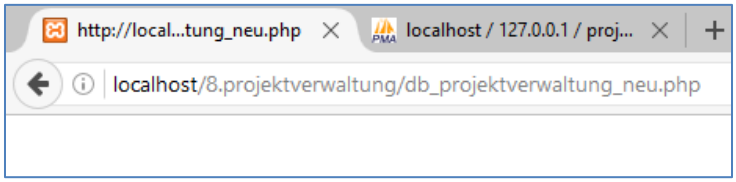
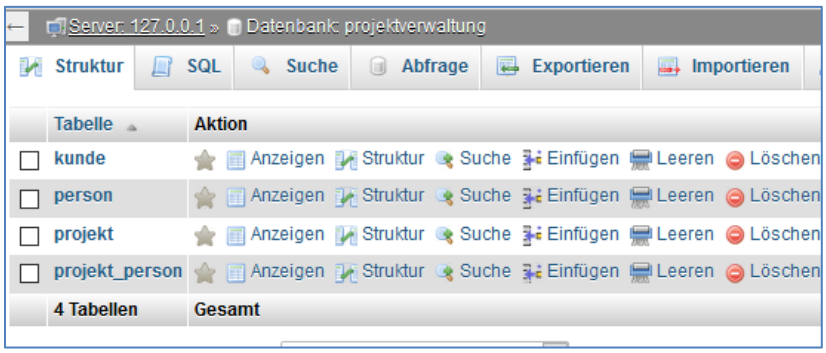
Übung 1: Projektverwaltung – fertige Datenbank kopieren

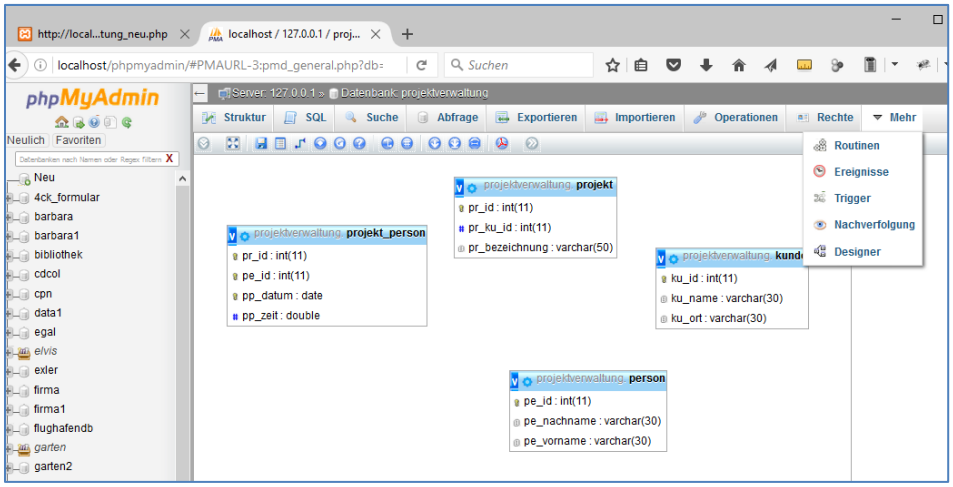
Kopiere die Datei „db_projektverwaltung_neu.php“ in das Verzeichnis von XAMPP und zwar in den Ordner „projektverwaltung“. Dann öffne den Browser und öffne bei laufendem XAMPP (Apache und MySQL) diese Datei:



Wenn keine Fehlermeldung kommt, dann wurde eine neue Datenbank mit vier Tabellen angelegt:



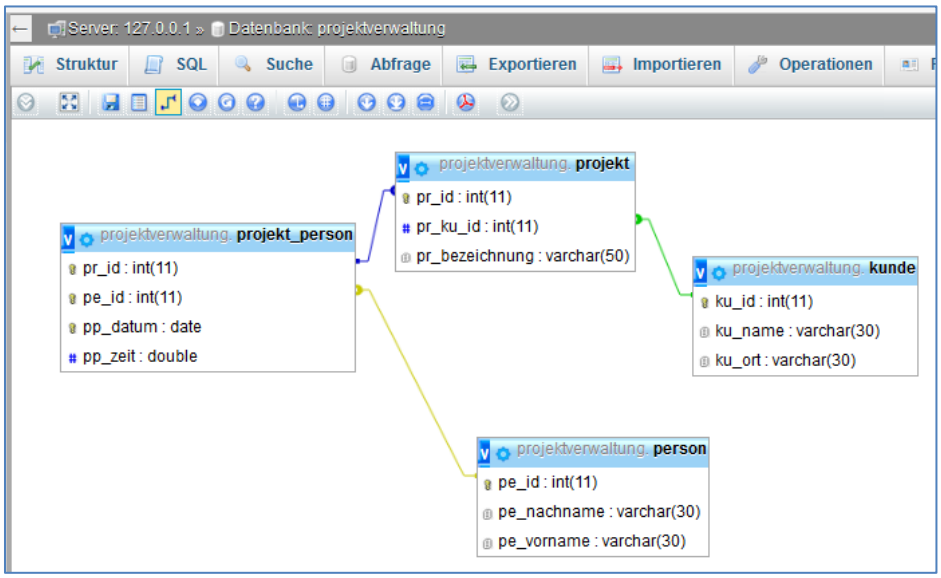
1. **Aufgabe: Zuerst im „Designer“ die Beziehungen herstellen:**



- Klicke rechts auf „mehr“ und dann „Designer“.
- Wenn keine Tabellen angezeigt werden klicke auf das Symbol „Tabellen-Liste anzeigen/ausbelnden“
- Verschiebe die Tabellen, so dass ein ähnliches Bild erscheint wie hier abgebildet
- Speichere die aktuelle Anzeige mit Hilfe des Symbols „Seite speichern“ bzw. „Position speichern“ - Diskettensymbol
- Verknüpfungen erzeugen:
    - Wähle das Symbol „Erzeuge Verknüpfung“
    - Wähle den referenzierten Schlüssel aus, z.B. das Feld „ku_id“ in der Zabelle „kunde“

- Wähle den dazugehörigen Fremdschlüssel aus, in diesem Fall das Feld „pr_ku_id“ in der Tabelle „projekt“
- Bestätige mit „OK“ das Erzeugen der Verknüpfung.

Ergebnis:



Speicher mit Hilfe des Symbols „Speichere Position“

2. Abfragen durchführen

2a) Anzahl der Kunden

- mithilfe der SQL-Funktion COUNT()
- das Ergebnisfeld, das die berechnete Anzahl beinhaltet, bekommt den frei gewählten Namen „count_ku_id“ für eine bessere Ausgabe.

  Ergebnis:

Anzahl der Kunden:
3

Hilfe für die Lösung:

```
10  echo "<br><b>Anzahl der Kunden:</b><br>";
11  $res = mysqli_query($con, "SELECT COUNT(ku_id) as count_ku_id FROM kunde");
12  while ($dsatz = mysqli_fetch_assoc($res))
13     echo $dsatz["count_ku_id"] . "<br>";
14
```

2b) Alle Kunden mit allen Projekten (zwei Tabellen)

- Es wird für jedes Projekt ein Datensatz ausgegeben
- In jedem Datensatz stehen die Daten des Projekts und des betreffenden Kunden
- Die Anzeige ist nach Namen, Ort und Bezeichnung sortiert

Alle Kunden mit allen Projekten:
Murchel, Dortmund, Nordbahnhof
Murchel, Dortmund, Westbahnhof
Schmidt, Hamburg, Alexanderstrasse
Schmidt, Hamburg, Peterstrasse
Weber, Frankfurt, Jahnplatz
Weber, Frankfurt, Lindenplatz

Info:
In jedem Datensatz werden Inhalte aus zwei Tabellen angezeigt. Beide Tabellennamen werden hinter FROM aufgeführt. Es werden nur die Datensätze zusammengestellt, bei denen die Feldinhalte aus der Bedingung nach WHERE übereinstimmen.

2c) Alle Personen mit allen Projektzeiten – drei Tabellen

- Es wird für jede eingetragene Arbeitszeit ein Datensatz ausgegeben
- In jedem Datensatz stehen die Daten der Arbeitszeit, des betreffenden Projekts und des betreffenden Kunden
- Die Ausgabe ist nach Nachnamen, Bezeichnung und Datum sortiert.

Alle Personen mit allen Projektzeiten:
Berger, Lindenplatz, 2015-12-01
Berger, Lindenplatz, 2015-12-02
Mohr, Alexanderstrasse, 2015-12-01
Mohr, Lindenplatz, 2015-12-01
Suhren, Alexanderstrasse, 2015-12-01
Suhren, Lindenplatz, 2015-12-01

Info:
In jedem Datensatz werden Inhalte aus drei Tabellen angezeigt. Alle drei Tabellennamen werden hinter FROM aufgeführt

```
21    echo "<br><b>Alle Personen mit allen Projektzeiten:</b><br>";
22    $res = mysqli_query($con, "SELECT * FROM projekt, projekt_person, person "
```

Es werden nur Datensätze zusammengestellt, bei denen die Feldinhalte aus den beiden Bedingungen nach WHERE übereinstimmen.

Die beiden Feldnamen „pr_id“ und „pe_id“ kommen in zwei Tabellen vor. Daher muss man jeweils den Tabellenamen, mit nachfolgendem Punkt, zusätzlich angeben. Ansonsten wären die Feldnamen in der SQL-Anweisung nicht eindeutig.

2d) Alle Personen mit Zeitsummen

- Es wird für jede Person ein Datensatz ausgegeben.
- Es werden alle Personen, denen mindestens eine Arbeitszeit zugeordnet ist, ausgegeben.
- Es wird die Summe der Arbeitszeiten pro Person mithilfe der SQL-Funktion „SUM()“ berechnet.
- Die Ausgabe ist nach Namen sortiert.

Alle Personen mit Zeitsumme:
Berger, 13.8
Mohr, 6.5
Suhren, 8

Lösung:

```
30    echo "<br><b>Alle Personen mit Zeitsumme:</b><br>";
31    $res = mysqli_query($con, "SELECT pe_nachname, SUM(pp_zeit) as sum_pp_zeit "
32      . "FROM person, projekt_person "
33      . "WHERE person.pe_id = projekt_person.pe_id "
34      . "GROUP BY person.pe_id, pe_nachname "
35      . "ORDER BY pe_nachname");
36    while ($dsatz = mysqli_fetch_assoc($res))
37       echo $dsatz["pe_nachname"] . ", " . $dsatz["sum_pp_zeit"] . "<br>";
```

Info:

- Die Anweisung „SUM … AS“ bewirkt, dass die SQL-Funktion „SUM()“ angewendet wird.
- Es werden alle Einträge im Feld „pp_zeit“ aufsummiert, nach denen gruppiert wird. Die Gruppierung wird mithilfe von GROUP BY durchgeführt.
- Es wird nach den Feldern „pe_id“ und „pe_nachname“ der Tabelle „person“ gruppiert, es werden also Arbeitszeiten einer Person summiert. Streng genommen hätte es gereicht, nach „pe_id“ zu gruppieren, da dadurch bereits alle Personen voneinander unterschieden werden. Allerdings soll das Feld „pe_nachname“ ausgegeben werden, daher muss es ebenfalls Teil der Gruppierungsfunktion sein.
- Das Ergebnisfeld, das die berechnete Summe beinhaltet, bekommt den frei gewählten Namen „sum_pp_zeit“.

2e) Alle Projekte mit allen Personenzeiten

- Es handelt sich um den gleichen Zusammenhang wie in der Abfrage „alle Personen mit allen Projektzeiten“.
- Die Ausgabe ist nur anders sortiert – nach Bezeichnung, dem Nachnamen und dem Datum.

Alle Projekte mit allen Personenzeiten:
Alexanderstrasse, Mohr, 2015-12-01
Alexanderstrasse, Suhren, 2015-12-01
Lindenplatz, Berger, 2015-12-01
Lindenplatz, Berger, 2015-12-02
Lindenplatz, Mohr, 2015-12-01
Lindenplatz, Suhren, 2015-12-01

2f) Alle Projekte mit Zeitsumme

- Es handelt sich um einen ähnlichen Zusammenhang wie in der Abfrage „Alle Personen mit Zeitsumme“
- Es wird nach Projekt statt nach Person gruppiert und entsprechend sortiert.

Alle Projekte mit Zeitsumme:
Alexanderstrasse, 7.5
Lindenplatz, 20.8

Lösung:

```
48      echo "<br><b>Alle Projekte mit Zeitsumme:</b><br>";
49      $res = mysqli_query($con, "SELECT pr_bezeichnung, SUM(pp_zeit) as sum_pp_zeit "
50        . "FROM projekt, projekt_person "
51        . "WHERE projekt.pr_id = projekt_person.pr_id "
52        . "GROUP BY projekt.pr_id, pr_bezeichnung "
53        . "ORDER BY pr_bezeichnung");
54      while ($dsatz = mysqli_fetch_assoc($res))
55         echo $dsatz["pr_bezeichnung"] . ", " . $dsatz["sum_pp_zeit"] . "<br>";
56
57      mysqli_close($con);
58  ?>
59  </body></html>
```

Quelle:

Thomas Theis: in Einstieg in PHP 7 und MySQL 5.6; Rheinwerk Verlag, 2017, S. 231-238

Beispiel 2: Literaturdatenbank – vom Anlegen bis zum Abfragen

Die Mini-Datenbank soll Autoren, deren Bücher und das Erscheinungsjahr eines Buches enthalten. Die Tabelle wirkt auf den ersten Blick vielleicht ganz in Ordnung, doch sobald Sie ein wenig darüber nachdenken, werden Sie rasch auf Probleme stoßen.

Wie Sie sehen, taucht der Name »Karl May« zweimal in der Tabelle auf. Das mag in diesem Fall noch nicht problematisch scheinen, doch wenn die Tabelle mehr Einträge enthält, wird es immer häufiger zu derartigen Mehrfachnennungen oder Redundanzen kommen.

id	Name	Vorname	Titel	Jahr
1	May	Karl	Winnetou I	1897
2	Schmidt	Arno	Leviathan	1949
3	Raabe	Wilhelm	Abu Telfan	1897
4	May	Karl	Der Schut	1892
5	Wolf	Ror	Pilzer und Pelzer	1967

Normalisieren:
Um diese ineffiziente und fehleranfällige Situation zu umgehen, werden die Informationen, die bislang in einer Tabelle verwaltet werden, auf zwei Tabellen aufgeteilt. In der Tabelle **autoren** werden die Autoren, in **werke** die Titel gespeichert. Damit man die Informationen der beiden Tabellen verknüpfen kann, besitzt jeder Datensatz der Tabelle autoren einen eindeutigen Index, etwa autor_id. In der Tabelle werke wird dann bei jedem Titel statt des Autorennamens der entsprechende Index eingetragen.

Damit ist das Problem gelöst. Bei einer nach Autor sortierten Ausgabe wird nun zuerst der Autor aus der Tabelle autoren ausgelesen, und anschließend werden aus werke alle Titel, die als Autor die passende autor_id aufweisen, angefordert. Umgekehrt wird bei einer Ausgabe nach Titeln zuerst der Titel aus werke ausgegeben und anschließend über die autor_id der dazugehörige Autor aus autoren ermittelt.

Tabelle: autoren

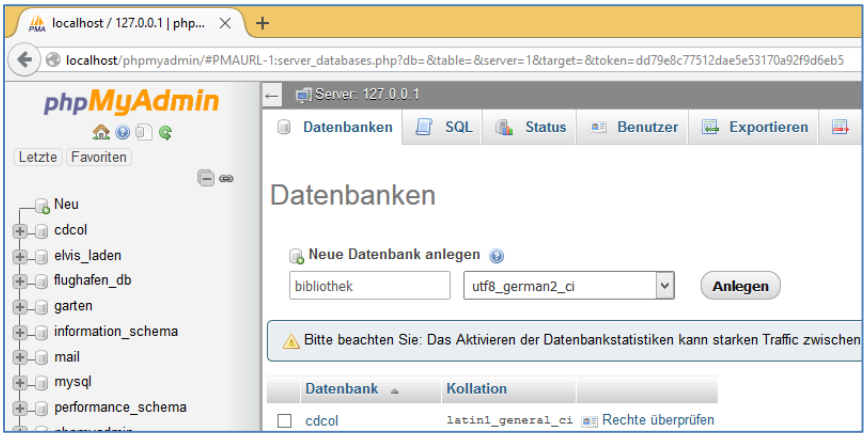
autor_id	name	vorname
1	Schmidt	Arno
2	May	Karl
3	Raabe	Wilhelm
4	Wolf	Ror

Tabelle: werke

werk_id	autor_id	werk	jahr
1	1	Leviathan	1949
2	2	Winnetou I	1893
3	3	Abu Telfan	1867
4	2	Der Schut	1892
5	4	Pilzer und Pelzer	1967

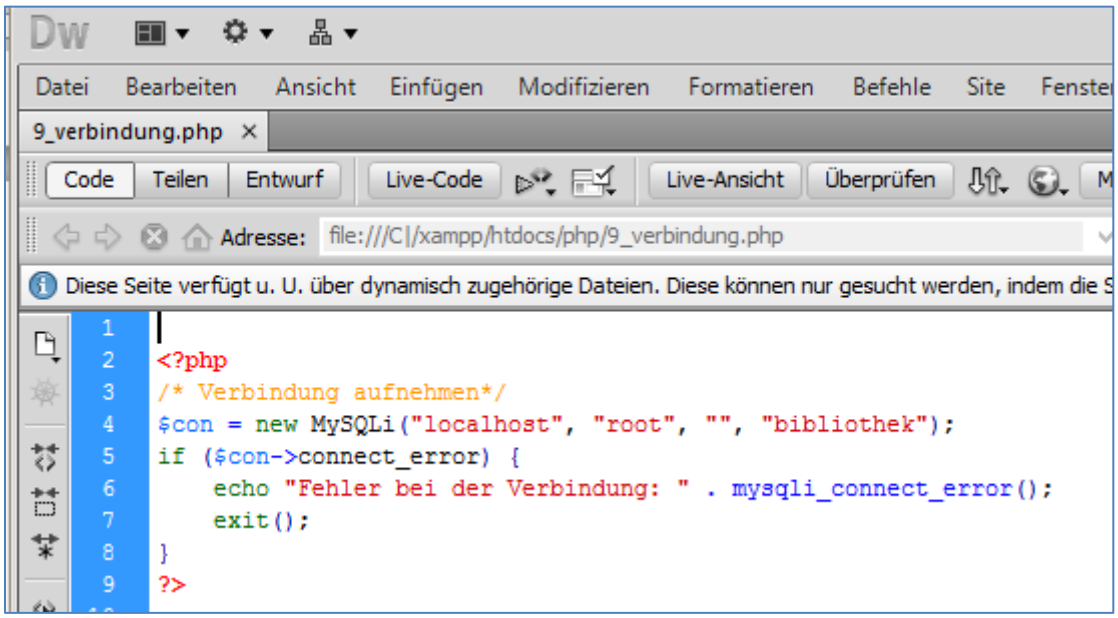
0.) Xampp starten und dann http://localhost/phpmyadmin

1.) Datenbank mit phpMySQL erstellen: bibliothek



2.)Verbindung aufnehmen mit dieser Datenbank

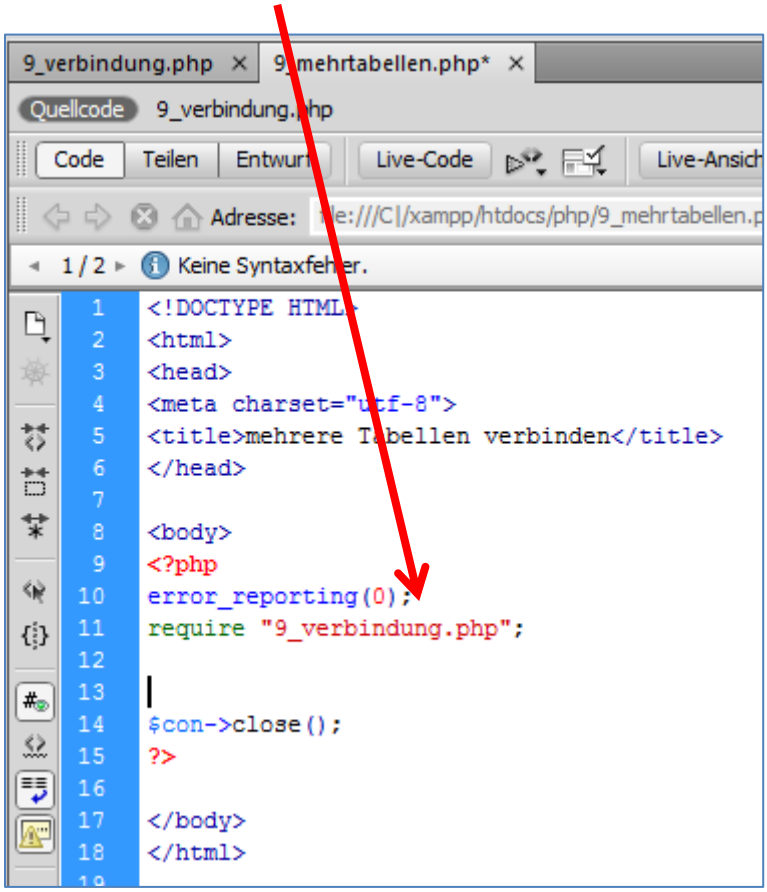
Erstelle die die „verbindung.php“



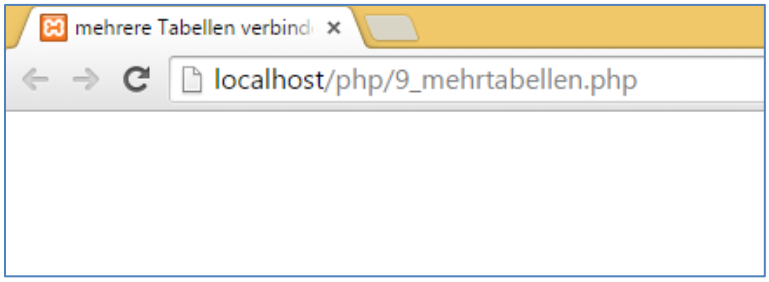
Code:

```
<?php
/* Verbindung aufnehmen*/
$con = new MySQLi("localhost", "root", "", "bibliothek");
if ($con->connect_error) {
        echo "Fehler bei der Verbindung: " . mysqli_connect_error();
        exit();
}
?>
```

Dann erstelle die PHP-Site „9_mehrtabellen.php“, die den Code beinhalten wird und auf die „9_verbindung.php“ zugreift. Dies muss auch eine PHP-Site sein! Keine HTML!



Test: wenn keine Fehlermeldung, dann OK:



3.)Tabellen per PHP anlegen

Ein Autor schreibt mehrere Werke, aber ein Werk ist immer genau von einem Autor.
Beziehung daher 1:n

Daher muss man die Einser-Beziehung in die N-Beziehung „einbauen“.

Theorie: Die Struktur von der Tabelle autoren:

- **autor_id:** Die eindeutige Kennung wird zwingend für die Verknüpfung benötigt. Sie ist vom Typ INT, hat die Eigenschaften UNSIGNED und AUTO_INCREMENT. Damit ist sie automatisch der Primärschlüssel der Tabelle (PRIMARY KEY).
- **name, vorname:** Hier werden zwei Felder vom Typ VARCHAR mit maximal 255 Zeichen benötigt.

Die Struktur von der Tabelle werke:

- **werk_id:** Eine eindeutige Kennzeichnung der Datensätze wird streng genommen für diese Tabelle nicht benötigt, aber es ist eine gute Idee, einen automatisch erzeugten Index mitlaufen zu lassen. Auch hier ist das Feld vom Typ INT, UNSIGNED, besitzt das Attribut AUTO_INCREMENT und ist der Primärschlüssel der Tabelle.
- **autor_id:** Hier wird die entsprechende autor_id aus autoren gespeichert, auch dieses Feld ist also vom Typ INT und UNSIGNED.
- **werk:** Dieses Feld soll den Werktitel enthalten. Wenn man auf Nummer sicher gehen will und auch sehr lange Titel erfassen möchten, dann sollte es vom Typ TEXT sein, in der Regel wird hier aber ein VARCHAR mit 255 Zeichen genügen.
- **jahr:** Hier scheint sich der Typ YEAR anzubieten – doch Vorsicht! Felder mit diesem Typ können keine Jahreszahl vor 1901 speichern. Statt YEAR wählen Sie hier also SMALLINT mit 4 Stellen und UNSIGNED.

Nun kann man die benötigten Tabellen samt Beispielinhalt mit diesem PHP-Skript anlegen:

Beachte:

- die SQL-Befehle schreibt man in Großbuchstaben

Das Skript legt zwei Tabellen an:

9_verbindung.php | 9_mehrtabellen.php*

Quellcode 9_verbindung.php

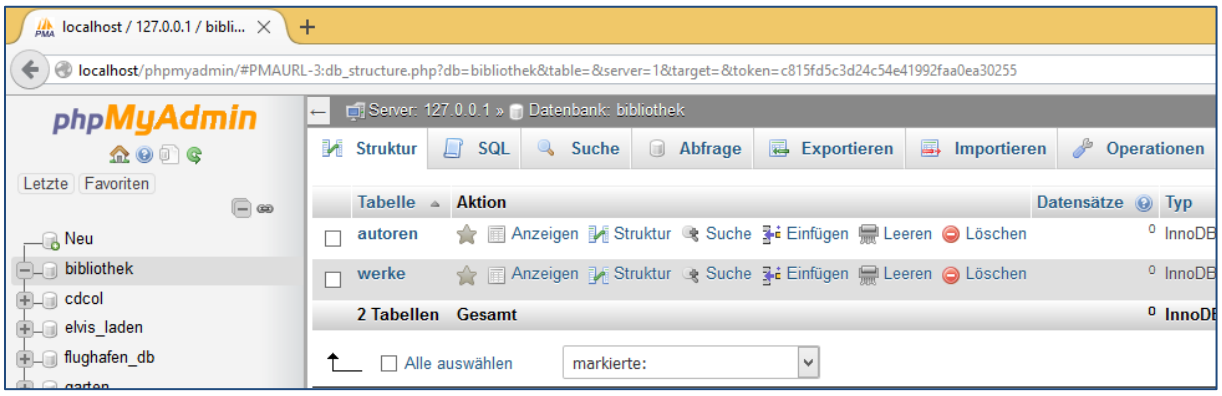
Code | Teilen | Entwurf | Live-Ansicht | Titel: mehr

Diese Seite verfügt u. U. über dynamisch zugehörige Dateien. Diese können nur gesucl

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>mehrere Tabellen verbinden</title>
</head>

<body>
<?php
error_reporting(0);
require "9_verbindung.php";

$con->query("CREATE TABLE autoren (
  autor_id INT UNSIGNED AUTO_INCREMENT,
  name VARCHAR (255),
  vornamer VARCHAR (255),
  PRIMARY KEY (autor_id)
)");

$con->query("CREATE TABLE werke (
  werk_id INT UNSIGNED AUTO_INCREMENT,
  autor_id INT UNSIGNED,
  werk VARCHAR (255),
  jahr SMALLINT (4) UNSIGNED,
  PRIMARY KEY (werk_id)
)");

$con->close();
?>

</body>
</html>
```

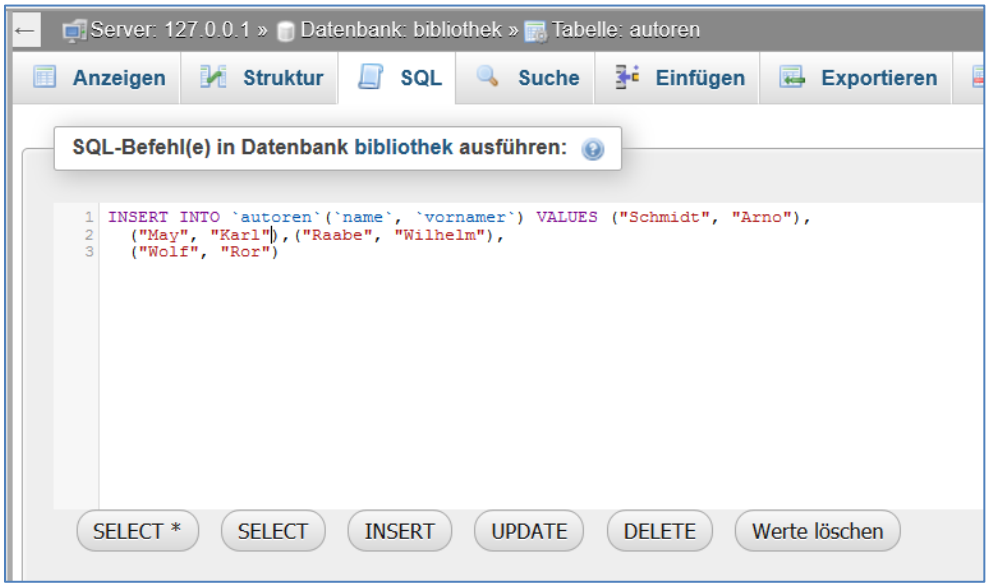
Habe versehentlich statt „vorname“ „vornamer“ geschrieben!?

Ergebnis in phpMyAdmin:

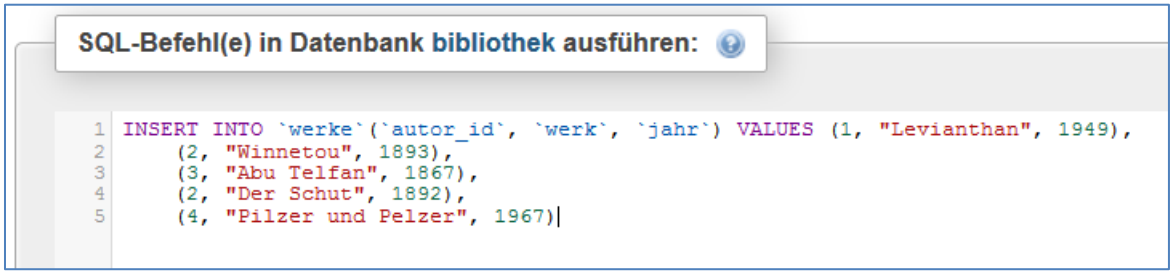


Mit Daten füllen in phpMyAdmin.

Einfach: mit SQL-Befehle INSERT INTO:
Klicke auf den Button „INSERT“, lösche nur den ersten Eintrag (Primärschlüssel) bei den „autoren“ und ersetzte die Begriffe in eckigen Klammern mit den einzufüllenden Namen in doppelten Anführungszeichen:

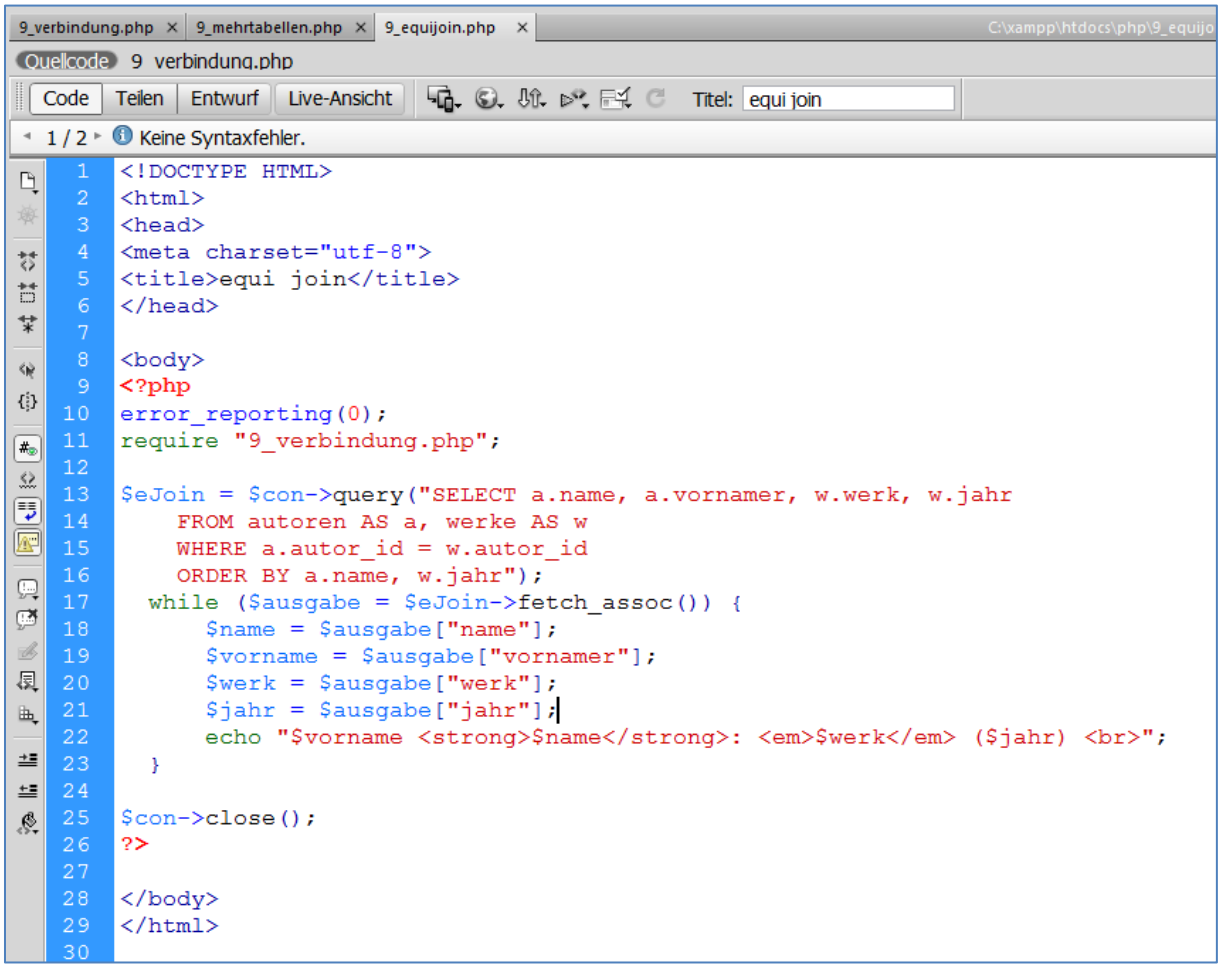


Werke einfügen:
wähle in Datenbank die Tabelle „werke“. Dann verwende wie oben den „INSERT“ Button.



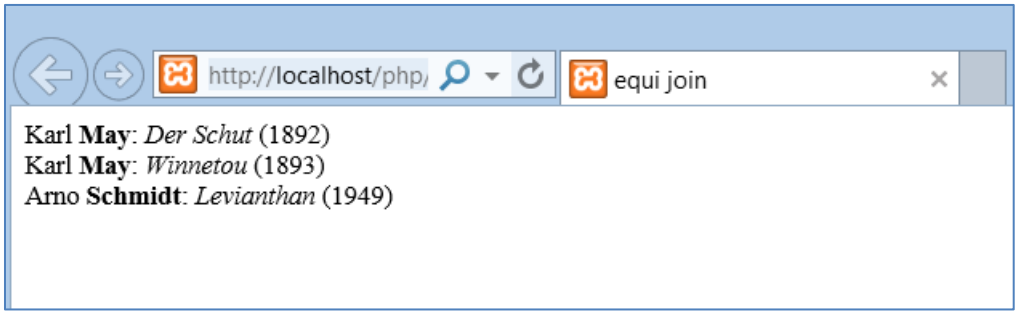
Ergebnis bei Werke:

Erstelle eine neue php-Datei mit folgendem Inhalt:




```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>equi join</title>
</head>

<body>
<?php
error_reporting(0);
require "9_verbindung.php";

$eJoin = $con->query("SELECT a.name, a.vornamer, w.werk, w.jahr
    FROM autoren AS a, werke AS w
    WHERE a.autor_id = w.autor_id
    ORDER BY a.name, w.jahr");
  while ($ausgabe = $eJoin->fetch_assoc()) {
      $name = $ausgabe["name"];
      $vorname = $ausgabe["vornamer"];
      $werk = $ausgabe["werk"];
      $jahr = $ausgabe["jahr"];
      echo "$vorname <strong>$name</strong>: <em>$werk</em> ($jahr) <br>";
  }

$con->close();
?>

</body>
</html>
```

Ergebnis:



Übung 3: Pizza

1)

- Gib alle Namen und Orte der Kunden aus, die in einem Ort wohnen, der mit M oder einem der davorstehenden Buchstaben im Alphabet beginnt.
- Suche nach allen Pizzen, die mehr als 5 aber weniger als 8 kosten.
- Zähle, wie viele Kunden der Pizzaservice zurzeit hat.

2)

- Erstelle eine weitere Tabelle namens bestellungen. Diese soll die Bestellungen verwalten. In der Tabelle sind die kunden_id, die id der bestellten Pizza, die Anzahl und das Datum gespeichert.
- Befülle die Tabelle mit ein paar Beispieldatensätzen.

3)

- Lasse die Anzahl der Bestellungen, das Datum und die Pizzasorte auslesen.
- Es sollen zusätzlich zum ersten Punkt die Namen der Kunden ausgegebene werden. Dafür muss man drei Tabellen verknüpfen.
- Schreibe den letzten Befehl mit einem INNER JOIN.
- Ermittle wie viele Kunden es pro Ort gibt – dafür muss man keine Tabellen verknüpfen.

Quelle:

Giesbert Damaschke in: PHP und MySQL: Der Web-Baukasten für Einsteiger und Individualisten, 2014, Wiley_Vch Verlag GmbH&Co.KGaA

Beispiel Pizza:

Florence Maurice in PHP 5.6 und MySQL 5.7, dpunkt Verlag, 2015, S.315-355