

SQL

Mit Beispielen in MySQL/MariaDB

Bearbeitet von
Ralf Adams

2., aktualisierte Auflage 2016. Buch inkl. Online-Nutzung. 454 S. Softcover

ISBN 978 3 446 45074 5

Format (B x L): 17,9 x 24,4 cm

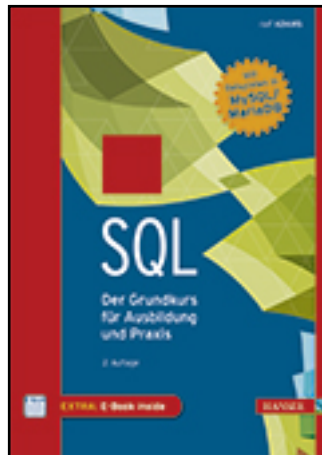
Gewicht: 945 g

[Weitere Fachgebiete > EDV, Informatik > Datenbanken, Informationssicherheit,
Geschäftssoftware](#)

schnell und portofrei erhältlich bei


DIE FACHBUCHHANDLUNG

Die Online-Fachbuchhandlung beck-shop.de ist spezialisiert auf Fachbücher, insbesondere Recht, Steuern und Wirtschaft. Im Sortiment finden Sie alle Medien (Bücher, Zeitschriften, CDs, eBooks, etc.) aller Verlage. Ergänzt wird das Programm durch Services wie Neuerscheinungsdienst oder Zusammenstellungen von Büchern zu Sonderpreisen. Der Shop führt mehr als 8 Millionen Produkte.



Leseprobe

Ralf Adams

SQL

Der Grundkurs für Ausbildung und Praxis. Mit Beispielen in
MySQL/MariaDB

ISBN (Buch): 978-3-446-45074-5

ISBN (E-Book): 978-3-446-45079-0

Weitere Informationen oder Bestellungen unter

<http://www.hanser-fachbuch.de/978-3-446-45074-5>

sowie im Buchhandel.

*Dieses Buch möchte ich allen Lehrerinnen und Lehrern der ehemaligen Aufbau-
realschule in Eslohe, Sauerland, widmen.*

*Besonders denke ich dabei an meinen Klassenlehrer und jetzigen Schulleiter, Herrn Schmidt.
Durch Ihr stetes Bemühen um jeden Einzelnen sind Sie mir menschlich und heute als
Lehrer fachlich ein Vorbild.*



Inhalt

Vorwort	XVII
Teil I Was man so wissen sollte	1
1 Datenbanksystem	3
1.1 Aufgaben und Komponenten	3
1.1.1 Datenbank	3
1.1.2 Datenbankmanagementsystem	5
1.2 Im Buch verwendete Server	6
1.2.1 MySQL und MariaDB	6
1.2.2 PostgreSQL	9
2 Einführung in relationale Datenbanken	11
2.1 Was ist eine relationale Datenbank?	11
2.1.1 Abgrenzung zu anderen Datenbanken	11
2.1.2 Tabelle, Zeile und Spalte	14
2.1.3 Schlüssel, Primärschlüssel und Fremdschlüssel	16
2.2 Kardinalitäten und ER-Modell	22
2.2.1 Darstellung von Tabellen im ER-Modell	22
2.2.2 <i>1:1</i> -Verknüpfung	24
2.2.2.1 Wann liegt eine <i>1:1</i> -Verknüpfung vor?	24
2.2.2.2 Wie kann ich eine <i>1:1</i> -Verknüpfung darstellen?	25
2.2.2.3 Kann man die Kardinalität genauer beschreiben?	26
2.2.3 <i>1:n</i> -Verknüpfung	27
2.2.3.1 Wann liegt eine <i>1:n</i> -Verknüpfung vor?	27
2.2.3.2 Wie kann ich eine <i>1:n</i> -Verknüpfung darstellen?	28
2.2.3.3 Kann man die Kardinalität genauer beschreiben?	28
2.2.4 <i>n:m</i> -Verknüpfung	29

2.2.4.1	Wann liegt eine $n:m$ -Verknüpfung vor?	29
2.2.4.2	Wie kann ich eine $n:m$ -Verknüpfung darstellen?	30
2.2.4.3	Kann man die Kardinalität genauer beschreiben?	31
2.2.5	Aufgaben zum ER-Modell	31
2.3	Referentielle Integrität	32
2.3.1	Verletzung der referentiellen Integrität durch Löschen	33
2.3.2	Verletzung der referentiellen Integrität durch Änderungen	34
2.4	Normalformen	34
2.4.1	Normalform 1	35
2.4.2	Normalform 2	37
2.4.3	Normalform 3	38
2.4.4	Normalform Rest	39
3	Unser Beispiel: Ein Online-Shop	41
3.1	Kundenverwaltung	41
3.2	Artikelverwaltung	42
3.3	Bestellwesen	43
Teil II	Datenbank aufbauen	47
4	Installation des Servers	49
4.1	MySQL unter Windows 10	49
4.2	MariaDB unter Windows 10	55
4.3	Andere Installationen	59
5	Datenbank und Tabellen anlegen	61
5.1	Die Programmiersprache SQL	61
5.2	Anlegen der Datenbank	62
5.2.1	Wie ruft man den MySQL-Client auf?	63
5.2.2	Wie legt man eine Datenbank an?	64
5.2.3	Wie löscht man eine Datenbank?	65
5.2.4	Wie wird ein Zeichensatz zugewiesen?	66
5.2.5	Wie wird eine Sortierung zugewiesen?	68
5.3	Anlegen der Tabellen	70
5.3.1	Welche Datentypen gibt es?	70
5.3.2	Wie legt man eine Tabelle an?	71
5.3.3	Wann eine Aufzählung (ENUM) und wann eine neue Tabelle?	74
5.3.4	Wann ein DECIMAL, wann ein DOUBLE?	76
5.3.5	Wann verwendet man NOT NULL?	77

5.3.6	Wie legt man einen Fremdschlüssel fest?	80
5.3.7	Wie kann man Tabellen aus anderen herleiten?	86
5.3.8	Ich brauche mal eben kurz 'ne Tabelle!	87
6	Indizes anlegen	89
6.1	Index für Anfänger	89
6.1.1	Wann wird ein Index automatisch erstellt?	91
6.1.2	Wie kann man einen Index manuell erstellen?	92
6.2	Und jetzt etwas genauer	95
6.2.1	Wie kann ich die Schlüsseleigenschaft erzwingen?	95
6.2.2	Wie kann ich Dubletten verhindern?	96
6.2.3	Was bedeutet Indexselektivität?	98
6.2.4	Wie kann man einen Index löschen?	100
7	Werte in Tabellen einfügen	101
7.1	Daten importieren	101
7.1.1	Das CSV-Format	102
7.1.2	LOAD DATA INFILE	103
7.1.3	Was ist, wenn ich geänderte Werte importieren will?	106
7.2	Daten anlegen	108
7.2.1	Wie legt man mehrere Zeilen mit einem Befehl an?	108
7.2.2	Wie kann man eine einzelne Zeile anlegen?	110
7.2.3	Vorsicht Constraints!	111
7.2.4	Einfügen von binären Daten	112
7.3	Daten kopieren	115
Teil III	Datenbank ändern	117
8	Datenbank und Tabellen umbauen	119
8.1	Eine Datenbank ändern	119
8.2	Ein Schema löschen	121
8.3	Eine Tabelle ändern	122
8.3.1	Wie kann ich den Namen der Tabelle ändern?	123
8.3.2	Wie kann ich eine Spalte hinzufügen?	124
8.3.3	Wie kann ich die Spezifikation einer Spalte ändern?	126
8.3.4	Zeichenbasierte Spalten in der Länge verändern	127
8.3.5	Zeichensatz verändern	127
8.3.6	Zeichenbasierte Spalten in numerische Spalten verändern	128
8.3.7	Numerische Spalten im Wertebereich verändern	129

8.3.8	Datum- oder Zeitspalten verändern.....	130
8.3.9	Wie kann ich aus einer Tabelle Spalten entfernen?.....	131
8.4	Eine Tabelle löschen.....	132
8.4.1	Einfach löschen.....	133
8.4.2	Was bedeuten die Optionen CASCADE und RESTRICT?	133
9	Werte in Tabellen verändern.....	135
9.1	WHERE-Klausel	135
9.1.1	Wie formuliert man eine einfache Bedingung?	136
9.1.2	Wird zwischen Groß- und Kleinschreibung unterschieden?	137
9.1.3	Wie formuliert man eine zusammengesetzte Bedingung?	138
9.2	Tabelleninhalte verändern	139
9.2.1	Szenario 1: Einfache Wertzuweisung	141
9.2.2	Szenario 2: Berechnete Werte	141
9.2.3	Szenario 3: Gebastelte Zeichenketten	142
9.2.4	Was bedeutet die Option LOW_PRIORITY?.....	143
9.2.5	Was bedeutet die Option IGNORE?.....	143
9.3	Tabelleninhalte löschen.....	143
9.3.1	Und was passiert bei Constraints?	145
9.3.2	Was passiert mit dem AUTO_INCREMENT?	145
9.3.3	Was bedeutet LOW_PRIORITY?	146
9.3.4	Was bedeutet QUICK?	147
9.3.5	Was bedeutet IGNORE?	147
9.3.6	Wie kann man eine Tabelle komplett leeren?	147
Teil IV	Datenbank auswerten	149
10	Einfache Auswertungen	151
10.1	Ausdrücke.....	152
10.1.1	Konstanten.....	152
10.1.2	Wie kann man Berechnungen vornehmen?	153
10.1.3	Wie ermittelt man Zufallszahlen?	154
10.1.4	Wie steckt man das Berechnungsergebnis in eine Variable?	155
10.2	Zeilen- und Spaltenwahl.....	156
10.3	Sortierung.....	157
10.3.1	Was muss bei der Sortierung von Texten beachtet werden?.....	159
10.3.2	Wird zwischen Groß- und Kleinschreibung unterschieden?	161
10.3.3	Wie werden Datums- und Uhrzeitwerte sortiert?.....	163

10.3.4	Wie kann man das Sortieren beschleunigen?	164
10.4	Mehrfachausgaben unterbinden	167
10.4.1	Fallstudie: Datenimport von Bankdaten	168
10.4.2	Was ist beim DISTINCT bzgl. der Performance zu beachten?	171
10.5	Ergebnismenge ausschneiden	171
10.5.1	Wie kann man sich die ersten n Datensätze ausschneiden?	171
10.5.2	Wie kann man Teilmengen mittendrin ausschneiden?	172
10.6	Ergebnisse exportieren.....	174
10.6.1	Wie legt man eine Exportdatei auf dem Server an?	174
10.6.2	Wie legt man eine Exportdatei auf dem Client an?	175
10.6.3	Wie liest man binäre Daten aus?	175
11	Tabellen verbinden.....	179
11.1	Heiße Liebe: Primär-/Fremdschlüsselpaare	180
11.2	INNER JOIN zwischen zwei Tabellen	183
11.2.1	Bauanleitung für einen INNER JOIN	184
11.2.2	Abkürzende Schreibweisen	188
11.2.3	Als Datenquelle für temporäre Tabellen	189
11.2.4	JOIN über Nichtschlüsselspalten	191
11.3	INNER JOIN über mehr als zwei Tabellen	193
11.4	Es muss nicht immer heiße Liebe sein: OUTER JOIN	196
11.5	Narzissmus pur: SELF JOIN	201
11.6	Eine Verknüpfung beschleunigen	204
12	Differenzierte Auswertungen.....	207
12.1	Statistisches mit Aggregatfunktionen	207
12.2	Tabelle in Gruppen zerlegen	210
12.3	Gruppenergebnisse filtern.....	214
12.4	Noch Fragen?.....	215
12.4.1	Kann ich nach Ausdrücken gruppieren?	215
12.4.2	Kann ich nach mehr als einer Spalte gruppieren?	216
12.4.3	Wie kann ich GROUP BY beschleunigen?.....	217
12.5	Aufgaben	218
13	Auswertungen mit Unterabfragen.....	219
13.1	Das Problem und die Lösung	219
13.2	Nicht korrelierende Unterabfrage	222
13.2.1	Skalarunterabfrage.....	222
13.2.1.1	Beispiel 1: Banken mit höchster BLZ.....	222

13.2.1.2	Beispiel 2: Überdurchschnittlich teure Artikel	223
13.2.1.3	Beispiel 3: Überdurchschnittlich wertvolle Bestellungen	224
13.2.2	Listenunterabfrage	226
13.2.2.1	Beispiel 1: IN()	226
13.2.2.2	Beispiel 2: ALL()	227
13.2.2.3	Beispiel 3: ALL()	228
13.2.2.4	Beispiel 4: ANY()	231
13.2.3	Unterschied zwischen IN(), ALL() und ANY()	233
13.2.4	Unterschied zwischen NOT IN() und <> ALL()	233
13.2.5	Tabellenunterabfrage	233
13.3	Korrelierende Unterabfrage	234
13.3.1	Beispiel 1: Rechnungen mit vielen Positionen	234
13.3.2	Beispiel 2: EXISTS	235
13.4	Fallstudie Datenimport	236
13.5	Wie ticken Unterabfragen intern?	239
13.6	Aufgaben	243
14	Mengenoperationen	245
14.1	Die Vereinigung mit UNION	245
14.2	Die Schnittmenge	248
14.2.1	Mit INTERSECT	248
14.2.2	Mit Unterabfragen	249
14.3	Die Differenzmenge	250
14.3.1	Mit EXCEPT	250
14.3.2	Mit Unterabfragen	251
14.4	UNION, INTERSECT und EXCEPT ... versteh' ich nicht!	252
15	Bedingungslogik	255
15.1	Warum ein CASE?	255
15.2	Einfacher CASE	257
15.3	SEARCHED CASE	259
15.4	Fallbeispiele	261
15.4.1	Lagerbestand überprüfen	261
15.4.2	Kundengruppen ermitteln	262
15.4.3	Aktive Lieferanten ermitteln	265
15.4.4	Aufgaben	266

16	Ansichtssache	267
16.1	Was ist eine Ansicht?	267
16.1.1	Wie wird eine Ansicht angelegt?	268
16.1.2	Wie wird eine Ansicht verarbeitet?	270
16.1.3	Wie wird eine Ansicht gelöscht?	273
16.1.4	Wie wird eine Ansicht geändert?	275
16.2	Anwendungsgebiet: Vereinfachung	276
16.3	Anwendungsgebiet: Datenschutz	278
16.4	Grenzen einer Ansicht	279
 Teil V Anweisungen kapseln		283
17	Locking	285
18	Transaktion	289
18.1	Das Problem	289
18.2	Was ist eine Transaktion?	291
18.3	Isolationsebenen	294
18.3.1	READ UNCOMMITTED	294
18.3.2	READ COMMITTED	296
18.3.3	REPEATABLE READ	297
18.3.4	SERIALIZABLE	298
18.4	Fallbeispiel in C#	299
18.5	Deadlock	301
19	STORED PROCEDURE	303
19.1	Einstieg und Variablen	304
19.2	Verzweigung	309
19.2.1	Einfache Verzweigung mit IF	309
19.2.2	Mehrfache Verzweigung mit CASE	312
19.3	Schleifen	315
19.3.1	LOOP-Schleife	316
19.3.2	WHILE-Schleife	318
19.3.3	REPEAT-Schleife	321
19.4	Transaktion innerhalb einer Prozedur	322
19.5	CURSOR	323
19.6	Aufgaben	330
20	Funktion	331

21 TRIGGER	333
21.1 Was ist das?	333
21.2 Ein Beispiel für einen INSERT-Trigger	335
21.3 Ein Beispiel für einen UPDATE-Trigger	336
21.4 Ein Beispiel für einen DELETE-Trigger	338
22 EVENT	341
22.1 Wie legt man ein Ereignis an?	341
22.2 Wie wird man ein Ereignis wieder los?	344
Teil VI Anhänge	345
23 Datenbank administrieren	347
23.1 Backup und Restore	347
23.1.1 Backup mit mysqldump	347
23.1.2 Restore mit mysqldump	349
23.2 Benutzerrechte	350
23.2.1 Benutzerrechte und Privilegien	350
23.2.2 Benutzer anlegen/Recht zuweisen	352
23.2.2.1 CREATE USER	352
23.2.2.2 GRANT	353
23.2.2.3 REVOKE	355
23.3 Datenbankreplikation	356
24 Rund um den MySQL-Client	361
24.1 Aufruf(parameter)	361
24.2 Befehle	364
25 SQL-Referenz	369
25.1 Datentypen	369
25.1.1 Numerische Datentypen	369
25.1.1.1 Ganze Zahlen	369
25.1.1.2 Gebrochene Zahlen	370
25.1.2 Zeichen-Datentypen	371
25.1.3 Datums- und Zeit-Datentypen	372
25.1.4 Binäre Datentypen	375
25.1.5 Standardwerte	375
25.1.6 Zusätze für Datentypen	376
25.2 Operatoren und Funktionen	377

25.2.1	Mathematische Operatoren	377
25.2.2	Mathematische Funktionen	377
25.2.3	Aggregatfunktionen	380
25.3	Bedingungen	382
25.3.1	Vergleichsoperatoren	382
25.3.2	Logikoperatoren	384
25.3.2.1	NOT, Negation, \neg	384
25.3.2.2	AND, Konjunktion, \wedge	385
25.3.2.3	OR, Disjunktion, \vee	386
25.3.2.4	XOR, Antivalenz, \otimes	386
25.4	Befehle	387
25.4.1	Data Definition Language	387
25.4.2	Data Manipulation Language	398
25.4.3	Benutzerverwaltung	402
26	Ausgewählte Quelltexte	405
26.1	DOUBLE versus DECIMAL	405
26.2	NULL versus NOT NULL	409
26.3	Suchen mit und ohne Index	411
26.4	Messen der Performance der Einfügeoperation	414
26.5	Messen der Indexselektivität	418
26.6	Sortieren ohne und mit Index	419
26.7	Rundungsfehler	422
27	Rund ums Zeichen	423
27.1	Für Deutsch relevante Zeichensätze	423
27.2	Für Deutsch relevante Sortierungen	424
28	Quelltexte	427
28.1	MySQL/MariaDB	427
28.2	PostgreSQL	488
Literatur	537
Stichwortverzeichnis	541



Vorwort

Vorwort zur 1. Auflage

Und noch'n SQL-Buch. Es gibt so viele SQL-Bücher, dass man berechtigt die Frage stellen kann, warum man noch eines braucht. Ich kann die Frage nur indirekt beantworten. Als Lehrer für Anwendungsentwicklung an einem Berufskolleg habe ich über Jahre erlebt, dass die Auszubildenden sich sehr mit den üblichen Büchern abmühen.

Die fachliche Qualität dieser Bücher ist unbestritten. Aber die Sprache ist meist von *IT-Profi* zu *IT-Profi*, und genau damit sind Auszubildende und Berufsanfänger oft überfordert – zumindest wird der Einstieg erschwert.

Ich habe daher begonnen, leicht verständliche Skripte zu schreiben, aus denen sich dieses Buch speist. Dabei werden Befehle didaktisch reduziert und Beispiele möglichst lebensnah ausgesucht. Fachbegriffe werden nur verwendet, wenn sie IT-sprachlicher Umgang sind; akademische Begriffe werden vermieden, wobei ich ihre Berechtigung nicht in Abrede stellen möchte.

Primärziel ist ein möglichst umfangreicher Ersteinstieg (Grundkurs), der dann durch berufliche Praxis ausgebaut werden kann. Trotzdem vertiefe ich an vielen Stellen im Buch den Einblick in SQL oder den MySQL-Server (Vertiefendes) – zum einen, um zu zeigen, dass ich auch ein bisschen was drauf habe, zum anderen, um Neugierde und Jagdtrieb beim Leser¹ zu wecken.

Ein weiterer Grund für dieses Buch ist, dass es mir großen Spaß gemacht hat, es zu schreiben. Ich hoffe, dass es Ihnen genau soviel Spaß macht, es zu lesen und damit zu arbeiten. Falls Sie mich fachlich korrigieren oder ergänzen möchten, senden Sie mir doch bitte eine E-Mail an sqlbuch@ralfadams.de.

Der Titel des Buches ist SQL und nicht MySQL. Ich habe deshalb an vielen Stellen den Unterschied zwischen MySQL und dem SQL92-Standard aufgezeigt. Trotzdem wird es schwer sein, die Beispiele *einfach so* auf andere DBMS wie Oracle oder MS-SQL zu übertragen. Die im Internet unter <http://downloads.hanser.de> verfügbaren Quelltexte sind jedenfalls nur unter MySQL 5.5 getestet worden. Trotzdem ist dies kein MySQL-Buch. Auf jeden Fall werden Sie ein Verständnis für den Aufbau und die Funktionsweise der Befehle erwerben, sodass Sie leicht die verschiedenen SQL-Dialekte adaptieren können.

¹ Der besseren Lesbarkeit wegen verzichte ich auf weiblich/männlich-Konstruktionen. Bitte verstehen Sie dies nicht als stillschweigende Hinnahme des geringen Frauenanteils in den IT-Berufen.

Wie ist das Buch zu lesen?

- **Teil I *Was man so wissen sollte***
Als Grundkurs werden hier die zentralen Begriffe wie Datenbank und Datenbankmanagementsystem eingeführt. Anschließend wird unser Beispiel-DBMS MySQL/MariaDB vorgestellt. Falls Sie auf anderen Systemen arbeiten, können Sie diesen Abschnitt überspringen. Nun folgt eine Einführung in die Theorie. Zwar wird in den nachfolgenden Kapiteln immer wieder auf diese Theorie zurückgegriffen, aber Sie können dieses Kapitel auch erst einmal überspringen, wenn Sie nur einen Befehl erlernen möchten. Allen, die sich auf eine Prüfung vorbereiten müssen, empfehle ich aber dringend, dieses Kapitel zu lesen und zu lernen. Zum Schluss wird das im Buch verwendete Beispiel eines Online-Shops inhaltlich entwickelt und als ER-Modell dokumentiert.
- **Teil II *Eine Datenbank aufbauen***
Es werden einige Installationen von MySQL vorgestellt. Leider kann ich nicht alle attraktiven Zielplattformen beschreiben. Es geht hier um die Bereitstellung einer Arbeitsumgebung für einen Entwickler oder besser für den Leser dieses Buches. Nach der Installation werden die Datenbank und die Tabellen des Online-Shops angelegt. Zu den Tabellen werden Indizes eingeführt. Dabei werden immer wieder vertiefende Inhalte angeboten, die bei einem Grundkurs erstmal übersprungen werden können. Zum Schluss werden noch Daten in die Tabellen eingefügt oder aus anderen Datenquellen übernommen.
- **Teil III *Eine Datenbank verändern***
Die Datenbankeigenschaften, die Tabellenstrukturen und die Tabelleninhalte werden verändert oder gelöscht.
- **Teil IV *Eine Datenbank auswerten***
Der umfangreichste Teil des Buchs. Hier werden die Auswertungen mit SELECT in allen möglichen Varianten vorgestellt. Besonders die Verbindung mit JOIN und Unterabfragen werden an vielen Beispielen verständlich gemacht. Fast alle Inhalte gehören zum Grundkurs und sollte man daher z.B. für Prüfungen wissen.
- **Teil V *Anweisungen kapseln***
Transaktionen, Prozeduren, Funktionen, Trigger und Ereignisse gehören nicht zwangsläufig zum Grundkurs, werden hier aber wegen ihrer fachlichen Attraktivität und ihrer wachsenden Bedeutung vorgestellt.
- **Teil VI *Anhänge***
Eine Datenbank zu administrieren, ist – bis auf die Benutzerrechte – so DBMS-spezifisch, dass ich dies in den Anhang geschoben habe. Leider musste ich aus Platzmangel das Thema Cluster aussparen. Es folgt eine (My)SQL-Referenz. Anschließend werden Zeichensätze und Sortierungen, die für die Sprache *Deutsch* relevant sind, aufgelistet. Da ich den MySQL-Client verwende, stelle ich auch noch den Client etwas genauer vor. Zum Schluss folgen noch ausgewählte Quelltexte zum Selbststudium.

Die Lösungen zu den Aufgaben können unter <http://downloads.hanser.de> zwecks Vergleich heruntergeladen werden.

Danksagung

Als Erstes möchte ich mich bei Frau Margarete Metzger vom Hanser Verlag bedanken. Sie hat dieses Buch erst möglich gemacht. Zum Ende hin wurde das Buchprojekt von Frau Brigitte Bauer-Schiewek betreut, der ich ebenfalls meinen Dank aussprechen möchte. Herr Jürgen Dubau hat sprachliche Ausrutscher und flapsige Formulierungen glatt gebügelt. Für die \LaTeX -Probleme war Herr Stephan Korell zuständig. Seine schnellen und guten Lösungen haben die Bearbeitung sehr erleichtert. Das Layout wurde von Frau Irene Weillhart betreut; wenn Ihnen das Layout gefällt, ist das ihrem Sinn für Ästhetik zu verdanken.

Ich möchte meinen Kollegen Dr. Andreas Alef und Marco Bakera, mit denen ich das Vergnügen habe, an der Technischen Beruflichen Schule 1 in Bochum (<http://www.tbs1.de>) zu unterrichten, für ihre kritischen und aufmunternden Kommentare danken.

Besonders will ich meine Schülerinnen und Schüler erwähnen. Die hier vorgestellten Beispiele und Konzepte sind in großen Teilen durch ihre schonungslose Kritik an bestehenden Lehrmaterialien entstanden. Das penetrante *Kapiere ich nicht!* hat mich immer weiter angespornt, es noch verständlicher zu versuchen. Falls dieses Buch SQL gut vermittelt, ist das auch deren Verdienst.

Zum Schluss noch eine Abbitte an meine Familie. Sie hat in den Monaten der Entstehung dieses Buchs oft zurückstecken müssen. Dafür, dass ich von allen – insbesondere von meiner Frau Barbara – so intensiv unterstützt wurde, bin ich sehr dankbar. Das ist nicht selbstverständlich, und ich freue mich darauf, dies mit vielen Runden Kniffel und Rummikub auszugleichen.

Ralf Adams, Juli 2012

Bemerkungen zur überarbeiteten 2. Auflage

- Ich beziehe mich nicht mehr auf den SQL92-Standard, sondern auf den SQL3-Standard.
- Skripte und Beispiele sind auf folgenden Servern getestet worden: *MySQL Community Server 5.7.12* unter Ubuntu, *MariaDB 10.1.13* unter Windows 10 und *PostgreSQL 9.5* unter Windows 10.
- MariaDB hat MySQL in vielen Anwendungen abgelöst. Die Gründe dafür und die entsprechende Diskussion darüber möchte ich hier nicht wiedergeben. Da beide den gleichen SQL-Dialekt sprechen, konnte ich an den meisten Stellen im Buch die Gültigkeit auf MariaDB erweitern. Unterschiede konnte ich bei den verwendeten Skripten nicht feststellen.
- Ich habe an mehreren Stellen auch PostgreSQL-Beispiele eingebaut. Zum einen, weil diese näher am SQL3-Standard sind, und zum anderen, weil es manche Befehlsvarianten in MySQL und MariaDB nicht gibt (z.B. der rekursive SELECT in [Abschnitt 11.5 auf Seite 201](#)).
- Aus diesem Grunde habe ich auch für die meisten Skripte PostgreSQL-Varianten erstellt. Diese können Sie im entsprechenden Verzeichnis des Hanser Verlags herunterladen.
- Die SQL-Skripte der ersten Auflage haben an manchen Stellen Fehlermeldungen ausgegeben. Diese Fehlermeldungen waren gewollt – z.B. das Löschen einer Zeile mit einem

Fremdschlüsselconstraint – und im Quelltext durch Kommentare dokumentiert. Leider ist dies von vielen so interpretiert worden, als seien die Skripte fehlerhaft. Ich habe deshalb die entsprechenden Stellen in den Skripten auskommentiert.

- Bitte beachten Sie, dass die Pfadangaben in den Skripten mit `LOAD DATA INFILE` angepasst werden müssen, je nachdem, wo Sie die Daten entpacken.
- Ich habe angefangen, für die Aufgaben Musterlösungen bei YouTube (<http://www.youtube.com/channel/UCu4ZybNXw1y4Rs4Mgx-4HKw>) einzustellen. In diesen Videos kann ich einfach besser erklären, worauf es bei den Lösungen ankommt.
- Beim Test der Skripte unter MySQL 5.6.19 ist ein Fehler des Servers aufgetreten (siehe [Ada14]).

Bei Frau Sylvia Hasselbach vom Hanser Verlag möchte ich mich dafür bedanken, dass sie die 2. Auflage angestoßen und ermöglicht hat.

Abschließend möchte ich mich für die vielen guten Kontakte und positiven wie kritischen Rückmeldungen über sqlbuch@ralfadams.de bedanken. Es hat mich sehr gefreut, dass durch mein Buch vielen das Erlernen von SQL Spaß gemacht hat.

Ralf Adams, Juni 2016

11

Tabellen verbinden



Daten für Auswertungen verbinden.

- Grundkurs
 - Kartesisches Produkt, CROSS JOIN
 - INNER JOIN mit zwei und mehr Tabellen
 - LEFT und RIGHT OUTER JOIN
- Vertiefendes
 - Abkürzung mit USING
 - NATURAL JOIN
 - JOIN als Datenquelle
 - Verknüpfung über Nichtschlüsselspalten
 - EQUI JOIN
 - OUTER JOIN und referentielle Integrität
 - SELF JOIN
 - Common Table Expression (WITH)
 - Einfluss von Indizes auf JOIN

Spätestens jetzt sind wir im nichttrivialen Bereich von SELECT angekommen. Daten aus mehreren Tabellen zusammenzuführen, ist Alltagsgeschäft und wird von vielen DBMS-Tools unterstützt. Trotzdem muss man das Handwerk dahinter verstehen, denn jeder DB-Designer muss wissen, wie die Daten später wieder zusammengebastelt werden müssen. Ohne den Aufwand zu kennen, lässt sich kein seriöses ER-Modell erstellen¹.



Die Quelltexte dieses Kapitels stehen in der Datei `listing08.sql` (siehe [Listing 28.8](#) auf Seite 445 und [Listing 28.27](#) auf Seite 502).

¹ So, wie wir es in den ersten Kapiteln getan haben ;-).

11.1 Heiße Liebe: Primär-/Fremdschlüsselpaare



SQL3

MySQL/MariaDB

```
SELECT [DISTINCT]
  {*|spaltenliste|ausdruck}
FROM tabellename[, tabellename]*
[WHERE bedingung]
[ORDER BY spaltenname [ASC|DESC] [,spaltenname [ASC|DESC]]*]
;
```

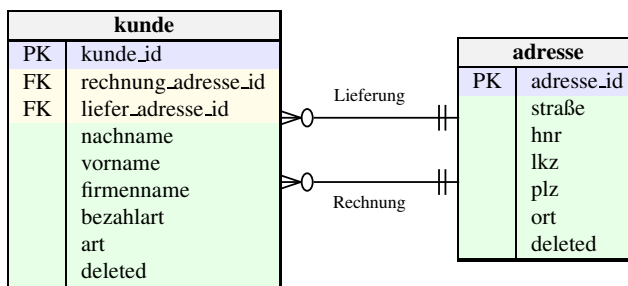


Bild 11.1 ER-Modell: kunde und adresse

Die bisher durchgeführten Auswertungen fanden immer auf *einer* Tabelle statt. Wenn wir uns die ER-Modelle für den Online-Shop anschauen, erkennen wir, dass oft inhaltlich zusammenhängende Informationen auf mehrere Tabellen verteilt sind.²

So stehen beispielsweise die Adressdaten eines Kunden in einer anderen Tabelle als sein Name. Für ein Rechnungsschreiben werden beide Informationen wieder miteinander zu verknüpfen sein. Eine Variante des SELECTs erlaubt die Verwendung mehrerer Tabellen in einem SELECT. Wollen wir mal schauen, was dabei herauskommt:

```
1 mysql> SELECT
2   -> kunde_id, nachname, vorname, rechnung_adresse_id, adresse_id
3   -> FROM
4   ->   kunde, adresse
5   -> ;
6
7 +-----+-----+-----+-----+-----+
8 | kunde_id | nachname | vorname | rechnung_adresse_id | adresse_id |
9 |         |          |         |                     |            |
10 |         |          |         |                     |            |
11 |         |          |         |                     |            |
12 |         |          |         |                     |            |
13 |         |          |         |                     |            |
14 |         |          |         |                     |            |
15 |         |          |         |                     |            |
```

² Dies ist gerade der entscheidende Unterschied zu objektorientierten Datenbanken oder NoSQL.

```

16 |      2 | Beutlin | Frodo |      2 |      2 |
17 |      3 | Beutlin | Bilbo |      2 |      2 |
18 |      4 | Telcontar | Elessar |      3 |      2 |
19 |      5 | Earendilionn | Elrond |      4 |      2 |
20 |      6 | Eichenschild | Thorin | NULL |      2 |
21 |      1 | Gamschie | Samweis |      1 |      3 |
22 |      2 | Beutlin | Frodo |      2 |      3 |
23 |      3 | Beutlin | Bilbo |      2 |      3 |
24 |      4 | Telcontar | Elessar |      3 |      3 |
25 |      5 | Earendilionn | Elrond |      4 |      3 |
26 |      6 | Eichenschild | Thorin | NULL |      3 |
27 |      1 | Gamschie | Samweis |      1 |      4 |
28 |      2 | Beutlin | Frodo |      2 |      4 |
29 |      3 | Beutlin | Bilbo |      2 |      4 |
30 |      4 | Telcontar | Elessar |      3 |      4 |
31 |      5 | Earendilionn | Elrond |      4 |      4 |
32 |      6 | Eichenschild | Thorin | NULL |      4 |
33 |      1 | Gamschie | Samweis |      1 |      5 |
34 |      2 | Beutlin | Frodo |      2 |      5 |
35 |      3 | Beutlin | Bilbo |      2 |      5 |
36 |      4 | Telcontar | Elessar |      3 |      5 |
37 |      5 | Earendilionn | Elrond |      4 |      5 |
38 |      6 | Eichenschild | Thorin | NULL |      5 |
39 |      1 | Gamschie | Samweis |      1 |     10 |
40 |      2 | Beutlin | Frodo |      2 |     10 |
41 |      3 | Beutlin | Bilbo |      2 |     10 |
42 |      4 | Telcontar | Elessar |      3 |     10 |
43 |      5 | Earendilionn | Elrond |      4 |     10 |
44 |      6 | Eichenschild | Thorin | NULL |     10 |
45 |      1 | Gamschie | Samweis |      1 |     11 |
46 |      2 | Beutlin | Frodo |      2 |     11 |
47 |      3 | Beutlin | Bilbo |      2 |     11 |
48 |      4 | Telcontar | Elessar |      3 |     11 |
49 |      5 | Earendilionn | Elrond |      4 |     11 |
50 |      6 | Eichenschild | Thorin | NULL |     11 |
51 |-----+-----+-----+-----+-----+-----+
52 42 rows in set (0.00 sec)

```

In [Zeile 4](#) werden zwei Tabellen hinter dem FROM angegeben. Das ist neu; bisher stand hier immer nur *die eine* Tabelle.

Im Ergebnis werden mir 42 Zeilen einer *neuen* Tabelle angezeigt. Aber wie sind diese Zeilen gebildet worden? Es fällt auf, dass die `kunde_id` sich nach dem Schema 1,2,3,4,5,6 wiederholt. Ebenso interessant ist, dass die `adresse_id` mit jeweils sechs gleichen Werten auftaucht.

Betrachtet man nur die Werte von `kunde_id` und `adresse_id`, so hat man Datenpaare, die wie folgt aufgebaut sind: Jede Adresse ist mit allen Kunden kombiniert worden. Adresse 1 mit Kunde 1 bis 6, Adresse 2 mit Kunde 1 bis 6 usw. Da es sechs Kunden und sieben Adressen gibt, erhalten wir 42 Kombinationen, das *Kartesische Produkt*.



Definition 38: Kartesisches Produkt

Seien A und B zwei endliche Mengen, $a \in A$ und $b \in B$, dann ist die Menge aller unterschiedlichen Paare (a, b) das *Kartesische Produkt* K der Mengen A und B .

Diese Definition lässt $A = B$ zu. Wenn n die Anzahl der Elemente in A ist und m die Anzahl der Elemente in B , dann hat K $n \times m$ viele Elemente.



Definition 39: CROSS JOIN

Das Kartesische Produkt zweier Mengen wird auch *CROSS JOIN* oder *Kreuzprodukt* genannt.

Toll, ein Kartesisches Produkt³, ja und?



Aufgabe 11.1: Betrachten Sie genau die Zahlenpaare `rechnung_adresse_id` und `adresse_id`. Formulieren Sie eine *WHERE*-Klausel, die genau die Zeilen übrig lässt, die zu einem Kunden die richtige Adressnummer angeben.

In den [Zeilen 9, 16, 17, 24](#) und [31](#) stehen jeweils die gleichen Werte. Ausformuliert bedeutet dies: Im Fremdschlüssel `rechnung_adresse_id` steht der gleiche Wert wie in `adresse_id`. Das sind genau die Elemente des Kartesischen Produkts, die eine gültige Verknüpfung zwischen den beiden Tabellen darstellen. Die *WHERE*-Klausel sollte somit nur diese Zeilen übrig lassen:

```

1  mysql> SELECT
2     -> kunde_id, nachname, vorname, rechnung_adresse_id, adresse_id
3     -> FROM
4     -> kunde, adresse
5     -> WHERE
6     -> rechnung_adresse_id = adresse_id
7     -> ;
8
9  +-----+-----+-----+-----+-----+
10 | kunde_id | nachname | vorname | rechnung_adresse_id | adresse_id |
11 |          | Gamschie | Samweis | 1 | 1 |
12 |          | Beutlin  | Frodo   | 2 | 2 |
13 |          | Beutlin  | Bilbo   | 2 | 2 |
14 |          | Telcontar | Elessar | 3 | 3 |
15 |          | Earendilionn | Elrond | 4 | 4 |
16 +-----+-----+-----+-----+-----+
17 5 rows in set (0.00 sec)

```



Aufgabe 11.2: Was ist mit den Adressen mit den Primärschlüsselwerten 5, 10 und 11 passiert? Wie kann man das inhaltlich interpretieren? Und was ist mit dem Kunden *Thorin Eichenschild*?

Die Reduzierung des Kartesischen Produkts auf die Zeilen mit passenden Schlüsselpaaren ist schon ein *INNER JOIN*. Was uns nun noch fehlt, ist ein *richtiger* Befehl dazu.

³ Wird von den Azubis gerne *Orgienjoin* genannt.

11.2 INNER JOIN zwischen zwei Tabellen



Definition 40: INNER JOIN

Der *INNER JOIN* zweier Tabellen ist die Teilmenge des kartesischen Produkts, für welche gilt, dass die Fremdschlüsselwerte zu den Primärschlüsselwerten passen.

Die Formulierung über das Kartesische Produkt mit der passenden *WHERE*-Klausel ist für die Programmierung ein wenig sperrig. Stellen Sie sich vor, dass die Ergebnismenge weitere Bedingungen erfüllen muss oder keine echten Tabellen verwendet werden, sondern Unterabfragen⁴. Deshalb gibt es eine eigene Syntax für den *INNER JOIN*:



SQL3

MySQL/MariaDB

```
SELECT [DISTINCT]
{*|spaltenliste|ausdruck}
FROM
    tabellennamefk INNER JOIN tabellennamepk
    ON tabellennamefk.fk = tabellennamepk.pk
[WHERE bedingung]
[ORDER BY spaltenname [ASC|DESC] [,spaltenname [ASC|DESC]]*]
;
```

Der obige Befehl sähe umgebaut so aus:

```
1 SELECT
2 kunde_id, nachname, vorname, rechnung_adresse_id, adresse_id
3 FROM
4 kunde INNER JOIN adresse ON rechnung_adresse_id = adresse_id;
```



Aufgabe 11.3: Bauen Sie den letzten Befehl so um, dass folgende Ausgabe erzeugt wird:

nachname	vorname	strasse	hnr	ort
Gamdschie	Samweis	Beutelhaldenweg	5	Hobbingen
Beutlin	Frodo	Beutelhaldenweg	1	Hobbingen
Beutlin	Bilbo	Beutelhaldenweg	1	Hobbingen
Telcontar	Elessar	Auf der Feste	1	Minas Tirith
Earendilonn	Elrond	Letztes Haus	4	Bruchtal

⁴ Siehe [Kapitel 13 auf Seite 219](#)

11.2.1 Bauanleitung für einen INNER JOIN

Die Programmierung eines INNER JOIN fällt meinen Schülerinnen und Schülern oft schwer. Deshalb will ich hier ein wenig ausführlicher beschreiben, wie man einen INNER JOIN zusammenbauen kann. Die passende Aufgabe: Wir wollen zu einer Bankverbindung die Bankleitzahl und den Banknamen wissen.

1. **Ermitteln der beteiligten Tabellen:** In unserem Fall sind es die Tabellen `bankverbindung` und `bank`. Schreiben Sie sich diese Tabellen auf: eine auf die linke Seite vom Blatt und eine auf die rechte Seite.
2. **Ermitteln der Primär- und Fremdschlüssel:** Schreiben Sie unter den Tabellennamen jeweils den Primärschlüssel und alle vorkommenden Fremdschlüssel.

`bankverbindung`

`kunde_id`
`bankverbindung_id`
`bank_id`

`bank`

`bank_id`

3. **Fremdschlüssel festlegen:** In einer der Tabellen muss ein Fremdschlüssel vorkommen, der auf die andere Tabelle zeigt. Wenn Sie beim Design alles richtig gemacht haben und die Namenskonvention beachten, finden Sie diesen sehr schnell. Es kommen zwei Fremdschlüssel infrage: `kunde_id` und `bank_id`. Da eine der Tabellen `bank` heißt und wir der Namenskonvention gefolgt sind, muss es `bank_id` sein. Markieren Sie den Fremdschlüssel z.B. mit einem Textmarker.

`bankverbindung`

`kunde_id`
`bankverbindung_id`
`bank_id`

`bank`

`bank_id`

4. **Primärschlüssel festlegen:** Jetzt markieren Sie in der gleichen Farbe in der anderen Tabelle den dazugehörigen Primärschlüssel. Hier ist es die Spalte `bank_id`

`bankverbindung`

`kunde_id`
`bankverbindung_id`
`bank_id`

`bank`

`bank_id`

5. **Schablone benutzen:** Jetzt schreiben Sie auf das Blatt die Schablone für den INNER JOIN. Das Ganze sollte jetzt so aussehen:


```
bankverbindung
kunde_id
bankverbindung_id
bank_id
```

```
bank
bank_id
```

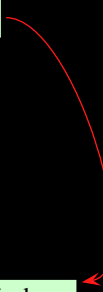
```
SELECT
kunde_id, kontonummer, blz, bankname
FROM
INNER JOIN
ON . = .
```

6. **Fremdschlüsseltabelle eintragen:** Fügen Sie links vom INNER JOIN den Tabellennamen für die Tabelle mit dem markierten Fremdschlüssel ein.

```
bankverbindung
kunde_id
bankverbindung_id
bank_id
```

```
bank
bank_id
```

```
SELECT
kunde_id, kontonummer, blz, bankname
FROM
bankverbindung INNER JOIN
ON . = .
```

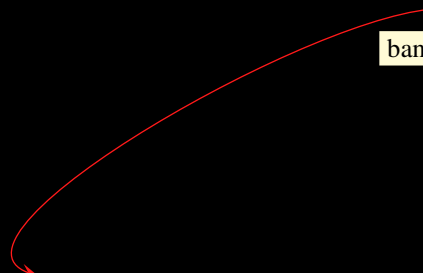


7. **Primärschlüsseltabelle eintragen:** Fügen Sie rechts vom INNER JOIN den Tabellennamen für die Tabelle mit dem markierten Primärschlüssel ein.

```
bankverbindung
kunde_id
bankverbindung_id
bank_id
```

```
bank
bank_id
```

```
SELECT
kunde_id, kontonummer, blz, bankname
FROM
bankverbindung INNER JOIN bank
ON . = .
```



8. **Fremdschlüssel eintragen:** Fügen Sie zwischen dem ON und dem Gleichheitszeichen den Namen der Fremdschlüsseltabelle, einen Punkt und den Namen des Fremdschlüssels ein.

bankverbindung

```
kunde_id
bankverbindung_id
bank_id
```

bank

```
bank_id
```

```
SELECT
kunde_id, kontonummer, blz, bankname
FROM
bankverbindung INNER JOIN bank
ON bankverbindung.bank_id = bank.bank_id
```

9. **Primärschlüssel eintragen:** Fügen Sie nach dem = den Namen der Primärschlüssel-tabelle, einen Punkt und den Namen des Primärschlüssels ein.

bankverbindung

```
kunde_id
bankverbindung_id
bank_id
```

bank

```
bank_id
```

```
SELECT
kunde_id, kontonummer, blz, bankname
FROM
bankverbindung INNER JOIN bank
ON bankverbindung.bank_id = bank.bank_id
```

10. Fertig!

```
1 mysql> SELECT
2   -> kunde_id, kontonummer, blz, bankname
3   -> FROM
4   -> bankverbindung INNER JOIN bank
5   ->   ON bankverbindung.bank_id = bank.bank_id
6   -> ;
7 +-----+-----+-----+-----+
8 | kunde_id | kontonummer | blz      | bankname |
9 +-----+-----+-----+-----+
10 |         1 | 1111111111 | 10010010 | Postbank |
11 |         1 | 1111111112 | 10010010 | Postbank |
12 |         2 | 2222222221 | 10060198 | Pax-Bank |
13 |         3 | 3333333331 | 10060198 | Pax-Bank |
14 |         4 | 4444444441 | 12070000 | Deutsche Bank |
15 |         5 | 5555555551 | 12070000 | Deutsche Bank |
16 +-----+-----+-----+-----+
```

Ein zweites Beispiel: Zu einem Kunden werden die Kontonummern ausgegeben.

1. **Ermitteln der beteiligten Tabellen:** Es sind kunde und bankverbindung.
2. **Ermitteln der Primär- und Fremdschlüssel:** In der Tabelle bankverbindung gibt es den zusammengesetzten Primärschlüssel mit kunde_id und bankverbindung_nr und den Fremdschlüssel bank_id. In der Tabelle kunde gibt es den Primärschlüssel

kunde_id und die beiden Fremdschlüssel rechnung_adresse_id und liefer_adresse_id.

3. **Fremdschlüssel festlegen:** Da wir die Namenskonvention beachtet haben, brauchen wir nur nach einem Fremdschlüssel suchen, der so wie die andere Tabelle heißt. Dies ist in unserem Fall kunde_id.
 4. **Primärschlüssel festlegen:** Jetzt wird die Spalte kunde_id in der Tabelle kunde markiert.
 5. **Schablone benutzen:** Jetzt schreiben Sie auf das Blatt die Schablone für den INNER JOIN.
 6. **Fremdschlüsseltabelle eintragen:** Fügen Sie links vom INNER JOIN den Tabellennamen für die Tabelle mit dem markierten Fremdschlüssel ein.
 7. **Primärschlüsseltabelle eintragen:** Fügen Sie an die passende Stelle den Tabellennamen für die Tabelle mit dem markierten Primärschlüssel ein.
 8. **Fremdschlüssel eintragen:** Fügen Sie an die passende Stelle den Spaltennamen des markierten Fremdschlüssels ein.
 9. **Primärschlüssel eintragen:** Fügen Sie an die passende Stelle den Spaltennamen des markierten Primärschlüssels ein.
10. **Fertig!**

```

1  mysql> SELECT
2      ->  nachname, vorname, kontonummer
3      ->  FROM
4      ->  bankverbindung b INNER JOIN kunde k ON b.kunde_id = k.kunde_id;
5  +-----+-----+-----+
6  | nachname      | vorname  | kontonummer |
7  +-----+-----+-----+
8  | Gamdschie     | Samweis  | 1111111111  |
9  | Gamdschie     | Samweis  | 1111111112  |
10 | Beutlin       | Frodo    | 2222222221  |
11 | Beutlin       | Bilbo    | 3333333331  |
12 | Telcontar     | Elessar  | 4444444441  |
13 | Earendilionn | Elrond   | 5555555551  |
14 +-----+-----+-----+
```

Haben Sie gesehen, dass in Zeile 4 die Tabellennamen durch Aliase ersetzt wurden?



Aufgabe 11.4: Geben Sie zu allen Kunden den Namen und die Rechnungsadresse aus.

Aufgabe 11.5: Geben Sie zu allen Kunden den Namen und die Lieferadresse aus. Interpretieren Sie das Ergebnis.

Aufgabe 11.6: Geben Sie zu jedem Kunden den Namen und die Bestellungen nach Kundennamen und Bestelldatum sortiert aus. Die letzte Bestellung des Kunden soll zuerst erscheinen.

Aufgabe 11.7: Geben Sie zu jeder Bestellung die Kundennummer, das Bestelldatum und die Positionen aus. Die Sortierung soll nach Kundennummer, Bestellnummer und Position erfolgen.

Aufgabe 11.8: Geben Sie zu jeder Position den Artikelnamen aus. Es soll nach Artikelname sortiert werden. ■

11.2.2 Abkürzende Schreibweisen



SQL3

MySQL/MariaDB

```
SELECT [DISTINCT]
  {*|spaltenliste|ausdruck}
FROM
  tabellennamefk {INNER|NATURAL} JOIN tabellennamepk
  [{ON tabellennamefk.fk = tabellennamepk.pk|USING (spaltenname)}]
[WHERE bedingung]
[ORDER BY spaltenname [ASC|DESC] [,spaltenname [ASC|DESC]]*]
;
```

Falls der Name des Fremdschlüssels genau der gleiche ist wie der des Primärschlüssels, kann man die ON-Klausel durch eine kürzere Variante mit USING ersetzen:

```
1 SELECT
2   kunde_id, kontonummer, blz, bankname
3 FROM
4   bankverbindung INNER JOIN bank USING (bank_id)
5 ;
```

Bei den Fremdschlüsseln, die anders als die Primärschlüssel heißen, wie beispielsweise `rechnung_adresse_id`, ist eine solche Abkürzung nicht möglich.



Aufgabe 11.9: Bauen Sie Ihre Lösungen zu den Aufgaben um, wenn ein USING möglich ist.

Sind die Fremdschlüssel-/Primärschlüsselspalten die einzigen Spalten, die in beiden Tabellen gleich sind und die Verknüpfung definieren, spricht man von einem NATURAL JOIN.



Definition 41: NATURAL JOIN

Werden zwei Tabellen über die Spalten verknüpft, die den gleichen Namen und Inhalt haben, spricht man von einem NATURAL JOIN.

```
1 SELECT
2   kunde_id, kontonummer, blz, bankname
3 FROM
4   bankverbindung NATURAL JOIN bank
5 ;
```



Hinweis: Bitte beachten Sie, dass bei unseren Tabellen in einem NATURAL JOIN auch die Spalte `deleted` in die Verknüpfung mit einfließt.

11.2.3 Als Datenquelle für temporäre Tabellen

Wir haben oben die Bankverbindung zu einem Kunden ermittelt. Annahme: Wir wollen jetzt viele Auswertungen der Kunden inklusive der Bankverbindung machen, dann müssen jedes Mal im entsprechenden SELECT die Informationen mit INNER JOIN verknüpft werden.

Obwohl INNER JOIN-Anweisungen durch passend gewählte Indizes sehr beschleunigt werden, entsteht trotzdem eine Rechnerlast auf dem Server, die jedes Mal das gleiche – oder fast das gleiche – Ergebnis liefert⁵.

Da es plausibel ist anzunehmen, dass sich die Kundenstammdaten (siehe [Definition 34](#)) für einen gewissen Auswertungszeitraum nicht ändern, könnte man statt dessen das Ergebnis des INNER JOIN in eine (temporäre) Tabelle ablegen und mit dieser weiterarbeiten.

Als Beispiel wollen wir die Kundendaten mit Rechnungsanschrift und allen Bestellungen ausgeben und danach die Kundendaten mit Rechnungsanschrift mit allen Bankverbindungen.

In beiden Fällen werden die Kundendaten mit Rechnungsanschrift benötigt. Das können wir schon:

```

1 SELECT
2   k.kunde_id, k.nachname, k.vorname, a.strasse, a.hnr, a.plz, a.ort
3 FROM
4   kunde k INNER JOIN adresse a
5   ON k.rechnung_adresse_id = a.adresse_id
6 ;

```

Sie bemerken bitte, dass hier abkürzende Alias ([Zeile 4](#)) verwendet werden. Das ist gerade bei Verknüpfungen sehr beliebt, da hier oft zwischen den Tabellen unterschieden werden muss. Diese Abfrage wird jetzt in eine Variante des CREATE TABLE eingebaut.



MySQL/MariaDB

```

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tabellenname
SELECT auswahl
;

```

```

1 mysql> CREATE TEMPORARY TABLE tmp_kadresse
2   -> SELECT
3   ->   k.kunde_id, k.nachname, k.vorname, a.strasse, a.hnr, a.plz, a.ort
4   -> FROM
5   ->   kunde k INNER JOIN adresse a
6   ->   ON k.rechnung_adresse_id = a.adresse_id;
7
8 Query OK, 5 rows affected (0.09 sec)
9 Records: 5 Duplicates: 0 Warnings: 0
10
11 mysql> SELECT kunde_id, nachname, ort FROM tmp_kadresse;
12 +-----+-----+-----+
13 | kunde_id | nachname | ort |

```

⁵ Wobei davon auszugehen ist, dass exakt gleiche Ergebnisse durch eine Cache-Strategie beschleunigt zur Verfügung gestellt werden können.

```

14 +-----+-----+-----+
15 |      1 | Gamdschie   | Hobbingen |
16 |      2 | Beutlin     | Hobbingen |
17 |      3 | Beutlin     | Hobbingen |
18 |      4 | Telcontar   | Minas Tirith |
19 |      5 | Earendilionn | Bruchtal  |
20 +-----+-----+-----+

```

Jetzt zur ersten Auswertung: Alle Kunden mit Adressdaten und ihren Bestellungen:

```

1  mysql> SELECT
2      -> t.kunde_id, t.nachname, t.ort, b.bestellung_id, DATE(b.datum)
3      -> FROM
4      -> bestellung b INNER JOIN tmp_kadresse t USING (kunde_id);
5
6 +-----+-----+-----+-----+-----+
7 | kunde_id | nachname   | ort        | bestellung_id | DATE(b.datum) |
8 +-----+-----+-----+-----+-----+
9 |          1 | Gamdschie  | Hobbingen | 1              | 2012-03-24    |
10 |          2 | Beutlin    | Hobbingen | 2              | 2012-03-23    |
11 +-----+-----+-----+-----+-----+

```



Aufgabe 11.10: Erzeugen Sie mit der Spaltenliste t.*, b.bestellung_id, datum eine neue temporäre Tabelle und verknüpfen Sie diese mit den Positionen der Bestellungen. Achten Sie auf eine sinnvolle Sortierung.

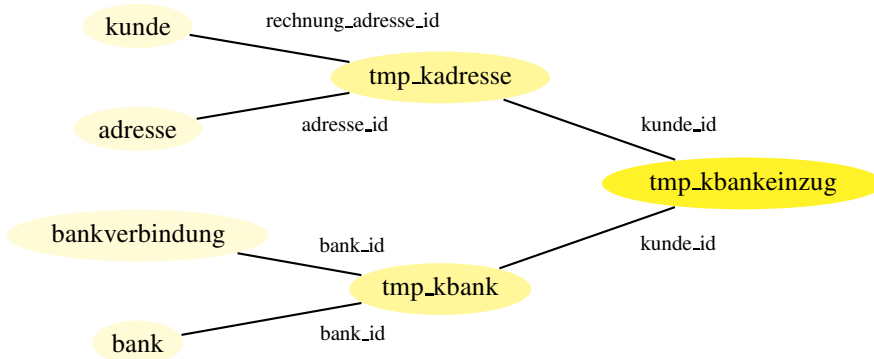
Jetzt zur zweiten Auswertung: Alle Kunden mit Adressdaten und allen Bankverbindungen. Das wollen wir jetzt besonders schön machen. Zuerst eine temporäre Tabelle für die Bankleitzahl und den Banknamen, und dann werden diese beiden temporären Tabellen wieder verknüpft. Und weil wir es jetzt können, wird am Ende noch eine temporäre Tabelle gebaut.

```

1  mysql> CREATE TEMPORARY TABLE tmp_kbank
2      -> SELECT
3      -> bv.kunde_id, bv.bankverbindung_nr, bv.kontonummer,
4      -> bv.iban, ba.blz, ba.bankname
5      -> FROM
6      -> bankverbindung bv INNER JOIN bank ba USING (bank_id);
7
8  mysql> CREATE TEMPORARY TABLE tmp_kbankeinzug
9      -> SELECT
10     -> ka.*, kb.bankverbindung_nr, kb.kontonummer,
11     -> kb.iban, kb.blz, kb.bankname
12     -> FROM
13     -> tmp_kadresse ka INNER JOIN tmp_kbank kb USING(kunde_id);
14
15  mysql> SELECT kunde_id, nachname, ort, bankname FROM tmp_kbankeinzug;
16
17 +-----+-----+-----+-----+
18 | kunde_id | nachname   | ort        | bankname       |
19 +-----+-----+-----+-----+
20 |          1 | Gamdschie  | Hobbingen | Postbank       |
21 |          1 | Gamdschie  | Hobbingen | Postbank       |
22 |          2 | Beutlin    | Hobbingen | Pax-Bank       |
23 |          3 | Beutlin    | Hobbingen | Pax-Bank       |
24 |          4 | Telcontar  | Minas Tirith | Deutsche Bank |
25 |          5 | Earendilionn | Bruchtal  | Deutsche Bank |
26 +-----+-----+-----+-----+

```

Das Zusammenspiel der tatsächlichen und temporären Tabellen sei hier in einer Baumstruktur dargestellt. Die Blätter⁶ sind die Ursprungstabellen. Zwei davon werden in jeweils eine temporäre Tabelle mit INNER JOIN zusammengefasst. Die beiden neu erzeugten temporären Tabellen werden wiederum in eine temporäre Tabelle vereinigt.



Jede der Tabellen kann nun unabhängig voneinander verwendet werden, da die Datensätze in den temporären Tabellen keine Verweise auf die ursprünglichen Datensätze sind, sondern Kopien. Diese *Unabhängigkeit* ist sehr wichtig, wenn man mit vielen konkurrierenden Zugriffen auf die Tabellen kunde etc. rechnet.

Bei der Betrachtung von Transaktionen (siehe [Kapitel 18 auf Seite 289](#)) werden wir noch sehen, dass Tabellen für Operationen gesperrt werden. Durch die temporären Tabellen kann aber auf der Tabelle kunde gearbeitet werden, während Auswertungen auf tmp_kadresse stattfinden.



Hinweis: Ich habe oben erwähnt, dass es sich um Stammdaten handelt und somit während der Auswertungen eine geringe Änderungswahrscheinlichkeit besteht. Bei Bewegungsdaten ist das Verfahren genau abzuwägen. Vielleicht lassen sich ja Änderungen auf eine gewisse Uhrzeit festmachen, und die Auswertungen passieren außerhalb dieses Zeitfensters.

11.2.4 Ist Ihnen was aufgefallen?

Beim Bau der letzten temporären Tabelle gab es keinen Primärschlüssel!

```

1  mysql> DESCRIBE tmp_kadresse;
2  +-----+-----+-----+-----+-----+-----+
3  | Field  | Type                | Null | Key | Default | Extra |
4  +-----+-----+-----+-----+-----+-----+
5  | kunde_id | int(10) unsigned    | NO   |     | 0        |       |
6  | nachname | varchar(255)        | NO   |     |          |       |
7  | vorname  | varchar(255)        | NO   |     |          |       |
8  | strasse  | varchar(255)        | NO   |     |          |       |
9  | hnr     | varchar(255)        | NO   |     |          |       |
  
```

⁶ die Knoten ganz links

```

10 | plz      | char(5)      | NO | | | |
11 | ort      | varchar(255) | NO | | | |
12 +-----+-----+-----+-----+-----+-----+
13
14 mysql> DESCRIBE tmp_kbank;
15 +-----+-----+-----+-----+-----+-----+
16 | Field          | Type                | Null | Key | Default | Extra |
17 +-----+-----+-----+-----+-----+-----+
18 | kunde_id       | int(10) unsigned    | NO   |     | 0        |       |
19 | bankverbindung_nr | int(10) unsigned    | NO   |     | 0        |       |
20 | kontonummer    | char(25)            | NO   |     |          |       |
21 | iban           | char(34)            | NO   |     |          |       |
22 | blz            | char(12)            | NO   |     |          |       |
23 | bankname       | varchar(255)        | NO   |     |          |       |
24 +-----+-----+-----+-----+-----+-----+

```

Und tatsächlich: Für einen INNER JOIN ist es nicht nötig, Primär-/Fremdschlüsselpaare zu bilden. Dem Befehl ist es völlig egal, was bei einem INNER JOIN in der ON- oder USING-Klausel steht. Es wird nur die Verträglichkeit der Datentypen untersucht.

Natürlich erfolgt die überwiegende Anzahl der Verknüpfungen auf Primär-/Fremdschlüsselpaaren. Aber hier haben wir ein Beispiel dafür, dass auch die Verknüpfung über Nichtschlüsselspalten⁷ sinnvoll sein kann.

Es geht sogar noch weiter. Bei der ON-Klausel ist das Gleichheitszeichen nicht zwingend vorgeschrieben:

```

1  mysql> SELECT
2      -> k.kunde_id, k.vorname, a.strasse, a.hnr, a.plz, a.ort
3      -> FROM
4      ->   kunde k INNER JOIN adresse a
5      ->   ON k.rechnung_adresse_id <= a.adresse_id;
6
7  +-----+-----+-----+-----+-----+-----+
8  | kunde_id | vorname | strasse          | hnr | plz | ort          |
9  +-----+-----+-----+-----+-----+-----+
10 |          |         | Beutelhaldenweg | 5   | 67676 | Hobbingen   |
11 |          |         | Beutelhaldenweg | 1   | 67676 | Hobbingen   |
12 |          |         | Beutelhaldenweg | 1   | 67676 | Hobbingen   |
13 |          |         | Auf der Feste    | 1   | 54786 | Minas Tirith |
14 |          |         | Auf der Feste    | 1   | 54786 | Minas Tirith |
15 |          |         | Auf der Feste    | 1   | 54786 | Minas Tirith |
16 |          |         | Auf der Feste    | 1   | 54786 | Minas Tirith |
17 |          |         | Letztes Haus     | 4   | 87567 | Bruchtal    |
18 |          |         | Letztes Haus     | 4   | 87567 | Bruchtal    |
19 |          |         | Letztes Haus     | 4   | 87567 | Bruchtal    |
20 [...]
21 |          |         | Baradur          | 1   | 62519 | Lugburz     |
22 |          |         | Baradur          | 1   | 62519 | Lugburz     |
23 |          |         | Baradur          | 1   | 62519 | Lugburz     |
24 |          |         | Baradur          | 1   | 62519 | Lugburz     |
25 |          |         | Hochstrasse      | 4a  | 44879 | Bochum      |
26 |          |         | Hochstrasse      | 4a  | 44879 | Bochum      |
27 |          |         | Hochstrasse      | 4a  | 44879 | Bochum      |
28 |          |         | Hochstrasse      | 4a  | 44879 | Bochum      |
29 |          |         | Hochstrasse      | 4a  | 44879 | Bochum      |

```

⁷ Immerhin sind es Fremdschlüssel.


```

30 |      1 | Samweis | Industriegebiet | 8 | 44878 | Bochum |
31 |      2 | Frodo   | Industriegebiet | 8 | 44878 | Bochum |
32 |      3 | Bilbo   | Industriegebiet | 8 | 44878 | Bochum |
33 |      4 | Elessar | Industriegebiet | 8 | 44878 | Bochum |
34 |      5 | Elrond  | Industriegebiet | 8 | 44878 | Bochum |
35 +-----+-----+-----+-----+-----+-----+
36 28 rows in set (0.00 sec)

```

In [Zeile 5](#) ist anstelle des = ein <= verwendet worden. Ich kann mir zwar keine vernünftige Anwendung dafür ausdenken, will aber nicht bestreiten, dass es sie irgendwo gibt.

Wird aber die Verknüpfung über die Gleichheit hergestellt, hat das Ganze einen schönen Namen:



Definition 42: EQUI JOIN

Wird bei INNER JOIN auf die Gleichheit von Fremdschlüsselwert und Primärschlüsselwert getestet, spricht man von einem *EQUI JOIN*.

Sie haben sicher bei der [Definition 40 auf Seite 183](#) bemerkt, dass nur allgemein von *passen* gesprochen wird. Was immer dieses *passen* auch bedeutet. Der Test auf Gleichheit ist aber so gängig, dass der Begriff INNER JOIN synonym für EQUI JOIN verwendet wird.



Hinweis: Lassen Sie sich nicht verwirren. Erstmal nach Primär-/ Fremdschlüsselpaaren suchen und diese mit = verknüpfen. In den absolut meisten Fällen sind Sie auf der sicheren Seite. Erst, wenn das so überhaupt nicht klappen sollte, denken Sie über Alternativen nach.

11.3 INNER JOIN über mehr als zwei Tabellen

Eine Möglichkeit, mehr als zwei Tabellen zu verknüpfen, haben Sie oben auf [Seite 189](#) kennengelernt. Die Verwendung von temporären Tabellen bietet sich aber nur bei Stammdaten an oder wenn die Änderungen unerheblich für das Gesamtergebnis sind.

Trotzdem können wir aus der Episode mit den temporären Tabellen eine wichtige Schlussfolgerung ziehen: Das Ergebnis eines INNER JOINs ist wieder eine Tabelle. Wie kann ich damit die Beschränkung umgehen, dass der INNER JOIN nur zwei Tabellen verknüpfen kann?

Betrachten wir dazu den Klassiker für die Verknüpfung von mehr als zwei Tabellen: das Auflösen einer $n:m$ -Verknüpfung (siehe [Abschnitt 2.2.4 auf Seite 29](#)). Wir haben eine $n:m$ -Verknüpfung zwischen den Tabellen `artikel` und `warengruppe`.

Unser erster Versuch besteht darin, wie oben beschrieben vorzugehen, indem wir zwei $1:n$ -Verknüpfungen erstellen:

1. **Ermitteln der beteiligten Tabellen:** `artikel_nm_warengruppe` und `artikel`.

2. **Ermitteln der Primär- und Fremdschlüssel:** In der Tabelle `artikel_nm_warengruppe` gibt es keine *echten* Primärschlüssel, aber die beiden Fremdschlüssel `warengruppe_id` und `artikel_id`. In der Tabelle `artikel` gibt es nur den Primärschlüssel `artikel_id`.
3. **Fremdschlüssel festlegen:** Laut Namenskonvention muss `artikel_id` in der Tabelle `artikel_nm_warengruppe` der gesuchte Fremdschlüssel sein.
4. **Primärschlüssel festlegen:** Wir markieren `artikel_id` in `artikel`.
5. **Schablone benutzen:** Jetzt schreiben Sie auf das Blatt die Schablone für den INNER JOIN.
6. **Fremdschlüsseltabelle eintragen:** Fügen Sie links vom INNER JOIN den Tabellennamen für die Tabelle mit dem markierten Fremdschlüssel ein.
7. **Primärschlüsseltabelle eintragen:** Fügen Sie an die passende Stelle den Tabellennamen für die Tabelle mit dem markierten Primärschlüssel ein.
8. **Fremdschlüssel eintragen:** Fügen Sie an die passende Stelle den Spaltennamen des markierten Fremdschlüssels ein.
9. **Primärschlüssel eintragen:** Fügen Sie an die passende Stelle den Spaltennamen des markierten Primärschlüssels ein.
10. **Fertig!**

```

1  mysql> SELECT
2      -> a.bezeichnung, nm.warengruppe_id
3      -> FROM
4      -> artikel_nm_warengruppe nm NATURAL JOIN artikel a;
5  +-----+-----+
6  | bezeichnung | warengruppe_id |
7  +-----+-----+
8  | Feder       |                | 1 |
9  | Papier (100)|                | 1 |
10 | Schaufel    |                | 3 |
11 | Schaufel    |                | 4 |
12 | Silberzwiebel |                | 2 |
13 | Silberzwiebel |                | 3 |
14 | Spaten      |                | 3 |
15 | Spaten      |                | 4 |
16 | Tinte (blau)|                | 1 |
17 | Tinte (gold)|                | 1 |
18 | Tinte (rot) |                | 1 |
19 | Tulpenzwiebel |                | 2 |
20 | Tulpenzwiebel |                | 3 |
21 +-----+-----+

```

Das ganze Prozedere mit den Tabellen `artikel_nm_warengruppe` und `warengruppe` führt zu einer zweiten Verknüpfung:

```

1  mysql> SELECT
2      -> w.bezeichnung, nm.artikel_id
3      -> FROM
4      -> artikel_nm_warengruppe nm NATURAL JOIN warengruppe w;
5  +-----+-----+
6  | bezeichnung | artikel_id |
7  +-----+-----+
8  | Bürobedarf  |          3001 |
9  | Bürobedarf  |          3005 |
10 | Bürobedarf  |          3006 |

```

```

11 | Bürobedarf |      3007 |
12 | Bürobedarf |      3010 |
13 | Gartenbedarf |     7856 |
14 | Gartenbedarf |     7863 |
15 | Gartenbedarf |     9010 |
16 | Gartenbedarf |     9015 |
17 | Pflanzen   |     7856 |
18 | Pflanzen   |     7863 |
19 | Werkzeug   |     9010 |
20 | Werkzeug   |     9015 |
21 | +-----+-----+

```

Und jetzt kommt's: Wir nehmen einfach den letzten SELECT, klammern die Verknüpfung und verwenden nicht mehr NATURAL, sondern INNER JOIN:

```

1 SELECT
2   w.bezeichnung, nm.artikel_id
3 FROM
4   (artikel_nm_warengruppe nm INNER JOIN warengruppe w USING(warengruppe_id))
5 ;

```

Der Inhalt der Klammer ist eine (!) Tabelle, und diese könnte der linke Teil einer neuen Verknüpfung sein:

```

1 SELECT
2   w.bezeichnung, a.bezeichnung
3 FROM
4   (artikel_nm_warengruppe nm INNER JOIN warengruppe w USING (warengruppe_id))
5   INNER JOIN artikel a USING(artikel_id)
6 ;

```

Und eine weitere gute Nachricht ist, dass man die Klammern nicht braucht; sie dienten nur dem besseren Verständnis:

```

1 SELECT
2   w.bezeichnung Warengruppe, a.bezeichnung Artikel
3 FROM
4   artikel_nm_warengruppe nm INNER JOIN warengruppe w USING (warengruppe_id)
5   INNER JOIN artikel a USING(artikel_id)
6 ;

```

Dieser SELECT liefert das gewünschte Ergebnis:

```

1 +-----+-----+
2 | Warengruppe | Artikel |
3 +-----+-----+
4 | Bürobedarf  | Papier (100) |
5 | Bürobedarf  | Tinte (gold) |
6 | Bürobedarf  | Tinte (rot)  |
7 | Bürobedarf  | Tinte (blau) |
8 | Bürobedarf  | Feder        |
9 | Gartenbedarf | Silberwiebel |
10 | Gartenbedarf | Tulpenzwiebel |
11 | Gartenbedarf | Schaufel     |
12 | Gartenbedarf | Spaten      |
13 | Pflanzen    | Silberwiebel |
14 | Pflanzen    | Tulpenzwiebel |
15 | Werkzeug    | Schaufel     |
16 | Werkzeug    | Spaten      |
17 +-----+-----+

```

Die Spezifikation des SELECT lässt sich somit erweitern:



SQL3

MySQL/MariaDB

```
SELECT [DISTINCT]
  {*|spaltenliste|ausdruck}
FROM
  tablename [{INNER|NATURAL} JOIN tablename
  [{(ON tablename.spaltenname = tablename.spaltenname|USING(spaltenname))}]]*
[WHERE bedingung]
[ORDER BY spaltenname [ASC|DESC] [,spaltenname [ASC|DESC]]*]
;
```

Der Abschnitt rechts vom ersten *tablename* kann beliebig oft wiederholt werden, wobei beliebig auch bedeuten kann, dass er gar nicht vorkommt. Es wäre dann ein normaler SELECT.



Aufgabe 11.11: Warum konnte beim letzten SELECT kein NATURAL JOIN verwendet werden?

Aufgabe 11.12: Erweitern Sie diesen SELECT um die Positionen, in denen der Artikel vorkommt.

Aufgabe 11.13: Erweitern Sie das Ergebnis der letzten Aufgabe um die Daten der Bestellung.

Aufgabe 11.14: Erweitern Sie das Ergebnis der letzten Aufgabe um die Daten des Kunden.

Aufgabe 11.15: Erweitern Sie das Ergebnis der letzten Aufgabe um die Rechnungsadresse des Kunden.

11.4 Es muss nicht immer heiße Liebe sein: OUTER JOIN

Wir wollen die Kunden mit ihren ggf. vorhandenen Bestellungen wissen:

```
1 mysql> SELECT
2     -> k.kunde_id, k.nachname, k.vorname, b.datum
3     -> FROM
4     ->  bestellung b INNER JOIN kunde k USING (kunde_id)
5     -> ORDER BY
6     ->  k.nachname, k.vorname;
7 +-----+-----+-----+-----+
8 | kunde_id | nachname | vorname | datum |
9 +-----+-----+-----+-----+
10 |         2 | Beutlin  | Frodo   | 2012-03-23 16:11:00 |
11 |         1 | Gamschie | Samweis | 2012-03-24 17:41:00 |
12 +-----+-----+-----+-----+
```

Stichwortverzeichnis

- ||, 386
- () , 138
- *, 156, 377
- +, 377
- , 377
- ~, 159
- .NET, 6
- /, 377
- ;;, 65
- <, 383
- <=, 383
- <=>, 383
- <>, 383
- =, 383
- >, 383
- >=, 383
- %, 377
- &&, 385
- !, 384
- !=, 383

- Abhängigkeit, mehrwertige, 25
- ABS(), 377
- Abweisende Schleife, 315
- ACID, 293
- ACOS(), 377
- AFTER, 125
- Aggregatfunktion, 207
- Alias, 156, 203
- ALL, 227, 228
- ALL(), 233
- ALTER DATABASE, 119, 387
- ALTER EVENT, 344, 387
- ALTER FUNCTION, 388
- ALTER LOGFILE GROUP, 387
- ALTER PROCEDURE, 388
- ALTER SCHEMA, 119, 387
- ALTER SERVER, 388
- ALTER TABLE, 123, 388
 - ... ADD, 124, 142
 - ... ADD FOREIGN KEY, 170
 - ... ADD INDEX, 170
 - ... ADD PRIMARY KEY, 170
 - ... DISABLE KEYS, 100
 - ... DROP, 131
 - ... DROP FOREIGN KEY, 169
 - ... DROP INDEX, 169
 - ... DROP PRIMARY KEY, 169
 - ... ENABLE KEYS, 100
 - ... MODIFY, 126
 - ... RENAME, 124
- ALTER TABLESPACE, 390
- ALTER VIEW, 275, 390
- AND, 385
- Änderung, kaskadierende, 34, 84
- Änderungsweitergabe, 34, 84
- Annehmende Schleife, 315
- ANSI, 62
- Ansicht, 4, 267
 - Projektions-, 279
 - Selektions-, 276
 - veränderbare, 281
 - Verbund-, 278
- Antivalenz, 386
- ANY(), 231, 233
- ApacheFriends, 55
- API, 5
- ARCHIVE, 82
- Aria, 82
- Artikelverwaltung, 42
- AS IDENTITY, 72
- ASC, 158
- ASIN(), 377
- ATAN(), 377
- ATAN2(), 377
- Atomare Tabelle, 36
- Atomarität, 293
- Atomicity, 293

- Attribut, 16
- Aufzählung, 74
- AUTO_INCREMENT, 72, 146, 376
- AUTOCOMMIT, 294
- AVG(), 208, 380
- AVG(DISTINCT), 380

- B-Baum, 89
- Bank, 41
- Bankverbindung, 41
- BCNF, 39
- BDB, 82
- Bedingung, 135
- BEGIN ... END, 305
- Benutzerrecht, 350
- Benutzerspezifikation, 402
- Bestellwesen, 43
- BETWEEN, 383
- Bewegungsdaten, 140, 191
- BIGINT, 369
- Binäres Logging, 356
- Binäre Mitschrift, 356
- BINARY, 137, 376
- BIT_AND(), 380
- BIT_OR(), 381
- BIT_XOR(), 381
- BLACKHOLE, 82
- BLOB, 112, 375
- BOOL, 369
- Breitenabdeckung, 371
- BSI, 353
- Bücherei, 31
- BULK INSERT, 103

- C, 6
- C++, 6
- C#, 6
- Cache, 8
- CALL, 398
- CASCADE, 66, 83, 134
- CASE, 255, 312
- Case insensitive, 423
- Case sensitive, 423
- Cassandra, 82
- CD-Sammlung, 31
- CEILING(), 378
- CHAR, 371
- CHARACTER SET, 66, 160
- CHECK, 376
- CHECK TABLE, 274
- Chen, Jolly, 9
- Chen-Notation, 22

- ci, 423
- CLOSE, 323
- Clown-Agentur, 31
- Cluster, 356
- COBOL Data Division, 12
- Codd, Dr. E. F., 11
- Codepage 850, 66
- COLLATE, 68
- Compilezeit, 152
- CONCAT(), 142, 216
- Condition, 135
- conditional comment, 120
- CONNECT, 82
- CONNECT_TIMEOUT, 306
- Connection-Pool, 7
- Consistency, 293
- CONSTRAINT, 82
- constraint check, 5
- CONV(), 378
- COPY, 103
- COS(), 378
- COT(), 378
- COUNT(), 208, 381
- COUNT(*), 208, 381
- COUNT(DISTINCT), 208, 381
- cp850, 67
- CRC32(), 378
- CREATE DATABASE, 390
- CREATE EVENT, 341, 390
- CREATE FUNCTION, 331, 391
- CREATE INDEX, 92, 391
- CREATE LOGFILE GROUP, 391
- CREATE PROCEDURE, 304, 392
- CREATE SCHEMA, 64, 392
- CREATE SERVER, 392
- CREATE TABLE, 71, 84, 86, 393
- ... LIKE, 396
- ... SELECT, 395
- CREATE TABLESPACE, 396
- CREATE TEMPORARY TABLE, 189, 220
- CREATE TRIGGER, 334, 396
- CREATE USER, 352, 402
- CREATE VIEW, 276, 396
- CROSS JOIN, 182
- Crowfoot, 23
- cs, 423
- CSV, 82
- CSV-Format, 102
- CURSOR, 323

- Data Control Language, 61
- Data Definition Language, 61

- Data Manipulation Language, 61
- DATE, 372
- DATE(), 131
- DATE_FORMAT(), 374
- Datensystem, 8
- Daten
 - Bewegungs-, 140, 191
 - redundante, 204
 - Stamm-, 140, 191
- Datenbank, 3, 13
 - Abgrenzung, 11
 - anlegen, 64
 - hierarchische, 12
 - löschen, 65
 - objektorientierte, 13
 - relationale, 11
- Datenbankmanagementsystem, 5
- Datenbanksystem, 3
 - Client/Server-, 7
- Datenfeld, 16
- Datenflussdiagramm, 22
- Datensatz, 16
- Datenschutz, 5, 278
- Datensicherheit, 5
- Datentyp, 70
- Datenverzeichnis, 58
- DATETIME, 372
- Dauerhaftigkeit, 293
- DBMS, 5
- DCL, 61
- DDL, 61, 387
- Deadlock, 301, 302
- DECIMAL, 76, 370, 405
- DECLARE, 305
 - ... FOR CURSOR, 323
 - CONTINUE, 323
 - EXIT, 323
- Dedicated Machine, 51
- DEFAULT, 376
- DEGREES(), 378
- Deklarative Programmiersprache, 61
- DELETE, 144, 398
 - ... IGNORE, 147
 - ... LOW_PRIORITY, 146
 - ... QUICK, 147
- deleted, 33
- DELIMITER, 305
- DESC, 158
- DESCRIBE, 74, 124
- Deutscher Brauereiverband, 31
- Deutscher Wetterdienst, 25
- Developer Server, 49
- Development Machine, 51
- Differenzmenge, 250, 253
- DIN, 62
- DIN 5007-1, 68
- DIN 5007-2, 68
- DIN 66001, 22
- DIN 66261, 22
- dirty read, 294
- DISABLE KEYS, 100
- Disjunktion, 386
- DISTINCT, 167
- DIV, 377
- DML, 61, 398
- DO, 399
- Domäne, 15, 16
- DOUBLE, 76, 370, 405
- DOUBLE PRECISION, 370
- Drei-Schichten-Architektur, 16, 77
- Dritte Normalform, 38
- DROP DATABASE, 121, 396
- DROP EVENT, 344, 397
- DROP FUNCTION, 332, 397
- DROP INDEX, 100, 397
- DROP LOGFILE GROUP, 397
- DROP PROCEDURE, 308, 397
- DROP SCHEMA, 121, 397
- DROP SERVER, 397
- DROP TABLE, 132, 397
- DROP TABLESPACE, 398
- DROP TRIGGER, 334, 398
- DROP USER, 350, 353, 403
- DROP VIEW, 273, 398
- Dublette, 96
- Durability, 293
- Eigenschaft, 16
- 1:1-Verknüpfung, 25
- 1:n-Verknüpfung
 - Definition, 27
 - identifizierende, 27
- ENABLE KEYS, 100
- Engine, 5, 8, 80
 - ARCHIVE, 82
 - Aria, 82
 - BDB, 82
 - BLACKHOLE, 82
 - Cassandra, 82
 - CONNECT, 82
 - CSV, 82
 - EXAMPLE, 82
 - FEDERATED, 82
 - FederatedX, 82

- InnoDB, 82
- MEMORY, 82
- MERGE, 82
- MyISAM, 82
- OQGRAPH, 82
- XtraDB, 82
- Entität, 16
- Entitätentyp, 16
 - schwacher, 18
 - starker, 18
- Entity, 16
- Entity Relationship Model, 22
- Entitytype, 16
- ENUM, 74, 369
- EQUI JOIN, 193
- ER-Modell, 22
- Eratosthenes, 318
- Ereignis, 5, 341
- ERM, 22
- Erste Normalform, 36
- Event, 341
- EXAMPLE, 82
- EXCEPT, 250, 251, 253
- Existenzabhängige Tabelle, 18
- Existenzunabhängige Tabelle, 18
- EXISTS, 235
- EXIT, 64
- EXP(), 378
- Experiment
 - DOUBLE vs. DECIMAL, 405
 - Einfügen mit Index, 414
 - Indexselektivität, 418
 - NULL vs. NOT NULL, 409
 - Rundungsfehler, 422
 - Sortierung, 419
 - Suchen, 411
- EXPLAIN, 164
 - EXTENDED, 242
- Exportieren
 - Binärdaten, 175
 - CSV-Daten, 174
- Fallunterscheidung, 255, 312
- FEDERATED, 82
- FederatedX, 82
- Feld, 16
- FETCH, 323
- FIRST, 125
- FIXED, 370
- FLOAT, 370
- FLOOR(), 154, 378
- FOREIGN KEY, 80, 376
- FORMAT(), 258, 378
- Formatierungszeichen, 374
 - Datum, 373
 - Uhrzeit, 373
- Fremdschlüssel, 20, 80
 - festlegen, 80
- FULL OUTER JOIN, 197
- Funktion, 332
- GENERATED, 72
- GENERATED ... AS IDENTITY, 376
- Geschlossene Schleife, 315
- GET_FORMAT(), 374
- GLOBAL, 88
- Globale Variable, 306
- GRANT, 353, 403
 - ALL PRIVILEGES, 354
 - CHARACTER SET, 354
 - COLLATION, 354
 - DOMAIN, 354
 - FUNCTION, 354
 - PROCEDURE, 354
 - PROXY, 403
 - REPLICATION SLAVE, 357
 - TABLE, 354
 - TRANSLATION, 354
 - WITH GRANT OPTION, 354
- GROUP BY, 210
- GROUP_CONCAT(), 381
- Hauptanweisung, 222
- HAVING, 214
- Herz, 151
- HEX(), 378
- Hierarchische Datenbank, 12
- Hilfstabelle, 30
- Hotel-Software, 32
- iconv, 67
- IDEFIX-Notation, 22
- Identifizierende 1:n-Verknüpfung, 27
- IE, 310
- IF EXISTS, 66
- IF NOT EXISTS, 65
- IGNORE, 106
- Imperative Programmiersprache, 61
- Importieren
 - Binärdaten, 112
 - CSV-Daten, 103
 - XML-Daten, 324
- IN(), 226, 233, 384
- Index, 4, 89, 411, 414, 418, 419

- anlegen, 92
- automatisch, 91
- Dublette, 96
- löschen, 100
- Schlüsseleigenschaft, 95
- Indexselektivität, 98, 418
- Informix, 9
- INHERITS, 87
- INNER JOIN, 183
- InnoDB, 82
- INSERT, 350
- INSERT INTO
 - ... SELECT, 115, 399
 - ... SET, 110, 399
 - ... VALUES, 108, 399
- INT, 369
- Integrität, referentielle, 33
- Interpreter, 5
- INTERSECT, 248, 253
- IS FALSE, 384
- IS NOT NULL, 384
- IS NULL, 384
- IS TRUE, 384
- IS UNKNOWN, 384
- ISAM, 6
- ISO, 62
- ISO/IEC 5807, 22
- ISO/IEC 8859-1, 66
- ISO/IEC 8859-2, 67
- ISO/IEC 8859-9, 67
- ISO/IEC 8859-13, 67
- ISO/IEC 9075:1999, 62
- ISO/IEC 9075:2011, 62
- ISO/IEC 9075, 374
- Isolation, 293
- Item, 16
- ITERATE, 316

- JDBC, 6
- JOIN
 - ... ON, 183
 - ... USING, 188
 - CROSS, 182
 - EQUI, 193
 - INNER, 183
 - NATURAL, 188
 - OUTER, 197
 - FULL, 197, 200
 - LEFT, 197
 - RIGHT, 197
 - rekursiv, 203
 - SELF, 202

- Kalender
 - gregorianisch, 372
 - julianisch, 372
 - proleptischer gregorianischer, 372
- Kardinalität, 24
- Kartesisches Produkt, 181
- Kaskadierende Änderung, 34, 84
- Kaskadierendes Löschen, 33, 83
- Klammern, 138
- Klasse, 16
- Kommentar
 - bedingter, 120
 - einzeilig, 159
- Konjunktion, 385
- Konnektor, 7
- Konsistenz, 293
- Konstante, 152
- Kopf-/Fußgesteuerte Schleife, 315
- Korrelierende Unterabfrage, 222
- Krähenfuß-Notation, 23
- Kreuzprodukt, 182
- Kunde, 41
- Kundenverwaltung, 41

- LAST_INSERT_ID(), 290
- latin1, 67
- Laufzeit, 152
- LE, 423
- LEAVE, 316
- LEFT OUTER JOIN, 197
- LIKE, 86, 384
- LIMIT, 171
- Listenunterabfragen, 226
- little endian, 423
- LN(), 379
- LOAD DATA INFILE, 103, 105, 400
- LOAD XML, 325, 400
- LOCAL, 87
- lock, 286
- LOCK TABLES, 287
- locking, 286
- LOG(), 379
- LOG10(), 379
- LOG2(), 379
- Logging, binäres, 356
- lokale Variable, 305
- LONGBLOB, 375
- LONGTEXT, 371
- LOOP-Schleife, 316
- Löschen, kaskadierendes, 33, 83
- Löschkennzeichen, 33
- Löschweitergabe, 33, 83

- lost update, 286
- MariaDB, 60
- MariaDB-Client, 63
- Martin-Notation, 23
- Master, 356
- MASTER_HEARTBEAT_PERIOD, 359
- MASTER_HOST, 358
- MASTER_LOG_FILE, 358
- MASTER_LOG_POS, 358
- MASTER_USER, 358
- Matrix, 16
- MAX(), 209, 381
- MEDIUMBLOB, 375
- MEDIUMINT, 369
- MEDIUMTEXT, 371
- Mehrwertige Abhängigkeit, 25
- MEMORY, 82
- Mengenverhältnis, 24
- MERGE, 82
- MERGED, 270
- MIN(), 209, 381
- Minimalität des Schlüssels, 17
- Minimumexistenz, 159
- Mitschrift, binäre, 356
- MOD, 377
- MOD(), 379
- Modellierung, 22
- Monolithische Anwendung, 77
- MONTH(), 216
- MONTHNAME(), 216
- MyISAM, 82
- MySQL, 6, 59
- mysql -e, 175
- MySQL Workbench, 23
- MySQL-Client, 63
- MySQL-Query Browser, 63
- mysql_error.log, 359
- mysqladmin, 59
- mysqldump, 347
- n:m-Verknüpfung, 29
- Nassi-Shneidermann-Diagramm, 22
- NATURAL JOIN, 188
- Negation, 384
- NEW, 334
- Nicht identifizierende 1:n-Verknüpfung, 27
- Nicht korrelierende Unterfrage, 222
- NO ACTION, 83
- Normalform, 34
 - Boyce-Codd, 39
 - erste, 36
 - zweite, 38
 - dritte, 38
 - vierte, 39
 - fünfte, 39
- Normalisierung, 37
- NOT, 384
- NOT BETWEEN, 384
- NOT FOUND, 323
- NOT IN, 384
- NOT IN(), 251
- NOT NULL, 77, 376, 409
- Notation
 - Chen, 22
 - Crowfoot, 23
 - IDEFIX, 22
 - Krähenfuß, 23
 - Martin, 23
 - UML, 23
- NULL, 77, 376, 409
- NUMERIC, 370
- Nummernkreis, 74
- Oberanweisung, 222
- Objekt, 16
- Objektorientierte Datenbank, 13
- Objekttyp, 403
- ODBC, 6
- Offene Schleife, 315
- OLD, 334
- Online-Shop, 41
 - Tabelle adresse, 41
 - Tabelle artikel, 42
 - Tabelle artikel_nm_lieferant, 88
 - Tabelle artikel_nm_warengruppe, 108
 - Tabelle bank, 41
 - Tabelle bankverbindung, 41
 - Tabelle bestellung, 43
 - Tabelle bild, 112
 - Tabelle kunde, 41
 - Tabelle lagerbestand, 210
 - Tabelle lieferant, 42
 - Tabelle position_bestellung, 43
 - Tabelle position_rechnung, 44
 - Tabelle rechnung, 43
 - Tabelle warengruppe, 42
- OPEN, 323
- Operatorenpriorität, 153
- Operatorenrangfolge, 153
- Optimierer, 5, 8
- Option, 403
- OQGRAPH, 82
- OR, 386

- Oracle, 6
- ORDER BY, 157
- Ordnung, 159
- Orgienjoin, 182
- OUTER JOIN, 197

- page lock, 286
- Parser, 8
- PASSWORD(), 352
- Performancemessung, 405, 411, 414, 419
- Perl, 6
- PHP, 6
- PI(), 379
- Platzhalter, 156
- Plausibilisierung, 16
- Plausibilitätsüberprüfung, 309
- PostgreSQL, 9, 60
- POSTQUEL, 9
- POWER(), 379
- Primärschlüssel, 19
- PRIMARY KEY, 376
- Privileg, 350
- Privilegtiefe, 404
- Programmablaufplan, 22
- Programmierschnittstelle, 5
- Programmiersprache
 - deklarative, 61
 - imperative, 61
- Programmverzeichnis, 57
- Projektionsansicht, 279
- Property, 16
- Prozedur, 4
- Puh, 124
- Python, 6

- RADIANS(), 379
- RAND(), 154, 379
- Randbedingungsprüfer, 5
- RDBMS, 6
- REAL, 370
- Record, 16
- Recordset, 16
- Redundante Daten, 204
- Redundanz, 20, 34
- REFERENCES, 80
- referentielle Integrität, 33, 41, 83
- Referenz, 20
- Relation, 11, 16
- Relationale Datenbank, 11
- relationales Datenbankmanagementsystem, 6
- RENAME TABLE, 398
- RENAME USER, 403

- REPLACE, 106
- Replikation, 356
 - Master, 356
 - Slave, 356
- RESTRICT, 66, 83, 85, 134
- REVOKE, 350, 355, 403
 - ALL, 404
 - ALL PRIVILEGES, 356
 - PROXY, 404
- RIGHT OUTER JOIN, 197
- ROLLBACK, 295
- ROUND(), 142, 379
- row lock, 287
- Rundungsfehler, 422

- Sakila, 6
- Satzkennzeichen, 12
- Schema, 16
- Schlüssel
 - Definition, 17
 - Fremd-, 20
 - Kandidat-, 19
 - Minimalität des, 17
 - Primär-, 19
 - Sekundär-, 19
- Schleife, 315
 - abweisende, 315
 - annehmende, 315
 - fußgesteuerte, 315
 - geschlossen, 315
 - kopfgesteuerte, 315
 - offene, 315
- Schnittmenge, 248, 253
- Schwache Tabelle, 18
- Seitensperre, 286
- Sekundärschlüssel, 19
- SELECT, 151, 401
 - ... INTO, 155, 172, 307
 - ... INTO OUTFILE, 174
- Selektionsansicht, 276
- SELF JOIN, 201, 202
- Semikolon, 65
- Seq_in_index, 94
- Server Machine, 51
- server-id, 357
- session variable, 306
- SET, 307, 369
 - SET GLOBAL, 343
 - SET NAMES, 160
 - SET NULL, 83
 - SET PASSWORD, 404
- SHOW

- CHARACTER SET, 67
- COLLATION, 68
- CREATE DATABASE, 120
- CREATE SCHEMA, 120
- CREATE TABLE, 81, 146
- DATABASES, 69
- EVENTS, 342
- FULL TABLES, 269
- GRANTS FOR, 352
- INDEX, 91
- MASTER STATUS, 358
- PROCEDURE STATUS, 308
- SCHEMAS, 69
- TABLES, 73
- TRIGGERS, 339
- VARIABLES, 343
- VARIABLES LIKE, 64
- VIEWS (work around), 269
- WARNINGS, 65, 100, 104
- Sicherheitseinstellung, 52
- Sieb des Eratosthenes, 318
- SIGN(), 380
- SIN, 380
- Single Responsibility Principle, 77
- Sitzung, 7
- Sitzungsvariablen, 306
- Sitzungsverwaltung, 5
- Skalarunterabfrage, 222, 223
- Slave, 356
- SMALLINT, 369
- Sortierreihenfolge, 68
- Sortierung
 - deutsch, 424
 - zuweisen, 68
- SP_RENAME, 124
- Spalte
 - Auswahl einer, 156
 - Definition, 15
 - Spezifikation einer, 71
- Sprunglogik, 385
- SQL, 61
- SQL HANDLER, 323
- SQL-Schnittstelle, 7
- SQL_NO_CACHE, 408
- SQL EXCEPTION, 323
- SQLWARNING, 323
- SQRT(), 380
- SRP, 77
- Stammdaten, 140, 191
- Standardwert, 375
- Starke Tabelle, 18
- START SLAVE, 358
- START TRANSACTION, 293
- STD(), 381
- STDDEV_POP(), 382
- STDDEV_SAMP(), 382
- Stonebraker, Michael, 9
- STOP SLAVE, 358
- Storage Engine, 5, 8
- strict-Modus, 351
- String, 369
- Struktogramm, 22
- Stundenplan-Software, 32
- SUBSELECT, 221
- SUBSTRING(), 216
- Suchpfad, 59
- SUM(), 209, 382
- SUM(DISTINCT), 382
- Sun Microsystems, 6

- Tabelle, 4, 16
 - anlegen, 70
 - atomar, 36
 - existenzabhängige, 18
 - existenzunabhängige, 18
 - herleiten, 86
 - schwach, 18
 - starke, 18
 - teilfunktional, 37
 - temporär, 87, 218
 - transitiv, 38
 - vollfunktional, 37
 - wiederholungsgruppenfrei, 35
- Tabellenreferenzen, 401
- Tabellensperre, 286
- Tabellenunterabfragen, 234
- table lock, 286
- TAN(), 380
- Tcl, 6
- Teilfunktionale Tabelle, 37
- Temporäre Tabelle, 4, 87, 218
- TEMPORARY, 87
- TEMPTABLE, 270
- TEXT, 371
- Tiefenabdeckung, 371
- TIME, 372
- TIME(), 163
- TIME_FORMAT(), 374
- Timeout, 5, 64
- TIMESTAMP, 372
- TINYBLOB, 375
- TINYINT, 369
- TINYTEXT, 371
- Transaktion, 293

- Transaktionsmanagement, 5
- Transitive Tabelle, 38
- Transitivität, 159
- Trennzeichen, 102
- Trichotomie, 159
- Trigger, 5, 333
- TRUNCATE, 147, 398
- TRUNCATE(), 380
- Tupel, 16
- Twebaze, Ambrose, 6

- Übergabeparameter, 306
- UML-Notation, 23
- UNDER, 86
- Unicode, 66
- UNION, 245, 253, 402
- UNIQUE, 376
- UNKNOWN, 135
- UNLOCK TABLES, 287
- UNSIGNED, 72, 376
- Unterabfrage, 221
 - korrelierende, 222
 - Listen-, 226
 - nicht korrelierende, 222
 - skalar, 222, 223
 - Tabellen-, 234
- UPDATE, 140, 402
 - ... IGNORE, 143
 - ... LOW_PRIORITY, 143
- USE, 72
- USING, 188
- utf8, 66, 67
- utf16, 66, 67
- UTF16LE, 423
- utf32, 66, 67

- VAR_POP(), 382
- VAR_SAMP(), 382
- VARCHAR, 72, 371
- Variable, 155, 172
 - global, 306
 - lokal, 305
 - Sitzungs-, 306
- VARIANCE(), 382
- Veränderbare Ansicht, 281
- Verbinder, 7
- Verbundansicht, 278
- Vereinigung, 245, 253
- Vererbung, 86

- Verknüpfung, 20
 - 1:1, Definition, 25
 - 1:n
 - Definition, 27
 - identifizierende, 27
 - n:m, Definition, 29
- Verletzte referentielle Integrität, 33
- Verschlüsselung, 404
- Verzeichnis
 - Daten, 58
 - Programm, 57
 - XAMPP, 56
- Verzweigung, 309
- VIEW, 267
- Vollfunktionale Tabelle, 37
- Vorbelegungen, 375

- Wartungsstabilität, 24
- Wertebereich, 15
- WHERE-Klausel, 135, 136, 215
- WHILE-Schleife, 318
- Widenius, Michael, 6
- Wiederholungsgruppe, 30, 35
- Wiederholungsgruppenfreiheit, 35
- Windows 10, 55
- Windows Service, 52
- WITH RECURSIVE, 204

- XAMPP, 55, 60
- XAMPP-Verzeichnis, 56
- XML-Format, 13
- XtraDB, 82

- YEAR, 372
- YEAR(), 216
- Yu, Anrew, 9

- Zeichenketten, 109
- Zeichensatz, 66
 - deutsch, 423
 - zuweisen, 66
- Zeile
 - Auswahl einer, 157
 - Definition, 16
- Zeilensperre, 287
- Zeilenumbruch, 102
- Zufallszahlen, 154
- Zweite Normalform, 38
- Zwischenspeicher, 8