Realisierung eines einfachen Datenmodells

Ziele:

- Anwendung des Clients phpMyAdmin
- Anwendung des Konsolenclients mysql.exe
- Datenimport von SQL-Statements
- Datenimport von CSV-Dateien

Es ist das folgende ERM gegeben:

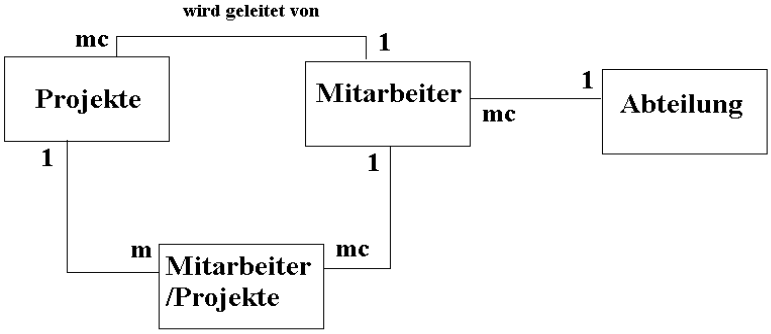


Tabelle: tbl_Projekte			
Feldname	**Datentyp**	**PS**	**FS**
ID_Projekt	INT/AI	✓	-
Projektbezeichnung	Text	-	-
Projektleiter_MA_FS	INT	-	✓

Tabelle: tbl_TT_Mitarbeiter_Projekte			
Feldname	**Datentyp**	**PS**	**FS**
MitarbeiterIDFS	INT	✓	✓
ProjekteIDFS	INT		✓

Tabelle: tbl_Mitarbeiter			
Feldname	**Datentyp**	**PS**	**FS**
ID_Mitarbeiter	INT/AI	✓	-
FamName	VARCHAR(50)	-	-
Vorname	VARCHAR(30)	-	-
FS_Abteilung	INT		✓

Tabelle: tbl_Abteilungen			
Feldname	**Datentyp**	**PS**	**FS**
ID_Abteilung	INT/AI	✓	-
AbtBezeichnung	Text	-	-

SQL-Code:

```
CREATE TABLE `Firma`.`tbl_Mitarbeiter` (
`ID_Mitarbeiter` INT NOT NULL AUTO_INCREMENT PRIMARY KEY ,
`FamName` VARCHAR( 50)NOT NULL ,
`Vorname` VARCHAR( 30)NOT NULL ,
`FS_Abteilung` INT NOT NULL
) ENGINE= InnoDB;
```

```
CREATE TABLE `Firma`.`tbl_Abteilungen` (
`ID_Abteilung` INT NOT NULL AUTO_INCREMENT PRIMARY KEY ,
`AbtBezeichnung` VARCHAR( 30)NOT NULL
) ENGINE= InnoDB;
```

```
CREATE TABLE `Firma`.`tbl_Projekte` (
`ID_Projekte` INT NOT NULL AUTO_INCREMENT PRIMARY KEY ,
`ProjektBezeichnung` VARCHAR( 50)NOT NULL ,
`FS_Mitarbeiter` INT NOT NULL
) ENGINE= InnoDB;
```

```
CREATE TABLE `Firma`.`tbl_TT_Mitarbeiter_Projekte` (
`FS_Mitarbeiter` INT NOT NULL ,
`FS_Projekte` INT NOT NULL ,
PRIMARY KEY (`FS_Mitarbeiter` ,`FS_Projekte`)
) ENGINE= InnoDB;
```

Systemumgebung:

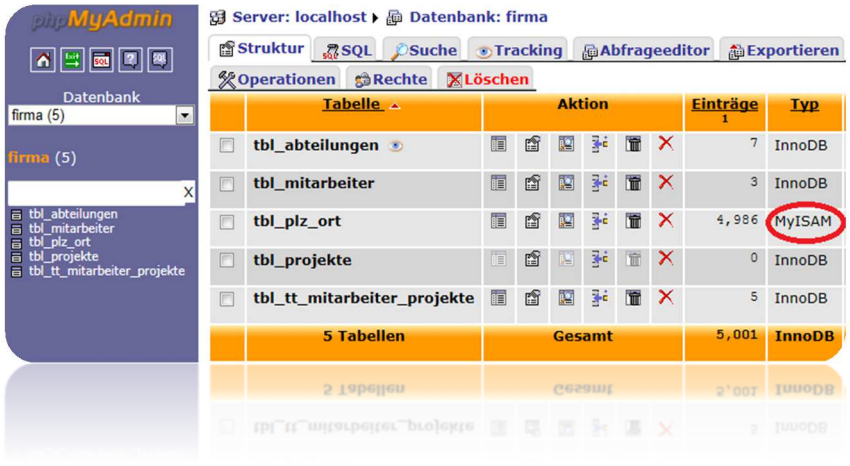
- Für die Erfassung der SQL-Befehle empfiehlt es sich einen **Editor** zu verwenden. Natürlich können Sie die Befehle auch in der Konsole eintippen, aber das ist auf Dauer recht mühsam, wenn Sie die Eingaben editieren müssen. Ausserdem sollten Sie die Skripte ebenfalls in einem Editor anschauen, bevor Sie sie importieren oder bearbeiten.
- **phpMyadmin** sollten Sie im Browser geöffnet habe.
- Das MySQL-**Konsolenfenster** (XAMPP-Verzeichnis/mysql/bin/mysql.exe) starten Sie mit root-Rechten: mysql.exe –u root –p

Aufgaben:

- Erstellen Sie die Datenbank („neue Datenbank anlegen“ → Firma), die Tabellen und die Attribute in MySQL; erstellen Sie ein oder zwei Tabellen über PHPMyAdmin und starten Sie dann das Konsolenfenster (mysql.exe) und erzeugen per SQL-Code die restlichen Tabellen. Den Quellcode zur Erzeugung finden Sie auch in der Datei *DDL_V2.txt*. Sie sollten ihn aber nicht einfach kopieren, sondern möglichst verstehen.
- Füllen Sie die Tabelle „tbl_Abteilungen“ mit Werten aus dem Textfile *Abteilungen.txt* ab (Importieren…). Es ist eine CSV-Datei. Das bedeutet zwar **C**omma**S**eparated**V**alue, die Werte in dieser Datei sind aber mit Semikolon getrennt (einstellbar). Experimentieren Sie mit den Werten des Primärschlüssels (vorgeben, nicht vorgeben, überschreiben…); achten Sie auf die Umlaute, importieren Sie die Datei mit verschiedenen Einstellungen mehrmals, löschen Sie den gesamten Inhalt der Tabelle (schauen Sie sich immer die entstandenen SQL-Befehle an). Öffnen Sie die Textdatei in einem Editor. Sie ist ANSI-codiert. Speichern Sie sie im utf-8-Format ab und importieren Sie sie neu. Jetzt sollten die Umlaute korrekt dargestellt werden.
- Als nächstes importieren wir eine Tabelle, die als SQL-Code vorliegt: tbl_plz_ort.sql. Also wieder „Importieren“, aber diesmal nicht wie oben CSV, sondern SQL. Damit haben wir ohne grossen Aufwand eine Tabelle mit allen Schweizer Ort- und Postleizahlkombinationen erhalten.

- Bevor wir weiterfahren, korrigieren wir noch eine Kleinigkeit: Beim Import sind zwei Indizes generiert worden (warum auch immer). Wir nutzen die Gelegenheit und schauen uns die Grösse der Datei tbl_plz_ort.MYI im Datenverzeichnis der Datenbanken an (…/datadir/firma – oder was auch immer in der my.ini definiert ist). Derzeitige Grösse vermutlich ca. 100 KB. Ein Index ist überflüssig – löschen Sie den Index *plz_ort_id* und kontrollieren Sie die Veränderung in der Grösse der MYI-Datei.

- Wenn Sie sich jetzt die Tabellentypen anschauen, werden Sie sehen, dass wir 4 InnoDB-Tabellen und eine MyISAM-Tabelle haben. Sie haben im Datenverzeichnis ja bereits die 3 zugehörigen Dateien gefunden. Für die InnoDB-Tabellen gibt es aber nur eine einzige Datei: *.frm. Jetzt ändern Sie auch für die Tabelle tbl_plz_ort den Tabellentyp auf InnoDB (Tabelle auswählen und den Reiter Operationen selektieren). Jetzt halt nochmal bei den Dateien nachschauen – irre, was!



- Wenn wir uns die Daten anschauen, sehen wir, dass die Textattribute in doppelten Anführungszeichen eingeschlossen sind. Das war so nicht geplant. Also leeren wir die Tabelle. ACHTUNG: leeren (SQL: truncate) und löschen (SQL: drop) ist ein Unterschied. Verwenden Sie für den nächsten Import eine aktuellere Version als CSV-Import: PLZ_ORT.csv. Anschliessend kontrollieren Sie noch, ob die französischen und deutschen Sonderzeichen korrekt dargestellt werden. Alternativ können Sie auch selber die aktuellsten Daten von der Schweizer Post verwenden. Es handelt sich dabei um eine gezippte Textdatei, die Sie zum Beispiel via Tabellenkalkulation bearbeiten können: überflüssige Spalten raus, Speicherung als csv-Text-Datei mit dem richtigen Zeichensatz (z. B. utf-8) und so weiter.

- Bis jetzt haben diese Daten mit unseren bisherigen Daten keinen Zusammenhang. Ziel ist es, die Tabellen tbl_mitarbeiter mit einem Fremdschlüssel zu erweitern, der auf den jeweiligen Wohnort der Mitarbeiter verweist. Erfassen Sie mindestens zwei oder drei Mitarbeiter bevor Sie weitermachen. Anschliessend erweitern Sie die Tabellenstruktur tbl_mitarbeiter um das Attribut FS_Wohnort mit dem Datentypen INT (integer). Falls Sie keine weiteren Angaben gemacht haben, sind die Zellen automatisch mit der Ziffer Null initialisiert worden.

- Um nicht alle Daten manuell erfassen zu müssen, können Sie jetzt wieder ein SQL-Skript verwenden, dass Ihnen etwas über tausend Personendatensätze zur Verfügung stellt: tbl_mitarbeiter.sql. Falls Sie Daten erfasst haben, die Sie behalten möchten, müssen Sie diese vorher sichern. Auf diesen Datenbestand werden wir jetzt unsere ersten SELECT-Übungen machen. Verwenden Sie hierzu das Skript oder ein Manual Ihrer Wahl aus dem Internet.

---

http://maczarr.de/wissen/ascii-ansi-utf-8-iso-unicode/ascii-ansi-utf-8-iso-unicode.php

Wenn man mit dem Computer und dem Internet arbeitet begegnet man des Öfteren der Möglichkeit, die Zeichencodierung auswählen zu können, so z.B. im Browser oder im Email-Programm. Die Wenigsten kennen überhaupt den Unterschied, daher stelle ich auf dieser Seite einmal die verschiedenen Codierungen vor.

Zeichensatzcodierungen sind im Grunde nur Tabellen, die bestimmten Byte-Werten konkrete Zeichen zuordnen. Anfangs waren dies nur alphanumerische Zeichen, heutzutage umfassen Zeichensätze z.B. auch Runen oder kyrillische Zeichen.

1. ASCII (standard): Der ASCII (American Standard Code for Information Interchange) entstand in den 60er Jahren, ist 7-Bit codiert, nutzt allerdings nur die ersten 128 Byte. Er umfasst alphanumerische, Sonder- und Steuerzeichen und ist von den verwendeten Zeichen her auf die englische Sprache ausgerichtet.
   Neben dem standard ASCII-Code gibt es noch den erweiterten ASCII-Code, bei dem nicht nur die ersten 128 Byte genutzt werden, sondern auch noch die "letzten" 128 Byte. Die Varianten des erweiterten ASCII-Codes nennt man *ANSI* und *ISO-8859*.
2. ANSI: ANSI ist eine Erweiterung des ASCII-Codes, er ist genormt vom **A**merican **N**ational-**S**tandards **I**nstitute (daher der Name) und hat sich als Standard auf den Windows- und Macintosh-Betriebssystemen durchgesetzt.
3. ISO-8859: Bei diesem Zeichensatz handelt es sich erstmal um eine Normserie mit länderspezifischen Zeichensätzen der ISO (International Organization for Standardization). Der Zeichensatz ISO-8859-1 (Latin) ist der erste von insgesamt 16 ländereigenen Zeichensätzen und ist für Westeuropa gemacht, beinhaltet also z.B. deutsche, französische oder skandinavische Sonderzeichen.
4. Unicode: Entstanden Ende der 80er Jahre mit dem Ziel alle Sprachen der Welt in einem Zeichensatz zu vereinen, ist der Unicode der größte und umfassendste Zeichensatz. Anfangs 16-Bit codiert, allerdings 2001 umgestellt auf 32-Bit beinhaltete Unicode 4 im Jahre 2003 ca. 100000 verschiedene Zeichen. Der Unicode vereint tote wie auch lebende Sprachen, so sind z.B. auch Runen Bestandteil.
5. UTF-8: Im Grunde eine Unicode-Variante mit hohem ASCII-Anteil, die Abkürzung bedeutet Unicode Transformation Format 8-Bit. Dieser Zeichensatz ist besonders im Internet weit verbreitet. Es ist angestrebt, dass er sich zum Standard entwickelt, indem alle neuen Internetkommunikationsprotokolle sich auf UTF-8 verstehen sollen. Da dies allerdings keine Verpflichtung, sondern nur eine Empfehlung oder "Bitte" ist, sieht es mit diesem Standard aktuell (März 2007) eher mäßig aus.

MySQL-Handbuch:

10.1. Zeichensätze und Sortierfolgen im Allgemeinen

Ein *Zeichensatz* ist eine Menge mit Symbolen und Kodierungen. Eine *Sortierfolge* ist ein Regelsatz für den Vergleich von Zeichen in einem Zeichensatz. Folgendes Beispiel, welches einen imaginären Zeichensatz verwendet, soll den Unterschied verdeutlichen.

Angenommen, wir haben ein Alphabet mit vier Buchstaben: '**A**', '**B**', '**a**', '**b**'. Jedem Buchstaben weisen wir nun eine Zahl zu: '**A**' = 0, '**B**' = 1, '**a**' = 2, '**b**' = 3. Der Buchstabe '**A**' ist ein Symbol, die Zahl 0 die ***Kodierung*** für '**A**'. Die Kombination aller vier Buchstaben und ihrer Kodierungen bildet den ***Zeichensatz***.

Angenommen, wir wollen zwei String-Werte '**A**' und '**B**' miteinander vergleichen. Die einfachste Möglichkeit, dies zu tun, besteht in einem Blick auf die Kodierungen: 0 für '**A**' und 1 für '**B**'. Da 0 kleiner als 1 ist, sagen wir, dass '**A**' kleiner als '**B**' ist. Was wir gerade getan haben, war die Anwendung einer Sortierfolge für unseren Zeichensatz. Die Sortierfolge ist eine Menge von Regeln – wenn auch in diesem Fall nur eine Regel: „Vergleiche die Kodierungen". Diese einfachste aller möglichen Sortierfolgen nennen wir *Binärsortierung*.

Was aber, wenn wir ausdrücken wollen, dass Klein- und Großbuchstaben äquivalent sind? Dann hätten wir schon zwei Regeln: (1) Betrachte die Kleinbuchstaben '**a**' und '**b**' als äquivalent zu den entsprechenden Großbuchstaben '**A**' und '**B**'. (2) Vergleiche die Kodierungen. Dies ist eine *Sortierfolge ohne Unterscheidung der Groß-/Kleinschreibung*. Sie ist ein kleines bisschen komplexer als eine Binärsortierung.

Im wirklichen Leben umfassen die meisten Zeichensätze viele Zeichen: nicht nur '**A**' und '**B**', sondern ganze Alphabete – manchmal sogar mehrere Alphabete oder fernöstliche Schreibsysteme mit Tausenden von Zeichen – sowie viele Sonder- und Interpunktionszeichen. Auch für Sortierfolgen gibt es im wirklichen Leben viele Regeln, die nicht nur die Behandlung der Groß-/Kleinschreibung angeben, sondern auch, ob Akzente und Tremata (die Punkte etwa auf den deutschen Umlauten Ä, Ö und Ü) unterschieden und wie Folgen aus mehreren Zeichen zugeordnet werden (beispielsweise die Regel, dass bei einer der beiden deutschen Sortierfolgen die Gleichung '**Ö**' = '**OE**' gilt).

MySQL erlaubt Ihnen

- das Speichern von Strings in einer Vielzahl von Zeichensätzen,
- das Vergleichen von Strings unter Verwendung einer Vielzahl von Sortierfolgen,
- das Mischen von Strings verschiedener Zeichensätze oder Sortierfolgen auf demselben Server, in derselben Datenbank oder sogar derselben Tabelle,
- die Angabe von Zeichensatz und Sortierfolge auf beliebiger Ebene.

In dieser Hinsicht ist MySQL den meisten anderen Datenbanksystemen weit überlegen. Allerdings müssen Sie, um diese Funktionen effizient nutzen zu können, wissen, welche Zeichensätze und Sortierfolgen verfügbar sind, wie Sie die Standardeinstellungen ändern und wie diese das Verhalten von String-Operatoren und -Funktionen beeinflussen.