Datenbanken und SQL –
Eine Einführung für Schüler mit MySQL
auf den Betriebssystemen Windows, Linux, Mac,
und Hinweisen zur Arbeit mit Oracle XE

Datenbanken und SQL

Eine Einführung für Schüler mit MySQL
auf den Betriebssystemen
Windows, Linux, Mac
und Hinweisen zur Arbeit mit Oracle XE

von
Norbert Kessel

www.verlagkessel.de

Verlag Kessel
Eifelweg 37
53424 Remagen-Oberwinter
Tel.: 02228-493
Fax: 03212-1024877
E-Mail: webmaster@forstbuch.de
Homepage: www.verlagkessel.de,
www.forstbuch.de
www.forestrybooks.com

ISBN: 978-3-945941-49-2

Inhalt

Einführende Hinweise 8
Zielgruppe, Werkzeuge 9
Wieso dieses Buch 9
Ein Vorwort für Schüler 10
... und eines für Erwachsene. 11
Eine erste SQL-Anweisung 12

Grundlagen bei der Arbeit mit Datenbanken 14
Datenbanken um uns herum. 15
Client und Server, eine wichtige Anmerkung. 16
Abgrenzung: Tabellen in Datenbanken, in der Textverarbeitung und in der Tabellenkalkulation 17
Was ist eine ‚richtige' Tabelle? 18
Wie kommt man zu einer Datenbank? 21
Datentypen 22
Für Musikliebhaber 24
Eine Lösung mit der Tabellenkalkulation 24
Die richtige Lösung mit einer Datenbank. 26
Skript-Datei zum Erstellen der Musik-Datenbank 34
Datenbanken aus kaufmännischer Sicht. 36
Eine Anekdote für die Fans der Tabellenkalkulation 36
Die Normalisierung von Daten 37
Drei Arten von Beziehungen zwischen Tabellen 41
1:n Beziehung 41
1:1 Beziehung 42
m:n Beziehung. 42

Datenbank und Workbench installieren 44
1. Mac 44
2. Windows 46
3. Linux 48

Datenbank und Workbench starten 51
1. Mac 51
2. Windows 53
3. Linux 54
Mac, Windows, Linux: Skriptdatei ausführen lassen. 55

Arbeiten mit SQL 57
Hinweise zur Arbeit mit MySQL Workbench 58
Allgemeine Hinweise zu SQL 63
Die Struktur der Sprache SQL, Begriffe 64
DDL (Data Definition Language) 65
CREATE DATABASE, DROP DATABASE. 65
CREATE TABLE, DROP TABLE 65
Einfache Tabelle für Freunde 66
Eine einfache Tabelle für Belege. 68

null, not null, 0 . . . 70
auto_increment . . . 75
Primary Key . . . 77
Primary Key und Foreign key in zwei Tabellen . . . 80
Löschen von Daten in zwei verknüpften Tabellen: on delete cascade . . . 81
Tabelle mit sehr vielen Datensätzen anfertigen . . . 85
Weitere Befehle zu Tabellen . . . 86
Tabellen aus dem Entity Relationship-Modell anfertigen lassen . . . 88
create index, drop index . . . 96
Ein Beispiel für den Gewinn an Geschwindigkeit . . . 98
Volltext-Suche mit FULLTEXT . . . 99
create view, drop view . . . 101
Die klassische Einteilung: Selektion, Projektion, Join . . . 101
DML (Data Manipulation Language) . . . 107
insert into mit Tabelle und View . . . 108
Weitere Möglichkeiten, Daten einzufügen . . . 109
Einfügen von CSV-Daten (Comma Separated Value) aus einer Access MDB . . . 109
Datenimport mit dem ‚Table Data Import Wizard‘ . . . 111
Datenimport mit der Anweisung LOAD DATA . . . 111
Der ‚safe update mode‘ für update- und delete-Anweisungen . . . 113
Eine Warnung vorweg: update ... ohne where-Klausel . . . 113
update . . . 114
delete . . . 118
DQL (Data Query Language) . . . 122
einfache Select-Anweisungen mit: all, distinct, distinctrow . . . 124
order by . . . 125
Select über mehrere Spalten . . . 130
group by, having . . . 131
subqueries (in, any, all, exists) . . . 132
union . . . 137
Joins . . . 138
Datenintegrität . . . 138
Inner Join (=Join) . . . 139
Left Join . . . 140
Right Join . . . 141
Bewusst herbeigeführte fehlerhafte Daten . . . 141
Primary Key und Foreign Key bei Tabellenerstellung zuweisen . . . 143
Pivot-Abfrage . . . 144
TCL (Transaction Control Language) . . . 147
DCL (Data Control Language) . . . 149
User und Passwörter . . . 149
Benutzer anlegen . . . 149
Passwort ändern . . . 150
Benutzer löschen . . . 150
Rechte zuweisen/entziehen mit grant und revoke . . . 150

Oracle XE – Installation und Einführung . . . 152
Installation . . . 152
Der Startbildschirm . . . 153
Das Schema und der User ‚HR‘ . . . 155
SQL-Anweisungen ausführen lassen . . . 159

Tabelle anlegen . . . 161
1. als Kopie einer bereits vorhandenen Tabelle . . . 161
2. mit der Anweisung CREATE TABLE . . . 162
3. Import von CSV-Daten . . . 164
Export von Tabellendaten im CSV-Format . . . 168
Anwendung erzeugen . . . 170
lokale Anwendung, mit den Daten von Beispiel-Tabellen . . . 170
Anwendung auf dem Server erstellt, mit eigenen Daten . . . 174
Anmelden, um Speicherplatz zu bekommen . . . 174
Neue Daten mit SQL Script . . . 177
Anwendung schreiben . . . 179
Anwendung auf Smartphone/Tablet ausführen lassen . . . 181

Anhang . . . 182
Einige Funktionen in SQL . . . 183
Aggregat-Funktionen . . . 183
String-Funktionen . . . 183
Datums- und Zeitfunktionen . . . 184
Mathematische Funktionen . . . 186
Beispiel-Datenbanken . . . 187
Download und Installation der World Database für MySQL . . . 187
Inhalte der Datenbank World . . . 189
Inhalte der Datenbank ‚sakila‘ . . . 190
Inhalte der Datenbank Menagerie . . . 193
Selbst gebaut: Datenbank für eine Tierarztpraxis . . . 194
Die Datenbank Nordwind (Access) . . . 196
Datenexport . . . 197
Export einer einzelnen Tabelle im CSV-Format . . . 197
Export der kompletten Datenbank . . . 198
Ergebnismenge einer select-Abfrage als Textdatei sichern . . . 199
Literatur . . . 201

Stichwortverzeichnis . . . 202

Einführende Hinweise

Zielgruppe, Werkzeuge

Dieses Buch ist für Schüler geschrieben, die schon etwas Übung im Umgang mit Computern und Office-Software haben. Zu Beginn wird anhand von einfachen Beispielen erläutert, was eine Datenbank ist, welche Art von Daten darin enthalten sein können und was man damit machen kann. Und schließlich wird die entscheidende Frage gestellt, warum uns das interessieren sollte.

Erst danach wird die Sprache SQL (Structured Query Language – strukturierte Abfragesprache) vorgestellt und es wird gezeigt, wie man damit auf Daten zugreifen kann. Im einfachsten Fall, um eine Liste anzuzeigen und sie ausdrucken zu lassen.

Von zentraler Bedeutung ist dabei die Software MySQL. Der Grund dafür ist, dass diese Software gratis aus dem Internet herunter geladen werden kann. Und zwar für alle drei gängigen Betriebssysteme Windows, Mac und Linux. Neben MySQL werden noch zwei andere Dinge benötigt: die MySQL Workbench, sozusagen die Werkbank, eine Software, über die man mit der Datenbank kommunizieren kann. Und schließlich Beispieldatenbanken, alles findet sich im Internet auf der Homepage von MySQL.

Neben MySQL ist auch ein Abschnitt zu Oracle XE enthalten. Auch diese Software vom Marktführer ist gratis im Internet zu bekommen.

Wieso dieses Buch

Die Sprache SQL, mit der man Informationen aus Datenbanken abfragen kann, hat sich weltweit durchgesetzt. Seit Jahrzehnten ist die Sprache standardisiert und lässt sich relativ leicht erlernen, da sie nur wenige Begriffe umfasst (weniger als 40). Außerdem entstammen diese Begriffe der ‚normalen Sprache': ‚delete' bedeutet zum Beispiel löschen, ‚update' bedeutet aktualisieren und ‚insert' bedeutet einfügen. Im richtigen Leben wie bei SQL auch.

> Ok, das ist etwas optimistisch ausgedrückt: wir müssten schreiben: der Kern der Sprache ist standardisiert, jeder Hersteller ‚erfindet' neue Begriffe, die nicht im Standard enthalten sind. Das führt dazu, dass eine Anweisung, die bei der einen Datenbank funktioniert, für eine andere Datenbank etwas abgeändert werden muss. Wir konzentrieren uns in diesem Buch meist auf den Standard und bleiben damit auf der sicheren Seite.

Somit lässt sich die Frage nach dem ‚Warum' bereits hier beantworten: die Sprache ist weltweit verbreitet, sie lässt sich leicht erlernen und man kann damit vom lokalen PC bis zum Großrechner auf Daten zugreifen, die in SQL-Datenbanken gespeichert sind. Außerdem macht es sich im Lebens-

lauf sehr gut, wenn man neben den üblichen Software-Produkten (Office-Pakete) auch etwas richtig Professionelles aufführen kann.

Ein Vorwort für Schüler ...

Einem Schüler zu erklären, was eine Datenbank ist, ist eigentlich ganz einfach, man kann es mit den Schülern einer Schule beschreiben: Alle Schüler einer Schule können zusammengefasst werden als ‚die Menge aller Schüler' (das bezeichnet man bei Datenbanken als Gesamtmenge). Als Teilmenge kann man die Schüler einer bestimmten Klasse bezeichnen.
Schüler haben Merkmale: Namen, Vornamen, Wohnorte und Telefon-Nummern. Diese Merkmale sind wichtig, oft interessieren wir uns für diese und möchten sie erfragen.
Man kann sich die gespeicherten Daten der Schüler folgendermaßen vorstellen:

Vorname	Nachname	Klasse	Schülernummer
Anna	Schulz	7c	67533
Paul	Müller	7c	68115
Lena	Maier	7c	68228
Jonas	Baier	7c	63498

Ein paar Anmerkungen zu der Tabelle mit Schülerdaten:

- im Beispiel sind nur vier Schüler einer Klasse, in normalen Klassen sind viel mehr Schüler;
- die Schülernummer ist leider nötig, denn in Schulen gibt es immer mehrere Kinder die ‚Anna' oder ‚Paul' heißen; wer denkt sich die Schülernummer aus? Das macht üblicherweise der Computer, der kann auch dafür sorgen, dass jede Nummer nur einmal vorkommt;

Man kann ganz unterschiedliche Daten in solchen Tabellen ablegen, zum Beispiel Informationen über Briefmarken oder Bücher, Adressdaten von Freunden oder Hinweise zu Musiktiteln.

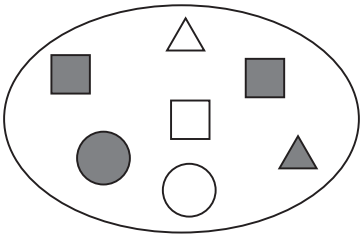
Man muss aber auch bedenken, dass es manchmal nicht sinnvoll ist, eine Datenbank anzulegen: kein Mensch hat mehr als 1000 Freunde, da würde eine einfache Namensliste genügen. Aber: hat man große Datenmengen, dann ist der Gebrauch einer Datenbank unbedingt zu empfehlen.

Zurück zu der Schulklasse: Wieso ist es nun sinnvoll, eine solche Tabelle anzulegen? Angenommen, die Schule möchte allen Schulkindern einen Brief schicken. Dann kann man diese Tabelle mit Schülerdaten mit der Textverar-

beitung verknüpfen. So kann mit wenig Aufwand ein Schreiben verschickt werden.

... und eines für Erwachsene

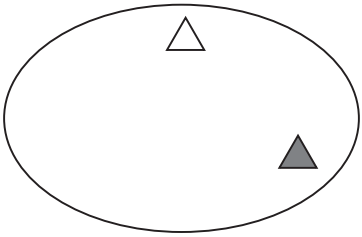
Möglicherweise hatten Sie in der Schule ‚Mengenlehre', das war das Teilgebiet im Mathematik-Unterricht, wo man rote Vierecke und blaue Dreiecke zeichnen musste, die zusammengefasst in einem sogenannten Venn-Diagramm (=Mengendiagramm) standen, das war ein Kreis oder ein Oval mit den genannten Elementen darin.



Venn-Diagramm, Gesamtmenge mit sieben Elementen

Falls Sie sich jemals gefragt haben, wozu das nützlich sein kann, dann erfahren Sie es jetzt: Die Arbeit mit Datenbanken und der Sprache SQL ist im Grunde dasselbe. Man schafft sich eine Menge mit Elementen und wählt einige davon aus, indem man ihre *Eigenschaften* beschreibt, zum Beispiel:

- zeige mir alle Dreieecke aus der Gesamtmenge aller Elemente
- zeige mir alle weißen Vierecke aus der Gesamtmenge aller Elemente



Venn-Diagramm mit zwei ausgewählten Elementen,
man bezeichnet die nun als Teil- oder Ergebnismenge

Übrtragen auf eine Tabelle mit Schülerdaten:

- zeige mir alle Schüler, die in der Stadt Mannheim wohnen
- zeige mir alle Schüler, deren Vornamen mit ‚A' beginnen

Eine erste SQL-Anweisung

Übertragen auf ein Datenbanksystem lautet die erste Anweisung:

```
SELECT vorname, name, wohnort
FROM schuelertabelle
WHERE wohnort = 'Mannheim'
```

Das ist leicht zu verstehen, hier ein paar Erläuterungen:

- SELECT ist die eigentliche SQL-Anweisung zur Auswahl der Daten; danach sind die Spalten genannt (*vorname, name, wohnort)* deren Inhalte angezeigt werden sollen (oft sind noch viel mehr Spalten in Tabellen enthalten)
- nach FROM folgt der Name der Tabelle, aus der die Informationen gelesen werden sollen
- WHERE leitet eine Bedingung ein, sie lautet hier: im Feld mit dem Namen ‚Wohnort' muss der Inhalt ‚Mannheim' stehen)

Wenn die Schülertabelle die folgenden Inhalte hat,

vorname	name	wohnort
Anna	Müller	Mannheim
Paul	Meyer	Speyer
Lena	Kress	Ludwigshafen
Jonas	Meier	Mannheim
Lisa	Schüller	Mannheim
Timo	Brauch	Heidelberg
Jan	Stark	Schifferstadt
Lars	Keller	Mannheim

dann könnte die Ergebnismenge (im Beispiel für den Wohnort) folgendermaßen aussehen:

vorname	name	wohnort
Anna	Müller	Mannheim
Jonas	Meier	Mannheim
Lisa	Schüller	Mannheim
Lars	Keller	Mannheim

Das ist nicht wirklich kompliziert. Und ist es nicht faszinierend, dass man vom kleinen PC bis hin zu den größten SQL-Datenbanken auf dieser Welt mit den gleichen SQL-Anweisungen arbeiten kann?

Zusammenfassung, Begriffe

Mengen-lehre	Theoretische Grundlage für die Arbeit mit Datenbanken. Umgangssprachlich: beschreibe mir die Eigenschaften der Ergebnismenge und ich liefere dir die einzelnen Elemente
SQL	Structured Query Language, eine standardisierte Sprache zur Arbeit mit Datenbanken.
SELECT	die wichtigste Anweisung der Sprache SQL. Mit dieser Anweisung filtert man Teilmengen aus einer Gesamtmenge heraus.
Tabelle	Tabellen enthalten Daten; es gibt immer mehrere Tabellen in einer Datenbank, die meist miteinander in Beziehung gesetzt werden. Neben den Tabellen, die man selbst erstellt, gibt es auch sogenannten Systemtabellen, die von der Datenbank angelegt werden. In ihnen werden beispielsweise die User-Namen gespeichert
Spalten, Felder	Wenn man Tabellen anlegt, dann definiert man verschiedene Spalten für verschiedene Arten von Informationen.So landen alle Vornamen in einer ersten Spalte und alle Nachnamen in einer zweiten Spalte. Das gleiche gilt für die Postleitzahl und den Wohnort, man verwendet jeweils eine separate Spalte dafür.

Grundlagen bei der Arbeit mit Datenbanken

In diesem theoretischen Teil werden Datenbanken und Tabellen vorgestellt. Wie zuvor erwähnt, ist die Datenbank die Heimat der Daten.

Datenbanken haben einen Dateinamen (wie ein Word- oder Excel-Dokument), sie sind aber viel größer und können Millionen Datensätze enthalten.

Die in den Datenbanken gespeicherten Tabellen haben zwar auch Namen, aber diese Namen sind keine (!) Dateinamen, sie tauchen in keiner Dateiübersicht auf, deshalb kann man sie mit dem *Betriebssystem* nicht kopieren oder löschen.

Die Grundlagen zu Datenbanken und Tabellen werden in diesem Abschnitt ganz ohne Computer vorgestellt, man kann dieses Kapitel getrost auf dem Liegestuhl lesen, durchaus auch mehrfach.

Datenbanken um uns herum

Wir alle verwenden Datenbanken, auch wenn wir es nicht unbedingt wissen. So sind zum Beispiel Millionen von Bücher, die im Buchhandel angeboten und verkauft werden, in einer Datenbank gespeichert. Diese Datenbank gehört der VLB-Agentur (VLB = Verzeichnis lieferbarer Bücher) in Frankfurt, sie gibt die Daten auch an Amazon weiter.

Auch die sehr polulären Internet-Seiten zum Verkauf von Gebrauchtwagen basieren auf Datenbanken. Dort kann man gezielt nach Hersteller, Farbe, Standort, und Leistung auswählen.

Aber auch viele Dinge, bei denen man nicht unbedingt an eine Datenbank denkt, werden in einer solchen gespeichert und verwaltet. Hier zunächst ein paar Beispiele aus dem Internet:

www.ebay.de	alle Artikel, alle Kundendaten sind in Datenbanken gespeichert.
www.buchhandel.de	Katalog lieferbarer Bücher (alle Bücher mit ISBN) Nach Eingabe von Buchtitel oder Autorennamen werden die im Buchhandel lieferbaren Bücher angezeigt. Für die Älteren unter uns: das war früher in den dicken blauen Büchern zu lesen (und hieß: VLB-Katalog des Buchhandels).
www.mobile.de www.autoscout24.de	Gebrauchtwagen (Stand Oktober 2018) 1,4 Millionen Automobile 1,1 Millionen Automobile
www.dasoertliche.de www.telefonbuch.de	Namen und Telefonnummern von Telefonkunden. Nach Eingabe von Namen und Ort werden die Telefonnummern (manchmal auch die Adressen) angezeigt.
www.zvab.de	Zentralverzeichnis Antiquarischer Bücher Viele (nicht alle) Antiquariate in Deutschland haben dort ihre Titel gelistet. Vergriffene Bücher (also die, die nicht mehr gedruckt werden) lassen sich dort finden.
www.freedb.org	Musikdatenbank Man kann online nach Interpreten, Alben und Titeln suchen.
www.pixelio.de www.pixabay.de	Bilder-Datenbanken: man kann Bilder nach Stichworten suchen, die Bilder sind meist frei verwendbar für eigene Aufsätze, Bücher und Homepages.
www.amazon.de	Alle Artikel sind in verschiedenen Datenbanken gespeichert.

Bei diesen Beispielen kann man bereits erkennen:

- Daten aus Datenbanken begegnen uns überall, oft sind sie nicht auf den ersten Blick erkennbar;
- einfache Daten (Telefonnummern) lassen sich oft in Form von Listen darstellen. Bei diesen Listen gibt es Spalten mit Überschriften (Vorname, Nachname, Straße, Ort).

Ein paar Beispiele für den Einsatz von Datenbanken:

- Musik bei Radiosendern: Früher legte man dort noch Schallplatten (oder CDs) auf, heute kommt die Musik aus einer Datenbank. Deshalb ist ein Titel sofort auffindbar und kann mit einem Mausklick abgespielt werden. Eine Anfrage bei WDR2 ergab, dass deren Musiktitel schon seit dem Jahr 1995 digitalisiert in einer Datenbank vorliegen;
- Giro-Konten bei Banken: Holt man sich Geld von der Bank, wird eine sogenannte Buchung ausgelöst. Diese Buchung erscheint später auf dem Kontoauszug;
- Verkehrssünderkartei in Flensburg: fährt man mit dem Auto zu schnell, bekommt man einen ‚Strafzettel'; die Strafzettel aller Verkehrsteilnehmer werden in einer Datenbank zentral gespeichert und können von dort jederzeit abgerufen werden, zum Beispiel von Polizisten bei einer Verkehrskontrolle;
- E-Mails, facebook, whatsapp, spotify und wie sie alle heißen: das eigene Profil, Vorlieben und alle Postings, die man macht, werden in Datenbanken gespeichert;
- Wikipedia: alle Artikel dieser Online-Enzyklopädie sind in Datenbanken gespeichert;
- Alle Internet-Seiten, die mit Wordpress erstellt sind, verwenden eine Datenbank (von MySQL).

Manche Datenbanken passen auf den Laptop, manche sind so groß, dass sie viele Rechner benötigen. Interessanterweise sind die kleinsten Versionen so klein, dass sie auf den Chip einer Kreditkarte passen. Allen gemeinsam ist aber, dass man auf die Daten mit der Sprache SQL zugreifen kann.

Nun gut, bei fast allen Datenbanken ist das so. Neben SQL-Datenbanken gibt es noch andere Datenbanken (Netzwerkdatenbanken, objektorientierte Datenbanken, hierarchische Datenbanken), die werden uns hier aber nicht beschäftigen.

Client und Server, eine wichtige Anmerkung

Eine wichtige Eigenschaft soll hier bereits erläutert werden: Wenn man mit dem Computer im Internet die Telefonnummer eines Menschen sucht, dann schickt man an die Datenbank im Internet eine Such-Anfrage. Diese lautet: ‚zeige mir die Telefonnummer von Anja Müller in Köln'.

Diese Anfrage wird abgeschickt und von einem anderen Computer bearbeitet, der irgendwo auf der Welt stehen kann. Von diesem Computer werden nun alle Menschen ausgewählt, die Anja Müller heißen und in Köln wohnen. Die Daten werden in Form einer Ergebnismenge an denjenigen geschickt, der danach gefragt hatte.

Man nennt dies ‚Client-Server-Architektur', es bedeutet, dass es einen Client gibt (hier: der oder die Suchende) und einen entfernt stehenden Server, der die Daten auswählt und an den oder die Suchende schickt. Typischerweise werden nur wenige Datensätze als Ergebnismenge verschickt.

Frühere, einfachere Datenverwaltungssysteme hatten ein anderes Konzept: dort wurden alle Daten zum Clienten geschickt und dessen Rechner hat anschließend ausgewählt. Von Nachteil war, dass immer große Mengen an Daten übertragen wurden, obwohl nur wenige tatsächlich gebraucht wurden.

Das folgende Schaubild fasst den Vorgang zusammen. Links ist der Laptop des Clients zu sehen, rechts ist ein Symbol für eine Datenbank abgebildet, typischerweise in Form einer Tonne.

Client-Server-Architektur: der Client schickt eine Anweisung an den Server, der führt diese auf dem Server aus und liefert eine Ergebnismenge zurück an den Client
Quelle: www.openclipart.org

Abgrenzung: Tabellen in Datenbanken, in der Textverarbeitung und in der Tabellenkalkulation

Tabellen gibt es überall, auch in der Textverarbeitung und in der Tabellenkalkulation. Allerdings sehen die nur so aus wie die Tabellen einer Datenbank, die folgende Übersicht vergleicht diese Typen.

Umgebung	Verwendung
Datenbanken	Datenbanken sind der Speicherort für die ‚echten' Tabellen. Von außen nicht sichtbar, sind diese Tabellen in einer Datenbank-Datei verpackt. Zusammen mit einer speziellen Software, dem Datenbank-Managemenst-System (DBMS) können leistungsstarke Anwendungen erstellt werden. Dabei kann die Zahl der Benutzer auf mehrere Tausend und die Zahl der Datensätze auf Millionen ansteigen. Interessant ist, wie wenig Zeit benötigt wird, um aus einer Tabelle mit beispielsweise einer Million Datensätze ein paar wenige auszuwählen.
Textverarbeitung	zur tabellarischen Darstellung von Texten oder Bildern; durch die einheitliche Ausrichtung schauen damit erstellte Listen immer sehr professionell aus.

Tabellen-kalkulation	Die mit Excel hergestellten Dateien haben die Extension ‚XLS‘, das steht für ‚Excel sheet‘, zu Deutsch: Arbeitsblatt. Von Tabelle keine Spur, eigentlich müssten die Excel-Anwender das Wort ‚Arbeitsblatt‘ verwenden. Hinzu kommt, dass in Excel zusätzlich der Begriff ‚Datenbank‘ für einen markierten Bereich eines Arbeitsblattes eingeführt wurde, damit war dann die Verwirrung komplett, denn selbstverständlich hat das mit einer richtigen Datenbank überhaupt nichts zu tun. Die Funktionen, die Excel dafür bereithält beginnen immer mit ‚DB‘, zum Beispiel ‚DBANZAHL()‘ oder ‚DBMAX()‘ (in der Online-Hilfe finden Sie die anderen Funktionen). Das Gleiche gilt für sogenannte ‚Tabellen‘, die mit LibreOffice bzw. OpenOffice hergestellt werden, dort haben sie die Extension ‚ODS‘, das ‚S‘ steht auch hier für „sheet“. Etwas anderes sind die MDB-Dateien, die innerhalb von LibreOffice/ OpenOffice geöffnet werden können: sie stammen ursprünglich von MS-Access und sind kleine Datenbank-Dateien. Auch sie können mit SQL verwendet werden. Diese MDB-Dateien sind für einzelne Anwender oder kleinere Netzwerke durchaus eine Alternative zu ‚richtigen‘, großen Datenbanken.

Was ist eine ‚richtige‘ Tabelle?

Wie zuvor erläutert, sind Tabellen immer Teil einer Datenbank. Man kann die Daten nach ihrem Inhalt oder ihrer Verwendung einteilen in *Stammdaten* und in *Bewegungsdaten*. Außerdem gibt es, wie oben schon erwähnt, die vom Anwender erstellten Tabellen und Systemtabellen.

Hier zum Beispiel zwei Tabellen mit Stammdaten.

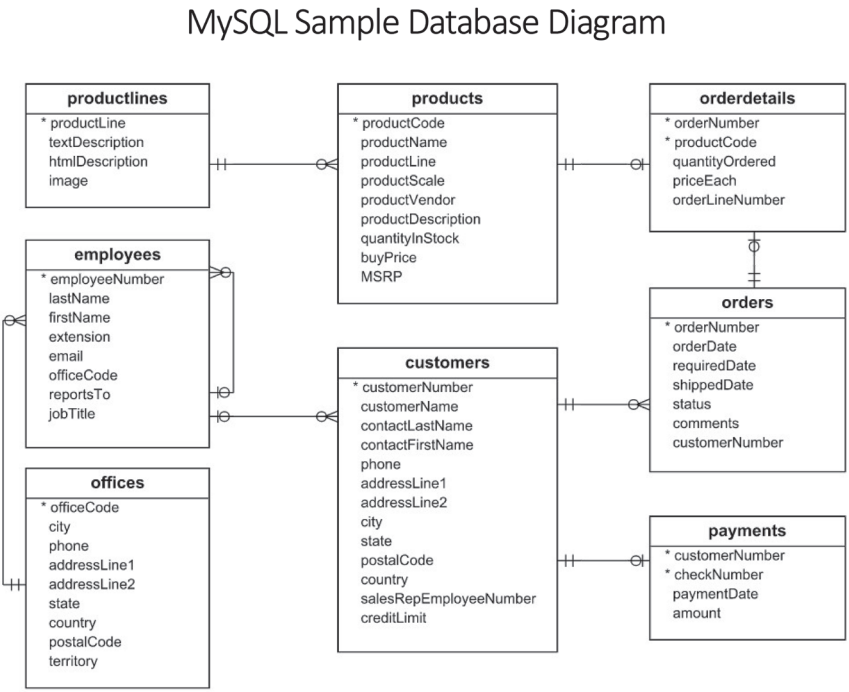
- Tabelle mit Kundendaten, sie enthält die Informationen zu Kunden: Kundennummern, Adressen, Kontaktdaten, Ansprechpartner;
- Tabelle mit Artikeldaten, sie enthält Informationen zu den Artikeln, die verkauft werden: Artikelnummern, Bezeichnungen, Preise, Lieferanten;

Stammdaten sind Tabellen, bei denen sich – nachdem sie einmal mit Daten gefüllt wurden – nicht mehr viel ändert. Im englischen spricht man auch von Master-Daten.

Daneben gibt es noch andere Tabellen, in die hinein *häufig* Daten geschrieben werden, zum Beispiel solche, die beim Verkauf von Artikeln anfallen. Aus ihr ergibt sich, wer, wann, wovon und wieviel gekauft hat. Das sind die sogenannten Detail-Daten.

Setzt man Detail-Daten in Verbindung mit den zuvor genannten Stamm-Daten, ergibt sich eine *Beziehung* (Relationen), bei vielen Tabellen spricht man von einem Netzwerk. Man spricht dann von einem *relationalen Datenbanksystem (RDBMS)*. Der Begriff ‚Master-Detail-Beziehung‘ kommt aus dieser Begriffs-Welt.

Hier ein Schema mit Tabellen, wie sie in einer Beispiel-Datenbank von MySql gespeichert sind (sie wird in vielen Beispielen dieses Buchs verwendet).



Übersicht über die in einer MySQL-Datenbank enthalten Tabellen und deren Beziehungen. Dieses Schema wird in der Literatur auch als ER-Diagramm bezeichnet, dabei steht ER für Entity Relationship, es versinnbildlicht die Relationen zwischen den Tabellen.

Ein paar Anmerkungen zu den Tabellen in der Beispiel-Datenbank:

- Später werden uns zuerst die in der Abbildung dargestellten Tabellen ‚customers' und ‚products' interessieren. Danach kommen die ‚orderdetails' und ‚orders' dazu.
- Die Linien deuten die Beziehungen an, in denen die Tabellen miteinander stehen. Sternchen (Asteriske) bedeuten, dass auf einem Feld ein sogenannter *Primärschlüssel* sitzt (engl.: *primary key*): er sorgt dafür, dass jeder Feldinhalt nur einmal vorkommt, zum Beispiel eine Kunden- oder Artikelnummer. Dies wird von der Datenbank überwacht.

Zeilen und Spalten: die Tabelle ‚employees'

Beispielhaft soll hier der Inhalt der Tabelle ‚employees' gezeigt werden. Sie enthält 23 Datensätze und besteht aus 8 Spalten.

Result Grid | Filter Rows: Search | Edit: | Export/Import:

employeeNumber	lastName	firstName	extension	email	officeCode	reportsTo	jobTitle
1002	Murphy	Diane	x5800	dmurphy@classicmodelcars.com	1	NULL	President
1056	Patterson	Mary	x4611	mpatterso@classicmodelcars.com	1	1002	VP Sales
1076	Firrelli	Jeff	x9273	jfirrelli@classicmodelcars.com	1	1002	VP Marketing
1088	Patterson	William	x4871	wpatterson@classicmodelcars.com	6	1056	Sales Manager (AP
1102	Bondur	Gerard	x5408	gbondur@classicmodelcars.com	4	1056	Sale Manager (EM
1143	Bow	Anthony	x5428	abow@classicmodelcars.com	1	1056	Sales Manager (NA
1165	Jennings	Leslie	x3291	ljennings@classicmodelcars.com	1	1143	Sales Rep
1166	Thompson	Leslie	x4065	lthompson@classicmodelcars.com	1	1143	Sales Rep
1188	Firrelli	Julie	x2173	jfirrelli@classicmodelcars.com	2	1143	Sales Rep
1216	Patterson	Steve	x4334	spatterson@classicmodelcars.com	2	1143	Sales Rep
1286	Tseng	Foon Yue	x2248	ftseng@classicmodelcars.com	3	1143	Sales Rep
1323	Vanauf	George	x4102	gvanauf@classicmodelcars.com	3	1143	Sales Rep
1337	Bondur	Loui	x6493	lbondur@classicmodelcars.com	4	1102	Sales Rep
1370	Hernandez	Gerard	x2028	ghernande@classicmodelcars.com	4	1102	Sales Rep
1401	Castillo	Pamela	x2759	pcastillo@classicmodelcars.com	4	1102	Sales Rep
1501	Bott	Larry	x2311	lbott@classicmodelcars.com	7	1102	Sales Rep
1504	Jones	Barry	x102	bjones@classicmodelcars.com	7	1102	Sales Rep
1611	Fixter	Andy	x101	afixter@classicmodelcars.com	6	1088	Sales Rep
1612	Marsh	Peter	x102	pmarsh@classicmodelcars.com	6	1088	Sales Rep
1619	King	Tom	x103	tking@classicmodelcars.com	6	1088	Sales Rep
1621	Nishi	Mami	x101	mnishi@classicmodelcars.com	5	1056	Sales Rep
1625	Kato	Yoshimi	x102	ykato@classicmodelcars.com	5	1621	Sales Rep
1702	Gerard	Martin	x2312	mgerard@classicmodelcars.com	4	1102	Sales Rep
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Inhalt der Tabelle ‚employees',
hervorgehoben ein Zeile (waagerecht) und eine Spalte (senkrecht)

Zur Erläuterung:

- waagerecht: die hervorgehobene *Zeile* markiert einen Datensatz; eine andere Bezeichnung dafür ist ‚row' oder ‚record'; dieser Begriff taucht in manchen Funktionen auf, die man in SQL verwenden kann;
- senkrecht: die hervorgehobene *Spalte* wird auch als *Feld* bezeichnet, der englische Ausdruck ist ‚column', auch dieses Wort kommt in SQL-Anweisung vor.

Reale und virtuelle Tabellen: Table und View (Query)

Neben den realen Tabellen (das sind die, in denen die Daten gespeichert sind), gibt es auch noch sogeneannte virtuelle Tabellen, die man auch ‚View' oder ‚Query' nennt. Man kann sie als eine spezielle Sicht auf die Daten beschreiben (beispielsweise zeigen sie Kunden aus dem Inland). Oben wurden bereits die Begriffe Gesamtmenge und Teilmenge eingeführt, die lassen sich hier verwenden. Die Gesamtmenge der Daten ist in der Tabelle gespeichert. Views dagegen holen eine Teilmenge davon und zeigen sie an.

Obwohl Views Daten anzeigen, sind in ihnen keine Daten enthalten, nur eine SQL-Anweisung, mit der die Daten aus einer Tabelle geholt und an-

gezeigt werden. Deshalb sind Views klein und benötigen nur wenig Speicherplatz.

Dieses Konzept ist phänomenal, denn fast immer benötigt man nur einen Teil der Daten, die in einer Tabelle enthalten sind. Beispielsweise alle unbezahlten Rechnungen oder alle Artikel eines bestimmten Lieferanten.

Schnellere Suche mit Index

Um Daten schneller zu finden, werden Indexe verwendet, das sind zusätzliche Hilfstabellen, die für schnelles Suchen verwendet werden. Man legt sie selbst an.

Die schnelle Suche funktioniert dann so wie ein Stichwortverzeichnis am Ende eines Buches. Sucht man einen Begriff, dann schaut man erst hinten im Stichwortverzeichnis nach, erfährt die Seitenzahl und schlägt vorne die entsprechende Seite im Buch auf.

Wie kommt man zu einer Datenbank?

Es gibt mehrere Arten, zu einer Datenbank zu kommen:

- man verwendet eine Software wie ‚MySQLWorkbench' und gibt die Anweisungen selbst ein;
- man lädt eine sogenannte Skript-Datei mit SQL-Anweisungen (die kann man im Internet finden oder selbst erstellen) und lässt die darin enthaltenen SQL-Anweisungen von ‚MySQL Workbench' ausführen. Mit Skript-Dabeien kann man eine Datenbank erzeugen, Tabellen anlegen und Daten in die Tabellen schreiben.

Offen gestanden ist es recht langweilig, die ganzen Anweisungen zum Anfertigen von Tabellen manuell einzugeben, deshalb benutzen wir für das Beispiel eine Skript-Datei, die von der MySQL-Homepage heruntergeladen werden kann. Sie findet sich dort:

http://www.mysqltutorial.org/mysql-sample-database.aspx

Die Script-Datei hat folgenden Namen:

mysqlsampledatabase.sql

Im Script enthalten sind zum Beispiel folgende Anweisungen:

```
CREATE DATABASE ...   (erzeugt eine neue Datenbank)
CREATE TABLE ...      (erzeugt eine neue Tabelle)
INSERT INTO ...       (fügt einen Datensatz in eine Tabelle ein)
```

Ein Skript ist eine normale Text-Datei ohne jegliche Formate, es kann mit jeder Textverarbeitung geöffnet und eingesehen werden. Allerdings darf es nicht mit Formatierungen gespeichert werden, sondern nur als reiner Text.

Die Verwendung von Skript-Dateien ist elegant, denn man kann sie – wenn sie einmal erstellt sind – auch auf anderen Rechnern ausführen lassen und kommt somit zu identischen Datenbanken.

Außerdem kann man, wenn man seine Daten zerstört hat, diese mit einer Skript-Datei neu aufbauen lassen.

Datentypen

Wir wissen, dass es verschiedene Arten von Daten gibt. Im Gegensatz zur Tabellenkalkulation, wo man alle möglichen Daten einfach so auf ein Arbeitsblatt schreiben kann, muss man bei Datenbanken immer festlegen, was gespeichert werden soll: ein Name, eine Zahl, ein Datum oder ein Foto.

Eine wichtige Frage dabei ist: muss mit der zu speichernden Information gerechnet werden? So verführt der Begriff ‚Telefonnummer' dazu, diese Zeichen als Zahl zu speichern. Aber muss damit jemals gerechnet werden? Nein.

Gleiches gilt für die ‚Postleitzahl' oder die ‚Hausnummer': Mit diesen Informationen muss niemals gerechnet werden, deshalb werden diese Informationen als Zeichen (character) gespeichert.

MySQL listet mittlerweile mehr als 50 Datentypen auf von denen uns hier aber nur wenige beschäftigen werden, sie sind in der folgenden Übersucht zusammengefasst:

Art der Information	zum Beispiel	Datentyp
ganze Zahlen, mit denen gerechnet werden soll	Rechnungsnnummer, Artikelnummer, Kundennummer, Lieferantennummer. Neue Kunden- und Rechnungsnummern kann das System berechnen. Sollen in den Kundennummern (oder Rechnungsnummern) Buchstaben enthalten sein, dann müssen sie allerdings vom Typ ‚character' sein)	integer
Zahl, mit der nicht gerechnet werden soll	Postleitzahl, Hausnummer (Hausnummern werden zusammen mit den Straßennamen gespeichert)	character
Namen	Vorname, Nachname, Ortsname, Straßenname, allgemeiner Text mit variabler Länge	varchar
Geburtsdatum, Rechnungsdatum	hierfür gibt es einen speziellen Datentyp, mit dem man zum Beispiel Differenzen zwischen zwei Datumswerten oder das Alter berechnen kann	date
Zeichen mit variabler Länge	Beschreibung eines Artikels, Straßenname, Ortsname	varchar
Zahl	Rechnungsbetrag	decimal

Zusammenfassung, Begriffe

Datenbank	Die Heimat der Daten. In ihr sind enthalten: • *Tabellen* für die Daten, • View (Query), besondere Ansichten für die Tabellendaten. Mit Views können auch Daten aus verschiedenen Tabellen zusammengeführt und angezeigt werden. So könnte man für die Person, die sich um den Zahlungseingang kümmert, eine View für unbezahlte Rechnungen machen • Indexe für die schnellere Suche, Daneben sind auch noch *stored Procedures* und *Functions* enthalten, gespeicherte Anweisungen, die in bestimmten Situationen ausgeführt werden. Zum Teil starten sie automatisch, wenn eine bestimmte Bedingung eintritt (zum Beispiel wenn eine Rechnung auch nach einem Monat noch nicht bezahlt ist).
Tabellen mit Stammdaten (Master-Daten)	Tabellen mit Master-Daten beschreiben Dinge, die für unsere Arbeit von Interesse sind: Artikelnamen, Einkaufspreise, Verkaufspreise und Artikelnummern. Auch Personaldaten sind Masterdaten: Besipielsweise Personalnummer, Name, Vorname, Abteilung.
Tabellen mit Detail-Daten	Als Detail-Daten bezeichnet man die Daten, in denen die Buchungen stehen, bei kaufmännischen Anwendungen das, was verkauft wird, wofür eine Rechnung geschrieben wird.
Master-Detail-Beziehung	Fast immer stehen die Master-Daten mit Detail-Daten in Beziehung: eine Tabelle mit Artikeldaten mit der Tabelle, in der die Rechnungen gespeichert sind. Oder eine Tabelle mit Lieferantendaten mit der Tabelle der Artikeldaten.
Client-Server	Eine besondere Architektur, in der ein Client eine Anweisung zu einem Server schickt. Diese Anweisung wird vom Server ausgeführt. Häufig schickt der Server dann eine Ergebnismenge zurück zum Client. Oder einen Hinweis, dass es keine Ergebnismenge gibt. Der Server muss nicht entfernt stehen, man kann Client-Server-Anwendungen auch auf demselben Rechner installieren und benutzen.
Datentypen	Man verwendet verschiedene Datentypen zum Speichern von Daten. Die am meisten verwendeten sind • ganze Zahlen (integer), • Dezimalzahlen (decimal), • Zeichenketten mit fester Länge (char) und • Zeichenketten mit variabler Länge (varchar). • Nicht zu vergessen die Datumswerte (date).
Skripte	Sie erlauben es, eine Reihe von SQL-Anweisungen auf Knopfdruck ausführen zu lassen, jederzeit und immer wieder.

Für Musikliebhaber

Eine Lösung mit der Tabellenkalkulation

Im folgenden Beispiel werden Lieder von verschiedenen Interpreten vorgestellt. Anhand dieser Daten soll gezeigt werden, was meist falsch gemacht wird und wie man es besser machen kann.

Zuerst die Rohdaten, eingegeben in der Tabellenkalkulation, wie z.B. Excel oder LibreOffice/OpenOffice:

	A	B	C
1	**Interpret**	**Album**	**Lied**
2	Avicii	True: Avicii By Avicii	Wake Me Up
3			You Make Me
4			Hey Brother
5			Addicted To You
6			Dear Boy
7			Liar Liar
8			Shame On Me
9			Lay Me Down
10			Hope There's Someone
11			
12	Maroon 5	V	Maps
13			Animals
14			It Way Always You
15			Unkiss Me
16			Sugar
17			Leaving California
18			In Your Pocket
19			New Love
20			Coming Back For You
21			Feelings
22			
23	Maroon 5	Songs About Jane	Harder to Breathe
24			This Love
25			Shiver
26			She Will be Loved
27			Tangled
28			The Sun
29			Must Get Out
30			Sunday Morning
31			Secret
32			Through With You
33			
34	David Guetta	Listen	Dangerous
35			What I did for Love
36			No Money no Love

Interpreten, Alben und Titel von Liedern, dargestellt mit einer Tabellenkalkulation

Jeder Interpret und jedes Album wird zunächst nur einmal eingegeben. Die Daten sind ‚unvollständig', um sie vollständig zu machen, müsste man immer wieder die gleichen Daten eingeben oder die Daten kopieren.

An dieser Stelle verwenden die meisten Benutzer und Freunde der Tabellenkalkulation nun eine Funktion mit dem Namen ‚Autoausfüllen()' oder sie gehen mit der Maus an die untere rechte Ecke der Zelle deren Inhalt kopiert werden soll, klicken und bewegen dann die Maus nach unten. Dadurch werden die Zellen darunter mit dem Inhalt der ursprünglich markierten Zelle ausgefüllt.

Nach getaner Arbeit schaut das dann folgendermaßen aus:

	A	B	C
1	**Interpret**	**Album**	**Lied**
2	Avicii	True: Avicii By Avicii	Wake Me Up
3	Avicii	True: Avicii By Avicii	You Make Me
4	Avicii	True: Avicii By Avicii	Hey Brother
5	Avicii	True: Avicii By Avicii	Addicted To You
6	Avicii	True: Avicii By Avicii	Dear Boy
7	Avicii	True: Avicii By Avicii	Liar Liar
8	Avicii	True: Avicii By Avicii	Shame On Me
9	Avicii	True: Avicii By Avicii	Lay Me Down
10	Avicii	True: Avicii By Avicii	Hope There's Someone

Dies wiederholt man dann in der Tabellenkalkulation so lange, bis alle ‚leeren' Zellen mit Informationen gefüllt sind. Bei Kopieren von ‚Maroon 5' gibt es Probleme, da automatisch mitgezählt wird, das muss dann manuell korrigiert werden.

Nach getaner Arbeit schaut die Übersicht folgendermaßen aus:

	A	B	C
1	**Interpret**	**Album**	**Lied**
2	Avicii	True: Avicii By Avicii	Wake Me Up
3	Avicii	True: Avicii By Avicii	You Make Me
4	Avicii	True: Avicii By Avicii	Hey Brother
5	Avicii	True: Avicii By Avicii	Addicted To You
6	Avicii	True: Avicii By Avicii	Dear Boy
7	Avicii	True: Avicii By Avicii	Liar Liar
8	Avicii	True: Avicii By Avicii	Shame On Me
9	Avicii	True: Avicii By Avicii	Lay Me Down
10	Avicii	True: Avicii By Avicii	Hope There's Someone
11			
12	Maroon 5	V	Maps
13	Maroon 5	V	Animals
14	Maroon 5	V	It Way Always You
15	Maroon 5	V	Unkiss Me
16	Maroon 5	V	Sugar
17	Maroon 5	V	Leaving California
18	Maroon 5	V	In Your Pocket
19	Maroon 5	V	New Love
20	Maroon 5	V	Coming Back For You
21	Maroon 5	V	Feelings
22			
23	Maroon 5	Songs About Jane	Harder to Breathe
24	Maroon 5	Songs About Jane	This Love
25	Maroon 5	Songs About Jane	Shiver
26	Maroon 5	Songs About Jane	She Will be Loved
27	Maroon 5	Songs About Jane	Tangled
28	Maroon 5	Songs About Jane	The Sun
29	Maroon 5	Songs About Jane	Must Get Out
30	Maroon 5	Songs About Jane	Sunday Morning
31	Maroon 5	Songs About Jane	Secret
32	Maroon 5	Songs About Jane	Through With You
33			
34	David Guetta	Listen	Dangerous
35	David Guetta	Listen	What I did for Love
36	David Guetta	Listen	No Money no Love
37	David Guetta	Listen	Lovers on the Sun
38	David Guetta	Listen	Goodbye Friend
39	David Guetta	Listen	Lift me up
40	David Guetta	Listen	Listen
41	David Guetta	Listen	Bang my Head
42	David Guetta	Listen	Yesterday
43	David Guetta	Listen	Hey Mama
44			

Musikdaten, die Inhalte von Interpreten und Alben wurden mit der Funktion ‚Autoausfüllen()' aufgefüllt

Die gute Nachricht ist: Die Daten sind nun vollständig und für Ihre private Musiksammlung durchaus zu gebrauchen. Die schlechte Nachricht ist: für professionelle Ansprüche ist diese Lösung unbrauchbar. Warum?

Die gleichen Daten sind mehrfach gespeichert, viel besser wäre es doch, wenn man z.B. die Namen der Interpreten überhaupt nur *einmal* speichern würde. Es wäre platzsparender und bei Änderungen (Rechtschreibkorrektur, Änderung/Erweiterung des Namens) wäre nur eine (!) Änderung nötig.

Oder stellen Sie sich vor, es kämen noch Adressdaten dazu: bei jeder Ergänzung oder Änderung müsste man sicherstellen, dass die Daten komplett geändert werden. Das lässt sich bei großen Datenmengen überhaupt nicht mehr garantieren.

Redundante Daten

Man nennt diese mehrfach gespeicherten Daten auch ‚redundant'. Es ist ein wichtiger Gedanke beim Aufbau von Datenbank-Tabellen, dass man redundante Daten möglichst vermeidet.

Die richtige Lösung mit einer Datenbank

Wie schaut nun eine ‚richtige' Lösung aus? Man versucht erst gar nicht alles in eine Tabelle zu schreiben, man legt stattdessen mehrere Tabellen an:

- eine Tabelle für die Interpreten,
- eine Tabelle für die Alben und
- eine Tabelle für die Liedtitel.

Es wird nun gezeigt, wie man Tabellen mit der SQL-Anweisung CREATE TABLE anfertigt. Am Ende dieses Abschnitts werden alle SQL-Anweisungen optimiert und zusammengetragen, so dass eine erste SQL-Skript-Datei entsteht.

Tabelle für die Interpreten

Hier ein Entwurf für die Tabelle der Interpreten:

InterpretNummer	Interpret
1	Avicii
2	Maroon 5
3	David Guetta
4	...

Hier fällt als erstes auf, dass eine zusätzliche Spalte eingefügt wurde: Die Nummer des Interpreten. Über diese Nummer wird diese Tabelle später mit einer anderen Tabelle verknüpft.

Bei der Interpretnummer ist es wichtig, dass jede Nummer nur einmal vorkommt. Ganz wichtig: Beim Löschen eines Datensatzes bleiben die restlichen Nummern so, wie sie waren, wer also einmal die Nummer 3 hatte, behält sie; sie wird auch später nicht wieder erneut verwendet.

Ein Hinweis zu den Spaltennamen: Um Schwierigkeiten aus dem Wege zu gehen, verwenden wir generell die Spaltennamen ohne jegliche Sonderzeichen. Das heißt Umlaute, ‚ß' und auch der Bindestrich ‚-' (der ja in Wirklichkeit ein Minuszeichen ist), lassen wir einfach weg. Es ist zwar möglich, solche Zeichen innerhalb von SQL zu benutzen, aber diese müssten dann immer speziell formatiert werden. Das ist umständlich und beim Datenaustausch mit anderen Programmen kann es trotzdem schwierig werden. Viele Programme fühlen sich als ‚Amerikaner': dort käme kein Mensch auf die Idee, einen Umlaut zu benutzen. Bewährt hat sich außerdem die oben gezeigte Schreibweise: ‚InterpretNummer' anstelle von ‚interpretnummer'. Wenn man größere Projekte betreut, mit anderen Menschen im Team zusammenarbeitet oder nach längerer Zeit wieder alte Anweisungen liest, fällt es viel leichter, wenn man so verfährt.

Bevor man Tabellen anfertigt, muss man entweder eine neue Datenbank anlegen oder eine vorhandene öffnen. Zum Anlegen einer neuen Datenbank gibt man die folgende Anweisung ein:

```
CREATE DATABASE musik;
```

Ist die Datenbank schon vorhanden, dann öffnet man sie mit der folgenden Anweisung:

```
USE database musik;
```

Die SQL-Anweisung zum Anfertigen der ersten Tabelle hat die folgende Form:

```
CREATE TABLE Interpreten
(
InterpretNummer integer,
InterpretName varchar(50)
);
```

Dass die Klammern alleine in einer Zeile stehen ist nicht erforderlich, sie können auch an die anderen Zeichen angehängt werden, zum Beispiel so:

```
CREATE TABLE Interpreten(
InterpretNummer integer,
InterpretName varchar(50));
```

Tabelle für Alben

Hier die zweite Tabelle mit den Alben-Namen, auch hier erhält jedes Album eine Nummer und jede Nummer kommt nur einmal vor:

AlbumNummer	AlbumName
1	True: Avicii By Avicii
2	V
3	Songs about Jane
4	Listen
5	...

Die SQL-Anweisung für diese Tabelle unterscheidet sich kaum von der ersten, ihre Struktur ist ja sehr ähnlich:

```
CREATE TABLE Alben
(
AlbumNummer integer,
AlbumName varchar(50)
);
```

Tabelle für Lieder

Und schließlich die Tabelle für die Lieder. Sie hat enthält später die meisten Datensätze, denn für jedes Lied wird eine Zeile gebraucht. Da jedes Lied einen Interpreten hat und sich auf einem bestimmten Album befindet, kommen die Interpret- und Album-Nummern mehrfach vor. Das ist zwar auch redundant, aber ganz ohne Redundanzen geht es nicht.

InterpretNummer	AlbumNummer	LiedTitel
1	1	Wake Me Up
1	1	You Make Me
1	1	Hey Brother
1	1	Addicted To You
1	1	Dear Boy
1	1	Liar Liar
1	1	Shame On Me
1	1	Lay Me Down
1	1	Hope There‘s Someone
2	2	Maps
2	2	Animals
2	2	It Way Always You
2	2	Unkiss Me
2	2	Sugar
2	2	Leaving California
2	2	In Your Pocket
2	2	New Love
2	2	Coming Back For You
2	2	Feelings

2	3	Harder to Breathe
2	3	This Love
2	3	Shiver
2	3	She Will be Loved
2	3	Tangled
2	3	The Sun
2	3	Must Get Out
2	3	Sunday Morning
2	3	Secret
2	3	Through With You
3	4	Dangerous
3	4	What I did for Love
3	4	No Money no Love
3	4	Lovers on the Sun
3	4	Goodbye Friend
3	4	Lift me up
3	4	Listen
3	4	Bang my Head
3	4	Yesterday
3	4	Hey Mama
...	...	...

Hier die SQL-Anweisung zum Anfertigen dieser Tabelle:

```
CREATE TABLE Lieder
(
InterpretNummer integer,
AlbumNummer integer,
Liedtitel varchar(50)
);
```

Nachdem drei Tabellen angefertigt sind, können Daten eingegeben werden (den Tabellen-Namen ist der Datenbank-Name vorangestellt, das ist nicht immer nötig, sorgt aber für Klarheit).

Einfügen von Daten in die Tabelle Interpreten

Hiermit werden drei Datensätze in die Tabelle der Interpreten eingefügt:

```
INSERT INTO Musik.Interpreten
(InterpretNummer, InterpretName)
VALUES
(1, 'Avicii'),
(2, 'Maroon 5'),
(3, 'David Guetta');
```

Einfügen von Daten in die Tabelle Alben

Die INSERT-INTO-Anweisung für die Tabelle Alben (vier Datensätze):

```
INSERT INTO Musik.Alben
(AlbumNummer, AlbumName)
VALUES
(1, 'True: Avicii By Avicii'),
(2, 'V'),
(3, 'Songs about Jane'),
(4, 'Listen');
```

Einfügen von Daten in die Tabelle Lieder (nur ein paar Titel)

```
INSERT INTO Musik.Lieder
(InterpretNummer, AlbumNummer, Liedtitel)
VALUES
(1, 1, 'Wake Me Up'),
(1, 1, 'You Make Me'),
(1, 1, 'Hey Brother'),
(1, 1, 'Addicted To You'),
(1, 1, 'Dear Boy'),
(1, 1, 'Liar Liar'),
(1, 1, 'Shame On Me'),
(1, 1, 'Lay Me Down'),
(1, 1, 'Hope There\'s Someone'),
(2, 2, 'Maps'),
(2, 2, 'Animals'),
(2, 2, 'It Way Always You'),
(2, 2, 'Unkiss Me'),
(2, 2, 'Sugar'),
(2, 2, 'Leaving California'),
(2, 2, 'In your Pocket'),
(2, 2, 'New Love'),
(2, 2, 'Coming Back For You'),
(2, 2, 'Feelings'),
(2, 3, 'Harder to Breathe'),
(2, 3, 'This Love'),
(2, 3, 'Shiver'),
(2, 3, 'She Will be Loved');
```

Bei den drei INSERT-INTO-Anweisungen sind vor dem Schlüsselwort VALUES die Feldnamen aufgeführt und zwar in der Reihenfolge, wie sie gefüllt werden sollen. Das ist nicht zwingend erforderlich, es geht auch ohne, also so:

```
INSERT INTO Musik.Lieder
VALUES
(1, 1, 'Wake Me Up'),
...
```

Aber: nur dann, wenn man die Feldnamen nennt, kann man wirklich sicher sein, dass die Daten in den richtigen Spalten landen.

Anzeigen der Datensätze aus den drei Tabellen

Nachdem Daten in den Tabellen eingefügt wurden, können sie einzeln oder zusammengeführt angezeigt werden. Hierzu verwendet man die SELECT-Anweisung. Zuerst in der einfachen Form mit Daten jeweils einer Tabelle, danach mit den Daten aus allen drei Tabellen:

```
select * FROM Interpreten;
```

InterpretNumm...	InterpretName
1	Avicii
2	Maroon 5
3	David Guetta

Die Liste der Interpreten

```
select * FROM Alben;
```

AlbumNummer	AlbumName
1	True: Avicii By Avicii
2	V
3	Songs about Jane
4	Listen

Die Liste der Alben

```
select * FROM Lieder;
```

InterpretNumm...	AlbumNummer	Liedtitel
1	1	Wake Me Up
1	1	You Make Me
1	1	Hey Brother
1	1	Addicted To You
1	1	Dear Boy
1	1	Liar Liar
1	1	Shame On Me
1	1	Lay Me Down
1	1	Hope There's Someone
2	2	Maps
2	2	Animals
2	2	It Way Always You
2	2	Unkiss Me
2	2	Sugar
2	2	Leaving California
2	2	In your Pocket
2	2	New Love
2	2	Coming Back For You
2	2	Feelings
2	3	Harder to Breathe
2	3	This Love
2	3	Shiver
2	3	She Will be Loved

Die Liste der Lieder

Durch eine etwas aufwändigere SQL-Anweisung, in der die drei Tabellen zusammengeführt werden, können die Daten nun in einer gemeinsamen

Liste angezeigt werden. Hierzu müssen zwei Bedingungen formuliert werden:

1. die InterpretNummer aus der Tabelle Interpreten muss der InterpretNummer aus der Tabelle Lieder entsprechen;
2. die AlbumNummer aus der Tabelle Alben muss der AlbumNummer der Tabelle Lieder entsprechen.

Dies macht man mit der folgenden SQL-Anweisung, in der zuerst die gewünschten Felder aufgezählt werden. Danach kommen die Namen der Tabellen und schließlich die zwei Bedingungen:

```
SELECT Interpreten.InterpretName, Alben.AlbumName, Lieder.Liedtitel
FROM Interpreten, Alben, Lieder
WHERE Interpreten.InterpretNummer = Alben.AlbumNummer
AND Lieder.InterpretNummer = Interpreten.InterpretNummer
```

Zur Erläuterung:

- die SELECT-Anweisung zählt die Felder auf, die angezeigt werden sollen; damit klar ist, aus welcher Tabelle die stammen, schreibt man den Tabellennamen davor;
- nach FROM sind die verwendeten drei Tabellen genannt;
- die beiden Bedingungen finden sich nach WHERE; mit dem Wort AND wird festgelegt, dass beide Bedingungen erfüllt sein müssen

InterpretName	AlbumName	Liedtitel
Avicii	True: Avicii By Avicii	Wake Me Up
Avicii	True: Avicii By Avicii	You Make Me
Avicii	True: Avicii By Avicii	Hey Brother
Avicii	True: Avicii By Avicii	Addicted To You
Avicii	True: Avicii By Avicii	Dear Boy
Avicii	True: Avicii By Avicii	Liar Liar
Avicii	True: Avicii By Avicii	Shame On Me
Avicii	True: Avicii By Avicii	Lay Me Down
Avicii	True: Avicii By Avicii	Hope There's Someone
Maroon 5	V	Maps
Maroon 5	V	Animals
Maroon 5	V	It Way Always You
Maroon 5	V	Unkiss Me
Maroon 5	V	Sugar
Maroon 5	V	Leaving California
Maroon 5	V	In your Pocket
Maroon 5	V	New Love
Maroon 5	V	Coming Back For You
Maroon 5	V	Feelings
Maroon 5	V	Harder to Breathe
Maroon 5	V	This Love
Maroon 5	V	Shiver
Maroon 5	V	She Will be Loved

Die aus drei Tabellen zusammengeführten Daten werden in einer Liste angezeigt

Schema mit den Beziehungen

Man kann die Beziehung zwischen den Tabellen als Schaubild darstellen:

Interpreten

Interpret-Nummer	Interpret
1	Avicii
2	Maroon 5
3	David Guetta
4	...

Alben

Album-Nummer	Album-Name
1	True: Avicii By Avicii
2	V
3	Songs about Jane
4	Listen
5	...

Lieder

Interpret-Nummer	Album-Nummer	Liedtitel
1	1	Wake Me Up
1	1	You Make Me
1	1	Hey Brother
1	1	Addicted To You
1	1	Dear Boy
1	1	Liar Liar
1	1	Shame On Me
1	1	Lay Me Down
1	1	Hope There's Someone
2	2	Maps
2	2	Animals
2	2	It Way Always You
2	2	Unkiss Me
2	2	Sugar
2	2	Leaving California
2	2	In Your Pocket
2	2	New Love
2	2	Coming Back For You
2	2	Feelings
2	3	Harder to Breathe
2	3	This Love
2	3	Shiver
2	3	She Will be Loved
2	3	Tangled
2	3	The Sun
2	3	Must Get Out
2	3	Sunday Morning
2	3	Secret
2	3	Through With You
3	4	Dangerous
3	4	What I did for Love
3	4	No Money no Love
3	4	Lovers on the Sun
3	4	Goodbye Friend
3	4	Lift me up
3	4	Listen
3	4	Bang my Head
3	4	Yesterday
3	4	Hey Mama
...	...	...

Über gemeinsame Felder werden die drei Tabellen miteinander verknüpft. Dieses Schema ist freihändig gezeichnet. Später wird gezeigt, wie man mit MySQL das professionell ausschauende ER-Diagramm (Entity Relationship) aus dem Kapitel ‚Grundlagen' anfertigen kann.

Skript-Datei zum Erstellen der Musik-Datenbank

In dieser Datei werden die SQL-Anweisungen zusammengefasst, die auf den vorhergehenden Seiten verwendet wurden. Sie können von der Homepage des Verlags heruntergeladen werden.

Das Zeichen ‚#' am Anfang einer Zeile macht diese Zeile zu einer Kommentar-Zeile, damit beschreibt man für sich oder andere, was die folgende Anweisung macht.

```
#————————————————————————————————
# Datenbank Musik löschen, falls die bereits vorhanden ist
# damit sind alle Tabellen, Views, Indexe und Daten gelöscht
DROP DATABASE musik;
#————————————————————————————————
# Anfertigen der Datenbank Musik
CREATE DATABASE Musik;
#————————————————————————————————
# musik zur aktuellen Datenbank machen
USE Musik;
#————————————————————————————————
# Tabelle Interpreten erstellen
CREATE TABLE Interpreten
(
  InterpretNummer integer,
  InterpretName VARCHAR(50)
);
#————————————————————————————————
# Tabelle Alben erstellen
CREATE TABLE Alben
(
AlbumNummer integer,
AlbumName varchar(50)
);
#————————————————————————————————
# Tabelle Lieder erstellen
CREATE TABLE Lieder
(
InterpretNummer integer,
AlbumNummer integer,
Liedtitel varchar(50)
);
#————————————————————————————————
# einfügen von Datensätzen in die Tabelle Interpreten
INSERT INTO Musik.Interpreten
values
(1, 'Avicii'),
(2, 'Maroon 5'),
(3, 'David Guetta')
;
#————————————————————————————————
# einfügen von Datensätzen in die Tabelle Alben
INSERT INTO Musik.Alben
```

```
values
(1, 'True: Avicii By Avicii'),
(2, 'V'),
(3, 'Songs about Jane'),
(4, 'Listen')
;
#————————————————————————————————
# einfügen von Datensätzen in die Tabelle Lieder
INSERT INTO Musik.Lieder
values
(1, 1, 'Wake Me Up'),
(1, 1, 'You Make Me'),
(1, 1, 'Hey Brother'),
(1, 1, 'Addicted To You'),
(1, 1, 'Dear Boy'),
(1, 1, 'Liar Liar'),
(1, 1, 'Shame On Me'),
(1, 1, 'Lay Me Down'),
(1, 1, 'Hope There\'s Someone'),
(2, 2, 'Maps'),
(2, 2, 'Animals'),
(2, 2, 'It Way Always You'),
(2, 2, 'Unkiss Me'),
(2, 2, 'Sugar'),
(2, 2, 'Leaving California'),
(2, 2, 'In your Pocket'),
(2, 2, 'New Love'),
(2, 2, 'Coming Back For You'),
(2, 2, 'Feelings'),
(2, 3, 'Harder to Breathe'),
(2, 3, 'This Love'),
(2, 3, 'Shiver'),
(2, 3, 'She Will be Loved')
;

#————————————————————————————————
# Anzeigen der Datensätze aus drei Tabellen
SELECT Interpreten.InterpretName, Alben.AlbumName, Lieder.Liedtitel
FROM Interpreten, Alben, Lieder
WHERE Interpreten.InterpretNummer = Alben.AlbumNummer
AND Lieder.InterpretNummer = Interpreten.InterpretNummer
```

Zwei Hinweise:

- In dem Lied ‚Hope There's Someone' wird ein Apostroph-s verwendet, um dies einzufügen wird der Backslash (\) als sog. Escape-Sequenz vor das einzufügende Zeichen gestellt.
- Man würde die vorstehende SELECT-Anweisung in Form einer sogenannten VIEW speichern, damit man nicht so viel tippen muss. Wie das geht, wird bei der Anweisung CREATE VIEW unten beschrieben.

Datenbanken aus kaufmännischer Sicht

Ob man eine Datenbank einsetzen sollte, kann man anhang des folgenden Vergleichs zwischen zwei Firmen (einer Galerie und einem Baumarkt) beurteilen.

In einer Galerie werden nur wenige Bilder pro Woche verkauft, es fallen also nur wenige Daten an. Dazu braucht es keine Datenbank. Rechnungen und andere Informationen lassen sich von Hand schreiben.

In einem Baumarkt dagegen werden (nach Rückfrage) 150.000 Artikel zum Verkauf angeboten; pro Tag kommen geschätzte 2000 Kunden, jeder kauft im Schnitt 2 Artikel. Das ergibt pro Tag:

2000 x 2 = 4000 Datensätze

Rechnet man das hoch auf ein Jahr (ca. 250 Arbeitstage), dann ergibt sich:

4000 x 250 = 1.000.000 Datensätze

Bei einer Million Datensätze pro Jahr leuchtet es jedem ein, dass man nicht mehr mit Karteikarten arbeiten kann. Auch ‚normale' Kassen (ohne Anbindung an eine Datenbank) sind nicht die richtigen Werkzeuge. Besser ist es, wenn alle Buchungen in einer zentralen Datenbank gespeichert werden, so dass man auf Knopfdruck erkennen kann, wie viele Schrauben, Grillgeräte, Strandkörbe oder Topfpflanzen pro Jahr oder pro Monat verkauft wurden.

Eine Anekdote für die Fans der Tabellenkalkulation

Im Jahr 1990 fragte der Besucher eines einführenden EDV-Kurses, wie er das folgende Problem lösen solle: in einer großen Firma bearbeitete er die Reklamationen von Disketten, Magnetbänder, Cassetten usw. Er benutzte dazu Lotus 1-2-3 (eine Tabellenkalkulation, die um diese Zeit populärer war als Excel). Leider erlaubte jedes Arbeitsblatt (sheet) nur 32.000 Zeilen (oder Datensätze), das reichte aber nach ein paar Jahren nicht mehr aus.

Die Empfehlung war, eine Datenbank zu verwenden. Zwar gibt es auch bei Tabellen in Datenbanken eine maximale Anzahl von Zeilen, aber ein Leben reicht nicht dafür aus, diese zu füllen.

Wenige Wochen später traf man sich wieder, der Lotus-Fan berichtete freudig: es war nicht nötig zu wechseln, die mittlerweile erschienene neue Version seiner Tabellenkalkulation erlaube 64.000 Datensätze je Arbeitsblatt.

Die Datenbank wäre immer noch die bessere Alternative gewesen, denn die in ihr gespeicherten Daten stehen auch für andere Anwendungen oder andere Kollegen zur Verfügung. Aber gegen die Fans der Tabellenkalkulation ist schwer anzukommen.

Die Normalisierung von Daten

Wie bereits bei dem Beispiel mit Liedern und Musikern auf den vorigen Seiten gezeigt wurde, speichert man Daten auf mehrere Tabellen verteilt und verknüpft sie miteinander über gemeinsame Felder. Man teilt die Tabellen dabei in Master- und Detail-Tabellen ein, bei Musikern und Liedern ist das genauso wie bei kaufmännischen Daten.

In der Master-Tabelle sind beispielsweise Kundendaten (alle Daten, die einen Kunden beschreiben), in der Detail-Tabelle sind die Bestellungen der Kunden. Man nennt die Aufteilung auch Normalisierung. Das Ziel dabei ist es, möglichst wenig redundante (sich wiederholende) Daten zu haben. Das ist bei relationalen Datenbanksystemen immer so.

Drei Normalformen

Wie immer, ist es auch bei der Normalisierung ein bisschen komplizierter: Man unterscheidet man üblicherweise drei Normalformen:

- Die 1. Normalform gibt vor, dass man verschiedenartige Daten (wie zum Beispiel Vorname, Nachname, Postleitzahl, Ort) in verschiedenen Feldern speichert. In diese Felder kommen bei jedem Datensatz immer wieder gleichartige Informationen, so stehen in dem Feld für Nachnamen immer nur die Nachnamen.
- Um die 2. Normalform zu erreichen, teilt man die sich wiederholende Daten so auf, dass zusätzliche Tabellen Wiederholungen vermeiden helfen. Man sorgt also dafür, dass eine Adresse eines Kunden nur ein einziges Mal gespeichert wird.
- In der 3. Normalform werden schließlich Schlüssel (Primärschlüssel und Fremdschlüssel) eingefügt. Ein Primärschlüssel in einer Master-Tabelle macht die Feldinhalte eindeutig und wird mit dem sogenannten Fremdschlüssel in einer Detail-Tabellen verknüpft.

Da das Normalisieren von Daten ein wesentlicher Gedanke bei der Arbeit mit Datenbanken ist, werden hierzu mehrere Beispiele folgen.

Schlüsselfelder, Primärschlüssel und Fremdschlüssel

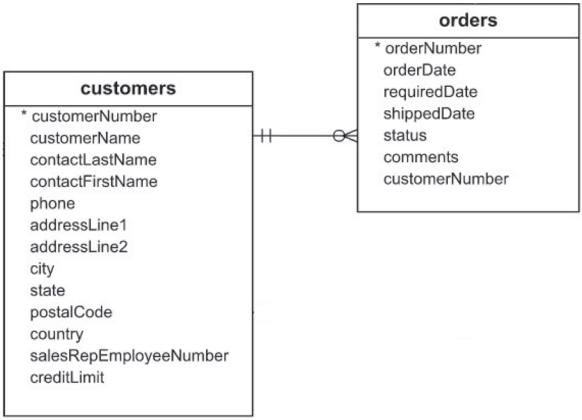
Schlüsselfelder (KundenNummer, LieferantenNummer, Artikelnummer) sind besondere Felder, über die man Tabellen miteinander verknüpft. Man unterscheidet Primär- und Fremdschlüssel:

- *Primärschlüssel* (Primary Key, abgekürzt: PK): Ein Primärschlüssel sorgt dafür, dass ein Feldinhalt eindeutig bleibt. Die Kundennummer 549873

beispielsweise gibt es nur einmal. Wird der Kunde gelöscht, dem diese Nummer gehört, wird diese Nummer nicht nochmals verwendet.

- *Fremdschlüssel* (Foreign Key, abgekürzt: FK): Ein Fremdschlüssel ist ein Feldinhalt, der erst durch die Verknüpfung mit einem Primärschlüssel Sinn ergibt, denn obwohl wir die Kundennummer in den Bestellungen sehen, wissen wir erst durch den Blick in die Kundentabelle, um wen es sich handelt.

Das folgende Schema zeigt die Beziehung zwischen den Tabellen ‚customers' und ‚orders' an.



Verknüpfung zwischen denTabellen ‚customers' und ‚orders'
(Kunden und Bestellungen)

Die customers-Tabelle (links) ist die Master-Tabelle, der Primary Key liegt in der Tabelle auf dem Feld ‚customerNumber' (dieses Feld ist mit mit einem * gekennzeichnet). Rechts ist die Detail-Tabelle. Die Verbindungslinie deutet an, dass *ein* Datensatz der linken Tabelle mit *mehreren* Datensätzen der rechten Tabelle in Beziehung steht (man bezeichnet das auch als 1:n-Verknüpfung).

Ein kurzes Beispiel soll zeigen, wie man Daten normalisiert. Das macht man durchaus mit Papier und Bleistift und bespricht die Entwürfe mit anderen. Erst danach legt man die Tabellen an. Spätere Änderungen im Entwurf (Design) der Tabellen sind zwar ohne Probleme möglich, aber das sollte nicht zur Gewohnheit werden.

Ein Beispiel zur Normalisierung von Daten

Um Artikeldaten eines Baumarktes zu speichern, soll eine Tabelle angelegt werden:

Artikel-Nummer	Bezeichnung	Einzelpreis	Lieferant
1	Hammer	12,98	Humm & Meier, Karlsruhe Tel.: 0721-123456
2	Eimer	4,78	Kiel und Schwund, Mannheim Tel.: 0621-4321
3	Kelle	9,45	Humm & Meier, Karlsruhe Tel.: 0721-123456

Anmerkungen zur Tabelle:

- Bei der Tabelle handelt es ich um eine Tabelle mit Artikel-Stammdaten (Masterdaten);
- die Spalten für Artkelnummer, Bezeichnung und Einzelpreis erklären sich selbst;
- in der vierten Spalte ist der Lieferantenname und alle sonstigen Daten des Lieferanten enthalten; wie zu sehen, ist der Lieferant ‚Humm & Meier' zweimal vorhanden. Hier muß eingegriffen werden, denn solche Wiederholungen sind immer auch Fehlerquellen: Tippfehler können im Extremfall dazu führen, daß ein Eintrag bei der Suche nicht mehr gefunden wird;
- in dieser vierten Spalte sind außerdem mehrere Arten von Information (Firmenname, Ort und Telefonnummer) gemeinsam abgelegt. Das vermeidet man und verwendet besser mehrere Felder.

Um die genannten Mängel in der vierten Spalte zu beseitigen, bietet sich die folgende Lösung an:

- Man macht eine zusätzliche Tabelle, in der nur die Daten der Lieferanten gespeichert werden und gibt jedem Lieferant zusätzlich eine eindeutige Lieferanten-Nummer;
- man fügt in die oben gezeigte Artikeltabelle eine Spalte für die Nummer des Lieferanten ein und löscht das vierte Feld;

Die dabei entstandenen zwei Tabellen für Artikel- und Lieferantendaten könnten folgendermaßen aufgebaut sein:

Eine Tabelle für Artikeldaten (als Ersatz für den Entwurf oben):

ArtikelNr	Bezeichnung	Preis	LieferantenNr
1	Hammer	12,98	1
2	Eimer	4,78	2
3	Kelle	9,45	1

... und eine Tabelle für Lieferantendaten

LieferantenNr	Firmenname	Ort	Telefon
1	Humm & Meier	Karlsruhe	0721-123456
2	Kiel & Schwund	Mannheim	0621-4321

Wir sehen sofort den Vorteil, dass Daten in der Lieferanten-Tabelle nur einmal erfasst werden müssen.

Demgegenüber gibt es auch einen Nachteil: in der Artikeltabelle steht nur noch eine Lieferanten-Nummer. Welcher Lieferant das ist, ergibt sich erst mit einem Blick in die zweite Tabelle. Hierzu werden die Tabellen miteinander verknüpft, oben wurde schon die 1:n-Verknüpfung erwähnt, das hier ist so eine: Ein Lieferant liefert mehrere Artikel.

In Firmen sind oft Dutzende von Tabellen miteinander in Beziehung gesetzt und die Master-Tabellen haben oft 30 und mehr Spalten.