ENUM vs. JOIN: Aufzählender Spaltentyp oder Nachschlagetabellen?

Dr. Volker Thormählen, 3. Sept. 2018

Inhaltsverzeichnis

1 Vorwort

Vorliegendes Dokument behandelt das Entscheidungsproblem, wie Tabellen des relationalen Datenbanksystems MySQL bzw. MariaDB bestmöglich angelegt werden können. Es wendet sich an Leser, die bereits Kenntnisse und Erfahrungen mit …

- dem relativen Datenbanksystem *MySQL bzw. MariaDB*,
- der dazugehörigen Benutzeroberfläche *phpMyAdmin,*
- den einschlägigen Fachbegriffen *ENUM* und *JOIN*
- sowie der Abfragesprache *SQL* besitzen.

2 Einleitung

MySQL bzw. MariaDB gilt als das populärste relationale Open-Source-Datenbanksystem der Welt. Dort kann bei der Einrichtung von Tabellen der aufzählende Spaltentyp ENUM genutzt werden.

Alternativ dazu können Nachschlagetabellen (engl. *lookup-tables*) angelegt werden, auf deren Inhalt mittels Fremdschlüsseln (engl. *foreign keys*) zugegriffen werden kann.

Im Folgenden wird anhand eines konkreten Beispiels dargestellt, welche Vor- und Nachteile mit der Verwendung des Spalten- bzw. Datentyps ENUM verbunden sind.

Insgesamt werden in MySQL 4 Tabellen mit der Abfragesprache SQL angelegt:

- Die Tabelle '***personen***'. Sie enthält u. a. die enumerativen Spalten '*geschlecht*' und '*anrede'*:
  - *Geschlecht* ENUM('divers','männlich','weiblich') NOT NULL DEFAULT 'männlich',
  - *anrede* ENUM('Frau','Herr') NOT NULL DEFAULT 'Herr'
- Die Tabelle ‚**personen_nt**'. Mit den dort definierten numerischen Fremdschlüsseln '*anredeID*' und '*geschlechtID*' ist sie mit den beiden Nachschlagtabellen
  - *'**anreden***' bzw.
  - '**geschlecht**'

  relational verbunden.

Zunächst wird der ENUM-Daten- bzw. Spaltentyp vorgestellt, der in der Tabelle *'personen*' vorkommt.

3 Datenbanktabelle mit dem Spaltentyp ENUM anlegen

Der ENUM-Spaltentyp ist nützlich beim Entwurf einer Datenbank. Er ermöglicht die Einfachauswahl aus einer Liste definierter Werte, vgl. [1], S. 397:

> *„Ein ENUM ist ein Zeichenketten-Objekt, dessen Wert normalerweise aus einer Liste zulässiger Werte ausgesucht wird, die explizit bei der Spaltenspezifizierung bei der Tabellenerzeugung aufgezählt werden. Der Wert kann unter bestimmten Umständen auch die leere Zeichenkette ("") oder NULL sein.“*

Wichtige Eigenschaften sind:

- Eine Liste von max. 65535 Werten kann definiert werden.
- Speichert entweder nichts oder nur einen der definierten Werte.
- Beim Speichern werden ungültige Werte als leere Zeichenfolge gespeichert.
- Bei Verwendung von NOT NULL ist der erste Wert in der Liste der Standardwert.
- Bei Verwendung von NULL ist dies der Standard-Wert.
- Die Werte werden in der Reihenfolge sortiert, in der sie eingeben werden.

Weitere nützlich Informationen über diesen Zeichenkettentyp: Siehe ebenda, S. 384 ff.

Diese beiden ENUM-Spalten werden im Folgenden benutzt (vgl. u. a. Abb. 1):

- `geschlecht ENUM('divers','männlich,'weiblich')`
- `anrede ENUM('Frau','Herr')`, *später auch:* `anrede ENUM('Familie','Frau','Herr')`

Die Tabelle '*personen*' anlegen und mit Werten belegen wird mit 3 SQL-Anweisungen durchgeführt, siehe Abb. 1.

SQL-Befehl(e) in Datenbank my_db ausführen:

```
1  DROP TABLE IF EXISTS personen;
2
3  CREATE TABLE IF NOT EXISTS personen (
4      id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
5      vollerName VARCHAR(30) NOT NULL UNIQUE,
6      lebensalter TINYINT UNSIGNED NOT NULL,
7      geschlecht ENUM('divers','männlich','weiblich') NOT NULL DEFAULT 'männlich',
8      anrede ENUM('Frau','Herr') NOT NULL DEFAULT 'Herr'
9  )
10 ENGINE=INNODB AUTO_INCREMENT=1 CHARACTER SET=utf8 COLLATE=utf8_unicode_ci;
11
12 INSERT INTO personen (id, vollerName,lebensalter, geschlecht, anrede)
13     VALUES('','V. Thormählen',78,2,2);
```

Abb. 1: Tabelle '*personen*' mit 5 Spalten in MySQL-Datenbank '*my_db*' anlegen und mit Werten belegen

Als Ergebnis der drei vorstehenden SQL-Anweisungen wird eine erste Zeile in die Tabelle '*personen*' eingefügt (siehe Abb. 2).

	id	vollerName	lebensalter	geschlecht	anrede
Bearbeiten Kopieren Löschen	1	V. Thormählen	78	männlich	Herr

Abb. 2: Ergebnis der drei vorstehenden SQL-Anweisungen

Die Liste der ENUM-Werte in Abb. 17 wird mit folgendem PHP-Skript erzeugt (siehe Listing 1):

```
<?php
function getOptionsFromEnum($conn, $tbl) {
// Alle Einträge in allen ENUM-Spalten der MySQL-Tabelle '$tbl' auflisten.
   $sql = "SHOW COLUMNS FROM $tbl WHERE TYPE LIKE 'enum%'";
   if ($result = mysqli_query($conn, $sql)) {
      echo "Die ENUM-Einträge in der MySQL-Tabelle '$tbl' lauten:<br>";
      // Alles zwischen Apostrophs auswählen.
      $pattern = "/'([^']+)'/";
      while ($row=mysqli_fetch_row($result)) {
         echo "<br>Tabellenspalte: <b>" . $row[0] . "</b><br>";
         preg_match_all($pattern, $row[1], $matches);
         $enums = $matches[1];
         If (sizeof($enums) > 0) {
            foreach($enums as $val) {
               echo "<br>    " . $val;
            }
         }
         echo "<br>";
      }
      mysqli_free_result($result);
   }
}
// Verbindung mit MySQL herstellen.
    $conn = mysqli_connect("localhost","…","…");
// Verbindung mit MySQL prüfen.
if (mysqli_connect_errno()) {
   echo "Fehler: Nicht mit MySQL verbunden: " . mysqli_connect_error();
}
// Zeichensatz bestimmen.
   mysqli_set_charset($conn,"utf8");
// Datenbank auswählen.
   mysqli_select_db($conn,"my_db");
// Tabelle mit ENUM-Spalten festlegen.
   $tbl = "personen";
// Alle Einträge in allen ENUM-Spalten der MySQL-Tabelle '$tbl' auflisten.
   getOptionsFromEnum($conn, $tbl);
?>
```

Listing 1: Alle Einträge in allen ENUM-Spalten der Tabelle '*personen*' ausgeben

3.1 Auswahllisten für **HTML-Formulare automatisch erzeugen**

Ausgehend von einer ENUM-Tabellenspalte können mit einem PHP-Skript Formulare relativ einfach generiert werden (siehe

```
<?php
function getEnumValues($conn, $tbl, $tblCol, $selected) {
   // Optionen für das HTML-Dropdown-Steuerelement mit Werten
   // der ENUM-Tabellenspalte '$tblCol' erzeugen.
   $list = getEnumFldValues($conn,$tbl,$tblCol);
   if (sizeof($list) > 0) {
      $label = strtoupper($tblCol);
      echo "<label>$label: </label>";
      echo "<select name='$tblCol'>";
      foreach($list as $val) {
         // Hochkomma am Anfang und Ende der Zeichenkette $val entfernen
         $val = trim($val, "'");
         // Option definieren
         echo "<option value='" . $val . "'";
         // Option auswählen
         if($val === $selected){
            echo " selected ";
         }
         echo ">" . $val . "</option>\n";
      }
      echo "</select>";
   }
}
function getEnumFldValues($conn,$tbl,$tblCol) {
   // ENUM-/SET-Werte aus Spalte $tblCol der Tabelle $tbl
   // für das HTML-Dropdown-Steuerelement ermitteln.
   $sql = "SHOW COLUMNS FROM $tbl WHERE FIELD='".$tblCol."'";
   $fldArr = array();
   if ($result=mysqli_query($conn,$sql)) {
      $fldDetail = mysqli_fetch_array($result);
      $type = preg_replace("/(^set\()|(^enum\()|\"|\)/i", "", $fldDetail["Type"]);
      $fldArr = explode(",",$type);
   }
   return $fldArr;
}
```

```
// Verbindung mit MySQL herstellen.
   $conn = mysqli_connect("localhost","root","mvt1940");
// Verbindung mit MySQL prüfen.
if (mysqli_connect_errno()) {
   echo "Fehler: Nicht mit MySQL verbunden: " . mysqli_connect_error();
}
// Zeichensatz bestimmen.
   mysqli_set_charset($conn,"utf8");
// Datenbank auswählen.
   mysqli_select_db($conn,"my_db");
// Tabelle mit ENUM-Spalten festlegen.
   $tbl = "personen";
// ENUM-Spalte vorgeben.
   $tblCol = "geschlecht";
// Standardwert für Drop-Down-Auswahliste bestimmen.
   $selected = "männlich";
// Alle Einträge in allen ENUM-Spalten der MySQL-Tabelle '$tbl' auflisten.
   getEnumValues($conn, $tbl, $tblCol, $selected);
?>
```

Listing 2):

```
<?php
function getEnumValues($conn, $tbl, $tblCol, $selected) {
   // Optionen für das HTML-Dropdown-Steuerelement mit Werten
   // der ENUM-Tabellenspalte '$tblCol' erzeugen.
   $list = getEnumFldValues($conn,$tbl,$tblCol);
   if (sizeof($list) > 0) {
      $label = strtoupper($tblCol);
      echo "<label>$label: </label>";
      echo "<select name='$tblCol'>";
      foreach($list as $val) {
         // Hochkomma am Anfang und Ende der Zeichenkette $val entfernen
         $val = trim($val, "'");
         // Option definieren
         echo "<option value='" . $val . "'";
         // Option auswählen
         if($val === $selected){
            echo " selected ";
         }
         echo ">" . $val . "</option>\n";
      }
      echo "</select>";
   }
}
function getEnumFldValues($conn,$tbl,$tblCol) {
   // ENUM-/SET-Werte aus Spalte $tblCol der Tabelle $tbl
   // für das HTML-Dropdown-Steuerelement ermitteln.
   $sql = "SHOW COLUMNS FROM $tbl WHERE FIELD='".$tblCol."'";
   $fldArr = array();
   if ($result=mysqli_query($conn,$sql)) {
      $fldDetail = mysqli_fetch_array($result);
      $type = preg_replace("/(^set\()|(^enum\()|\"|\)/i", "", $fldDetail["Type"]);
      $fldArr = explode(",",$type);
   }
   return $fldArr;
}
```

```php
// Verbindung mit MySQL herstellen.
   $conn = mysqli_connect("localhost","root","mvt1940");
// Verbindung mit MySQL prüfen.
if (mysqli_connect_errno()) {
   echo "Fehler: Nicht mit MySQL verbunden: " . mysqli_connect_error();
}
// Zeichensatz bestimmen.
   mysqli_set_charset($conn,"utf8");
// Datenbank auswählen.
   mysqli_select_db($conn,"my_db");
// Tabelle mit ENUM-Spalten festlegen.
   $tbl = "personen";
// ENUM-Spalte vorgeben.
   $tblCol = "geschlecht";
// Standardwert für Drop-Down-Auswahliste bestimmen.
   $selected = "männlich";
// Alle Einträge in allen ENUM-Spalten der MySQL-Tabelle '$tbl' auflisten.
   getEnumValues($conn, $tbl, $tblCol, $selected);
?>
```

Listing 2: Werte einer ENUM-Tabellenspalte als Auswahlliste eines HTML-Formulars anzeigen, Quelle: in Anlehnung an [2]

Mit dem PHP-Skript in Listing 5 wird eine HTML-Auswahlliste mit den Werten der ENUM-Tabellenspalte '*geschlecht*' der Tabelle '*personen*' erzeugt, bei der zunächst nur ein Element sichtbar ist (siehe Abb. 3).

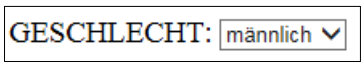

Abb. 3: Mit vorstehendem PHP-Skript erzeugte HTML-Auswahlliste im Browser

Der erste Listeneintrag (hier: '*männlich*') ist vorausgewählt. Die Vorauswahl erfolgt mit dem Attribut "*selected*" (siehe Listing 2, Funktion '*getEnumValues*', gelbe Hervorhebung). Ohne Vorauswahl würde der erste Listeneintrag '*divers*' lauten (vgl. Abb. 18 und Abb. 17).

3.2 Abfrage erstellen und als Ansicht speichern

Im folgenden Beispiel wird die bereits erstellte Tabelle '*personen_nt*' mit den beiden Nachschlagetabellen '*anreden'* und '*geschlecht*' mittels LEFT JOIN relational verbunden (siehe Abb. 4):

SQL-Befehl(e) in Tabelle my_db.persons ausführen:

```
1 SELECT a.anrede, p.vollerName,p.lebensAlter, g.sex
2 FROM (persons as p
3     LEFT JOIN anreden    AS a ON p.AnredeID     = a.id)
4     LEFT JOIN geschlecht AS g ON p.geschlechtID = g.id
5 WHERE p.lebensAlter = 78 AND a.anrede = "Herr";
```

Abb. 4: SQL-Anweisung zur N:1-Verknüpfung von 2 Datenbanktabellen mit LEFT JOIN

Das Ergebnis dieser komplexen Abfrage wird in Abb. 5 präsentiert:

anrede	vollerName	lebensalter	sex
Herr	V. Thormählen	78	männlich

Abb. 5: Ergebnis der vorstehenden komplexen SQL-Abfrage

Wenn das Ergebnis einer Abfrage als Ansicht (sog. *VIEW*) gespeichert werden soll, muss das obige Beispiel etwas geändert werden:

Die bereits erwähnte Tabelle '*personen_nt*' wird wiederum mit den beiden Nachschlagetabellen *'anreden*' und '*geschlecht*' verbunden (siehe Abb. 6) und das Abfrageergebnis wird als Ansicht unter dem Namen *'personen_nt_by_sex'* (siehe Abb. 7) gespeichert. Mit der SQL-Anweisung *CREATE VIEW* wird die Abfrage eingeleitet. Mit den beiden JOIN-Anweisungen werden die beiden Nachschlagetabellen '*anreden*' und '*geschlecht*' relational mit der Tabelle '*personen_nt*' verknüpft:

SQL-Befehl(e) in Datenbank my_db ausführen:

```
1  DROP VIEW personen_nt_by_sex;
2
3  CREATE VIEW personen_nt_by_sex
4  AS SELECT
5      p.vollerName  AS 'voller Name',
6      p.lebensAlter AS Lebensalter,
7      a.anrede      AS Anrede,
8      g.sex         AS Geschlecht
9  FROM personen_nt AS p
10     JOIN anreden    AS a ON p.anredeID     =  a.id
11     JOIN geschlecht AS g ON p.geschlechtID =  g.id;
```

Abb. 6: Die Ansicht *'personen_nt_by_sex'* mit einer komplexen SQL-Anweisung erzeugen

Das Ergebnis dieser *CREATE VIEW*-Anweisung ist in Abb. 7 zu sehen:

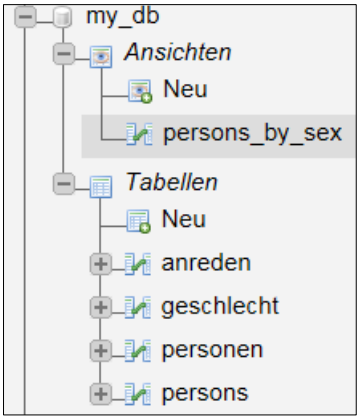

Abb. 7: Erzeugte Ansicht '*personen_nt_by_sex*' im Navigationsmenü von phpMyAdmin

Zum Vergleich wird eine analoge Auswahlabfrage mit der Tabelle '*personen*' durchgeführt, die zwei Spalten vom Typ ENUM enthält (vgl. Abb. 1):

Abb. 8: Einfache SQL-Abfrage der Tabelle '*personen*' mit WHERE Klausel

Diese Abfrage ist einfacher und kürzer als die zuvor in Abb. 4 gezeigte, weil keine N:1-Verknüpfung mit Nachschlagetabellen erforderlich ist.

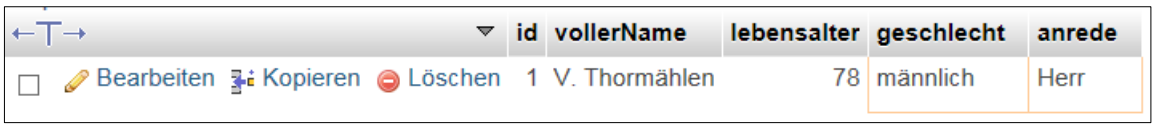

Abb. 9: Ergebnis der vorstehenden einfachen SQL-Abfrage

3.3 Tabelle mit Spalten vom Typ ENUM ändern

Wenn beispielsweise die ENUM-Spalte '*anrede*' der Tabelle '*personen*' durch den neuen Eintrag '*Familie*' erweitert werden soll, kann das nicht einfach mit der SQL-Anweisung "INSERT INTO …" erfolgen, wie das bei der Nachschlagetabelle '*anreden*' möglich wäre. Stattdessen muss die betreffende ENUM-Spalte mit der SQL-Anweisung "ALTER TABLE …" geändert werden (siehe Abb. 10):

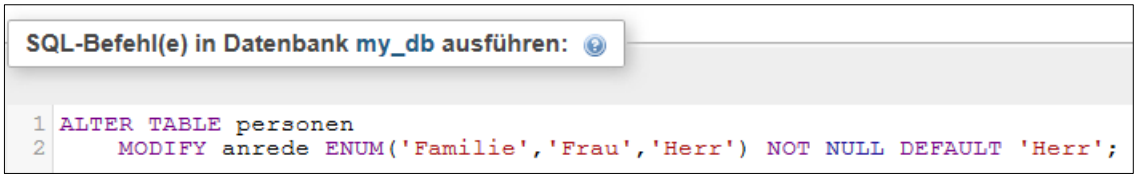

Abb. 10: Tabellenänderung

Bei häufigen Änderungen ist das lästig und nicht ohne Risiko. In vorliegenden Fall klappt die Änderung. Überprüft werden kann das mit folgender SQL-Anweisung (siehe Abb. 11):

Abb. 11: ENUM-Spalte 'anrede' der Tabelle '*personen*' anzeigen

Field	Type	Null	Key	Default	Extra
anrede	enum('Familie','Frau','Herr')	NO		Herr	

Abb. 12: Ergebnis der vorstehenden SQL-Anweisung

3.4 Spaltentyp ENUM abfragen

Die Abfrage des ENUM-Spaltentyps '*geschlecht*' der Tabelle '*personen*' kann mit der Tabelle COLUMNS des Informationsschemas erfolgen (siehe Abb. 13):

SQL-Befehl(e) in Tabelle my_db.COLUMNS ausführen:

```
1 SELECT
2     column_type
3 FROM
4     information_schema.COLUMNS
5 WHERE
6     TABLE_NAME = 'personen'
7     AND COLUMN_Name = 'geschlecht';
```

Abb. 13: Abfrage des Spaltentyps '*geschlecht*' in der Tabelle '*personen*'

Das Ergebnis der vorstehenden Abfrage ist in Abb. 14 zu sehen:

column_type
enum('divers','männlich','weiblich')

Abb. 14: Ergebnis der Abfrage des Typs der Spalte '*geschlecht*' in der Tabelle '*personen*'

3.5 Datensatz mit undefiniertem ENUM-Wert einfügen

Angenommen, ein unbekannter ENUM-Wert (beispielsweise '*Fräulein*') wird mit folgender SQL-Anfügeabfrage in die Tabelle '*personen*' eingefügt:

```
INSERT INTO personen(id,vollername,lebenssalter,geschlecht,anrede)
  VALUES(Null,'M. Thormählen',78,2,'Fräulein');
```

Listing 3: Neuen Datensatz in Tabelle '*personen*' einfügen mit undefiniertem ENUM-Wert

Ergebnis der SQL-Anweisung "*INSERT INTO* …" in Listing 3 ist, dass zwar ein neuer Datensatz in die Tabelle '*personen*' eingefügt, aber die Spalte '*anrede*' nicht belegt wird, weil die Zeichenfolge '*Fräulein*' kein vorher definierter Aufzählungswert ist (siehe Abb. 15, Sp. anrede).

	id	vollerName	lebensalter	geschlecht	anrede
Bearbeiten Kopieren Löschen	1	V. Thormählen	78	männlich	Herr
Bearbeiten Kopieren Löschen	2	M. Thormählen	78	männlich	

Abb. 15: Datensätze in der Tabelle '*personen*' nach vorstehender SQL-INSERT INTO-Anweisung

Was nun? Wie kann mit einer SQL-Auswahlabfrage herausgefunden werden, ob der Spalte '*anrede*' ein undefinierter ENUM-Wert zugewiesen wurde?

Beim Einfügen eines ungültigen Wertes in eine ENUM-Spalte wird statt einer leeren Zeichenfolge ein spezieller Fehlerwert eingefügt. Diese spezielle Zeichenfolge kann von einer normalen leeren Zeichenfolge dadurch unterschieden werden, dass der zugehörige Indexwert 0 ist[1]. Folglich kann mit folgender SQL-Auswahlabfrage bestimmt werden, welche Zeilen ungültige ENUM-Werte enthalten[2]:

SQL-Befehl(e) in Tabelle my_db.personen ausführen:

```
1 SELECT * FROM personen WHERE anrede = 0;
```

Abb. 16: Alle Datensätze der Tabelle '*personen*' finden mit ungültigem ENUM-Wert in der Spalte '*anrede*'

[1] Der Begriff "Index" bezieht sich hier auf die relative Position innerhalb der Liste der ENUM-Werte. Es hat nichts mit Tabellenindizes zu tun.

[2] Der Index der ENUM-Spalte '*anrede*' der Tabelle '*personen*' lautet:

Wert	Index
NULL	NULL
''	0
Familie	1
Frau	2
Herr	3

3.6 Alle Werte der ENUM-Spalten ausgeben

In Abb. 17werden alle ENUM-Einträge der Tabelle '*personen*' nochmals präsentiert:

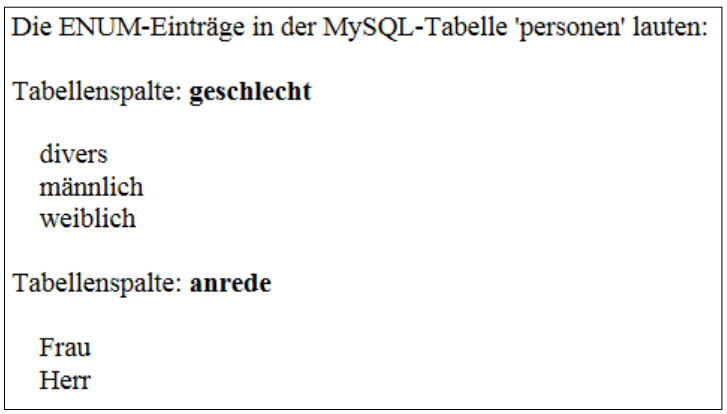

Abb. 17: Alle ENUM-Werte in der Tabelle '*personen*' auflisten

4 Verwendung von Nachschlagetabellen

4.1 Nachschlagetabellen anlegen und mit Werten belegen

Ausgehend von den beiden Arrays

$*geschlecht* = array("divers","männlich","weiblich");
$anreden = array("Frau","Herr");

können die benötigten Nachschlagetabellen mit PHP-Code automatisch generiert werden. Gezeigt wird im Folgenden, wie die Nachschlagetabelle '*geschlecht*' erstellt und mit Werten belegt werden kann. Die Nachlagetabelle '*anreden*' wird analog erstellt und belegt.

Drei Voraussetzungen müssen erfüllt sein, um eine Fremdschlüsselbeziehung zwischen zwei MySQL-Tabellen herzustellen:

1. Beide Tabellen müssen vom Typ **InnoDB** sein.
2. Die für die Fremdschlüsselbeziehungen benutzten Spalten müssen **indiziert** sein.
3. Die für die Fremdschlüsselbeziehung benutzten Spalten müssen vom **gleichen Datentyp** sein.

Listing 4 beinhaltet das Treiber-Skript zum Aufruf von zwei PHP-Funktionen (siehe Tabelle 1):

PHP-Funktion	Zahl d. Argumente	Aufgabe
createLookupTable	3	Nachschlagetabelle erstellen
insertIntoLookupTable	4	Spaltenwerte in eine Nachschlagetabelle einfügen

Tabelle 1: PHP-Funktionen zum Erstellen und Belegen einer Nachschlagetabelle

```
$conn =                    // Verbindung zur Datenbank MySQL herstellen
$tableNm = "geschlecht";   // Tabellenname
$column  = "sex";          // Spaltenname
// Nachschlagetabelle 'geschlecht' erstellen.
$retVal = createLookupTable($conn, $tableNm, $column);
if($retVal) {
     echo "Tabelle $tableNm wurde erfolgreich erstellt!<br>";
} else {
     echo "Tabelle $tableNm NICHT erstellt!<br>";
}
// Spaltenwerte in die Nachschlagetabelle 'geschlecht' einfügen.
$geschlecht = array( "divers", "männlich", "weiblich");
foreach ($geschlecht as $sex ) {
   $retVal = insertIntoLookupTable($conn, $tableNm, $column, $sex);
   if($retVal) {
        echo "Geschlecht $sex wurde erfolgreich eingefügt!<br>";
   } else {
        echo "Geschlecht $sex wurde NICHT eingefügt!<br>";
   }
}
```

Listing 4: PHP-Treiber-Skript mit 2 Funktionsaufrufen zum Erstellen u. Belegen der Nachschlagetabelle '*geschlecht*'

```php
// Eine Nachschlagetabelle erstellen.
function createLookupTable($conn, $tableNm, $column) {
   $sql =  "DROP TABLE IF EXISTS " . $tableNm;
   mysqli_query($conn,$sql);
   $sql = "CREATE TABLE IF NOT EXISTS " . $tableNm .
   "(id SMALLINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,".$column."
     VARCHAR(20) NOT NULL UNIQUE)
   ENGINE=InnoDB DEFAULT CHARACTER SET=utf8 COLLATE=utf8_unicode_ci";
   $retVal = mysqli_query($conn,$sql);
   return $retVal;
}
```

Listing 5: PHP-Funktion zur Erstellung einer Nachschlagetabelle

```php
// Spaltenwerte in eine Nachschlagetabelle einfügen.
function insertIntoLookupTable($conn, $tableNm, $column, $value) {
   $sql = "INSERT INTO $tableNm (id, $column) VALUES (NULL, '$value');";
   $retVal = mysqli_query($conn,$sql);
   return $retVal;
}
```

Listing 6: PHP-Funktion zum Einfügen von Spaltenwerten in eine Nachschlagetabelle

Das Ergebnis der oben genannten PHP-Skripte wird in Abb. 18 gezeigt:

Inhalt der Nachschlagetabelle '*geschlecht*'

	id	sex
Bearbeiten Kopieren Löschen	1	divers
Bearbeiten Kopieren Löschen	2	männlich
Bearbeiten Kopieren Löschen	3	weiblich

Inhalt der Nachschlagetabelle 'anreden'

	id	anrede
Bearbeiten Kopieren Löschen	1	Frau
Bearbeiten Kopieren Löschen	2	Herr

Abb. 18: Jeweiliger Inhalt der beiden Nachschlagetabellen

4.2 Haupttabelle 'personen_nt' erstellen

Mit einem *JOIN* lassen sich Verknüpfungen zwischen Datenbanktabellen oder innerhalb einer einzigen Datenbanktabelle herstellen. Voraussetzung ist eine indizierte Spalte (sog. Join-Spalte), die in beiden Tabellen vorkommt und den gleichen Datentyp besitzt.

In MySQL bzw. MariaDB stehen vier *JOIN*-Typen zur Verfügung. Nur der nützlichste und am leichtesten zu verstehende *JOIN*-Typ, nämlich *LEFT JOIN*, wird im Folgenden angewandt, um eine N:1-Beziehung zwischen der Haupttabelle ('*personen_nt*'[3]) und zwei dazugehörigen Nachschlagetabellen ('*anreden*' bzw. '*geschlecht*') zu erstellen.

Die Tabelle *'personen_nt*' anlegen, verknüpfen und mit Werten belegen wird wiederum mit SQL-Anweisungen durchgeführt, siehe Abb. 19.

SQL-Befehl(e) in Datenbank my_db ausführen:

```
1  DROP TABLE IF EXISTS personen_nt;
2  CREATE TABLE IF NOT EXISTS personen_nt(
3     id SMALLINT UNSIGNED AUTO_INCREMENT NOT NULL,
4     vollerName    VARCHAR(30)         NOT NULL,
5     lebensAlter   TINYINT UNSIGNED    NOT NULL,
6     anredeID      SMALLINT UNSIGNED NOT NULL,geschlechtID SMALLINT UNSIGNED NOT NUll,
7     CONSTRAINT FK_anrede FOREIGN KEY (anredeID) REFERENCES anreden(id)
8          ON DELETE RESTRICT ON UPDATE CASCADE,
9     CONSTRAINT FK_geschlecht FOREIGN KEY (geschlechtID) REFERENCES geschlecht(id)
10         ON DELETE RESTRICT ON UPDATE CASCADE,
11    PRIMARY KEY(id), UNIQUE KEY(vollerName)
12 )
13 ENGINE=INNODB CHARACTER SET=utf8 COLLATE=utf8_unicode_ci;
14 INSERT INTO personen_nt(id,vollerName,lebensAlter,anredeID,geschlechtID)
15 VALUES('','V. Thormählen',78,2,2);
```

Abb. 19: Tabelle *'personen_nt*' mit 5 Spalten in MySQL-Datenbank anlegen und mit Werten belegen

4.3 Datensatz in Tabelle 'personen_nt' einfügen

Als Ergebnis der drei vorstehenden SQL-Anweisungen wird eine erste Zeile in die Tabelle '*personen_nt'* eingefügt (siehe Abb. 20).

	id	vollerName	lebensAlter	anredeID	geschlechtID
☐ Bearbeiten Kopieren Löschen	1	V.Thormählen	78	2	2

Abb. 20: Erster Datensatz der Tabelle *'personen_nt*' in der MySQL-Datenbank 'my_db'

[3] Das Kürzel 'nt' steht für **N**achschlageTabellle

4.4 Abfrage erstellen und als Ansicht speichern

Im folgenden Beispiel wird die bereits erstellte Tabelle *'personen_nt'* mit den beiden Nachschlagetabellen '*anreden*' und '*geschlecht*' mittels LEFT JOIN relational verbunden (siehe Abb. 21):

SQL-Befehl(e) in Tabelle my_db.personen_nt ausführen:

```
1 SELECT a.anrede, p.vollerName, p.lebensAlter, g.sex
2 FROM (personen_nt AS p
3       LEFT JOIN anreden    AS a on p.anredeID     = a.id)
4       LEFT JOIN geschlecht AS g ON p.geschlechtID = g.id
5 WHERE p.lebensAlter = 78 AND a.anrede = "Herr";
```

Abb. 21: SQL-Anweisung zur N:1-Verknüpfung von 2 Datenbanktabellen mit LEFT JOIN

Das Ergebnis dieser komplexen Abfrage wird in Abb. 22präsentiert:

anrede	vollerName	lebensAlter	sex
Herr	V. Thormählen	78	männlich

Abb. 22: Ergebnis der vorstehenden komplexen SQL-Abfrage

Wenn das Ergebnis einer Abfrage als Ansicht (sog. *VIEW*) gespeichert werden soll, muss das obige Beispiel etwas geändert werden:

Die bereits erwähnte Tabelle *'personen_nt*' wird wiederum mit den beiden Nachschlagetabellen '*anreden*' und '*geschlecht*' verbunden (siehe Abb. 23) und das Abfrageergebnis wird als Ansicht unter dem Namen '*personen_nt_by_sex*' (siehe Abb. 24) gespeichert. Mit der SQL-Anweisung CREATE VIEW wird die Abfrage eingeleitet. Mit den beiden JOIN-Anweisungen werden die beiden Nachschlagetabellen '*anreden*' und '*geschlecht*' relational mit der Tabelle *'personen_nt*' verknüpft:

SQL-Befehl(e) in Datenbank my_db ausführen:

```
1  DROP VIEW personen_nt_by_sex;
2
3  CREATE VIEW personen_nt_by_sex
4  AS SELECT
5      p.vollerName  AS 'voller Name',
6      p.lebensAlter AS Lebensalter,
7      a.anrede      AS Anrede,
8      g.sex         AS Geschlecht
9  FROM personen_nt AS p
10     JOIN anreden    AS a ON p.anredeID     =  a.id
11     JOIN geschlecht AS g ON p.geschlechtID =  g.id
```

Abb. 23: Die Ansicht *'personen_nt_by_sex*' mit einer komplexen SQL-Anweisung erzeugen

Das Ergebnis dieser CREATE VIEW-Anweisung ist in Abb. 24 zu sehen:

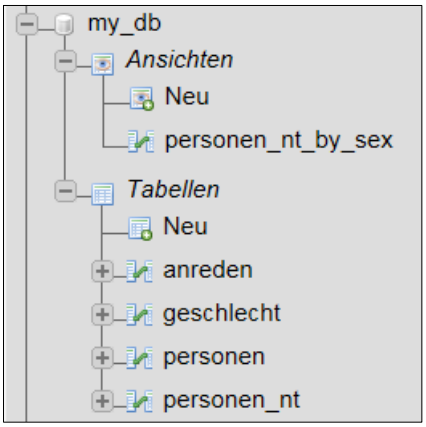

Abb. 24: Erzeugte Ansicht '*personen_nt_by_sex*' im Navigationsmenü von phpMyAdmin

Zum Vergleich wird eine analoge Auswahlabfrage mit der Tabelle '*personen*' durchgeführt, die bekanntlich zwei Spalten vom Typ ENUM enthält (vgl. Abb. 1 in Verbindung mit Abb. 25):

Abb. 25: Einfache SQL-Abfrage der Tabelle '*personen*' mit WHERE Klausel

Diese Abfrage ist einfacher und kürzer als die zuvor in Abb. 21 gezeigte, weil keine N:1-Verknüpfung mit Nachschlagetabellen erforderlich ist.

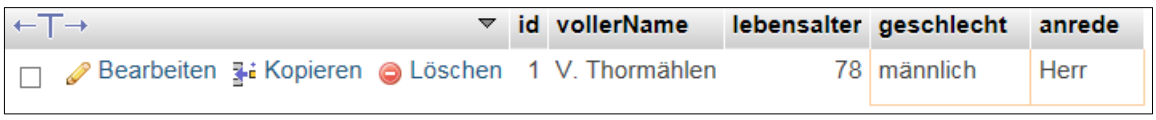

Abb. 26 :Ergebnis der vorstehenden einfachen SQL-Abfrage

5 Erstellte Tabellen und ihre Beziehungen

Ausführlich beschrieben wurde bisher, wie folgende vier Tabellen erzeugt und mit Werten versorgt werden können (siehe Abb. 27):

Abb. 27: Erstellte Tabellen und Beziehungen in der aktuellen MySQL-Datenbank

Die Tabelle '*personen*' (links) enthält zwei Spalten des Typs ENUM: '*geschlecht*' und '*anrede*'.
Die Tabelle *'personen_nt*' (rechts) besitzt eine N:1 Beziehung zu den Nachschlagetabellen '*geschlecht*' und '*anreden*' (siehe dazu die Beziehungslinien in Abb. 27)[4].

Im Gegensatz zur Tabelle '*personen_nt*' ist der Inhalt der Tabelle '*personen'* besser erkennbar. Bevor auf die Nachteile des Daten- bzw. Spaltentyps ENUM eingegangen wird, soll im nächsten Gliederungspunkt noch ein weiterer Vorteil hervorgehoben werden: Die automatische Erzeugung von Auswahllisten für HTML-Formulare mit Einträgen aus ENUM-Spalten.

[4] Verknüpfungen der Tabelle '*personen*' über Fremdschlüssel mit 2 dazugehörigen Nachschlagetabellen wurde erstellt mit dem Werkzeug '***Designer***', verfügbar in der Verwaltungs-Software '***phpMyAdmin***' für MySQL/MariaDB-Datenbanken.

6 Vor- und Nachteile des Spaltentyps ENUM

Dieser Spaltentyp hat folgende **Vorteile**:

- Kompakte Datenspeicherung in Situationen, in denen eine Tabellenspalte eine begrenzte Anzahl von Aufzählungswerten enthält und diese nie oder selten Änderungen unterliegen. Das gilt beispielsweise für die mehrmals erwähnte Spalte '*geschlecht*' der Tabelle '*personen*'. Es ist nicht zu erwarten, dass ein vierter Wert hinzukommt.
- Leicht lesbare Abfragen und Ausgaben. Die Zahlen werden zurück in die entsprechenden Zeichenfolgen in Abfrageergebnissen übersetzt.

Chris Komlenic [3]nennt drei Kriterien, die für Verwendung von ENUM sprechen (übersetzt v. Verf.):

1. Wenn Sie unterschiedliche, unveränderbare Wertemengen speichern …
2. Du wirst nie zusätzliche verwandte Informationen speichern müssen …
3. Die Aufzählungsliste enthält mehr als 2 und weniger als 20 Elemente.

Die **Nachteile** sind:

Der Spaltentyp ENUM ist ein zusammengesetzter Datentyp, der nicht dem Ideal der '*atomaren*' Datenspeicherung im Rahmen eines relationalen Datenbankentwurfs entspricht. Verständlich ist daher, dass die negative Kritik daran ziemlich umfangreich ist. Acht negative Kritikpunkte werden in dem bereits erwähnten Beitrag von Chris Komlenic [3]mit dem Titel

8 Reasons Why MySQL's ENUM Data Type Is Evil

geübt. Die Überschriften lauten (ebenda, übersetzt v. Verf.):

1. Daten werden nicht wie atomare Daten behandelt.
2. Das Ändern der Auflistungswerte von ENUM-Spalten ist aufwändig.
3. Es ist nicht möglich, zusätzliche Attribute oder zugehörige Informationen hinzuzufügen.
4. Eine Liste der verschiedenen ENUM-Werte zu erzeugen ist schwierig.
5. ENUM-Spalten haben nur begrenzte oder vernachlässigbare Auswirkungen auf die Optimierung.
6. Die Auflistungswerte einer ENUM-Spalte sind in anderen Tabellen nicht wieder verwendbar.
7. ENUM-Spalten haben bemerkenswerte Fehler.
8. ENUM ist nur eingeschränkt auf andere relationale Datenbanksysteme übertragbar.

Der erwähnte Beitrag [3] ist im Internet verfügbar, deshalb werden die Einzelheiten der 8 Kritikpunkte hier nicht wiedergegeben. Weitere Quellen zur Beurteilung des Spaltentyps ENUM sind im Literatur- und Quellenverzeichnis zu finden (siehe Seite IV).

Abbildungen

Listings

Tabellen

Literatur- und Quellenverzeichnis

[1] S. Hinz, „MySQL, Das offizielle Handbuch,“ verlag moderne industrie Buch AG & Co. KG, Bonn, 2002. [Online]. Available: http://www.theopenunderground.de/@pdf/ulinuz/mysql.pdf. [Zugriff am 30 08 2018].

[2] o. V., „HTML/PHP Populating Dropdown from ENUM values in Form,“ 09 06 2011. [Online]. Available: http://www.pcserviceselectronics.co.uk/php-tips/enum.php. [Zugriff am 30 08 2018].

[3] C. Komlenic, „8 Reasons Why MySQL's ENUM Data Type Is Evil,“ 02 03 2011. [Online]. Available: http://komlenic.com/244/8-reasons-why-mysqls-enum-data-type-is-evil/. [Zugriff am 26 08 2018].

[4] R. Bradford, „To enum or not to enum?,“ 22 01 2006. [Online]. Available: http://ronaldbradford.com/blog/to-enum-or-not-to-enum-2006-01-22/. [Zugriff am 30 08 2018].

[5] T. Klenke, „Database Enums,“ 20 09 2017. [Online]. Available: http://geekswithblogs.net/TimothyK/archive/2017/09/20/database-enums.aspx. [Zugriff am 30 08 2018].

[6] G. Richards, „The "What" of MySQL ENUMs (as well as the "Why"?),“ 27 03 2017. [Online]. Available: https://endertech.com/blog/the-what-of-mysql-enums-as-well-as-the-why. [Zugriff am 30 08 2018].

[7] o. V., „MySQL Sample Databases,“ 2012 10 01. [Online]. Available: SAMPE DATABASE http://www.ntu.edu.sg/home/ehchua/programming/sql/sampledatabases.html. [Zugriff am 2018 08 29].

[8] o. V., „A Comprehensive Guide to Using MySQL ENUM,“ o. J.. [Online]. Available: http://www.mysqltutorial.org/mysql-enum/. [Zugriff am 30 08 2018].

[9] o. V., „Kabinett bringt drittes Geschlecht "divers" auf den Weg,“ 15 08 2018. [Online]. [Zugriff am 26 08 2018].

[10] S. „Enum – Performance, Nutzen und Alternativen,“ o. J.. [Online]. Available: http://www.wissenstausch.com/2010/09/enum-performance-nutzen-und-alternativen/. [Zugriff am 29 08 2018].