# Serverseitige Web-Programmierung unter Linux

*Michael Kugler*

Eine der unter Linux zur freien Verfügung stehenden Datenbank heißt **mysql**. Dieses Datenbankprogramm arbeitet eng mit der Skriptsprache **php** zusammen. Der Webserver Apache wiederum unterstützt diese Scriptsprache. In diesem Artikel geht es um das Zusammenspiel dieser drei Komponenten. Die verwendete Distribution ist SuSe 7.0.

## 1. Die Datenbank mysql

Das Datenbanksystem **mysql** besitzt eine Multiuser Multithread Architektur. Es existieren verschiedene Programminterfaces, z.B. für C, C++, php, Perl und viele mehr. mysql gibt es für über 20 verschiedenen Betriebssysteme(z.B. Linux, Windows9x, Windows NT . . . ). Weitere Leistungsmerkmale: ODBC-Standard 2.5, hochoptimierte SQL-Funktionen, sehr schnelle Implementation von JOIN's,... mysql Datenbanken können über 50.000.000 Datensätze enthalten und unterstützen den Zeichensatz ISO-9959-1 Latin 1.

### 1.1 Installation

Die Installation erfolgt problemlos mit dem Installationstool **yast**. Die Frage, ob die Datenbank beim Starten aktiviert werden soll, ist mit ja zu beantworten. Dadurch wird beim nächsten Starten des Betriebssystems das Programm **mysqld** automatisch gestartet. Für das manuelle Starten, das ist z.B. unmittelbar nach der Installation notwendig, genügt es, von der **shell**-Konsole als Superuser **mysqld &** als Hintergrundjob zu starten.

### 1.2 Anlegen einer Datenbank

Das Datenbanksystem **mysql** kann nun über mehrere Schnittstellen angesprochen werden. Ich bespreche hier die Kommandozeile, in der hauptsächlich die Administration durchgeführt wird. Im folgenden wird eine kleine Datenbank implementiert. Für einen Verein soll eine Mitgliederverwaltung geschrieben werden. Eine der Tabellen enthält die persönlichen Daten der Mitglieder. In einer weiteren werden die verschieden Einträge, die im Laufe einer Mitgliedschaft entstehen, eingetragen. In der dritten werden alle für eine Vereinsverwaltung notwendigen Textbausteine verwaltet. (Der Aufbau einer Tabellenstruktur ist in der Regel ein sehr aufwendiger Prozess, da die Tabellenstruktur einigen Regeln, so genannten Normalformen, gehorchen soll. Warum diese Tabellenstruktur für dieses Beispiel gewählt wurde, ist Gegenstand eines anderen Artikels.)

Der erste Schritt ist es, **mysql** mitzuteilen, dass es eine neue Datenbank geben wird. Diese bekommt den Titel **VIT**.

Welche Datenbanken kennt das System? Am Datenbank-Prompt **mysql>** wird **show Databases;** eingeben. (Der abschließende Strichpunkt ist wie bei allen Anfragen an die Datenbank unbedingt notwendig). Angezeigt wird:

```
mysql> show databases;
+----------+
| Database |
+----------+
| BACKUP   |
| Schule1  |
| mysql    |
| test     |
+----------+
4 rows in set (0.57 sec)
```

**Create Database VIT;** erzeugt die neue Datenbank. Mit dem Befehl **use VIT;** teilt man dem Datenbankmanagment mit, dass man mit der Datenbank VIT arbeiten möchte. Alternativ kann man schon beim Aufruf von **mysql** den Datenbanknamen angeben (**mysql VIT**).

### 1.3 Anlegen von Tabellen

Eine Datenbank besteht aus Tabellen. Wir werden nun in dieser Datenbank die notwendigen Tabellen erzeugen. Dies geschieht mit einem Standard SQL Befehl (Hinweis: Jeder SQL-Befehl endet mit einem Strichpunkt). Jedes Mitglied wird in der Datenbank über eine eindeutige Nummer, die `MitgliederNummerID`, identifiziert. Durch die Option `auto_increment`, wird bei jedem Eintrag der Wert automatisch um eins erhöht. Die Spalte Land besitzt einen Standardwert, dieser wird immer bei der Eingabe eines neuen Datensatzes verwendet, wenn kein Wert angegeben wird. Mit `PRIMARY KEY (MitgliederNummerID)` wird die Spalte `MitgliedsnummerID` zu einem speziellen Index, dem Primary Key.

```
CREATE TABLE Mitglieder (
  FamilienName varchar(30),
  VorName varchar(30),
  MitgliederNummerID
          int(5)
          unsigned DEFAULT '0' NOT NULL,
          auto increment,
  PLZ varchar(7),
  Land varchar(30)
          DEFAULT 'Österreich',
  Ort varchar(40),
  Strasse varchar(60),
  Geburtsdatum date,
  Arbeitsgeber varchar(40),
  PRIMARY KEY (MitgliederNummerID)
);
```

Welche Tabellen sind in dieser Datenbank?

```
mysql> Show Tables;
+---------------+
| Tables in VIT |
+---------------+
| Eintraege     |
| Ereignisse    |
| Mitglieder    |
+---------------+
3 rows in set (0.00 sec)
```

Wie sieht die Struktur der Tabelle Mitglieder aus?

```
mysql> describe Mitglieder;
+-------------------+---------------+------+-----+------------+----------------+
| Field             | Type          | Null | Key | Default    | Extra          |
+-------------------+---------------+------+-----+------------+----------------+
| FamilienName      | varchar(30)   | YES  |     | NULL       |                |
| VorName           | varchar(30)   | YES  |     | NULL       |                |
| MitgliederNummerID| int(5)unsigned|      | PRI | 0          |auto_increment  |
| PLZ               | varchar(7)    | YES  |     | NULL       |                |
| Land              | varchar(30)   | YES  |     | Österreich |                |
| Ort               | varchar(40)   | YES  |     | NULL       |                |
| Strasse           | varchar(60)   | YES  |     | NULL       |                |
| Geburtsdatum      | date          | YES  |     | NULL       |                |
| Arbeitsgeber      | varchar(40)   | YES  |     | NULL       |                |
+-------------------+---------------+------+-----+------------+----------------+
9 rows in set (0.00 sec)
```

### 1.4 Tabellen mit Daten füllen

Jede Datenbank lebt von Daten. Üblicherweise hat man am Anfang die einzutragenden Daten in irgend einer Form und können als ASCII-Text gespeichert werden. Dieser kann dann mittels des folgenden Befehls in die entsprechende Tabelle eingetragen werden.

```
LOAD DATA LOCAL
    INFILE 'Namen.txt'
    INTO TABLE Mitglieder;
```

Das Textfile, in unserem Fall **Namen.txt**, muss nach der folgenden Spezifikation erstellt werden. Die einzelnen Spalteneinträge sind mit einem Tabulator getrennt, eine Zeile wird mit einem Linefeed abgeschlossen. Ist das Textfile anders aufgebaut, so ist das Einlesen ebenfalls möglich, die mitgelieferte Doku liefert die notwendige Info. (MS-Dos verwendet üblicherweise CR-LF als Zeilentrenner!) Eine zweite Art des Eintragens von Daten ist die Verwendung des SQL Befehls `INSERT`.

```
INSERT INTO Mitglieder
        VALUES ('Franz','Müller',1,NULL,
                'Österreich',NULL,NULL,NULL,NULL);
```

Das interaktive Eingeben von größeren SQL-Befehlen kann schnell zum Problem werden. In diesem Fall ist es günstig, die notwendige SQL-Zeichenkette in einem Editor zu bearbeiten und mit einem externen Aufruf von mysql zu verarbeiten. Der obige Befehl zum Erzeugen der Mitgliedertabelle sei in dem File **sql.txt**

abgespeichert. Mit **mysql VIT <sql.txt** wird **mysql** aufgerufen, in die Datenbank **VIT** gewechselt und der Inhalt des Files **sql.txt** ausgeführt.

### 1.5 erste eine Abfrage

Nun sind die Daten in der Tabelle. Mit einem einfachen SELECT-Befehl aus dem SQL-Befehlssatz kann das überprüft werden.

```
mysql> select Vorname, FamilienName from Mitglieder;
+---------+--------------+
| Vorname | FamilienName |
+---------+--------------+
| Franz   | Müller       |
| Fritz   | Meier        |
| Rosi    | Bauer        |
| Lisi    | Berger       |
+---------+--------------+
4 rows in set (0.00 sec)
```

### 1.6 Ein kleiner Einblick in das Sicherheitssystem von mysql

Das Sicherheitssystem von **mysql** wird in der Datenbank **mysql** verwaltet.

```
mysql> use mysql;
Database changed
mysql> show tables;
+-----------------+
| Tables in mysql |
+-----------------+
| columns_priv    |
| db              |
| func            |
| host            |
| tables_priv     |
| user            |
+-----------------+
6 rows in set (0.00 sec)
```

#### 1.6.1 1. Schritt: Berechtigung zum Verbindungsaufbau

In der Tabelle host werden die Daten zum Verbindungsaufbau gespeichert.

```
select * from host;
```

Die in dieser Tabelle eingetragene Host/User Kombination gilt für alle Datenbanken. Im folgenden Beispiel soll ein User mit dem Namen testuser am localhost mit dem Passwort testpassword in die Tabelle eingetragen werden.

```
INSERT INTO user (host, user, password)
       VALUES (localhost, 'testuser',password('testpassword'));
```

Die Passwörter werden in der Datenbank so abgelegt, wie sie im SQL-String angegeben sind. Da beim Verbindungsaufbau das Passwort verschlüsselt übertragen wird, ist die Funktion password() im SQL-String wichtig. (Aus der Sicht von **mysql** ist das in der Tabelle eingetragene Passwort das wirkliche Passwort.)

#### 1.6.2 2. Schritt: Berechtigungen für die einzelnen Datenbanken

In der Tabelle db werden die Berechtigungen für die einzelnen Tabellen geregelt:

```
select * from db;
```

Im folgenden Beispiel wird dem Benutzer michi in der Datenbank VIT vom Host localhost aus das Recht, Abfragen zu stellen (das sind alle SELECT-Anweisungen) zugewiesen. Alle anderen Rechte sind, da sie nicht gesetzt sind, defaultmäßig auf N gestellt.

```
INSERT  INTO db (host, db, user, Select priv)
        VALUES ('localhost','VIT','michi','Y')
```

Ist der Wert in einer Spalte egal, so kann dafür der Platzhalter (%) verwendet werden. (z.B: alle Hosts, jede Datenbank,. . . )

Im folgendem Beispiel wird dem Benutzer michi das Recht, etwas in die Datenbank einfügen zu dürfen, gesetzt:

```
update db set Insert_priv ='Y' where user ='michi';
```

#### 1.6.3 Finalisierung

Damit von der Datenbank diese Werte übernommen werden, ist noch ein weiterer Schritt notwendig: Entweder wird von der **bash** das Skript

```
mysqladmin -u root -p reload
```

gestartet, oder von der **mysql**-Konsole aus

```
mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)
```

Damit wird das Datenbankmanagment angewiesen, die Berechtigungen neu einzulesen. (Die verwendeten Optionen bedeuten -u verwende den nachfolgenden Benutzer, -p frage nach dem Passwort. Achtung: Das **root**-Passwort ist das in der Datenbank, nicht das des Linux-Systems.)

## 2 php-eine Skriptsprache für das Web

PHP ist ein relativ junge Sprache, die Anfang 1995 von Rasmus Lerdorf für die Realisierung von Webanwendungen geschaffen wurde. (Die ursprüngliche Bedeutung der Buchstaben PHP stand für Personal Homepage Tools, heute wird darunter meist PHP Hypertext Preprocessor verstanden. PHP ist eine Skriptsprache, die in HTML Code eingebettet ist. Das Script wird vom HTTP-Server, unter Linux ist dies meist das Programm Apache, verarbeitet. Damit der Webserver den HTML-Code als PHP-Skript erkennt, und damit vor Absenden an den Client bearbeitet, bekommt jedes PHP Skript die Endung **.php** . Der auf diese Art veränderte HTML-Code wird anschließend dem Client gesendet.

Der Client erhält reinen HTML-Kode. Damit ist es möglich, auf Datenbanken zuzugreifen, ohne dass der Client erfährt, in welcher Form dies geschehen ist.

PHP stellt zur Zeit etwa 1200 Funktionen zu Verfügung; es kann sowohl prozedural als auch objektorientiert programmiert werden. Neben den Funktionen, die sowieso von jeder Skriptsprache erwartet werden, sind hervorzuheben: HTTP-Funktionen, Mail-Funktionen, Unterstützung bei der Abfrage von LDAP-Servern, Session-Handling, Unterstützung von Cookies und ODBC.

### 2.1 Das erste PHP Skript

PHP wird im HTML Text eingebettet, daher beginnt jedes Skript mit den einleitenden HTML-Tags. PHP-Skripts werden in <?.....?> eingeschlossen. Jeder Befehl endet mit einem Strichpunkt. Kommentare werden wie in C bzw. C++ geschrieben. Hier das traditionelle Einsteigebeispiel Hallo Welt.

```
<html>
<body>
<?   /* Hier beginnt der PHP Code*/
echo ("Hallo Welt");
?></body>
</html>
```

### 2.2 Einige Strukturelemente von PHP

Variablen beginnen mit einem $. Man muss sie nicht deklarieren. Ihren Typ erhalten sie bei der ersten Wertzuweisung. Groß- Kleinschreibung ist von Bedeutung. Grundsätzlich sind die Variablen nur im lokalen Kontext sichtbar, jedoch können sie explizit sichtbar gemacht werden.

```
$meinWert=2;
```

Arrays beginnen ebenfalls mit einen $. Der Index wird in eckigen Klammern angegeben und beginnt immer mit 0.

```
$Feld[2]=2;
```

Eine andere Form der Arrays sind die assoziativen Arrays (Hash), bei ihnen wird statt des nummerischen Index eine eindeutige Zeichenkette zur Identifizierung herangezogen.

```
$Feld[Strasse]="Apfelweg";
```

Für Abfragen stehen die bekannten if, elseif, else und endif zur Verfügung. Für die Auswahl mehrerer Alternativen gibt es das switch-case Konstrukt. Bei den Schleifen gibt es die üblichen while-Konstrukte, die unter C bekannte for -Schleife wird durch ein foreach ergänzt (nur in PHP4).

### 2.3 Ein einfaches Formular

HTML-Formulare stellen eine einfache Art dar, Daten vom Client zum Server zu übertragen. Jede im Formular angegebene Bezeichnung wird im PHP-Skript automatisch zu einer Variablen. Im folgenden soll ein Formular erzeugt werden, das die Eingabe in ein Textfeld ermöglicht. Die Daten werden mit der Methode **GET** übertragen. Wird auf den Submitbutton gedrückt, soll das Skript **f2.php** (In diesem Beispiel ist das aufzurufende Skript gleich dem aufgerufenen.) ausgeführt werden.

```
<html>
<body>
<form method=get action=f2.php>
```

Diese Sequenz erstellt das Texteingabefeld mit dem Namen `Familienname` und der Länge 45.

```
<input type=text name="Familienname" size = 45>
```

Mit der Erstellung des Submitbuttons ist das HTML-Formular fertig.

```
<input type=submit name ="submit" value =senden>
</form>
```

Nach einem Zeilenumbruch (`<br>`) beginnt das PHP-Skript. Der Client ruft das Skript erneut auf und übergibt dabei dem Skript in der Variablen `$Familienname` den im Textfeld eingetragenen String. Der `echo`-Befehl des Skripts gibt diesen Wert aus. Der Rest des Skripts beendet den HTML-Code.

```
<br>
<? echo $Familienname;
?>
</body>
</html>
```

## 2.4 Zugriff auf die Datenbank

Um auf die Datenbank zugreifen zu können, sind folgende Informationen notwendig:

- **Auf welchem Server liegt die Datenbank?**
  Es ist nicht notwendig, dass der Webserver und der Rechner der Datenbank identisch sein müssen. Es ist notwendig, entweder die IP-Adresse oder den Namen des Rechners zu kennen. (Im Fall des selben Rechners wird der Name `localhost` verwendet.)
- **Wie heißt die Benutzerkennung unter dem der Datenbankzugriff erfolgen soll?**
  Diese Benutzerkennung ist nicht jene des Betriebssystems. Der Benutzer **root** der Datenbank ist daher nicht der gleiche, wie der Benutzer **root** des Betriebssystems.
- **Das Passwort der dazugehörigen Benutzerkennung**
  Da das Skript am Server zuerst ausgeführt wird, gehen diese Daten (Benutzerkennung und Passwort) nicht an den Client weiter. (Es sei denn, der Programmierer sieht dies ausdrücklich vor.)
- **Den Namen der Datenbank, mit der gearbeitet werden soll**

  Das folgende PHP-Skript verwendet die Funktion `mysql_connect` um die Verbindung zum Datenbanksystem herzustellen. `$link_id` liefert einen Handle zurück, falls die Verbindung hergestellt werden konnte. Dieser Handle wird für die weiteren Datenbankaktivitäten benötigt.

```
<html><body>
<?
 $MySQL Host  = "localhost";
 $MySQL User  = "michi";
 $MySQL_Passw = "12345";
    // mit der Datenbank verbinden
 $link_id = mysql_connect("$MySQL_Host",
                          "$MySQL User",
                          "$MySQL Passw"
                        );
 if (! $link_id )
   echo "<br> ".
     "Die Verbindung zur Datenbank konnte nicht hergestellt werden".
     "<br>";
 else
   echo "<br>".
     "Die Verbindung zur Datenbank ist
hergestellt <br>";
?>
 </html></body>
```

## 2.5 Eine erste Abfrage

Nachdem die Verbindung zur Datenbank erstellt wurde, ist der Zugriff auf die Daten möglich. Dazu sind folgende Voraussetzungen notwendig:

- Ein Verbindungshandle, wie im vorigen Kapitel beschrieben,
- den Namen der Datenbank, die die Daten liefern soll,
- eine SQL-Zeichenkette, für die Abfrage der Daten aus der Datenbank.
- Das folgende Skript stellt die Verbindung zur Datenbank `VIT` her, und übergibt dieser den SQL-String. Rückgabewert ist ein Handle, mit dessen Hilfe, die aus der Datenbank gewonnenen Werte dem Skript übergeben werden können.

```
<html><body>
<?
  $MySQL Host  = "localhost";
  $MySQL_User  = "michi";
  $MySQL Passw = "12345";
  $DataBase    = "VIT";
  $SQL         = "Select FamilienName from Mitglieder;";
  // mit der Datenbank verbinden
  $link id =mysql connect("$MySQL Host",
                          "$MySQL_User",
                          "$MySQL Passw "
                         );
  // Abfrage durchführen
  $res = mysql_db_query ($DataBase,$SQL);
  if (! $res )
   echo "<br> Die Abfrage konnte nicht durchgeführt werden <br>";
  else
   echo "<br> Die Abfrage war erfolgreich <br>";
?>
</html></body>
```

## 2.6 Ausgabe der Daten in HTML

Die Ausgabe einer Tabelle beginnt mit

```
  echo "<table border>";
```

Um für den Tabellekopf die Spaltennamen ausgeben zu können, werden in einer Schleife alle Spaltennamen durchlaufen. `mysql_num_fields($res)` gibt an, wie viele Spalten die Abfrage hat; `mysql_field_name($res,$i)` gibt den Namen an der Position `$i` an.

```
echo "<tr>";
for ($i = 0; $i < mysql num fields($res); $i++)
    {
      echo "<th>";
      echo  mysql_field_name($res,$i);
      echo "</th>";
    }
 echo "</tr>";
```

Verwendet wird `mysql_fetch_row ($res)`, um einen kompletten Datensatz zu bekommen. Jeder Aufruf dieser Funktion erhöht automatisch den Datensatzzeiger. Solange es Daten gibt, wird das Array `$zeile` gefüllt, gibt es keine Daten mehr, wird der Wert `false` zurückgegeben.

```
 while ($zeile = mysql fetch row ($res) )
   {
      echo "<tr>";          //eine Zeile beginnt
      for ($i = 0; $i < mysql_num_fields($res); $i++)
           echo "<td>". $zeile[$i]."</td>";
      echo "</tr>\n";     // eine Zeile endet
   }
```

Fehlt nur mehr der Befehl zum Beenden der Tabelle:

```
  echo "</table border>";
```