

Tutorial 5: *Einführung in MySQL*

Bei MySQL handelt es sich um ein Datenbanksystem, das genauso wie Microsoft Access auf SQL (Abk. für **Structured Query Language**) basiert. MySQL-Datenbanken bestehen aus mehreren Tabellen, die unabhängig voneinander sein können oder auch miteinander verknüpft werden können – z. B. über gemeinsame Felder.

Die Homepage-Produkte der Telekom beinhalten ab der Homepage Basic Skriptfähigkeit (PHP 4, 5 und Perl) sowie eine bzw. mehrere MySQL-Datenbanken. Sie können zwischen zwei Versionen für Ihre Datenbank wählen: MySQL 4.x oder MySQL 5.x. Falls Sie beide Versionen benötigen, z. B. weil Sie bestehende MySQL 4.x Datenbanken importieren möchten, empfehlen wir Ihnen die Homepage Professional, da bei diesem Produkt 10 MySQL-Datenbanken enthalten sind.

Wie nutze ich die Anbindung an eine MySQL-Datenbank?

Neben Serveradresse und Port benötigen Sie einen Benutzernamen mit dazugehörigem Passwort. Alle Informationen finden Sie im Homepagecenter unter "Profi-Anwendungen / MySQL-Datenbank".

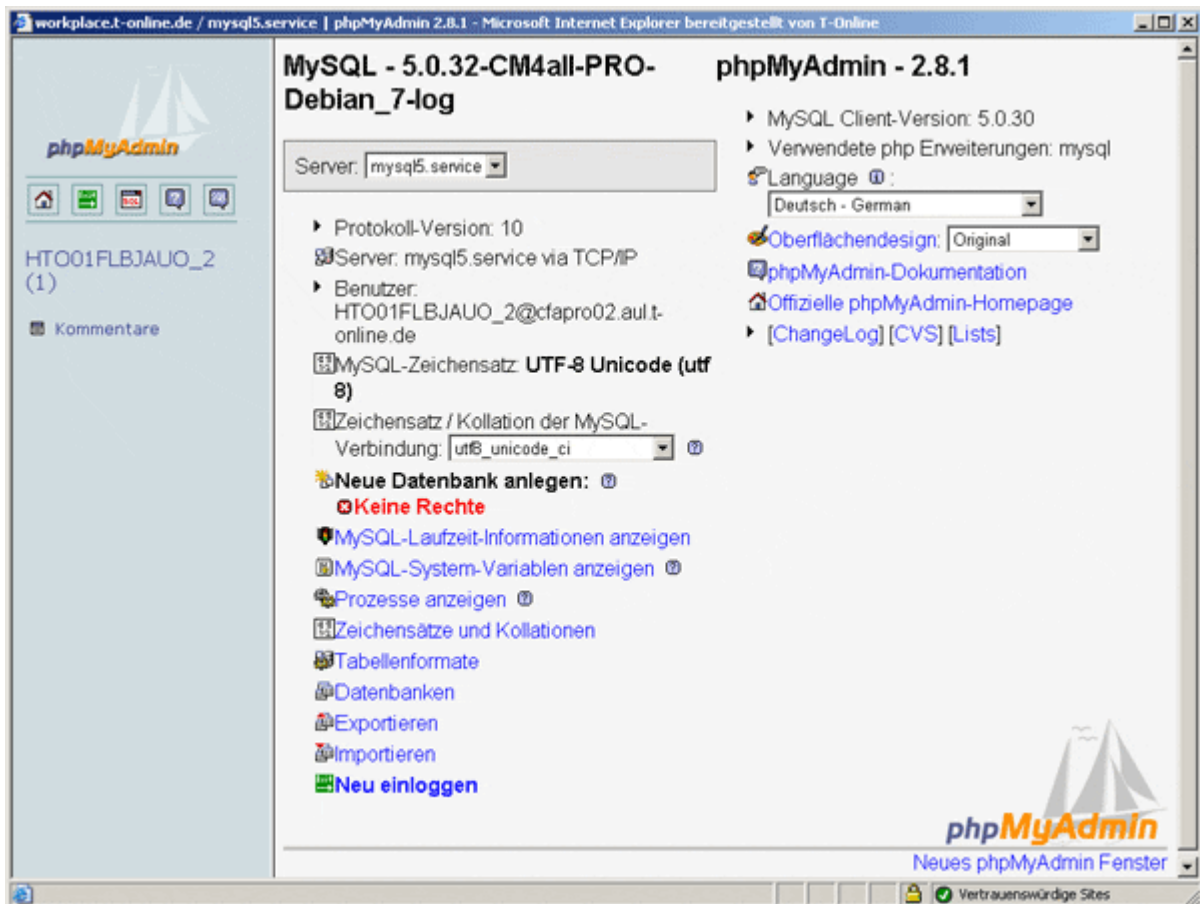
Das für die MySQL-Datenbank benötigte Passwort können Sie im Homepagecenter jederzeit wieder ändern. Über das Homepage Rechtemanagement ist sichergestellt, dass neben dem Besitzer nur Mitarbeiter mit der Rolle "Administrator" oder "Redakteur" Zugriff auf diesen Reiter haben.

Die MySQL-Zugangsdaten werden u.a. benötigt, wenn Skripte auf Datenbanken zugreifen sollen. D.h. sie müssen in den Skripten integriert werden.

Für den komfortablen Zugriff auf Ihre Datenbank steht Ihnen im Homepagecenter das Tool "phpMyAdmin" zur Verfügung. Durch Klicken auf den Namen der Datenbank gelangen Sie direkt zu dieser Anwendung.

Unter der Internet-Adresse <https://workplace.t-online.de/pma/> ist ein direktes Login in das phpMyAdmin-Tool möglich. Sie benötigen hierzu die Zugangsdaten Ihrer MySQL-Datenbank, die Sie wie oben beschrieben im Homepagecenter auslesen können.

Auf der Startseite des phpMyAdmin-Tools sehen Sie links den Namen Ihrer Datenbank (oder Datenbanken, wenn Sie die Homepage Professional nutzen) und in Klammern die Anzahl der darin angelegten Tabellen. Klicken Sie auf den Namen der Datenbank, um neue Tabellen anzulegen.



Tipp: In phpMyAdmin können Sie bestehende Tabellen bequem in Ihre MySQL-Datenbank importieren.

Der große Vorteil von phpMyAdmin ist die bequeme Benutzeroberfläche, in der Sie Ihre Tabellen anlegen, verwalten, ändern oder löschen können. Selbstverständlich können Sie auch gleich Daten eingeben, obwohl die Datenbank ja meist dazu verwendet wird, Informationen zu speichern, die die Besucher Ihrer Website eingeben.

Als erstes sehen Sie die Liste Ihrer Tabellen und zwei Eingabefelder, in denen Sie den Namen für Ihre neue Tabelle sowie die Anzahl der Felder eingeben können. Sie können zwar später jederzeit neue Felder hinzufügen, dennoch sollten Sie sich schon bei der Anlage der Tabelle genau überlegen, welche Informationen Sie abspeichern möchten. Sobald die Tabelle mit den ersten Daten befüllt ist, könnte es unter Umständen zu spät sein, die Struktur der Tabelle noch einmal anzupassen.

Als Beispiel wollen wir eine Tabelle anlegen, in der wir die Kommentare abspeichern, die die Besucher unserer Homepage zu unserem Internetauftritt abgeben. Diese Kommentare sollen auch auf der Homepage nachzulesen sein. Allerdings wollen wir uns die Möglichkeit vorbehalten, die Anzeige der Kommentare zu unterdrücken, ohne dass dazu der Datensatz gelöscht werden muss.

Selbstverständlich wollen wir auch die Namen und E-Mail-Adressen sammeln, die uns die Besucher hinterlassen. Noch bevor wir den HTML- und PHP-Code programmieren, der dazu auf unserer Homepage eingebunden werden muss, müssen wir uns überlegen, wie die Datenbank im Hintergrund aufgebaut sein muss, damit sie allen Anforderungen genügt.

Wir benötigen mindestens 5 Felder:

- Ein Index-Feld, das eindeutig ist und dazu dient, den Datensatz zu identifizieren (ein solches Feld sollten Sie immer einplanen).
- Ein Feld für den Namen
- Ein Feld für die E-Mail-Adresse
- Ein Feld für den Text, den der Besucher eingegeben hat.
- Und ein Feld, in dem festgelegt wird, ob der Kommentar angezeigt wird.

Server: mysql5.service ▶ Datenbank: HTO01FLBJAUO_2 ▶ Tabelle: Kommentare

Feld	Typ	Länge/Set ¹	Kollation	Attribute	Null	Standard ²	Extra
Index	INT				not null		auto_increment
Name	VARCHAR	50			not null		
eMail-Adresse	VARCHAR	40			not null		
Kommentar	TEXT				not null		
Anzeige	BINARY				not null		

Tabellen-Kommentar: Tabellenformat: MyISAM Kollation:

Speichern oder 1 Felder hinzufügen OK

¹ Wenn das Feld vom Typ 'ENUM' oder 'SET' ist, benutzen Sie bitte das Format 'a','b','c',... Wann immer Sie ein Backslash ("\") oder ein einfaches Anführungszeichen (""') verwenden, setzen Sie bitte ein Backslash vor das Zeichen. (z. B.: \'xyz' oder 'a\b')

² Bitte geben Sie jeweils nur einen Standardwert ohne Escape- oder Anführungszeichen an.

Das Index-Feld nennen wir "Index". Es erhält den Typ **INT** (für Integer), weil es ganze Zahlen enthält. Bei Feldern des Typs **INT** müssen wir keine Länge definieren. In der Spalte Aktion können wir festlegen, dass diese Spalte als Index verwendet werden soll, also als das Feld, das den Datensatz eindeutig identifiziert. Damit wir den Index nicht manuell eintragen müssen, wählen wir in der Spalte "Extras" das Attribut "auto_increment". Damit erreichen wir, dass der Index automatisch hochgezählt wird; bei jedem neuen Datensatz ist der Wert für den Index genau 1 größer als beim vorherigen.

Die Felder "Name", "E-Mail-Adresse" und "Kommentar" sind jeweils Textfelder, mit unterschiedlicher Länge. **varchar** Felder sind Textfelder, die nicht größer als 255 Zeichen sind. Da wir für das Eingabefeld "Kommentar" keine solche Begrenzung wünschen, müssen wir hier den Datentyp **text** verwenden.

Das Feld "Anzeige" ist vom Typ **BINARY**. Hier kann also nur 0 oder 1 eingegeben werden. Das Skript auf der Homepage soll später so programmiert werden, dass der Wert immer automatisch auf 1 gesetzt wird, damit die Kommentare sofort auf der Homepage angezeigt werden. Wollen wir einen Kommentar unterdrücken, müssen wir den Wert beim jeweiligen Datensatz manuell auf 0 setzen.

In der Spalte "Null" können Sie festlegen, ob ein Feld leer bleiben darf. Sie können z. B. auch anonyme Kommentare zulassen. Hierzu müssten Sie den Feldern "Name" und "E-Mail-Adresse" den Wert "null" zuweisen.

Unsere Eingaben bestätigen wir anschließend mit **Speichern**. Daraufhin sehen wir die Tabellenstruktur. Oben wird übrigens der verwendete SQL-Code angezeigt.

Server: mysql5.service | Datenbank: HT001FLBJAUO_2 | Tabelle: Kommentare

Anzeigen | Struktur | SQL | Suche | Einfügen | Exportieren | Importieren | Operationen | Leeren

Löschen

Feld	Typ	Kollation	Attribute	Null	Standard	Extra	Aktion
<input type="checkbox"/> Index	int(11)			Nein		auto_increment	[Icon] [Icon] [Icon] [Icon] [Icon] [Icon]
<input type="checkbox"/> Name	varchar(50)	utf8_general_ci		Nein			[Icon] [Icon] [Icon] [Icon] [Icon] [Icon]
<input type="checkbox"/> eMail-Adresse	varchar(40)	utf8_general_ci		Ja	NULL		[Icon] [Icon] [Icon] [Icon] [Icon] [Icon]
<input type="checkbox"/> Kommentar	text	utf8_general_ci		Nein			[Icon] [Icon] [Icon] [Icon] [Icon] [Icon]
<input type="checkbox"/> Anzeige	binary(1)			Nein			[Icon] [Icon] [Icon] [Icon] [Icon] [Icon]

Alle auswählen / Auswahl entfernen markierte: [Icon] [Icon] [Icon] [Icon] [Icon]

Druckansicht | Tabellenstruktur analysieren

Felder hinzufügen | An das Ende der Tabelle | An den Anfang der Tabelle | Nach [Index] OK

Indizes: 0					Speicherplatzverbrauch		Zeilenstatistik	
Name	Typ	Kardinalität	Aktion	Feld	Typ	Verbrauch	Angaben	Wert
Index	INDEX	keine	[Icon] [Icon]	Index	Daten	0 Bytes	Format	dynamisch
Index über [1]		Spalten anlegen OK			Index	1.024 Bytes	Kollation	utf8_general_ci
					Insgesamt	1.024 Bytes	Zellen	0
							Nächste Autoindex	1
							Erzeugt am	22. Mai 2007 um 12:49
							Aktualisiert am	22. Mai 2007 um 12:49

Wenn Sie auf den Reiter "Einfügen" klicken, können Sie auch direkt Daten in die Tabelle eingeben. Auf der linken Seite sehen Sie unter dem Namen der Datenbank übrigens Links zu allen bereits angelegten Tabellen.

Zugriff auf die Datenbank mit PHP

Um mit PHP auf die Daten in unserer Datenbank zugreifen zu können, müssen wir zunächst eine Verbindung herstellen. Dies erfolgt über den Befehl **mysql_connect**. Der Befehl benötigt den Servernamen, den Benutzernamen und das Kennwort als Attribute.

```
<?php
mysql_connect("mysql5.service",
"meinbenutzername","meinkennwort");
?>
```

Damit haben wir eine Verbindung zum Server hergestellt.

Als nächstes benötigen wir eine Verbindung zu der Datenbank, in der wir die Daten abspeichern bzw. auslesen wollen. Dafür gibt es den Befehl **mysql_select_db**.

```
<?php
mysql_select_db("meinedatenbank");
?>
```

Da diese Daten in den meisten Fällen gleich bleiben werden, kann man diesen Teil in einer eigenen Datei speichern und sie anschließend über den Befehl **include** wieder einbinden.

```
<?php
include("verbindungsdaten.php");
?>
```

Da man eine Datenbankverbindung immer auch wieder schließen soll, kann man die Verbindungsdaten beim Verbindungsaufbau auch in einer Variable speichern. Dadurch kann die Verbindung zu dieser Datenbank sehr einfach wieder geschlossen werden:

```
<?php
    $verbindung = mysql_connect("mysql5.service",
    "meinbenutzername","meinkennwort");
    mysql_select_db("meinedatenbank");

    ...

    mysql_close($verbindung);
?>
```

Einfügen von Daten in eine Tabelle

Das Einfügen von Daten in eine Datenbank funktioniert am sinnvollsten mit einem Formular, in das diese Daten eingegeben werden können und das danach übermittelt wird. Dieses Formular befindet sich auf unserer Homepage. Es enthält 3 Eingabefelder, die im HTML-Code bereits eindeutige Bezeichnungen erhalten müssen. Wenn ein Besucher einen Eintrag hinterlässt und die Formulardaten absendet, müssen die Daten erst mal in PHP-Variablen abgespeichert werden.

In unserem Beispiel haben wir uns dafür entschieden, die HTML-Eingabefelder und die dazugehörigen PHP-Variablen gleich zu benennen ('name', 'mail' und 'nachricht'). Das ist aber nicht zwingend erforderlich.

```
<?php
    $name = $_POST["name"];
    $mail = $_POST["mail"];
    $nachricht = $_POST["nachricht"];
?>
```

Der SQL-Befehl zum Einfügen dieser Daten ist **INSERT INTO** mit dem Tabellennamen in Folge. In Klammern dahinter stehen die Bezeichnungen der Spaltennamen (Tabellenfelder). Im zweiten Teil des Befehls, der mit **VALUES** eingeleitet wird, folgen die Werte, die in den neuen Datensatz eingefügt werden sollen. Die Werte sind die zuvor in PHP-Variablen abgespeicherten Benutzereingaben (in einfachen Anführungszeichen) und der Wert 1 für das Binärfeld "Anzeige", das festlegt, ob der neue Kommentar auf unserer Homepage dargestellt werden darf.

Der Befehl **INSERT INTO** wird erst mal in einer weiteren PHP-Variable gespeichert.

```
<?php
    $eintrag = "INSERT INTO Kommentare (Name, E-Mail-Adresse,
    Kommentar, Anzeige) VALUES ('$name', '$mail',
    '$nachricht', 1)";
    $eintragen = mysql_query($eintrag);
?>
```

Da wir für die Spalte "Index" auto_increment verwenden, müssen wir diesen Wert nicht eintragen lassen – er wird ja automatisch um eins höher gesetzt als der vorherige Wert war.

Um den Eintrag auch tatsächlich durchführen zu lassen, verwenden wir abschließend den Befehl **mysql_query**. Erst wenn dieser Befehl ausgeführt ist, stehen die Daten in der Datenbank.

Löschen und Ändern von Datensätzen mit SQL-Befehlen

Das Löschen von Datensätzen aus einer Datenbank ist sehr einfach – manchmal leider zu einfach. Deshalb sollten Sie bei der Programmierung dieses Codes sehr vorsichtig vorgehen. Die Alternative dazu ist natürlich das manuelle Löschen der Datensätze über phpMyAdmin.

```
<?php
$zuloeschen = "DELETE FROM Kommentare WHERE Index < 100";
$loeschen = mysql_query($zuloeschen);
?>
```

Mit dem Befehl **DELETE FROM** wird festgelegt, aus welcher Tabelle Datensätze entfernt werden sollen und nach dem "WHERE" folgt die Bedingung, unter der ein Datensatz gelöscht werden soll. In unserem Beispiel alle Datensätze, deren Index kleiner als 100 ist, also alle älteren Einträge.

Die Änderung von Datensätzen erfolgt in ähnlicher Weise.

```
<?php
$zuaendern = "UPDATE Kommentare Set E-Mail-Adresse =
'max.mustermann@t-online.de' WHERE Name = 'Mustermann';
$aendern = mysql_query($zuaendern);
?>
```

Nach dem Befehl **UPDATE** wird der Tabellename angegeben, gefolgt von einem **Set**, das bestimmt, welche Spalte auf welchen Wert geändert werden soll. In unserem Fall ändern wir die E-Mail-Adresse des Besuchers Mustermann. Der Datensatz wird anhand des Feldes "Name" mit **WHERE** identifiziert.

Wir könnten auch das Feld "Index" heranziehen. Diese Möglichkeit ist besonders interessant, wenn wir unseren Besuchern z. B. die Möglichkeit bieten wollen, Ihre Kommentare zu editieren.

Wenn mehrere Werte auf einmal geändert werden sollen, werden die Daten durch Komma trennt.

```
<?php
$zuaendern = "UPDATE Kommentare Set E-Mail-Adresse =
'max.mustermann@t-online.de', Kommentar = 'Ihre Homepage
gefällt mir sehr gut.' WHERE Name = 'Mustermann';
$aendern = mysql_query($zuaendern);
?>
```

Auslesen von Daten aus einer Tabelle

Zunächst legen wir die gewünschte Abfrage festlegen und speichern sie in einer PHP-Variablen. Wir wollen in diesem Beispiel nur den Namen, nicht aber die E-Mail-Adresse unserer Besucher zusammen mit ihrem Kommentar anzeigen. Deshalb fragen wir das Feld "E-Mail-Adresse" erst gar nicht ab.

```
<?php
$abfrage = "SELECT Name, Kommentar FROM Kommentare";
?>
```

Damit wählen wir die Spalten "Name" und "Kommentar" aus der Tabelle "Kommentare" aus. Mehrere Spaltennamen werden durch Komma getrennt. Wollen wir alle Spalten aus einer Tabelle auslesen, können wir einen Stern verwenden.

```
<?php
  $abfrage = "SELECT * FROM Kommentare";
?>
```

Bei dieser Vorgehensweise wird jeweils der Spaltenname als Variablenname verwendet. Das ist meist ausreichend. Wenn wir andere Variablennamen verwenden möchten, können wir folgende Ergänzung vornehmen:

```
<?php
  $abfrage = "SELECT E-Mail-Adresse AS mail FROM
  Kommentare";
?>
```

Damit ist die E-Mail-Adresse nicht mehr unter der Variable "E-Mail-Adresse" sondern unter "mail" gespeichert.

Um unsere Abfragen auch auszuführen, verwenden wir wieder den Befehl **mysql_query**. Das Ergebnis ist ein ResultSet, das in einer neuen Variable (\$daten) gespeichert wird.

```
<?php
  $daten = mysql_query($abfrage);
?>
```

Die bisherigen Abfragen haben Daten aller Zeilen der kompletten Tabelle zurückgegeben, auch wenn nicht alle Spalten ausgewählt waren. Selbstverständlich können wir unsere Abfragen auch nach bestimmten Kriterien verfeinern, so dass nur Daten aus ausgewählten Zeilen zurückgegeben werden.

Wollen wir z. B. nur die Datensätze auslesen, bei denen der Wert für "Anzeige" weiterhin 1 ist, können wir das durch folgende Bedingung erreichen:

```
<?php
  $abfrage = "SELECT * FROM Kommentare WHERE Anzeige = 1";
?>
```

Das Auswahlkriterium wird mit dem Befehl **WHERE** direkt an den **SELECT**-Teil angeschlossen.

Möchten wir alle Kommentare eines bestimmten Besuchers auslesen, lautet die Bedingung:

```
<?php
  $abfrage = "SELECT * FROM Kommentare WHERE Name =
  'Max Mustermann'";
?>
```

Mit einer weiteren Abfrageart können wir Datensätze sortieren lassen. Das erfolgt mit dem Befehl **ORDER BY**, gefolgt von dem Namen des Feldes, nach dem sortiert werden soll.

```
<?php
  $abfrage = "SELECT * FROM Kommentare ORDER BY Name";
?>
```

Standardmäßig erfolgt die Sortierung von A bis Z. Möchte man sie jedoch umgekehrt sortieren, kann man den Befehl **DESC** anhängen (der Befehl für eine Sortierung von A bis Z wäre **ASC**). Die Bezeichnungen sind vom englischen *descending* (absteigend) bzw. *ascending* (ansteigend) abgeleitet.

```
<?php
$abfrage = "SELECT * FROM Kommentare ORDER BY
Name DESC";
?>
```

Ein weiteres Auswahlkriterium, das wir für unser Vorhaben benötigen, ist der Befehl **LIMIT**. Damit kann erreicht werden, dass nur eine bestimmte Anzahl von Datensätzen abgefragt wird.

Der Befehl wird von einer oder zwei Zahlen gefolgt. Die erste legt fest, ab welchem Datensatz die Abfrage beginnt und die zweite, wieviele Datensätze abgefragt werden sollen. Möchte man beim ersten Datensatz beginnen, ist nur eine Zahl erforderlich: die Anzahl der Datensätze die abgefragt werden sollen.

```
<?php
$abfrage = "SELECT * FROM Kommentare LIMIT 200, 100";
?>
```

In diesem Beispiel werden 100 Datensätze ab dem 200. abgefragt.

Möchte man – was in unserem Beispiel ja sinnvoll ist – anstatt der ersten 100 Datensätze die letzten 100 angezeigt bekommen, müssen die Datensätze nur vorher nach dem Indexwert sortiert und anschließend in eine absteigende Reihenfolge gebracht werden.

```
<?php
$abfrage = "SELECT * FROM Kommentare ORDER BY Index
DESC LIMIT 100";
?>
```

Zum Schluss wird bestimmt, dass lediglich 100 Datensätze ausgegeben werden sollen.

Verarbeitung von ausgelesenen Datensätzen

Da unsere Tabelle ja nicht nur aus einem Datensatz besteht, sind die PHP-Variablen "Index", "Name", "E-Mail-Adresse", "Kommentar" und "Anzeige" keine einfachen Variablen mit nur einem Wert. Sie müssen also in ein Array übergeben werden. Dies erfolgt mit dem Befehl **mysql_fetch_object**.

```
<?php
$daten = mysql_query($abfrage);
while($array1 = mysql_fetch_object($daten))
{
print $array1->Name;
print $array1->E-Mail-Adresse;
print $array1->Kommentar;
}
?>
```

In der ersten Zeile wird die Abfrage ausgeführt. Danach kommt eine while-Schleife, die bewirkt, dass die einzelnen Datensätze während der Übergabe ins Array nach und nach verarbeitet werden. Solange ein Eintrag aus der Variable "ergebnis" als Arrayinhalt in die Variable "array1" geschrieben wird, wird die Schleife ausgeführt. In der Schleife stehen die Befehle, die die tatsächliche Ausgabe auf der Webseite verursachen. Selbstverständlich würde dieser Teil in der Praxis etwas ausführlicher aussehen – Sie wollen ja schließlich, dass die Seite auch ansprechend aussieht.

Variablen freigeben

Nach jeder Datenbankabfrage oder Änderung sollten die Ergebnisvariablen wieder freigegeben werden, bevor eine neue Abfrage erfolgen kann. Dabei bleibt die Variable erhalten, der Inhalt wird aber wieder aus dem Speicher entfernt. Dies erfolgt mit dem Befehl **mysql_free_result(\$freizugebende_Variable);**

Zum Beispiel:

```
<?php
  $abfrage = "SELECT * FROM Kommentare";
  $daten = mysql_query($abfrage);
  ...
  mysql_free_result($daten);
?>
```

Zum Nachlesen und Nachschlagen: Ressourcen im Internet

- Die Homepage der Herstellerfirma ist unter <http://www.mysql.com/> zu finden.
- Eine deutsche Übersetzung des Referenzhandbuches finden Sie unter <http://dev.mysql.com/doc/refman/5.1/de/index.html>
- Die Neuerungen in der Version 5.0 des Datenbankmanagementsystems MySQL können Sie hier nachlesen: <http://dev.mysql.com/doc/refman/5.0/en/mysql-5-0-nutshell.html>
- Auf der Website php.net befindet sich ein eigener Bereich zum Thema MySQL mit PHP: <http://de.php.net/mysql>
- Im PHP-Handbuch werden die MySQL-Funktionen ausführlich erklärt und mit vielen Beispielen veranschaulicht: <http://www.php.net/manual/de/ref.mysql.php>