



create\_inse  
rt\_update...

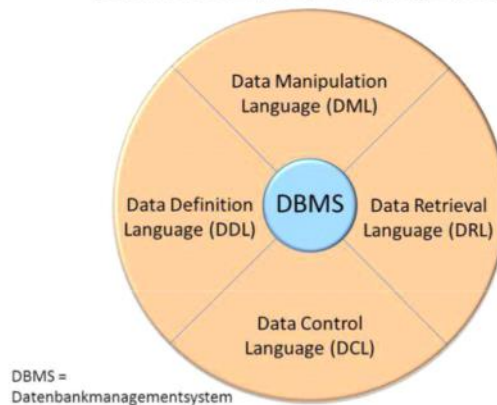
## 5 SQL-DDL

SQL Standard.

Teilsprachen.

SQL ist eine Sprache zur Bearbeitung und Auswertung von relationalen Datenbanken. Sie umfasst drei Bereiche:

### Structured Query Language (SQL)



- Data Definition Language (DDL): Befehlssatz zum Anlegen, Ändern und Löschen von Datenbanken, Tabellen usw. und ihren Strukturen.
- Data Manipulation Language (DML): Befehlssatz zum Einfügen, Ändern, Löschen und Auslesen von Daten aus den Tabellen.
- Data Control Language (DCL): Befehlssatz zur Administration von Datenbanken

Anders als bei imperativen Programmiersprachen wie C#, C++, Java oder Pascal wird durch die Befehle nicht die Art und Weise bestimmt, wie man ein Ergebnis erhält; es wird kein Algorithmus implementiert. Vielmehr sagt man, was man haben möchte, und der Datenbankserver ermittelt das Ergebnis. Solche Arten von Programmiersprachen nennt man *deklarativ*.

Obwohl es viele SQL-Dialekte gibt, ist der offizielle SQL-Standard in vielen Systemen implementiert und garantiert eine Wiederverwendbarkeit oder Übertragbarkeit der Befehle.

1986 wurde der erste SQL-Standard vom ANSI verabschiedet, der 1987 von der ISO ratifiziert wurde. 1992 wurde der Standard überarbeitet und als SQL-92 (oder auch SQL2) veröffentlicht. Alle aktuellen Datenbanksysteme halten sich im Wesentlichen an diese Standardversion. Die Version SQL:1999 (ISO/IEC 9075:1999, auch SQL3 genannt) ist noch nicht in allen Datenbanksystemen implementiert. Das gilt auch für die nächste Version SQL:2003. Der aktuelle Standard wurde 2008 unter SQL:2008 verabschiedet.

## 5.1 Anlegen/Löschen einer Datenbank

Wir gehen davon aus, dass die Datenbank komplett neu erstellt werden soll. Der Befehl zum Anlegen einer Datenbank ist CREATE SCHEMA. Damit werden allerdings noch keine Tabellen angelegt, sondern nur die diese umfassende Datenbank.

Bibliothek

```
--SQL 92:
CREATE SCHEMA datenbankname
    [ DEFAULT CHARACTER SET zeichensatz]

--Mysql:
CREATE { DATABASE | SCHEMA } [ IF NOT EXISTS ] datenbankname
    [[ DEFAULT ] CHARACTER SET zeichensatz]
    [ COLLATE sortierung]
;
```

### 5.1.1 Zuweisen eines CharacterSets

Was ist ein Zeichensatz? Auf dem Computer werden Buchstaben, Ziffern, Satz- und Sonderzeichen durch Zahlen kodiert. So ist beispielsweise der Buchstabe A im ASCII-Zeichensatz die Zahl 0x41 und a die Zahl 0x61. Da für die Kodierung des ASCII nur ein Byte (= 8 Bit) zur Verfügung steht, können nur 256 verschiedene Zahlen zur Kodierung verwendet werden. Die ersten 128 sind im Wesentlichen die Steuerzeichen (wie z.B. der Zeilenumbruch), das Leerzeichen, die lateinischen Buchstaben, die Ziffern 0 – 9, Satz- und einfache Sonderzeichen. Die restlichen 128 wurden mehr oder weniger willkürlich dazu verwendet, Umlaute oder andere sprachspezifische Sonderzeichen abzubilden. Und hier fing das Unglück an. Fast jeder Computer- oder Betriebssystemhersteller hat da sein eigenes Süppchen gekocht. So ist beispielsweise das Zeichen Ü im Zeichensatz ISO/IEC 8859-1 mit der Zahl 0xDC kodiert und in Codepage 850 mit 0x9A. Wird nun ein Text unter Windows erfasst, wird das Ü als 0xDC in die Datei geschrieben. Öffnet man nun diese Datei mit einem COMMAND-Editor wie EDIT, so erscheint aber ein anderes Zeichen und umgekehrt. Dieses Problem und die Beschränkung auf 256 Zeichen, was die Darstellung z.B. ostasiatischer Schriften unmöglich macht, haben dazu geführt, dass man eine neue, leicht erweiterbare Kodierung von Schriftzeichen baute. Unicode ward geboren! Unicode selbst liegt in verschiedenen Formatierungen vor. Derzeit gerne verwendet werden utf8, utf16 und utf32. Welche Zeichensätze von Ihrem Server unterstützt werden, können Sie leicht mit SHOW CHARACTER SET herausfinden.

### 5.1.2 Sortierreihenfolge

Für jede Sprache gibt selbst bei gleichen Zeichensätzen oft mehrere Arten, die Texte wie z.B. für eine Namensliste zu sortieren. In MySQL wird die Sortierreihenfolge über die Option **COLLATE** im CREATE SCHEMA festgelegt.

Die verfügbaren Sortierungen lassen sich leicht mit SHOW COLLATION anzeigen, wobei schnell deutlich wird, dass es mehrere Sortierreihenfolgen für einen Zeichensatz geben kann. Betrachten wir die Sortierreihenfolgen für den Zeichensatz cp850:

```
mysql> SHOW COLLATION LIKE 'cp850%';
+-----+-----+-----+-----+-----+-----+
+
+ | Collation          | Charset | Id | Default | Compiled | Sortlen |
+-----+-----+-----+-----+-----+-----+
+
+ | cp850_general_ci   | cp850   | 4  | Yes     | Yes      | 1        |
+ | cp850_bin          | cp850   | 80 |         | Yes      | 1        |
+-----+-----+-----+-----+-----+-----+
+
+
2 rows in set (0.00 sec)
```

Bei cp850\_general\_ci wird nicht und bei cp850\_bin wird zwischen Groß- und Kleinschreibung unterschieden.

### 5.1.3 Löschen einer Datenbank

Das Löschen einer Datenbank erfolgt analog zum Erstellen mit dem Befehl **DROP SCHEMA**

```
--SQL92
DROP SCHEMA datenbankname
[ CASCADE | RESTRICT ]
;
--MySQL
DROP { DATABASE | SCHEMA } [ IF EXISTS ] datenbankname;
```

Mit Hilfe von CASCADE bzw. RESTRICT werden in der Datenbank existierende Tabellen entweder gleich mitgelöscht bzw. das Löschen der Datenbank wird verhindert, solange diese noch Tabellen aufweist.

## 5.2 Anlegen einer Tabelle

```
CREATE TABLE tabellenname (
    spaltenspezifikation
    [, spaltenspezifikation]*
    [, PRIMARY KEY (spaltenliste)]
) [tabellenoptionen];
```

Der erste Teil des Befehls ist selbsterklärend. Danach kommt der Tabellenname, den wir der Namenskonvention entsprechend klein schreiben. Was ist aber eine *spaltenspezifikation*? Eine spaltenspezifikation besteht aus drei Teilen:

1. Spaltenname: Er wird klein geschrieben und ergibt sich aus dem ER-Modell.
2. Datentyp: Dieser legt fest, was für eine Art von Information in der Spalte verwaltet und wie diese kodiert wird. Mögliche Datentypen finden Sie in Abschnitt 25.1 auf Seite 371.
3. Zusätze: Mit diesen kann man eine Spalte ausführlicher bestimmen. Eine Liste möglicher Zusätze finden Sie weiter unten.

Das aus der Notation für reguläre Ausdrücke entnommene Sternchen \* hinter der optionalen zweiten Spaltenspezifikation bedeutet: eine beliebige Anzahl viele, also auch 0.

```
use database artikel;
CREATE TABLE adresse (
    adresse_id INT UNSIGNED AUTO_INCREMENT,
    strasse VARCHAR(255),
    hnr VARCHAR(255),
    lkz CHAR(2),
    plz CHAR(5),
    ort VARCHAR(255),
    deleted TINYINT UNSIGNED NOT NULL DEFAULT 0,
    PRIMARY KEY (adresse_id)
);
```

## Beispiele

```

drop database if exists employee; (1)
create database employee;

use employee; (2)

create table department (3)
(
    departmentID int not null auto_increment primary key,
    name varchar(20)
) type=InnoDB;

create table employee
(
    employeeID int not null auto_increment primary key,
    name varchar(80),
    job varchar(15),
    departmentID int not null
    references department(departmentID)
) type=InnoDB; (4)

create table employeeSkills
(
    employeeID int not null references employee(employeeID),
    skill varchar(15) not null,
    primary key (employeeID, skill) (5)
) type=InnoDB;

create table client
(
    clientID int not null auto_increment primary key,
    name varchar(40),
    address varchar(100),
    contactPerson varchar(80),
    contactNumber char(12)
) type=InnoDB;

create table assignment
(
    clientID int not null references client(clientID),
    employeeID int not null references employee(employeeID),
    workdate date not null,
    hours float,
    primary key (clientID, employeeID, workdate)
) type=InnoDB;

```

1. Diese Anweisung stellt fest, ob die DB schon existiert und löscht sie gegebenenfalls. Die Anweisung ist u.U. mit Vorsicht zu genießen.
2. Mit Hilfe von CREATE wird die Datenbank erstellt und durch USE zur aktuellen DB erklärt.
3. Die Tabelle verfügt über 2 Spalten, departmentID als Primärschlüssel und name als Abteilungsnamen. Als Tabellentyp ist InnoDB angegeben.
  - departmentID: Datentyp ist int (Integer)
  - not null; die Spalte muss in jeder Zeile einen Wert haben
  - auto-increment; der Wert wird von MySQL automatisch hochgezählt
  - primary key: Die Spalte ist der Primärschlüssel der Tabelle
  - Name: Die Spalte kann maximal 20 alphanumerische Zeichen aufnehmen (varchar(20))
  - Die Spalten werden in Form einer kommagetrennten Liste geführt. Üblicherweise wird immer eine Objektbeschreibung je Zeile geschrieben. Der SQL-Interpreter führt die Zeile erst aus, nachdem er ein Semikolon (;) gefunden hat.
4. Mit Hilfe des Schlüsselwortes REFERENCES weist man die Spalte departmentID als Fremdschlüssel aus. Sie verweist auf den Primärschlüssel der Tabelle department..
5. Ein zusammengesetzter Primärschlüssel wird gebildet, indem in der Klammer auf mehrere Felder verwiesen wird. Die einzelnen Spalten werden durch Komma getrennt

### 5.2.1 Datentypen

Datentypen.

### 5.2.2 Constraints

Constraints definieren ganz allgemein gesprochen *Einschränkungen*, denen ein relationales Datenmodell entsprechen muss. Diese Einschränkungen können sein:

#### 5.2.2.1 NOT NULL constraints

Die Einschränkung bedeutet, dass der Wert einer Spalte kein NULL-Wert sein darf.

```
CREATE TABLE products (
  product_no integer NOT NULL,
  name text NOT NULL,
  price numeric
);
```

### 5.2.2.2 CHECK constraint

Damit wird ausgedrückt, dass Werte in Spalten gewisse Bedingungen einzuhalten haben, z.B. bestimmte Größenbereiche

```
CREATE TABLE products (  
    product_no integer,  
    name text,  
    price numeric CONSTRAINT positive_price CHECK (price > 0)  
); --column check constraint (benannt positive_price)
```

```
CREATE TABLE products (  
    product_no integer,  
    name text,  
    price numeric,  
    CHECK (price > 0),  
    discounted_price numeric,  
    CHECK (discounted_price > 0),  
    CHECK (price > discounted_price)  
); -- letzte zwei Zeilen sind table check constraints
```

Der Constraint wird erfüllt, wenn die Bedingung true ist oder einen NULL-Wert annimmt.

Der check-Constraint wird von MySQL zwar auf Syntaxfehler hin geparkt, jedoch nicht weiter umgesetzt :-).

### 5.2.2.3 UNIQUE constraint

Dieser Constraint fordert, dass für eine Spalte(n) innerhalb einer Tabelle ein Wert nur einmal vorkommt. NULL-Werte sind dennoch über mehrere Zeilen hinweg erlaubt. Er wird durch folgende Befehle abgebildet.

```
CREATE TABLE products (  
    product_no integer UNIQUE,  
    name text,  
    price numeric  
); -- column constraint  
CREATE TABLE products (  
    product_no integer,  
    name text,  
    price numeric,  
    UNIQUE (product_no)  
); -- table constraint  
  
CREATE TABLE example (  
    a integer,  
    b integer,  
    c integer,  
    UNIQUE (a, c) -- eindeutig über mehrere Spalten  
);
```



#### 5.2.2.4 PRIMARY/FOREIGN KEY Constraint

Ein PRIMARY KEY ist technisch gesehen ein UNIQUE constraint, der einen INDEX führt. Im Gegensatz zum UNIQUE Constraint dürfen die jeweiligen Spalten aber keinen NULL-Wert besitzen.

```
CREATE TABLE products (
  product_no integer PRIMARY KEY,
  name text,
  price numeric
);
CREATE TABLE example (
  a integer,
  b integer,
  c integer,
  PRIMARY KEY (a, c)
);
```

Ein FOREIGN KEY ist eine Einschränkung, dass sich der Wert einer Spalte an einem UNIQUE Wert einer anderen Tabelle orientieren muss. Ein FOREIGN KEY kann NULL-Werte besitzen, die *gegenüberliegende* Seite muss ein PRIMARY KEY oder ein UNIQUE constraint sein.

```
CREATE TABLE orders (
  order_id integer PRIMARY KEY,
  product_no integer REFERENCES products (product_no),
  quantity integer
);
-- Zusammengesetzter Fremdschlüssel mit Namen fk_myFKey
-- Gut zum späteren Löschen des FK.
CREATE TABLE t1 (
  a integer PRIMARY KEY,
  b integer,
  c integer,
  CONSTRAINT fk_myFKey FOREIGN KEY (b, c)
    REFERENCES other_table (c1, c2)
);
CREATE TABLE products (
  product_no integer PRIMARY KEY,
  name text,
  price numeric
);

CREATE TABLE orders (
  order_id integer PRIMARY KEY,
  shipping_address text,
  ...);

-- Lösch/Änderungsweitergabe verhindern oder erlauben
```

```
CREATE TABLE order_items (
    product_no integer REFERENCES products ON DELETE RESTRICT,
    order_id integer REFERENCES orders

ON DELETE CASCADE ON UPDATE CASCADE,
    quantity integer,
    PRIMARY KEY (product_no, order_id)
);
```

In MySQL muss als Tabellentyp InnoDB angegeben sein. Weiterhin muss vorher ein Index auf die Fremdschlüsselspalte vergeben worden sein.

```
CREATE TABLE product (
    category INT NOT NULL, id INT NOT NULL,
    price DECIMAL,
    PRIMARY KEY (category, id)
) TYPE=INNODB;

CREATE TABLE customer (
    id INT NOT NULL,
    PRIMARY KEY (id)
) TYPE=INNODB;

CREATE TABLE product_order (
    no INT NOT NULL AUTO_INCREMENT,
    product_category INT NOT NULL,
    product_id INT NOT NULL,
    customer_id INT NOT NULL,
    PRIMARY KEY (no),
    INDEX (product_category, product_id),
    FOREIGN KEY (product_category, product_id)
    REFERENCES product(category, id)
    ON UPDATE CASCADE ON DELETE RESTRICT,
    INDEX (customer_id),
    FOREIGN KEY (customer_id) REFERENCES customer(id)
) TYPE=INNODB;
```

### 5.2.3 Ändern/Löschen von Datenstrukturen

Mit Hilfe des Befehl **ALTER** kann die Struktur einer bestehenden Tabelle verändert werden. Dazu wird dem Statement je nach Bedarf eine drop, add, change, modify - Klausel hinzugefügt.

Der grundlegende Aufbau sieht wie folgt aus.

```
ALTER [IGNORE] TABLE tbl_name alter_specification [, alter_specification ...]

alter_specification:
    ADD [COLUMN] create_definition [FIRST | AFTER column_name ]
  | ADD [COLUMN] (create_definition, create_definition,...)
  | ADD INDEX [index_name] (index_col_name,...)
  | ADD PRIMARY KEY (index_col_name,...)
  | ADD UNIQUE [index_name] (index_col_name,...)
  | ADD FULLTEXT [index_name] (index_col_name,...)
  | ADD [CONSTRAINT symbol] FOREIGN KEY [index_name] (index_col_name, ...)
    [reference_definition]
  | ALTER [COLUMN] col_name {SET DEFAULT literal | DROP DEFAULT}
  | CHANGE [COLUMN] old_col_name create_definition
    [FIRST | AFTER column_name]
  | MODIFY [COLUMN] create_definition [FIRST | AFTER column_name]
  | DROP [COLUMN] col_name
  | DROP PRIMARY KEY
  | DROP INDEX index_name
  | DISABLE KEYS
  | ENABLE KEYS
  | RENAME [TO] new_tbl_name
  | ORDER BY col
  | table_options
```

Das folgende Beispiel zeigt den Umgang mit dem ALTER TABLE-Statement.

```
C:\mysql\bin>mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 5.0.1-alpha-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> create database testAlter;
mysql> use testAlter;
      Wir beginnen mit dem Erzeugen einer Demodatenbank

mysql> -- Wir erzeugen eine Tabelle t1
      CREATE TABLE t1 (a INTEGER,b CHAR(10));

      -- Wir ändern den Tabellennamen von t1 in t
mysql> ALTER TABLE t1 RENAME t2;

      --Wir ändern den Spaltentyp von a in TINYINT NOT NULL und
      ändern
      --den Spaltentyp von b nach CHAR(20) und geben der Spalte b
      den
      -- Namen c .
mysql> ALTER TABLE t2 MODIFY a TINYINT NOT NULL, CHANGE b c
      CHAR(20);

      --Wir fügen eine neue Spalte d mit Datentyp TIMESTAMP hinzu.
mysql> ALTER TABLE t2 ADD d TIMESTAMP;

      --Wir fügen einen Index auf die Spalte d hinzu und machen aus
      --der Spalte a einen Primärschlüssel.
mysql> ALTER TABLE t2 ADD INDEX (d), ADD PRIMARY KEY (a);

      --Wir löschen die Spalte c
mysql> ALTER TABLE t2 DROP COLUMN c;

      --Wir füen eine neue Spalte c mit dem Datentyp INTEGER hinzu.
      -- Der Wert soll sich automatisch hochzählen.
mysql> ALTER TABLE t2 ADD c INT UNSIGNED NOT NULL AUTO_INCREMENT,
      ADD INDEX (c);
```

## 5.2.4 Einfügen/Ändern/Löschen von Daten

### 5.2.4.1 INSERT

Mit Hilfe des Befehls INSERT wird ein neuer Datensatz in eine Tabelle eingefügt. Der INSERT-Befehl ist in verschiedenen Formen verfügbar.

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
INTO tbl_name [(Feld1, Feld2, ...)]
VALUES ({expr | DEFAULT},...), (...), ...
[ ON DUPLICATE KEY UPDATE col_name=expr, ... ]
```

Nach dem Nennen der Tabelle erfolgt die Definition der Feldnamen; mit Hilfe des Schlüsselwortes **VALUES** werden dann die jeweiligen Werte übergeben.

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
INTO tbl_name
SET col_name={expr | DEFAULT}, ...
[ ON DUPLICATE KEY UPDATE col_name=expr, ... ]
```

Das Einfügen der Werte erfolgt explizit durch eine **Feld=Wert-Zuweisung**

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
INTO tbl_name [(col_name,...)]
SELECT ...
```

Die zum Füllen der Felder erforderlichen Werte kommen aus einem SELECT Statement

### 5.2.4.2 REPLACE

Mit Hilfe des Befehls REPLACE wird ebenfalls ein neuer Datensatz in eine Tabelle eingefügt. Wird jedoch ein Wert für einen Primärschlüssel oder einen UNIQUE KEY mit übergeben, so wird der alte Datensatz vor dem Einfügen des neuen Datensatzes gelöscht. Verschiedene Formen sind möglich.

```
REPLACE [LOW_PRIORITY | DELAYED]
[INTO] tbl_name [(col_name,...)]
VALUES ({expr | DEFAULT},...), (...), ...

REPLACE [LOW_PRIORITY | DELAYED]
[INTO] tbl_name
SET col_name={expr | DEFAULT}, ...

REPLACE [LOW_PRIORITY | DELAYED]
[INTO] tbl_name [(col_name,...)]
SELECT ...
```

### 5.2.4.3 DELETE, UPDATE

Mit Hilfe des **DELETE**-Befehls werden Datensätze in einer Tabelle gelöscht. Es ist dabei zu beachten, dass **ohne einen WHERE-Teil alle Datensätze in einer Tabelle gelöscht werden**.

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM tbl_name  
[WHERE where_definition]  
[ORDER BY ...]  
[LIMIT row_count]
```

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE]  
tbl_name[.*] [, tbl_name[.*] ...]  
FROM table_references  
[WHERE where_definition]
```

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE]  
FROM tbl_name[.*] [, tbl_name[.*] ...]  
USING table_references  
[WHERE where_definition]
```

#### 5.2.4.4 Aufgaben zu UPDATE, INSERT, DELETE

##### Lösung Link.

1. Folgende Tabelle ist gegeben

```
mysql> DESCRIBE twounique; SELECT * FROM twounique;
```

Field	Type	Null	Key	Default	Extra
id1	tinyint(3) unsigned		PRI	0	
id2	tinyint(3) unsigned		UNI	0	

id1	id2
1	2
3	4
5	6

Sie führen folgende beiden Befehle aus. Was ist anschließend der Inhalt der Tabelle?

```
mysql> REPLACE INTO twounique VALUES (2,2);
mysql> REPLACE INTO twounique VALUES (2,6);
```

2. Wie fängt man mehrere Datensätze mit Hilfe eines einzigen INSERT-Statements hinzu
3. INSERT unterstützt den sog. IGNORE Modifier, REPLACE jedoch nicht. Warum ist das so ?
4. Mit welchem Statement kann man eine Tabelle komplett leeren ?
5. Mit welchem Statement kann man eine Tabelle teilweise leeren ?
6. Welcher Unterschied besteht in der Art und Weise, wie MySQL Fehler behandelt für ein single-row oder multiple-row-statement, wenn NULL-Werte in ein NOT NULL-Feld eingefügt werden?
7. Was für Gründe kann es geben, wenn ein UPDATE-Statement keine Auswirkungen hat, d.h. keinen einzigen Wert ändert.
8. Warum ist die Zahl der betroffenen Zeilen im folgenden UPDATE-Statement 0, obwohl die WHERE-Klausel auf 5 Zeilen zutrifft ? Warum ist die Anzahl der zutreffenden Zeilen 5 und nicht 10?

```
mysql> SELECT pid, grade FROM personnel;
+-----+-----+
| pid | grade |
+-----+-----+
| 1 | 1 |
| 2 | 2 |
| 3 | NULL |
| 4 | NULL |
| 5 | NULL |
| 6 | 1 |
| 7 | 1 |
| 8 | 1 |
| 9 | NULL |
| 10 | NULL |
| 11 | NULL |
| 12 | 1 |
| 13 | NULL |
+-----+-----+
13 rows in set
mysql> UPDATE personnel SET grade = 1 WHERE grade != 2;
Query OK, 0 rows affected (0.00 sec)
rows matched: 5 Changed: 0 Warnings: 0
```

9. Ist das folgende Statement WAHR oder FALSCH?

Um zu verhindern, dass UPDATE-Statements alle Zeilen einer Tabelle ändern würden, kann man mysql mit der **–safe-updates-Option** starten

10. Folgende Tabelle ist gegeben.

```
mysql> SELECT * FROM personnel;
+-----+-----+-----+
| pid | unit | grade |
+-----+-----+-----+
| 1 | 42 | 1 |
| 2 | 42 | 2 |
| 3 | 42 | NULL |
| 4 | 42 | NULL |
| 5 | 42 | NULL |
| 6 | 23 | 1 |
| 7 | 23 | 1 |
| 8 | 23 | 1 |
| 9 | 23 | NULL |
| 10 | 42 | NULL |
| 11 | 23 | NULL |
| 12 | 23 | 1 |
| 13 | 42 | NULL |
+-----+-----+-----+
```

Mit welchem einzigen UPDATE-Statement würde man alle Zeilen, die im Feld grade keinen Wert besitzen, auf 3 ändern.



11. Beziehen Sie sich auf die vorhergehende Tabelle. Welches REPLACE-Statement würden Sie benutzen, um das grade-Feld auf 4 und das unit-Feld auf 45 für alle Zeilen zu ändern, wo das pid-Feld den Wert 10 hat.

12. Die Tabelle personell hat den folgenden Aufbau:

```
mysql> DESCRIBE personell; SELECT * FROM personell;
```

```
+-----+-----+-----+-----+-----+-----+
+
| Field | Type                | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
+
| pid   | smallint(5) unsigned |      | PRI | NULL    | auto_incremnt |
|
| unit  | tinyint(3) unsigned  | YES  |     | NULL    |               |
| grade | tinyint(3) unsigned  | YES  |     | NULL    |               |
+-----+-----+-----+-----+-----+-----+
+

+-----+-----+-----+
| pid | unit | grade |
+-----+-----+-----+
|  1  |  42  |    1  |
|  2  |  42  |    2  |
|  3  |  42  |    3  |
|  4  |  42  |    3  |
|  5  |  42  |    3  |
|  6  |  23  |    1  |
|  7  |  23  |    1  |
|  8  |  23  |    1  |
|  9  |  23  |    3  |
| 10  |  42  |    3  |
| 11  |  23  |    3  |
| 12  |  23  |    1  |
| 13  |  42  |    3  |
+-----+-----+-----+
```

Welches UPDATE-Statement benutzen Sie, um die Werte des Feldes grade mit 1000 zu multiplizieren. Welche Werte würde das Statement erzeugen.

13. In the table `personnel`, the unit numbers were interchanged for some reason. Unit 23 is supposed to be 42, and 42 is supposed to be 23. What statement would you use to resolve this problem? Currently the table looks like this:

```
mysql> SELECT * FROM personnel;
```

pid	unit	grade
1	42	255
2	42	255
3	42	255
4	42	255
5	42	255
6	23	255
7	23	255
8	23	255
9	23	255
10	42	255
11	23	255
12	23	255
13	42	255

14. The table `petnames` contains the following data:

```
mysql> SELECT * FROM petnames;
```

name
Lucy
Macie
Myra
Cheep
Lucy
Myra
Cheep
Macie
Pablo
Stefan

Assume that you issue the following statement:

```
UPDATE petnames SET name = CONCAT(name, 1) ORDER BY name LIMIT 1;
```

What will the table's contents be after the UPDATE?

15. Will the following statement delete all rows from the table mytable ?

```
TRUNCATE TABLE mytable;
```

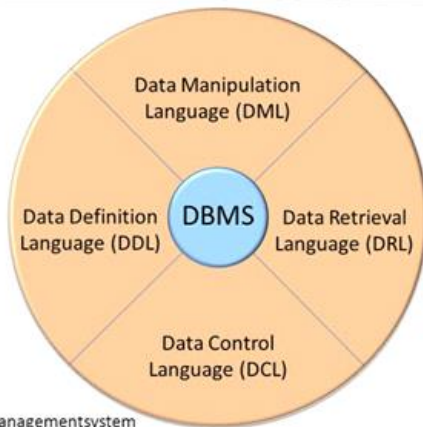
16. Will the following statement delete all rows from the table mytable ?

```
DELETE FROM mytable;
```

# SQL-DDL

Donnerstag, 26. Oktober 2017 16:14

## Structured Query Language (SQL)



Deklarative Sprache: Der Schwerpunkt liegt auf der Darstellung, was man will und nicht, wie man das machen muss, um es zu bekommen

DDL: Erzeugen von Strukturen (DB, Tabelle, Felder, Datentypen, FK, PK)  
CREATE, ALTER, DROP

DML: Managen der Werte in den Tabellen (INSERT, UPDATE, DELETE)

DRL: Auswählen der Daten (SELECT)

DCL: Managen des Datenbanksystems (GRANT, REVOKE, Replikation, Backup .....)

### Erstellen

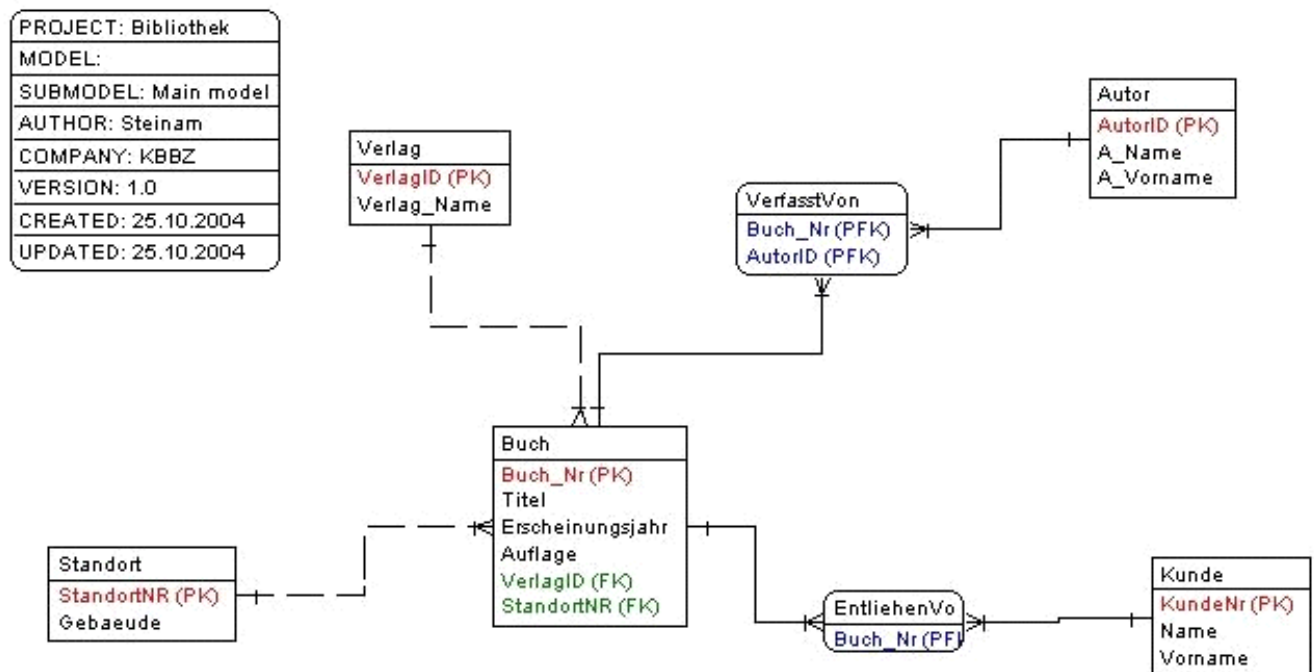
- Anlegen Datenbank
- Anlegen Tabelle
- Datentypen
- Auto\_Increment
- Tabellentyp
- Constraints
  - Foreign Key
  - Null/Not Null
  - Primary Key
  - Unique Key

### Ändern

Sprocedures, Triggers, Functions

# Übung

Donnerstag, 26. Oktober 2017 16:11



```
drop database if exists demo_11fi5;
```

```
create database if not exists demo_11fi5; -- DB erzeugen
```

```
/* Umlenken der befehle auf die neue DB */  
use demo_11fi5;
```

```
create table Autor(  
    AutorID int primary key auto_increment,  
    A_Name char(30),  
    A_Vorname varchar(30)  
) engine = innodb;
```

```
create table Kunde(  
    KundeNr int primary key auto_increment,  
    Name char(30),  
    Vorname char(30)  
) engine = innodb;
```

```
create table Verlag(  
  
    VerlagID int primary key auto_increment,  
    Verlag_Name char(30) unique key  
) engine = innodb;
```

```
create table Standort(  
    StandortNr int primary key auto_increment,
```

```
`Gebaeude` varchar(20)
```

```
)engine = innodb;
```

```
create table Buch(
```

```
    BuchNr int primary key auto_increment,
```

```
    Titel varchar(30) not null default 'unbekannt',
```

```
    Erscheinungsjahr int null,
```

```
    Auflage int,
```

```
    VerlagID integer,
```

```
    StandortNr integer not null,
```

```
    foreign key (VerlagID) references Verlag(VerlagID),
```

```
    constraint `myownFK` foreign key(StandortNr) references Standort(StandortNr)
```

```
)engine = innodb;
```

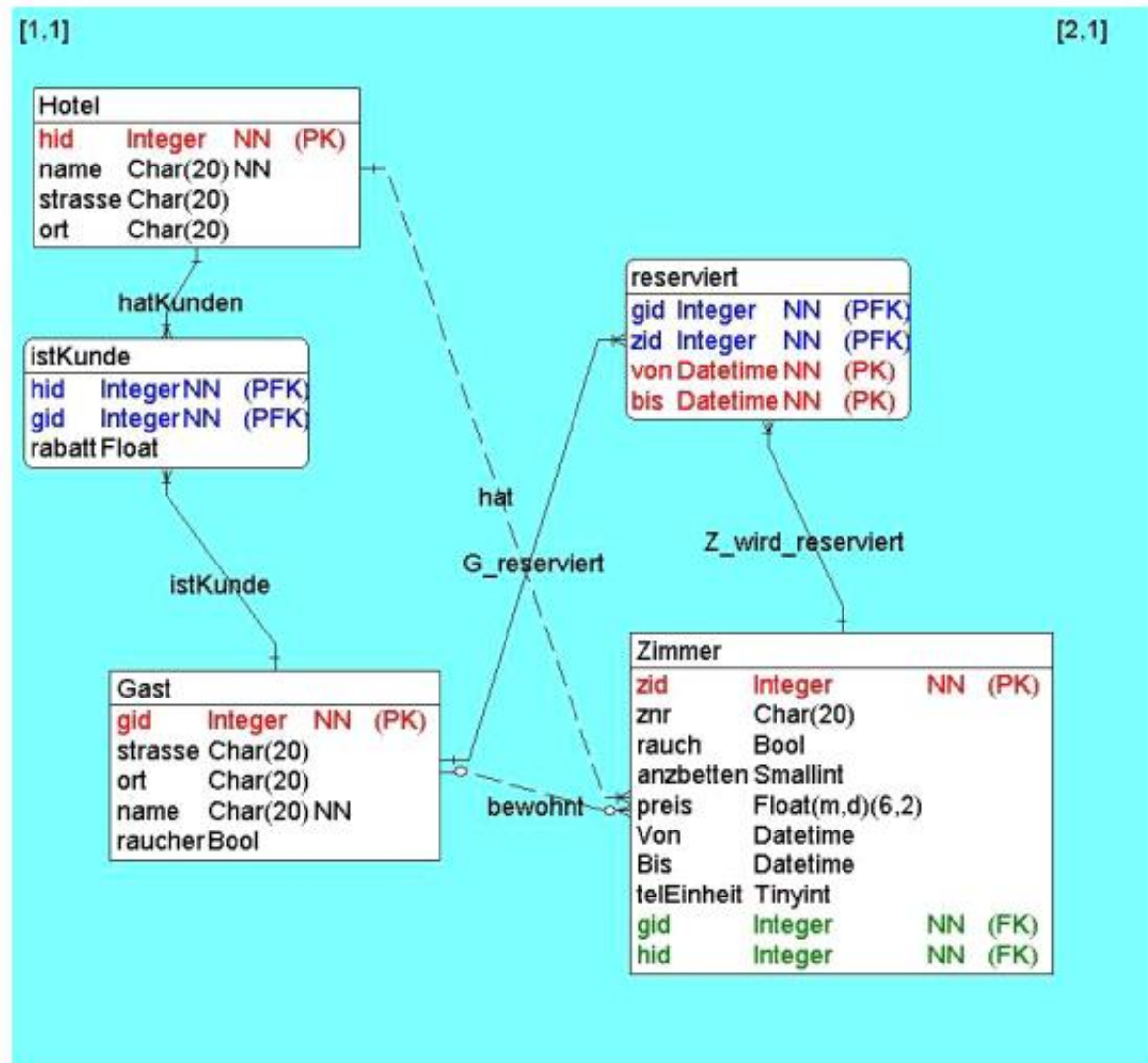
```
ALTER TABLE `buch` DROP FOREIGN KEY `buch_ibfk_1`;
```

```
ALTER TABLE `buch` DROP FOREIGN KEY `myownFK`;
```

# Übung

Donnerstag, 26. Oktober 2017

16:19



## Lösung

```
/* SQL Manager Lite for MySQL 5.7.2.52112 */
```

```
/* ----- */
```

```
/* Host : localhost */
```

```
/* Port : 3306 */
```

```
/* Database : demo_11fi5 */
```

```
/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES 'latin1' */;
```

```
SET FOREIGN_KEY_CHECKS=0;
```

```
CREATE DATABASE `demo_11fi5`
  CHARACTER SET 'latin1'
  COLLATE 'latin1_swedish_ci';
```

```
USE `demo_11fi5`;
```

```
/* Struktur für die Tabelle `autor`: */
```

```
CREATE TABLE `autor` (  
  `AutorID` INTEGER(11) NOT NULL AUTO_INCREMENT,  
  `A_Name` CHAR(30) COLLATE latin1_swedish_ci DEFAULT NULL,  
  `A_Vorname` VARCHAR(30) COLLATE latin1_swedish_ci DEFAULT NULL,  
  PRIMARY KEY USING BTREE (`AutorID`)  
) ENGINE=InnoDB  
AUTO_INCREMENT=1 CHARACTER SET 'latin1' COLLATE 'latin1_swedish_ci'  
;
```

```
/* Struktur für die Tabelle `verlag`: */
```

```
CREATE TABLE `verlag` (  
  `VerlagID` INTEGER(11) NOT NULL AUTO_INCREMENT,  
  `Verlag_Name` CHAR(30) COLLATE latin1_swedish_ci DEFAULT NULL,  
  PRIMARY KEY USING BTREE (`VerlagID`),  
  UNIQUE KEY `Verlag_Name` USING BTREE (`Verlag_Name`)  
) ENGINE=InnoDB  
AUTO_INCREMENT=1 CHARACTER SET 'latin1' COLLATE 'latin1_swedish_ci'  
;
```

```
/* Struktur für die Tabelle `standort`: */
```

```
CREATE TABLE `standort` (  
  `StandortNr` INTEGER(11) NOT NULL AUTO_INCREMENT,  
  `Gebaeude` VARCHAR(20) COLLATE latin1_swedish_ci DEFAULT NULL,  
  PRIMARY KEY USING BTREE (`StandortNr`)  
) ENGINE=InnoDB  
AUTO_INCREMENT=1 CHARACTER SET 'latin1' COLLATE 'latin1_swedish_ci'  
;
```

```
/* Struktur für die Tabelle `buch`: */
```

```
CREATE TABLE `buch` (  
  `BuchNr` INTEGER(11) NOT NULL AUTO_INCREMENT,  
  `Titel` VARCHAR(30) COLLATE latin1_swedish_ci NOT NULL DEFAULT 'unbekannt',  
  `Erscheinungsjahr` INTEGER(11) DEFAULT NULL,  
  `Auflage` INTEGER(11) DEFAULT NULL,  
  `VerlagID` INTEGER(11) DEFAULT NULL,  
  `StandortNr` INTEGER(11) NOT NULL,  
  PRIMARY KEY USING BTREE (`BuchNr`),  
  KEY `VerlagID` USING BTREE (`VerlagID`),  
  KEY `myownFK` USING BTREE (`StandortNr`),  
  CONSTRAINT `buch_ibfk_1` FOREIGN KEY (`VerlagID`) REFERENCES `verlag` (`VerlagID`),  
  CONSTRAINT `myownFK` FOREIGN KEY (`StandortNr`) REFERENCES `standort` (`StandortNr`)  
) ENGINE=InnoDB  
AUTO_INCREMENT=1 CHARACTER SET 'latin1' COLLATE 'latin1_swedish_ci'  
;
```

```
/* Struktur für die Tabelle `gast`: */
```

```
CREATE TABLE `gast` (  
  `gid` INTEGER(11) NOT NULL AUTO_INCREMENT,  
  `strasse` CHAR(20) COLLATE latin1_swedish_ci DEFAULT NULL,
```



```

`ort` CHAR(20) COLLATE latin1_swedish_ci DEFAULT NULL,
`name` CHAR(20) COLLATE latin1_swedish_ci NOT NULL,
`raucher` TINYINT(1) DEFAULT NULL,
PRIMARY KEY USING BTREE (`gid`)
) ENGINE=InnoDB
AUTO_INCREMENT=1 CHARACTER SET 'latin1' COLLATE 'latin1_swedish_ci'
;

/* Struktur für die Tabelle `hotel`: */

CREATE TABLE `hotel` (
  `hid` INTEGER(11) NOT NULL AUTO_INCREMENT,
  `name` CHAR(20) COLLATE latin1_swedish_ci NOT NULL,
  `strasse` CHAR(20) COLLATE latin1_swedish_ci DEFAULT NULL,
  `ort` CHAR(20) COLLATE latin1_swedish_ci DEFAULT NULL,
  PRIMARY KEY USING BTREE (`hid`)
) ENGINE=InnoDB
AUTO_INCREMENT=1 CHARACTER SET 'latin1' COLLATE 'latin1_swedish_ci'
;

/* Struktur für die Tabelle `istkunde`: */

CREATE TABLE `istkunde` (
  `hid` INTEGER(11) NOT NULL,
  `gid` INTEGER(11) NOT NULL,
  `rabatt` FLOAT DEFAULT NULL,
  PRIMARY KEY USING BTREE (`hid`, `gid`),
  KEY `IX_hatKunden` USING BTREE (`hid`),
  KEY `IX_istKunde` USING BTREE (`gid`),
  CONSTRAINT `istkunde_ibfk_1` FOREIGN KEY (`hid`) REFERENCES `hotel` (`hid`),
  CONSTRAINT `istkunde_ibfk_2` FOREIGN KEY (`gid`) REFERENCES `gast` (`gid`)
) ENGINE=InnoDB
CHARACTER SET 'latin1' COLLATE 'latin1_swedish_ci'
;

/* Struktur für die Tabelle `kunde`: */

CREATE TABLE `kunde` (
  `KundeNr` INTEGER(11) NOT NULL AUTO_INCREMENT,
  `Name` CHAR(30) COLLATE latin1_swedish_ci DEFAULT NULL,
  `Vorname` CHAR(30) COLLATE latin1_swedish_ci DEFAULT NULL,
  PRIMARY KEY USING BTREE (`KundeNr`)
) ENGINE=InnoDB
AUTO_INCREMENT=1 CHARACTER SET 'latin1' COLLATE 'latin1_swedish_ci'
;

/* Struktur für die Tabelle `zimmer`: */

CREATE TABLE `zimmer` (
  `zid` INTEGER(11) NOT NULL,
  `znr` CHAR(20) COLLATE latin1_swedish_ci DEFAULT NULL,
  `rauch` TINYINT(1) DEFAULT NULL,
  `anzbetten` SMALLINT(6) DEFAULT NULL,
  `preis` FLOAT(6,2) DEFAULT NULL,
  `Von` DATETIME DEFAULT NULL,
  `Bis` DATETIME DEFAULT NULL,
  `telEinheit` TINYINT(4) DEFAULT NULL,

```

```

`gid` INTEGER(11) NOT NULL,
`hid` INTEGER(11) NOT NULL,
PRIMARY KEY USING BTREE (`zid`),
KEY `IX_bewohnt` USING BTREE (`gid`),
KEY `IX_hat` USING BTREE (`hid`),
CONSTRAINT `zimmer_ibfk_1` FOREIGN KEY (`gid`) REFERENCES `gast` (`gid`),
CONSTRAINT `zimmer_ibfk_2` FOREIGN KEY (`hid`) REFERENCES `hotel` (`hid`)
) ENGINE=InnoDB
CHARACTER SET 'latin1' COLLATE 'latin1_swedish_ci'
;

/* Struktur für die Tabelle `reserviert`: */

CREATE TABLE `reserviert` (
`gid` INTEGER(11) NOT NULL,
`zid` INTEGER(11) NOT NULL,
`von` DATETIME NOT NULL,
`bis` DATETIME NOT NULL,
PRIMARY KEY USING BTREE (`gid`, `zid`, `von`, `bis`),
KEY `IX_G_reserviert` USING BTREE (`gid`),
KEY `IX_Z_wird_reserviert` USING BTREE (`zid`),
CONSTRAINT `reserviert_ibfk_1` FOREIGN KEY (`gid`) REFERENCES `gast` (`gid`),
CONSTRAINT `reserviert_ibfk_2` FOREIGN KEY (`zid`) REFERENCES `zimmer` (`zid`)
) ENGINE=InnoDB
CHARACTER SET 'latin1' COLLATE 'latin1_swedish_ci'
;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;

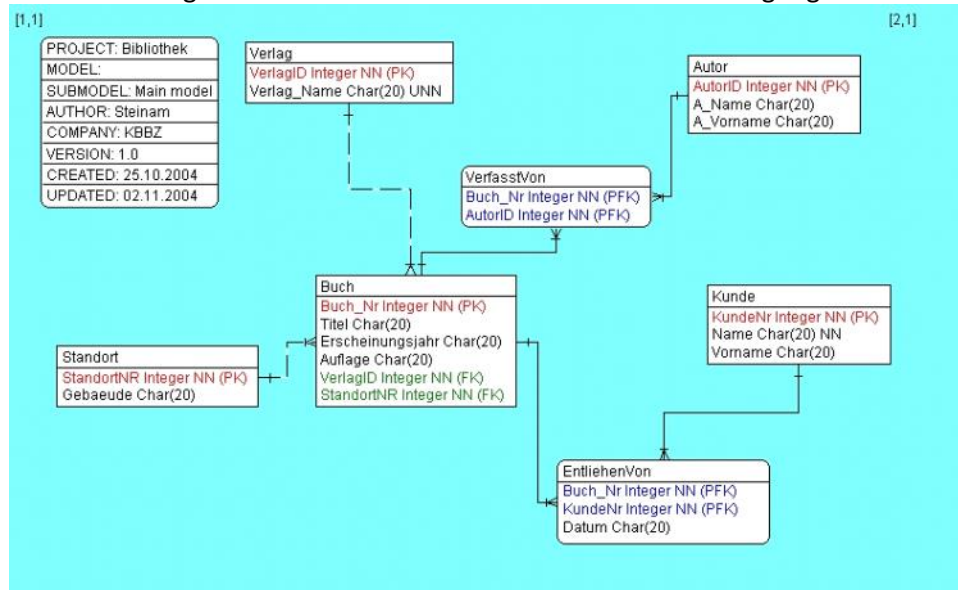
```

# Übung Alter - Table

Mittwoch, 22. Januar 2020 11:06

## Bibliothek -- ALTER

Ihnen steht folgendes Datenmodell der Bibliothek-Db zur Verfügung.



Ein Kollege hat bereits ein SQL-Skript zum Erzeugen der DB angefertigt und auf den Produktivserver eingespielt. Sie stellen fest, dass sich zwischen Skript und Datenmodell einige Unterschiede befinden. Finden Sie diese heraus und bereinigen Sie diese mit Hilfe von ALTER-Statements.

--

-- Tabellenstruktur für Tabelle `autor`

--

```
DROP TABLE IF EXISTS `autor`;
```

```
DROP TABLE IF EXISTS `buch`;
```

```
CREATE TABLE IF NOT EXISTS `autor` (  
  `AutorID` int(11) NOT NULL,  
  `A_Name` varchar(20) default NULL,  
  `A_Vorname` char(20) default NULL  
  ) Engine=MyISAM ;
```

--

-- Tabellenstruktur für Tabelle `buch`

--

```
CREATE TABLE IF NOT EXISTS `buch` (  
  `Buch_Nr` int(11) NOT NULL auto_increment,  
  `Titel` char(20) default NULL,  
  `Erscheinungsjahr` char(20) default NULL,  
  `Auflage` char(20) default NULL,  
  `StandortNr` int(11) NOT NULL default '0',  
  PRIMARY KEY (`Buch_Nr`)  
  ) TYPE=MyISAM AUTO_INCREMENT=1 ;
```

--

-- Tabellenstruktur für Tabelle `kunde`

--

```
DROP TABLE IF EXISTS `kunde`;
```

```
CREATE TABLE IF NOT EXISTS `kunde` (  
  `KundeNr` int(11) NOT NULL auto_increment,  
  `Name` char(20) NOT NULL default "",
```

```

`Vorname` char(20) default NULL,
PRIMARY KEY (`KundeNr`)
) TYPE=MyISAM AUTO_INCREMENT=1 ;
--
-- Tabellenstruktur für Tabelle `standort`
--
DROP TABLE IF EXISTS `standort`;
CREATE TABLE IF NOT EXISTS `standort` (
  `StandortNr` int(11) NOT NULL auto_increment,
  `Gebaeude` char(20) default NULL,
  PRIMARY KEY (`StandortNr`)
) TYPE=MyISAM AUTO_INCREMENT=1 ;
--
-- Tabellenstruktur für Tabelle `verlag`
--
DROP TABLE IF EXISTS `verlag`;
CREATE TABLE IF NOT EXISTS `verlag` (
  `VerlagID` int(11) NOT NULL auto_increment,
  `Verlag_Name` char(20) NOT NULL default "",
  PRIMARY KEY (`VerlagID`)
) TYPE=MyISAM AUTO_INCREMENT=1 ;
DROP TABLE IF EXISTS `Verfasst_von`;
CREATE TABLE IF NOT EXISTS `Verfasst_von` (
  id integer
);
DROP TABLE IF EXISTS `Entliehen_von`;
CREATE TABLE IF NOT EXISTS `Entliehen_von` (
  id integer
);

-- ToDo --
-- Tabelle autor:
    -- Primary Key für tabelle 'autor' auf das feld AutorID
    -- AName von varchar(20) auf char(20) ändern
    -- AutorID als AUTO_INCREMENT

-- Tabelle Buch:
    -- verlagID hinzufügen (als Fremdschlüssel)
    -- Erscheinungsjahr auf Datentyp YEAR ändern
-- Tabelle Verlag:
    -- UNIQUE KEY auf Verlag_Name hinzufügen
-- Sonstiges:
    -- Spalten aller Zwischentabellen anlegen
    -- ID-Spalte aller Zwischentabellen löschen

```

# Übung Wawi - Alter

Mittwoch, 22. Januar 2020 11:07

## WAWI ALTER

Erzeugen Sie eine Datenbank gemäß untenstehender Abbildung.  
Benutzen sie als Tabellenformat InnoDB.

tblkunden	
K_Nummer	Integer NN (PK)
K_Name	Char(50) NN
K_Vorname	Char(50) NN
K_Straße	Char(50) NN
K_PLZ	Char(5) NN
K_Ort	Char(50) NN
K_Fax	Char(20)

tblbestelldetails	
A_Nr	Integer NN (PK)
B_Nr	Integer NN (PK)
B_Menge	Integer NN
A_VK_Preis	Integer NN

tblartikel	
A_Nr	Integer NN (PK)
A_Bez	Char(50) NN
A_Art	Char(50) NN
A_VK	Decimal(20,4) NN
A_Bestand	Integer NN

tblbestellungen	
B_Nr	Integer NN
B_Datum	Datetime NN
B_Lieferdatum	Datetime NN
B_Erledigt	Tinyint NN
B_Rechnung	Tinyint NN

### Aufgaben zum ALTER-Teil

Ändern Sie die oben erzeugte Datenbank gemäß den untenstehenden Anweisungen.

- tblbestellungen
  - Fügen Sie in der Tabelle tblbestellungen das Fremdschlüsselfeld KundenNr hinzu.
  - Machen Sie das Feld B\_Nr zum Primärschlüssel (AUTO\_INCREMENT)
  - Wandeln Sie den Datentyp des Feldes B\_Datum in ein TIMESTAMP-Feld um.
  - Informieren Sie sich über die Bitbreite des Datentypes TINYINT.
- tblBestelldetails
  - Ändern sie den Datentyp des Feldes A\_VK\_Preis in einen Gleitkomma-Datentyp
  - Erzeugen Sie die jeweiligen Fremdschlüsselbeziehungen für die Felder A\_Nr und B\_Nr
- tblArtikel
  - Die Werte des Feldes A\_Bez soll eindeutig sein.
  - Fügen sie ein Feld A\_Minbestand (integer, not null) hinzu
- tblKunden
  - Das Feld K\_PLZ soll mehr als 10 Zeichen aufnehmen können.
  - Lesen Sie in der Online-Dokumentation nach, was passieren würde, wenn sie das Feld verkleinern würden und bereits Werte in der Spalte enthalten sind.
  - Machen Sie den Primärschlüssel AUTO\_INCREMENT fähig.
  - Fügen Sie das Feld K\_Telefon hinzu (Breite: 50 Zeichen)
  - Löschen Sie das Feld K\_Fax
  - Legen Sie einen gemeinsamen Index auf das Feld K\_Name und K\_Ort.

# Alter table

Mittwoch, 22. Januar 2020 11:07

## ALTER - Statement

Mit Hilfe des Befehl *ALTER* kann die Struktur einer bestehenden Tabelle verändert werden. Dazu wird dem Statement je nach Bedarf eine *drop*, *add*, *change*, *modify* - Klausel hinzugefügt.

Der grundlegende Aufbau sieht wie folgt aus.


ALTER [IGNORE] TABLE tbl\_name alter\_specification [, alter\_specification ...]

alter\_specification:

- ADD [COLUMN] create\_definition [FIRST | AFTER column\_name ]
- | ADD [COLUMN] (create\_definition, create\_definition,...)
- | ADD INDEX [index\_name] (index\_col\_name,...)
- | ADD PRIMARY KEY (index\_col\_name,...)
- | ADD UNIQUE [index\_name] (index\_col\_name,...)
- | ADD FULLTEXT [index\_name] (index\_col\_name,...)
- | ADD [CONSTRAINT symbol] FOREIGN KEY [index\_name] (index\_col\_name,...)  
[reference\_definition]
- | ALTER [COLUMN] col\_name {SET DEFAULT literal | DROP DEFAULT}
- | CHANGE [COLUMN] old\_col\_name create\_definition  
[FIRST | AFTER column\_name]
- | MODIFY [COLUMN] create\_definition [FIRST | AFTER column\_name]
- | DROP [COLUMN] col\_name
- | DROP PRIMARY KEY
- | DROP INDEX index\_name
- | DISABLE KEYS
- | ENABLE KEYS
- | RENAME [TO] new\_tbl\_name
- | ORDER BY col
- | table\_options

Das folgende Beispiel zeigt den Umgang mit dem ALTER TABLE-Statement.

Befehl	Ziel
C:\mysql\bin>mysql Welcome to the MySQL monitor. Commands end with ; or \g. Your MySQL connection id is 1 to server version: 5.0.1-alpha-nt Type 'help;' or '\h' for help. Type '\c' to clear the buffer. mysql> create database testAlter; mysql> use testAlter;	Wir beginnen mit dem Erzeugen einer Demodatenbank
mysql> CREATE TABLE t1 (a INTEGER, b CHAR(10));	Wir erzeugen eine Tabelle t1
mysql> ALTER TABLE t1 RENAME t2;	Wir ändern den Tabellennamen von t1 in t2
ALTER TABLE t2 MODIFY a TINYINT NOT NULL, CHANGE b c CHAR(20);	Wir ändern den Spaltentyp von <i>a</i> in TINYINT NOT NULL und ändern den Spaltentyp von <i>b</i> nach CHAR(20) und geben der Spalte <i>b</i> den Namen <i>c</i> .
mysql> ALTER TABLE t2 ADD d TIMESTAMP;	Wir fügen eine neue Spalte <i>d</i> mit Datentyp TIMESTAMP hinzu.
mysql> ALTER TABLE t2 ADD INDEX (d), ADD PRIMARY KEY (a);	Wir fügen einen Index auf die Spalte <i>d</i> hinzu und machen aus der Spalte <i>a</i> einen Primärschlüssel.

mysql> ALTER TABLE t2 DROP COLUMN c;	Wir löschen die Spalte c
mysql> ALTER TABLE t2 ADD c INT UNSIGNED NOT NULL AUTO_INCREMENT, ADD INDEX (c); <div>  <b>Anmerkung</b> <p>Note that we indexed c, because AUTO_INCREMENT columns must be indexed, and also that we declare c as NOT NULL, because indexed columns cannot be NULL.</p> <p>When you add an AUTO_INCREMENT column, column values are filled in with sequence numbers for you automatically. You can set the first sequence number by executing SET INSERT_ID=# before ALTER TABLE or using the AUTO_INCREMENT = # table option.</p> </div>	Wir fügen eine neue Spalte c mit dem Datentyp INTEGER hinzu. Der Wert soll sich automatisch hochzählen.

**Tabelle 2.14. Gebrauch des ALTER-Statements**

## 2.7.4. INSERT

Mit Hilfe des Befehls INSERT wird ein neuer Datensatz in eine Tabelle eingefügt. Der INSERT-Befehl ist in verschiedenen Formen verfügbar.

Statement	Erklärung
<pre>INSERT [LOW_PRIORITY   DELAYED   HIGH_PRIORITY] [IGNORE] INTO tbl_name [(Feld1, Feld2, ...)] VALUES ({expr   DEFAULT},...), (...), ... [ ON DUPLICATE KEY UPDATE col_name=expr, ... ]</pre>	<p>Nach dem Nennen der Tabelle erfolgt die Definition der Feldnamen; mit Hilfe des Schlüsselwortes <b>VALUES</b> werden dann die jeweiligen Werte übergeben.</p> <p>Insert into buch(buch_nr, name, standortid) values (1, 'VWV', 1);  Insert into buch(name, standortid) values ('TopGun', 2);</p>
<pre>INSERT [LOW_PRIORITY   DELAYED   HIGH_PRIORITY] [IGNORE] INTO tbl_name SET col_name={expr   DEFAULT}, ... [ ON DUPLICATE KEY UPDATE col_name=expr, ... ]</pre>	<p>Das Einfügen der Werte erfolgt explizit durch eine <b>Feld=Wert-Zuweisung</b></p>
<pre>INSERT [LOW_PRIORITY   DELAYED   HIGH_PRIORITY] [IGNORE] INTO tbl_name [(col_name,...)] SELECT ...</pre>	<p>Die zum Füllen der Felder erforderlichen Werte werden einen SELECT Statement entnommen.</p> <p>Insert into buch_hist  Select id, name, standortNr from buch</p>

Tabelle 2.21. Verschiedene Formen des INSERT-Statements

Where Date = YMD

## 2.7.5. REPLACE

Mit Hilfe des Befehls REPLACE wird ebenfalls ein neuer Datensatz in eine Tabelle eingefügt. Wird jedoch ein Wert für einen Primärschlüssel oder ein UNIQUE KEY mit übergeben, so wird der alte Datensatz vor dem Einfügen des neuen Datensatzes gelöscht.

Statement 1	Statement 2	Statement 3
<pre>REPLACE [LOW_PRIORITY   DELAYED] [INTO] tbl_name [(col_name,...)] VALUES ({expr   DEFAULT},...), (...), ...</pre>	<pre>REPLACE [LOW_PRIORITY   DELAYED] [INTO] tbl_name SET col_name={expr   DEFAULT}, ...</pre>	<pre>REPLACE [LOW_PRIORITY   DELAYED] [INTO] tbl_name [(col_name,...)] SELECT ...</pre>

Tabelle 2.22. Verschiedene Formen des REPLACE-Statements

## 2.7.6. delete, update

Mit Hilfe des **DELETE** -Befehls werden Datensätze in einer Tabelle gelöscht. Es ist dabei zu beachten, dass ohne einen WHERE-Teil alle Datensätze in einer Tabelle gelöscht werden.

Single-table syntax:	Multiple-table syntax	Multiple-table syntax
<pre>DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM tbl_name [WHERE where_definition] [ORDER BY ...] [LIMIT row_count]</pre>	<pre>DELETE [LOW_PRIORITY] [QUICK] [IGNORE] tbl_name[*] [, tbl_name[*] ...] FROM table_references [WHERE where_definition]</pre>	<pre>DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM tbl_name[*] [, tbl_name[*] ...] USING table_references [WHERE where_definition]</pre>

Tabelle 2.23. Verschiedene Formen des DELETE-Statements



Single-table syntax:

```
1  UPDATE [LOW_PRIORITY] [IGNORE] table_reference
2      SET assignment_list
3      [WHERE where_condition]
4      [ORDER BY ...]
5      [LIMIT row_count]
6
7  value:
8      {expr | DEFAULT}
9
10 assignment:
11     col_name = value
12
13 assignment_list:
14     assignment [, assignment] ...
```

# Aufgaben zu update, insert, delete

Sonntag, 26. Januar 2020 20:53

1. Folgende Tabelle ist gegeben

```
mysql> DESCRIBE twounique; SELECT * FROM twounique;
```

Field	Type	Null	Key	Default	Extra
id1	tinyint(3) unsigned		PRI	0	
id2	tinyint(3) unsigned		UNI	0	

id1	id2
1	2
3	4
5	6

Sie führen folgende beiden Befehle aus. Was ist anschließend der Inhalt der Tabelle?

```
mysql> REPLACE INTO twounique VALUES (2,2);
```

```
mysql> REPLACE INTO twounique VALUES (2,6);
```

- Wie fängt man mehrere Datensätze mit Hilfe eines einzigen INSERT-Statements hinzu
- INSERT unterstützt den sog. IGNORE Modifier, REPLACE jedoch nicht. Warum ist das so ?
- Mit welchem Statement kann man eine Tabelle komplett leeren ?
- Mit welchem Statement kann man eine Tabelle teilweise leeren ?
- Welcher Unterschied besteht in der Art und Weise, wie MySQL Fehler behandelt für ein single-row oder multiple-row-statement, wenn NULL-Werte in ein NOT NULL-Feld eingefügt werden?
- Was für Gründe kann es geben, wenn ein UPDATE-Statement keine Auswirkungen hat, d.h. keinen einzigen Wert ändert.
- Warum ist die Zahl der betroffenen Zeilen im folgenden UPDATE-Statement 0, obwohl die WHERE-Klausel auf 5 Zeilen zutrifft ? Warum ist die Anzahl der zutreffenden Zeilen 5 und nicht 10?

```
mysql> SELECT pid, grade FROM personnel;
```

pid	grade
1	1
2	2
3	NULL
4	NULL
5	NULL
6	1
7	1
8	1
9	NULL
10	NULL
11	NULL
12	1
13	NULL

13 rows in set

```
mysql> UPDATE personnel SET grade = 1 WHERE grade != 2;
Query OK, 0 rows affected (0.00 sec)
rows matched: 5 Changed: 0 Warnings: 0
```

9. Ist das folgende Statement WAHR oder FALSCH?

Um zu verhindern, dass UPDATE-Statements alle Zeilen einer Tabelle ändern würden, kann man mysql mit der **—safe-updates-Option** starten

10. Folgende Tabelle ist gegeben.

```
mysql> SELECT * FROM personnel;
```

pid	unit	grade
1	42	1
2	42	2
3	42	NULL
4	42	NULL
5	42	NULL
6	23	1
7	23	1
8	23	1
9	23	NULL
10	42	NULL
11	23	NULL
12	23	1
13	42	NULL

Mit welchem einzigen UPDATE-Statement würde man alle Zeilen, die im Feld grade keinen Wert besitzen, auf 3 ändern.

11. Beziehen Sie sich auf die vorhergehende Tabelle. Welches REPLACE-Statemen würden Sie benutzen, um das grade-Feld auf 4 und das unit-Feld auf 45 für alle Zeilen zu ändern, wo das pid-Feld den Wert 10 hat.

12. Die Tabelle personell hat den folgenden Aufbau:

```
mysql> DESCRIBE personnel; SELECT * FROM personnel;
```

Field	Type	Null	Key	Default	Extra
pid	smallint(5) unsigned		PRI	NULL	auto_increment
unit	tinyint(3) unsigned	YES		NULL	
grade	tinyint(3) unsigned	YES		NULL	

pid	unit	grade
1	42	1
2	42	2
3	42	3
4	42	3
5	42	3
6	23	1
7	23	1
8	23	1
9	23	3

10	42	3
11	23	3
12	23	1
13	42	3

Welches UPDATE-Statement benutzen Sie, um die Werte des Feldes grade mit 1000 zu multiplizieren. Welche Werte würde das Statement erzeugen.

13. In the table personnel, the unit numbers were interchanged for some reason. Unit 23 is supposed to be 42, and 42 is supposed to be 23. What statement would you use to resolve this problem? Currently the table looks like this:

```
mysql> SELECT * FROM personnel;
```

pid	unit	grade
1	42	255
2	42	255
3	42	255
4	42	255
5	42	255
6	23	255
7	23	255
8	23	255
9	23	255
10	42	255
11	23	255
12	23	255
13	42	255

14. The table petnames contains the following data:

```
mysql> SELECT * FROM petnames;
```

name
Lucy
Macie
Myra
Cheep
Lucy
Myra
Cheep
Macie
Pablo
Stefan

Assume that you issue the following statement:

```
mysql> UPDATE petnames SET name = CONCAT(name, 1) ORDER BY name LIMIT 1;
```

What will the table's contents be after the UPDATE?

15. Will the following statement delete all rows from the table mytable ?  
TRUNCATE TABLE mytable;
16. Will the following statement delete all rows from the table mytable ?  
DELETE FROM mytable;

# Insert\_update\_create\_alter

Dienstag, 28. Januar 2020 09:44



create\_alter\_insert\_update\_delete

kunden		
Kunden_Code	Varchar(5)	NN
Firma	Varchar(40)	NN
Strasse	Varchar(60)	NN
Ort	Varchar(15)	NN
PLZ	Varchar(10)	NN
Land	Varchar(15)	NN
Telefon	Varchar(24)	NN

bestellungen		
Bestell_Nr	Bigint	NN
Kunden_Code	Varchar(5)	NN
Bestelldatum	Datetime	NN
Lieferdatum	Datetime	NN
Versanddatum	Datetime	NN
Frachtkosten	Double	NN

bestelldetails		
Bestell_Nr	Bigint	NN
Artikel_Nr	Bigint	NN
Einzelpreis	Double	NN
Anzahl	Integer	NN
Rabatt	Double	NN

artikel		
Artikel_Nr	Bigint	NN
Artikelname	Varchar(40)	NN
Einzelpreis	Double	NN
Lagerbestand	Integer	NN
Mindestbestand	Integer	NN
Auslaufartikel	Tinyint	NN

Die Konvertierung der bekannten Datenbank Nordwind.mdb aus dem Access-Format in eine MySQL-Datenbank führte zu folgender (reduzierter) Abbildung.

Erstellen Sie ein Skript, mit der Sie diese Grafik in eine MySQL-Datenbank überführen. 8 P  
Benutzen Sie als Tabellentyp durchgängig InnoDB.

Speichern Sie ihre Skript unter den Namen : **11FI3\_'IhrName'\_CREATE.sql**

---

**Verändern Sie die Struktur der obigen Datenbank gemäß folgenden Anweisungen:  
Sammeln Sie diese Veränderungen ebenfalls in einem Editor. Nennen Sie diese Datei  
11FI3\_'IhrName'\_ALTER.sql**

Die Datenbank ist noch nicht optimiert. Führen Sie folgende Änderungen an der oben erstellten Datenbank durch.

- Erstellen Sie Primärschlüssel auf die Spalten 3 P  
Kunden\_Code der Tabelle Kunden  
Bestell\_Nr der Tabelle bestellungen  
Artikel\_Nr der Tabelle artikel
- Erzeugen Sie einen zusammengesetzten Primärschlüssel auf die Spalten Bestell\_Nr und 2 P  
Artikel\_Nr der Tabelle bestelldetails
- Der Wert eines Artikelnamens darf in der Tabelle artikel in allen Datensätzen nur ein 2 P  
einziges Mal vorkommen
- Vergrößern Sie das Feld Kunden.Ort auf 60 Zeichen 2 P
- Stellen Sie Fremdschlüsselbeziehungen her zwischen den Tabellen 4 P

bestellungen ---> kunden  
 bestelldetails ---> bestellungen  
 artikel

6. Löschen Sie das Feld kunden.Land 2 P

7. Ändern Sie den Spaltennamen des Felds Artikel.Artikelname in Artikel.A\_Name  
 2 P

8. Fügen Sie folgenden Datensatz in die Tabelle Kunden ein: 2 P

Kundencode	Firma	Strasse	Ort	PLZ	Land	Telefon
STE11	Steinam OHG	Bauchweg 3	Würzburg	97071	Grmany	0931/111
SIER2	Sierl KG	Herbstweg 7	TBB	95434	Austria	09341/552

9. Fügen Sie einen Datensatz mit beliebigen Werten in die Tabelle Artikel ein.

10. Der Kunde Sierl tätigt folgende Bestellung. 2 P

Beziehen Sie sich dabei auf den unter 9. eingefügten Datensatz. Sie bestellen 9 Einheiten des Artikels.

Bestelldatum: 20.11.2005  
 Lieferdatum: 23.11.2005  
 Versanddatum: 22.11.2005  
 Frachtkosten: 2345,87 Euro

10. Sie haben sich beim Eintrag für das Land im Datensatz des Kunden Steinam verschrieben. Ändern Sie den Wert in Germany 2 P

11. Löschen Sie den Kunden Sierl aus der Tabelle Kunden. 3 P

=====

Bearbeiten Sie die Aufgaben unter Zuhilfenahme der MySQL-Dokumentation !

12. Nennen Sie zwei wesentliche Unterschiede zwischen den Tabellentypen MyIsam und InnoDB.

13. Folgende Ausgangssituation: 3 P

Datenbankserver-Rechnername: AlphaCentauri  
 Datenbank-User: PalimPalim  
 Datenbank-Passwort: FooBar  
 Datenbankname: JohnDoe

Sie sitzen am Rechner 'Saturn' und möchten die Struktur (ohne Inhalt) der obigen Datenbank in die Datei **/home/user/steinam/FooBar.sql** dumpen. Als Klientprogramm steht auf ihrem Rechner das Programm **mysqldump** zur Verfügung. Die Erreichbarkeit und die Rechte auf den Maschinen und der Datenbank sind vorhanden.

Wie lautet der Befehl zum Erzeugen der Datenbankstruktur ?

# Ddl\_megasoft

Dienstag, 28. Januar 2020 09:45



megasoft



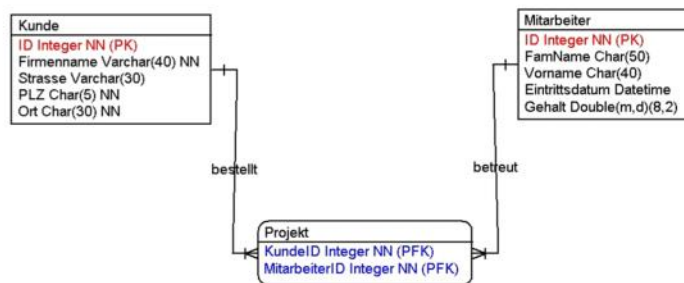
### Übungsaufgabe zu SQL-Statements:

Kommentiert [1]: <!--StartFragment-->

Die MegaSoft-GmbH erstellt Individualsoftware im Kundenauftrag. Die Verwaltung und Abrechnung der Projekte erfolgt über eine Datenbank. Auch die Kundendaten werden dort gespeichert. Insgesamt arbeiten in der MegaSoft GmbH zur Zeit 12 feste Mitarbeiter und 25 Freiberufler

Erzeugen Sie die **Datenbank** MegaSoft. Erstellen Sie nun alle Tabellen mittels SQL-Statements..

[1,1]



1. Tragen Sie nun folgende Daten ein:  
3 Datensätze für Kunden  
2 Datensätze für Mitarbeiter
2. Fügen Sie der Tabelle Projekt 2 Felder hinzu:  
Projektname: varchar(30) null unique  
Projektbeginn: datetime
3. Tragen Sie in die Tabelle Projekt 2 Datensätze ein. Benutzen Sie dabei alle Kunden und Mitarbeiter
4. Erstellen Sie eine weitere Tabelle tblProjekte\_alt, die im Aufbau mit der Projekttabelle identisch ist.
5. Fügen Sie alle Datensätze aus der Tabelle Projekt in die Tabelle tblProjekte\_alt hinzu.
6. Ändern Sie den Namen des ersten Kunden.
7. Tragen Sie nun zwei weitere Projekte in Ihre Datenbank ein. Testen Sie auch den Fall, daß ein Projekt von einem Sachbearbeiter erledigt wird, den Sie noch nicht angelegt haben.
8. Löschen Sie die Tabelle tblProjekte\_alt.
9. Fügen Sie der Projekt-Tabelle eine Spalte 'Projektmerkmale' hinzu.
10. Tragen Sie hier für das erste Projekt 'Access, ERM, Projektmanagement' ein. Welche Vor- bzw. Nachteile hat dieses Feld?
11. Löschen Sie die Spalte Projektmerkmale