



Entwurf und Verarbeitung relationaler Datenbanken

Eine durchgängige und praxisorientierte
Vorgehensweise

von

Prof. Dr. Nikolai Preiß
Berufsakademie Stuttgart

R. Oldenbourg Verlag München Wien

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über [<http://dnb.d-nb.de>](http://dnb.d-nb.de) abrufbar.

© 2007 Oldenbourg Wissenschaftsverlag GmbH
Rosenheimer Straße 145, D-81671 München
Telefon: (089) 45051-0
oldenbourg.de

Das Werk einschließlich aller Abbildungen ist urheberrechtlich geschützt. Jede Verwertung außerhalb der Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Verlages unzulässig und strafbar. Das gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Bearbeitung in elektronischen Systemen.

Lektorat: Wirtschafts- und Sozialwissenschaften, wiso@oldenbourg.de
Herstellung: Anna Grosser
Satz: DTP-Vorlagen des Autors
Coverentwurf: Kochan & Partner, München
Cover-Illustration: Hyde & Hyde, München
Gedruckt auf säure- und chlorfreiem Papier
Druck: Grafik + Druck, München
Bindung: Thomas Buchbinderei GmbH, Augsburg

ISBN 978-3-486-58369-4

Inhalt

Abbildungsverzeichnis	XI
1 Einführung	1
1.1 Datenbanken und Datenbankentwurf	1
1.2 Arbeiten mit relationalen Datenbanken	5
1.3 Vom Entwurf zur Verarbeitung relationaler Datenbanken	9
2 Entity-Relationship-Datenmodellierung	11
2.1 Entitätstyp	11
2.1.1 Strukturierungselement Entitätstyp	12
2.1.2 Attribut	12
2.1.3 Wertebereich	13
2.1.4 Primärschlüssel	14
2.1.5 Weak-Entitätstyp	15
2.1.6 Integritätsbedingungen	16
2.1.7 Übungsaufgabe 1	17
2.2 Beziehungstyp	18
2.2.1 Allgemeiner Beziehungstyp: Assoziation	18
2.2.2 Spezielle Beziehungstypen: Abhängigkeit, Aggregation, Generalisierung	20
2.2.3 Beziehungsentitätstyp	23
2.2.4 Multiplizität	25
2.2.5 Rolle	28
2.2.6 Attribut und Fremdschlüssel	29
2.2.7 Primärschlüssel	30
2.2.8 Beziehungstypen mit einem Grad > 2	33
2.2.9 Vermeidung höhergradiger Beziehungstypen	36
2.2.10 Übungsaufgabe 2	42
2.2.11 Übungsaufgabe 3	43
2.3 Klassische ER-Datenmodellierung	44
2.3.1 Entitätstyp	44
2.3.2 Beziehungstyp	45
2.3.3 Beziehungskomplexitäten	46
2.3.4 Übungsaufgabe 4	50

3	Relationale Datenmodellierung und Normalisierung	51
3.1	Grundlagen der relationalen Datenmodellierung	51
3.1.1	Relation	52
3.1.2	Attribut	52
3.1.3	Wertebereich	53
3.1.4	Primärschlüssel	54
3.1.5	Fremdschlüssel	54
3.1.6	Integritätsbedingung	55
3.2	Überführung des ER-Datenmodells in ein Relationenmodell	56
3.2.1	Überführung eines Entitätstyps	56
3.2.2	Überführung eines Beziehungstyps	59
3.2.3	Übungsaufgabe 5	66
3.2.4	Übungsaufgabe 6	66
3.3	Normalisierung im Relationenmodell	66
3.3.1	Grundidee der Normalisierung	66
3.3.2	Attribut-Abhängigkeiten	68
3.3.3	Erste Normalform	71
3.3.4	Zweite Normalform	73
3.3.5	Dritte Normalform	76
3.3.6	Boyce-Codd-Normalform	78
3.3.7	Vierte Normalform	81
3.3.8	Übungsaufgabe 7	83
3.3.9	Übungsaufgabe 8	84
4	Datenbanksprache SQL	87
4.1	Datendefinition	87
4.1.1	Schema und Tabellen	88
4.1.2	Datentypen und weitere Integritätsbedingungen	89
4.1.3	Änderung einer Tabelle	92
4.1.4	Views und Indexe	93
4.1.5	Übungsaufgabe 9	95
4.2	Datenabfrage	96
4.2.1	Datenbankoperatoren Projektion, Selektion und Verbund (Join)	96
4.2.2	Die SELECT-Anweisung	97
4.2.3	Ausgabe aller Datensätze einer Relation	99
4.2.4	Ausgabe spezieller Attribute einer Relation	99
4.2.5	Ausgabe ohne Duplikate	100
4.2.6	Ausgabe berechneter Attribute	101
4.2.7	Ausgabe spezieller Datensätze einer Relation	101
4.2.8	Ausgabe sortierter Datensätze	104
4.2.9	Ausgabe aggregierter Werte	105
4.2.10	Ausgabe aggregierter Werte mit Gruppierung	107
4.2.11	Ausgabe aggregierter Werte mit ausgewählter Gruppierung	108

4.2.12	Verwendung von Subabfragen	109
4.2.13	Verwendung von Subabfragen mit Existenz-Prüfung	113
4.2.14	Verknüpfung von Relationen	115
4.2.15	Verknüpfung von Relationen mit Bedingungen	116
4.2.16	Neue Formulierungsformen für Verknüpfungsbedingungen.....	120
4.2.17	Übungsaufgabe 10	125
4.2.18	Übungsaufgabe 11	126
4.2.19	Übungsaufgabe 12	126
4.3	Datenmanipulation	127
4.3.1	Einfügen von Datensätzen	127
4.3.2	Ändern von Datensätzen	129
4.3.3	Löschen von Datensätzen	130
4.3.4	Übungsaufgabe 13	131
4.3.5	Übungsaufgabe 14	132
5	Zusammenfassung und Ausblick	133
Anhang	139
Anhang 1:	Strukturierungselemente der Entity-Relationship-Datenmodellierung	139
Anhang 2:	Strukturierungselemente der relationalen Datenmodellierung.....	141
Anhang 3:	Abbildung ER-Datenmodell ins normalisierte Relationenmodell.....	142
Anhang 4:	Normalformen für relationale Datenbanken	144
Anhang 5:	Sprachelemente der Datenbanksprache SQL	145
Anhang 6:	Musterlösung Übungsaufgabe 1	151
Anhang 7:	Musterlösung Übungsaufgabe 2	152
Anhang 8:	Musterlösung Übungsaufgabe 3	154
Anhang 9:	Musterlösung Übungsaufgabe 4	157
Anhang 10:	Musterlösung Übungsaufgabe 5	160
Anhang 11:	Musterlösung Übungsaufgabe 6	162
Anhang 12:	Musterlösung Übungsaufgabe 7	165
Anhang 13:	Musterlösung Übungsaufgabe 8	169
Anhang 14:	Musterlösung Übungsaufgabe 9	172
Anhang 15:	Musterlösung Übungsaufgabe 10	175
Anhang 16:	Musterlösung Übungsaufgabe 11	177
Anhang 17:	Musterlösung Übungsaufgabe 12	178
Anhang 18:	Musterlösung Übungsaufgabe 13	180
Anhang 19:	Musterlösung Übungsaufgabe 14	181

Glossar	183
Internet-Links für den Download relationaler Datenbanksysteme	191
Literaturverzeichnis.....	192
Index	193

Vorwort

Relationale Datenbanken haben sich in den vergangenen 30 Jahren zu einem fundamentalen Bestandteil betrieblicher Informationssysteme entwickelt. Sie sind dementsprechend weit verbreitet und leisten in vielen Bereichen zuverlässige Dienste. Komfortable grafische Oberflächen erlauben EDV-Laien und Datenbankadministratoren eine einfache Verwaltung der gespeicherten Daten. Allerdings ist es alles andere als trivial, eine relationale Datenbank so zu strukturieren, dass sich bei deren Verarbeitung keine Datenunstimmigkeiten ergeben. Dies gilt insbesondere dann, wenn die Datenbank mehrere hundert Datenfelder und mehrere Millionen Datensätze umfasst.

Glücklicherweise kennt man heute geeignete Methoden, mit denen man sowohl kleine als auch große Datenstände optimal strukturieren und verwalten kann. Mit den entsprechenden Kenntnissen ist es möglich, ausgehend von der fachlichen Datenanalyse über den relationalen Datenbankentwurf nahtlos zur Implementierung einer optimalen Datenbankarchitektur zu gelangen. Eine solche Vorgehensweise und die zugehörigen Methoden sind das zentrale Thema des vorliegenden Lehrbuchs. Anhand vieler Beispiele wird in umfassender und praxisorientierter Weise gezeigt, wie relationale Datenbanken idealerweise entworfen werden sollten und wie die Daten in einer relationalen Datenbank verarbeitet werden können.

Entsprechend der Vorgehensweise gliedert sich das Buch in drei große Blöcke. Nach einer kurzen allgemeinen Einführung in die Welt der relationalen Datenbanken wird in einem ersten großen Block zunächst gezeigt, wie die Zusammenhänge der fachlichen Datenwelt in einem Entity-Relationship-Datenmodell aufbereitet werden können. Im zweiten Block wird dann das relationale Datenmodell vorgestellt und das Entity-Relationship-Datenmodell in ein relationales Datenmodell überführt. Dabei wird auch die Normalisierung von Relationen behandelt. Schließlich geht es im dritten und letzten Teil um die Datenbanksprache SQL, mit der Relationen definiert, manipuliert (Datensätze einfügen, ändern, löschen) und insbesondere abgefragt werden können.

Das vorliegende Buch ist gedacht als Basis für eine Grundlagenvorlesung über den Entwurf und die Verarbeitung relationaler Datenbanken.¹ Es eignet sich aber auch für ein Selbststudium oder als Nachschlagewerk für den Datenbankspezialisten, der in der Praxis relationale Datenbanken entwickelt und verarbeitet. Die vorgestellten Methoden und Sprachkonstrukte werden umfassend anhand vieler praktischer Beispiele veranschaulicht und können in zahl-

¹ Die zugehörigen Vorlesungsfolien sind beim Autor erhältlich (preiss@ba-stuttgart.de).

reichen Übungsaufgaben getestet werden, wobei sich eine größere Übungsaufgabe durch alle Kapitel zieht.

Im Gegensatz zu den meisten Lehrbüchern im Datenbankbereich behandelt das vorliegende Buch nicht das gesamte Spektrum aller aktuellen Datenbankthemen, sondern konzentriert sich auf die zentrale und grundlegende Fragestellung, wie relationale Datenbanken entworfen und verarbeitet werden. Insofern handelt es sich also nicht um ein klassisches Lehrbuch, sondern vielmehr um ein kompaktes Lehr- und Arbeitsbuch, das dem Leser anhand vieler Beispiele detailliert und anschaulich zentrale Datenbankthemen vermittelt.

Das Buch ist entstanden aus den Erkenntnissen und Erfahrungen, die sich über viele Jahre im Umgang mit Datenbanken - sowohl im akademischen Bereich als auch in der Praxis - ergeben haben. Für die Durchsicht der Formulierungen und die damit verbundenen Diskussionen, die zu zahlreichen wertvollen Klarstellungen und Ergänzungen geführt haben, möchte ich mich bei Prof. Dr. Andreas Oberweis (Universität Karlsruhe), Prof. Dr. Manfred Sander und Prof. Dr. Jürgen Schwille (beide Berufsakademie Stuttgart) bedanken.

Wenn Sie als Leser eine Anregung haben, wie man das vorliegende Buch weiter verbessern kann, würde ich mich über einen entsprechenden Hinweis freuen. Eine diesbezügliche E-Mail adressieren Sie bitte an preiss@ba-stuttgart.de.

Stuttgart, im Februar 2007

Abbildungsverzeichnis

Abb. 1.1 Datenbank mit zwei Tabellen.....	1
Abb. 1.2 Kundendatensatz in zwei unterschiedlichen Datenbanken.....	3
Abb. 1.3 Kundendatensatz in zentraler Datenbank.....	3
Abb. 1.4 Kunde-orientierte Strukturierung der Bestellungsdaten.....	4
Abb. 1.5 Artikel-orientierte Strukturierung der Bestellungsdaten.....	4
Abb. 1.6 Bestellung-orientierte Strukturierung der Bestellungsdaten.....	5
Abb. 1.7 Optimale Datenbankstruktur für die Bestellungsdaten.....	5
Abb. 1.8 3-Ebenen-Architektur eines Datenbanksystems in Anlehnung an ANSI/SPARC	7
Abb. 2.1 ER-Diagramm mit Entitätstypen.....	12
Abb. 2.2 ER-Diagramm mit Entitätstypen und deren Attributen.....	13
Abb. 2.3 Grafische Darstellung für Entitätstyp und für Informationsobjekte (Entitäten)	14
Abb. 2.4 Entitätstyp mit Primärschlüssel.....	15
Abb. 2.5 ER-Diagramm mit einem Weak-Entitätstyp.....	16
Abb. 2.6 Entitätstyp mit Integritätsbedingungen.....	17
Abb. 2.7 Beziehungstypen vom Grad 2 und 3.....	19
Abb. 2.8 ER-Diagramm mit Abhängigkeitsbeziehungstyp.....	20
Abb. 2.9 Aggregationen von Teilen zu einem Ganzen.....	21
Abb. 2.10 ER-Diagramm mit Generalisierung.....	22
Abb. 2.11 Sub-Entitätstypen mit speziellen Attributen und Beziehungstypen.....	23
Abb. 2.12 ER-Diagramm mit Beziehungsentitätstyp.....	24
Abb. 2.13 ER-Diagramm mit Multiplizitäten.....	26
Abb. 2.14 Abhängigkeitsbeziehungstyp mit Multiplizitäten.....	26
Abb. 2.15 Aggregationen mit Multiplizitäten.....	27
Abb. 2.16 Multiplizitäten bei Generalisierung/Spezialisierung.....	28
Abb. 2.17 Rekursiver Beziehungstyp mit Rollen.....	29
Abb. 2.18 Beziehungstyp mit Attributen.....	30

Abb. 2.19 Beziehungstypen mit unterschiedlichen Multiplizitäten	31
Abb. 2.20 Projekt-Angestellter-Abteilung-Datenbank	31
Abb. 2.21 ER-Diagramm mit allgemeinem binären Beziehungstyp.....	32
Abb. 2.22 ER-Diagramm mit Beziehungstyp für Einkäufe	33
Abb. 2.23 Beziehungstyp zur Verwaltung von Wareneinkäufen.....	34
Abb. 2.24 allgemeiner Beziehungstyp mit dem Grad 3	35
Abb. 2.25 Zerlegung eines Beziehungstyps mit Grad 3 in binäre Beziehungstypen	37
Abb. 2.26 Beziehungstyp kauft...bei mit Grad 3.....	38
Abb. 2.27 Zerlegter Beziehungstyp kauft...bei	39
Abb. 2.28 Umwandlung eines Beziehungstyps mit Grad 3 in einen Entitätstyp	40
Abb. 2.29 Umwandlung eines Beziehungstyps mit Grad 3 und „0..1“-Multiplizitäten	42
Abb. 2.30 Attribute und Weak-Entitätstyp bei klassischer ER-Datenmodellierung	45
Abb. 2.31 Beziehungstypen bei klassischer ER-Datenmodellierung.....	46
Abb. 2.32 Darstellungsform der 1:n-Notation	47
Abb. 2.33 Darstellungsformen der (min,max)-Notation.....	48
Abb. 2.34 Darstellung Beziehungstyp mit Grad 3	49
Abb. 2.35 Darstellung von Beziehungstypen mit Pfeil-Notation und Krähenfuß-Notation ..	50
Abb. 3.1 Relation mit Datensätzen	52
Abb. 3.2 Relation mit Attributen und Wertebereichen	53
Abb. 3.3 Relation mit Primär- und Fremdschlüssel.....	54
Abb. 3.4 Relationendiagramm mit Fremdschlüsselbeziehung.....	55
Abb. 3.5 Relation mit Integritätsbedingungen	56
Abb. 3.6 Entitätstyp Angestellter.....	58
Abb. 3.7 Relationenmodell für den Entitätstyp Angestellter	59
Abb. 3.8 Beziehungstypen mit unterschiedlichen Multiplizitäten	60
Abb. 3.9 Relationendiagramm mit verschiedenen Fremdschlüssel-Beziehungen	61
Abb. 3.10 Überführung des Abhängigkeitsbeziehungstyps in ein Relationenmodell.....	62
Abb. 3.11 Überführung von Aggregationsbeziehungstypen in ein Relationenmodell.....	63
Abb. 3.12 Überführung der Generalisierung/Spezialisierung in ein Relationenmodell.....	64

Abb. 3.13 Überführung eines höhergradigen Beziehungstyps in ein Relationenmodell.....	65
Abb. 3.14 Überführung eines rekursiven Beziehungstyps in ein Relationenmodell	65
Abb. 3.15 Relation zur Verwaltung von Prüfungsergebnissen	67
Abb. 3.16 Relationen zur Verwaltung von Mitarbeitern und Abteilungen	69
Abb. 3.17 Relation Prüfung mit atomaren, einwertigen Attributen	71
Abb. 3.18 Datenbank mit 1NF-Relationen.....	73
Abb. 3.19 Datenbank mit 2NF-Relationen.....	75
Abb. 3.20 Datenbank mit 3NF-Relationen.....	78
Abb. 3.21 Datenbank mit BCNF-Relationen	80
Abb. 3.22 Relation mit mehrwertiger Abhängigkeit.....	82
Abb. 3.23 Datenbank mit 4NF-Relationen.....	83
Abb. 4.1 Relationale Datenbank mit Basistabellen (oben) und View (unten).....	94
Abb. 4.2 Anwendung von Verbund, Selektion und Projektion in einer relationalen Datenbank.....	97
Abb. 4.3 Datenbank Verkauf	98
Abb. 4.4 Datenbank Handel	125

1 Einführung

1.1 Datenbanken und Datenbankentwurf

In vielen Bereichen unseres Alltags wird heute mit **Datenbanken** gearbeitet. Beim Einkaufen im Supermarkt, beim Geldabheben am Geldautomaten, bei der Behandlung im Krankenhaus, bei der Reisebuchung im Reisebüro oder auch bei Auktionen im Internet, überall sind Computerprogramme im Einsatz, deren Basis eine Datenbank bildet.

Eine Datenbank besteht im Allgemeinen aus einer Sammlung von **strukturierten Datensätzen**, deren Anzahl nahezu beliebig groß werden kann. Dabei enthält jeder Datensatz spezielle Informationen über ein bestimmtes Objekt, bspw. über einen bestimmten *Artikel* eines Herstellers oder über einen bestimmten *Kunden* eines Unternehmens. Zur Verwaltung dieser Informationen werden standardmäßig **Tabellen** verwendet, wobei für jeden Objekttyp eine eigene Tabelle angelegt wird (s. Beispiel 1.1). Es ist zu beachten, dass eine solche Tabelle immer zwei Arten von Informationen enthält:

- Struktur (Datenfelder, auch Attribute genannt, als Spalten) und
- Inhalt (Datensätze als Zeilen).

Beispiel 1.1

In einer Datenbank sollen die Daten über die Artikel und die Kunden eines Unternehmens verwaltet werden. Dazu werden die beiden Tabellen *Artikel* und *Kunde* eingesetzt, die entsprechend den jeweils relevanten Informationen strukturiert sind (s. Abb. 1.1).

Artikel			Kunde		
Artikel-Nr	Bezeichnung	Preis	Kunden-Nr	Name	Anschrift
4711	Schreibtisch	99,99	JB-007	James Bond	Palastweg 7, 00707 Buckingham
0815	Wasserbett	555,55	ME-100	Max Einstein	Ideengasse 1, 12121 Entenhausen
1234	Liegestuhl	22,22	JF- 987	Jutta Feldbusch	Hauptstr. 6, 60606 Witzigheim

Abb. 1.1 Datenbank mit zwei Tabellen

Die Verwaltung einer Datenbank erfolgt durch eine spezielle Software, das **Datenbankmanagementsystem** (DBMS). Dieses DBMS bietet

- Funktionen zum Anlegen, Ändern und Löschen der Tabellenstrukturen,
- Funktionen zum Einfügen, Ändern und Löschen von Datensätzen in Tabellen,
- Funktionen zum Abfragen von Datensätzen in Tabellen.

Das DBMS bildet zusammen mit der Datenbank (dem eigentlichen Datenspeicher) das Datenbanksystem. Da der Fachbegriff für die Tabelle **Relation** lautet, spricht man von einem **relationalen Datenbanksystem** (rDBS).

Relationale Datenbanksysteme stellen die Standard-Technologie im Datenbankbereich dar und sind in fast allen Unternehmen zu finden. Am häufigsten anzutreffen sind Oracle vom gleichnamigen Hersteller, DB2 von IBM und der Microsoft SQL Server, zunehmend aber auch MySQL aus dem Open Source-Bereich². Der hohe Verbreitungsgrad erklärt sich mit einer ganzen Reihe erheblicher **Vorteile**, die ein rDBS bietet:

- zentrale Datenverwaltung mit unterschiedlichen Benutzersichten,
- koordinierter Mehrbenutzerbetrieb,
- hohe Verfügbarkeit und Absturzsicherheit (durch Wiederherstellungsmechanismen für den Fehlerfall),
- schnelles Zugriffsverhalten, auch bei großen Datenbanken (Millionen von Datensätzen),
- Zugriffsschutz,
- einfache, umfassende und standardisierte Datenbanksprache SQL³ (angelehnt an die englische Sprache),
- Unabhängigkeit von technischen Aspekten wie Speicherstrukturen, Zugriffsstrukturen, usw.,
- graphische Oberfläche für interaktive Benutzer und Aufrufmöglichkeiten für Programmierer (Programmierschnittstellen).

Bevor aber ein rDBS die genannten Vorteile ausspielen kann, muss im Vorfeld zunächst einmal geklärt werden, welche **Struktur** eine Datenbank haben sollte. Dabei geht es darum, welche Tabellen für den späteren Einsatzbereich benötigt werden, welche Strukturen diese Tabellen haben müssen und wie diese Tabellen zusammenhängen. Diese Fragestellungen betreffen die Architektur einer Datenbank und sind nicht so einfach zu beantworten. Sie erfordern nämlich ein spezielles methodisches Vorgehen, das die geeigneten Tabellen im Rahmen eines so genannten **Datenbankentwurfs** ermittelt.

Oberstes Gebot bei einem Datenbankentwurf sollte sein, dass die Verwaltung der Daten an zentraler Stelle organisiert wird, unabhängig von speziellen Anwendungen erfolgt und auf **redundanzfreie** bzw. redundanzarme (mit kontrollierter Redundanz⁴ realisierte) Datenbestände zielt. Dadurch erhält man nicht nur einen Überblick über die gesamte Datenwelt, sondern es wird auch eines der großen Probleme bei der Verarbeitung von Datenbanken vermieden, nämlich die durch Mehrfachspeicherung entstehenden **Datenunstimmigkeiten** (s. Beispiel 1.2).

² Die Web-Seiten dieser Datenbanksysteme sind bei den Internet-Links am Ende des Buches zu finden.

³ SQL steht für **Structured Query Language**.

⁴ Unter Redundanz versteht man die Mehrfachspeicherung von Daten. Die Mehrfachspeicherung kann bei verteilten Datenbanken sehr sinnvoll sein. Dieser Aspekt wird hier aber nicht behandelt.

Beispiel 1.2

In einem Unternehmen werden die Kundendaten unabhängig voneinander in zwei unterschiedlichen Datenbanken verwaltet. Dabei hat die Abteilung Marketing einen Kundendatensatz gemäß Abb. 1.2 oben. Die Abteilung Vertrieb hat denselben Kunden im Datenbestand, allerdings mit einem Kundendatensatz gemäß Abb. 1.2 unten.

<i>Kunde</i>		
<i>Name</i>	<i>Straße</i>	<i>Ort</i>
Reiner Mayer	Waldweg 8	45454 Piepenstadt

<i>Kunde</i>		
<i>Name</i>	<i>Wohnort</i>	<i>Beruf</i>
Rainer Maier	Bahnhofstr. 5, 12121 Glückshausen	Wirtschaftsinformatiker

Abb. 1.2 Kundendatensatz in zwei unterschiedlichen Datenbanken

Bei getrennter Datenhaltung bleibt normalerweise unentdeckt, dass es sich um denselben Kunden handelt. Es bleibt auch ungeklärt, wie die richtige Schreibweise für den Namen lautet und wie die unterschiedlichen Adressen zu interpretieren sind (Umzug? Zweitwohnsitz?). Nicht zuletzt wäre auch eine einheitlich Struktur für die Kundendatensätze hilfreich (bspw. für die Suche).

Alle genannten Probleme können dadurch vermieden werden, dass die Daten zentral und ohne Redundanzen gemäß Abb. 1.3 verwaltet werden (mit eindeutigem Namen und zwei Wohnsitzen).

<i>Kunde</i>		
<i>Name</i>	<i>Anschrift</i>	<i>Beruf</i>
Rainer Mayer	Bahnhofstr. 5, 12121 Glückshausen	Wirtschaftsinformatiker
	Waldweg 8, 45454 Piepenstadt	

Abb. 1.3 Kundendatensatz in zentraler Datenbank

Das zentrale Problem beim Datenbankentwurf ist die Klärung der Frage, wie man zu einer **optimalen Datenbankstruktur** kommt. Das Beispiel 1.3 veranschaulicht dies an einem einfachen Fall.

Beispiel 1.3

Ein Online-Shop möchte die Bestellungen seiner Kunden in einer Datenbank verwalten und überlegt, in welcher Form die Datensätze abzulegen sind. Dabei geht es um die Datfelder *Bestell-Nr*, *Bestell-Datum*, *Kunden-Nr*, *Kunden-Name*, *Anschrift*, *Artikel-Nr*, *Artikel-Name*, *Preis* und *Bestell-Menge*.

Die DV-Spezialisten des Online-Shops haben zunächst die Idee, die Bestellinformationen als Ergänzung in einem *Kunde-Datensatz* abzulegen (s. Abb. 1.4). Bei diesem Lösungsvorschlag fallen aber sofort einige Nachteile auf:

- Die Artikeldaten können nur gespeichert werden, wenn eine Bestellung vorliegt.
- Bei jeder Bestellung werden sämtliche Artikeldaten gespeichert, was zu vielen Mehrfachspeicherungen führt, verbunden mit einem entsprechend hohen Aufwand bei Änderungen (bspw. bei Artikel-Name).

Die *Kunde*-orientierte Datenstruktur ist also keine gute Lösung.

<i>Kunde</i>								
<i>K-Nr</i>	<i>K-Name</i>	<i>Anschrift</i>	<i>Bestellung</i>					
			<i>B-Nr</i>	<i>B-Datum</i>	<i>Art-Nr</i>	<i>Art-Name</i>	<i>Preis</i>	<i>Menge</i>
007	Bond	Buckingham	1001	1.1.2006	4711	Stuhl	29,90	4
			1002	2.1.2006	4712	Tisch	49,90	1
008	Bean	Oxford	999	1.1.2006	4711	Stuhl	29,90	1
009	Miller	Stanford						

Abb. 1.4 *Kunde-orientierte Strukturierung der Bestellsdaten*

Ein weiterer denkbarer Lösungsvorschlag wäre ein *Artikel*-orientierter Datensatz (Bestellinformationen als Ergänzung in einem *Artikel-Datensatz* – s. Abb. 1.5). Dieser Ansatz bereitet aber dieselben Probleme wie der *Kunde*-orientierte Datensatz. Die *Artikel*-orientierte Datenstruktur ist also auch keine gute Lösung.

<i>Artikel</i>								
<i>Art-Nr</i>	<i>Art-Name</i>	<i>Preis</i>	<i>Bestellung</i>					
			<i>B-Nr</i>	<i>B-Datum</i>	<i>K-Nr</i>	<i>K-Name</i>	<i>Anschrift</i>	<i>Menge</i>
4711	Stuhl	29,90	999	1.1.2006	008	Bean	Oxford	1
			1001	1.1.2006	007	Bond	Buckingham	4
4712	Tisch	49,90	1002	2.1.2006	007	Bond	Buckingham	1
4713	Schrank	99,90						

Abb. 1.5 *Artikel-orientierte Strukturierung der Bestellsdaten*

Als dritte Alternative bleibt noch der Ansatz, die Bestellinformationen in einem **Bestellung**-orientierten Datensatz abzulegen (s. Abb. 1.6). Leider weist aber auch diese Struktur die bekannten Nachteile auf, sogar in verstärktem Maße:

- Die Daten über Kunden und Artikel können nur gespeichert werden, wenn diese an Bestellungen beteiligt sind.
- Bei jeder Bestellung werden sämtliche Kundendaten und sämtliche Artikeldaten gespeichert, mit den bekannten Redundanzproblemen.

Bestellung								
B-Nr	B-Datum	K-Nr	K-Name	Anschrift	Art-Nr	Art-Name	Preis	Menge
999	1.1.2006	008	Bean	Oxford	4711	Stuhl	29,90	1
1001	1.1.2006	007	Bond	Buckingham	4711	Stuhl	29,90	4
1002	2.1.2006	007	Bond	Buckingham	4712	Tisch	49,90	1

Abb. 1.6 Bestellung-orientierte Strukturierung der Bestellsungsdaten

Mit einer einzigen Datensatzart bzw. Tabelle kommt man also im vorliegenden Fall nicht zu einer guten Datenbankstruktur. Eine gute Datenbankstruktur findet man nur dann, wenn die Datenfelder auf mehrere Datensatzarten bzw. Tabellen verteilt werden, wie dies die Abb. 1.7 zeigt. Dabei ist zu beachten, dass die Verbindungen zwischen den Tabellen über die gleichnamigen Spalten hergestellt werden (bspw. zwischen *Bestellung* und *Artikel* über *Art-Nr*).

<i>Kunde</i>			<i>Artikel</i>		
<i>K-Nr</i>	<i>K-Name</i>	<i>Anschrift</i>	<i>Art-Nr</i>	<i>Art-Name</i>	<i>Preis</i>
007	Bond	Buckingham	4711	Stuhl	29,90
008	Bean	Oxford	4712	Tisch	49,90
009	Miller	Stanford	4713	Schrank	99,90

<i>Bestellung</i>				
<i>B-Nr</i>	<i>B-Datum</i>	<i>K-Nr</i>	<i>Art-Nr</i>	<i>Menge</i>
999	1.1.2006	008	4711	1
1001	1.1.2006	007	4711	4
1002	2.1.2006	007	4712	1

Abb. 1.7 Optimale Datenbankstruktur für die Bestellsungsdaten

Daten über Kunden und Artikel können nun gespeichert werden, ohne dass diese an einer Bestellung beteiligt sind. Ferner ist bspw. der Artikel-Name oder die Anschrift an lediglich einer Stelle gespeichert, wodurch eine Datenänderung einfach durchzuführen ist.

Mit schlecht strukturierten Datenbanken lassen sich keine guten Informationssysteme realisieren. Gut strukturierte Datenbanken sind eine unabdingbare Voraussetzung für eine gut funktionierende Informationsverarbeitung. Daher ist ein zentrales Anliegen des vorliegenden Buches, ein Verfahren aufzuzeigen, wie man zu **guten Datenbankstrukturen** kommt (s. Kapitel 2 und 3).

1.2 Arbeiten mit relationalen Datenbanken

Der Entwurf und die Verarbeitung relationaler Datenbanken erfolgt heute nach einem bestimmten Prinzip, das sich in den vergangenen 40 Jahren Schritt für Schritt entwickelt hat.

Die ersten Datenbanksysteme, die Mitte der Sechziger-Jahre auf den Markt kamen, waren satzorientiert und noch nicht relational organisiert. Es gab zunächst hierarchische Datenbanksysteme (bspw. IMS von IBM) und kurz darauf netzwerkorientierte Datenbanksysteme (bspw. IDS von General Electric). Erst 1970 erschien der grundlegende Artikel von E. F. Codd (s. [Cod70]), damals Mitarbeiter bei IBM, über die Basiskonzepte mengenorientierter relationaler Datenbanken. Es dauerte dann aber noch einige Zeit, bis Ende der Siebziger-Jahre mit System R von der IBM das erste relationale Datenbanksystem verfügbar war. Dieses Datenbanksystem gilt als Vorläufer heutiger **relationaler Datenbanksysteme** wie DB2 und Oracle und lieferte insbesondere auch die Vorlage für die Entwicklung der heutigen Datenbanksprache **SQL**.

Mitte der Siebziger-Jahre erschienen zwei weitere Artikel, die das Arbeiten mit relationalen Datenbanken wesentlich beeinflussten. Zum einen erschien 1976 der grundlegende Artikel von P. Chen (s. [Che76]) über die Basiskonzepte der **Entity-Relationship-Datenmodellierung** (ERDM) und zum anderen erschien 1975 von der ANSI/SPARC-Standardisierungsgruppe ein Vorschlag über die **3-Ebenen-Architektur** eines Datenbanksystems (s. [AS75]).

Mit der ERDM wurde eine Methode vorgestellt, mit der die Zusammenhänge der Datenwelt unabhängig von DV-technischen Aspekten beschrieben werden konnten. Diese Methode hat sich in den letzten drei Jahrzehnten in der Praxis sehr bewährt und stellt heute in Verbindung mit der relationalen Datenmodellierung die Standard-Methode für den **Datenbankentwurf** dar. Dabei ist zu beobachten, dass in jüngster Zeit die ERDM in einigen Punkten der Modellierungssprache **UML**⁵ angeglichen wurde, die sich seit der Jahrtausendwende immer mehr zum Standard für die Systemmodellierung entwickelt.

Ebenso wie die ERDM hat sich auch der Architekturvorschlag der ANSI/SPARC-Gruppe zu einem Standard entwickelt. Alle großen relationalen Datenbanksysteme funktionieren heute im Prinzip nach dieser **3-Ebenen-Architektur** (s. Abb. 1.8), die im Folgenden als grober Überblick vorgestellt wird:

- Das zentrale Element der Architektur bildet die **konzeptionelle Ebene**, auf welcher mittels ER-Datenmodellierung und relationaler Datenmodellierung die Struktur der relationalen Datenbank ermittelt wird. Das DV-unabhängige Arbeitsergebnis (ER-Datenmodell) nennt man konzeptionelles Schema, das DV-abhängige Arbeitsergebnis (relationales Datenmodell) heißt logisches Schema.

Schemainformationen stellen Strukturbeschreibungen dar, die auch Metadaten heißen und in einer eigenen systeminternen Datenbank, dem sogenannten Systemkatalog (Data Dictionary), verwaltet werden (im Hintergrund, unbemerkt vom Benutzer).

Da der Datenbankentwurf auf der konzeptionellen Ebene stattfindet, steht diese Ebene im Mittelpunkt der Betrachtungen und wird daher im vorliegenden Buch ausführlich behandelt (s. Kapitel 2 und 3).

⁵ UML steht für Unified Modeling Language; für eine Einführung s. www.uml.org

- Oberhalb der konzeptionellen Ebene befindet sich die **Benutzerebene** (externe Ebene), auf der jeder Anwender bzw. jedes Anwendungsprogramm seine speziellen Sichten auf die Daten erhält. Dabei werden die Benutzersichten aus dem relationalen Datenmodell der mittleren Ebene abgeleitet. Die Sichten für einen bestimmten Anwender bzw. für ein bestimmtes Anwendungsprogramm werden in einem externen Schema zusammengefasst.

Auf die Benutzersichten wird zwar in Kapitel 4 noch näher eingegangen, die externe Ebene stellt aber kein zentrales Thema für das vorliegende Buch dar.

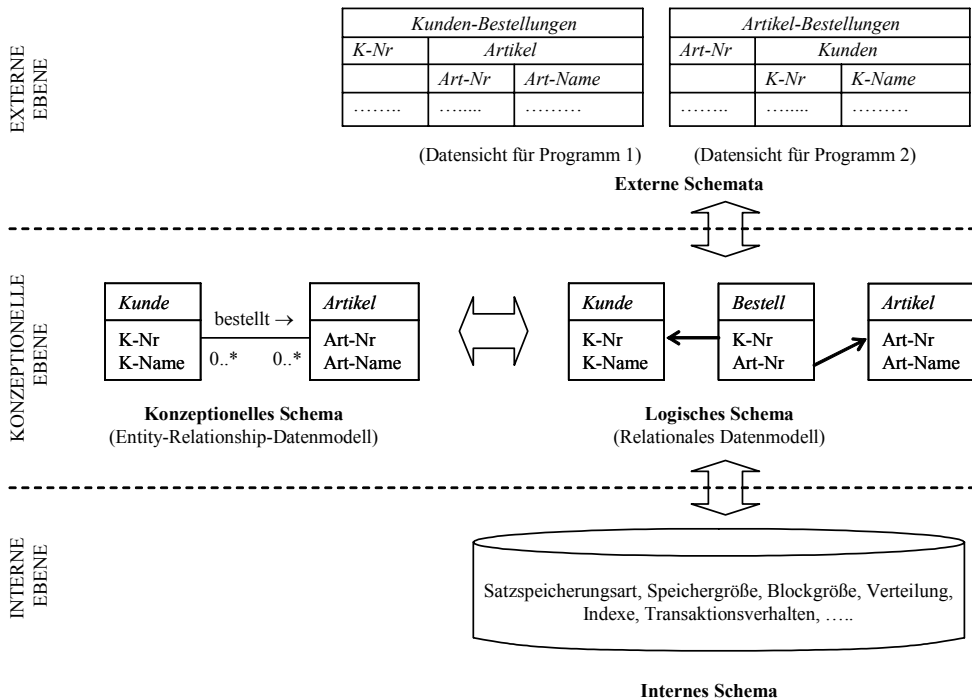


Abb. 1.8 3-Ebenen-Architektur eines Datenbanksystems in Anlehnung an ANSI/SPARC

- Unterhalb der konzeptionellen Ebene befindet sich die **interne Ebene**, auf der geregelt wird, wie die Datensätze physisch auf der Festplatte zu organisieren sind. Die Vereinbarungen werden in einem internen Schema festgehalten, das einen direkten Bezug zum relationalen Datenmodell besitzt. Dieses meist DBS-spezifische interne Schema wird hier aber nur der Vollständigkeit halber erwähnt und im Folgenden nicht näher betrachtet (mit Ausnahme der Indexe⁶ – s. Kapitel 4.1.4).

⁶ Ein Index stellt eine Art Stichwortverzeichnis für die Datensätze einer Relation dar und ermöglicht einen gezielten und damit schnellen Zugriff auf die Datensätze.

Der große Vorteil der 3-Ebenen-Architektur besteht darin, dass in vielen Fällen das interne Schema und das logische Schema geändert werden können (bspw. *Index löschen* oder *neues Datenfeld hinzufügen*), ohne dass dies Auswirkungen auf die Benutzersichten hat. Man spricht dann von **physischer** bzw. **logischer Datenunabhängigkeit**, was bedeutet, dass die Anwenderprogramme bei Änderungen am internen oder logischen Schema nicht angepasst werden müssen. Dies erhöht deutlich die Flexibilität beim Umgang mit Datenbanken und bietet große Einsparungspotentiale für die Wartungsphase und für das Reengineering⁷ (im Vergleich zu monolithischen (nicht modular aufgebauten) Software-Systemen).

Der Entwurf und die Verarbeitung relationaler Datenbanken gestaltet sich gemäß der 3-Ebenen-Architektur nun wie folgt:

- **Fachliche Analyse und fachlicher Entwurf:**

Zunächst werden aus fachlicher Sicht (ohne DV-Aspekte) die Anforderungen an die Datenbank formuliert, d.h. die Datenfelder, die in der Datenbank verwaltet werden sollen, werden gesammelt und entsprechend ihrer Zusammengehörigkeit in einem Entity-Relationship-Datenmodell geordnet (s. Kapitel 2).

- **DV-technischer Entwurf:**

Das ER-Datenmodell wird dann in ein relationales Datenmodell überführt (s. Kapitel 3), welches aus Performance-Gesichtspunkten heraus anschließend noch modifiziert werden kann (bspw. durch die Zusammenlegung von Tabellen).

- Codierung:

- **Datendefinition:**

Die fertigen Tabellen des relationalen Datenmodells werden in einem dritten Schritt mit der Datenbanksprache SQL definiert (s. Kapitel 4.1). Danach können mit SQL auf der externen Ebene die gewünschten Benutzersichten über die zuvor vereinbarten Basistabellen definiert werden (s. Kapitel 4.1.4). Hinsichtlich des internen Schemas können mit SQL lediglich einige wenige Angaben gemacht werden. Meist sind dies Index-Vereinbarungen (s. Kapitel 4.1.4).

- **Datenmanipulation:**

Im letzten Schritt erfolgt nun das eigentliche Arbeiten mit der Datenbank, also das Einfügen, Ändern, Löschen und Abfragen der Daten (s. Kapitel 4.2 und 4.3). Dies kann entweder über die Benutzersichten erfolgen oder in freier Form über die interaktive Benutzerschnittstelle des Datenbanksystems.

Hinweis: In der Praxis werden für die Datendefinition und die Datenmanipulation oft so genannte GUI-Werkzeuge eingesetzt, die den jeweiligen SQL-Code vor dem Benutzer verbergen und statt dessen mittels einer grafischen Oberfläche interaktiv mit dem System Definitionen und Manipulationen ermöglichen.

SQL hat sich in den letzten 25 bis 30 Jahren zu einer sehr **umfassenden Datenbanksprache** entwickelt. Für die eingangs erwähnten datenbankbasierten Informationssysteme, die große

⁷ Unter Reengineering versteht man im Software-Bereich die Neu- oder Umstrukturierung bestehender Systeme.