

---

## 9 Datenbank-Anbindung

### 9.1 Einführung

#### In diesem Kapitel lernen Sie

- ▶ wie Sie Webseiten dynamisch aus einer MySQL-Datenbank generieren lassen.
- ▶ wie Sie auf diese Datenbank mit PHP und Perl zugreifen.

☞ *Hinweis:* Dieses Kapitel ist mehr als Überblick gedacht, damit Sie einen Eindruck bekommen, wie die Datenbank-Anbindung an Apache *prinzipiell* funktioniert. Um dieses Thema (Datenbanken, PHP-Programmierung) befriedigend zu behandeln, wären mindestens zwei eigene Kurse mit eigenen Kurshandbüchern notwendig. Sie können daher bei Zeitnot dieses Kapitel (und Übung) gerne überspringen oder nur überfliegen.

Dies hat nicht unmittelbar etwas mit der Konfiguration des Apache-Webservers zu tun, sondern eher mit den individuellen CGI-Programmen oder in Webseiten eingebetteten Skripten. Speziell im Fall von PHP ist aber auch die Installation eines zusätzlichen Moduls notwendig.

#### Voraussetzungen:

- ☑ Apache ist bereits für die Benutzung von CGI-Skripten vorbereitet,
- ☑ `mod_php` ist installiert und in die Apache-Konfiguration eingebunden,
- ☑ Sie sind mit dem Verfahren von in Webseiten eingebetteten Skripten bzw. der für den Apache relevanten `Handler`-Direktiven, die für bestimmte Dateiendungen gelten, vertraut (siehe auch Abschnitt 8 Seite 83).

### 9.2 MySQL

*MySQL* (<http://www.mysql.com/>) ist eine zum SQL-Standardsprachsatz kompatible, relationale Datenbank mit Erweiterungen wie Replikation und ODBC-Support sowie offenen Schnittstellen für viele Programmiersprachen. Aufgrund der hohen Verbreitung kann MySQL als die beliebteste Open Source Datenbank bezeichnet werden.

☞ *Hinweis:* Im Webserver-Bereich hat MySQL den Begriff *LAMP* mitgeprägt, der für

## Datenbank-Anbindung

---

die beliebte Kombination aus Linux Betriebssystem, Apache Webserver, MySQL Datenbank und PHP bzw. Perl als unterstützende Programmiersprachen für dynamisch generierte Webseiten steht.

Seit Mitte 2000 untersteht MySQL der *GNU GENERAL PUBLIC LICENSE* und ist somit ohne Nutzungsgebühr sowohl kommerziell als auch nicht-kommerziell inklusive eventueller eigener Quelltext-Modifikationen und Erweiterungen einsetzbar. Natürlich ist auch bezahlter Support von der Herstellerfirma verfügbar.

Für die verschiedenen Linux-Distributionen gibt es die aktuelle MySQL-Version fertig übersetzt und vorkonfiguriert als RPM- oder Debian-Paket, so dass die Installation der MySQL-Komponenten (shared libs, Server, verschiedene Clients und Programmierschnittstellen) wie gewohnt mit

```
# rpm -Uvh mysql-version.rpm
# rpm -Uvh mysql-client-version.rpm
```

bzw. bei Fedora yum mit

```
# yum install mysql
# yum install mysql-server
# yum install mysql-client
```

bzw. bei SuSE zypper mit

```
# zypper install mysql
# zypper install mysql-client
```

bzw. bei Debian mit

```
# apt-get install mysql-server
# apt-get install mysql-client
```

durchgeführt werden kann. Anschließend sollte ein Passwort für den Datenbank-Zugriff mit Hilfe der mitinstallierten Clients gesetzt werden:

```
# /etc/init.d/mysql start
# mysqladmin -u root password 'new-password'
```

Die MySQL-Konfigurationsdatei `my.cnf` (üblicherweise im Verzeichnis `/etc` zu finden bzw. `/etc/mysql` unter Debian) muss ggf. angepasst werden, nebst Neustart des MySQL-Servers mit:

```
# /etc/init.d/mysql restart
```

### 9.3 php-mysql

Für die Datenbank-Anbindung über in HTML-Seiten eingebettete PHP-Skripte sind Zusatzmodule verfügbar. PHP benötigt zum Zugriff auf verschiedene Datenbank-Typen (z.B. `postgres`, `oracle`, `mysql`) unterschiedliche und teilweise nicht parallel installierbare (weil nicht miteinander verträgliche) Zusatzmodule nach dem Plugin-Prinzip.<sup>16</sup> PHP-Skripte müssen bei Wechsel des Datenbanktyps z.B. von `postgres` nach `mysql` wegen der syntaktisch unterschiedlichen Funktionen mit i.d.R. erheblichem Aufwand umgeschrieben werden.

Für die Kommunikation mit einer MySQL-Datenbank ist das PHP-Modul `php-mysql` verantwortlich und muss, günstigstenfalls wieder als Softwarepaket mit automatischer Konfiguration, zunächst auf dem Webserver-Rechner installiert werden.

```
# rpm -Uvh php-mysql-version.rpm (RedHat)
# yum install php-mysql (Fedora/RedHat)
# zypper install php5-mysql (SuSE)
# apt-get install php5-mysql (Debian)
```

Bei SuSE ist die MySQL-Anbindung zu PHP schon im PHP-Paket enthalten.

Da es sich hier um ein Plugin für den PHP-Interpreter handelt, ist nicht wie bei der Installation von `mod_php` ein Eintrag in die Apache-Konfigurationsdateien, sondern lediglich eine Änderung in der Konfigurationsdatei des PHP-Interpreters, `/etc/php.ini`, notwendig.

---

<sup>16</sup>Von einigen Programmierern wird dies als Designfehler von PHP angesehen. Eine universelle Programmierschnittstelle zum Zugriff auf Datenbanken wie bei Perl, bei der lediglich einige wenige datenbankspezifische Direktiven bei der Initialisierung gesetzt werden, wäre vielleicht vorteilhafter gewesen.

## Datenbank-Anbindung

---


```
; /etc/php.ini
[PHP]
...
extension=mysql.so

[MySQL]
mysql.allow_persistent = On      ; allow or prevent persistent link
mysql.max_persistent   = -1      ; maximum number of persistent links.
                                ; -1 means no limit
...
```

Diese Änderungen werden i.d.R. automatisch von den in den **php-mysql**-Paketen integrierten Shell-Skripten durchgeführt. Da das PHP-Modul während der Laufzeit von Apache speicherpersistent ist, kann es dennoch notwendig sein, den Apache-Server zum erneuten Laden der Module (mindestens des **mod\_php**-Moduls) zu veranlassen, damit die zusätzlichen Datenbankfunktionen zur Verfügung stehen.

Anschließend sollten die in der `php-mysql`-Erweiterungsbibliothek definierten Funktionen wie `mysql_pconnect()`, `mysql_select_db()`, `mysql_query()`, `mysql_fetch_array()` und viele andere für Ihre PHP-Skripte zur Verfügung stehen.

### Beispiel:



```
<HTML><HEAD><TITLE>Meine Werkzeugkiste</TITLE></HEAD>
<BODY>
<?
// Mit Datenbank-Server verbinden, Datenbank auswählen
mysql_pconnect("localhost","db_benutzer","passwort")
    or die("Datenbankfehler.");
mysql_select_db("kiste");
// Tabelle 'werkzeug' enthält Einträge der Form 'typ','name'
$query="SELECT name FROM werkzeug WHERE typ = 'schraube'";
$result=mysql_query($query);
// Werte aller Tabellenzeilen mit typ='schraube' ausgeben
while($schraube=mysql_fetch_array($result))
    { echo "Schraube gefunden: ".$schraube['name']."<br>\n"; }
?>
</BODY>
</HTML>
```

Abbildung 15: Eine Datenbankabfrage mit der PHP-Erweiterung `php-mysql`

Falls Ihre PHP-Skripte beim Aufruf der MySQL-Zusatzfunktionen Fehlermeldungen der Art „Call to undefined function“ liefern, wurde die `mysql.so`-Zusatzbibli-

othek nicht richtig eingebunden oder passt nicht zu Ihrer installierten PHP-Version. Überprüfen Sie ggf. erneut die PHP-Konfigurationsdatei `/etc/php.ini`.

Einige Websites generieren den vollständigen Inhalt aller Seiten durch `php-mysql`-Datenbankabfragen. Dies hat Vor- und Nachteile:

- + Die Webinhalte können leicht über ein datenbankbasiertes Contentmanagement-System verwaltet werden.
- + Mit Hilfe der Datenbankreplikation ist es möglich, die gleichen Inhalte auf mehreren Webservern anzubieten (Redundanz, Verfügbarkeit).
- + Die Seiten können mit Hilfe von SQL-Statements leicht durchsucht und durch ein on-the-fly-generiertes Inhaltsverzeichnis referenziert werden.
- + Der Zugriff auf die Daten wird durch speicherresistente Bereiche gegenüber einem Dateisystembasierten Server beschleunigt.
- + Die Seiten können, da sie dynamisch erzeugt werden, individuell mit Hilfe von Cookies oder Variablen benutzerspezifisch angepasst werden.
- Da über (CGI-)Skripte zugegriffen wird und die Seiten bei jedem Abruf neu dynamisch generiert werden, ist ein Caching über HTTP-Proxies nicht mehr ohne weiteres möglich. Daher werden alle Seiteninhalte (ggf. auch Bilder) bei jedem Zugriff erneut übertragen, statt aus dem Webcache geladen zu werden. Dies ist schlecht für den Netztraffic, die verfügbare Bandbreite und das monatliche Datenvolumen.
- Große Binärobjekte (sog. BLOBs) müssen beim Laden aus und Speichern in der Datenbank jedesmal eine Konvertierungsroutine durchlaufen. Dieses kostet Rechenzeit und Performance.
- Es ist schwer möglich, individuelle Webseiten oder Bereiche per Backup zu sichern, da die einzelnen Komponenten an verschiedenen Stellen/in verschiedenen Tabellen in der Datenbank untergebracht sind.
- Das Einspielen von neuen Seiten ist nur mit SQL-Kenntnissen oder über entsprechende GUIs möglich, da es keine „Dateien“ im eigentlichen Sinne mehr gibt.

Abbildung 16 zeigt anhand eines kurzen Beispiels, wie WWW-Seiten komplett aus Datenbankinhalten generiert werden können.

## Datenbank-Anbindung

---



*Beispiel:*

```
<? // Mit Datenbank-Server verbinden, Datenbank auswählen
mysql_pconnect("localhost","db_benutzer","password");
mysql_select_db("web");
$query="SELECT header,menu,main,footer FROM inhalt ".
        "WHERE item = '$item'";
if($ergebnis=mysql_query($query) &&
    $seite=mysql_fetch_array($ergebnis))
{
    echo "$seite[header]"; // HTML-Header und Titel
    echo "$seite[menu]"; // Navigationsleiste
    echo "$seite[main]"; // Hauptinhalt
    echo "$seite[footer]"; // Kontaktadresse und HTML-Footer
}
?>
```

Abbildung 16: Ein sehr einfaches PHP-Skript zum Generieren einer Webseite vollständig aus der Datenbank

Die URLs sämtlicher Webseiten könnten in diesem Beispiel so aussehen:

```
http://www.webdb-test.de/system/showitem.php?item=1234
```

Mit der durch `item=` angegebenen Nummer wird in der Datenbank die entsprechende Tabellenspalte mit dem darzustellenden Webinhalt ausgewählt. Hierbei geht (ein weiterer Nachteil dieses Verfahrens) die Zuordnung zwischen den sonst üblichen Dateinamen von HTML-Seiten und dem Inhalt bzw. Titel verloren, und die URLs sind auch nicht so einfach zu merken. Etwas Abhilfe kann hier das im Abschnitt 7.4 auf Seite 78 behandelte Modul **mod\_rewrite** bieten, das die kompliziert aussehenden URLs (fast) wieder wie „normale“ HTML-Dateien aussehen lässt.

```
RewriteEngine on
RewriteRule ^/(.*)\.html$ /system/showitem.php3?item=$1 [PT]
```

Abbildung 17: Vereinfachung der URLs von DB-generierten Seiten mit **RewriteRule**

Nach Einfügen dieser Regel werden die vom Benutzer eingegebenen URLs der Form

```
http://www.webdb-test.de/123.html
```

Apache-intern umgeschrieben in

```
http://www.webdb-test.de/system/showitem.php?item=123
```

Eine ausführliche Anleitung zu `php-mysql` finden Sie auf den Online-Dokumentationsseiten zu PHP unter <http://www.php.net/manual/de/ref.mysql.php>.

### 9.4 Perl-DBI

*Perl-DBI* ist ein Perl-Zusatzmodul, das den Zugriff auf Datenbanken aus Perl-Skripten (bzw. aus in Perl geschriebenen CGI-Programmen) heraus gestattet.

Dies ist zunächst vom Apache-Server unabhängig. Dieser muss lediglich angewiesen werden, die entsprechenden CGI-Skripte (siehe Abschnitt 8.5) auszuführen.

Auf Linux-Standardinstallationen sind normalerweise bereits die wichtigsten Perl-Module (inclusive Perl-DBI mit MySQL-Support) installiert. Die „Nachrüstung“ sollte also in den wenigsten Fällen nötig sein:

SuSE:

```
# rpm -Uvh perl-DBI-version.rpm
# zypper install perl-DBI
```

Fedora/RedHat:

```
# rpm -Uvh perl-DBI-version.rpm
# rpm -Uvh perl-DBD-MySQL-version.rpm
# yum install perl-DBI perl-DBD-MySQL
```

Debian:

```
# apt-get install libdbi-perl
# apt-get install libdbd-mysql-perl
```

Anschließend steht das `DBI.pm`-Perlmodul zur Verfügung, und kann mit `use DBI` in Perl-Skripten verwendet werden.

## **Datenbank-Anbindung**

---

Das folgende Perl-CGI Skript läuft prinzipiell auch ohne Webserver, und liefert eine HTML-Seite (inklusive der notwendigen HTTP-Header) gefüllt mit Datenbankinhalten bei der Ausführung.



*Beispiel:*

```
#!/usr/bin/perl

use DBI;          # Datenbank-Funktionen importieren
use CGI qw(:standard); # CGI-Funktionen importieren

# Übergabeparameter "item" auslesen
$item = param('item');

# Verbindung zur Datenbank herstellen, als "dbuser" einloggen
$dbh = DBI->connect("DBI:mysql:webdb","dbuser","dbpasswd");

# SQL-Kommando vorbereiten
$sql = "SELECT title,main FROM inhalt WHERE item = '$item'";
$stmt = $dbh->prepare($sql);

# Und an die Datenbank schicken
$stmt->execute;

# Ergebnisse abholen
$ret = $stmt->fetchrow_hashref;

print header($ret->{'header'},      # HTTP-Header ausgeben
            start_html($ret->{'title'}), # HTML-Header+Fenstertitel
            h1($ret->{'title'}),      # Überschrift ausgeben
            $ret->{'main'},           # Seiteninhalt ausgeben
            end_html, "\n");         # HTML-Footer ausgeben

# Session schließen und Verbindung zur Datenbank beenden
$stmt->finish;
$dbh->disconnect;
```

Abbildung 18: **gethtmlpage.cgi** - Ein Perl-CGI Programm zum Auslesen und Darstellen einer HTML-Seite aus einer MySQL-Tabellenspalte



*Hinweis:* Das angegebene Beispielprogramm ist wahrlich kein Musterbeispiel für gute Programmieretechnik, da weder Rückgabewerte abgefragt noch Fehlerbehandlungen vorgesehen sind. Hier soll nur der eigentliche Datenbankzugriff veranschaulicht werden.

