

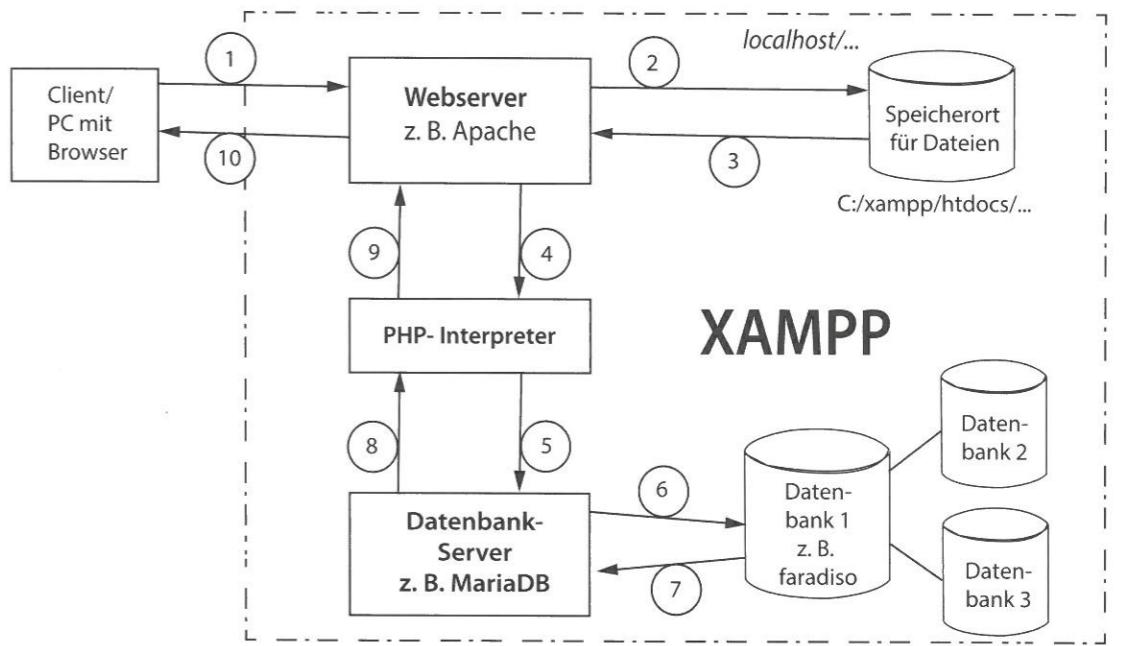
# 8 Datenbanken im Internet

## 8.1 Entwicklungsumgebung XAMPP

Web-Datenbanken sind Datenbanken, die für den Einsatz im Internet geeignet sind, z. B. MySQL-Datenbanken. Zur Entwicklung einer Web-Datenbank benötigt man eine lokale Entwicklungsumgebung. Diese besteht z. B. aus

- dem **Webserver Apache**, der im Internet angeforderte Daten an den Client sendet,
- dem Datenbanksystem **MySQL** (von **M**y **Sc**tured **Q**uery **L**anguage), neu MariaDB,
- und der Programmiersprache **PHP** (von **P**HP **H**ypertext **P**reprocessor).

Diese Elemente sind z. B. in der Entwicklungsumgebung **XAMPP** zusammengefasst (siehe Bild).



## 8.2 Funktionsweise der Komponenten

### 8.2.1 Der Webserver

Der Webserver, z. B. Apache, organisiert das Zusammenwirken der einzelnen Internet-Komponenten, z. B. des Browsers und des Servers, auf dem die Daten gespeichert sind.

Die vom Client (= Kunde) angeforderte URL (= Uniform Resource Locator, Bezeichnungsstandard für Netzwerkressourcen) ① wird vom Webserver als Adresse identifiziert und am Speicherort abgeholt ② und zurückgesendet ③. An der Dateiendung erkennt der Webserver, ob es sich um eine reine HTML-Seite handelt oder ob andere Dateitypen, z. B. PHP-Dateien, angefordert werden.

Dateien mit der Endung .PHP leitet der Webserver zunächst an das PHP-Programm zur Interpretation weiter ④. Dort werden alle PHP-Code-Anteile erkannt und interpretiert. Sind z. B. Anfragen an eine Datenbank enthalten, so werden diese vom PHP-Interpreter an das Datenbanksystem weitergegeben ⑤. Das Datenbanksystem führt die SQL-Anweisungen in der Datenbank aus (⑥ und ⑦) und liefert die Daten an den PHP-Interpreter ⑧. Diese Ergebnisse (z. B. einer SQL-Abfrage) werden in HTML-Code umgewandelt und an den Webserver zurückgeleitet ⑨. Von dort erhält der Client ein reines HTML-Dokument ⑩, das mit einem Browser, z. B. Mozilla Firefox, am Bildschirm dargestellt wird.

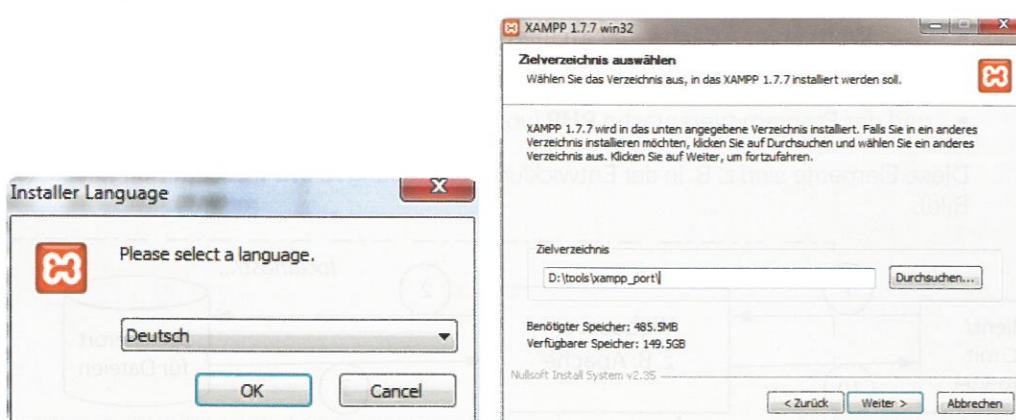
Webserver, Interpreter-Programme und Datenbankserver müssen nicht zwingend auf dem gleichen PC verfügbar sein, sondern können auf verschiedenen Servern im Internet weltweit verteilt sein.

### 8.2.2 Installation der Entwicklungsumgebung XAMPP

Nach Herunterladen der frei verfügbaren Entwicklungsumgebung (z. B. von [www.apache-friends.org/de/xampp-windows.html](http://www.apache-friends.org/de/xampp-windows.html)) startet man die Datei XAMPP xampp-windows-x64.x.x-VCxx-installer.exe.

Die Sprache wird über das Auswahlmenü eingestellt:

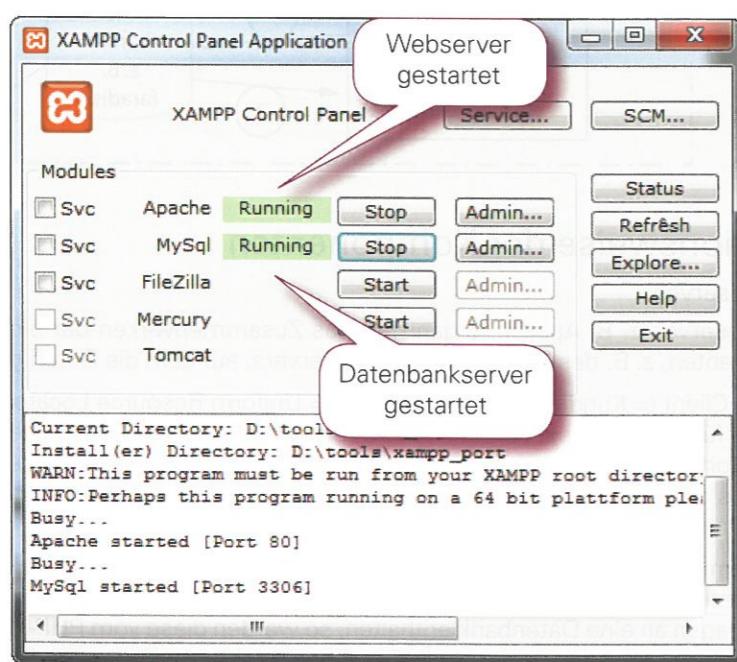
Das Ziellaufwerk der Installation wird im Fenster Zielverzeichnis auswählen festgelegt.



Um die Verzeichnispfade der Anwendung auf dieses Verzeichnis einzustellen, wird die Datei setup\_xampp.bat durch Doppelklick ausgeführt.

### 8.2.3 Starten der Komponenten

Die Datei xampp-control.exe startet die XAMPP-Entwicklungsumgebung und es öffnet sich das Control Panel von XAMPP. Ein Klick auf die Schaltflächen Start startet jeweils den Webserver Apache sowie den Datenbankserver MariaDB.



## 8.3 Die Skriptsprache PHP

### 8.3.1 Einführung

**PHP** ist eine Skriptsprache (von scriptum = Schriftstück, im Deutschen: Skript), die in HTML-Code eingebettet werden kann. Eine Skriptsprache wird nicht vorab kompiliert, sondern das Skript wird zur Laufzeit interpretiert und ausgeführt. Im Unterschied zu JavaScript überträgt der Webserver diesen Code jedoch nicht zum Client, sondern führt ihn auf der Serverseite aus. Der Client erhält das Ergebnis des Programmlaufes als HTML-Datei, die keinen PHP-Quellcode mehr enthält.

#### Hinweis:

PHP verhindert den Diebstahl von Code, indem der Quellcode (das Skript) nicht zum Client übertragen wird, sondern auf dem Server interpretiert wird.

Während zum Verarbeiten von Seiten mit Java-Code der HTML-Browser des Clients dafür geeignet und eingestellt sein muss, stellt PHP keine Anforderungen an den Browser.

In der Entwicklungsumgebung XAMPP ist der PHP-Interpreter enthalten.

Um PHP-Quellcode in der Testumgebung XAMPP auszuführen,

- müssen die Skripte im Unterverzeichnis C:\xampp\htdocs mit der Extension .php gespeichert sein,
- muss der Webserver gestartet sein und
- muss der Aufruf eines Skripts über den Webserver mit der IP 127.0.0.1/... oder der URL localhost/... erfolgen.

#### Hinweis:

Wird die Datei test.php direkt vom Browser an der Speicheradresse z. B. c:\xampp\htdocs\... gelesen, so wird der PHP-Interpreter umgangen und der PHP-Quelltext wird direkt im Browser angezeigt. Datenbankzugriffe werden dann nicht durchgeführt.

### 8.3.2 Schreiben eines PHP-Skripts

Die Sprachelemente von PHP sind weitgehend von den Programmiersprachen C, Java und Perl abgeleitet.

#### Beispiel:

Ein Skript soll am Bildschirm den Inhalt einer Textvariablen, z. B. „Hallo, Datenbank-Experten!“, ausgeben.

```
<html>
  <head>
    <title>Übung 1</title>
  </head>
  <body>

    <?php
      $daten = "Hallo, Datenbank-Experten!";
      //Deklaration der Variablen
      echo $daten; // Ausgabe der Variablen
    ?>

  </body>
</html>
```

#### Vorgehensweise:

Mit einem Texteditor oder einem HTML-Editor erstellt man zunächst das Grundgerüst einer HTML-Datei.

Das PHP-Skript wird zwischen die Tags (= Markierungen) <?php und ?> in den Body-Bereich des HTML-Skripts eingefügt.

Jede Anweisungszeile muss mit einem Semikolon beendet werden.

Kommentare werden mit // begonnen.

#### Hinweis:

PHP-Skripte sollten immer schrittweise erstellt und getestet werden, um Fehler rechtzeitig einzufangen, zu erkennen und zu beheben.

#### 8.3.3 Variablen in PHP

Variablen werden in PHP durch ein vorangestelltes \$-Zeichen benannt, z. B. \$daten. Sie müssen nicht ausdrücklich deklariert werden, sondern werden durch die Wertzuweisung in das geeignete Datenformat gebracht. Die Ausgabe der Variablen \$daten am Bildschirm erfolgt durch die Anweisung print \$daten oder echo \$daten.

Die Datei mit dem PHP-Skript muss mit der Erweiterung .php, z. B. script1.php, im Veröffentlichungsverzeichnis des Webservers gespeichert werden. Der Aufruf erfolgt im Browser nicht offline über das Öffnen der Datei im Pfad C:\...\xampp\htdocs..., sondern im Online-Modus, z. B. über die Adresse http://localhost/script1.php.



Damit PHP-Skripte in HTML-Seiten interpretiert werden, müssen die Dateien z. B. mit der Dateierweiterung .php gespeichert werden und über einen Webserver ausgeführt werden.

#### 8.3.4 Arrays

Ein Array (= Feld) besteht aus einer bestimmten Anzahl von Feldelementen, die jedoch nicht vom gleichen Datentyp sein müssen. So können z. B. Zahlenfelder, Textfelder und Bildobjekte zu einem Array zusammengefasst werden.

Arrays können aus Feldelementen mit unterschiedlichen Datentypen bestehen.

Deshalb sind PHP-Arrays für die Bearbeitung von Datensätzen in Datenbanken besonders geeignet.

Ein Array besitzt, wie andere Variablen auch, einen Namen, z. B. \$kunden. Die einzelnen Feldelemente werden bei numerischen Arrays durch einen fortlaufenden Index unterschieden, z. B. \$kunden[1] = "Huber". Der Index beginnt bei PHP standardmäßig bei [0].

#### Beispiel:

Mit einem Skript wird ein numerisches Array angelegt mit den folgenden Daten des Kunden Huber:

```
<?php
$kunden[0]= 1;
$kunden[1]= "Huber";
$kunden[2]= "Wilhelm";
$kunden[3]= "Hauptstr. 345";
$kunden[4]= "D";
$kunden[5]= "76567";
$kunden[6]= "Oberberg";
$kunden[7]= "07654-3210";
$kunden[8]= "07654-3211";
$kunden[9]= "Wilhelm.Huber@mail.com";
?>
```

Die Ausgabe des Arrays am Bildschirm ergibt:



#### Assoziative Arrays

Assoziative (mit Vorstellungen verbindende) Arrays verwenden als Index ihrer Elemente nicht Zahlen, sondern Zeichenfolgen oder Worte, Keys genannt. Anstatt \$kunden[0] und \$kunden[1] erhalten die Arrayelemente die Bezeichnung \$kunden[Nr] und \$kunden[Name].

Durch Indizierung mit diesem Key, z. B. [Nr] oder [Name], wird die Bedeutung des einzelnen Elementes einfacher vermittelt.

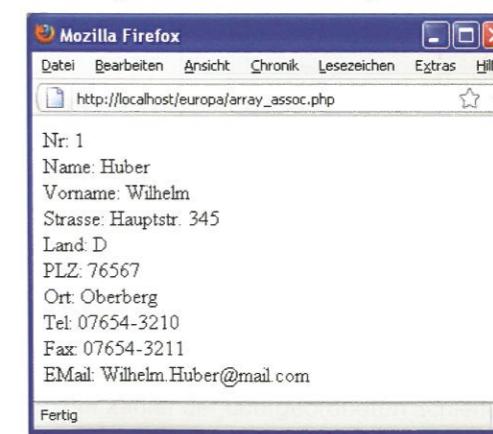
#### Beispiel:

Mit einem PHP-Skript werden die Daten des Kunden Huber in einem assoziativen Array gespeichert und ausgegeben.

```
<?php
$kunden[Nr]= 1;
$kunden[Name]= "Huber";
$kunden[Vorname]= "Wilhelm";
$kunden[Strasse]= "Hauptstr. 345";
$kunden[Land]= "D";
$kunden[PLZ]= "76567";
$kunden[Ort]= "Oberberg";
$kunden[Tel]= "07654-3210";
$kunden[Fax]= "07654-3211";
$kunden[EMail]= "Wilhelm.Huber@mail.com";

foreach($kunden as $key => $element)
    echo $key, ': ', $element, '<br>';
?>
```

Die Ausgabe am Bildschirm ergibt:



Im Beispiel erfolgt die Ausgabe der Feldinhalte durch eine `foreach`-Schleife. Die allgemeine Syntax dieser Schleife (bei assoziativem Array) lautet:

```
foreach($arrayname as $keyname => $elementname)
    Anweisung;
```

Bei numerischen Arrays entfällt die Angabe des Schlüssels. Die Syntax der Schleife lautet dann allgemein:

```
foreach($arrayname as $elementname)
{
    Anweisung 1;
    Anweisung 2;
    Anweisung 3
}
```

Anstelle einer einzelnen Anweisung können mehrere mit geschweiften Klammern umschlossene Anweisungen zu einem Anweisungsblock zusammengefasst werden.

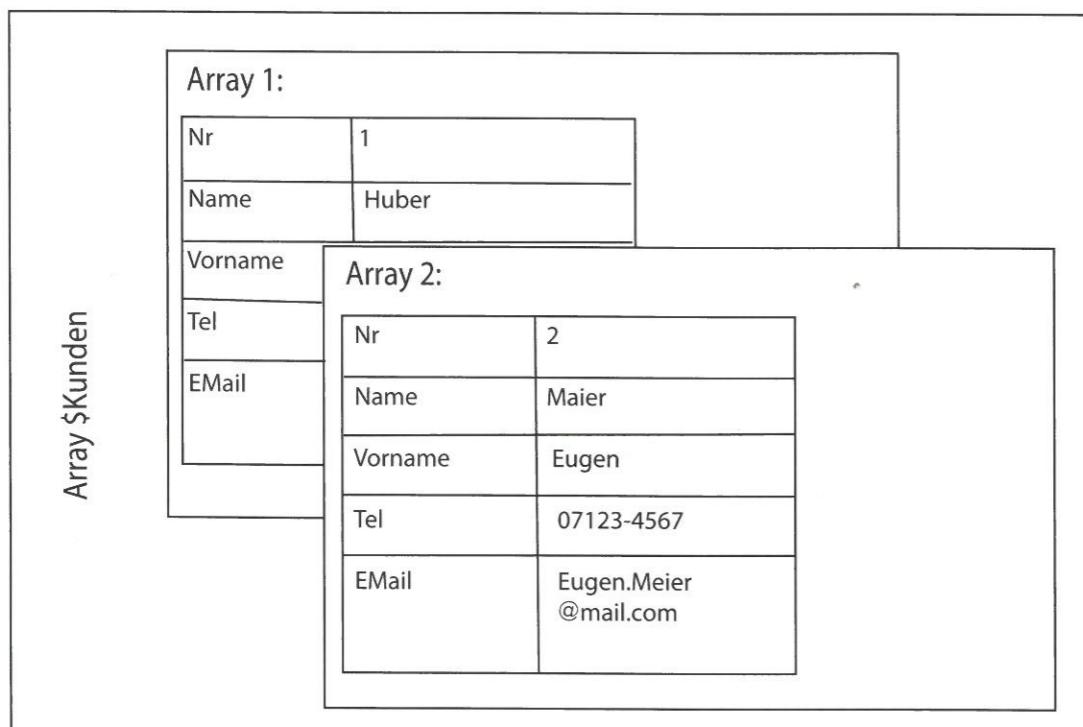
Zur Vereinbarung assoziativer Arrays stellt man den Arraynamen voran und gibt das Schlüsselwort `array` an. Die Keys der einzelnen Elemente werden in Anführungszeichen genannt, z. B. "Name", und nach dem Zuweisungsoperator `=>` mit Werten gefüllt:

```
$kunden=array(
    "Nr"=>1,
    "Name"=>"Huber",
    "Vorname"=>"Wilhelm",
    ...
    "EMail"=>"Wilhelm.Huber@mail.com");
```

### Mehrdimensionale Arrays

Einfache numerische oder assoziative Arrays können beliebig erweitert werden. Um jedoch bei mehreren Kunden die Übersicht zu bewahren, empfiehlt es sich, für jeden Kunden ein neues Array anzulegen. Die Arrays der einzelnen Kunden (entsprechend den Datensätzen einer Datenbank) werden zu einem übergeordneten Array zusammengefasst.

Das Prinzip eines mehrdimensionalen Arrays sieht folgendermaßen aus:



Mehrdimensionale Arrays sind ihres Aufbaus wegen geeignet, den Inhalt einer gesamten Tabelle einer relationalen Datenbank aufzunehmen.

### Beispiel:

Durch ein PHP-Script werden zwei Datensätze mit zwei Adressen gespeichert und am Bildschirm ausgegeben.

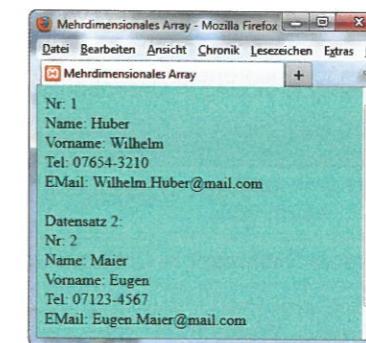
```
<html>
    <head>
        <title>Mehrdimensionales Array</title>
    </head>
    <body text=#000000    bgcolor=#00FFFF
        <?php
            $kunde[1] = array(
                "Nr"=> 1,
                "Name"=> "Huber",
                "Vorname"=> "Wilhelm",
                "Tel"=> "07654-3210",
                "EMail"=> "Wilhelm.Huber@mail.com"
            );
            $kunde[2] = array(
                "Nr"=> 2,
                "Name"=> "Maier",
                "Vorname"=> "Eugen",
                "Tel"=> "07123-4567",
                "EMail"=> "Eugen.Maier@mail.com"
            );
            foreach($kunde as $zaehler => $datensatz) {
                echo 'Datensatz ', $zaehler, ':', '<br>';
                foreach($datensatz as $index=>$feldinhalt)
                {
                    echo $index, ':      ', $feldinhalt, '<br>';
                }
                echo '<br>';
```

?>

```
</body>
</html>
```

### Erläuterung:

Nach der Festlegung der beiden Arrays `$kunde[1]` und `$kunde[2]` werden die Daten über eine doppelt verschachtelte `foreach`-Schleife ausgegeben. Dabei wird in der äußeren Schleife zunächst der interne Zähler `$zaehler` auf das erste Element gesetzt. Dieses erste untergeordnete Array `$kunde[1]` wird nun der Variablen `$datensatz` übergeben. Nach der Ausgabe von „Datensatz 1:“ wird die innere Schleife begonnen. Hier wird der Zähler `$index` gestartet und das erste Element des aktiven Arrays `$datensatz` der Variablen `$feldinhalt` übergeben. Mit `echo ...` werden der Index und der Feldinhalt am Bildschirm angezeigt:



Wenn der Zähler `$index` die gesamten Arrayelemente eines Datensatzes durchlaufen hat, wird der Zähler der übergeordneten Schleife `$zaehler` um 1 erhöht und der Vorgang wiederholt sich mit dem nächsten Sub-Array `$kunde[2]`.

### 8.3.5 Arbeiten mit Arrays

PHP bietet für die Verarbeitung von Arrays spezielle Funktionen:

Funktion	Beispiel	Wirkung
array_walk	\$ergebnis = array_walk(\$zahlen, 'quadrat');	Verändert die Elemente des Arrays, indem eine benutzerdefinierte Funktion (z. B. 'quadrat') auf die einzelnen Elemente angewendet wird.
key	\$schluessel = key(\$array);	Gibt den Schlüssel der aktuellen Position des Arrayzeigers zurück.
array_pop	array_pop(\$kunden);	Entfernt das letzte Element eines Arrays.
array_push	array_push(\$kunden, \$elem);	Fügt \$elem als weiteres Element an ein Array an.

#### Beispiel:

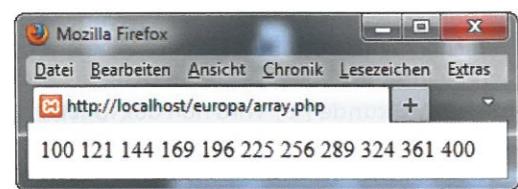
In einem Array werden die Zahlen 10 bis 20 gespeichert. Anschließend werden die Elemente dieses Arrays mithilfe einer selbstdefinierten Funktion quadriert und die Ergebnisse am Bildschirm ausgegeben.

```
<?
    function quadrat (&$x) {
        $x=$x*$x;
    }
$zahlen=array(10,11,12,13,14,15,16,17,18,19,20);
$ergebnis = array_walk($zahlen, 'quadrat');
foreach ($zahlen as $element) {
    echo $element, ' ';
}
?>
```

Zunächst wird die Funktion zum Quadrieren der Elemente definiert.

Nach dem Festlegen des ursprünglichen Arrays \$zahlen quadriert die Anweisung `array_walk($zahlen, 'quadrat')` die Elemente des Arrays und schreibt sie jeweils an ihren ursprünglichen Speicherplatz. Die Ausgabe erfolgt über die Anweisungen in der `foreach`-Schleife.

Das Ergebnis sieht folgendermaßen aus:



### 8.3.6 Bearbeiten von Zeichenketten

PHP stellt verschiedene Funktionen zur Bearbeitung von Zeichenketten zur Verfügung (siehe Tabelle).

Z. B. kann die Groß- und Kleinschreibung in Datenbanken gesteuert werden:

Funktion	Beispiel	Beschreibung
strtolower()	\$name = strtolower(\$name);	Wandelt in Kleinbuchstaben um.
strtoupper()	\$name = strtoupper(\$name);	Wandelt in Großbuchstaben um.
ucfirst()	\$name = ucfirst(\$name);	Wandelt das erste Zeichen in Großbuchstaben um.
ucwords()	\$str1 = strtolower(\$str);	Wandelt das erste Zeichen eines jeden Wortes in Großbuchstaben um.
ord()	\$wert = ord("A")	Gibt den ASCII-Wert eines Zeichens zurück.

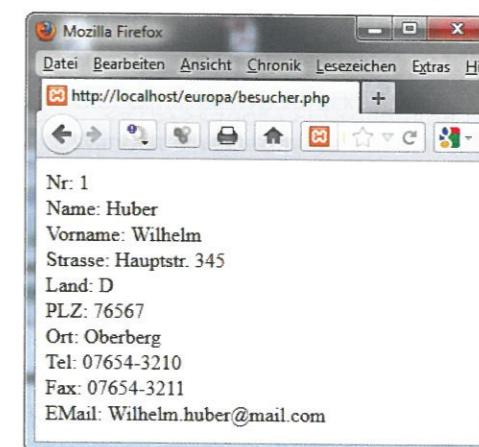
#### Beispiel:

Unabhängig von der Groß- und Kleinschreibung sollen die Daten des Kunden Huber in einem Array gespeichert werden. Bei der Ausgabe soll der erste Buchstabe von Textelementen groß geschrieben werden, alle anderen Buchstaben klein.

```
<?
    $kunden = array(
        "Nr"=> 1,
        "Name"=> "HubEr",
        "Vorname"=> "WilHelM",
        "Strasse"=> "hauptstr. 345",
        "Land"=> "d",
        "PLZ"=> "76567",
        "Ort"=> "OberBerg",
        "Tel"=> "07654-3210",
        "Fax"=> "07654-3211",
        "EMail"=> "Wilhelm.Huber@mail.com"
    );
    foreach($kunden as $zaehler => $feldinhalt) {
        $feldinhalt = strtolower($feldinhalt);
        $feldinhalt = ucfirst($feldinhalt);
        echo $zaehler, ': ', $feldinhalt, '<br>';
    }
?>
```

Innerhalb der `foreach`-Schleife wandelt die Funktion `strtolower($feldinhalt)` den Inhalt der Variablen in Kleinbuchstaben um und speichert ihn wieder unter gleichem Namen. Zahlenvariablen werden nicht bearbeitet.

Mit der Funktion `ucfirst()` wird der erste Buchstabe in einen Großbuchstaben umgewandelt und anschließend der Feldinhalt am Bildschirm ausgegeben:



### 8.3.7 Dateioperationen mit PHP

Für Dateioperationen stehen verschiedene Funktionen zur Verfügung:

Funktion	Beispiel	Bedeutung
fopen()	\$fp = fopen('filename', \$mode);	Öffnet eine Datei in einer bestimmten Zugriffsart
fclose()	fclose(\$fp);	Schließt eine geöffnete Datei
fgets()	fgets(\$fp, 12);	Liest eine Zeile mit maximaler Länge 12 aus einer Datei.
fgetc()	fgetc(\$fp);	Liest ein einzelnes Zeichen einer Datei.
fwrite()	fwrite(\$fp, \$zaehler);	Schreibt eine Zeichenkette \$str in eine Datei.
file_exists()	if(!file_exists(\$zaehldatei) ...;	Prüft, ob eine Datei existiert.

Funktion	Beispiel	Bedeutung
rewind()	rewind(\$fp);	Setzt den Dateizeiger auf den Anfang einer Datei.
unlink()	unlink('filename');	Löscht eine Datei.
require()	require(\$zaehldatei);	Fügt eine Datei in das PHP-Skript ein.

**Beispiel:**

Um die Besucher einer Homepage zu zählen, wird ein Zähler programmiert, der den Zählerstand in einer Textdatei `counter.txt` speichert und am Bildschirm ausgibt.

```
<html>
<head>
    <title>PHP-Zähler</title>
</head>
<body bgcolor="#87ceff">
    <?php
        $zaehldatei="counter.txt";
        if(!file_exists($zaehldatei)) {
            $fp=fopen($zaehldatei,"w+");
            fwrite($fp,"0");
            fclose($fp);
        }
        $fp=fopen($zaehldatei,"r+");
        $zaehler=(int)fgets($fp,12);
        $zaehler++;
        rewind($fp);
        fwrite($fp,$zaehler);
        fclose($fp);
    ?>
    <div align="Center">
        <font size="+2">Sie sind der
        <? require($zaehldatei); ?>
        . Besucher!</font></div>
    </body>
</html>
```

In der Datei `counter.txt` wird nur der Zählerstand als Ganzzahl abgelegt. Der Name dieser Datei wird in einer Variablen festgelegt.

Wenn diese Datei (noch) nicht existiert, wird die `if`-Anweisung durchlaufen. Es wird diese Datei erzeugt und im Modus `w+` geöffnet. Diese Betriebsart der Datei legt fest, dass sie zum Lesen und Schreiben geöffnet wird und – falls sie nicht existiert – zunächst erstellt wird. Mit der Funktion `fwrite()` wird die Zahl 0 in die Datei geschrieben. `fclose()` schließt die Datei wieder.

Die Funktion `fopen()` öffnet die nun sicher existierende Datei im Modus `r+` zum Lesen und Schreiben. Mit `fgets()` wird die maximal 12 Zeichen lange Zahl als String ausgelesen und mit `int()` in eine Ganzzahl umgewandelt. Sie wird in der Variablen `$zaehler` zwischengespeichert und mit `$zaehler++` um Eins erhöht. Die Funktion `rewind()` setzt den internen Zeiger auf den Anfang der Datei. Dorthin wird nun die um Eins erhöhte Zahl mit `fwrite()` geschrieben und die Datei wieder geschlossen.

Die Ausgabe erfolgt im HTML-Format, wobei der Zählerstand durch Einbinden der Datei mittels der Funktion `require($zaehlerstand)` angezeigt wird.



Dateien können mit der Funktion `fopen()` in verschiedenen Modi geöffnet werden. Die Betriebsart `a` öffnet die Datei und setzt den internen Dateizeiger auf das Ende der Datei. Dort sind jedoch nur Schreibvorgänge zugelassen. In der Betriebsart `a+` ist diese Einschränkung nicht vorhanden, es kann auch gelesen werden. In beiden Betriebsarten wird die gewählte Datei erstellt, falls sie noch nicht existiert.

Hier eine Übersicht über die möglichen Betriebsarten für Dateioperationen:

Betriebsart	Erläuterung
"r"	Öffnet eine Datei nur zum Lesen.
"r+"	Öffnet eine Datei zum Lesen und zum Schreiben.
"w"	Öffnet eine Datei nur zum Schreiben.
"w+"	Öffnet eine Datei zum Lesen und Schreiben. Der Inhalt wird zunächst gelöscht. Falls die Datei nicht existiert, wird sie erstellt.
"a"	Öffnet die Datei nur zum Schreiben und setzt den Dateizeiger auf das Ende der Datei. Falls die Datei nicht existiert, wird sie erstellt.
"a+"	Öffnet die Datei zum Lesen und Schreiben und setzt den Dateizeiger auf das Ende der Datei.

## 8.3.8 Zugriffsrechte auf Dateien

Eine wichtige Information bei der Arbeit mit Dateien, speziell mit Datenbanken, sind die Zugriffsrechte. Sie informieren über die Berechtigungen verschiedener Benutzer bezüglich dieser Dateien.

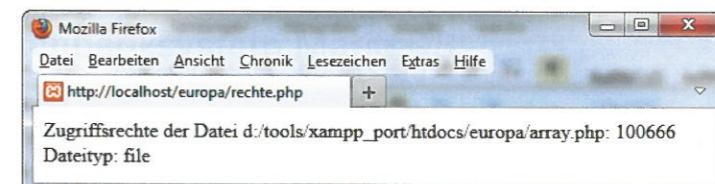
PHP bietet für die Verwaltung von Dateien und Zugriffsrechten geeignete Funktionen an:

Funktion	Beschreibung
filegroup();	Gibt die Gruppe des Dateibesitzers zurück.
fileowner();	Ermittelt den Besitzer der Datei.
fileperms();	Ermittelt die Zugriffsrechte der Datei.
filesize();	Ermittelt die Größe der Datei.
filetype();	Gibt den Dateityp zurück.
decodt();	Wandelt eine Dezimalzahl in eine Oktalzahl um.

**Beispiel:**

Mit einem PHP-Skript sollen die Zugriffsberechtigungen und der Dateityp, z. B. für die Datei `D:/tools/xampp_port/htdocs/europa/array.php`, ermittelt werden.

```
<?php
    $datei = "d:/tools/xampp_port/htdocs/europa/array.php";
    $erg=fileperms($datei);
    echo "Zugriffsrechte der Datei $datei: ", decodt($erg),
    "<br>";
    echo "Dateityp: ", filetype($datei);
?>
```



Nach der Festlegung des Pfades der Datei in einer Variablen ermittelt die Funktion `fileperms()` die Zugriffsrechte auf diese Datei. Da das Ergebnis in dezimaler Schreibweise vorliegt, aber erst in oktaler Schreibweise richtig interpretiert werden kann, muss es mit Hilfe der Funktion `decodt()` umgewandelt werden. Die ersten drei Ziffern, hier 100, informieren über den Dateityp. Die Funktion `filetype()` gibt diesen Dateityp in einer Textvariablen als `file` aus.

Mögliche Filetypen sind:

- block: block special device
- char: character special device
- dir: directory
- fifo: FIFO (named pipe)
- file: regular file
- link: symbolic link
- unknown: unknown file type

Die Ziffernfolge 666 informiert über die Rechte des Besitzers, der Gruppe und der Anderen. Dabei bedeutet die erste Ziffer 6, dass der Besitzer der Datei lesen (4) und schreiben (2), aber nicht ausführen darf. Die zweite Ziffer 6 informiert über die – hier gleichen – Rechte der Gruppe und die letzte Ziffer über die aller anderen Benutzer.

Die möglichen Rechte der einzelnen Benutzer sind:

Besitzer		Gruppe		Andere	
r	w	x	r	w	x
4	2	1	4	2	1
4	2	1	4	2	1
r von read = lesen, w von write = schreiben, x von execute = ausführen					

#### Beispiel:

Für eine Datei wird als Schlüssel für die Zugriffsrechte die Zahl 754 geliefert. Die einzelnen Berechtigungen werden folgendermaßen ermittelt:

- 7 = Besitzer: lesen (4) + schreiben (2) + ausführen (1);
- 5 = Gruppe: lesen (4) + ausführen (1);
- 4 = Andere: lesen (4).

#### 8.3.9 Arbeiten mit Formularen

Mit PHP-Skripten können Daten aus HTML-Formularen verarbeitet werden.

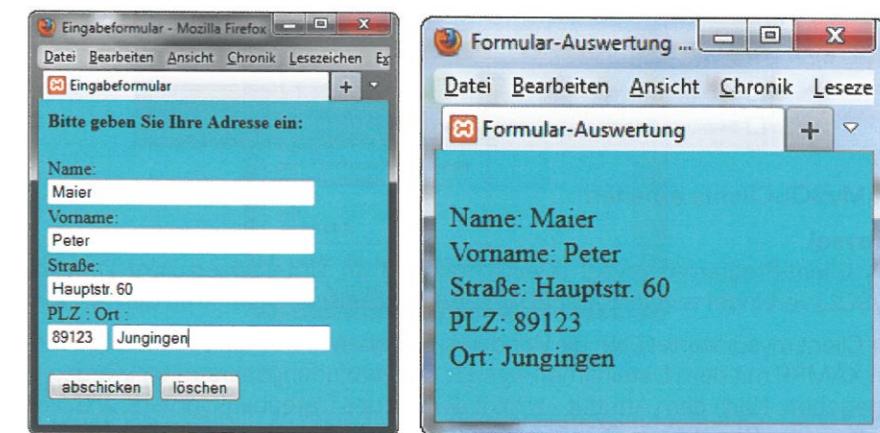
#### Beispiel:

In einem HTML-Formular wird die Adresse (Name, Vorname, Straße, PLZ, Ort) eingegeben. Über eine Befehlsschaltfläche sollen die Daten an ein PHP-Skript gesendet und dort angezeigt werden.

Das Listing des Eingabeformulars lautet z. B.:

```
<html>
  <head><title>Eingabeformular</title></head>
  <body text="#000000" bgcolor="#00E0FF">
    <h4>Bitte geben Sie Ihre Adresse ein:</h4>
    <form action="form_aus.php" method="POST">
      Name:<br><input type="Text" name="Name"
      size="35"><br>
      Vorname:<br><input type="Text" name="Vorname"
      size="35"><br>
      Straße:<br><input type="Text" name="Strasse"
      size="35">
      <br>
      PLZ : Ort :<br><input type="Text" name="PLZ"
      size="5" >
      <input type="Text" name="Ort"
      size="28"><br><br>
      <input type="Submit" name="submit" value="ab-
```

```
schicken">
<input type="reset" name="loeschen" value="löschen">
</form>
</body>
</html>
```



Zunächst erstellt man ein Formular im HTML-Code. Dort wird festgelegt, dass z. B. durch die Methode POST die Formulardaten an ein PHP-Skript, z. B. form\_aus.php, gesendet werden. Dies geschieht durch die HTML-Anweisung

```
<form action="form_aus.php" method="post">.
```

Durch Betätigen der Schaltfläche abschicken werden die Daten an die Datei form\_aus.php gesendet.

Das Listing der Datei form\_aus.php lautet z. B.:

```
<html><head>
  <title>Formular-Auswertung</title></head>
<body text="#000000" bgcolor="#00E0FF">
<?php
  $name      =$_POST["Name"];
  $vorname   =$_POST["Vorname"];
  $strasse   =$_POST["Strasse"];
  $plz       =$_POST["PLZ"];
  $ort       =$_POST["Ort"];
  echo "<br>Name:      ", $name;
  echo "<br>Vorname:   ", $vorname;
  echo "<br>Straße:   ", $strasse;
  echo "<br>PLZ:      ", $plz;
  echo "<br>Ort:      ", $ort;
?
</body></html>
```

Die Standardvariable \$\_POST[] überträgt die Daten im Datenformat eines Arrays. Die Elemente werden z. B. mit \$vorname=\$\_POST["Vorname"] ausgelesen und einer lokalen Variablen zugewiesen. Mithilfe des Befehls echo... können sie angezeigt werden.

## 8.4 Das Datenbanksystem Maria DB

#### Hinweis:

Die Datenbanksysteme MariaDB und MySQL sind nahe zu gleichwertig zu verwenden.

MariaDB ist ein relationales Datenbanksystem. Es wird eine Client-Server-Architektur benutzt, wobei der Datenbankserver mysqld.exe als Hintergrundprozess auf einem Server läuft. Es können beliebig viele Clients auf den Server zugreifen. Das Multithreading-Konzept ermöglicht, verschiedene Anfragen an den Server abzuarbeiten.

Eine MariaDB-Distribution ist in dem freien Datenbank-Entwicklungstool XAMPP enthalten. Starten lässt sich der Server-Prozess z. B. über das XAMPP Control Panel durch Betätigen der Schaltfläche Start neben dem Eintrag MariaDB.



#### 8.4.1 Mit MySQL-Clients arbeiten

##### Der Client mysql

Das Client-Programm mysql (Kleinschreibung im Unterschied zum Datenbanksystem MySQL) verbindet mit dem MariaDB-Server und erlaubt Befehle an den Server zu senden.

Der Client mysql startet über die Windows-Eingabeaufforderung im Installationsverzeichnis von XAMPP mit dem Dateinamen mysql. Die Verbindungsparameter werden beim Aufruf angegeben. Nach dem Attribut -h folgt der Ort des Datenbankservers, z. B. localhost, nach -u der Benutzer, z. B. root. Ein Passwort folgt hinter dem Schalter -p. Standardmäßig ist für den Benutzer root kein Passwort festgelegt. Um die Sicherheit zu erhöhen, ist es sinnvoll ein Passwort einzurichten.

```
Windows [Version 10.0.18363.778]
(c) 2019 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\User>cd\

C:\>cdxampp/mysql/bin

C:\xampp\mysql\bin>mysql -hlocalhost -uroot -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 67
Server version: 10.4.11-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

Nach einem Begrüßungstext können Befehle an den Client eingegeben werden. Monitorbefehle enden immer mit dem Zeichen ';'. Der Aufruf der Hilfe erfolgt durch den Befehl \h.

```
Windows [Version 10.0.18363.778]
(c) 2019 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\User>cd\

C:\>cdxampp/mysql/bin

C:\xampp\mysql\bin>mysql -hlocalhost -uroot -p
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>\h

General information about MariaDB can be found at
http://mariadb.org

List of all client commands:
Note that all text commands must be first on line and end with ;
?          (\?) Synonym for 'help'.
clear      (\c) Clear the current input statement.
connect    (\r) Reconnect to the server. Optional arguments are db and host.
delimiter  (\d) Set statement delimiter.
ego        (\G) Send command to MariaDB server, display result vertically.
exit       (\q) Exit mysql. Same as quit.
go         (\g) Send command to MariaDB server.
help       (\h) Display this help.
noteee    (\t) Don't write into outfile.
print     (\p) Print current command.
prompt    (\R) Change your mysql prompt.
quit      (\q) Quit mysql.
rehash    (\#) Rebuild completion hash.
source   (...) Execute an SQL script file. Takes a file name as an argument.
status   (\s) Get status information from the server.
tee      (\t) Set outfile [to_outfile]. Append everything into given outfile.
use      (\u) Use another database. Takes database name as argument.
charset  (\c) Switch to another charset. Might be needed for processing binlog with multi-byte charsets.
warnings (\W) Show warnings after every statement.
nowarning (\w) Don't show warnings after every statement.

For server side help, type 'help contents'

MariaDB [(none)]>
```

Am Prompt mysql werden Anweisungen an den Datenbankserver eingegeben.

Der Befehl show databases; gibt z. B. alle auf dem Server liegenden Datenbanken aus. Im Bild sind dies 11 Datenbanken.

##### Der Client mysqladmin

Der Client mysqladmin ermöglicht z. B. Administrationsaufgaben am Datenbankserver zu erledigen.

##### Beispiel:

Für den Benutzer root wird das Passwort Hus89adRBSiu eingerichtet.

In der Eingabeaufforderung des mysql-Verzeichnisses ruft man das Programm mysqladmin auf, mit den Attributen für den Server localhost und dem Benutzer root. Der Schlüsselbegriff password legt das nachgestellte Passwort für den Datenbankserver fest.

##### Beispiel:

Das Passwort wird in das neue Passwort geheim geändert.

Die zweite Anmeldung am Datenbankserver erfordert das vorher festgelegte Passwort hinter dem Attribut -p, um anschließend dieses Passwort ändern zu können.

##### Der Client mysqldump

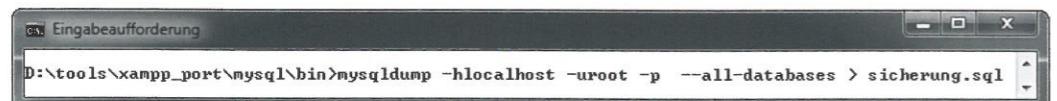
Der Client mysqldump wird zur Datensicherung verwendet. Das Programm schreibt den Inhalt von Datenbanken in eine Textdatei.

##### Beispiel:

Der Inhalt der Datenbank faradiso wird in der Datei backup\_file.sql gesichert.

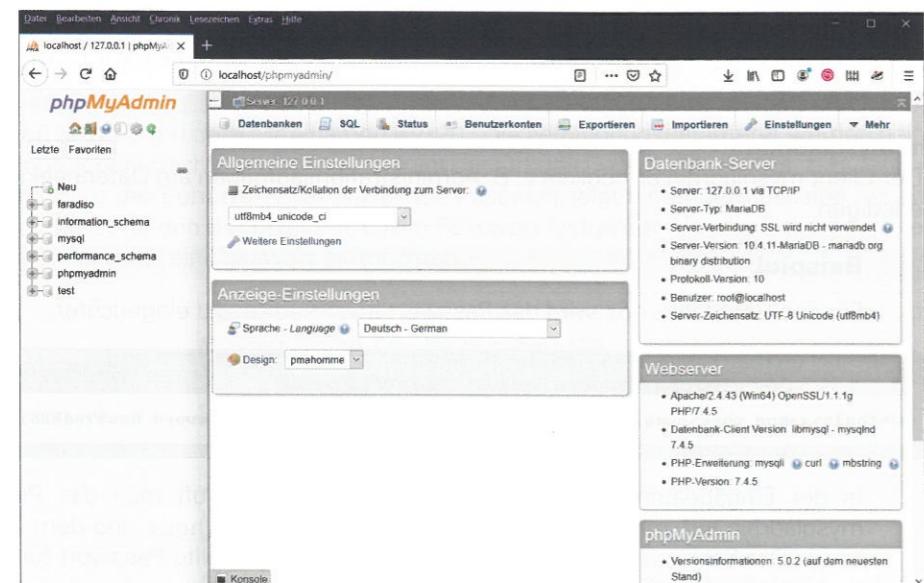
Nach den Anmeldeattributen gibt man die Datenbank an und legt nach dem Zeichen > den Namen der Exportdatei, z. B. backup\_file.sql, fest. In diese Textdatei werden SQL-Befehle geschrieben, die beim Ausführen wieder die Tabellen erzeugen und die Daten einfügen.

Mit der Option `--all-databases` werden alle Datenbanken des Datenbankservers z. B. in der Datei `sicherung.sql` gesichert:



#### Der Client phpMyAdmin

Die Entwicklungsumgebung XAMPP bietet standardmäßig den Client phpMyAdmin an. Er startet durch Aufruf der Adresse `localhost/phpmyadmin` im Browser. Mit diesem Werkzeug können nahezu alle Administrationsaufgaben am MySQL-Datenbankserver komfortabel durchgeführt werden. Deshalb wird im Weiteren vor allem mit diesem Client gearbeitet.

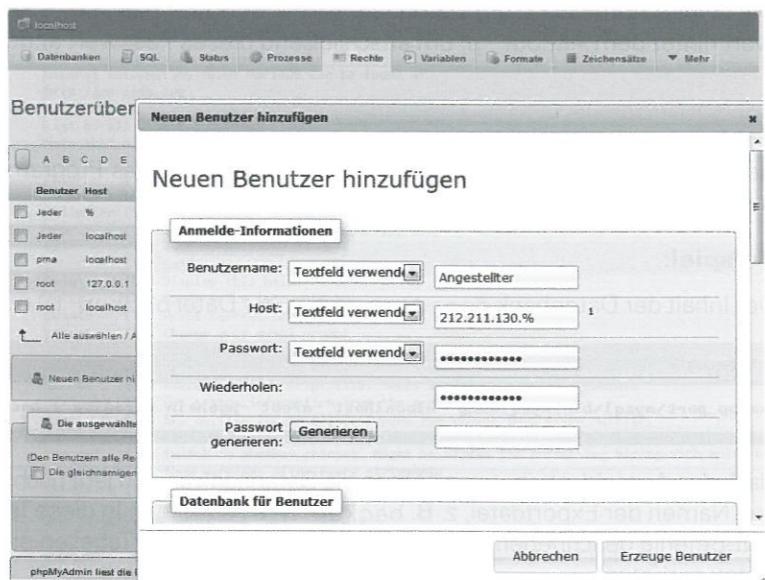


#### 8.4.2 Zugriffsrechte gewähren und widerrufen

MySQL speichert Rechte der Benutzer sehr differenziert in der Datenbank `mysql`.

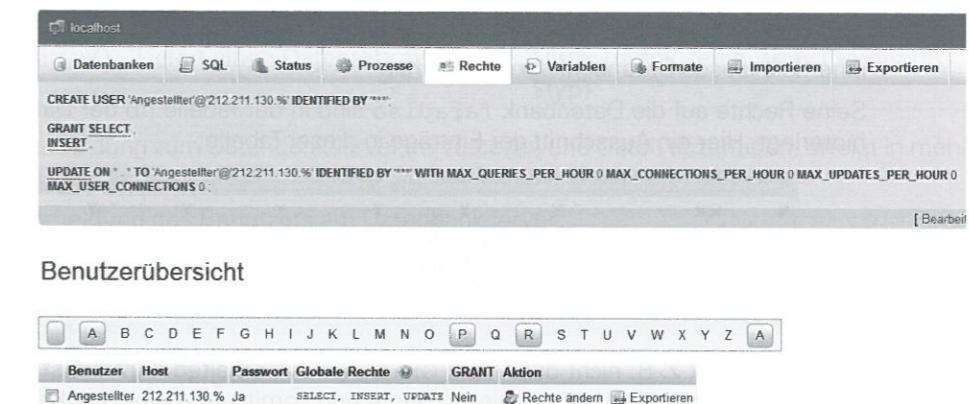
##### Beispiel:

Mit dem Client phpMyAdmin wird z. B. ein neuer Benutzer mit dem Benutzernamen 'Angestellter' und dem Passwort 'Angestellter' eingerichtet. Er soll sich nur von Hosts der Domäne 212.211.130.xxx anmelden können.



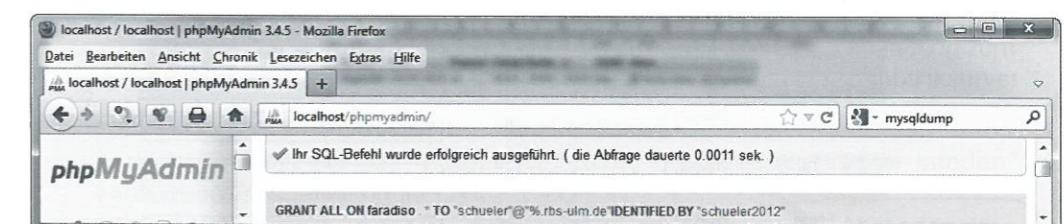
Über die Karteikarte **Rechte** auf der Startseite gelangt man zur Ansicht der in der Datenbank `mysql` bestehenden Benutzer. Hier wählt man **Neuen Benutzer hinzufügen**, wodurch sich obige Eingabemaske öffnet. Dort werden die gewünschten Anmelde-Informationen eingegeben. In der Zeile **Host** muss im Pulldown-Fenster die Option **Textfeld verwenden** verwendet werden eingestellt sein, um einen konkreten Host einzugeben, von dem aus sich der Benutzer am Datenbankserver anmelden darf. Soll sich der neue Benutzer von allen Rechnern eines bestimmten Netzes anmelden können, so legt man z. B. für das letzte Byte der Adresse den Platzhalter `%` fest.

Im Bereich **Globale Rechte** werden die Rechte des Benutzers ausgewählt, z. B. `SELECT` zum Lesen von Daten, `INSERT` zum Einfügen von Datensätzen in Tabellen und `UPDATE` zum Verändern vorhandener Daten. Mit Klick auf die Schaltfläche **Erzeuge Benutzer** wird der neue Benutzer durch eine `CREATE`-Anweisung angelegt und seine Benutzerrechte vergeben. Die erfolgreiche Ausführung wird gemeldet, indem der ausgeführte Befehl im SQL-Code angezeigt wird.



##### Beispiel:

Mittels eines SQL-Befehls wird ein neuer Benutzer `schueler` mit dem Passwort `schueler2012` angelegt. Er soll sich nur von Hosts der Domäne `rbs-ulm.de` anmelden können und nur die Datenbank `faradiso` mit allen Rechten bearbeiten können.



Das Schlüsselwort `GRANT` leitet den SQL-Befehl zur Gewährung von Benutzerrechten ein. Das Wort `ALL` legt alle Rechte für den Benutzer fest. Mit dem Attribut `ON faradiso.*` werden diese gewährten Rechte auf alle Tabellen der Datenbank `faradiso` begrenzt. Mit `*.*` könnte der Benutzer alle vorhandenen Datenbanken bearbeiten. Nach dem Begriff `TO` steht der Anmeldename des neuen Benutzers, z. B. `schueler` und nach dem Zeichen `@` der Host oder der Netzbereich, von dem aus der Benutzer sich anmelden darf. Das Anmeldepasswort folgt nach dem Schlüsselbegriff `IDENTIFIED BY`.

Der Vorteil dieser Methode der Rechtefestlegung mit SQL-Befehlen gegenüber der direkten Bearbeitung der Rechtebank mysql ist, dass der Administrator die Inhalte der Tabellen dieser Rechtebank nicht kennen muss und nicht entscheiden muss, in welche dieser Tabellen die Einträge verteilt werden müssen.

Die Rechte der einzelnen Benutzer können in der Benutzerübersicht des Clients phpmyadmin eingesehen werden.

Die Rechte des Benutzers schueler des Beispiels sind mit dem Begriff USAGE beschrieben. Dieses Schlüsselwort sagt aus, dass der Benutzer schueler keine globalen Rechte innerhalb dieses Datenbankservers besitzt.

Seine Rechte auf die Datenbank faradiso sind in der Tabelle db der Datenbank mysql hinterlegt. Hier ein Ausschnitt der Einträge in dieser Tabelle:

Host	Db	User	Select_priv	Insert_priv	Update_priv	Delete_priv	Create_priv	Drop_priv	Grant_priv
%	test		Y	Y	Y	Y	Y	Y	N
%	test\_%		Y	Y	Y	Y	Y	Y	N
localhost	phpmyadmin	pma	Y	Y	Y	Y	N	N	N
%:rbs-ulm.de	faradiso	schueler	Y	Y	Y	Y	Y	Y	N

Der Schüler hat z. B. nicht das Recht Grant\_priv erhalten (Eintrag ist N), da er sonst anderen Benutzern Rechte zuweisen könnte und somit seine Beschränkungen umgehen könnte.

Damit der Benutzer schueler z. B. keine Tabellen oder Datenbanken löschen kann, wird ihm das Recht Drop\_priv entzogen. Dies geschieht durch Bearbeiten seiner Rechte im Client phpmyadmin. Dort wird der Button N in der Zeile Drop\_priv ausgewählt und bestätigt.

Spalte	Typ	Funktion	Null	Wert
Host	char(60)			%:rbs-ulm.de
Db	char(64)			faradiso
User	char(16)			schueler
Select_priv	enum	-		<input type="radio"/> N <input checked="" type="radio"/> Y
Insert_priv	enum	-		<input type="radio"/> N <input checked="" type="radio"/> Y
Update_priv	enum	-		<input type="radio"/> N <input checked="" type="radio"/> Y
Delete_priv	enum	-		<input type="radio"/> N <input checked="" type="radio"/> Y
Create_priv	enum	-		<input type="radio"/> N <input checked="" type="radio"/> Y
Drop_priv	enum	-		<input checked="" type="radio"/> N <input type="radio"/> Y
Grant_priv	enum	-		<input type="radio"/> N <input checked="" type="radio"/> Y
References_priv	enum	-		<input type="radio"/> N <input checked="" type="radio"/> Y
Index_priv	enum	-		<input type="radio"/> N <input checked="" type="radio"/> Y
Alter_priv	enum	-		<input type="radio"/> N <input checked="" type="radio"/> Y

#### Hinweis:

Globale Berechtigungen gelten für alle Datenbanken auf einem Server.

Benutzer	Host	Passwort	Globale Rechte	GRANT	Aktion
Jeder	%	-	USAGE	Nein	<input type="checkbox"/> Rechte ändern <input type="checkbox"/> Exportieren
Jeder	localhost	Nein	USAGE	Nein	<input type="checkbox"/> Rechte ändern <input type="checkbox"/> Exportieren
Angestalter	212.211.130.%	Ja	SELECT, INSERT, UPDATE	Nein	<input type="checkbox"/> Rechte ändern <input type="checkbox"/> Exportieren
pma	localhost	Nein	USAGE	Nein	<input type="checkbox"/> Rechte ändern <input type="checkbox"/> Exportieren
root	127.0.0.1	Nein	ALL PRIVILEGES	Ja	<input type="checkbox"/> Rechte ändern <input type="checkbox"/> Exportieren
root	localhost	Nein	ALL PRIVILEGES	Ja	<input type="checkbox"/> Rechte ändern <input type="checkbox"/> Exportieren
schueler	%:rbs-ulm.de	Ja	USAGE	Nein	<input type="checkbox"/> Rechte ändern <input type="checkbox"/> Exportieren

#### 8.4.3 Bearbeiten einer MySQL-Datenbank mit PHP

PHP stellt verschiedene Funktionen zur Arbeit mit MariaDB zur Verfügung:

PHP-Funktion	Beschreibung	Beispiel
mysqli_connect()	Stellt eine Verbindung zum Datenbankserver her.	\$erg=mysqli_connect (\$host, \$user, \$password);
mysqli_db_query()	Schickt eine SQL-Abfrage an den Datenbankserver.	\$erg=mysqli_db_query ('faradiso', 'select * from kunden');
mysqli_num_rows()	Liefert die Anzahl der Datensätze einer Abfrage zurück.	\$anzahl=mysqli_num_rows(\$erg);
mysqli_num_fields()	Liefert die Anzahl der Felder eines Datensatzes der Abfrage.	\$anzahl=mysqli_num_fields(\$erg);
mysqli_close()	Schließt eine Verbindung zum Datenbankserver.	mysqli_close();
mysqli_fetch_array()	Liefert einen Datensatz als Array.	\$liste=mysqli_fetch_array (\$ergebnis, \$typ)

Eine Verbindung zum Datenbankserver herzustellen und Daten abzufragen, erfolgt in mehreren Schritten:

1. Anmeldung des Benutzers am Datenbankserver
2. Festlegen der Datenbank
3. Absenden der Abfrage an den Datenbankserver, speichern des Ergebnisses in einer Variablen
4. Schließen der Verbindung

Die Anmeldung des Benutzers am Datenbankserver geschieht mit der Funktion mysqli\_connect (\$host, \$user, \$passwd).

Eine SQL-Abfrage wird mit der Funktion mysqli\_db\_query (\$db, \$sql) abgeschickt. Die Funktion mysqli\_num\_rows (\$erg\_sql) liest z. B. die Anzahl der Datensätze einer Tabelle aus und speichert sie in der Variablen \$anz. Die Funktion mysqli\_close () trennt die Verbindung zum Datenbankserver.

Das Skript zur Ausgabe der Anzahl der Datensätze lautet:

```
<?
$db = mysqli_connect('localhost','root','');
$erg_sql = mysqli_db_query('faradiso', 'select * from kunden');
$anz = mysqli_num_rows($erg_sql);
echo $anz;
mysqli_close();
?>
```

Herstellen einer Verbindung zum Datenbankserver

Absenden der SQL-Anweisung  
Auslesen des Ergebnisses der SQL-Anweisung

Anzeigen des Ergebnisses  
Schließen der DB-Verbindung

#### Beispiel:

Es wird ein PHP-Skript entworfen, welches die Tabelle Kunden der Datenbank faradiso am Bildschirm als HTML-Tabelle ausgibt.

Das Skript baut zunächst die Verbindung auf und schickt die SQL-Anweisung ab. Die Anzahl der benötigten Spalten wird mit der PHP-Funktion mysqli\_num\_fields(\$erg\_sql) ermittelt und der Variablen \$anzahl übergeben. Mit einer for-Schleife wird der Tabellenkopf erzeugt, der mittels der Funktion mysqli\_field\_name (\$erg\_sql, \$i) mit den Feldnamen beschriftet wird.

Innerhalb einer While-Schleife wird mithilfe der Funktion mysqli\_fetch\_array (\$erg\_sql, MYSQL\_ASSOC) bei jedem Durchlauf ein Datensatz des Abfrageergebnisses in ein assoziatives Array \$zeile eingelesen. Eine Foreach-Schleife ermöglicht

für jeden einzelnen Datensatz den Zugriff auf die einzelnen Felder des Datensatzes und die Ausgabe innerhalb der HTML-Tabelle. Nachdem die Tags für das Tabellenende angegeben worden sind, wird die Verbindung zum Datenbankserver durch `mysqli_close()` geschlossen.

Das Skript lautet somit:

```
<?php
$db = mysqli_connect('localhost', 'root', '');
$erg_sql = mysqli_query('faradiso', 'select * from kunden');



||
||
||


```

Ausgabe der Spaltenüberschriften

```
</tr>
<tr> <?
while ($zeile = mysqli_fetch_array ($erg_sql, MYSQL_ASSOC)) {
    foreach ($zeile as $elem) {
        echo "<td bgcolor='#EFEFEF'><font size='1'> $elem
</font></td>";
    }
    ?></tr><?
}
?>
</table> <?
mysqli_close();
?>
```

Auslesen der einzelnen Datensätze

```
        ?>
        Ausgabe der einzelnen Feldinhalte eines Datensatzes
```

## 8.5 Daten über ODBC-Schnittstellen austauschen

Der direkte Austausch von Daten aus Datenbanken verschiedener Formate ist in der Regel nicht möglich. Deshalb wurde ein ODBC-Standard (von Open Database Connectivity = offene Datenbankverbindung) geschaffen, der über spezielle ODBC-Treiber den Austausch ermöglicht. Dabei übergibt das Datenbanksystem, z. B. MariaDB, die Daten an den ODBC-Treiber, der sie dann in das Format des zweiten Datenbanksystems, z. B. Access, umsetzt.



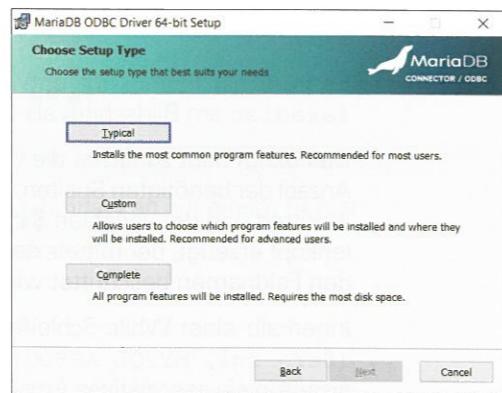
### Hinweis:

Über ODBC-Schnittstellen können verschiedene Datenbanksysteme zusammenarbeiten.

Für den Informationsaustausch zwischen Access und MariaDB stehen im Internet passende Treiber für Linux und für Windows zur Verfügung, z. B. als Installationsdatei `mariadb-connector-odbc-3.x.x-win64.msi`.

### Installation des MyODBC-Treibers

Die Installation erfolgt durch Starten der Installationsdatei. Der Begrüßungsbildschirm wird mit `Next >` bestätigt. Anschließend fordert das Programm zur Auswahl der gewünschten Installationsart auf (siehe Bild). Hier klickt man auf die Checkbox `Typical` und bestätigt mit `Next >`. Das anschließende Fenster zeigt den Verlauf der Installation und bestätigt den erfolgreichen Abschluss.



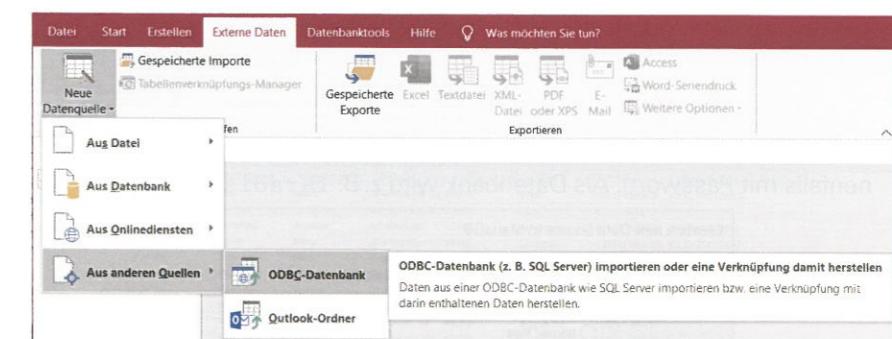
### Bearbeitung einer MariaDB-Datenbank mit Access

Nach dem Start von Access und dem Anlegen einer neuen Datenbank kann über die ODBC-Schnittstelle auf die Tabellen einer bestehenden MariaDB-Datenbank zugegriffen werden.

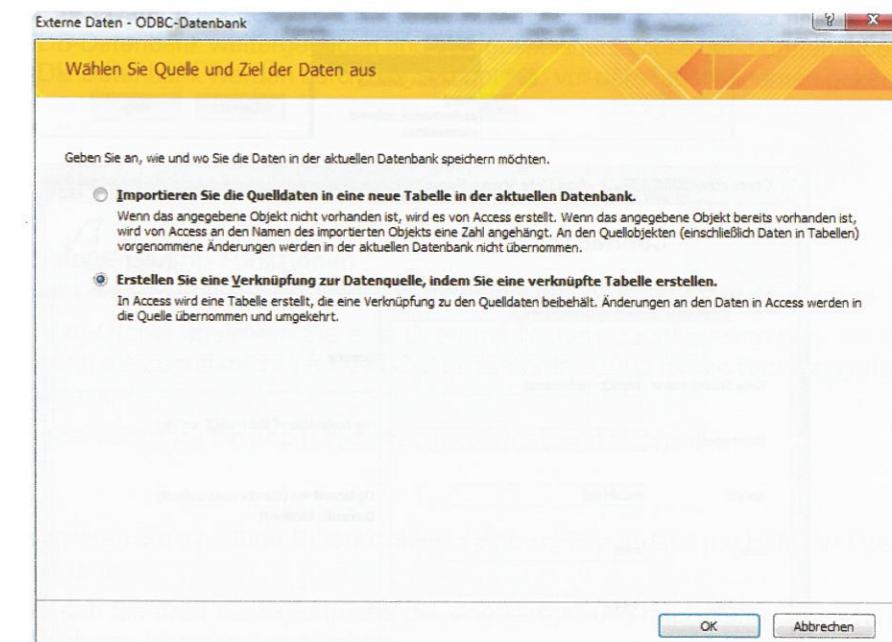
#### Beispiel:

Die Tabelle `Kunden` der MariaDB-Datenbank `faradiso` wird mit dem Datenbanksystem Access bearbeitet.

Dazu wählt man nach dem Start des Datenbanksystems Access im Menü `Externe Daten` die Option `ODBC-Datenbank`.



Der folgende Bildschirm lässt die Auswahl zu zwischen dem Import von Daten und dem Erstellen einer Verknüpfung. Man klickt auf die Auswahl `Erstellen Sie eine Verknüpfung ...` und bestätigt mit `OK`.



Die Daten werden dadurch nicht in die Access-Datenbank importiert, sondern bleiben in der MariaDB-Datenbank, können aber über die ODBC-Schnittstelle von Access aus bearbeitet werden. Würde die Option `Importieren...` gewählt, so würden die Daten als eigenständige Tabellen in die Access-Datenbank kopiert werden. Die MariaDB-Datenbank würde durch Bearbeitungen nicht mehr verändert werden.

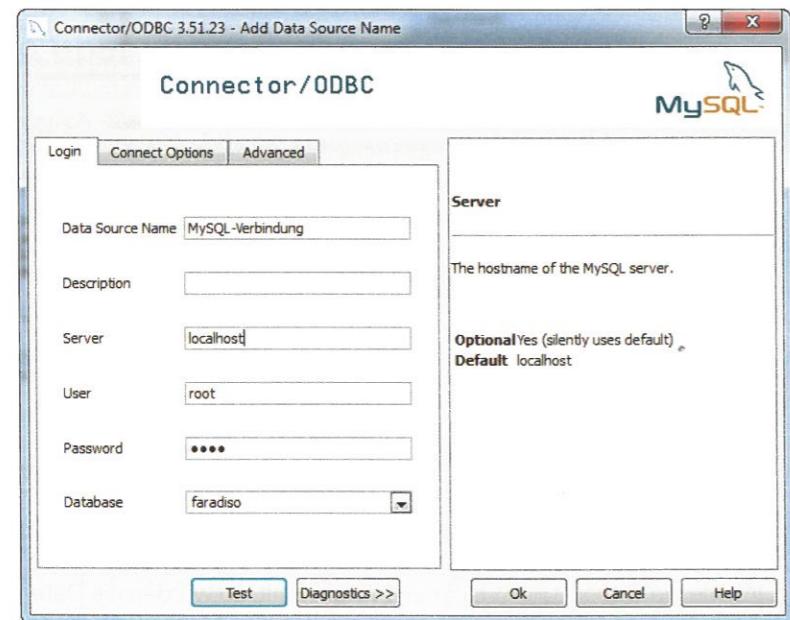
Das folgende Fenster stellt einen Dialog zur Auswahl der Datenquelle zur Verfügung. Auf der Registerkarte `Computerdatenquelle` wird die Schriftfläche `Neu...` gewählt.

Im nächsten Fenster wird die Verbindung als Systemdatenquelle festgelegt und im anschließenden Fenster als Treiber MariaDB ODBC 3.1 Driver bestimmt.

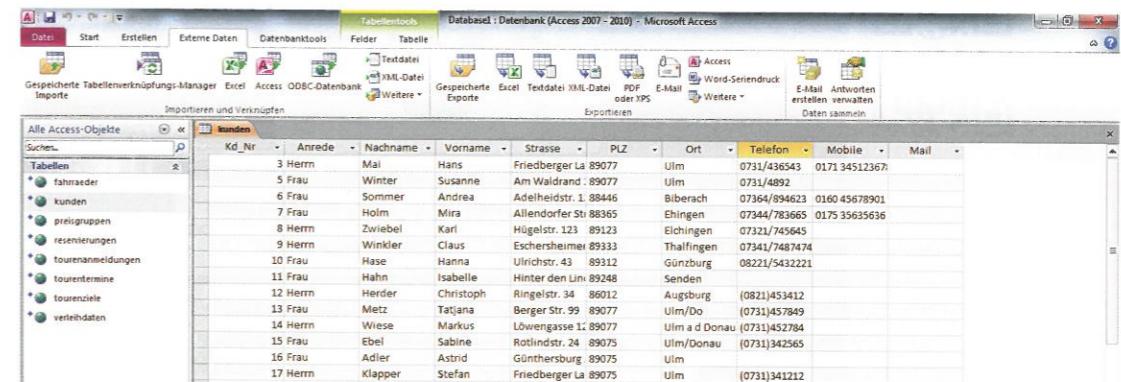
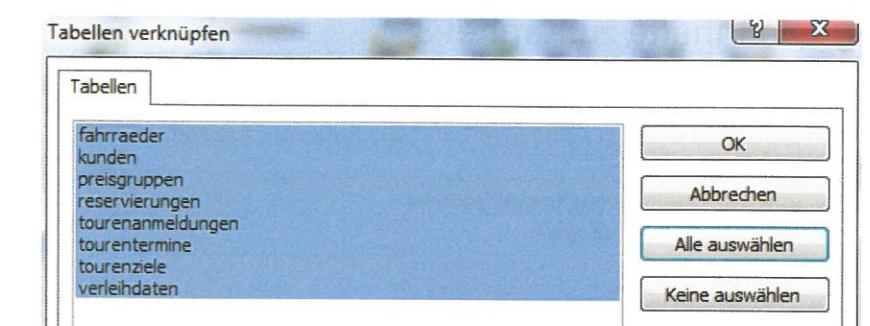
Der Name kann z. B. MariaDB-Verbindung sein.



Als Servername wird localhost eingetragen, der Standardbenutzer ist root (gegebenenfalls mit Passwort). Als Datenbank wird z. B. faradiso gewählt.



Nach Bestätigung mit Ok ist Access am MariaDB-Datenbankserver angemeldet und zeigt die in der MySQL-Datenbank faradiso vorhandenen Tabellen. Nach Betätigen der Schaltfläche Alle auswählen und Bestätigung mit Ok stehen alle Tabellen der MariaDB-Datenbank zur Bearbeitung zur Verfügung.



Änderungen, die nun z. B. in der Tabelle Kunden durchgeführt werden, werden nicht in der Access-Datenbank gespeichert, sondern werden über die ODBC-Schnittstelle an die MariaDB-Datenbank weitergegeben und dort gespeichert. Diese verknüpften Tabellen der MariaDB-Datenbank werden durch das Symbol \* vor dem Tabellennamen gekennzeichnet.

## 8.6 Aufgaben Kapitel 8

### Aufgabe 1. Arbeiten mit php-Funktionen

Erzeugen Sie im Ordner 'htdocs' des Webservers einen Unterordner 'testordner'.

In diesem Ordner speichern Sie eine Datei mit Namen 'zugriffsrechte.php', die den Datentyp und die Zugriffsrechte für den Ordner 'testordner' und für die Datei 'zugriffsrechte.php' ausgibt.

Verwenden Sie dazu die php-Funktionen fileperms() und filetype().

### Aufgabe 2.

a) Exportieren Sie eine Ihrer Datenbanken in eine sql-Exportdatei mit Hilfe der Oberfläche phpMyAdmin.

Verwenden Sie dazu die Exportmethode „angepasst“ und informieren Sie sich über die verschiedenen Einstellungsmöglichkeiten.

b) Importieren Sie eine Datenbank eines Ihrer KollegInnen.

## 8.7 Digitale Inhalte zu Kapitel 8

## Aufgabe 1

— Spielen Sie selbst oder im Klassenverband das Kahoot!-Quiz mit dem Titel „Datenbanken 36087 Kapitel 8 MariaDB“.

Quelle: Kahoot-App oder [www.kahoot.com](http://www.kahoot.com)

## Aufgabe 2

<https://vel.plus/mXHB>



### Aufgabe 3

<https://vel.plus/NAjk>



Datenbanken im Internet

Was ist Apache?

2020-05-01

---

A B C D E F G H I J K L M N O P  
Q R S T U V W X Y Z Ä Ö Ü

## Aufgabe 4

<https://vel.plus/lkmQ>



PHP-Programm - Ausgabe eines Arrays

2022-01

```
$ergebnis = array_walk($zahlen, 'quadrat');

for ($i = 0; $i < count($zahlen); $i++) {
    echo $zahlen[$i] . " ";
}

echo "
```

**Aufgabe**

In einem Array werden die Zahlen 10 bis 20 gespeichert. Anschließend werden die Elemente dieses Arrays mithilfe einer selbstdefinierten Funktion quadriert und die Ergebnisse am Bildschirm ausgegeben.

Ordnen Sie die Zeilen des Quelltextes.

}

## Aufgabe 5

<https://vel.plus/d67k>



Datenbanken mit PHP bearbeiten

Schließt eine Verbindung zum Datenbankserver

Liefert die Zeilenanzahl

Liefert die Felder des Datensatzes

Stellt eine Verbindung zum Datenbankserver her.

mysql\_close()

mysql\_connect()

mysql\_num\_rows()

mysql\_fetch\_array()

OK

Aufgabe

PHP stellt verschiedene Funktionen zur Arbeit mit MariaDB oder MySQL zur Verfügung. Ordnen Sie die Erklärungen den Befehlen zu.