M Y S Q L - Dokumentation

inhalt

autor: **B. Hartard**

version: **3.2 / 04.2021**

1. einleitung

1.1 MySQL - Structured Query Language

1.1.1 begriffe

MySQL ist ein verwaltungssystem für **SQL**-datenbanken (**My** ist der vorname einer Tochter von Michael Videnius, Miteigentümer der Firma, die diese system geschaffen hat). Das system besteht aus vielen funktionen auf der basis von **PHP**, die funktionen werden in PHP- abschnitten verwendet, Kenntnisse von PHP werden also vorausgesetzt. **SQL** selbst ist eine sprache zur bearbeitung von relationalen datenbanken. Das grundwissen über relationale datenbanken wird vorausgesetzt. Hier werden nur einige begriffe erklärt, die in dieser dokumentation verwendet werden.

Eine datenbank enthält eine oder mehrere datenbank-tabellen, meist als **DB-tabelle** oder noch kürzer als **tabelle** bezeichnet. Eine tabelle enthält **zeilen**, oft auch als **sätze** bezeichnet. Eine zeile besteht aus **spalten**, auch als **tabellen-element** oder einfach als **element** bezeichnet.

hinweis
In dieser dokumentation werden nur die wichtigsten funktionen und anweisungen beschrieben, die benötigt werden, um eine DB-tabellen einzurichten und auf einfache weise abzufragen und zu bearbeiten.

1.1.2 versionen

Es sind zwei versionen in gebrauch: MySQL und MySQLI; sie unterscheiden sich geringfügig in der syntax. MySQL gilt inzwischen als veraltet. In der dokumentation werden beide schreibweisen erklärt, in den beispielen wird ausschließlich MySQLI verwendet.

1.1.3 formales

Bei der beschreibung der MySQL-funktionen und der SQL-anweisungen gilt folgendes:

NORMAL	die angabe ist genau so zu schreiben, ob klein- oder großbuchstaben ist gleich, in dieser dokumentation werden großbuchstaben verwendet.
kursiv oder ***kursiv***	das ist ein platzhalter für einen wert, der hier anzugeben ist
[ ]	angaben in eckigen klammern können wahlweise gemacht werden.
. . .	die vorangehende angabe kann mehrfach wiederholt werden
|	durch einen senkrechten strich getrennte angaben sind alternativ
`Courier`	beispiel in MySQL oder SQL-code

1.2 datentypen

In einer DB-tabelle können die unterschiedlichsten daten gespeichert werden, es gibt nur eine einschränkung: in einer spalte muß der datentyp in allen zeilen gleich sein.

1.2.1 numerische typen

Typ	*Bytes*	*Minimum*	*Maximum*
TINYINT	1	128	127
TINYINT UNSIGNED	1	0	255
SMALLINT [(n)]	2	32768	32767
SMALLINT [(n)] UNSIGNED	2	0	65535
INT [(n)]	4	2147483648	2147483647
INT [(n)] UNSIGNED	4	0	4294967295
FLOAT [(v,n)]	4	dezimalzahl ggf mit angabe der vor und nachkommastellen	
DOUBLE [(v,n)]	8	dezimalzahl ggf mit angabe der vor- und nachkommastellen	

1.2.2 zeichentypen

einfache zeichenketten

VARCHAR(n)	zeichenkette mit variabler länge 1 bis n, n-max = 65.535
CHAR(n)	zeichenkette mit fester länge n, n-max = 255
TEXT	maximal 65.535 bytes, für große zeichen-mengen
BLOB	maximal 65.535 bytes, für große binärdaten-mengen (z.b. grafiken)

datum / uhrzeit

DATE	datum in der form JJJJ-MM-TT
TIME	uhrzeit in der form HH:MM:SS
DATETIME	datum und uhrzeit in der form JJJJ-MM-TT/HH:MM:SS

1.2.3 optionen

Bei der beschreibung einer spalte ist neben dem datentyp meist noch eine option anzugeben.

NULL	spalte muß nicht belegt werden
NOT NULL	spalte muß belegt werden
DEFAULT *wert*	spalte wird mit *wert* vorbelegt, wenn NOT NULL
ZEROFILL	typ INT wird mit führenden nullen aufgefüllt
UNIQUE	inhalt der spalte darf nur in einer zeile vorkommen

1.3 hinweise zu den beispielen

1.3.1 DB-tabelle mitglieder

Die datenbank enthält für die noch folgenden beispiele die DB-tabelle **mitglieder**, die hier kurz vorgestellt wird.

struktur
Die zeilen der tabelle enthalten folgende spalten

name	***datentyp***	***bedeutung***
nummer	int	personalnummer
name	varchar(30)	familienname
vorname	varchar(30)	vorname
abteil	int	abteilung
beitrag	double	monatsbeitrag

1.3.2 darstellung, ausführung der beispiele

Bei den beispielen wird jeweils nur der teil eines PHP-abschnitts gezeigt, der dazu dient, die jeweils behandelte funktion oder anweisung zu erklären. Es wird dabei davon ausgegangen, dass die verbindung zum server bereits hergestellt und die datenbank ausgewählt ist. Auch die mögliche fehlerauswertung wird in der regel nicht gezeigt. Soweit möglich werden bei der anzeige der verschiedenen abschnitte der dokumentation die beispiele auch ausgeführt und das ergebnis angezeigt.

1.3.3 download der beispiele

Die meisten beispiele sind kapitelweise zu einem großen beispiel zusammengefaßt, das ausgeführt werden kann. Beim aufruf der jeweiligen seite wird die variable **$modus** mit dem wert **test** oder **std** übergeben. Das ist nur wegen der einbettung in die dokumentation notwendig. Der code der beispiele kann heruntergeladen werden.

Alle beispiele binden die datei **verbinden-inc.php** ein; mit der datei wird die verbindung zu datenbank hergestellt. Eine geringfügig geänderte fassung dieser datei kann als **verbinden-inc.txt** heruntergeladen werden.

Abschnitt A - mysql-funktionen

2. an- und abmelden

2.1 connect - anmelden

Vor dem ersten zugriff zu einer datenbank muß man sich bei dem server anmelden, wo die datenbank eingerichtet ist.

$con = **mysql_connect**("*host*", "*user*" [, "*kennwort*"]);

$con = **mysqli_connect**("*host*", "*user*" [, "*kennwort*"] [, "*dbname*"]);

$con	als ergebnis wird in einer variablen ein **handle** der verbindung gespeichert, auf den man sich in den sonstigen funktionen bezieht. Wenn keine verbindung zustande gekommen ist, enthält $con den wert **false**.
host	was man hier angibt, erfährt der benutzer von seinem provider
user	auch diese angabe erfährt man vom provider
kennwort	wenn mit dem provider ein kennwort vereinbart ist, muss man es hier angeben
dbname	name der datenbank, die geöffnet werden soll; nur bei **MySQLI** möglich.

Alle angaben können auch in PHP-variablen enthalten sein

hinweis
Es ist sehr praktisch, dass man die seiten einer homepage mit XAMPP auf einem PC austesten und dabei auch eine datenbank bearbeiten kann. In diesem fall gibt man für *host* und *kennwort eine* leere zeichenkette ("") und für *user* "root" an.

2.2 close - abmelden

[$bool =] **mysql_close**([$con]);

[$bool =] **mysqli_close**([$con]);

Die angabe $con ist notwendig, wenn mehrere verbindungen bestehen und nur eine bestimmte geschlossen werden soll. Wenn $con fehlt, werden alle verbindungen geschlossen. Als ergebnis wird true oder false zurückgegeben.

2.3 select_db - datenbank auswählen

Mit dieser funktion wird eine datenbank ausgewählt und geöffnet

[$bool =] **mysql_select_db**("*dbname*" [, $con]);

[$bool =] **mysqli_select_db**($con, "*dbname*");

Für *dbname* ist der name einer datenbank anzugeben. Manche provider lassen nur eine datenbank zu und legen auch deren name fest, trotzdem muss man ihn hier angeben.

Die angabe $con muß bei **MySQL** nur angegeben werden, wenn mehrere verbindung bestehen,
bei **MySQLI** ist die angabe notwendig.

Der name der datenbank kann auch als PHP-variable angegeben werden!

2.4 beispiel

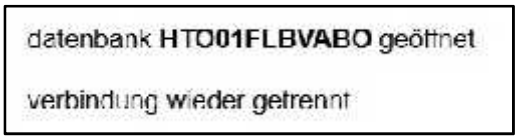
Die aufgerufene seite enthält eine funktion, mit der die verbindung zu einem server hergestellt wird und, abhängig von dem parameter **$dbnr**, eine datenbank geöffnet wird. Die variablen **$con** mit dem handle der verbindung und **$dbname** mit dem namen der datenbank sind global, damit sie auf der seite auch außerhalb der funktion verwendet werden können. Diese funktion wird in allen weiteren beispielen verwendet, der aufruf wird meist nicht gezeigt.

```
<?php
function verbinden($dbnr=0)
{   global $modus, $con, $dbname;

    $host = ". . .";                //  diese angaben werden
    $user = ". . .";                //  natürlich nicht gezeigt
    $pw   = ". . .";
    $dbname = "HTO01FLBVABO";
    $con = mysqli_connect($host, $user, $pw); //  verbindung zum DB-server
    if (!$con)
        $ergeb = false;
    else
    {   if ($dbnr > 0)
        {   $ergeb = mysqli_select_db($con, $dbname); //  wählt DB aus
            if (!$ergeb)
                mysqli_close($con);
        }
        else
            $ergeb = false;
    }
    return $ergeb;
}
?>
```

Irgendwo auf der aufgerufenen seite wird die verbindung hergestellt

```
<?php
   $erg = verbinden(1);               //  mit test-datenbank verbinden
   if ($erg)
   {  echo "<p>datenbank <b>$dbname</b> geöffnet</p>" . $ende;
      mysqli_close($con);
      echo "<p>verbindung wieder getrennt</p>" . $ende;
   }
   else
   {  echo "<p><b>ERR</b> keine verbindung zur datenbank "
            . "<b>$dbname</b></p>" . $ende;
   }
?>
```



3. query - sql-anweisung ausführen

3.1 syntax

Mit dieser funktion wird eine SQL-anweisung ausgeführt; die anweisung wird in anführungszeichen eingeschlossen oder wird in einer PHP-variablen als zeichenkette übergeben.

[$erg =] **mysql_query**("*anweisung*" [, $con]);

[$erg =] **mysqli_query**($con, "*anweisung*");

beispiel

$tabelle = "mitglieder"

$con = wird gesetz durch **mysqli_connect** siehe 2.1

$erg = **mysqli_query**($con, "UPDATE **mitglieder** SET **beitrag** = 25.00 WHERE **nummer** = 1245");

$sql = "UPDATE **$tabelle** SET **vorname** = 'Hans' WHERE **name** = 'Meyer'";

$erg = **mysqli_query**($con, $sql);

UPDATE, SET und WHERE sind spezifischer, fixer text der UPDATE-anweisung, alles andere ist variabel. Die anweisung muss hier noch nicht verstanden werden, sie dient hier nur dazu, die syntax zu erklären.

In dem beispiel wird die SQL-anweisung UPDATE ausgeführt. Dabei werden in der DB-tabelle **mitglieder** folgende änderungen durchgeführt:

- in der zeile, die in der spalte **nummer** den wert **1245** enthält, wird der inhalt der spalte **beitrag** geändert. Die anweisung steht als zeichenkette in der funktion **mysqli_query**.
- In der zeile, die in der spalte **name** den familiennamen **Meyer** enthält, wird die spalte **vorname** geändert. Die anweisung steht in variablen **$sql**, die dann in der funktion steht. Der name der DB-tabelle steht hier in der variablen **$tabelle**.

regeln der syntax
Fehlerträchtig ist die falsche verwendung von anführungszeichen und apostrophen. Es ist unbedingt notwendig, die anweisung in **anführungszeichen** einzuschließen und dann folgende regeln einzuhalten:

1. Zeichenketten in vergleichen (WHERE) mit spalten vom typ zeichen (CHAR, VARCHAR usw.) oder bei zuweisungen (SET) zu solchen spalten werden in apostrophe eingeschlossen, numerische werte werden dagegen direkt in die anweisung geschrieben.
2. PHP-variable, die zeichenketten enthalten, werden in vergleichen mit spalten vom typ zeichen oder bei zuweisungen zu solchen spalten in apostrophe eingeschlossen, numerische variable dagegen werden direkt in die anweisung geschrieben.
3. PHP-variable, die anweisungselemente wie tabellennamen, spaltenamen u.ä. enthalten, werden direkt in die anweisung geschrieben.
4. In den fällen 1 und 2 kann man statt apostrophen auch entwertete anführungszeichen verwenden.

ergebnis von query

Abhängig von der ausgeführten SQL-anweisung ist das ergebnis der funktion **query** unterschiedlich, entsprechend unterschiedlich ist mit dem ergebnis dann umzugehen.

Wie und wo man die folgenden funktionen in der praxis einsetzt, wird bei der behandlung der SQL-anweisungen gezeigt.

beispiele

```
$nam  = "Meyer";            //  zeichenkette in variable
$vor  = "Hans";             //  zeichenkette in variable
$neu  = 25.00;              //  numerischer wert in variable
$num  = 1245;
$tab  = "mitglieder";       //  tabellenname in variable
$par1 = "name";             //  spaltenname in variable
$sql = "UPDATE mitglieder SET vorname = 'Hans'  WHERE name = 'Meyer'"; regel 1

$sql = "UPDATE mitglieder SET beitrag = 25.00 WHERE nummer = 1245";    regel 1

$sql = "UPDATE mitglieder SET vorname = '$vor' WHERE name ='$nam'";    regel 2

$sql = "UPDATE mitglieder SET beitrag = $neu WHERE nummer = $num";     regel 2

$sql = "UPDATE $tab SET vorname = '$vor'  WHERE $par1 = '$nam'";       regel 2 +
                                                                       regel 3
$sql = "UPDATE mitglieder SET vorname = \"Hans\" WHERE name = \"Meyer\"";
                                                                       regel 4
$sql = "UPDATE $tab SET vorname = \"$vor\" WHERE $par1 = \"$nam\"";    regel 3 +
                                                                       regel 4
```

3.2 affected_rows - betroffene tabellen-zeilen

Nach der erfolgreichen ausführung der SQL-anweisungen INSERT, UPDATE, REPLACE und DELETE kann mit dieser funktion abgefragt werden, wieviel zeilen einer DB-tabelle von der jeweiligen aktion betroffen sind. Wenn die funktion **query** das ergebnis **false** geliefert hat, darf die funktion **affected_rows** nicht verwendet werden.

Achtung

Die funktion erwartet als parameter nicht die ergebnis-variable der funktion **query** sondern den verbindungs-**handle.**

$num = **mysql_affected_rows**($con);

$num = **mysqli_affected_rows**($con);

3.3 funktionen nach der SHOW- oder SELECT-anweisung

Die ausführung einer **SHOW**- oder **SELECT**-anweisung liefert als ergebnis eine oder mehrere ergebnis-zeilen, zu deren auswertung verschiedene funktionen verwendet werden. Zu beachten ist, die funktion **query** hat **nicht** das ergebnis **false**, wenn die ausgeführte anweisung zu keinem ergebnis geführt hat, denn auch kein ergebnis ist rein formal eine richtige ausführung der funktion.

3.3.1 num_rows - anzahl der ergebniszeilen

Mit dieser funktion wird die anzahl der ergebniszeilen abgefragt.

$num = **mysql_num_rows**($erg);

$num = **mysqli_num_rows**($erg);

$num enthält die anzahl der zeilen im ergebnis, kann auch 0 (null) sein.

3.3.2 free_result - ergebnis freigeben

Wenn nach ausführung einer **SHOW**- oder **SELECT**-anweisung das ergebnis nicht mehr gebraucht wird, sollte man es freigeben.

mysql_free_result($erg);

mysqli_free_result($erg);

3.3.3 ergebnis auswerten

Mit den folgenden funktionen wird das ergebnis einer **SHOW**- oder **SELECT**-anweisung zeilenweise ausgewertet, d.h. beim ersten aufruf wird die erste ergebniszeile bereitgestellt und intern ein zeiger auf die nächste zeile gesetzt, die dann beim nächsten aufruf geliefert wird. Sinnvoll setzt man die funktionen nur ein, wenn ergebniszeilen geliefert wurden. Die funktionen unterscheiden sich nur in der art und weise, wie die ergebniszeile bereitgestellt wird.

fetch_assoc

Die ergebniszeile wird als assoziatives feld zur verfügung gestellt, als key oder index für die elemente einer zeile dienen die spaltennamen.

$zeile = **mysql_fetch_assoc**($erg);

$zeile = **mysqli_fetch_assoc**($erg);

$vorn = $zeile["**vorname**"]; zugriff auf die spalte **vorname** der ergebniszeile

fetch_row

Die ergebnis-zeile wird als feld mit numerischen indices zur verfügung gestellt.

$zeile = **mysql_fetch_row**($erg);

$zeile = **mysqli_fetch_row**($erg);

$vorn = $zeile[2]; zugriff auf die dritte spalte, d.h. die spalte **vorname** der ergebniszeile

fetch_array

Die ergebnis-zeile wird als feld mit numerischen indices **und** als assoziatives feld zur verfügung gestellt.

$zeile = **mysql_fetch_array**($erg);

$zeile = **mysqli_fetch_array**($erg);

$vorn = $zeile["**vorname**"]; zugriff auf die spalte **vorname** bzw die **dritte** spalte
$vorn = $zeile[2]; der ergebniszeile;

hinweis
Wenn bei den funktionen **fetch_assoc** und **fetch_array** die ergebniszeile als assoziatives feld verwendet wird und die zeile mehrere elemente enthält, kann man die funktion **extract** einsetzen, die für jedes element eine variable mit dem namen des zugehörigen key erzeugt.

extract($zeile); liefert die variablen mit den namen der keys (vgl. PHP-dokumentation)

abschnitt B - SQL-anweisungen

4. datenbank bearbeiten

4.1 datenbank erzeugen / löschen

Die beiden folgenden SQL-anweisungen können nur ausgeführt werden, wenn der provider dem benutzer das anlegen von mehreren datenbanken gestattet oder im testbetrieb unter XAMPP auf einem PC.

CREATE DATABASE [IF NOT EXISTS] *dbname*

dbname name der datenbank als zeichenkette oder in einer variablen.

Mit dieser anweisung wird eine neue datenbank angelegt. Die funktion **query** hat als ergebnis **true**, wenn die datenbank angelegt wurde oder wenn sie schon vorhanden ist und IF NOT EXISTS angegeben wurde. Andernfalls ist das ergebnis **false**.

DROP DATABASE [IF EXISTS] *dbname*

dbname name der datenbank als zeichenkette oder in einer variablen.

Mit dieser anweisung wird eine datenbank gelöscht. Die funktion **query** hat als ergebnis **true**, wenn die datenbank gelöscht wurde oder wenn sie nicht vorhanden ist und IF EXISTS angegeben wurde. Andernfalls ist das ergebnis **false**.

beispiel

```
<?php
   $erg = mysqli_query($con, "CREATE DATABASE IF NOT EXISTS testdatenbank");
   if ($erg)
       echo "<p>datenbank <b>$dbname</b> erzeugt oder vorhanden</p>" . $ende;
   else
       echo "<p>datenbank <b>$dbname</b> nicht erzeugt</p>" . $ende;

   $dbname = "testdatenbank";
   $erg = mysqli_query($con, "DROP DATABASE IF EXISTS  $dbname");
   if ($erg)
       echo "<p>datenbank <b>$dbname</b> gelöscht oder nicht "
            . "vorhanden</p>" . $ende;
   else
       echo "<p>datenbank <b>$dbname</b> nicht geloscht</p>" . $ende;
?>
```

4.2 SHOW DATABASES - alle datenbanken anzeigen

SHOW DATABASES

Die funktion **query** liefert als ergebnis die namen aller datenbanken, die über eine bestehende verbindung zu erreichen sind oder **false** wenn kein zugriff auf datenbanken gegeben ist. Wenn keine datenbanken vorhanden sind, ist das ergebnis **true** ! Man muss beachten, dass das ergebnis in jeder zeile nur den namen einer datenbank enthält, trotzdem ist jede zeile ein feld (array) und der name steht im ersten element des feldes. Vor der auswertung sollte man mit der funktion **num_rows** die anzahl der ergebniszeilen feststellen und dann die zeilen der reihe nach mit der funktion **fetch_row** lesen.

beispiel

```
<?php
   $erg = mysqli_query($con, "SHOW DATABASES");
   if (!$erg)
       echo "<p>ERR: kein zugriff auf datenbanken</p>" . $ende;
   else
   {   $num = mysqli_num_rows($erg);                //  zeilenzahl
       if ($num > 0)
       {   echo "<p>es sind $num datenbanken vorhanden<br />" . $ende;
           for ($z=1; $z<=$num; $z++)
           {   $zeile = mysqli_fetch_row($erg);     //  nächste zeile
               echo "DB nr $z: <b>$zeile[0]</b><br />" . $ende;
           }
           echo "</p>" . $ende;
           mysqli_free_result($erg);                //  ergebnis freigeben
       }
       else
           echo "<p>ERR: keine datenbanken gefunden</p>" . $ende;
   }
?>
```

es sind 2 datenbanken vorhanden
DB nr 1: **Information_schema**
DB nr 2: **HTO01FLBVABO**

4.3 SHOW TABLES - alle tabellen einer datenbank zeigen

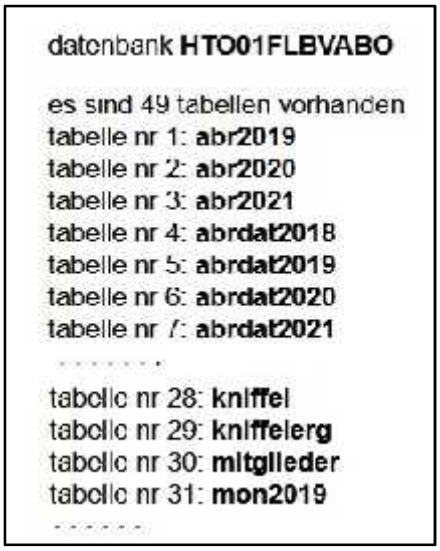
SHOW TABLES FROM *dbname*

dbname name der datenbank als zeichenkette oder in einer variablen.

Die funktion **query** liefert als ergebnis die namen aller DB-tabellen einer datenbank oder **false**, wenn auf die datenbank nicht zugegriffen werden kann. Wenn keine tabellen vorhanden sind, ist das ergebnis **true**! Man muss beachten, dass das ergebnis in jeder zeile nur den namen einer tabelle enthält, trotzdem ist jede zeile ein feld (array) und der name steht im ersten element des feldes. Vor der auswertung sollte man mit der funktion **num_rows** die anzahl der ergebnis-zeilen feststellen und dann die zeilen der reihe nach mit der funktion **fetch_row** lesen.

beispiel
Das beispiel ist dem vorhergehenden sehr ähnlich, es wird daher nur das wesentliche gezeigt.

```
<?php
   verbinden(1);                                  //  verbindung herstellen
   $erg = mysqli_query($con, "SHOW TABLES FROM $dbname");
   $num = mysqli_num_rows($erg);                  //  zeilenzahl
   if ($num > 0)
   {   echo "<p>datenbank <b>$dbname</b></p>" . $ende;
       echo "<p>es sind $num tabellen vorhanden<br />" . $ende;
       for ($z=1; $z<=$num; $z++)
       {   $zeile = mysqli_fetch_row($erg);    //  nächste zeile
           echo "tabelle nr $z: <b>$zeile[0]</b><br />" . $ende;
       }
       echo "</p>" . $ende;
       mysqli_free_result($erg);                  //  ergebnis freigeben
   }
?>
```



4.4 SHOW COLUMNS - tabellen-struktur anzeigen

Die funktion **query** liefert als ergebnis für jede spalte einer DB-tabelle eine zeile oder **false**, wenn die tabelle nicht vorhanden ist. Jede ergebniszeile ist ein feld (array) mit sechs elementen (numerische indices !). Vor der auswertung sollte man mit der funktion **num_rows** die anzahl der ergebniszeilen feststellen und dann die zeilen der reihe nach mit der funktion **fetch_row** lesen und auswerten.

SHOW COLUMNS FROM *tabelle*

tabelle name der DB-tabelle als zeichenkette oder in einer variablen.

Jede ergebniszeile enthält fünf elemente, deren bedeutung nur zum teil bekannt ist.

$zeile[0]	spaltenname	$zeile[3]	key
$zeile[1]	datentyp	$zeile[4]	default-wert
$zeile[2]	?? (Null)	$zeile[5]	weitere angaben

beispiel
Es wird wieder nur das wesentliche angezeigt.

```
<?php
   $tabelle = "mitglieder";                          //  name der tabelle
   $erg = mysqli_query($con, "SHOW COLUMNS FROM $tabelle");
   $num = mysqli_num_rows($erg);                     //  zeilenzahl
   if ($num != 0)
   {   echo "<p>test-datenbank <b>$dbname</b> "
            . "tabelle <b>$tabelle</b></p>" . $ende;
       echo "<p>es sind $num spalten vorhanden</p>" . $ende;
       echo "<table class='tbmit' style='width: 500px'>" . $ende;
       echo "<tr>" . $ende;
       echo "<th>spaltenname</th><th>typ</th><th>Null</th>" . $ende;
       echo "<th>key</th><th>default</th><th>sonstiges</th>" . $ende;
       echo "</tr>" . $ende;
       for ($z=1; $z<=$num; $z++)
       {   $zeile = mysqli_fetch_row($erg);          //  nächste zeile
           echo "<tr>" . $ende;
           echo "<td>$zeile[0]</td><td>$zeile[1]</td>"
                . "<td>$zeile[2]</td><td>$zeile[3]</td>"
                . "<td>$zeile[4]</td><td>$zeile[5]</td>" . $ende;
           echo "</tr>" . $ende;
       }
       echo "</table>" . $ende;
       mysqli_free_result($erg);                     //  ergebnis freigeben
   }
?>
```

test-datenbank **HTO01FLBVABO** tabelle mitglieder

es sind 5 spalten vorhanden

spaltenname	*typ*	*Null*	*key*	*default*	*sonstiges*
nummer	int(11)	YES	UNI		
name	varchar(30)	YES			
vorname	varchar(30)	YES			
abteil	int(11)	YES			
beitrag	double	YES			

5. SELECT - abfragen

5.1 alle spalten abfragen

Bei erfolgreicher ausführung dieser SQL-anweisung liefert die funktion **query** für alle zeilen einer DB-tabelle oder für die zeilen, die der ***bedingung*** entsprechen eine ergebnis-zeile mit allen spalten der tabellen-zeile. Eine ***sortierung*** der ergebniszeilen ist möglich, außerdem kann deren anzahl mit ***limit*** begrenzt werden. Vor der auswertung sollte man mit der funktion **num_rows** die anzahl der ergebniszeilen feststellen und dann die zeilen der reihe nach mit der funktion **fetch_assoc** oder **fetch_array** lesen und auswerten. Wegen et waiger probleme mit geschlossenen umlauten und einigen sonderzeichen siehe nr. 8.4 - besonderheiten.

SELECT * FROM *tabelle* [*bedingung*] [*sortierung*] [*limit*]

tabelle	name der DB-tabelle als zeichenkette oder in einer variablen
bedingung	bedingung für die auswahl der zeilen siehe ziffer 5.3 und 5.4
sortierung	sortierung der ergebnis zeilen siehe ziffer 5.5
limit	beschränkung der ergebniszeilen siehe ziffer 5.6

beispiel
Meist wird man eine ergebniszeile als assoziatives feld behandeln, als **key** dienen dabei die namen der spalten. Es ist sehr einfach, die spalten aus der ergebniszeile in PHP-variable zu bringen und damit weiter zu arbeiten. Wenn man die spalten mit **echo** direkt in einer HTML-anweisung verwenden will, muss man sich strikt an die regeln halten, d.h. entweder den namen der spalte in eine variable bringen oder den namen in anführungszeichen oder apostrophe setzen und den ganzen ausdruck in geschweifte klammern schreiben oder die ausgabe von echo aus passenden teilen zusammen setzen (vgl. PHP-dokumentation, assoziatives feld). Auf die prüfung der anzahl der ergebniszeilen kann man hier verzichten, weil die funktion **fetch_assoc** als bedingung in einer while-schleife steht und diese schleife auch richtig funktioniert, wenn keine ergebnis-zeile vorhanden ist.

Für die echo-anweisung werden die werte aus den spalten wie folgt ausgewertet:

nummer	$ix = "nummer"; damit aus der spalte $zeile[$ix]
name	direkt aus der spalte mit {$zeile["name"]}
vorname	direkt aus der spalte mit $zeile['vorname']
abteil	$abt = $zeile["abteil"];
beitrag	$bei = $zeile["beitrag"]; dann sprintf aufbereiten

```
<?php
   verbinden(1);     //  verbindung herstellen
   $tabelle = "mitglieder";
   $sql = "SELECT * FROM  $tabelle";
   $erg = mysqli_query($con, $sql);
   if (!$erg)
   {  echo "<p>ERR: kein zugriff zur tabelle "
            . "<b>$tabelle</b></p>" . $ende;
   }
```

```
   else
   {   echo "<p> </p>" . $ende;
       echo "<table class='tbmit' "
            . "style='width: 500px'>" . $ende;
       echo "<tr>" . $ende;
       echo "<th>nummer</th><th>name</th>"
            . "<th>vorname</th><th>abt</th>"
            . "<th>beitrag</th>" . $ende;
       echo "</tr>" . $ende;
       $ix = "nummer";
       while ($zeile = mysqli_fetch_assoc($erg))
       {   $abt = $zeile["abteil"];
           $bei = $zeile["beitrag"];
           $bei = sprintf("% 8.2f", $bei);
           echo "<tr>" . $ende;
           echo "<td>$zeile[$ix]</td><td>{$zeile["name"]}</td>"
                . "<td>" . $zeile['vorname'] . "</td>"
                . "<td>$abt</td>" . "<td>$bei</td>" . $ende;
           echo "</tr>" . $ende;
       }
       echo "</table>" . $ende;
       echo "<p> </p>" . $ende;
       mysqli_free_result($erg); //  ergebnis frei geben
   }
   mysqli_close($con);           //  verbindung trennen
?>
```

nummer	name	vorname	abt	beitrag
6714	Maier	Hans	4	15.00
81343	Schmitz	Peter	2	37.50
2297	Mertens	Julia	1	21.50
6715	Maier	Frieda	3	10.00
1723	Rembremerdeng	Wrdlbrmft	1	5.00
6716	Maier	Bubi	6	5.00

hinweise

Die überschriftszeile verwendet die namen der spalten. Da bei der tabelle **mitglieder** diese namen bekannt sind, können sie so wie gezeigt verwendet werden. Sind die namen nicht bekannt, muss man sie mit der anweisung **SHOW COLUMS** ermitteln (siehe 4.4).

Die von der **SELECT**-anweisung gelieferten ergebniszeilen sind **assoziierte** felder, die mit **fetch_array** (siehe 3.3) ausgewertet werden. Dabei werden als schlüssel die spaltennamen verwendet.

Ein etwas aufwendigeres beispiel für die anzueige des tabelleninhalts wird bei 7.4 gezeigt.

5.2 spalten auswählen und abfragen

Bei erfolgreicher ausführung dieser SQL-anweisung liefert die funktion **query** für allen zeilen einer DB-tabelle oder für die zeilen, die der ***bedingung*** entsprechen eine ergebniszeile mit den ausgewählten spalten. Im übrigen gilt alles wie bei 5.1.

SELECT *spalte* [, *spalte* …] FROM *tabelle* [*bedingung*] [*sortierung*] [*limit*]

beispiel
In dem beispiel wird zum lesen der ergebniszeile die funktion **fetch_array** verwendet. Im übrigen gelten die hinweise vom vorhergehenden beispiel.

```
<?php
   $tabelle = "mitglieder";
   $sql = "SELECT name, vorname FROM  $tabelle";
   $erg = mysqli_query($con, $sql);
```

vgl. beispiel von 5.1

```
   while ($zeile = mysqli_fetch_array($erg))
   {   $name = $zeile["name"];
       $vorname = $zeile["vorname"];
       echo "<tr>" . $ende;
       echo "<td>$name</td><td>$vorname</td>" . $ende;
       echo "</tr>" . $ende;
   }
```

name	*vorname*
Maier	Hans
Schmitz	Peter
Merlens	Julia
Maier	Frieda
Rembremerdeng	Wrdlbrmfl
Maier	Bubi

5.3 bedingungen

Bei der SELECT-anweisung kann die auswahl der tabellen-zeilen an bedingugnen verknüpft werden.

WHERE *spalte1 op spalte2* | *wert* [**AND | OR | NOT** *bedingung* . . .]

Der inhalt von ***spalte1*** wird mit dem inhalt von ***spalte2*** oder dem ***wert*** verglichen. Die bedingung kann mit AND, OR oder NOT mit weiteren bedingungen verknüpft werden.

wert spaltten, die verglichen werden, müssen typengleich sein, ein wert muß dem typ der spalte entsprechen, mit deren inhalt er verglichen wird und kann direkt oder in einer variablen angegeben werden. Die regeln für die schreibweise sind strikt einzuhalten, d.h. eine zeichenketten oder eine variable, die eine zeichenkette enthält, steht in apostropen oder entwerteten anführungszeichen (vgl. funktion query 3.1).

op vergleichsoperand =, <>, <, >, <=, >=
man beachte, dass der operand für gleichheit anders als bei PHP hier nur ein zeichen = ist.

beispiel
Gesucht werden die spalten **name, vorname** und **abteil** aus allen tabellen-zeilen in denen die spalte **name** "Maier" enthält oder die spalte **abteil** den wert **2**.

```
<?php
   $tabelle = "mitglieder";
   $sql = "SELECT name, vorname, abteil FROM  $tabelle "
          . "WHERE name = 'Maier' OR abteil = 2";
   $erg = mysqli_query($con, $sql);
```

vgl. beispiel bei 5.1

```
   while ($zeile = mysqli_fetch_assoc($erg))
   {   $name = $zeile["name"];
       $vorname = $zeile["vorname"];
       $abt = $zeile["abteil"];
       echo "<tr>" . $ende;
       echo "<td>$name</td><td>$vorname</td>"
            . "<td>$abt</td>" . $ende;
       echo "</tr>" . $ende;
   }
?>
```

name	*vorname*	*abt*
Maier	Hans	4
Schmitz	Peter	2
Maier	Frieda	3
Maier	Bubi	6

5.4 weitere bedingungen

WHERE spalte **BETWEEN** wert1 **AND** wert2

die bedingung ist erfüllt, wenn für den inhalt der ausgewählten ***spalte*** gilt: >= ***wert1*** und <= ***wert2***
die beiden werte müssen dem typ der *spalte* entsprechen.

WHERE *spalte* **IN** (*wert1* , *wert2* . . .)

es werden nur tabellen-zeilen ausgewählt, bei denen die ausgewählte ***spalte*** einen wert aus der klammer enthält, die werte müssen dem typ der *spalte* entsprechen.

WHERE *spalte* **LIKE** '*muster*'

es werden nur zeilen ausgewählt, bei denen der inhalt der ausgewählten ***spalte*** dem ***muster*** entspricht.
Das *muster* kann auch als variable angegeben werden und muß in apostrophen oder entwerteten anführungszeichen stehen.

Im *muster* bedeutet **%** beliebig viele beliebige zeichen und _ (unterstrich) genau ein beliebiges zeichen.

'**abc**%'	die *spalte* beginnt mit **abc**, gefolgt von beliebig vielen zeichen
'%**abc**'	die *spalte* beginnt mit beliebig vielen zeichen und endet mit **abc**
'**abc**'	die *spalte* enthält nur **abc**
'_**abc**%'	die *spalte* beginnt mit **einem** beliebigen zeichen, gefolgt von **abc**, gefolgt von beliebig vielen zeichen
'__**abc**'	die *spalte* beginnt mit **zwei** beliebigen zeichen, gefolgt von **abc** und nichts weiter

beispiel
Gesucht werden alle zeilen mit allen spalten, in denen die spalte **name** mit einem beliebigen zeichen beginnt, gefolgt von **ai**, gefolgt von beliebig vielen zeichen.

```
<?php
   $tabelle = "mitglieder";
   $sql = "SELECT * FROM  $tabelle WHERE name LIKE '_ai%'";
   $erg = mysqli_query($con, $sql);
```

vgl. beispiel bei 5.1

```
   while ($zeile = mysqli_fetch_assoc($erg))
   {   $name = $zeile["name"];
       $vorname = $zeile["vorname"];
       echo "<tr>" . $ende;
       echo "<td>$name</td><td>$vorname</td>" . $ende;
       echo "</tr>" . $ende;
   }
?>
```

name	*vorname*
Maier	Hans
Maier	Frieda
Maier	Bubi

5.5 ergebniszeilen sortieren

ORDER BY *spalte1* [DESC | ASC] [, *spalte2* [DESC | ASC] . . .]

Die ergebniszeilen werden nach dem inhalt von *spalte1* absteigend (DESC) oder aufsteigend (ASC) sortiert. Wenn die angabe der sortierrichtung fehlt, wird aufsteigend sortiert. Es ist möglich nach mehreren spalten zu sortieren.

beispiel
Gesucht werden alle tabellen-zeilen mit allen spalten. Die ergebniszeilen werden aufsteigend nach der spalte **name** und absteigend nach der spalte **vorname** sortiert.

```
<?php
   $tabelle = "mitglieder";
   $sql = "SELECT * FROM  $tabelle ORDER BY name ASC, vorname DESC";
   $erg = mysqli_query($con, $sql);
```

vgl. beispiel bei, 5.1

```
   while ($zeile = mysqli_fetch_assoc($erg))
   {   $name = $zeile["name"];
       $vorname = $zeile["vorname"];
       echo "<tr>" . $ende;
       echo "<td>$name</td><td>$vorname</td>" . $ende;
       echo "</tr>" . $ende;
   }
?>
```

name	*vorname*
Maier	Hans
Maier	Frieda
Maier	Bubi
Mertens	Julia
Rembremerdeng	Wrdlbrmft
Schmitz	Peter

5.6 ergebniszeilen beschränken

LIMIT *nn*

Die anzahl der ergebniszeilen wird auf ***nn*** zeilen beschränkt

beispiel
Ändert man das vorstehende beispiel auf die folgende abfrage, werden nur 5 tabellen-zeilen ausgewählt, sortiert und ausgewertet, d.h. im ergebnis fehlt "Schmitz Peter". .

```
$sql = "SELECT * FROM  $tabelle ORDER BY name ASC, vorname DESC LIMIT 5";
$erg = mysqli_query($con, $sql);
```

6. SELECT mit funktion

6.1 COUNT - tabellenzeilen zählen

In der SELECT-anweisung kann die funktion COUNT aufgerufen werden, mit der alle zeilen einer DB-tabelle oder die zeilen, die eine bedingung erfüllen gezählt werden. Es können auch die zeilen gezählt werden, die eine bestimmte spalte enthalten (was meist auf das gleiche hinausläuft).

SELECT COUNT(*) [AS *ergeb*] FROM *tabelle* [*bedingung*]

SELECT COUNT(*spalte*) [AS *ergeb*] FROM *tabelle* [*bedingung*]

Wichtig: zwischen COUNT und der öffnenden klammer darf kein zwischenraum sein.

Bei erfolgreicher ausführung der anweisung liefert die funktion **query** genau eine ergebniszeile. Die zeile ist ein feld (array), das im ersten element das ergebnis enthält. Die zeile kann mit der funktion **fetch_row** oder **fetch_array** (mit numerischem index) gelesen werden. Wenn man in der anweisung angibt **AS *ergeb,*** definiert man damit für die ergebniszeile die spalte ***ergeb*** und man kann die ergebniszeile als assoziatives feld behandeln und mit der funktion **fetch_assoc** oder **fetch_array** lesen. Da immer nur eine ergebniszeile geliefert wird, muß man vor der auswertung die anzahl nicht feststellen.

beispiel
Es wird nur der wesentliche teil des codes gezeigt.

```
<?php
   $tabelle = "mitglieder";
   $sql = "SELECT COUNT(*) FROM $tabelle";
   $erg = mysqli_query($con, $sql);
   . . .
   $zeile = mysqli_fetch_row($erg);
   echo "<p>die DB-tabelle $tabelle enthält "
        . "$zeile[0] zeilen</p>" . $ende;
?>
```

die DB-tabelle mitglieder enthält 0 zeilen

6.2 AVG, SUM, MAX, MIN - spalten auswerten

In der SELECT-anweisung können funktionen verwendet werden, mit denen für eine spalte mit numerischem inhalt der durchschnitt, die summe, das maximum oder das minimum aller zeilen einer DB-tabelle oder der zeilen, die eine bedingung erfüllen ermittelt wird. Im übrigen ist alles wie bei der funktion COUNT.

SELECT *function* (*spalte*) [AS *ergeb*] FROM *tabelle* [*bedingung*]

function	**wirkung**
SUM	summe der ausgewählten spalte
AVG	durchschnitt der ausgewählten spalte
MAX	maximum der ausgewählten spalte
MIN	minimum der ausgewählten spalte

Wichtig: zwischen dem namen der funktion und der öffnenden klammer darf kein zwischenraum sein.

beispiel
Es wird nur der wesentliche teil des codes gezeigt.

```
<?php
   $tabelle = "mitglieder";
   $sql = "SELECT SUM(beitrag) AS ergeb "
          . "FROM $tabelle";
   $erg = mysqli_query($con, $sql);
   . . .
   $zeile = mysqli_fetch_array($erg);
   $summe = $zeile["ergeb"];
   $summe = sprintf("% 6.2f", $summe);
   echo "<p>Die summe der beiträge ist "
        . "$summe €</p>" . $ende;
?>
```

6.3 funktion MIN, MAX in einer bedingung

Man kann die funktion **MAX** oder **MIN** auch in einer bedingung der SELECT-anweisung verwenden, um die zeilen zu finden, in denen sich das maximum bzw. minimum einer spalte befindet. In diesem fall liefert die ausführung der funktion **query** nicht das gesuchte maximum oder minimum, sondern die zeilen, die diesen wert enthalten.

SELECT * FROM *tabelle* **WHERE** *spalte* = (**SELECT MAX**(*spalte*) FROM *tabelle*)";

SELECT * FROM *tabelle* **WHERE** *spalte* = (**SELECT MIN**(*spalte*) FROM *tabelle*)";

Wichtig: zwischen dem namen der funktion und der öffnenden klammer darf kein zwischenraum sein.

beispiel
Es wird nur der wesentliche teil des codes gezeigt.

```
<?php
   $tabelle = "mitglieder";
   $sql = "SELECT * FROM $tabelle "
          . "WHERE beitrag = (SELECT MIN(beitrag) FROM $tabelle)";
   $erg = mysqli_query($con, $sql);
   . . .
   $num = mysqli_num_rows($erg);
   echo "<p>$num mitglieder mit minimalbeitrag</p>" . $ende;
   while ($zeile = mysqli_fetch_array($erg))
   {   $name = $zeile["name"];
       $vorname = $zeile["vorname"];
       $beitrag = $zeile["beitrag"];
       $beitrag = sprintf("% 6.2f", $beitrag);
       echo "<p>$name $vorname $beitrag</p>" . $ende;
   }
?>
```

7. DB-tabellen bearbeiten

7.1 DB-tabelle erzeugen

Mit dieser anweisung wird eine neue DB-tabelle erzeugt, bei erfolgreicher ausführung gibt die funktion **query** den wert **true** zurück. Für die anweisung gibt es eine unzahl von argumenten, hier werden nur die allernotwendigsten aufgeführt, die man benötigt um eine ganz einfache tabelle aufzubauen. Bisher ist noch kein wirklicher bedarf für mehr aufgetreten.

CREATE TABLE [IF NOT EXISTS] *tabelle* (*spalte eigenschaften*
[, *spalte eigenschaften*] . . .
[, UNIQUE KEY (*spalte*)])
ENGINE= MyISAM DEFAULT CHARSET=latin1 | utf8

IF NOT EXISTS	diese angabe sollte man machen, für den fall, dass es die tabelle schon gibt.
tabelle	name der tabelle, die erzeugt werden soll.
definition einer spalte	die definition der spalte/n ist in klammern eingeschlossen, mehrere definitionen werden durch komma getrennt.
spalte	name einer spalte, der name muß in der tabelle eindeutig sein
eigenschaften	*datentyp NULL* | NOT NULL | DEFAULT NULL | DEFAULT *wert* [AUTO_INCREMENT] [UNIQUE]
datentyp	datentyp der spalte (vgl. 1.2 datentypen)
NULL	beim erzeugen der zeile, darf für die spalte kein wert angegeben werden
NOT NULL	beim erzeugen der zeile muss für die spalte ein wert angegeben werden
DEFAULT NULL	wenn beim erzeugen der zeile für die spalte kein wert angegeben ist gilt der wert NULL.
DEFAULT *wert*	wenn beim erzeugen der zeile für die spalte kein wert angegeben ist gilt dieser wert, der zum datentyp passen muss
AUTO_INCREMENT	beim erzeugen der zeile wird der bisher höchste in der spalte erreichte wert um eins erhöht und eingetragen. Die angabe DEFAULT *wert* führt zu einem fehler, die anderen werden ignoriert.
UNIQUE	der wert in dieser spalte darf nur in einer zeile auftreten
weitere angaben	
UNIQUE KEY	alternative für die angabe UNIQUE in den eigenschaften
ENGINE=	MyISAM DEFAULT was das bedeutet ist nicht bekannt, aber man muss es angeben
CHARSET=	bestimmt den zeichensatz der daten für die DB-tabelle **latin1** bedeutet **Ansicode, utf8** bedeutet **Unicode**. Empfohlen wird Unicode, denn bei Ansicode kann es mit geschlossenen umlauten und einigen sonderzeichen probleme geben. (siehe 8.4 besonderheiten).

beispiel

```
<?php
   $tabelle = "testdaten";
   $sql = "CREATE TABLE IF NOT EXISTS $tabelle (
       nummer      int           DEFAULT NULL,
       name        varchar(30)   DEFAULT NULL,
       vorname     varchar(30)   DEFAULT NULL,
       abteil      int           DEFAULT NULL,
       beitrag     double        DEFAULT NULL,
       UNIQUE KEY (nummer))
       ENGINE=MyISAM DEFAULT CHARSET=utf8;

   $erg = mysqli_query($con, $sql);
?>
```

Das ergebnis kann mit **SHOW COLUMNS** angezeigt werden und entspricht dem beispiel bei 4.4.

7.2 DB-tabelle kopieren

7.2.1 DB-tabelle vollständig kopieren

Mit dieser anweisung wird eine DB-tabelle vollständig in eine neue tabelle kopiert, bei erfolgreicher ausführung gibt die funktion **query** den wert **true** zurück. Ist die neue tabelle schon vorhanden, wird **false** zurück gegeben, hat man aber IF NOT EXISTS angegeben, ist die rückgabe **true**, kopiert wird natürlich nicht.

CREATE TABLE [IF NOT EXISTS] *tabelle-neu* **SELECT * FROM** *tabelle-alt*

7.2.1 struktur einer DB-tabelle kopieren

Mit dieser anweisung wird nur die struktur einer DB-tabelle in eine neue DB-tabelle kopiert, bei erfolgreicher ausführung gibt die funktion **query** den wert **true** zurück. Ist die neue tabelle schon vorhanden, wird **false** zurück gegeben, hat man aber IF NOT EXISTS angegeben, ist die rückgabe **true**, kopiert wird natürlich nicht.

CREATE TABLE [IF NOT EXISTS] tabelle-neu **LIKE** tabelle-alt

beispiele
Die DB-tabelle **mitglieder** wird in die neue DB-tabelle **allesneu** kopiert, dann wird die struktur von **mitglieder** in die neue DB-tabelle **struktur** kopiert.

```
<?php
   $tabalt  = "mitglieder";
   $tabneu1 = "allesneu";
   $tabneu2 = "struktur";
   $sql = "CREATE TABLE IF NOT EXISTS "
          . "$tabneu1 SELECT * FROM $tabalt";
   $erg = mysqli_query($con, $sql);
   $sql = "CREATE TABLE IF NOT EXISTS "
          . "$tabneu2 LIKE $tabalt";
   $erg = mysqli_query($con, $sql);
?>
```

tabelle **mitglieder** in neue tabelle **allesneu** kopiert

nummer	*name*	*vorname*	*abt*	*beitrag*
6714	Maier	Hans	4	15.00
81343	Schmitz	Peter	2	37.50
2297	Mertens	Julia	1	21.50
6715	Maier	Frieda	3	10.00
1723	Rembremerdeng	Wrdlbrmft	1	5.00
6716	Maier	Bubi	0	5.00

struktur von tabelle **mitglieder** in neue tabelle **struktur** kopiert

Das ergebnis des kopierens der struktur kann mit **SHOW COLUMNS** angezeigt werden und entspricht dem beispiel bei 4.4.

7.3 DB-tabelle umbenennen, löschen

RENAME TABLE *tabelle-alt* **TO** *tabelle-neu*

Mit dieser anweisung wird eine DB-tabelle umbenannt; bei erfolgreicher ausführung gibt die funktion **query** den wert **true** zurück. Wenn die tabelle nicht vorhanden ist oder wenn es schon eine tabelle mit dem neuen namen gibt, ist das ergebnis **false.**

DROP TABLE [IF EXISTS] *tabelle*

Mit der anweisung wird eine tabelle gelöscht, bei erfolgreicher ausführung gibt die funktion **query** den wert **true** zurück. Für den fall, dass es die tabelle nicht gibt, sollte man IF EXISTS angeben, andernfalls gibt **query** den wert **false** zurück.

beispiele
Die DB-tabelle **allesneu** wird gelöscht und die DB-tabelle **struktur** wird in **allerlei** umbenannt

```
<?php
   $tab1 = "allesneu";
   $tab2 = "struktur";
   $tab3 = "allerlei";
   $sql = "DROP TABLE IF EXISTS $tab1";
   $sql = "RENAME TABLE $tab2 TO $tab3";
?>
```

7.4 ALTER TABLE - tabellenstruktur ändern

ALTER TABLE *tabelle aktion*

Mit dieser anweisung wird, abhängig von der ***aktion*** die struktur einer DB-tabelle geändert. Folgende aktionen sind möglich:

ALTER TABLE *tabelle* **CHANGE** *spalte-alt spalte-neu eigenschaften*

ALTER TABLE *tabelle* **MODIFY** *spalte eigenschaften-neu*

ALTER TABLE *tabelle* **ADD** *spalte eigenschaften* [AFTER *spalte*]

ALTER TABLE *tabelle* **DROP** *spalte*

CHANGE	name und eigenschaften einer spalte ändern; man muss die eigenschaften auch angeben, wenn sie sich nicht ändern
MODIFY	eigenschaften einer spalte ändern
ADD	eine neue spalte und ihre eigenschaften wird eingefügt, entweder nach einer bestimmten spalte (bei verwendung von AFTER) oder nach der letzten spalte..
DROP	eine spalte löschen

beispiele

nummer	*einschub*	*name*	*vorn*	*abteil*	*beitrag*	*extra*
6714		Maier	Hans	4	15	
81343		Schmitz	Peter	2	37.5	
2297		Mertens	Julia	1	21.5	
6715		Maier	Frieda	3	10	
1723		Rembremerdeng	Wrdlbrmft	1	5	
6716		Maier	Rubi	6	5	

Es werden vier aktionen durchgeführt:

- die spalte **vorname** wird in **vorn** umbenannt

- nach der spalte **nummer** wird die spalte **einschub** eingefügt

- als neue letzte spalte wird die spalte **extra** angefügt

```
<?php
   $tabelle = "allesneu";
   $sql = "ALTER TABLE  $tabelle CHANGE  vorname vorn "
          . "VARCHAR(30) DEFAULT NULL";
   $erg = mysqli_query($con, $sql);
   $sql = "ALTER TABLE  $tabelle  ADD einschub "
          . "VARCHAR(30) DEFAULT NULL  AFTER nummer";
   $erg = mysqli_query($con, $sql);
   $sql = "ALTER TABLE  $tabelle  ADD  extra "
          . "varchar(20) DEFAULT NULL";
   $erg = mysqli_query($con, $sql);
?>
```

Der inhalt der DB-tabelle wird hier mit einer etwas komplizierteren routine angezeigt als bei kapitel 5.1. Gezeigt wird, wie man verfährt, wenn die anzahl und der name der spalten nicht bekannt ist.

```
//       anzahl und namen der spalten ermitteln
$erg = mysqli_query($con, "SHOW COLUMNS FROM $tabelle");
$nsp = mysqli_num_rows($erg);              //  zeilenzahl = spaltenzahl
$spalte[0] = " ";
for ($z=1; $z<=$nsp; $z++)
{    $zeile = mysqli_fetch_row($erg);      //  nächste zeile
     $spalte[$z] = $zeile[0];              //  spaltenname aus zeile
}
mysqli_free_result($erg);                  //  ergebnis freigeben
//       alle zeilen der tabelle anzeigen
$sql = "SELECT * FROM  $tabelle";          //  alle zeilen aus DB-tab
$erg = mysqli_query($con, $sql);
$z = mysqli_num_rows($erg);                //  anzahl ergebnis-zeilen
echo "<table class='tbmit' style='width: 500px'>" . $ende;
echo "<tr>" . $ende;
for ($x=1; $x<=$nsp; $x++)                 //  spaltennamen in titelzeile
     echo "<td><i>$spalte[$x]</i></td>";
echo "</tr>" . $ende;
//       ergebnis zeilenweise verarbeiten
while ($zeile = mysqli_fetch_assoc($erg))
{    echo "<tr>" . $ende;
     for ($x=1; $x<=$nsp; $x++)            //  ergebnis-zeile spaltenweise
     {    $key = $spalte[$x];              //  key ist spaltenname
               $wert = $zeile[$key];       //  wert aus spalte
               echo "<td>$wert</td>";
     }
     echo "</tr>" . $ende;
}
echo "</table>" . $ende;
```

8. zeilen bearbeiten

8.1 INSERT - zeilen erzeugen

Mit dieser anweisung werden eine oder mehrere neue zeilen in eine DB-tabelle eingefügt. Wenn die anweisung mit **query** erfolgreich ausgeführt wurde, kann man die anzahl der eingefügten zeilen mit der funktion **affected_rows** (vgl. 3.2) abfragen.

INSERT INTO *tabelle* (*spalte1* [, *spalte2* . . .])
VALUES (*wert1* [, *wert2* . . .])
[, (*wert1* [, *wert2* . . .])] . . .

tabelle name der tabelle, für die zeilen erzeugt werden

spalte name einer spalte, für die ein wert eingefügt wird, es können mehrere namen angegeben werden, die durch komma getrennt sind, die namen sind in eine runde klammer eingeschlossen. Beim erzeugen einer zeile müssen hier nur die spalten mit der eigenschaft NOT NULL angeführt werden.

wert für jede zuvor aufgeführte spalte muss ein wert angegeben werden, die werte für eine zeile sind in runde klammern eingeschlossen. Für mehrere zeilen wird für jede zeile eine solche folge von werten, getrennt durch komma, angegeben.

hinweise zu den werten
- Die werte müssen dem datentyp der jeweiligen spalte entsprechen.
- Werte können auch in variablen angegeben werden.
- Numerische werte werden einfach als numerischer ausdruck in die anweisung geschrieben.
- Zeichenketten und variable, die zeichenketten enthalten sind, sind in apostrophe oder entwertete anführungszeichen einzuschließen.
- Mit geschlossene umlaute und einige sonderzeichen kann es besonders beim zeichensatz Ansicode Probleme geben. (siehe dazu. 8.4 besonderheiten).

beispiel
Es werden vier zeilen erzeugt und dabei die verschiedenen möglichkeiten für die angabe der werte gezeigt. Zu beachten sind hier die werte für die erste spalte (nummer). In dem beispiel wird die funktion **affected_rows** verwendet, sie liefert die anzahl der betroffenen, d.h. erzeugten zeilen. Die tabelle **testdaten** hat die gleiche struktur wie die bekannte tabelle **mitglieder**.

```
<?php
     verbinden(1)
     $tabelle = "testdaten";
     $name = "Meyer";
     $vorn1 = "Otto";
     $vorn2 = "Lisa";
     $abt = 2;
     $bei = 15.50;
     $num = 1001;
     $sql = "INSERT INTO $tabelle (nummer, name, vorname,
            abteil, beitrag) VALUES
            (1000,'Schulze', 'Hans', 1, 15.00),
            ($num, '$name', '$vorn1', $abt, $bei),
            ($num + 1, \"$name\", \"$vorn2\", $abt, $bei),
            ($num + 2, 'Müller', 'Frieda', 3, 10.00)";
     $erg = mysqli_query($con, $sql);
     if ($erg)
     {   $num = mysqli_affected_rows($con);
         echo "<p>$num zeilen erzeugt</p>";
     }
?>
```

4 zeilen erzeugt

nummer	*name*	*vorname*	*abt*	*beitrag*
1000	Schulze	Hans	1	15.00
1001	Meyer	Otto	2	15.50
1002	Meyer	Lisa	2	15.50
1003	Müller	Frieda	3	10.00

8.2 UPDATE - zeilen ändern

Mit der anweisung werden in allen oder ausgewählten zeilen einer DB-tabelle eine oder mehrere spalten geändert. Wenn keine bedingung angegeben ist, werden alle zeilen geändert. Für die werte gelten die hinweise der INSERT-anweisung. Wenn die anweisung mit **query** erfolgreich ausgeführt wurde, kann man die anzahl der geänderten zeilen mit der funktion **affected_rows** (vgl. 3.2) abfragen.

UPDATE *tabelle* **SET** *spalte* = *wert* [, *spalte* = *wert* . . .] [**WHERE** *bedingung*]

beispiel
In allen zeilen in denen die spalte **name** den wert "Schulze" enthält, wird die spalte **vorname** in "Fritz" geändert. Dann werden in allen zeilen die spalten **abteil** und **beitrag** geändert.

```
<?php
   verbinden(1);
   $tabelle = "testdaten";
   $sql = "UPDATE $tabelle SET vorname = 'Fritz' "
          . "WHERE name = 'Schulze' ";
   $erg = mysqli_query($con, $sql);
   if ($erg)
   {   $num = mysqli_affected_rows($con);
       echo "<p>$num zeile geändert: vorname bei schulze</p>";
   }
   $sql = "UPDATE $tabelle SET abteil = 5, beitrag = 25.00";
   $erg = mysqli_query($con, $sql);
   if ($erg)
   {   $num = mysqli_affected_rows($con);
       echo "<p>$num zeilen geändert: abteilung, beitrag bei allen</p>";
   }
?>
```

1 zeile geändert: vorname bei Schulze

4 zeilen geändert: abteilung und beitrag bei allen

nummer	*name*	*vorname*	*abt*	*beitrag>*
1000	Schulze	Fritz	5	25.00
1001	Meyer	Otto	5	25.00
1002	Meyer	Lisa	5	25.00
1003	Muller	Frieda	5	25.00

8.3 DELETE - zeilen löschen

Mit der anweisung kann man zeilen einer DB-tabelle löschen. Wenn keine bedingung angegeben ist, werden alle zeilen der tabelle gelöscht. Wenn die anweisung mit **query** erfolgreich ausgeführt wurde, kann man die anzahl der gelöschten zeilen mit der funktion **affected_rows** (vgl. 3.2) abfragen.

DELETE FROM *tabelle* [**WHERE** *bedingung*]

beispiel
Es werden alle zeilen gelöscht, bei denen die spalte **nummer** einen wert < 2000 enthält.

```
<?php
   verbinden(1);                  //  liefert $con
   $tabelle = "mitglieder";
   $sql = "DELETE FROM $tabelle WHERE nummer < 2000 ";
   $erg = mysqli_query($con, $sql);
   if ($erg)
   {   $num = mysqli_affected_rows($con);
       echo "<p>$num zeilen gelöscht</p>";
   }
?>
```

8.4 besonderheiten

8.4.1 zeichensatz festlegen

Daten, die in einer DB-tabelle gespeichert werden, stammen aus verschiedenen quellen:
- aus einer datei, die von der seite eingelesen wird, mit der die zeilen einer DB-tabelle erzeugt oder geändert werden;
- aus einem formular, mit dem die daten erfaßt werden;
- direkt aus der seite, mit der die DB-tabelle bearbeitet wird.

Für die speicherung ist der zeichensatz der daten von großer bedeutung. In der regel ist das **Ansicode** oder **Unicode**. In der **HTML**-beschreibung wird ausgeführt, dass der zeichensatz , mit dem eine seite erstellt wird und der zeichensatz, der im header der seite vereinbart wird der gleiche sein sollte (HTML-beschreibung 2.5 und 9.9). Nur so ist eindeutig festgelegt, mit welchem zeichensatz eine seite arbeitet, d.h. daten ausgibt und darstellt. Vertieft wird diese information in der **PHP**-beschreibung (ziffer 8.9). Bei **MySQL** kommt jetzt hinzu, dass man festlegen muss, mit welchem zeichensatz die daten in der datenbank gespeichert werden.

vereinbarung im header einer seite

```
<meta http-equiv="Content-Type" content="text/html;      veraltet
      charset=windows-1252 | iso-8859-1 | utf-8">

<meta charset="iso-8858-1 | utf-8">                      HTML5
```

```
windows-1252   Ansicode
iso-8859-1

utf-8          Unicode
```

zeichensatz für datenbank festlegen

Das geschieht mit der anweisung **CREATE TABLE** (vgl. 7.1)

```
ENGINE=MyISAM DEFAULT  CHARSET=latin1 | utf8

latin1         Ansicode

utf8           Unicode
```

8.4.2 probleme mit zeichensätzen

Die beiden zeichensätze Ansicode und Unicode unterscheiden sich nur bei der codierung von geschlossenen umlauten und einigen sonderzeichen (?, €, §, °, ², ³, µ); während im Ansicode alle zeichen mit **einem** byte verschlüsselt sind, verwendet der Unicode bei den genannten zeichen **zwei** bytes. Wenn man konsequent (auch bei den eingabedaten) den gleichen zeichensatz verwendet, sollte es eigentlich keine probleme geben. Leider stimmt das nicht, zudem treten die probleme recht uneinheitlich auf, ob das versions- oder herstellerabhängig ist war nicht zu klären.

probleme mit Unicode
Liegen die daten im **Unicode** vor und wird dieser zeichensatz auch für die DB-tabelle vereinbart, gibt es keine probleme, die speziellen zeichen werden mit zwei bytes gespeichert, seltsamerweis auch dann, wenn man mit der anweisung **Create Table** Ansicode vereinbart. Bei der dimensionierung der spalten muss man natürlich berücksichtigen, dass bestimmte zeichen zwei bytes platz benötigen. Es gibt nur einen schönheitsfehler: das weitverbreitete programm für die verwaltung einer datenbank **phpmyadmin** stellt die zwei-bytes-zeichen nicht richtig dar, sondern zeigt kryptische zeichenfolgen.

probleme mit Ansicode
Liegen die daten im **Ansicode** vor, funktioniert es manchmal richtig, selbst dann, wenn für die datenbank Unicode vereinbart wird. Meist aber geht es schief, d.h. die o.g. zeichen werden entweder als fragezeichen oder auch gar nicht gespeichert.

8.4.3 zeichen maskieren

Alle probleme mit den geschlossenen umlauten und den sonderzeichen kann man umgehen, wenn man diese zeichen maskiert (siehe HTML-beschreibung 2.5).. Allerdings muss man dann die spalten der DB-tabelle entsprechend groß machen, denn die maskierten zeichen benötigen viel platz, beispielsweise besteht ein maskiertes **Ä** aus 6 zeichen: & A u m l ;

codes, maskierung

zeichen	*bedeutung*	*Ansicode*	*maskierung*	*Unicode*
Ä	umlaut	C4	Ä	C384
Ö	umlaut	D6	Ö	C396
Ü	umlaut	DC	Ü	C395
ä	umlaut	E4	ä	C3A4
ö	umlaut	F6	ö	C3B6
ü	umlaut	FC	ü	C3BC
ß	scharfes s	DF	ß	C39F
€	euro	80	€	E282
§	paragraph	A7	&#0167	C2A7
°	hochg. 0	B0	&#0176	C2B0
²	hochg. 2	B2	&#0178	C2B2
³	hochg. 3	B3	&#0179	C2B3
µ	my	B5	&#0181	C2B5

9. beispiel datenbank-anwendung

9.1 einleitung

Zum abschluß der dokumentation wird hier ein beispiel für eine datenbank-anwendung gezeigt; das beispiel arbeitet konsequent mit dem zeichensatz **UTF-8**. Die daten für die DB-tabelle werden mit einem formular eingegeben. Die auswertung von etwaigen fehlern, die bei der ausführung von MySQL-funktionen auftreten können, ist in dem beispiel vorhanden, wird hier aber nicht gezeigt.

Das beispiel ist in der datei **beispiel.php** enthalten. Die datei ist in eine textdatei, die mit dem zeichensatz **UTF-8** erstellt wurde, ein- und ausgaben der seite erfolgen mit diesem zeichensatz Die seite kann her-untergeladen werden.

9.2 start der anwendung

Der aufruf der anwendung erfolgt mit drei aufruf-parametern, dabei sind die variablen **$modus** und **$anz** durch die einbettung des beispiels in die dokumentation bereits gesetzt, sie haben nur interne bedeutung. Mit dem parameter **art** wird der wert "start" übergeben, d.h. die startroutine der anwendung wird ausgeführt. Aus verschiedenen routinen ruft sich die anwendung selbst wieder auf, dabei ändert sich im aufruf nur der wert für den aufruf-parameter **art**.

```
<?php
   echo "<p><a href=\"doku/MYSQL/beispiel.php?modus=$modus&art=start"
        . "&anz=$anz\">aufruf</a></p>";
?>
```

In die seite wird wie bei allen beispielen die datei **verbinden-inc.php** eingebunden. Die datei enthält die funktion **verbinden**, mit der die verbindung zum DB-server hergestellt, und eine datenbank zugewiesen wird. In der funktion werden die globalen variablen **$con** (handle der daten-bank) und **$dbname** (name der datenbank) versorgt. Bei jedem aufruf der seite werden die aufrufparameter übernommen und die variable **$tabelle** wird gesetzt, soweit nötig, wird die verbindung zur datenbank hergestellt.

```
<?php
   //   parameter übernehmen
   $art   = $_GET["art"];            //  auszuführende routine
   $modus = $_GET["modus"];
   $anz   = $_GET["anz"];
   $ende    = chr(13) . chr(10);     //  zeilenende
   $tabelle = "testdaten";           //  DB-tabelle
   include 'verbinden-inc.php';
```

start-routine

Der aufrufparameter **art** hat den wert **start**. Die verbindung zur datenbank wird hergestellt und, falls vorhanden, wird die DB-tabelle **testdaten** gelöscht und dann neu eingerichtet. Obwohl bei der eingabe für **name** und **vorname** nur 20 zeichen eingegeben werden können, erhalten die entsprechenden spalten in der DB-tabelle die größe 100, weil hier ggf. platz für geschlossene umlaute und sonderzeichen benötigt wird, die im Unicode mit zwei bytes verschlüsselt werden. Für die spalte **nummer** ist die option **AUTO_INCREMENT** gesetzt, dadurch wird beim erstellen der zeilen in diese spalte ein mit eins beginnender und dann fortlaufend um eins erhöhter wert eingetragen. Das einrichten der tabelle wird protokolliert, dann wird die verbindung zur datenbank getrennt. Mit **weiter** wird erneut die seite **beispiel.php** aufgerufen, jetzt mit **art=bau**, d.h. das eingabeformular soll aufgebaut werden.

```
if ($art == "start")
{    $erg = verbinden(1);            //  verbindung zur datenbank
     mysqli_query($con, "DROP TABLE IF EXISTS $tabelle");
      $sql = "CREATE TABLE IF NOT EXISTS $tabelle (
           nummer   int          AUTO_INCREMENT,
           name     varchar(100) DEFAULT NULL,
           vorname  varchar(100) DEFAULT NULL,
           abteil   int          DEFAULT NULL,
           beitrag  double       DEFAULT NULL,
           UNIQUE KEY (nummer))
           ENGINE=MyISAM DEFAULT CHARSET=utf8";
     $erg = mysqli_query($con, $sql);
     echo "<p>DB-tabelle <b><i>$tabelle</i></b> erzeugt</p>" . $ende;
     $erg = mysqli_query($con, "SHOW COLUMNS FROM $tabelle");
      $num = mysqli_num_rows($erg);    //  zeilenzahl
     echo "<p>es sind $num spalten vorhanden</p>" . $ende;
     //   hier wird jetzt die struktur der tabelle gezeigt
     //   vgl. dazu kapitel 4.4
     mysqli_free_result($erg);        //  ergebnis freigeben
     mysqli_close($con);              //  verbindung trennen
     echo "<p><a href=\"beispiel.php?modus=$modus&art=bau"
               . "&anz=$anz\">weiter</a>"
               . "   zur datenerfassung</p>" . $ende;
}
```

beispiel für eine datenbank-anwendung

DB-tabelle *testdaten* einrichten

DB-tabelle ***testdaten*** erzeugt

es sind 5 spalten vorhanden

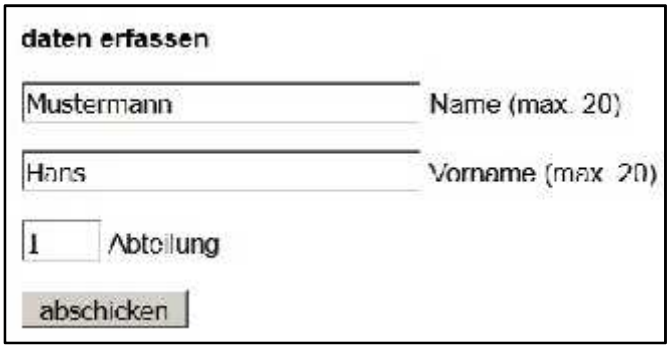
spalte	typ	*Null*	*key*	*default*	*sonstiges*
nummer	int(11)	NO	PRI		auto_increment
name	varchar(100)	YES			
vorname	varchar(100)	YES			
abteil	int(11)	YES			
beitrag	double	YES			

9.3 routine datenerfassung

9.3.1 daten eingeben

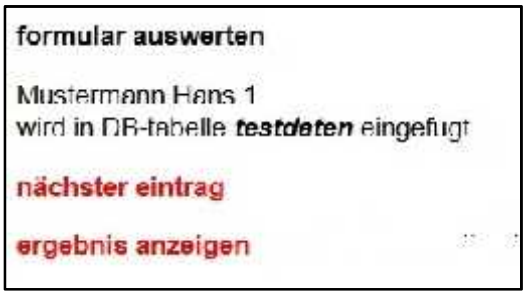
Das erfassungsformular wird aufgebaut und angezeigt. Das ausgefüllte formular wird mit **abschicken** an die seite **beispiel.php** geschickt. Jetzt aber mit **art = form**, d.h. das formular soll ausgewertet werden. Während diese routine abläuft besteht keine verbindung zur datenbank.

```
<?php
   else if ($art == "bau")
   {   echo "<p><b>daten erfassen</b></p>" . $ende;
echo <<<DOC
       <form action="beispiel.php?modus=$modus&art=form&anz=$anz"
             accept-charset="utf-8" method="POST">
       <p><input type="text" name="name" size="20" maxlength="40" /> Name</p>
       <p><input type="text" name="vorn" size="20" maxlength="40" /> Vorname</p>
       <p><input type="text" name="abt"  size="3" maxlength="3" /> Abteilung</p>
       <p><input type="submit" name="sender" value="abschicken" /></p>
       </form>
DOC;
   }
?>
```



9.3.2 erfasste daten verarbeiten

Aus dem übergebenen formular werden die daten übernommen. Wenn keine abteilung angegeben wurde, wird sie auf 1 gesetzt. Dann wird die verbindung zur datenbank hergestellt und mit der anweisung INSERT INTO eine zeile in die DB-tabelle gespeichert. Dann wird die verbindung getrennt.



nächster eintrag

die seite beispiel.php wird erneut mit **art = bau** aufgerufen

ergebnis anzeigen

die seite beispiel..php wird aufgerufen mit **art = zeig**, d.h. routine ergebnis anzeigen ausführen

```
<?php
   else if ($art == "form")
   {   $name    = $_POST["name"];     //  daten aus formular
       $vorname = $_POST["vorn"];     //  übernehmen
       if (strlen($name) <= 0)
           $name = "Anonymus";
       if (strlen($vorname) <= 0)
           $vorname = "namenlos";
       $abteil  = $_POST["abt"];
       $abteil  = intval($abteil);
       if ((!is_int($abteil)) || ($abteil <= 0))
           $abteil = "1";
       $erg = verbinden();            //  verbindung zur DB
       echo "<p>$name $vorname $abteil<br />"
            . "wird in DB-tabelle <b><i>$tabelle</i></b> "
            . "eingefügt</p>" . $ende;
       $sql = "INSERT INTO $tabelle (name, vorname, abteil)
               VALUES ('$name', '$vorname', $abteil)";
       $erg = mysqli_query($con, $sql);
       mysqli_close($con);            //  verbindung trennen
       echo "<p><a href=\"beispiel.php?modus=$modus&art=bau"
            . "&anz=$anz\">nächster eintrag</a></p>";
       echo "<p><a href=\"beispiel.php?modus=$modus&art=zeig"
            . "&anz=$anz\">ergebnis anzeigen</a></p>";
   }
```

9.4 routine ergebnis anzeigen

Die routine wird schrittweise erklärt: zunächst wird die verbindung zur datenbank hergestellt. Dann wird in allen zeilen die spalte **beitrag** geändert; die anzahl der geänderten zeilen wird ermittelt und angezeigt.

```
<?php
   else if ($art == "zeig")
   {   $erg = verbinden();          //  verbindung zur DB
       $sql = "UPDATE $tabelle SET beitrag = 25.50";
       $erg = mysqli_query($con, $sql);
       if ($erg)
       {   $n = mysqli_affected_rows($con);
           echo "<p>$n zeilen geändert</p>";
       }
```

Bei der anzeige der DB-tabelle mit allen vorhandenen spalten wird gezeigt, wie das möglich ist, ohne zu wissen, wieviel spalten vorhanden sind und welche namen die spalten haben. Es werden also zunächst die anzahl der spalten je zeile und die namen der spalten ermittelt. Dazu muss man wissen, dass SHOW COLUMNS für jede spalte als ergebniszeile ein feld liefert, bei dem im ersten element der name der spalte steht. Die namen werden in dem feld **$spalte** gespeichert, die anzahl in der variablen **$nsp**. Das ergebnis wird freigegeben.

```
       $erg = mysqli_query($con, "SHOW COLUMNS FROM $tabelle");
       $nsp = mysqli_num_rows($erg);     //  anzahl ergebnis-zeilen
                                         //  ist anzahl spalten
       echo "<p>es sind $nsp spalten vorhanden</p>" . $ende;
       $spalte[0] = " ";
       for ($z=1; $z<=$nsp; $z++)
       {   $zeile = mysqli_fetch_row($erg);   //  nächste zeile
           $spalte[$z] = $zeile[0];           //  spaltenname aus zeile
       }
       mysqli_free_result($erg);              //  ergebnis freigeben
```

Nun werden alle zeilen der DB-tabelle gelesen (das ergebnis steht in **$erg**) und die anzahl der ergebnis-zeilen ermittelt und angezeigt. Das ergebnis soll zeilenweise in einer tabelle angezeigt werden, in die titel-zeile der tabelle kommen die namen der spalten aus dem feld **$spalte**, die anzahl der spalten steht in **$nsp.**

```
        $sql = "SELECT * FROM  $tabelle";     //  alle zeilen aus der
        $erg = mysqli_query($con, $sql);      //  DB-tabelle
        $z = mysqli_num_rows($erg);           //  anzahl ergebnis-zeilen
        echo "<p>die tabelle <b><i>$tabelle</i></b> enthält $z "
             . "zeilen</p>" . $ende;
        echo "<table class='tbmit' style='width: 500px'>" . $ende;
        echo "<tr>" . $ende;
        for ($x=1; $x<=$nsp; $x++)            //  spaltennamen in titelzeile
            echo "<th>$spalte[$x]</th>";
        echo "</tr>" . $ende;
```

Jetzt wird endlich das ergebnis aus **$erg** zeilenweise nach **$zeile** gelesen; das ist keine einfache variable sondern ein assoziatives feld. Damit man das feld auswerten kann, braucht man als key die spaltennamen der DB-tabelle. Die namen stehen im feld **$spalte**, die anzahl der spalten in **$nsp**. Zuletzt wird das ergebnis freigegeben, die DB-tabelle wird gelöscht und die verbindung getrennt. Dann erfolgt die rückkehr zur doku-mentation

```
        while ($zeile = mysqli_fetch_assoc($erg))
        {   echo "<tr>" . $ende;
            for ($x=1; $x<=$nsp; $x++)        //  ergebnis-zeile spaltenweise
            {   $key  = $spalte[$x];          //  key ist spaltenname
                $wert = $zeile[$key];
                echo "<td>$wert</td>";
            }
            echo "</tr>" . $ende;
        }
        echo "</table>" . $ende;

        mysqli_free_result($erg);             //  ergebnis freigeben
                                              //  DB-tabelle löschen
        $erg = mysqli_query($con, "DROP TABLE IF EXISTS $tabelle");
        mysqli_close($con);                   //  verbindung trennen

        //   zurück zur dokumentation
        echo "<p><a href='../../seite.php?modus=$modus&dir=doku/MYSQL"
             . "&name=mysqldoku9&mkz=54&anz=$anz'>beenden</a>"
             . "   die anwendung beenden</td>" . $ende;
    }
```

hinweis
Die hier gezeigte beendigung der anwendung ist spezifisch für die dokumentation, sie ist ein rücksprung zum untermenü von kapitel 9. In der praxis wird man wohl zu der stelle zurückkehren, an der die anwendung aufgerufen wurde.

ergebnis anzeigen
zunächst in allen zeilen 25.00 in spalte ***beitrag*** eintragen

3 zeilen geändert

es sind 5 spalten vorhanden

die tabelle ***testdaten*** enthält 3 zeilen

nummer	*name*	*vorname*	*abteil*	*beitrag*
1	Mustermann	Hans	1	25.5
2	Musterfrau	Gretl	4	25.5
3	Anonymus	namenlos	1	25.5

die DB-tabelle ***testdaten*** wurde geloscht