Relationale Datenbanken und SQL

Forschungsabend zum Thema:

Einführung in SQL anhand der freien Datenbanksoftware MySQL

Dieter Schicker
Gewilab
(dieter.schicker@uni-graz.at)

27.4.2006

Das relationale Datenbankmodell

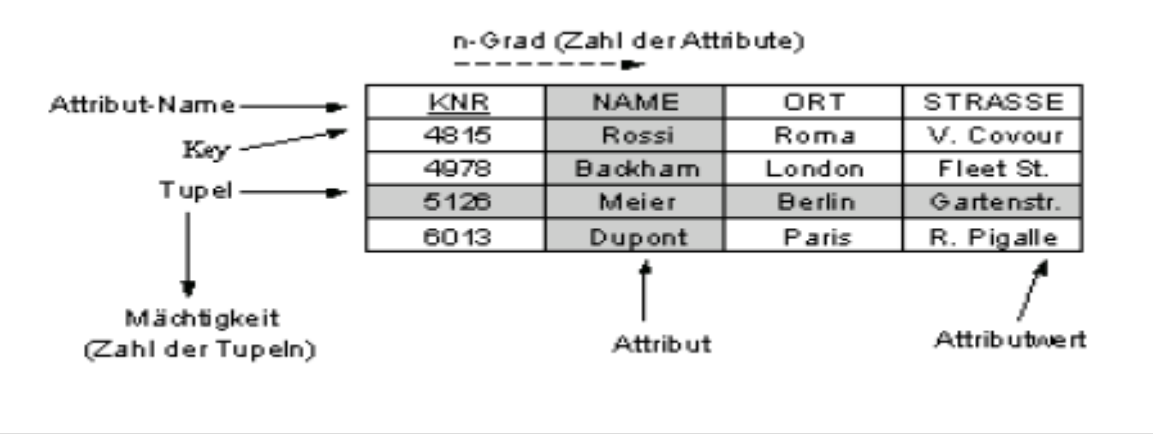
- In relationalen Datenbanken werden Daten in Tabellenform, so genannten Relationen gespeichert.
- Zwischen den Relationen können Beziehungen definiert werden. Wobei abhängig von der Anzahl der in Verbindung stehenden Datensätze verschieden Beziehungsarten existieren (1:1, 1:n, m:n).
- Die graphische Darstellung der Relationen und der zugehörigen Beziehungen erfolgt meist über ein Entity-Relationship-Modell (ERM), wobei die Beziehungen über Linien dargestellt werden, die mit Kardinalitäten versehen sind.

Relationale Datenbankmanagementsysteme

- Ein relationales Datenbankmanagementsystem (DBMS) bietet Funktionen zur Verwaltung der Datenbank udn steuert alle Zugriffe auf die Daten. Als Blackbox betrachtet nimmt ein DBMS BenutzerInnenanfragen sowie Aufträge zur Datenänderung entgegen und sorgt für die Erledigung der Aktionen.
- Beispiele für Datenbankmanagementsysteme: postgreSQL, MySQL, Sybase, Oracle, Microsoft SQL Server

Grundbegriffe

- Relation (Tabelle): Sammlung aller zugehörigen Datensätze (Tupeln)
- Tupel: Einzelner Satz einer Relation (Zeile) aus Attributen zusammengesetzt
- Attribut: Teil (Feld) eines Tupels
- Schlüssel: Eindeutige Charakterisierung der Tupel einer Relation, aus einem oder mehreren Attributen zusammengesetzt (<=>Primärschlüssel).



Normalisierung

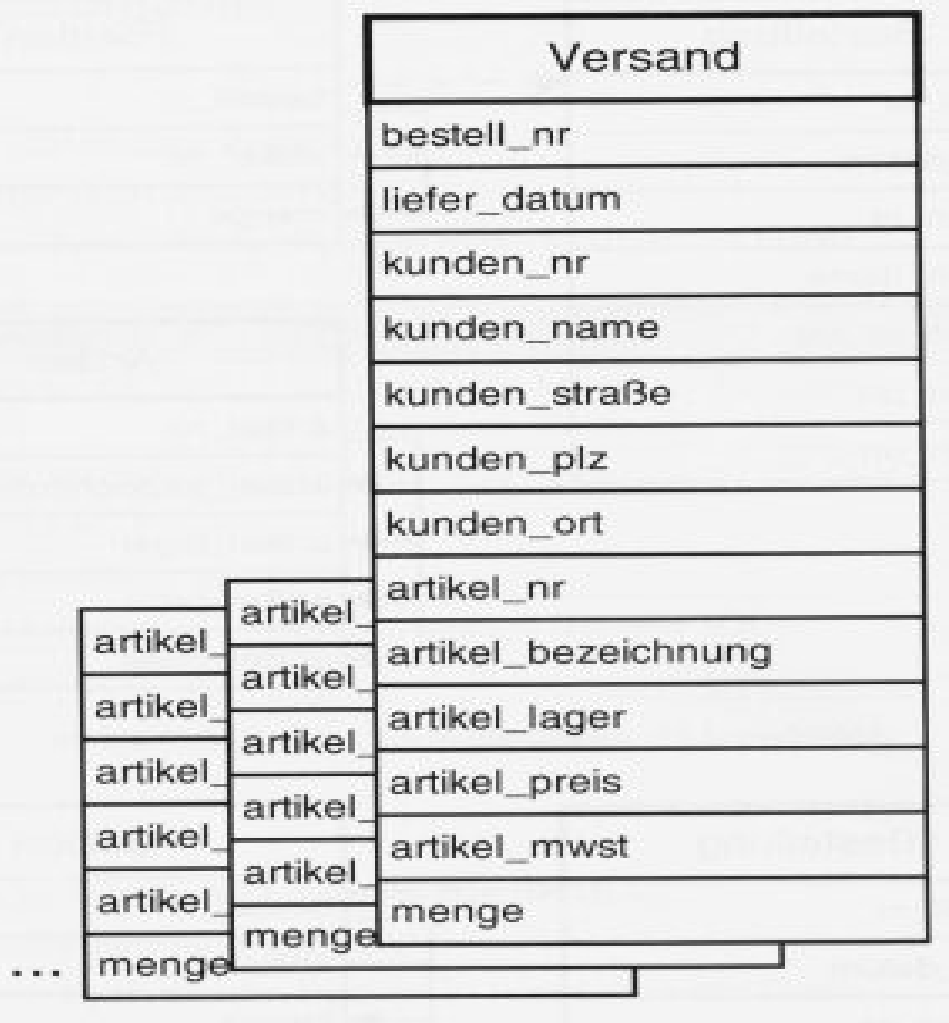
Unter Normalisierung versteht man die Überführung komplexer Beziehungen (Tabellen) in einfache Beziehungen durch Aufteilung der Attribute einer Tabelle auf mehrere Tabellen. Ziel sind stabile und flexible Datenstrukturen.

- [Nullte Normalform: Tabelle ist unnormalisiert
- 1. Normalform (1NF): Alle Attribute sind atomar (d.h. an jeder Reihen- und Spaltenposition steht genau ein Wert)
- 2. Normalform (2NF): Relation ist in der 1. Normalform und "funktional abhängige" Attribute werden zu eigenen Relationen zusammengefasst.
- 3. Normalform (3NF): Alle Relationen sind in der 2. Normalform

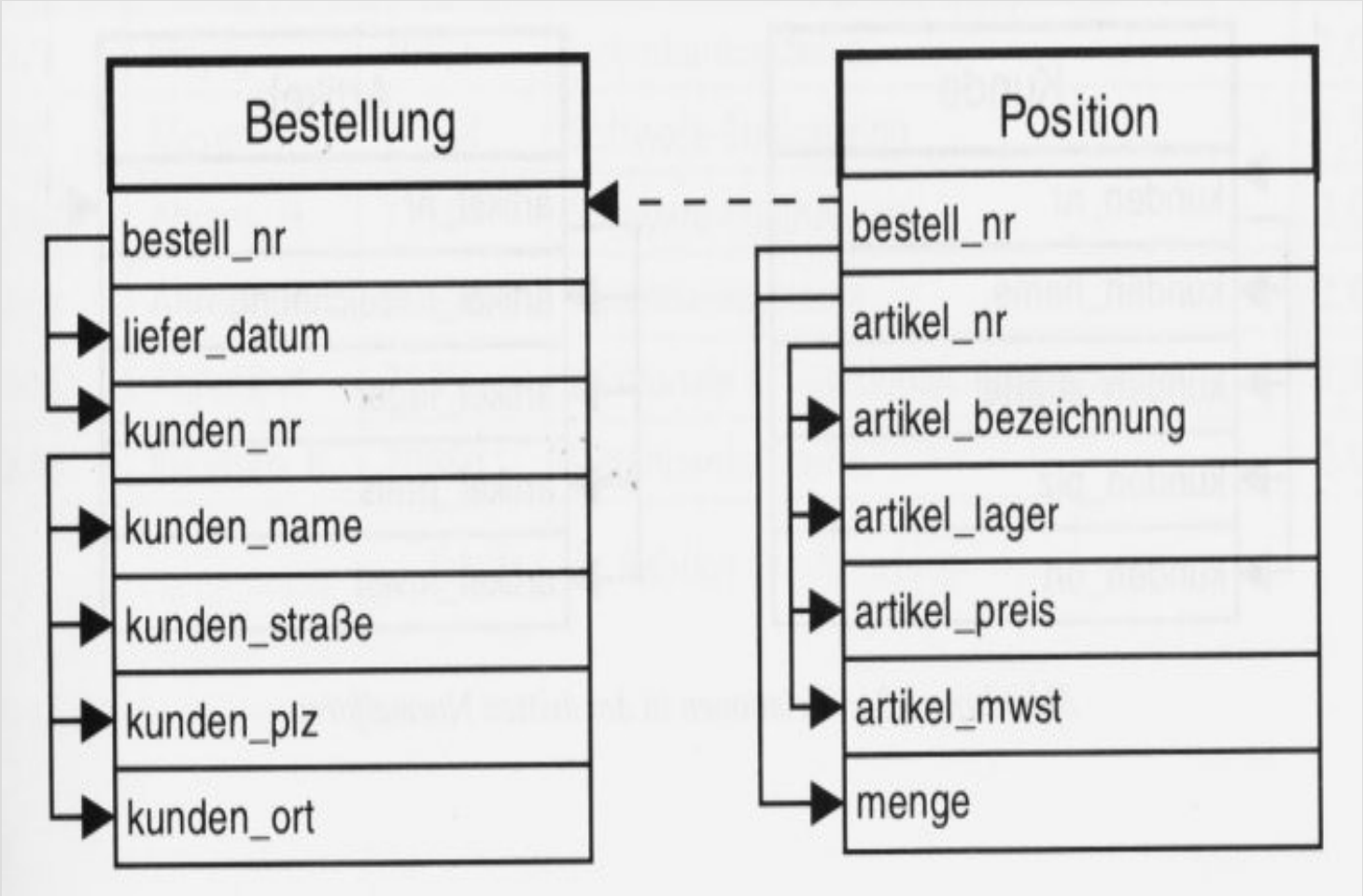
Beispiel: Nullte Normalform (1)

Bestell		Kunden					Artikel						Menge
Nr	Datum	Nr	Name	Straße	PLZ	Ort	Nr	Pa-ckung	Bezeich-nung	La-ger	Preis	MwSt	
151	02.05.2000	101	Stein, Peter	Moordamm 34	23863	Kayhude	G002	0,5 l	Portwein	7	12,45	2	4
							G003	6er Pack	Bier	7	5,20	2	3
							K002		Hose	2	112,80	2	3
							K003	Karton	Damenhut	2	65,70	2	1
							L002	125 g	China-Tee	5	8,35	1	5
152	02.05.2000	103	Randers, Nis	Am Seeufer 12	23845	Oering	K001	Karton	Schuhe	2	98,50	2	10
							K003	Karton	Damenhut	2	65,70	2	2
							K004	Karton	Sonnen-brille	2	76,00	1	12

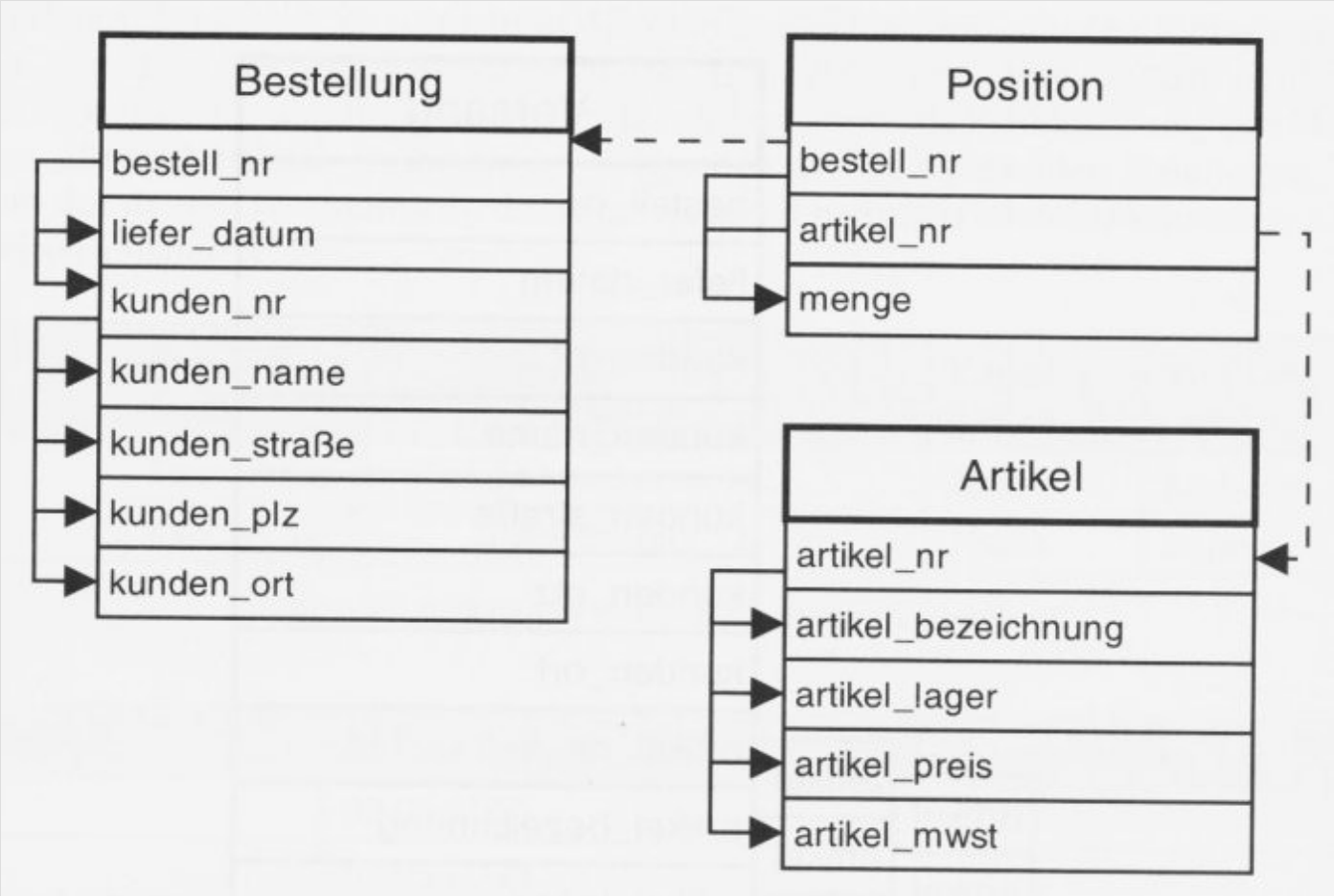
Beispiel: Nullte Normalform (2)



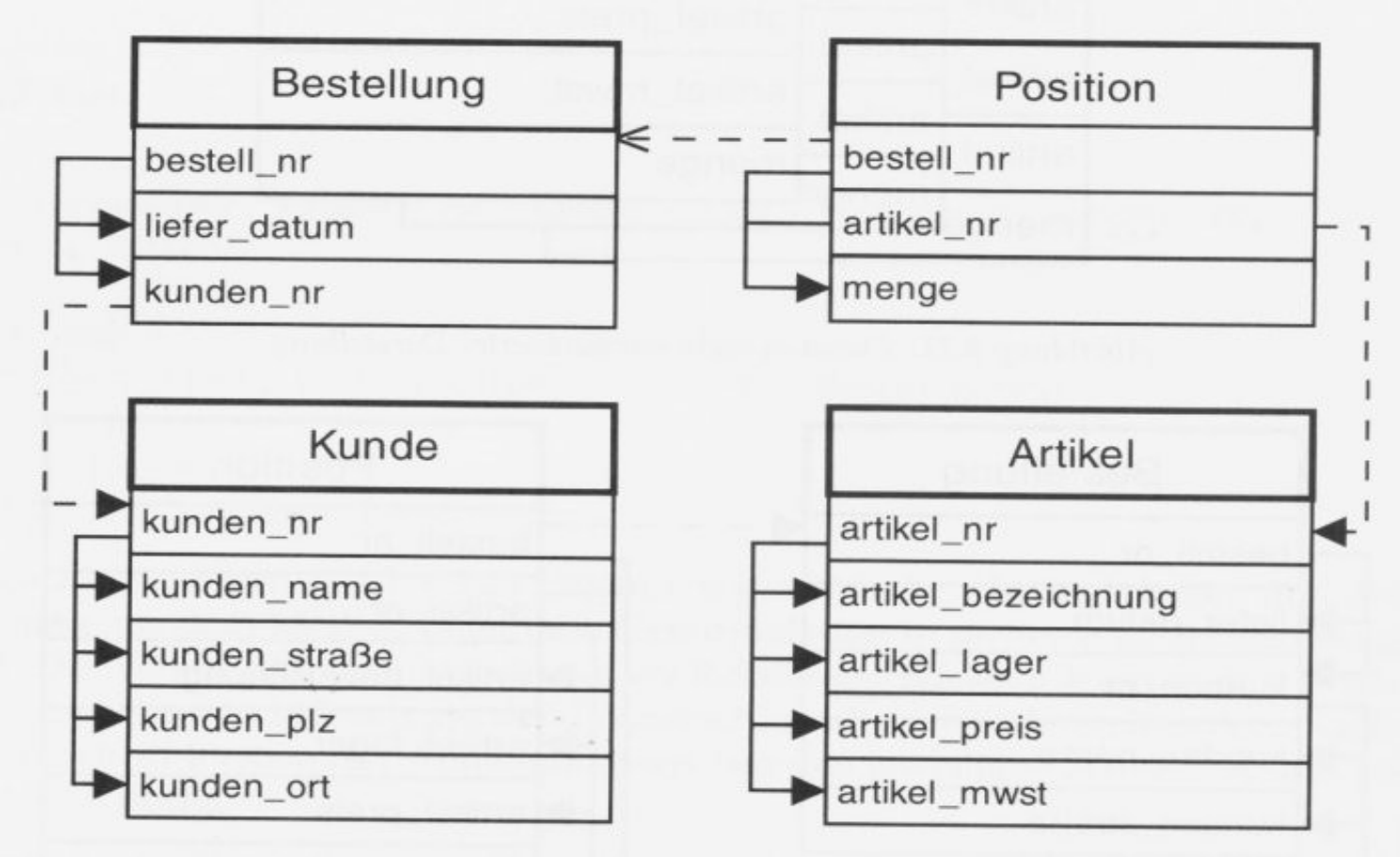
Beispiel: 1. Normalform



Beispiel: 2. Normalform



Beispiel: 3. Normalform



Basisfunktionen von Datenbanksprachen

- Anlegen neuer Relationen und Tupel
- Verändern von Relationen und Tupeln
- Löschen von Relationen und Tupeln
- Erzeugen von Relationen aus vorhandenen Relationen
- Auffinden (Suche) von Tupeln in Relationen

SQL

- SQL steht für Structured Query Language und wird als standardisierte Schnittstelle zu relationalen Datenbanken verwendet. SQL ist ursprünglich nach dem ANSI (American National Standards Institute) – Standard genormt, in weiterer Folge ein ISO-Standard geworden. SQL-Anweisungen werden sowohl zur Daten-Abfrage und -veränderung, als auch zur Datendefinition verwendet. SQL unterstützt eine deskriptive (beschreibende) Vorgehensweise, d.h. der/die BenutzerIn spezifiert, was er haben will, nicht wie es zu ermitteln ist. Man spricht daher auch von einer funktionalen Programmiersprache.

Geschichte von SQL

1970: Als DB-Sprache des relationalen DB-Prototyps SYSTEM R von IBM entwickelt
1986: SQL-Normierung durch ANSI: ANSI-Standard (X3. 135-1986)
1987: Übernahme des ANSI-Standards durch ISO: ISO-Standard (ISO 9075-1987)
1989: Erweiterung des SQL 1-Standards: ISO-Standard (ISO 9075-1989)
1992: SQL 2-Standard: ISO -Standard ISO 9075 - 1992

SQL-Befehlübersicht

- CREATE: Definition von Relationen, Views ...
- ALTER: Ändern von Relationen
- GRANT: Vergabe von Zugriffsrechten
- SELECT: Datenauswahl, Abfrage
- INSERT: Hinzufügen von Tupel
- UPDATE: Verändern von Tupeln
- DELETE: Löschen von Tupeln

SQL-Datentypen

- VARCHAR(n): Zeichenkette mit maximal n Zeichen
- SMALLINT: Ganzzahlige Werte zw. -32768 u. 32767
- INTEGER: Ganzzahlige Werte zw. -2147483648 u. 2147483647
- REAL: Fließkommazahlen
- BOOLEAN: boolsche Werte (true/false)
- TEXT: Zeichenketten in beliebiger Länge
- SERIAL: Autoinkrementierter BIGINT-Wert
- DATE: Datumsangaben
- u.v.a.m

SQL - CREATE

- CREATE TABLE Tabellenname (Attributsname Datentyp, Attributsname Datentyp, ...) PRIMARY KEY (Attributname));

Beispiel:

```
CREATE TABLE adressen (
    id              SERIAL,
    familienname    VARCHAR(128),
    vorname         VARCHAR(128),
    PRIMARY KEY (id)
);
```

SQL - INSERT

- INSERT INTO Tabellenname (Attributsname1, Attributsname2, ...) VALUES (Wert1, Wert2, ...);

Beispiel:

```
INSERT INTO adressen
(familienname, vorname)
VALUES ('Schicker',
'Dieter');
```

SQL - SELECT

- SELECT * FROM Tabellenname WHERE Bedingung ORDER BY Attributsname, Attributsname, ...
- SELECT Attributsname, Attributsname ... FROM Tabellenname WHERE Bedingung ORDER BY Attributsname, Attributsname, ...

```
SELECT * FROM Adressen;
SELECT familienname FROM Adressen ORDER BY familienname;
SELECT * FROM Adressen WHERE familienname = 'Schicker';
SELECT * FROM Adressen WHERE familienname LIKE 'Sch_cker';
SELECT * FROM Adressen WHERE familienname LIKE 'S%';
```

SQL – UPDATE

- UPDATE Tabellenname SET Attributsname1=Wert1, Attributsname2=Wert2 WHERE Bedingung;

Beispiel:

```
UPDATE adressen SET familienname="Mayer"
WHERE familienname="Schicker";
```

SQL - DELETE

- DELETE FROM Tabellenname WHERE Bedingung;

Beispiel:

DELETE FROM Adressen WHERE familienname LIKE 'M%';

MySQL

- URL: http://dev.mysql.com
- MySQL ist ein Open-Source-RDBMS ("Relational database management system")
- Server-Client-Anwendung
- Notwendige Dowloads (von obigem URL):

  * MySQL5 Community Edition – Database Server and Client
  * MySQL Administrator
  * MySQL Query Browser
  * (optional:) MySQL Workbench

MySQL - Server-Installation

Installieren: Setup aufrufen und folgende Einstellungen vornehmen:

* "Typical"
* => Install
* "Detailed Configuration
* Developer Machine
* Multifunctional Database
* C: Installation Path
* Decision Support (DSS)/OLAP
* Enable TCP/IP Networking: YES, Port: 3306
* Enable StrictMode: YES
* Default Character Set auswählen
* Install As Windows Service: YES
* root password setzen
* create an anonymous account: NO

MySQL - Tools

- MySQL-Administrator: Tool zum Anlegen von Datenbanken, Usern, Tabellen-Schemata
- MySQL-Query-Browser: Tool zum Abfragen der Datenbank
- phpMyAdmin für browserbasierte Administration von MySQL-Datenbanken (siehe "http://ginko.uni-graz.at/phpmyadmin")

MySQL-Beispiel Database-Creation

- Erzeugen Sie eine Datenbank mit folgenden Tabellen und Feldern:

Tabelle: Person
Felder:

* **id (NOT NULL, AUTOINCREMENT)**
* **vorname VARCHAR(128) NN**
* **familienname VARCHAR(128) NN**
* **adresse VARCHAR(128)**
* **plz VARCHAR(10)**
* **ort VARCHAR(128)**
* **geburtsdatum DATE**
* **geburtsort VARCHAR(128)**
* **institution_id INT**

Tabelle: Institution
Felder:

* **id (NOT NULL, AUTOINCR.)**
* **name VARCHAR(256) NN**
* **adresse VARCHAR(128)**
* **plz VARCHAR(10)**
* **ort VARCHAR(128)**
* **land VARCHAR(128)**

MySQL-Beispiel Database-Creation

```
CREATE TABLE Person (
      id  INTEGER NOT NULL AUTO_INCREMENT,
      vorname VARCHAR(128) NOT NULL,
      familienname VARCHAR(128) NOT NULL,
      adresse VARCHAR(128),
      plz VARCHAR(10),
      ort VARCHAR(128),
      geburtsdatum DATE,
      geburtsort VARCHAR(128),
      institution_id INTEGER
);

CREATE TABLE Institution (
      id INTEGER NOT NULL AUTO_INCREMENT,
      name VARCHAR(256) NOT NULL,
      adresse VARCHAR(128),
      plz VARCHAR(10),
      ort VARCHAR(128),
      land VARCHAR(128) );
```

MySQL - Beispiel INSERT-Operationen

```
INSERT INTO Institution (name, adresse, plz,
ort, land) VALUES ("Gewilab",
"Merangasse 70", "8010", "Graz", "Austria");

INSERT INTO Person (vorname, familienname,
adresse, plz, ort, geburtsdatum,
geburtsort, institution_id) VALUES ("Dieter",
"Schicker", "Lendplatz 98",
"8020", "Graz", "1970-11-14", "Bruck/Mur", 1);

INSERT INTO Person (vorname, familienname,
adresse, plz, ort, geburtsdatum,
geburtsort, institution_id) VALUES ("Peter",
"Langmann", "Ankerstrasse 24", "8054",
"Graz", "1969-04-24", "Graz", 1);
```

MySQL - Beispiel UPDATE/DELETE-Operationen

```
UPDATE Person SET vorname="Christian" WHERE familienname="Schicker";

UPDATE Institution SET name="INIG" WHERE name="Gewilab";

DELETE FROM Person WHERE familienname LIKE 'L%';
```

MySQL - Beispiel Queries

```
SELECT * FROM Person;

SELECT vorname, familienname, ort FROM Person;

SELECT vorname, familienname FROM Person ORDER BY
familienname;

SELECT vorname, ort, geburtsdatum FROM Person WHERE
familienname LIKE "L%" ORDER BY vorname;

SELECT * FROM Person p, Institution i WHERE p.institution_id = i.id
ORDER BY familienname;

SELECT  familienname, vorname, name, i.adresse, i.ort FROM Person p,
Institution i WHERE p.institution_id = i.id AND vorname LIKE 'Christ%'
ORDER BY familienname;
```

Relationale Datenbanken und SQL

Danke für Ihr Interesse

und:

Verwenden Sie Open-Source-Produkte! ;-)