Andreas Heuer
Gunter Saake
Kai-Uwe Sattler

4. Auflage

mitp

Datenbanken

Implementierungstechniken

Andreas Heuer
Gunter Saake
Kai-Uwe Sattler

Datenbanken

Implementierungstechniken

Vierte Auflage

mitp

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.

ISBN 978-3-95845-780-5
4. Auflage 2019

www.mitp.de
E-Mail: mitp-verlag@sigloch.de

Telefon: +49 7953 / 7189 - 079
Telefax: +49 7953 / 7189 - 082

Lektorat: Lisa Kresse, Sabine Janatschek
Sprachkorrektorat: Astrid Sander, Petra Heubach-Erdmann
Covergestaltung: Christian Kalkert, www.kalkert.de
Bildnachweis Cover: iStock.com/Shawn Hempel | fotolia.com/karpenko_ilia
Satz: Andreas Heuer, Rostock; Gunter Saake, Magdeburg; Kai-Uwe Sattler, Ilmenau
Druck: Impress Media GmbH, Mönchengladbach

Vorwort zur vierten Auflage

Das Gebiet der *Datenbanksysteme* gehört zu den klassischen Ausbildungsgebieten der Informatikstudiengänge. Datenbanksysteme kommen immer dann zum Einsatz, wenn an die Datenhaltung besondere Anforderungen hinsichtlich der Zuverlässigkeit, des zu speichernden Volumens, der Ausfallsicherheit, des Mehrbenutzerzugriffs, der Komplexität der Datenbeschreibung oder der Datenqualität gestellt werden. Zu Beginn des Informationszeitalters ist es daher nicht verwunderlich, dass der Umgang mit Datenbanksystemen für viele Absolventinnen und Absolventen der Informatikstudiengänge zum Berufsalltag gehört.

Viele Grundkonzepte der Datenbanktechnologie wurden in den 70er-Jahren entwickelt und seitdem beständig weiterentwickelt. Nach einer Stabilisierung in den 80er-Jahren – die damalige relationale Basistechnologie wurde fälschlicherweise als das Universalrüstzeug für alle Arten von Anwendungen angesehen – wird die aktuelle Entwicklung von speziellen Anwendungsanforderungen (Datenanalyse, Maschinelles Lernen, Data Science), speziellen Datentypen und Operationen (Geoinformationssysteme, Multimediaanwendungen) und neuen Architekturen (auch der zugrunde liegenden Hardware) geprägt, die die etablierten Techniken in vielfältiger Weise weiterentwickeln. Als Resultat ist das Gebiet der Implementierungstechniken derart vielgestaltig geworden, dass es in einem üblichen Grundlagenlehrbuch und einer Datenbankgrundvorlesung nur noch knapp angerissen werden kann.

Bereits seit der ersten Auflage des Buches *Datenbanken – Konzepte und Sprachen* [HS95a], das sich auf Modelle und Benutzersprachen konzentriert, wird daher immer ein spezieller Folgeband zum Thema Implementierungstechniken für Datenbanken bereitgestellt. Diese Aufteilung auf zwei Bände erlaubt es, Implementierungskonzepte ausführlicher vorzustellen. Aufgrund der Covergestaltung hat sich für diese Lehrbücher die Bezeichnung *Biberbücher* etabliert. Das Biber-1-Buch ist das für Konzepte und Sprachen, das hier vorliegende Buch über Implementierungstechniken ist dann das Biber-2-Buch.

Früher waren nur wenige in Deutschland ausgebildete Informatiker oder Wirtschaftsinformatiker an der Implementierung eines *echten* Datenbankmanagementsystems beteiligt. Wie üblich bei derartiger Grundsoftware wurde der Markt von wenigen (in der Regel amerikanischen) Firmen dominiert. Mittlerweile gibt es aber diverse Entwicklungsabteilungen von Datenbankmanagementsystemen oder wesentlicher Komponenten davon auch in Deutschland. Als größter Vertreter sei SAP erwähnt, mit einem eigenen Datenbank-Campus in Deutschland.

Aber auch außerhalb der Datenbankhersteller gibt es eine Reihe guter Gründe, sich intensiv mit den in diesem Buch diskutierten Implementierungstechniken zu beschäftigen:

- Ursprünglich für Datenbanksysteme entwickelte Einzelalgorithmen und Datenstrukturen können auch in anderer Software erfolgreich eingesetzt werden.
- Komplexe Anwendungen von Datenbanksoftware, die einen Schwerpunkt in kommerziellen Arbeitsgebieten für Absolventen bilden, erfordern ein tiefes Verständnis der Abläufe in Datenbanksystemen, um diese auch bei sehr großen Datenbeständen und Transaktionslasten effizient zu tunen.

Nicht zuletzt ist die Implementierung von Datenbanksystemen ein Gebiet, das hohe Anforderungen an Datenstrukturen und Algorithmen stellt und daher per se interessant für alle ist, die sich intensiv mit Informatik und Software Engineering beschäftigen.

Die erste und zweite Auflage

Abbildung 1: Die zweite Auflage des Biber-2-Buchs

Das Gebiet der Implementierung von Datenbanksystemen entwickelt sich stetig weiter. Schon die zweite Auflage dieses Buches [SHS05] erreichte eine Dicke von über 850 Seiten. Auf dem Weg zur dritten Auflage hätte im Jahr 2010 ein einfaches Hinzufügen neuerer Entwicklungen zu einem Lehrbuch von über 1000 Seiten geführt — zu viel für das Lehrbuchbudget eines typischen Studenten und zu schwer, um es zum Stöbern in die Tasche zu stecken. Schweren Herzens haben sich die Autoren daher bereits vor der dritten Auflage dazu entschlossen, drastische Einschnitte bei der Themenauswahl vorzunehmen.

Die dritte Auflage

Die dritte Auflage des Biber-2-Buchs [SSH11] behandelt die Basistechnologien zentraler, insbesondere relationaler Datenbankmanagementsysteme: Architektur, Datenorganisation, Anfragebearbeitung, Synchronisation im Mehrbenutzerbetrieb und Recovery. Diese Themen werden detailliert und ergänzt um neuere Entwicklungen dargestellt und behandelt. Dieses Basiswissen soll den Leser insbesondere in die Lage versetzen, die Eigenschaften eines relationalen DBMS zu verstehen und bei auftretenden Problemen kompetent kausale Zusammenhänge zwischen Problemsituationen und Implementierungs- bzw. Parametrisierungsentscheidungen herzustellen.

Abbildung 2: Die dritte Auflage des Biber-2-Buchs

Weggefallen sind bereits in der dritten Auflage einige Techniken, die für spezielle Anwendungsszenarien und andere Datenmodelle relevant sind, so Indexstrukturen für multimediale Daten oder die Realisierung von Objektidentifikatoren in objektorientierten Datenbanken.

Der größte Einschnitt allerdings betrifft den Verzicht auf die Behandlung der Aspekte verteilter Datenhaltung sowie der Parallelisierungspotenziale. Einige wenige Punkte nur erschienen den Autoren so zentral, dass sie weiterhin behandelt werden, so die Fragen der Partitionierung oder der Commit-

Protokolle. Dieser Verzicht ist den Autoren besonders schwergefallen, da die aktuellen Entwicklungen im Hardwarebereich und der Softwareinfrastruktur (Stichworte Cloud-Computing, P2P-Datenhaltung, Multi-Core-Prozessoren) gerade hier viel Entwicklungspotenzial zeigen. Aber ohne diesen Einschnitt wäre ein Seitenlimit bei etwa 700 Seiten nicht zu halten gewesen. Hier liegt aber mittlerweile mit [RSS15] ein dediziertes Lehrbuch zu diesen Themenkreisen vor.

Die aktuelle vierte Auflage

In der hier vorliegenden vierten Auflage des Biber-2-Buches konnten wir glücklicherweise ohne drastische Kürzungen einige Techniken ergänzen, die entweder in dem bisherigen Buch weggelassen wurden oder erst im letzten Jahrzehnt eine entscheidende Bedeutung gewonnen haben. Dazu gehören neue Techniken der semantischen Anfrageoptimierung und erweiterte Transaktionsmodelle, hier insbesondere Techniken der Snapshot Isolation, der Prädikatsperren, der Sperren in Indexstrukturen sowie der Varianten von bestimmten Sperrprotokollen. Natürlich wurden alle Angaben zu Größenordnungen von Speichern und zu Prozessoren an die Aktualität angepasst.

Noch wesentlicher ist aber die Neustrukturierung des Buches. Ihren Ursprung hat diese darin, dass wir ein extrem langes (und immer länger werdendes) Kapitel über Anfrageoptimierung nun in drei Kapitel zerlegt haben. Danach haben wir das Buch nun in drei fachlich zusammenhängende Teile gegliedert:

- Teil I behandelt in den Kapiteln 3 bis 6 Speichermodelle und Zugriffspfade.
- Teil II stellt in den Kapiteln 7 bis 10 die Techniken der Anfragebearbeitung und -optimierung vor. Hier befindet sich auch das alte Kapitel 8 der dritten Auflage, das nun in die Kapitel 8 bis 10 zerlegt wurde.
- Teil III schließlich umfasst die Kapitel 11 bis 13 und behandelt die Transaktionsverarbeitung und das Recovery.

Die Kapitel 1 und 2 vor Teil I wiederholen einige Grundbegriffe aus dem Biber-1-Buch und stellen die internen Datenbankarchitekturen im Überblick vor.

Neu ist auch das abschließende Kapitel 14 über *moderne Datenbanksystem-Architekturen*, das das einzige Kapitel des Teils IV bildet. Diesen letzten, noch kleinen, Teil haben wir mit *aktuellen Entwicklungen* überschrieben. Während die Teile I, II und III des Buches sich auf klassische Datenbanksysteme für OLTP-Anwendungen (transaktionsverarbeitende Anwendungen) konzentrieren, befasst sich Kapitel 14 mit den neuen Datenbank-Architekturen für OLAP-Anwendungen (analysierende Anwendungen), Data Science und Big Data Analytics. Letztere sind derzeit noch ein Hype in der Datenbank-Forschung. Sobald der Hype abflaut, müssen die dann sich etablierenden Techniken in einem eigenen Lehrbuch zu dieser Art von Datenbanksystemen ergänzt werden.

Das Themenspektrum dieser vierten Auflage deckt aber – zusammen mit dem Lehrbuch über Konzepte und Sprachen von Datenbanken [SSH18] – genau die Aspekte von Datenbanksystemen ab, die nach den Empfehlungen des deutschen Fakultätentags für Informatik in einem Informatik-Bachelor-Studium zwingend erforderlich sind. Natürlich werden innerhalb dieses Themenspektrums die Inhalte in diesem Buch so breit gefächert und vertieft, dass auch spezielle Vorlesungen für Master-Studiengänge aus dem hier dargestellten Stoff abgeleitet werden können. Umgekehrt sind aber keine Themengebiete in dieser vierten Auflage entfallen, die zum Pflichtkanon eines Informatik-Bachelors gehören.

Schreibweisen und Umgebungen

In diesem Buch werden wir folgende Schreibweisen verwenden:

- Wortsymbole aus Programmier- oder Datenbanksprachen werden wie bei **`select`** oder **`where`** geschrieben.

- Namen von Elementen eines Datenbankschemas, wie Attribut- oder Relationennamen, werden wie bei `KUNDE` oder `Nachname` geschrieben.

- Einträge in einer Datenbank oder Attributwerte in Programmen oder Datenbankanfragen werden wie in 4711 oder Meyer notiert.

- Begriffe, die an der betreffenden Stelle im Buch gerade definiert werden, werden wie in *Tableau-Optimierung* oder *Serialisierbarkeit* hervorgehoben. Diese Kursivschrift wird teilweise auch für andere Hervorhebungen wie *Betonungen* oder englische (nicht übersetzte) *Fachbegriffe* verwendet.

Wir werden viele Konzepte anhand von Beispielen erklären. Die Beispiele beziehen sich zum größten Teil auf eine Datenbank, die im Anhang noch einmal unter „Laufendes Beispiel“ aufgeführt ist. Dieses Beispiel beschreibt eine kleine Anwendung für den Vertrieb von Kaffeespezialitäten. Wenn wir im laufenden Text die allgemeine Erläuterung von Konzepten deutlicher von veranschaulichenden Beispielen abheben wollen, so verwenden wir in einigen Kapiteln des Buches eine eigene Beispielumgebung.

Beispiel 0.1 Diese Beispielumgebung ist dann pro Kapitel durchnummeriert, sodass wir auf Beispiele verweisen können. Das Beispiel endet schließlich mit einem kleinen Kästchen am rechten Spaltenrand. □

Durch dieses Kästchen kann man das Ende des Beispiels, in diesem Fall Beispiel 0.1, und damit die Fortsetzung des erklärenden Textes leichter finden.

Danksagungen

Buchprojekte wie dieses haben – zumindest, wenn man die Autorentätigkeit nicht als Vollzeitjob betreibt – die Eigenschaft, dass der zeitliche Aufwand meist völlig unterschätzt wird. So müssen dann immer die Familien gestresste Autoren ertragen, die „nur noch“ ein Kapitel, einen Abschnitt etc. fertigstellen müssen. Natürlich wissen wir, dass dies durch Widmungen oder Danksagungen nicht ausgeglichen werden kann. Dennoch möchten wir uns an dieser Stelle bei unseren Familien für ihr Verständnis, ihre Geduld und ihren Rückhalt bedanken – ohne sie wäre dieses Buch sicher nie fertig geworden.

Ein besonderer Dank gilt David Broneske für die Unterstützung bei Abbildungen und Beispielen und Holger Meyer für Zuarbeiten im Bereich von Snapshot Isolation, Prädikatsperren sowie strikten und rigoros strikten Schedules und Protokollen. Für das Korrekturlesen der dritten Auflage bedanken wir uns bei Hannes Grunert, Holger Meyer, Ilvio Bruder, Tanja Auge, Mark Lukas Möller, Johannes Goltz, Michael Poppe, Henrik Hertel, Enrico Gruner, Frank Röger, Dirk Hamann und der Sprachkorrektorin Petra Heubach-Erdmann. Die Übungsaufgaben entstammen dem in den Jahren entstandenen Fundus der beteiligten Lehrstühle, ein spezieller Dank gilt Andreas Meister. Und schließlich gilt den Lektorinnen des mitp-Verlags, Sabine Janatschek und Lisa Kresse, ein Dank für die unendliche Geduld: Die Fertigstellung des endgültigen Manuskripts hat schließlich über ein halbes Jahr länger gedauert als ursprünglich geplant.

Ergänzende Informationen zum Buch wie Verweise zu begleitenden Vorlesungsmaterialien und gegebenenfalls erforderliche Fehlerkorrekturen sind im Web unter folgender Adresse zu finden:

`http://www.biberbuch.de`

Rostock, Magdeburg und Ilmenau, im April 2019

Andreas Heuer, Gunter Saake und Kai-Uwe Sattler

Inhaltsverzeichnis

1

Aufgaben und Prinzipien von Datenbanksystemen

Datenbanksysteme ermöglichen die integrierte Speicherung von großen Datenbeständen, auf die mehrere Anwendungen gleichzeitig zugreifen können. Hierbei garantiert das Prinzip der *Datenunabhängigkeit* die weitestgehende Unabhängigkeit der Datenrepräsentation von Optimierung und Änderung der Speicherstrukturen. Sie ermöglicht auch eine Reaktion auf Änderungen der Anwendungsanforderungen, ohne die logische Struktur der Daten ändern zu müssen. Diese allgemeinen Anforderungen stellen zusammen genommen hohe Anforderungen an die interne Realisierung von Datenbanksystemen.

Für Datenbanksysteme müssen daher speziell zugeschnittene Algorithmen und Datenstrukturen insbesondere für die folgenden internen Aufgaben entwickelt werden:

- Um eine effiziente Speicherung der Daten und ein schnelles Wiederauffinden zu ermöglichen, werden unterschiedliche, auf große Datenbestände optimierte, interne *Zugriffsdatenstrukturen* eingesetzt.

- Die Datenunabhängigkeit erzwingt, dass Benutzer und Anwendungsprogrammierer die vom Datenbanksystem bereitgestellten Zugriffsdatenstrukturen nicht direkt ausnutzen können – die *Zugriffsoptimierung* muss durch das Datenbanksystem erfolgen.

- Das System muss den Mehrbenutzerbetrieb kontrollieren, um unerwünschte Konflikte beim gleichzeitigen Zugriff auf Daten auszuschließen.

- Weitere Aufgaben umfassen Vorkehrungen zur Wiederherstellung der Daten nach Systemausfällen, Kooperation zwischen verteilten Datenhaltungssystemen und Unterstützung der Wartungsphase.
- Als standardkonforme Systemsoftware müssen Datenbanksysteme auf verschiedenen Rechnerarchitekturen effizient arbeiten.

Die Realisierung dieser Aufgaben in der Implementierung von Datenbanksystemen ist Inhalt dieses Buches. Der Schwerpunkt liegt dabei auf Konzepten kommerzieller, meist relationaler Datenbanksysteme, wobei aber auch weitere zukunftsweisende Entwicklungen sowie Spezialentwicklungen betrachtet werden.

In diesem ersten Kapitel werden wir zunächst die notwendigen, grundlegenden Datenbankkonzepte wiederholen. Die Wiederholung orientiert sich an der Darstellung im ersten Band dieser Buchreihe (dem „Biber-Buch“ [SSH18]). Insbesondere werden für die weiteren Kapitel des Buches Grundkenntnisse der theoretischen Grundlagen von Datenbanksystemen, insbesondere der Relationenalgebra, und von Datenbanksprachen vorausgesetzt. Speziell werden Basiskenntnisse von SQL erwartet.

Eine detaillierte Kapitelübersicht des Buches findet sich in Abschnitt 1.2. Ferner werden wir eine weitgehend durchgängig verwendete Beispielanwendung vorstellen.

1.1 Wiederholung der Datenbank-Grundbegriffe

Dieser Abschnitt wiederholt wichtige Grundkenntnisse über Datenbanksysteme, die zum Lesen dieses Buches vorausgesetzt werden. Sollte die Darstellung in diesem Kapitel zu knapp sein oder sollten Verständnisschwierigkeiten bei den hier gegebenen Erläuterungen auftreten, so empfehlen wir zunächst das Studium der einschlägigen Kapitel des Biber-Buches [SSH18].

In diesem Abschnitt werden die Bereiche *Architekturen von Datenbanksystemen*, *Datenmodelle und Datendefinition* sowie *Anfragen und Änderungsoperationen* kurz angesprochen.

1.1.1 Architektur eines Datenbanksystems

Die Abbildung 1.1 zeigt einen Überblick über die prinzipielle Aufteilung eines Datenbankmanagementsystems in Funktionsmodule. Die Darstellung ist angelehnt an eine Aufteilung in drei Abstraktionsebenen. Die *externe Ebene* beschreibt die Sicht, die eine konkrete Anwendung auf die gespeicherten Daten hat. Da mehrere angepasste externe Sichten auf eine Datenbank existieren können, gibt die *konzeptuelle Ebene* eine logische und einheitliche Gesamtsicht

auf den Datenbestand. Die *interne Ebene* beschreibt die tatsächliche interne Realisierung der Datenspeicherung.

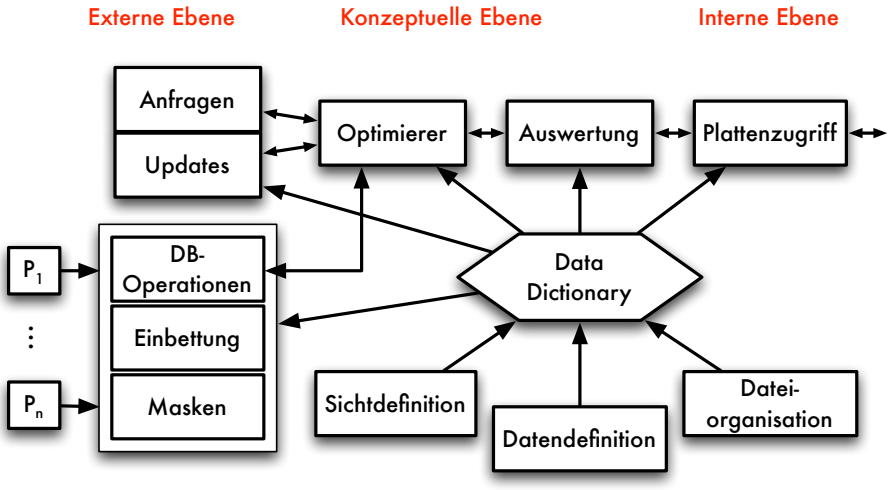

Abbildung 1.1: Vereinfachte Architektur eines DBMS

Die in Abbildung 1.1 gezeigten Komponenten können wie folgt kurz charakterisiert werden:

- Die Komponente *Dateiorganisation* beinhaltet die Definition der Dateiorganisation und Zugriffspfade auf der internen Ebene.
- Die Komponente *Datendefinition* betrifft die konzeptuelle Datendefinition, das heißt die Definition des *konzeptuellen Schemas*.
- In der Komponente *Sichtdefinition* erfolgt die Definition von Benutzersichten, also die Deklaration der Datendarstellung auf der externen Ebene.
- Die Komponente zur Definition von *Masken* beinhaltet den Entwurf von Menüs und Masken für die Benutzerinteraktion.
- Die Komponente *Einbettung* von Konstrukten der Datenbanksprache in eine Programmiersprache bildet die Schnittstelle zur Anwendungsprogrammierung (vgl. [SSH18, Kapitel 13]).
- Die Komponenten zur Bearbeitung von *Anfragen* und *Änderungen* (*Updates*) ermöglichen einen interaktiven Zugriff auf den Datenbestand.

- Die Komponente *Datenbankoperationen* (kurz *DB-Operationen*) realisiert die Datenbankoperationen für Anfragen und Änderungen, die von Anwendungen genutzt werden.
- Der Komponente *Optimierer* übernimmt die Optimierung der Datenbankzugriffe.
- Die Komponente des *Plattenzugriffs* realisiert die Plattenzugriffssteuerung.
- Die Komponente *Auswertung* betrifft die Auswertung der Ergebnisse von Anfragen und Änderungen.
- $P_1 \ldots P_n$ sind verschiedene Datenbankanwendungsprogramme.
- Das *Data Dictionary* (oft auch Katalog oder Datenwörterbuch genannt) bildet den zentralen Katalog aller für die Datenhaltung relevanten Informationen.

Die verschiedenen Komponenten eines Datenbanksystems können dabei zu folgenden Klassen zusammengefasst werden (siehe Abbildung 1.2):

- Die *Definitionskomponenten* bieten Datenbank-, System- und Anwendungsadministratoren die Möglichkeit zur Datendefinition, Definition der Dateiorganisationsformen und Zugriffspfade sowie zur Sichtdefinition.
- Die *Programmierkomponenten* beinhalten eine vollständige Entwicklungsumgebung in einer höheren Programmiersprache, einer 4GL oder einer grafischen Sprache, die Datenbankoperationen und in den meisten Fällen auch Werkzeuge zur Definition von Menüs, Masken und andere Primitiven einer grafischen Benutzeroberfläche einbettet.
- Die *Benutzerkomponenten* umfassen die interaktiven Anfrage- und Änderungswerkzeuge für anspruchsvolle Laien und die vorgefertigten Datenbankanwendungsprogramme für den unbedarften Benutzer („parametric user“).
- Die *Transformationskomponenten* wandeln Anfrage- und Änderungsoperationen schrittweise über Optimierung und Auswertung in Plattenzugriffsoperationen um. Umgekehrt werden die in Blöcken der Platte organisierten Bytes außerdem in die externe Benutzerdarstellung (im Relationenmodell: Tabellen) transformiert.
- Der zentrale Kern des ganzen Systems ist das *Data Dictionary*, das die Daten aus den Definitionskomponenten aufnimmt und die Programmier-, Benutzer- und Transformationskomponenten mit diesen Informationen versorgt.

Gerade die Transformationskomponenten sind in der Drei-Ebenen-Architektur noch etwas ungenau beschrieben. Die im nächsten Kapitel beschriebene Fünf-Schichten-Architektur wird die schrittweise Transformation von Operationen und Daten genauer darlegen.

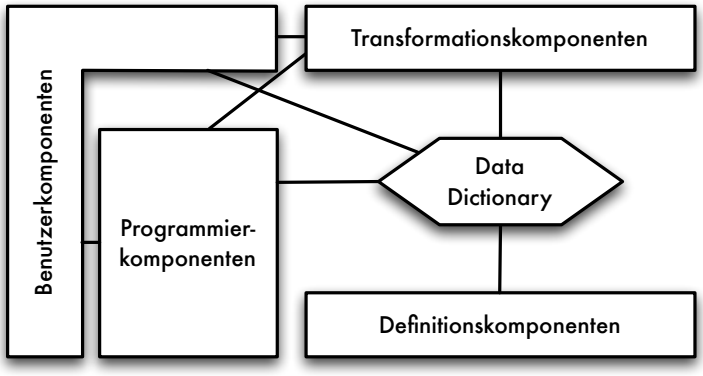

Abbildung 1.2: Klassifikation der Komponenten eines DBMS aus Abbildung 1.1

1.1.2 Neun Funktionen nach Codd

Im Laufe der Jahre hat sich eine Basisfunktionalität herauskristallisiert, die von einem Datenbankmanagementsystem erwartet wird. Codd hat 1982 diese Anforderungen in einer Liste von neun Punkten[1] zusammengefasst [Cod82]:

1. *Integration*

   Die Datenintegration erfordert die *einheitliche* Verwaltung *aller* von Anwendungen benötigten Daten. Hier verbirgt sich die Möglichkeit der kontrollierten nicht-redundanten Datenhaltung des gesamten relevanten Datenbestands.

2. *Operationen*

   Auf der Datenbank müssen Operationen möglich sein, die Datenspeicherung, Suchen und Änderungen des Datenbestands ermöglichen.

3. *Katalog*

   Der Katalog, auch „Data Dictionary“ genannt, ermöglicht Zugriffe auf die Datenbeschreibungen der Datenbank.

[1] Wer in [Cod82] nur acht Anforderungen entdeckt: die von uns mit Nummer 4 erwähnte Regel *Benutzersichten* führt Codd als *additional service* nach der Aufzählung der anderen acht Anforderungen ein.

4. *Benutzersichten*

   Für unterschiedliche Anwendungen sind unterschiedliche Sichten auf den Datenbestand notwendig, sei es in der Auswahl relevanter Daten oder in einer angepassten Strukturierung des Datenbestands. Die Abbildung dieser speziellen Sichten auf den Gesamtdatenbestand muss vom System kontrolliert werden.

5. *Konsistenzüberwachung*

   Die Konsistenzüberwachung, auch als *Integritätssicherung* bekannt, übernimmt die Gewährleistung der Korrektheit von Datenbankinhalten und der korrekten Ausführung von Änderungen, sodass diese die Konsistenz nicht verletzen können.

6. *Datenschutz*

   Aufgabe des Datenschutzes ist der Ausschluss unautorisierter Zugriffe auf die gespeicherten Daten. Dies umfasst datenschutzrechtlich relevante Aspekte personenbezogener Informationen ebenso wie den Schutz firmenspezifischer Datenbestände vor Werksspionage.

7. *Transaktionen*

   Unter einer Transaktion versteht man eine Zusammenfassung von Datenbankoperationen zu Funktionseinheiten, die als Ganzes ausgeführt werden sollen und deren Effekt bei Erfolg permanent in der Datenbank gespeichert werden soll.

8. *Synchronisation*

   Konkurrierende Transaktionen mehrerer Benutzer müssen synchronisiert werden, um gegenseitige Beeinflussungen, etwa versehentliche Schreibkonflikte auf gemeinsam benötigten Datenbeständen, zu vermeiden.

9. *Datensicherung*

   Aufgabe der Datensicherung ist es, die Wiederherstellung von Daten etwa nach Systemfehlern zu ermöglichen.

Die Punkte 1, 2, 5 und 6 werden schwerpunktmäßig in [HS00, SSH18] behandelt, während die anderen Punkte Inhalt dieses Buches sind.

1.1.3 Datenbankmodelle und Datendefinition

Es gibt verschiedene Datenbankmodelle, die für die Datenbeschreibung auf der konzeptuellen Ebene eingesetzt werden. Kommerziell am erfolgreichsten sind dabei zurzeit ohne Zweifel die relationalen Datenbanksysteme bzw. deren Erweiterung um objektrelationale Konzepte. Der Sprachstandard für diese Systeme ist SQL.

Wir werden in diesem Buch daher primär die Implementierung von relationalen Datenbanksystemen als Motivation für die vorgestellten Techniken verwenden. Natürlich können diese Techniken (teils in abgewandelter Form) auch für andere Datenbankmodelle eingesetzt werden. Am Ende des Buches werden wir auch auf Implementierungstechniken für andere Datenbankmodelle eingehen, die etwa den NoSQL-Systemen zugrunde liegen (siehe Kapitel 14).

Konzeptionell ist eine relationale Datenbank eine Ansammlung von *Tabellen*. Hinter den Tabellen steht mathematisch die Idee einer *Relation*, ein grundlegender Begriff, der dem Ansatz den Namen gegeben hat.

Die folgenden zwei Tabellen sollen die Produkt- und Lieferantendaten eines Kaffeehändlers repräsentieren.

PRODUKT

ProdNr	Bezeichnung	Preis	LName
1042	Jamaica Blue	14,90	CoffeeShop
1043	Arabica Black	10,90	Kaffeebude
1044	New York Espresso	18,20	Kaffeebude
1045	Arabica Black	12,00	CoffeeDealer

LIEFERANT

LName	Adresse
CoffeeShop	München
Kaffeebude	Berlin
CoffeeDealer	Magdeburg

Wir verwenden in diesem Abschnitt die folgenden begrifflichen Konventionen: Die erste Zeile gibt jeweils die Struktur einer Tabelle an (Anzahl und Benennung der Spalten). Diese Strukturinformation bezeichnen wir als *Relationenschema* (als Pluralform von Schema verwenden wir *Schemata*). Die weiteren Einträge in der Tabelle bezeichnen wir als *Relation* zu diesem Schema. Eine einzelne Zeile der Tabelle bezeichnen wir als *Tupel*. Spaltenüberschriften werden als *Attribut(namen)* bezeichnet.

Formalisierung von Relationenschemata und Relationen

Im weiteren Verlauf des Buches, insbesondere im Teil der Anfragebearbeitung, müssen wir die Konzepte des Relationenmodells etwas genauer betrachten. Daher wiederholen wir hier kurz die wichtigsten Aspekte der Formalisierung des relationalen Datenbankmodells, die wir in [SSH18, Kapitel 5] ausführlicher vorgestellt haben. In diesem Absatz geht es dabei zunächst um die Begriffe *Attribut*, *Relationenschema* und *Relation*.

Attribute und Wertebereiche

Die elementaren Bausteine einer relationalen Datenbank sind Attribute. Jedem Attribut wird ein Wertebereich zugeordnet. Mit $\mathcal{U}$ wird das *Universum* der

Attribute bezeichnet, jedes $A_i \in \mathcal{U}$ heißt *Attribut*. Jedem Attribut wird mit D_i ein *Wertebereich* oder eine *Domäne* zugeordnet, $\text{dom}(A_i)$ heißt dann der *Wertebereich von* A_i. Ein $w \in \text{dom}(A)$ wird *Attributwert* für A genannt.

Relationenschemata und Relationen

Eine Menge $R \subseteq \mathcal{U}$ heißt *Relationenschema*. Eine *Relation* r *über* $R = \{A_1, \ldots, A_n\}$ (kurz: $r(R)$) mit $n \in \mathbb{N}$ ist eine endliche Menge von Abbildungen

$$t : R \longrightarrow \bigcup_{i=1}^{m} D_i$$

die *Tupel* genannt werden, wobei $t(A) \in \text{dom}(A)$ gilt. $t(A)$ ist dabei die Restriktion der Abbildung t auf $A \in R$. Die Menge aller Relationen über einem Relationenschema R wird mit $\mathbf{REL}(R)$ bezeichnet und folgendermaßen definiert:

$$\mathbf{REL}(R) := \{r \mid r(R)\}$$

$\mathbf{REL}(R)$ besteht aus allen Relationen, die ich aus den Attributwerten der den Attributen zugeordneten Wertebereichen bilden kann. Es sind also alle Kombinationen von Attributwerten innerhalb einer Relation erlaubt. Wir werden diese Menge der möglichen Relationen im nächsten Absatz durch Integritätsbedingungen einschränken.

Vorher definieren wir noch die Begriffe Datenbankschema und Datenbank. Eine Menge von Relationenschemata $S := \{R_1, \ldots, R_p\}$ mit $p \in \mathbb{N}$ heißt *Datenbankschema*. Eine *Datenbank* über einem Datenbankschema S ist eine Menge von Relationen

$$d := \{r_1, \ldots, r_p\}$$

wobei $r_i(R_i)$ für alle $i \in \{1, \ldots, p\}$ gilt. Eine Datenbank d über S wird mit $d(S)$ bezeichnet, eine Relation $r \in d$ heißt *Basisrelation*.

Integritätsbedingungen

Selbst in einem derart einfachen Datenstrukturierungsmodell wie dem relationalen ist es sinnvoll, bestimmte *Konsistenzforderungen* oder *Integritätsbedingungen* an gespeicherte Datenbanken zu stellen, die vom System gewährleistet werden müssen.

Betrachten wir die `PRODUKT`-Tabelle erneut. Die Einträge für Lieferanten in der `LName`-Spalte, in der folgenden Tabelle kursiv hervorgehoben, sollten sicher nicht beliebig gewählt werden dürfen.

PRODUKT	ProdNr	Bezeichnung	Preis	LName
	1042	Jamaica Blue	14,90	*CoffeeShop*
	1043	Arabica Black	10,90	*Kaffeebude*
	1044	New York Espresso	18,20	*Kaffeebude*
	1045	Arabica Black	12,00	*CoffeeDealer*

Von jedem `LName`-Eintrag erwarten wir, dass er tatsächlich auf einen Lieferanteneintrag in der `LIEFERANT`-Tabelle verweist. Dies ist aber nur möglich, wenn diese Lieferantennamen eindeutig je eine Zeile identifizieren. Wir bezeichnen diese Eigenschaft als *Schlüsseleigenschaft*.

LIEFERANT	LName	Adresse
	CoffeeShop	München
	Kaffeebude	Berlin
	CoffeeDealer	Magdeburg

Derartig einfache Integritätsbedingungen sind im relationalen Datenbankmodell fest integriert. Wir werden darum im Folgenden jeweils Relationenschema *plus* Integritätsbedingungen betrachten.

Unter *lokalen* Integritätsbedingungen verstehen wir Bedingungen, die für genau eine Tabelle gewährleistet sein müssen. Etwa ist das Attribut `ProdNr` *Schlüssel* für `PRODUKT`, das heißt, eine `ProdNr` darf nicht doppelt vergeben werden oder anders ausgedrückt: In der Spalte `ProdNr` dürfen keine zwei gleichen Werte auftauchen.

Unter *globalen* Integritätsbedingungen verstehen wir Bedingungen, die über den Bereich einer Tabelle hinausreichen. Wir sagen, dass `LName` in der Tabelle `PRODUKT` ein *Fremdschlüssel* bezüglich `LIEFERANT` ist. Dies bedeutet, dass `LName` in einem anderen Relationenschema als Schlüssel auftaucht und die Lieferantennamen in `PRODUKT` auch in `LIEFERANT` auftreten müssen. Es sei noch angemerkt, dass es in einer realen Anwendung besser wäre, künstliche Lieferantennummern als Schlüssel zu verwenden, um bei Namensänderungen nicht Referenzierungsprobleme zu riskieren.

Formalisierung von Integritätsbedingungen

Die einfachsten lokalen Integritätsbedingungen sind identifizierende Attributmengen, Schlüssel und Primärschlüssel. Eine *identifizierende Attributmenge* für ein Relationenschema R ist eine Menge $K := \{B_1, \ldots, B_k\} \subseteq R$, sodass für jede Relation $r(R)$ gilt:

$$\forall t_1, t_2 \in r \; [\; t_1 \neq t_2 \quad \Longrightarrow \quad \exists B \in K : t_1(B) \neq t_2(B) \;].$$

Ein *Schlüssel* ist eine bezüglich $\subseteq$ minimale identifizierende Attributmenge. In einem Relationenschema kann es mehrere Schlüssel geben. Ein *Primärschlüssel* ist ein ausgezeichneter Schlüssel, pro Relationenschema gibt es nur einen Primärschlüssel.

Die Menge aller Relationen aus $\mathbf{REL}(R)$, die noch zusätzlich die lokalen Integritätsbedingungen $\mathcal{B}$ erfüllen, bezeichnen wir mit

$$\mathbf{SAT}_R(\mathcal{B}) := \{r \mid r(R) \text{ und } r \text{ genügt } \mathcal{B}\}.$$

SAT ist abgeleitet vom englischen Wort *satisfy*.

Die einfachsten globalen Integritätsbedingungen sind Fremdschlüssel. Ein *Fremdschlüssel* (engl. *Foreign Key*) ist eine Attributliste X in einem Relationenschema R_1, wenn in einem Relationenschema R_2 eine Attributmenge Y Primärschlüssel ist und die Attributwerte zu X in der Relation $r_1(R_1)$ auch in den entsprechenden Spalten Y der Relation $r_2(R_2)$ enthalten sind. Wir bezeichnen einen solchen Fremdschlüssel dann mit $X(R_1) \rightarrow Y(R_2)$, die zugehörige *Fremdschlüsselbedingung* für eine Relation $r_1(R_1)$ ist ein Ausdruck

$$X(R_1) \rightarrow Y(R_2)$$

mit $X \subseteq R_1, Y \subseteq R_2$. X nennt man dann *Fremdschlüssel* für R_1 bezüglich Y in R_2. Eine Datenbank d *genügt* $X(R_1) \rightarrow Y(R_2)$ genau dann, wenn eine Relation $r_2(\mathcal{R}_2)$ mit Y Primärschlüssel für r_2 in der Datenbank existiert und Folgendes erfüllt ist:

$$\{t(X) | t \in r_1\} \subseteq \{t(Y) | t \in r_2\}$$

Einfache lokale Integritätsbedingungen wie Schlüssel und Primärschlüssel werden bereits im ersten Teil des Buches eine Rolle spielen, da wir bei Dateiorganisationsformen und Zugriffspfaden (Kapitel 5) zwischen Schlüsseln und anderen Attributmengen (die wir dann Sekundärschlüssel nennen) unterscheiden müssen. Fremdschlüssel sowie Abhängigkeiten zwischen Attributen als Verallgemeinerungen von Schlüsseln und Fremdschlüsseln werden wir im zweiten Teil des Buches bei der Anfragebearbeitung und -optimierung benötigen (Kapitel 9).

Datendefinition in SQL

Zur Umsetzung der entworfenen Relationenschemata mit ihren Integritätsbedingungen in das Data Dictionary eines Datenbanksystems verwenden wir den Teil der Sprache SQL, der die *Datendefinition* ermöglicht und daher auch als DDL (Data Definition Language) bezeichnet wird.

Die folgende einfache SQL-Deklaration zeigt den prinzipiellen Aufbau von Tabellendeklarationen. Neben einem eindeutigen *Tabellennamen* werden insbesondere die Attribute mit ihren Wertebereichen aufgeführt.

```
create table PRODUKT (
   ProdNr int not null,
   Bezeichnung varchar(200),
   Preis decimal(5,2),
   LName varchar(30) )
```