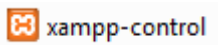
Management of Information Systems

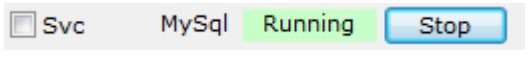
Tutorat: Session 3

Einführung in SQL

Setup

1. XAMPP-Console starten 

in `C:\xampp\`
    1. Apache starten bis 

erscheint
    2. MySQL starten bis 

erscheint
    3. An den Rechnern im PC Pool sollten diese beiden Dienste schon automatisch gestartet sein, aber zur Sicherheit nachschauen

2. Über http://localhost/phpmyadmin oder über das XAMPP-Menü kommt ihr in den phpMyAdmin
    - Dieses Programm benutzen wir heute um SQL zu lernen

Achtung: Skype muss vor den obigen Schritten geschlossen sein

Was ist SQL?

- SQL steht für **S**tuctured **Q**uery **L**anguage und ist eine sehr verbreitete Datenbanksprache.
- Mit SQL kann man Datenbank und deren Daten erstellen, löschen, verändern und ansehen
- Innerhalb der Datenbank werden die Daten in Relationen gespeichert, die man gut als Tabellen visualisieren kann

Turnier	Jahr	Sieger
French Open	2009	Roger Federer
US Open	2009	Juan Martin Del Potro
Australian Open	2010	Roger Federer
French Open	2010	Rafael Nadal

Was ist SQL?

- SQL-Anweisungen lassen sich in 3 Kategorien unterteilen:
  - **Data Definition Language (DDL)**

    Anlegen, Ändern und Löschen von Datenbanken und Relationen, Namen und Datentyp der Spalten festlegen, etc. (Struktur)

  - **Data Manipulation Language (DML)**

    Abfragen, Einfügen, Ändern und Löschen von Tupel, also Inhalten in den Tabellen (Inhalt)

  - **Data Control Language (DCL)**

    Datenzugriffskontrolle (Zugriff; für uns nicht relevant)

Datentypen

- Jede Tabellenspalte hat einen festgelegten Datentyp
  - **Ganzzahlen**

`int` bzw. `integer`	-2^{31} bis $2^{31} - 1$
`bigint`	-2^{63} bis $2^{63} - 1$
`smallint`	-2^{15} bis $2^{15} - 1$
`tinyint`	0 bis 255

  - **Gleitkommazahlen**

`decimal(p,s)` *bzw.* numeric(p,s)	p: Länge, s: Dezimalstellen
float(m)	m: maximale Dezimalstellen
real	Abhängig vom Datenbanksystem

Datentypen

- **Zeichenketten**

`char(n)`	Zeichenkette mit gleich Länge $n \leq 255$
`varchar(n)`	Zeichenkette mit Länge von bis zu $n \leq 255$
`text`	Text mit Länge von bis zu 65 535 Zeichen

- **Logische Werte**

`boolean`	*true* oder *false*

- **Sonstige**

`date`	Datum in dem Format *YYYY-MM-DD*
`time`	Uhrzeit im Format *HH:MM:SS*
`datetime`	Zeitstempel im Format *YYYY-MM-DD HH:MM:SS*

SQL-Befehlsübersicht

- Es folgt eine Aufzählung gebräuchlicher SQL-Befehle
- Nicht erschrecken, es sind einige
- Diese sind ***klausurrelevant***, allerdings könnt ihr für die Tutorate die Befehlsübersicht benutzen
- Wenn man damit einige Aufgaben bearbeitet, findet man sich schnell zurecht
- Tipp: zahlreiche Tutorials im Internet helfen euch das Gelernte zu vertiefen (siehe Referenzen auf letzter Folie)
- Ohne praktische Erfahrung wird es euch schwer fallen alles zu verstehen.

SQL Befehlsübersicht

- Erstellung einer neuen Datenbank

  CREATE DATABASE <Name der Datenbank>

- Erstellung einer neuen Relation

  CREATE TABLE <Name der Relation>

  (<Spaltenname$_1$> <Datentyp> (<Länge>), …,

  <Spaltenname$_n$> <Datentyp> (<Länge>))

- **AUTO_INCREMENT** (Automatische Erhöhung des Werts**)**
- **PRIMARY KEY** (Festlegung einer Spalte als Primärschlüssel)
- **UNIQUE** (Unterbindet die Erstellung von Duplikaten)
- **NULL** (Leeren Datenfeldern wird der Wert Null zugewiesen)
- **NOT NULL** (Der Inhalt der Datenfelder muss sich von Null unterscheiden)
- **DEFAULT** (Leeren Datenfeldern wird ein Standardwert zugewiesen)

SQL Befehlsübersicht

- Änderung des Spaltennamens, Datentyps und/oder der Speicherlänge

  ALTER TABLE <Name der Relation> **CHANGE** <Spaltenname$_{alt}$> <Spaltenname$_{neu}$> <Datentyp> (<Länge>)

- Hinzufügen einer Spalte vor bzw. nach einer bereits bestehenden Spalte

  ALTER TABLE <Name der Relation> **ADD** <Spaltenname$_{neu}$>

  <Datentyp> (<Länge>) **FIRST / AFTER** <Spaltenname>

- Löschen einer Spalte

  ALTER TABLE <Name der Relation> **DROP** <Spaltenname>

- Hinzufügen bzw. Löschen eines Primärschlüssels

  ALTER TABLE <Name der Relation> **ADD / DROP**

  PRIMARY KEY (<Spaltenname>**)**

- Löschen einer Relation oder Datenbank

  DROP TABLE/DATABASE <Name der Relation/Datenbank>

- Löschen aller Tupel in einer Relation

  TRUNCATE TABLE <Name der Relation>

SQL Befehlsübersicht

- Einfügen von Inhalten in Datenfelder bzw. Tupel

  INSERT INTO <Tabellenname>(<Spaltenname>*, [etc.]*) **VALUES** (<Wert für die erste Spalte>*, [etc.]*)

  Zeichenketten werden in Hochkommata gestellt, Bsp. 'Jeans'

- Auswahl aller Datenfelder einer bestimmten Tabellenspalte

  SELECT <spaltenname(n)> **FROM** <tabellenname(n)>

- Auswahl aller Datenfelder in bestimmten Tabellen

  SELECT * **FROM** <tabellenname(n)>

- Auswahl von Datenfeldern unter Berücksichtigung einer Bedingung

  SELECT <spaltenname(n)> **FROM** <tabellenname(n)> **WHERE** <Bedingung>

- Auswahl von Datenfeldern mit Hilfe einer Bedingungsliste

  SELECT <spaltenname(n)> **FROM** <tabellenname(n)>

  WHERE <Bedingung1> **AND/OR** <Bedingung2> *[etc.]*

SQL Befehlsübersicht

- Auswahl von Datenfeldern ohne Duplikate

  SELECT DISTINCT <spaltenname(n)> **FROM** <tabellenname(n)>

- Sortierte Ausgabe von Datenfeldern

  SELECT <spaltenname(n)> **FROM** <tabellenname(n)>

  ORDER BY <spaltenname(n)> **ASC/DESC**

  ASC = ansteigend (Standardausgabe), **DESC** = absteigend

- Verwendung alternativer Spaltentitel

  SELECT <spaltenname> **AS** <alias> *[etc.]* **FROM** <tabellenname(n)>

- Aktualisierung von Datenfeldern

  UPDATE <Tabellenname> **SET** <Feldänderungen>

  WHERE <Bedingungsliste>

- Bedingtes Löschen von Tupeln

  DELETE FROM <Tabellenname> **WHERE** <Bedingungen>

SQL Funktionen

- Zählen von Zeilen

  SELECT **COUNT**(<spaltenname>) FROM <Tabellenname>
- Ermittlung eines Durchschnitts

  SELECT **AVG**(<spaltenname>) FROM <Tabellenname>
- Ermittlung einer Standardabweichung

  SELECT **STD**(<spaltenname>) FROM <Tabellenname>
- Ermittlung eines Maximalwertes

  SELECT **MAX**(<spaltenname>) FROM <Tabellenname>
- Ermittlung einer Summe von Werten

  SELECT **SUM**(<spaltenname>) FROM <Tabellenname>

SQL Befehlsübersicht

- Um Ergebnisse zu gruppieren (insbesondere unter Verwendung von Aggregationsfunktionen wie COUNT und SUM) kann bzw. muss das GROUP BY Statement verwendet werden.

  SELECT <Name of column>, **SUM/AVG/COUNT/etc.**(<Name of column>) **FROM** <Name of table>

  WHERE <Condition>

  GROUP BY <Name of column>;

- Um die Anzahl der ausgegeben Datensätze zu begrenzen kann das Schlüsselwort LIMIT verwendet werden.

  LIMIT <Grenze>

  Beispiel: **SELECT** * **FROM** tabelle **LIMIT** 10

SQL einfache Tabellenverknüpfung

- Um zwei Tabellen einfach miteinander zu verknüpfen, können die gemeinsamen Spalten der Tabellen mittels WHERE-Bedingung verknüpft werden.

tabelle1

KundenId	Vorname	Nachname
1	Bernd	Becker
2	Christina	Checker

tabelle2

Auftrag	KundenId	Datum
1	1	02.10.2012
2	2	19.12.2012
3	2	10.02.2013

Beispiel:

```
SELECT * FROM tabelle1, tabelle2
WHERE tabelle1.KundenId = tabelle2.KundenId;
```

KundenId	Vorname	Nachname	Auftrag	Datum
1	Bernd	Becker	1	02.10.2012
2	Christina	Checker	2	19.12.2012
2	Christina	Checker	3	10.02.2013

SQL Operatoren

Die folgenden Operatoren können innerhalb einer WHERE Anweisung stehen:

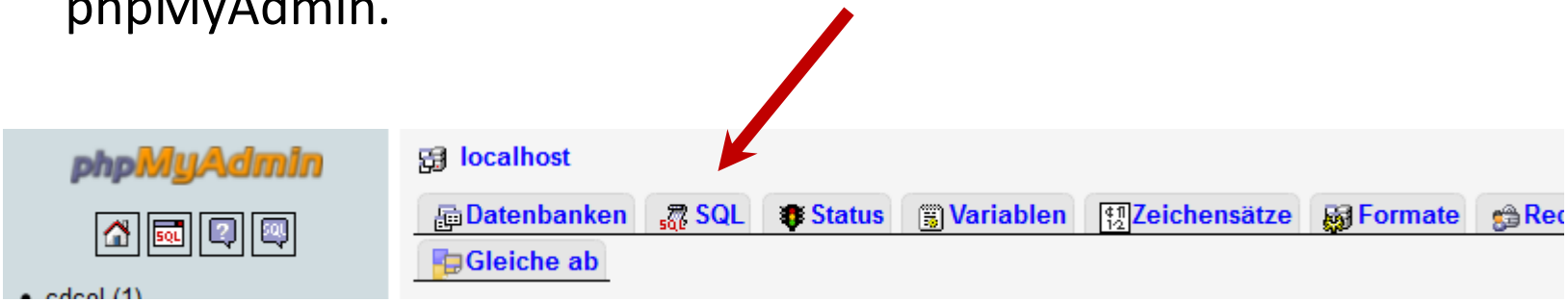
- **=** ist gleich
- **<>** ist ungleich
- **>** ist größer als
- **<** ist kleiner als
- **>=** ist größer oder gleich als
- **<=** ist kleiner oder gleich als
- **BETWEEN** liegt innerhalb eines bestimmten Bereiches
- **LIKE** wird zum Suchen von Mustern verwendet
- **IN** bzw. **NOT IN** um mehrere gültige/ungültige Werte zu spezifizieren

Arbeiten mit phpMyAdmin

- phpMyAdmin bietet für alles, was wir jetzt lernen eine WYSIWYG-Oberfläche.

- Da wir für die zukünftigen Tutorate aber SQL verstehen müssen, benutzen wir diese allerdings nur zum nachprüfen.

- Stattdessen benutzen wir das Eingabefeld für SQL-Befehle im phpMyAdmin.



Arbeiten mit phpMyAdmin

- Fall 1) Wir haben unsere Datenbank noch nicht erstellt.

  Dann auf der phpMyAdmin-Startseite auf den SQL-Link (siehe vorherige Folie) klicken und dort die Datenbank erstellen.

- Fall 2) Wir haben unsere Datenbank schon erstellt.

  Dann zuerst auf die Datenbank im linken Auswahlmenü klicken und dort auf den Link zur SQL-Eingabe. Nun können alle weiteren Befehle eingegeben werden.

  Immer darauf achten, dass ihr in der richtigen Datenbank seid.

Beispielaufgabe Nachtleben (1)

- Erstelle die Datenbank Nachtleben
- Erstelle die Tabelle Locations, mit den Spalten: Zaeler, Name, Mindestalter und einer Spalte die eine Identifizierung als Bar oder Club angibt. Zaehler soll der Primärschlüssel sein und sich mit jedem neuen Eintrag um 1 erhöhen.

- `CREATE DATABASE Nachtleben;`
- `CREATE TABLE Locations (Zaehler INT(2) PRIMARY KEY AUTO_INCREMENT, Name VARCHAR(20), Mindestalter VARCHAR(2), Bar VARCHAR(1));`

Beispielaufgabe Nachtleben (2)

- Füge einige Bars und Clubs zu der Tabelle Locations hinzu, achte darauf, dass mindestens eine Bar und ein Club sowie das Mindestalter 16 und 18 vorhanden sind

- Gebe alle Locations aus bei denen das Mindestalter 18 ist

- `INSERT INTO Locations (Name, Mindestalter, Bar) VALUES(‘Cheers‘,16,1);`
- `INSERT INTO Locations (Name, Mindestalter, Bar) VALUES(‘Stusi‘,18,0);`
- `INSERT INTO Locations (Name, Mindestalter, Bar) VALUES(‘Karma‘,21,0);`

- `SELECT * FROM Locations WHERE Mindestalter = 18;`

Beispielaufgabe Nachtleben (3)

- Durch eine Gesetzesänderung muss jeder Club das Mindestalter auf 21 erhöhen und jede Bar auf 18. Update deine Tabelle
- Löscht zuerst die Tabelleninhalte und dann die Tabelle.
- Löscht die Datenbank.

- `UPDATE Locations SET Mindestalter = 18 WHERE Bar = 1;`
- `UPDATE Locations SET Mindestalter = 21 WHERE Bar = 0;`

- `TRUNCATE Locations;`
- `DROP TABLE Locations;`
- `DROP DATABASE Nachtleben;`

Beispielaufgabe Tutorat

- Erstellt die Datenbank *Tutorat.*
- Erstellt die Tabelle *Teilnehmer* mit den Spalten *Zaehler, Vorname, Nachname, Lieblingsfarbe.* Zaehler soll der Primärschlüssel sein und sich mit jedem neuen Eintrag um 1 erhöhen.

- Fügt willkürlich ein paar Teilnehmer des Tutorats in die Tabelle ein. Achtet darauf, dass mindestens zwei die Lieblingsfarbe Blau haben.

- Gebt den Vornamen aller Teilnehmer aus, deren Lieblingsfare Rot ist.
- Gebt alle Tutoratsteilnehmer aus, die die Lieblingsfarbe Blau haben, alphabetisch aufsteigend nach Nachname.
- Gebt die Anzahl der Tutoratsteilnehmer aus und nennt die Spalte *Teilnehmeranzahl*.
- Gebt eine Liste mit allen unterschiedlichen Lieblingsfarben und deren Anzahl aus.
- Gebt Vor- und Nachnamen des Teilnehmers aus, der zuletzt hinzugefügt wurde (den größten Zähler hat). Nehmen sie dabei an, dass der größte Zähler zunächst nicht bekannt ist.

- Löscht zuerst die Tabelleninhalte und dann die Tabelle.
- Löscht die Datenbank.

Beispielaufgabe Tutorat Lösung

- `CREATE DATABASE Tutorat;`
- `CREATE TABLE Teilnehmer (Zaehler INT(2) PRIMARY KEY AUTO_INCREMENT, Vorname VARCHAR(20), Nachname VARCHAR(20), Lieblingsfarbe VARCHAR(20));`

- `INSERT INTO Teilnehmer (Vorname, Nachname, Lieblingsfarbe) VALUES('Bernd','Becker','Blau');`
- `INSERT INTO Teilnehmer (Vorname, Nachname, Lieblingsfarbe) VALUES('Christina','Checker','Blau');`
- `INSERT INTO Teilnehmer (Vorname, Nachname, Lieblingsfarbe) VALUES('Doreen','Drechsler','Rot');`

- `SELECT * FROM Teilnehmer WHERE Lieblingsfarbe = 'Blau' ORDER BY Nachname ASC;`
- `SELECT Vorname FROM \`teilnehmer\` WHERE Lieblingsfarbe = 'Rot';`
- `SELECT COUNT(*) AS Teilnehmeranzahl FROM \`teilnehmer\`;`
- `SELECT Lieblingsfarbe, COUNT(*) FROM \`teilnehmer\` GROUP BY Lieblingsfarbe;`
- `SELECT Vorname, Nachname FROM \`teilnehmer\` ORDER BY Zaehler DESC LIMIT 1;`

- `TRUNCATE Teilnehmer;`
- `DROP TABLE Teilnehmer;`
- `DROP DATABASE Tutorat;`

Referenzen

- http://www.w3schools.com/sql/default.asp
- http://www.sqlcourse.com/intro.html
- en.wikipedia.org/wiki/SQL