Datenbanktabelle pflegen mit PHP und MySQL

Anzeigen, Hinzufügen, Ändern und Löschen von Datensätzen

Dr. Volker Thormählen, 13. März 2018

Inhalt

Abbildungen

Skripte

Listings

Tabellen

1 Datenbank und Datenbanktabelle einrichten

In diesem Beitrag wird demonstriert, wie die Tabelle „*players*“ in der relationalen Datenbank „*team*“ mit vier web-orientieren Sprachen gepflegt werden kann. Zum Einsatz kommen folgende vier Programmiersprachen.

- Die Skriptsprache Hypertext Preprocessor (PHP)[1]
- Die Auszeichnungssprache *Hypertext Markup Language* (HTML)
- Die Formatierungssprache Cascading Style Sheets (CSS)
- Die Abfragesprache *Structured Query Language* (SQL)

Zunächst werden zwei PHP-Skripte vorgestellt, die zum automatischen Erstellen der Datenbank mit dem Namen „*team*“ und der Datenbanktabelle „*players*“ dienen.

Voraussetzung dafür ist, dass die Open Source Software XAMPP[2] im Verzeichnis C:\xampp installiert ist. Dieses Softwarepaket umfasst u. a. einen lokalen Apache Webserver, ein relationales Datenbanksystem und die Skriptsprache PHP.

Mit dem Skript create-db.php (siehe Tabelle 1) wird die Datenbank erstellt. Es befindet sich im Verzeichnis C:\xampp\htdocs\players. Im Internet-Browser wird es mit folgender Eingabe aufgerufen: http://localhost/players/create-db.php. Die Verkettung der PHP-Skripte erfolgt jeweils mit dem href-Attribut des HTML a-Tags[3]. Beispiel: Das Skript create-db.php enthält am Ende folgenden Link (siehe Skript 1)

```
<a href="create-tbl.php">Datenbanktabelle 'players' erstellen.</a>
```

Der Link wird dazu verwendet, um das nächste PHP-Skript in der Vorgangskette auszurufen.

Mit dem Skript „insert-data.php“ (siehe Skript 3Skript 4) werden danach fünf Datensätze in die Datenbanktabelle *players* eingefügt. Das Skript connect.php (siehe Skript 4) wird in jedes der drei zuvor genannten PHP-Skripte eingebunden.

Skript	Skripte im Verzeichnis C:\xampp\htdocs\players	PHP-Datei
Vorgänge zur Einrichtung der relationalen Datenbank mit dem Namen 'team'		
1	Datenbank 'team' neu erstellen	create-db.php
2	Datenbanktabelle 'players' neu erstellen.	create-tbl.php
3	Datensätze in die Datenbanktabelle 'players' einfügen.	insert-data.php
4	Verbindung zum lokalen Webserver herstellen.	connect.php
Vorgänge zur Pflege der Datenbanktabelle mit dem Namen 'players'		
5	Alle Datensätze anzeigen.	view.php
6	Datensätze seitenweise anzeigen.	view-paginated.php
7	Anwendung beenden	quit.php
8 ff	Neuen Datensatz hinzufügen oder vorhandenen ändern.	records.php
12 f	Vorhandenen Datensatz löschen.	delete.php
Die mit 1 bis 4 und 7 nummerierten PHP-Skripte stammen vom Autor. Die übrigen PHP-Skripte sind im Tutorial (siehe [1]) enthalten. Sie wurden vom Autor geändert bzw. erweitert und mit deutschen Kommentaren versehen.		

[1] PHP ist eine Skriptsprache, die hauptsächlich zur Erstellung dynamischer Webseiten verwendet wird.
[2] Das Akronym XAMPP enthält die Anfangsbuchstaben **X** für ein beliebiges Betriebssysteme, **A**pache, **M**ariaDB, **P**HP und **P**erl.
[3] Der HTML a-Tag wird auch als Anker oder Link bezeichnet.

Tabelle 1: Benötigte PHP-Skripte

Skript	Formatvorlagen im Verzeichnis C:\xampp\htdocs\players …	CSS-Datei
5 und 6	für die Skripte view.php und view-paginated	table.css
8 ff	für das Skript records.php	styles.css

Tabelle 2: Benötigte mehrstufige Formatvorlagen (CSS)

1.1 Datenbank '*team*' einrichten

```
<?php
/* Datei:   create-db.php
   Aufgabe: Neue Datenbank erstellen.
*/
   $server  = "localhost"; // lokaler Webserver
   $user    = "*****";     // Benutzername (bitte anpassen)
   $pass    = "*****";     // Passwort (bitte anpassen)
   $db      = "team";      // Datenbankname
   // Datenbankverbindung öffnen.
   $conn = mysqli_connect($server, $user, $pass);
   if (!$conn) {
      die("Beim Verbindungsaufbau ist ein Fehler aufgetreten: " .
         mysqli_connect_error() . "<br>\n");
   } else {
      echo "Die Datenbankverbindung wurde erfolgreich aufgebaut. <br>\n";
      if(mysqli_select_db($conn,$db)) {
         die ("FEHLER: Die Datenbank '$db' ist bereits vorhanden. <br>\n");
      } else {
         // Datenbank erstellen
         $sql  = "CREATE DATABASE IF NOT EXISTS $db ";
         $sql .= "CHARACTER SET utf8 ";      // oder CHARACTER SET latin1
         $sql .=  "COLLATE utf8_general_ci"; // oder COLLATE latin1_swedish_ci
         if (mysqli_query($conn, $sql)) {
            echo "Die Datenbank '$db' wurde erfolgreich erstellt. <br>\n";
         } else {
            die ("FEHLER: Die Erstellung der Datenbank '$db' ist fehlgeschlagen: <br>\n" .
               mysqli_error($conn) . "<br>\n");
         }
      }
   }
   // Zuvor aufgebaute Datenbankverbindung schliessen.
   mysqli_close($conn);
?>
   <!-- Nächstes Skript aufrufen -->
   <p><a href="create-tbl.php">Datenbanktabelle 'players' erstellen.</a></p<
```

Skript 1: Datenbank 'team' erstellen (create-db.php)

Das obige Skript gibt folgendes aus*:*

Die Datenbankverbindung wurde erfolgreich aufgebaut.

Bei erfolgreicher Verbindung zusätzlich: Die Datenbank '*team*' wurde erfolgreich erstellt.

Bei missglückter Verbindung zusätzlich: FEHLER: Die Datenbank '*team*' ist bereits vorhanden.

Datenbanktabelle 'players' erstellen.

Wie oben erwähnt, falls die Datenbank mit dem Namen '*team*' bereits existiert, wird eine entsprechende Fehlermeldung ausgegeben: „FEHLER: Die Datenbank 'team' ist bereits vorhanden.“

Die letzte Zeile in Skript 1 enthält also einen Verweis auf das nächste Skript in der oben erwähnten Vorgangskette: *create-tbl.php*.

1.2 Datenbanktabelle '*players*' einrichten

Das Skript *create-tbl.php* erstellt automatisch eine Datenbanktabelle mit dem Namen '*players*' in der Datenbank „team“. Die Tabelle umfasst nur 3 Spalten. Die Namen dieser Spalten lauten: '*id*', '*first-name*' und '*lastname*'.

```
/* Datei:   create-tbl.php
   Aufgabe: Datenbanktabelle 'players' neu erstellen.
*/
   // Verbindung zum lokalen Webserver aufbauen.
   include_once('connect-db.php');
   $tbl = "players";
   // Datenbanktabelle 'players' mit 3 Feldern neu erstellen.
   $sql = "CREATE TABLE IF NOT EXISTS $tbl (
      id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
      firstname VARCHAR(32) NOT NULL,
      lastname VARCHAR(32) NOT NULL
   ) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci AUTO_INCREMENT=1";
   if (mysqli_query($conn, $sql)) {
      echo "Die Datenbanktabelle '$tbl' wurde erfolgreich eingerichtet. <br>\n";
   } else {
      echo "Das Einrichten der Datenbanktabelle '$tbl' ist misslungen: "
         . mysqli_error($conn) . "<br>\n";
   }
   mysqli_close($conn);
?>
   <!-- Nächstes Skript aufrufen -->
   <p>
      <a href="insert-data.php">Datensätze in die Datenbanktabelle 'players' einfügen.</a>
   </p>
```

Skript 2: Datenbanktabelle 'players' mit 3 Feldern neu erstellen (create-tbl.php)

Die erste Spalte ('id') wird automatisch mit nummerischen Werten belegt, siehe Klausel „AUTO_INCREMENT“ in **Fehler! Verweisquelle konnte nicht gefunden werden.**. Die beiden folgenden Spalten enthalten variabel lange Zeichenketten des Datentyps *varchar*[4].

Die Erstellung der Datenbanktabelle '*players*' kann mit folgender Eingabe in die Befehlszeile eines Internet-Browsers aufgerufen werden: http://localhost/players/create-tbl.php.

Die Klausel 'IF NOT EXISTS' in der SQL 'CREATE TABLE' Anweisung verhindert, dass die Datenbanktabelle 'players' doppelt angelegt wird.

[4] Der SQL Datentyp *varchar* repräsentiert eine Zeichenkette variabler Länge.

Skript 2 gibt folgendes aus:

> Die Datenbanktabelle 'players' wurde erfolgreich eingerichtet.
> Datensätze in die Datenbanktabelle 'players' einfügen.

Die letzte Zeile in **Fehler! Verweisquelle konnte nicht gefunden werden.** enthält also den Verweis auf das nächste Skript in der oben erwähnten Vorgangskette, nämlich insert-data.php

1.3 Datensätze in die Datenbanktabelle '*players*' einfügen

Das Skript „*insert-data.php*“ fügt automatisch fünf Datensätze in die zuvor erzeugte Datenbanktabelle '*players*' ein. Die Befehlszeile des jeweiligen Internet-Browsers ist dafür wie folgt zu belegen: „*http://localhost/players/insert-data*.php“.

```
<?php
/*    Datei:   insert-data.php
      Aufgabe: Datensätze in Datenbanktabelle einfügen.
*/
   // Datenbankverbindung öffnen.
   include_once ('connect-db.php');
   // Datensätze in der Zieltabelle zählen.
   function countRows($conn, $sql){
      return mysqli_num_rows(mysqli_query($conn, $sql));
   }
   // Datensätze in die Datenbanktabelle 'players' einfügen.
   $tbl = "players";
   $sql = "INSERT INTO $tbl (id, firstname, lastname) VALUES
      ('', 'Bob', 'Baker'),
      ('', 'Tim', 'Thomas'),
      ('', 'Jim', 'Connor'),
      ('', 'Daniel', 'Greyson'),
      ('', 'Sam', 'Smith')";
   // Meldung ausgeben.
   if (mysqli_query($conn, $sql)) {
      // Datensätze in der Datenbanktabelle 'players' zählen.
      $sql = "SELECT * FROM $tbl";
      $count = countRows($conn, $sql);
      echo "Die Datenbanktabelle '$tbl' enthält derzeit $count Datensätze. <br>\n";
   } else {
      echo "Fehler: " . $sql . "<br>\n" . mysqli_error($conn) . "<br>\n";
   }
   // Zuvor geöffnete Datenbankverbindung schliessen.
   mysqli_close($conn);
?>
   <!-- Nächstes Skript aufrufen -->
   <p>
      <a href="view.php">Alle Datensätze anzeigen</a>
   </p>
```

Skript 3: Fünf Datensätze in die Datenbanktabelle 'players' einfügen (insert-data.php)

Skript 3 gibt folgendes aus:

> Die Datenbanktabelle '*players*' enthält derzeit 5 Datensätze.
> Alle Datensätze anzeigen

Die letzte Zeile in Skript 3 enthält also einen Verweis auf das Skript *view.php* in der oben erwähnten Vorgangskette.

Die Verbindung zum lokalen Webserver erfolgt mit einem Verweis auf die externe PHP-Datei mit dem Namen *„connect-db.php“*. Diese PHP-Datei enthält folgenden Code:

```
<?php
/* Datei:   connect-db.php
   Aufgabe: Verbindung zur Datenbank herstellen
*/
   error_reporting(E_ALL);

   $server = 'localhost';  // lokaler Webserver
   $user   = '******';     // Benutzername   (bitte anpassen)
   $pass   = '******';     // Passwort       (bitte anpassen)
   $db     = 'team';       // Datenbankname

   // PHP mit der Datenbank 'team' verbinden.
   $conn = mysqli_connect($server, $user, $pass, $db);

   if (!$conn) {
      die("FEHLER: Keine Verbindung zur Datenbank '$db' aufgebaut: "
         . mysqli_connect_error() . "<br>\n");
   }

   // Zeichensatz utf8 verwenden.
   mysqli_set_charset($conn,"utf8");
?>
```

Skript 4: Verbindung zur Datenbank 'team' herstellen (connect-db.php)

2 Zwischenstand

Die im Ablaufplan (siehe Abbildung 1) links angeordneten Vorgänge wurden vorstehend ausführlich dargestellt. Die detaillierte Beschreibung der übrigen Vorgänge erfolgt nachstehend.

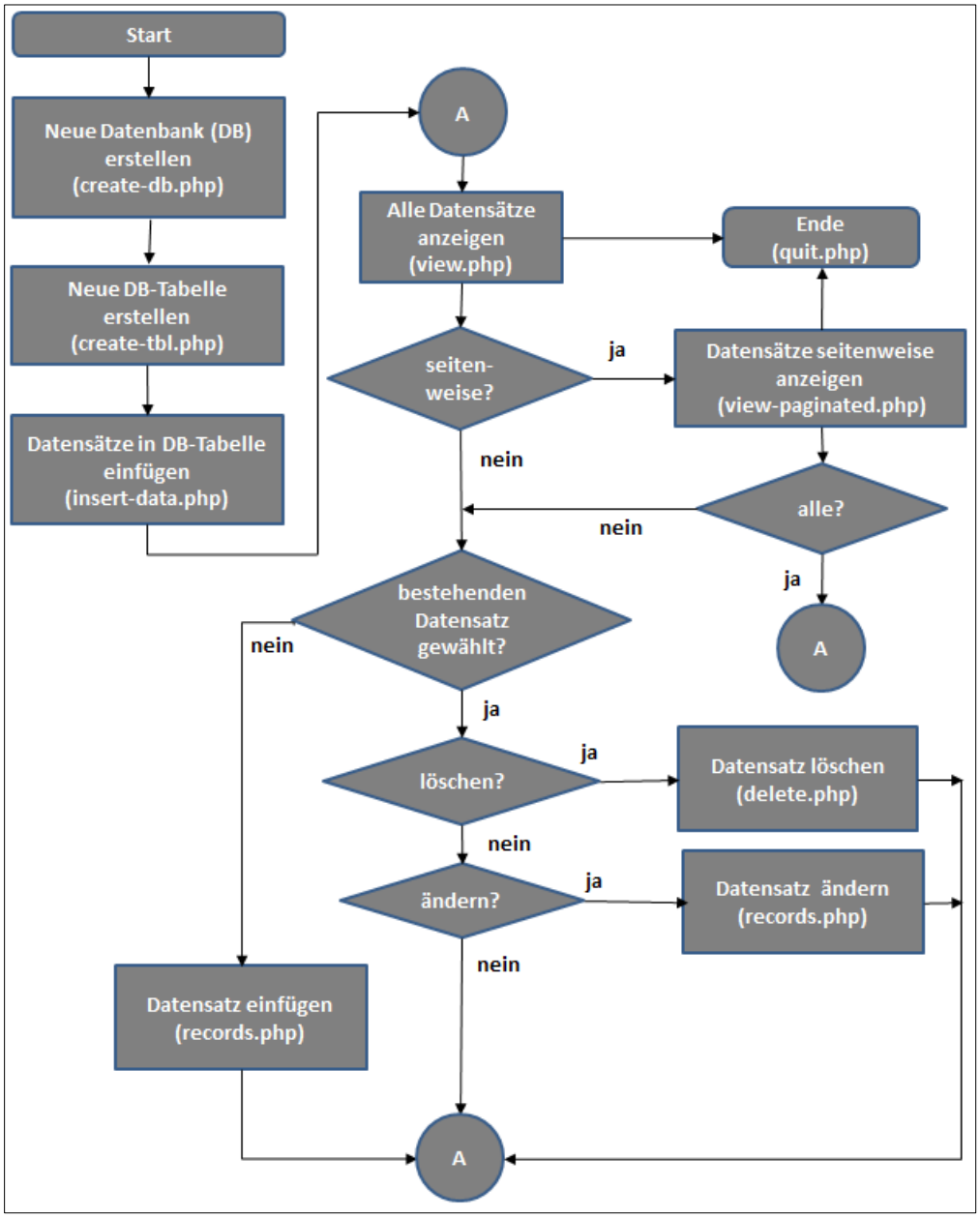

Abbildung 1: Grober Ablaufplan

3 Datensätze in der Datenbanktabelle '*players*' anzeigen

3.1 Alle Datensätze anzeigen

Datensätze anzeigen

Alle anzeigen | Seitenweise anzeigen

ID	Vorname	Nachname		
1	Bob	Baker	ändern	löschen
2	Tim	Thomas	ändern	löschen
3	Jim	Connor	ändern	löschen
4	Daniel	Greyson	ändern	löschen
5	Sam	Smith	ändern	löschen

Datensatz hinzufügen
Vorgang abbrechen

Abbildung 2: Alle Datensätze (ID = 1, ..., 5) anzeigen

Mit dem Skript *insert-data.php* werden die Datensätze (ID = 1, ..., 5) in der Datenbanktabelle 'players' automatisch erzeugt (siehe Abbildung 2).

Datensätze anzeigen

Alle anzeigen | Seitenweise anzeigen

ID	Vorname	Nachname		
1	Bob	Baker	ändern	löschen
2	Tim	Thomas	ändern	löschen
3	Jim	Connor	ändern	löschen
4	Daniel	Greyson	ändern	löschen
5	Sam	Smith	ändern	löschen
6	Hans	Hansen	ändern	löschen
7	Siegfried	Sauerland	ändern	löschen
8	Volker	Thormählen	ändern	löschen

Datensatz hinzufügen
Vorgang abbrechen

Abbildung 3: Alle Datensätze (ID = 1, ..., 8) anzeigen

Die Datensätze (ID = 6, 7, 8) in der Datenbanktabelle 'players' wurden manuell hinzugefügt (siehe Abbildung 3).

3.2 Quellcode des Skripts view.php

```
<!--Datei:   view.php
 Aufgabe: Alle Datensätze anzeigen.
-->
<!DOCTYPE html>
<html lang="DE">
<head>
   <title>Datensätze anzeigen</title>
   <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
   <link rel="stylesheet" type="text/css" href="table.css" media="all" />
</head>
<body>
   <h1>Datensätze anzeigen</h1>
   <p>
      <b>Alle anzeigen</b> | <a href="view-paginated.php"><b>Seitenweise anzeigen</b></a>
   </p>
   <?php
       // Verbindung zur Datenbank herstellen.
      include_once('connect-db.php');
      // Alle Datensätze in der Datenbanktabelle 'players' anzeigen.
      $sql = "SELECT * FROM players ORDER BY id";
      if ($result = mysqli_query($conn, $sql)) {
         // Datensätze anzeigen, falls vorhanden.
         if (mysqli_num_rows($result) > 0) {
            // Datensätze in einer html-Tabelle anzeigen.
            echo "<table border='1' cellpadding='10'>";
            // Tabellenkopf ausgeben und Datensätze auflisten.
            echo "<tr><th>ID</th> <th>Vorname</th><th>Nachname</th><th></th><th></th></tr>";
            while($row = mysqli_fetch_assoc($result)) {
               // Für jeden Datensatz eine Tabellenzeile ausgeben.
               echo "<tr>";
                  echo "<td>" . $row["id"]         . "</td>";
                  echo "<td>" . $row["firstname"]  . "</td>";
                  echo "<td>" . $row["lastname"]   . "</td>";
                  echo "<td><a href='records.php?id=" . $row["id"] . "'>ändern</a></td>";
                  echo "<td><a href='delete.php?id="  . $row["id"] . "'>löschen</a></td>";
               echo "</tr>";
            }
            // Ausgabe der Tabelle abschliessen.
            echo "</table>";
         } else {
            echo "FEHLER: Keine Datens&auml;tze zum Anzeigen in Tabelle 'players' gefunden!";
         }
         mysqli_free_result($result);
      } else {
         echo ("FEHLER: SQL SELECT Abfrage ist fehlgeschlagen!" . mysqli_error($conn));
      }
      mysqli_close($conn);
   ?>
   <br>
   <a href="records.php">Datensatz hinzufügen</a>
   <br>
   <a href="quit.php">Vorgang abbrechen</a>
</body>
</html>
```

Skript 5: Alle Datensätze anzeigen (view.php)

```
/* table.css */
table, th, td {
     border: 1px solid black;
}
table {
   border-collapse: collapse;
}
tr:hover {
   background-color: #f5f5f5;
}
th {
   background-color: #4CAF50;
   color: white;
}
a {
    color: hotpink;
}
```

Listing 1: Mehrstufige Formatvorlage (*table.css*) für die Skripte *view.php* und *view-paginated.php*

3.3 Datensätze seitenweise anzeigen

Abbildung 4: Datensätze seitenweise anzeigen, 1. Seite

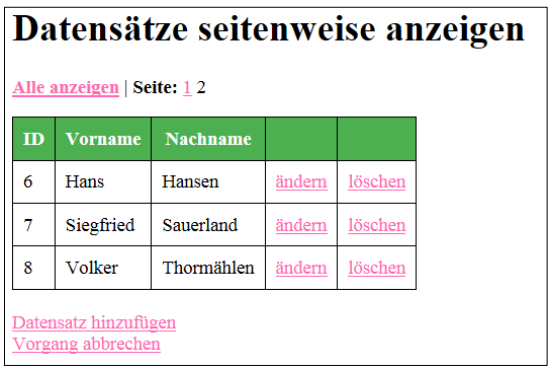

Abbildung 5: Datensätze seitenweise anzeigen, 2. Seite

Wenn insgesamt nur fünf Datensätze in der Datenbanktabelle 'players' enthalten sind, ist die seitenweise Anzeige der Datensätze identisch mit der Anzeige aller Datensätze.

Bei der seitenweisen Anzeige der Datensätze erfolgt das Blättern mit den verlinkten Seitenzahlen hinter dem Führungstext '**Seite:**' (siehe Abbildung 4 und Abbildung 5, über dem Tabellenkopf).

3.4 Quellcode des Skripts *view-paginated.php*

```
<!--Datei:  view-paginated.php
   Aufgabe: Datensätze seitenweise anzeigen.
-->
<!DOCTYPE html>
<html lang="DE">
   <head>
      <title>Datensätze seitenweise anzeigen</title>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
      <link rel="stylesheet" type="text/css" href="table.css" media="all" />
   </head>
   <body>
      <h1>Datensätze seitenweise anzeigen</h1>
      <?php
         // Verbindung zur Datenbank herstellen.
         include_once('connect-db.php');
         // Maximale Zahl der Datensätze je HTML-Seiten bestimmen.
         $per_page = 5;
         // Gesamtzahl der Datensätze in der Datenbanktabelle ermitteln.
         $sql = "SELECT * FROM players ORDER BY id";
         if ($result = mysqli_query($conn, $sql)) {
            if (mysqli_num_rows($result) > 0) {
               $total_results = mysqli_num_rows($result);
               $total_pages = ceil($total_results / $per_page);
               if (isset($_GET['page']) && is_numeric($_GET['page'])) {
                  $show_page = $_GET['page'];
                  if ($show_page > 0 && $show_page <= $total_pages) {
                     $start = ($show_page - 1) * $per_page;
                     $end = $start + $per_page;
                  } else {
                     $start = 0;
                     $end = $per_page;
                  }
               } else {
                  $start = 0;
                  $end = $per_page;
               }
               // Seitenzahlen aufsteigend anzeigen.
               echo "<p><a href='view.php'><b>Alle anzeigen</b></a> | <b>Seite:</b> ";
               for ($i = 1; $i <= $total_pages; $i++) {
                  if (isset($_GET['page']) && $_GET['page'] == $i) {
                     echo $i . " ";
                  } else {
                     echo "<a href='view-paginated.php?page=$i'>$i</a> ";
                  }
               }
               echo "</p>";
               // Gefundene Datensätze tabellarisch anzeigen.
               echo "<table border='1' cellpadding='10'>";
               echo "<tr><th>ID</th><th>Vorname</th><th>Nachname</th><th></th><th></th><th></th></tr>";
               for ($i = $start; $i < $end; $i++) {
                  if ($i == $total_results) { break; }
                  // bestimmte Zeile finden
                  if (mysqli_data_seek($result, $i)) {
                     $row = mysqli_fetch_row($result);
                     // HTML-Tabelle füllen und anzeigen
                     echo "<tr>";
                     echo "<td>" . $row[0] . "</td>";
                     echo "<td>" . $row[1] . "</td>";
```

```
                    echo "<td>" . $row[2] . "</td>";
                    echo "<td><a href='records.php?id=" . $row[0] . "'>ändern</a> </td>";
                    echo "<td><a href='delete.php?id="  . $row[0] . "'>löschen</a></td>";
                    echo "</tr>";
                } else {
                    echo "Die Suche nach Zeile $i ist fehlgeschlagen:" . mysqli_error($conn) . "\n";
                    continue;
                }
            }
            echo "</table><br />";
            mysqli_free_result($result);
        } else {
            echo "FEHLER: Keine Datensätze in der Datenbanktabelle gefunden!";
        }
    } else {
        echo ("FEHLER: SQL SELECT Abfrage ist fehlgeschlagen!" . mysqli_error($conn));
    }
    // Verbindung zur Datenbank lösen
    mysqli_close($conn);
?>
<a href="records.php">Datensatz hinzufügen</a>
<br>
<a href="quit.php">Vorgang abbrechen</a>
</body>
</html>
```

Skript 6: Datensätze seitenweise anzeigen (view-paginated.php)

Im obigen Skript bestimmt die gelb hervorgehobene Code-Zeile **$per_page = 5** die höchste Zahl der Datensätze je angezeigter HTML-Seite.

Durch das Anklicken des Links „Vorgang abbrechen wird jeweils das Skript 'quit.php' (siehe oben) aufgebrufen. Es dient zur Beendigung der Anwendung durch den Benutzer:

```
<?php
session_start();
session_destroy();
echo "<b>Abbruch durch Benutzer!</b>
<br>";
?>
```

Skript 7: Anwendung beenden (quit.php)

4 Datenbanktabelle '*players*' pflegen

Egal, ob die bestehenden Datensätze insgesamt (siehe Skript 5) oder seitenweise (siehe Skript 6) angezeigt werden, die beiden Verweise

```
<a href="records.php">Datensatz hinzufügen</a>
<td><a href='records.php?id=" . $row[0] . "'>ändern</a></td>;
```

führen jeweils zum Skript mit dem Namen *'records.php'*. Der Pseudocode für dieses Skript wird in Abbildung 6 dargestellt.

```
Verbindung zur Datenbank aufbauen (connect-db.php)
HTML-Formular anzeigen (renderForm).
WENN die Variable 'id' gesetzt ist (bestehender Datensatz),
|  WENN im HTML-Formular die Schaltfläche 'submit' gedruckt wurde,
|  |  WENN die Variable 'id' nummerisch ist,
|  |  |  die Werte der Felder des HTML-Formular auslesen.
|  |  |  Überprüfen, ob beide Pflichtfelder gefüllt sind.
|  |  |  WENN die Eingabedaten ungültig sind,
|  |  |  |  Fehlermeldung ausgeben: Pflichtfeld/er ist/sind leer!
|  |  |  |  Das HTML-Formular anzeigen (renderForm).
|  |  |  SONST
|  |  |  |  Die SQL UPDATE Anweisung vorbereiten.
|  |  |  |  WENN die Vorbereitung der UPDATE Anweisung geklappt hat,
|  |  |  |  |  die DB-Tabelle 'players' aktualisieren.
|  |  |  |  SONST
|  |  |  |  |  Fehlermeldung: Vorbereitete UPDATE Anweisung fehlerhaft.
|  |  |  |  ENDEWENN
|  |  |  |  Den Benutzer umleiten zur Datensatzanzeige (view.php).
|  |  |  ENDEWENN
|  |  SONST
|  |  |  Fehlermeldung ausgeben: Die Variable 'id' ist nicht nummerisch!
|  |  ENDEWENN
|  } SONST
|  |  WENN der Wert der Variablen 'id' nummerisch und positiv ist:
|  |  |  Die SQL SELECT Anweisung vorbereiten.
|  |  |  WENN die Vorbereitung der SELECT Anweisung geklappt hat,
|  |  |  |  den Datensatz in der DB-Tabelle 'players' auswählen.
|  |  |  |  das HTML-Formular anzeigen (renderForm).
|  |  |  SONST
|  |  |  |  Fehlermeldung ausgeben: SELECT Anweisung fehlerhaft.
|  |  |  ENDEWENN
|  |  SONST
|  |  |  Den Benutzer umleiten zur Datensatzanzeige (view.php).
|  |  ENDEWENN
|  ENDEWENN
SONST (neuer Datensatz)
|  WENN im HTML-Formular die Schaltfläche 'submit' gedruckt wurde,
|  |  die Eingabedaten des HTML-Formulars bereinigen und aufbereiten.
|  |  WENN die Eingabedaten fehlerhaft sind,
|  |  |  Fehlermeldung ausgeben.
|  |  |  HTML-Formular anzeigen (renderForm).
|  |  SONST
|  |  |  Die SQL INSERT INTO Anweisung vorbereiten.
|  |  |  WENN die Vorbereitung der INSERT INTO Anweisung geklappt hat,
|  |  |  |  neuen Datensatz in DB-Tabelle 'players' einfügen.
|  |  |  |  Erfolgsmeldung ausgeben.
|  |  |  SONST
|  |  |  |  Fehlermeldung ausgeben.
|  |  |  ENDEWENN
|  |  |  Den Benutzer umleiten zur Datensatzanzeige (view.php).
|  |  ENDEWENN
|  SONST
|  |  Das HTML-Formular anzeigen (renderForm).
|  ENDEWENN
ENDEWENN
Verbindung zur Datenbank beenden.
```

Abbildung 6: Pseudocode des PHP-Skripts 'records.php'

Im Wesentlichen kann anhand von Abbildung 6 nachverfolgt werden, unter welchen Bedingungen ein Datensatz in der Datenbanktabelle '*players*' mittels SQL

- geändert (UPDATE Anweisung),
- angezeigt (SELECT Anweisung) oder
- hinzugefügt (INSERT Anweisung)

wird.

4.1 Bestehenden Datensatz ändern

Wenn der bestehende Datensatz mit der Identnummer 6 geändert werden soll, ist folgendes HTML-Formular auszufüllen und abzuschicken (siehe Abbildung 7):

Abbildung 7: Mit CSS gestyltes HTML-Formular für den Vorgang 'Datensatz ändern'

Voraussetzung für diesen Vorgang mit der Identnummer 6 ist, dass der Vorgang 'Datensatz hinzufügen' vorher erfolgreich abgeschlossen wurde.

Das wiederum setzt voraus, dass die rot markierten Eingabefelder (sog. Pflichtfelder) ordnungsgemäß gefüllt wurden (siehe Abbildung 7).

4.2 Datensatz hinzufügen

In Abbildung 8 ist die Erfassungsmaske für den Vorgang ‚*Datensatz hinzufügen*' dargestellt.

Abbildung 8: Mit CSS gestyltes HTML-Formular für den Vorgang 'Datensatz hinzufügen'

Nach erfolgreichem Abschluss dieses Vorgangs wird der neu hinzugefügte Datensatz sofort angezeigt (siehe Abbildung 9, letzte Tabellenzeile):

Datensätze anzeigen

Alle anzeigen | Seitenweise anzeigen

ID	Vorname	Nachname		
1	Bob	Baker	ändern	löschen
2	Tim	Thomas	ändern	löschen
3	Jim	Connor	ändern	löschen
4	Daniel	Greyson	ändern	löschen
5	Sam	Smith	ändern	löschen
6	Hans-Werner	Hansen	ändern	löschen
7	Siegfried	Sauerland	ändern	löschen
8	Volker	Thormählen	ändern	löschen

Datensatz hinzufügen
Vorgang abbrechen

Abbildung 9: Alle Datensätze anzeigen nach dem Vorgang 'Datensatz hinzufügen'

4.3 Quellcode des Skripts *records.php*

Gegenüber dem Original (siehe [1]) ist das Skript *records.php* u.a. wie folgt geändert bzw. ergänzt worden:

- Das ergänzte Skript enthält die Funktion (*clean_data*) zur Bereinigung der manuellen Eingaben des Benutzers.
- Strukturierung des Codes durch Einrücken der Zeilen.
- Ausführliche Kommentierung des PHP-Codes.

Trotz Pseudocode (siehe Abbildung 6), Strukturierung und Kommentierung wird der Code des Skripts 'records.php' möglicherweise nicht bis ins Detail verständlich sein. Offene Fragen können durch eine Recherche im Internet beantwortet werden können.

```
<?php
/*  Datei:    records.php
    Aufgabe: Neuen Datensatz erstellen bzw. vorhandenen Datensatz bearbeiten.
*/

// Verbindung zur Datenbank herstellen.
include("connect-db.php");

// Funktion zur Bereinigung der Eingabedaten des Benutzers.
function clean_data($data) {
       // Entfernt alle Leerzeichen am Anfang und Ende einer Zeichenfolge.
   $data = trim($data);
       // Entfernt umgekehrte Schrägstriche aus einer Zeichenfolge.
   $data = stripslashes($data);
       // Wandelt Sonderzeichen in HTML-Codes um.
   $data = htmlspecialchars($data);
       // Entfernt Anweisungen in spitzen Klammern (sog. HTML-Tags) aus einer Zeichenfolge.
   $data = strip_tags($data);
   return $data;
}
```

Skript 8: Neuen Datensatz erstellen bzw. vorhandenen bearbeiten (records.php), Teil 1 von 4

```
// HTML-Formular zum Hinzufügen und Ändern von Datensätzen
function renderForm($first = '', $last ='', $error = '', $id = '') {
?>
   <!DOCTYPE html>
   <html lang = "de">
      <head>
         <meta charset="utf-8" />
         <title>
            <?php
               if ($id != '') {
                  echo "Datensatz &auml;ndern";
               } else {
                  echo "Datensatz hinzuf&uuml;gen";
               }
            ?>
         </title>
         <meta name="viewport" content="width=device-width, initial-scale=1.0" />
         <link rel="stylesheet" type="text/css" href="styles.css" media="all" />
      </head>

      <body>
         <h1>
            <?php
               if ($id != '') {
                  echo "Datensatz &auml;ndern";
               } else {
                  echo "Datensatz hinzuf&uuml;gen";
               }
            ?>
         </h1>
         <?php
            if ($error != '') {
               echo "<div style='padding:4px; border:1px solid red; color:red'>"
                  . $error . "</div>";
            }
         ?>
         <form action="" method="post">
            <div>
               <?php if ($id != '') { ?>
                  <input type="hidden" name="id" value="<?php echo $id; ?>" />
                  <p>ID: <?php echo $id; ?></p>
               <?php } ?>
            </div>
            <div>
               <label for="firstname"><strong>Vorname:</strong>
                  <span style="color:red">* </span></label>
               <input type="text" id="firstname" name="firstname" size="30"
                  value="<?php echo $first; ?>" required="required" />
            </div>
            <br>
            <div>
               <label for="lastname"><strong>Nachname:</strong>
                  <span style="color:red">*</span></label>
               <input type="text" id="lastname" name="lastname" size="30"
                  value="<?php echo $last; ?>" required="required"/>
            </div>

               <p><span style="color:red">*</span> erforderlich</p>
               <input type="submit" name="submit" class="button" value="Abschicken" />

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

Skript 9: Neuen Datensatz erstellen bzw. vorhandenen bearbeiten (records.php), Teil 2 von 4

```php
<?php }
/* DATENSATZ ÄNDERN: */
if (isset($_GET['id'])) {
    /* Wenn im Formular die Schaltfläche 'submit' gedrückt wird,
       werden die Felder des Formulars ausgewertet. */
    if (isset($_POST['submit'])) {
        // Gültigkeit der Variablen 'id' prüfen.
        if (is_numeric($_POST['id'])) {
            // Die Werte der Felder des Formulars auslesen.
            $id = $_POST['id'];
            $firstname = htmlentities($_POST['firstname'], ENT_QUOTES);
            $lastname =  htmlentities($_POST['lastname'],  ENT_QUOTES);
            // Prüfen, ob die Pflichtfelder für Vor- und Nachname gefüllt sind.
            if ($firstname == '' || $lastname == '') {
                // Wenn beide leer sind, eine Fehlermeldung ausgeben und das Formular anzeigen.
                $error = 'FEHLER: Bitte Vor- und Nachnamen erfassen!';
                renderForm($firstname, $lastname, $error, $id);
            } else {
                // Wenn die Daten im Formular OK sind, die Datenbanktabelle 'players' aktualisieren.
                $sql = "UPDATE players SET firstname = ?, lastname = ? WHERE id=?";
                if ($stmt = $conn->prepare($sql)) {
                    $stmt->bind_param("ssi", $firstname, $lastname, $id);
                    $stmt->execute();
                    $stmt->close();
                } else {
                    echo "FEHLER: Die UPDATE-Anweisung konnte nicht vorbereitet werden.";
                }
            }
            // Den Benutzer nach der Aktualisierung umleiten.
            header("Location: view.php");
        } else {
            // Wenn der Inhalt der Variable 'id' ungültig ist, eine Fehlermeldung ausgeben.
            echo "FEHLER: Die Variable 'id' ist nicht nummerisch!";
        }
        /* Wenn im Formular die Schaltfläche 'submit'  noch nicht betätigt wurde,
           die Daten aus der Datenbanktabelle 'players' entnehmen und im Formular anzeigen: */
    } else {
        // Prüfen, ob der Wert der Variablen 'id' nummerisch und größer als null ist.
        if (is_numeric($_GET['id']) && $_GET['id'] > 0) {
            $id = $_GET['id'];
            // Bestimmten Datensatz aus der Datenbanktabelle 'players' auswählen und anzeigen.
            $sql = "SELECT * FROM players WHERE id=?";
            if($stmt = $conn->prepare($sql)) {
                $stmt->bind_param("i", $id);
                $stmt->execute();
                $stmt->bind_result($id, $firstname, $lastname);
                $stmt->fetch();
                // HTML-Formular anzeigen
                renderForm($firstname, $lastname, NULL, $id);
                $stmt->close();
            } else {
                echo "FEHLER: Die SELECT-Anweisung konnte nicht vorbereitet werden.";
            }
        } else {
            /* Wenn der Wert der Variablen 'id' ungültig ist,
               den Benutzer zur Seite view.php umleiten. */
            header("Location: view.php");
        }
    }
} else {
```

Skript 10: Neuen Datensatz erstellen bzw. vorhandenen bearbeiten (records.php), Teil 3 von 4

```
    /* Die Variable 'id' ist nicht gesetzt, folglich:
    NEUER DATENSATZ:  */
    if (isset($_POST['submit'])) {
       /* Wenn im Formular die Schaltfläche 'submit' gedrückt wird,
          muss das Formular ausgewertet werden. */
       // Die Eingabedaten mit der Funktion 'clean_data' bereinigen.
       $firstname = clean_data($firstname);
       $lastname  = clean_data($lastname);
       // Eingabedaten aufbereiten.
       $firstname = htmlentities($_POST['firstname'], ENT_QUOTES);
       $lastname  = htmlentities($_POST['lastname'],  ENT_QUOTES);
       /* Prüfen, ob im Formular die Pflichtfelder für Vor- und Nachname
          gefüllt sind, sonst eine Fehlermeldung ausgeben
          und das HTML-Formular anzeigen. */
       if ($firstname == '' || $lastname == '') {
          $error = 'FEHLER: Bitte alle Pflichtfelder ausf&uuml;llen!';
          renderForm($firstname, $lastname, $error);
       } else {
          /* Einen neuen Datensatz in die Datenbanktabelle 'players' einfügen,
             oder eine Fehlermeldung ausgeben.*/
          $sql = "INSERT INTO players (firstname, lastname) VALUES (?, ?)";
          if ($stmt = $conn->prepare($sql)) {
             $stmt->bind_param("ss", $firstname, $lastname);
             $stmt->execute();
             $stmt->close();
             echo  "Einen neuen Datensatz erfolgreich erstellt.";
          } else {
             echo "FEHLER: Die INSERT-Anweisung konnte nicht vorbereitet werden.";
          }
          // Den Benutzer umleiten.
          header("Location: view.php");
       }
    } else {
       // Wenn das HTML-Formular noch NICHT abgesandt wurde, dieses anzeigen.
       renderForm();
    }
}
// Die Verbindung zur Datenbank beenden.
mysqli_close($conn);
?>
```

Skript 11: Neuen Datensatz erstellen bzw. vorhandenen bearbeiten (records.php), Teil 4 von 4

Das Skript 'records.php' wurde in Anlehnung an das gleichnamige Skript in [1] erstellt. Hinzugefügt wurde u. a. das HTML5-Attribut 'required', das die <input>-Tags für die Felder Vorname (*firstname)* und Nachname (*lastname)* zu Pflichtfeldern deklariert (siehe Skript 9, Seite 16). Deshalb kann später im zugehörigen Skript auf eine entsprechende Plausibilitätsprüfung mit der PHP-Funktion **empty()**[5] oder sonst wie verzichtet werden.

[5] Die PHP-Funktion **empty()** liefert TRUE zurück, wenn eine Variable nicht definiert ist oder einen Leerstring oder den Wert Null enthält.

```
/* styles.css */
form > div { margin-bottom: 1em; }
form {
   border: 1px solid #000;
   padding: 5px;
   background-color: grey;
   width: 400px;
   margin-bottom: 2em;
}
input[type="text"] {
   width: 280px;
   padding: 0.25em;
   margin-bottom: 0.25em;
}
.button {
    background-color: #4CAF50;
    border: none;
    color: white;
    padding: 15px 32px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 16px;
    margin: 4px 2px;
    cursor: pointer;
}
```

Listing 2: Mehrstufige Formatvorlage (*styles.css*) für das Skript records.php

4.4 Bestehenden Datensatz löschen

Das Skript zum Löschen eines bestimmten Datensatzes (*delete.php*) wurde neu geschrieben, weil im Original [1] eine Sicherheitsabfrage vor Ausführung des Lösch-Vorgangs fehlt. Diese wurde vom Autor realisiert mit:

- vorbereiteten SQL-Anweisungen und
- einer formatierten Zeichenkette (siehe gelbe Hervorhebung in Skript 12 auf Seite 20).

Die Sicherheitsabfrage (siehe Abbildung 10) im neu gefassten Skript wird am Bildschirm wie folgt dargestellt:

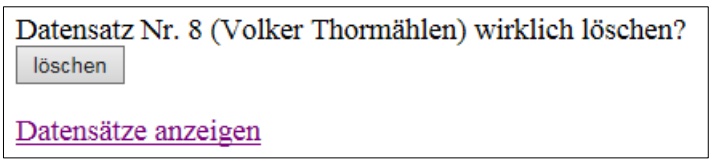

Abbildung 10: Sicherheitsabfrage vor dem Löschen des Datensatzes mit der Nr. 8

4.5 Quellcode des Skripts delete.php

```
<?php
/* Datei:   delete.php
   Aufgabe: Bestehenden Datensatz löschen
*/
   include_once('connect-db.php');
?>
<html>
   <head>
      <meta charset="UTF-8">
      <title>Datensatz löschen</title>
   </head>
<body>
   <?php
   if (!isset($_GET['id'])) {
      header('Location: view.php');
   } elseif (!isset($_POST['ok'])) {
      $id = (int)$_GET['id'];
      // SQL-Anweisung definieren.
      $sql = "SELECT id, firstname, lastname FROM players WHERE id=?";
      // SQL-Anweisung vorbereiten.
      if($stmt = $conn->prepare($sql)) {
         // Felder an die vorbereitete Anweisung binden.
         $stmt->bind_param('i', $id);
         // Vorbereitete Anweisung ausführen.
         $stmt->execute();
         // Variablen an Felder binden.
         $stmt->bind_result($id, $firstname, $lastname);
         // Ergebnis abgeholen.
         $stmt->fetch();
         // Zeichenkette definieren.
         $format = 'Datensatz Nr. %d (%s %s) wirklich löschen?<br>'
                 . '<form method="post" action="#">'
                 . '<input type="hidden" name="ok" value="true" />'
                 . '<input type="submit" value="löschen" />'
                 . '</form>';
         // Zeichenkette formatiert ausgeben.
         echo sprintf($format, $id, $firstname, $lastname);
      } else {
         echo 'Fehler: ' . $stmt->error;
      }
```

Skript 12: Bestehenden Datensatz nach Sicherheitsabfrage löschen (delete.php), Teil 1 von 2

```
    } else {
        $id = (int)$_GET['id'];
        // SQL-Anweisung definieren.
        $sql = "DELETE FROM players WHERE id = ? LIMIT 1";
        // SQL-Anweisung vorbereiten.
        if($stmt = $conn->prepare($sql)) {
            // Feld an die vorbereitete Anweisung binden.
            $stmt->bind_param('i', $id);
            // Vorbereitete Anweisung ausführen.
            $stmt->execute();
            // Vorbereitete Anweisung schliessen.
            $stmt->close();
            // Zuvor geöffnete Datenbankverbindung schliessen.
            $conn->close();
            // Erfolgsmeldung am Bildschirm ausgeben.
            echo 'Datensatz gelöscht!';
        } else {
            echo 'Fehler: ' . $stmt->error;
        }
    }
    ?>
    <p>
        <a href="view.php">Datensätze anzeigen</a>
    </p>
</body>
</html>
```

Skript 13: Bestehenden Datensatz nach Sicherheitsabfrage löschen (delete.php), Teil 2 von 2

Nach dem Löschen eines Datensatzes wird zum Vorgang *'Datensätze anzeigen'* verzweigt.

5 Pflege größerer Datenbanktabellen

Auch Datenbanktabellen mit mehr als 3 Spalten lassen sich mit den gezeigten PHP-Skripten bequem pflegen. Sie müssen aber entsprechend angepasst werden.

Literaturverzeichnis

[1] falkencreative, „Basic PHP System: View, Edit, Add, Delete records with MySQLi,“ 9 5 2010. [Online]. Available: https://www.killersites.com/community/index.php?/topic/3064-basic-php-system-view-edit-add-delete-records-with-mysqli/&. [Zugriff am 13 3 2018].

[2] T. Theis, Einstieg in PHP 5.6 und MySQL 5.6, Bonn: Rheinwerk-Verlag, 2014/2015, p. 601.

[3] J. Brandes, „OOP MySQLi - Prepared Statements,“ 17 05 2017. [Online]. Available: http://www.pcsystembetreuer.de/ii-php-a-mysql/oop-mysqli-prepared-statements.html. [Zugriff am 05 04 2018].

[4] nurhodelta_17, „Easy and Simple Add, Edit, Delete MySQL Table Rows using PHP/MySQLi,“ sourcecodester.com, 21 08 2017. [Online]. Available: https://www.sourcecodester.com/php/11546/easy-and-simple-add-edit-delete-mysql-table-rows-using-phpmysqli.html. [Zugriff am 05 04 2018].

[5] P. Khodke, „Multiple Insert, Update, Delete example usinf OHP & MySQLi,“ 06 02 2015. [Online]. Available: http://www.codingcage.com/2015/06/multiple-insert-update-delete-crud.html. [Zugriff am 05 04 2018].

[6] o. V., „INSERT, DELETE, EDIT, UPDATE OPERATION USING PHP AND MY SQL,“ 21 03 2014. [Online]. Available: http://www.sanwebcorner.com/2014/03/insert-delete-edit-update-operation.html. [Zugriff am 05 04 2018].