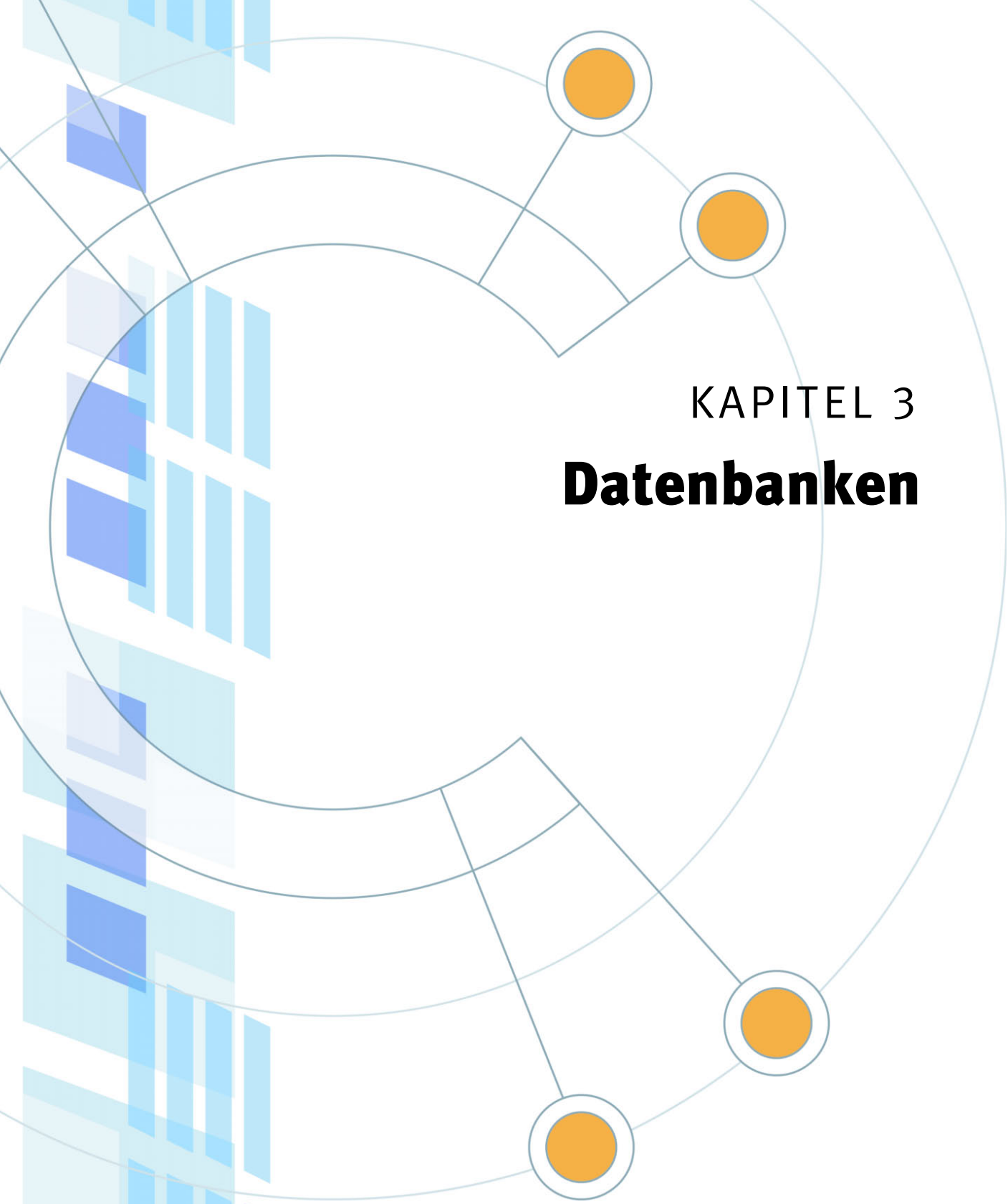
dpi

Charles Wyke-Smith

Codin´ For The Web

Eine Anleitung für Designer, dynamische Websites zu entwickeln

ADDISON-WESLEY



KAPITEL 3

Datenbanken

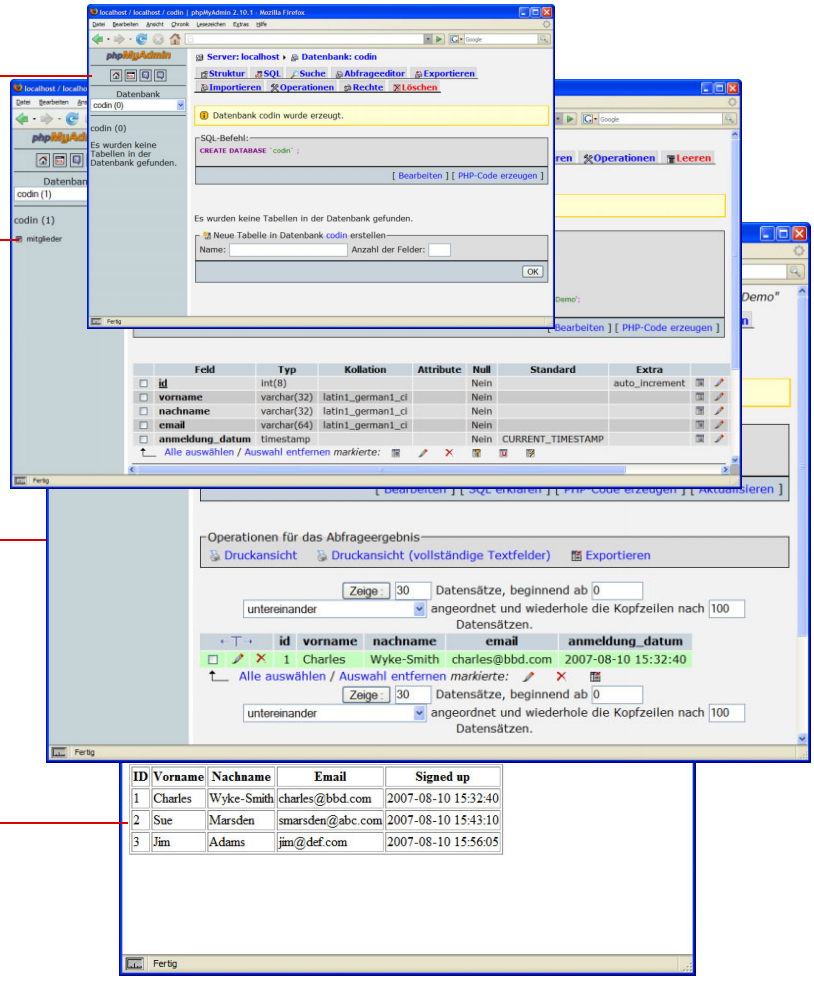
Datenbanken entwickeln

Datenbank erstellen und dort Tabellen einfügen

Datentypen definieren, die die Tabelle enthalten soll

Der Tabelle Einträge hinzufügen

Die Einträge im Browser darstellen



Mehr über die Konzepte von Datenbanken erfahren Sie in Kapitel A.

Nur bei den einfachsten Sites ist es praktisch, die verknüpften Daten in einfachen Textdateien zu speichern. Sobald die Site auch nur ein wenig komplexer wird, erscheinen normalerweise verschiedene Datensätze, und solche Daten werden am besten in einer Datenbank gespeichert. Datenbanken bestehen aus Tabellen, die so wie eine Microsoft Excel-Tabelle eingerichtet sind. Jede Tabelle enthält einen anderen, mit der Site zusammenhängen Datensatz, zum Beispiel Informationen über Mitglieder, Produktspezifikationen und Dateinamen (abhängig davon, was aufgrund der Funktionalität der Site erforderlich ist).

Eine Tabelle hat ein gitterähnliches Layout aus Zeilen und Spalten. Jede Zeile repräsentiert einen *Eintrag* (*record*): eine Instanz der Daten, die in der Tabelle enthalten sind, wie zum Beispiel Informationen über ein bestimmtes eingetragenes Mitglied (Abbildung 3.1). Jede Spalte enthält ein bestimmtes Datenelement, das in den Einträgen enthalten ist. Die jeweiligen Datenelemente nennt man *Werte* (*values*). Der Kasten, der den Wert enthält, wird als *Feld* (*field*) bezeichnet.

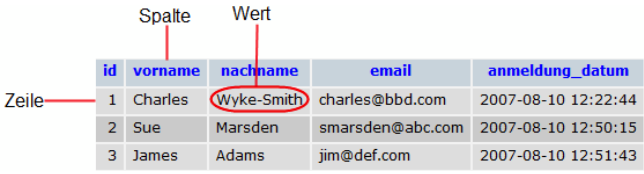


Abbildung 3.1: Anatomie einer SQL-Tabelle

Datenbanktabellen werden über SQL verwaltet. Man kann SQL-Anweisungen an die Datenbank schicken, um erst einmal die Struktur der Tabellen zu erstellen und bei Bedarf zu bearbeiten. Bei den meisten Sites werden solche Aufgaben in erster Linie während der grundlegenden Erstellung der Site durchgeführt. Häufiger wird SQL für Abfragen (Queries) benutzt, also für das Lesen, Schreiben, Verändern und Löschen von Daten innerhalb der Tabellen während des Betriebs der Site.

Die Verwaltung einer Datenbank wird höchst vereinfacht durch solche Tools wie phpMyAdmin – das ist das irreführend bezeichnete Datenbanktool für die Open-Source-Datenbank MySQL.

Bevor Sie überhaupt Daten in einer Datenbank speichern können, müssen Sie die Tabellen erstellen, die die Daten enthalten sollen. In diesem Kapitel werden wir mit einer Tabelle arbeiten, um die Namen und E-Mail-Informationen der Mitglieder unserer Site zu speichern. Diese Daten entsprechen denjenigen aus dem Formular, mit denen wir bereits im vorigen Kapitel gearbeitet haben. Bevor wir diese Tabelle erstellen und Daten einfügen, schauen wir uns die verschiedenen Datentypen an, die in einer Datenbank enthalten sein können.

Datentypen und Datenlängen

Mit jeder Spalte einer Datenbanktabelle ist ein bestimmter Datentyp verknüpft. Alle Felder (die Kästen, in denen die jeweiligen Datenelemente enthalten sind) einer Spalte weisen somit den gleichen Datentyp auf. Bei Erstellung einer Tabelle wird der Datentyp definiert, den die Felder einer jeden Spalte enthalten werden, zum Beispiel Integer (Ganzzahlen), Floats (Dezimalzahlen) oder Varchars (Strings unterschiedlicher Länge), damit die Datenbank weiß, welcher Datentyp in welcher Spalte zu verwalten ist. Es gibt mehr als 25 verschiedene Datentypen, die den Feldern einer Datenbank zugewiesen werden können. In Tabelle 3.1 finden Sie diejenigen, die am häufigsten verwendet werden.

Tabelle 3.1: Häufige Datentypen bei MySQL

TYP	GRÖßE (VERWENDETE DATENMENGE)	BESCHREIBUNG (BEREICH DER KLEINSTEN UND GRÖßTEN ZULÄSSIGEN WERTE)
CHAR[Länge]	Länge Bytes	Ein Feld mit festgelegter Länge zwischen 0 und 255 Zeichen.
VARCHAR(Länge)	String-Länge + 1 Byte	Ein Feld mit festgelegter Länge zwischen 0 und 255 Zeichen.
TINYTEXT	String-Länge + 1 Byte	Ein String mit einer maximalen Länge von 255 Zeichen.
TEXT	String-Länge + 2 Bytes	Ein String mit einer maximalen Länge von 65.535 Zeichen.
MEDIUMTEXT	String-Länge + 3 Bytes	Ein String mit einer maximalen Länge von 16.777.215 Zeichen.
LONGTEXT	String-Länge + 4 Bytes	Ein String mit einer maximalen Länge von 4.294.967.295 Zeichen.
TINYINT[Länge]	1 Byte	Bereich von -128 bis 127 oder 0 bis 255 ohne Vorzeichen.
SMALLINT[Länge]	2 Bytes	Bereich von -32.768 bis 32.767 oder 0 bis 65.535 ohne Vorzeichen.
MEDIUMINT[Länge]	3 Bytes	Bereich von -8.388.608 bis 8.388.607 oder 0 bis 16.777.215 ohne Vorzeichen
INT[Länge]	4 Bytes	Bereich von -2.147.483.648 bis 2.147.483.647 oder 0 bis 4.294.967.295 ohne Vorzeichen
BIGINT[Länge]	8 Bytes	Bereich von -9.223.372.036.854.775.808 bis 9.223.372.036.854.775.807 oder 0 bis 18.446.744.073.709.551.615 ohne Vorzeichen
FLOAT	4 Bytes	Eine kleine Zahl mit einem Gleitkomma
DOUBLE[Länge, Dezimalzahl]	8 Bytes	Eine große Zahl mit einem Gleitkomma
DECIMAL[Länge, Dezimalzahl]	Länge + 1 oder Länge + 2 Bytes	Ein DOUBLE, als String gespeichert, was einen fixen Dezimalpunkt ermöglicht
DATE	3 Bytes	Im Format YYYY-MM-DD
DATETIME	8 Bytes	Im Format YYYY-MM-DD HH:MM:SS
TIMESTAMP	4 Bytes	Im Format YYYYMMDDHHMMSS. Die akzeptierte Bandbreite endet im Jahr 2037.
TIME	3 Bytes	Im Format HH:MM:SS
ENUM	1 oder 2 Bytes	Kurz für *enumeration* (Aufzählung), d.h. jede Spalte kann einen von mehreren möglichen Werten haben.
SET	1, 2, 3, 4 oder 8 Bytes	Wie ENUM, außer dass jede Spalte mehr als einen von mehreren möglichen Werten haben kann.

Wie Sie sehen können, haben viele dieser Datentypen ein optionales Längenattribut (die maximale Zeichenanzahl) für ein Feld. Die für den Speicher benötigte Größe in Bytes, die jedes Feld erfordert, wird erst dann relevant, wenn Sie große Datenbanken erstellen (ist also an dieser Stelle eher von akademischem Interesse). Allerdings sollten Sie die Länge der Felddaten angeben, wo Sie es können. Das zugrunde liegende Konzept ist, den größten Wert (also die größte Anzahl von Zeichen) bereitzustellen, den Sie an Speicherbedarf haben könnten, doch nicht mehr, damit die Datenbank diesem Feld nicht über Gebühr Speicher zuweist. Im Zweifelsfall sollten Sie aber großzügig sein.

Wir schauen uns nun die Daten des Formulars an, um den Feldern die Datentypen und -längen zuzuweisen.

Jeder Dateneintrag besteht aus den folgenden Feldern, die hier mit ihren Feldtypen und der Länge der Daten gezeigt wird, die in jedem Feld enthalten sein können.

FELD	TYP	LÄNGE ODER WERT	WEITERE EINSTELLUNGEN
id	int	20	auto-increment, primary key
vorname	varchar	100	
nachname	varchar	100	
email	varchar	64	
anmeldung_datum	timestamp		

Das `id`-Feld enthält den primären Schlüssel (*unique identifier*) für jeden Eintrag in dieser Tabelle. Dabei handelt es sich um einen Integer, und SQL wird diese Zahl generieren. Das Feld `anmeldung_datum` enthält einen Zeitstempel des Datums und der Erstellungszeit dieses Eintrages. Wie bei `id` wird dieser Wert durch SQL generiert. Das Format ist festgelegt, von daher ist es nicht notwendig, eine Länge zuzuweisen.

Erstellung der Datenbanken und Tabellen

Loggen Sie sich bei phpMyAdmin mit dem Benutzernamen und Passwort ein, das für Sie vom Systemadministrator erstellt wurde. Wenn Sie bei einem kommerziellen Provider sind, bekommen Sie die Informationen von dort.

Eine Instanz (eine installierte Version) von MySQL kann verschiedene Datenbanken enthalten, die jeweils einen eigenen Benutzer-

namen und Passwort haben. Oft ist schon vom Administrator oder ISP, der MySQL auf dem Server installiert hat, eine Datenbank auf dem Server vorinstalliert worden.

Falls das nicht der Fall ist (wie das linke Panel der phpMyAdmin-Oberfläche in Abbildung 3.2 zeigt), müssen Sie eine Instanz erstellen.

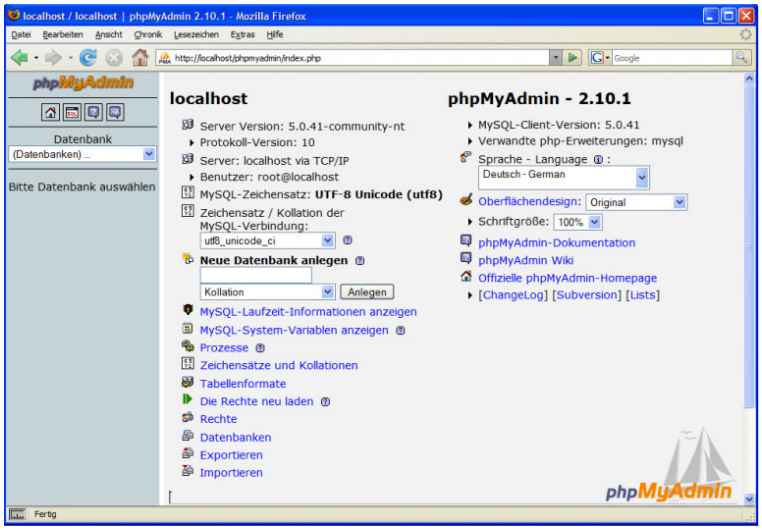


Abbildung 3.2: Im linken Panel der Hauptansicht von phpMyAdmin ist zu sehen, dass noch keine Datenbank erstellt wurde.

Erstellung einer Datenbank

Zum Erstellen einer Datenbank gehen Sie wie folgt vor:

1. Tippen Sie den Namen der zu erstellenden Datenbank in das Feld NEUE DATENBANK ANLEGEN in der ersten Spalte des Hauptbildschirms von phpMyAdmin.

   Unsere Datenbank soll `codin` heißen.

2. Lassen Sie das Dropdown-Menü auf KOLLATION eingestellt (außer Sie wollen mit SQL kyrillische Textstrings oder andere Zeichensätze vergleichen, die nicht Latin sind).

   Ihr Bildschirm sollte wie Abbildung 3.3 aussehen.

Abbildung 3.3: Die Datenbank wird erstellt.

3. Klicken Sie auf ANLEGEN.

Die Datenbank wird erstellt (Abbildung 3.4). Sie sehen eine Bestätigung und das SQL, mit dem die Datenbank erstellt wurde:

```
CREATE DATABASE 'codin';
```

Ferner können Sie sehen, dass in dieser Datenbank noch keine Tabellen enthalten sind – diese werden wir als Nächstes erstellen.

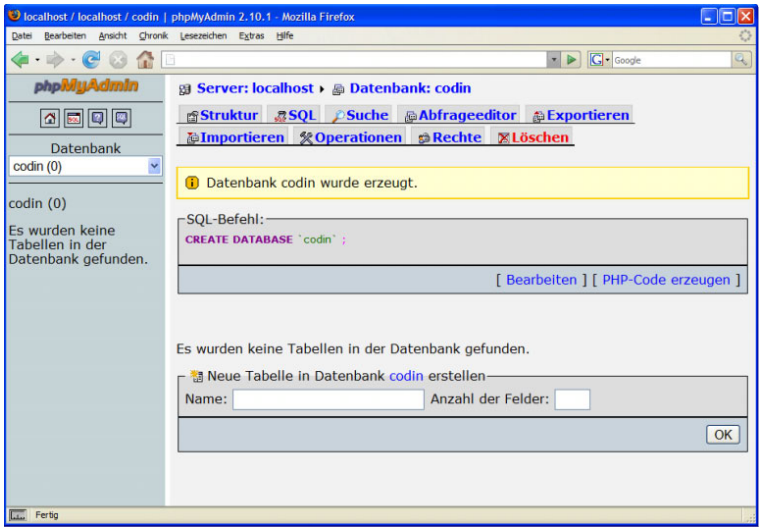


Abbildung 3.4: Die Datenbank wurde erstellt. Beachten Sie, dass die SQL-Query, mit der die Datenbank generiert wurde, angezeigt wird.

Erstellen einer Tabelle

Die MySQL-Datenbank bekommt nun Tabellen.

1. Klicken Sie auf den Tab STRUKTUR oben im Hauptpanel von phpMyAdmin. (Beachten Sie, dass Sie zuerst auf den Link DATENBANK: DATENBANKNAME über den Tabs klicken müssen, wenn Sie weitere Tabellen einfügen wollen.)
2. Im nun angezeigten Formular tragen Sie den Namen der Tabelle und die Anzahl der Felder ein (Abbildung 3.5).

   Die Tabelle bekommt den Namen `mitglieder` und soll `5` Felder haben.

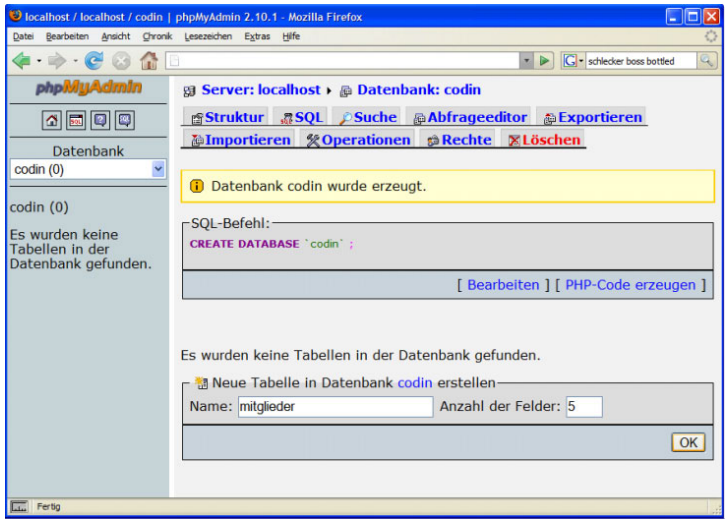


Abbildung 3.5: Der Tab STRUKTUR zeigt die Struktur der Tabellen. Noch gibt es keine Tabellen, doch wir erstellen gleich eine.

In Abbildung 3.5 ist der Tab LÖSCHEN rot hervorgehoben – nicht weil es die aktuelle Auswahl ist, sondern als Warnung: Mit LÖSCHEN können Sie eine Datenbank komplett mit allen Tabellen löschen.

3. Klicken Sie auf OK.

   Nun stellen Sie die Attribute für die Tabelle ein. Der Bildschirm, mit dem Sie nun arbeiten, ist sehr breit. Wahrscheinlich müssen Sie horizontal scrollen, um alle Attribute zu sehen, die man für jedes Feld einstellen kann.

4. Links weisen Sie jedem der fünf Felder entsprechend des darin enthaltenen Datentyps den Feldnamen, Datentyp und Länge oder Wert zu (Abbildung 3.6).

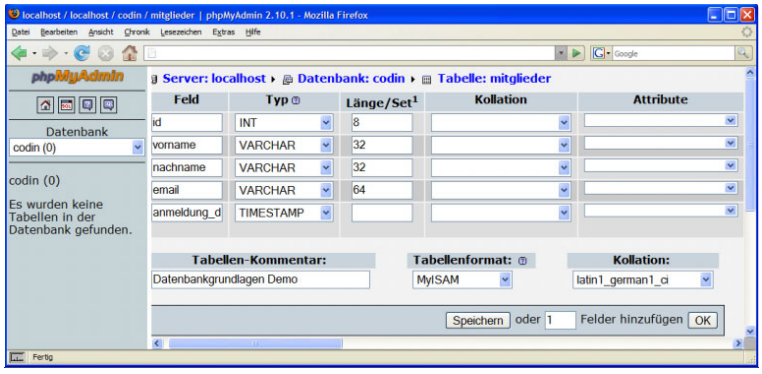


Abbildung 3.6: Einrichten der Datenbankfelder, Datentypen und Längen oder Werte: die linke Bildschirmseite

Sie können später noch den Datentyp eines Feldes ändern. Falls Sie also unsicher sind, tragen Sie einfach etwas ein und prüfen, ob der Datentyp des Feldes Ihnen erlaubt, die gewünschte Art von Daten einzutragen. Sie können zum Beispiel keinen Text in ein `INT`-Feld (Integer) eintragen, doch sobald Sie auf `TEXT` ändern, geht das. Schauen Sie sich die Übersicht der Datentypen auf der MySQL-Seite unter http://dev.mysql.com/doc/refman/5.1/de/data-type-overview.html an.

5. Scrollen Sie im Bildschirm nach rechts und setzen das Attribut EXTRA des `id`-Feldes auf AUTO_INCREMENT.

   Wenn dieses Attribut gesetzt ist, wird der erste Eintrag automatisch den Wert 1 bekommen, der nächste den Wert 2 usw. Wenn ein Eintrag gelöscht wird, wird dessen `id` in der Tabelle nie wieder benutzt.

Wir werden das `id`-Feld auch als primären Schlüssel der Tabelle nutzen.

6. Klicken Sie auf den Radio-Button des primären Schlüssels für das `id`-Feld, um das Feld zu wählen, das der primäre Schlüssel der Tabelle sein wird (Abbildung 3.7).

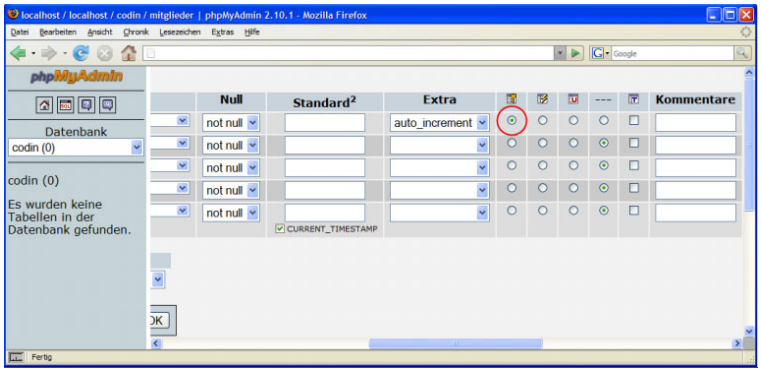


Abbildung 3.7: Klicken Sie auf den Radio-Button, um festzulegen, welches Feld der primäre Schlüssel sein soll.

7. Wenn Sie fertig sind, klicken Sie auf SPEICHERN (siehe Abbildung 3.6).

   Die neue Tabelle ist erstellt (Abbildung 3.8). Beachten Sie die SQL-Query.

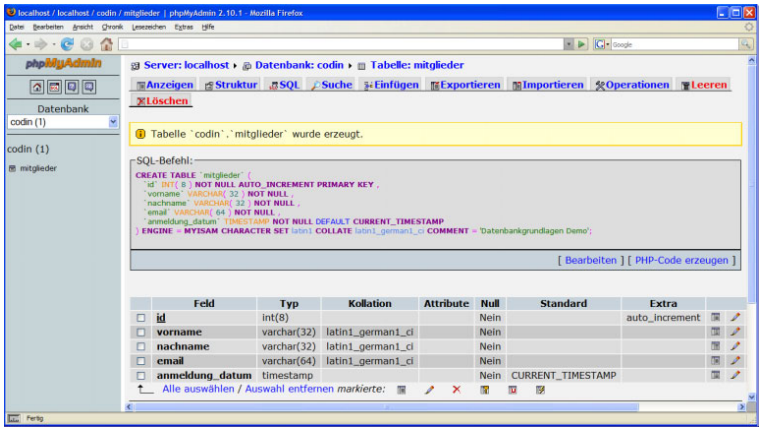


Abbildung 3.8: Die neue Tabelle ist erstellt. Die KOLLATION-Einstellung (für den Zeichensatz)ist auf latin1_german1_ci eingestellt, doch das funktioniert problemlos in allen Situationen, die ich kenne, und wird sich nicht auf die Performance der Tabelle auswirken.

Sie können den Text der SQL-Query kopieren und zur Sicherheit in einer Textdatei ablegen. Wenn Sie die Tabelle erneut erstellen müssen, nachdem Sie vielleicht ein paar Tests gemacht und sie gelöscht haben, können Sie die Tabelle sofort wieder erstellen. Dafür klicken Sie auf den SQL-Tab, fügen die Query in das SQL-Query-Feld ein und klicken auf OK.

Verbindung mit der Datenbank

Nachdem nun die Tabellen in phpMyAdmin erstellt wurden, können wir mit unseren PHP-Skripten die Tabellendaten abfragen.

Jedes SQL, das mit den Daten in einer Datenbank hantiert, wird allgemein als *Query* bezeichnet. Praktisch alle Queries, die Sie schreiben, gehören zu einer der folgenden Arten:

- SELECT: Holt Daten.
- INSERT: Fügt Daten hinzu.
- UPDATE: Ändert Daten.
- DELETE: Löscht Daten.

Bevor Sie von PHP eine SQL-Query ausgeben, öffnen Sie zuerst eine Verbindung und wählen dann eine Datenbank. Denken Sie daran, dass eine einzelne MySQL-Installation mehrere Datenbanken enthalten kann. Also müssen Sie sich nicht nur mit dem Server verbinden, auf dem MySQL läuft, sondern auch angeben, bei welcher Datenbank Sie die Query ausführen wollen.

Öffnen einer Verbindung

Um eine Verbindung zu öffnen, wird folgendes Format verwendet:

```
mysql_connect(Hostserver, Benutzername, Passwort)
```

Eine Verbindung wird mit der Funktion `mysql_connect` aufgebaut. Das erste Argument dieser Funktion ist die Adresse des Servers, auf dem die Datenbank gehostet wird. Dabei kann es sich um eine IP-Adresse oder den Domainnamen eines MySQL-Servers handeln. Falls Datenbank und Webserver auf demselben Computer laufen (ist meist der Fall), funktioniert auch `localhost`. Die anderen beiden Argumente der Funktion `mysql_connect` sind der Benutzername und das Passwort, das bei der Erstellung der Datenbank zugewiesen wurden.

Üblicherweise werden diese drei Zugangsinformationen in Variablen gespeichert, oft in einer `include`-Datei oberhalb des root-Ordners. Doch es ist auch relativ sicher, sie in PHP-Dateien unterhalb der root-Ebene einzubinden, da nur der Output der PHP-Dateien überhaupt auf einer Webseite dargestellt wird.

Listing 3.1: 1_connect.php

```
$hostUrl="localhost";
$benutzerName="codin";
$passwort="codin1234";
$verbindungID = mysql_connect($hostUrl, $benutzerName, $passwort)
  or die ("Tut mir Leid, kann keine Verbindung zur Datenbank
aufbauen");
print $verbindungID . "<br />";
```

Temporäre Darstellung des Verbindungsstatus = Resource id #2 oder ähnlich, wenn okay

Wenn die Verbindung erfolgt, wird `$verbindungID` (oder wie Sie diese Variable auch genannt haben) auf einen Zeiger für diese Verbindung gesetzt. Wenn Sie diesen Codeabschnitt tatsächlich starten, sehen Sie die Ausgabe der Zeigervariablen (Abbildung 3.9).

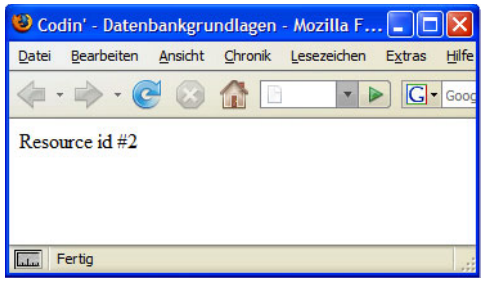


Abbildung 3.9: Die Verbindung mit der Datenbank ist erfolgreich gewesen, der Verbindungszeiger ist gesetzt.

Wie Sie erkennen, ist der Wert für den Zeiger `$verbindungID` nicht sonderlich informativ, doch solange er gesetzt ist, brauchen wir nicht mehr zu wissen. Wie Sie sehen werden, nutzen alle folgenden Interaktionen mit der Datenbank die Variable `$verbindungID` als Referenz.

Beachten Sie auch den Einsatz der Funktion `die`, die aufgerufen wird, falls die Verbindung nicht hergestellt werden konnte. Der Name ist ganz treffend (engl. für *stirb*), weil Ihr Skript abbricht, wenn dieser Fehler ausgelöst wird. In diesem Fall gibt `die` einen Textstring aus, wenn die Verbindung nicht aufgebaut wurde. Allerdings ist `die` auch ein nützliches Tool für kreativen und anwenderfreundlichen Umgang mit Fehlermeldungen bei der Datenbank, weil Sie den Textstring durch den Aufruf einer Funktion ersetzen können, die eine von Ihnen gewünschte Aktion beim Misslingen des Verbindungsaufbaus ausführt. Das könnte zum Beispiel die Umleitung auf eine Seite sein, auf der die Anwender Ihre unterwürfigen Entschuldigungen für das Problem lesen und vielleicht einen Weg aufgezeigt bekommen, wie es gemeldet werden kann.

Auswahl einer Datenbank

Nach Aufbau der Verbindung kann über die PHP-Funktion `mysql_select_db` die Datenbank ausgewählt werden.

Das Format ist wie folgt:

```
mysql_select_db (Datenbankname, Verbindungszeiger)
```

Die Verbindung mit der Datenbank wird so aufgebaut:

```
mysql_select_db("codin", $verbindungID)
  or die ("Auswahl der Datenbank nicht möglich");
```

Das einzige Problem, dem Sie bei diesem recht einfachen Schritt begegnen könnten, wäre ein Tippfehler, zum Beispiel der falsche Datenbankname. Für dieses Beispiel werden wir also diesen Schritt mit dem nächsten kombinieren: der Datenbank einen Eintrag hinzufügen.

Mit INSERT Daten in einer Tabelle einfügen

Nach dem Aufbau der Verbindung und der Auswahl der Datenbank können wir bei der Datenbank eine Query, also eine Operation mit den Daten in der Datenbank, durchführen.

Bisher sind noch keine Einträge in der Mitgliedertabelle vorhanden, also wird zuerst einmal einer eingefügt. Das geschieht über eine SQL-INSERT-Anweisung. Das Format ist:

```
INSERT INTO (feld1, feld2, feld3,...) VALUES (wert1, wert2,
wert3,...)
```

Eine solche INSERT-Anweisung kann so aussehen:

```
INSERT INTO mitglieder (vorname, nachname, email) VALUES
("Charles", "Wyke-Smith", "charles@bbd.com")
```

In diesem Beispiel ist die gesamte Query in einfache Anführungszeichen und Elemente innerhalb der Query in doppelte Anführungszeichen eingeschlossen. Abhängig davon, was sich im String der Query befindet (zum Beispiel PHP-Variablen), müssen Sie möglicherweise diese Reihenfolge umkehren – siehe weiter unten in diesem Kapitel.

Um diese SQL-Anweisung aus dem PHP-Skript in die MySQL-Datenbank zu schicken, verwenden wir es als Argument in der Funktion `mysql_query`.

Das Format der Funktion `mysql_query` ist wie folgt:

```
mysql_query (SQL Query, Verbindungszeiger)
```

Also würde eine Query, mit der ein Eintrag mit meinem Vor- und Nachnamen sowie meiner E-Mail-Adresse eingefügt wird, so aussehen:

```
mysql_query ('INSERT into mitglieder (vorname, nachname, email)
VALUES ("Charles", "Wyke-Smith", "charles@bbd.com")',
$verbindungID)

  or die ("Schreiben des Eintrags in die Datenbank nicht
möglich");
```

Beenden der Verbindung

Obwohl PHP eine Verbindung am Ende eines Skripts automatisch schließt, ist es eine gute Praxis, das nach Ende der Interaktion mit der Datenbank explizit über die Funktion `mysql_close` zu machen. Diese Funktion braucht als Argument den Verbindungszeiger:

```
mysql_close($verbindungID);
```

INSERT: Ein Beispiel

Das folgende Beispiel fasst alles zusammen, was wir bisher angesprochen haben. Dieses Skript wird in der neuen Datenbanktabelle einen Eintrag erstellen.

Listing 3.2: 2_connect.php

```
$hostUrl="localhost";

$benutzerName="codin";

$passwort="codin1234";

$verbindungID = mysql_connect($hostUrl, $benutzerName, $passwort)

  or die ("Tut mir Leid, kann keine Verbindung zur Datenbank
aufbauen");

print $verbindungID . "<br />";

mysql_select_db("codin", $verbindungID)

  or die ("Auswahl der Datenbank nicht möglich");

mysql_query ('INSERT into mitglieder (vorname, nachname, email)
VALUES ("Charles", "Wyke-Smith", "charles@bbd.com")',
$verbindungID)

  or die ("Schreiben des Eintrags in die Datenbank nicht
möglich");

mysql_close($verbindungID);

?>
```

Datenbankverbindung aufbauen

Temporäre Darstellung des Verbindungsstatus = Resource id #2 oder ähnlich wenn okay

Datenbank auswählen

In Datenbank schreiben

Verbindung schließen

Wir haben uns noch nicht angeschaut, wie unser PHP-Skript wieder an die Daten kommt, können aber die erfolgreiche Ausführung des Skripts erkennen, indem wir einfach unsere Tabelle in phpMyAdmin auswählen und auf den Tab ANZEIGEN klicken (Abbildung 3.10).

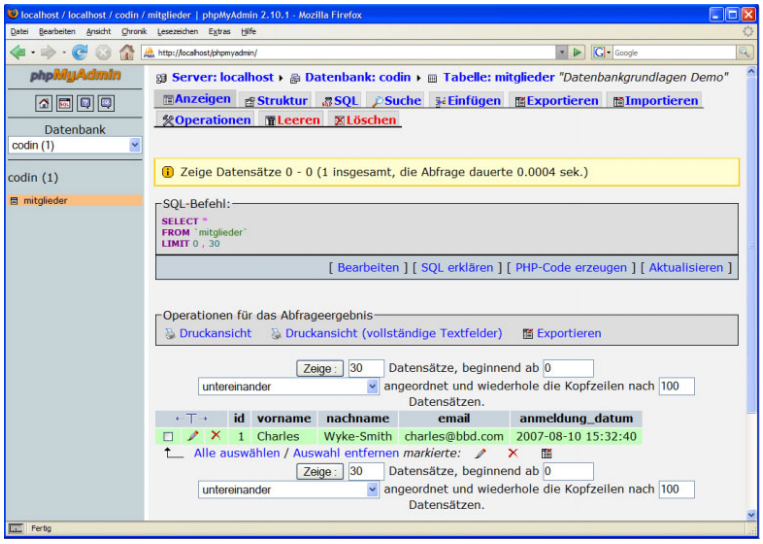


Abbildung 3.10: Der Mitgliedertabelle wurde ein Eintrag hinzugefügt (grün hervorgehoben).

Die SQL-Query in Abbildung 3.10 ist diejenige, die phpMyAdmin gestartet hat, als der BROWSE-Button angeklickt wurde, um die Aufzeichnungen aus der Datenbank zur Anzeige bei diesem Bildschirm anzuwählen.

Sie sehen, dass der neue Eintrag wegen der `auto_increment`-Einstellung im `id`-Feld automatisch die ID 1 bekommen hat. Beachten Sie, dass wegen der Zeitstempeleinstellung im `anmeldung_datum`-Feld auch Datum und Erstellungszeit automatisch eingefügt wurden. Ab jetzt wird bei jedem Laden dieser Seite ein neuer Eintrag mit einer eindeutigen ID und einem Zeitstempel erstellt.

Mit SELECT Daten aus der Datenbank lesen

Um aus der Datenbank etwas zu lesen, muss eine SELECT-Query durchgeführt werden, die wie folgt aussieht:

```
SELECT feld1, feld2,... FROM tabellenname WHERE feldN = ein wert
```

Wie Sie gleich sehen werden, werden wir diese Query in der PHP-Funktion `sql_query` auf die gleiche Weise verwenden wie die INSERT-Query.

Zuerst schauen wir uns an, wie *alle* Einträge in einer Tabelle ausgewählt werden. Also nutzen wir SELECT entsprechend ohne die optionale WHERE-Bedingung:

```
SELECT * FROM mitglieder
```

Das * (Shift-+) bedeutet „alles", und so wird dieser Code alle Daten aus der Mitgliedertabelle auswählen. Nach der Auswahl stehen sie PHP zur Verfügung.

Mit PHP auf Datenbankergebnisse zuzugreifen, erscheint anfangs kompliziert, bis man den Dreh heraus hat. Jedes zurückgegebene Resultat (jeder Eintrag), das oft aus mehreren Feldern besteht – in unserem Beispiel fünf – wird in einer Zeile festgehalten. Eine Zeile im Kontext eines Query-Ergebnisses ist ein assoziatives Array der Namen und Werte der ausgegebenen Datenbankfelder. Nach einer SELECT-Query können wir der Reihe nach auf die Zeilen der Resultate zugreifen, indem wir die Funktion `mysql_fetch_row` in einer `while`-Schleife nutzen:

```
while ($ergebnisreihe = mysql_fetch_row($meineDatenID)) {

}
```

Mach etwas mit den Feldern dieser Zeile

Die `while`-Schleife wird so oft wiederholt, wie Zeilen zurückgegeben werden. Wenn die Schleife also Code enthält, wird dieser Code auf jede Zeile eine nach der anderen angewendet.

Doch wenn Sie versuchen, direkt auf die Variable `$ergebnisreihe`, die das Array enthält, zuzugreifen (zum Beispiel über `print $ergebnisreihe;`), dann gibt der Code nur das Wort `Array` zurück, anstatt die Felder der Zeile auszugeben, was nicht sonderlich hilfreich ist. Wenn Sie den Befehl `print_r $ergebnisreihe;` nehmen, wird PHP die Felder des Arrays ausgeben, doch der Output ist nur im Quellcode formatiert. Also ist es manchmal ganz nützlich, den Output für `print_r` in `<pre></pre>`-Tags einzuschließen. Ich persönlich

arbeite lieber mit einer foreach-Schleife, bei der man dem Output ganz leicht Tags zur Formatierung mitgeben kann. Innerhalb der while-Schleife, die auf jede Zeile zugreift, nehmen wir eine foreach-Schleife, über die wir auf jedes Feld der Zeilen individuell zugreifen können.

Wird so oft wiederholt wie Anzahl Zeilen

Wird so oft wiederholt wie Anzahl Felder in der Zeile

```
while ($ergebnisreihe = mysql_fetch_row($meineDatenID)) {
  foreach ($ergebnisreihe as $feld) {
      print $feld;
  }
   print "<br />";
}
```

Dieser Code wird jedes Feld aus allen Zeilen auf dem Bildschirm ausgeben und fügt nach jeder Zeile einen Umbruch ein.

Listing 3.3: 3_connect_insert_select.php

Nun soll dieser Code direkt nach der INSERT-Query dem vorigen Code für das INSERT-Beispiel hinzugefügt werden.

Code für Verbindungsaufbau und Datenbankauswahl hier ausgelassen – siehe voriger Code

In Datenbank schreiben

Aus Datenbank lesen

Verbindung schließen

```
mysql_query ('INSERT into mitglieder (vorname, nachname, email)
VALUES ("Sue", "Marsden", "smarsden@abc.com")', $verbindungID)
  or die ("Schreiben des Eintrags in die Datenbank nicht möglich");
$meineDatenID = mysql_query("SELECT * from mitglieder",
$verbindungID);
while ($ergebnisreihe = mysql_fetch_row($meineDatenID)) {
foreach ($ergebnisreihe as $feld) {
    print $feld;
    }
    print "<br />";
}
mysql_close($verbindungID);
```

Jetzt werden jedes Mal, wenn ein neuer Eintrag in der Datenbank erfolgt, sofort alle Aufzeichnungen einschließlich des neuen in den Browser geschrieben (Abbildung 3.11).

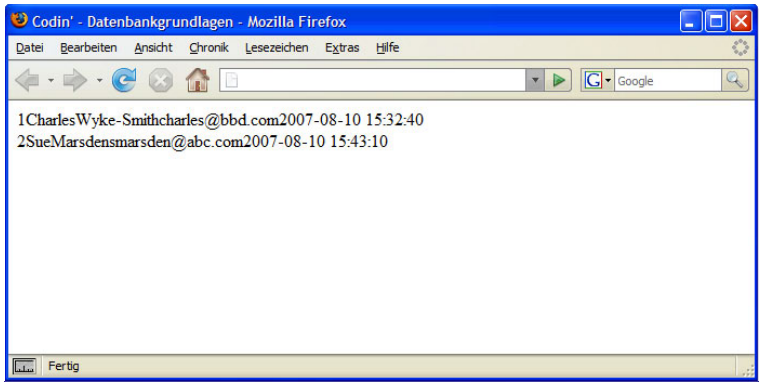


Abbildung 3.11: Die einzigen beiden Einträge in der Datenbanktabelle werden ausgegeben. Die Formatierung könnte allerdings noch verbessert werden.

Ich habe für diesen Schritt die Namen- und E-Mail-Werte in den festkodierten Testvariablen geändert, damit wir nicht in jedem Eintrag die gleichen Daten haben.

Wie Sie sehen können, wird jede Zeile einfach als langer String ohne Leerzeichen zwischen den Werten ausgegeben. Durch etwas XHTML können wir den Code so verbessern, dass das Ergebnis auf dem Bildschirm in einer XHTML-Tabelle ausgegeben wird (Abbildung 3.12).

Listing 3.4: 4_connect_insert_select_table.php

```
$meineDatenID = mysql_query("SELECT * from mitglieder",
$verbindungID);
print "<table border=\"1\">\n";
print '<tr style="font-weight:bold; text-align:center;">
<td>ID</td><td>Vorname</td><td>Nachname</td><td>E-Mail
</td><td>Anmeldedatum</td></tr>';
while ($ergebnisreihe = mysql_fetch_row($meineDatenID)) {
print '<tr>';
foreach ($ergebnisreihe as $feld) {
    print '<td>'.$feld.'</td>';
    }
  print "</tr>\n";
}
print '</table>';
```

Tabelle öffnen und Überschriften schreiben

Tabellenzeile öffnen

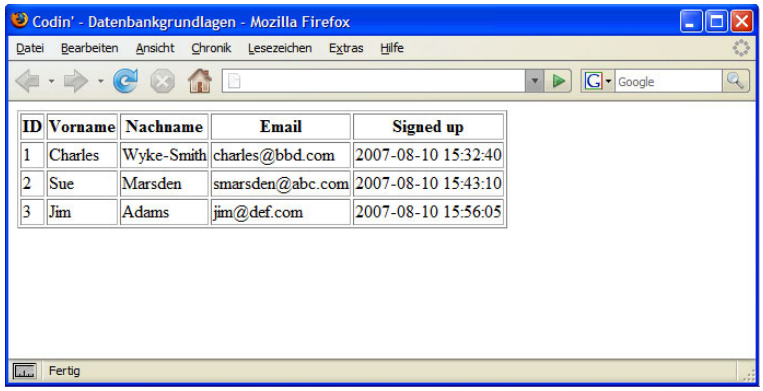
Für jedes Feld

Schreib den Wert in eine Zelle

Zeile schließen

Tabelle schließen

Abbildung 3.12: Durch etwas zusätzliches XHTML werden die Resultate als Tabelle ausgegeben.



ID	Vorname	Nachname	Email	Signed up
1	Charles	Wyke-Smith	charles@bbd.com	2007-08-10 15:32:40
2	Sue	Marsden	smarsden@abc.com	2007-08-10 15:43:10
3	Jim	Adams	jim@def.com	2007-08-10 15:56:05

Auch hier habe ich die festkodierten Namensvariablen geändert, bevor ich den Code gestartet habe.

Es braucht ein wenig Übung, um zu bestimmen, wo die XHTML-Elemente der Tabelle im PHP hinzugefügt werden sollen. Schauen Sie sich den Quellcode in Ihrem Browser an, um sicherzugehen, dass die Tabellenelemente korrekt im XHTML-Output verschachtelt sind.

Die Tabelle wird geöffnet und eine Tabellenzeile mit Kopfzeile wird geschrieben, bevor die `while`-Schleife startet. Bei jedem Start der `while`-Schleife wird eine neue Tabellezeile geöffnet, und bevor sie beendet wird, wird die Zeile geschlossen. Jedes Mal, wenn die `for-each`-Schleifen laufen, wird eine Tabellenzelle geöffnet, die Daten hineingeschrieben und die Zelle dann wieder geschlossen. Nach dem letzten Durchlauf der `while`-Schleife wird die Tabelle geschlossen.

Da Sie nun wissen, wie die Daten einer Tabelle angezeigt werden, können Sie diese Prozedur abändern, um die erhaltenen Daten nach Ihren Wünschen in beliebigen Formaten oder Variablen zu bekommen. Wir wollen nun etwas wählerischer sein – sowohl mit den SQL-Queries als auch den Elementen der Query-Resultate, die mit PHP ausgewählt werden.

Auswahl spezieller Felder

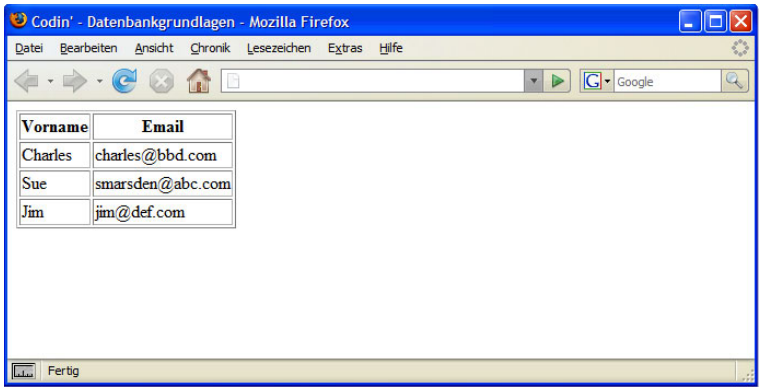
Bisher wurde * („alle“) als Argument für die SELECT-Anweisungen benutzt, und so haben wir auch alle Felder der zurückgegebenen Einträge bekommen. Das soll nun etwas genauer geschehen. Wir modifizieren die vorige Query und ersetzen das * durch die Namen zweier Felder.

Listing 3.5: Ausschnitt aus 5_select_by_field.php

```
$meineDatenID = mysql_query("SELECT vorname, email FROM
mitglieder", $verbindungID);
```

Außer der Modifizierung der festkodierten Spaltenüberschriften sind keine weiteren Änderungen erforderlich (Abbildung 3.13).

Abbildung 3.13: Diese Query gibt nur den Vornamen und die E-Mail-Adresse aller Einträge aus.



Übereinstimmung mit WHERE in SELECT-Anweisungen

Wenn bestimmte Informationen aus dem Eintrag eines Mitglieds gebraucht werden, könnte diese Person ihre E-Mail-Adresse angeben. Die können wir dann nutzen, um den Eintrag dieser Person zu finden und uns die gewünschten Felder ausgeben lassen. Der Query-Code wird dafür wie folgt verändert:

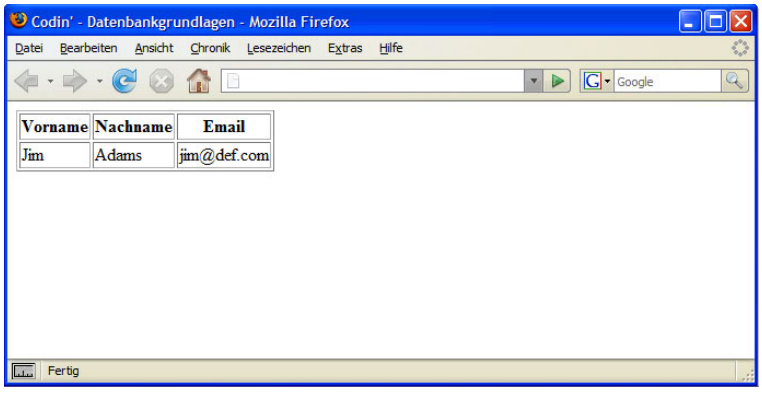
Listing 3.6: Ausschnitt aus 6_select_by_value.php

```
$zielEmail="jim@def.com";

$meineDatenID = mysql_query("SELECT vorname, nachname, email
from mitglieder WHERE email = '$zielEmail'", $verbindungID);
```

Diese Query ergibt nur eine Person (Abbildung 3.14).

Abbildung 3.14: Über die E-Mail-Adresse in der WHERE-Bedingung der SELECT-Anweisung können wir den Namen der Person mit dieser E-Mail-Adresse herausfinden.



Die Werte eines Eintrags mit UPDATE modifizieren

Wenn der Wert in einem Eintrag durch einen anderen ersetzt wird, ist das eine Aktualisierung (Update). Solche Modifikationen werden über die UPDATE-Anweisung durchgeführt. Das Format ist:

```
UPDATE tabellenname SET feld1 = wert1, feld2 = wert2,... WHERE feldN = wertN
```

Nehmen wir an, dass Jim seinen Vornamen auf James ändern will. Wir rufen die UPDATE-Anweisung auf und lassen sie arbeiten, bevor die SELECT-Anweisung aufgerufen wird. So können wir im Browser sehen, wie sich die UPDATE-Anweisung auswirkt (Abbildung 3.15).

Listing 3.7: Ausschnitt aus 7_modify_by_value.php

```
$zielEmail="jim@def.com";

$name_update = "James";

Eintrag bearbeiten
$meineDatenID = mysql_query("UPDATE mitglieder SET vorname =
'$name_update' WHERE email = '$zielEmail'", $verbindungID);

Aus Datenbank lesen
$meineDatenID = mysql_query("SELECT vorname, nachname, email
from mitglieder WHERE email = '$zielEmail'", $verbindungID);
```

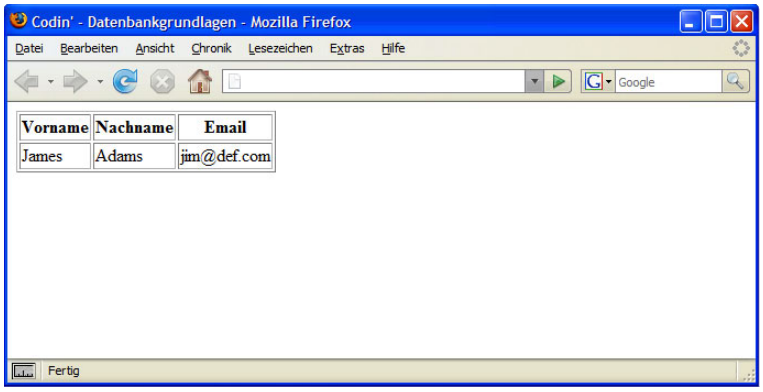


Abbildung 3.15: Durch die UPDATE-Anweisung können die Werte der Felder in einem Eintrag verändert werden. Hier wurde der Vorname des Anwenders verändert (vergleichen Sie mit Abbildung 3.14).

Mit DELETE einen Eintrag entfernen

Es wird unausweichlich der Punkt kommen, an dem Einträge aus der Datenbank gelöscht werden sollen. Das Format der DELETE-Anweisung ist

```
DELETE FROM tabellenname WHERE feldname = wert
```

Momentan enthält die Datenbank drei Mitgliedereinträge: für Charles, Sue und Jim. Sue ist ausgeschieden, also wird ihr Eintrag gelöscht. Hier löschen wir Sues Eintrag und geben dann alle verbleibenden Einträge der Tabelle aus (Abbildung 3.16).

Listing 3.8: Ausschnitt aus 8_delete_by_value.php

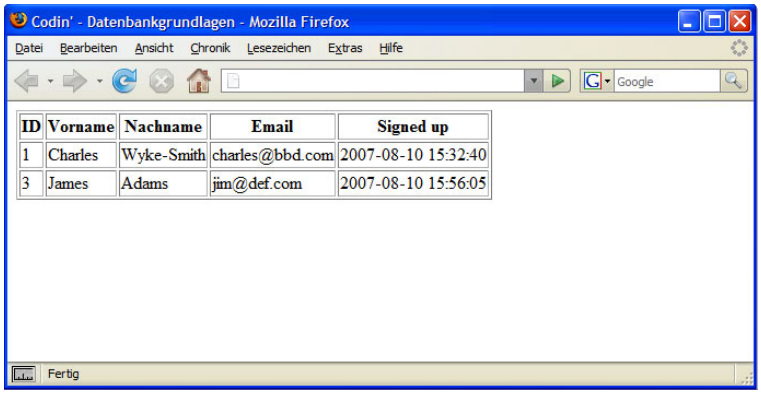
Eintrag löschen

```
$meineDatenID = mysql_query("DELETE FROM mitglieder WHERE
vorname = 'Sue'", $verbindungID);
```

Aus Datenbank lesen

```
$meineDatenID = mysql_query("SELECT * from mitglieder",
$verbindungID);
```

Abbildung 3.16: Sues Eintrag ID 2 wurde aus der Tabelle gelöscht.



ID	Vorname	Nachname	Email	Signed up
1	Charles	Wyke-Smith	charles@bbd.com	2007-08-10 15:32:40
3	James	Adams	jim@def.com	2007-08-10 15:56:05

Warnhinweis für DELETE

Achten Sie besonders darauf, eine WHERE-Bedingung anzugeben, wenn Sie eine DELETE-Query ausgeben. Diese Query wird sofort und unwiderruflich alle Einträge aus der Mitgliedertabelle löschen. Sagen Sie hinterher nicht, ich hätte Sie nicht gewarnt!

Zusammenfassung

Es kann anfänglich eine ziemliche Herausforderung sein, SQL-Anweisungen in die Argumente der PHP-Funktionen, die sich auf SQL beziehen, einfließen zu lassen. Die Beispiele dieses Kapitels sind für Sie hoffentlich nützliche Vorlagen für Ihre speziellen Bedürfnisse.

Wir haben uns noch keine JOIN-Anweisungen angeschaut, also Queries, die auf Daten verschiedenen Tabellen zugreifen. Darum wird es in den nächsten Kapiteln gehen.

Gewappnet mit diesen grundlegenden PHP- und SQL-Techniken können wir uns nun anspruchsvollere Projekte vornehmen.