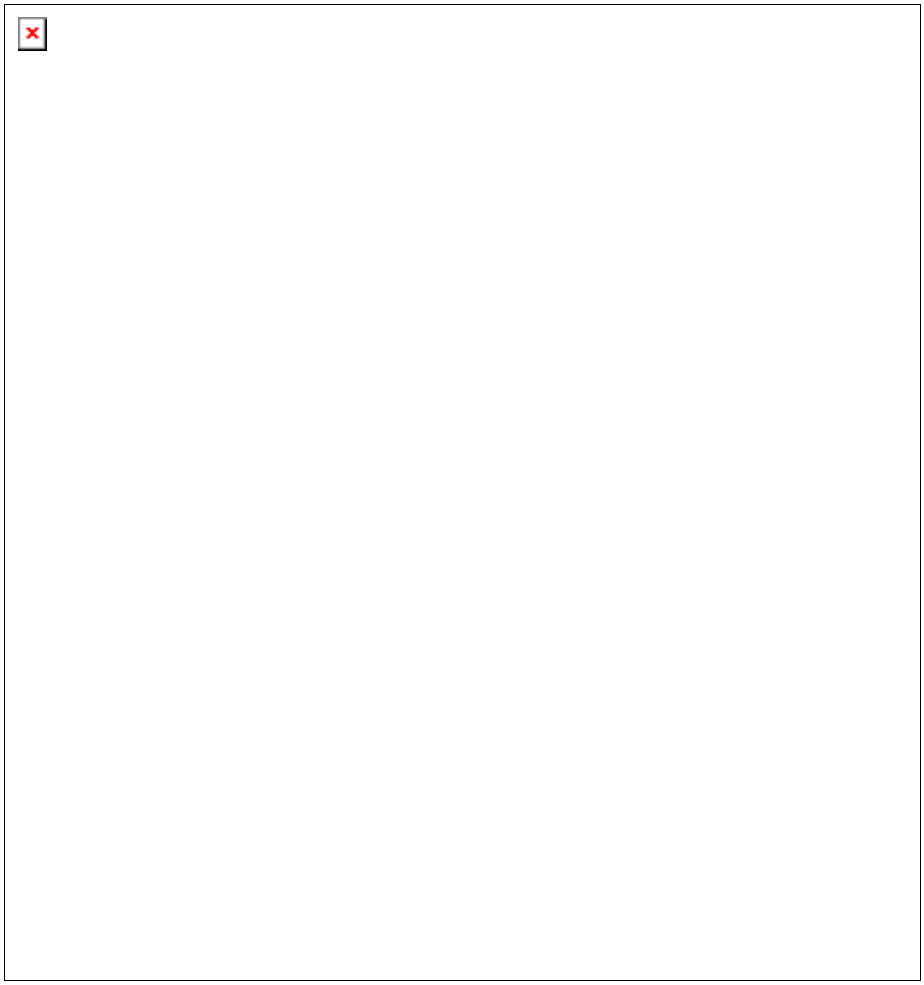
Nachdem der erste Installationsschritt erfolgt ist, wird nach dem gewünschten ODBC-Treiber gefragt. Hier muss der MySQL-Eintrag ausgewählt und dann auf „OK" geklickt werden.



Der nächste Schritt besteht in der Auswahl einer Datenquelle. Vom Installationsprogramm vorbereitet wurde „sample-MySQL". Diesen Eintrag kann man durch einen Doppelklick aktivieren und verändern.



In dem auftauchenden Fenster werden die notwendigen Eingaben gemacht. Es muss aber sichergestellt sein, dass der angegebene Benutzer auch die notwendigen Rechte für die gewählte Datenbank besitzt.

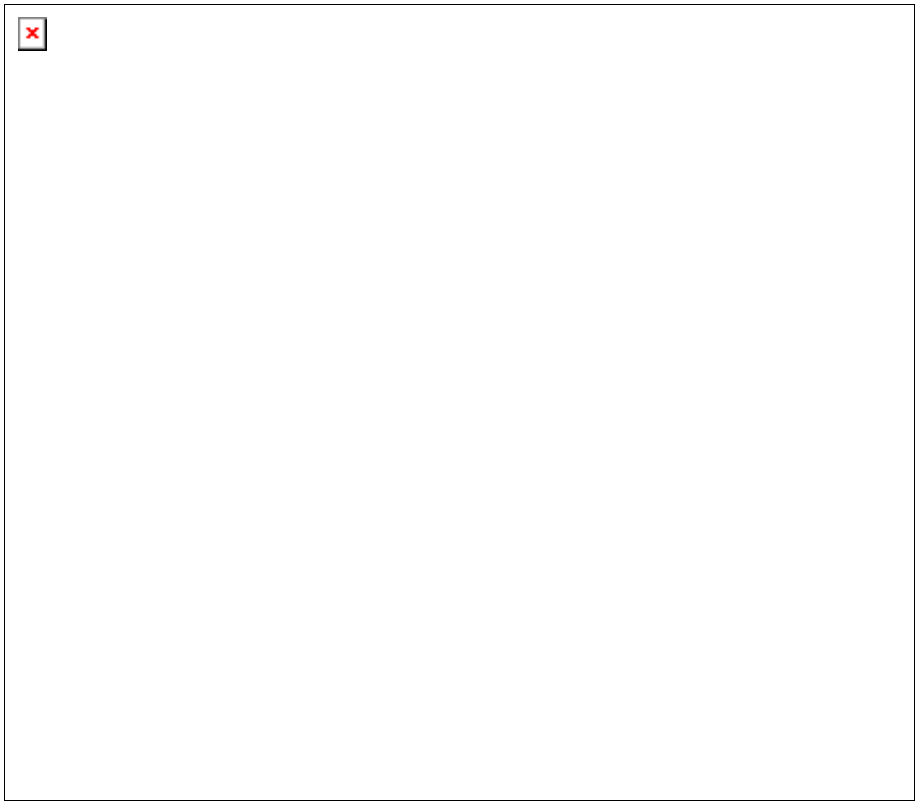


Nach einem Klick auf „OK" erscheint das folgende Fenster, in dem man die vorgenommenen Eintragungen wiederfindet. Ein Klick auf „Close" beendet die Installation des ODBC-Treibers.

6.2. MySQL-Anbindung mit Access

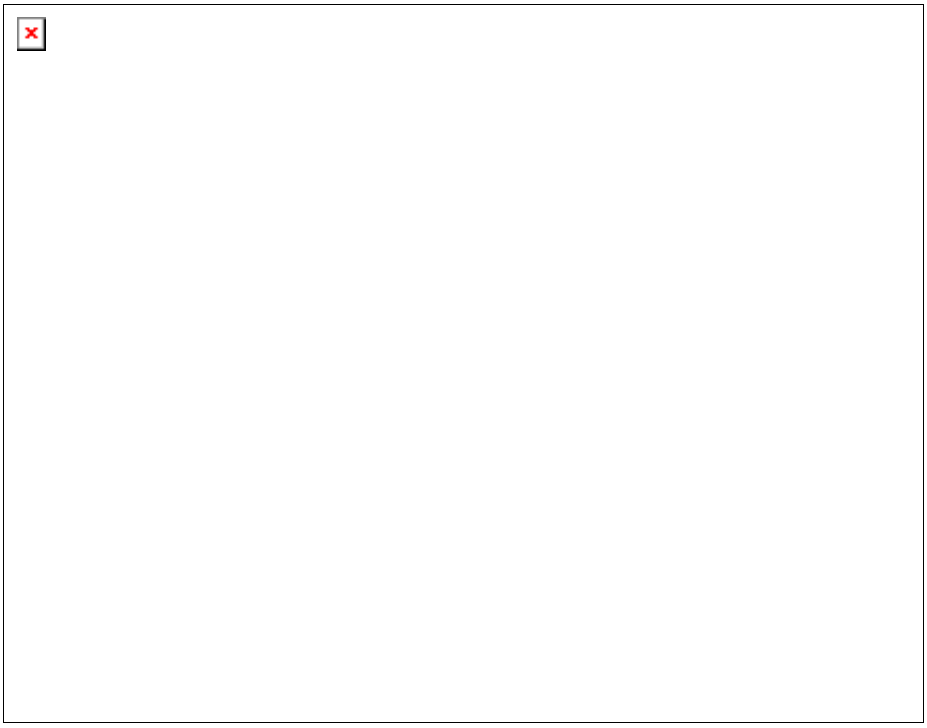
Nach der erfolgreichen Installation des ODBC-Treibers kann Access gestartet werden. Hier erzeugt man erst einmal eine leere Datenbank und kann dann die MySQL-Datenbank einbinden,

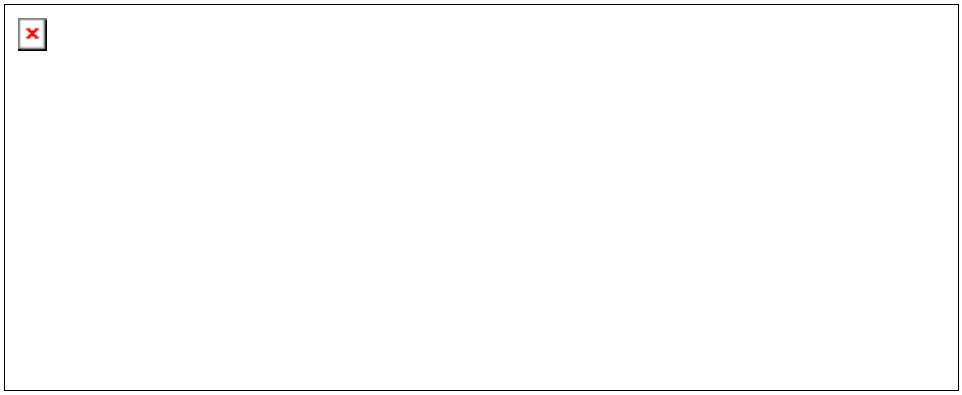
indem man „Externe Daten", „Tabellen verknüpfen" anwählt.



Es erscheint ein Dialog, in dem man die passende Datenbank auswählen kann. Hier wählt man „ODBC-Datenbanken" als Dateityp.

Im nächsten Dialog wird die Datenquelle erfragt. Unter „Computerdatenquelle" findet sich der von uns erzeugte Eintrag „IfL-Kurs". Diesen Eintrag bestätigen wir mit „OK".





Nun müssen wir innerhalb der Datenbank noch die Tabelle auswählen:

Hier wählen wir unsere Tabelle aus. Access benötigt nun für diese Tabelle noch einen Primärschlüssel.

</I> Im vorliegenden Beispiel müssten wir eigentlich alle drei Felder angeben, um die Eindeutigkeit zu gewährleisten. Nach Abschluss der Eingaben steht uns die Tabelle zur Verfügung und kann benutzt werden.

Access macht über ein verändertes

Symbol in der Tabellen- Liste deutlich, dass es sich um eine externe Tabelle handelt.

7. Ein vollständiges Beispiel: Gästebuch

Im folgenden Abschnitt soll eine vollständige Anwendung realisiert werden. Dazu nehmen wir als Beispiel ein Gästebuch. Dabei handelt es sich um eine Online-Anwendung bei der jeder Benutzer Einträge machen kann, die dann allgemein zugänglich sind.

7.1. Die Datenstruktur

Für das Gästebuch benutzen wir folgende Datenstruktur:

- `id` `int` `Laufende Nummer für den Eintrag`
- `datum` `char(10)` `Datum des Eintrages`
- `aktiv` `char(1)` `Für eine eventuelle Moderation`
- `name` `char(60)` `Name des Absender`
- `email` `char(60)` `Mailadresse des Absenders`
- `kommentar` `text` `Die Nachricht im Gästebuch`

Die Tabelle richten wir im einfachsten Fall von der Konsole aus ein, wobei die Datenbank schon eingerichtet sein muß:

```
/usr/bin/mysql kurs -u root
```

Im mySQL-Monitor erfolgen dann die notwendigen Eingaben:

```
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 11 to server version:
3.22.21
Type 'help' for help.
mysql> CREATE TABLE gaesteb (
 -> id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
 -> datum CHAR(10),
 -> aktiv CHAR(1),
 -> name CHAR(60),
```

```
-> email CHAR(60),
-> kommentar TEXT
-> );
Query OK, 0 rows affected (0.03 sec)
```

Damit ist die Tabelle angelegt. Die Definition stellt sicher, dass die Einträge alle eine unterschiedliche ID bekommen, da diese vom mySQL automatisch vergeben wird. In das Datumsfeld soll der Eintag automatisch vorgenommen werden und das Feld aktiv dient dazu in einem zweiten Schritt eine Moderation zu realisieren.

Interessant sind hier die zusätzlichen Angaben im Feld *id*

- NOT NULL verhindert eine leere Eingabe
- AUTO_INCREMENT erhöht den Wert automatisch
- PRIMARY KEY definiert das Feld als Primärschlüssel

7.2 Das Formular

Zur Eingabe dient eine einfache HTML-Seite, die ein Formular definiert. Die ersten drei Felder der Definition tauchen hier nicht auf, das sie automatisch generiert werden sollen.

gaesteb_ein.htm

```
<HTML><HEAD><TITLE>Gästebuch</TITLE></HEAD><BODY>
<CENTER><H1>G&auml;stebuch-Eingabeformular</H1></CENTER>
<FORM ACTION="gaesteb_ein.php3" METHOD="get">
Name<br>
<INPUT TYPE="text" NAME="name" SIZE=30 MAXLENGTH=60>
<p>E-Mail Adresse<br>
<INPUT TYPE="text" NAME="email" SIZE=30 MAXLENGTH=60>
<p>Kommentar<br>
<TEXTAREA NAME="kommentar" ROWS="6" COLS="40"></TEXTAREA>
<p><INPUT TYPE="submit" VALUE="Absenden">
<INPUT TYPE="reset" VALUE="Verwerfen">
</FORM>
</BODY></HTML>
```

7.3 Das Script zur Auswertung

Zum Eintragen der Daten in die Datenbank wird ein PHP3-Script benötigt, das recht kurz ausfallen kann, wenn man darauf verzichtet fehlerhafte Eingaben abzufangen.

gaesteb_ein.php3

```
<HTML><HEAD>
<TITLE>Gästebuch Neueintrag in Datenbank</TITLE>
</HEAD><BODY>
<H1 align=center>Neueintrag in das G&auml;stebuch</H1>
<?
mysql_connect("localhost");
$aktiv="1";
$datum=date("d.m.Y");
$result=mysql_db_query("kurs", "INSERT INTO gaesteb VALUES(NULL, '$datum', '$akt:
if (mysql_errno() == 0):
  echo "Datensatz ". mysql_insert_id(). " erfolgreich eingetragen";
else:
  echo "Fehler ".mysql_errno().": ".mysql_error();
endif;
```

```
?>
<br>
</BODY></HTML>
```

Das Feld *aktiv* setzen wir hier auf den Wert *1* für aktiv, so dass erst einmal keine Moderation notwendig ist. Das aktuelle Datum lassen wir vom System ermitteln mit dem Befehl *date*. Der Vorteil dieses Befehls besteht darin, dass man das Datumsformat frei definieren kann. Im Listing wird das Datum in der Form 23.05.2000 definiert. Das *d* steht für Tag, das *m* für Monat und das *Y* für Jahr in vierstelliger Form.

Statt einer id übergeben wir NULL (also keinen Wert) an mySQL, damit der Wert automatisch festgelegt werden kann.

Über die bedingte Anweisung wird zurückgemeldet, ob das Anlegen des Datensatzes erfolgreich war oder nicht. Die Funktion

```
mysql_errno()
```

liefert eines eventuelle Fehlermeldung von mySQL als Zahl zurück. Der Wert 0 steht für fehlerfreie Ausführung. Wenn die Ausführung fehlerfrei war, dann wird ein entsprechender Text und die id des Datensatzes zurückgeliefert. Die id müssen wir mit

```
mysql_insert_id()
```

ermitteln, da sie von mySQL festgelegt wurde. Die Punkte in der Ausgabezeile dienen der Stringverknüpfung. Es werden hier zwei Texte mit dem Ergebnis eines Funktionsaufrufes verknüpft. PHP3 konvertiert den Wert der Funktion dazu automatisch in einen String.

Falls ein Fehler aufgetreten ist, wird die Fehlernummer und die zugehörige Fehlerbeschreibung ausgegeben. Das ist für die Fehlersuche ganz hilfreich.

7.4. Das Script zur Anzeige der Datensätze

Will man alle vorhandenen und aktivierten Datensätze anzeigen, so kann das mit dem folgenden PHP3-Script geschehen:

gaesteb_aus.php3

```
<HTML><HEAD>
<TITLE>Gästebuch, vorhandene Einträge</TITLE>
</HEAD><BODY>
<H1 align=center>Vorhandene Eintr&auml;ge im
G&auml;stebuch</H1>
<?
mysql_connect("localhost");
$result=mysql_db_query("kurs", "SELECT * FROM gaesteb");
$anzahl=mysql_numrows($result);
if ($anzahl == 0):
  echo "Es liegen bisher keine Eintr&auml;ge vor";
else:
  for ($i=0;$i<$anzahl;$i++) {
    $id=mysql_result($result, $i, "id");
    $datum=mysql_result($result, $i, "datum");
    $aktiv=mysql_result($result, $i, "aktiv");
    $name=mysql_result($result, $i, "name");
    $email=mysql_result($result, $i, "email");
    $kommentar=mysql_result($result, $i, "kommentar");
```

```
    if ($aktiv==1):
      echo "Datensatz $id: ";
      echo "<a href=\"mailto:$email\">$name</a> meinte am $datum<p>";
      echo "$kommentar<p><hr><p>";
    endif;
  }
endif;
?>
<br>
</BODY></HTML>
```

Dieses Script wertet zuerst aus, ob überhaupt Datensätze vorhanden sind. Wenn ja, dann werden alle Datensätze ausgegeben, die aktiviert wurden.

7.5 Moderation des Gästebuches

Der Eigentümer des Gästebuches muß die Möglichkeit haben Datensätze zu bearbeiten, zu löschen und vor allem auch zu aktivieren bzw. deaktivieren.

Im Einfachsten Fall bekommt er dazu eine Sammlung von Scripten, die eventuell auch noch mit einem Passwort gegen unbefugten Zugriff gesichert werden könnten.

Zuerst ein Script, das die Datensätze auflistet und jeweils auf das Bearbeitungsformular verlinkt.

gaestb_mod1.php3

```
<HTML><HEAD>
<TITLE>Gästebuch Moderation Liste vorhandener Einträge</TITLE>
</HEAD><BODY>
<H1 align=center>Moderation: Liste vorhandener Eintr&auml;ge</H1>
<?
 mysql_connect("localhost");
 $result=mysql_db_query("kurs", "SELECT * FROM gaesteb");
 $anzahl=mysql_numrows($result);
 if ($anzahl == 0):
   echo "Es liegen bisher keine Eintr&auml;ge vor";
 else:
   for ($i=0;$i<$anzahl;$i++) {
     $id=mysql_result($result, $i, "id");
     $datum=mysql_result($result, $i, "datum");
     $aktiv=mysql_result($result, $i, "aktiv");
     $name=mysql_result($result, $i, "name");
     $email=mysql_result($result, $i, "email");
     $kommentar=mysql_result($result, $i, "kommentar");
     echo "<a href=\"mailto:$email\">$name</a> am $datum";
     if ($aktiv==1):
       echo " (aktiviert)";
     endif;
     echo "<p>$kommentar<p>";
     echo "<a href=\"gaesteb_mod2.php3?id=$id\">bearbeiten</a>   ";
     echo "<a href=\"gaesteb_mod4.php3?id=$id\"> l&ouml;schen</a>";
     echo "<hr>";
   }
 endif;
?>
<br>
</BODY></HTML> FACE="Courier"
```

Für jeden vorhandenen Eintrag wird ein Link zum Bearbeiten erzeugt, der das Moderationsformular

aufruft und die ID des Datensatzes als Parameter übergibt. Entsprechend wird ein Link zum Löschen eingefügt,

Der zweite Teil des Programmes zum Bearbeiten erwartet als Übergabeparameter die ID des Datensatzes und holt damit alle Angaben aus der Datenbank und stellt sie in einem Formular dar.

gaestb_mod2.php3

```
<HTML><HEAD>
<TITLE>Gästebuch Moderation vorhandener Einträge Teil 2</TITLE>
</HEAD><BODY>
<H1 align=center>Moderation vorhandener Eintr&auml;ge Datensatz
<?
echo " $id</H1>";
mysql_connect("localhost");
$result=mysql_db_query("kurs", "SELECT * FROM gaesteb where id=$id");
$anzahl=mysql_numrows($result);
if ($anzahl == 0):
  echo "Es liegt kein Eintrag vor";
else:
  $id=mysql_result($result, $i, "id");
  $datum=mysql_result($result, $i, "datum");
  $aktiv=mysql_result($result, $i, "aktiv");
  $name=mysql_result($result, $i, "name");
  $email=mysql_result($result, $i, "email");
  $kommentar=mysql_result($result, $i, "kommentar");
  echo "<FORM ACTION=\"gaesteb_mod3.php3\" METHOD=\"get\">";
  echo "ID: $id";
  echo "<INPUT TYPE=\"HIDDEN\" NAME=\"id\" VALUE=\"$id\">";
  echo "<p>Datum: $datum";
  echo "<INPUT TYPE=\"HIDDEN\" NAME=\"datum\" VALUE=\"$datum\">";
  echo "<p>Aktiviert <INPUT TYPE=\"CHECKBOX\" NAME=\"aktiv\"";
  if ($aktiv=="1"):
    echo "CHECKED";
  endif;
  echo ">";
  echo "<p>Name<br><INPUT TYPE=\"text\" NAME=\"name\" SIZE=30 MAXLENGTH=60 VALUE=
  echo "<p>E-Mail Adresse<br><INPUT TYPE=\"text\" NAME=\"email\" SIZE=30 MAXLENG
  echo "<p>Kommentar<br><TEXTAREA NAME=\"kommentar\" ROWS=\"6\" COLS=\"40\">$komm
  echo "<p><INPUT TYPE=\"submit\" VALUE=\"Absenden\">";
  echo "<INPUT TYPE=\"reset\" VALUE=\"Verwerfen\">";
  echo "</FORM>";
endif;
?>
<br>
</BODY></HTML>
```

Der letzte Teil des Moderationsprogrammes muss dann die gemachten Änderungen in die Datenbank eintragen. Dazu wird zuerst der vorhandene Datensatz gelöscht und dann neu eingerichtet.

gaestb_mod3.php3

```
<HTML><HEAD>
<TITLE>Gästebuch Neueintrag in Datenbank</TITLE>
</HEAD><BODY>
<H1 align=center>&Auml;nderungseintrag in das G&auml;stebuch</H1>
<?
mysql_connect("localhost");
$result=mysql_db_query("kurs", "DELETE FROM gaesteb WHERE id=$id");
if (mysql_errno() == 0):
  echo "Datensatz $id erfolgreich entfernt";
```

```
  if ($aktiv):
    $aktiv=1;
  else:
    $aktiv=0;
  endif;
  $result=mysql_db_query("kurs", "INSERT INTO gaesteb VALUES($id, '$datum', '$akt
  if (mysql_errno() == 0):
    echo "Datensatz ". mysql_insert_id(). " erfolgreich neu eingetragen";
  else:
    echo "Fehler ".mysql_errno().": ".mysql_error();
  endif;
else:
  echo "Fehler ".mysql_errno().": ".mysql_error();
endif;
?>
<br>
</BODY></HTML>
```

Nun fehlt nur noch das Script zum Löschen eines Datensatzes. Die Ähnlichkeit mit dem letzten Script dürfte nicht überraschen:

gaestb_mod4.php3

```
<HTML><HEAD>
<TITLE>Gästebuch L&oschen eines Eintrages</TITLE>
</HEAD><BODY>
<H1 align=center>L&ouml;schen eines Eintrages im G&auml;stebuch</H1>
<?
mysql_connect("localhost");
$result=mysql_db_query("kurs", "DELETE FROM gaesteb WHERE id=$id");
if (mysql_errno() == 0):
  echo "Datensatz $id erfolgreich entfernt";
else:
  echo "Fehler ".mysql_errno().": ".mysql_error();
endif;
?>
<br>
</BODY></HTML>
```

Mit diesem Beispiel ist das Grundgerüst für ein Gästebuch erstellt. Es fehlen aber noch eine Reihe von Sicherheitsabfragen, z.B. vor dem Löschen eines Datensatzes.

---

geschrieben von Uwe Debacher, letzte Änderung am 23.5.2000