Objekt-orientierte Datenbanken: Modelle und Sprachen

von
Prof. Dr. Georg Lausen,
Universität Freiburg
Prof. Dr. Gottfried Vossen,
Universität Münster

R. Oldenbourg Verlag München Wien 1996

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Lausen, Georg:
Objekt-orientierte Datenbanken : Modelle und Sprachen / von Georg Lausen ; Gottfried Vossen. - München ; Wien : Oldenbourg, 1996
ISBN 3-486-22370-4
NE: Vossen, Gottfried:

© 1996 R. Oldenbourg Verlag GmbH, München

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.

Gesamtherstellung: R. Oldenbourg Graphische Betriebe GmbH, München

ISBN 3-486-22370-4

Inhaltsverzeichnis

Vorwort ix

I Grundlagen 1

1 Objekt-Orientierung in Datenbanken 3
1.1 Einführung . . . 3
1.2 Historische Entwicklung . . . 10
1.3 Relationale Datenbanken und SQL . . . 12
1.3.1 Strukturelle Aspekte . . . 13
1.3.2 Änderungen und Anfragen in SQL . . . 14
1.4 Eine Beispiel-Anwendung . . . 16
1.4.1 Darstellung im Relationenmodell . . . 17
1.4.2 Nicht-relationale Darstellung . . . 20
1.4.3 Erkenntnisse . . . 22
1.5 Objekt-orientierte Datenbanken . . . 23
1.5.1 Anforderungen . . . 23
1.5.2 Das Paradigma der Objekt-Orientierung . . . 24
1.5.3 OO-Eigenschaften . . . 27
1.5.4 DB-Eigenschaften . . . 40
1.5.5 Diskussion . . . 41
1.6 Bibliographische Hinweise . . . 43

2 Aspekte objekt-orientierter Datenbanksprachen 45
2.1 Allgemeine Anforderungen . . . 45
2.2 Wünschenswerte Eigenschaften . . . 47
2.2.1 Elementarer Zugriff auf Objekte . . . 47
2.2.2 Zugriff auf komplexe Objekte . . . 48
2.2.3 Expliziter Verbund . . . 50
2.2.4 Gleichbehandlung von Attributen und Methoden . . . 50
2.2.5 Zugriff auf abstrakte Typen und Klassen . . . 51
2.2.6 Elementarer Zugriff auf Objekt-Mengen . . . 52
2.2.7 Mengen-Zugriff im Kontext einer Klassenhierarchie . . . 55
2.2.8 Erzeugung und Veränderung von Objekten . . . 56

2.2.9 Erzeugung neuer Klassen, Instanzen und Operationen . . 58
2.2.10 Abschließende Bemerkungen 60
2.3 Navigation über Pfad-Ausdrücke 61
2.4 Vererbung 66
2.4.1 Allgemeines 67
2.4.2 Typen, Ersetzbarkeit und spätes Binden 71
2.4.3 Sicherheit 74
2.4.4 Diskussion 80
2.5 Bibliographische Hinweise 82

3 Ein formaler Rahmen für Struktur und Verhalten 83
3.1 Modellierung von Struktur 83
3.2 Modellierung von Verhalten 95
3.3 Formale Behandlung von Pfad-Ausdrücken 100
3.3.1 Skalare Pfad-Ausdrücke 100
3.3.2 Mengenwertige Pfad-Ausdrücke 105
3.4 Bibliographische Hinweise 108

II Sprachen 109

4 Fallstudien 111
4.1 Architektur und Persistenzmodell objekt-orientierter Datenbanken 112
4.1.1 Architektur 112
4.1.2 Persistenzmodell 116
4.2 Illustra 117
4.3 O_2 118
4.3.1 Die Deklarationssprache 119
4.3.2 Die Anfragesprache 128
4.4 GemStone 133
4.4.1 System-Übersicht 133
4.4.2 Einführung in OPAL 136
4.4.3 Struktur-Definition in OPAL 138
4.4.4 Methoden-Erzeugung in OPAL 143
4.4.5 Datenmanipulation 146
4.5 ObjectStore 152
4.6 Bibliographische Hinweise 157

5 Standardisierungsaktivitäten 159
5.1 SQL3 160
5.1.1 Erweiterungen von SQL2 160
5.1.2 Wert- und Objekttypen 161
5.1.3 Untertabellen 164
5.2 OMG-Standards 166

5.2.1 Verteilte Objektverwaltung 167
5.2.2 Object Management Architecture 170
5.2.3 CORBA 173
5.2.4 Zum Objektmodell der OMG 175
5.3 Die ODMG-Vorschläge 178
5.3.1 Grundlagen des ODMG-93-Objektmodells 178
5.3.2 Die Objekt-Definitionssprache ODL 182
5.3.3 Die Objekt-Anfragesprache OQL 185
5.4 Bibliographische Hinweise 188

III Theoretische Konzepte 189

6 Algebraische Operationen auf Datenbanken 191
6.1 Algebraische Operationen auf Relationen 191
6.1.1 Die Relationenalgebra und ihre Eigenschaften 191
6.1.2 Algebraische Optimierung 193
6.1.3 Die Relationenalgebra als Sprachmaßstab 195
6.2 Algebraische Operationen auf geschachtelten Relationen 197
6.3 Algebraische Operationen auf Objektbanken 200
6.3.1 Einführende Überlegungen 200
6.3.2 Objekterhaltende Operationen 201
6.3.3 Objekterzeugende Operationen 204
6.4 Vollständigkeit objekt-orientierter Sprachen 205
6.5 Bibliographische Hinweise 207

7 Objekt-Orientierung und Regeln 209
7.1 Regeln 210
7.2 Objekt-Orientierung 215
7.2.1 Komplexe Objekte 216
7.2.2 Regeln 218
7.2.3 Objekt-Identitäten 220
7.2.4 Verarbeitung von Mengen 222
7.2.5 Klassen 223
7.3 Bibliographische Hinweise 225

Literaturverzeichnis 227

Index 233

Vorwort

Objekt-orientierte Datenbanken werden seit etwa 15 Jahren in der Forschung untersucht und sind rund halb so lange auch schon kommerziell verfügbar. Das Paradigma der Objekt-Orientierung steht heute in vielen Bereichen der Informatik im Mittelpunkt. Bei Datenbanken bewirkt die Einführung der Objekt-Orientierung eine Abkehr von zahlreichen Beschränkungen, die dem Anwender durch die vorher verfügbare (insbesondere die relationale) Technologie auferlegt wurden; exemplarisch seien nur zwei genannt:

- Durch eine Aufgabe des Zwangs, daß Daten und ihre Beziehungen untereinander unausweichlich in (normalisierter) Tabellenform zu repräsentieren sind, ergibt sich für viele Anwendungen eine erheblich angemessenere Modellierung und Verarbeitung.

- Aus der Sicht des Software Engineering und der Anwendungsentwicklung findet ein Bruch statt, wenn ein gesamter Entwurf objekt-orientiert durchgeführt wird, das unterliegende Datenbanksystem jedoch relational ist. Ist auch das Datenbanksystem objekt-orientiert, läßt sich die Datenverwaltung nahtlos in die Methodik der Softwaretechnik einpassen.

Darüber hinaus kann man feststellen, daß die im Zeitalter der globalen Vernetzung immer bedeutender werdende verteilte Informationsverarbeitung, welche auf Aspekte wie Interoperabilität, Kooperation, Kollaboration oder Integration abzielt, auf Objekt-Orientierung basiert, da man erkannt hat, daß ein verteiltes Objekt-Management ein hierfür vielversprechender Ansatz ist. In diesem Kontext ist es sinnvoll, auch die Datenverwaltung objekt-orientiert zu organisieren.

Im vorliegenden Buch wählen wir zur Motivation objekt-orientierter Datenbanken den Aspekt der angemessenen Modellierung und Verarbeitung. Wir behandeln vornehmlich die benutzernahen Aspekte der Modelle und Sprachen objekt-orientierter Datenbanken; wir schreiben über systemnahe Aspekte nur dann, wenn es zum Verständnis erforderlich ist.

Ein charakteristisches Merkmal dieses Buches ist seine Ausrichtung an SQL und die Berücksichtigung der aktuellen Standardisierungsbemühungen im Bereich der objekt-orientierten Datenbanken. Die relationale Sprache SQL ist seit vielen Jahren der Standard für Datenbanken, und es ist heute abzusehen, daß sich weitere Entwicklungen hieran orientieren und stark anlehnen werden; da dies ein Buch über objekt-orientierte *Datenbanken* und nicht etwa

ein Buch über persistente Programmiersprachen ist, erscheint eine Betonung von SQL gerechtfertigt. Ferner ist heute unbestritten, daß Standards eine hohe Bedeutung haben; speziell im Datenbankbereich ist jedoch die für die Zukunft endgültige Richtung, bei der Definition von SQL3 oder bei der Arbeit der ODMG, noch nicht eindeutig absehbar. Daher behandeln wir das Thema Standards etwas breiter, so daß im Hinblick auf ODMG auch der stabilere OMG-Hintergrund aufgegeriffen wird.

Unter diesen Prämissen geht es uns im vorliegenden Text vor allem um eine kompakte Darstellung von *Modellen* und von *Sprachen* für objekt-orientierte Datenbanken. Wir fahren über eine längere Strecke zweigleisig; wir betrachten die Kernaspekte sowohl aus einer informalen, an Beispielen orientierten, von den möglichen Anwendungen getriebenen Sicht, als auch aus einem formalen Blickwinkel, aus dem die zentralen Aspekte soweit präzisiert werden, daß ihr Zweck und ihre Wechselwirkungen klar erkennbar werden. Dabei nehmen wir, im Unterschied zu anderen Büchern zum gleichen Thema, ganz bewußt keinen Rundumschlag vor; wir bieten stattdessen eine sicher subjektive Stoffauswahl, in der allerdings unserer Meinung nach nichts Wesentliches über Modelle und Sprachen fehlt.

Das Buch besteht aus drei Teilen. In Kapitel 1 von Teil I geben wir eine Einführung in den hier relevanten Datenbankkontext mit seiner bisherigen Evolution bei Datenmodellen und Datenbanksystemen und beschreiben die wesentlichen Eigenschaften der Objekt-Orientierung im allgemeinen und der objekt-orientierten Datenbanken im besonderen. In den Kapiteln 2 und 3 stecken wir den Rahmen der weiteren Betrachtungen genauer ab und behandeln in Kapitel 2 die zentralen Eigenschaften objekt-orientierter Datenbank*sprachen*, jedoch noch nicht in bezug zu einer konkreten Sprache bzw. einem Produkt. In Kapitel 3 untersuchen wir die Modelleigenschaften objekt-orientierter Datenbanken genauer und stellen einen formalen Rahmen her, in welchem sich die diversen, zuvor informal diskutierten Zusammenhänge präzisieren und Probleme lösen lassen.

Teil II behandelt sodann in Kapitel 4 verschiedene *Fallstudien*, um an diesen die unterschiedlichen Realisierungsmöglichkeiten objekt-orientierter Datenbanken studieren zu können; die keinen Anspruch auf Vollständigkeit erhebende Systemauswahl umfaßt Vertreter der derzeit wichtigsten Entwicklungslinien, nämlich objekt-orientierte Systeme in Datenbank- bzw. in Programmiersprachentradition sowie objekt-relationale Systeme. Kapitel 5 ist aktuellen Standardisierungsaktivitäten gewidmet, denn es ist zu hoffen und zu erwarten, daß SQL3 sowie die OMG und die ODMG für eine weitere Vereinheitlichung im Bereich objekt-orientierter Datenbank-Sprachen sorgen werden.

In Teil III wollen wir den auch an formalen Untersuchungen interessierten Leser in einige *theoretische Konzepte* einführen, welche im Zusammenhang mit objekt-orientierten Datenbank-Sprachen relevant sind. In Kapitel 6 behandeln wir algebraische Ansätze zum Sprachentwurf; in Kapitel 7 kontrastieren wir

diese mit regelbasierten Konzepten.

Das Buch ist ursprünglich aus einem Tutorium hervorgegangen, welches wir 1993 an verschiedenen Orten für die Deutsche Informatik-Akademie gehalten haben; darüber hinaus haben wir Material verarbeitet, das wir auch in der universitären Lehre verwenden. Dementsprechend richtet sich das Buch an Leserinnen und Leser, die mit Grundlagen von Datenbanken vertraut sind und sich lediglich mit objekt-orientierten Datenbanken näher beschäftigen wollen; solche Vorkenntnisse besitzt man z.B. als Informatik-Studentin oder -Student nach dem Vordiplom.

Wie stets bei solchen Gelegenheiten sind wir zahlreichen Leuten zu Dank verpflichtet. Herrn Klaas Kruithoff danken wir für das sorgfältige Anfertigen der Abbildungen, unsern Mitarbeitern Margret Groß-Hardt, Paul-Thomas Kandzia und Heinz Uphoff für die Durchsicht von Teilen dieses Buches. Wir bedanken uns bei unserer Lektorin Margarete Metzger für die Geduld, die sie diesem Projekt (das schon längst abgeschlossen sein sollte) entgegen gebracht hat. Schließlich danken wir unseren Familien für das Verständnis, mit welchem unsere Arbeit an diesem Buch selbst an sonnigen Sonntagen im Breisgau begleitet wurde.

Freiburg und Münster, im November 1995

Georg Lausen, Gottfried Vossen

Teil I

Grundlagen

Kapitel 1

Objekt-Orientierung in Datenbanken

In diesem einführenden Kapitel wird die Verwendung von Objekt-Orientierung in Datenbanken und Datenbanksystemen motiviert, um so den Rahmen für die nachfolgenden Betrachtungen abzustecken. Dazu wollen wir zunächst Datenbanken und deren typische Einsatzgebiete skizzieren, um so insbesondere Anforderungen an Datenbanksysteme abzuleiten. Die historische Entwicklung von Datenmodellen und Datenbanksystemen läßt erkennen, in welcher Weise unterschiedliche Systemgenerationen diese Anforderungen erfüllen. Von besonderer Bedeutung sind heute *relationale* Datenbanken, die jedoch insbesondere in neueren Datenbank-Anwendungen an ihre Grenzen stoßen. Dies wiederum motiviert speziell die Einbeziehung objekt-orientierter Konzepte in Datenbanken, sowie die Entwicklung eigenständiger *objekt-orientierter Datenbanksysteme*. Wir geben eine Charakterisierung derartiger Systeme und diskutieren die Probleme und Aufgaben, die zum Zwecke einer angemessenen Integration von Objekt-Orientierung und Datenbanken zu lösen sind. In nachfolgenden Kapiteln wird sich dann unter anderem zeigen, welcher Stand heute erreicht bzw. für die nahe Zukunft absehbar ist.

1.1 Einführung

Datenbank-Management spielt heutzutage eine zentrale Rolle in Anwendungen, in welchen große Datensammlungen in adäquater und effizienter Weise organisiert, erzeugt, unterhalten und verwaltet werden müssen und welche sich demzufolge auf ein rechnerbasiertes *Informationssystem* abstützen. Datenbanksysteme werden seit rund 30 Jahren erforscht, entwickelt und eingesetzt, und zwar in Anwendungen unterschiedlichster Natur und Zielsetzung.

Ein *Datenbanksystem* (DBS) basiert, grob gesehen, auf der Idee, Daten und zugehörige Programme zu *trennen*, und unterscheidet sich damit wesentlich von einem Dateisystem. Es sei allerdings bemerkt, daß ein zentrales An-

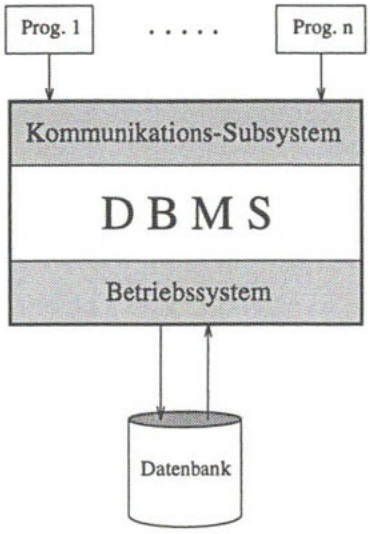

Abbildung 1.1: Generische Sicht eines DBMS.

liegen objekt-orientierter Datenbaken wiederum die *Integration* von Daten und Programmen ist, die jetzt aber in einem angemessenen methodischen Rahmen (abstrakte Datentypen) vorgenommen wird.

Ein Datenbanksystem besteht aus Software, dem *Datenbankmanagement-system* (DBMS), sowie einer oder mehreren *Datenbanken.* Das DBMS ist ein Programmsystem, welches im Hauptspeicher eines Rechners unter der Kontrolle des jeweiligen Betriebssystems abläuft. Eine Datenbank ist eine Sammlung von Daten, die Informationen über eine spezielle Anwendung der realen Welt repräsentieren, die aufgrund ihres Umfangs typischerweise im Sekundärspeicher gehalten wird. Das DBMS agiert dann als Schnittstelle zwischen den Benutzern und einer Datenbank; es stellt sicher, daß Benutzer auf die Daten in angemessener und effizienter Weise zugreifen können, und daß die Daten selbst gegen Hard- und Softwarefehler resistent sind und unabhängig von den Programmen, die auf sie zugreifen, über lange Zeiträume hinweg persistent gespeichert werden können.

Technisch gesehen ist ein DBMS im allgemeinen eingebettet in den Software-Kontext eines Rechnersystems in der in Abbildung 1.1 gezeigten Weise. Nach außen ist es angebunden an das Kommunikationssubsystem des betreffenden Rechners, nach innen arbeitet es unter der Kontrolle des Betriebssystems.

Die Unterscheidung zwischen einer Datenbank und dem sie verwaltenden DBMS ermöglicht mindestens zwei Arten der Betrachtung eines DBS:

(1) aus der Benutzersicht,

(2) aus der Entwicklersicht.

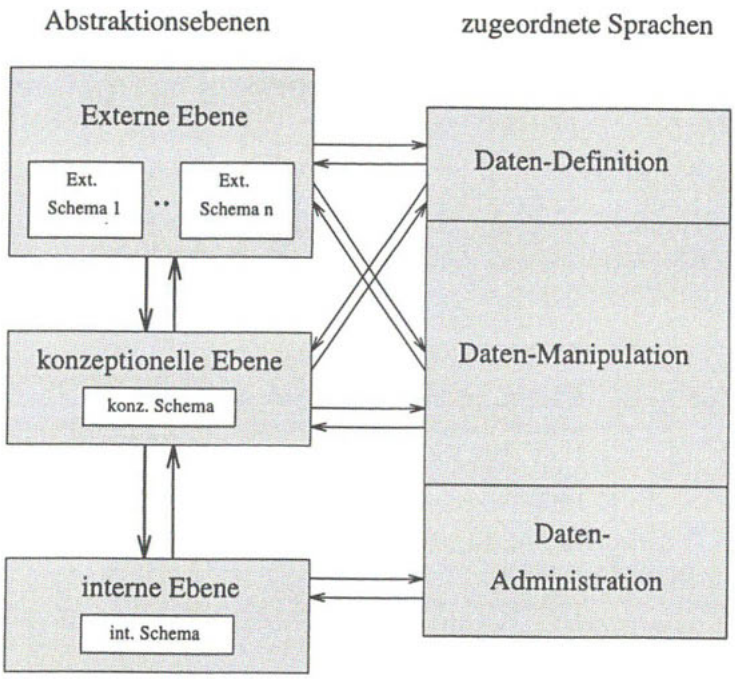

Abbildung 1.2: 3-Ebenen-Datenbank-Architektur nach ANSI/SPARC.

Aus der Sicht eines *Benutzers* wird eine Datenbank im allgemeinen auf unterschiedlichen Abstraktionsebenen gesehen, auf denen unterschiedliche Arten von Daten identifiziert werden können. So kennt z.B. das allgemein akzeptierte ANSI/SPARC-Modell die drei in Abbildung 1.2 gezeigten Abstraktionsebenen, welche auf die Erzielung logischer und physischer *Datenunabhängigkeit* ausgerichtet sind. Die (unterste) *interne Ebene* behandelt die physische Definition und Organisation von Daten (in Form adäquater Datenstrukturen mit zugehörigen Zugriffspfaden). Die nächst höhere *konzeptionelle Ebene* beschreibt die zeitinvariante globale Struktur einer Datenbank im konzeptionellen *Schema* und in der Sprache eines spezifischen *Datenmodells*, welches von den Details der physischen Ebene abstrahiert. Die höchste, *externe Ebene* gibt einzelnen Benutzern oder Benutzergruppen die Möglichkeit, Teile des konzeptionellen Schemas für ihre speziellen Anwendungen in Form eines externen Schemas zu definieren.

Zentral für diese Sichtweise einer Datenbank ist die konzeptionelle Ebene und ihr Datenmodell. Grundsätzlich hat ein Datenmodell eine *Spezifikationskomponente*, welche die Beschreibung struktureller Aspekte einer Anwendung und deren Semantik ermöglicht, sowie eine *operationale Komponente*, welche eine Manipulation dieser Strukturen erlaubt. Dies stellt sich dem Benutzer in Form spezifischer *Sprachen* dar, mit welchen er Datenbank-Strukturen defi-

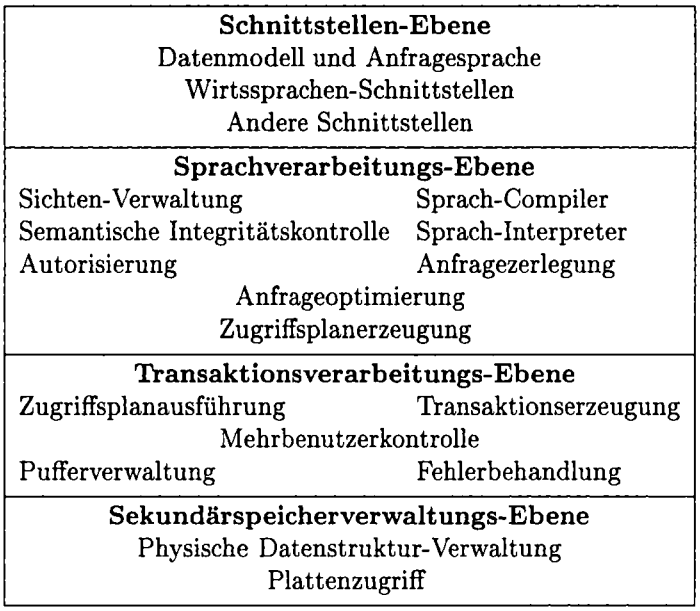

Abbildung 1.3: Funktionale DBMS-Ebenen.

nieren, manipulieren und verwalten kann. Diese Sprachen bzw. Sprach-Anteile sind in Abbildung 1.2 mit ihrer Zuordnung zu den jeweiligen Ebenen angedeutet.

Aus der Sicht eines *Systementwicklers* muß ein DBMS eine umfangreiche Funktionalität abdecken, welche sich teilweise direkt aus der Benutzersicht ergibt. Grundsätzlich kann die Funktionalität eines DBMS als in vier Ebenen organisiert betrachtet werden (vgl. Abbildung 1.3). Die *Schnittstellen-Ebene* stellt Schnittstellen für die verschiedenen Klassen von Benutzern bereit, darunter der Datenbank-Administrator, Ad-hoc-Benutzer und Anwendungsprogrammierer. Die *Sprachverarbeitungs-Ebene* hat die verschiedenen Arten von Aufträgen an eine Datenbank (wie Anfragen oder Änderungen) zu behandeln. So wird z.B. eine Anfrage im allgemeinen zerlegt in eine Reihe elementarer Datenbank-Operationen, welche einer *Optimierung* unterzogen werden können mit dem Ziel einer Vermeidung von Ausführungen mit schlechter Laufzeit. Eine ausführbare Anfrage bzw. ein ausführbares Programm wird an die *Transaktionsverarbeitungs-Ebene* übergeben, welche zeitlich überlappte Zugriffe auf eine von vielen Anwendern gemeinsam benutzte Datenbank kontrolliert (Mehrbenutzerkontrolle, *Concurrency Control*) und das System gleichzeitig resistent gegen gewisse Arten von Fehlern macht (Fehlerbehandlung, *Recovery*). Die *Sekundärspeicherverwaltungs-Ebene* schließlich ist zuständig für physische

Datenstrukturen (wie z.B. Dateien, Seiten oder Indexe) und führt Plattenzugriffe aus.

Aus dieser groben Beschreibung der zwei zentralen Sichtweisen einer Datenbank lassen sich bereits die vier größeren Bereiche identifizieren, auf welche sich die Datenbank-Forschung und -Entwicklung in den vergangenen 25 Jahren konzentriert hat:

(1) *Datenmodelle:*
Datenmodelle stellen geeignete Konzepte zur Abstraktion von Realwelten und der damit verbundenen Modellierung von Struktur und Semantik von Daten bereit. Datenmodelle haben bis heute eine umfangreiche Evolution durchlaufen (vgl. Abbildung 1.5).

(2) *Datenbanksprachen:*
Gegenstand ist hier die Bereitstellung von Sprachen, mit welchen insbesondere in geeigneter Weise auf Datenbanken operiert bzw. zugegriffen werden kann. Wesentlich war z.B. die Erkenntnis, daß Datenbanksprachen im Vergleich zu höheren Programmiersprachen nur über eine eingeschränkte Ausdruckskraft verfügen können, da aus praktischer Sicht der Aspekt der effizienten Auswertbarkeit von hoher Bedeutung ist. Sprachen wie der relationale Standard SQL (vgl. Abschnitt 1.3) besitzen unter anderem keine Schleifen-Kontrollstruktur. Bei Bedarf ist es jedoch möglich, die Anfragesprache des DBS in eine höhere Programmiersprache *einzubetten*, welche ja stets über Kontrollstrukturen verfügt und so „Turing-Berechenbarkeit" gewährleistet. Hierbei stößt man jedoch in der Regel auf den sogenannten *Impedance Mismatch*, da Datenbanken und Programmiersprachen unterschiedliche Datentypen besitzen. In einer Programmiersprache existiert beispielsweise ein Datentyp „Menge" im allgemeinen nicht; Mengenoperationen auf der Datenbank sind daher auf der Ebene der Programmiersprache unter Verwendung eines *Cursors* in Tupel-Operationen auf den Elementen der Menge zu zerlegen. Die „Abstraktionsebene" der Datenbanksprache geht an dieser Stelle verloren.

(3) *Transaktionen und Mehrbenutzerkontrolle:*
Da es Benutzern einer Datenbank möglich sein soll, zeitlich verzahnt auf einem gemeinsamen Datenbestand zu operieren, müssen Synchronisations- und Wiederanlauf-Strategien bereitgestellt werden, die dieses ermöglichen. In Datenbanksystemen wird hierzu das Transaktionskonzept herangezogen, welches sich in unterschiedlicher Weise präzisieren und zum Entwurf entsprechender Protokolle einsetzen läßt.

(4) *Datenstrukturen:*
Da ein Datenbanksystem über eine Sekundärspeicherverwaltung verfügen muß, ist die Entwicklung entsprechender Datenstrukturen und Zugriffstechniken für diese wesentlich. Auch hier verfügt man heute über ein um-

fangreiches Wissen, welches in konkreten Systemen zur effizienten Speicherung und Wiederauffindung von Daten verwendet wird.

Die Entwicklung und Untersuchung dieser Teilaspekte wurde stets durch Anforderungen aus den verschiedenen Anwendungsbereichen von Datenbanksystemen gesteuert bzw. beeinflußt. Von besonderer Bedeutung waren dabei kommerzielle Anwendungen, insbesondere in Wirtschaft und Verwaltung, von denen wir beispielhaft die folgenden erwähnen:

- Banken und Versicherungen,
- öffentliche Verwaltungen,
- Bibliotheken,
- Buchungs- und Reservierungssysteme (*START* und *AMADEUS*),
- Materialwirtschaft und -verwaltung, Lagerhaltung,
- Personalverwaltung,
- Auftragserfassung und Rechnungswesen,
- Einwohnermelde- und Auskunftswesen,
- Produktionsplanung und -steuerung.

Diese Bereiche haben zahlreiche gemeinsame Charakteristika, die erlauben, daß Datenbank-Technologie tatsächlich zu einem wirksamen Hilfsmittel wird, z.B.:

- einfache, fest-formatierbare Datensätze, welche insbesondere durch satz- bzw. record-orientierte Datenmodelle auf logischer Ebene angemessen beschreibbar sind,
- einfache Datentypen für die Wertebereiche der Felder bzw. Attribute eines Datensatzes (numerische sowie alphanumerische Typen reichen häufig aus),
- viele parametrisierbare, vorformulierbare Anfragen, welche in einer solchen Anwendung häufig wiederkehrende Aufträge an die Datenbank beschreiben,
- Änderungen an Ort und Stelle („In-Place-Updates"), durch welche alte Werte durch neue ersetzt werden, ohne den jeweiligen alten Wert noch zu erhalten,
- kurze Transaktionen, welche die an die Datenbank gerichteten Aufträge (Anfragen oder Änderungs-Operationen) in interner Form repräsentieren und welche im allgemeinen mit hoher Rate (z.B. 1.000 Transaktionen pro Sekunde) ausführbar sein müssen.

Aus denselben Gründen, die vor über 20 Jahren in den gerade genannten Anwendungsbereichen den Übergang von Datei- zu Datenbanksystemen motivierten, also z.B.

- physische und logische Datenunabhängigkeit,
- redundanzfreie Speicherung,
- zentrale Integritätskontrollen,
- auf hohem Abstraktionsniveau angesiedelte Sprachen,

verlangen heute auch zunehmend wissenschaftlich-technische Bereiche nach einer DBS-Unterstützung. Exemplarisch seien hier genannt:

- die CAx-Bereiche, also z.B. CAD, CAM, CAE, CASE oder auch CIM,
- Büro-Informationssysteme,
- geographische Informationssysteme,
- Versuchsdatenerfassung und -auswertung,
- Multimedia-Systeme (für Bilder, Texte, Sprache, Daten).

Diese haben eigene Charakteristika, die von traditionellen DBS bisher nicht oder nur schlecht unterstützt werden, wobei man anwendungs*spezifische* Charakteristika von anwendungs*unabhängigen* unterscheiden muß; zu den letzteren zählen insbesondere:

(1) die Notwendigkeit der Modellierung hochstrukturierter Information, d.h. die Darstellbarkeit komplexer Objekte und Datentypen,

(2) der Wunsch der Modellierung von Verhalten, d.h. von objekt- bzw. typspezifischen Operationen auf den Strukturen, die durch das jeweilige Datenmodell beschreibbar sind.

Wenn man berücksichtigt, daß häufig die einem Datenmodell mitgegebenen Möglichkeiten zur Beschreibung von Struktur stark eingeschränkt sind gegenüber dem, was höhere Programmiersprachen in dieser Richtung erlauben, so verwundert es nicht, daß im Bereich der Datenmodelle seit längerem versucht wird, von Programmiersprachen her bekannte Paradigmen auch für Datenbanken nutzbar zu machen. Besonderes Interesse hat in diesem Zusammenhang *Objekt-Orientierung* erlangt, um die es in diesem Text gehen soll.

Buch	*InvNr*	*ErstAutor*	*WeitAut*	*Titel*	...
	123	Date	n	Intro DBS	
	234	Jones	y	Algorithms	
	345	King	n	Operating Syst.	

Leser	*LeserNr*	*Name*	...
	225	Peter	
	347	Laura	

Ausleihe	*InvNr*	*LeserNr*	*AusleihEnde*
	123	225	07-22-94
	234	347	08-02-94

Abbildung 1.4: Beispiel einer relationalen Datenbank.

1.2 Historische Entwicklung

Die historische Entwicklung von Datenbanksystemen kann in drei Generationen eingeteilt werden. Die erste Generation (zeitlich den späten 60er sowie den 70er Jahren zuzuordnen) war durch die Einführung einer ersten Trennung von *logischer* und *physischer* Information gekennzeichnet. Datenmodelle wurden erstmals zur Beschreibung physischer Strukturen aus logischer Sicht eingesetzt. Speziell wurden das *hierarchische Modell* sowie das *Netzwerk-Modell* auf der Grundlage von Graphenkonzepten entwickelt. Da diese Modelle aus heutiger Sicht nur noch wenig interessant sind, gehen wir auf sie hier nicht weiter ein.

Die zweite Systemgeneration ist geprägt durch relationale Systeme, die zu Beginn der 80er Jahre kommerziell verfügbar wurden. *Relationale* Systeme basieren auf einem neuen und einfacheren Ansatz zur Datenorganisation, der *Relation* bzw. *Tabelle*, welcher eine erheblich klarere Unterscheidung zwischen einem logischen und einem physischen Datenmodell ermöglicht. Relationale Systeme bieten insbesondere einen hohen Grad an physischer Datenunabhängigkeit und verfügen über mächtige, wenn auch berechnungsmäßig eingeschränkte Sprachen. Physische Datenunabhängigkeit bedeutet dabei, daß die physische Speicherung von Daten für einen Benutzer transparent (im Sinne von unsichtbar) ist und daher im Prinzip verändert werden kann, ohne daß sich die logische Sicht auf die Daten ebenfalls verändern müßte. Relationale Sprachen sind *mengen-orientiert* (anstelle von satz-orientiert) und können daher nicht-prozedural oder *deklarativ* sein. In einer relationalen Datenbank sieht ein Benutzer die Daten als Tabellen. Als Beispiel zeigt Abbildung 1.4 eine kleine Datenbank für eine Bibliotheks-Anwendung: Die *Buch*-Relation beschreibt die in einer Bibliothek vorhandenen Bücher, die *Leser*-Relation die der Bibliothek

bekannten Benutzer, die *Ausleihe*-Relation stellt Bücher mit Lesern in Beziehung. Mengen-orientierte Verarbeitung bedeutet nun, daß die Tabellen einer relationalen Datenbank durch spezielle Operatoren als Ganzes angesprochen werden können; man braucht also nicht durch eine Relation Tupel für Tupel zu iterieren. Da Relationen ein wichtiges und bekanntes mathematisches Konzept sind und auch z.B. als Prädikate der mathematischen Logik aufgefaßt werden können, hat diese Datenbanksystem-Generation eine intensive Durchdringung aus theoretischer Sicht erfahren, welche bereits in den frühen 70er Jahren begann, kurz nachdem das Modell von Codd vorgeschlagen worden war.

Während Datenbanksysteme zunächst hauptsächlich in Anwendungen mit betriebswirtschaftlichem Hintergrund eingesetzt wurden, wurde in den frühen 80er Jahren auch für wissenschaftliche, technische und andere Bereiche der durch Datenbanken erzielbare Nutzen erkannt. Relationale Systeme stoßen in Anwendungen wie z.B. CAD, CASE oder CIM jedoch aus einer Reihe von Gründen an ihre Grenzen, so daß hier eine Weiterentwicklung notwendig ist. Diese ist heute in vollem Gange; insbesondere ist mit *objekt-orientierten*, *objekt-relationalen* sowie mit *erweitert-relationalen Datenbanksystemen* inzwischen eine dritte Generation verfügbar, die über die Möglichkeiten rein relationaler Systeme weit hinausgeht und unter anderem auf eine angemessene Integration von Programmiersprachen und Datenbanken abzielt.

Neben dieser Systemevolution kann eine zweite, eng verwandte Evolutionslinie im Datenbankbereich verfolgt werden, die sich auf *Datenmodelle* konzentriert. Die Evolution von Datenmodellen ist in Abbildung 1.5 überblicksartig gezeigt. Motivation für die Entwicklung des Entity-Relationship-Modells (ER-Modells), semantischer Modelle bzw. von Komplex-Objekt-Modellen war unter anderem der Wunsch nach graphischen Hilfsmitteln für den Datenbankentwurf, eine Abkehr von satz-orientierter Denkweise, die Herstellung einer klareren Trennung von logischen und physischen Konzepten, eine Reduzierung semantischer Überladung von Beziehungs-Typen durch explizite Einführung entsprechender Konstrukte (z.B. IS-A-Typ-Beziehungen, d.h. *Spezialisierungen*, im Unterschied zu Komponenten-Beziehungen, d.h. *Aggregationen*) sowie eine manchmal nur indirekte Verfügbarkeit zentraler Abstraktionsmechanismen in satz-orientierten Modellen.

Während das geschachtelte Relationenmodell sich als direkte Verallgemeinerung des flachen Relationenmodells ergibt, gehen das ER-Modell und nachfolgend entwickelte semantische Modelle anders vor bei der Beschreibung von gegebenen Anwendungen. Wesentliches Ziel war zunächst die Herstellung von Möglichkeiten einer adäquateren Beschreibung von Objekten mit komplexer Struktur sowie die Erfassung von semantischer Information. Dies leisten auch die sogenannten Komplex-Objekt-Modelle, die sich von semantischen Modellen allenfalls in der Verfügbarkeit von Typ-Konstruktoren unterscheiden. Den vorläufigen Abschluß der Evolution von Datenmodellen bilden objektorientierte Modelle, die neben einer Struktur-Modellierung auch eine Modellierung von Objekt-Verhalten erlauben.

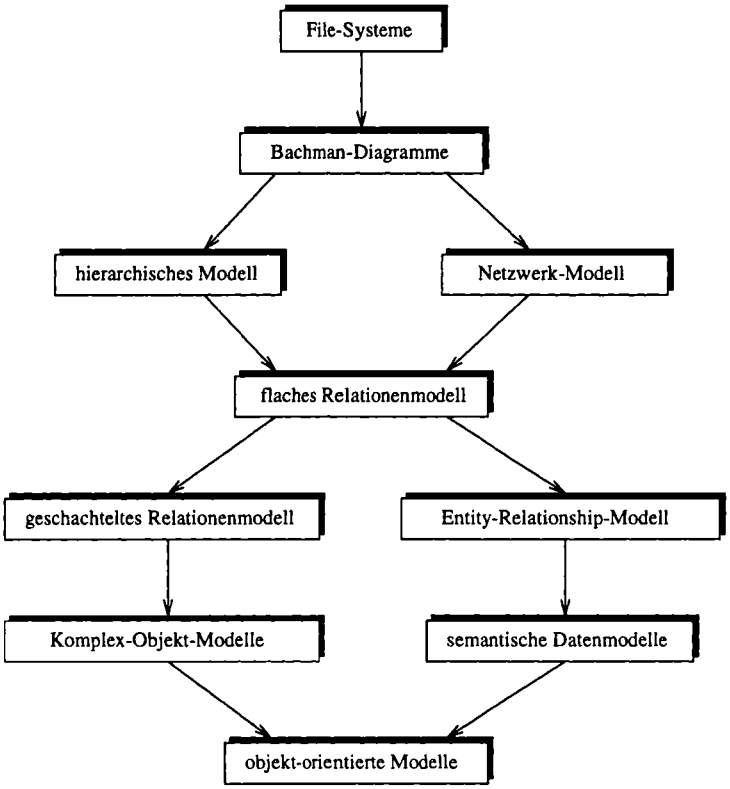

Abbildung 1.5: Evolution von Datenmodellen.

1.3 Relationale Datenbanken und SQL

Ein systematisches Studium der Grundlagen von Datenbanken begann mit der Einführung des relationalen Modells. Wie jedes andere Datenmodell verfügt auch das relationale über einen strukturellen Anteil (Schemata mit Abhängigkeiten) und einen operationalen Anteil (die Relationenalgebra). Im relationalen Modell sind die logische und die physische Betrachtungsebene (im Sinne des ANSI/SPARC-Modells) unabhängig voneinander. Auf der logischen Ebene existiert als einziges Grundkonstrukt die *Relation*; Relationen können den Benutzern anschaulich in Form einer Tabelle präsentiert werden (vgl. Abbildung 1.4). Wir wollen in diesem Abschnitt einige gebräuchliche Konzepte des Relationenmodells und der Sprache SQL einführen. Dies soll einmal als Motivation für die Betrachtung objekt-orientierter Datenbanken dienen und zum anderen berücksichtigen, daß aller Voraussicht nach SQL, entsprechend erweitert, auch für objekt-orientierte Datenbanken eine große Rolle spielen wird.

1.3.1 Strukturelle Aspekte

Eine relationale Datenbank wird auf konzeptioneller Ebene durch ein *Schema* beschrieben, in welchem die strukturellen Aspekte der Datenbank festgelegt sind. Ein relationales Datenbankschema besteht aus einer Menge von Relationenschemata, welche über interrelationale Bedingungen miteinander in Beziehung stehen. Ein Relationenschema wiederum besteht aus einem Namen, einer Menge von Attributen und ihren Wertebereichen sowie einer Menge intrarelationaler Bedingungen; eine Relation über diesem Schema besteht aus Tupeln, welche die Datenwerte enthalten und die inter- sowie intrarelationalen Bedingungen erfüllen. Eine Relation wird im allgemeinen als Tabelle dargestellt. Die in SQL für Relation, Tupel bzw. Attribut üblichen Bezeichnungen sind *table* (Tabelle), *row* (Zeile) bzw. *column* (Spalte).

Das zentrale SQL-Kommando zur Daten-Definition ist das Kommando `CREATE TABLE`, mit welchem neue Relationenschemata (im Kontext eines zuvor deklarierten Datenbankschemas) definiert werden können. Für jedes solche Schema können als Integritätsbedingungen zumindest Schlüssel-Definitionen, Entity- und referentielle Integrität unterschieden werden. Die Syntax dieses Kommandos lautet:

```
CREATE TABLE table-name
   ( column-name-1 type [ NOT NULL | UNIQUE ]
     [, column-name-2 ... ]
        .
        .
     [, UNIQUE ( list-of-column-names ) [, UNIQUE ... ] ]
     [, PRIMARY KEY ( list-of-column-names ) ]
     [, FOREIGN KEY ( list-of-column-names )
          REFERENCES table-name-2 [( list-of-column-names )]
     [, FOREIGN KEY ... ] ]
     [, CHECK ( condition ) [, CHECK ... ] ]   )
```

Verschiedene, uns hier nicht weiter interessierende Optionen einer Tabellen-Definition sind dabei nicht angegeben. In einem solchen Kommando werden also zunächst Attributnamen und deren Typ festgelegt; daneben kann jedes Attribut optional als frei von Nullwerten oder als eindeutig (d.h. alle Werte sind verschieden) deklariert werden. Als Datentypen stehen z.B. die folgenden zur Verfügung: `CHAR`, `VARCHAR`, `INT`, `SMALLINT`, `DEC`, `FLOAT`, `BIT`, `DATE`, `TIME`, `TIMESTAMP`.

Nach obiger Syntax kann eine Integritätsbedingung in der Definition einer Tabelle insbesondere eine Primärschlüssel-Definition, eine Fremdschlüssel-Definition oder eine Check-Klausel sein; daneben kennt SQL Domain- sowie Attribut-Bedingungen sowie sogenannte „Assertions". Wir beschränken uns hier auf die Beschreibung von Schlüsseln und Fremdschlüsseln.