Xampp
Administration und Grundlagen

Stefan Maihack Dipl. Ing. (FH)
Datum: 10.04.2015

Die wichtigsten Befehle zur Administration

- **Betriebssystem-Befehle**

→ cmd : Console starten
→ dir : Ordner anzeigen
→ cd : In einen Ordner wechseln
→ cd \ : Wechseln auf die oberste Ebene eines Ordners
→ cd <Ordner> : Wechseln in einen anderen Ordner
→ cd <Ordner1>\<Ordner2> : Wechsel in den Ordner2 von Ordner0 aus
→ cd .. : Wechseln in den oberen Ordner
→ <Laufwerksbuchstaben> : Wechseln in auf ein anderes Laufwerk

- **MySQL-Befehle**

→ mysql -h localhost -u root (aus Betriebsystem eingeben, um in MySQL zu wechseln)

- In MySQL

→ show databases; : Zeigt alle Datenbanken an
→ use <datenbankname>; : Verbindet sich mit einer Datenbank
→ show tables; : Zeigt alle Tabellen der verbundenen Datenbank
→ SELECT * FROM <Tabellenname>; : Zeigt den Inhalt einer Tabelle
→ DESC <Tabellenname>; : Tabellendefinition anzeigen

MySQL-Administration

- Um eine MySQL-Datenbank abzufragen, zu aktualisieren oder zu erstellen, wird das Kommandozeilenprogramm „mysql.exe“ benutzt. MySQL-Monitor
- Der MySQL-Monitor „mysql.exe“ steht unter:

  …\xampp\mysql\bin\mysql.exe

- Mit dem MySQL-Monitor ist es möglich, eine Verbindung zu jeder beliebigen MySQL-Datenbank im Internet aufzubauen.
- Folgende Parameter können dem MySQL-Monitor mitgegeben werden:

  -h <host> ➔ Der Monitor verbindet sich mit dem angegeben Host.
  -u <user> ➔ Der Monitor verbindet sich mit dem angegeben User.
  -p<passwort> ➔ Der Monitor benutzt das angegebene Passwort.
  -C ➔ Während der Verbindung werden die Daten komprimiert.
  -D <database> ➔ Der Monitor verwendet die angegebene Datenbank.
  -P <port> ➔ Der Monitor verwendet den angegebenen Verbindungsport.

MySQL-Administration

- Aufrufbeispiele:

Der Monitor verbindet sich lokal mit der Datenbank
…\xampp\mysql\bin\mysql –h localahost –u root

Der Monitor verbindet sich remote mit der Datenbank
…\xampp\mysql\bin\mysql –h www.mysql.com –u root –p<meinpasswort>

Der Monitor verbindet sich mit einer bestimmten Datenbank
…\xampp\mysql\bin\mysql –u root –D meineDatenbank

MySQL-Administration

- Folgende Steuerkommandos kennt der MySQL-Monitor:

show databases;	➔ Gibt alle erzeugten Datenbanken zurück. Es sollte mindestens die Datenbank MySQL angezeigt werden.
use <Datenbank>;	➔ Verwendet die Datenbank, welche angegeben wurde. Alle weiteren SQL-Kommandos beziehen sich darauf.
show tables;	➔ Zeigt alle erzeugten Tabellen der aktuellen Datenbank an.
desc <Tabelle>;	➔ Zeigt die Definition der Tabelle an.
drop database <Datenbank>;	➔ Löscht die angegebene Datenbank
quit; oder exit;	➔ Beendet den SQL-Monitor.
Show columns from <table>;	➔ Zeigt die Spalten einer Tabelle.
Show variables;	➔ Zeigt die Enviroment-Variablen des Systems
help;	➔ Zeigt die MySQL-Kommandos

MySQL-Administration

- Bevor Tabellen angelegt werden können, muss eine Datenbank angelegt werden:

  CREATE DATABASE [IF NOT EXISTS] <Datenbankname>;

- Durch den CREATE DATABASE-Befehl wird folgendes Verzeichnis für die Datenbank angelegt: „…\xampp\mysql\data\<Datenbankname>“.

- In diesem Verzeichnis entstehen dann folgende Files:

  ➔ NAME.FRM: Speichert die Tabellendefinition
  ➔ NAME.MYD: Enthält die Daten der Tabelle
  ➔ NAME.MYI: Enthält den Index der Tabelle

  Modifizieren sie diese Dateien niemals, es sei denn, sie benutzen ein Dienstprogramm wie den MySQL-Monitor oder Ähnliches.

SQL-Befehle
Allgemeines

- Was ist SQL (Structured Query Language)?
  SQL ist von IBM in Verbindung mit dem Datenbanksystem DB2 entwickelt worden. Erst als ORACLE diese Sprache für sein Datenbank einsetzte wurde SQL zum Standard erklärt.

- MySQL ist kompatibel mit dem SQL-Standard, wenn gleich MySQL zum einem nicht alle Elemente dieses Standards unterstützt und zum anderen zahlreiche Erweiterungen zu diesem Standard bietet.

- MySQL kann zum SQL92-Standard gezwungen werden, in dem der Dienst/Daemon mit „--ansi“ gestartet wird.

SQL-Befehle
SQL-Schreibweise

- SQL kann in drei Kategorien eingeteilt werden, die in der Literatur auch als selbständige Sprachen betrachtet werden:

  DDL ➔ Data Definition Language

  DML ➔ Data Manipulation Language

  DCL ➔ Data Control Language

- DDL: Enthält Befehle zur Erzeugung und Manipulation von Datenbankobjekten. Anlegen oder Bearbeiten von Datenbanken, Tabellen, Sichten oder Prozeduren z.B. CREATE, USE.
- DML: Enthält Befehle die auf die Daten einwirken, z.B. INSERT, UPDATE, DELETE, SELECT.
- DCL: (weniger umfangreich) Enthält Befehle zur Transaktionssteuerung.

SQL-Befehle
Schreibweise

- Eine SQL-Anweisung beginnt immer mit einem Verb, dass die Art der Anweisung festlegt. Solchen Verben sind: CREATE, SELECT oder UPDATE.
- Eine SQL-Anweisung endet mit einem Semikolon „;“.
- Zeichenketten stehen in Anführungszeichen – einfache oder doppelte.
- Es gibt reservierte Wörter: AS oder WHERE.
- Ist ein Attributname in einem Ausdruck nicht eindeutig – beispielsweise weil durch eine Verknüpfung mehrere Tabellen ein Attribut doppelt ist, kann dem Attributnamen der Tabellennamen durch einen Punkt getrennt voran gestellt werden.

  Z.B.: *sapr3.gehaltstabelle*

- Groß- und Kleinschreibung der Syntax wird nicht berücksichtigt.
- Groß- und Kleinschreibung in Tabellen- und DB-Namen wird nur dann berücksichtigt, wenn das darunter liegende Dateisystem diese berücksichtigt.
- Tabellennamen und Sonderzeichen – insbesondere Leerzeichen werden mit dem „Accent grave“ – Zeichen ` eingeschlossen.

SQL-Befehle
Schreibweise

- Im Allgemeinen wird empfohlen, reservierte Wörter nicht als Tabellen- oder Attributnamen zu verwenden. Ebenso sollten sie auf Sonder- und Leerzeichen in diesen verzichten.

- Der Ausdruck NULL steht für ein leeres Datum (NICHTS). NULL ist nicht zu verwechseln mit 0 oder „ “ (leere Zeichenkette).

- SQL-Befehle sollten in Großbuchstaben, Tabellen- und Attributnamen dagegen in Kleinbuchstaben geschrieben werden.

MySQL-Datentypen

- Der Datentyp bestimmt, welche Art von Operationen mit einem gespeicherten Wert zulässig sind.

- Numerische Typen:
  ➔ speichert Zahlen
  ➔ Operationen wie Addition, Subtraktion, Division und Multiplikation
  ➔ Numerische Datentypen verbrauchen wenig Speicherplatz, insbesondere Integer-Zahlen.
  ➔ Es gibt prinzipiell zwei Arten von Numerischen Datentypen: Integer-Zahlen, Fließkommazahlen

MySQL-Datentypen

- Integer-Zahlen

TINYINT	Ganzzahlen von -128 bis +127
TINYINT UNSIGNED	Ganzzahlen von 0 bis 255
SMALLINT	Ganzzahlen von -32768 bis 32767
SMALLINT UNSIGNED	Ganzzahlen von 0 – 65535
MEDIUMINT	Ganzzahlen von -8 Mio bis +8Mio
MEDIUMINT UNSIGNED	Ganzzahlen von 0 bis 16Mio
INT oder INTEGER	**Ganzzahlen von -2Mrd bis +2Mrd**
INT UNSIGNED	Ganzzahlen von 0 bis 4Mrd
BIGINT	Ganzzahlen von -9*1018 bis +9*1018
BIGINT UNSIGNET	Ganzzahlen von 0 bis 18*1018

MySQL-Datentypen

- Fließkommanzahlen

FLOAT (X)	➔ -3*1038 - +3*1038
FLOAT (M, D)	➔ **M=Stellen gesamt; D=Dezimalstellen**
DOUBLE	➔ Fließkommanzahl mit doppelter Genauigkeit -2*10308 - +2*10308
DOUBLE(M, D)	➔ wie Double
DECIMAL	➔ gepackte Fließkommazahl -999.99 - +999.99

MySQL-Datentypen

- Zeichenkettentypen

  CHAR (M) ➔ **Zeichenkette mit fester Länge und höchstens M Zeichen; M < 256**

  VARCHAR (M) ➔ **Zeichenkette mit variabler Länge M < 256**

  TINYTEXT, TINYBLOB ➔ Zeichenkette oder Blob bis 255 Zeichen

  TEXT, BLOB ➔ Zeichenkette oder Blob bis 65535 Zeichen

  MEDIUMTEXT, MEDIUMBLOB ➔ 16 Mio Zeichen

  LONGTEXT, LONGBLOB ➔ 4Mrd Zeichen

  ENUM (´wert1´, ´wert2´,…) ➔ Die Zelle kann einen Wert aus einer Liste bis zu 65535 Werten annehmen.

  SET (´wert1´, ´wert2´,…) ➔ Menge. Die Zelle kann ein oder mehrere Elemente aus einer Liste bis zu 64 Werten enthalten.

- Groß- und Kleinschreibung nur bei den Typen TINYBLOB, BLOB, MEDIUMBLOB, LONGBLOB
- Bei CHAR und VARCHAR den Zusatz BINARY angeben, um zwischen Groß- und Kleinschreibung zu unterscheiden.

MySQL-Datentypen

- Datumtypen für Datum und Zeit

DATE	➔ 01.01.1000 – 32.12.9999 im Format „YYYY-MM-DD“
TIME	➔ -838:59:59 - +838:59:59 im Format „HH:MM:SS“
DATETIME	➔ YYYY-MM-DD HH-MM-SS Kombination aus DATE und TIME
YEAR	➔ Jahreszahlen 1901 – 2155 Format „YYYY“
TIMESTAMP	➔ Unix-Zeit seit 01.01.1979 werden die Sekunden gezählt. Erlaubt das Speichern von Werten bis 2037. Format „YYYYMMDDHHMMSS“

MySQL-Tabellenadministration

- Erzeugen einer Datenbank
  Bevor Tabellen angelegt werden können, muss eine DB angelegt werden.

  CREATE DATABASE [IF NOT EXISTS] <db_name>;

  ➔ Im Dateiverzeichnis ..\XAMPP\mysql\data wird ein Unterverzeichnis mit dem Namen der Datenbank erzeugt „..\XAMPP\mysql\data\<db_name>“
- Connect zur Datenbank
  Nun muss zur erzeugten Datenbank eine Verbindung aufgebaut werden

  …\xampp\mysql\bin\mysql –u root
  show databases;
  use <db_name>;

MySQL-Tabellenadministration

- Nun kann in mit der verbundenen Datenbank eine Tabelle angelegt werden

  CREATE [TEMPORARY] TABLE [IF NOT EXIST] <tabellenname>
  [(Tabellendefinition,…)] [Tabellenoptionen] [SELECT-Befehl]

  TEMPORARY: erzeugt ein flüchtige Tabelle. Sie besteht nur solange die Verbindung zur Datenbank besteht.

  Tabellenoptionen: Angabe des Tabellentyps.
  ➔ BDM: Berkeley Tabellen, die transaktionssichere Sperren auf Tabellen unterstützen.
  ➔ GEMINI: Transaktionssichere Sperren auf Datensatzebene.
  ➔ HEAP: Tabellen im Hauptspeicher erzeugen.
  ➔ ISAM: Die original MySQL-Tabellen.
  ➔ InnoDB: Transaktionssichere Sperren auf Datensatzebene.
  ➔ MERGE: Vereinigung mehre physikalischer MyISAM Tabellen zu einer logischen Tabelle.
  ➔ **MyISAM**: Ersetzt ISAM-Tabellen und ist Standard seit MySQL 3.23.xx

MySQL-Tabellenadministration

Eigenschaften des MyISAM-Tabellentyps

- → Gute Kompatibilität aufgrund der Little-Endian-Architektur.
- → Maximal 64 Indizies
- → Index sind maximal 16 Spalten möglich
- → Eine AUTO_INCREMENT-Spalte pro Tabelle
- → NULL-Werte sind in indizierten Spalten zulässig. Hierzu werden 0 bis 1 Byte pro Schlüssel gebraucht.
- → MyISAM-Tabellen sind nicht transaktionssicher

Was ist Transaktionssicherheit?

In der Informatik bezeichnet man eine Transaktion, als Folge von Programmschritten, die als eine logische Einheit betrachtet werden, weil sie den Datenbestand nach fehlerfreier und vollständiger Ausführung in einem konsistenten Zustand hinterlassen.

Fazit: Die Tabellentypen „InnoDB, Gemini und BDB sollen dann eingesetzt werden, wenn die Konsistenz der Datenbank auf keinen Fall gefährdet werden darf

MySQL-Tabellenadministration

- Beispiel: CREATE TABLE

  CREATE TABLE strassen (ID INT NOT NULL AUTO_INCREMENT,
  ~~kartenausschnitt INT NOT NULL,~~
  strassenname VARCHAR(50) NOT NULL,
  plz VARCHAR(5) NOT NULL,
  ort VARCHAR (50) NOT NULL,
  ~~hausnummern VARCHAR(5) NOT NULL,~~
  ~~xkorrd INT NOT NULL,~~
  ~~ykorrd INT NOT NULL,~~
  PRIMARY KEY (ID));

- Ein externes Script aufrufen, innerhalb des MySQL-Monitors

  \. d:\sql\<Scriptname>.sql

MySQL-Tabellenadministration

- Löschen einer Tabelle

  DROP <Tabellenname>;

- Tabellendefinition anzeigen

  DESC <Tabellenname>;

- Umbenennen einer Tabelle

  RENAME TABLE <Tabellenname_alt> TO <Tabellenname_neu>
  [,<Tabellename_alt> TO <Tabellename_neu>,…];

MySQL-Tabellenadministration

- Spalten/Attribute einer Tabelle hinzufügen

  ADD <Spaltendefinition> [FIRST | AFTER <Spaltenname>];
  ➔ FIRST: Die neue Spalte wird am Anfang hinzugefügt.
  ➔ AFTER: Die Spalte wird nach dem angegebenen Spaltenname hinzugefügt.

- Eine Spalte/Attribut ändern

  CHANGE <Spaltenname> <Spaltendefinition>;
  ➔ Ändert den Datentyp und den Spaltennamen

  MODIFY <Spaltendefinition>;
  ➔ Ändert nur den Datentyp der Spalte

- Ein Attribut löschen
  ALTER TABLE <Tabellenname> DROP COLUMN <Spalte>;

MySQL - Referenzhandbuch

- https://dev.mysql.com/doc/refman/5.1/de/

Übung 1 - MySQL

- Aufgabenstellung: Erzeugen sie eine Strassentabelle mit folgenden Spalten:

  ➔ StrassenID
  ➔ Kartenausschnitt
  ➔ Strassenname
  ➔ Postleitzahl
  ➔ Name der Stadt
  ➔ und einen primären Schlüssel

- Lösung:
- DROP DATABASE IF EXISTS stadt;
  CREATE DATABASE stadt;
  CREATE TABLE strassen (strassenID INT NOT NULL AUTO_INCREMENT,
  kartenausschnitt INT UNSIGNED NOT NULL,
  strassenname VARCHAR(50) NOT NULL,
  plz VARCHAR(5) NOT NULL,
  ort VARCHAR (50) NOT NULL,
  ~~hausnummern VARCHAR(5) NOT NULL,~~
  ~~xkorrd INT NOT NULL,~~
  ~~ykorrd INT NOT NULL,~~
  PRIMARY KEY (strassenID));

Übung 2 - MySQL

- Fügen sie in die eben angelegt Tabelle 3 Datensätze ein. Die Datensätze sollen Umlaute enthalten.

  Beispiel:
  INSERT INTO strassen (kartenausschnitt, strassenname, plz)
  VALUES (1234, ´Au in den Buchen´, ´76646´);

  oder

  INSERT INTO strassen
  VALUES (1234, ´Au in den Buchen´,´76646´);

  oder

  INSERT INTO strassen
  SET kartenausschnitt = 1234, strassenname = ´Au in den Buchen´, plz = ´76646´;

Übung 3 - MySQL

- Erzeugen sie eine Bewohnertabelle mit folgenden Spalten:
  - ➔ bewohnerID
  - ➔ Vorname
  - ➔ Nachname
  - ➔ Postleitzahl
  - ➔ Stadt
  - ➔ Straße
  - ➔ Hausnummer
  - ➔ Familienstand
  - ➔ Eigentum
  - ➔ strassenID
- Lösung:

```
CREATE TABLE bewohnertab (bewohnerID INT NOT NULL AUTO_INCREMENT,
                          vorname VARCHAR NOT NULL,
                          nachname VARCHAR NOT NULL,
                           plz VARCHAR(5) NOT NULL, ort VARCHAR (50) NOT NULL,
                          strassenID INT NOT NULL,
                          hausnummern VARCHAR(5) NOT NULL,
                          familienstand VARCHAR(10) NOT NULL,
                          eigentum INT NOT NULL,
                         PRIMARY KEY (bewohnerID)
                         FOREIGN KEY (strassenID) REFERENCES (strassen));
```


-

Übung 4 - MySQL

- Fügen sie in die eben angelegt Tabelle 3 Datensätze ein. Die Datensätze sollen Umlaute enthalten.

  Beispiel:
  INSERT INTO strassen (kartenausschnitt, strassenname, plz)
  VALUES (1234, „Au in den Buchen“, „76646“);

Übung 5 - MySQL

- Erstellen sie eine MySQL-Loaderdatei für diese beiden Tabellen.
  Beispiel:
- 003 **DROP** DATABASE **IF EXISTS** sample**;**
- 005 **CREATE** DATABASE **IF NOT EXISTS** sample
- 006 **DEFAULT CHARACTER SET** utf8 **COLLATE** utf8_unicode_ci**;**
- 008 **USE** sample**;**

- 056 **DROP TABLE IF EXISTS** s_r**;**
- 058 **CREATE TABLE IF NOT EXISTS** s_r **(**
- 059 s_id INT2**,**
- 060 r_id INT2**,**
- 061 seq INT2 UNSIGNED **DEFAULT NULL,**
- 062 spa INT4 **DEFAULT NULL,**
- 063 dur **TIME DEFAULT NULL,**
- 064 **PRIMARY KEY (**s_id**,** r_id**),**
- 065 **FOREIGN KEY (**s_id**) REFERENCES** stops**(**id**),**
- 066 **FOREIGN KEY (**r_id**) REFERENCES** routes**(**id**)**
- 067 **);**