



Andreas Heuer
Gunter Saake
Kai-Uwe Sattler
4. Auflage

Datenbanken

Implementierungstechniken





Hinweis des Verlages zum Urheberrecht und Digitalen Rechtemanagement (DRM)

Der Verlag räumt Ihnen mit dem Kauf des ebooks das Recht ein, die Inhalte im Rahmen des geltenden Urheberrechts zu nutzen. Dieses Werk, einschließlich aller seiner Teile, ist urheberrechtlich geschützt. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Verlages unzulässig und strafbar. Dies gilt insbesondere fürervielfältigungen, Übersetzungen, Mikroverfilmungen und Einspeicherung und Verarbeitung in elektronischen Systemen.

Der Verlag schützt seine ebooks vor Missbrauch des Urheberrechts durch ein digitales Rechtemanagement. Bei Kauf im Webshop des Verlages werden die ebooks mit einem nicht sichtbaren digitalen Wasserzeichen individuell pro Nutzer signiert.

Bei Kauf in anderen ebook-Webshops erfolgt die Signatur durch die Shopbetreiber. Angaben zu diesem DRM finden Sie auf den Seiten der jeweiligen Anbieter.

*Andreas Heuer
Gunter Saake
Kai-Uwe Sattler*

Datenbanken

Implementierungstechniken

Vierte Auflage



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

© 2019 mitp Verlags GmbH & Co KG, Frechen

Dieses Werk, einschließlich aller seiner Teile, ist urheberrechtlich geschützt. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Verlages unzulässig und strafbar. Dies gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Verarbeitung in elektronischen Systemen.

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zu der Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Lektorat: Lisa Kresse, Sabine Janatschek

Sprachkorrektorat: Astrid Sander, Petra Heubach-Erdmann

Covergestaltung: Christian Kalkert, www.kalkert.de

Bildnachweis Cover: [iStock.com/Shawn Hempel](http://iStock.com/ShawnHempel) | 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 Ausbildungsbereichen 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 Co-vergestaltung 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.

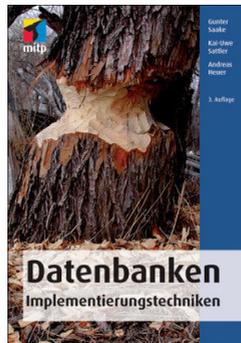


Abbildung 2: Die dritte Auflage des Biber-2-Buchs

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 schmerzlich, 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 Sperrungen 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 Datenbankabfragen 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

Vorwort zur vierten Auflage	v
Inhaltsverzeichnis	xi
1 Aufgaben und Prinzipien von Datenbanksystemen	1
1.1 Wiederholung der Datenbank-Grundbegriffe	2
1.1.1 Architektur eines Datenbanksystems	2
1.1.2 Neun Funktionen nach Codd	5
1.1.3 Datenbankmodelle und Datendefinition	6
1.1.4 Anfrage- und Änderungsoperationen	12
1.1.5 Sprachen und Sichten	13
1.2 Wann kommt was?	15
1.2.1 Optimierer	15
1.2.2 Dateiorganisation und Zugriffspfade	17
1.2.3 Transaktionen	20
1.2.4 Recovery und Datensicherheit	20
1.3 Vertiefende Literatur	21
1.4 Übungen	22
2 Architektur von Datenbanksystemen	23
2.1 Betrachtete Fragestellungen	24
2.2 Schichtenmodell eines relationalen DBMS	26
2.3 Hardware und Betriebssystem	29
2.4 Pufferverwaltung	31
2.5 Speichersystem	34
2.6 Zugriffssystem	35
2.7 Datensystem	39
2.8 Katalog und Data Dictionary	40
2.9 Vertiefende Literatur	42
2.10 Übungen	43

I	Speichermodelle und Zugriffspfade	45
3	Verwaltung des Hintergrundspeichers	47
3.1	Speichermedien	48
3.1.1	Speicherhierarchie	48
3.1.2	Cache, Hauptspeicher und Sekundärspeicher	51
3.1.3	Die Magnetplatte	52
3.1.4	Flash-Laufwerke	55
3.1.5	Speicherkapazität, Geschwindigkeit und Kosten	59
3.2	Speicher-Arrays: RAID	61
3.2.1	Ziele von RAID-Systemen	61
3.2.2	RAID-Levels	62
3.3	Sicherungsmedien: Tertiärspeicher	68
3.3.1	Optische Platten	69
3.3.2	Bänder	70
3.3.3	Jukeboxes und Roboter	70
3.3.4	Langzeitarchivierung	71
3.4	Modell des Hintergrundspeichers	72
3.4.1	Betriebssystemdateien	72
3.4.2	Abbildung der konzeptuellen Ebene auf interne Strukturen	74
3.4.3	Einpassen von Datensätzen auf Blöcke	75
3.4.4	Modell des Sekundärspeichers	77
3.5	Seiten, Sätze und Adressierung	78
3.5.1	Struktur der Seiten	78
3.5.2	Satztypen	79
3.5.3	Adressierung von Datensätzen	85
3.5.4	Alternative Speichermodelle und Kompression	87
3.6	Speicherorganisation und physische Datendefinition in SQL-Systemen	89
3.7	Vertiefende Literatur	93
3.8	Übungen	94
4	Pufferverwaltung	97
4.1	Einordnung und Motivation	98
4.2	Suche von Seiten und Speicherzuteilung	101
4.2.1	Suchen einer Seite	101
4.2.2	Speicherzuteilung im Puffer	102
4.3	Seitenersetzungsstrategien	102
4.3.1	Merkmale gängiger Strategien	105
4.3.2	Konkrete Seitenersetzungsstrategien	106
4.3.3	Fazit	118
4.4	Vertiefende Literatur	120

4.5	Übungen	120
5	Dateiorganisation und Zugriffsstrukturen	123
5.1	Klassifikation der Speichertechniken	124
5.1.1	Primärschlüssel vs. Sekundärschlüssel	125
5.1.2	Primärindex vs. Sekundärindex	126
5.1.3	Dateiorganisationsform vs. Zugriffspfad	127
5.1.4	Dünn besetzter vs. dicht besetzter Index	129
5.1.5	Geclusterter vs. nicht-geclusterter Index	131
5.1.6	Schlüsselzugriff vs. Schlüsseltransformation	132
5.1.7	Ein-Attribut- vs. Mehr-Attribut-Index	133
5.1.8	Eindimensionale vs. mehrdimensionale Zugriffsstruktur	133
5.1.9	Nachbarschaftserhaltende vs. streuende Zugriffsstruktur	134
5.1.10	Statische vs. dynamische Zugriffsstruktur	135
5.1.11	Beispiele für Klassifikationen	136
5.1.12	Alternative Klassifikationen von Zugriffsverfahren . . .	137
5.1.13	Anforderungen an Speichertechniken	139
5.2	Sequenzielle und indexierte Dateien	140
5.2.1	Heap-Organisation	140
5.2.2	Sequenzielle Speicherung	144
5.2.3	Indexsequenzielle Dateiorganisation	145
5.2.4	Indexiert-nichtsequenzieller Zugriffspfad	150
5.3	Suchbäume	154
5.3.1	B-Bäume	155
5.3.2	B-Bäume und Varianten in Datenbanken	163
5.3.3	B-Bäume in der Praxis	170
5.4	Hashverfahren	173
5.4.1	Grundprinzipien von Hashverfahren	173
5.4.2	Hashverfahren für Datenbanken	175
5.5	Cluster-Bildung	179
5.5.1	Index-organisierte Tabellen	180
5.5.2	Cluster für Verbundanfragen	181
5.6	Partitionierung	184
5.6.1	Fragmentierung und Allokation in verteilten Datenban- ken	185
5.6.2	Formen der horizontalen Partitionierung	187
5.6.3	Bereichspartitionierung	188
5.6.4	Hash-Partitionierung	189
5.7	Vertiefende Literatur	190
5.8	Übungen	191

6	Spezielle Indexstrukturen	193
6.1	Dynamisches Hashing	194
6.1.1	Hashfunktionen mit erweiterbarem Bildbereich	194
6.1.2	Lineares Hashen	195
6.1.3	Erweiterbares Hashen	198
6.1.4	Spiralhashen	201
6.1.5	Kombinierte Methoden	204
6.2	Mehrdimensionale Speichertechniken	205
6.2.1	Mehrdimensionale Baumverfahren	206
6.2.2	Mehrdimensionales Hashen	212
6.2.3	Grid-File	216
6.2.4	UB-Baum	222
6.3	Geometrische Zugriffsstrukturen	225
6.3.1	Probleme und Aufgaben	225
6.3.2	Eignung klassischer Suchbäume und Indexstrukturen	228
6.3.3	Prinzipien nachbarschaftserhaltender Suchbäume	229
6.3.4	R-Bäume und Varianten	233
6.3.5	Rechteckspeicherung durch Punktdatenstrukturen	239
6.3.6	Klassifizierung und Vergleich	246
6.4	Hochdimensionale Daten	247
6.4.1	Hochdimensionale Feature-Vektoren	247
6.4.2	Operationen auf Feature-Vektoren	248
6.4.3	Metriken für Abstände	250
6.4.4	Nächster-Nachbar-Suche in R-Bäumen	252
6.4.5	Der X-Baum	255
6.4.6	Alternativen zu Baumverfahren	257
6.5	Bitmap-Indexe	259
6.5.1	Vor- und Nachteile von Bitmap-Indexen	260
6.5.2	Varianten von Bitmap-Indexen	261
6.5.3	Implementierung von Bitmap-Indexen	262
6.6	Indexierung von Texten	263
6.6.1	Eignung von B-Bäumen: Probleme und Präfix-B-Baum	263
6.6.2	Digitale Bäume	264
6.6.3	Invertierte Listen	269
6.7	Relationenübergreifende Indexe	270
6.7.1	Verbundindexe	270
6.7.2	Multi-Join-Indexe	272
6.7.3	Pfadindexe	273
6.7.4	Zugriffsunterstützungsrelationen	275
6.7.5	Zugriffspfade für berechnete Werte	276
6.8	Vertiefende Literatur	277
6.9	Übungen	278

II Anfragebearbeitung

281

7	Basisalgorithmen für Datenbankoperationen	283
7.1	Benötigte Grundalgorithmen	285
7.1.1	Parameter für Kostenbestimmung	285
7.1.2	Grundannahmen	286
7.1.3	Hauptspeicheralgorithmen	287
7.1.4	Zugriffe auf Datensätze	287
7.1.5	Externe und interne Sortieralgorithmen	288
7.2	Navigationsoperationen: Scans	292
7.2.1	Arten von Scans	292
7.2.2	Operationen auf Scans	293
7.2.3	Scan-Semantik	296
7.3	Unäre Operationen: Selektion, Projektion und Gruppierung	297
7.3.1	Selektion	297
7.3.2	Projektion	299
7.3.3	Aggregatfunktionen und Gruppierung	300
7.4	Binäre Operationen: Mengenoperationen	303
7.4.1	Techniken für binäre Operatoren	304
7.4.2	Klassen binärer Operatoren	305
7.4.3	Vereinigung mit Duplikateliminierung	306
7.5	Berechnung von Verbunden	308
7.5.1	Nested-Loops-Verbund	308
7.5.2	Merge-Techniken	310
7.5.3	Hashverbund	312
7.5.4	Vergleich der Techniken	316
7.6	Operationen für spezielle Anwendungen	318
7.6.1	Cube-Berechnung	319
7.6.2	Skyline-Operator	325
7.7	Vertiefende Literatur	331
7.8	Übungen	332
8	Optimierung von Anfragen	333
8.1	Grundprinzipien der Optimierung	335
8.2	Motivierende Beispiele	336
8.3	Phasen der Anfragebearbeitung	341
8.4	Anfrageübersetzung und -vereinfachung	343
8.4.1	Parsen und Analysieren der Anfrage	343
8.4.2	Übersetzung in Relationenalgebra	345
8.4.3	Auflösung von Sichten	346
8.4.4	Standardisierung und Vereinfachung von Ausdrücken	347
8.4.5	Entschachteln von Anfragen	349
8.5	Weitere Phasen der Optimierung	353

8.6	Vertiefende Literatur	354
8.7	Übungen	355
9	Logische Optimierung	357
9.1	Algebraische Optimierung	358
9.1.1	Entfernen redundanter Operationen	359
9.1.2	Änderung der Reihenfolge von Operationen	360
9.1.3	Optimierungsregeln	361
9.1.4	Ein einfacher Optimierungsalgorithmus	364
9.1.5	Vorgruppierungen	367
9.1.6	Erkennung gemeinsamer Teilanfragen	369
9.1.7	Ergebnis der algebraischen Optimierung	370
9.2	Verbundoptimierung mit Tableaus	370
9.2.1	Tableaus – Eine informale Einführung	371
9.2.2	Formale Definition einer Tableau-Anfrage	373
9.2.3	Konstruktion einer Tableau-Anfrage	376
9.2.4	Äquivalenz von Tableau-Anfragen	379
9.2.5	Minimalität	380
9.2.6	Optimierung von Tableau-Anfragen	381
9.2.7	Erweiterung der Tableau-Optimierung	384
9.3	Semantische Optimierung	385
9.3.1	Darstellungsvarianten für Anfragen	386
9.3.2	Berücksichtigung von Integritätsbedingungen	387
9.3.3	Äquivalenz von Anfragen unter Integritätsbedingungen	390
9.3.4	Tableau-Optimierung mit CHASE	391
9.4	Vertiefende Literatur	395
9.5	Übungen	396
10	Interne Optimierung und kostenbasierte Planauswahl	399
10.1	Physische oder interne Optimierung	400
10.1.1	Planoperatoren und Planrepräsentation	401
10.1.2	Plangenerierung und Suchstrategien	412
10.2	Kostenmodelle und Kostenabschätzung	416
10.2.1	Komponenten von Kostenmodellen	416
10.2.2	Histogramme	422
10.2.3	Kostenabschätzungen am Beispiel	429
10.2.4	Statistiken in DBMS	433
10.3	Strategien zur kostenbasierten Planauswahl	435
10.3.1	Greedy-Suche	436
10.3.2	Dynamische Programmierung	438
10.3.3	Anfragedekomposition	442
10.3.4	Iterative Improvement und Simulated Annealing	445
10.3.5	Optimierung mit genetischen Algorithmen	448

10.4	Beeinflussung von Anfrageoptimierern	450
10.4.1	Ausgabe von Plänen	451
10.4.2	Optimizer Hints	454
10.5	Vertiefende Literatur	457
10.6	Übungen	458

III Transaktionsverarbeitung und Recovery 461

11	Transaktionsmodelle	463
11.1	Transaktionen im Mehrbenutzerbetrieb	463
11.2	Transaktionseigenschaften	465
11.3	Probleme im Mehrbenutzerbetrieb	467
11.3.1	Inkonsistentes Lesen: Nonrepeatable Read	468
11.3.2	Lesen inkonsistenter Zustände	469
11.3.3	Abhängigkeiten von nicht freigegebenen Daten: Dirty Read	469
11.3.4	Das Phantom-Problem	471
11.3.5	Verloren gegangene Änderungen: Lost Update	471
11.3.6	Integritätsverletzung durch Mehrbenutzer-Anomalie	472
11.3.7	Cursor-Referenzen	473
11.3.8	Problemklassifikation	474
11.3.9	Isolation: Serialisierbarkeit oder Snapshot Isolation	475
11.4	Serialisierbarkeit	476
11.4.1	Einführung in die Serialisierbarkeitsthematik	476
11.4.2	Der Begriff des Schedules	481
11.4.3	Grundlegende Definitionen	484
11.4.4	Das Konzept der Serialisierbarkeit	485
11.4.5	Sichtserialisierbarkeit	486
11.4.6	Konfliktserialisierbarkeit	489
11.4.7	Graphbasierter Test auf Konfliktserialisierbarkeit	491
11.4.8	Abgeschlossenheitseigenschaften	493
11.5	Transaktionsabbruch und Fehlersicherheit	496
11.5.1	Rücksetzbarkeit	497
11.5.2	Vermeidung kaskadierender Abbrüche	498
11.5.3	Striktheit	498
11.5.4	Rigorese Striktheit oder Strenge	500
11.5.5	Operationen für den Transaktionsabbruch	502
11.6	Mehrversionen-Serialisierbarkeit	504
11.6.1	Idee des MVCC	504
11.6.2	Ein- und Mehrversionen-Schedules	505
11.6.3	Serialisierbarkeitsgraph für MV-Schedules	508
11.6.4	Serielle und serialisierbare MV-Schedules	508

11.6.5	Mehrversionen-Serialisierbarkeitsgraph	510
11.6.6	MVCC in DBMS	513
11.7	Snapshot Isolation	514
11.7.1	Definition der Snapshot Isolation	515
11.7.2	Vergleich zur Serialisierbarkeit	516
11.7.3	Serialisierbare Snapshot Isolation	519
11.8	Ausnutzung semantischer Informationen	520
11.8.1	Vertauschbarkeit von Operationen	520
11.8.2	Kompensierende Aktionen	523
11.9	Vertiefende Literatur	525
11.10	Übungen	526
12	Transaktionsverwaltung	529
12.1	Der Scheduler	529
12.2	Sperrmodelle	532
12.2.1	Sperrdisziplin	532
12.2.2	Verklemmungen	533
12.2.3	Livelock-Problem	534
12.3	Sperrprotokolle	535
12.3.1	Notwendigkeit von Sperrprotokollen	535
12.3.2	Zwei-Phasen-Sperrprotokoll	536
12.3.3	Striktes und strenges Zwei-Phasen-Sperrprotokoll	538
12.3.4	Aggressive und konservative Protokolle	538
12.4	Sperrgranulate	539
12.4.1	Hierarchisches Sperren	540
12.4.2	Prädikatsperren	544
12.4.3	Baumprotokolle für Sperren in Indexstrukturen	545
12.5	Nichtsperrende Verfahren zur Synchronisation	549
12.5.1	Zeitmarkenverfahren	550
12.5.2	Serialisierbarkeitsgraphentester	553
12.5.3	Optimistische Verfahren	554
12.6	Mehrversionen-Synchronisation	557
12.6.1	Begrenzung der Anzahl der Versionen	557
12.6.2	Synchronisation von MV-Schedules	559
12.7	Commit-Protokolle	564
12.7.1	Verteiltes Commit	564
12.7.2	Das Zwei-Phasen-Commit-Protokoll	565
12.7.3	Lineares 2PC	568
12.7.4	Verteiltes 2PC	570
12.7.5	Hierarchisches 2PC	571
12.7.6	Das Drei-Phasen-Commit-Protokoll	572
12.8	Transaktionen in SQL-DBMS	575
12.8.1	Aufweichung von ACID in SQL-92: Isolationsebenen	575

12.8.2	Explizite Sperren in SQL	577
12.9	Vertiefende Literatur	579
12.10	Übungen	579
13	Wiederherstellung und Datensicherung	581
13.1	Beteiligte Systemkomponenten	582
13.2	Fehlerklassifikation und Recovery-Klassen	584
13.2.1	Fehlerklassifikation	584
13.2.2	Fehlerkategorien und zugehörige Recovery-Maßnahmen	587
13.3	Protokollierungsarten	588
13.3.1	Aufbau des Logbuchs	588
13.3.2	Physisches vs. logisches Logbuch	591
13.3.3	Sicherungspunkte	595
13.4	Recovery-Strategien	599
13.4.1	Seitenersetzungsstrategien	599
13.4.2	Propagierungsstrategien	600
13.4.3	Einbringstrategien	601
13.4.4	Konkrete Recovery-Strategien	602
13.4.5	Wiederanlauf nach einem Fehlerfall	604
13.4.6	Das REDO-Protokoll als konkrete Realisierung	605
13.4.7	Abbrüche im Recovery-Prozess	606
13.5	Das ARIES-Verfahren	607
13.5.1	Vorgehensweise in ARIES	607
13.5.2	Grundprinzipien und Datenstrukturen	608
13.5.3	Phasen des Wiederanlaufs	609
13.6	Schattenspeicherverfahren	611
13.7	Backup-Strategien und Archivierung	613
13.7.1	Backups und Archivierung	614
13.7.2	Spiegelung von Datenbanken	616
13.8	Vertiefende Literatur	616
13.9	Übungen	617
IV	Aktuelle Entwicklungen	619
14	Moderne Datenbanksystem-Architekturen	621
14.1	Alternative Speichermodelle: DSM und PAX	622
14.2	Kompression von Daten	625
14.3	Multicore- und Spezialprozessoren	632
14.3.1	Hashverbunde für Multicore-Systeme	633
14.3.2	GPGPU-Beschleunigung von Datenbankoperationen	635
14.4	Alternative transaktionale Garantien	638
14.4.1	Von ACID zu BASE	639
14.4.2	Das CAP-Theorem	640

14.4.3	Abgeschwächte Konsistenzmodelle	642
14.5	Vertiefende Literatur	644
	Laufendes Beispiel	647
	Abbildungsverzeichnis	651
	Tabellenverzeichnis	660
	Liste der Codefragmente	663
	Sachindex	665
	Literaturverzeichnis	679

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 Relationalalgebra, 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.

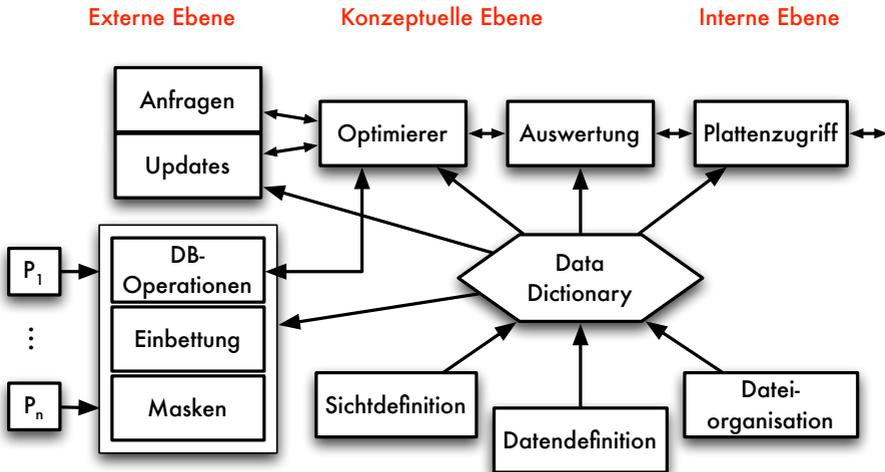


Abbildung 1.1: Vereinfachte Architektur eines DBMS

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

- Die Komponente *Dateiorganisation* beinhaltet die Definition der Dateior-
ganisation 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 Benutzersich-
ten, 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 Anwendungsprogram-
mierung (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.
- $P_1 \dots P_n$ 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.

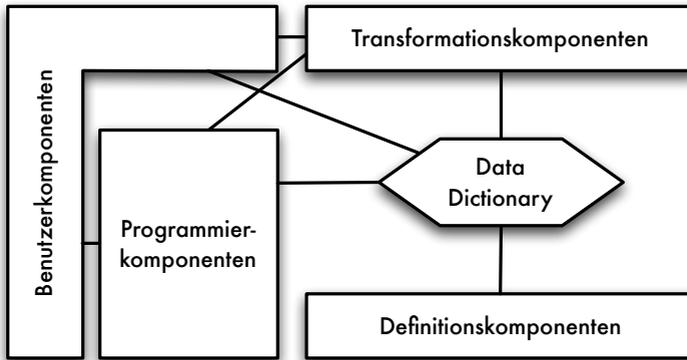


Abbildung 1.2: Klassifikation der Komponenten eines DBMS aus Abbildung 1.1

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 Punkten¹ 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.

¹Wer 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 $R = \{A_1, \dots, A_n\}$ (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 sich 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 $S := \{R_1, \dots, R_p\}$ mit $p \in \mathbb{N}$ heißt *Datenbankschema*. Eine *Datenbank* über einem Datenbankschema S ist eine Menge von Relationen

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

wobei $r_i(R_i)$ für alle $i \in \{1, \dots, p\}$ 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 $K := \{B_1, \dots, B_k\} \subseteq R$, sodass für jede Relation $r(R)$ gilt:

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

Ein *Schlüssel* ist eine bezüglich \subseteq minimale identifizierende Attributmenge. In einem Relationenschema kann es mehrere Schlüssel geben. Ein *Primärschlüssel* ist ein ausgezeichnete 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 $r_2(R_2)$ mit Y Primärschlüssel für r_2 in der Datenbank existiert und Folgendes erfüllt ist:

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

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) )
```