Dynamische Web-Dokumente mit PHP
Teil 3: PHP und MySQL

Dr. Wolfgang Riedel, TU Chemnitz, URZ

Zi. 1/B301b, Tel.: 1422, E-Mail: w.riedel@hrz

Stand: 29. März 2012

Ziel

- Entwicklung von Webseiten mit dynamischem Inhalt
- Inhalt kommt aus einer Datenbank
- Daten sollen auch änderbar sein
- häufigste Lösung: PHP + MySQL

Was ist MySQL?

- relationales Datenbank-Management-System
  d.h. Anlegen, Nutzen und Verwalten von Datenbanken beliebigen Inhalts
- basiert auf SQL-Standard („Structured Query Language“)

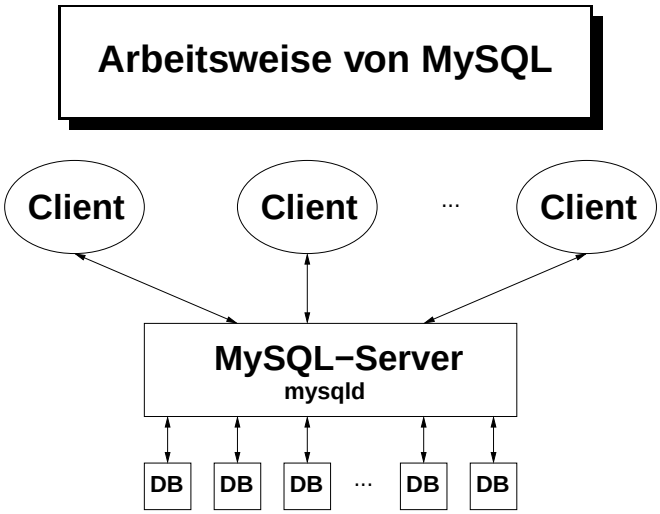
- Client-Server-Architektur
- „Väter“: David Axmark, Allan Larsson, Michael „Monty“ Widenius
- dann: verwaltet und weiterentwickelt durch die Firma *MySQL AB*
- $\Rightarrow$ Sun Microsystems $\Rightarrow$ Oracle
- populärste freie Datenbanksoftware, mehrere Millionen Installationen
- aktuell: Version 5.1

Vorteile von MySQL

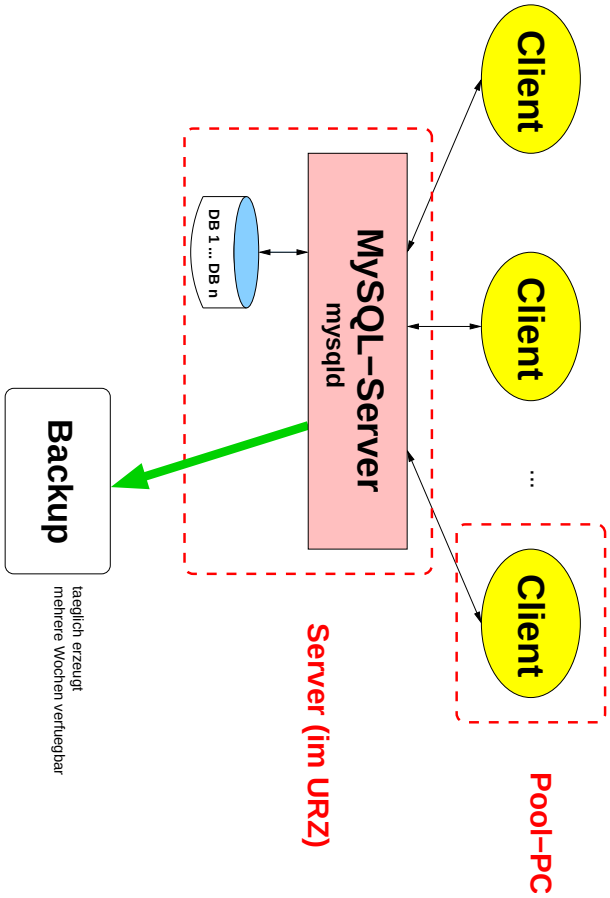
- stabile, sichere Software
- einfach zu bedienende Klienten
- Schnittstellen zu vielen Programmiersprachen, insbesondere Skriptsprachen im Web-Umfeld
- existiert für verschiedenste Plattformen
- duale Lizenz: GPL und kommerziell

Dokumentation, weitere Informationen

- MySQL-Homepage `www.mysql.com` (mit Online-Doku)
- Diverse Bücher (meist Englisch)
- Stepken: „MySQL Datenbankhandbuch“
  `www.little-idiot.de/mysql`
- Tutorium „MySQL, phpMyAdmin und PHP“
  `http://v.hdm-stuttgart.de/~riekert/lehre/php/mysql_php.html`
- RRZN-Handbücher im Nutzerservice:
  - „SQL (Grundlagen)“, „MySQL (Administration)“
  - „PHP (Grundlagen)“, „PHP (Fortgeschrittene Techniken)“
- URZ-Kurs „MySQL“



In der URZ-Infrastruktur (MySQL-Server Version 5.0.77):



kein direkter Zugriff von außerhalb der TU-Domäne!
Ausnahme (indirekt): Webserver

Was ist ein „Client“?

- Kommandozeilenprogramm

```
% mysql        (aus /usr/bin)
```

bzw.

```
C:\Programme\MySQL\MySQL Server n.n\bin\mysql.exe
(oder aus Startmenü – Problem: Parameterübergabe)
```

- „komfortables“ Programm
- Web-Applikation
- . . .

in der praktischen Anwendung: nur Variante 2 + 3
jetzt: Variante 1, um Zusammenhänge besser zeigen zu können

Erster Aufruf der Software (Linux)

```
% mysql
```

Ergebnis:

```
ERROR 2002: Can't connect to local MySQL server through socket
'/var/lib/mysql/mysql.sock' (2)
```

→ auf diesem Rechner läuft kein MySQL-Server (lokal)

```
% mysql -h mysql[.hrz.tu-chemnitz.de]
```

Ergebnis:

```
ERROR 1045: Access denied for user: 'otto@tacco.hrz.tu-chemnitz.de'
(Using password: NO)
```

→ *otto* hat keinen Zugriff, wir müssen einen „geeigneten“ Nutzer verwenden

```
% mysql -h mysql -u db-nutzer -ppasswort
% mysql -h mysql -u db-nutzer -p
```

Ergebnis nun:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 123 to server version: 3.23.56

Type 'help' for help.

mysql>
```

Erste Kommandos:

Liste von - wichtigen - SQL-Kommandos:

```
mysql> help;

MySQL commands:
Note that text commands must be first on line and end with ';'
help    (\h)    Display this help.
?       (\?)    Synonym for `help'.
clear   (\c)    Clear command.
connect (\r)    Reconnect to the server.
...
```

Liste aller vorhandenen Datenbanken:

```
mysql> show databases;
+----------+
| Database |
+----------+
| mysql    |
| test     |
+----------+
2 rows in set (0.20 sec)
```

Nach Neuinstallation eines MySQL-Servers standardmäßig:

- `mysql`: „System-Datenbank“
- `test`: „Spiel-Datenbank“

Zugriff auf eine bestimmte Datenbank:

```
% mysql -h mysql[....] ... Datenbank
```

Alternative: Klientenaufruf ohne Datenbankname

```
mysql> use mysql;
ERROR 1044: Access denied for user: 'otto@localhost'
to database 'mysql'
```

WER darf überhaupt WAS?

Etwas Theorie . . .

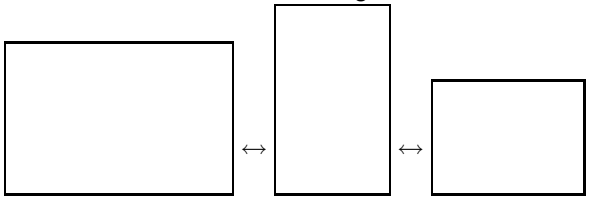
Daten: einzelne Informationen beliebiger Art und beliebigen Inhalts
Beispiele: "`Müller`", `0815`, `3.1415`, `CF081A37`

Tabelle (Relation): Sammlung von Daten zu einem bestimmten Thema, alle zusammengehörenden Daten stehen in einem Satz (*Datensatz*) oder Zeile, alle Daten einer bestimmten Semantik bilden eine Spalte (Feld)

1	Müller	31
2	Meier	59
3	Schulze	17

Beziehungen zwischen Tabellen sind üblich

Datenbank: besteht aus einer Menge von Tabellen



Datenbanknutzer: virtueller Nutzer eines Datenbanksystems
→ MySQL hat eine eigene Nutzerverwaltung (Datenbank-Login + Passwort)

Zugriffsrechte: in MySQL kann spezifiziert werden, welcher (Datenbank-)Nutzer von welchem/er Rechner/Rechnergruppe aus mit welcher Datenbank welche Operationen ausführen darf

keine Beziehung zu Loginkennzeichen und Zugriffsrechten im Filesystem!

Nutzungsszenarien:

1. für eine Datenbank **ein** Datenbanknutzer: wird von *allen* Personen (Nutzern) verwendet, die die Datenbank benutzen wollen
2. für eine Datenbank **zwei** Datenbanknutzer:
   - ➤ **ein** Datenbanknutzer mit Schreibrechten in der DB (Administrator)
   - ➤ **ein** Datenbanknutzer ohne Schreibrechte mit Nur-Leserechten (ggf. „öffentliche“ Nutzung)

Der Weg zu einer DB-Applikation

Neuaufbau einer Datenbank bedeutet:

1. ggf. Anlegen eines neuen Nutzers (oder mehrere)
2. Einstellen der Zugriffsberechtigungen
3. Anlegen der Datenbank
4. Anlegen der Tabellen der Datenbank
5. Füllen der Tabellen mit Daten
6. Entwickeln einer Applikation (bequemerer Zugriff)

Datenbankservice des URZ

Schritte 1. – 3.: brauchen *root*- bzw. *Admin*-Rechte — für einen Nutzer auf einem zentralen Server ausgeschlossen . . .

deshalb: Nutzen eines speziellen Werkzeugs =
Online-Formular zum automatisierten Anlegen einer Datenbank, das Verarbeitungsskript hat dann die notwendigen Rechte

URZ-Homepage ⇒ Datenbank-Dienst:
http://www.tu-chemnitz.de/urz/db/
Anlagen einer DB für diesen Kurs: `kursdb`

Und nun weiter?

Schritte 4. – 6.: macht Nutzer selbst (mit den Rechten des angelegten Datenbanknutzers)

zuerst: Struktur der geplanten Datenbank überlegen!

Projekt: primitive Adressdatenbank, 1 Tabelle (Adressen)

Relevante Daten: Vorname + Name, Postleitzahl + Ort, Straße mit Hausnummer
zusätzlich „Satznummer“

diese Daten bilden die Spalten der Tabelle, jede Spalte hat einen Datentyp

Mögliche Datentypen

1. Numerische Typen

`int`	Integerzahl
`float`	Gleitkommazahl, einfache Genauigkeit
`double`	Gleitkommazahl, doppelte Genauigkeit
`decimal`	Gleitkommazahl, als String dargestellt

   auch: `tinyint`, `smallint`, `bigint`
   die physischen Größen sind implementationsabhängig
   zusätzlich Angabe von Stellenanzahl möglich

2. Zeichenkettentypen

`char`	String fester Länge
`varchar`	String variabler Länge
`blob`	BLOB (Binary Large Object)
`text`	Textstring
`enum`	Auflistung
`set`	Menge

   auch: `tinyblob`,`mediumblob`,`longblob`, `tinytext`, `mediumtext`, `longtext`

3. Typen für Datum und Uhrzeit

`date`	Datum *JJJJ-MM-TT*
`time`	Uhrzeit *hh:mm:ss*
`datetime`	Datum und Uhrzeit
`timestamp`	Zeitstempel
`year`	Jahresangabe *JJJJ*

Weitere Spezifikationen

`default`	Standardwert
`not null`	Spaltenwert darf nicht NULL werden
`auto_increment`	automatischer Zähler
`primary key`	„Primärschlüssel“

Beachte: Bei Spaltennamen und Daten (Zeichenketten) wird keine Groß-Klein-Schreibung unterschieden!

Außerdem: Festlegen der verwendeten „Datenbank-Engine“:
`type=`*engine*

`MyISAM`	Standard in MySQL
`InnoDB`	unterstützt Transaktionen, ...
`...`	

Spezifikation für unser Beispiel

Feldname	*Typ*	*konkreter Typ*
Name	String	`char(128)`
Vorname	String	`char(128)`
PLZ	(ganze) Zahl	`int`
Ort	String	`char(128)`
Strasse	String	`char(128)`
PersonNr	(ganze) Zahl	`int`
		`primary key`
		`auto_increment`

SQL-Kommando zum Anlegen einer Tabelle

```
create table tabellenname (spaltendefinition,...)
```

Spaltendefinition

```
spaltenname typ [attribute]
```

Eingabe für unser Beispiel

Nutzung des MySQL-Kommandozeilen-Klienten (Alternativen später):

```
mysql -h mysql.hrz.tu-chemnitz.de -u kurs1 -p kursdb
```

Passwort: ...

oder:

```
mysql -h mysql.hrz.tu-chemnitz.de -u kurs1 -pK-Tn-ddam kursdb
```

```
create table adressen (
        Name     char(128),
        Vorname  char(128),
        PLZ      int,
        Ort      char(128),
        Strasse  char(128),
        PersonNr int primary key auto_increment);
```

Zur Kontrolle:

```
mysql> show tables;
+------------------+
| Tables_in_kursdb |
+------------------+
| adressen         |
+------------------+
1 row in set (0.00 sec)
```

```
mysql> describe adressen;
+----------+-----------+------+-----+--------- ...
| Field    | Type      | Null | Key | Default  ...
+----------+-----------+------+-----+--------- ...
| Name     | char(128) | YES  |     | NULL     ...
| Vorname  | char(128) | YES  |     | NULL     ...
| PLZ      | int(11)   | YES  |     | NULL     ...
| Ort      | char(128) | YES  |     | NULL     ...
| Strasse  | char(128) | YES  |     | NULL     ...
| PersonNr | int(11)   |      | PRI | NULL     ...
+----------+-----------+------+-----+--------- ...
6 rows in set (0.01 sec)
```

Füllen mit Daten

Name	*Vorname*	*PLZ*	*Ort*	*Strasse*
Mustermann	Max	12345	Adorf	Feldweg 3
Beispiel	Minna	99999	Dburg	Schlossgasse 9
Normalo	Otto	08150	Xberg	Hauptstraße 123

SQL-Kommando

```
insert into tabellenname [(spalte,...)] values(daten)
```

```
mysql> insert into adressen (Name,Vorname,PLZ,
Ort,Strasse) values ('Mustermann','Max','12345',
'Adorf','Feldweg 3');

mysql> insert into adressen (Name,Vorname,PLZ,
Ort,Strasse) values ('Beispiel','Minna','99999',
'Dburg','Schlossgasse 9');

mysql> insert into adressen (Name,Vorname,PLZ,
Ort,Strasse) values ('Normalo','Otto','08150',
'Xberg','Hauptstraße 123');
```

Kontrolle - Ausgabe der vorhandenen Daten

```
select * from tabellenname
select spalte1,spalte2,... from tabellenname
```

```
mysql> select * from adressen;
+------------+---------+-------+--------+-----------------+----------+
| Name       | Vorname | PLZ   | Ort    | Strasse         | PersonNr |
+------------+---------+-------+--------+-----------------+----------+
| Mustermann | Max     | 12345 | Adorf  | Feldweg 3       |        1 |
| Beispiel   | Minna   | 99999 | Dburg  | Schlossgasse 9  |        2 |
| Normalo    | Otto    |  8150 | Xberg  | Hauptstraße 123 |        3 |
+------------+---------+-------+--------+-----------------+----------+
3 rows in set (0.00 sec)
```

Erkenntnis: PLZ als `integer` ist ungünstig, weil führende Nullen fehlen

Ausweg: Feld als `char` vereinbaren

⇒ mit phpMyAdmin Änderung durchführen:
Definition ändern, Datum neu eingeben

Die wichtigsten SQL-Operationen

Einfügen von Datensätzen	`insert ...`
Anzeigen von Datensätzen	`select ...`
Ändern von Datensätzen	`update ...`
Löschen von Datensätzen	`delete ...`

genaue Syntax siehe MySQL-Dokumentationen

Fazit:

- ist eigentlich nicht schwierig
- aber doch etwas unhandlich
- damit ungeeignet als Angebot an andere Nutzer

Komfortablere Schnittstellen

MySQL-Konnektoren: APIs zur Entwicklung von Anwendungen
Connector/J, Connector/Net, Connector/MXJ

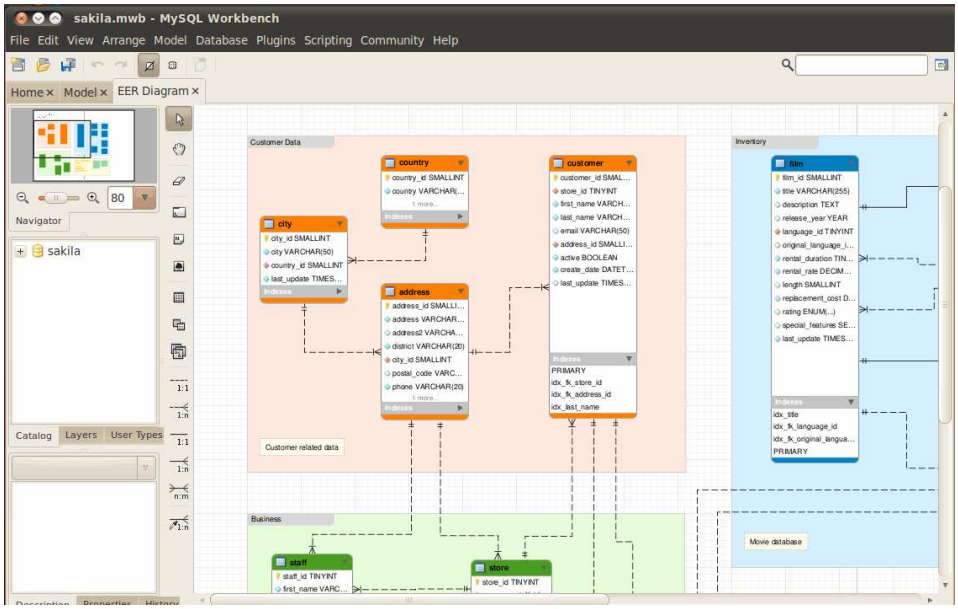
ODBC-Treiber (= MyODBC) verallgemeinerte Schnittstelle, damit bspw. auch Verbindung mit anderen Anwendungen möglich (OpenOffice, MS Office [Access], Delphi)

kommerzielle Klienten für Windows: Mascon, SQLyog, Navicat, ...

MySQL Query Browser Grafisches Datenbankmanagement für verschiedene Plattformen (Windows, Linux, ...)

knoda „Knorr's Datenbank“
`www.knoda.org`
Grafischer Klienten für verschiedene Datenbanksysteme

MySQL Workbench: GUI für Datenbankentwurf und -administration



Resüme:
alles nicht so recht geeignet zum Entwickeln von Applikationen (die andere nutzen sollen)

PHP

Datenbankanbindung

- Es gibt Funktionen für Adabas, dBase, Informix, Interbase, mSQL, MySQL, Oracle, Sybase, Postgres, . . .
- Die SQL-Statements werden als im Prinzip als Funktionsaufrufe im PHP-Programm hingeschrieben

1. Herstellen der Verbindung zum Server (≙ Aufruf des `mysql`-Klienten)

`mysql_connect("`*mysqlserver*`","`*nutzer*`","`*passwort*`");`

```
mysql_connect("mysql.hrz.tu-chemnitz.de",
              "kurs1","K-Tn-ddam");
```

Probleme: Nutzerkennzeichen und Passwort im Klartext!

- Quelltext ist eigentlich nicht sichtbar, weil Webserver immer das Ergebnis der Abarbeitung ausliefert (außer: `.phps`)
- jedes php-Skript ist File im Filesystem, das auch von anderen Programmen gelesen werden kann
  ⇒ Zugriffsrechte

- Bei AFS: geeignete ACL (Kommando `fs`)
  dabei muss der Webserver selbst Zugriff haben (`urz:www-server rl`)
- Zugriff auf Webseite (Skript) *über* Webserver wird mittels `.htaccess` gesteuert
- dort ggf. Authentifizierung per WTC (Shibboleth)
- das „wirkt“, wenn Zugriff per „Webseiten-Request“ erfolgt
- Zugriff durch PHP-Skript mit `open/read`-Funktionen ist aber *immer* möglich
  notwendige Voraussetzung: Schreibrechte im Webbaum *dieses* Webservers
- **Konsequenz: Passworte sind so nicht sicher!**
- Auswege:
  1. Passwort verschlüsseln, nur der Webserver kennt das Geheimnis („sectoken“):
     https://www.tu-chemnitz.de/urz/www/sectoken.html
  2. PROWebserver benutzen: http://www.tu-chemnitz.de/urz/www/pro/

`mysql_connect` liefert Rückkehrcode: „MySQL Verbindungs-Kennung“ oder „FALSE“
→ Fehlerbehandlung

```
$rc = mysql_connect("mysql.hrz.tu-chemnitz.de",
        "kurs1","K-Tn-ddam");
if (!$rc)
        die("Unable to connect to SQL server");
```

oder eleganter:

```
mysql_connect("mysql.hrz.tu-chemnitz.de",
        "kurs1","K-Tn-ddam") or
    die("Unable to connect to SQL server");
```

Boolsche Ausdrücke . . .

Teilausdr1	Teilausdr2	OR
True	True	True
False	True	True
False	False	False

2. Auswählen einer Datenbank (≙ SQL-Kommando `use` bzw. Datenbankname beim Aufruf des `mysql`-Klienten)

`mysql_select_db("`*datenbankname*`");`

```
mysql_select_db("kursdb") or
        die("Unable to select database");
```

3. Ausgabe des Inhalts (`adressen1`)

Allgemeine PHP-Anweisung zum Zugriff auf Inhalte (Lesen, Ändern, Löschen usw.)

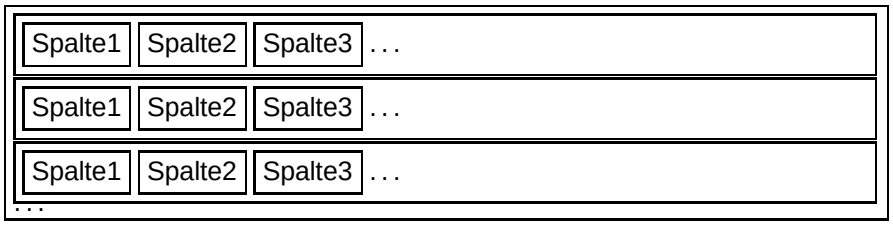
`$result = mysql_query("`*sql-kommando*`");`

Zur Erinnerung: SQL-Kommando zum Lesen:

`select * from` *tabelle*
`select` *spalte1,spalte2,...* `from` *tabelle*

```
$result = mysql_query("select * from adressen");
```

Ergebnis `$result`: „Matrix“ der betreffenden Daten:
Feld aus den „Zeilen“
jede Zeile ist Feld aus den gewünschten Spalteneinträgen



- Schleife über die Zeilen: `while`
- Zugriff auf eine Zeile:
  - `$row = mysql_fetch_row`
    dann innerhalb einer Zeile Zugriff auf die Spaltenwerte mittels Index: `$row[0]`,
    ...
  - `$row = mysql_fetch_array`
    liefert Assoziativfeld, Zugriff auf die Spaltenwerte mittels Spaltenname: `$row['Vorname']`,
    ...
- (stückweises) Erzeugen einer HTML-Tabelle

```
echo("<table><tr><th>Name</th><th>Vorname</th>
      <th>PLZ</th>...</tr>\n");
$query = mysql_query("select * from adressen");
while($row = mysql_fetch_array($query)) {
    echo("<tr><td>".$row['PersonNr']."</td>");
    echo("<td>".$row['Name']."</td>");
    echo("<td>".$row['Vorname']."</td>");
    ...
    echo("</tr>\n");
}
echo("</table>");
```

Test – Quelltext

4. Weitere Funktionen

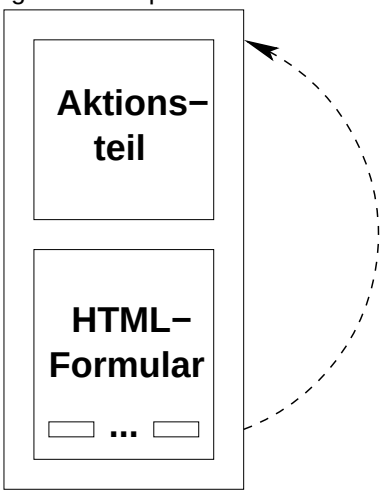
Denkbar (und sinnvoll):

- Einfügen neuer Datensätze
- Suchen von Sätzen
- Löschen von Sätzen
- Ändern von Daten

- ...

2 prinzipielle Möglichkeiten zur Realisierung weiterer Funktionalität:

- Entwickeln zusätzlicher Skripte, Hauptseite mit Links notwendig
- Erweitern des vorhandenen Skripts
  - Anlegen als HTML-Formular
  - Entgegennehmen von Nutzereingaben ⇒ HTML
  - Ausführen der gewünschten Aktionen ⇒ PHP
  - Formularziel: das gleiche Skript → rekursiv



Zugriff auf die Variablen des HTML-Formulars:
`if (isset($variable)) $variable = $_REQUEST["variable"]`

Sicherheit bei PHP-Skripten beachten!

- Überprüfung **aller** übergebenen Werte auf Sinnfälligkeit
- Ziel: Verhinderung des „SQL-Injection“
  Werte dürfen keine versteckten SQL-Kommandos enthalten
- Maskieren von Sonderzeichen:
  `$variable = mysql_escape_string($variable)`
- Wert soll Zahl sein:
  - Test: `if (is_numeric($nr)) ...`
  - Erzwingen: `settype($nr, 'integer')`
    `$nr = $nr + 0`

5. Anlegen des Skripts als Formular (`adressen2`)

```
<form action="adressen2.php" name="form" method="post">
<table align=center>
    <tr><td align=right><b>Name:</b></td>
        <td align=left><input name="name" maxlength="64">
            </td></tr>
    ...
...
<input type="submit" name="add" value="Neuen Satz einfügen">
</form>
```

Test – Quelltext

6. Kodieren des „Einfügen“-Falls (`adressen3`)

SQL-Kommando zum Einfügen in die DB:

```
insert into tabelle (spalte1,spalte2,...)
       values("wert1","wert2",...)
```

Als PHP-Anweisung so:

```
mysql_query("insert into tabelle (spalte1,spalte2,...)
        values(\"wert1\",\"wert2\",...)");
```

```
$name = $_REQUEST["name"];
$laenge = strlen($name);
if ($laenge > 128)
        die("falscher Parameter");
$name = ereg_replace(';', '', $name);
$name = addslashes($name);
...
```

```
if ($add) {
    mysql_query("insert into adressen
        (Name,Vorname,PLZ,Ort,Strasse)
        values (\"$name\",\"$vorname\",\"$plz\",
                \"$ort\",\"$strasse\")") or
        die("Einfügen geht schief");
}
```

Test – Quelltext

7. Suchen von Sätzen (`adressen4`)

konkret: Suchen nach Familienname, Ausgabe der „passenden“ Sätze rot markiert

1. Neuer `submit`-Button: „Sätze suchen“, Variable `search`
2. PHP: Suchoperation programmieren
   Suchen in SQL ist Lesen mit „Suchmaske“ (where-Klausel)
   `select * from tabelle where spalte="wert"`
   `select * from tabelle where spalte like "%wert%"`

3. „Gefundene“ Sätze markieren: die entsprechenden Indizes merken

```
else if ($search) {
    $name = "%$name%";
    $query1 = mysql_query("select * from adressen
        where (Name like \"$name\")");
    $rows = mysql_num_rows($query1);
    $i = 0;
    while($row = mysql_fetch_array($query1))
        $match[$i++] = $row['PersonNr'];
}
```

```
$query = mysql_query("select * from adressen");
while($row = mysql_fetch_array($query)) {
    $font = "<font>";
    if ($search) {
        for ($i = 0; $i < $rows; $i++) {
            if ($match[$i] == $row['PersonNr'])
                $font = "<font color=\"red\">";
    }   }
    echo("<tr><td>".$font$row['PersonNr']."</font></td>");
    echo("<td>".$font$row['Name']."</font></td>");
    ...
}
```

Test – Quelltext

8. Löschen, Ändern eines Satzes (`adressen5`)

Welcher Satz? ⇒ Markierungsbutton in der Ausgabeliste

```
while($row=mysql_fetch_array($query)) {
    echo("<tr><td><input name=\"satznr\"
          value=\"".$row['PersonNr']."\" type=\"radio\"></td>");
    echo("<td>".$row['Name']."</td>");
    echo("<td>".$row['Vorname']."</td>");
    ...
    echo("</tr>\n");
}
```

Test – Quelltext

Neue Funktionsbuttons (`adressen6`)

```
<input type="submit" name="add"
      value="Neuen Satz einfügen">
<input type="submit" name="search"
      value="Satz suchen">
<input type="submit" name="change"
      value="Markierten Satz ändern">
<input type="submit" name="delete"
      value="Markierten Satz löschen">
```

Löschen implementieren (adressen7)

`delete from` *tabelle* `where` *spalte*`="`*wert*`"`

```
else if ($delete) {
    $query = mysql_query("delete from adressen
             where (PersonNr = \"$satznr\")");
```

Test – Quelltext

Ändern (adressen8)

`update` *tabelle* `set` *spalte1*`="`*wert*`"` `where` ...
`update` *tabelle* `set` *spalte1*`="`*wert*`",` *spalte2*`="`*wert*`"` ... `where` ...

Noch behandeln: welche Felder sollen geändert werden – nur diese in die `update`-Anweisung aufnehmen
⇒ prüfen, welche Variable belegt ist

```
else if ($change) {
    $set = "";
    if ($name) $set .= "Name=\"$name\",";
    if ($vorname) $set .= "Vorname=\"$vorname\",";
    ...
    if ($set != "") {
        $setlength = strlen($set);
        $set = substr($set,0,$setlength-1);
        mysql_query("update adressen set $set
                     where (PersonNr = \"$satznr\")");
}
```

Test – Quelltext

9. Komfortable Sortierung der Ausgabe (adressen9)

Die Spaltenüberschriften werden „Button“, Anklicken bewirkt Sortieren der Tabelle (der Ausgabe) nach den Werten dieser Spalte

den Sortiervorgang überlassen wir MySQL:
`select * from` *tabelle* `order by` *spalte*

```
echo("<table><tr><th> </th>");
echo("  <th><input type=\"submit\" name=\"sort\"
        value=\"Name\"></th>");
echo("  <th><input type=\"submit\" name=\"sort\"
        value=\"Vorname\"></th>");
echo("  <th><input type=\"submit\" name=\"sort\"
        value=\"PLZ\"></th>");
...
```

```
switch ($sort) {
    case "Name":    $orderstring = "order by name";
                    break;
    case "Vorname": $orderstring = "order by vorname";
                    break;
    ...
    default:        $orderstring = "";
}
$query = mysql_query("select * from adressen $orderstring");
```

Test – Quelltext

10. Funktionelle Erweiterung: Eine primitive Bibliotheksanwendung
Anlegen einer Bücherliste, Verleih der Bücher

Neue Tabelle `buecher`:

Spalte	*Typ*
Autor	`char(128)`
Titel	`char(128)`
Verlag	`char(128)`
Anzahl	`int`
BuchNr	`int auto_increment`

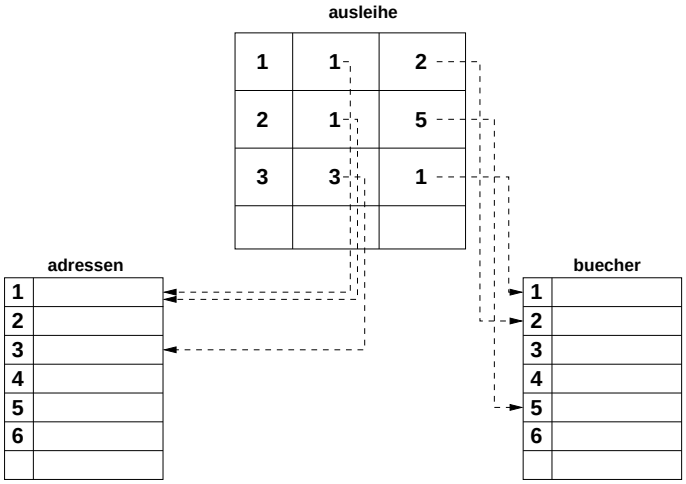
Welche Schnittstellen sind jetzt notwendig:

- Verwalten der Bücher: Einfügen, Suchen, Ändern, Löschen – analog zum Skript `adressen`
- Ausgeben von Büchern
- Anzeige, welche Bücher verfügbar sind
- Anzeige, welche Person hat welche Bücher ausgeliehen
- Anzeige, welches Buch von welchen Personen ausgeliehen ist
- Rücknahme von Büchern
- ...

Ausgeben von Büchern

Verknüpfung zwischen einem Satz der Tabelle `adressen` und einem Satz der Tabelle `buecher`
Ablegen in separater Tabelle `ausleihe`:

Spalte	*Typ*
Vorgang	`int auto_increment`
PersonNr	`int`
BuchNr	`int`



Aufbau eines Formulars:

- Anzeige aller vorhandenen Personen, z.B. als „Aufklappliste“:
  Holen der Daten aus `adressen`

```
$q1=mysql_query("select Name,Vorname,PersonNr from adressen");
echo("<select name=\"pliste\">");
while($row1 = mysql_fetch_array($q1)) {
    echo("<option value=\"".$row1["PersonNr"]."\">".
    $row1["Name"].", ".$row1["Vorname"]."</option>");
}
echo("</select>");
```

- für die vorhandenen Bücher analog
- Variablen `pliste` und `bliste` am Anfang des Skripts auswerten
- Schreiben eines Satzes in Tabelle `ausleihe`
  `$p`: Nummer der Person, `$b`: Nummer des Buches

```
mysql_query("insert into ausleihe (PersonNr, BuchNr)
    values($p,$b)");
```

- dazu vorher prüfen, ob Buch noch verfügbar

```
$erg1 = mysql_query
   ("select Anzahl from buecher where BuchNr=$b");
$row = mysql_fetch_array($erg1);
$anz = $row['Anzahl'];
$erg2 = mysql_query
   ("select count(*) from ausleihe where BuchNr=$b");
$row = mysql_fetch_row($erg2);
$aus = $row[0];
if ($aus < $anz)  ...
```

- nun gleich noch Anzeige, welche Bücher die Person nun hat

Variante 1: schrittweise

1. Holen der relevanten Sätze aus `ausleihe`
2. Für jede Buchnummer: Holen der Buchdaten

```
$erg1 = mysql_query
   ("select BuchNr from ausleihe where PersonNr=$p");
while ($row = mysql_fetch_array($erg1)) {
   $buch = $row['BuchNr'];
   $erg2 = mysql_query
     ("select Autor, Titel, Verlag from buecher
       where BuchNr=$buch");
   ...
}
```

Variante 2: eine Abfrage über mehrere Tabellen

`select` *spalte1*,*spalte2*[,...] `from` *tabelle1*,*tabelle2*[,...]

liefert das Kreuzprodukt: **alle** Möglichkeiten der Kombination der Daten aus *tabelle1* und *tabelle2*
= (Anzahl der Zeilen in *tabelle1*) $\times$ (Anzahl der Zeilen in *tabelle2*)

```
mysql_query("select Autor,Titel,Name,Vorname
    from buecher,adressen");
```

Autor	Titel	Name	Vorname
Dubois	MySQL 4	Mustermann	Max
Dubois	MySQL 4	Beispiel	Minna
Dubois	MySQL 4	Normalo	Otto
-	MySQL Administration	Mustermann	Max
-	MySQL Administration	Beispiel	Minna
-	MySQL Administration	Normalo	Otto
Born	HTML Kompendium	Mustermann	Max
Born	HTML Kompendium	Beispiel	Minna
Born	HTML Kompendium	Normalo	Otto
...	...	...	...

wir wollen aber nur die miteinander verknüpften Zeilen „sehen“: welche Bücher ein bestimmter Nutzer hat . . .

$\Rightarrow$ `where`-Klausel

bei gleichen Spaltennamen in unterschiedlichen Tabellen: *tabelle.spalte*

```
mysql_query("select Autor,Titel,Verlag,ausleihe.BuchNr
    from ausleihe,buecher
    where PersonNr=$p and buecher.BuchNr=ausleihe.BuchNr");
```

zu lesen als: suche in Tabelle `ausleihe` die Sätze zur Person `p` und dazu die Daten „Autor“, „Titel“ und „Verlag“ aus der Tabelle `buecher` mit der betreffenden Buchnummer, die die Person ausgeliehen hat

Test Endversion – Quelltext

Danke für die Aufmerksamkeit