



mitp

Andreas
Heuer

Gunter
Saake

Kai-Uwe
Sattler

Holger
Meyer

Hannes
Grunert

Datenbanken

Kompaktkurs



Inhaltsverzeichnis

Inhaltsverzeichnis	v
Vorwort	xiii
1 Was sind Datenbanken?	1
1.1 Warum Datenbanken?	1
1.2 Datenbanksysteme	3
1.3 Anforderungen: Die Codd'schen Regeln	6
1.4 DBMS-Architektur	8
1.5 Datenunabhängigkeit	10
1.6 Transaktionen	13
1.7 Konkrete Datenbankmanagementsysteme	14
1.8 Einsatzgebiete und Grenzen	15
1.9 Beispielanwendungen	16
1.10 Übersicht über die Kapitel des Buches	16
1.11 Übungsaufgaben	17
2 Relationale Datenbanken – Daten als Tabellen	19
2.1 Relationen für tabellarische Daten	20
2.2 Integritätsbedingungen: Schlüssel und Fremdschlüssel	21
2.3 Tabellendefinition in SQL	23
2.4 Anfrageoperationen auf Tabellen	25
2.4.1 Selektion σ	25
2.4.2 Projektion π	26
2.4.3 Natürlicher Verbund \bowtie	27
2.4.4 Umbenennung β	29
2.4.5 Mengenoperationen $\cup, \cap, -$	30
2.5 Anfragen in SQL	33
2.6 Änderungsoperationen in SQL	35
2.7 Zusammenfassung	36
2.8 Übungsaufgaben	36

3	Das Entity-Relationship- Modell	39
3.1	Eigenschaften von Datenbankmodellen	39
3.2	Einführung in das Entity-Relationship-Modell	40
3.2.1	Grundkonzepte	40
3.2.2	Ein einfaches Beispiel für ein ER-Schema	41
3.3	ER-Modellierungskonzepte	43
3.3.1	Wert und Attribut	43
3.3.2	Entity und Entity-Typ	44
3.3.3	Beziehung und Beziehungstyp	44
3.3.4	Identifizierung durch Schlüssel	45
3.4	Kardinalitäten	46
3.4.1	n:m-Beziehung	47
3.4.2	1:n-Beziehung	47
3.4.3	1:1-Beziehung	48
3.4.4	Optionale und zwingende Beziehungen	49
3.4.5	Abhängige Entity-Typen	49
3.4.6	Die IST-Beziehung	50
3.4.7	Alternative Notationen für Kardinalitäten	51
3.5	Zusammenfassung	53
3.6	Übungsaufgaben	54
4	Datenbankentwurf	57
4.1	Der Datenbankentwurfsprozess	57
4.1.1	Ziele des Datenbankentwurfs	57
4.1.2	Das Phasenmodell	58
4.2	Formalisierung des Relationenmodells	60
4.3	Logischer Datenbankentwurf: Abbildung auf Relationen	62
4.3.1	Übersicht über die Abbildungsregeln	64
4.3.2	Abbildung von Entity-Typen	64
4.3.3	Abbildung von Beziehungstypen	65
4.3.4	Abbildung eines abhängigen Entity-Typs	70
4.3.5	Abbildung der IST-Beziehung	71
4.3.6	Komplexere Beispiele	72
4.4	Zusammenfassung	75
4.5	Übungsaufgaben	76
5	Normalisierung für eine redundanzfreie Datenbank	79
5.1	Funktionale Abhängigkeiten	80
5.2	Schema-Eigenschaften	81
5.2.1	Update-Anomalien	82
5.2.2	Erste Normalform	83
5.2.3	Zweite Normalform	84
5.2.4	Dritte Normalform	85

5.2.5	Boyce-Codd-Normalform	88
5.2.6	Minimalität	88
5.3	Transformationseigenschaften	89
5.3.1	Abhängigkeitstreue	89
5.3.2	Verbundtreue	91
5.4	Mehrwertige Abhängigkeiten	92
5.4.1	Definition der mehrwertigen Abhängigkeit	93
5.4.2	Vierte Normalform	93
5.5	Zusammenfassung	95
5.6	Übungsaufgaben	96
6	Datendefinition und Updates in SQL	99
6.1	Datendefinition in SQL	99
6.1.1	Die create table-Klausel	100
6.1.2	Die create domain-Klausel	102
6.1.3	Die Klauseln alter und drop	103
6.2	Änderungsoperationen in SQL	103
6.2.1	Die insert-Klausel	104
6.2.2	Die update-Klausel	105
6.2.3	Die delete-Klausel	107
6.3	Zusammenfassung	108
6.4	Übungsaufgaben	108
7	Anfragen in SQL	109
7.1	Kriterien für Anfragesprachen	110
7.2	Der SFW-Block	111
7.3	Auswahl von Tabellen: Die from-Klausel	112
7.3.1	Kartesisches Produkt	112
7.3.2	Verbunde als explizite Operatoren	113
7.4	Projektionsattribute: Die select-Klausel	114
7.5	Selektionsbedingungen: Die where-Klausel	118
7.5.1	Verbundbedingung	118
7.5.2	Schachtelung von Anfragen	119
7.6	Mengenoperationen	120
7.7	Mächtigkeit des SQL-Kerns	123
7.8	Weitere Verbunde in SQL	124
7.8.1	Äußere Verbunde	124
7.8.2	Selbstverbunde	125
7.9	Weitere Selektionen in SQL	126
7.9.1	Bereichsselektion	126
7.9.2	Muster in Zeichenketten: Ungewissheitsselektion	127
7.9.3	Selektionen nach Nullwerten	128
7.9.4	Bedingungen mit Quantoren	129

7.9.5	exists-Prädikat	130
7.10	Skalare Ausdrücke: Funktionen zeilenweise	131
7.11	Aggregatfunktionen und Gruppierung: Funktionen spaltenweise	134
7.11.1	Aggregatfunktionen	134
7.11.2	Die group by- und having-Klausel	137
7.12	Sortierung mit der order by-Klausel	142
7.13	SQL-Standards und ihr Funktionsumfang	143
7.13.1	Teilsprachen von SQL	143
7.13.2	Entwicklung der SQL-Standards	145
7.14	Zusammenfassung	150
7.15	Übungsaufgaben	150
8	Sichten und Datenschutz	153
8.1	Sichten	153
8.1.1	Definition von Sichten	154
8.1.2	Definition von Sichten in SQL	155
8.1.3	Vorteile von Sichten	157
8.1.4	Probleme mit Sichten	158
8.1.5	Effektkonformität bei Updates auf Sichten	161
8.1.6	Behandlung von Sichten in SQL	165
8.2	Rechtevergabe und Zugriffskontrolle	167
8.2.1	Datensicherheit, Datenschutz, Integrität	168
8.2.2	Rechtevergabe in Datenbanksystemen	168
8.2.3	Rechtevergabe in SQL	169
8.2.4	Zurücknahme von Rechten	171
8.2.5	Authentifikation und Autorisierung	172
8.3	Datenschutz	173
8.3.1	Privatsphäre und Datenbanken	173
8.3.2	Statistische Auswertungen	174
8.3.3	Anonymisierung und Pseudonymisierung	176
8.3.4	Quasi-Identifizierer	178
8.3.5	Datensparsame Anfrageverarbeitung	180
8.4	Zusammenfassung	181
8.5	Übungsaufgaben	182
9	Integrität und Trigger	183
9.1	Integritätsbedingungen	184
9.1.1	Integritätsbedingungen in SQL	184
9.1.2	Die check-Klausel	186
9.1.3	Die assertion-Klausel	188
9.2	Trigger	190
9.2.1	Grundprinzipien von Triggern	190
9.2.2	Trigger in SQL	193

9.2.3	Integritätssicherung durch Trigger	194
9.3	Zusammenfassung	196
9.4	Übungsaufgaben	196
10	Arbeitsweise eines DBMS und Tuning	199
10.1	Schichten-Architektur für Datenbanken	199
10.2	Interne Datenstrukturen und Zugriffspfade	204
10.2.1	Organisation von Dateien	204
10.2.2	Dateiorganisationsformen	204
10.2.3	Indexdateien als Zugriffspfade	208
10.2.4	Weitere Unterstützungen des Zugriffs	212
10.2.5	Indexdateien in SQL: create index	216
10.2.6	Pufferverwaltung	217
10.3	Optimierung von Anfragen	217
10.3.1	Motivation der Optimierung	217
10.3.2	Phasen der Anfrageverarbeitung	218
10.3.3	Logische Optimierung	219
10.3.4	Interne Optimierung: Effiziente Basisalgorithmen	221
10.3.5	Kostenbasierte Auswahl	223
10.4	Transaktionen und Mehrbenutzerbetrieb	225
10.4.1	Motivation für Transaktionen	226
10.4.2	Eigenschaften von Transaktionen: Das ACID-Prinzip	227
10.4.3	Isolation im Mehrbenutzerbetrieb	228
10.4.4	Verfahren zur Synchronisation	229
10.4.5	Transaktionen in SQL	232
10.5	Datensicherheit: Recovery und Logbuch	234
10.5.1	Fehlerklassifikation	235
10.5.2	Aufbau des Logbuchs	238
10.5.3	Recovery: UNDO und REDO	239
10.6	Tuning von Datenbanksystemen	240
10.6.1	Tuning von Schemata	240
10.6.2	Tuning der internen Strukturen	241
10.6.3	Tuning von Anfragen und Transaktionen	242
10.6.4	Tuning-Werkzeuge in SQL-Systemen	244
10.7	Zusammenfassung	244
10.8	Übungsaufgaben	245
11	OLTP- und OLAP-Systeme	247
11.1	OLTP-Systeme	248
11.1.1	Anfragen in OLTP-Anwendungen	248
11.1.2	Transaktionen und Updates in OLTP-Anwendungen	249
11.2	OLAP-Systeme	249
11.2.1	Datenanalysen in OLAP-Anwendungen	250

11.2.2	Vergleich OLTP und OLAP	250
11.3	Data Warehouses und OLAP	252
11.3.1	Konzepte und Modelle für OLAP und Data Warehouses	252
11.3.2	Prinzipien von Data Warehouses und OLAP-Systemen	255
11.3.3	Data Warehouses und klassische Datenbanken	258
11.3.4	SQL-Unterstützung für Data Warehouses	261
11.4	Data Mining in Datenbanken	266
11.4.1	Data Mining im Vergleich zu OLAP	267
11.4.2	Mining-Probleme und -Verfahren	268
11.4.3	Data Mining in SQL	272
11.5	Zusammenfassung	273
11.6	Übungsaufgaben	274
12	Row und Column Stores, NoSQL und NewSQL	275
12.1	Einführung in Big Data	276
12.2	Speicherung und Verarbeitung von Tabellen	279
12.2.1	Zeilenorientierte Verarbeitung von Tabellen	279
12.2.2	Spaltenorientierte Analyse von Tabellen	280
12.2.3	Kompression von Spalten	282
12.3	Semistrukturierte Datenbankmodelle	283
12.4	NoSQL-Datenbankmodelle	285
12.4.1	Schlüssel-Wert-Paare	285
12.4.2	Wide Column	286
12.4.3	Dokumenten-Modell und JSON	287
12.5	NewSQL: SQL und JSON	288
12.6	Zusammenfassung	292
12.7	Übungsaufgaben	292
13	Ausblick	293
13.1	Objektorientiertes Datenbankmodell	294
13.2	Objektrelationales Datenbankmodell	296
13.2.1	Typkonstruktoren: row, array und multiset	297
13.2.2	Untertabellen und Untertypen	301
13.3	Multimediale Daten	304
13.3.1	Multimedia in Datenbanken	304
13.3.2	Der SQL/MM-Standard	306
13.3.3	Texte in Datenbanken: SQL/MM Full Text	307
13.3.4	Bilder in Datenbanken: SQL/MM Still Image	309
13.4	Verarbeitung von Big Data	310
13.4.1	Verteilte Datenbanken	310
13.4.2	Parallele Datenbanken	312
13.4.3	Big-Data-Frameworks	316

A Hotel-Beispiel	319
A.1 ER-Modellierung	319
A.2 Relationale Repräsentation	321
B Universitäts-Beispiel	323
B.1 ER-Modellierung	324
B.2 Relationale Repräsentation	326
B.3 Beispiel-Datenbank	328
Literaturverzeichnis	331
Sachindex	335

Vorwort

Das Gebiet der *Datenbanksysteme* gehört zu den klassischen Ausbildungsbereichen der Informatik-Studiengänge. Datenbanksysteme kommen immer dann zum Einsatz, wenn an die Datenhaltung besondere Anforderungen betreffend 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.

Zielgruppe dieses Buches

Durch die zunehmende Verbreitung des Informationsaustausches über das Internet werden Datenbanksysteme aber nicht nur von Informatikern benutzt und weiterentwickelt. Die Bereitstellung von Informationen über das Internet und die Verwaltung dieser Informationen mit Datenbanksystemen werden auch für Schüler, Anwender und andere an dieser Technologie interessierte Personen ein ernst zu nehmendes Thema. Allerdings sind bei der Benutzung von Datenbanksystemen Grundkenntnisse über den Datenbankentwurf und die Datenbanksprachen unerlässlich.

Dies ist ein Buch zum Studium und Selbststudium für Schüler und Studenten insbesondere außerhalb des Fachbereichs Informatik. Natürlich kann es auch von Informatikern als einfache Einführung genutzt werden. In kompakter Form und anhand von zwei durchgängigen Beispielen vermittelt es Grundkenntnisse in den folgenden Fragestellungen:

- Wie entwerfe ich eine Datenbankstruktur für meine Anwendung?
- Wie definiere ich die Datenbankstruktur im Datenbanksystem?
- Wie baue ich den Datenbestand auf? Wie ändere ich den Datenbestand?
- Wie selektiere ich aus dem Datenbestand mithilfe einer Anfragesprache gesuchte Informationen?

- Wie benutze ich die Grundkonstrukte der relationalen Datenbanksprache SQL?
- Wie kann ich die Korrektheit (Integrität) der Daten sichern und die Daten vor unerlaubten Zugriffen schützen?
- Wie kann ich die Effizienz des Systems bei steigenden Datenmengen verbessern?
- Wie kann ich die Daten nicht nur auffinden und selektieren, sondern auch mit statistischen Methoden analysieren?
- Was unterscheidet die klassischen relationalen Datenbanksysteme von den aktuellen NoSQL-Systemen?

Für Details zu jedem dieser Themen sei auf den ersten Band des Biberbuchs [SSH18] von Saake, Sattler und Heuer im gleichen Verlag verwiesen, für Details zu Kapitel 10 auf den zweiten Band des Biberbuchs von Heuer, Saake und Sattler [HSS19].

Was dieses Buch NICHT bietet

Dieses Buch führt in die Prinzipien der Nutzung von Datenbanksystemen ein, bietet aber keinen Ersatz für Systemhandbücher. Das Buch bietet also kein Praxiswissen über die Installation von Datenbanksystemen, die konkrete Nutzung eines Systems, die Abweichungen der Datenbanksysteme vom Sprachstandard SQL, dem Aufruf der Systeme aus Webpräsentationen heraus etc. Für die Nutzung eines Datenbanksystems x, y und z muss man sich neben den Systemhandbüchern auf Spezialbücher verlassen, die es aber weitaus häufiger auf dem Markt gibt als dieses hier vorliegende: die anwendungsgerechte Erläuterung der Prinzipien (insbesondere relationaler) Datenbanksysteme.

Einsatz in Kursen

Das Buch kann auch als Grundlage für einen 20- bis 30-stündigen Kurs dienen. Hat man mehr Zeit als eine Doppelstunde pro Kapitel, also insgesamt mehr Zeit als 30 Stunden, so können Kapitel 4 über den Datenbankentwurf, Kapitel 7 über Anfragen in SQL und das Kapitel 10 über die Arbeitsweise eines Datenbankmanagementsystems auf je zwei Doppelstunden ausgedehnt werden. Alternativ kann man auch bei einem 30-Stunden-Kurs die Kapitel 4, 7 und 10 jeweils in zwei Doppelstunden behandeln und dafür die Kapitel 11 bis 13 weglassen.

An der Universität Rostock haben zwei der Autoren (Holger Meyer und Hannes Grunert) bereits seit vielen Jahren regelmäßig eine Vorlesung *Datenbanken für Anwender* gehalten, die von Naturwissenschaftlern, Ingenieurwissenschaftlern, Wirtschaftswissenschaftlern und Geisteswissenschaftlern gehört wird, stetig wachsende Hörerzahlen aufweist und die mittlerweile

auch in einigen Studiengängen wie Biologie und Umweltingenieurwesen zum Standard-Kanon gehört. Die Vorlesung läuft in 14 Wochen mit 2 Semesterwochenstunden (SWS) Vorlesung (sie ist also auf 28 Stunden ausgelegt). Dazu gibt es insgesamt 14 Stunden Übungen und ein 14-stündiges Datenbankpraktikum. Im Datenbankpraktikum entwickeln kleine Arbeitsgruppen mit bis zu fünf Studierenden jeweils eine Datenbankanwendung, inklusive Datenbankentwurf, Datendefinition in SQL, Füllen der Datenbank mit Daten und das Stellen von Anfragen bzw. das Durchführen von Auswertungen auf der Datenbank. Viele Studierenden-Teams bringen dazu auch Problemstellungen aus ihrem eigenen Studienfach mit: So wurden schon biologische Artendatenbanken, Gendatenbanken, Datenbanken für Messreihen in Laborversuchen oder von Messfahrten in der Ostsee entworfen, in einem Datenbanksystem umgesetzt und für Auswertungen genutzt.

Insbesondere werden im Praktikum Hinweise gegeben, wie freie Datenbanksysteme wie MySQL oder PostgreSQL oder (für Zwecke der Lehre) freie Versionen kommerzieller Datenbanksysteme wie IBM Db2 und Oracle bezogen, installiert und benutzt werden können. Drei dieser Systeme (außer Oracle) können die Studierenden dann auch mit den vorgefertigten Beispiel-Datenbanken dieses Buches zum Nachvollziehen der Vorlesungs- und Übungs-Beispiele über ein Web-Portal nutzen.

Zum Cover

Dieses Buch ist eine auf Nicht-Informatiker zugeschnittene, extrem kompakte Version der für Bachelor-Studenten von Informatik-Studiengängen geeigneten Datenbankehrbücher *Datenbanken – Konzepte und Sprachen* [SSH18] und *Datenbanken – Implementierungstechniken* [HSS19] von Gunter Saake, Kai-Uwe Sattler und Andreas Heuer. Das Symbolbild auf dieser Buchreihe ist schon seit 25 Jahren der Biber. Die Bücher werden deshalb auch kurz als Biber-1- und Biber-2-Buch bezeichnet. Da dieses Buch *Datenbanken – Kompaktkurs* auch vom Umfang und Detaillierungsgrad her als Auftakt dieser Reihe angesehen werden kann, bezeichnen die Autoren es auch als das Biber-0-Buch. Auf dem Cover ist *Biber 0* dadurch gekennzeichnet, dass zwei Biber in unterschiedliche Richtungen schauen – und sich somit wie bei +1 und –1 gegenseitig aufheben. Nun ist das Buch natürlich keine Null, sondern in der Informatikerzählweise das Bit 0 und damit der Anfang von allem (Wissen über Datenbanken).

Der Vorgänger

Dieses Buch *Datenbanken – Kompaktkurs* liegt nun in der ersten Auflage vor. Es gab aber bereits einen – mittlerweile lange vergriffenen – Vorgänger, der in den Jahren 2001 bis 2003 in zwei Auflagen erschienen ist: *Datenbanken kompakt* (Abbildung 1), ebenfalls vom selben Verlag. Da dieses Buch aber komplett aktualisiert, vom Inhalt her um mehrere veraltete Kapitel erleichtert, gleichzeitig

aber um mehrere aktuelle Kapitel ergänzt wurde, haben sich die Autoren entschlossen, dieses Buch nicht als 3. Auflage des alten Buches anzusehen. Auch die Zielgruppe war beim Biber-Kompakt-Buch noch anders: Es war vor allem auf Bachelor-Studiengänge der Informatik an Fachhochschulen ausgerichtet.



Abbildung 1: Der knapp 20 Jahre alte Vorgänger dieses Buches

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 *Person* oder *Nachname* geschrieben.
- Platzhalter für konkrete Namen wie Attribut- oder Relationennamen, die oft in Syntaxdefinitionen auftauchen, werden wie in *Attributname* oder *Bedingungsname* notiert.
- 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 *Relationenschema* oder *natürlicher Verbund* 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 zwei Datenbanken, die im Anhang noch einmal

unter „Laufende Beispiele“ (Anhänge A und B) aufgeführt sind. Diese Beispiele beschreiben kleine Anwendungen in einer Universität und für den Betrieb eines Hotels. 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.

Die Strukturierung des Buches erfolgt in nummerierte Kapitel (etwa 2), Abschnitte (etwa 3.4) und Unterabschnitte (etwa 4.1.2). Paragraphen und Absätze haben eine Zwischenüberschrift, tragen aber keine Nummer.

Danksagungen

Zu danken haben wir einer großen Anzahl von Personen, die direkt oder indirekt zum Entstehen dieses Buches beigetragen haben. Insbesondere aber geht unser Dank an die jeweiligen Mitarbeiter der Lehrstühle für Datenbank- und Informationssysteme an den Universitäten Ilmenau, Magdeburg und Rostock sowie an die Studierenden der Vorlesung *Datenbanken für Anwender* an der Universität Rostock. Weiterhin danken wir Lisa Kresse vom mitp-Verlag für das Lektorat und insbesondere ihre Geduld: Das Erscheinen des Buches war eigentlich bereits vor Beginn der Corona-Krise geplant und wurde durch Letztere dann über Monate verzögert. Bedanken möchten wir uns auch bei Petra Heubach-Erdmann (und ihrem Familien-Team inklusive mitkorrigierender Katze) für das sehr hilfreiche Sprachkorrektorat.

Weitere Informationen

Ergänzende Informationen zum Buch, wie Verweise zu begleitenden Vorlesungsmaterialien, gegebenenfalls erforderliche Fehlerkorrekturen und alle virtuellen Teile, die aus Platzgründen nicht mehr in dieses Buch passten (Praktikumsaufgaben, Abbildungs- und Tabellenverzeichnis, Glossar), sind im Web unter folgender Adresse zu finden:

<http://www.biberbuch.de>

Weiterhin stehen dort auch für alle am Ende der einzelnen Kapitel aufgeführten Übungsaufgaben Musterlösungen zum Download bereit. Sollte die eigene Lösung von der bereitgestellten Musterlösung abweichen, sind die Autoren gern bereit, hierzu Fragen zu beantworten. Kontaktadressen der Autoren finden sich auch unter der obigen Web-Adresse.

Informationen und Downloads zur Buchreihe gibt es auch auf der Verlagsseite:

<http://www.mitp.de/782>

Ilmenau, Magdeburg und Rostock, im August 2020

Andreas Heuer, Gunter Saake, Kai-Uwe Sattler,
Holger Meyer und Hannes Grunert

Was sind Datenbanken?

Der Leser dieses Buches hat sicher bereits eine Vorstellung davon, was man unter Datenbanken zu verstehen hat. Allerdings wird dieser Begriff im Alltagsdeutsch sehr großzügig benutzt, sodass wir doch mit einer etwas genaueren Charakterisierung dieses Begriffes beginnen wollen.

1.1 Warum Datenbanken?

Daten ist die Pluralform von *Datum*, lateinisch für das Geschriebene, das Gegebene, Wert, Tatsache, Information. Eine Datenbank ist nun eine Bank für Daten, also im übertragenen Sinne ein sicherer Aufbewahrungsort für (wertvolle) Informationen. Diese übertragene Bedeutung einer Art Geldinstitut für Daten gibt uns bereits einige Charakteristika von Datenbanken:

- Eine Datenbank hat die (langfristige) *Aufbewahrung* von Daten als Aufgabe.
- Die *Sicherheit vor Verlusten* ist eine Hauptmotivation, etwas „auf die Bank zu bringen“.
- Eine Bank bietet *Dienstleistungen für mehrere Kunden* an, um effizient arbeiten zu können.

Eine Bank kostet Gebühren, auch eine Datenbank verursacht Kosten, also bringt man natürlich nur Daten auf die Bank, für die sich das tatsächlich lohnt. In einer Datenbank bewahren wir daher operative, sozusagen für einen Betrieb „lebenswichtige“ Daten, sehr große Datenbestände, langfristig vorzuhaltende und von vielen gleichzeitig zu nutzende Daten auf.

Betrachten wir als typisches Beispiel die Daten einer Hotelkette. In einer Datenbank aus obigen Gründen zu speichernde Informationen sind hier Reservierungen von Kunden und Preise sowie Verfügbarkeiten von Zimmern, sowie alle Angaben für die Rechnungslegung. Hingegen würden die Geburtstage und Hochzeitstage der Geschäftsführung nicht unbedingt in einer Datenbank aufzubewahren sein.

Für ein Hotel ist es wichtig, die Rechnungsdaten längere Zeit korrekt zu speichern (Regressansprüche von Gästen, Jahresumsatz), sowie verschiedene Arten von Software auf diese Daten zugreifen zu lassen (Rechnungserstellung, Umsatzberechnung, Vorratshaltung der Hotelküche, Datenaustausch zu Buchungsportalen, Erstellung von Gästeprofilen etc.).

Eine Datenbank hat nun die Aufgabe, die Daten in strukturierter Form zur Verwendung durch mehr als ein Software-System zu speichern. Die obigen Beispiele zeigen unterschiedlichen Zugriffsbedarf: In einem Fall werden die Daten eines Gastes analysiert, im Fall der Vorratshaltung die Nachbestellung von Lebensmitteln ausgelöst. Es ist daher sinnvoll, die Daten in einem Format zu speichern, das für viele Zugriffsarten offen ist. Bei Datenbanken ist insbesondere die Speicherung in *Tabellenform in SQL-Datenbanken* verbreitet.

Die Abbildung von Datenbeständen in eine Tabellenform wird uns in diesem Buch intensiv beschäftigen. Die oben genannten Daten liegen durchaus teilweise bereits in Tabellenform vor. Auf einer Rechnung werden aber auch verschiedene Informationen gemischt, die man in einer Datenbank trennen sollte. Auch macht es Sinn, bestimmte Daten (etwa die Adresse des Gastes oder die Informationen zum Zimmertyp) nur einmal abzuspeichern, um Tippfehler bei der erneuten Eingabe von vornherein zu verhindern. Für unser Beispiel einer Hotelrechnung bedeutet dies, dass wir mehrere Tabellen zur Speicherung nutzen sollten:

- Eine Gästetabelle speichert relevante Informationen des Gastes (Namen, Adresse etc.).
- Analog gibt es Tabellen mit Informationen zu Zimmertypen, Zimmern und – im Falle einer Hotelkette – zu den einzelnen Hotels.
- Weitere Tabellen können Angaben zu reservierten sowie belegten (und damit in Rechnung zu stellenden) Zimmern verwalten.
- Eine weitere Tabelle enthält die Daten einzelner Rechnungen.

Abbildung 1.1 zeigt die Daten des Rechnungsausschnitts in einer derartigen Tabellenform.

Die Einträge in den verschiedenen Tabellen werden über die identifizierenden Nummern miteinander verknüpft. Die Speicherung in Tabellen erscheint bei den ersten Einträgen nicht besonders sinnvoll, da es sich ja jeweils nur um eine einzelne Zeile handelt – wir dürfen aber nicht vergessen, dass wir in diesen

Tabellen die Daten *aller* Gäste und *aller* Rechnungen über mehrere Jahrzehnte speichern wollen.

Datenbanksysteme sind nun Software-Systeme, die derartig strukturierte Datenbestände verwalten. Die folgenden Abschnitte dieses Kapitels sollen diese Systeme, die Anforderungen und deren Eigenschaften genauer charakterisieren.

Gast	KNr	Vorname	Name	
	103	Lilo	Pause	
Rechnung	RNr	Datum	KNr	Status
	1014	03.07.2020	103	beendet
Zimmer	ZNr	Zimmertyp	Preis	
	201	Einzelzimmer	99.00	
	202	Doppelzimmer	129.00	
	203	Suite	199.00	
Übernachtungen	RNr	ZNr	Anzahl_Nächte	
	1014	202	4	
	1014	203	1	

Abbildung 1.1: Rechnungsdaten in Tabellenform

1.2 Datenbanksysteme

Die Rolle der Datenbanksysteme ist eine sehr elementare: Sie sollen andere Softwaresysteme wie Buchhaltungssysteme, Rechnungslegungssysteme, Zimmerverwaltungssysteme, Schnittstellen zu Buchungsportalen und weitere Anwendungssoftware mit standardisierten Informationen unterstützen.

Das Problem der Datenredundanz

Ohne den Einsatz von Datenbanksystemen tritt dabei das Problem der *Datenredundanz* auf. Die Anwendungssoftware verwaltet in diesem Szenario jeweils ihre eigenen Daten in ihren eigenen Dateien, jeweils in eigenen speziellen Formaten. Ein typisches Szenario in unseren Hotels gibt die folgende Auflistung wieder:

- Die Buchhaltung speichert Gäste-, Adressinformationen sowie Reservierungen und Belegungen.
- In der Vorratshaltung der Küche werden Reservierungs- und Belegungszahlen benötigt.
- Das Marketing braucht die Zahlen der Vorreservierungen, außerdem zur Direktwerbung die Gäste- und Adressinformationen und die Gästeprofile.

In diesem Szenario sind die Daten *redundant*, also mehrfach gespeichert. Etwa werden die Gästeadressen und die Reservierungen/Belegungen von mehreren Anwendungen in jeweils eigenen Dateien verwaltet. Die entstehenden Probleme sind Verschwendung von Speicherplatz und „Vergessen“ von lokalen Änderungen, die typisch für das Fehlen einer zentralen, „genormten“ Datenhaltung sind. Ein Ziel der Entwicklung von Datenbanksystemen ist die Beseitigung der Datenredundanz. Die Situation wird in Abbildung 1.2 verdeutlicht.

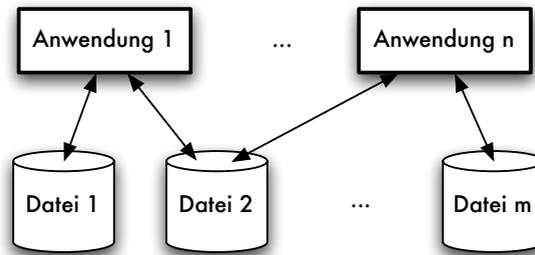


Abbildung 1.2: Zugriff auf Dateien ohne spezielle Verwaltung

Auch in der Anwendungserstellung führt der fehlende Einsatz einer zentralen Datenhaltungskomponente zu erheblichen Defiziten. Die Anwendungsprogrammierer oder auch Endanwender können Anwendungen nicht programmieren bzw. benutzen, ohne

- die interne Darstellung der Daten sowie
- Speichermedien oder Rechner (bei verteilten Systemen)

zu kennen. Dieses Problem wird als fehlende *Datenunabhängigkeit* bezeichnet und in Abschnitt 1.5 noch intensiver diskutiert. Auch ist die Sicherstellung der *Zugriffskontrolle* und der *Datensicherheit* ohne zentrale Datenhaltung nicht gewährleistet.

Vermeidung der Datenredundanz durch Datenbanksysteme

Die obigen Probleme können mit der Datenbanktechnologie gelöst werden. Wir sprechen dann im Gegensatz zur Datenredundanz von einer *Datenintegration*. Das Prinzip der Datenintegration basiert auf folgenden Überlegungen:

Die gesamte Anwendungssoftware arbeitet auf denselben Daten, die in einer zentralen Datenhaltungskomponente verwaltet werden. Der Gesamtbestand der Daten wird nun als *Datenbank* bezeichnet.

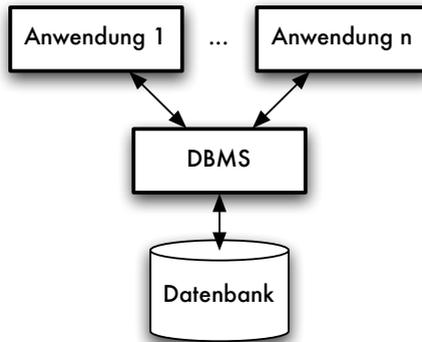


Abbildung 1.3: Datenbankmanagementsysteme

Diese Architekturvorstellung wird in Abbildung 1.3 grafisch verdeutlicht. Eine derartige Datenbank muss natürlich äußerst sorgfältig entworfen und in einer geeigneten Datendefinitionssprache beschrieben werden. Auch andere Probleme im Umgang mit großen Datenbeständen, etwa Fragestellungen der Effizienz, Parallelität, Zugriffskontrolle und Datensicherheit, können mit heutigen kommerziellen Datenbankmanagementsystemen (DBMS) zufriedenstellend gelöst werden. Diese Systeme zeichnen sich durch folgende Eigenschaften aus:

- Datenbanksysteme können große Datenmengen effizient verwalten. Sie bieten benutzergerechte *Anfragesprachen* an, die eine komfortable Anfrageformulierung ohne Rücksichtnahme auf die interne Realisierung der Datenspeicherung ermöglichen. Eine interne *Optimierung* ermöglicht trotzdem einen effizienten Zugriff auf die Datenbestände.
- Viele Benutzer können parallel auf Datenbanken arbeiten. Das *Transaktionskonzept* verhindert hier unerwünschte Nebeneffekte beim Zugriff auf gemeinsam genutzte Daten.
- Die *Datenunabhängigkeit* wird durch ein Drei-Ebenen-Konzept gewährleistet, das eine externe Ebene der Anwendungssicht, eine konzeptuelle Ebene der logischen Gesamtsicht auf den Datenbestand und eine interne Ebene der implementierten Datenstrukturen unterscheidet.
- Zugriffskontrolle (kein unbefugter Zugriff) und Datensicherheit (kein – ungewollter – Datenverlust) werden vom System gewährleistet.

1.3 Anforderungen: Die Codd'schen Regeln

Im Laufe der Jahre hat sich eine Basisfunktionalität herauskristallisiert, die von einem Datenbankmanagementsystem erwartet wird. Bereits Anfang der 80er Jahre hat Edgar F. („Ted“) Codd diese Anforderungen in einer Liste von neun Punkten zusammengefasst, die noch heute ihre Gültigkeit besitzt [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.

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. Zugriffskontrolle

Aufgabe der Zugriffskontrolle 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 Datenbank-Änderungen 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.

Basierend auf diesen Anforderungen können wir einige grundlegende Begriffsbildungen vornehmen, die uns im Laufe dieses Buches begleiten werden:

Unter dem Begriff *Datenbankmanagementsystem*, kurz DBMS, verstehen wir die Gesamtheit der Software-Module, die die Verwaltung einer Datenbank übernehmen. Ein *Datenbanksystem*, kurz DBS, ist die Kombination eines DBMS mit einer Datenbank¹.

Diese Begriffsbildung ist für das Verständnis der Datenbankkonzepte essenziell und wird in Tabelle 1.1 zusammengefasst.

Kürzel	Begriff	Erläuterung
DB	Datenbank	Strukturierter, von einem DBMS verwalteter Datenbestand
DBMS	Datenbankmanagementsystem	Software zur Verwaltung von Datenbanken
DBS	Datenbanksystem	DBMS plus Datenbank(en)

Tabelle 1.1: Begriffsbildungen für Datenbanksysteme

Grundmerkmale von modernen Datenbanksystemen sind die folgenden, die eine direkte Umsetzung der aufgeführten neun Punkte von Codd darstellen:

- DBMS verwalten *persistente* (langfristig zu haltende) Daten, die einzelne Läufe von Anwendungsprogrammen überstehen sollen.

¹Vereinfachend werden wir im Verlaufe dieses Buches ein Datenbankmanagementsystem auch als Datenbanksystem bezeichnen, wenn es aus dem Kontext ersichtlich ist, dass hier keine konkrete Bindung an eine Datenbank vorliegt.

- Sie haben die Aufgabe, *große* Datenmengen *effizient* zu verwalten.
- DBMS definieren ein *Datenbankmodell*, mit dessen Konzepten alle Daten *einheitlich beschrieben* werden.
- Sie stellen *Operationen und Sprachen* (Datendefinitionssprache, interaktive Anfragesprachen, Datenmanipulationssprachen ...) zur Verfügung. Derartige Sprachen sind *deskriptiv*, verzichten also auf die explizite Angabe von Berechnungsschritten. Die Sprachen sind getrennt von einer Programmiersprache definiert. Für die Speicherung in Tabellenform in relationalen Datenbanken bildet die Sprache *SQL* hierbei den Standard.
- Datenbankmanagementsysteme unterstützen das *Transaktionskonzept* inklusive *Mehrbenutzerkontrolle*: Logisch zusammenhängende Operationen werden zu Transaktionen zusammengefasst, die als atomare (unteilbare) Einheit bearbeitet werden. Auswirkungen von Transaktionen sind langlebig. Transaktionen können parallel durchgeführt werden, wobei sie voneinander isoliert werden.
- Sie unterstützen die Einhaltung des *Datenschutzes*, gewährleisten *Datenintegrität* (Konsistenz) und fördern die *Datensicherheit* durch geeignete Maßnahmen.

1.4 DBMS-Architektur

Abbildung 1.4 zeigt einen Überblick über die prinzipielle Aufteilung eines Datenbankmanagementsystems in Funktionsmodule, 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.

Die in Abbildung 1.4 gezeigten Komponenten können wie folgt kurz charakterisiert werden:

- **Dateiorganisation:** Definition der Dateiorganisation und Zugriffspfade auf der internen Ebene
- **Datendefinition:** Konzeptuelle Datendefinition (konzeptuelles Schema)
- **Sichtdefinition:** Definition von Benutzersichten (externe Ebene)
- **Masken:** Entwurf von Menüs und Masken für die Benutzerinteraktion

- **Einbettung:** Einbettung von Konstrukten der Datenbanksprache in eine Programmiersprache
- **Anfragen/Updates:** Interaktiver Zugriff auf den Datenbestand
- **DB-Operationen:** Datenbank-Operationen (Anfrage, Änderungen)
- **Optimierer:** Optimierung der Datenbankzugriffe
- **Plattenzugriff:** Plattenzugriffssteuerung bzw. Ansteuerung anderer Speichermedien
- **Auswertung:** Auswertung von Anfragen und Änderungen
- **P1 ... Pn:** Verschiedene Datenbank-Anwendungsprogramme
- **Data Dictionary** (oder auch Datenwörterbuch): Zentraler Katalog aller für die Datenhaltung relevanten Informationen

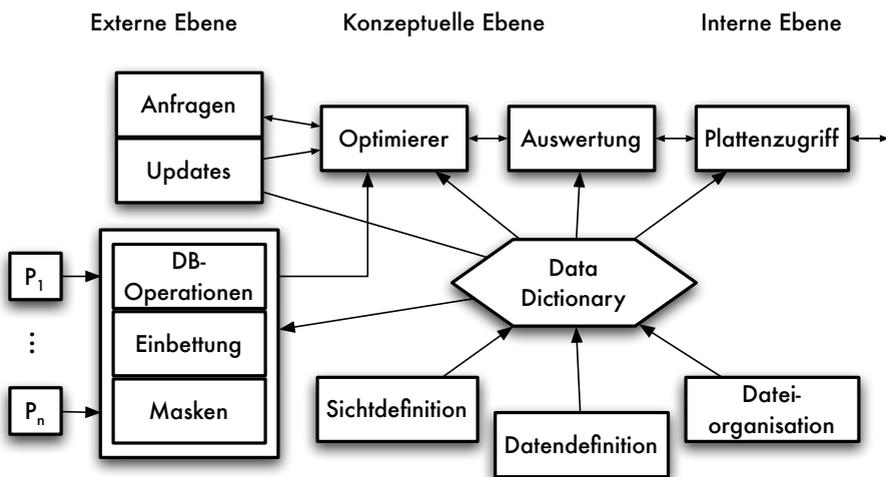


Abbildung 1.4: Vereinfachte Architektur eines DBMS

In den folgenden Abschnitten werden wir einzelne Komponenten kurz erläutern. Hierzu diskutieren wir exemplarisch einige Funktionen, die von einem Datenbankmanagementsystem ausgeführt werden müssen, sowie die zugehörigen datenbankspezifischen Sprachen.

1.5 Datenunabhängigkeit

Ein wesentlicher Aspekt bei Datenbank Anwendungen ist die Unterstützung der *Datenunabhängigkeit* durch das Datenbankmanagementsystem. Sowohl Datenbanken als auch Anwendungssysteme haben in der Regel eine lange Lebensdauer, während derer sowohl die Realisierung der Datenspeicherung als auch externe Schnittstellen aus verschiedensten Gründen modifiziert oder erweitert werden. Das Konzept der Datenunabhängigkeit hat das Ziel, eine (oft langlebige) Datenbank von notwendigen Änderungen der Anwendung abzukoppeln (und umgekehrt). Die Datenunabhängigkeit kann in zwei Aspekte aufgeteilt werden:

- Die *Implementierungsunabhängigkeit* oder auch *physische Datenunabhängigkeit* bedeutet, dass die konzeptuelle Sicht auf einen Datenbestand unabhängig von der für die Speicherung der Daten gewählten Datenstruktur besteht.
- Die *Anwendungsunabhängigkeit* oder auch *logische Datenunabhängigkeit* hingegen koppelt die Datenbank von Änderungen und Erweiterungen der Anwendungsschnittstellen ab.

Zur Unterstützung der Datenunabhängigkeit in Datenbanksystemen wurde bereits in den 70er Jahren von der „ANSI/X3/SPARC² Study Group on Database Management Systems“ eine *Drei-Ebenen-Schema-Architektur* als Ergebnis einer mehrjährigen Studie vorgeschlagen. ist das Kürzel für die amerikanische Standardisierungsbehörde „American National Standards Institute“. Die dort vorgeschlagene Aufteilung in drei Ebenen ist im Datenbankbereich inzwischen allgemein akzeptiert. Abbildung 1.5 zeigt die diesem ANSI-Vorschlag folgende, prinzipielle Schema-Architektur.

Die ANSI-Schema-Architektur teilt ein Datenbankschema in drei aufeinander aufbauende Ebenen auf. Von unten nach oben werden die folgenden Ebenen vorgeschlagen:

- Das *interne Schema* beschreibt die systemspezifische Realisierung der Datenbank, etwa die eingerichteten Zugriffspfade. Die Beschreibung des internen Schemas ist abhängig vom verwendeten Basissystem und der von diesem angebotenen Sprachschnittstelle.
- Das *konzeptuelle Schema* beinhaltet eine implementierungsunabhängige Modellierung der gesamten Datenbank in einem systemunabhängigen Datenmodell, zum Beispiel dem ER-Modell oder dem relationalen Modell. Das konzeptuelle Schema beschreibt die Struktur der Datenbank vollständig.

²SPARC steht für Standards Planning and Requirements Committee.

- Basierend auf dem konzeptuellen Schema können mehrere *externe Schemata* definiert werden, die anwendungsspezifische (Teil-)Sichten auf den gesamten Datenbestand festlegen.

Oft beschreiben externe Sichten einen anwendungsspezifischen Ausschnitt des konzeptuellen Schemas unter Benutzung desselben Datenmodells. Es ist aber auch möglich, unterschiedliche Datenbankmodelle für verschiedene externe Schemata zu verwenden.

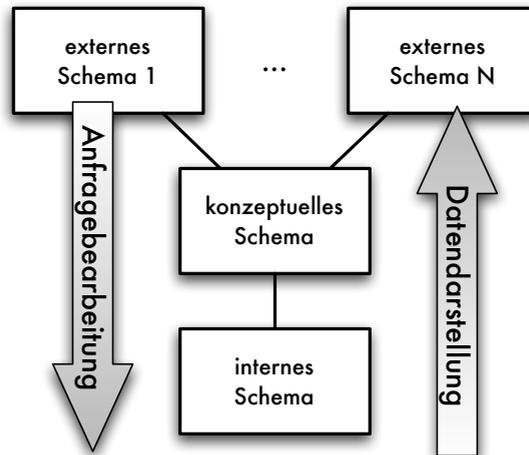


Abbildung 1.5: Drei-Ebenen-Schemaarchitektur für Datenbanken

Die Sprachmittel und typischen Konzepte auf den verschiedenen Ebenen werden im Folgenden exemplarisch anhand einer kleinen Beispielanwendung vorgestellt.

Zwischen den verschiedenen Schemaebenen müssen Abbildungen festgelegt werden, die die Transformation von Datenbankzuständen, Anfragen und Änderungstransaktionen zwischen den Ebenen ermöglichen. Da diese Transformationen vom Datenbankmanagementsystem durchgeführt werden, müssen diese Abbildungen in einer formalen Beschreibungssprache mit festgelegter Semantik notiert werden.

Die Aufgabe der Abbildungen zwischen den Ebenen kann in zwei Problem-bereiche aufgeteilt werden:

- Die *Anfragebearbeitung* erfordert eine Übersetzung von Anfragen und Änderungsoperationen, die bezüglich der externen Schemata formuliert wurden, in Operationen auf den internen Datenstrukturen (über den Zwischenschritt der konzeptuellen Ebene).

- Die *Datendarstellung* erfordert eine Transformation in der umgekehrten Richtung: Die internen Datenstrukturen von Anfrageergebnissen müssen derart transformiert werden, dass sie den Beschreibungskonzepten der externen Darstellungen entsprechen.

Ebenen-Architektur am Beispiel

Wir wollen die Basisidee der Drei-Ebenen-Architektur von Datenbankschemata im Folgenden anhand einer kleinen Beispielmodellierung diskutieren. Die modellierte Datenbank enthält Daten über Produkte und deren Lieferanten. Die konzeptuelle Gesamtsicht ist im relationalen Datenbankmodell beschrieben.

Die konzeptuelle Gesamtsicht

Die konzeptuelle Gesamtsicht erfolgt in relationaler Darstellung. Die Datenbank ist in zwei Relationen gespeichert, wie in Abbildung 1.6 dargestellt.

Zimmer	<u>ZimmerId</u>	Zimmertyp	Preis	HotelId → Hotels.HotelId
	O-201	Einzelzimmer	99.0	901
	O-203	Suite	199.0	901
	M-205	Suite	129.0	902

Hotels	<u>HotelId</u>	HotelName
	901	Ostseeraum
	902	Mecklenburger Hof

Abbildung 1.6: Konzeptuelle Beispieldatenbank in Relationendarstellung

Schlüssel, also identifizierende Attribute in Relationen, werden durch Unterstreichung gekennzeichnet: Die Einträge in diesen Spalten der Tabelle müssen eindeutig sein. Bezüge zwischen Relationen, die sogenannten Fremdschlüssel als Verweise auf Schlüssel, sind in der Beispielrelation mit dem Bezug zu dem Schlüssel einer anderen Relation angegeben.

Externe Sichten

Eine mögliche Anwendungssicht wäre dadurch gegeben, dass die Daten in *einer* Relation dargestellt werden, wobei die `HotelId` ausgeblendet werden soll (wie in Abbildung 1.7 gezeigt).

Diese externe Sicht kann in SQL-Datenbanksystemen einfach durch eine Sicht-Definition (**create view**, vgl. Abschnitt 8.1) definiert werden.

Dieses erste Beispiel definiert eine flache Tabelle als Sicht auf andere flache Tabellen, verlässt also den verwendeten Beschreibungsrahmen im gewissen Sinne nicht. Aber auch Sichten in einem anderen Datenbankmodell sind möglich, etwa als eine hierarchisch aufgebaute Relation wie in Abbildung 1.8.

<u>ZimmerId</u>	Zimmertyp	Preis	HotelName
O-201	Einzelzimmer	99.0	Ostseetraum
O-203	Suite	199.0	Ostseetraum
M-205	Suite	129.0	Mecklenburger Hof

Abbildung 1.7: Externe Sicht auf zwei Relationen, dargestellt als eine Relation

Hotels	Zimmer		
	ZimmerId	Bezeichnung	Preis
Ostseetraum	O-201	Einzelzimmer	99.0
	O-203	Suite	199.0
Mecklenburger Hof	M-205	Suite	129.0

Abbildung 1.8: Externe Sicht als hierarchisch aufgebaute Relation

Diese externe Darstellung ist in den meisten SQL-Datenbanken nicht möglich, entspricht aber der hierarchischen Darstellung von Tabellen, wie sie in vielen Anwendungen üblich ist. Eine derartige Datenrepräsentation ist allerdings in *objektrelationalen Datenbanken* möglich, die Sie später noch kennenlernen werden (Kapitel 13).

Interne Darstellung

Für die interne Darstellung kann ein Datenbankmanagementsystem optimierte Datenstrukturen verwenden, etwa eine Baumstruktur über die ZimmerIDs und eine Hash-Tabelle über die Namen der Hotels. Die Verbindung zwischen den beiden Datenstrukturen könnte dadurch erfolgen, dass die Fremdschlüsseleigenschaft des Attributs `HotelId` in die Definition physischer Zeiger umgewandelt wird. Die interne Speicherung von Relationen wird in Abschnitt 10.2 noch erläutert.

1.6 Transaktionen

Ein wichtiges Merkmal von Datenbanksystemen, das für den Einsatz im Mehrbenutzerbetrieb von großer Bedeutung ist, ist die Unterstützung des Transaktionskonzepts. Unter einer *Transaktion* versteht man eine Folge von Datenbankoperationen, die einen konsistenten Datenbankzustand in einen neuen, wiederum konsistenten Zustand überführen und dabei entweder vollständig oder gar nicht ausgeführt werden. Aus Sicht des Anwendungsentwicklers kann eine Transaktion damit als Einheit zur Konsistenzbewahrung angesehen wer-

den. Ein typisches Beispiel für eine Transaktion ist eine Banküberweisung, die aus zwei Datenbankoperationen besteht: die Abbuchung von einem Konto und die Gutschrift des abgebuchten Betrags zum zweiten Konto. Diese Operationsfolge muss entweder ganz oder gar nicht ausgeführt werden: Sollte zwischen beiden Operationen ein Fehler auftreten oder die zweite Operation nicht durchgeführt werden können, so muss auch die Wirkung der ersten Operation rückgängig gemacht werden.

Trotz Einhaltung der semantischen Integrität durch eine einzelne Transaktion können jedoch fehlerhafte Datenbankzustände eintreten, indem sich parallel laufende Transaktionen im Mehrbenutzerbetrieb gegenseitig beeinflussen. Daher wird gefordert, dass Transaktionen das *ACID-Prinzip* einhalten müssen, auf das wir in Abschnitt 10.4 näher eingehen.

Aus Anwender- und Entwicklersicht manifestiert sich die Unterstützung des Transaktionskonzepts durch ein Datenbanksystem insbesondere in speziellen Kommandos zur Transaktionssteuerung. Diese Kommandos definieren das erfolgreiche Abschließen der Transaktion (das sogenannte *Commit*) sowie den Abbruch und damit das Zurücksetzen der Datenbank in den Ausgangszustand der Transaktion (das *Rollback*).

1.7 Konkrete Datenbankmanagementsysteme

Relationale Datenbanksysteme (RDBS) sind seit den 80er Jahren in einer Vielzahl auf dem Markt präsent. Zu den wichtigsten RDBS zählen Oracle, IBM Db2, Microsoft SQL-Server sowie PostgreSQL und MySQL. Gemeinsame Merkmale dieser Systeme sind

- eine Drei-Ebenen-Architektur nach ANSI-SPARC,
- eine einheitliche Datenbanksprache (SQL; Structured Query Language),
- eine Einbettung dieser Sprache in kommerzielle Programmiersprachen,
- diverse Werkzeuge für die Definition, Anfrage und Darstellung von Daten und den Entwurf von Datenbank-Anwendungsprogrammen und der Benutzer-Interaktion sowie
- kontrollierter Mehrbenutzerbetrieb, Zugriffskontrolle und Datensicherheitsmechanismen.

Daneben gibt es gerade im PC-Bereich noch viele Dateiverwaltungssysteme mit tabellarischer Benutzeroberfläche, oder Pseudo-Datenbanksysteme, die nicht alle Eigenschaften eines Datenbanksystems besitzen. Wir bezeichnen sie deshalb als Pseudo-RDBS, weil sie zwar Verwaltungssysteme für strukturierte

Dateien sind, aber nicht die volle Funktionalität eines Datenbanksystems erreichen. MS Access ist ein Datenbankprogramm für Windows, das über eine moderne Benutzeroberfläche verfügt, mit der man bequem und grafisch Anfragen, Änderungen, Sichten, Berichte und Anwendungsmodule erstellen kann. Die grafischen Operationen werden auf eine SQL-Variante abgebildet, die leider nicht dem Standard genügt. Es fehlen einige Konzepte von Datenbanksystemen wie kontrollierter Mehrbenutzerzugriff (für PC-Datenbanksysteme auch nicht so interessant) und eine getrennt einstellbare interne Ebene. Einen Anfrageoptimierer in unserem Sinne bietet MS Access auch nicht. Allerdings ist aus Access heraus der Zugriff auf „echte“ DBMS möglich, sodass die komfortable Benutzerschnittstelle und Entwicklungsumgebung auch für diese Systeme genutzt werden kann.

Im Open-Source-Umfeld haben sich einige (oben schon erwähnte) ernst zu nehmende Lösungen für Datenbanksysteme entwickelt. Hier sind insbesondere MySQL³ und PostgreSQL⁴ zu nennen. MySQL ist aufgrund der Schnelligkeit und des geringen Installations- und Administrationsaufwands speziell bei Entwicklern von Web-Anwendungen beliebt. Gegenüber den „großen“ Systemen weist MySQL jedoch einige Einschränkungen auf. So werden nicht alle SQL-Konstrukte unterstützt, die in diesem Buch beschrieben werden. PostgreSQL entstand aus dem Forschungsprototyp Postgres der University of Berkeley, indem die dort verwendete Anfragesprache durch SQL ersetzt wurde. Ein besonderes Merkmal dieses Systems ist die Unterstützung objektrelationaler Features, wie sie in SQL:1999 und SQL:2003 Einzug in den SQL-Standard gefunden haben.

1.8 Einsatzgebiete und Grenzen

Die klassischen Einsatzgebiete von Datenbanksystemen sind die Verwaltung vieler Objekte, zum Beispiel 150.000 Hotels, 10.000 Gäste und 300 Buchungsvorgänge pro Tag. Die zu verwaltenden Objekte gehören allerdings zu relativ wenigen Objekttypen (in unserem Beispiel die fünf Objekttypen Hotel, Zimmertyp, Zimmer, Rechnung, Gast). Die Objekte eines Objekttyps sind immer gleich strukturiert, so hat etwa jedes Zimmer unserer Datenbank dieselben zu speichernden Angaben (oder in Tabellen-Sprechweise: dieselben Spalten).

In diesem Buch werden die Objekttypen immer durch einfache Tabellenstrukturen beschrieben, die in relationalen Datenbanksystemen auf Basis der Sprache SQL verwendet werden.

Normalerweise sind herkömmliche Datenbanksysteme überfordert mit sehr heterogenen Objekten, die zu vielen unterschiedlichen Objekttypen gehö-

³<http://www.mysql.com>

⁴<http://www.postgresql.org>

ren, sowie stark strukturierten Objekten. Nur objektorientierte und objektrelationale Datenbanksysteme bieten geeignete Konzepte für stark strukturierte Objekte an (siehe Kapitel 13).

Auch die Verwaltung semistrukturierter Daten bereitet Probleme. Semistrukturiert bedeutet dabei, dass Objekte unterschiedlich strukturierte und auch unstrukturierte Anteile haben können. Multimedia-Datenbanken (siehe auch Kapitel 13), die Texte, Bilder und andere unstrukturierte Informationen verwalten können, eignen sich dafür wie auch NoSQL-Datenbanksysteme (siehe Kapitel 12).

1.9 Beispielanwendungen

In den folgenden Kapiteln werden wir alle Konzepte an zwei einheitlichen Beispielen erläutern. Unsere Beispielszenarien sind dabei

- eine Hotel-Anwendung, in der eine Hotelkette Hotels, Zimmer, Zimmertypen, Gäste, Reservierungen und Buchungen sowie Rechnungen verwalten kann, und
- eine Universitäts-Anwendung, in der Studierende und MitarbeiterInnen der Universität, die Ausleihe von Büchern und Modulprüfungen gespeichert werden.

Wir werden sowohl die konzeptuelle Modellierung im Entity-Relationship-Modell und den Datenbankentwurf, als auch die Umsetzung des Datenbankentwurfs im relationalen Datenbankmodell und die Anfragen an die Datenbank mit SQL an diesen Beispielen demonstrieren.

In den Anhängen A und B sind diese Szenarien im Entity-Relationship-Modell vollständig dargestellt. Weiterhin wird jeweils eine relationale Repräsentation angegeben. Im Universitäts-Beispiel ist diese auch in SQL-Datendefinitionsanweisungen umgesetzt. Weiterhin haben wir für das Universitäts-Beispiel auch vollständige Beispieltabellen angegeben, auf der unsere SQL-Anfragen beispielhaft ausgeführt werden können. An einigen Stellen dieses Buches werden wir diese Anwendungen jedoch gegebenenfalls erweitern, um auf spezielle Probleme einzugehen, die mit diesen einfachen Schemata nicht verdeutlicht werden können.

1.10 Übersicht über die Kapitel des Buches

Dieses Buch gibt eine Einführung in grundlegende Konzepte und Techniken der Datenbanksysteme. Dazu werden zuerst einige wesentliche Prinzipien des rela-

tionalen Datenbankmodells im nachfolgenden Kapitel 2 dargestellt. Anschließend folgt eine Einführung in das Entity-Relationship-Modell, das die Grundlage für den logischen Datenbankentwurf ist (Kapitel 3), und eine Darstellung des Datenbankentwurfsprozesses in Kapitel 4. Um eine redundanzfreie Datenbank zu erreichen, müssen die entworfenen Relationen oft noch weiter normalisiert werden. Der Normalisierungsprozess wird in Kapitel 5 vorgestellt.

Relationale Datenbanksysteme und ihre objektrelationalen Erweiterungen sind heute die Standardtechnik für Datenbankanwendungen. SQL ist hier die dominierende Anfragesprache. Sie enthält Anteile sowohl zur Datendefinition (Kapitel 6), zum Abfragen von Datenbanken (Kapitel 7) sowie Möglichkeiten zur Sichtdefinition und Zugriffskontrolle (Kapitel 8) und zur Vereinbarung von Integritätsbedingungen und Triggern (Kapitel 9).

Einige wesentliche Implementierungsaspekte wollen wir uns in Kapitel 10 anschauen. Da moderne Datenbanksysteme auch komplexe Datenanalysen durchführen können, werden wir uns Themen wie Online-Analysen (OLAP), Data Warehousing und Data Mining in Kapitel 11 anschauen. Gerade für die Nutzung in komplexen Analysen wurden neue Datenbanksystem-Architekturen (column stores: spalten- statt der sonst üblichen zeilenweise Speicherung von Tabellen) und flexiblere Datenbankmodelle in NoSQL-Datenbanksystemen entwickelt. Diese sind Gegenstand von Kapitel 12. Im Ausblick werden wir Lösungen für komplexe Datenbankobjekte, für Multimedia-Daten (wie Bilder) und für die Analyse von Big Data skizzieren (Kapitel 13).

Sollten die Darstellungen dieses Buches einmal nicht ausreichen: Für ein tiefergehendes Verständnis der Konzepte und Sprachen von Datenbanksystemen sei auf den ersten Band des Biberbuches von Saake, Sattler und Heuer verwiesen [SSH18], für eine ausführliche Darstellung der internen Funktionsweise und aller Implementierungsaspekte von Datenbanksystemen auf den dazu passenden zweiten Band [HSS19].

1.11 Übungsaufgaben

Übung 1-1 Nennen Sie grundlegende Eigenschaften eines Datenbanksystems.

Übung 1-2 Erklären Sie die logische und physische Datenunabhängigkeit anhand der 3-Ebenen-Architektur.

Übung 1-3 In welchen Bereichen und für welche Aufgaben sind klassische Datenbanksysteme gut beziehungsweise nicht sinnvoll einsetzbar?

Übung 1-4 Denken Sie an Ihren Alltag: Wo begegnen Ihnen Datenbanksysteme? Sind diese immer auf den ersten Blick zu erkennen oder arbeiten sie im Hintergrund?

Sachindex

Symbole

Integritätssicherung 145
Sichtdefinitionssprache 144
Spezialisierung 296
Transaktionsdefinition 145
2-Phasen-Sperrprotokoll 230
2NF 84
2PL *siehe* 2-Phasen-Sperrprotokoll
3NF 86
4NF 95

A

Abbildung zwischen Schemaebenen
11
Abhängigkeit
funktionale 80
mehrwertige 92, 93
nicht-triviale 95
transitive 85
Abhängigkeitstreue 89
abhängiger Entity-Typ 50
Ablaufintegrität 183, 226
Abort 232, 234, 238, 240
ACID
bis BASE 290
Eigenschaften 227
Prinzip 14, 227, 230, 248, 289
Transaktionen 290, 291
Ähnlichkeitsfunktion 268
Änderungskomponente 33

Änderungsoperationen 103
After image 238, 240
Aggregatfunktion 134, 263
algebraische Optimierung 217
Anforderungsanalyse 58
Anfrage 109
rekursive 190
verzahnt geschachtelte 131
Anfragesprache 33, 144
Anomalie 82
Anonymität
k-Anonymität 179
ANSI 145
ANSI/X3/SPARC 10
Anwendungsunabhängigkeit 10
Apache 318
Archivierung 237, 240
Array-Typkonstruktor 298
Assertion 184
Assoziationsregeln 270
Atomarität 227, 240
Atomicity 227
Attribut 20, 41, 43
Attributselektion 118
Auditing 168
Auswahl
kostenbasierte 219, 224
Authentifikation 172
Auto-Commit-Modus 228
Autorisierungsidentifikator 169

B

B-Baum 208
Backup 240
BASE 289, 291
BASE-Transaktionen 290
Basisrelation 21
Before image 238, 240
Begin-of-Transaction 228
Bereichspartitionierung 313
Bereichsselektion 126
Beziehung
 [-, *] - [-, *] 47
 [-, 1] - [-, *] 47
 IST- 50, 71
 1:1- 48, 65
 1:n- 65
 bidirektionale 296
 mehrstellige 73
 n:m- 65
 optionale 65
 rekursive 72
 Teilnahme- 46
Beziehungsattribut 44
Beziehungstyp 44, 65
Big Data 276, 310
Big Data Analytics 277
Big-Data-Framework 316
Binary large object 101, 305
Binärobjekt 305
Bitmap-Index 211
BLOB *siehe* Binary large object
Boyce-Codd-Normalform 88
built-in-Funktion *siehe*
 Aggregatfunktion

C

Call Level Interface 145
Character large object 101
Checkpoint 238, 240
CLI 145
CLOB *siehe* Character large object
Cluster 212, 219
Cluster-Schlüssel 213

Clustering 268
Code-Erzeugung 219
Column Store 279, 280, 290
Commit 232, 238, 240
Complex Event Processing 272
Consistency 228
Controller-Fehler 236
CRUD 287

D

Datenbankprogrammiersprache 144
Data Base Programming Language
 siehe Datenbankprogrammiersprache
Data Cleaning 255
Data Cube 251, 253
Data Definition Language *siehe*
 Datendefinitionssprache
Data Manipulation Language *siehe*
 Datenmanipulationssprache
Data Mart 256
Data Mining 266
Data Warehouse 211, 251, 252, 282
Dateiorganisation 206
Dateischnittstelle 203
Datenbank 4, 21
 aktive 190
 statistische 174
Datenbankadministrator 143
Datenbankentwurf 57
 Phasenmodell 58
Datenbankmanagementsystem 7
Datenbankschema 21, 40
Datenbanksystem 3, 7
 NoSQL 285
 paralleles 312
 relationales 208
 verteiltes 310
Datendefinition 59, 99
Datendefinitionssprache 23, 143
Datenintegration 4
Datenmanipulationssprache 144
Datenmodell

- multidimensionales 252, 253
- Datenorganisation
 - spaltenorientierte 280, 282
- Datenredundanz 3
- Datenschutz 168
- Datensicherheit 168, 234
- Datensicherung 235
- Datensystem 201
- Datentyp
 - kontinuierlicher 305
- Datenunabhängigkeit 10
 - logische 10, 154
 - physische 10
- Datenwürfel 253
- Dauerhaftigkeit 228, 240
- Db2 14
- DBA *siehe* Datenbankadministrator
- DBMS *siehe* Datenbankmanagementsystem
- DBPL *siehe* Datenbankprogrammiersprache
- DBS *siehe* Datenbanksystem
- DDL *siehe* Datendefinitionssprache
- Deadlock 234
- Delta Encoding 282
- Denormalisierung 215
- Deskriptive Probleme im Mining 268
- dice 257
- Differenzrelation 192
- Dimension 253
- Dimensionstabelle 258
- dirty read 226
- Distanzfunktion 268
- DML *siehe*
 - Datenmanipulationssprache
- Document Stores 287
- Drei-Ebenen-Schema-Architektur 10
- dreiwertige Logik 128
- drill down 256
- dritte Normalform 86
- Duplikat in SQL 115
- Durability 228
- Dynamic Programming 225

- Dynamic SQL 145
- dynamische Eigenschaften 40
- dynamische Verfahren 207
- E**
- Ebene
 - externe 5, 8, 99, 153, 159, 199
 - interne 8, 100, 201, 240
 - konzeptuelle 8, 99, 199
- ECA-Regel 190
- Effektkonformität 160
- Ein- und Auslagerungsstrategie 217
- Eindimensionale Suchverfahren 206
- Embedded SQL 145
- Entität 40
- Entity 41
 - abhängiges 49
 - im ER-Modell 44
 - schwaches 70
 - Typ 44, 64
- Entwurfsdokument 58
- ER-Modell 39, 40
- Erkennung von Tupelmustern 272
- ETL-Prozess 255
- Extension 295
- externes Schema 11
- F**
- Faktentabelle 258
- FD *siehe* funktionale Abhängigkeit
- Feature 305
- Feature-Vektor 305
- Festplatten-orientiert 290
- Fixierung 217
- Flink 318
- Freigabephase 230
- Fünf-Schichten-Architektur 199
- funktionale Abhängigkeit 80
- G**
- geclusterter Index 213
- Generalisierung 179
- Greedy-Strategie 225
- Gruppeneigenschaft 137

Gruppierung 137

H

Hadoop 318

Hash

 Funktion 205

 Partitionierung 313

 Tabelle 210

Hauptspeicher

 optimiert 290, 291

 orientiert 290

Head-Crash 236

Header 204

Heap 204

Heuristik 219

Hierarchie

 extensionale 301

 intensionale 301

Histogramm 224

HOLAP 256

horizontale Partitionierung 215

hybrides OLAP 256

Häufigkeit 270

I

IBM 201

Identifikator

 Quasi-Identifikator 178

Impedance Mismatch 294

Implementierung 59

Implementierungsunabhängigkeit 10

Index 206, 219

Index-organisierte Tabelle 212

Instanz 40

Integration 6

Integrität 168

 operationale 183

 referenzielle 185

 semantische 226

Integritätsbedingung 40, 184

 dynamische 190

Integritätserhaltung 228

integrity constraint 184

Integrity Enhancement Feature 145

Integrität 183

 referenzielle 296

 semantische 183

Inter-Operator-Parallelität 314

Interactive Query Language *siehe*

 Anfragesprache

Interne Optimierung 221

interne Satzchnittstelle 202

internes Schema 10

Intervallnotation 46

Intra-Operator-Parallelität 315

IQL *siehe* Anfragesprache

ISO 145

Isolation 228, 230

Isolationsniveau 232

ISS 202

Iterator 297

J

JavaScript 287

JavaScript Object Notation 287

join

 self 117, 125

Join *siehe* Verbund

JSON 287

K

k-Anonymität 179

k-means-Cluster-Verfahren 268

kartesisches Produkt 28

Katalog 6

Kategorisierung 270

KDD 266

Key-Value 285

Key-Value-Datenbank 286

Key-Value-Store 285

Klasse 295

Klassenbeziehung 295

Knotenfehler 236

Knowledge Discovery in Databases
 266

Kollektionsdatentyp 297

kompatible Attribute 120

Kompression 282

Konfidenz 270
 Konfliktserialisierbarkeit 229
 Konsistenz 228, 230
 Konsistenzerhaltung
 Sichtänderungen 160
 Konstantenselektion in SQL 118
 konzeptioneller Entwurf 58
 konzeptuelles Schema 10
 KV-Store 285, 286

L

Language Bindings 145
 Lauflängencodierung 281, 282
 LDB 311
 lineare Ordnung 26
 Lock 230
 Log 238
 Logbuch 238
 Logische Optimierung 219
 logischer Entwurf 59
 lokale Datenbank 311
 lost update 226

M

Main-Memory-Datenbanksysteme
 282
 Map-Reduce 316
 Map-Reduce-Paradigma 316
 media failure *siehe* Mediafehler
 Mediafehler 235, 236
 Mehrdimensionale Suchverfahren 206
 mehrwertig 64
 mehrwertige Abhängigkeit 92, 93
 Menge in SQL 115
 mengenorientierte Schnittstelle 201
 Merge-Join 223
 Mindesthäufigkeit 271
 Mindestkonfidenz 271
 Minimalität 88
 Sichtänderungen 160
 MMDB *siehe* Multimedia-Datenbank
 MOLAP 256
 MOS 201
 MS Access 15

Multidimensionales Datenmodell 252
 multidimensionales OLAP 256
 Multimedia-Datenbank 304
 Multimenge
 in SQL 115
 Multimengen-Typkonstruktor 300
 Multimengensemantik 115
 multiple Hierarchie 254
 multivalued dependency 93
 MVD 93
 MySQL 14, 15

N

natural join *siehe* Natürlicher
 Verbund
 Natürlicher Verbund 27
 Nested-Loops-Join 223
 Nestung 139
 NewSQL 288
 NF²-Relationen 137
 nicht-geclusterter Index 213
 Non-First-Normal-Form-Relationen
 137
 nonrepeatable read 227
 Normalform 81
 erste 140
 zweite 84
 dritte 86, 96
 vierte 95
 NoSQL 285
 Null-Selektion 128
 Nullwert 101, 128, 162
 in SQL DDL 24
 Nur-Lese-Transaktionen 232

O

Objekt
 persistentes 296
 transientes 296
 Objektdatenbankmanagementsystem
 294
 Objektdatenbankmodell 294
 Objektorientiertes Datenbankmodell
 294

Objektrelationales Datenbankmodell
294

ODL 295

ODMG 294

offenes Tupel 286

OID-Spalte 302

OLAP 247, 249, 273

OLAP-Funktion 263

OLTP 247, 273

Online Analytical Processing 249

Online Transactional Processing 248

Optimierung 217

 algebraische 219

 interne 219

 logische 219

OQL 295

Oracle 14, 145

P

Paralleles Datenbanksystem 312

Parallelisierungsprinzip 313

Parallelität 313

Parametric User 252

Partial match 206

Partitionierung 311, 313

Partitionierungsklausel 264

Partitionierungsparallelität 315

Persistent Stored Modules 145, 243

Persistenz 296

Phantom-Problem 226, 227

Phasenmodell 58, 60

physischer Entwurf 59

Pipelining-Parallelität 314

Postgres 15

PostgreSQL 14, 15

Primärschlüssel 46, 206

Primattribut 85

Privacy 173

 by Design 180

Programmiersprachenanbindung 145

Projektion 26, 27

Projektionssicht 161

Pseudo-RDBS 14

PSM 145, 243

Pufferverwaltung 203, 217

Q

QBIC 309

Quantor 130

Quasi-Identifikator 178

R

range partitioning 313

RDBS 14

Read-write-locks 230

Rechtevergabe 144, 167, 168

Recovery 235, 239

Referenzattribut 296

referenzielle Integrität 185

Refresh 255

Relation 20

Relationale Vollständigkeit 123

relationales Datenbankmodell 19, 293

relationales Datenbanksystem 14

relationales OLAP 256

Relationenalgebra 25, 115

Relationenschema 20, 64

Relationship 41, 44

Replikation 311

Rewriting 219

ROLAP 256

roll up 257

Rollback 234, 240

Rollenname 44

Round-Robin-Partitionierung 313

Row Pattern Recognition 272

Row Store 279, 290

Row/Column Store 290

Run Length Encoding *siehe*

 Lauflängencodierung

S

satzorientierte Schnittstelle 201

Satzschnittstelle

 interne 202

Schedule 229

 serialisierbarer 229

Schema-Architektur 10
 Schema-arm 290
 Schema-flexibel 290, 291
 Schema-los 290
 Schema-starr 290
 Schemalosigkeit 287
 Schlüssel 45
 Schlüsselabhängigkeiten 80
 Schlüsselintegrität 185
 Schlüsselkandidat 46
 Schlüssel-Werte-Datenbank 286
 Schneeflocken-Schema 258, 260
 Schnittstellen der Fünf-Schichten-Architektur 201
 Schnittstellenbeschreibung 295
 schwache Typisierung 286
 SDDL *siehe* Sichtdefinitionssprache
 Seiten 204
 Sekundärschlüssel 206
 Selbstverbund 125
 in SQL 113
 Selektion 25, 26
 Selektionssicht 163
 Selektivität 219, 224
 self join *siehe* Selbstverbund
 semantische Integrität 226
 Semistrukturierte Daten 284
 sensibles Attribut 179
 SEQUEL 145
 SEQUEL2 145
 Sequenzielle Dateiorganisation 205
 Serialisierbarkeit 229
 serieller Schedule 229
 SFW-Block 111
 Shared Disk 312
 Shared Memory 312
 Shared Nothing 312
 Sicherungskopien 240
 Sicht 103, 109
 SQL-92 165
 SQL:2003 165
 Sichtbarkeit 232
 Sichtdefinition in SQL 155
 Sichtdefinitionssprache 144
 Sichtexpansion 218
 Simulated Annealing 225
 site failure 236
 Skalierbarkeit 289
 slice 257
 Snapshot 109
 snowflake schema 260
 Sortierungsklausel 264
 SOS 201
 Spaltenorientiertes Speichermodell 280
 Spark 318
 Speichermodell
 spaltenorientiertes 280
 zeilenweises 279
 Speicherstruktursprache 144
 Speichersystem 202
 Sperrern 217, 230
 Sperrphase 230
 Sperrverwaltung 230
 Spezialisierungshierarchie 301
 SQL 14, 17, 33, 99, 233
 SQL-86 145
 SQL-89 145
 SQL-92 99, 146, 165
 SQL-Server 14
 SQL/DS 145
 SQL/JSON 291
 SQL/MM 306
 SQL/MM Full Text 307
 SQL/MM Still Image 309
 SQL2 146
 SQL3 147
 SQL:1999 148, 261
 SQL:2003 148, 165, 261
 SQL:2006 148
 SQL:2011 148
 SQL:2016 148, 291
 SSL *siehe* Speicherstruktursprache
 star schema 258
 statische Dateiorganisation 207
 statische Eigenschaften 39

Stern-Schema 258
Storage Structure Language *siehe*
 Speicherstruktursprache
Stored Procedures 243
Stromdaten 276
Structured Query Language 14, 33
Suchbaum 205
system failure 236
System R 145, 201
Systemfehler 235, 236
Systempufferschnittstelle 202

T

Tabelle 20
 typisierte 302
Tabellenhierarchie 301
Tag 284
timestamp-ordering protocol 231
Timestamp-Ordering-Regel 231
transaction failure 235
Transaktion 225–227
Transaktionsfehler 235
Transitionsrelation 194
Transitionsvariable 194
transitive Abhängigkeit 85
Trigger 190, 194
Trigger-Aktivierungszeit 193
Trigger-Granularität 193
Tuning 240
Tupel 20
Tupelidentifikator 209
Tupelkonstruktor 302
Tupelmigration 163
Tupelmuster
 Erkennung 272
Tupelvariable
 in SQL 116
Two phase locking 230
Typhierarchie 301
Typintegrität 185

U

UDT 307
Überlaufseite 210

Übersetzung 218
unabhängige Parallelität 313
Ungewissheitsselektion 127
Unlock 230
Untertabelle 302
Update-Anomalie 82
Update-Komponente 33

V

Variety 276
VDBS 310
VDLL *siehe* Sichtdefinitionssprache
Velocity 276
Veracity 276
Verbund
 äußerer 124
 in SQL-92 114
Verbundbedingung in SQL 118
Verbundsicht 164
Verbundtreue 91
Vereinigung
 in SQL 120
Verteiltes Datenbanksystem 310
Verteilung 311
Verteilungsentwurf 59
vertikale Partitionierung 216
Vertraulichkeit 168
View *siehe* Sicht
View Definition Language *siehe*
 Sichtdefinitionssprache
Vollständigkeit
 relationale *siehe* Relationale
 Vollständigkeit
Volume 276
Vorhersage-Probleme 268

W

Warenkorbanalysen 270
Wartung 59
Weak Entity 70
Wert
 im ER-Modell 43
Wertebereiche 26
Wide-Column-Datenmodell 286

Wide-Column-Store 285, 286
Wiederherstellung 235
Windows 15
Word Count 316

X

xRDBMS 290

Z

Zeilenweises Speichermodell 279

Zeitmarkenverfahren 231

Zeitreihenanalyse 271

Zugriffsattributwert 206

Zugriffskontrolle 160, 167, 168

Zugriffspfad 205, 206, 208

Zugriffsrecht 169

Zugriffssystem 202

zweite Normalform 84