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> abrufbar.

Rosenheimer Straße 145, D-81671 München
Telefon: (089) 45051-0
oldenbourg.de

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

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

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		
Artikel-Nr	*Bezeichnung*	*Preis*
4711	Schreibtisch	99,99
0815	Wasserbett	555,55
1234	Liegestuhl	22,22

Kunde		
Kunden-Nr	*Name*	*Anschrift*
JB-007	James Bond	Palastweg 7, 00707 Buckingham
ME-100	Max Einstein	Ideengasse 1, 12121 Entenhausen
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-Bereich[2]. 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 SQL[3] (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 Redundanz[4] 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 **S**tructured **Q**uery **L**anguage.

[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.

Kunde		
Name	*Straße*	*Ort*
Reiner Mayer	Waldweg 8	45454 Piepenstadt

Kunde		
Name	*Wohnort*	*Beruf*
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).

Kunde		
Name	*Anschrift*	*Beruf*
Rainer Mayer	Bahnhofstr. 5, 12121 Glückshausen Waldweg 8, 45454 Piepenstadt	Wirtschaftsinformatiker

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 Datenfelder *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.

Kunde								
K-Nr	*K-Name*	*Anschrift*	*Bestellung*					
			B-Nr	*B-Datum*	*Art-Nr*	*Art-Name*	*Preis*	*Menge*
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 Bestellungsdaten

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.

Artikel								
Art-Nr	*Art-Name*	*Preis*	*Bestellung*					
			B-Nr	*B-Datum*	*K-Nr*	*K-Name*	*Anschrift*	*Menge*
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 Bestellungsdaten

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 Bestellungsdaten

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*).

Kunde		
K-Nr	*K-Name*	*Anschrift*
007	Bond	Buckingham
008	Bean	Oxford
009	Miller	Stanford

Artikel		
Art-Nr	*Art-Name*	*Preis*
4711	Stuhl	29,90
4712	Tisch	49,90
4713	Schrank	99,90

Bestellung				
B-Nr	*B-Datum*	*K-Nr*	*Art-Nr*	*Menge*
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 Bestellungsdaten

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 **U**nified **M**odeling **L**anguage; 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.

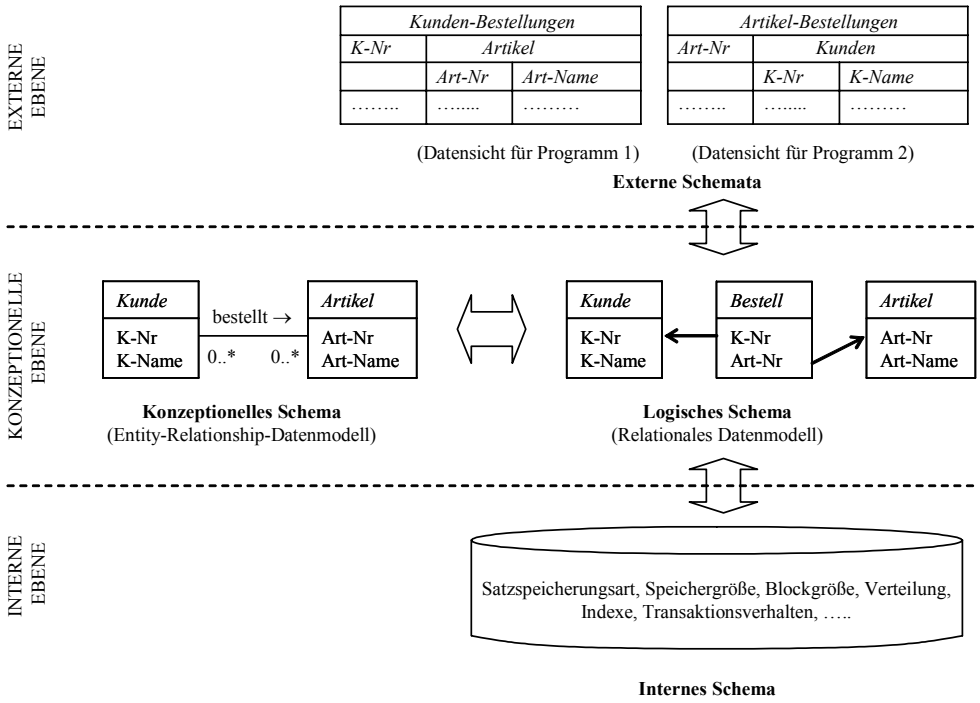

Abb. 1.8 3-Ebenen-Architektur eines Datenbanksystems in Anlehnung an ANSI/SPARC

- 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 Indexe[6] – 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 Reengineering[7] (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.