

Mark Maslakowski



MySQL

Schritt für Schritt eine
Datenbank aufbauen

MySQL administrieren,
optimieren, absichern

Daten importieren
und exportieren



MySQL-Server für Windows
und Unix/Linux sowie das komplette
Buch im HTML-Format

Widmung

In herzlicher Erinnerung möchte ich dieses Buch meinem Vater widmen, der diese Welt früh verlassen hat, aber denjenigen, mit denen er in Berührung stand, viele Erinnerungen und Lehren hinterlassen hat.

– Mark

Mark Maslakowski

Deutsche Übersetzung:

Wolfgang Drews

Thorsten Wendt

MySQL

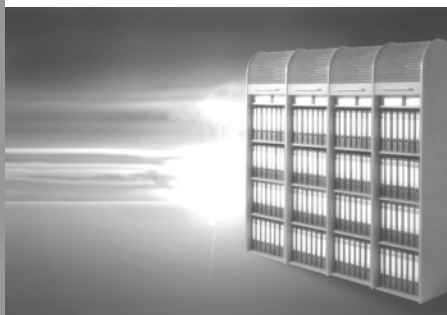
in 21 Tagen

Bitte beachten Sie:

Der originalen Printversion liegt eine CD-ROM bei.

In der vorliegenden elektronischen Version ist die Lieferung einer CD-ROM nicht enthalten.

Alle Hinweise und alle Verweise auf die CD-ROM sind ungültig.



Markt+Technik Verlag

Die Deutsche Bibliothek – CIP-Einheitsaufnahme

Ein Titeldatensatz für diese Publikation ist bei
Der Deutschen Bibliothek erhältlich.

Die Informationen in diesem Produkt werden ohne Rücksicht auf einen
eventuellen Patentschutz veröffentlicht.
Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt.
Bei der Zusammenstellung von Texten und Abbildungen wurde mit größter
Sorgfalt vorgegangen.
Trotzdem können Fehler nicht vollständig ausgeschlossen werden.
Verlag, Herausgeber und Autoren können für fehlerhafte Angaben
und deren Folgen weder eine juristische Verantwortung noch
irgendeine Haftung übernehmen.
Für Verbesserungsvorschläge und Hinweise auf Fehler sind Verlag und
Herausgeber dankbar.

Autorisierte Übersetzung der amerikanischen Originalausgabe:
MySQL in 21 Days © 2000 by Sams Publishing

Alle Rechte vorbehalten, auch die der fotomechanischen
Wiedergabe und der Speicherung in elektronischen Medien.
Die gewerbliche Nutzung der in diesem Produkt gezeigten
Modelle und Arbeiten ist nicht zulässig.

Fast alle Hardware- und Software-Bezeichnungen, die in diesem Buch
erwähnt werden, sind gleichzeitig auch eingetragene Warenzeichen
oder sollten als solche betrachtet werden.

Umwelthinweis:
Dieses Buch wurde auf chlorfrei gebleichtem Papier gedruckt.
Die Einschrumpffolie – zum Schutz vor Verschmutzung – ist aus
umweltverträglichem und recyclingfähigem PE-Material.

10 9 8 7 6 5 4 3 2 1

04 03 02 01

ISBN 3-8272-5850-2

© 2001 by Markt+Technik Verlag,
ein Imprint der Pearson Education Deutschland GmbH.
Martin-Kollar-Straße 10–12, D–81829 München/Germany
Alle Rechte vorbehalten
Übersetzung: Wolfgang Drews, Thorsten Wendt, Trier
Fachliches Lektorat: Wolfgang Drews, Trier, Christian Wenz, München
Lektorat: Boris Karnikowski, bkarnikowski@pearson.de
Herstellung: Claudia Bäurle, cbaurle@pearson.de
Satz: reemers publishing services gmbh, Krefeld
Einbandgestaltung: Heinz H. Rauner, Gmund
Druck und Verarbeitung: Bercker, Kevelaer
Printed in Germany

Inhaltsverzeichnis

	Über den Autor	13
	Danksagungen	15
	Einleitung	17
	Wie ist dieses Buch aufgebaut?	18
	Über dieses Buch	20
	Wer sollte dieses Buch lesen?	20
	Wie ist dieses Buch strukturiert?	21
	Konventionen	21
	Woche 1 im Überblick	23
Tag 1	Was ist MySQL?	25
	Die Geschichte von MySQL	26
	Was ist ein Unternehmen?	28
	Was ist eine relationale Datenbank?	30
	Das Client/Server-Paradigma	30
	Merkmale von MySQL	32
	Zusammenfassung	33
	Fragen und Antworten	34
	Übungen	34
Tag 2	Los geht's	35
	Lizenzierung	36
	Der Download von MySQL	36
	Die Linux-Installation	37
	Die Windows-Installation	38
	Datei-Inhalt	39
	MySQL unter Linux benutzen	42
	MySQL unter Windows benutzen	43
	MySQL benutzen – die Kommandozeile	44



	Zusammenfassung	48
	Fragen und Antworten	48
	Übungen	49
Tag 3	Das Entwerfen Ihrer ersten Datenbank	51
	Der Design-Prozess	52
	Die letzte Phase – Erstellen einer Datenbank	61
	Zusammenfassung	68
	Fragen und Antworten	69
	Übungen	70
Tag 4	Das Erstellen Ihrer ersten Datenbank	71
	Die Befehle CREATE und DROP	72
	Benutzer hinzufügen	78
	Das Erstellen der Meet_A_Geek-Datenbank	80
	Zusammenfassung	80
	Fragen und Antworten	81
	Übungen	82
Tag 5	Datennormalisierung	83
	Was ist Normalisierung?	84
	Grade der Normalisierung	85
	Welchen Grad der Normalisierung soll man wählen?	90
	Zusammenfassung	92
	Fragen und Antworten	92
	Übungen	93
Tag 6	Das Hinzufügen von Tabellen, Spalten und Indizes	95
	Das Erstellen von Tabellen	96
	Das Ändern bestehender Tabellen	101
	Das Hinzufügen/Löschen von Tabellen und Spalten	102
	Der Gebrauch von Indizes	104
	Zusammenfassung	107
	Fragen und Antworten	107
	Übungen	108

Tag 7	MySQL-Datentypen	109
	Die von MySQL unterstützten Typen	110
	Nummerische Typen	111
	Zeichen- oder Zeichenketten-Datentypen	115
	Vermischte Typen	117
	Zusätzliche Spaltenattribute	118
	Zusammenfassung	120
	Fragen und Antworten	120
	Übungen	121
	Woche 1 im Rückblick	123
	Woche 2 im Überblick	125
Tag 8	Das Füllen der Datenbank mit Daten	127
	Methoden des Datenimports	128
	MySQL-Import-Tools	129
	Datenbank-Import-/Exportwerkzeuge	144
	Methoden des Datenexports	145
	Zusammenfassung	150
	Fragen und Antworten	151
	Übungen	152
Tag 9	Datenbankabfragen	153
	Was ist SQL und was sind seine Wurzeln?	154
	Datenbearbeitung mit SQL	158
	Arbeiten mit Joins	167
	Aggregat-Funktionen	171
	Sortieren	174
	Zusammenfassung	175
	Fragen und Antworten	176
	Übungen	177
Tag 10	Interne MySQL-Funktionen	179
	Programmieren auf Server-Ebene	180
	Wie kann MySQL Ihnen helfen?	183
	Die grundlegenden MySQL-Funktionen	183



	Zusammenfassung	193
	Fragen und Antworten	193
	Übungen	194
Tag 11	Sperren und Schlüssel in MySQL	195
	Was ist eine Sperre?	196
	Der Gebrauch von Sperren in MySQL	199
	Der Gebrauch von LOCK TABLES	201
	Was ist ein Schlüssel?	203
	Schlüssel in MySQL	205
	Primärschlüssel	211
	Sinnvolle Auswahl der Schlüssel	213
	Zusammenfassung	216
	Fragen und Antworten	217
	Übungen	218
Tag 12	Wie komme ich an die Daten ran? – DBIs	219
	Verbindung aufnehmen	220
	Verbindungsanforderungen	222
	An die Daten rankommen	223
	Zusammenfassung	226
	Fragen und Antworten	227
	Übungen	227
Tag 13	Wie benutze ich MyODBC?	229
	Woher bekomme ich einen Treiber?	230
	Einrichten des Datenquellennamens (DSN)	233
	Active Server Pages und VBScript	237
	ActiveX Data Objects (ADO)	249
	Die Meet_A_Geek-Datenbank im Web	258
	Zusammenfassung	267
	Fragen und Antworten	267
	Übungen	268
Tag 14	Das Perl-Datenbank-Interface	269
	Installation der Perl DBI und der MySQL DBD	270
	Ihr erstes Perl-DBI/DBD-Skript	275
	CGI, Perl, DBI und MySQL	281

	Zusammenfassung	289
	Fragen und Antworten	290
	Übungen	291
	Woche 2 im Rückblick	293
	Woche 3 im Überblick	295
Tag 15	MySQL und PHP.	297
	Was ist PHP?	298
	Ein schneller Einstieg in PHP	302
	PHP trifft MySQL	315
	Zusammenfassung	322
	Fragen & Antworten	322
	Übungen	323
Tag 16	MySQL und die Zeit	325
	Die Handhabung des Datums in MySQL	326
	Datumsbezogene Datentypen	327
	Interne MySQL-Datums-Funktionen.	331
	Zusammenfassung	344
	Fragen und Antworten	345
	Übungen	346
Tag 17	Datenbanksicherheit in MySQL	347
	Wie Sicherheit in MySQL implementiert ist	348
	Kontrollebenen	353
	Vergabe der Benutzerrechte	356
	Wie Sie geeignete Sicherheitsmaßnahmen ergreifen	364
	Zusammenfassung	365
	Fragen und Antworten	366
	Übungen	367
Tag 18	MySQL im Vergleich	369
	Schnelligkeit muss sein	370
	Datenbanken und Transaktionen	371
	SQL-bedingte Beschränkungen (Constraints)	381
	Zusammenfassung	382



	Fragen und Antworten	383
	Übungen	384
Tag 19	MySQL administrieren	385
	Datensicherung	386
	Protokollieren von Transaktionen	394
	Verschieben einer MySQL-Datenbank	396
	Systemeigene Funktionen von MySQL	397
	Datenbank-Management mit mysqladmin	401
	Reparieren einer beschädigten Datenbank	407
	Zusammenfassung	412
	Fragen und Antworten	413
	Übungen	414
Tag 20	MySQL optimieren	415
	Performance-Tuning	416
	Erstellen von besseren SQL-Anweisungen	421
	Erstellen besserer WHERE-Klauseln	423
	Ungenutzten Speicherplatz freigeben	423
	Kompilieren und Komprimieren	426
	Zusammenfassung	426
	Fragen und Antworten	427
	Übungen	428
Tag 21	Ein Projekt wird verwirklicht	429
	Erstellen der Meet_A_Geek-Datenbank	430
	Importieren existierender Daten	437
	Sicherheitsaspekte	438
	Erstellen der Webseiten und Scripts	438
	Zusammenfassung	469
	Fragen und Antworten	469
	Übungen	470
	Woche 3 im Rückblick	471

Anhang A	SQL-Syntax und Befehle	473
Anhang B	Aktuelle MySQL-Funktionen	487
Anhang C	Lösungen zu den Übungen	527
	Tag 1	528
	Tag 2	528
	Tag 3	529
	Tag 4	530
	Tag 5	531
	Tag 6	531
	Tag 7	533
	Tag 8	533
	Tag 9	534
	Tag 10	536
	Tag 11	536
	Tag 12	537
	Tag 13	538
	Tag 14	539
	Tag 15	539
	Tag 16	540
	Tag 17	541
	Tag 18	542
	Tag 19	542
	Tag 20	543
	Tag 21	544
	Stichwortverzeichnis	545

Über den Autor

Mark Maslakowski arbeitet als Berater für Thor Systems Inc., eine Firma, die sich auf System-Integration spezialisiert hat. Als Senior Solutions Developer ist er zuständig für die Erarbeitung von Konzepten, die den Bedürfnissen der Kunden entsprechen – egal für welches Betriebssystem. Mark war an der Entwicklung umfangreicher, vielschichtiger Applikationen für mehrere Großunternehmen beteiligt. Seine Laufbahn begann er in der U.S. Army. Seitdem ist er unentwegt und überall beschäftigt. Er war vielseitig tätig, so zum Beispiel als Datenbank-Administrator, leitender Entwickler und leitender Systemanalytiker. Mark ist erreichbar unter markm@thorinc.com.

Danksagungen

In erster Linie möchte ich Ken Robertson danken. Er hat mir die Zeit, das Verständnis und die Möglichkeit zur Verwirklichung dieses Buches gegeben.

Ferner möchte ich Carla Maslakowski danken. Ohne ihre Motivation und ihre Richtungsweisung hätte ich nicht diese Laufbahn einschlagen können.

Weiterhin möchte ich David Smith von Cornerstone Information Systems danken. Er gab mir eine Chance, als keiner mir eine Chance geben wollte.

Ich danke allen meinen Freunden im Club, die mir geholfen haben, Dampf abzulassen, wenn es am nötigsten war.

Ich möchte an dieser Stelle auch meiner Familie danken, die immer für mich da war und meine schlechten Angewohnheiten ertragen hat.

Darüber hinaus möchte ich mich beim Team von Sams Publishing und besonders Shelley Johnston bedanken, dass sie mir dieses Unterfangen angenehm gestaltet haben.

– Mark Maslakowski

Einleitung

Schon vor dem Beginn des Computerzeitalters haben Menschen Datenbanken benutzt. Bevor es Computer gab, hätte Ihre Datenbank ein Rolodex mit Ihren wichtigsten Telefon-Nummern sein können oder ein Aktenschrank, der alle persönlichen Daten einer Firma beinhaltet. Heute sind Datenbanken computergestützt und befinden sich praktisch überall. Es gibt Datenbanken in allen Formen und Größen – von Desktop-Datenbanken für Ihre Plattensammlung bis hin zu internettauglichen Datenbanken, die von großen Firmen eingesetzt werden. Deswegen ist die Datenbank-Industrie genauso schnell gewachsen wie die übrige Computer-Industrie.

Bis vor kurzem mussten Sie für die meisten leistungsstarken Datenbanken Unsummen ausgeben. Diese konnten zwar alle Werkzeuge und Funktionen zur Verfügung stellen, die für ein Geschäft notwendig sind – doch zu einem sehr hohen Preis. Die meisten Firmen haben daher für eine billigere Datenbank auf einen Teil der Funktionalität verzichtet.

Darüber hinaus hat auch das Internet einen neuen Bedarf nach Datenbanken, die via Web zugänglich sind, hervorgerufen. Dieses Bedürfnis hat Software-Hersteller dazu veranlasst, Produkte herzustellen, die diese Technologie nutzen können. Auch hier spielt der Preis eine große Rolle. Denn diese Produkte sind in der Regel sehr teuer und hängen stark vom Betriebssystem ab, so dass nicht alle Internet Service Providers (ISPs) oder kleine Unternehmen von dieser Technologie profitieren können.

Springen Sie auf den Linux- und Open Source-Zug auf. Die Idee eines kostenlosen Betriebssystems und Quellcodes veränderte die Einstellung der Industrie zur bisherigen Geschäftspraxis. Nun da die Open Source-Revolution Fuß gefasst hat, strömen die Menschen in Massen herbei, um zu sehen, wie sie diese neuen, erschwinglichen Konzepte nutzen können.

MySQL ist Teil dieses Konzepts. Es wurde 1996 von den Entwicklern von TcX geschaffen, weil sie eine relationale Datenbank benötigten, die große Mengen an Daten auf relativ günstiger Hardware handhaben sollte. Doch sie konnten auf dem Markt nichts Passendes finden. Aus diesem Grund entwickelten sie ein eigenes Produkt.

MySQL ist die schnellste relationale Datenbank auf dem Markt. Sie übertrifft alle großen Datenbanken in nahezu sämtlichen Kategorien. MySQL besitzt annähernd die gleiche Funktionalität wie die großen Datenbanken, ist aber nicht so teuer. Das mag alles nach viel Hype und Feilbieterei klingen aber nachdem Sie nur kurze Zeit mit MySQL gearbeitet haben, werden Sie mir zustimmen.



Wenn MySQL so gut ist, weshalb hat es dann noch nicht die Aufmerksamkeit der Industrie auf sich gezogen? Die Antwort lautet: Bis 1999 waren Linux und die Open Source-Bewegung praktisch unbekannt.

MySQL läuft hauptsächlich auf UNIX-basierten Systemen – auch wenn es auf dem Markt für fast jedes Betriebssystem Portierungen gibt. MySQL wurde erst beachtet, als die UNIX-basierten Betriebssysteme zu erschwinglichen Preisen auf den Markt kamen und die Open Source-Bewegung allmählich an Popularität gewann.

Wegen des jüngsten Erfolges von Linux hat MySQL ebenfalls an Popularität gewonnen. Leider gibt es hierfür kaum ausführliche Dokumentationen. In diesem Augenblick kommt *MySQL in 21 Tagen* ins Spiel. Darin werden verschiedenen Komponenten von MySQL wie zum Beispiel Installation, Administration und Schnittstellen vorgestellt.

Wenn Sie mit dem Buch fertig sind, werden Sie mit diesen und anderen Komponenten bestens vertraut sein. Sie werden verstehen, weshalb MySQL eines der besten bis dato erhältlichen relationalen Datenbank Management Systeme (RDBMS) ist.

Wie ist dieses Buch aufgebaut?

Dieses Buch behandelt das relationale Datenbank-Management-System von MySQL in 21 Tagen – aufgliedert in drei Wochen. Jede Woche umfasst einen anderen Bereich von MySQL und baut auf den Grundlagen der vorherigen Tage auf.

In der ersten Woche werden Sie die Grundlagen der MySQL-Komponenten lernen:

- ▶ Tag 1 bietet eine Einführung in MySQL – was es ist und was es von anderen RDBMS seiner Größenordnung unterscheidet. Sie werden etwas über die verschiedenen Komponenten einer Datenbank lernen. Zudem werden Sie in die Welt der relationalen Datenbanken eingeführt.
- ▶ An Tag 2 lernen Sie, wie Sie MySQL unter Linux und Windows installieren.
- ▶ Tag 3 behandelt die Grundlagen des Datenbank-Designs. Sie werden eine Muster-Datenbank entwerfen, die das ganze Buch hindurch als Beispiel dient.
- ▶ An Tag 4 erstellen Sie Ihre erste MySQL-Datenbank. Sie werden die Befehle kennenlernen, mit denen Sie dies erreichen sowie einige MySQL-Utilities.
- ▶ Tag 5 behandelt das Thema Normalisierung – ein äußerst wichtiger Aspekt bei der Arbeit mit relationalen Datenbanken.
- ▶ An Tag 6 werden Sie Ihre Datenbank weiter entwickeln, indem Sie lernen, Tabellen, Spalten und Indizes hinzuzufügen. Diese geben Ihrem Design eine Struktur.
- ▶ Tag 7 befasst sich mit den verschiedenen Datentypen, die MySQL benutzt.



Die zweite Woche dient dazu, Ihnen den Umgang mit Ihrer Datenbank beizubringen. Sie werden verschiedene Methoden kennenlernen, um die in Ihrer MySQL-Datenbank gespeicherten Daten zu bearbeiten. Sie werden etwas über die systemeigenen Funktionen von MySQL erfahren. Außerdem wird auf Schnittstellen, u.a. auf die bekannte Perl DBI/DBD-Schnittstelle, eingegangen:

- ▶ An Tag 8 werden Sie die Werkzeuge und Tricks kennenlernen, die Sie verwenden können, um Ihre Datenbank mit Daten zu füllen.
- ▶ Tag 9 bietet einen SQL-Leitfaden. Dieser wird Ihnen die nötige Grundlage zur Bearbeitung Ihrer Daten verschaffen.
- ▶ Tag 10 behandelt die systemeigenen Funktionen von MySQL.
- ▶ Tag 11 macht Sie mit Sperren und Schlüsseln bekannt. In dieser Lektion wird erläutert, wie diese Leistungsmerkmale gebraucht werden und weshalb sie in MySQL enthalten sind.
- ▶ An Tag 12 beginnt der Themenkomplex Schnittstellen. Diese Lektion behandelt die verschiedenen APIs, die unter MySQL verfügbar sind. Sie werden die gemeinsamen Funktionen der verschiedenen Schnittstellen kennenlernen.
- ▶ Tag 13 behandelt die ODBC-Treiber-Schnittstelle. Sie werden mit den Techniken vertraut gemacht, die Sie beim Zugriff auf eine MySQL-Datenbank mit dieser Technologie benötigen.
- ▶ Tag 14 behandelt ausführlich das Perl DBI für MySQL. Aufbauend auf den vorherigen Lektionen werden Sie Ihr eigenes Perl-Programm erstellen, um auf Daten via Internet zuzugreifen.

In der dritten Woche werden einige der anspruchsvolleren Techniken, eine MySQL-Datenbank zu administrieren, vorgestellt. Sie werden erfahren, wie ein MySQL-Datenbank-Server inklusive Sicherheits- und Optimierungstechniken administriert wird. Am Ende dieser Woche erstellen Sie eine Website, wobei Sie MySQL benutzen, um diese dynamisch zu gestalten und um Benutzer-Eingaben zu verarbeiten.

- ▶ An Tag 15 lernen Sie, wie Sie mit PHP Schnittstellen zu MySQL erstellen.
- ▶ An Tag 16 wird erläutert, wie Daten in MySQL behandelt werden. Sie werden etwas über die verschiedenen Funktionen, die in MySQL enthalten sind, erfahren.
- ▶ Tag 17 geht auf die Sicherheits in MySQL ein. Sie werden lernen, wie die Sicherheits-Features in MySQL implementiert sind und wie sie arbeiten.
- ▶ An Tag 18 wird MySQL mit anderen Datenbanken verglichen. Sie werden einen Blick werfen auf Leistungsmerkmale anderer Datenbanken, die MySQL nicht besitzt. Zudem werden verschiedene Workarounds vorgestellt, mit denen Sie diese Leistungsmerkmale implementieren können.



- ▶ Tag 19 befasst sich ausführlich mit der Administration von MySQL. Sie werden lernen, wie Log-Files erstellt werden und wie man diese bei Problemen durchsucht. Ein weiterer Schwerpunkt dieses Kapitels sind Backup und Wiederherstellung der MySQL-Datenbank.
- ▶ An Tag 20 erfahren Sie etwas über Performance-Tuning der Datenbank. Sie lernen, wie Sie bessere Abfragen erstellen, und wie sie das letzte Quäntchen an Performance aus der Datenbank holen.
- ▶ Am letzten Tag werden Sie das Gelernte anwenden, um eine funktionstüchtige Applikation mit Hilfe von MySQL zu erstellen.

Über dieses Buch

Dieses Buch bringt Ihnen alles über das MySQL-Datenbank-Management-System bei. Sie werden etwas über das Erstellen und die Administration einer Datenbank sowie über Schnittstellen zu Datenbanken erfahren. Am Ende dieses Buches werden Sie eine MySQL-Datenbank installieren, erstellen, benutzen und pflegen können. Nebenbei werden Sie auch etwas über Datenbank-Design und den Umgang mit relationalen Datenbanken erfahren.

Wer sollte dieses Buch lesen?

Wenn folgendes auf Sie zutrifft, ist dieses Buch genau das Richtige für Sie:

- ▶ Ihre momentane Datenbank genügt Ihnen nicht mehr, und Sie suchen nach einem guten Ersatz.
- ▶ Sie entwickeln gerade eine Webseite, die Datenbankzugang erfordert.
- ▶ Sie haben noch nie mit einer relationalen Datenbank gearbeitet und möchten lernen, wie man eine benutzt.
- ▶ Sie wechseln zu Linux und suchen nach einem geschäftstauglichen RDBMS.

Dieses Buch wird Ihnen alle Schritte zeigen, die Sie benötigen, um MySQL zu benutzen. Hierbei werden sämtliche Aspekte von MySQL behandelt.

Wenn Sie noch nie ein relationales Datenbank-Management-System (RDBMS) benutzt haben, fragen Sie sich vielleicht, ob dieses Buch für Sie geeignet ist. Das ist es auf jeden Fall. Es führt Sie Schritt für Schritt durch die schwierigsten Situationen, wobei es Ihnen Beispiele und Illustrationen liefert, die Ihnen bei den schwierigsten Aspekten von MySQL zur Seite stehen.



Wenn Sie schon einmal ein RDBMS wie zum Beispiel Microsoft SQL Server, Sybase oder Oracle benutzt haben, ist dieses Buch ebenfalls etwas für Sie. Es bietet Ihnen die Möglichkeit, eines der weniger bekannten Systeme kennenzulernen, das es aber durchaus mit den Großen aufnehmen kann.

Wenn Sie MySQL bereits benutzt haben, oder es momentan benutzen, ist dieses Buch erst recht etwas für Sie. Hier können Sie Abkürzungen und Erklärungen finden, die in anderen Büchern möglicherweise nicht vorgestellt werden. Viele der Fragen, die Sie sich vielleicht schon lange gestellt haben, werden in diesem Buch beantwortet.

MySQL in 21 Tagen geht davon aus, dass Sie keinerlei Vorkenntnisse in Bezug auf Datenbank-Management-Systeme haben. Gehen Sie die Sache also langsam an und vergewissern Sie sich, dass Sie jede Lektion voll und ganz verstanden haben bevor Sie weitermachen. Jedes Kapitel baut auf dem vorher behandelten Material auf. Sie werden das, was Sie in einem Kapitel gelernt haben, später anwenden können.

Wie ist dieses Buch strukturiert?

Dieses Buch sollte innerhalb von drei Wochen durchgearbeitet werden. Sie bearbeiten sieben Kapitel pro Woche und führen die Übungen am Ende jedes Kapitels aus.

Konventionen



Ein Hinweis bietet interessante, manchmal technische Informationen, die sich auf die Thematik, die in diesem Abschnitt diskutiert wird, beziehen.



Ein Tip gibt Ratschläge oder liefert eine einfachere Methode, wie Sie etwas bewerkstelligen können.



Eine Warnung weist auf mögliche Probleme hin und hilft Ihnen, eventuelle Katastrophen zu vermeiden.



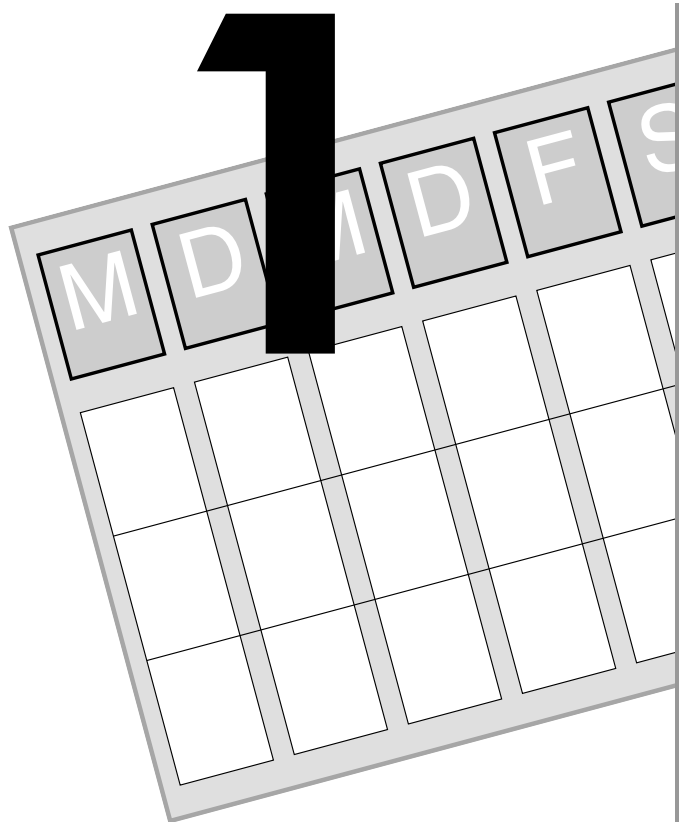
Hier wird ein neu eingeführter Begriff erklärt.



Neu auftauchende Begriffe werden in den Paragraphen, in denen sie definiert werden, *kursiv* gedruckt. Am Ende eines Kapitels befindet sich eine nützliche Zusammenfassung sowie ein Abschnitt mit Fragen und Antworten.

Außerdem finden Sie im gesamten Buch verschiedene typographische Konventionen:

- ▶ Anweisungen, Variablen, Verzeichnisse und Dateien werden `nicht-proportional` gedruckt.
- ▶ Platzhalter in Syntaxbeschreibungen werden `nicht-proportional` & *kursiv* gedruckt. Dies zeigt an, dass Sie den Platzhalter durch einen Dateinamen, einen Parameter oder durch ein anderes entsprechendes Element ersetzen.



Woche

Überblick

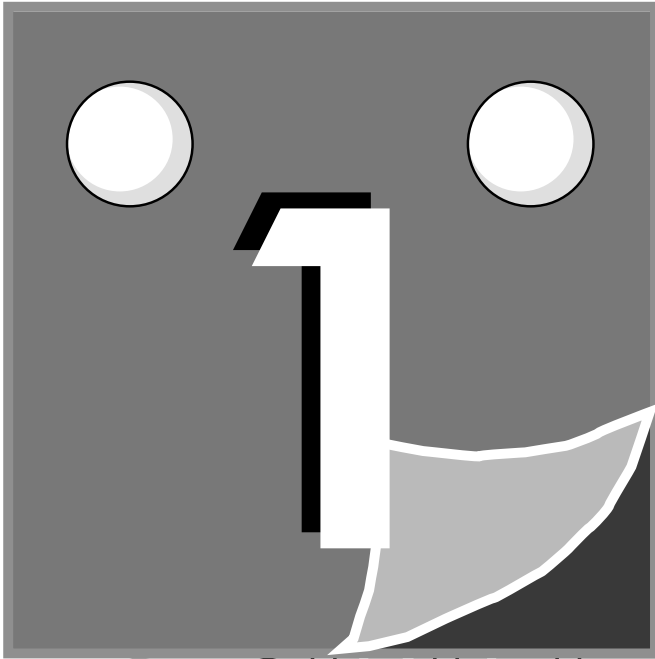


Bevor Sie anfangen, MySQL zu lernen, benötigen Sie einige Dinge: einen Computer, einen Web-Server und dieses Buch. Falls Sie die ersten beiden Sachen nicht besitzen sollten, können Sie diese Buch trotzdem nutzen. Dennoch müssen Sie das Gelernte irgendwo ausprobieren. Ohne Praxis geht es nicht. Dieses Buch wird Sie Schritt für Schritt durch jeden Aspekt von MySQL führen. Es ist so aufgebaut, dass jeder Tag mit Übungen endet. Nutzen Sie diese Übungen, denn sie können Ihnen helfen, ein selbständiger MySQL-Datenbank-Administrator zu werden.

Was Sie diese Woche lernen werden

Diese Woche behandelt die Grundlagen von MySQL. Am ersten Tag werden Sie erfahren, was MySQL überhaupt ist und wie man es u. a. einsetzen kann. An Tag 2 lernen Sie, wie Sie MySQL auf einer Windows- und Linux-Plattform installieren. An den nächsten beiden Tagen befassen Sie sich mit dem Design und der Erstellung einer Datenbank. Tag 5 behandelt die Normalisierung Ihrer Daten. An Tag 6 werden Sie lernen, Ihrer Datenbank Tabellen, Spalten und Indizes hinzuzufügen. Die Woche schließt mit den MySQL-Datentypen. Sie werden die verschiedenen Datentypen kennen lernen und erfahren, wie sie in MySQL angewendet werden.

Das ist eine Menge Stoff für eine Woche, wenn Sie sich aber genügend Zeit nehmen und die Übungen machen, werden Sie es schon packen.



Was ist MySQL?

**Woche
1**

Ich begrüße Sie zu *MySQL in 21 Tagen*. Heute werden sie ihre Entdeckungsreise in eines der besten relationalen Datenbank-Management-Systeme auf dem heutigen Markt antreten.

Sie werden heute Folgendes lernen:

- ▶ was ist eine relationale Datenbank, und wofür kann man sie gebrauchen?
- ▶ was bedeutet es für eine Datenbank, auf Unternehmensebene eingesetzt zu werden?
- ▶ worum geht es bei einer Client/Server-Programmierung?
- ▶ Einige Merkmale von MySQL

Die Geschichte von MySQL

MySQL [*mai es kju el*] ist ein Open-Source-Produkt, welches auf Unternehmensebene eingesetzt werden kann. Es ist eine Multi-Threaded, Multi-User RDBMS (relationales Datenbank-Management-System). Das mag sich alles nach Marketingstrategie oder Hype anhören, definiert MySQL aber am besten. Vielleicht sind Sie mit einigen dieser Begriffe noch nicht vertraut; am Ende dieses Tages werden Sie es aber bestimmt sein.

MySQL wurde von einer schwedischen Consultingfirma namens TcX entwickelt. Diese benötigte ein extrem schnelles und flexibles Datenbanksystem. Leider (oder zum Glück, ganz wie man es sieht) konnte sie auf dem Markt nichts Entsprechendes finden. Aus diesem Grund wurde MySQL, das zu einem geringen Teil auf einem anderen Datenbank-Management-System (DBMS) namens mSQL basiert, entwickelt. Das geschaffene Produkt ist schnell, verlässlich und extrem flexibel. Es wird mittlerweile auf der ganzen Welt benutzt. Universitäten, Internet Service Provider (ISPs) und gemeinnützige Organisationen sind vor allem auf Grund seines Preises (meistens umsonst) die hauptsächlichen Benutzer von MySQL. Seit kurzem durchdringt das verlässliche und schnelle Datenbanksystem MySQL auch die Geschäftswelt. Auf der dem Buch beigelegten CD-ROM finden Sie einige Beispiele für den kommerziellen Gebrauch von MySQL.

Der Popularitätsschub von MySQL ist einerseits im Aufkommen der Open-Source-Bewegung und andererseits in der unglaublichen Entwicklung von Linux in der Computerindustrie begründet. Die Open-Source-Bewegung, falls Sie noch nicht davon gehört haben, resultiert aus der Bereitstellung nicht nur eines Produkts, sondern auch des Source Codes seitens mehrerer Software-Hersteller. Auf diese Weise können Konsumenten sehen, wie ihr Programm arbeitet, und dieses, wenn nötig, verändern. Dies und die Popularität von Linux ließ den Gebrauch von Open-Source-Produkten in der Geschäftswelt in die Höhe schnellen. Auf Grund der immensen Popularität von Linux suchen Benutzer nach Produkten, die auf dieser Plattform funktionieren. MySQL ist eines dieser Produkte.

MySQL wird oft mit SQL, der von IBM entwickelten *Structured Query Language*, verwechselt. MySQL ist nicht eine Art SQL, sondern ein Datenbanksystem, das SQL benutzt, um Daten zu bearbeiten, zu erstellen und anzuzeigen. MySQL ist ein Programm, das Datenbanken benutzt, wie z.B. Microsoft Excel Spreadsheets benutzt. SQL ist eine Programmiersprache, die von MySQL genutzt wird, um Arbeiten innerhalb einer Datenbank zu verrichten, so wie Excel VBAs (Visual Basic for Applications) benutzt, um Aufgaben mit Spreadsheets und Workbooks zu bearbeiten. Microsofts SQL Server, der Sybase Adaptive Server und DB2 sind weitere Programme, die Datenbankfunktionalität zur Verfügung stellen.

Jetzt, da Sie wissen wie MySQL entstanden ist, werden Sie erfahren, was es ist. Zuerst beginnen wir mit dem Terminus *Datenbank*. Was ist eigentlich eine Datenbank? Wahrscheinlich haben Sie in ihrem Leben schon einmal eine Datenbank benutzt. Falls Sie jemals etwas über das Internet gekauft oder einen Online-Katalog durchstreift haben, können Sie davon ausgehen, dass Sie bereits eine benutzt haben. Eine *Datenbank* ist eine Reihe strukturierter Dateien in einem Computer, die auf höchst effiziente Art und Weise gegliedert sind. Diese Dateien können tonnenweise Informationen speichern, welche bearbeitet und abgerufen werden können, wenn sie benötigt werden. Eine Datenbank ist in folgender hierarchischer Art und Weise, von oben nach unten betrachtet, aufgebaut. Es beginnt mit einer Datenbank, die eine gewisse Anzahl an Tabellen enthält. Jede Tabelle besteht aus einigen Spalten. Die Daten sind in Zeilen gespeichert, und wo Zeilen und Spalten sich schneiden, entsteht ein Feld. Die Abbildung 1.1 gibt diese Aufgliederung wieder.

Ihre bevorzugte Online-Buchhandlung basiert beispielsweise auf einer Datenbank. Diese Datenbank besteht aus vielen Tabellen. Jede Tabelle enthält bestimmte allgemeine Daten. Wahrscheinlich würde die Datenbank eine *Authors*-Tabelle und eine *Books*-Tabelle beinhalten. Diese Tabellen bestehen aus benannten Spalten, die anzeigen, welche Daten sie enthalten. Wird in eine Tabelle ein Datensatz eingefügt, entsteht eine Datensatzzeile. Wo eine Zeile und eine Spalte sich schneiden, entsteht ein Feld. Auf diese Weise sind Datenbanken aufgegliedert.

MySQL ist jedoch mehr als nur eine Datenbank. Es ist ein System, das Datenbanken benutzt. Es kontrolliert, wer Datenbanken benutzen kann und wie man sie handhabt. Es protokolliert die Handlung und läuft kontinuierlich im Hintergrund. Das ist vielleicht anders als das, woran Sie gewöhnt sind. Beim Thema Datenbanken denken viele an Microsoft Access oder Lotus Approach. Dieses sind zwar Datenbanken, aber keine Management-Systeme. Ein DBMS kann zahlreiche Datenbanken beinhalten. Benutzer stellen eine Verbindung zum Datenbankserver her und senden Anfragen. Der Datenbankserver befragt seine Datenbanken und gibt den Absendern die Ergebnisse zurück. Datenbanken wie Approach und Access sind von diesem Systemtyp noch einen Schritt entfernt. Sie teilen ihre Dateien mit einer Vielzahl von Benutzern, wobei es allerdings kein Interface gibt, das die Verbindungen oder das Beantworten von Anfragen kontrolliert.

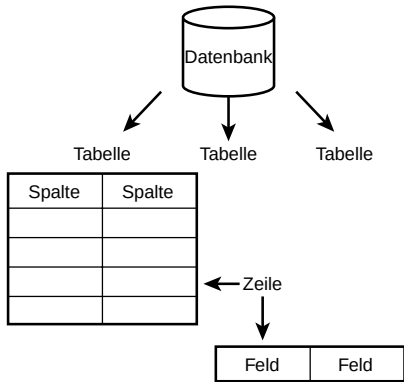


Abbildung 1.1:
Der Aufbau einer Datenbank

Für ein DBMS wie MySQL gibt es jede Menge Einsatzmöglichkeiten – von Help-Desk-Systemen bis Web-Applikationen. Es ist wichtig festzuhalten, dass MySQL groß und schnell genug ist, um in nahezu jeder Situation zu funktionieren. Ein Unternehmen ist der Ort, an dem MySQL am besten Anwendung findet.

Was ist ein Unternehmen?

Das *Unternehmen* ist der Bereich der Geschäftswelt, in dem viele große Systeme interagieren, um ein gemeinsames Ziel zu erreichen. SAP, Microsoft SQL Server, Oracle 8i und Sybase Adaptive Server sind einige Applikationen, die sich auf dieser Ebene der Geschäftswelt bewegen. Die Computerapplikationen, die sich auf dieser Ebene befinden, haben häufig bestimmte Merkmale. Sie sind von Haus aus Multi-User-Systeme – d. h., viele Personen können die gleiche Applikation zur gleichen Zeit nutzen. Außerdem müssen Applikationen auf diesem Level sehr sicher und flexibel sein.

Ein Charakteristikum einer Applikation auf Unternehmensebene ist die gleichzeitige Nutzung von mehr als einer Person. Auf dieser Ebene gilt dies als Grundvoraussetzung, schließlich kann es vorkommen, dass zu einem bestimmten Zeitpunkt mehr als eine Person Zugang zu den Geschäftsinformationen benötigt. Dies zu ermöglichen, ist entscheidend für den Erfolg eines Unternehmens. MySQL wird dieser Anforderung gerecht. Es kann bis zu 101 gleichzeitige Verbindungen aufrechterhalten. Das bedeutet nicht, dass nur 101 Personen diese Applikation nutzen, sondern dass 101 Verbindungen gleichzeitig laufen können – das ist etwas anderes. Eine Verbindung stellt die Zeit dar, die ein Benutzer benötigt, um die angeforderten Daten zu erhalten. Mit MySQL wird die Verbindung in null Komma nix hergestellt. Der Zeitaufwand ist verschwindend gering. Die meisten Datenbanken der Größenordnung von MySQL lassen weniger Verbindungen zu. Das einzige DBMS, das MySQL in diesem Punkt noch übertrifft, ist Microsofts SQL Server.

Ein weiteres Merkmal, das eine Applikation auf Unternehmensebene haben muss, ist Sicherheit. Wird mit kritischen Informationen gearbeitet, sollte die Einsicht nur den Personen gestattet sein, die diese unbedingt haben müssen. Sicherheitskonzepte halten Leute, die dem System Schaden zufügen wollen, fern. Bei fehlender Sicherheit kann es zu ernsthaften Problemen kommen. MySQL wird auch dieser Anforderung gerecht; seine Sicherheitsmechanismen suchen ihresgleichen. Der Zugang zu einer MySQL-Datenbank kann sogar von einem entfernten Rechner kontrolliert und von diesem aus kann bestimmt werden, welcher Benutzer eine Tabelle einsehen darf. Die Datenbank kann weiter abgesichert werden, indem das Betriebssystem ebenfalls zur Sicherung herangezogen wird. Nur wenige Datenbanken dieser Größenordnung sind so sicher wie MySQL.

Eine Applikation auf Unternehmensebene muss zudem flexibel sein. Man stellt sich folgende Fragen: Wie flexibel ist die Applikation? Ist sie wandlungsfähig genug, um den ständig wachsenden Bedürfnissen der Geschäftswelt nachzukommen? In wieweit können die notwendigen Veränderungen vollzogen werden? Wie schwer ist es, diese Veränderungen vorzunehmen? MySQL liefert Antworten auf diese Fragen. Es ist sehr flexibel und einfach zu benutzen. Außerdem läuft es auf fast jeder Plattform. Möchte ein neuer CIO von Windows NT zu Linux wechseln, so passt MySQL sich problemlos an. MySQL wird ebenfalls mit dem Quellcode ausgeliefert. Falls Sie tiefgreifende Veränderungen vornehmen müssen, können Sie den Quellcode verändern und somit die Änderungen selbst herbeiführen. Fehlt MySQL ein Feature, das Sie unbedingt brauchen, fügen Sie es einfach selbst hinzu. Keine andere Datenbank auf dem Markt ist so flexibel wie MySQL. Darüber hinaus besitzt MySQL mehrere Interfaces (Schnittstellen) auf Applikationsebene; und diese in einer Vielzahl von Sprachen. Arbeiten Sie vorwiegend mit Microsoft, so können Sie ODBC benutzen, um mit MySQL zu interagieren. Arbeitet Ihre Firma mit Unix, dann können Sie C, PHP, Perl oder JDBC verwenden. Wie Sie sehen, ist MySQL unwahrscheinlich flexibel.

Zusätzlich zu den oben erörterten Charakteristika müssen Datenbanken auf Unternehmensebene zusammenarbeiten können. Datawarehousing ist eine Technik, die alle Daten in einem Geschäft verbindet. Auf Grund seiner Flexibilität und Schnelligkeit, findet MySQL sich in jeder Situation zurecht.

Das Internet ist ebenfalls Teil eines jeden Unternehmens geworden. Alle größeren Firmen sind im Internet präsent. Diese Firmen benötigen Datenbanken, um auch auf dieser Geschäftsebene verkaufen und wettbewerbsfähig bleiben zu können. MySQL lässt sich hervorragend als Datenbankserver im Internet einsetzen. Es hat sich auf diesem Gebiet bewährt und ist zur bevorzugten Datenbank vieler ISPs aufgestiegen. Wegen seiner Schnelligkeit und seiner vielfachen Applikationsinterfaces ist MySQL die optimale Wahl für ein Unternehmen.

Was ist eine relationale Datenbank?

Eine relationale Datenbank ist, einfach gesagt, eine Datenbank, die aus Tabellen und Spalten besteht, welche sich aufeinander beziehen. Diese Beziehungen gründen auf einem Schlüsselwert, der in einer Spalte enthalten ist. Sie können beispielsweise eine Tabelle namens *Orders* haben, in der sämtliche Informationen, wie zum Beispiel die Bestellnummer, das Bestelldatum oder das Versanddatum, enthalten sind, die sie benötigen, um eine Bestellung zu bearbeiten. Ebenso können Sie eine Tabelle namens *Customers* haben, die alle Daten über die Kunden, zum Beispiel deren Namen oder Adresse, beinhaltet. Diese beiden Tabellen können aufeinander bezogen werden – keine Bestellung ohne Kunden, nicht wahr? Alles über Beziehungen erfahren Sie an Tag 3, »Das Entwerfen Ihrer ersten Datenbank«.

Das Konzept für eine relationale Datenbank wurde in den frühen 70ern von E.F.Codd entwickelt. Er schlug vor, Daten in einer Datenbank in Spalten und Tabellen, die aufeinander bezogen werden können, zu speichern. Dieses Konzept unterscheidet sich sehr stark von dem hierarchischen Dateisystem, das damals benutzt wurde. Seine Denkweise hat das Erstellen und Nutzen von Datenbanken revolutioniert.

Eine relationale Datenbank arbeitet sehr intuitiv. Sie imitiert die Denkweise des Menschen. Menschen neigen dazu, ähnliche Objekte in Gruppen zu vereinen und komplexe Vorgänge in einfachere aufzugliedern. Relationale Datenbanken machen das genauso. Da sie Ihre Denkweise nachahmen, sind sie einfach zu benutzen und zu begreifen.

Die meisten modernen Datenbanken verwenden zur Bewerkstelligung Ihrer Aufgaben das relationale Modell. MySQL bildet da keine Ausnahme. Es entspricht absolut dem relationalen Modell, was zusätzlich zum leichteren Gebrauch von MySQL beiträgt.

Das Client/Server-Paradigma

Das Client/Server-Paradigma gibt es schon länger, als die meisten denken. Wenn Sie zurückdenken an die Anfänge des Programmierens, werden Sie sich an die riesigen Mainframe Computer mit ihren vielen kleineren »dummen« Terminals erinnern. Diese Terminals wurden zurecht als »dumm« bezeichnet, da an ihnen weder Programmlogik noch Verarbeitung ausgeführt wurde. Sie waren lediglich Behälter für das Output des Mainframe-Computers. Dies waren die Anfänge des Client/Server-Zeitalters, allerdings war der Begriff Client/Server nicht das Modewort, das es heute ist.

Die zunehmende Verbreitung des PC trug nicht nur zur Entstehung des Local Area Network (LAN), sondern auch zur Entfaltung des Client/Server-Modells bei. Von nun an konnte Verarbeitung auch auf dem Client erledigt werden. Clients begannen,

Daten zu teilen. Die Daten wurden in Computern namens File Server (Datei-Server) gespeichert. Jetzt wurde die ganze Verarbeitung nicht mehr auf dem Server, sondern auf dem Client durchgeführt. Der Server oder zentralisierte Computer war von da an nichts anderes mehr als ein riesiges Datenlager. Er erledigte wenig oder gar keine Verarbeitung – die totale Umkehr der früheren Denkweise.

Nach einigen Jahren wurden Desktop-Applikationen immer leistungstärker. Immer mehr Informationen mussten immer schneller aufgeteilt werden. Dies führte zu leistungstärkeren Servern, die Anfragen von Clients beantworteten und verarbeiteten. Diese Server entsprechen dem, was Sie heute als Datenbank-, Web- und File-Server kennen.

Aus dieser Zeit stammt der Begriff »Client/Server-Computing«. Das »Client/Server-Computing« ist im Grunde ein *Two-Tier-Design*; der Client sendet Anfragen und der Server beantwortet sie. All die Logik befindet sich auf der Applikationsebene des Clients. *Two-Tier-Design* ist auch heute noch sehr weit verbreitet. Es ist auch unter dem Namen *Fat Client* bekannt, da die gesamte Applikationsverarbeitung auf der Client-Ebene erledigt wird.

Nach einigen Jahren wurden die Server auf Grund Ihrer Leistungsfähigkeit zu den Schaltzentralen der Geschäftsorganisation. Sie waren normalerweise Systeme auf höchster Ebene die auf Schnelligkeit ausgerichtet waren, mit der besten Hardware. Es war nur eine Frage der Zeit, bis jemand auf die Idee kam, die Fähigkeiten seiner Programme auf den Server zu übertragen. Der Client sollte nur eine grafische Benutzeroberfläche (GUI) sein und die Hauptapplikation bzw. Logik sollte auf dem Server laufen. Der Server sollte dann je nach Notwendigkeit die benötigten Anfragen an andere Server wie zum Beispiel Datenbank- oder File-Server stellen. Das ist die Geburtsstunde des *Three-Tier-Designs* bzw. *Thin Client*. Bei diesem Design wird sämtliche Verarbeitung der Logik auf der Serverebene ausgeführt. Dies ermöglicht dem leistungstärkeren Rechner, die Logik zu handhaben und dem leistungsschwächeren Rechner das Output anzuzeigen. Kommt Ihnen das nicht irgendwie bekannt vor? Das sollte es – Sie sehen, der Kreis beginnt sich zu schließen. Die schwierige Verarbeitung wird wiederum auf den stärkeren, zentralisierten Maschinen erledigt, während die Clients lediglich das Output anzeigen.

Das Internet ist ein erstklassiges Beispiel für eine *Thin-Client*-Konstruktion. Der Browser sendet Anfragen an einen Web-Server, der die Anfrage verarbeitet und eine Antwort an diesen zurückschickt. Der Browser zeigt dann die angeforderte Information an – der Kreis schließt sich.

Und wieder befinden wir uns an der Schwelle zu einem neuen Computer-Zeitalter. Die verschiedenen Applikationen werden mehr im gesamten Netzwerk verteilt. Auf Grund der sinkenden Computerpreise tauchen auf den Bürotischen immer bessere Rechner, die als Clients fungieren, auf. Dies ermöglicht den Anwendungen, Leerlaufzeiten auszunutzen und eigene Prozesse durchzuführen. Server-Applikationen sind ebenfalls auf

dem Vormarsch. Sie können nun Funktionen auf anderen Rechnern ausführen und verteilte System sehr einfach realisieren. Diese Fortschritte machen Ihre Applikationen robuster und geschäftstauglicher.



Verteilte Systeme erlauben es den Client-Programmen mit mehreren Server-Prozessen zu interagieren, welche selbst wiederum mit anderen Servern zusammenarbeiten können. Die Serverkomponenten können so alle Ressourcen des Netzwerks ausnutzen.

MySQL kann sich sehr gut an all diese Konstruktionen anpassen. Es verrichtet sowohl in einer *Two-Tier*-als auch in einer *Three-Tier*-Konstruktion sehr gute Arbeit, kann aber auch alleine hohe Leistungen erbringen.

Merkmale von MySQL

MySQL ist ein perfekt ausgerüstetes relationales Datenbank-Management-System. Es ist sehr stabil und hat sich über die Jahre bewährt. MySQL befindet sich seit mehr als zehn Jahren im Produktionseinsatz.

MySQL ist ein Multi-Threaded Server. *Multi-Threaded* bedeutet, dass jedes Mal, wenn jemand eine Verbindung zum Server herstellt, das Serverprogramm einen Thread oder Prozess erstellt, um die Anfragen des Clients zu bearbeiten. Das erfordert einen extrem schnellen Server. In der Tat erhält jeder Client, der eine Verbindung zu einem MySQL-Server herstellt, seinen eigenen Thread.

Zudem ist MySQL vollständig ANSI-SQL92-kompatibel. MySQL hält sich an alle vom American National Standards Institute festgelegten Regeln.



ANSI SQL92 ist ein Regelwerk für die Structured Query Language, das 1992 vom American National Standards Institute vereinbart wurde.

Ein weiteres wichtiges und sehr nützliches Feature von MySQL ist sein Online-Hilfssystem. Alle Befehle für MySQL können auf der Kommandozeilenebene eingegeben werden. Um herauszufinden, welche Optionen die Befehle bereitstellen oder was ein Utility oder ein Befehl bewirken, müssen Sie nur den entsprechenden Befehl und zusätzlich entweder `-help` oder `-?` eingeben. Auf diese Weise erhalten Sie jede Menge Informationen über diesen Befehl.

Portabilität ist ein weiteres Feature von MySQL. Es wurde schon auf fast jede Plattform portiert. Das bedeutet, dass Sie Ihre Hauptplattform nicht wechseln müssen, um von MySQL zu profitieren. Und falls Sie Ihre Plattform doch wechseln wollen, so gibt es für Ihre neue Plattform wahrscheinlich eine eigene MySQL-Portierung.

MySQL besitzt zudem viele verschiedene *Application Programming Interfaces* (APIs) – unter anderem APIs für PHP, Perl, TCL, Python, C/C++, Java (JDBC) und ODBC. Es ist also fast völlig egal, welcher Variante Ihre Firma den Vorzug gibt – MySQL bietet einen Weg für den Zugriff.

MySQL ist zudem sehr günstig. Für eine nicht lizenzierte vollständige Version von MySQL sind die Kosten gleich Null. Seit Version 3.23.19 steht MySQL unter GPL und kostet ab dieser Version auch auf Windows-Plattformen kein Geld mehr. Ihre Kopie lizenzieren zu lassen, kostet Sie im Moment etwas mehr als 170 EUR. In Anbetracht dessen, was Sie für Ihr Geld bekommen, ist das ein Bombengeschäft. Datenbanksysteme, die nur halb so gut ausgerüstet sind wie MySQL, können Zehntausende kosten. MySQL kann das, was diese können, günstiger und besser erledigen. Detaillierte Informationen darüber, ob und wann Sie Ihre MySQL-Version lizenzieren müssen, erhalten Sie auf folgender Webseite: <http://www.mysql.com/support/arrangements/policy.html>.

Zusammenfassung

Wie Sie sehen, ist MySQL ein sehr robuster Datenbank-Server. Es ist absolut unternehmenstauglich und es beinhaltet die fortschrittlichsten Sicherheitskonzepte, die auf dieser Geschäftsebene benötigt werden. Zudem ist MySQL so schnell und flexibel wie keine andere Datenbank seiner Größenordnung.

MySQL ist eine relationale Datenbank. Es benutzt Tabellen und Spalten, um Daten zu speichern, die durch Schlüssel verbunden werden.

Es ist sehr gut für verschiedene Architekturen geeignet. MySQL kann sowohl in einer reinen Client/Server-Architektur als auch als allein arbeitende Datenbank verwendet werden. Was auch immer Sie benötigen, MySQL kann dem gerecht werden.

Heute haben Sie die wichtigsten Merkmale von MySQL kennen gelernt. Sie haben beispielsweise erfahren, dass MySQL Multi-Threaded und ANSI-SQL92-kompatibel ist. Darüber hinaus wurde auf die verschiedenen Plattformen und APIs, die MySQL benutzen kann, eingegangen.

Schließlich haben Sie erfahren, dass MySQL meistens kostenlos ist (wenn Sie etwas über Lizenzierungsregeln erfahren wollen, besuchen Sie die MySQL Website unter <http://www.mysql.com>) – kaum zu glauben bei einer solch robusten, flexiblen und schnellen RDBMS.

Fragen und Antworten

- F** Meine Mutter hat immer gesagt, dass man bekommt, wofür man bezahlt. Wenn MySQL so großartig ist, warum ist es dann so billig?
- A** *Wenn etwas billig ist, dann taugt es nichts – diese Überzeugung teilen die meisten Leute. Auf einige Dinge trifft dies vielleicht zu, nicht aber auf MySQL. MySQL ist Teil der Open-Source-Bewegung. Es wurde von einer Gruppe von Entwicklern geschaffen, die mit der Weiterentwicklung auf eigene Kosten fortfahren. Dies ermöglicht dem Benutzer, in den Genuss eines großartigen Produkts zu kommen – und das für wenig Geld oder sogar umsonst.*
- F** Wenn MySQL all das kann, was Sie sagen, warum habe ich dann noch nichts davon gehört?
- A** *MySQL hatte bisher nicht die gleiche Popularität wie manch andere Datenbank-Produkte, weil ihm kein großes Unternehmen den Rücken stärkte. Es wurde von einer Consulting-Firma für einen Klienten entwickelt. Diese Firma hat es allerdings nicht vermarktet. Der einzige Grund, weshalb MySQL jetzt an Popularität gewinnt, ist der Erfolg von Linux und der Open-Source-Bewegung. Durch dieses Buch und die Stärke des Produktes werden hoffentlich bald mehr Menschen die Vorteile von MySQL kennen lernen.*

Übungen

1. Vergleichen Sie die Preise einiger anderer Datenbanken mit der gleichen Ausstattung wie MySQL – das sind zum Beispiel SQL Server, Oracle, Adaptive Server und DB2. Überprüfen Sie nun, wie viel Sie mit MySQL sparen können.
2. Besuchen Sie Websites und testen Sie Produkte, die MySQL benutzen. (Einige befinden sich auf der beigefügten CD-ROM.) MySQL in Aktion zu sehen, kann Ihre Meinung über Open-Source-Produkte wirklich verändern.



Los geht's

**Woche
1**

Heute werden Sie lernen, wie MySQL auf Linux- und Windows-Plattformen installiert wird. Zudem erfahren Sie etwas über Lizenzierung und wo Sie MySQL herbekommen. Außerdem werden Sie lernen, wie man das Root-Passwort ändert, wie Sie den MySQL-Server-Daemon starten und beenden, und schließlich, wie Sie den auf der Kommandozeile basierenden MySQL-Monitor benutzen.

Lizenzierung

Nun, da Sie wissen, was MySQL ist und wie man es am Arbeitsplatz einsetzen kann, können sie anfangen, damit zu arbeiten. Wie schon erwähnt, ist MySQL eine Open-Source-Applikation. Der Quellcode ist für jedermann einseh- und abänderbar.

Allerdings ist MySQL nicht immer umsonst. MySQL steht zwar seit der Version 3.23.19 unter der GPL und ist ab dieser Version auch für Windows-Plattformen kostenlos, es gibt jedoch einige Ausnahmefälle, in denen Sie eine Lizenz erwerben müssen (Sie können natürlich auch freiwillig eine Lizenz kaufen, wenn Sie die Entwickler unterstützen möchten). Um mehr über Lizenzierungsregeln zu erfahren, empfehle ich Ihnen den Besuch der MySQL-Webseite (<http://www.mysql.com/support/arrangements/>).

Der Download von MySQL

Nachdem Sie geklärt haben, ob Sie Ihre Version von MySQL lizenzieren müssen, können Sie es jetzt herunterladen. Klicken Sie auf der MySQL Website »Downloads« an. Es erscheint eine Liste mit den zur Zeit erhältlichen Binär- und Quellcode-Versionen von MySQL für verschiedenste Plattformen. Auch auf der buchbegleitenden CD-ROM finden Sie verschiedene Versionen. Dadurch können Sie beim Installationsvorgang einen Schritt einsparen. Wollen Sie die CD-ROM benutzen, halten Sie sich an die Ihrem System entsprechende Anleitung.

MySQL wird ständig verändert und verbessert. Die jüngsten Ausgaben sind die Alpha-Versionen, die neben einigen Verbesserungen und Fehlerbeseitigungen gegenüber den Vorgängern die neuesten Features von MySQL enthalten. Diese Versionen wurden nicht so gründlich geprüft wie die »stabilen« Endversionen (*stable releases*). Daher empfiehlt es sich, die neuesten Versionen, die auf der ganzen Welt in der Produktion eingesetzt werden, zu benutzen.

MySQL läuft auf vielen Plattformen. Für die meisten dieser Plattformen sind auch Binär-Distributionen erhältlich. Diese sind das Ergebnis der Kompilierung des Quellcodes. Auf diese Weise können Sie MySQL am einfachsten erhalten. Die andere Möglichkeit besteht darin, den Quellcode für Ihre Plattform herunterzuladen und anschlie-

ßend selbst zu kompilieren. Dies kann Ihnen allerdings Schwierigkeiten bereiten. Sie benötigen hierzu die passenden Bibliotheken sowie einen Compiler. Dieser Vorgang wird in diesem Buch nicht behandelt, wenn Sie den Quellcode aber unbedingt selbst kompilieren wollen, lesen Sie sich die Anleitung gründlich durch. Bei Problemen werfen Sie einen Blick in die MySQL-Mailing-Listen. Sie sind eine nicht versiegende Informationsquelle für MySQL-Administratoren. Die Mitglieder des Entwicklungsteams gehen die Beiträge regelmäßig durch und geben gerne Auskünfte oder Orientierungshilfen.

Besuchen Sie eine Mirror Site, um die Linux-Version herunterzuladen. Eine *Mirror Site* ist eine exakte Kopie einer Website auf einem anderen Server. Dadurch wird der Haupt-Webserver entlastet und andere Benutzer können den Webserver problemlos nutzen. Wenn Sie sich auf der Mirror Site befinden, klicken Sie die benötigte Binärdistribution an.

Dieses Buch behandelt sowohl die Installation der Linux- als auch die der Windows-Version.

Die Linux-Installation

Nach Beendigung des Downloads haben Sie eine gepackte .tar.gz-Datei namens `mysql-3.22.23b-pc-linux-gnu-i686.tar.gz` vor sich. Es wird empfohlen, diese nach `/usr/local` zu entpacken – alle Standardpfade zeigen auf diesen Ort. Es ist möglich, dass Sie `root`-Privilegien benötigen, um das `/usr`-Verzeichnis zu verändern. Ich habe die Erfahrung gemacht, dass man MySQL am besten als `root` installiert; auf diese Weise scheint es weniger Probleme mit den Benutzerrechten zu geben. Um die Datei wie gerade erklärt zu entpacken, müssen Sie Folgendes auf der Kommandozeilenebene eingeben:

```
cd /usr/local
gunzip < mysql-3.22.23b-pc-linux-gnu-i686.tar.gz | tar xvf -ln -s mysql-3.22.23b-
pc-linux-gnu-i686 mysql
```

Das ist lediglich ein Beispiel – der Dateiname kann sich wieder ändern, sobald neue Versionen auf den Markt gebracht werden. Es entpackt das MySQL-Binärpaket und erstellt die Verzeichnisstruktur. Die letzte Zeile erstellt ein symbolisches Link zu diesem Verzeichnis. Wechseln Sie in dieses Verzeichnis und rufen Sie dessen Inhalt auf:

```
cd mysql
ls
```

Sie sollten jetzt folgendes Ergebnis vor sich haben:

```
ChangeLog      bin lib mysql-for-dummies
INSTALL-BINARY configure manual.htm scripts
PUBLIC        data manual.txt share
README        include manual_toc.html sql-bench
Support-files  tests
```

Die Windows-Installation

Die Windows-Installation ähnelt sehr der Linux-Installation. Besuchen Sie wiederum eine Mirror Site, von der Sie Ihre Windows-Version herunterladen. Bei der Windows-Datei handelt es sich um eine selbst extrahierende und installierende WinZip-Datei. Nach Beendigung des Downloads starten Sie den Installationsvorgang, indem Sie die entpackte Datei doppelklicken. Nach der Installation und einem Neustart finden Sie sich auf Ihrem Desktop wieder.

Um die neuen Dateien einzusehen, starten Sie den Windows Explorer und gehen in das C:\-Verzeichnis. Sie sollten jetzt etwas vor sich haben, das Abbildung 2.1 gleicht.

Linux- und Windows-Installationen haben nahezu die gleiche Verzeichnisstruktur.

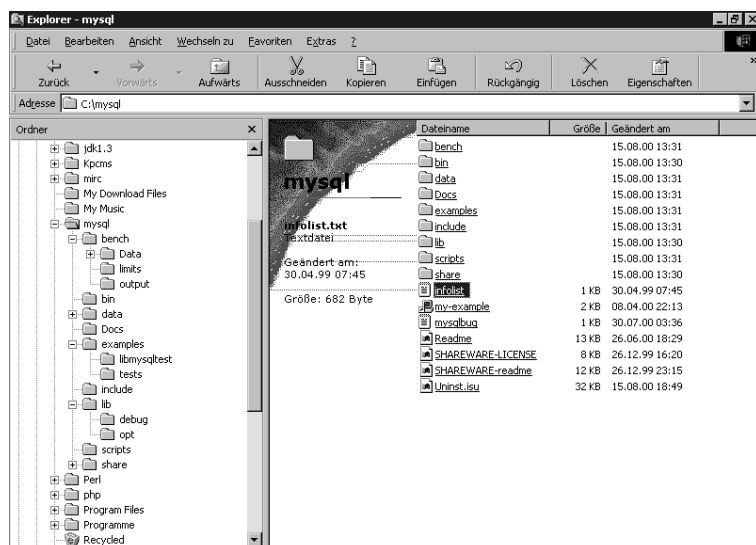


Abbildung 2.1:
Nach der
MySQL-Windows-
Installation

Datei-Inhalt

Bevor Sie weitermachen, überprüfen Sie, was Sie haben.

`ChangeLog` ist eine Datei, die alle Veränderungen sowie Bugfixes für diese bestimmte Version enthält.

`INSTALL-BINARY` ist eine Informationsdatei, die erklärt, wie MySQL installiert wird und Hinweise über die Plattform, die diese Binärdistribution benutzt, gibt.

Die `PUBLIC`-Datei beinhaltet das Copyright und das Lizenzrecht.

Die `README`-Datei enthält Informationen bezüglich der Einrichtung Ihres MySQL-Servers.

Das `support-files`-Verzeichnis enthält Dateien, die Ihnen bei der Konfiguration Ihres MySQL-Servers behilflich sind. Darüber hinaus beinhaltet es eine Spezifikationsdatei, die erklärt, warum TcX MySQL geschaffen hat.

Das `bin`-Verzeichnis enthält sämtliche MySQL-Programme, wie beispielsweise `mysqladmin` und `mysqlaccess`. Sie werden sich damit in späteren Kapiteln noch ausführlicher befassen. Die Windows-Version beinhaltet einige zusätzliche Programme. Der `MySQLManager` (Abb. 2.2) ist die grafische Benutzeroberfläche (GUI) von MySQL. Es zeigt die aktuellen Datenbanken sowie die Tabellenstruktur an und ermöglicht dem Benutzer, Anfragen zu stellen – und das alles in einer übersichtlichen grafischen Umgebung. Wie bei jedem GUI wurde etwas Leistungsfähigkeit zugunsten des Erscheinungsbildes geopfert. Nicht alle in der Konsole erhältlichen Features sind vertreten. GUIs sind eine feine Sache – aber um dieses Werkzeug handhaben zu können, müssen Sie zuerst wissen, wie man die Konsole benutzt.

Die `configure`-Datei beinhaltet ein Script, das die `GRANT`-Tabellen in Ihrem MySQL-Server erstellt.

Das `data`-Verzeichnis ist der Ort, an dem sämtliche Daten für die Datenbank gespeichert werden. Jede Datenbank besitzt ein entsprechendes Verzeichnis sowie Dateien, die die Daten speichern, die Tabellen konfigurieren und Zugang zu den in den Dateien gespeicherten Daten verschaffen.

Das `include`-Verzeichnis enthält alle C-Dateien. Diese werden vom C/C++ API sowie vom `MyODBC`-Treiber benutzt.

Das `tests`-Verzeichnis beinhaltet mehrere Perl-Scripts, um Ihren MySQL-Server zu testen.

Das `lib`-Verzeichnis enthält die Bibliotheken, die von der C++ API benutzt werden.



Abbildung 2.2:
Der MySQL-
Manager

`manual.txt`, `manual.htm` und `manual_toc.htm` sind außer den Datendateien die wohl wichtigsten Ressourcen für einen MySQL-Verwalter oder -Entwickler. Sie liefern eine unschätzbare Fülle an Informationen. Im Übrigen ist die `mysql_for_dummies`-Datei ein geeigneter Startpunkt für den MySQL-Neuling.

Das `scripts`-Verzeichnis enthält das Installationsskript für MySQL. Es wird durch den `configure`-Befehl aufgerufen.

Das `share`-Verzeichnis enthält die Fehlerprotokolle und Nachrichten.

Das `mysql_bench`-Verzeichnis beinhaltet das `crash_me`-Tool, das benutzt wird, um Vergleiche zwischen verschiedenen Datenbanken zu generieren. Vergleichsinformationen sowie Standardbeispiele finden Sie auch auf der MySQL-Webseite.

Wie ändere ich das Passwort?

Nun, da die Verzeichnisstruktur erstellt ist und die Daten entpackt sind, können Sie anfangen, MySQL zu konfigurieren und laufen zu lassen. Um die `GRANT`-Tabelle zu erstellen, vergewissern Sie sich, dass Sie sich im `/usr/local/mysql`-Verzeichnis befinden, und geben Folgendes auf der Kommandozeilenebene ein:

```
scripts/mysql_install_db
```

Während das Script die `GRANT`-Tabelle für MySQL erstellt, erhalten Sie auf Ihrem Bildschirm ein Durcheinander an Output. Die Tabelle enthält Informationen darüber, wer

eine Verbindung zur Datenbank herstellen kann. Jetzt ist ein guter Zeitpunkt, um das `root`-Passwort zu ändern. Die Datenbank- und die Dateisystem-Privilegien sind zwei verschiedene Paar Schuhe. Wenn es also einen Systembenutzer Tamara gibt, heißt das noch lange nicht, dass es auch einen Datenbankbenutzer Tamara gibt. Den gibt es erst, wenn Sie ihn erstellen. Die Sicherheit von MySQL ist unabhängig von der Sicherheit des Systems. An Tag 17, »Datenbank-Sicherheit in MySQL«, erfahren Sie mehr über Sicherheit und Privilegien. Für den Moment wählen Sie einfach ein Passwort anstelle von `root`.

Auf der Kommandozeilebene geben Sie Folgendes ein, wobei Sie `newpassword` durch Ihr neues Passwort ersetzen.

```
bin/mysqladmin -password newpassword
```

Dies setzt an die Stelle des bisherigen Passworts das neue Passwort. Sicherheit hat in jeder Umgebung, besonders aber im Umgang mit dem Internet, höchste Priorität. Ändern Sie Ihr Passwort also auf jeden Fall. Sollten Sie dies nicht tun, riskieren Sie, dass Ihre Daten in die falschen Hände geraten.

Hoch- und Herunterfahren des Servers

Wie die meisten Datenbank-Management-Systeme (DBMS) seiner Größenordnung läuft MySQL als Service bzw. Daemon. Ein *Service* oder *Daemon* ist ein Programm, das kontinuierlich im Hintergrund läuft. Normalerweise hat es kein Benutzer-Interface und wird erst sichtbar, wenn Sie ein `ps` unter Linux durchführen oder im Task Manager von Windows nachsehen (siehe Abbildung 2.3). `mysqld` ist ein Serverprogramm, dessen einzige Aufgabe darin besteht, zu warten, bis jemand eine Verbindung mit ihm herstellt und eine Anfrage sendet. Anschließend antwortet es auf diese Anfrage.

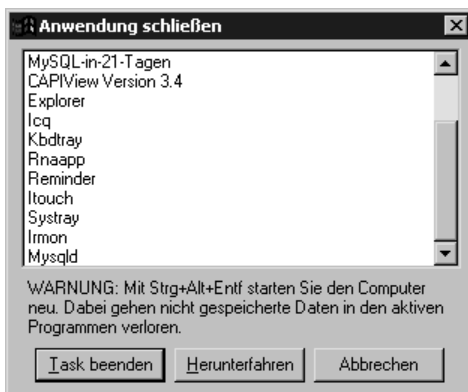


Abbildung 2.3:
Der Windows Task-Manager: `mysqld` läuft
als Hintergrundprozess

Stellen Sie sich das Serverprogramm als einen Informationsschalter in einem Einkaufszentrum vor. Es sitzt da so rum und hat nichts anderes zu tun, als Fragen von neugierigen Kunden zu beantworten. Die Leute, die am Schalter arbeiten und auf die Fragen der Kunden antworten, haben natürlich nicht alle Antworten parat. Sie müssen sie sich mit Hilfe der zur Verfügung stehenden Mittel beschaffen. Der `mysqld`-Daemon macht es ungefähr genauso. Nachdem er gestartet wurde, hält er sich im Hintergrund auf und wartet auf Anfragen. Je nach Anfrage wird er sie direkt beantworten oder die entsprechende Datenbank befragen. Das ist es, was eine Datenbank auf Unternehmensebene von einer Desktop-Applikation unterscheidet.

MySQL unter Linux benutzen

Um Ihre Datenbank benutzen zu können, muss der MySQL-Daemon laufen. (Wenn das Serverprogramm nicht läuft, kann es auf keine Anfragen antworten.) Um den Server unter Linux zu starten, vergewissern Sie sich, dass Sie sich im `mysql`-Verzeichnis befinden, und geben Folgendes auf der Kommandozeilenebene ein:

```
cd mysql
bin/safe_mysql &
```

Der `safe_mysqld`-Befehl startet den Server. Das `&`-Zeichen veranlasst das Programm dazu, im Hintergrund zu laufen. Es gibt mehrere Möglichkeiten, sich zu vergewissern, ob der Prozess läuft. Versuchen Sie es mit Folgendem auf der Kommandozeilenebene:

```
cd mysql
bin/mysqladmin -p ping
```

Jetzt sollte Folgendes erscheinen:

```
Enter password: Yourpassword <enter>
Mysqld is alive
```

Das `ping`-Argument des `mysqladmin`-Befehls stellt einen schnellen und einfachen Weg dar, um zu sehen, ob der `mysql`-Prozess läuft. Sie können auch eine andere Technik anwenden, indem Sie Folgendes auf die Kommandozeilenebene eingeben:

```
%> ps -aux | grep mysql
```

Wenn `mysqld` läuft, werden Sie den Prozess mit dieser Anweisung angezeigt bekommen.

Der `safe_mysqld`-Befehl ist das beste Mittel, um MySQL hochzufahren. Stürzt es ab, so wird es automatisch neu gestartet. Sie können MySQL auch starten, indem Sie den `mysqld`-Befehl verwenden. Das ist allerdings bei Produktionsumgebungen nicht sehr empfehlenswert, da es nicht automatisch neu gestartet wird.

Zum Herunterfahren benutzen Sie den `mysqladmin`-Befehl mit dem `shutdown`-Argument, und zwar folgendermaßen:

```
cd mysql
bin/mysqladmin -p shutdown
```

Dies fährt MySQL sicher herunter. Der `kill`-Befehl bietet eine drastischere Art, MySQL herunterzufahren. Auch dies wird jedoch nicht empfohlen, da es zur Verfälschung von Daten kommen kann.

Bei den meisten Server-Daemons wird empfohlen, deren Prozess gleichzeitig mit dem des Servers zu beginnen und zu beenden. Verwenden Sie dazu das `mysql.server`-Script mit dem `start`-Argument (`mysql.server start`), das sich im `support_files`-Verzeichnis befindet. Benutzen Sie dieses Script im `rc.d`-Verzeichnis. Halten Sie sich dabei an die Anleitung von Linux.

MySQL unter Windows benutzen

Wie bei Linux läuft der MySQL-Server auch hier im Hintergrund. Um den Server auf einer Windows-Plattform zu starten, doppelklicken Sie die `mysqld.exe`-Datei im `\mysql\bin`-Verzeichnis.

Um den Service zu beenden, geben Sie den `mysqladmin shutdown`-Befehl am DOS-Prompt ein. Dies fährt den Server-Prozess langsam herunter. Eine drastischere, vielleicht sogar schädliche Methode, ist der Gebrauch des Task Manager. Hierbei kann es zur Verfälschung von Daten kommen. Daher wird diese Methode nicht empfohlen.

Erwähnenswert ist, dass im Unterschied zum Linux-Daemon der Windows-Service eine kleines Speicherloch aufweist. Nach einer Weile wird dieses Loch Probleme verursachen. Das System kann sich sperren, so dass die Programme nicht mehr laufen. Um dieses Problem zu lösen, starten Sie MySQL alle paar Wochen neu. Gemäß der Dokumentation hat das TcX-Entwicklungsteam dieses Problem gelöst. In der nächsten Ausgabe wird dieses Problem nicht mehr auftauchen.

Damit Windows NT den MySQL-Daemon schon beim Start automatisch startet, klicken Sie in der Systemsteuerung auf Services – jetzt sollten Sie den `mysqld`-Service vor sich haben. Aktivieren Sie diesen, indem Sie die Box anklicken, damit der Prozess automatisch startet. Unter Windows 95/98 müssen Sie eine Verknüpfung auf die `mysqld.exe` in das Autostart-Verzeichnis einfügen.

MySQL benutzen – die Kommandozeile

Nun da MySQL läuft, wird es Zeit, dass Sie Ihre Datenbank ausprobieren. Das Kommandozeilen-Interface von MySQL kann beim ersten Mal entmutigen – besonders wenn Sie an GUIs gewöhnt sind, die die anderen Datenbanken der gleichen Größenordnung wie MySQL anbieten.

Um das Kommandozeilen-Interface unter Linux oder Windows zu starten, vergewissern Sie sich, dass Sie sich im `mysql`-Verzeichnis befinden. Unter Windows müssen Sie den DOS-Prompt verwenden. Geben Sie Folgendes auf der Kommandozeile ein:

```
bin/mysql -p
```

Jetzt sollten Sie Folgendes vor sich haben:

```
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2 to server version: 3.23.23
```

```
Type 'help' for help.
```

```
mysql>
```

Nachdem die Ausgabe erschienen ist, stehen Sie mit einem leeren Prompt da (siehe Abbildung 2.4). An dieser Stelle kann es für jemanden, der MySQL zum ersten Mal benutzt, zu Problemen kommen. Hier werden meistens alle Befehle, mit denen Sie Ihre Daten bearbeiten können, eingetragen. Das kann ein Fluch oder ein Segen sein – je nachdem, wie Sie es sehen. Ohne einen guten SQL-Befehl kommen Sie nicht weit.

Wenn Sie ein Kommandozeilen-Prompt haben, benötigen Sie kein GUI. Sie könnten jetzt natürlich behaupten, es sei gut, ein GUI zu haben; es vereinfache die Dinge. Mit ein paar Mausklicks können Sie alle Ihre Tabellen und wie diese sich aufeinander beziehen, überblicken. Mit einem GUI können Sie Genehmigungen und aktive Verbindungen einsehen. Sie können mit einem GUI zwar allerhand bewerkstelligen, jedoch können Sie nicht von einem entfernten Rechner aus Ihre Datenbank schnell und effektiv verwalten. Die meisten Windows-Administratoren benutzen PC AnyWhere oder etwas Ähnliches, um Ihre Server zu verwalten. Obwohl diese Programme recht gut sind, lassen sie doch sehr zu wünschen übrig – besonders bei langsamen Internet-Verbindungen.

Durch das Kommando-Prompt werden diese Unzulänglichkeiten beseitigt. Durch eine einfache Telnet-Sitzung können Sie Ihre Datenbank schnell und effektiv von einem entfernten Rechner aus verwalten. Auf diese Weise können Sie eine Datenbank erstellen, löschen und füllen, als wenn Sie direkt auf dem Rechner arbeiten würden, auf dem der MySQL-Server läuft.

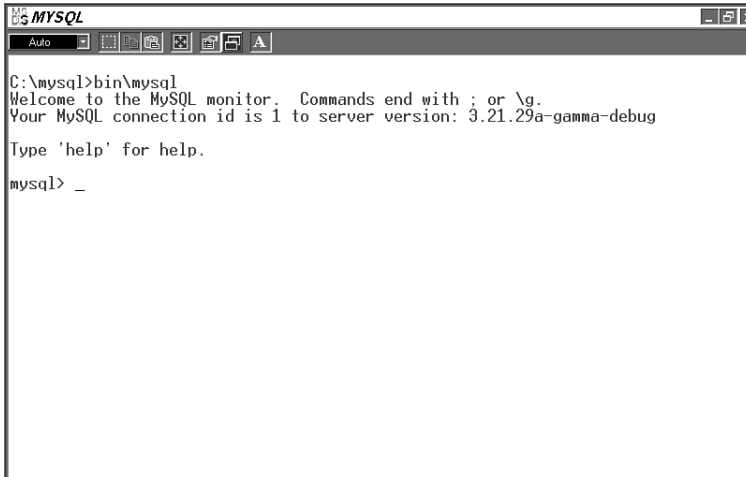


Abbildung 2.4:
Der MySQL-
Monitor

Angenommen Sie haben sich als `root` eingeloggt. Werfen Sie nun einen Blick auf die in der Standardinstallation existierenden Datenbanken. Wenn Sie sehen wollen, welche Datenbanken dieser Server enthält, müssen Sie Folgendes eingeben:

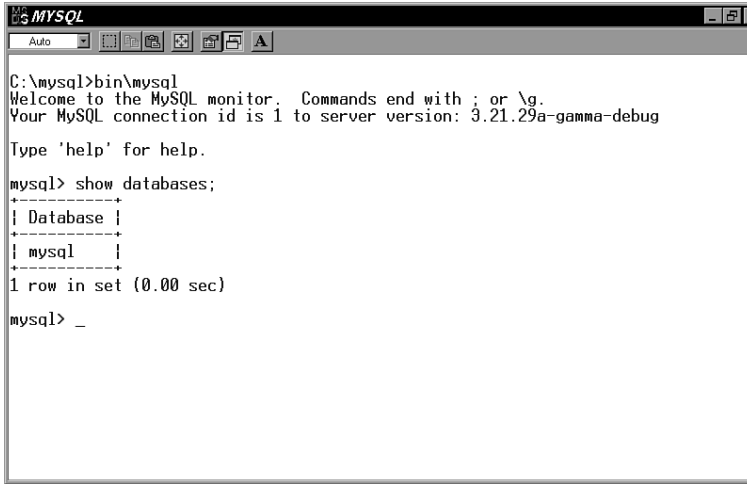
```
show databases;
```

Ihr Output sollte dem der Abbildung 2.5 gleichen.

Damit die eingegebene SQL-Anweisung ausgeführt wird, müssen Sie am Zeilenende ein Semikolon (;) oder `\g` setzen. Dies übermittelt dem MySQL-Monitor, dass Sie Ihre Anweisung beendet haben und diese nun ausgeführt werden soll. Das Drücken der `Return`- bzw. `Enter`-Taste verursacht lediglich einen Zeilenwechsel. Dadurch können Sie eine lange SQL-Anweisung lesbar eingeben. Die `Recall`-Taste ist ein weiteres sehr nützliches Feature. Unter Linux ruft der `↑` Ihre zuletzt eingegebenen Zeilen wieder auf, und zwar genau so, wie sie es von der Konsole her gewöhnt sind. Der MySQL-Monitor benutzt eine andere History-Datei als das Betriebssystem, so dass nur die Befehle wieder aufgerufen werden, die auf MySQL-Promptebene eingegeben wurden. Für die Benutzer von Windows sieht es schlecht aus. Denn hier gibt es keine History-Recall-Taste (nicht einmal die `F3`-Taste – die standardgemäße Rückruf-Taste für DOS). Die `Recall`-Taste ist sehr angenehm, besonders wenn Sie in einer langen SQL-Anweisung einen Fehler gemacht haben. Sie macht lästiges Wiedereintippen überflüssig.

Um mit einer der aufgelisteten Datenbanken zu arbeiten, müssen Sie dem MySQL-Monitor übermitteln, welche Datenbank er benutzen soll. Der Befehl hierfür ist ganz einfach. Geben Sie Folgendes ein, um die `mysql`-Datenbank zu benutzen:

```
USE mysql;
```



```

C:\mysql>bin\mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 3.21.29a-gamma-debug

Type 'help' for help.

mysql> show databases;
+-----+
| Database |
+-----+
| mysql    |
+-----+
1 row in set (0.00 sec)

mysql> _

```

Abbildung 2.5:
Ausgabe existie-
render Daten-
banken mit dem
MySQL-Monitor

Die Ausgabe sollte Abbildung 2.6 gleichen.

Um die Struktur oder das Schema einer Datenbank einzusehen, erlassen Sie folgenden Befehl:

```
SHOW TABLES FROM mysql;
```

Die Ausgabe sollte Abbildung 2.7 gleichen.

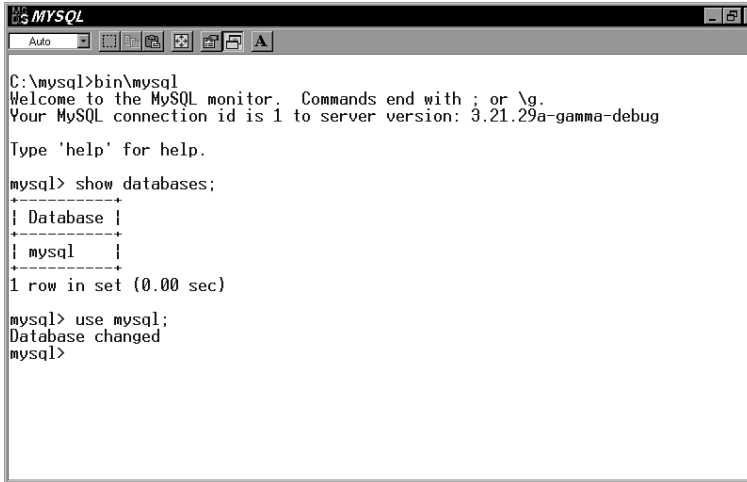
Dieser einfache Befehl liefert eine Auflistung aller Tabellen der ausgewählten Datenbank. Der folgende Befehl zeigt eine ausführliche Auflistung der Spalten Ihrer Datenbank.

```
SHOW COLUMNS FROM user;
```

Diesmal sollte Ihre Ausgabe Abbildung 2.8 gleichen.

Diese Anweisung zeigt die Beschreibungen der Felder in der Datenbanktabelle an. Sie gibt den Typ, den Standardwert, ob NULL oder nicht NULL, und sämtliche Schlüssel-Felder an. Dies ist eine sehr nützliche Anweisung, die im Allgemeinen auch häufig benutzt wird.

Wie Sie sehen, ist die Kommandozeile ein großartiges Werkzeug bei der Arbeit mit Ihrer Datenbank. Es mag vielleicht ein wenig dauern, bis Sie sich an die Kommandozeile gewöhnen, aber letztendlich ist sie sehr nützlich – schnell, stark und zuverlässig.



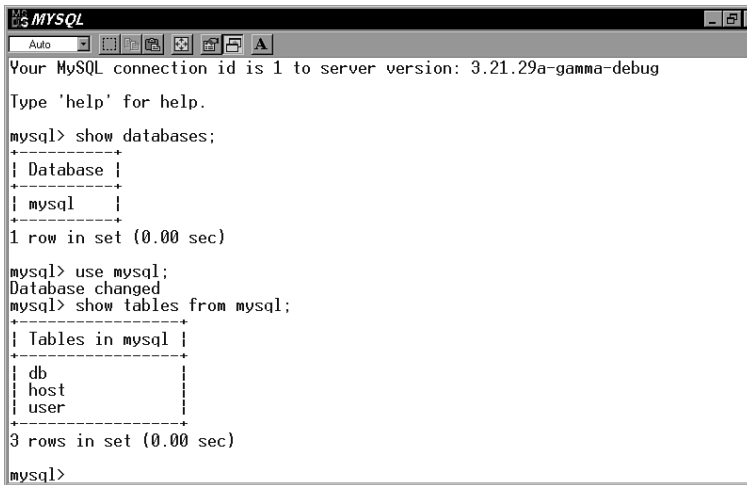
```
C:\mysql>bin\mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1 to server version: 3.21.29a-gamma-debug

Type 'help' for help.

mysql> show databases;
+-----+
| Database |
+-----+
| mysql    |
+-----+
1 row in set (0.00 sec)

mysql> use mysql;
Database changed
mysql>
```

Abbildung 2.6:
Ergebnis der
USE-Anweisung



```
Your MySQL connection id is 1 to server version: 3.21.29a-gamma-debug

Type 'help' for help.

mysql> show databases;
+-----+
| Database |
+-----+
| mysql    |
+-----+
1 row in set (0.00 sec)

mysql> use mysql;
Database changed
mysql> show tables from mysql;
+-----+
| Tables in mysql |
+-----+
| db               |
| host             |
| user             |
+-----+
3 rows in set (0.00 sec)

mysql>
```

Abbildung 2.7:
Anzeige der
Struktur der
mysql-Datenbank

Die vorgefertigte mysql-Datenbank

Am Ende Ihrer kurzen MySQL-Exkurses werden Sie wahrscheinlich gemerkt haben, dass bereits zwei Datenbanken in MySQL vorhanden sind: Zum einem die `mysql`-Datenbank und zum anderen eine `test`-Datenbank. Sie fragen sich jetzt wahrscheinlich, was diese Datenbanken eigentlich sollen und wie sie verwendet werden. Im folgenden Abschnitt wird die `mysql`-Datenbank näher erläutert.

```

mysql> show columns from user;
+-----+-----+-----+-----+-----+-----+
| Field      | Type                | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| Host       | char(60)             |      | PRI |          |       |
| User       | char(16)             |      | PRI |          |       |
| Password   | char(16)             |      |     |          |       |
| Select_priv| enum('N','Y')         |      |     | N        |       |
| Insert_priv| enum('N','Y')         |      |     | N        |       |
| Update_priv| enum('N','Y')         |      |     | N        |       |
| Delete_priv| enum('N','Y')         |      |     | N        |       |
| Create_priv| enum('N','Y')         |      |     | N        |       |
| Drop_priv  | enum('N','Y')         |      |     | N        |       |
| Reload_priv| enum('N','Y')         |      |     | N        |       |
| Shutdown_priv| enum('N','Y')        |      |     | N        |       |
| Process_priv| enum('N','Y')         |      |     | N        |       |
| File_priv  | enum('N','Y')         |      |     | N        |       |
+-----+-----+-----+-----+-----+-----+
13 rows in set (0.00 sec)

mysql>

```

Abbildung 2.8:
Auflistung der
Spalten der user-
Tabelle

Die `mysql`-Datenbank ist die Standarddatenbank. Diese Datenbank speichert alle Berechtigungen. Wie das MySQL Privilegien-System funktioniert, wird ausführlich an Tag 17 erklärt. Im Moment sollten Sie sich einfach merken, dass die `mysql`-Datenbank alle Benutzer, die Datenbank und Host-Privilegien speichert. Das Verändern oder Löschen einer dieser Tabellen verursacht Probleme. Deshalb wird empfohlen, die Standardstruktur der Tabellen nicht zu ändern.

Zusammenfassung

MySQL ist ein sehr leistungsstarkes RDBMS, das unter Windows und Linux ganz einfach zu installieren ist. Der MySQL-Monitor ermöglicht Ihnen, eine MySQL-Datenbank von einem entfernten Rechner aus zu benutzen, ohne dabei an Leistungsstärke oder Schnelligkeit einzubüßen. Grundlegende Verwaltungsangelegenheiten wie das `root`-Passwort zu ändern, den Server hoch- und runterfahren oder Informationen bezüglich der Datenbank anzuzeigen, sind leicht durchzuführen.

Fragen und Antworten

- F** Was mache ich, wenn ich das `root`-Passwort vergesse?
- A** Zuerst *loggen Sie sich als dieselbe Person, die den `mysqld`-Daemon betreibt, ein (wahrscheinlich `root`)*. Danach beenden Sie den Prozess, indem Sie `kill` einsetzen. Starten Sie MySQL mit den folgenden Parametern neu:

Beim nächsten Einloggen können Sie Ihr neues Passwort verwenden.

- F** Wie verändere ich die Startoptionen von MySQL, wenn ich es mit einem `mysql.server`-Eintrag im `rc.d`-Verzeichnis starte?
- A** *mysql.server ist ein Script, das den `mysqld`-Befehl enthält. Um dem Server-Daemon Optionen wie »logging« und »debugging« hinzuzufügen, müssen Sie dieses Script editieren. Benutzen Sie Ihren bevorzugten Editor, öffnen Sie die Datei und platzieren Sie die Argumente an den passenden Stellen.*
- F** Wo werden in einer MySQL-Datenbank die Daten gespeichert?
- A** *MySQL speichert Daten in Dateien, die Sie im `data/Datenbankname`-Verzeichnis finden, wobei `Datenbankname` der Name der Datenbank ist. Es gibt drei Dateitypen: `.ISM`, `.FRM` und `.ISD`. Die `.FRM`-Datei enthält das Tabellenschema, die `.ISD`-Datei enthält die Daten und die `.ISM`-Datei sichert den schnellen Zugriff zwischen diesen beiden.*

Übungen

1. Lassen Sie alle in der Datenbank befindlichen Daten erscheinen. Benutzen Sie dazu die Kommandozeile.
2. Überprüfen Sie, ob der `mysqld`-Daemon läuft. Wenden Sie zwei verschiedene Methoden an.
3. Verbinden Sie sich mit einem entfernten Rechner, und fahren Sie den MySQL-Server hoch und wieder runter.
4. Benutzen Sie den MySQL-Monitor auf einem entferntem Rechner (z.B. Ihrem Web-Server).



Das Entwerfen Ihrer ersten Datenbank

**Woche
1**

Das wichtigste beim Erstellen einer Datenbank ist das Design. Ist das Design gut, kann die Datenbank in der Geschäftswelt von großem Nutzen sein. Der heutige Tag behandelt Folgendes:

- ▶ den Design-Prozess
- ▶ die verschiedenen Typen von Beziehungen
- ▶ Einführung in die Musterdatenbank

Der Design-Prozess

Mit einem guten Design steht und fällt die Datenbank. Um eine gut funktionierende Datenbank zu erstellen, muss man sich erst ein paar Gedanken über ihr Design machen. Eine gut entworfene Datenbank lässt sich einfach und gut ausbauen und weiterentwickeln. Informationen in einer gut entworfenen Datenbank zu verwalten, ist ein wahrer Segen. Leider nehmen sich die meisten Leute nicht genügend Zeit für das Design einer Datenbank. Sie beginnen in medias res, erstellen Tabellen und Felder für ihre momentanen Bedürfnisse, ohne auch nur einen Schritt weiter zu denken. Diese Technik führt letztendlich zu nichts. Sie erhalten eine schlecht strukturierte Datenbank, aus der man nur sehr mühsam Informationen zurückbekommt. Eine solche miserabel konstruierte Datenbank wird nie im Leben den Bedürfnissen einer Firma gerecht werden.

Das Erstellen einer Datenbank ist ungefähr so, als würden Sie ein Haus bauen. Die Bauunternehmen errichten kein Haus ohne Plan. Der Architekt fertigt einen Plan für das Bauunternehmen an, nach dem das Haus gebaut wird. Das Bauunternehmen legt besonders Wert auf den Grundstein des Hauses, weil es ohne diesen einstürzen würde. Die gleichen grundlegenden Prinzipien sind auch beim Errichten einer Datenbank zu beachten.

Sie werden beide Rollen spielen – den Architekten und den Baumeister. Als Architekt werden Sie den Entwurf der Datenbank anfertigen. Sie müssen entscheiden, welche Informationen Sie speichern und aufnehmen möchten. Zudem müssen Sie die Beziehungen zwischen den Tabellen, die sie einrichten werden, definieren. Das ist lebenswichtig für eine gute, solide relationale Datenbank.

Als Baumeister werden Sie die SQL-Anweisungen eingeben, mit denen Sie die Datenbank dann erstellen. Sie werden entscheiden, welche Datentypen zu benutzen sind, um die Daten effektiv und richtig zu speichern. Damit legen Sie den Grundstein für Ihre Datenbank. Um eine solide Grundlage für Ihre Datenbank zu erstellen, müssen Sie wissen, welche Typen wann zu benutzen sind und wie Sie die richtigen Beziehungen aufbauen. Ich habe eine Reihe von Schritten aufgestellt, die Ihnen helfen, ein korrektes und vollständiges Design zu erstellen. Im ersten Schritt wird der momentane Geschäftsablauf definiert. Im nächsten Schritt definieren Sie die Geschäftsobjekte und die Geschäftsre-

gel und wie sie sich auf diese Objekte beziehen. Danach entwerfen Sie das Layout der Datenbank. Anschließend definieren Sie die Beziehungen zwischen den Tabellen sowie die Datentypen, die Sie in jedem Feld speichern wollen. Wenn Sie nach all diesen Schritten Ihre Arbeit gründlich überprüft haben, können Sie die Datenbank erstellen.

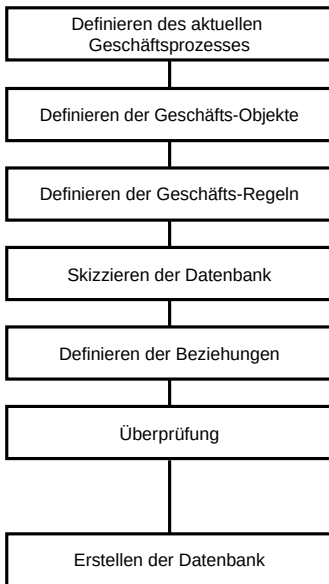


Abbildung 3.1:
Der Datenbank-Entwurfsprozess

Das Definieren des Geschäftsablaufs

Der erste Schritt beim Entwerfen einer Datenbank besteht darin, Einblick in den momentanen Geschäftsablauf zu erhalten. Unter Geschäftsablauf versteht man die Art und Weise, wie ein Geschäft seine Aufgaben erfüllt, um seine Ziele zu erreichen. Ein Online-Buchladen könnte beispielsweise folgenden Geschäftsablauf haben:

1. Eine Bestellung wird von einem Kunden via Internet aufgegeben.
2. Die Kreditkarte wird verifiziert.
3. Das Buch wird aus dem Inventar herausgenommen und die Bestellung an die Versandabteilung weitergegeben.
4. Die Versandabteilung verpackt das Produkt, prüft die Adresse und schickt das Päckchen raus.

In einigen Fällen werden Sie einen schon computergestützten Prozess aktualisieren, in anderen wiederum einen computergestützten Prozess auf der Basis eines Entwurfs erstellen.

Es gibt viele verschiedene Techniken, die Ihnen helfen, ein Verständnis für den Geschäftsablauf zu entwickeln. Am hilfreichsten ist, Leute zu interviewen, die jeden Tag mit diesem System arbeiten und mit dem Geschäftsablauf vertraut sind. Es kann sein, dass Sie mehrere Personen befragen müssen, um den Ablauf ganz und gar zu verstehen. Darauf sind wir in diesem Buch nicht eingegangen. Ein wirklich gutes Buch, das Sie durch diesen gesamten Ablauf führt, ist *Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design* von Michael J. Hernandez (veröffentlicht bei Addison Wesley).

Es ist unerlässlich, dass Sie diesen Ablauf voll und ganz verstehen. Nur dadurch werden Sie ein Verständnis für alle involvierten Objekte entwickeln. Das ist der Grundstein für Ihre Datenbank.

Das Definieren der Geschäftsobjekte

Nachdem Sie den Geschäftsablauf definiert haben, bestimmen Sie nun die Geschäftsobjekte. Geschäftsobjekte sind Komponenten, aus denen der Geschäftsablauf zusammengesetzt ist. Im letzten Beispiel sind das Buch und der Kunde Geschäftsobjekte. Geschäftsobjekte beinhalten diejenigen Informationen, die Sie in Ihrer Datenbank pflegen wollen. Es handelt sich hier um einen zweiteiligen Prozess. Der erste Teil besteht darin, das Objekt zu identifizieren, der zweite, die Felder, die diese Objekte beschreiben, zu erstellen.



Ein *Geschäftsobjekt* stellt eine Komponente des Geschäftsablaufs dar. Es ist eine der Säulen, auf denen das Geschäft steht.

Die Geschäftsobjekte sind normalerweise leicht zu identifizieren. Manchmal allerdings sind sie nicht so leicht zu erkennen. Im letzten Beispiel konnten Sie ganz leicht das Buch und den Kunden als die Geschäftsobjekte ausmachen. Aber wie sieht es mit der Transaktion aus, die stattfindet, wenn ein Kunde ein Buch wirklich kauft? Die Transaktion enthält wesentliche Informationen, ist aber nicht ohne Weiteres als Geschäftsobjekt auszumachen. Aus diesem Grund ist eine vollständige Kenntnis des Geschäftsprozesses unerlässlich, um eine gute Datenbank zu erstellen. Der zweite Teil dieses Prozesses besteht darin, Felder und Adjektive, die die Geschäftsobjekte beschreiben, zu erstellen bzw. aufzulisten. Denken Sie dabei an Dinge, die mit dem Geschäftsobjekt verbunden sind. Um bei dem Buchladen-Beispiel zu bleiben – das Buchobjekt kann zusätzlich aus einem Titel-, Verleger-, Autor-, Preis-, Auflage-, Datum-Feld etc. bestehen. Das Transaktionsobjekt kann unter anderem durch Transaktionsdatum, Anzahl, Beschreibung und Zahlungsmethode näher bestimmt werden. Diese Felder definieren zusätzlich Ihr Objekt. Es sind ebenfalls Felder, die Sie in Ihrer Datenbank pflegen. Meiner Meinung nach ist es auch sehr nützlich, alle Adjektive, die das Objekt beschreiben, aufzulisten. Später eliminiere ich unnötige Adjektive und füge neue, die ich vorher vergessen hatte, hinzu.

Das Definieren der Geschäftsobjekte ist der eigentliche Anfang beim Erstellen Ihrer Datenbank. Später werden diese Objekte in Ihrer Datenbank zu Tabellen und die Beschreibungen in Ihrer Tabelle zu Feldern.

Das Definieren der Geschäftsregeln

Der dritte Schritt beim Entwerfen einer Datenbank besteht darin, die Geschäftsregeln festzulegen. Eine Geschäftsregel ist eine Anweisung oder eine Reihe von Anweisungen, die bestimmt, wie ein Geschäft geführt wird. Angewandt auf das Buchladen-Beispiel könnte man als eine mögliche Geschäftsregel definieren: Es gibt keine negativen Transaktionen. Selbstverständlich könnte es negative Transaktionen wie zum Beispiel Rückzahlungen geben, doch der Geschäftsführer könnte die oben genannte Anweisung zur Regel erheben. Ein weiteres Beispiel wäre: Jedes Mal, wenn eine Bestellung den Verarbeitungsanforderungen genügt, soll die Ware verschickt werden. Diese Art von Regel hilft, die notwendigen Beziehungen zwischen Geschäftsobjekten festzulegen.

Es gibt zwei Arten von Regeln. Die *festgelegte Regel* (eine vom Unternehmen vorgeschriebene Regel) und die *selbstverständliche Regel* (eine auf gesundem Menschenverstand basierende Regel). Eine festgelegte Regel, um beim Buchladen-Beispiel zu bleiben, wäre, dass ein Kunde mehr als eine Bestellung aufgeben kann. Eine selbstverständliche Regel wäre, dass jedes Buch einen Titel haben muss. Das alles mag dumm oder albern erscheinen, spielt aber eine wichtige Rolle beim Bestimmen der Datentypen für Ihre Felder und bei der Frage, ob ein Feld leer sein kann oder nicht.



Eine *festgelegte Regel* wird von einem Unternehmen definiert. Eine *selbstverständliche Regel* kann nicht von einem Unternehmen bestimmt werden, sie basiert normalerweise auf gesundem Menschenverstand.

Die besten Werkzeuge zum Festlegen der Geschäftsregeln sind immer noch Papier und Bleistift. Schreiben Sie jede Regel nieder – ob Sie diese dumm finden oder nicht. Ziehen Sie jemanden, der sich mit dem Geschäftsablauf auskennt, beim Festlegen der Regeln zu Rate. So jemand kann Ihnen wahrscheinlich Regeln nennen, die Sie nicht sofort wahrnehmen. Später, während des eigentlichen Erstellungsprozesses, wird Ihnen diese Ansammlung jede Menge Zeit sparen; jetzt hilft sie Ihnen zunächst einmal bei der Vorbereitung auf den nächsten Schritt.

Die Modellierung der Datenbank

Der nächste Abschnitt beim Entwerfen Ihrer Datenbank beinhaltet das Skizzieren Ihres Schemas. Das mag Ihnen zuerst wie Zeitverschwendung vorkommen; ich bin aber zu dem Schluss gekommen, dass alles viel einfacher und sinnvoller erscheint, wenn Sie es vor sich ausgebreitet sehen. Indem ich etwas vorher skizziert habe, konnte ich schon x-mal Fehler im Design aufdecken.

Auf dem heutigen Markt gibt es eine Menge Programme, die Sie bei Ihrem Datenbankentwurf unterstützen und den Entwurfsprozess automatisieren. Diese Programme sind gut, aber beim ersten Entwurf bevorzuge ich die altmodische Art und Weise. Auf diese Weise kann ich etwas schnell und einfach entfernen und hinzufügen. Nachdem Sie die Datenbank erstellt haben, sind die von diesen Programmen erzeugten Diagramme ein unschätzbares Werkzeug, wenn Sie Anfragen erstellen oder sich mit einem Datenbankschema vertraut machen möchten.

Nach Vollendung der Skizze ist es an der Zeit, einige Lücken auszufüllen. Es kann sein, dass Sie Lücken vor sich haben, die ausgefüllt werden müssen; oder vielleicht wird Ihnen jetzt klar, dass einige Beschreibungen, die Sie für ein Objekt verwendet haben, eigentlich besser zu einem anderen passen.

Wenn Sie all diese Veränderungen verdaut haben, sollten Sie beginnen, Ihren Feldern Datentypen zuzuordnen. Die Typen, die Sie zuweisen, und die Frage, ob ein Feld Null sein kann oder nicht, können teilweise von den im letzten Schritt definierten Geschäftsregeln bestimmt werden. Die von MySQL unterstützten Datentypen werden an Tag 7, »MySQL-Datentypen«, behandelt. Im Augenblick müssen Sie nur wissen, dass dies die Phase im Design-Prozess ist, in der Datentypen zugewiesen werden.

Wenn Sie diese Schritte ausgeführt haben, werden Sie den grundlegenden Aufbau Ihrer Datenbank vor sich haben. Die Tabellen und Spalten werden größtenteils definiert sein. Im nächsten Schritt wird die bestehende Struktur leistungsfähiger gemacht.

Das Festlegen von Beziehungen

Dies ist der letzte Schritt, bevor Sie Ihre Datenbank erstellen. Es ist nicht immer einfach, Beziehungen zwischen Tabellen festzulegen. Zuerst müssen Sie bestimmen, ob eine Beziehung überhaupt besteht. Falls dies der Fall ist, um welchen Typ von Beziehung es sich handelt.

Der einfachste Weg, Beziehungen zu bestimmen besteht darin, einen Blick auf die im vorherigen Schritt erstellten Diagramme zu werfen. Wählen Sie eine Tabelle aus und prüfen Sie, ob sie sich auf eine andere Tabelle bezieht oder mit ihr interagiert. In der Buchladen-Datenbank haben Sie zum Beispiel einen Kunden, ein Buch und ein Transaktionsobjekt. Ich würde mich zuerst auf den Kunden konzentrieren und überlegen, ob irgendwelche Beziehungen oder Interaktionen zwischen ihm und dem Buch bestehen. In diesem Beispiel gibt es eine Beziehung. Ein Kunde muss schließlich zuerst ein Buch aus Ihrem Laden kaufen, um ein Kunde zu werden. Danach würde ich mir die gleiche Frage in Bezug auf das Transaktionsobjekt stellen. Auch hier besteht eine Beziehung. Erwirbt ein Kunde ein Buch, so kommt es zu einer Transaktion. Schließlich würde ich mich auf das Buch konzentrieren und prüfen, ob es irgendwelche Beziehungen zu den anderen Objekten aufweisen kann. Zwischen Buch und Kunde besteht, wie bereits gesagt, eine Beziehung, nicht aber zwischen Buch und Transaktionsobjekt. Ein Buch

gibt es auch ohne Transaktion. Das Transaktionsobjekt interagiert mit dem Kunden, jedoch nicht mit dem Buch. Das alles mag am Anfang etwas verwirrend erscheinen, aber mit der Zeit und durch Erfahrung werden Sie sehr schnell und einfach Beziehungen erkennen.

Im nächsten Schritt dieses Prozesses werden Sie bestimmen, welche Beziehungstypen es gibt. In einer relationalen Datenbank gibt es drei Beziehungstypen: eins-zu-eins (1:1), eine-zu-vielen (1:n) und viele-zu-vielen (m:n).

Eins-zu-eins (1:1)

Bei einer Eins-zu-eins- Beziehung muss ein Datensatz in Tabelle 1 einen entsprechenden Datensatz in Tabelle 2 haben und umgekehrt. In Abbildung 3.2 wird dies verdeutlicht.

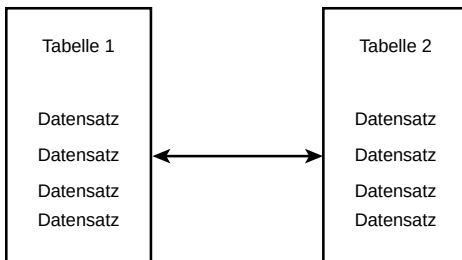


Abbildung 3.2:
Eine 1:1-Beziehung

Im Buchladenladen-Beispiel könnte eine Eins-zu-eins-Beziehung zwischen der Bestellungen- und der Transaktionstabelle bestehen. Ohne Bestellung keine Transaktion und ohne Transaktion keine Bestellung. Um diese Beziehung in der Datenbank zu erstellen, müssen Sie ein Feld hinzufügen, das diese Beziehung hält. Dies ist normalerweise Aufgabe eines Schlüsselfeldes. Schlüsselfelder werden an Tag 11, »Sperren und Schlüssel in MySQL«, ausführlicher behandelt. Im Moment müssen Sie nur wissen, dass ein *Schlüssel-Feld* unter anderem beim Bestimmen von Beziehungen behilflich ist.

Das Schlüsselfeld ist ein eindeutiges Feld innerhalb der Tabelle. Kein anderer Datensatz in diesem Feld hat den gleichen Wert. Dies dient dazu, einen Datensatz von allen anderen Datensätzen in dieser Tabelle zu unterscheiden.

Aus diesem Grund nennen die meisten Datenbank-Designer dieses Feld das ID-Feld. Die Books-Tabelle zum Beispiel würde ein Book_ID-Feld haben und die Transaction-Tabelle ein Trans_ID-Feld.

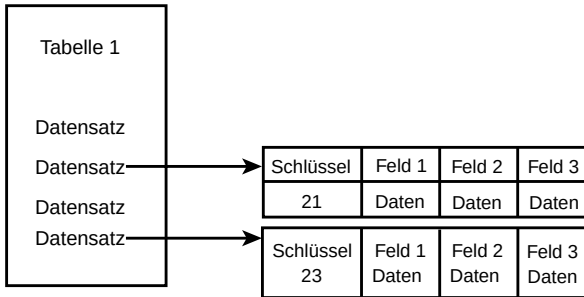


Abbildung 3.3:
Schlüsselfelder in einer 1: 1-
Beziehung

Um eine Eins-zu-eins-Beziehung festzulegen, müssen Sie eine Tabelle als primäre und die andere als sekundäre Tabelle bezeichnen. Im Allgemeinen ist dies eine willkürliche Entscheidung. Um es kurz zu machen, wählen Sie, wenn Sie der Datenbank einen neuen Datensatz hinzufügen, die zuerst betroffene Tabelle als primäre Tabelle. Diese primäre Tabelle enthält ein Schlüsselfeld. In unserem Beispiel wird die `Orders`-Tabelle ein eindeutiges `Order_ID`-Feld haben. Die sekundäre Tabelle enthält ebenfalls ein eigenes eindeutiges Schlüsselfeld sowie das Schlüsselfeld der Tabelle, mit der sie in Beziehung steht. Diese beiden Felder innerhalb der sekundären Tabelle sind auch eindeutig. Damit ist eine Eins-zu-eins-Beziehung hergestellt.

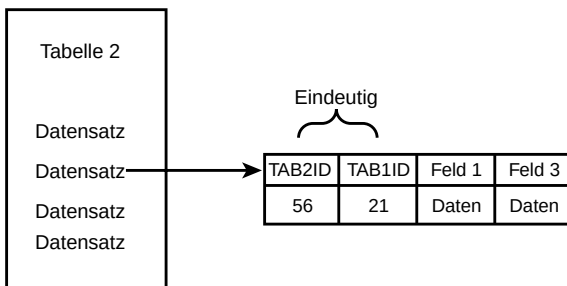
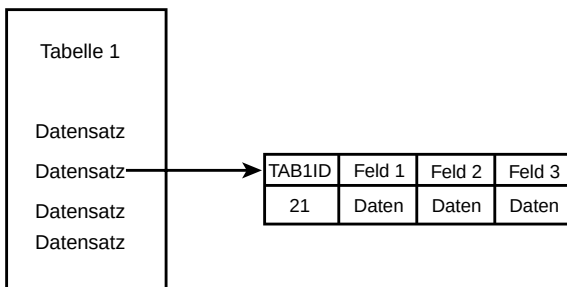


Abbildung 3.4:
Eine 1: 1-Beziehung in
einer Datenbank

Eine-zu-vielen (1:n)

Eine Eine-zu-vielen-Beziehung tritt auf, wenn ein Datensatz in Tabelle 1 viele entsprechende Datensätze in Tabelle 2 aufweist und wenn viele Datensätze in Tabelle 2 nur einem Datensatz in Tabelle 1 entsprechen (Abbildung 3.5). Im Buchladen-Beispiel besteht eine eine-zu-vielen-Beziehung zwischen der *Customers*- und der *Orders*-Tabelle. Ein Kunde kann viele Bestellungen aufgeben, aber all diese Bestellungen führen auf nur einen Kunden zurück. Abbildung 3.6 illustriert diesen Sachverhalt.

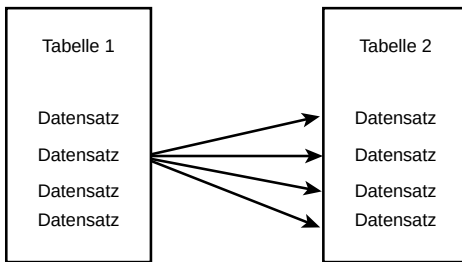


Abbildung 3.5:
Eine 1: n-Relation

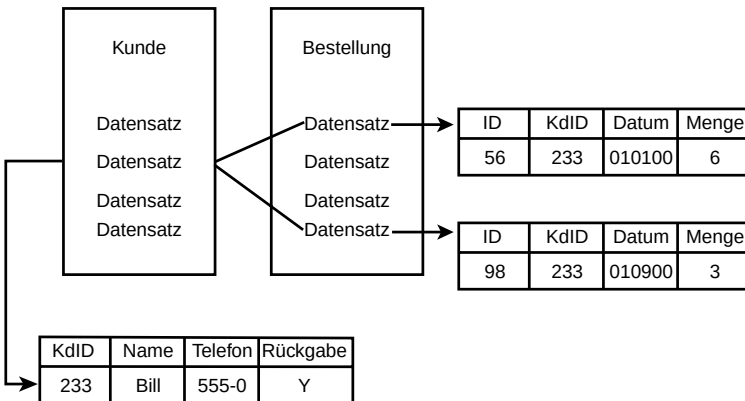


Abbildung 3.6:
Ein Kunde
mit mehreren
Bestellungen

Das Erstellen einer eine-zu-vielen-Beziehung unterscheidet sich kaum von dem Erstellen einer Eins-zu-eins-Beziehung. Auch hier müssen Sie Schlüssel benutzen. Zuerst müssen Sie eine primäre Tabelle auswählen. Im Gegensatz zur Eins-zu-eins-Beziehung ist diese Entscheidung hier nicht willkürlich. Die primäre Tabelle enthält einen einzelnen Datensatz und die sekundäre Tabelle mehrere Datensätze. Das Schlüsselfeld der primären Tabelle ist auch in der sekundären Tabelle vorhanden, allerdings ist es nicht eindeutig. Das Schlüsselfeld der sekundären Tabelle ist im Unterschied zum Fremdschlüssel eindeutig. Dies gestattet Ihnen, so viele Datensätze, wie Sie wollen, einzufügen und trotzdem jeden einzelnen Datensatz zu unterscheiden. Zudem können Sie

diese Datensätze zu einem einzelnen Datensatz einer anderen Tabelle in Beziehung setzen. Werfen Sie noch einmal einen Blick auf Abbildung 3.6, die diesen Aspekt illustriert.

Viele-zu-vielen (m:n)

Eine Viele-zu-vielen-Beziehung liegt vor, wenn Tabelle 1 einen Datensatz enthält, dem viele entsprechende Datensätze in Tabelle 2 gegenüberstehen und umgekehrt, das heißt, wenn Tabelle 2 einen Datensatz enthält, dem viele entsprechende Datensätze in Tabelle 1 gegenüberstehen (siehe Abbildung 3.7).

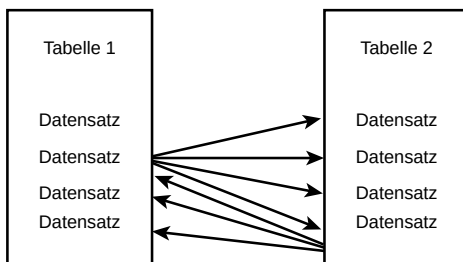


Abbildung 3.7:
Eine m: n-Beziehung

Die Viele-zu-vielen-Beziehung kann auch zu einigen Problemen führen. Sie könnte zum Beispiel zu überflüssigen (redundanten) Daten führen und damit die Regeln der Normalisierung missachten. Normalisierung wird an Tag 5, »Datennormalisierung« ausführlich behandelt. Zudem ist eine Viele-zu-vielen-Beziehung nicht einfach aufrechtzuerhalten. Es ist sehr riskant, Datensätze zu löschen und einzufügen. Unser Buchladen hat beispielsweise im ganzen Land zahlreiche Warenlager. Jedes Warenlager fasst einen großen Vorrat an Büchern. In einem Warenlager können sich viele Bücher befinden und genauso kann sich ein bestimmtes Buch in vielen Warenlagern befinden. Was passiert also, wenn Sie ein weiteres Warenlager hinzufügen? Sie müssten erneut jeden Buchtitel Ihrer Warenlager-Tabelle hinzufügen. Das könnte eine haarige Angelegenheit werden. Um diese Situation in den Griff zu bekommen, müssten Sie eine »Zwischentabelle« einrichten, die diese beiden Tabellen verbindet. Dadurch würden zwei Eine-zu-vielen-Beziehungen entstehen. Diese »Zwischentabelle« würde aus den primären Schlüsseln beider Tabellen bestehen. Wenn ein Buch in ein Warenlager gestellt wird, müsste ein weiterer Datensatz, der sich aus den Schlüsselfeldern des Buches und des Warenlagers zusammensetzt, der »Zwischentabelle« hinzugefügt werden. Um herauszufinden, welche Bücher sich in dem Warenlager befinden, könnten Sie diese »Zwischentabelle« befragen. Auf den ersten Blick kommt Ihnen das alles vielleicht als eine Komplizierung der Datenbank vor. Ich kann Ihnen aber versichern, dass sich diese Arbeit auf jeden Fall lohnt. Ohne eine »Zwischentabelle« zu implementieren, erweist sich eine Viele-zu-vielen-Beziehung als äußerst schwierig.

Nachdem die Beziehungen identifiziert wurden, sollten Sie diese jetzt in Ihr Schema einfügen, damit Sie nicht vergessen, diese Beziehungen auch tatsächlich zu implementieren, wenn Sie Ihre Datenbank erstellen. Sie sollten auch die neu erstellten Schlüsselfelder hinzufügen. Beachten Sie, dass ein Schlüsselfeld einen Datensatz eindeutig identifiziert. Es sollte nicht NULL sein. Nachdem Sie dies ausgeführt haben, kommen Sie zum nächsten Schritt des Design-Prozesses.

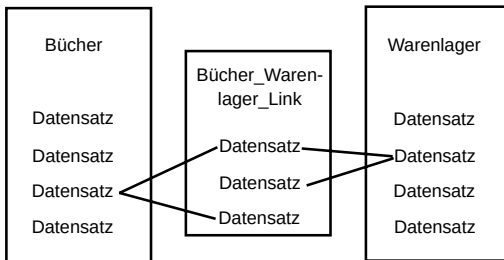


Abbildung 3.8:
Eine m: n-Beziehung wird in zwei 1: n-Beziehungen aufgelöst. Dies wird gemacht, um die Verwaltung zu vereinfachen und flexibel auf Änderungen zu reagieren – und um den Regeln der Normalisierung folgen zu können

Die letzte Phase – Erstellen einer Datenbank

Der letzte Schritt des Prozesses ist das eigentliche Erstellen der Datenbank. Sie sollten bereits genaue Kenntnis über den Geschäftsablauf sowie die Geschäftsobjekte und -regeln haben. Sie sollten das Schema Ihrer geplanten Datenbank vor sich haben. Das kann sehr hilfreich sein, wenn Sie Ihre Datenbank verändern oder abfragen. Jetzt ist der richtige Augenblick, um alles noch einmal genau zu überdenken. Checken Sie den Geschäftsablauf, prüfen Sie, ob noch irgendetwas fehlt. Gehen Sie die Geschäftsobjekte noch einmal durch, um sich zu vergewissern, dass Sie nichts vergessen haben. Sie sollten jetzt Felder und Tabellen hinzufügen oder streichen. Dadurch können Sie Ihr System verbessern.

Nachdem Sie alles überprüft haben, können Sie jedem der Felder Datentypen zuweisen. Die MySQL-Datentypen werden an Tag 7 vorgestellt. Indem Sie die passenden Datentypen zuweisen, unterstützen Sie nicht nur die festgelegten Regeln, sondern Sie machen damit auch Ihre Datenbank effektiver. Es ist auch von Vorteil, wenn Sie jeder Tabelle einen Schlüssel hinzufügen. Jede Tabelle sollte ein Schlüsselfeld beinhalten. Wenn Sie das alles auf Papier haben, können Sie nun mit dem Erstellen Ihrer Datenbank beginnen. Halten Sie sich auf jeden Fall an Ihren Entwurf. Improvisation ist nicht zu empfehlen. Das kann zu einem schlechten Design führen – was Sie ja gerade zu vermeiden suchen.

Nachdem Sie Ihre Datenbank erstellt haben, werden Sie Privilegien festlegen, Benutzer hinzufügen und unzählige verwaltungstechnische Aufgaben erledigen müssen.

Die Muster-Datenbank – die Meet_A_Geek-Datenbank

Um das, was Sie heute gelernt haben zu festigen, werden Sie nun den neu erworbenen Stoff an einem Musterprojekt anwenden. Ihre Aufgabe ist es, eine Datenbank für einen Online-Dating-Service namens Meet_A_Geek zu erstellen. Meet_A_Geek basiert auf einer realen Website, die MySQL benutzt. Die URL lautet: *http://www.meetageek.com*. Besuchen Sie ruhig mal diese Seite und schauen Sie, was sie anzubieten hat. Sie werden dieses Projekt in den folgenden Lektionen fortführen. Jetzt zunächst mal zum Design.

Der erste Schritt besteht darin, den Geschäftsablauf zu bestimmen. Nachdem Sie den Kunden interviewt haben, können Sie folgenden Plan erstellen:

1. Ein potentieller Kunde besucht Ihre Seite und möchte eine Anzeige aufgeben.
2. Der Kunde wird aufgefordert, einen Mitgliedsantrag auszufüllen. Dieser Antrag beinhaltet allgemeine persönliche Daten sowie einen Fragebogen mit spezifischen Fragen bezüglich der Präferenzen und Abneigungen des Kunden.
3. Nachdem der Kunde den Antrag komplett ausgefüllt hat, ist es ihm gestattet, die Datenbank nach potentiellen Dates abzufragen.
4. Nachdem ein Kunde Rückmeldung erhalten hat, kann er Blumen, eine Schachtel Pralinen oder irgendein anderes Geschenk an eines der potentiellen Dates, die er in der Datenbank gefunden hat, schicken. Der Kunde kann dann ein Hyperlink anklicken, das ihn zu einem Geschenkkatalog führt.
5. Sobald der Kunde ein Geschenk ausgesucht hat, kann er es per Internet kaufen und es zusammen mit einer Nachricht direkt dem oder der Auserwählten schicken. Geschenkgeber und -empfänger müssen allerdings beide Mitglieder dieser Website sein.

An Hand dieses Prozesses können Sie ganz einfach einige Geschäftsobjekte ausmachen. Da wäre zunächst das Customers-Objekt, das Products-Objekt und schließlich das Orders and Transactions-Objekt.

Im zweiten Schritt beschreiben Sie Ihre Objekte. Der Meet_A_Geek-Client hat Ihnen das Antragsformular für potentielle Kunden geliefert. Damit haben Sie schon einen sehr guten Anfang gemacht. Die Beschreibung eines Kunden müsste Ihnen jetzt leichter fallen. Werfen Sie jetzt einen Blick auf folgende Abbildung:

Book	Customer	Transaction
Author	Name	Trans Date
Title	Address	Amount
Publisher	Phone #	Description
Price	First Timer	Quantity
	Age	Item
	Gender	

Abbildung 3.9:
Meet_A_Geek-Antragsformular

Anhand dieses Antrags können Sie Ihr `Customers`-Objekt mit den folgenden Merkmalen beschreiben:

- ▶ Vorname, Nachname und Anfangsbuchstaben des Zweitnamens (First Name, Last Name, Middle Initial)
- ▶ E-Mail-Adresse (E-mail)
- ▶ Telefon-Nummer (Phone)
- ▶ Alter (Age)
- ▶ Geschlecht (Gender)
- ▶ Haarfarbe (Hair-Color)
- ▶ Augenfarbe (Eye-Color)
- ▶ Nationalität (Race)

Der Fragebogen liefert weitere Punkte, um Ihren Kunden zu beschreiben (siehe Abbildung 3.10)

Aus dem Fragebogen können Sie folgende Informationen entnehmen :

- ▶ Lieblingsbeschäftigung
- ▶ Lieblingsfilm
- ▶ Beruf
- ▶ Raucher (Ja/Nein)

Nach dem Sammeln dieser Informationen entspricht Ihre `Customers`-Tabelle Abbildung 3.11.

The screenshot shows a web browser window titled "Meet - Microsoft Internet Explorer". The address bar is empty. The menu bar includes "File", "Edit", "View", "Favorites", "Tools", and "Help". The main content area displays a form titled "Meet-A-Geek Questionnaire". The form contains the following fields and controls:

- Favorite Activity:** A dropdown menu with "Programming" selected.
- Favorite Movie:** A text input field.
- Occupation:** A dropdown menu with "Consultant" selected.
- Smoker:** Two radio buttons, "No" (selected) and "Yes".
- Buttons:** "Reset" and "Submit" buttons at the bottom left.

The status bar at the bottom shows "Done" and "My Computer".

Abbildung 3.10:
Meet_A_Geek-
Kunden-Frage-
bogen

This screenshot is identical to the one in Abbildung 3.10, showing the "Meet-A-Geek Questionnaire" form in Microsoft Internet Explorer. The form fields and controls are the same: Favorite Activity (Programming), Favorite Movie (text input), Occupation (Consultant), Smoker (No/Yes radio buttons), and Reset/Submit buttons.

Abbildung 3.11:
Das Kunden-
Objekt

Die Beschreibung der anderen Objekte gestaltet sich ein wenig schwieriger. Beginnen wir mit dem `Products`-Objekt. Jedes Produkt hat einen Preis und einen Namen. Weitere Punkte, die man mit dem Produkt in Verbindung bringen kann, sind beispielsweise Produktbeschreibung und Hersteller. Wenn es sich anbietet, können Sie auch ein Bild des Produkts in Ihre Liste aufnehmen. Das Ergebnis sieht folgendermaßen aus:

Produkte (Products)
Name (Name)
Preis (Price)
Produktbeschreibung (Description)
Hersteller (Manufacturer)
Bild (Picture)

Tabelle 3.1: Das Produkt-Objekt

Das Gleiche machen Sie mit den restlichen Objekten, bis Sie mit Ihren Beschreibungen zufrieden sind. Beachten Sie dabei, dass Sie nicht nur die Objekte beschreiben, sondern auch nach Informationen suchen, die zum Erreichen des Geschäftsziels beitragen. Sie haben beispielsweise Ihrer `Products`-Tabelle eine Bildbeschreibung hinzugefügt. Ein Bild beschreibt nicht unbedingt Ihr Produkt, Sie wissen aber, dass auf Ihrer Website ein Bild des Artikels erscheinen muss – darin besteht die Verbindung zum Produkt. Werfen Sie einen Blick auf Tabelle 3.2 und prüfen Sie, ob Sie das Gleiche zusammengestellt haben.

Bestellungen (Orders)	Transaktionen (Transactions)	Versender (Shippers)	Produkte (Products)	Kunden (Customers)
Bestelldatum (Order Date)	Transaktionsdatum (Transaction Date)	Name (Name)	Name (Name)	Vorname (First Name)
Menge (Quantity)	Versender (Shipper)	Ort (Location)	Produktbeschreibung (Description)	Nachname (Last Name)
(Item)	Gebühr bezahlt (Amount Paid)	Versandart (Shipping Method)	Preis (Price)	Adresse (Address)
Kunde (Customer)	Bestellnummer (Order Number)	Aktiv (Active)	Hersteller (Manufacturer)	Stadt (City)
Persönliche Nachricht (Custom Message)	Versanddatum (Ship Date)		Bild (Picture)	Bundesstaat (State)
				Zip (Zip)

Tabelle 3.2: Die vervollständigten Geschäftsobjekte

Bestellungen (Orders)	Trans- aktionen (Transactions)	Versender (Shippers)	Produkte (Products)	Kunden (Customers)
				E-Mail (Email)
				Alter (Age)
				Gender (Geschlecht)
				Augenfarbe (Eye Color)
				Haarfarbe (Hair Color)
				Lieblingsbeschäftigung (Favorite Activity)
				Lieblingofilm (Favorite Movie)
				Beschäftigung (Occup- ation)
				Raucher (Smoker)

Tabelle 3.2: Die vervollständigten Geschäftsobjekte (Forts.)

Man könnte diese Liste endlos erweitern; sie soll hier aber klein und übersichtlich gehalten werden. Der wichtigste Aspekt ist hier nicht der Inhalt, sondern der Nutzwert des Konzepts. Hoffentlich erkennen Sie jetzt die Wichtigkeit eines guten Datenbank-Designs.

Im nächsten Schritt definieren Sie Ihre Geschäftsregeln. Wenn Sie sich den Geschäftsablauf anschauen, können Sie einige Regeln bestimmen. Gesunder Menschenverstand und Ihre Erfahrung können Ihnen dabei ebenfalls behilflich sein. Wie Sie bereits wissen, haben nur Mitglieder Zugriff auf die Datenbank. Ihnen ist ebenfalls bekannt, dass ein Geschenk nicht verschickt wird, wenn es nicht bestellt wurde. Dies ist vielleicht offensichtlich, aber wie bereits gesagt, werden sogar die Regeln, die Ihnen albern und überflüssig vorkommen, beim Gesamt-Design sehr behilflich sein. Denken Sie sich ein paar weitere Regeln aus und schauen Sie sich anschließend die von mir zusammengestellte Liste an.

Meet_A_Geek-Geschäftsregeln:

- ▶ Nur Mitglieder haben Zugang zur Datenbank.
- ▶ Ein Geschenk wird erst verschickt, wenn es bestellt wurde.
- ▶ Ein Kunde muss einen Namen und eine Adresse haben.

- ▶ Ein Produkt muss einen Namen und einen Preis haben.
- ▶ Das Versanddatum kann nicht vor dem Bestelldatum liegen.
- ▶ Das Geschlecht des Kunden kann nur männlich oder weiblich sein.
- ▶ Der Kunde ist entweder Raucher oder Nichtraucher (eine Ja/Nein-Frage).
- ▶ Die bezahlte Summe entspricht dem Preis des Produkts mal die Bestellmenge.
- ▶ Die bezahlte Summe kann kein Minusbetrag sein.

Hören Sie der Einfachheit halber auf. Es gibt natürlich noch viel mehr Regeln, aber halten Sie sich an das, was Sie haben. Nun, da Sie Ihren Geschäftsablauf bestimmt, Ihre Geschäftsobjekte definiert und beschrieben sowie Ihre Geschäftsregeln festgelegt haben, sind Sie soweit, Ihre Datenbank zu modellieren. In Abbildung 3.12 sehen Sie das von mir erstellte Schema für Ihre momentane Datenbank. Vergleichen Sie es mit Ihrem eigenen.

Das Definieren von Beziehungen

Als Nächstes definieren Sie die Beziehungen. Dazu betrachten Sie zunächst das gesamte Bild, das Sie bis jetzt gezeichnet haben. Auf Anhieb können Sie eine 1:n-Beziehung zwischen Kunden und Bestellungen ausmachen. Ein Kunde kann viele Bestellungen aufgeben, aber all diese Bestellungen gehen auf diesen einen Kunden zurück. Eine weitere Beziehung besteht zwischen Produkten und Bestellungen. Es handelt sich hier um eine m:n-Beziehung. Eine Bestellung kann viele Produkte beinhalten und ein Produkt kann in vielen Bestellungen enthalten sein. Nachdem, was Sie über die m:n-Beziehungen gelernt haben, wissen Sie, dass Sie Ihrer Datenbank nun eine weitere Tabelle hinzufügen müssen. Eine weitere wichtige Beziehung besteht zwischen Bestellungen und Transaktionen. Für jede Transaktion gibt es eine Bestellung. Es handelt sich hier ganz eindeutig um eine 1:1-Beziehung.

Nun müssen Sie die Datentypen, die Sie benutzen werden, identifizieren. Da Sie Datentypen erst an Tag 7 behandeln werden, verwenden Sie einfache Typen, um auszudrücken, was eigentlich verwendet werden soll. Beginnen Sie mit der `Customers`-Tabelle.



In der `Customers`-Tabelle haben Sie die Felder `First_Name` und `Last_Name`. Aus Ihren Geschäftsregel geht hervor, dass diese Felder notwendig sind. Um diese Notwendigkeit in Ihrer Datenbank aufzuzeigen, ordnen Sie diesen Feldern den Wert `NOT NULL` zu. `NULL` kann am besten erklärt werden, indem ich Ihnen sage, was es nicht ist. Ein `NULL`-Wert ist nicht 0. Er ist keine leere Zeichenkette wie zum Beispiel » ». Er ist auch keine negative Zahl. Ein `NULL`-Wert ist nichts. Ein Feld, das keinen Wert enthält, wird als `NULL` bezeichnet. Wenn Sie nicht wollen, dass ein Feld leer ist, können Sie anweisen, dass es `NOT NULL` ist. Dadurch stellen Sie sicher, dass in diesem Feld ein Wert enthal-

ten ist. Da diese Felder aus Namen bestehen und Namen wiederum aus Buchstaben, werden diese Felder als Zeichen-Felder definiert. Das gleiche gilt für die Felder `Address`, `City`, `State` und `Zip`. Auch die `Email`-Adresse definieren Sie als Zeichen-Feld. Das `Age`-Feld kann nur eine positive Zahl sein. Die Felder `Gender`, `Race`, `Hair_Color` und `Eye_Color` können zunächst auch als Zeichen-Felder definiert werden. An Tag 7 werden Sie lernen, dass diese als ENUM (aufzählende) Typen effektiver sind. Lediglich das `Smoker`-Feld wird nicht als Zeichen-Feld definiert. Aus Ihren Geschäftsregeln geht hervor, dass das `Smoker`-Feld nur mit Ja oder Nein beantwortet werden kann; daher könnten Sie dieses als Boolean-Feld definieren. Ein *Boolean*-Typ ist entweder WAHR oder FALSCH.

Customers
First Name
Last Name
Address
City
State
Zip
E-mail
Age
Race
Gender
Eye Color
Hair Color
Favorite Activity
Occupation
Favorite Movie
Smoker

Abbildung 3.12:
Das entgültige Meet_A_Geek-Datenbank-Modell

Das Gleiche müssten Sie bei den übrigen Tabellen durchführen. Vergessen Sie an dieser Stelle also nicht, jeder Tabelle ein Schlüsselfeld hinzuzufügen. Nachdem Sie diese hinzugefügt haben, modellieren Sie Ihre Datenbank am besten noch einmal. Das ist die Basis für die morgige Lektion, in der Sie die Meet_A_Geek-Datenbank erstellen werden.

Zusammenfassung

Heute haben Sie die Bedeutung eines korrekten Datenbank-Designs kennen gelernt. Eine Datenbank ist wie ein Gebäude. Ein genauer Plan und ein korrektes Design sind unerlässlich, um dem Zahn der Zeit standzuhalten. Ich habe Ihnen den Design-Prozess

vorgestellt. Wenn Sie diesen einhalten, wird Ihre Datenbank sehr zuverlässig und solide sein. Der Ablauf besteht aus sechs aufeinander aufbauenden Schritten. Im ersten Schritt legen Sie den Geschäftsablauf fest. Danach definieren Sie die Geschäftsobjekte, aus denen letztlich die Tabellen, die eine Datenbank ausmachen, hervorgehen. Die Beschreibungen dieser Objekte ergeben die Felder. Im dritten Schritt bestimmen Sie die Geschäftsregeln. Diese Regeln bestimmen, wie die Daten gespeichert werden und wie die Geschäftsobjekte zueinander in Beziehung stehen. Anschließend modellieren Sie die Datenbank. Dies hilft Ihnen, die Struktur der Datenbank zu verinnerlichen. Auf diese Weise können Sie ganz einfach Mängel und Fehler im Design und der Logik erkennen. Im letzten Schritt identifizieren Sie die Beziehungen zwischen den Objekten. Sie haben erfahren, dass es drei Arten von Beziehungen gibt: eins-zu-eins (1:1), eins-zu-vielen (1:n) und viele-zu-vielen (m:n). Sie wissen jetzt, was diese Beziehungen bewirken und wie sie implementiert werden. Außerdem habe ich heute das Meet_A_Geek-Projekt vorgestellt. Dieses Projekt basiert auf einer realen Website, auf der MySQL eingesetzt wird. Sie haben die heute gelernten Techniken und Konzepte an diesem Musterprojekt angewandt.

Die heute vorgestellten Konzepte können bei jeder Datenbank angewandt werden. Sie wurden in diesem Buch behandelt, da meines Erachtens das Design beim Erstellen einer Datenbank am wichtigsten ist. Für Sie als zukünftigen MySQL-Administrator oder Programmierer, der ein Programm schreibt, um Zugang zu den in einer MySQL-Datenbank enthaltenen Daten zu erhalten, ist es unerlässlich, zu wissen, wie man eine Datenbank entwirft.

Fragen und Antworten

- F** Dieser Design-Prozess scheint Zeitverschwendung zu sein. Warum sollte ich so viel Zeit für das Design aufwenden, wenn ich sie doch viel besser für das Erstellen der Datenbank nutzen könnte?
- A** *Die auf das Design einer Datenbank verwendete Zeit ist auf jeden Fall gut angelegt. Beim Erstellen und Aufrechterhalten Ihrer Datenbank werden Sie dadurch jede Menge Zeit einsparen. Wenn die Datenbank nicht gut durchdacht und korrekt entworfen ist, werden Sie endlose Stunden damit zubringen, Felder und Beziehungen, die Sie möglicherweise vergessen haben, hinzuzufügen. Auch wird es mehr Zeit in Anspruch nehmen, Informationen abzufragen, da hastig erstellte Datenbanken häufig nicht den Regeln der Normalisierung entsprechen. Die Folge sind überflüssige Daten und Verschwendung von Speicherkapazität. Die zusätzlichen Stunden, die Sie für das Design aufwenden, werden sich für Sie auf alle Fälle auszahlen.*

F Wozu dient das Modellieren der Datenbank?

A *Das Modellieren der Datenbank dient mehreren Zwecken. Zum einen liefert es eine visuelle Darstellung der Logik und des Flusses Ihres Designs. Die ermöglicht Ihnen, die in Ihrem Design enthaltenen Fehler und Schwächen zu erkennen. Zum anderen bietet es eine gute Quelle, um Ihre Datenbank zu dokumentieren. Das Modell enthält sämtliche Tabellen, deren Beziehungen sowie alle Felder und Feldtypen. Es gibt keine bessere Quelle für eine Dokumentation als ein gutes Modell.*

Übungen

1. Im Meet_A_Geek-Projekt haben Sie mehrere Geschäftsobjekte definiert. Können Sie weitere Objekte bestimmen?
2. Erstellen Sie die noch fehlenden Geschäftsregeln für das Meet_A_Geek-Projekt.



Das Erstellen Ihrer ersten Datenbank

**Woche
1**

Das Erstellen einer Datenbank gehört wahrscheinlich zu den wichtigsten, aber am wenigsten genutzten MySQL-Funktionalitäten. MySQL bietet viele Möglichkeiten, eine Datenbank zu erstellen.

Heute werden Sie Folgendes lernen:

- ▶ die Befehle `CREATE` und `DROP`
- ▶ wie Sie das `mysqladmin`-Programm benutzen
- ▶ wie Sie Ihrer Datenbank Benutzer hinzufügen
- ▶ wie Sie die `Meet_A_Geek`-Datenbank erstellen

Die Befehle `CREATE` und `DROP`

Wenn Sie an die Befehle `CREATE` und `DROP` denken, sollten Sie sich darunter Raupen, Bagger, Kipper und Kräne vorstellen, weil diese die Werkzeuge sind, die Sie zum Erstellen Ihrer Datenbank benutzen. Obwohl sie selten angewandt werden, sind diese Befehle die wichtigsten. Bevor Sie einen dieser beiden Befehle erlassen, haben Sie sich hoffentlich ein paar Gedanken darüber gemacht.

Der Befehl `CREATE`

Es gibt viele verschiedene Methoden, Datenbanken in MySQL zu erstellen. Wenn Sie eine Datenbank erstellen, haben Sie das gesamte Layout normalerweise bereits vor sich. Im Normalfall wird die Tabelle direkt nach dem Erstellen der Datenbank hinzugefügt. Da dieses Buch aber dem Training dient, werden Sie alles Schritt für Schritt durchgehen.

Die erste Möglichkeit, eine Datenbank in MySQL zu erstellen, besteht darin, den SQL-Befehl `CREATE DATABASE>databasename` in den MySQL-Monitor einzugeben; wobei `databasename` für den Namen der Datenbank, die Sie gerade erstellen, steht. Führen Sie die folgenden Schritte aus, um diese Musterdatenbank zu erstellen:

Der Ablauf beim Erstellen einer Datenbank ist bei den meisten Betriebssystemen gleich. Falls etwas nicht in einem bestimmten Betriebssystem ausgeführt werden kann, so wird dies angemerkt.

Sie sollten Ihr `root`-Passwort für das MySQL-Datenbank-System geändert haben. Um das `mysqladmin`-Programm zu benutzen und den MySQL-Monitor zu starten, müssen Sie Ihr neues Passwort eingeben. Um der Kürze willen habe ich das Passwort-Argument `(-p)` bei den folgenden Anweisungen weggelassen.

1. Öffnen Sie die Konsole (MS-DOS-Eingabeaufforderung unter Windows bzw. eine Konsole Ihrer Wahl unter Linux).
2. Wechseln Sie ins `mysql`-Verzeichnis. Nachdem Sie einen symbolischen Link erstellt haben, können Sie Folgendes eingeben:

```
cd mysql
```

Wenn Sie keinen symbolischen Link erstellt haben, müssen Sie den ganzen Pfad folgendermaßen eingeben:

```
cd/usr/local/mysql
```

(vorausgesetzt, MySQL wurde im Standardverzeichnis installiert)



Symbolische Links werden im Allgemeinen zum Abkürzen benutzt. Sie können einen langen Pfad-Namen auf ein Wort verkürzen und es auf diese Weise dem Benutzer leichter machen.

3. Vergewissern Sie sich, dass der `mysqld`-Daemon läuft. Um dies zu überprüfen, geben Sie Folgendes ein:

```
bin/mysqladmin ping
```

4. Sind Sie sicher, dass der `mysqld`-Daemon läuft, starten Sie den MySQL-Monitor, indem Sie Folgendes auf Kommandozeilenebene eingeben:

```
bin/mysql
```

5. Geben Sie Folgendes am Monitor-Prompt ein:

```
CREATE DATABASE sample_db;
```

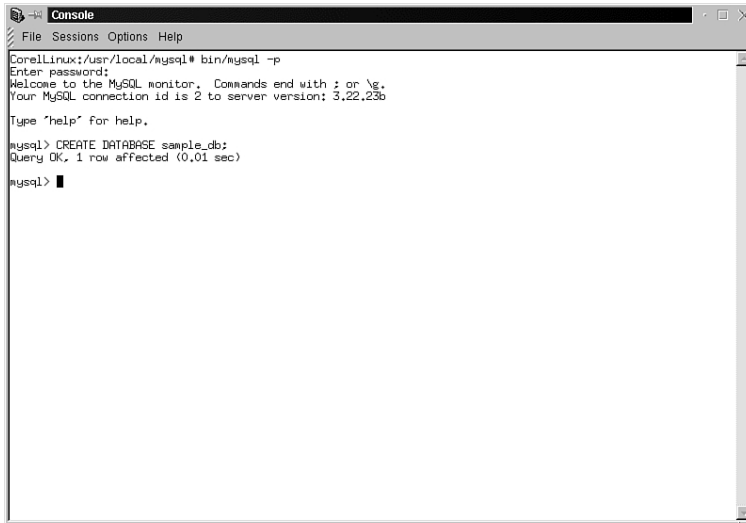
Beachten Sie, dass Sie es genauso eintippen müssen, wie es hier abgebildet ist. Vergessen Sie nicht, dass Sie am Zeilenende ein Semikolon oder `\g` setzen müssen.

Ihre Ergebnisse sollten Abbildung 4.1 gleichen.

Beim Gebrauch von SQL-Befehlen unterscheidet der MySQL-Monitor nicht zwischen Groß- und Kleinschreibung. Deshalb bestehen keine Unterschiede zwischen den folgenden Befehlen; sie bewirken alle das Gleiche:

```
Create Database sample_db;
CrEaTe DaTaBaSe sample_db;
create database sample_db;
```

Diese Befehle erstellen alle die gleiche Datenbank namens `sample_db`. Gemäß der allgemeinen Konvention, die auch dieses Buch befolgt, werden alle SQL-Befehle groß geschrieben. Ein wichtiger Punkt, den Sie beachten sollten, ist allerdings, dass Groß- und Kleinschreibung relevant ist, wenn es um Objekte innerhalb Ihrer Datenbank geht; beispielsweise ist `sample_db` nicht das Gleiche wie `sample_DB`.



```

CoreLinux:/usr/local/mysql# bin/mysql -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2 to server version: 3.22.23b

Type 'help;' for help.

mysql> CREATE DATABASE sample_db;
Query OK, 1 row affected (0,01 sec)

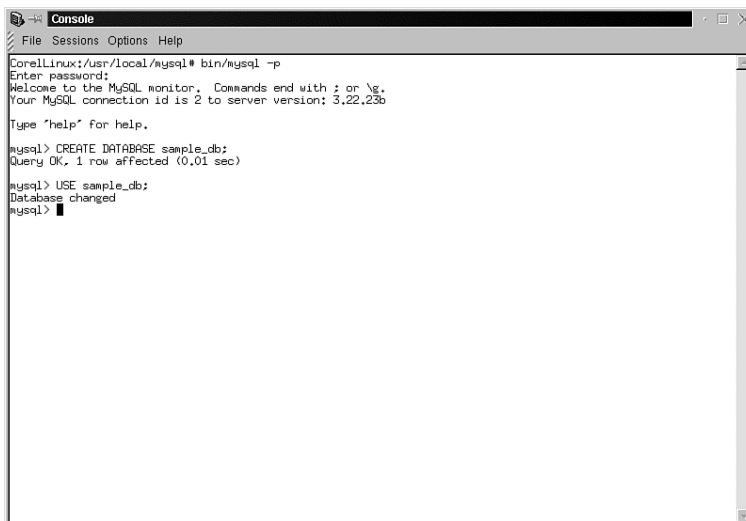
mysql>
  
```

Abbildung 4.1:
Ergebnisse einer
erfolgreichen
Datenbank-
Erstellung

Nachdem Sie Ihre Datenbank erfolgreich erstellt haben, können Sie nun dazu übergehen, sie auch zu benutzen. Wenn Sie sich an das, was Sie an Tag 1, »Was ist MySQL?«, gelernt haben, erinnern, dann sollte Ihnen der `USE`-Befehl einfallen, denn diesen brauchen Sie, um Ihre Datenbank zu benutzen. Um mit der `sample_db` zu arbeiten, geben Sie Folgendes am MySQL-Monitor-Prompt ein:

`USE sample_db;`

Die Ergebnisse Ihres Befehls sollten Abbildung 4.2 gleichen.



```

CoreLinux:/usr/local/mysql# bin/mysql -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2 to server version: 3.22.23b

Type 'help;' for help.

mysql> CREATE DATABASE sample_db;
Query OK, 1 row affected (0,01 sec)

mysql> USE sample_db;
Database changed

mysql>
  
```

Abbildung 4.2:
Die neue Daten-
bank benutzen

Sehr wichtig ist – und das sollten Sie sich auf jeden Fall merken –, dass MySQL die eben erstellte Datenbank nicht automatisch zu einer aktiven Datenbank macht. Sie müssen auf jeden Fall angeben, welche Datenbank Sie mit Ihrer `USE`-Anweisung aktivieren wollen.

Der Befehl DROP

Der `DROP`-Befehl ähnelt dem `CREATE`-Befehl. Während `CREATE` eine Datenbank erstellt, können Sie mit `DROP` eine Datenbank löschen. Seien Sie vorgewarnt – der `DROP`-Befehl ist gnadenlos. Es erscheint keine Bestätigungsbox, die noch einmal nachfragt, ob Sie die Datenbank auch wirklich löschen wollen. `DROP` löscht die Datenbank samt den darin enthaltenen Daten. Darin wird die Stärke dieses Befehls deutlich – einmal ausgeführt, gibt es kein Zurück mehr. Das entspricht allerdings nicht ganz der Wahrheit – Sie können Ihre Daten über eine Log-Datei zurückbekommen. Seien Sie also sehr vorsichtig, wenn Sie den `DROP`-Befehl einsetzen. Um diesen Befehl zu verwenden, führen Sie folgende Schritte aus:

1. Versichern Sie sich, dass der `mysqld`-Daemon läuft und Sie sich im `mysql`-Verzeichnis befinden.
2. Geben Sie auf dem Kommandozeilenebene Folgendes ein:

```
bin/mysql
```

Damit starten Sie den MySQL-Monitor.

3. Geben Sie am Monitor-Prompt Folgendes ein:

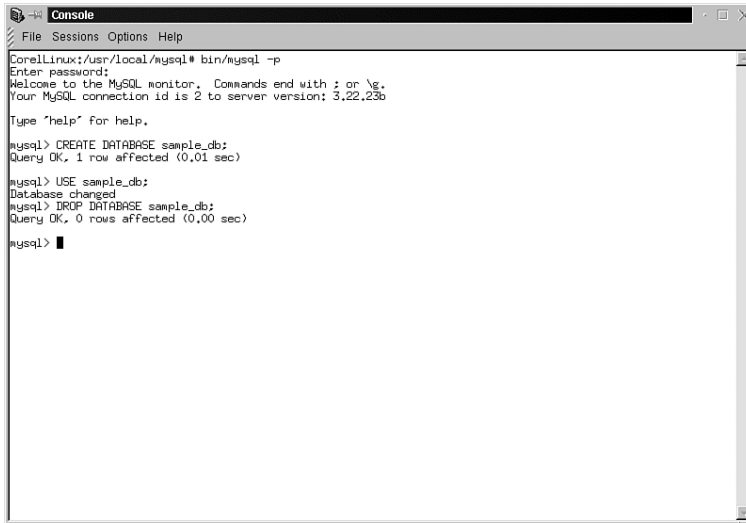
```
DROP DATABASE sample_db;
```

Dies löscht die `sample_db`-Datenbank samt ihren Daten.

Der Output der letzten Schritte sollte Abbildung 4.3 gleichen.

mysqladmin

So wie dies meist der Fall ist in der Welt der Computer, gibt es auch in MySQL mehrere Möglichkeiten, eine Aufgabe zu lösen. MySQL stellt ein äußerst leistungsstarkes Programm zur Verfügung, das Ihnen beim Erstellen und Löschen einer Datenbank behilflich sein kann – `mysqladmin`. Dieses Programm liefert Ihnen zudem viele weitere Funktionen; einigen dieser Funktionen werden Sie in späteren Kapiteln begegnen. Im Augenblick werden Sie dieses Programm nur einsetzen, um eine Datenbank zu erstellen oder zu löschen.



```

CoreLinux:/usr/local/mysql# bin/mysql -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2 to server version: 3.22.23b

Type 'help' for help.

mysql> CREATE DATABASE sample_db;
Query OK, 1 row affected (0,01 sec)

mysql> USE sample_db;
Database changed
mysql> DROP DATABASE sample_db;
Query OK, 0 rows affected (0,00 sec)

mysql>
  
```

Abbildung 4.3:
Das Löschen
einer Datenbank

Eine Datenbank mit Hilfe von `mysqladmin` zu erstellen ist sehr einfach. Um eine Musterdatenbank zu erstellen, müssen Sie folgende Schritte ausführen:

1. Versichern Sie sich, dass der `mysqld`-Daemon läuft und Sie sich im `mysql`-Verzeichnis befinden.
2. Geben Sie folgenden Befehl ein, um Ihre Musterdatenbank zu erstellen:

```
bin/mysqladmin -p CREATE sample_db
```

Ihr Output sollte Abbildung 4.4 gleichen.

Das Löschen einer Datenbank ist genauso einfach. Führen Sie folgende Schritte aus, um Ihre Musterdatenbank zu löschen:

1. Versichern Sie sich wiederum, dass der `mysqld`-Daemon läuft und Sie sich im `mysql`-Verzeichnis befinden.
2. Geben Sie folgenden Befehl ein, um die Datenbank zu löschen:

```
bin/mysqladmin -p DROP sample_db
```

Ihr Output sollte der Abbildung 4.5 gleichen.

Sie haben vielleicht bemerkt, dass beim Gebrauch von `mysqladmin`, bevor Sie eine Datenbank löschen, eine Warnung an Sie ergeht. Dies ist sowohl für einen Neueinsteiger als auch für einen alten Hasen eine große Hilfe. Es ermöglicht Ihnen, Ihren Schritt noch einmal zu überdenken, bevor alle Daten weg sind.



```
CoreLinux:usr/local/mysql# bin/mysqladmin -p CREATE sample_db
Enter password:
Database "sample_db" created.
CoreLinux:usr/local/mysql#
```

Abbildung 4.4:
Erstellen einer
Datenbank mit
mysqladmin



```
CoreLinux:usr/local/mysql# bin/mysqladmin -p DROP sample_db
Enter password:
Dropping the database is potentially a very bad thing to do.
Any data stored in the database will be destroyed.

Do you really want to drop the "sample_db" database [y/N]
y
Database "sample_db" dropped
CoreLinux:usr/local/mysql#
```

Abbildung 4.5:
Löschen einer
Datenbank mit
mysqladmin

Die Befehle `CREATE` und `DROP` des `mysqladmin`-Programms unterscheiden im Gegensatz zum Namen der Datenbank nicht zwischen Groß- und Kleinschreibung. Ein anderer wichtiger Punkt ist, dass Sie berechtigt sein müssen, `CREATE` und `DROP` zu benutzen. Als `root` haben Sie diese Berechtigung; aber wenn Sie kein Verwalter sind, können Sie diese Befehle nicht benutzen.

Benutzer hinzufügen

Da Ihre Datenbank nun läuft, sollten Sie anderen Benutzern die Möglichkeit geben, die Datenbank zu nutzen. Heute werden Sie lernen, wie Sie Benutzer hinzufügen. Berechtigungen und Benutzerprivilegien werden ausführlicher an Tag 17 behandelt.

Um einem Benutzer von Ihrem lokalen Rechner aus – im Folgenden als *localhost* bezeichnet – Zugang zu Ihrer Datenbank zu ermöglichen, muss dieser Benutzer an mehreren Orten eingetragen sein. Die MySQL RDBMS beinhaltet eine Datenbank namens *mysql*. Diese verfügt über sämtliche Berechtigungen für alle MySQL Datenbanken. Die *mysql*-Datenbank besteht aus folgenden Tabellen:

- ▶ **user:** Die Tabelle, die alle Namen, Passwörter, Gäste und Privilegien aller Benutzer dieses MySQL RDBMSs enthält
- ▶ **db:** Die Tabelle, die alle Benutzer, Datenbanken und Gästennamen für dieses MySQL RDBMS beinhaltet
- ▶ **host:** Die Tabelle, die alle Host-Namen, Datenbanken und Privilegien, die sie für dieses MySQL RDBMS brauchen, enthält

Von jedem, der Ihre Datenbank nutzen möchte, muss der Host-Name des Rechners, von dem aus die Verbindung hergestellt wird, in der *host*-Tabelle stehen. Der Benutzer muss sich in die *user*-Tabelle eintragen und die Datenbank steht in der *db*-Tabelle. Führen Sie folgende Schritte aus, um einem anderen Benutzer zu ermöglichen, Ihre Datenbank vom lokalen Rechner aus zu benutzen.

1. Vergewissern Sie sich zuerst, dass der *mysqld*-Daemon läuft und Sie sich im *mysql*-Verzeichnis befinden.
2. Fügen Sie der *host*-Tabelle den Host-Name und die Datenbank hinzu. Dafür müssen Sie den MySQL-Monitor einsetzen.

```
bin/mysql -p
```

3. Um anschließend die *mysql*-Datenbank zu einer aktiven Datenbank zu machen, geben Sie Folgendes ein:

```
USE mysql;
```



Vergessen Sie nicht, dass Befehle im Gegensatz zu Datenbank-Objekten zwischen Groß- und Kleinschreibung unterscheiden.

4. Um die Kombination Host-Name/Datenbank diesem MySQL RDBMS hinzuzufügen, müssen Sie einen **SQL-INSERT-Befehl** verwenden. Geben Sie Folgendes auf Kommandozeilenebene ein:

```
INSERT INTO mysql VALUES(localhost, sample_db,
'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y');
```

Beachten Sie, dass der MySQL-Monitor Ihre Anweisung in der folgenden Zeile weiterführt, wenn Sie kein Semikolon (;) oder \g eingeben. Das ist von Vorteil, da sich die Befehle auf diese Weise gut lesen lassen, und wenn Sie einen Fehler machen, können Sie den »history key« (↑), nur unter Unix verfügbar) einsetzen, um die Anweisung wiederherzustellen und nicht alles noch einmal eintippen zu müssen.

Ihr Output sollte dem in Abbildung 4.6 gleichen.

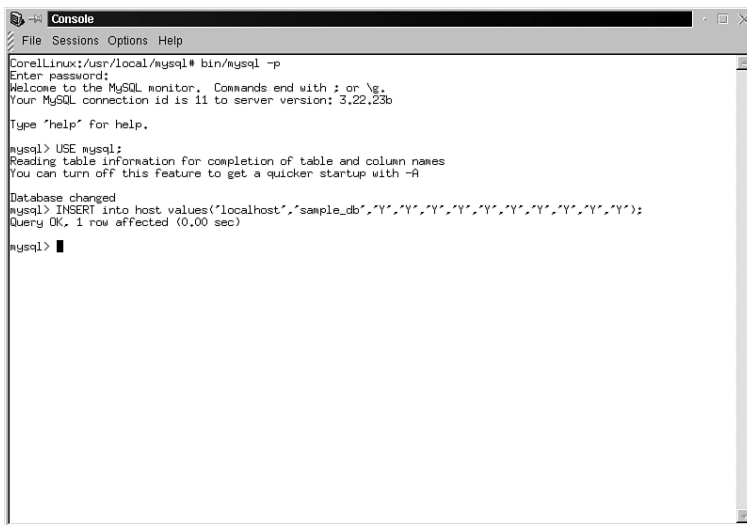


Abbildung 4.6:
Hinzufügen eines
Hosts in die host-
Tabelle

Anschließend sollten Sie sich vergewissern, dass Sie auch Benutzer haben, die Sie Ihrer Datenbank hinzufügen können. Ist dies der Fall, dann fügen Sie jetzt einen Benutzer hinzu, indem Sie folgende Anweisung benutzen:

```
INSERT INTO user VALUES('localhost', 'TestUser',
PASSWORD('pass123'),'Y','Y','Y','Y','Y','Y','Y',
'Y','Y','Y','Y','Y','Y','Y')
```

Die `PASSWORD()`-Funktion ist eine *intrinsische Funktion*, d.h. eine Funktion, die in MySQL selbst als systemeigene Funktion zur Verfügung steht. Sie werden ausführlicher über intrinsische Funktionen an Tag 10, »Interne MySQL-Funktionen«, informiert. Die `PASSWORD`-Funktion beinhaltet als Argument eine Zeichenkette, die sie dann verschlüsselt. Dieses verschlüsselte Wort wird in der Datenbank gespeichert. Dies verhindert, dass neugierige Personen allzu leicht die Passwörter aller Ihrer Benutzer mit einer einfachen Anfrage an die `mysql`-Datenbank entschlüsseln. Am besten gewöhnen Sie sich daran, Benutzer auf diese Art und Weise hinzuzufügen.

Sie sind nun so weit, der `mysql`-Datenbank Ihre Datenbank sowie Benutzer hinzuzufügen. Geben Sie dazu Folgendes ein:

```
INSERT INTO db VALUES('localhost','sample_db',
'TestUser','Y','Y','Y','Y','Y','Y','Y','Y',
'Y')
```

Jetzt ist es Zeit für einen kleinen Rückblick. Um einer Person den Gebrauch der `sample_db`-Datenbank vom lokalen Rechner aus zu ermöglichen, benötigt man so einiges – beispielsweise den Host-Namen des Computers, den der Benutzer verwendet, um eine Verbindung zu Ihrer Datenbank herzustellen. Im Beispiel werden Sie den gleichen Rechner, den das MySQL RDBMS installiert hat, benutzen. Ihr Rechner hat vielleicht einen tollen Namen, doch MySQL erfordert nur den Namen `localhost`, um einen lokalen Rechner zu beschreiben. Wenn Sie von Ihrem Rechner aus mit einer anderen Datenbank in Verbindung treten, muss der Name Ihres Rechners sich in dieser Datenbank befinden. Neben dem Host-Namen brauchen Sie einen Benutzer. Sie können jederzeit Benutzer hinzufügen. Da ich davon ausgehe, dass Sie neu installieren, habe ich das Hinzufügen eines Benutzers bereits abgehandelt. Nachdem ein Benutzer hinzugefügt ist, können Sie einen Schritt weiter gehen und diesem Benutzer die Erlaubnis erteilen, Ihre Datenbank zu nutzen. Dies haben Sie bereits getan, indem Sie den Benutzer der `db`-Tabelle hinzugefügt haben. Die Vorteile dieser Vorgehensweise werden ausführlich an Tag 17 erörtert.

Das Erstellen der Meet_A_Geek-Datenbank

Sie erstellen die `Meet_A_Geek`-Datenbank, indem Sie das `mysqladmin`-Programm benutzen. (Benutzer fügen Sie in einem späteren Kapitel hinzu). Diese Datenbank dient das ganze Buch hindurch als Beispiel und wird von Lektion zu Lektion immer weiter ausgebaut. Um die Datenbank zu erstellen, führen Sie folgende Schritte aus:

1. Vergewissern Sie sich, dass der `mysqld`-Daemon läuft und Sie sich im `mysql`-Verzeichnis befinden.
2. Geben Sie Folgendes ein:

```
bin/mysqladmin -p CREATE Meet_A_Geek
```

Zusammenfassung

Gestern haben Sie die Bedeutung des korrekten Designs kennen gelernt. Heute haben Sie den ersten Schritt getan, um Ihren Entwurf zu realisieren. Sie haben einen wichtigen Punkt erreicht: Sie haben Ihre erste MySQL-Datenbank erstellt. Darüber hinaus

haben Sie gelernt, dass es mehrere Möglichkeiten gibt, diese Aufgabe zu lösen. Wenn Sie auf Befehle stehen, können Sie das `mysqladmin`-Programm benutzen, wenn Sie ein Fan von SQL sind, können Sie eine SQL-Anweisung benutzen, um zum gleichen Ergebnis zu kommen. Mit beiden Methoden befinden sie sich auf dem richtigen Weg, ein MySQL-Datenbank-Administrator zu werden.

Fragen und Antworten

- F** Gibt es eine andere Möglichkeit, Befehle in `mysql` einzugeben? Der MySQL-Monitor kann ein wenig mühsam sein, und ich habe eine Menge Anweisungen einzugeben.
- A** *Es gibt eine andere Möglichkeit: Sie können eine Datei in das `mysql`-Programm »pipen«, d.h. es dem `mysql`-Programm in einem Stück übergeben. Erstellen Sie eine Datei, in der Sie Ihren bevorzugten Editor benutzen (emacs, vi oder gedit). Geben Sie dieser Datei einen beliebigen Namen. Ihre Befehle sollten jetzt so wie auf dem MySQL-Monitor aussehen. Vergessen Sie nicht, die Befehle mit einem Semikolon oder \g zu beenden. Wenn Sie damit fertig sind, können Sie die Datei in MySQL importieren. Vergewissern Sie sich, dass der `mysqld`-Daemon läuft und Sie sich im `mysql`-Verzeichnis befinden. Geben Sie Folgendes ein:*

```
cat /Pfad/Dateiname | bin/mysql
```

Bzw. unter Windows:

```
C:\mysql\bin>mysql < Pfad\Dateiname
```

Damit werden alle Anweisungen in der Datei ausgeführt. Das ist eine gute Methode, Daten hinzuzufügen oder ein Datenbank-Schema zu erstellen. Damit haben Sie auch die Möglichkeit, das Schema für die Datenübertragung und das Wiederherstellen der Datenbank zu speichern, falls die Daten einmal gelöscht werden.

- F** Wie viele Datenbanken kann ein MySQL RDBMS enthalten?
- A** *Da MySQL das Dateisystem des Betriebssystems benutzt, sind der Anzahl der in einem einzigen MySQL RDBMS befindlichen Datenbanken keine Grenzen gesetzt. Die Größe der Datenbank ist allerdings durch das Betriebssystem begrenzt. Die Tabellen der Datenbank können nur so groß sein, wie es das Dateisystem des Betriebssystems zulässt. Bei Linux zum Beispiel beträgt die maximale Größe 4 Gbyte.*



Übungen

1. Erstellen und löschen Sie Datenbanken, indem Sie das `mysqladmin`-Programm sowie den Monitor benutzen.
2. Fügen Sie der Datenbank einige Benutzer hinzu und versuchen Sie, diese Benutzerkonten zu benutzen.



Daten- normalisierung

**Woche
1**

Wenn Sie eine Datenbank strukturieren, kann das Einfügen von Spalten in die passenden Tabellen sehr entmutigend sein. Beim Strukturieren Ihrer Datenbank können Sie auf logische Fehler stoßen – besonders dann, wenn Sie in der Zeit der »nichtrelationalen« Datenbanken aufgewachsen sind, als noch alles in ein- und derselben Datei gespeichert wurde. Die Idee, Ihre sämtlichen Daten in einer Tabelle einer relationalen Datenbank unterzubringen, ist nicht nur veraltet, sondern beinahe ein Sakrileg. Um Datenbank-Designern zu helfen, wurde ein neues Regelwerk festgelegt. Wenn Sie sich an diese Richtlinien halten, können Sie eine Datenbank ohne logische Fehler erstellen. Das Anwenden dieser Regeln auf Ihre Datenbankstruktur wird als *Normalisierung* bezeichnet.

Heute werden Sie Folgendes lernen:

- ▶ was Normalisierung bedeutet und welche Vorteile sich daraus ergeben
- ▶ die verschiedenen Grade der Normalisierung

Was ist Normalisierung?

Normalisierung ist ein Regelwerk, das dem Datenbank-Designer hilft, ein Schema zu entwickeln, das logische Probleme minimiert. Jede Regel baut auf der vorherigen auf. Die Normalisierung wurde eingeführt, weil die alte Methode, alle Daten in einer Datei oder Datenbank-Tabelle zu speichern, große Mängel aufweist und häufig zu logischen Fehlern führte, wenn man die gespeicherten Daten bearbeiten wollte. Werfen Sie zum Beispiel einen Blick auf die *Meet_A_Geek*-Datenbank. Möchten Sie alle Daten in der *Customers*-Tabelle speichern, sieht die Tabelle ungefähr folgendermaßen aus:

```
Customers
Customer_ID
Last_Name
First_Name
Address
Product_Name1
Product_Cost1
Product_Picture1
Product_Name2
Product_Cost2
```

Product_Picture2

Order_Date

Order_Quantity

Shipper_Name

Dies ist nur ein Auszug aus der Tabelle, aber die Idee, die dahinter steht, wird trotzdem deutlich. Wie fügen Sie nun einen neuen Kunden in Ihre *Customers*-Tabelle ein? Sie müssen ebenfalls das Produkt und die Bestellung eingeben. Wie können Sie eine Liste all Ihrer zum Verkauf stehender Produkte abrufen? Sie können die Produkte nicht einfach von den Kunden trennen. Das Gute an einer relationalen, korrekt entworfenen Datenbank ist, dass sie genau diese Aufgaben meistert.

Normalisierung vereinfacht viele Dinge. Menschen neigen dazu, immer alles auf den kleinsten gemeinsamen Nenner zu bringen. Alles wird vereinfacht. Wenn wir beispielsweise ein großes Bild betrachten, so versuchen wir zu vereinfachen, indem wir ähnliche Dinge zusammenfügen. Die Richtlinien, die die Normalisierung liefert, schaffen den Rahmen dafür, die Struktur aufzugliedern. In Ihrer Musterdatenbank können Sie ganz einfach drei verschiedene Gruppen erkennen: Kunden, Produkte und Bestellungen. Den Richtlinien der Normalisierung folgend, würden Sie Ihre Tabellen auf der Grundlage dieser Gruppen erstellen.

Der Normalisierungsprozess besitzt einen Namen und ein Regelwerk für jede Phase der Aufgliederung/Einteilung. Das mag Ihnen am Anfang vielleicht etwas verwirrend vorkommen, aber ich hoffe, Sie werden den Prozess und auch die Gründe dafür, auf diese Weise zu verfahren, verstehen. Die meisten Leute geben sich mit einem Spreadsheet, das all Ihre Daten enthält, zufrieden. Die Zeit, die Sie darauf verwenden, Ihr Schema aufzugliedern, indem Sie den Normalisierungsprozess durchgehen, ist gut investiert. Das Durchlaufen dieses Prozesses geht schneller, wenn Sie die mit Daten angefüllten Spalten »auseinanderschneiden« und anschließend wieder »zusammenkleben«, so dass sie den gestellten Anforderungen genügen.

Ein weiterer Vorteil der Normalisierung ist, dass Sie dadurch Speicherkapazität einsparen. Eine normalisierte Datenbank nimmt insgesamt weniger Platz in Anspruch als eine Datenbank, die nicht normalisiert wurde. Durch Normalisierung wird die Wiederholung von Daten reduziert, so dass der Platzverbrauch beim Speichern Ihrer Daten sehr viel geringer ist.

Grade der Normalisierung

Der Normalisierungsvorgang beinhaltet prinzipiell drei Level: Erste Normalform (1NF), Zweite Normalform (2NF) und Dritte Normalform (3NF). Jede Form besitzt ein eigenes Regelwerk. Eine Datenbank kann einem der drei Level angepasst werden.

Nehmen wir an, Ihre Datenbank richtet sich nach den Regeln des zweiten Grades der Normalisierung, dann gilt sie als in der »2. Normalform« normalisiert. Es ist nicht immer am besten, den höchsten Grad der Normalisierung zu wählen. Dies kann zu unnötigen Schwierigkeiten führen, die bei einem niedrigeren Level vermieden werden könnten.



Insgesamt gibt es neun Grade der Normalisierung. Es gibt 1. Normalform, 2. Normalform, 3. Normalform, Boyce-Codd Normalform, 4. Normalform (4.NF), 5. Normalform (5.NF) oder Join-Projection Normalform, Strong Join Projection Normalform, Over-Strong Join Projection Normalform und Domain Key Normalform. Hier werden nur die ersten drei Formen behandelt.

Die erste Normalform

Die Regel der Ersten Normalform besagt, dass alle sich wiederholenden Spalten ausgesondert und in separate Spalten verfrachtet werden sollen. Dies ist eine relativ einfach zu befolgende Regel. Werfen Sie einen Blick auf das Schema der Customers-Datenbank in Tabelle 5.1.

Customers
Customer_ID
Last_Name
First_Name
Address
Product_Name1
Product_Cost1
Product_Picture1
Product_Name2
Product_Cost2
Product_Picture2
Order_Number
Order_Date
Order_Quantity
Shipper_Name

Tabelle 5.1: Schema für die Customers-Datenbank

In Tabelle 5.1 finden Sie mehrere sich wiederholende Spalten, die sich hauptsächlich mit Produkten befassen. Laut Regel müssen Sie die Wiederholungen aussondern und in eine gesonderte Tabelle verfrachten. Das ist ganz einfach. Die daraus resultierenden Datenbank-Tabellen sind in Tabelle 5.2 angeführt.

Customers	Products
Customer_ID	Product_Name1
Last_Name	Product_Cost1
First_Name	Product_Picture1
Address	
Order_Number	
Order_Date	
Order_Quantity	
Shipper_Name	

Tabelle 5.2: Eliminierung von Datenwiederholungen in einer Datenbank

Jetzt haben Sie zwar zwei Tabellen, aber immer noch ein Problem. Sie können im Augenblick noch nicht die Daten aus der ursprünglichen Tabelle zu den Daten aus der neuen Tabelle in Beziehung setzen. Um eine Beziehung herzustellen, muss der zweiten Tabelle ein Schlüssel hinzugefügt werden. Fügen Sie der `Products`-Tabelle einen primären Schlüssel namens `Product_ID` hinzu und der `Customers`-Tabelle einen Schlüssel, der die `Products`-Tabelle zur `Customers`-Tabelle in Beziehung setzt. Das `Product_ID`-Feld ist die ideale Wahl. Die daraus resultierenden Tabellen gleichen Tabelle 5.3.

Customers	Products
Customer_ID	Product_ID
Last_Name	Product_Name1
First_Name	Product_Cost1
Address	Product_Picture1
Order_Number	
Order_Date	
Order_Quantity	
Shipper_Name	

Tabelle 5.3: Die erste Normalform

Jetzt wurde eine 1:n-Beziehung festgelegt. Daran wird deutlich, was die Datenbank im Alltag zu bewerkstelligen hat. Der Anbieter kann beliebig viele Produkte verkaufen, unabhängig davon, wie viele Kunden es gibt, die diese kaufen. Ein Kunde muss trotzdem erst ein Produkt bestellt haben, bevor er ein Kunde wird. Sie müssen jedoch nicht mehr jedes Mal, wenn Sie Ihrem Inventar ein neues Produkt hinzufügen, gleichzeitig auch einen neuen Kunden hinzufügen.

Indem eine Datei in die 1. Normalform gebracht wird, löst sich das Problem der vielfachen Spaltennamen. Zu oft fabrizieren unerfahrene Datenbank-Designer etwas, das der nicht normalisierten Tabelle aus dem ersten Beispiel gleicht. Häufig werden viele Spalten erstellt, die immer wieder die gleichen Daten anzeigen. In einem Elektrizitätswerk im Nordwesten der USA gab es beispielsweise einmal eine Datenbank, die Teile eines Atomkraftwerks beinhaltete. Die Tabelle, in der die Nummer der Teile enthalten war, beinhaltete eine Spalte, die sich über 30-mal wiederholte. Jedes Mal wenn eine neue Information gespeichert wurde, wurde gleichzeitig eine neue Spalte geschaffen, die diese Daten speicherte. Dies war, wie Sie sich leicht vorstellen können, eine miserabel konstruierte Datenbank und ein Alptraum für jeden Programmierer.

Normalisierung ermöglicht, eine Datenbank übersichtlicher zu machen und sie in kleinere, verständlichere Einheiten zu gliedern. Statt eine riesige, monolithische Tabelle, die viele unterschiedliche Aspekte enthält, begreifen zu müssen, müssen Sie nur kleinere, verständlichere Objekte und die einfachen Beziehungen, die diese mit all den anderen Objekten unterhalten, verstehen. Es ist wohl überflüssig zu erwähnen, dass eine bessere Kenntnis darüber, wie eine Datenbank funktioniert, zu einer besseren Verwendung Ihrer Vorteile führt.

Die zweite Normalform

Die Regel der 2. Normalform besagt, dass alle Teilabhängigkeiten ausgesondert und in Ihre eigenen Tabellen verfrachtet werden müssen. *Teilabhängigkeit* ist ein Begriff, um Daten zu beschreiben, die sich nicht auf den Tabellenschlüssel stützen, um sie eindeutig zu identifizieren. In der Musterdatenbank findet sich die Bestellinformation in jedem Datensatz. Es wäre einfacher, nur die Bestellnummer zu benutzen. Die restlichen Informationen könnten in Ihrer eigenen Tabelle bleiben. Nach dem Aussondern der Bestellinformationen gleicht Ihr Schema Tabelle 5.4.

Customers	Products	Orders
Customer_ID	Product_ID	Order_Number
Product_ID	Order_Date	Product_Name
Order_Number	Product_Cost	Order_Quantity

Tabelle 5.4: Teilabhängigkeiten beseitigen

Customers	Products	Orders
Last_Name	Product_Pictures	
First_Name		
Shipper_Name		

Tabelle 5.4: Teilabhängigkeiten beseitigen (Forts.)

Wenn Sie Ihr Schema auf diese Weise aufbauen, finden Sie darin wiederum ein Abbild der Alltagswelt. Sie müssten einige Änderungen an Ihren Geschäftsregeln vornehmen, um diese wieder anwendbar zu machen, aber um die Normalisierung zu illustrieren, ist es erst mal okay.

Ihnen sollte jetzt einiges klar sein. Die einst schwer zu entschlüsselnde Tabelle ergibt nun mehr Sinn, und die Beziehungen unter den zu speichernden Daten treten deutlicher hervor und sind leichter zu verstehen. Alles erscheint logischer. Damit haben Sie einige Vorteile der Normalisierung einer Datenbank kennen gelernt.

Einer der Hauptnachteile der Normalisierung ist der Zeitaufwand. Die meisten Menschen denken, Sie hätten nicht genug Zeit, um auch noch zu überprüfen, ob Ihre Daten normalisiert sind. Normalisierung wird häufig als Zeitverschwendung betrachtet – schließlich läuft ja auch alles, ohne dass man die Daten normalisiert. Das ist jedoch eine falsche Einschätzung. Sie werden letztlich viel mehr Zeit brauchen, um eine nicht normalisierte Datenbank zu reparieren, als dies bei einer normalisierten, gut entworfenen Datenbank der Fall wäre.

Passen Sie Ihre Datenbank der 2. Normalform an, so kommen Sie in den Genuss einiger Vorteile einer relationalen Datenbank. Sie können beispielsweise der `Customers`-Tabelle neue Spalten hinzufügen, ohne die Tabellen `Products` und `Orders` zu beeinflussen. Das Gleiche gilt auch für die anderen Tabellen.

Zudem sind Sie, wenn Sie den zweiten Level erreicht haben, die meisten logischen Probleme los. Sie können in die meisten Tabellen einen Datensatz ohne überflüssige Daten einfügen. Bei näherem Hinsehen finden Sie in der `Customers`-Tabelle die `Shipper_Name`-Spalte. Diese Spalte ist unabhängig vom Kunden. Der nächste Level der Normalisierung zeigt Ihnen, wie Sie das korrigieren können.

Die dritte Normalform

Die Regel der dritten Normalform besagt, dass alle Daten, die kein Schlüssel sind, auszusondern und zu trennen sind. Eine dem dritten Level angepasste Spalte muss wegen ihrer Werte vom Schlüssel abhängig sein. Alle Werte müssen eindeutig durch den Schlüssel identifiziert werden. In der Musterdatei enthält die `Customers`-Tabelle die

Shipper_Name-Spalte, die vom Schlüssel nicht eindeutig identifiziert wird. Sie können diese Daten von der aktuellen Tabelle abtrennen und sie in Ihre eigene Tabelle einfügen. Tabelle 5.5 zeigt das daraus resultierende Datenbank-Schema:

Customers	Products	OrderMaster	OrderDetail	Shippers
Customer_ID	Product_ID	Order_Number	Order_Detail_ID	Shipper_ID
Product_ID	Product_Name	Order_Date	Order_Number	Shipper_Name
Order_Number	Product_Cost	Order_Quantity	Order_Date	
Shipper_ID	Product_Pictures		Order_Quantity	
Last_Name				
First_Name				
Address				

Tabelle 5.5: Nicht-Schlüssel-Daten in der 3. Normalform aussondern

Jetzt sind alle Ihre Tabellen dem dritten Level angepasst. Die 3. Normalform bietet die größte Flexibilität und beugt logischen Fehlern beim Einfügen und Löschen der Datensätze vor. Jede Spalte wird durch den Schlüssel eindeutig identifiziert, so dass keine Daten wiederholt werden. Damit haben Sie die Möglichkeit, ein klares, elegantes, sauberes Schema zu erstellen, mit dem Sie gut arbeiten und das Sie leicht ausweiten können.

Welchen Grad der Normalisierung soll man wählen?

Nachdem Sie die drei verschiedenen Level der Normalisierung kennen gelernt haben, sollten Sie nun entscheiden, wie weit Sie Ihre Daten normalisieren wollen. Normalisierung ist eine subjektive Wissenschaft. Es liegt an Ihnen, zu bestimmen, was aufgegliedert werden muss. Falls Ihre Datenbank Daten für nur einen Benutzer liefert und eine Erweiterung nicht in Betracht kommt, dann wäre es sicherlich übertrieben, die 3. Normalform zu wählen. Die Regeln der Normalisierung sind Richtlinien, die Ihnen helfen sollen, leicht bedienbare, flexible und effiziente Tabellen zu erstellen.

Manchmal ist es nicht sinnvoll, Ihre Daten auf dem höchsten Level zu normalisieren. Nehmen wir an, Sie haben Ihrer Datenbank eine weitere Adressen-Spalte hinzugefügt – es ist nichts Ungewöhnliches, zwei Zeilen für eine Adresse zu haben. Das Tabellenschema könnte folgendermaßen aussehen:

Customer_ID
Last_Name
First_Name
Address1
Address2

Gemäß den Regeln würde diese Tabelle der ersten Normalform entsprechen. Die Adressen-Spalten würden herausgenommen und durch den Schlüssel für die neue Tabelle ersetzt werden. Das Resultat würde folgendem Schema entsprechen:

Customer_ID	Address_ID
Last_Name	Customer_ID
First_Name	Address

Die Datenbank entspricht jetzt der 1. Normalform. Ihre Kunden können nun mehr als eine Adresse haben. Problematisch ist allerdings, dass Sie, indem Sie die Regeln der Normalisierung angewandt haben, eine einfache Sache letztlich komplizierter als nötig gemacht haben. In unserem Beispiel ist die zweite Adresse fakultativ. Sie kann, muss aber nicht, angegeben werden. Diese Information soll lediglich den Kontakt zu den Kunden erleichtern. Es ist in diesem Fall wirklich nicht notwendig, diese Informationen in eine eigene Tabelle abzusondern und zu normalisieren. Das wesentliche Ziel der Normalisierung ist es, wie bereits gesagt, Dinge zu vereinfachen. In diesem Beispiel aber wird das Gegenteil erreicht; eine relativ einfache Sache wird verkompliziert. Eine gute Methode zu überprüfen, ob Ihre Normalisierung zu weit geht, ist, immer die Anzahl Ihrer Tabellen im Auge zu behalten. Haben Sie eine große Anzahl an Tabellen, kann das unter Umständen bedeuten, dass Sie zu viel normalisieren. Gehen Sie einen Schritt zurück und betrachten Sie Ihr Schema. Gliedern Sie Dinge nur um der Regeln willen auf oder weil es die Arbeit tatsächlich einfacher und übersichtlicher macht? Dies sind alles Entscheidungen, die Sie, der Datenbank-Designer, treffen müssen. Sie sollten sich dabei von Ihrer Erfahrung und Ihrem gesundem Menschenverstand leiten lassen. Normalisierung ist nun mal keine exakte, sondern eine subjektive Wissenschaft.

Es gibt sechs weitere Level der Normalisierung, auf die bisher nicht eingegangen wurde. Es handelt sich um folgende Formen: Boyce-Codd-Normalform, 4. Normalform (4NF), 5. Normalform (5NF), Strong-Join-Projection-Normalform, Over-Strong-Join-Projection-Normalform und Domain-Key-Normalform. Diese Formen der Normalisierung sind nicht in allen Fällen angebracht. Ihre Aufgabe besteht darin, eine Datenbank wirklich relational zu machen. Diese sechs Level befassen sich hauptsächlich mit multiplen Abhängigkeiten und relationalen Schlüsseln. Wenn Sie sich damit auskennen, dann brauchen Sie dieses Buch wahrscheinlich nicht.

Zusammenfassung

Normalisierung ist eine Technik, die eingesetzt wird, um klare, logische Beziehungen zwischen Tabellen zu erstellen. Bei der Bearbeitung von Daten können mit Hilfe dieser Technik logische Fehler vermieden werden. Die Normalisierung vereinfacht zudem das Hinzufügen von Spalten, da das aktuelle Schema und die aktuellen Beziehungen nicht getrennt werden.

Es gibt verschiedene Formen der Normalisierung: 1. Normalform (1NF), 2. Normalform (2NF), 3. Normalform (3NF), Boyce-Codd-Normalform, 4. Normalform (4NF), 5. Normalform (5NF), Strong-Join-Projection-Normalform, Over-Strong-Join-Projection-Normalform und Domain-Key-Normalform. Jedes neue Level bedeutet einen Schritt weiter in Richtung wirkliche relationale Datenbank. Die ersten drei Formen wurden ausführlich erläutert. Sie liefern ausreichend Möglichkeiten, um den Ansprüchen der meisten modernen Datenbanken zu genügen.

Falls Sie einen zu hohen Grad der Normalisierung wählen, kann dies zu einer ineffizienten Datenbank führen und Ihr Schema unnötig verkomplizieren, so dass Ihre Arbeit nicht vereinfacht, sondern im Gegenteil erschwert wird. Gesunder Menschenverstand und Ihre Erfahrung können Ihnen bei der Entscheidung, wann eine Normalisierung sinnvoll ist und welcher Grad der Normalisierung der richtige ist, helfen.

Fragen und Antworten

- F** Wann ist der beste Zeitpunkt, um meine Daten zu normalisieren?
- A** *Meistens wird normalisiert, nachdem alles ins Rollen gekommen ist und Ihre Datenbank startbereit ist. Dies ist aber nicht der beste Zeitpunkt. Sie sollten Ihre Daten am besten direkt nach dem Entwerfen und Planen Ihrer Tabelle normalisieren, – dies ist nämlich der Augenblick, in dem Sie mögliche Probleme am ehesten erkennen und insofern am besten entscheiden können, ob und wie normalisiert werden soll.*
- F** Sollte ich meine Datenbank in die 3. Normalform bringen?
- A** *Nur wenn es sinnvoll ist. Wie an dem in dieser Lektion vorgestellten Beispiel deutlich geworden ist, ist es manchmal besser, nicht zu normalisieren. Normalisierung gliedert die Dinge in ihre kleinste Form – denn klein ist ein Synonym für schnell. Je schneller eine Datenbank arbeitet, desto besser sind Sie dran.*

Übungen

1. Erläutern Sie die Vorteile der Normalisierung.
2. Nennen Sie Bereiche, die man nicht normalisieren sollte.

A light gray grid representing a spreadsheet is visible in the background, tilted slightly. The top row contains the letters 'M', 'D', 'M', 'D', 'I' in white boxes. Below this are several empty white boxes.

Das Hinzufügen von Tabellen, Spalten und Indizes

**Woche
1**

Das Hinzufügen von Spalten, Tabellen und Indizes stellt einen grundlegenden Schritt beim Erstellen einer Datenbank dar. Eine Datenbank wird durch Tabellen und deren Spalten definiert. Heute werden Sie Folgendes lernen:

- ▶ wie erstelle ich Tabellen und Spalten
- ▶ wie editiere ich bestehende Spalten
- ▶ was ist ein Index, und wie wird er gebraucht

Das Erstellen von Tabellen

Das Erstellen von Tabellen in MySQL ist eine relativ einfache Aufgabe. Es gibt mehrere Möglichkeiten, diese Aufgabe zu lösen. In diesem Kapitel werden zwei Möglichkeiten, die Ihnen bereits im dritten Kapitel begegnet sind, vorgestellt. Zuerst benutzen Sie den MySQL-Monitor. Der Monitor ist bei der Interaktion mit Ihrer Datenbank das wichtigste Werkzeug. Um Ihre erste Tabelle zu erstellen, führen Sie folgende Schritte aus:

1. Vergewissern Sie sich, dass der `mysqld`-Daemon läuft (indem Sie `mysqladmin ping` eingeben) und dass Sie sich im `mysql`-Verzeichnis befinden (angenommen Sie haben `mysql` im Standardverzeichnis installiert; der Befehl `pwd` sollte `/usr/local/mysql` zurückgeben).
2. Starten Sie den MySQL-Monitor, indem Sie Folgendes eintippen:

```
bin/mysql -u root -p MEET_A_Geek
```

Sie sollten nun aufgefordert werden, ein Passwort einzugeben. Nachdem Sie Ihr Passwort eingegeben haben, befinden Sie sich im MySQL-Monitor mit der `MEET_A_Geek`-Datenbank als aktiver Datenbank (siehe Abbildung 6.1).

Anhand des Entwurfs, der an Tag 3, »Das Entwerfen Ihrer ersten Datenbank«, entwickelt wurde, werden Sie die `Customers`-Tabelle erstellen. Dazu geben Sie folgende Befehle in genau dieser Reihenfolge ein. Achten Sie darauf, dass Sie Ihren Befehl durch Semikolon (;) oder `\g` abschließen, sonst wird er beim Betätigen der -Taste nicht ausgeführt.

```
CREATE TABLE Customers (
Customer_ID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
First_Name VARCHAR(20) NOT NULL,
Last_Name VARCHAR(30) NOT NULL,
Address VARCHAR(50),
City VARCHAR(20),
State VARCHAR(2),
Zip VARCHAR(20),
```



```
E_Mail VARCHAR(20),
Age INT, Race VARCHAR(20),
Gender ENUM('M', 'F') DEFAULT 'F',
Eye_Color VARCHAR(10),
Hair_Color VARCHAR(10),
Favorite_Activity ENUM('Programming', 'Eating', 'Biking', 'Running',
None') DEFAULT 'None',
Favorite_Movie VARCHAR(50),
Occupation VARCHAR(30),
Smoker CHAR(0));
```

Ihr Output sollte Abbildung 6.2 entsprechen.

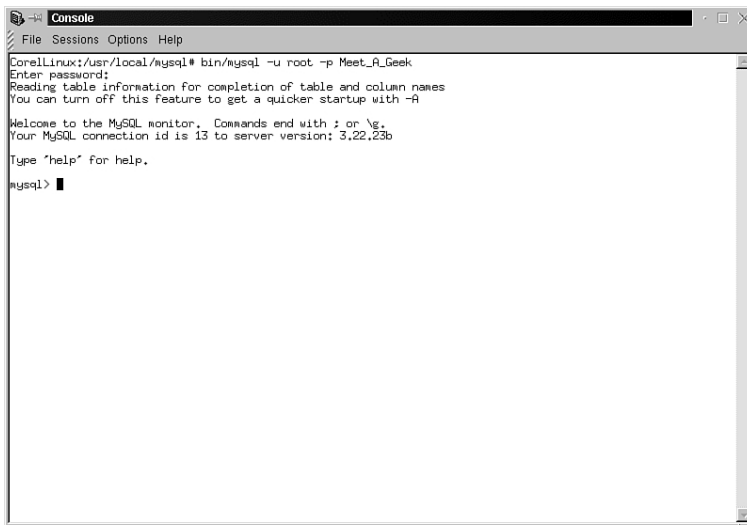


Abbildung 6.1:
MySQL-Monitor
mit der aktiven
Meet_A_Geek-
Datenbank

1. Um den Erfolg Ihrer Aktionen zu prüfen, geben Sie Folgendes ein:

```
SHOW TABLES FROM Meet_A_Geek;
```

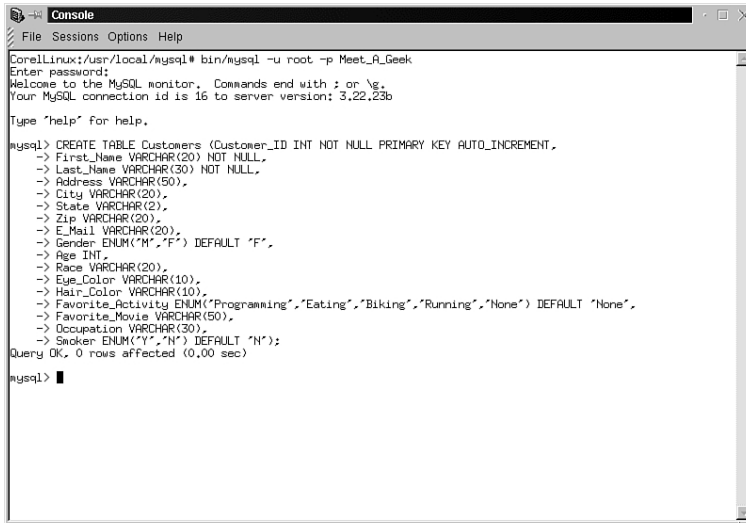
Sie sollten eine Liste von Tabellen, die in der Meet_A_Geek-Datenbank verfügbar sind, erhalten. Wenn Sie sich an die Vorgaben des Buches gehalten haben, sollte es jetzt nur die Customers-Tabelle geben.

2. Um die Beschreibung der Tabelle zu sehen, können Sie einen der beiden folgenden Befehle eingeben:

```
SHOW COLUMNS FROM Customers;
```

oder

```
DESCRIBE Customers;
```



```

CoreLinux:/usr/local/mysql# bin/mysql -u root -p Meet_A_Geek
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 16 to server version: 3.22.23b

Type 'help' for help.

mysql> CREATE TABLE Customers (Customer_ID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    -> First_Name VARCHAR(20) NOT NULL,
    -> Last_Name VARCHAR(30) NOT NULL,
    -> Address VARCHAR(50),
    -> City VARCHAR(20),
    -> State VARCHAR(2),
    -> Zip VARCHAR(20),
    -> E_Mail VARCHAR(20),
    -> Gender ENUM('M','F') DEFAULT 'F',
    -> Age INT,
    -> Race VARCHAR(20),
    -> Eye_Color VARCHAR(10),
    -> Hair_Color VARCHAR(10),
    -> Favorite_Activity ENUM('Programming','Eating','Biking','Running','None') DEFAULT 'None',
    -> Favorite_Movie VARCHAR(50),
    -> Occupation VARCHAR(30),
    -> Smoker ENUM('Y','N') DEFAULT 'N');
Query OK, 0 rows affected (0.00 sec)

mysql>
  
```

Abbildung 6.2:
Das Erstellen
einer neuen
Tabelle für die
MEET_A_Geek-
Datenbank



Ich bevorzuge den zweiten Befehl, da man weniger einzutippen hat. Beide Befehle geben die gleiche Information zurück.

Nachdem Sie Ihre Daten geprüft haben, können Sie weitere Tabellen hinzufügen.

MySQL ermöglicht Ihnen zudem, temporäre Tabellen zu erstellen. Vorübergehende Tabellen bestehen lediglich für den Zeitraum der aktuellen Session. Sie verschwinden wieder, wenn die Verbindung beendet wird. Temporäre Tabellen können nur vom Erstellenden eingesehen werden – d.h., wenn ich MySQL lokal starte und eine temporäre Tabelle erstelle, kann kein anderer mit seinem Rechner und einer anderen Verbindung auf diese Tabelle zugreifen. Diese Art von Tabellen ist vor allem ein sehr nützliches Werkzeug, wenn Sie Ihre Daten nur vorübergehend speichern wollen oder wenn Sie Ergebnisse einer Abfrage speichern, um sie mit den Ergebnissen einer anderen zu vergleichen. Um eine temporäre Tabelle zu erstellen, erlassen Sie folgenden Befehl:

```
CREATE TEMPORARY TABLE Tabellenname (Spaltenname Datentyp);
```

Wie Sie sehen, unterscheidet sich das Erstellen einer temporären Tabelle kaum von dem Erstellen einer permanenten Tabelle. Der einzige Unterschied besteht darin, dass Sie für das Erstellen einer temporären Tabelle den Begriff `TEMPORARY` einfügen müssen.

Eine weitere nützliche Funktion, die kürzlich in MySQL eingeführt wurde, ist die Möglichkeit, eine auf den Ergebnissen einer Abfrage basierende Tabelle zu erstellen. Das ist ein sehr nützliches Feature. Es ermöglicht Ihnen, eine Tabelle zu erstellen, ohne alle Spaltendaten eintippen zu müssen. Zudem können Sie auf sehr einfache Weise

eine Kopie einer bestehenden permanenten Tabelle anfertigen. Möchten Sie eine temporäre Kopie der `Customers`-Tabelle erstellen, geben Sie folgende Anweisung ein:

```
CREATE TEMPORARY TABLE SELECT *FROM Customers;
```

Sie können auch den Begriff `TEMPORARY` weglassen und hinter `TABLE` den neuen Tabellennamen einfügen. Die Syntax hierfür sieht folgendermaßen aus:

```
CREATE TABLE Tabellenname SELECT *FROM Customers;
```

Ein weiteres wichtiges Feature ist der `IF NOT EXISTS`-Parameter. Mit dieser Anweisung können Sie, bevor Sie eine Tabelle erstellen, prüfen, ob diese bereits existiert. Das ist sehr nützlich, wenn Sie eine Tabelle erstellen müssen und nicht wissen, ob diese schon existiert. Um dies auszuführen, geben Sie folgende Anweisung ein:

```
CREATE TABLE IF NOT EXISTS Tabellenname (Spaltenname Daten Typ);
```

Vergessen Sie nicht, dass auf Grund der Bedingungsanweisung die Tabelle nur erstellt wird, falls sie nicht schon existiert. Ansonsten geschieht gar nichts.



Namenskonventionen sind ein notwendiges Übel. Sie können das gesamte Projekt unterstützen und Neueinsteigern den Einstieg erleichtern.

Über die Namenskonvention muss noch etwas gesagt werden. Namenskonventionen sind eine gute Sache. Sie ermöglichen Ihnen, auf eine standardisierte Art und Weise Objekte, die Sie und andere nutzen, zu benennen. Die konkrete und meist anschauliche Begrifflichkeit hilft Neueinsteigern, sich schnell und einfach mit einem Datenbank-Schema vertraut zu machen. Wenn Sie zum Beispiel die Tabelle, die all Ihre Kunden beinhaltet, `Customers` und die Tabelle, die all Ihre Produkte enthält, `Products` nennen, ist das vor allem für Neueinsteiger sehr viel anschaulicher und leichter zu erlernen, als wenn Sie die Tabellen `01` und `02` genannt hätten. Wie Sie Ihre Tabellen benennen, liegt ganz bei Ihnen, dem Datenbank-Designer. Ich bevorzuge die folgenden Konventionen:

- ▶ Tabellen stehen im Plural, Feldnamen im Singular. Demnach heißt eine Tabelle, die beispielsweise alle Kundennamen enthält, `Customers` und nicht `Customer`. Meiner Ansicht nach ist es sinnvoll, Tabellen in den Plural zu stellen, weil in einer Tabelle sämtliche Objekte verzeichnet sind und nicht nur eins.
- ▶ Der erste Buchstabe eines Namens wird immer groß geschrieben. Das entspricht zum einen der Orthographie und zum anderen sieht es meiner Meinung nach auch besser aus.
- ▶ Zusammengesetzte Namen werden durch Unterstrich getrennt und der Anfangsbuchstabe jedes Namens wird groß geschrieben (zum Beispiel `Meet_A_Geek`). Das erschwert zwar vielleicht das Eintippen, erleichtert aber das Lesen. Zudem sind Bindestriche und Leerräume bei keinem Datenbank-Objekt erlaubt.

- Benutzen Sie anschauliche, beschreibende Namen, und seien Sie konsequent in Ihrer Benennungspraxis. Wenn Sie einen ganzen Berg an Tabellen erstellen, ist es gut zu wissen, dass zum Beispiel `Last_Name` immer `Last_Name` bedeutet – egal um welche Tabelle es sich handelt. Dies ist besonders hilfreich, wenn Sie Programme und Anweisungspläne entwickeln, die immer wieder auf viele Tabellen zugreifen.

Ich werde diese Konventionen das ganze Buch hindurch verwenden. Sie können natürlich benutzen, was Ihnen gefällt. Diese Regeln wurden an vielen Datenbanken ausprobiert und haben immer wieder bewiesen, dass Namenskonventionen sehr sinnvoll sind – auch wenn sie etwas zusätzliche Arbeit bedeuten.

Eine Möglichkeit, das Schema einer Datenbank zu erstellen, besteht darin, die Befehle in den MySQL-Monitor einzugeben. Zum anderen kann man auch ein Script benutzen. Ein Script ist eine Textdatei, die alle SQL-Befehle, die für die Konstruktion einer Datenbank erforderlich sind, enthält. Dies ist wahrscheinlich die beste Art, Ihre Datenbank zu erstellen, da Sie bei dieser Methode die Möglichkeit haben, Ihre Datenbank jederzeit wieder zu erstellen (allerdings ohne Daten). Zudem können Sie den gleichen Code wieder benutzen – im Allgemeinen sind Computer-Freaks ein ziemlich fauler Haufen; je weniger Arbeit es gibt, desto besser.

Um diesen Prozess zu starten, öffnen Sie Ihren bevorzugten Text-Editor. Geben Sie folgende Anweisungen ein:

```
CREATE DATABASE Temp;
USE DATABASE Temp;
CREATE TABLE Test_Table
    (Test_ID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    Test_Name VARCHAR(30),
    Test_Date DATETIME,
    Test_Giver VARCHAR(30));
INSERT INTO Test_Table
    (Test_ID, Test_Name, Test_Date, Test_Giver)
    VALUES (NULL, 'Test', '2000-01-01', 'Glen');
```

Das ist die geläufige Art, Ihre SQL-Anweisungen zu formatieren – ob sie gut oder schlecht ist, sei dahingestellt. Ich mag sie wegen der Übersichtlichkeit. Es dauert zwar ein bisschen, bis man sich daran gewöhnt hat, aber es ist wirklich gut lesbar und übersichtlich. Natürlich können Sie Ihre Befehle so eingeben, wie Sie möchten – solange diese in der gleichen Reihenfolge sind und mit einem Semikolon (;) enden.

In diesem Script erstellen Sie eine Datenbank namens `Temp`. Sie machen daraus die aktive Datenbank. Dann erstellen Sie eine Tabelle mit dem Namen `Test_Table` und fügen Ihrer Tabelle vier Spalten und einen Datensatz hinzu. Wenn Sie dies in einer Monitor-Session eintippen, so nimmt es einige Zeit in Anspruch, und beim Beenden der Session sind all Ihre Ausführungen verschwunden. Speichern Sie diese Datei unter `Temp.sql`. Sie könnten jeden beliebigen Namen benutzen; ich habe diesen gewählt, weil er sehr anschaulich ist.

Bevor Sie dieses Script benutzen, sollten Sie einiges beachten. Zunächst sollten Sie sich vergewissern, dass der `mysqld`-Daemon läuft und Sie sich im `mysql`-Verzeichnis befinden. Um die Datei an MySQL zu übergeben, tippen Sie schließlich folgende Anweisung auf Kommandozeilenebene ein:

```
bin/mysql-p </complete path/Temp.sql
```

Sie werden nun aufgefordert, ein Passwort einzugeben. Benutzen Sie das `root`-Passwort. Um das Script zu starten, müssen Sie die `CREATE`-Berechtigung haben. Nachdem Ihre Datei verarbeitet wurde, starten Sie den Monitor, indem Sie `Temp` als aktive Datenbank benutzen. Führen Sie eine `SHOW TABLE`-Anweisung aus. Jetzt sollten Sie die `Test_Table`-Tabelle vor sich haben. Geben Sie nun folgende Anweisung ein:

```
SELECT * FROM Test_Table;
```

Wie Sie sehen, ist das eine brauchbare Methode, SQL-Anweisungen über eine Textdatei auszuführen.

Das Ändern bestehender Tabellen

Jetzt haben Sie Ihre Tabelle erstellt. Was machen Sie aber, wenn Sie etwas ändern möchten? Tabellen zu verändern, ist fast so einfach wie Tabellen zu erstellen. Sie müssen nur wissen, was Sie verändern wollen. Der Name der Spalte unterscheidet sich grundlegend von dem der Tabelle. Den Spaltentyp zu verändern, ist etwas anderes als den Spaltennamen zu ändern. Um zu lernen, wie man den Namen einer Tabelle oder den Namen und Typ einer Spalte ändert, sehen Sie sich die folgenden Beispiele an.

Das Ändern eines Spaltennamens

Manchmal kann es vorkommen, dass Sie den Namen einer Ihrer Spalten ändern müssen. Ihr Chef kann zum Beispiel eine neue Namenskonvention vorgeben, die Sie befolgen müssen, oder Sie haben eine Spalte einfach falsch benannt, als Sie sie erstellt haben, und Sie bemerken diesen Fehler erst, als ein Kollege Sie darauf aufmerksam macht. Wie dem auch sei – einen Spaltennamen zu ändern, ist jedenfalls keine große Sache. Wollen Sie den Namen einer Spalte ändern, machen Sie Folgendes:

1. Vergewissern Sie sich, dass der `mysqld`-Daemon läuft und Sie sich im `mysql`-Verzeichnis befinden.
2. Starten Sie den MySQL-Monitor wie gehabt, und machen Sie die `Meet_A_Geek`-Datenbank zu der aktiven.
3. Um in der `Customers`-Tabelle den Namen der Spalte `First_Name` in `FirstName` abzuändern, geben Sie Folgendes auf Kommandozeilenebene ein:

```
ALTER TABLE CUSTOMERS
CHANGE First_Name FirstName VARCHAR(20);
DESCRIBE Customers;
```

Sie müssen den Datentyp genau eingeben, sonst erhalten Sie einen Fehler. Ich habe aus Gewohnheit den Befehl `DESCRIBE` benutzt, um diese Änderungen zu prüfen. Es ist allerdings nicht nötig, diesen Befehl beim Ändern der Tabellenstruktur zu benutzen.

Das Ändern eines Spaltentyps

Das Ändern eines Spaltentyps läuft ähnlich ab wie das Ändern eines Spaltennamens. Sie ändern beispielsweise den `Last_Name` von einem `Varchar(30)` in `Varchar (50)`. Befolgen Sie hierzu die Schritte 1 und 2 aus dem letzten Beispiel (Ändern eines Spaltennamens). Statt sich an die Ausführungen, die unter 3. stehen, zu halten, geben Sie schließlich Folgendes ein:

```
ALTER TABLE CUSTOMERS
CHANGE Last_Name VARCHAR(50);
DESCRIBE Customers;
```

Vergessen Sie nicht, dass Sie den Spaltennamen zweimal aufführen müssen. Der Grund dafür ist, dass MySQL eine temporäre Tabelle erstellt, die Ihre Änderungen enthält. Dies ermöglicht dem Benutzer, die Datenbank auch während Sie die Veränderungen vornehmen, zu benutzen.

Die Umbenennung einer Tabelle

Wenn Sie einen Tabellennamen ändern möchten, sollten Sie sich vergewissern, dass der `mysqld`-Daemon läuft und Sie sich im `mysql`-Verzeichnis befinden. Sind Sie sicher, das alles O.K. ist, starten Sie den MySQL-Monitor. Geben Sie Folgendes ein:

```
ALTER TABLE Customers RENAME Customer_Table;
SHOW TABLES FROM Meet_A_Geek;
```

Eine bestehende Tabelle oder Spalte zu verändern ist – trotz einiger syntaktischer Fallen – im Allgemeinen ziemlich einfach. Das Design ist dabei der eindeutig schwierigste Teil. Das sollten Sie beachten, wenn Sie die Arbeitszeit einplanen.

Das Hinzufügen/Löschen von Tabellen und Spalten

Wie Sie sehen, sind erstellte Tabellen oder Spalten nichts Statisches – sie können im allgemeinen ganz einfach verändert werden. Das Hinzufügen und Löschen von Spalten oder Tabellen bereitet keine großen Schwierigkeiten.

Das Löschen von Tabellen und Spalten

Um Tabellen oder Spalten hinzuzufügen oder zu löschen, vergewissern Sie sich, dass der `mysqld`-Daemon läuft und Sie sich im `mysql`-Verzeichnis befinden. Starten Sie den MySQL-Monitor mit der Datenbank, in der Sie Veränderungen vornehmen müssen, als aktive Datenbank.

Um eine bestehende Tabelle zu löschen, geben Sie Folgendes ein:

```
DROP Tabellenname;
```

Mit *Tabellenname* geben Sie die Tabelle an, die Sie löschen wollen. Möchten Sie zum Beispiel die `Customers`-Tabelle der `Meet_A_Geek`-Datenbank löschen, geben Sie folgenden Befehl ein:

```
DROP Customers;
```

Damit löschen Sie die ganze Tabelle samt Daten. Seien Sie vorsichtig bei der Ausführung solcher Befehle, denn es gibt keinerlei Warnungen. Haben Sie etwas gelöscht, können Sie es nur durch ein Backup-Log zurückholen.

Wenn Sie eine Spalte aus einer Tabelle löschen müssen, geben Sie Folgendes ein:

```
ALTER TABLE Tabellenname DROP Spaltenname;
```

Tabellenname steht wiederum für die zu löschende Tabelle und *Spaltenname* für die zu löschende Spalte.

Möchten Sie eine Spalte aus einer `Customers`-Tabelle löschen, geben Sie folgende Anweisung ein:

```
ALTER TABLE Customers DROP Last_Name;
```

Damit löschen Sie die Spalte samt der darin gespeicherten Daten.

Das Hinzufügen von Spalten

Das Hinzufügen von Tabellen in eine Datenbank wurde bereits behandelt. Sie haben gelernt, wie man eine Tabelle erstellt und löscht. Um aber eine Spalte hinzuzufügen, müssen Sie eine Variation des `ALTER TABLE`-Befehls benutzen. Sie müssen folgenden Befehl eingeben, um eine Spalte einem bestehenden Schema hinzuzufügen:

```
ALTER TABLE Tabellenname ADD Spaltenname Datentyp;
```

Tabellenname steht für die Tabelle, der eine Spalte hinzugefügt werden soll, und *Spaltenname* für die hinzugefügte Spalte. Wenn Sie die Spalte `Last_Name` der `Customers`-Tabelle wieder hinzuzufügen wollen, geben Sie Folgendes ein:

```
ALTER TABLE Customers ADD Last_Name Varchar(30);
```

Auf diese Weise fügen Sie Ihrer Tabelle eine Spalte hinzu. Ein wichtiger Punkt, auf den Sie achten sollten, ist, dass die Spalte, die hinzugefügt werden soll, einen Standardwert haben muss. Eine `NOT NULL`-Spalte kann nicht hinzugefügt werden. Die hinzuzufügende Spalte muss `NULL` oder irgendeinen anderen Standardwert enthalten. Der Grund hierfür ist ganz einfach. Wenn Sie eine Spalte, die `NOT NULL` ist, hinzufügen wollen, weiß MySQL nicht, welcher Wert gespeichert werden soll. Sie müssen immer genau angeben, was MySQL speichern soll.

Der Gebrauch von Indizes

Ein Index ist eine strukturierte Datei, die den Zugang zu den Daten erleichtert.

Das bedeutet für Sie, den Datenbank-Designer, Folgendes: Ein Index in der richtigen Spalte steigert die Schnelligkeit einer Anfrage enorm. Ein Index funktioniert ungefähr so wie alphabetische Trennblätter in einem Aktenschrank. Er ermöglicht Ihnen, zu der Stelle des Alphabets zu gelangen, die Sie brauchen. Stellen Sie sich vor, Sie bräuchten einen Datensatz von Glen Brazil. Mit Hilfe eines Indexes können Sie direkt zur Sektion B springen, ohne erst jeden einzelnen Datensatz durchsehen zu müssen. Auf diese Weise finden Sie das, was Sie suchen, sehr viel schneller.

Indizes sind eine tolle Sache, aber sie haben auch einige Nachteile. Zu viele Indizes können die gegenteilige Wirkung haben – die Arbeit wird nicht erleichtert, sondern unter Umständen erschwert. In dem eben erwähnten Beispiel sind Sie direkt zur Sektion B gesprungen. Was aber, wenn Sie ein Register hätten, das nach einzelnen Namen gegliedert ist. Sie hätten jede Menge Trennblätter – fast so viele wie Personen, die im Aktenschrank archiviert sind. Dies würde die Arbeit nicht einfacher und schneller machen, sondern im Gegenteil enorm verlangsamen. Es ist davon abzuraten, zu viele Indizes zu benutzen.

Ein weiterer Nachteil ist, dass das Hinzufügen einer Zeile in eine indizierte Tabelle etwas länger dauern kann, als dies bei einer nichtindizierten Tabelle der Fall wäre. In diesem Beispiel dauert es eine gewisse Zeit, einen Datensatz an der richtigen Stelle einzufügen. Sie müssen die »Trenner« durchblättern und diese dann im Aktenschrank in die richtige Reihenfolge bringen. Das dauert natürlich sehr viel länger, als ihn einfach irgendwo in den Aktenschrank zu werfen. Indizierte Spalten geben Informationen sehr viel schneller zurück. Wie Sie vorgehen, liegt ganz bei Ihnen.



Indizes beschleunigen den Datenzugang bei `SELECT`-Anweisungen, verlangsamten die Arbeit aber bei `INSERT`, `UPDATE` und `DELETE`-Anweisungen.

Entscheiden, welche Spalten indiziert werden sollen

Nachdem Sie sich entschieden haben, Indizes zu benutzen, müssen Sie die Spalte bzw. Spalten, die Sie indizieren möchten, auswählen. Das ist keine ganz einfache Aufgabe. Sie möchten die Spalte/n indizieren, die Sie in Ihren Anweisungen am häufigsten als Filter verwenden. Das sind die Spalten, die nach der `WHERE`-Klausel angeführt sind. In der `SELECT`-Anweisung `SELECT LAST_NAME FROM Customers WHERE Customer_ID < 10` wäre zum Beispiel die `Customer_ID`-Spalte ein potentieller Kandidat. Vergessen Sie nicht, die in Ihrer Anweisung am meisten benutzten Spalten zu indizieren. Wenn Sie viele Anfragen ausführen, in denen Sie den Nachnamen eines Kunden suchen, ist es sinnvoll, die Spalte `LAST_NAME` zu indizieren.

Zudem bearbeiten Indizes auch Spalten, die eindeutige Daten enthalten, besser. Aus diesem Grund sind Schlüssel normalerweise die beste Wahl für Indizes. Schlüssel und Indizes werden sehr häufig verwechselt. Sie sollten sich Folgendes merken: Schlüssel helfen, die Struktur einer Datenbank zu definieren, wohingegen Indizes die Ausführung verbessern.

Ein Index kann aus einer oder mehreren Spalten bestehen. Die `Meet_A_Geek`-Datenbank beispielsweise enthält einen Index, der auf den Spalten `LAST_NAME` und `FIRST_NAME` basiert. Dies ist sehr nützlich, wenn beide Spalten als Kriterium in der `WHERE`-Klausel einer SQL-Anweisung benutzt werden.

Außerdem können Sie in einer Tabelle bis zu 16 Indizes verwenden. Es wäre aber besser, wenn Sie niemals so viele Indizes benutzen müssten. Falls doch, werfen Sie einen Blick auf Ihr Datenbank-Design, denn beim Gebrauch vieler Indizes kann es zu Problemen kommen. Wenn Sie allerdings nur einige Indizes in einer Tabelle verwenden und dabei die vorhin vorgestellten Kriterien beachten, so dürften sich keine großen Probleme ergeben.

Das Erstellen eines Indexes

Im allgemeinen erstellt MySQL für Sie einen Index, wenn Sie eine Spalte als Primärschlüssel ausweisen. Es ist nicht notwendig, für diese Spalte einen Index zu erstellen; sonst hätten Sie zwei Indizes für die gleiche Spalte. Möchten Sie eine Spalte erstellen, geben Sie folgende Syntax ein:

```
CREATE INDEX Indexname ON Tabellename (Spaltennamen-liste);
```

Der *Indexname* ist beliebig. Es ist aber sinnvoll, einen beschreibenden Namen zu wählen, der Sie daran erinnert, was der Index beinhaltet. Beachten Sie das Schlüsselwort `ON`. Wenn Sie einen Index erstellen, sollten Sie sich auf jeden Fall vergewissern, dass Sie dieses Wort nicht vergessen haben – falls doch, wird mit Sicherheit ein Syntaxfehler angezeigt. Dem Schlüsselwort `ON` folgt der Name der Tabelle, welche die zu

indizierende Spalte enthält. Die *Spaltennamen*-liste ist eine Liste von Spalten, die Ihren Index enthält. Beachten Sie, dass ein Index aus einer oder mehreren Spalten bestehen kann.

Um einen Index hinzuzufügen, können Sie auch die Anweisung `ALTER TABLE` benutzen: Möchten Sie zum Beispiel der Spalte `LAST_NAME` der `Customers`-Tabelle einen Index hinzufügen, geben Sie Folgendes ein:

```
ALTER TABLE Customers ADD INDEX (IDX_Last_Name);
```

Die gleiche Syntax wird benutzt, um einer Tabelle einen Primärschlüssel hinzuzufügen. Die Anweisung hierfür sieht folgendermaßen aus:

```
ALTER TABLE Customers ADD PRIMARY KEY (Customer_ID);
```

Einen Index zu erstellen, ist ein relativ einfacher Prozess. Indizes spielen eine wesentliche Rolle in einer schnellen Datenbank, und MySQL arbeitet sehr gut mit ihnen. Achten Sie darauf, nicht zu viele Indizes zu benutzen, weil dies, wie bereits erwähnt, zu Problemen führen kann.

Das Löschen von Indizes

Einen Index zu löschen ist genauso einfach wie einen zu erstellen. Die Syntax ist die gleiche wie beim Löschen einer Spalte oder Tabelle. Sie können eine der beiden folgenden Anweisungen benutzen:

```
DROP INDEX Indexname ON Tabellenname;
```

oder

```
ALTER TABLE Tabellenname DROP INDEX Indexname;
```

Beide Anweisungen sind äquivalent. Vergessen Sie nicht: Wenn Sie eine Spalte löschen, die einen Index enthält, so wird dieser ebenfalls gelöscht. Wenn eine Spalte eines Multi-Spalten-Indexes gelöscht wird, wird auch nur diese gelöschte Spalte aus dem Index entfernt. Wenn alle Spalten, die einen Index bilden, gelöscht werden, so wird auch der gesamte Index gelöscht.

Wenn Sie einen `PRIMARY KEY` löschen müssen, benutzen Sie folgende Syntax:

```
ALTER TABLE Tabellenname DROP PRIMARY KEY;
```

Beachten Sie, dass eine Tabelle nur einen Primärschlüssel beinhalten kann. Falls Sie zu dem Schluss kommen, dass eine andere Spalte besser als Primärschlüssel geeignet ist, müssen Sie zuerst den vorherigen löschen.

Zusammenfassung

Heute haben Sie eine Menge Stoff durchgenommen. Sie haben die verschiedenen Tabellentypen, die MySQL anbietet, kennen gelernt und erfahren, wie Sie Tabellen erstellen, hinzufügen und löschen. Außerdem wurden einige Namenskonventionen und ihre Vorteile für die tägliche Arbeit mit einer Datenbank vorgestellt. Es wurde auch eingegangen auf die Bedeutung von Indizes und darauf, wie diese Ihre Datenbank schneller machen. Und was am wichtigsten ist: Sie haben gelernt, wie Sie Ihren Entwurf aus der Design-Phase verwirklichen.

Fragen und Antworten

F Ich finde temporäre Tabellen gut, möchte aber, dass sie für jeden einsehbar sind. Gibt es eine Möglichkeit, temporäre Tabellen für alle einsehbar zu machen?

A *Es gibt durchaus einen Weg. MySQL bietet eine temporäre Tabelle namens `HEAP` an, die für jeden, der eine Verbindung zum Server herstellt, zugänglich ist. Diese Tabelle verschwindet beim Herunterfahren des Servers. Sie existiert nur im Speicher und ist sehr schnell. Es gibt jedoch einige Einschränkungen. `TEXT`- und `BLOB`-Datentypen können nicht benutzt werden. Sie würden zu viel Speicher in Anspruch nehmen und somit den Ablauf verlangsamen. Alle Datentypen müssen eine bestimmte Länge haben. Dies trägt dazu bei, Ressourcen zu bewahren und den Ablauf zu beschleunigen. Auch können Indizes keine `NULL`-Werte beinhalten, und sie können nur in »ist gleich«(=) und »ist nicht gleich«(!=)Vergleichen benutzt werden. Um eine solche Tabelle zu erstellen, benutzen sie folgende Syntax:*

```
CREATE TABLE Tabellename (Spaltennamen Spaltentypen) TYPE=HEAP;
```

Sie können auch eine bestehende Tabelle in eine `HEAP`-Tabelle umändern. Vergessen Sie aber nicht, dass diese Tabelle automatisch gelöscht wird, wenn der Server herunterfährt. Die Anweisung hierfür sieht folgendermaßen aus:

```
ALTER TABLE tablename TYPE=HEAP;
```

F Ich habe andere RDBMS, wie Sybase und SQL Server 7, benutzt. Diese beinhalten Beschränkungen (Constraints) und Checks -Tabellen. Besitzt MySQL ebenfalls solche Tabellen?

A *Auf die Schnelle gesagt: Nein. MySQL kann diese Syntax aber so zerlegen, dass sie ohne große Schwierigkeiten bestehende Schemata in MySQL einführen kann.*

F Wie viele Daten kann eine MySQL-Tabelle speichern?

A *Die Beantwortung dieser Frage ist nicht ganz einfach. Die maximale Größe einer MySQL-Tabelle hängt ab vom Dateisystem des Betriebssystems. MySQL lagert seine Daten in Dateien, deren Größe absolut vom Betriebssystem abhängt. Bei 64-Bit-Systemen kann eine Tabelle im Prinzip unendlich groß sein. Bei einer Windows-NT-Plattform können Sie ungefähr mit einem 2-TB-Limit rechnen.*

Übungen

1. Erstellen Sie das Schema für ein Meet_A_Geek-Projekt. Bauen Sie das Schema auf dem Plan, der am dritten Tag entwickelt wurde, auf.
2. Erstellen Sie Indizes für die Meet_A_Geek-Datenbank. Bestimmen Sie zudem, welche Spalten katalogisiert werden sollen, indem Sie das Gelernte anwenden.



MySQL-Daten- typen

Woche
1

Wie die meisten relationalen Datenbank-Management-Systeme (RDBMS) verfügt auch MySQL über spezifische Spaltendatentypen. Heute werden Sie die folgenden kennen lernen:

- ▶ MySQL-Zeichenketten- und Zeichentypen
- ▶ MySQL-nummerische Typen
- ▶ Aufzählungen und Sets
- ▶ Spaltenattribute

Die von MySQL unterstützten Typen



MySQL besitzt verschiedene Datentypen, die unterschiedliche Funktionen unterstützen. Ein *Datentyp* ist der Typ von Daten, den eine Spalte speichert. Innerhalb einer Tabelle kann es viele verschiedene Datentypen geben, allerdings speichert jede Spalte ihren eigenen spezifischen Typ. Sie können sich einen Datentyp als eine Art Spaltendefinition vorstellen. Eine als Integer definierte Spalte beinhaltet nur numerische Informationen, wohingegen eine als `CHAR(10)` definierte Spalte bis zu zehn alphanumerische Zeichen enthält. Diese Definitionen sind der Schlüssel zu einer schnellen und effektiven Datenbank.

Es gibt im Großen und Ganzen drei Gruppen von Datenformaten. Numerische Formate beinhalten positive oder negative Zahlen wie zum Beispiel 4 und -50. Numerische Daten können auch im hexadezimalen (2ee250cc), wissenschaftlichen (2×10^{23}) oder dezimalen Format vorliegen. Das Zeichenketten- oder Zeichenformat besteht beispielsweise aus Buchstaben und Zahlen, vollständigen Wörtern, Adressen, Telefonnummern sowie aus allem, was Sie in Anführungszeichen setzen müssen. Den dritten und letzten Datentyp möchte ich in die Kategorie »Verschiedenes« einreihen – denn er beinhaltet alles, was nicht so recht in die beiden anderen Datentypen passt, wie zum Beispiel Daten und Zeiten, die als alphanumerisch gelten können, aber wie Zahlen gespeichert werden. Auf die verschiedenen Datentypen wird im Laufe des Kapitels noch ausführlicher eingegangen.

Neben Datentypen bietet MySQL auch Spaltenattribute mit den Bezeichnungen `AUTO_INCREMENT`, `UNSIGNED`, `PRIMARY KEY`, `NULL`, `NOT NULL` und `BINARY`. Diese Attribute tragen dazu bei, die Eigenschaften einer Spalte näher zu bestimmen. Eine ausführliche Beschreibung der Spaltenattribute folgt der Abhandlung über die grundlegenden Datentypen.

Numerische Typen

Numerische Typen speichern Zahlen. Sie können beispielsweise einen Buchstaben oder eine Zeichenkette folglich nicht in eine als numerisch definierte Spalte einfügen. Der numerische Datentyp kann weiter aufgegliedert werden in ganze Zahlen, Brüche, negative und positive Werte. Da die einzelnen numerischen Typen unterschiedlich groß sind, nehmen sie im Speicher auch unterschiedlich viel Platz ein.

»Warum all diese Schwierigkeiten?«, – fragen Sie sich vielleicht. »Eine Zahl ist doch eine Zahl.« Das mag zwar sein, aber das menschliche Gedächtnis speichert Zahlen anders als ein Computer. Stellen Sie sich vor, Sie gehen shoppen und kaufen in einem großen Süßwarenladen Süßigkeiten. Sie wissen schon: die Art von Läden, in denen Süßigkeiten in großen Behältern gelagert werden. Nehmen wir an, 500 g Süßigkeiten würden 1 \$ kosten. Es ist Halloween und Sie möchten 3 Pfund Süßigkeiten aus einem Behälter, der ungefähr 50 Pfund fasst. Als Sie an der Theke bezahlen, sagt Ihnen der Verkäufer, es würde 50 \$ kosten. Sie sind geschockt, denn Sie brauchen nur 3 Pfund Süßigkeiten, die eigentlich 3 \$ kosten sollten. Statt dessen sollen Sie für den gesamten Behälter bezahlen. Sie wollen nur das bezahlen, was Sie auch kaufen – aus dem gleichen Grund gibt es auch verschiedene Klassen von numerischen Typen. Denn gäbe es keine unterschiedlichen Klassen, müssten Sie jedes Mal, wenn Sie eine Zahl benutzen, acht Bytes an Speicherkapazität verbrauchen, weil MySQL nicht wüsste, welche Zahl gespeichert werden soll. Die verschiedenen Klassen ermöglichen also, Speicherkapazität zu sparen.

Die Namen der einzelnen MySQL-Typen und der dazugehörigen Klassen sind in der folgenden Tabelle aufgelistet.

Typenname	Speicherplatz
TINYINT	1 Byte
SMALLINT	2 Byte
MEDIUMINT	3 Byte
INT	4 Byte
BIGINT	8 Byte
FLOAT(M, D)	4 Byte
DOUBLE(M, D)	8 Byte
DECIMAL(M, D)	Wert von M + 2 Bytes

Tabelle 7.1: Numerische Speicherung

Ist eine Spalte als numerisch und `UNSIGNED` definiert, verdoppelt sich der Umfang für den jeweiligen Typ. Der Umfang einer Spalte, die durch `UNSIGNED TINYINT` bestimmt ist, beträgt 0 bis 255. Wird eine Spalte als `UNSIGNED` deklariert, dann beinhaltet sie nur positive Werte. Die Größe des Typs (`TINYINT`; `BIGINT`), den Sie verwenden, ändert sich nicht, nur der Umfang der gespeicherten Werte.

Typenname	Wertebereich	UNSIGNED
TINYINT	-128 bis 127	0 bis 255
SMALLINT	-32768 bis 32767	0 bis 65535
MEDIUMINT	-8388608 bis 8388607	0 bis 16777215
INT	-2147483648 bis 2147483647	0 bis 4294967295
BIGINT	-9223372036854775808 bis 9223372036854775807	0 bis 18446744073709550615
FLOAT(M, D)	Abhängig von den Werten	
DOUBLE(M, D)	Abhängig von den Werten	
DECIMAL(M, D)	Abhängig von den Werten	

Tabelle 7.2: Wertebereiche numerischer Datentypen

`FLOATS`, `DOUBLES` und `DECIMALS` sind numerische Typen, die Brüche enthalten können. MySQL ermöglicht Ihnen, die Anzahl der Dezimalstellen hinter dem Komma zu begrenzen. Stellen Sie sich vor, Sie sollen den Wert 5,6876 in einer Spalte des Typs `FLOAT (4,2)` speichern. Da MySQL die Dezimalstelle auf die im `FLOAT` hinter dem Komma angegebene Stelle auf- oder abrundet, wird die Zahl 5,69 gespeichert. Die gleiche Zahl, angegeben als `FLOAT (4,3)`, wird als 5,688 gespeichert und angezeigt. Diese Zahlen stellen also die Anzeigebreite dar. Die erste Zahl gibt die Anzahl der gesamten Dezimalstellen, die zweite die für den Bruch an. Die Anzeigenwerte sind nicht für alle numerischen Datentypen erforderlich.

Vergessen Sie nicht, dass MySQL normalerweise numerische Daten schneller verarbeitet als andere Datentypen. Wenn Sie also schnellere Anweisungen wollen, verwenden Sie für die Suche numerische Datentypen. Numerische Indizes sind im Allgemeinen ebenfalls schneller als Zeichen-gestützte.

Es ist sehr wichtig, den richtigen Typ für die jeweilige Arbeit auszuwählen. Beim Definieren Ihrer Spalten sollten Sie sich vergewissern, dass Sie auch den maximalen benötigten Wert berücksichtigt haben. Sollte dies nicht der Fall sein, kann es zu großen Problemen kommen – insbesondere wenn Ihre Datenbank für kritische Geschäftsanwendungen benutzt wird. Im Folgenden sind einige allgemeine Regeln für die Auswahl der richtigen numerischen Typen angeführt:

- ▶ Benutzen Sie numerische Typen für Primärschlüssel.
- ▶ Benutzen Sie `DECIMAL` für wirklich große Zahlen. `DECIMALS` werden unterschiedlich gespeichert und kennen keine Grenzen.
- ▶ Benutzen Sie `DECIMALS` bei Währungen, um Genauigkeit zu gewährleisten.
- ▶ Benutzen Sie auf jeden Fall den richtigen Datentyp. Verwenden Sie eine Zahl, die größer ist als benötigt, führt dies zu enormer Verschwendung und Unwirtschaftlichkeit.

Bei Datenübertragung von einer Datenbank auf eine andere sollten Sie sich vergewissern, dass Sie die passenden Datentypen ausgewählt haben. Ein falscher Datentyp kann zu enormen Problemen führen. Stellen Sie sich vor, Sie hätten den Datentyp `UNSIGNED TINYINT`, dessen Maximalwert 255 ist, gewählt und würden versuchen, den Wert 1000 einzufügen. MySQL würde lediglich den Wert 255 speichern, was im Rahmen einer wichtigen Geschäftsanwendung einer Katastrophe gleichkäme.

Die Attribute `AUTO_INCREMENT`, `UNSIGNED` und `ZEROFILL` können nur in Zusammenhang mit numerischen Daten benutzt werden. Sie führen Operationen aus, die nur mit Zahlen funktionieren. Sie haben bereits etwas über `UNSIGNED` gelernt – es macht Spalten positiv und lässt keine negativen Zahlen zu – lesen Sie weiter, um etwas über die anderen Attribute zu erfahren.

AUTO_INCREMENT

Das Spaltenattribut `AUTO_INCREMENT` erhöht automatisch den Wert einer Spalte, indem er dem aktuellen Maximalwert eins hinzufügt. Es liefert einen Zähler, der beim Erstellen eindeutiger Werte nützlich ist. Der Wert einer Zeile, die neu in eine `AUTO_INCREMENT`-Spalte eingefügt wurde, beginnt bei eins und erhöht sich mit jedem in die Tabelle eingefügten Datensatz um eins. Erstellen Sie eine Tabelle, die eine `AUTO_INCREMENT`-Spalte enthält. Fügen Sie zunächst einen Datensatz ein. Der Wert der `AUTO_INCREMENT`-Spalte beträgt eins. Fügen Sie einen weiteren Datensatz hinzu, beträgt der Wert zwei. Wie groß, glauben Sie, ist der Wert des Datensatzes, wenn Sie die allererste Zeile löschen und einen anderen Datensatz einfügen? Falls Sie auf drei getippt haben, liegen Sie richtig. Löschen Sie nun die neu eingefügte Zeile, deren Wert drei war, und fügen Sie eine andere Zeile ein. Was, denken Sie, ist nun der Wert der Zeile? Wenn Sie auf vier getippt haben, liegen Sie wieder richtig. Die `AUTO_INCREMENT`-Spalte verwendet einen (Maximal-)Wert nicht noch einmal, falls dieser gelöscht wurde. Es wird auch kein Wert, der kleiner als der aktuelle Maximalwert ist, vergeben.

Um vom `AUTO_INCREMENT`-Feature zu profitieren, müssen Sie beim Einfügen eines neuen Datensatzes in dem Feld, das automatisch um eins erhöht wird, eine `NULL`, 0 oder einen Leerraum benutzen. Mit folgender Anweisung können Sie von der `AUTO_INCREMENT` profitieren:

```
CREATE TABLE Test (Auto_Test int NOT NULL AUTO_INCREMENT);
INSERT INTO Test (Auto_Test) values(NULL);
INSERT INTO TEST (Auto_Test) values(0);
INSERT INTO Test (Auto_Test) values();
```

Sie können auch eine Zahl mit einschließen. Falls diese Zahl allerdings schon existiert, wird ein Fehler generiert, und falls sie dem höchsten Wert in der Spalte entspricht, wird der nächsteingefügte Wert um eins größer sein als der aktuelle. Wollen Sie also eine durch `AUTO_INCREMENT` veränderte Spalte, die bei 9000 beginnt, so muss Ihr erster Datensatz 9000 sein und alle nachfolgenden jeweils um eins höher. Sie können den Anfangspunkt durch folgende Syntax ganz exakt eingeben:

```
CREATE TABLE Test(
Test_ID INT NOT NULL AUTO_INCREMENT
AUTO_INCREMENT = 9000,
Another_Column INT);
```

Es ist völlig egal, wo in Ihrer Syntax Sie `AUTO_INCREMENT` platzieren. Denn in einer Tabelle kann es nur eine `AUTO_INCREMENT`-Spalte geben und MySQL ist clever genug, um zu wissen, wo dieser Wert einzusetzen ist.

Um die zuletzt eingefügte Datensatznummer zu erhalten, benutzen Sie die `LAST_INSERT_ID()`-Funktion. Diese Funktion gibt die zuletzt eingefügte Datensatznummer zurück – das bedeutet, es wird nicht der Datensatz irgendeines Benutzers Ihrer Datenbank zurückgegeben, sondern lediglich der von Ihnen zuletzt eingefügte Datensatz. Dies ist auch der Fall, wenn ein anderer Benutzer kurze Zeit nach Ihnen einen Datensatz eingibt. Sie sehen, diese Funktion kann sehr nützlich sein. Diese Funktion ist an Ihre Sitzung gebunden. Wenn Ihre Sitzung, nachdem Sie gerade erst einen Datensatz eingegeben haben, abstürzt und Sie nach Wiederherstellen der Verbindung versuchen, diese Funktion abzurufen, erhalten Sie eine Null, weil den Regeln von MySQL entsprechend gar nichts eingegeben wurde.

Eine letzte Sache, die Sie über das `AUTO_INCREMENT`-Attribut wissen sollten, ist, dass es nicht wieder von vorne anfängt zu zählen. Wenn Sie also den Maximalwert des von Ihnen ausgewählten Datentyps erreicht haben, wird ein Fehler angezeigt. Wählen Sie zum Beispiel ein `TINYINT` für eine `AUTO_INCREMENT`-Spalte, beträgt der maximale Wert 255. Versuchen Sie, nachdem der 255. Datensatz hinzugefügt wurde, einen weiteren hinzuzufügen, wird ebenfalls ein Fehler generiert. Um dies zu vermeiden, verwenden Sie den Typ `INT`. Dieser sollte Ihnen eine ausreichende Zahlenmenge zur Verfügung stellen.

Das `AUTO_INCREMENT`-Attribut funktioniert nur bei ganzen Zahlen. `FLOATS`, `DOUBLES` und `DECIMALS` können in einer `AUTO_INCREMENT`-Spalte nicht benutzt werden.

ZEROFILL

Das Spaltenattribut `ZEROFILL` wird gebraucht, um führenden Nullen bei einer Zahl, je nach Anzeigenbreite, anzuzeigen. Wie bereits erwähnt, besitzen alle numerischen Datentypen eine wählbare Anzeigenbreite. Geben Sie beispielsweise `INT(8) ZEROFILL` ein und beträgt der von Ihnen gespeicherte Wert `23`, so wird `00000023` angezeigt.

Zeichen- oder Zeichenketten-Datentypen

Die zweite wichtige Gruppe von Datentypen sind Zeichen- oder Zeichenketten-Datentypen. Ein Zeichenketten-Typ kann Daten wie »Hamburg« oder »Lindenstr. 17« speichern. In diesem Datentyp kann fast jeder Wert gespeichert werden. Die Größe des Speichers hängt davon ab, welchen Zeichenketten-Typ Sie wählen. Der für den jeweiligen Typ benötigte Speicher hängt mit der Länge der Zeichenketten zusammen. Maximalgröße und Speicherbeschreibung sind in der folgenden Tabelle aufgeführt.

Typenname	Max. Größe	Speicherplatz
CHAR(X)	255 Byte	X Byte
VARCHAR(X)	255 Byte	X+1 Byte
TINYTEXT	255 Byte	X+1 Byte
TINYBLOB	255 Byte	X+2 Byte
TEXT	65535 Byte	X+2 Byte
BLOB	65535 Byte	X+2 Byte
MEDIUMTEXT	1.6 Mbyte	X+3 Byte
MEDIUMBLOB	1.6 Mbyte	X+3 Byte
LONGTEXT	4.2 Gbyte	X+4 Byte
LOB	4.2 Gbyte	X+4 Byte

Tabelle 7.3: Zeichenketten-Typen

CHAR und VARCHAR

Von all diesen Typen sind `CHAR` und `VARCHAR` die am meisten gebrauchten. Sie unterscheiden sich dadurch, dass `VARCHAR` im Gegensatz zu `CHAR` eine variable Länge ist. `CHAR`-Typen werden für festgelegte Längen benutzt. Folglich verwenden Sie diese Typen, wenn sich Ihre Werte kaum verändern. Geben Sie zum Beispiel `CHAR(10)` ein, so sind alle in dieser Spalte angegebenen Werte 10 Byte lang – dies ist auch der Fall, wenn ein Wert beispielsweise nur drei Schriftzeichen beinhaltet. MySQL füllt alle Wer-

te auf, damit sie in die vorgegebene Größe passen. Der `VARCHAR`-Typ macht genau das Gegenteil. Wenn Sie `VARCHAR (10)` eingeben und einen Wert speichern, der drei Schriftzeichen enthält, beträgt der gesamte Speicherplatz vier Byte (die Länge + eins).

Der Vorteil der `CHAR`-Typen liegt darin, dass Tabellen, die diese festgelegten Werte enthalten, schneller verarbeitet werden als Tabellen, die aus `VARCHAR`-Typen gebildet sind. Ihr Nachteil besteht in der enormen Platzvergeudung. Für welchen Typ Sie sich entscheiden, liegt ganz bei Ihnen.

Normalerweise können `CHAR`- und `VARCHAR`-Typen nicht in derselben Tabelle verwendet werden. Falls Sie die beiden vermischen sollten, wechselt MySQL automatisch zu `VARCHAR`. Die einzige Ausnahme bilden kleine `VARCHAR`-Typen – »klein« bedeutet Werte von vier und kleiner. Benutzen Sie kleine `VARCHAR`-Typen, so ändert MySQL diese in `CHAR` um. Dies geschieht, wenn Sie zum Beispiel Folgendes eingeben:

```
CREATE TABLE Test (Fixed_Col CHAR (5), Var_Col VARCHAR (15));
```

MySQL ändert die Spalte `Fixed_Col` automatisch um in eine variable Länge. Den umgekehrten Vorgang erhalten Sie, wenn Sie Folgendes eingeben:

```
CREATE TABLE Test (Var_Col 1 VARCHAR (3))
```

MySQL ändert die Spalte in `CHAR`.

TEXT und BLOB

`TEXT` und `BLOB` (Binary Large Object) sind variable Längen, die große Mengen an Daten speichern können. Sie verwenden diese Typen, um Bilder, Geräusche oder große Mengen an Text wie Dokumente oder Websites zu speichern. Diese Typen sind zudem besonders gut geeignet, um Werte zu speichern, die in ihrer Größe von Datensatz zu Datensatz variieren. Wenn Sie zum Beispiel den Inhalt eines `<TEXTAREA>`-Feldes in einem Webseiten-Formular speichern wollen, wäre ein `TEXT`-Typ eine gute Wahl, da der eine vielleicht ein ganzes Buch in das Formular schreibt und ein anderer fast nichts. Sie können sich `TEXT` und `BLOB` als einen sehr großen `VARCHAR` vorstellen.

Der Vorteil von `TEXT` und `BLOB` gegenüber anderen Typen besteht darin, dass diese riesige Mengen an Daten speichern können. Sie können mit Hilfe dieser Typen ganze Dateien speichern. Von Nachteil ist allerdings, dass Dateien langsamer verarbeitet und riesige Speicherkapazitäten gebraucht werden, was unter Umständen zum Zusammenbruch führen kann.

In der neuesten Version von MySQL können Sie diesen Typen einen Index beifügen, was allerdings wegen der daraus resultierenden Performance-Verschlechterung Ihrer Datenbank nicht unbedingt zu empfehlen ist. Bei so großen Typen wie `TEXT` und `BLOB` können Indizes den Ablauf beträchtlich verlangsamen.

Vermischte Typen

Es gibt im Wesentlichen drei vermischte Typen: `ENUM`, `SET` und `DATE/TIME`. Diese drei Typen werden in diesem Abschnitt zusammengefasst, weil sie nicht ganz zu den beiden bereits vorgestellten Datentypen passen.

`SET` und `ENUM` werden heute vorgestellt, der `DATE/TIME`-Typ wird an Tag 16, »MySQL und die Zeit«, behandelt.

ENUM

Der `ENUM`-Typ ist eine aufzählende Liste, was bedeutet, dass diese Spalte nur einen der in der jeweiligen Liste angegebenen Werte speichern kann. Die `ENUM`-Spalte kann lediglich einen dieser Werte beinhalten. Um einen `ENUM`-Typ anzugeben, benutzen Sie folgende Syntax:

```
CREATE TABLE Test(  
    Return ENUM('Y','N') DEFAULT 'N',  
    Size ENUM('S','M','L','XL','XXL'),  
    Color ENUM('Black','Red','White')  
);
```

Sie können in Ihrer aufzählenden Liste bis zu 65.535 Posten angeben. `ENUM`-Typen eignen sich sehr gut für Auswahlboxen auf Webseitenformularen oder wenn Sie eine Wahl aus einer Werteliste treffen müssen. Achten Sie darauf, dass `ENUM` entweder einen Wert aus der Liste oder einen `NULL`-Wert enthalten muss. Versuchen Sie einen Wert einzufügen, der sich nicht auf der Liste befindet, wird eine Leerstelle eingefügt.

SET

Der `SET`-Typ ist `ENUM` sehr ähnlich. Er speichert ebenso wie `ENUM` eine Werteliste. Allerdings bietet `SET` im Unterschied zu `ENUM` mehrere Wahlmöglichkeiten. Er kann bis zu 64 Posten speichern. `SET`-Typen eignen sich besonders gut für Webseiten, auf denen der Benutzer mehr als einen Wert auswählen kann. Um einen `SET`-Typ anzugeben, benutzen Sie folgende Syntax:

```
CREATE TABLE Test(  
    Advertiser SET('Web Page','Television','Newspaper');  
);
```

Die anhand dieser Anweisung erstellte Spalte kann folgende Werte beinhalten:

```
"Web Page"  
"Television"  
"Newspaper"  
" "
```

Fügen Sie einen Datensatz in eine `SET`-Spalte ein, indem Sie die Werte mit Anführungszeichen umschließen und die einzelnen Werte durch Komma trennen. Um beispielsweise einen Datensatz, der zwei Werte aus der Tabelle des letzten Beispiels enthält, einzufügen, gebrauchen Sie folgende Syntax:

```
INSERT INTO Test (Advertiser) values('Web Page, Television')
```

Da die einzelnen `SET`-Werte durch Komma getrennt werden, sollten Sie niemals einen `SET`-Wert, der ein Komma enthält, benutzen, sonst werden Sie Probleme bekommen.

`SET`- und `ENUM`-Werte sind unter der Rubrik »Verschiedene Typen« aufgeführt, weil sie genauso wie Zeichenketten-Typen aussehen und arbeiten, MySQL sie aber als Zahlen speichert. Aus diesem Grund werden sie effektiver als Zeichenketten-Typen verarbeitet. Sie können auch bei numerischen Operationen eingesetzt werden. Sie können zum Beispiel die Zahl verwenden, die MySQL einsetzt, um die in der Tabelle gespeicherten Werte zurückzugeben. Werfen Sie einen Blick auf das folgende Beispiel:

```
SELECT * FROM Test WHERE Advertiser = 1
```

Diese Anweisung gibt sämtliche Werte aus `Test` zurück, wobei der `Advertiser` einer Webseite entspricht. Um die Werte, die von MySQL verwendet werden, einzusehen, können Sie folgende Anweisung verwenden:

```
SELECT Advertiser, Advertiser + 0 FROM Test
```

Die `SET`- und `ENUM`-Typen sind äußerst nützliche Spaltentypen, die Sie beim Entwerfen Ihrer Datenbank verwenden sollten. Denken Sie daran, dass `ENUM`-Typen nur eine Wahl aus der Liste ermöglichen, während `SET`-Typen mehrere Wahlmöglichkeiten zulassen.

Zusätzliche Spaltenattribute

MySQL besitzt mehrere Schlüsselwörter, die näher bestimmen, wie eine Spalte funktioniert. Sie haben zum Beispiel schon etwas über die Attribute `AUTO_INCREMENT`, `UNSIGNED` und `ZEROFILL` gehört und gelernt, wie diese die Spalten, in denen sie benutzt werden, beeinflussen. Einige Attribute passen nur zu bestimmten Spaltentypen. Werfen Sie einen Blick auf Tabelle 7.4, um zu sehen, welche Attribute für welche Spaltentypen anzuwenden sind.

Attributname	Verfügbare Typen
AUTO_INCREMENT	Alle Integer-Typen
BINARY	CHAR, VARCHAR
DEFAULT	Alle Typen außer BLOB und TEXT
NOT NULL	Alle Typen
NULL	Alle Typen
PRIMARY KEY	Alle Typen
UNIQUE	Alle Typen
UNSIGNED	Nummerische Datentypen
ZEROFILL	Nummerische Datentypen

Tabelle 7.4: Spaltenattribute

Das `BINARY`-Attribut sorgt dafür, dass die in diesen Typen gespeicherten Werte als binäre Zeichenketten behandelt werden und sie somit zwischen Klein- und Großschreibung unterscheiden. Wenn Sie diese Zeichenketten sortieren oder vergleichen, so werden diese ebenfalls Klein- und Großschreibung berücksichtigen. `VARCHAR`- und `CHAR`-Typen werden standardgemäß nicht als binär gespeichert.

Das `DEFAULT`-Attribut ermöglicht Ihnen, einen Wert in einer Spalte zu speichern, wenn keiner angegeben wird. Für alle Typen außer `ENUM` ist der MySQL-Standardwert `NULL`. MySQL benutzt bei `ENUM` den ersten Wert der aufzählenden Liste als Standard. Bei `SET`-Typen ist dies der »leere String«. Um einen `DEFAULT`-Wert zu bestimmen, benutzen Sie folgende Syntax:

```
CREATE TABLE Test(State char(2) NOT NULL DEFAULT "KY")
```

Diese Anweisung ordnet allen Datensätzen in der `State`-Spalte, die einen `NULL`- oder »leeren String«-Wert haben, den Standardwert »KY« zu.

Die Attribute `NULL` und `NOT NULL` geben an, ob eine Spalte einen Wert enthalten muss. In eine als `NOT NULL` definierte Spalte beispielsweise muss ein Wert eingefügt werden. Vergessen Sie nicht, dass `NULL` kein Wert ist. Ein »leerer String« – auch wenn er so aussieht, als ob er `NULL` wäre – ist `NOT NULL`. Der Gebrauch von `NULL` und `NOT NULL` kann den gespeicherten Daten die erforderlichen Einschränkungen auferlegen. Für weitere Informationen gehen Sie noch einmal Tag 3, »Das Entwerfen Ihrer ersten Datenbank«, durch.

Der `PRIMARY KEY` ist ein Index, der einzelne Werte enthalten muss. Er kann nicht `NULL` sein. Jede Tabelle sollte einen Schlüssel haben; MySQL ermöglicht Ihnen, indem es diesen Schlüssel als `PRIMARY KEY` angibt, ganz einfach einen Index zu erstellen.



Auch wenn keys für die Arbeit mit MySQL nicht absolut erforderlich sind, so sind sie doch ein wichtiges Werkzeug für die Performance einer Datenbank. Schlüssel werden ausführlich an Tag 11 behandelt.

Das `UNIQUE`-Attribut bewirkt, dass alle Daten innerhalb der angegebenen Spalte eindeutig sein müssen. Versuchen Sie Daten einzufügen, die nicht eindeutig sind, wird ein Fehler generiert.

Zusammenfassung

Wie Sie sehen, gibt es viele verschiedene Datentypen. Jeder Typ hat seine Besonderheiten. Es liegt ganz bei Ihnen, dem Datenbank-Designer den passenden Typ für die Daten, die Sie speichern wollen, auszuwählen. Das ist nicht immer ganz einfach. Denken Sie aber daran, dass Sie, falls Sie den falschen Typ gewählt haben, diesen mit der Anweisung `ALTER TABLE` ändern können.

Heute haben Sie die verschiedenen Datentypen kennen gelernt, wie zum Beispiel die numerischen Typen oder die Zeichenketten-Typen und Ihre jeweiligen Einsatzgebiete. Sie haben erfahren, dass es zwei verschiedene Zeichenketten-Typen gibt – die mit variabler und die mit festgelegter Länge –, und Sie haben die Vor- und Nachteile dieser beiden Typen kennen gelernt. Zudem wurde eingegangen auf die `SET`- und `ENUM`-Typen, die Unterschiede, die zwischen diesen beiden Typen bestehen, und auf die Vorteile gegenüber gewöhnlichen Zeichenketten-Typen. Schließlich wurden die verschiedenen Spaltenattribute und Ihre Funktionsweise vorgestellt.

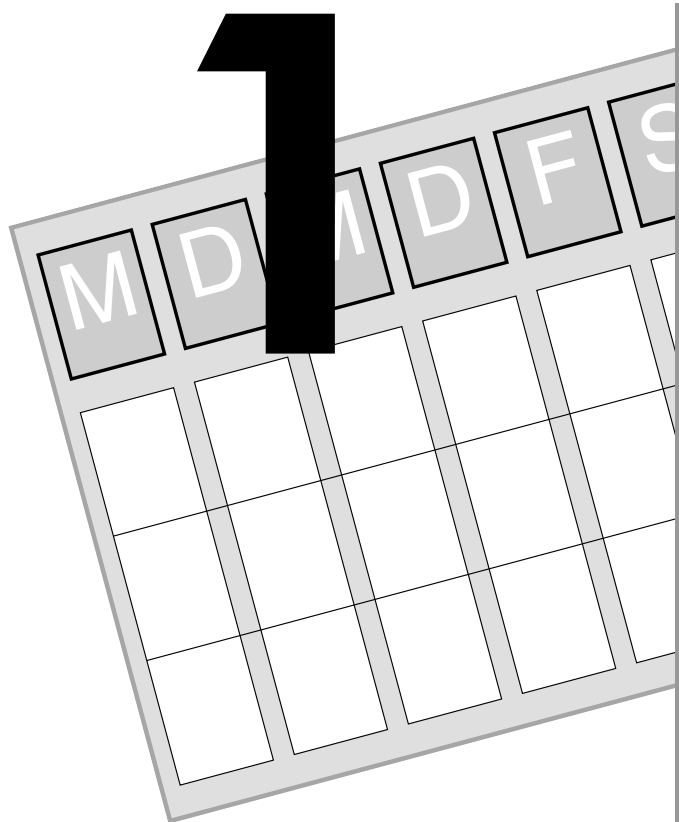
Fragen und Antworten

- F** Ich möchte die Werte meiner `SET`- und `ENUM`-Spalte sortieren. Wie mache ich das?
- A** *Die Reihenfolge, in der Sie sortieren, hängt von der Reihenfolge ab, in der die Werte eingefügt wurden. `SET`- und `ENUM`-Typen machen keinen Unterschied zwischen Klein- und Großschreibung. Die Werte, die eingefügt werden, gehen auf die Werte, die Sie beim Erstellen der Spalte bestimmt haben, zurück. Haben Sie zum Beispiel eine `ENUM`-Spalte mit den Werten »SCHWARZ«, »GRÜN« und »ROT« und fügen den Wert »rot« ein, wird dieser vor dem Speichern in »ROT« konvertiert.*

- F** Ich möchte Bilder in meiner Datenbank speichern. Wie mache ich das am besten?
- A** *Darüber gibt es unterschiedliche Meinungen. Ein Argument, das gegen das Speichern von Bildern und große Dateien in MySQL spricht, ist, dass MySQL das Dateisystem zur Datenspeicherung benutzt. Warum sollte das Dateisystem dann nicht direkt das betreffende Bild speichern und MySQL lediglich den Pfad zu diesem Bild? So müssten Sie sich keine Gedanken über die Fragmentierung Ihrer Datenbank machen. Das Problem ist, wenn Sie von einem entfernten Rechner auf Ihren Datenbankserver zugreifen, so müssen Sie erst einen Weg finden, wie Sie das Bild auf diesem im Dateisystem abspeichern können. Es gibt bei beiden Varianten Vor- und Nachteile. Wie Sie diese Varianten benutzen, werden Sie an den Tagen 12–15 lernen, wenn Sie mit den Schnittstellen zu MySQL vertraut gemacht werden.*

Übungen

1. Verwenden Sie das hier Gelernte und definieren Sie im Meet_A_Geek-Projekt alle Datentypen der Spalten neu.
2. Üben Sie das Einfügen von Datensätzen in eine Tabelle, indem Sie diese in Spalten mit den Datentypen SET und ENUM einsetzen.



Woche

Rückblick



Am Ende dieser Woche sollten Sie ganz gut mit dem MySQL-Monitor arbeiten können. Sie sollten ebenfalls gefestigte Kenntnisse über das Datenbank-Design besitzen. Außerdem sollten Sie jetzt wissen, wie eine relationale Datenbank mit den richtigen Datentypen erstellt wird.



Woche

Überblick



Sie sollten sich bei der Arbeit mit MySQL eigentlich ganz gut fühlen. Sie müssen keine GUIs lernen – nur einfache Befehle. Bis dato sollten Sie auch schon eine Muster-Datenbank erstellt haben, auf der sie in den folgenden Lektionen aufbauen werden.

Was Sie in dieser Woche lernen werden

Diese Woche behält das schnelle Tempo der ersten Woche bei. Es werden verschiedene Möglichkeiten vorgestellt, wie Sie MySQL benutzen können. Zudem werden Sie weiterhin mit der von Ihnen erstellten Musterdatenbank arbeiten.

In den Lektionen dieser Woche liegt der Schwerpunkt auf der Frage, wie Sie eine Datenbank benutzen. In der ersten Woche wird Ihnen gezeigt, wie Sie bestehende Datensätze in eine Datenbank einfügen und Datensätze von einer Datenbank auf die andere übertragen. Tag 9 behandelt die SQL-Programmiersprache. In dieser Lektion lernen Sie, wie Sie mit Ihrer Datenbank kommunizieren können. An Tag 10 erfahren Sie etwas über systemeigene Funktionen von MySQL. Tag 11 führt Sie in Tabellensperren und verschiedene Schlüssel in MySQL ein. Die Woche endet mit einer Reihe von Kapiteln, die sich mit programmierbaren Schnittstellen zur MySQL-Datenbank befassen.

Ihnen wird mal wieder jede Menge Information präsentiert. Wenn Sie ein wenig Zeit auf die Beispiele und Übungen verwenden, wird Ihnen alles klar werden.



Das Füllen der Datenbank mit Daten

**Woche
2**

Nun da Sie Ihre Datenbank entworfen und erstellt haben, sind Sie soweit, einige Daten einzufügen. Sie könnten zum Beispiel alle Daten auf Kommandozeilenebene eingeben – das könnte aber unter Umständen etwas langweilig sein, ganz zu schweigen von der Zeitverschwendung. MySQL bietet viele Möglichkeiten, Daten zu importieren und zu exportieren. Legen wir los! Der wichtigste Teil einer Datenbank sind die Daten, und MySQL stellt viele Werkzeuge bereit, die bei der Datenbewegung inner- und außerhalb der Datenbank behilflich sind.

Es gibt viele Arten, Ihre Datenbank mit Daten zu füllen. Es gibt auch zahlreiche Werkzeuge, die Ihnen hilfreich zur Seite stehen. Die CD-ROM enthält einige dieser *Third-Party-Tools* (Dienstprogramme anderer Hersteller). Heute lernen Sie Folgendes:

- ▶ einige Techniken, Daten zu importieren
- ▶ wie Sie Daten aus einer Microsoft-Access-Datenbank importieren
- ▶ wie Sie Daten aus einer MySQL-Datenbank exportieren

Methoden des Datenimports

Es gibt viele Gründe, weshalb man lernen sollte, wie Daten von einer Datenbank in eine andere importiert werden. Ein Grund hierfür könnte sein, dass Ihre aktuelle Datenbank nicht ausreicht. Sie müssen Datenbank-Produkte austauschen, wollen aber nicht alle über die Jahre angesammelten Daten verlieren. Ein weiterer Grund könnte sein, dass Sie nicht alle Daten über die Kommandozeile eingeben wollen. Dieser Vorgang kann äußerst langweilig, zeitaufwendig und fehleranfällig sein. Und letztendlich könnte es sein, dass sie Ihre Datenbank mit einer anderen synchronisieren möchten. Die Daten müssen übertragen werden, doch die Datenbanken besitzen nicht die gleichen Datenformate. Dies sind triftige Gründe, weshalb Sie die vielen verschiedenen Möglichkeiten, wie MySQL Ihnen beim Datenimport helfen kann, kennen lernen sollten.

Es gibt zahlreiche Methoden, Daten von einer bestehenden Datenbank in eine andere zu importieren. Bulk Copy Protocol (BCP) ist die gängigste Form, dies zu bewerkstelligen. Mit dieser Methode des Datenimports können textbasierte Daten sehr schnell in eine bestehende Datenstruktur importiert werden. Viele auf dem Markt befindliche Datenbanken, vornehmlich Microsoft SQL Server und Sybase Adaptive Server, benutzen das Bulk Copy Protocol.

Eine weitere gängige Methode des Datenimports ist der Dateikonvertierungs-Vorgang. Eine Datei, die in einer Datenbank in einem bestimmten Format geschrieben ist, wird für die neue Datenbank in ein anderes Format konvertiert. Dieses Verfahren lernen sie kennen, wenn Sie versuchen, Daten in MS Access zu importieren. Das Importwerkzeug lässt Sie einen Dateityp auswählen. Nachdem Sie Ihren Dateityp gewählt haben, wird dieser automatisch so konvertiert, dass er in den neuen MS-Access-Typ passt. Alte Datenbank-Dateien werden in der gleichen Art und Weise konvertiert.

Eine weitere Möglichkeit besteht darin, die Daten einer Datenbank in ein Format zu exportieren, das die andere Datenbank lesen kann. Durch Komma abgetrennte Dateien sind ein sehr gutes Beispiel. Eine Datenbank exportiert all ihre Daten in eine durch Komma abgetrennte Textdatei und die andere Datenbank liest diese ein. Dieses Vorgehen ähnelt Bulk Copy Protocol, doch arbeitet die eben vorgestellte Methode auf einem niedrigeren Niveau.

Extensible Markup Language (XML) ist die neueste Methode, Daten zu übertragen. Dies ist eine Sprache, die Hypertext Markup Language (HTML) ähnelt; doch ist sie robuster. XML ermöglicht Programmierern, ihre eigenen »Mark up Tags« zu erstellen, und gibt Entwicklern mehr Freiraum als HTML. Zudem bietet XML die Möglichkeit, Daten einfach und zuverlässig via Netzwerke zu übertragen. Die aktuellen Ausgaben aller großen Datenbanken haben diese neue Technologie auf die eine oder andere Weise übernommen. (Um weitere Informationen über XML zu erhalten, lesen Sie *XML in 21 Tagen*, erschienen bei Markt+Technik.) Die Entwickler von MySQL sind der Meinung, dass Parsen jeglicher Art keinesfalls auf Datenbank-Level vollzogen werden soll. Das ist sinnvoll. Es macht die Datenbank schneller, spart sämtliche Betriebskosten und überlässt dem Programmierer die freie Wahl des Formats, in dem die Daten ausgegeben werden. Die Programmierer haben die Möglichkeit, alles, was in ihre Datenbanken reinkommt oder aus diesen rausgeht, mit ihren eigenen Programmen zu parsen.

Wie Sie sehen, gibt es zahlreiche Möglichkeiten, Daten in eine Datenbank zu importieren. Welche Methode Sie wählen, liegt ganz bei Ihnen. Es gibt auch viele hilfreiche Werkzeuge, von denen MySQL glücklicherweise einige beinhaltet.

MySQL-Import-Tools

MySQL beinhaltet ein zwar etwas primitives aber äußerst effektives Utility, das speziell für das Verfrachten von Daten in eine Tabelle geschaffen wurde. Dieses Utility nennt sich `mysqlimport`. Es befindet sich im `mysql/bin`-Verzeichnis. Es handelt sich um ein Kommandozeilen-Utility, das zwei Parameter und eine Anzahl von Optionen beinhaltet. Dieses Hilfsprogramm importiert eine Textdatei in die von Ihnen angegebene Datenbank und Tabelle. Wenn Sie zum Beispiel eine Datei samt Daten in die `Customers`-Tabelle übertragen wollen, müssen Sie `mysqlimport` folgendermaßen einsetzen:

```
bin/mysqlimport Meet_A_Geek Customers.txt
```

Dieser Befehl überträgt den Inhalt der Textdatei in die Tabelle, die durch den Dateinamen bis zum ersten Punkt bestimmt wird (in den meisten Fällen also der Dateiname ohne die Dateiendung). In diesem Beispiel werden die Daten aus der Textdatei in die `Customer`-Tabelle übertragen. Trägt Ihre Datei beispielsweise den Namen `Cus.to.mer.txt`, so gehen die Daten aus dieser Tabelle in die `Cus`-Tabelle über. An diesem Beispiel wird deutlich, dass dieses Werkzeug zwar etwas primitiv, aber doch recht

effektiv ist; es erfüllt durchaus die gestellten Anforderungen. Erwähnenswert ist zudem, dass ein Fehler erscheint, falls eine Tabelle nicht existiert oder die Daten in der Datei nicht zu denen in den Tabellenspalten passen. Die Daten müssen in Typ und Anzahl zusammenpassen – wie dies auch bei einer `INSERT`-Anweisung der Fall ist. Wenn die Daten empfangende Tabelle aus dem letzten Beispiel die `Customer_ID INT`-, `Last_Name VARCHAR (25)`-und `First_Name VARCHAR(15)`-Spalten und ihre Datei die durch Komma abgegrenzten Werte "I", 1, "Stan", "Behm" enthielte, so würde ein Fehler erscheinen. Die Daten passen vom Typ her nicht zusammen – der erste Wert in der Datendatei ("I") passt nicht zum Datentyp dieser Spalte, der ein ganze Zahl ist. Das zweite Problem besteht darin, dass die Datendatei mehr Datensätze enthält, als die Tabelle Spalten hat. Um diese Art von Panne zu vermeiden, sollten Sie, bevor Sie die Daten einfügen, überprüfen, ob alles richtig ist.

mysqlimport-Optionen

Wie bereits erwähnt, kann das `mysqlimport`-Utility ebenfalls viele Optionen wahrnehmen. Folgende Tabelle zeigt eine Reihe möglicher Optionen:

Option	Bedeutung
<code>-d, --delete</code>	Leert die Tabellen, bevor die <code>INSERT</code> -Anweisungen, die in der Datei spezifiziert sind, ausgeführt werden.
<code>-f, --force</code>	Diese Option veranlasst <code>mysqlimport</code> , auch bei dem Auftreten von Fehlern die Daten weiter einzufügen.
<code>-h Hostname, --host Hostname</code>	Importiert die Daten auf den angegebenen <i>Hostnamen</i> statt auf den Standardrechner <i>localhost</i> .
<code>-L, --local</code>	Diese Option veranlasst <code>mysqlimport</code> , eine Datei von Ihrem Rechner, und nicht vom MySQL-Server, zu importieren, wenn Sie das Programm von einem entfernten Rechner ausführen.
<code>-l, --lock-tables</code>	Sperrt alle Tabellen, bevor die <code>INSERT</code> -Anweisungen ausgeführt werden.
<code>-u Benutzername, --user Benutzername</code>	Übergibt den MySQL-Benutzernamen, wenn mit dem Server Verbindung aufgenommen wird.
<code>-p [Passwort], --password[=Passwort]</code>	Wird normalerweise in Zusammenhang mit der Option <code>-u</code> verwendet und übergibt dann das zugehörige <i>Passwort</i> bei der Verbindungsaufnahme. Beachten Sie, dass keine Leerzeichen zwischen <code>-p</code> und dem <i>Passwort</i> stehen dürfen. Wenn Sie kein <i>Passwort</i> angeben, werden Sie von MySQL auf der Konsole dazu aufgefordert.

Tabelle 8.1: `mysqlimport`-Optionen

Option	Bedeutung
<code>-r, --replace</code> und <code>-i, --ignore</code>	Ermöglicht Ihnen die Kontrolle über Datensätze, die einzigartige Schlüsselwerte haben (»unique keys«). <code>-r</code> oder <code>--replace</code> bewirkt, dass importierte Datensätze bestehende Datensätze überschreibt, wenn Sie die gleichen einzigartigen Schlüsselwerte haben. <code>-i</code> oder <code>--ignore</code> bewirkt, dass solche Datensätze übergangen und nicht importiert werden. Wenn keine dieser beiden Optionen angegeben ist, und ein solcher doppelter Datensatz auftaucht, wird eine Fehlermeldung ausgegeben und der Import abgebrochen (beachten Sie, dass, wenn Sie eine <code>DROP TABLE</code> -Anweisung zu Beginn des Imports ausführen, das Auftauchen einer solchen Situation sehr unwahrscheinlich ist).
<code>-s, --silent</code>	Unterdrückt MySQLs Nachrichten und gibt nur Fehlermeldungen an der Standardausgabe aus.
<code>--fields-terminated-by="Trennzeichen"</code>	Gibt das Zeichen an, welches als Trenner nach jedem Feld benutzt wird. Standard ist hier der Tabulator <code>\t</code> .
<code>--fields-enclosed-by="Trennzeichen"</code>	Spezifiziert das Trennzeichen, welches jedes Feld umschließt.
<code>--fields-optionally-enclosed-by="Trennzeichen"</code>	Spezifiziert das Trennzeichen, welches <code>CHAR</code> - oder <code>VARCHAR</code> -Felder umschließt.
<code>--fields-escaped-by="Escape-Zeichen"</code>	Gibt das <i>Escape-Zeichen</i> an, welches vor spezielle Zeichen gesetzt wird. Standard ist <code>\\</code> (führt zu einem Backslash).
<code>--lines-terminated-by='Trennzeichen'</code>	Gibt das <i>Zeilen-Trennzeichen</i> an. Standard ist <code>\n</code> (newline).
<code>-, --help</code>	Zeigt eine Liste aller Optionen des <code>mysqlimport</code> -Programms an und beendet es anschließend wieder.

Tabelle 8.1: `mysqlimport`-Optionen (Forts.)

Import einer durch Komma abgegrenzten Datei

Stellen Sie sich vor, Sie müssten eine durch Komma abgegrenzte Textdatei, die Sie aus einem Spreadsheet-Programm generiert haben, importieren. Jede Spalte mit Daten wird durch ein Komma getrennt und mit Anführungszeichen umschlossen. Dies sieht folgendermaßen aus:

```
"1", "ORDDD89876", "1 Dozen Roses", "19991226"
```

Einige der Dateidaten stammen aus der Datenbank. Sie wurden möglicherweise bereits aktualisiert – vielleicht aber auch noch nicht. Die Aktualisierung der Daten muss am Ende dieses Tages auf dem Produktionssystem durchgeführt werden. Ihre Aufgabe besteht darin, diese Daten in die `Orders`-Tabelle der `Meet_A_Geek`-Datenbank zu importieren. Sie müssen dazu folgenden Befehl benutzen:

```
%> bin/mysqlimport -prl fields-enclosed-by="
-fields-terminated-by=, Meet_A_Geek Orders.txt
```

Der Anblick dieses Befehls mag Sie vielleicht ein wenig einschüchtern, doch wenn Sie erst einmal drin sind, ist es ziemlich einfach. Der erste Teil, `bin/mysqlimport`, übermittelt dem Betriebssystem, welchen Befehl Sie ausführen wollen. Die Option `p` wurde benutzt, weil Sie wissen, wie man eine Datenbank sichert, und Sie sich ausweisen müssen, bevor Sie irgendwelche strukturellen Änderungen vornehmen. Die Option `r` wird benutzt, weil Sie jegliche Datensätze, die den gleichen eindeutigen Schlüssel aufweisen, ersetzen wollen. Dies ist notwendig, weil Ihnen übermittelt wurde, dass das Spreadsheet in diesem Szenario möglicherweise aktualisiert wurde. Sie müssen sich vorstellen, dass Ihr Spreadsheet die aktuellsten Daten enthält und Sie sämtliche Daten in der Datenbank durch die aktuellen Daten ersetzen wollen. Die `l`-Option wird benutzt, weil Sie die Tabelle sperren wollen, während Sie Datensätze einfügen. Dadurch wird verhindert, dass ein Benutzer Daten auswählt oder abändert, während Sie die Tabelle aktualisieren. Sie werden mehr über Locks (Sperren) an Tag 11 erfahren. Die nächsten Optionen beschreiben die Daten, die in Ihrer Importdatei gespeichert werden. Sie übermitteln `mysqlimport`, dass Ihre Daten durch Anführungszeichen umschlossen und die Felder durch Komma abgetrennt werden. Schließlich geben Sie `mysqlimport` vor, welche Datenbank benutzt werden soll und in welche Tabelle die Daten einzufügen sind. Vergessen Sie nicht, dass `mysqlimport` den Tabellennamen vom Dateinamen übernimmt. `mysqlimport` geht davon aus, dass der gesamte Raum links vor dem ersten Punkt für den Tabellennamen zur Verfügung steht. Versichern Sie sich also, dass Ihr Dateiname zu Ihrem Tabellennamen passt.

Import im Batchmodus

MySQL kann Daten auch im Batchmodus importieren. Diese Methode ermöglicht es, das `mysql`-Programm mehr passiv als interaktiv ablaufen zu lassen – genauso wie Sie es bei der Arbeit mit dem MySQL-Monitor kennen gelernt haben. Dennoch benutzen Sie bei dieser Methode den gleichen Befehl. Dazu weisen Sie dem `mysql`-Programm erneut eine Datei zu. Das alles klingt etwas verwirrend? Ist es nicht. In Wirklichkeit ist es ziemlich einfach.

Sie benötigen eine Textdatei, die den gleichen Text, den Sie auch in den MySQL-Monitor eintippen würden, enthält. Stellen Sie sich beispielsweise vor, Sie möchten einige Daten einfügen. Sie könnten eine Textdatei mit folgendem Text erstellen:

```
USE Meet_A_Geek;  
INSERT INTO Customers (Customer_ID, Last_Name)  
VALUES(NULL, "Block");  
INSERT INTO Customers (Customer_ID, Last_Name)  
VALUES(NULL, "Newton");  
INSERT INTO Customers (Customer_ID, Last_Name)  
VALUES(NULL, "Simmons");
```

Nehmen Sie zur Kenntnis, dass diese Befehle syntaktisch korrekt sind (SQL-Anweisungen werden an Tag 9 ausführlicher behandelt) und mit einem Semikolon beendet werden. Sie verfahren ebenso wie bei der Arbeit mit dem MySQL-Monitor. Im nächsten Abschnitt werden Sie diese Daten in MySQL importieren. Bevor Sie beginnen können, muss der `mysqld`-Daemon laufen und Sie müssen sich im `mysql`-Verzeichnis befinden. Geben Sie Folgendes auf der Kommandozeile ein:

```
bin/mysql -p < /home/mark/New_Data.sql
```



Daemon ist ein Unix-Begriff, der im Großen und Ganzen Windows NT Service entspricht.

Danach werden Sie aufgefordert, ein Passwort einzugeben. Die Datei wird schließlich in das `mysql`-Programm übertragen. `mysql` verarbeitet jede Zeile so, als ob Sie diese in den Monitor eingetippt hätten. Dies ist eine sehr schnelle und effektive Art und Weise, SQL-Anweisungen zu verarbeiten. Solange Ihre Anweisungen korrekt sind, werden sie auch ausgeführt.

Die `USE`-Anweisung wird benötigt, da die Standarddatenbank nicht angegeben wurde. Sie hätten die Datenbank auch angeben und die `USE`-Anweisung auslassen können. Diese Anweisung wird hier benutzt, um zu zeigen, wie Befehle, die Sie in das Monitor-Programm eingeben könnten, in einer Datei benutzt werden können.

Jetzt fragen Sie sich vielleicht: »Warum um alles in der Welt sollte ich all diese SQL-Anweisungen in eine Datei eingeben und sie anschließend durch das Programm laufen lassen? Das sieht nach einer Menge Arbeit aus.« Sie haben wahrscheinlich Recht. Was wäre aber, wenn Sie ein Log hätten, das für Sie alle Befehle generieren würde. Das wäre ziemlich klasse, nicht wahr? Die meisten Datenbanken generieren ein Ergebnisprotokoll, das in der Datenbank erscheint. Ein Großteil dieser Logs enthält benutzte SQL-Anweisungen, die noch nicht verarbeitet wurden. Wenn Sie also die Datei nicht aus Ihrer aktuellen Datenbank exportieren können, um sie in Ihrer neuen MySQL-Datenbank zu benutzen, können Sie dieses Log und den Batchmodus von MySQL benutzen, um Ihre Daten schnell und einfach zu importieren. Sie werden mir zustimmen, das ist allemal besser, als alles einzutippen.

LOAD DATA INFILE

Die letzte Methode, Daten in eine MySQL-Datenbank zu importieren, ähnelt dem `mysqlimport`-Utility. Im Gegensatz zu diesem kann sie allerdings im MySQL-Monitor ausgeführt werden, was bedeutet, dass sie auch von jeder Anwendung ausgeführt werden kann, die Sie mit Hilfe von APIs geschrieben haben. Wenn Sie wollen, können Sie auf diese Weise Daten anhand Ihrer eigenen Anwendungen importieren.

Um diesen Prozess zu starten, muss der `mysqld`-Daemon laufen und Sie müssen sich im `mysql`-Verzeichnis befinden. Starten Sie den MySQL-Monitor. Falls Sie es vergessen haben sollten – Sie müssen folgendes auf Kommandozeilenebene eintippen:

```
bin/mysql -p
```

Sie werden aufgefordert, ein Passwort einzugeben. Nachdem Sie den Monitor erfolgreich gestartet haben, geben Sie folgenden Befehl ein, um die `Meet_A_Geek`-Datenbank benutzen zu können:

```
USE Meet_A_Geek;
```

Der Monitor zeigt Ihnen an, dass sich die Datenbank verändert hat. Geben Sie folgenden Befehle auf Kommandozeilenebene ein:

```
LOAD DATA INFILE "/home/mark/data.sql"
INTO TABLE Orders;
```

Genauso wie das `mysqlimport`-Utility führt dies den Inhalt der `data.sql`-Datei in die `Orders`-Tabelle ein. Diese Anweisung besitzt, wie auch das Utility, einige fakultative Argumente. Befindet sich zum Beispiel die zu importierende Datei auf Ihrem PC, der Datenbank-Server aber in einem anderem Gebäude, können Sie folgenden Befehl eingeben:

```
LOAD DATA LOCAL INFILE "C:\MyDocs\SQL.txt"
INTO TABLE Orders;
```

Dies führt den Inhalt der Datei in die `Orders`-Tabelle ein. Sie müssen die Datei nicht selbst übertragen, MySQL erledigt das für Sie.

Sie können auch der `INSERT`-Anweisung den Vorrang geben. Markieren Sie dies als `LOW_PRIORITY`, wartet MySQL mit der Dateneinfügung so lange, bis kein anderer Benutzer die Tabelle einsieht. Dieser Befehl sieht folgendermaßen aus:

```
LOAD DATA LOW_PRIORITY INFILE "/home/mark/data.sql"
INTO TABLE Orders;
```

Sie können zudem bestimmen, ebenso wie mit dem `mysqlimport`-Utility, ob Sie Positionen mit doppelten Schlüsselwerten ersetzen oder ignorieren wollen. Um doppelte Werte zu ersetzen, benutzen Sie folgende Syntax:

```
LOAD DATA LOW_PRIORITY INFILE "/home/mark/data.sql"
REPLACE INTO TABLE Orders;
```

Das erscheint zwar etwas unpraktisch, aber das Schlüsselwort muss an dieser Stelle stehen, damit der Parser versteht, was Sie wollen.

Die nächsten Optionen beschreiben das Layout der Datei. Es handelt sich um die gleichen Optionen, die auch im `mysqlimport`-Utility erhältlich sind. Sie unterscheiden sich lediglich optisch. Zunächst wird das Schlüsselwort `FIELDS` benutzt. Beim Gebrauch dieses Wortes erwartet der MySQL-Parser eines der folgenden Schlüsselwörter:

- ▶ `TERMINATED BY` *Zeichen*
- ▶ `ENCLOSED BY` *Zeichen*
- ▶ `ESCAPED BY` *Zeichen*

Diese Schlüsselwörter und deren Argumente werden genauso wie die `mysqlimport`-Optionen gebraucht. Der Ausdruck `TERMINATED BY` zeigt an, wodurch die Felder in der Datei getrennt werden. Im Normalfall ist es das Tab-Zeichen (`\t`). Der Ausdruck `ENCLOSED BY` gibt die umschließenden Zeichen an. Sind die Felder von Anführungszeichen umschlossen, geben Sie an, dass diese diesen Befehl benutzten. Der Ausdruck `ESCAPED BY` beschreibt letztendlich das Escape-Zeichen. Gewöhnlich ist dies der Backslash (`\`). Um bei dem vorherigen Beispiel zu bleiben, würde Ihre Anweisung nun folgendermaßen aussehen:

```
LOAD DATA INFILE "/home/mark/Orders.txt" REPLACE
INTO TABLE Orders
FIELDS
TERMINATED BY ','
ENCLOSED BY ''';
```

Ein Feature, das in der `LOAD DATA INFILE`-Anweisung zwar nicht aber im `mysqlimport`-Utility verfügbar ist, ist die Fähigkeit, bestimmte Spalten, in die Daten abgelegt werden sollen, zu benennen. Dies ist äußerst wichtig, wenn Sie nur einen Teil der Daten importieren. Wenn Sie von Access zu MySQL wechseln und Sie der MySQL-Datenbank Spalten hinzugefügt haben, um zusätzliche Anforderungen unterzubringen, können Sie die Daten in Ihrer aktuellen Datenbank immer noch gebrauchen. Da aber die Anzahl der Spalten jetzt unterschiedlich ist, können Sie das Import-Utility nicht mehr benutzen. Hierbei kann Ihnen die `LOAD DATA INFILE`-Anweisung Hilfestellung geben. Im Folgenden wird gezeigt, wie Sie Daten in die von Ihnen gewünschte Spalte bugsieren:

```
LOAD DATA INFILE "/home/Order.txt"
INTO TABLE Orders
(Order_Number, Order_Date, Customer_ID);
```

Wie Sie sehen, können Sie so viele Spalten angeben, wie Sie wollen. Die Spaltennamen stehen ebenfalls in Klammern und werden durch Komma getrennt. Falls Sie eine der beiden Ausführungen vergessen sollten, wird MySQL sich schon melden.

Datenimport aus Microsoft Access

Im nächsten Schritt soll das Gelernte umgesetzt werden. Sie werden Daten aus einer Microsoft-Access-Datenbank importieren. Zur Bewerkstelligung dieses Datenimports stehen Ihnen einige *Third-Party-Tools* zur Verfügung. Sie werden dies aber auf die altmodische Art und Weise durchführen – durch harte Arbeit und Scharfsinn.

Es gibt zahlreiche Möglichkeiten, um Daten aus Microsoft Access zu importieren. Die einfachste besteht darin, einen Datenquellennamen (DSN) zu erstellen und ODBC zu benutzen. Bevor Sie eine DSN erstellen können, müssen Sie den *Open-Data-Base-Connectivity*- (ODBC-) Treiber installieren. Dieser Treiber ist eine von Microsoft entworfene Technologie, die behilflich ist bei der Erstellung einer Umgebung, in der Daten ohne Beachtung der Datenquelle ganz einfach unter den Applikationen verteilt werden können. ODBC agiert als »Vermittler« zwischen Applikation und Datenbank. Die Software, die dies bewerkstelligt, heißt *Treiber*. Jede Datenbank besitzt ihren eigenen Treiber.

Das ODBC-Interface und DSN werden an Tag 13, »Wie benutze ich MyODBC?«, behandelt. Der Gebrauch von DSN und ODBC ist aber nicht immer der beste Weg. Manchmal müssen Sie einige Arbeitsschritte einschieben, um letztendlich etwas Zeit zu sparen. Heute lernen Sie, dies zu bewerkstelligen, indem Sie Daten in Textdateien exportieren und diese Dateien anschließend in MySQL unterbringen. Dieses Verfahren ist zwar nicht sehr beliebt, aber es kann sehr nützlich sein, wenn Sie einmal nicht die Möglichkeit haben sollten, einen ODBC-Treiber zu installieren und einen DSN zu benutzen, der auf eine MySQL-Datenbank hinweist. In dieser Situation ist es also sehr hilfreich, dieses Verfahren zu kennen.

Der nächste Abschnitt setzt voraus, dass Sie Zugang zu Microsoft Access haben. Sollte dies nicht der Fall sein, lesen Sie dennoch diesen Abschnitt. Hier werden jede Menge Screenshots vorgestellt, die zum Verständnis der Techniken, die in den Prozess der Datenübertragung zwischen Datenbanken eingebunden sind, beitragen.

Zuerst benötigen Sie eine Datenbank, die Tabellen beinhaltet, die von Ihnen exportiert werden können. Sie werden hier die *Nordwind*-Datenbank verwenden, die standardmäßig bei der Microsoft-Access-Distribution enthalten ist. Aus dieser sollen Dateien in Ihre MySQL-Datenbank exportieren werden.

1. Öffnen Sie die Datenbank. Sie sollten jetzt alle Datenbank-Objekte – Tabellen, Abfragen, Berichte und Formen – vor sich haben (siehe Abbildung 8.1).
2. Sie werden die *Bestellungen*- und *Kunden*-Tabellen benutzen. Öffnen Sie die *Bestellungen*-Tabelle. Sie werden sehen, dass die Daten im *dd-mm-yyyy*-Format gespeichert sind. Beim Datenimport in MySQL kann dies allerdings zu Problemen führen, da MySQL-Daten im *yyyy-mm-dd*-Format gespeichert werden. In der Access-Datenbank kann dieses Problem am einfachsten behoben werden. Wechseln Sie in die Entwurfsansicht und ändern Sie das Datenformat derart, dass es dem von MySQL entspricht.

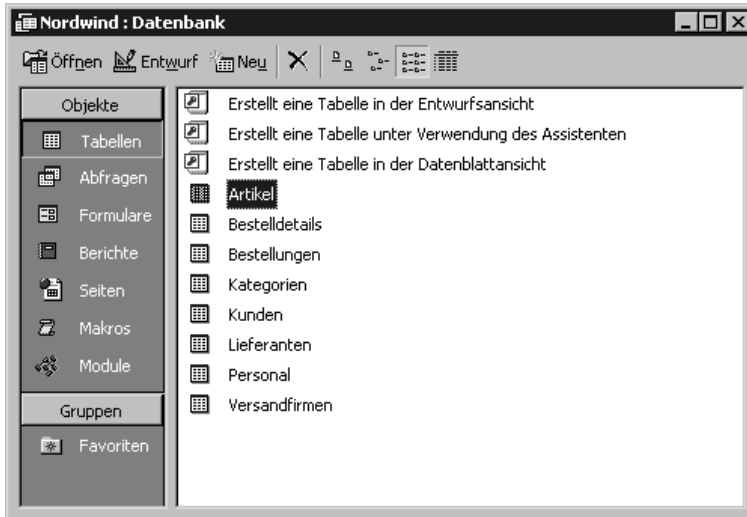


Abbildung 8.1:
Die Nordwind-
Datenbank

3. Versuchen Sie nun, die Daten der *Bestellungen*-Tabelle in die gleichnamige Tabelle der MySQL-Datenbank zu exportieren. Hierbei ergeben sich aber einige Probleme. Die Nordwind *Bestellungen*-Tabelle ist nicht ebenso normalisiert wie Ihre Tabelle. Es erfordert von Ihnen etwas Geschick, diese Tabelle auf den Stand Ihrer Datenbank zu bringen.
4. Das Vorhaben besteht darin, einen Anweisungsplan zu erstellen, der die Daten Ihrer Wahl in eine Komma separierte Textdatei (csv-Datei) exportiert. Dazu klicken Sie das Icon *Abfragen* auf der linken Seite des Datenbankobjekt-Fensters an (siehe Abbildung 8.2).

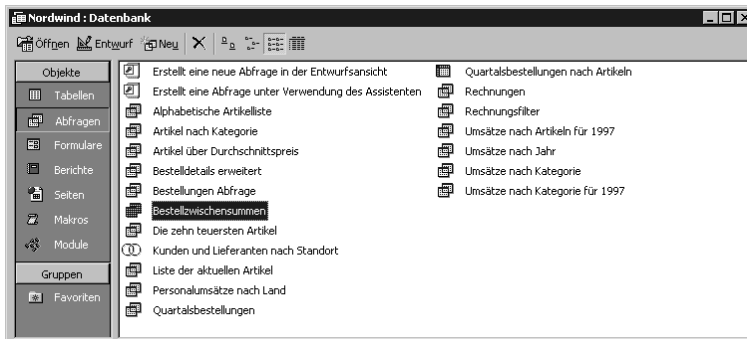


Abbildung 8.2:
Das Nordwind-
Abfrage-Objekt

5. Sie werden jetzt einen Anweisungsplan entwerfen, der die Daten herauszieht, die Sie zum Füllen der Spalten in Ihren Tabellen benötigen. Klicken Sie im Abfrage-Fenster *Neu* an. Das »Neue Abfrage«-Dialog-Fenster wird geöffnet. Klicken Sie auf *Entwurfsansicht* und bestätigen Sie mit *OK*.

6. Nun erscheint das »Tabellen anzeigen«-Fenster. Wählen Sie die Tabellen, welche die hier benötigten Daten enthalten. Sie müssen die `Kunden`- und `Bestellungen`-Tabellen auswählen, da die `Bestellungen`-Tabelle den Kundennamen, nicht aber das `Kunden-Code`-Feld enthält.
7. Nachdem Sie die Tabellen ausgewählt haben, müssen Sie die Felder, die in Ihrer Anweisung erscheinen sollen, auswählen. Bei einem Blick auf die ausgewählten Felder, können Sie feststellen, dass sich darunter einige Felder aus der `Meet_A_Geek`-Datenbank befinden. Da Sie Ihre Tabelle bereits in der `MySQL`-Datenbank erstellt haben, müssen Sie den Befehl `LOAD DATA INFILE` statt des `mysqlimport`-Utilities benutzen. Wählen Sie folgende Felder:
 - ▶ Das `Kunden-Code`-Feld aus der `Kunden`-Tabelle.
 - ▶ Die `Bestell-Nr`-, `Bestelldatum`-, `Lieferdatum`- und `Versanddatum`-Felder aus der `Bestellungen`-Tabelle. Das Ergebnis sollte der Abbildung 8.3 gleichen.

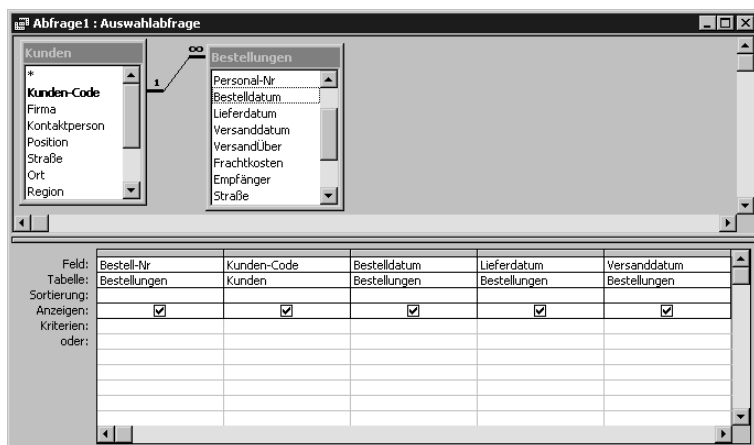


Abbildung 8.3:
Die Abfrage-
auswahl

8. Nachdem Sie die Felder ausgewählt haben, starten Sie die Anfrage. Klicken Sie dazu das Ausrufezeichen auf der Symbolleiste an. Ihr Bildschirm sollte die Abbildung 8.4 anzeigen.
9. Speichern Sie die Anweisung unter `Bestellungen_Import`. Nachdem Sie die Anweisung gespeichert haben, wechseln Sie in die Datenbank-Objektansicht und klicken mit Rechts auf den Namen der soeben erstellten Abfrage. Ein Pop-Up-Menü sollte erscheinen. Wählen Sie in diesem `Export` aus.
10. Sie sollten nun das »Exportieren einer Anfrage«-Dialogfeld vor sich haben (siehe Abbildung 8.5). Der Dateiname sollte auch, wie üblich, der Name der Anweisung sein – `Bestellungen_Import`. Unter dem Dateinamen befinden sich Ihre Export-Auswahlmöglichkeiten. Wählen Sie die Option `Textdatei` und klicken Sie auf `Speichern`.

Bestell-Nr	Kunden-Code	Bestelldatum	Lieferdatum	Versanddatum
10646	ALFKI	1997-08-25	1997-09-22	1997-09-02
10692	ALFKI	1997-10-03	1997-10-31	1997-10-13
10702	ALFKI	1997-10-13	1997-11-24	1997-10-21
10835	ALFKI	1998-01-15	1998-02-12	1998-01-21
10952	ALFKI	1998-03-16	1998-04-27	1998-03-24
10308	ANATR	1996-09-18	1996-10-16	1996-09-24
10625	ANATR	1997-08-08	1997-09-05	1997-08-14
10759	ANATR	1997-11-28	1997-12-26	1997-12-12
10926	ANATR	1998-03-04	1998-04-01	1998-03-11
10365	ANTON	1996-11-27	1996-12-25	1996-12-02
10507	ANTON	1997-04-15	1997-05-13	1997-04-22
10535	ANTON	1997-05-13	1997-06-10	1997-05-21
10573	ANTON	1997-06-19	1997-07-17	1997-06-20
10677	ANTON	1997-09-22	1997-10-20	1997-09-26
10682	ANTON	1997-09-25	1997-10-23	1997-10-01
10856	ANTON	1998-01-28	1998-02-25	1998-02-10
10355	AROUT	1996-11-15	1996-12-13	1996-11-20
10383	AROUT	1996-12-16	1997-01-13	1996-12-18
10453	AROUT	1997-02-21	1997-03-21	1997-02-26

Abbildung 8.4:
Die Abfrage-
ergebnisse

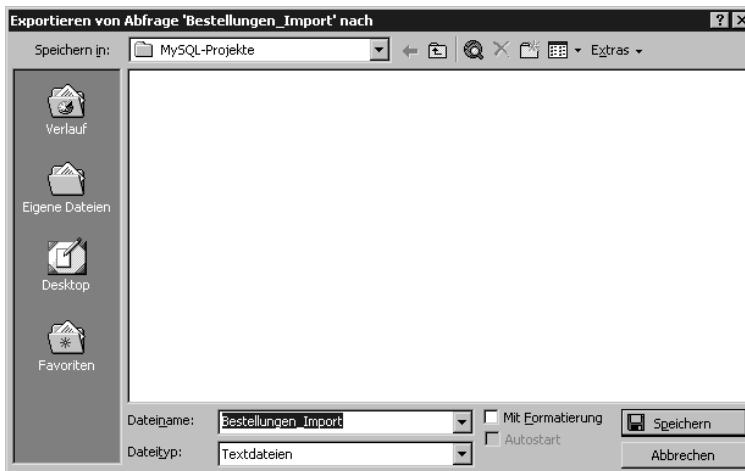


Abbildung 8.5:
Die »Exportieren
einer Anfrage«-
Dialogbox

11. Der Export-Text-Assistent startet automatisch (siehe Abbildung 8.6). Der Assistent hilft Ihnen beim Formatieren Ihrer Datei. Denken Sie daran, dass Sie eine Datei wollen, in der die Felder durch Komma getrennt und die Textfelder von Anführungszeichen umschlossen sind. Klicken Sie auf *Weitere* und geben Sie bei der *Datumsreihenfolge* das richtige Format an (JMT). Geben Sie als *Datumstrennzeichen* den Bindestrich (-) an und als *Dezimalzeichen* den Punkt (vergessen Sie dieses, erhalten Sie später beim Export einen Fehler, da beim Dezimalzeichen als Standard das Komma verwendet wird, dieses aber in Ihrer Exportdatei als Trennzeichen fungieren soll). Vergewissern Sie sich, dass die Option »Mit Trennzeichen« angeklickt wurde, und klicken Sie anschließend auf *Weiter*.

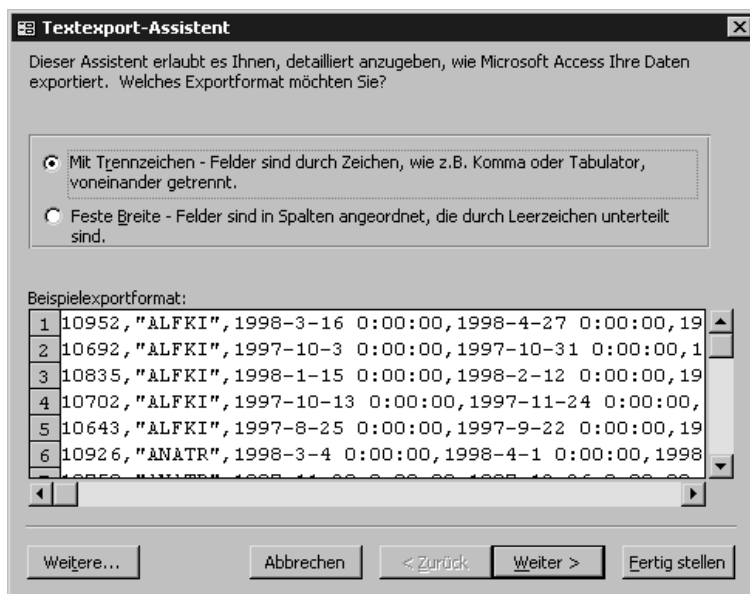


Abbildung 8.6:
Der Export-
Assistent

12. Die nächste Abbildung überlässt Ihnen die Wahl, welches Abgrenzungszeichen Sie benutzen wollen – standardgemäß ist dies das Semikolon. Benutzen Sie hier das Komma. Ebenfalls wird der Text (Dateninhalt) standardmäßig durch Anführungszeichen eingeschlossen (Textbegrenzer). Vergewissern Sie sich, dass Sie diese Optionen ausgewählt haben, und klicken Sie auf *Weiter*.
13. Die letzte Abbildung (Abbildung 8.8) des Assistenten gibt an, wo sich die Datei befindet. Geben Sie den Pfad an, in dem Sie diese Datei speichern möchten, und klicken Sie dann auf *Finish*. Falls nicht unvorhergesehene Probleme aufgetreten sind, sollte die Datei in dem von Ihnen angegebenen Verzeichnis erstellt worden sein.
14. Sie müssten jetzt eigentlich eine durch Komma abgegrenzte Textdatei mit den Daten Ihrer Anfrage vor sich haben. Um dies noch einmal zu überprüfen, gehen Sie in das Verzeichnis, in dem Sie die Datei gespeichert haben. Wenn Sie den Dateinamen doppelklicken, sollte sich die Datei öffnen. Sie sollten jetzt Abbildung 8.9 vor sich haben.

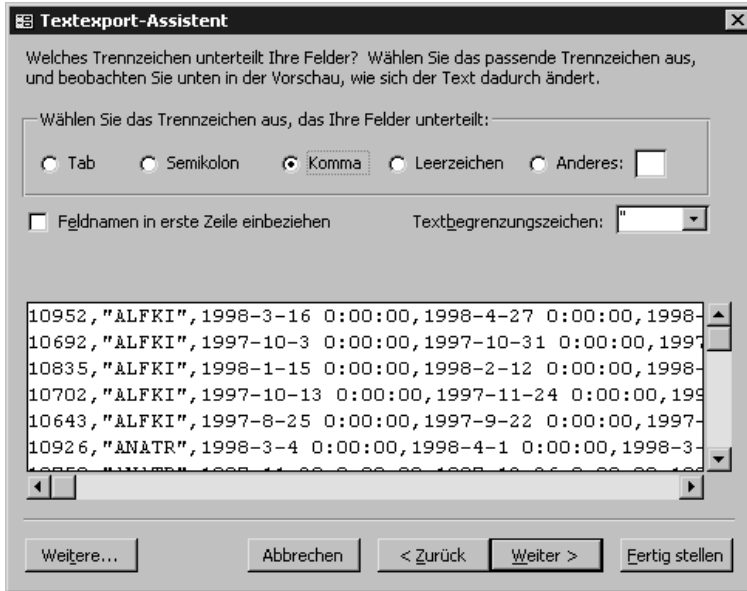


Abbildung 8.7:
Wählen Sie Ihre
Textbegrenzer
und Abgren-
zungszeichen



Abbildung 8.8:
Auswahl des
Pfades und des
Dateinamens

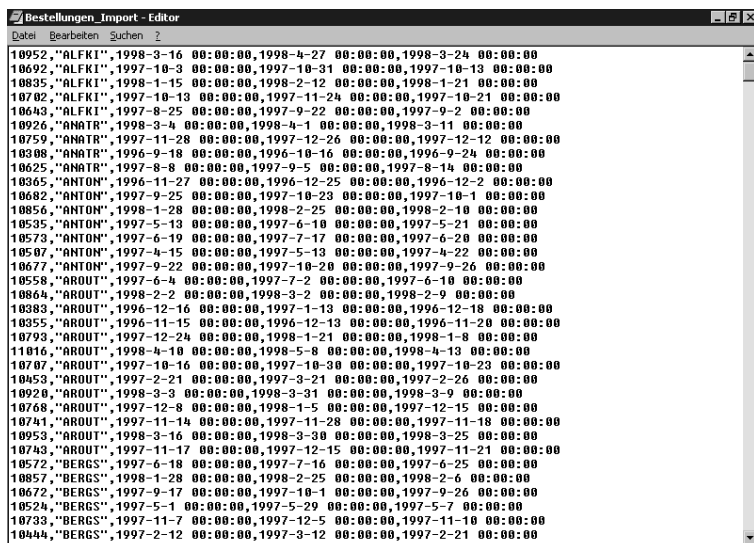


Abbildung 8.9:
Die fertige
Import-Datei

Nun, da Sie Ihren Datenimport beendet haben, ist die Schlacht schon halb gewonnen (oder hat erst begonnen, je nachdem, wie Sie es sehen). Der zweite Teil besteht daraus, Daten in die MySQL-Datenbank zu importieren. Dies ist ein ziemlich einfacher Vorgang. Der erste Schritt besteht darin, Zugang zum MySQL Server zu erhalten. Dabei können Sie aus zwei Möglichkeiten auswählen: Sie können entweder die grafische Benutzeroberfläche (GUI) oder aber Telnet benutzen. Nun werden Sie Telnet benutzen, um mehr Übung zu bekommen. Da sich Ihre Importdatei auf Ihrem Rechner und nicht auf Ihrem MySQL-Server (ich gehe davon aus, dass Sie zwei Rechner verwenden) befindet, werden Sie die Datei mit Hilfe der `LOAD DATA LOCAL INFILE`-Anweisung importieren. Falls Sie nur einen Rechner haben, können Sie diese Übung trotzdem machen. Sie werden einfach kein Telnet benutzen und Sie müssen auch nicht die Option `LOCAL` in Ihrer `LOAD DATA`-Anweisung verwenden. Alles andere ist identisch.

1. Starten Sie eine Telnet-Sitzung. Dies geht am einfachsten von einem DOS-Prompt oder von der Kommandozeile aus. Klicken Sie auf *Start*, anschließend auf *Run* und geben Sie danach Folgendes auf Kommandozeilenebene ein:

```
telnet 10.1.1.50 (use the name or IP of the machine that is running MySQL)
```

Dies sollte eine Telnet Sitzung starten, die Sie mit Ihrem MySQL-Rechner verbindet (siehe Abbildung 8.10).

2. Geben Sie Ihren Unix-/Linux-Benutzernamen und Ihr Passwort ein. Wenn Sie an eine Kommandozeile gelangen, wechseln Sie in das `mysql`-Verzeichnis und starten den MySQL-Monitor.



Abbildung 8.10:
Die Telnet-
Sitzung

```
cd /usr/local/mysql  
bin/mysql -p Meet_A_Geek
```

Dieser Befehl bringt Sie in die Meet_A_Geek-Datenbank, wenn Sie das root-Passwort korrekt eingegeben haben.

3. Nachdem der MySQL-Monitor gestartet wurde, geben Sie folgenden Befehl ein:

```
DESCRIBE Orders;
```

Haben Sie die Meet_A_Geek-Datenbank schon erstellt, sollte der Output demjenigen in Abbildung 8.11 gleichen.

4. Im nächsten Schritt importieren Sie die Daten in die Datenbank. Geben Sie folgende Anweisung auf Kommandozeilenebene ein:

```
LOAD DATA LOCAL INFILE "E:\\Bestellungen_Import.txt"  
INTO TABLE Orders  
FIELDS TERMINATED BY ','  
ENCLOSED BY '''  
(Order_ID, Customer_ID, Order_Date, Required_Date, Ship_Date);
```

Damit sollte der Datenimport erfolgreich abgeschlossen sein.

```

Telnet - 10.1.1.50
Connect Edit Terminal Help
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 99 to server version: 3.22.23b

Type 'help' for help.

mysql> DESCRIBE Orders;
+-----+-----+-----+-----+-----+-----+
| Field          | Type          | Null | Key | Default | Extra          |
+-----+-----+-----+-----+-----+-----+
| Order_ID       | int(11)       |      | PRI | 0        | auto_increment |
| Order_Date     | datetime      | YES  |     | NULL     |                |
| Required_Date  | datetime      | YES  |     | NULL     |                |
| Quantity       | int(11)       | YES  |     | NULL     |                |
| Item           | int(11)       | YES  |     | NULL     |                |
| Customer_ID    | int(11)       | YES  |     | NULL     |                |
| Custom_Message | varchar(255)  | YES  |     | NULL     |                |
| Amount_Due     | float(4,2)    | YES  |     | NULL     |                |
| Ship_Date      | datetime      | YES  |     | NULL     |                |
+-----+-----+-----+-----+-----+-----+
9 rows in set (0.00 sec)

mysql>

```

Abbildung 8.11:
Ausgabe der
DESCRIBE
Orders-
Anweisung

Datenbank-Import-/Exportwerkzeuge

Wie Sie sehen, besitzt MySQL zahlreiche integrierte Werkzeuge, um Ihnen bei der Datenübertragung von einer Datenbank auf die andere behilflich zu sein. MySQL ist in hohem Maße flexibel. Obwohl es bestimmte Features, die andere Datenbanken besitzen, wie zum Beispiel FOREIGN KEYS, nicht unterstützt, wird es keinen Fehler generieren, wenn es diesen beim Datenimport begegnet. Da bei der Entwicklung darauf geachtet wurde, dass MySQL bei der Datenübertragung sehr flexibel ist, mussten diese Unregelmäßigkeiten in Kauf genommen werden.

Da MySQL aus dem Open Source-Bereich stammt, haben viele Personen Programme und andere nützliche Werkzeuge beigesteuert, die die Arbeit mit MySQL erleichtern. Darunter befinden sich GUIs, Import Assistenten und API Wrappers. Diese Werkzeuge sind ebenso wie die, die Sie erstellen können, von großem Vorteil für jeden MySQL-Datenbank-Administrator.

Die MySQL-Website und die dem Buch beigefügte CD-ROM beinhalten ebenfalls einige nützliche Werkzeuge, die Sie ausprobieren sollten. Es gibt darauf ein sehr nützliches Access-Exportwerkzeug namens `exportsql.txt`, welches das Schema und die Daten aus der Datenbank in eine Textdatei exportiert. Das Gegenstück, `importsql`, bewerkstelligt das Gleiche nur in umgekehrter Richtung, indem es eine MySQL-Datenbank in Access konvertiert.

Methoden des Datenexports

Sie können erkennen, dass MySQL ein reichhaltiges Angebot an Importwerkzeugen besitzt. Aber diese stellen lediglich eine Hälfte der Datenübertragung dar. Die andere Hälfte besteht daraus, Daten aus MySQL zu exportieren. Es gibt zahlreiche Gründe, Daten zu exportieren.

Ein wichtiger Grund ist das Sichern Ihrer Datenbank (Backup). Daten sind sehr wertvoll und es zahlt sich aus, in sie zu investieren. Regelmäßige Datensicherungen können dazu beitragen, den Verlust wertvoller Daten zu verhindern. An Tag 19, »MySQL administrieren«, wird der Backup-Vorgang ausführlich behandelt. Für den Augenblick reicht es zu wissen, dass Datensicherung ein Grund für den Datenexport ist.

Es kann auch sein, dass Sie Ihre Daten exportieren möchten, um sie mit anderen zu teilen. In der ständig wachsenden Welt der Informationstechnologie wird das Teilen von Daten immer alltäglicher. Macmillan USA zum Beispiel unterhält eine riesige Datenbank, welche die geplanten Neuerscheinungen enthält. Diese Datenbank wird unter den Buchläden verteilt, damit diese wissen, welche Bücher bald erscheinen. Auch Krankenhäuser gehen immer mehr zu papierlosen Krankenberichten über, die jeder Arzt einsehen kann. Dieser Trend breitet sich zunehmend aus.

Es gibt viele unterschiedliche Methoden, Daten zu exportieren. Sie ähneln alle dem Datenimport – da letztlich alles nur eine Frage der Perspektive ist. Daten, die von einer Datenbank exportiert werden, werden gleichzeitig von anderer Seite als importierte Daten betrachtet. Anstatt zu erörtern, auf welche Art und Weise andere Datenbanken ihre Daten exportieren, werden Sie lernen, wie MySQL es bewerkstelligt.

Das Arbeiten mit mysqldump

Das `mysqldump`-Utility ist seinem Gegenstück, `mysqlimport`, sehr ähnlich. Sie teilen einige der allgemeinen Optionen, wobei `mysqldump` etwas mehr leisten kann. Es lagert die gesamte Datenbank in einer einzigen Textdatei. Diese Datei enthält sämtliche SQL-Anweisungen, die Sie benötigen, um Ihre Datenbank wiederherzustellen. Sie konvertiert das Schema in die passende DDL-Syntax (`CREATE`-Anweisungen) und erstellt aus allen Daten `INSERT`-Anweisungen. Dieses Utility konstruiert aus Ihren Daten SQL-Anweisungen, mit denen Sie Ihre Datenbank komplett wiederherstellen können. Da sich alle Daten in nur einer Textdatei befinden, kann es alles im Batchmodus und durch die passende SQL-Syntax zurück in MySQL importieren. Das geht alles unglaublich schnell und einfach und wird Ihnen überhaupt keine Sorgen bereiten.

Wollen Sie etwa den gesamten Inhalt der `Meet_A_Geek`-Datenbank in eine Datei ausgeben (dumpen), benutzen Sie folgenden Befehl:

```
bin/mysqldump -p Meet_A_Geek > MeetAGeek_Dump_File.txt
```

Dieses Utility ermöglicht Ihnen darüber hinaus, eine Tabelle zu bestimmen, die Sie auslagern möchten. Wollen Sie nur den Inhalt der `Orders`-Tabelle in der `Meet_A_Geek`-Datenbank in einer einzigen Datei auslagern, müssen Sie folgenden Befehl benutzen:

```
bin/mysqldump -p Meet_A_Geek Orders >MeetAGeek_Orders.txt
```

Dieses Utility ist so flexibel, dass es über eine `WHERE`-Klausel ermöglicht, nur die Datensätze auszuwählen, die Sie in der Datei gelagert haben wollen. Dazu müssen Sie folgenden Befehl benutzen:

```
bin/mysqldump -p  
-where="Order_ID > 2000"Meet_A_Geek Orders > Special_Dump.txt
```

Das `mysqldump`-Utility bietet eine Vielfalt an Optionen, die alle in der folgenden Tabelle erläutert werden.

Option	Bedeutung
<code>--add-locks</code>	Fügt <code>LOCK TABLES</code> -Anweisungen am Anfang und <code>UNLOCK TABLES</code> -Anweisungen am Ende des Tabellen-Dumps ein. Dieses sorgt dafür, dass Tabellen während des Imports gesperrt werden, und führt somit zu einem Geschwindigkeits-zuwachs während des Wiederherstellungsprozesses.
<code>--add-drop-table</code>	Fügt eine <code>DROP-TABLE</code> -Anweisung vor jeder <code>CREATE-TABLE</code> -Anweisung ein. Wenn schon eine Kopie der Datenbank am Zielort besteht, stellt diese Option sicher, dass keine Duplikate entstehen und dass die Tabellenspezifikation alle vorherigen Spezifikationen ersetzt.
<code>-c, --complete-insert</code>	Diese Option bewirkt, dass alle Spalten einer <code>INSERT</code> -Anweisung benannt werden. Das kann von Vorteil sein, wenn Sie Ihre Daten in eine andere Datenbank exportieren möchten.
<code>--delayed_insert</code>	Diese Option bewirkt, dass die <code>DELAYED</code> -Option in <code>INSERT</code> -Anweisungen verwendet wird.
<code>-e, --extended-insert</code>	Veranlasst <code>mysqldump</code> , eine mehrzeilige <code>INSERT</code> -Syntax zu verwenden. Dieses soll <code>INSERT</code> -Anweisungen kompakter und schneller machen.
<code>-q, --quick</code>	Gibt das Ergebnis des Dumps direkt auf der Standardausgabe aus. Sie brauchen diese Option unter Umständen, wenn Sie knappe Systemressourcen haben.
<code>--opt</code>	Diese Option umfasst die Optionen <code>--add-locks</code> , <code>--add-drop-tables</code> , <code>--extended-insert</code> und <code>--quick</code> . Dieses ist ein bequemer Weg, die schnellstmögliche Dump- und Importdatei zu erstellen.

Tabelle 8.2: `mysqldump`-Optionen

Option	Bedeutung
-f, --force	Diese Option veranlasst das Programm auch dann weiterzumachen, wenn ein Fehler auftritt.
-F, --flush-logs	Flusht die MySQL-Server-Logdateien, bevor die Dump-Erstellung gestartet wird.
--full	Diese Option bewirkt, dass zusätzliche Informationen bei den CREATE TABLE-Anweisungen hinzugefügt werden. Diese wurde an Tag 4 näher behandelt.
-h= <i>Hostname</i> , --host= <i>Hostname</i>	Bewirkt, dass der Dump auf <i>Hostname</i> statt auf localhost (welches die Standardeinstellung ist) erstellt wird.
-l, --lock-tables	Veranlasst, dass die Tabellen vor dem Dump gesperrt werden. Wenn komplexere Aktualisierungen in Ihrem System vorkommen, stellt diese Option sicher, dass Sie einen vollständigen Dump erhalten, der nicht nur teilweise aktuell ist.
-t, --no-create-info	Mit dieser Option werden nur die Daten gesichert. CREATE TABLE-Anweisungen werden ausgelassen.
-d, --no-data	Lässt die eigentlichen Daten weg und sichert nur die Tabellendefinitionen. Diese Option ist nützlich, wenn Sie nur das Design Ihrer Datenbank sichern möchten, jedoch nicht den Inhalt.
--tab= <i>Pfad</i> , -T= <i>Pfad</i>	Speichert den Dump in Dateien Ihres Dateisystems, die mit <i>Pfad</i> spezifiziert sind. mysqldump erstellt für jede Tabelle eine Datei namens <i>Tabellenname.sql</i> , die die CREATE TABLE-Anweisung, und eine Datei namens <i>Tabellenname.txt</i> , die den Inhalt der Tabelle enthält. Die Daten werden in einem Format gespeichert, welches Tabs als Trennzeichen verwendet (wenn --tab angegeben wurde). Andernfalls werden die Daten in einem Format gespeichert, das mit der --fields-terminated-by-Option angegeben wird (bedenken Sie, dass dieses nur funktioniert, wenn mysqldump auf demselben Rechner wie der mysql-Daemon läuft).
--fields-terminated-by=' <i>Trennzeichen</i> '	Gibt das <i>Trennzeichen</i> an, mit dem die Felder voneinander abgetrennt werden. Wird kein <i>Trennzeichen</i> angegeben, wird als Standardwert das Tabzeichen verwendet (\t = Tab). Wird bei der -T-Option verwendet.
--fields-enclosed-by=' <i>Trennzeichen</i> '	Gibt das <i>Trennzeichen</i> an das benutzt wird, um jedes Feld einzuschließen. Wird bei der -T-Option verwendet.
--fields-optimally-enclosed-by=' <i>Trennzeichen</i> '	Gibt das <i>Trennzeichen</i> an das benutzt wird, um CHAR- und VARCHAR-Felder einzuschließen. Wird bei der -T-Option verwendet.

Tabelle 8.2: mysqldump-Optionen (Forts.)

Option	Bedeutung
--fields-escaped-by='Escape-Zeichen'	Gibt das <i>Escape-Zeichen</i> an, welches vor spezielle Zeichen gesetzt wird. Standard ist \ (führt zu einem Backslash). Wird bei der -T-Option verwendet.
--lines-terminated-by='Trennzeichen'	Gibt das <i>Zeilen-Trennzeichen</i> an. Standard ist \n (newline). Vergleichen Sie hierzu auch Tag 8.
-u <i>Benutzername</i> , --user= <i>Benutzername</i>	Ermöglicht Ihnen, einen <i>MySQL-Benutzernamen</i> anzugeben, um mit dem Server in Verbindung zu treten. Andernfalls wird Ihr Unix-Login-Name benutzt.
-p[<i>Passwort</i>], --password[= <i>Passwort</i>]	Übergibt Ihr <i>Passwort</i> bei der Verbindungsaufnahme mit dem MySQL-Server. Beachten Sie, dass Sie kein Leerzeichen zwischen -p und Ihrem <i>Passwort</i> einfügen dürfen. Wenn Sie nur -p oder --password benutzen, fordert MySQL Sie auf, Ihr <i>Passwort</i> an der Konsole einzugeben (verhindert somit also, dass es auf dem Bildschirm sichtbar wird).
-w='Bedingung', --where='Bedingung'	Gibt nur Datensätze in den Dump aus, die der <i>WHERE-Bedingung</i> entsprechen.
-, --help	Zeigt eine Liste aller Optionen des mysqldump-Programms an und beendet es anschließend wieder.

Tabelle 8.2: mysqldump-Optionen (Forts.)

mysqlimport besitzt ebenso wie sein Gegenstück Wahlmöglichkeiten für die Formatierung von Dateien. Sie haben diese bereits kennen gelernt, sie sind aber in der folgenden Tabelle noch einmal aufgelistet.

Option	Bedeutung
--fields-terminated-by=" <i>Trennzeichen</i> "	Gibt das Zeichen an, welches als Trenner nach jedem Feld benutzt wird. Standard ist hier der Tabulator \t.
--fields-enclosed-by=" <i>Trennzeichen</i> "	Spezifiziert das Trennzeichen, welches jedes Feld umschließt.
--fields-optionally-enclosed-by=" <i>Trennzeichen</i> "	Spezifiziert das Trennzeichen, welches CHAR- oder VARCHAR-Felder umschließt.
--fields-escaped-by="Escape-Zeichen"	Gibt das <i>Escape-Zeichen</i> an, welches vor spezielle Zeichen gesetzt wird. Standard ist \ (führt zu einem Backslash).
--lines-terminated-by='Trennzeichen'	Gibt das <i>Zeilen-Trennzeichen</i> an. Standard ist \n (newline).

Tabelle 8.3: mysqlimport-Optionen

Stellen Sie sich folgende Aufgabe vor: Sie müssen eine Datei erstellen, welche die Buchhalter des Unternehmens in einem Spreadsheet benutzen. Die Geschäftsführer möchten alle Bestellungen dieses Jahres einsehen. Als Erbsenzähler sind sie nicht wirklich an DDL interessiert. Sie sollen die mit Kommata aufgeteilte Datei verwenden, da auf diese Weise der Datenimport in Excel sehr einfach ist. Um diese Aufgabe auszuführen, können Sie folgende Anweisung benutzen:

```
bin/mysqldump -p
-where "Order_Date >='2000-01-01'"
-tab = /home/mark -no-create-info
-fields-terminated-by=,
Meet_A_Geek Orders
```

Auf diese Weise lösen Sie die gestellte Aufgabe.

SELECT INTO OUTFILE

Falls Sie das `mysqldump`-Utility nicht gerade vom Hocker reißt, bietet MySQL Ihnen zudem `SELECT INTO OUTFILE`, das Gegenstück zum `LOAD DATA INFILE`-Befehl. Diese beiden Befehle haben vieles gemeinsam. Sie beinhalten im Großen und Ganzen die gleichen Optionen. Der einzige wichtige Unterschied ist, dass der eine Befehl die Daten importiert, wohingegen der andere Daten exportiert.

Um zu erfahren, wie Sie diesen Befehl benutzen, verfahren Sie ebenso wie beim `mysqlimport`-Utility. Führen Sie folgende Schritte aus:

1. Vergewissern Sie sich, dass der `mysqld`-Daemon läuft und Sie sich im `mysql`-Verzeichnis befinden.

```
cd /usr/local/mysql
bin/mysqladmin -ping
```

2. Starten Sie das MySQL-Monitor-Programm.

```
bin/mysql -p Meet_A_Geek
```

Dieser Befehl startet die `Meet_A_Geek`-Datenbank als aktive Datenbank. Natürlich werden Sie aufgefordert, ein Passwort einzugeben.

3. Geben Sie folgenden Befehl auf Kommandozeilenebene ein:

```
SELECT * INTO OUTFILE '/home/mark/Orders.txt'
FIELDS TERMINATED BY = ','
FROM Orders
WHERE Order_Date >= '2000-01-01'
```

Durch Drücken der -Taste wird Ihre Datei erstellt. Diese Anweisung funktioniert wie eine normale `SELECT`-Anweisung mit dem Unterschied, dass die Ausgabe in die von Ihnen benannte Datei und nicht auf den Bildschirm zurückgeführt wird. Dies be-

deutet, dass Sie komplexere Anfragen durchführen können, indem Sie `JOINS` und Mehrfachtabellen benutzen. Dieses Feature kann ebenso als »Berichterstatter« verwendet werden. Sie könnten zum Beispiel einige der in diesem Kapitel vorgestellten Methoden verbinden, um sehr interessante Berichte zu erzeugen. Versuchen Sie es beispielsweise mit dieser Methode:

1. Erstellen Sie im `mysql`-Verzeichnis eine Textdatei namens `Report_G.rpt` und geben Sie folgende Zeilen ein:

```
USE Meet_A_Geek;
INSERT INTO Customers (Customer_ID, Last_Name, First_Name)
VALUES (NULL, "Kinnard", "Vicky");
INSERT INTO Customers (Customer_ID, Last_Name, First_Name)
VALUES (NULL, "Kinnard", "Steven");
INSERT INTO Customers (Customer_ID, Last_Name, First_Name)
VALUES (NULL, "Brown", "Sam");
SELECT Last_Name INTO OUTFILE '/home/mark/Report.rpt'
FROM Customers
WHERE Customer_ID > 1;
```

2. Als Nächstes vergewissern Sie sich, dass der `mysqld`-Daemon läuft und Sie sich im `mysql`-Verzeichnis befinden.
3. Geben Sie nun folgenden Befehl ein:

```
bin/mysql < Report_G.rpt
```

4. Überprüfen Sie die Datei, die Sie soeben exportiert haben. Sie sollte alle Nachnamen der Kunden enthalten, die Sie in die `Customers`-Tabelle eingegeben haben.

Wie Sie sehen, können Sie die heute gelernten Import-/Export-Methoden auch verwenden, um Berichte zu erstellen.

Zusammenfassung

Heute haben Sie gelernt, weshalb Datenbanken Daten exportieren und importieren müssen.

MySQL besitzt zahlreiche Hilfsprogramme (Utilities) sowohl für den Datenimport als auch für den Datenexport. Sie haben gelernt, wie Sie das `mysqlimport`-Utility verwenden, um Daten in die verschiedensten Formattypen zu bringen. Darüber hinaus haben Sie erfahren, wie Sie das Gegenstück zum MySQL-Monitor – die `LOAD DATA INFILE`-Anweisung – benutzen. Sie wissen jetzt, dass diese Anweisungen von den verschiedenen APIs benutzt werden können, um Daten in die Datenbank zu importieren. Zudem haben Sie etwas über die Fähigkeit des `mysql`-Befehls, eine Datei mit dem »<«-Befehl zu verarbeiten, erfahren.

Schließlich haben Sie Ihre neugewonnenen Kenntnisse angewandt, indem Sie eine Access-Tabelle in eine Textdatei konvertiert und diese Datei anschließend in MySQL importiert haben. Sie haben erfahren, wie die Daten manchmal »zurechtgebogen« werden mussten, bevor sie von MySQL akzeptiert wurden.

Außerdem wurden einige der verschiedenen Shareware-Utilities, die den DBAs das Leben leichter machen, erläutert.

Ihnen wurden des Weiteren viele unterschiedliche Methoden, wie MySQL Daten exportiert, vorgestellt. Auch haben Sie den `mysqldump`-Befehl kennen gelernt und wie dieser Ihre Datenbank vom Schema zu den gespeicherten Datensätzen konstruiert. Letztendlich haben Sie etwas über die `SELECT.....INTO OUTFILE`-Anweisung erfahren. Sie wissen nun, wie diese Anweisung benutzt werden kann, um einige äußerst komplexe Berichte, die in verschiedene Einzelheiten formatiert sind, zu erzeugen.

Hoffentlich haben Sie heute gelernt, wie Daten ganz einfach in MySQL importiert und exportiert werden können.

Fragen und Antworten

F Was kann ich mit dem Inhalt einer `mysqldump`-Datei machen?

A *Diese Datei ist ein genaues Abbild Ihrer Datenbank in SQL-Format. Mit diesen Daten können Sie eine Menge machen. Sie können zum Beispiel Ihre Datenbank auf dem Microsoft SQL Server oder auf Sybase wieder erstellen, indem Sie den Inhalt der Datei einfach »auseinander schneiden« und wieder »zusammenkleben«. Auch können Sie Ihre Datenbank wieder herstellen, indem Sie die Datenbankdump-Datei sowie den Batchmodus des `mysql`-Programms benutzen. Um Ihre Datenbank beispielsweise an Hand des Outputs eines `mysqldump`-Befehls neu zu erstellen, müssen Sie folgende Syntax anwenden:*

```
bin/mysql < Dateiname_der_Dump.Datei
```

Damit wäre Ihre Datenbank wieder komplett hergestellt.

F Ist der Datenexport auf Textformat begrenzt?

A *Denken Sie beim Datenexport nicht an Grenzen, sondern an unbegrenzte Möglichkeiten. Die Entwickler von TcX sind der Auffassung, dass das Parsen der Daten in verschiedene Formate nicht auf dem Server-Level verrichtet werden soll und das Formatieren der Daten Aufgabe des Programmierers sowie der Applikation sein sollte. Sie haben ein Werkzeug zur Verfügung gestellt bekommen, das den Datenexport erleichtert. Sobald Sie die Daten erhalten, liegt es an Ihnen, was Sie damit machen wollen. Das Gleiche gilt für den*

Datenimport. Sie schreiben das Front-End, welches die Daten erhält – Sie müssen lediglich eine Textdatei für MySQL herstellen. Das ist ziemlich einfach.

Übungen

1. Benutzen Sie den `mysqlimport`-Befehl samt den passenden Optionen, um folgende Aufgabe zu lösen.

Sie müssen einen Spreadsheet aus der Versandabteilung importieren. Die Versandabteilung stellt Ihnen dafür einen Worksheet zur Verfügung, der alle ausgegangenen Bestellungen beinhaltet. Die Felder sind durch Slashes (/) voneinander getrennt und die Daten mit einfachen Anführungszeichen umschlossen. Alle Daten auf dem Sheet sind neue Daten, alte Daten wurden nicht aktualisiert. Importieren Sie die Datei.

2. Wenden Sie den `mysqldump`-Befehl im folgenden Szenario richtig an:

Der Chef möchte den Buchhaltern einen auf der `Orders`-Tabelle basierenden Bericht übergeben. Die Buchhalter benötigen die Anzahl und den Preis jedes bestellten Artikels, um den Umsatz zu berechnen. Dafür brauchen Sie nicht die DDL, jedoch eine durch Komma abgegrenzte Datei. Erstellen Sie die Datei.



Datenbank- abfragen

Woche
2

Die Struktur einer Datenbank zu definieren, stellt lediglich den ersten Teil eines RDBMS dar. Die eigentliche Leistungskraft und Nützlichkeit einer Datenbank beruhen auf der Art ihrer Handhabung, wozu MySQL die *Structured Query Language* (SQL) einsetzt.

Heute lernen Sie Folgendes:

- ▶ was Structured Query Language (SQL) ist und wie ihre Entstehungsgeschichte aussieht
- ▶ SQL-Grundlagen – SELECT, UPDATE, INSERT, DELETE
- ▶ wie Sie Joins benutzen
- ▶ Aggregat-Funktionen (zusammenfassende Funktionen)

Was ist SQL und was sind seine Wurzeln?

Über SQL wurden jede Menge Bücher geschrieben (z.B. *SQL in 21 Tagen*, erschienen bei Markt+Technik, ist für Einsteiger besonders geeignet). Das Ziel dieses Buches ist, Sie über diese Sprache und ihre Syntax ausreichend in Kenntnis zu setzen, um die Vorgänge innerhalb einer Datenbank bewerkstelligen zu können. Da die SQL-Programmiersprache sehr einfach aufgebaut ist, kann man sie sich auch sehr leicht aneignen. Die Schwierigkeit besteht darin, sie produktiv einzusetzen, was allerdings nur durch Praxis erlernt werden kann.

Die Structured Query Language oder SQL wurde Mitte der 70er von IBM für die logische Handhabung von Daten in einer relationalen Datenbank entwickelt, was bis dato nicht möglich war. Diese Sprache sollte einfach zu benutzen sein und in ihrer Syntax stark dem Englischen ähneln. IBM stellte sich Geschäftsleute vor, die diese Sprache benutzen sollten, um mit einer Datenbank zu »reden« und die gewünschte Information zu erhalten. Die in SQL gesetzten Hoffnungen erfüllten sich vorerst jedoch nicht. Erst als Oracle diese Sprache als Basis für seinen Datenbankzugang übernahm, wurde SQL auch wirklich benutzt. Seitdem gilt SQL als Standard, um in einer relationalen Datenbank auf Daten zuzugreifen.

Wie schon erwähnt, ist SQL eine Sprache, die die Handhabung von relationalen Datenbanken ermöglicht und deren Syntax stark dem Englischen ähnelt. Wollen Sie beispielsweise alle in Ihrer Datenbank befindlichen Namen der im Januar geborenen Personen in Erfahrung bringen, so könnte Ihre SQL-Anweisung folgendermaßen aussehen:

```
SELECT First_Name, Last_Name FROM Customers WHERE DOB= "January"
```

Diese Sprache ist eher ein Kommunikationsmittel als eine Programmiersprache. Die meisten Programmiersprachen werden kompiliert und anschließend ausgeführt. SQL wird aber im Allgemeinen nicht kompiliert. Es gibt einige Formen von SQL wie zum Beispiel T-SQL, die zwar kompiliert werden, aber hierbei handelt es sich um eine andere Datenbank und eine andere Sprache. SQL ist die Sprache der Datenbanken ebenso wie Englisch Weltsprache ist. Wenn Sie mit jemandem kommunizieren und eine Frage stellen, so bekommen Sie in der Regel eine Antwort. SQL funktioniert ungefähr genauso.

Um ein »Gespräch« mit Ihrer Datenbank zu führen, müssen Sie zuerst einen Kommunikationskanal öffnen. Hier bietet MySQL Ihnen Unterstützung in Form des MySQL-Monitors an. Wenn Sie von Anfang an bei der Sache waren, sollten Sie mit diesem bereits vertraut sein. Die Befehle, die Sie benutzt haben, um in Ihrer Musterdatenbank Tabellen und Spalten zu handhaben und zu erstellen, waren auch schon SQL-Anweisungen. Sie haben SQL also bereits in Aktion erlebt.

SQL ist eine sehr konsequente Sprache. Sobald Sie eine Frage gestellt oder einen Befehl erteilt haben, gibt es kein Zurück mehr. Der Befehl wird unverzüglich und ohne Warnung ausgeführt. Gehen Sie daher sehr vorsichtig mit Ihren Befehlen um, besonders auf einem Produktionssystem. Ich kann Ihnen jetzt nicht unzählige Geschichten über Daten, die wegen Nachlässigkeit bei einer SQL-Anweisung verloren gegangen sind, erzählen. Auch ich habe diesen Fehler begangen, war verantwortlich für das Löschen von ungefähr 20.000 Produktionsdatensätzen auf Grund einer schlampigen DELETE-Anweisung. Gehen Sie also sehr achtsam mit SQL-Anweisungen um, die Sie auf einem Produktionssystem ausführen.

Die Komponenten von SQL

Es besteht die Möglichkeit, SQL in Subkomponenten aufzugliedern. Da die Sprache ein sehr breites Spektrum abdeckt, wurde sie zerlegt, damit Benutzer sie besser verstehen können. SQL kann in zwei Teile aufgegliedert werden – im ersten werden die Datenbank-Objekte erstellt, im zweiten gehandhabt. Der Teil, der die Objekte erstellt, wird oft DCL oder *data creation language* genannt. Es handelt sich hierbei immer noch um SQL, die Anweisungen dienen allerdings nur dem Erstellen und nicht der Handhabung. DCL haben Sie bereits beim Erstellen ihrer Tabellen und Spalten kennen gelernt und benutzt. Der Teil, der die Daten handhabt, wird DML oder *data manipulation language* genannt.

Wie schon gesagt ist SQL eine Sprache, die der Kommunikation mit Datenbanken dient. Da SQL eine Sprache ist, unterliegt sie bestimmten Regeln und einer Grammatik. Darüber hinaus existiert eine Vokabelliste der in SQL benutzten Wörter. Da sie immer eine besondere Bedeutung haben, können Sie diese als *reservierte* Wörter betrachten. SQL kann folgende Verben verstehen: CREATE, DROP, ALTER, DELETE, INSERT,

UPDATE und SELECT, ebenso folgende Wörter: SET, FROM, INTO, WHERE, JOIN, LEFT JOIN, CROSS JOIN, RIGHT JOIN, FULL JOIN, INNER JOIN, ON, ORDER BY und GROUP BY. Eine vollständige Liste reservierter Wörter können Sie der folgenden Tabelle entnehmen:

CREATE	DROP
ALTER	DELETE
INSERT	UPDATE
SELECT	SET
FROM	INTO
ON	WHERE
ORDER BY	GROUP BY
JOIN	LEFT JOIN
CROSS JOIN	FULL JOIN
RIGHT JOIN	AND
LIMIT	OR
LIKE	AS

Tabelle 9.1: Reservierte Wörter in MySQL

Grundlegende SQL-Syntax

Beim »Gespräch« mit Ihrer Datenbank setzen Sie eins oder mehr dieser besonderen Wörter sowie die Information, die Sie bearbeiten wollen, in einer Anweisung, die an die Datenbank gegeben wird, ein. Stellen Sie sich vor, Sie haben in Ihrer Datenbank eine Tabelle namens `Customers`, die aus folgenden Spalten besteht: `First_Name`, `Last_Name`, `Address` und `DOB` (Geburtsmonat). Wollen Sie in Ihrer Datenbank alle Kunden, die im Januar geboren sind, einsehen, sieht Ihre Anweisung folgendermaßen aus:

```
SELECT First_Name, Last_Name FROM Customers WHERE DOB= "January"
```

Das Verb steht in einer SQL-Anweisung immer an erster Stelle. Es übermittelt der Datenbank, was Sie möchten. Im Laufe dieses Kapitels werden Sie erfahren, was ein jedes Verb bewirkt. Im letzten Beispiel haben Sie das Verb `SELECT` benutzt. Dies übermittelt der Datenbank, welche Informationen Sie auswählen möchten.

Die nach dem Verb stehenden Wörter bilden eine Liste von Parametern, die Sie einsehen möchten. Diese Parameter stellen die Namen der Spalten dar, die die von Ihnen erwünschten Informationen enthalten. Stellen Sie sich vor, Sie benötigen für Ihren Chef einen Bericht, in dem alle Kunden, die im Januar Geburtstag haben, aufgeführt sind, damit die Firma Geburtstagskarten verschicken kann. Sie können diesen Bericht

mit Hilfe der oben genannten Anweisung generieren. Stellen Sie sich die Parameterliste als Spaltennamen Ihres Berichts vor. Diese Liste enthält die Informationen, die Sie für Ihren Bericht benötigen.

Nach der Liste mit Spaltennamen steht das reservierte Wort `FROM`, das der Datenbank übermittelt, wo sie die Information suchen muss.

Dem folgt der Name der Tabelle, in der sich die von Ihnen erwünschten Informationen befinden. Es können durchaus auch mehrere Tabellen angegeben werden, in diesem Beispiel möchten Sie allerdings nur die Daten der `Customers`-Tabelle benutzen.

Der nächste Abschnitt der SQL-Anweisung ist die `WHERE`-Klausel, die die von Ihnen erwünschte Information eingrenzt. Im letzten Beispiel drückt die `WHERE`-Klausel aus, dass Sie lediglich die Datensätze von Kunden, deren Geburtstag im Januar liegt, möchten. In Ihrer `WHERE`-Klausel können durchaus mehrere Ausdrücke stehen. So können Sie beispielsweise Folgendes hinzufügen, um die Information weiter einzuschränken:

```
SELECT First_Name, Last_Name FROM Customers  
WHERE DOB = "January" AND Last_Name LIKE "M%"
```

Diese Anweisung zeigt alle Vor- und Nachnamen der Kunden, die im Januar geboren sind und deren Nachname mit M beginnt.

Noch ein Wort zur `WHERE`-Klausel. Wenn Sie diese eingeben, vergewissern Sie sich, dass Ihre Anweisung exakt und Ihre Bedingung logisch ist. Ist dies nicht der Fall, kann es sein, dass Sie nicht die gewünschten Ergebnisse erhalten und später bei der Interaktion mit Ihrer Datenbank auf gewaltige Probleme stoßen. Beim Gebrauch der `WHERE`-Klausel entstehen die meisten Fehler; insofern sollten Sie dabei gut aufpassen.

Die einzigen obligatorischen Felder sind die Verb-, Spaltenüberschrifts- und `FROM`-Felder. Die `WHERE`-Klausel ist fakultativ. Wollen Sie alle Datensätze Ihrer `Customers`-Tabelle einsehen, können Sie folgende Anweisung geben:

```
SELECT * FROM Customers
```

Durch diese Anweisung erscheinen alle Spalten und Daten. Einfache Anweisungen wie diese zeigen Ihnen in kürzester Zeit an, welche Informationen Ihre Tabelle enthält. Beachten Sie, dass im Falle einer großen Tabelle solche Anweisungen sehr viel Prozessorzeit in Anspruch nehmen können. Stellen Sie sich die Ausführung dieser Anfrage bei einer Tabelle, die eine halbe Million Datensätze beinhaltet, vor. Um eine Auswahl der in einer Tabelle enthaltenen Daten zu treffen, benutzen Sie die `LIMIT`-Anweisung. Diese Anweisung limitiert die Anzahl der Datenzeilen, die bei einer Anfrage zurückgegeben werden. Falls Sie Daten aus der `Customers`-Tabelle auswählen möchten, können Sie bedenkenlos folgende Anfrage ausführen:

```
SELECT * FROM Customers LIMIT 10
```

Diese Anweisung gibt lediglich die ersten zehn Datensätze zurück.

Ein weiterer wichtiger Aspekt ist, dass Zeilen in der Reihenfolge angezeigt werden, in der sie in der Muster-Datenbank eingegeben wurden. Wenn also Mia Claymore vor Ana Akin eingegeben wurde, würde Mia zuerst angezeigt werden. Dies kann zu einigen Problemen führen; insbesondere wenn Sie einen Bericht in alphabetischer Reihenfolge erstellen müssen. Glücklicherweise besitzt SQL Funktionen, die genau das, bewerkstelligen – Aggregat-Funktionen, die aus der `ORDER BY`- und `GROUP BY`-Klausel bestehen. Diese werden im Laufe dieses Kapitels noch ausführlich erläutert.

Wie Sie sehen, kann SQL ein sehr leistungsstarkes und nützliches Werkzeug sein, das Ihnen benötigte Daten zugänglich macht. Sie fragen sich vielleicht, was Ihnen das alles im Alltag bringen soll. Die Antwort ist denkbar einfach: Sie betten Ihre SQL-Anweisungen in einer Programmiersprache ein, um die von Ihnen gewünschten Daten zu erhalten. Dies wird ausführlich an den Tagen 12, 13 und 14 und 15 behandelt. Heute lernen Sie lediglich die Grundlagen von SQL und an den folgenden Tagen erfahren Sie, wie Daten zu bearbeiten sind. Sie können ganz einfach Ihren Tabelleninhalt einsehen, unnötige Daten löschen, und Sie können Ihrer Datenbank sogar neue Daten hinzufügen.

Datenbearbeitung mit SQL

Der Themenkomplex Datenverarbeitung kann in folgende vier Befehle gegliedert werden: `SELECT`, `INSERT`, `UPDATE` und `DELETE`. Diese Befehle decken alles ab, was Sie mit Ihren Daten machen können. Jeder Befehl hat seine Feinheiten und wird im Laufe dieses Kapitels erklärt. An dieser Stelle möchte ich einige Dinge, die allen Anweisungen gemein sind, ansprechen.

Nach jedem Verb steht eine Liste von Spaltennamen. Sie können jede Spalte, die Sie einsehen möchten, benennen oder Sie können alle durch das Wildcard-Zeichen aufrufen. Bei MySQL ebenso wie bei allen anderen RDBMS ist der Asterisk (*) das Wildcard-Zeichen. Wollen Sie alle Spalten der `Customers`-Tabelle einsehen, können Sie folgende Anweisung benutzen:

```
SELECT * FROM Customers
```

Diese Anweisung zeigt alle Spalten an.

Es gibt einige Regeln, die für alle Anweisungen gelten. Es handelt sich um allgemeine Syntaxregeln, die befolgt werden müssen, damit die Datenbank versteht, was Sie ihr sagen wollen.

Im Folgenden sind einige der zu befolgenden Regeln angeführt:

- Benutzen Sie in der `FROM`-Klausel mehr als eine Tabelle, muss allen Feldern gleichen Namens der Tabellename vorangestellt werden, ganz gleich wo diese benutzt werden. Beispielsweise besitzen die `Orders`- und die `Customers`-Tabelle beide

ein `Customer_ID`-Feld. Wenn in beiden Tabellen Abfragen auszuführen sind, so muss jedes Mal der genaue Tabellenname für das `Customer_ID`-Feld angegeben werden:

```
SELECT Orders.Order_ID FROM Orders, Customers
WHERE Customers.Customer_ID = Orders.Customer_ID
```

- ▶ Mehrfachanweisungen in einer `WHERE`-Klausel müssen entweder durch ein `AND`- oder ein `OR`-Schlüsselwort verbunden werden. Achten Sie auf die Reihenfolge, in der Sie diese Befehle benutzen. Sie können Anweisungen gruppieren, indem Sie Klammern verwenden, um die gewünschte Anweisung zu erhalten.
- ▶ Alle SQL-Anweisungen müssen ein Verb, eine `FROM`- oder `INTO`-Klausel sowie eine Parameter-Liste der Spaltennamen, die entweder eine oder alle Spalten einer Tabelle beinhaltet, haben.

Eine weitere Gemeinsamkeit aller Verben ist die Fähigkeit einer Tabelle, einen Alias zu vergeben. Dies spart Zeit und Platz beim Identifizieren der Tabellen, aus denen Sie Informationen haben möchten. Um Tabellen auf diese Weise »umzubenennen«, benutzen Sie Folgendes:

```
SELECT O.Order_ID FROM Orders AS O, Customers AS C
WHERE C.Customer_ID = O.Customer_ID
```

Wie Sie sehen, ermöglicht Ihnen diese, eine Art Kurzschrift für Tabellen zu benutzen. Sie müssen dabei das Schlüsselwort `AS` benutzen. Das erspart Ihnen, folgende (lange) Anweisung einzutippen.

```
SELECT Orders.Order_ID FROM Orders, Customers
WHERE Customers.Customer_ID = Orders.Customer_ID
```

Dadurch können Sie etwas Zeit sparen und Sie müssen nicht den ganzen Tabellennamen ausschreiben. Wenn Sie in der `FROM`-Klausel mehr als eine Tabelle benutzen, müssen Sie den Tabellennamen angeben, so dass die Datenbank weiß, welches Feld Sie meinen. Wenn Sie das tun müssen, können Sie genauso gut die oben ausgeführte Abkürzung nehmen.

Seien Sie vorsichtig, wenn Sie bei einer `SELECT`-, `UPDATE`- oder `DELETE`-Anweisung mehr als eine Tabelle benutzen. Vergewissern Sie sich, dass Ihre Joins vollständig und kurz sind (Joins werden im Abschnitt »Arbeiten mit Joins« behandelt). Wenn dies nicht der Fall sein sollte, kann es zu einem karthesischen Join kommen, was bedeutet, dass Sie jede Zeile der einen Tabelle mit jeder Zeile der anderen Tabelle verbinden müssen. Dies ist sehr CPU-intensiv. Um dies zu vermeiden, sollten Sie überprüfen, was Sie verbinden und dass die Verbindung vollständig ist. Dadurch verhindern Sie, dass unter Umständen wichtige Daten gelöscht werden und diese von einem Log aus wieder erstellt werden müssen.

SELECT

Die Anweisung `SELECT` ist wahrscheinlich die am häufigsten benutzte SQL-Anweisung. Sie gibt nur Daten aus, die in der Datenbank enthalten sind. MySQL führt diese Art von Abfrage wahrscheinlich schneller aus als jede andere auf dem Markt befindliche Datenbank. Die `SELECT`-Syntax sieht folgendermaßen aus:

```
SELECT Spaltenname FROM Tabellenname WHERE Bedingung
```

Der Spaltenname kann aus einer oder mehreren Spalten, die sich in der Tabelle, die in der `FROM`-Klausel angegeben ist, bestehen. Sie können keinen Spaltennamen aussuchen, den es nicht in der von der `FROM`-Klausel spezifizierten Tabelle gibt. Wenn Sie zum Beispiel das `Last_Name`-Feld aus der `Customers`-Tabelle auswählen möchten, diese aber nicht in der `FROM`-Klausel angegeben haben, so generiert MySQL einen Fehler. Wie schon erwähnt, können Sie diesen Vorgang durch die Vergabe von Aliasen verkürzen. Nur wenn Unklarheit bezüglich der Feldnamen herrscht, müssen Sie den Tabellennamen angeben. Wenn beispielsweise zwei Tabellen ein `Order_ID`-Feld enthalten, müssen Sie entscheiden, welchen Tabellen- und somit auch Feldnamen Sie benutzen möchten. Sie müssen keine Tabellennamen bestimmen, wenn jeder Feldname in allen Tabellen Ihrer Tabellen-Parameterliste nur einmal vorhanden ist.

Die Reihenfolge, in der Sie die Spaltennamen angeben, entspricht derjenigen, in der diese angezeigt werden. Wählen Sie zum Beispiel `First_Name`, `Last_Name`, werden die Daten in dieser Reihenfolge angezeigt. Wählen Sie hingegen `Last_Name`, `First_Name`, so werden die Daten konsequenterweise in umgekehrter Reihenfolge angezeigt.

Die `SELECT`-Anweisung kann zudem eingesetzt werden, um mathematische Gleichungen zu lösen. Die Anweisung

```
SELECT 1 + 1
```

ergibt beispielsweise folgendes Ergebnis:

```
2
```

Die nächste Anweisung ist ebenfalls anwendbar:

```
SELECT concat(First_Name, " ", Last_Name) AS Name FROM Customers
```

Diese Anweisung gibt einer Spalte mit der Bezeichnung `Name` samt den Ergebnissen der Verknüpfungsfunktion zurück. Die `concat()`-Funktion ist eine interne Funktion von MySQL und wird in der morgigen Lektion ausführlicher behandelt. Wenn Sie also ein Ergebnis erhalten möchten, das Name und Adresse einer Person enthält, können Sie folgende SQL-Anweisung benutzen:



Die `concat()`-Funktion verknüpft zwei Zeichenketten zu einer (*String-Konkatenation*). Interne Funktionen von MySQL werden Sie an Tag 10 kennen lernen.


```
SELECT concat(First_Name, " ", Last_Name)
AS Name, Adresse, City, State
FROM Customers;
```

Das Ergebnis sehen Sie in der folgenden Tabelle:

Name	Adresse	City	State
Glen Brazil	133 Foxview Ln.	Hoptown	KY
Roxanne Tor	123 Center Ave.	Glen Burnie	MD

Tabelle 9.2: Ergebnisse der Konkatenation

Beachten Sie, dass die Spalten die Namen tragen, die Sie ihnen in der `AS`-Klausel zugewiesen haben. Diese Technik ist besonders beim Gebrauch interner MySQL-Funktionen äußerst nützlich. Ein weiteres Beispiel stellt die `MAX()`-Funktion dar, die den größten Wert einer numerischen Spalte zurückgibt. Diese kann an Stelle des `AUTO_INCREMENT`-Attributs benutzt werden. Um beispielsweise den aktuellen Maximalwert einer numerischen Spalte zu erhalten, können Sie folgende Anweisung verwenden:

```
SELECT MAX(Customer_ID) as Curent_Num FROM Customers
```

Das Ergebnis lautet:

```
Current_Num
78
```

Bei einer MySQL-`SELECT`-Anweisung können Sie ebenfalls die Reihenfolge, in der MySQL diese Anfrage bearbeiten soll, festlegen. Wenn MySQL gleichzeitig eine Anfrage für eine `INSERT`- und eine `SELECT`-Anweisung erhält, wird `INSERT` vor `SELECT` verarbeitet. Um zu erreichen, dass MySQL die `SELECT`-Anweisung zuerst ausführt, können Sie die Reihenfolge der Anfrage folgendermaßen bestimmen:

```
SELECT HIGH_PRIORITY Customer_ID FROM Customers
```

Das `HIGH_PRIORITY`-Argument sollte nur bei kleinen Anfragen angewandt werden.

Der Subselect ist ein Leistungsmerkmal, das in MySQL noch nicht erhältlich ist. Es ermöglicht Ihnen, eine Anfrage, die auf den Ergebnissen einer anderen Anfrage basiert, zu formulieren. Zum Beispiel:

```
SELECT * FROM Customers
WHERE Customer_ID
IN (SELECT Customer_ID FROM Orders AS O, Shippers As S
WHERE S.Shipper_ID = O.Shipper_ID)
AND S.Shipper_ID = O.Shipper_ID)
```

Das ist zwar nicht das allerbeste Beispiel, aber es macht deutlich, was ein Subselect ist. Es gibt natürlich Workarounds für diese Art von Select. Sie könnten innerhalb einer einzigen Abfrage mehrere Multiple-Joins ausführen. Das ist natürlich nicht die beste Methode, aber solange MySQL noch keine Subselects besitzt, bleibt Ihnen nichts anderes übrig.

Ein weiteres neues Leistungsmerkmal, das in der Version 3.23 erhältlich ist, ist die Fähigkeit, einen SELECT auszuführen und das Ergebnis in eine Tabelle zu schreiben. Besonders beim Batch-Programmieren ist dies ein sehr nützliches Feature. Bei der hier erstellten Tabelle handelt es sich um eine temporäre Tabelle, die gelöscht wird, sobald die momentane Verbindung beendet wird. Um diese zu erstellen, geben Sie folgenden Befehl ein:

```
CREATE TEMPORARY TABLE temp_table SELECT * FROM Customers
```

Dieser Befehl erstellt eine temporäre Tabelle, die eine exakte Kopie der in der Customers-Tabelle enthaltenen Datensätze beinhaltet. Dieses Merkmal ähnelt dem SELECT INTO-Befehl, der von anderen RDBMS der gleichen Kategorie angeboten wird. Ein weiteres Workaround für SELECT INTO ist der INSERT INTO-Befehl, der praktisch das Gleiche bewirkt.

INSERT

Der INSERT INTO-Befehl fügt einer bestehenden Tabelle neue Datenzeilen hinzu. Das Format gleicht im Prinzip dem einer SELECT-Anweisung. In der Tat können Sie die INSERT INTO und SELECT-Syntax kombinieren, um Datenzeilen einer Tabelle in eine andere einzufügen. Der INSERT INTO-Befehl sieht folgendermaßen aus:

```
INSERT INTO Customers VALUES(NULL, "Glen", "Brazil")
```

Der Name der Tabelle, in die Sie den neuen Datensatz einfügen wollen, steht nach dem INTO-Schlüsselwort. Dem VALUES-Schlüsselwort folgt eine Liste mit Werten, die Sie einfügen wollen. Wenn Sie einen Datensatz auf diese Weise einfügen, muss jede Spalte einen Wert enthalten und die Reihenfolge der Daten derjenigen der Spalten entsprechen. Im untenstehenden Beispiel haben Sie eine Tabelle, die durch folgende Spalten definiert ist: Customer_ID, First_Name, Last_Name. Fügen Sie in diese Tabelle eine Datenzeile ein, so müssen diese Daten in der vorgegebenen Reihenfolge – Customer_ID-Wert, Vorname usw. – angeordnet sein. Entsprechen die eingefügten Werte nicht genau der vorgegebenen Spaltenreihenfolge, erscheint ein Fehler. Um das zu vermeiden, ermöglicht SQL die Benennung der einzufügenden Werte. Ihre SQL-INSERT-Anweisung könnte zum Beispiel folgendermaßen aussehen:

```
INSERT INTO Customers (Customer_ID, Last_Name, First_Name)
VALUES (NULL, "Brazil", "Glen")
```

Hierbei spielt die Reihenfolge oder Anzahl der in der Tabelle befindlichen Spalten keine Rolle. Dies ist die meistbenutzte Methode, Datenzeilen einzufügen. Auf diese Weise bestimmen Sie die Reihenfolge sowie die Spalten, die Sie einfügen wollen. Sie müssen nicht die Reihenfolge Ihrer Tabelle, sondern Ihre eigene beachten. Diese Methode ermöglicht Ihnen zudem, Daten hinzuzufügen, die Sie auch wirklich haben wollen – Sie müssen keine Werte hinzufügen, die Ihnen nicht zur Verfügung stehen oder die Sie gar nicht hinzufügen wollen, wie es beispielsweise bei der vorangegangenen Methode der Fall ist.

Ab MySQL Version 3.23.xx können Sie die `INSERT INTO...SELECT`-Anweisung benutzen. Diese Anweisung ermöglicht Ihnen, einer Tabelle Daten hinzuzufügen, die auf den in der `SELECT`-Anweisung angegebenen Kriterien basieren. Sie können zum Beispiel erforderliche Daten aus einer Tabelle herausziehen und diese in eine andere einfügen. Stellen Sie sich vor, Sie müssten aus der `Customers`-Tabelle alle Kunden aus Maryland herausziehen und diese in eine von Spediteuren benutzte Tabelle einfügen, damit diese ihre LKW-Routen planen können. Dazu benötigen Sie eine bestehende `Shipper_Info`-Tabelle mit folgendem Schema:

```
CREATE TABLE Shipper_Info (  
  Shipper_Info_ID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,  
  Customer_Name VARCHAR(50),  
  Address VARCHAR(30),  
  City VARCHAR(20),  
  Zip VARCHAR(15),  
  Truck_NUMBER INT,  
  Driver_ID INT);
```

Daraufhin können Sie eine SQL-Anweisung erlassen, die die Bedürfnisse des Spediteurs erfüllt und folgendermaßen aussieht:

```
INSERT INTO Shipper_Info  
(Customer_Name, Address, City, Zip)  
VALUES  
SELECT CONCAT(First_Name + " " + Last_Name) AS Name, Address, City, Zip FROM  
Customers WHERE State = "MD";
```

Diese fügt für jede Zeile des Ergebnissatzes eine weitere Zeile hinzu. Entsprechen zum Beispiel 100 Datenzeilen diesen Kriterien, so werden der `Shipper_Info`-Tabelle 100 Zeilen hinzugefügt. Wichtig ist zudem, dass Sie Daten nicht in die Tabelle einfügen können, aus der Sie gerade Daten auswählen. Außerdem müssen die Anzahl der Spalten sowie die in den Spalten enthaltenen Daten den Spalten- und Datentypen der Spalten, in die Sie die Daten einfügen, entsprechen. Beispielsweise können Sie keine Datenzeilen einfügen, die nicht zu den Spalten in der Tabelle passen. In diesem Fall würde MySQL einen Fehler generieren.

Eine weitere nützliche Fähigkeit von MySQL besteht darin, eine `INSERT`-Anweisung so lange zu verzögern, bis die Tabelle nicht mehr benutzt wird – was insbesondere bei langen `SELECT`-Anweisungen, die Vorrang vor `INSERT`-Anweisungen haben, von Vorteil ist. Vergessen Sie nicht, dass MySQL jegliche Anweisungen, durch die Datensätze bearbeitet werden sollen, vor `SELECT`-Anweisungen ausführt. So gesehen hat eine `INSERT`-Anweisung also Vorrang vor einer `SELECT`-Anweisung. Um eine `INSERT`-Anweisung zu verzögern, müssen Sie die `INSERT DELAYED`-Syntax, die folgendermaßen aussieht, einsetzen:

```
INSERT DELAYED INTO Customers (Last_Name) VALUES ("Newton")
```

Das Beste daran ist, dass neue Daten erst in die Tabelle eingefügt werden, wenn diese nicht mehr in Verwendung ist. Werden eine lange `SELECT`-Anweisung und mehrere `INSERT DELAYED`-Anweisungen zur gleichen Zeit ausgeführt, werden die Ausführungen der `INSERT`-Anweisungen so lange zurückgestellt, bis die `SELECT`-Anweisung ausgeführt wurde. Der größte Nachteil der `INSERT DELAYED`-Syntax ist, dass beim Herunterfahren des `mysqld`-Daemons alle Anweisungen, die in die Warteschleife gestellt wurden, gelöscht werden, ohne dass sie ausgeführt wurden. Zum Glück aber ist der `mysqld`-Daemon ein sehr stabiler Prozess, der nur selten, wenn überhaupt, von alleine abstürzt. Im Allgemeinen bedarf es dazu eines `kill-9` oder eines Neustarts.

UPDATE

Der `UPDATE`-Befehl ermöglicht Ihnen, die Werte bestehender Daten zu ändern. Dieser Befehl ähnelt sehr stark der `SELECT`-Anweisung, doch im Gegensatz zu dieser können mit `UPDATE` die von Ihnen ausgewählten Werte verändert werden. Die Syntax einer `UPDATE`-Anweisung sieht folgendermaßen aus:

```
UPDATE Tabellennamen SET Spaltenname = Wert WHERE x = y
```

Wie bei allen SQL-Anweisungen, steht das reservierte Verb an erster Stelle. Direkt nach dem Verb steht der Name der zu bearbeitenden Tabelle. Nach dem Tabellennamen wird das reservierte Wort `SET` eingesetzt. Dem `SET`-Befehl folgt eine Liste von Spaltennamen und Werten, die Sie einsetzen wollen. Möchten Sie zum Beispiel die `City`-Spalte in `Dunkirk` umändern, so sieht Ihre `WHERE`-Bedingung folgendermaßen aus:

```
City = "Dunkirk"
```

Nach dem `SET`-Befehl können auch mehrere Werte stehen. Jedes Spalten/Werte-Paar muss durch ein Komma abgetrennt werden. Mit nur einer `UPDATE`-Anweisung können Sie jede Spalte in der Tabelle aktualisieren. Der `WHERE`-Klausel folgt eine Spalten/Werte-Liste. Diese Klausel bestimmt, welche Zeilen aktualisiert werden. Gehen Sie äußerst vorsichtig vor, sonst könnte es zu Problemen kommen. Möchten Sie beispielsweise den Datensatz einer Ihrer Kundinnen aktualisieren, weil sie geheiratet und einen anderen Namen angenommen hat, müssen Sie folgende Anweisung ausführen:

```
UPDATE Customers SET Last_Name = "Smith" WHERE Customer_ID = 12
```

Benutzen Sie in dieser Anweisung keine `WHERE`-Klausel, so werden alle Werte der `Last_Name`-Spalte zu `Smith`. Sie können sich bestimmt vorstellen, was das bedeutet. Vergessen Sie nicht, dass SQL keine Warnungen ausgibt – bevor Sie also etwas verändern, sollten Sie sich vergewissern, dass Sie auch das Richtige verändern.

Es ist üblich, eine `SELECT`-Anweisung auszuführen, bevor die Daten aktualisiert oder gelöscht werden. Dadurch wird sichergestellt, dass nicht die falschen Daten entfernt oder verändert werden. Im letzten Beispiel müssten Sie demnach die folgende `SELECT`-Anweisung vor Ihrer `UPDATE`-Anweisung erteilen:

```
SELECT * FROM Customers WHERE Customer_ID = 12
```

Diese Anweisung gibt nur eine Datenzeile zurück, welche die Informationen enthält, die Sie aktualisieren wollen. Da lediglich eine Datenzeile zurückgegeben wird, können Sie sicher sein, dass beim Ausführen Ihrer `UPDATE`-Anweisung auch nur eine Zeile betroffen ist. Dieser Zwischenschritt kann Ihnen letztendlich eine Menge Kummer ersparen.

MySQL hat der `UPDATE`-Anweisung ein zusätzliches Leistungsmerkmal beigelegt. Wollen Sie eine `UPDATE`-Anweisung ausführen, während kein anderer die Tabelle, die Sie aktualisieren möchten, benutzt, können Sie `LOW_PRIORITY` einsetzen. Daraus folgt, dass die `UPDATE`-Anweisung nur dann ausgeführt wird, wenn die Tabelle nicht benutzt wird. Die Syntax ist die folgende:

```
UPDATE LOW_PRIORITY Tabellenname SET Spaltenname = Wert WHERE x = y
```

MySQL ist auch in der Lage zu erkennen, wenn Sie einen Wert durch einen Wert ersetzen wollen, der bereits in dem Feld enthalten ist. Versuchen Sie also beispielsweise einen Kundendatensatz zu aktualisieren, der ursprünglich den Namen `Steve` in der `First_Name`-Spalte enthielt und den Sie nun durch `Steve` ersetzen möchten, so ist MySQL clever genug, um dies zu erkennen und den Vorgang zu ignorieren. Dies spart CPU-Zeit und macht MySQL insgesamt leistungsfähiger.

Mit MySQL können Sie auch einen Wert aktualisieren, der auf einem bereits enthaltenen Wert basiert. Wollen Sie beispielsweise dem Alter aller Kunden ein Jahr hinzufügen, können Sie folgende Anweisung verwenden:

```
UPDATE Customers SET Age = Age + 1
```

Beachten Sie, dass MySQL-Update-Anweisungen von links nach rechts gelesen werden.

Das `LIMIT`-Schlüsselwort bietet weiteren Schutz beim Aktualisieren der Datenbank. Sie schränkt die Anzahl der durch Ihre `UPDATE`-Anweisung betroffenen Datenzeilen ein. Wollen Sie die Anzahl der im letzten Beispiel aktualisierten Datenzeilen einschränken, so können Sie Folgendes eingeben:

```
UPDATE Customers SET Age = Age + 1 LIMIT 10
```

Diese Anweisung aktualisiert nur die ersten zehn Datenzeilen. Auf diese Weise können Sie Ihre Arbeit überprüfen und sicherstellen, dass Ihre SQL-Anweisung genau das gemacht hat, was Sie auch wollten.

DELETE

Die `DELETE`-Anweisung ähnelt stark der `SELECT`-Anweisung. Der einzige Unterschied besteht darin, dass `DELETE` nicht Datensätze zur Einsicht auswählt, sondern löscht. Auch bei dieser Anweisung sollten Sie vorsichtig sein, um nicht unbeabsichtigt Daten zu löschen, da MySQL Sie nicht vorher warnt. Sie können die vorhin beschriebenen Vorsichtsmaßnahmen anwenden, das heißt, eine `SELECT`-Anweisung durchführen, um sicherzugehen, dass Sie nur die Datensätze löschen, die Sie auch wirklich löschen wollen. Zudem können Sie das `LIMIT`-Schlüsselwort benutzen, um sicherzustellen, dass nur die von Ihnen bestimmte Anzahl von Datenzeilen gelöscht wird.

Die `DELETE`-Syntax sieht folgendermaßen aus:

```
DELETE FROM Tabellenname WHERE x = y
```

Wie bei allen SQL-Anweisungen steht auch hier das Verb an erster Stelle. Dem Verb folgt die `FROM`-Klausel und schließlich der Name der Tabelle, aus der Sie die Datensätze löschen wollen. In einer `DELETE`-Anweisung ist die `WHERE`-Klausel von sehr großer Bedeutung. Sie begrenzt die Anzahl der Löschungen, die auf den hier angegebenen Kriterien basieren. Ohne die `WHERE`-Klausel werden alle Datensätze automatisch aus der Datenbank gelöscht. Folgende Anweisung löscht sämtliche Daten der `Customers`-Tabelle:

```
DELETE FROM Customers
```

Wie bei allen vorhergehenden Anweisungen bietet MySQL auch bei der `DELETE`-Anweisung die Möglichkeit, die Priorität der Anweisung anzugeben. Wollen Sie, dass eine `DELETE`-Anweisung ausgeführt wird, während kein anderer die Tabelle, aus der Sie Datensätze löschen wollen, einsieht, können Sie folgende Anweisung erlassen:

```
DELETE LOW_PRIORITY FROM Customer WHERE State = "KY"
```

Eine `DELETE`-Anweisung kann bei einer sehr großen Tabelle viel Zeit in Anspruch nehmen. Um die Tabelle für andere Anfragen verfügbar zu halten, können Sie multiple `LIMIT`-Anweisungen benutzen. Sie können die Grenze bei zehn ziehen und die Anweisung mehrfach ausführen. Die Anweisung löscht jedes Mal höchstens zehn Zeilen. Ist die `WHERE`-Klausel nicht länger WAHR, werden keine Zeilen mehr zurückgegeben. Ein Beispiel: Ihr Client möchte, dass Sie alle Kunden aus Kentucky löschen, weil sein Geschäft nicht mehr in diesem Staat tätig ist. Führen Sie folgende Query aus:

```
SELECT * FROM Customers WHERE State = "KY"
```

Diese Abfrage gibt 2.756 Datensätze aus Ihrer Datenbank zurück. Nun möchten Sie vom `LIMIT`-Schlüsselwort profitieren, um die Performance Ihrer Tabelle zu optimieren:

```
DELETE FROM Customers WHERE State = "KY" LIMIT 100
```

Diese Anweisung löscht bis zu 100 Datenzeilen, die den Kriterien in Ihrer `WHERE`-Klausel entsprechen. Jetzt bleiben Ihnen noch 2.656 Datensätze. Anschließend führen Sie die Anweisung erneut aus und löschen somit weitere 100 Datensätze – womit Sie nun bei 2.556 sind. Jedes Mal, wenn diese Query durchgeführt wird, nimmt die Anzahl der Datensätze, die den Kriterien Ihrer `WHERE`-Klausel entsprechen, um bis zu hundert ab. Dieser Prozess läuft so lange, bis es keine Datensätze mehr gibt, die diese Kriterien erfüllen. Durch diese Technik können Sie Ihre Datenbank löschen, ohne Performance einzubüßen.

Sie sollten zudem beachten, dass ein gelöschter Datensatz nicht einfach verschwindet. Er wird Teil einer internen Linkliste. Das bedeutet, dass gelöschte Datensätze immer noch Platz auf der Festplatte einnehmen. Um die Überbleibsel dieser gelöschten Datensätze zu entfernen, müssen Sie eine `OPTIMIZE`-Anweisung gegen die Tabellen ausführen oder das `myisamchk`-Programm benutzen. Beide Varianten säubern den Platz, der von diesen ungenutzten (*toten*) Datensätzen eingenommen wird.

Arbeiten mit Joins

Joins sind ein wesentlicher Bestandteil einer relationalen Datenbank. Sie ermöglichen dem Benutzer, von den in der Designphase hergestellten Beziehungen zu profitieren. Der Begriff *Join* beschreibt die Verknüpfung von zwei oder mehr Tabellen, um benötigte Daten auf der Basis der zwischen den Tabellen bestehenden Beziehungen zu erhalten. In Ihrer `Meet_A_Geek`-Datenbank besteht zum Beispiel eine Beziehung zwischen den Tabellen `Customers` und `Orders`. Wollen Sie Daten, die auf irgendeinem in der `Orders`-Tabelle enthaltenen Kriterium beruhen, aus der `Customers`-Tabelle auswählen, müssen Sie eine irgendwie geartete Verknüpfung verwenden. Um beispielsweise alle Kunden, die eine Bestellung aufgegeben haben, zu erfassen, können Sie folgende Syntax benutzen:

```
SELECT First_Name, Last_Name
FROM Customer AS C, Orders AS O
WHERE C.Customer_ID = O.Customer_ID
```

Diese Syntax gibt nur die Datensätze derjenigen Kunden zurück, die Bestellungen aufgegeben haben. Hat ein Kunde mehrere Bestellungen aufgegeben, erscheinen dessen sämtliche Bestellungen.



ANSI-92-Kompatibilität: 1992 entwickelte das American National Standards Institute eine Reihe von Standards für SQL. Diese Standards werden ANSI-92 genannt. MySQL ist sehr stolz darauf, ANSI-92-kompatibel zu sein. ANSI standardisiert die Methoden der Join-Erstellung. Es ermöglicht Ihnen, die Verknüpfungslogik in die `FROM`-Klausel statt in die `WHERE`-Klausel zu stellen. Es gibt zwei Schulen: Die eine nimmt die neue Methode, ANSI-92, an, die andere benutzt weiterhin die `WHERE`-Klausel, um Verknüpfungen

zu realisieren. Die alte Methode ist einfach zu lesen und sehr sinnvoll. Darüber hinaus ist die Mehrheit der Datenbank-Administratoren und Programmierer mit der alten Methode aufgewachsen. Die neue Methode hat aber einige Vorteile. Zum einen folgt sie dem ANSI-Standard. Alle neuen Datenbanken müssen die ANSI-92-Methode benutzen, wenn Sie kompatibel sein wollen. Seien Sie sich also sicher, dass die neue Methode in Zukunft mit von der Partie sein wird. Zudem verbessert Sie die Performance. Anstatt MySQL eine `WHERE`-Klausel verarbeiten zu lassen, um dann einen Abfrageplan zu erstellen, wird der Join vor der `WHERE`-Klausel angegeben, was die Anfragen in Hinsicht auf Performance besser macht.

Was die Syntax anbelangt, unterscheidet sich die ANSI-Methode, einen Join auszuführen, von der alten Methode. Das Konzept allerdings stimmt überein. Möchten Sie Informationen aus zwei Tabellen, die in Beziehung zueinander stehen, verbinden, um eine Ergebnismenge zu erhalten, benutzen Sie einen ANSI-92-JOIN. Um die Ergebnismenge aus dem vorherigen Beispiel zu erhalten, verwenden Sie folgenden Join:

```
SELECT First_Name, Last_Name
FROM Customer as C
JOIN Orders as O ON C.Customer_ID = O.Customer_ID
```

Das grundlegende Eingabeformat des ANSI-92-JOIN sieht folgendermaßen aus:

```
SELECT Spaltennamen-Liste
FROM Tabellennamen
JOIN Tabellennamen ON Verknüpfungsbedingung
WHERE Bedingung
```

Der JOIN folgt der grundlegenden SELECT-Syntax. Die JOIN-Klausel ist das Einzige, was hinzugefügt wird. Nach der JOIN-Klausel steht der Name der Tabelle, die in Beziehung zu derjenigen Tabelle, die nach dem Schlüsselwort `FROM` angegeben ist, steht. Dem Tabellennamen folgt das JOIN-Kriterium, welches im Grunde die Definition der Beziehung der beiden Tabellen darstellt. Das JOIN-Kriterium funktioniert nach der gleichen Logik wie Ihre `WHERE`-Klausel. Nach der JOIN-Klausel steht die `WHERE`-Klausel, die Ihrer SELECT-Anweisung weitere Limitierungskriterien hinzufügt. Wollen Sie sämtliche Bestellungen eines Kunden einsehen, können Sie folgende Syntax benutzen:

```
SELECT C.First_Name, C.Last_Name, O.Order_Date, O.Order_ID
FROM Customers as C
JOIN Orders as O ON C.Customer_ID = O.Customer_ID
WHERE Customer_ID = 12
```

Nach der alten Methode sieht dies folgendermaßen aus:

```
SELECT C.First_Name, C.Last_Name, O.Order_Date, O.Order_ID
FROM Customers as C, Orders as O
WHERE Customer_ID = O.Customer_ID
AND C.Customer_ID=12
```


Welche der beiden Methoden Sie wählen, liegt wiederum ganz bei Ihnen. Es ist eine Frage des Geschmacks. Der Gebrauch von ANSI-92 kann zwar einen kleinen Leistungszuwachs mit sich bringen, doch wird die alte Methode auf Grund der besseren Lesbarkeit und Klarheit bevorzugt.

MySQL unterstützt folgende JOINS: CROSS JOIN, INNER JOIN, LEFT JOIN und NATURAL LEFT JOIN. Lassen Sie sich nicht von den Namen oder der Methode verwirren. Beachten Sie einfach, dass Sie einen JOIN ausführen können, indem Sie in Ihrer WHERE-Klausel logisch vorgehen. ANSI-92 und die alte Methode werden beide in diesem Kapitel abgehandelt; benutzen Sie also die Methode die Ihnen gefällt und am sinnvollsten erscheint.

CROSS JOIN

Der CROSS JOIN wird nicht sehr häufig eingesetzt – um bei der Wahrheit zu bleiben: eigentlich nur aus Versehen. Ein CROSS JOIN gibt alle Datensätze der in der JOIN-Klausel aufgeführten Tabellen zurück. Diese Art von JOIN wird auch *karthesischer Join* genannt. Diese Joins sind sehr Prozessor-intensiv und sollten daher möglichst vermieden werden. Die Syntax für einen CROSS JOIN sieht folgendermaßen aus:

Nach der alten Methode:

```
SELECT C.First_Name, C.Last_Name, O.Order_ID
FROM Customers as C, Orders as O
```

Nach ANSI-92:

```
SELECT C.First_Name, C.Last_Name, O.Order_ID
FROM Customers as C
CROSS JOIN Orders as O
```

Die von diesem Join-Typ zurückgegebene Ergebnismenge ist enorm. Es werden alle Daten sämtlicher Tabellen zurückgegeben. Einen solchen Join braucht man nicht sehr oft.

INNER JOIN

INNER JOINS sind wahrscheinlich die gängigsten. Ein INNER JOIN sondert alle Datensätze, die nicht zueinander passen, aus. Nur zueinander passende Datenzeilen werden in der Ergebnismenge angezeigt. Der INNER JOIN stellt den Normaltyp eines Joins dar. Aus diesem Grund ist der Zusatz *INNER* auch nur fakultativ. Dieser Join-Typ basiert auf den Kriterien in der JOIN-Klausel. Ihr erstes Beispiel war ein Beispiel für einen INNER JOIN:

```
SELECT C.First_Name, C.Last_Name, O.Orders_ID
FROM Customers as C
(INNER) JOIN Orders as O ON C.Customer_ID = O.Customer_ID
```

Das Wort `INNER` steht hier in Klammern, weil es fakultativ ist. Die alte Methode, einen `INNER JOIN` zu erstellen, sieht folgendermaßen aus:

```
SELECT C.First_Name, C.Last_Name, O.Order_ID
FROM Customer AS C, Order as O
WHERE C.Customer_ID = O.Customer_ID
```

Der `INNER JOIN` ist wahrscheinlich der am häufigsten verwendete Join – das ist auch der Hauptgrund, warum er der ANSI-92-Standard ist.

LEFT JOIN

Ein `LEFT JOIN` gibt alle Datenzeilen der in einem Join links angegebenen Tabelle an. Anhand Ihres Beispiels mit den Tabellen `Customers` und `Orders` können Sie in der folgenden Anweisung die Syntax für einen `LEFT JOIN` sehen:

```
SELECT C.First_Name, C.Last_Name, O.Orders_ID
FROM Customers as C
LEFT JOIN Orders as O ON C.Customer_ID = O.Customer_ID
```

In diesem Beispiel werden alle Datenzeilen der Tabelle, die sich links der Gleichung befinden, zurückgegeben, egal ob es in der Tabelle auf der rechten Seite eine Entsprechung gibt oder nicht. Ist das nicht der Fall, wird ein `NULL`-Wert zurückgegeben.

In anderen Datenbank-Systemen wird der `LEFT JOIN` nach der alten Methode, durch Gebrauch eines Symbols, ausgeführt. Dieses Symbol ist hier eine Kombination aus Asterisk (*) und Gleichzeichen (=). Die Position des Asterisks (links oder rechts des Gleichzeichens) zeigt den Join-Typ an. Um also einen `LEFT JOIN` auszuführen, müssen Sie folgende Syntax verwenden:

```
SELECT C.First_Name, C.Last_Name, O.Order_ID
FROM Customers AS C, Orders as O
WHERE C.Customer_ID *= O.Orders_ID
```

Die Entwickler von TcX haben beschlossen, dieses Feature nicht in MySQL einzuführen. Sie taten dies wahrscheinlich aus mehreren Gründen. Zum einen werden Joins dieses Typs nur selten benutzt. Verwenden Sie einen solchen Join, können Sie genauso gut das ANSI-92-Äquivalent benutzen. Zum anderen arbeitet TcX mit einem sehr niedrigen Budget, und da der `LEFT JOIN` ein selten benutztes Extra-Feature ist, verwenden die Entwickler ihre Zeit lieber auf die Standards.

Die `USING`-Klausel kann als Ersatz für die `ON`-Klausel benutzt werden. Die `USING`-Klausel ermöglicht Ihnen, eine Liste von Spalten, die in beiden Tabellen enthalten ist, zu benutzen und ist äquivalent mit `C.Customer_ID = O.Customer_ID AND S.Shipper_ID = O.Shipper_ID`. Die Syntax sieht folgendermaßen aus:

```
SELECT C.First_Name, C.Last_Name, O.Order_ID
FROM Customers AS C
LEFT JOIN Order AS O USING Customer_ID
```

In diesem Beispiel entspricht die `USING`-Klausel der `ON`-Klausel mit dem Zusatz `C.Customer_Id = O.Customer_ID`. Dies ist eine weitere Methode, um Zeit zu sparen.

NATURAL LEFT JOIN

Außer der Tatsache, dass er alle zusammenpassenden Spalten automatisch als Teil eines Joins benutzt, entspricht der `NATURAL LEFT JOIN` einem `LEFT JOIN`. Was die Syntax betrifft, ist er äquivalent mit einem `LEFT JOIN + USING`-Klausel, die alle identischen Spalten der beiden Tabellen benennt. Die Syntax eines `LEFT JOIN` sieht folgendermaßen aus:

```
SELECT C.First_Name, C.Last_Name, O.Order_ID
FROM Customers AS C
NATURAL LEFT JOIN Orders as O
```

Dies gibt alle Datensatzzeilen der `Customers`-Tabelle zurück, ungeachtet der Tatsache, ob sie einen entsprechenden Datensatz in einer der gleichen Spalten der `Orders`-Tabelle haben oder nicht. Gibt es keine Entsprechung, so gibt die `Orders`-Tabelle einen `NULL`-Wert zurück.

Dieser Join wird ebenfalls kaum benutzt. Die einzige Möglichkeit, diesen Join-Typ auszuführen, besteht in der Anwendung der oben genannten Syntax.

Joins sind ziemlich einfach. Lassen Sie sich da von der Syntax nicht täuschen. Mit der Hilfe von Joins profitieren Sie von den Beziehungen, die Sie beim Erstellen Ihrer Tabellen definiert haben. Übung und Erfahrung werden Ihnen beim Erstellen von Joins eine große Hilfe sein.

Aggregat-Funktionen

Funktionen, die in Spalten zusammenfassende, mathematische Berechnungen durchführen, nennt man Aggregat-Funktionen. MySQL führt folgende aus: `COUNT()`, `COUNT(DISTINCT)`, `MAX()`, `MIN()`, `SUM()` und `STD()`.

COUNT()

Die `COUNT`-Funktion gibt an, wie viele Nicht-Null-Werte in einer Spalte vorkommen/enthalten sind. Zum Beispiel:

```
SELECT COUNT(C.First_Name)
FROM Customers as C, Orders as O
WHERE C.Customer_ID = O.Customer_ID
```

Diese Anweisung gibt die Anzahl der Datensätze zurück, die der WHERE-Klausel entsprechen und Nicht-Null-Werte in der `First_Name`-Spalte der `Customers`-Tabelle enthalten.

Weiterhin ist die `COUNT()`-Funktion in folgender Weise zu gebrauchen:

```
SELECT COUNT(*) as Num
FROM Customers as C, Orders as O
WHERE C.Customer_ID = O.Customer_ID
```

Diese Anweisung gibt die Gesamtanzahl der Datensätze zurück, die der WHERE-Klausel entsprechen. Diese Funktion erweist sich als besonders nützlich, wenn man herausfinden möchte, wie viele Datensätze es in einer Tabelle gibt.

COUNT(DISTINCT)

Die `COUNT(DISTINCT)`-Funktion gibt die eindeutigen Nicht-Null-Werte einer oder mehrerer Spalten an. Zum Beispiel:

```
SELECT COUNT(DISTINCT First_Name,)
FROM Customers
```

Diese Anweisung gibt die Anzahl der Datensatzzeilen mit verschiedenen Vornamen zurück. In der `Customers`-Tabelle gibt es beispielsweise zehn Datensatzzeilen. In den ersten drei Zeilen der `First_Name`-Spalte steht `Taylor`, in den darauffolgenden zwei `Sydney`, in der nächsten `Jill` und in den letzten vier `Trisha`. Die obige Anweisung gibt die Zahl 4, also die Gesamtanzahl der eindeutigen Namen, die in der `First_Name`-Spalte gespeichert sind, zurück.

Die `COUNT` und `COUNT(DISTINCT)`-Funktionen können bei jeder Art von Datentyp, die folgenden allerdings nur bei numerischen angewendet werden:

MAX()

Die `MAX()`-Funktion gibt den höchsten in einer Spalte befindlichen Wert zurück. Sie ist sehr nützlich, wenn Sie Ihre eigene Folgenummern erstellen möchten. Die Syntax sieht folgendermaßen aus:

```
SELECT MAX(Customer_ID)
FROM Customers
```

Diese Anweisung gibt den höchsten in einer Spalte befindlichen Wert wieder.

MIN()

Die `MIN()`-Funktion gibt den niedrigsten in einer Spalte befindlichen Wert wieder. Die Syntax sieht folgendermaßen aus:

```
SELECT MIN(Customer_ID)
FROM Customers
```

AVG()

Die `AVG()`-Funktion gibt den Durchschnittswert der Werte einer angegebenen Spalte zurück. Die Funktion wird folgendermaßen benutzt:

```
SELECT AVG(Price)
FROM Orders
```

Diese Anweisung gibt den Durchschnittspreis aller in der `Orders`-Tabelle enthaltenen Preise an. Die Rechnungen führt MySQL für Sie aus.



Beachten Sie, dass ein `NULL`-Wert überhaupt kein Wert ist. Wenn Sie also `SUM`- oder `AVG`-Funktionen bei einer Spalte, die `NULL`-Werte enthält, anwenden, werden die `NULL`-Werte Ihr Ergebnis in keiner Weise beeinflussen.

SUM()

Diese Funktion rechnet die in einer Spalte befindlichen Werte zusammen und wird folgendermaßen benutzt:

```
SELECT SUM(Price)
FROM Orders
```

Diese Funktion addiert alle Preise und gibt deren Summe zurück.

STD()

Die `STD`-Funktion gibt, wie im folgenden Beispiel gezeigt wird, die Standardabweichung einer spezifizierten Spalte zurück:

```
SELECT STD(Price)
FROM Orders
```



Als *Standard-Abweichung* bezeichnet das Maß der Abweichung. Es wird benutzt, um zu ermitteln, wie weit ein Wert vom Mittelwert entfernt ist. Mathematisch kann dieses folgendermaßen ausgedrückt werden

$\text{Sqrt } E[(X-E(X))^2]$

wobei E ein bekannter Wert ist.

Aggregat-Funktionen und die WHERE-Klausel

Beachten Sie, dass die `WHERE`-Klausel auch in jedem der vorangegangenen Beispiele angewandt werden kann. Sie schränkt ein, welche Werte in der Funktion berechnet werden. Die `WHERE`-Klausel liefert gemeinsam mit den Aggregat-Funktionen leistungsstarke Werkzeuge, mit denen Sie die meisten finanziellen oder wissenschaftlichen Berechnungen ausführen können.

Aggregat-Funktionen und die GROUP BY-Klausel

MySQL unterstützt auch die `GROUP BY`-Klausel. Diese Klausel kann nur in Verbindung mit einer Aggregat-Funktion angewandt werden. Sie ermöglicht, Ergebnismengen zu gruppieren. Wollen Sie zum Beispiel die Anzahl der Bestellungen, die eine Person an einem bestimmten Tag aufgegeben hat, wissen, so können Sie folgende Anweisung benutzen:

```
SELECT C.First_Name, C.Last_Name, O.Order_Date
Count(C.Customer_ID)AS Orders
FROM Customer as C
JOIN Orders as O ON C.Customer_ID = O.Customer_ID
WHERE O.Order_Date = '2000-01-13'
GROUP BY C.First_Name
```

Die Ergebnisse sind in Tabelle 9.3 angeführt.

First_Name	Last_Name	Order_Date	Orders
Taylor	Smith	2000-01-13	3
Sydney	Sue	2000-01-13	1
Stan	Behm	2000-01-13	6

Tabelle 9.3: Aggregat-Funktionen mit GROUP BY

Beachten Sie, dass die zusammengefassten Ergebnisse vom Spaltennamen der `GROUP BY`-Klausel abhängen. Beim Generieren von Berichten ist diese Klausel ein unschätzbares Werkzeug.

Sortieren

Die meisten RDBMS ermöglichen den Benutzern, Ergebnisse zu sortieren. MySQL bildet hier keine Ausnahme. Um eine Ergebnismenge, die von einer bestimmten Spalte abhängt, zu sortieren, können Sie die `ORDER BY`-Klausel benutzen. Diese Klausel sorgt dafür, dass eine Ergebnismenge nach den von Ihnen benannten Spalten und von

Ihnen vorgegebenen Richtlinien sortiert wird. Möchten Sie das letzte Beispiel nach Nach- und Vornamen in absteigender alphabetischer Reihenfolge (Z–A) sortieren, benutzen Sie folgende Anweisung:

```
SELECT C.First_Name, C.Last_Name, O.Order_Date  
Count(C.Customer_ID)AS Orders  
FROM Customer as C  
JOIN Orders as O ON C.Customer_ID = O.Customer_ID  
WHERE O.Order_Date = '2000-01-13'  
GROUP BY C.First_Name  
ORDER BY Last_Name, First_Name DESC
```

Wollen Sie Ihre Daten in aufsteigender alphabetischer Reihenfolge (A–Z) sortieren, ersetzen Sie lediglich `DESC` durch `ASC` – das ist schon alles. Um Spalten des Datentyps `TEXT` und `BLOB` zu sortieren, müssen Sie die `ORDER BY`-Klausel auf einen mit der `SUBSTRING()`-Funktion begrenzten Bereich anwenden. Das wird in der morgigen Lektion behandelt.

In dem oben angeführten Beispiel finden Sie in der `ORDER BY`-Klausel zwei Spaltennamen. Die `ORDER BY`-Klausel bewirkt, dass zunächst nach der ersten und dann nach der zweiten Spalte sortiert wird. Das ist die gängige Reihenfolge – normalerweise sortieren Sie nach Nachnamen, und nur wenn zwei Personen den gleichen Nachnamen besitzen, benutzen Sie auch den Vornamen, um die Reihenfolge zu bestimmen. Genau auf diese Weise handelt auch die `ORDER BY`-Klausel, wenn mehrere Spaltennamen angegeben werden. Sie sortiert nach der ersten Spalte und falls diese Spalte zwei gleiche Werte enthält, wechselt sie in die zweite Spalte und vergleicht hier die betreffenden Werte, um zu bestimmen, wo der Wert im Ergebnissatz stehen soll.

Zusammenfassung

SQL ist ein leistungsstarkes Werkzeug, das ursprünglich von IBM entwickelt wurde, um Otto Normalverbraucher den Zugang zu Datenbanken zu ermöglichen. SQL ist die Sprache, die Datenbanken verstehen. Sie besitzt eine Syntax und ein Vokabular, die sie zur Kommunikation verwenden kann. In SQL hat jeder reservierte Wert eine bestimmte Bedeutung und Funktion.

Heute haben Sie eine Menge Stoff erarbeitet. Vergessen Sie aber nicht, dass dies nur ein kleiner Einblick in SQL war – Überblicke sind naturgemäß unvollständig. Das Ziel dieses Buches ist nicht, Sie mit einer Flut von Informationen zuzudecken, sondern Hilfsmittel vorzustellen, mit denen Sie Ihre Daten handhaben, und letztendlich eine Grundlage zu schaffen, auf der Sie aufbauen können. In späteren Lektionen werden Sie APIs kennen lernen, die das, was Sie heute gelernt haben, nutzen, um Berichte zu erstellen und Anwendungen zu unterstützen. Das Schwierigste an SQL ist nicht die

Syntax oder das Vokabular, sondern die dahinter stehende Logik. Nehmen Sie sich Zeit, um zu lernen, wie Sie gute `WHERE`-Klauseln erstellen und `JOINS` benutzen. Es wird Ihnen das Leben in Zukunft leichter machen.

Fragen und Antworten

F In der heutigen Lektion werden *LEFT JOINS* behandelt. Gibt es auch *RIGHT JOINS*?

A *Ja, die gibt es. Die meisten großen RDBMS unterstützen RIGHT JOINS. MySQL unterlässt dies aus gutem Grund. Wenn Sie einmal kurz darüber nachdenken, was ein RIGHT JOIN ist – nämlich die Umkehrung eines LEFT JOINS – dann gibt es für Sie einen einfachen Lösungsansatz. Vertauschen Sie in Ihrer Abfrage einfach die Tabellennamen. Ist dies zum Beispiel Ihr LEFT JOIN*

```
SELECT C.First_Name
FROM Customer AS C
LEFT JOIN Orders AS O ON C.Customer_ID = O.Customer_ID
```

dann sieht Ihr RIGHT JOIN folgendermaßen aus:

```
SELECT C.First_Name
FROM Orders AS O
LEFT JOIN Customers AS C ON C.Customer_ID = O.Customer_ID
```

Wie Sie sehen, ist es absolut unnötig, einen RIGHT JOIN zu codieren und auszuführen – indem Sie die Reihenfolge der Tabellen umdrehen, erhalten Sie das gleiche Resultat.

F Ich habe mir das MySQL-Handbuch durchgelesen und bin auf eine *REPLACE*-Anweisung gestoßen. Was ist das und wie kann ich sie benutzen?

A *Die REPLACE-Anweisung entspricht im Wesentlichen dem INSERT INTO-Befehl. Der einzige Unterschied besteht darin, dass MySQL bei der REPLACE-Anweisung den alten Datensatz löscht und ihn durch neue Werte ersetzt – daher auch die Bezeichnung REPLACE. Die Syntax für eine REPLACE-Anweisung, die im Großen und Ganzen der INSERT INTO-Syntax entspricht, sieht folgendermaßen aus:*

```
REPLACE INTO Customers
(Customer_ID, First_Name, Last_Name)
VALUES (NULL, "Dina", "Newton")
```

REPLACE- und INSERT-Anweisung besitzen auch die gleichen Regeln und Parameter, um die REPLACE INTO...SELECT-Anweisung sowie das LOW_PRIORITY-Attribut einzuschließen.

Übungen

1. Erstellen Sie aus folgenden Beispielen SQL-Anweisungen:
 - ▶ Geben Sie alle Datensätze der Kunden aus der `Customers`-Tabelle aus, die Bestellungen aufgegeben haben.
 - ▶ Geben Sie alle Spediteure, die im letzten Monat beauftragt wurden, aus.
 - ▶ Geben Sie alle Kunden aus.
2. Fügen Sie folgenden Kunden in die `Customers`-Tabelle ein:
 - ▶ Vorname: Jerry
 - ▶ Nachname: Maxon
 - ▶ Adresse: 123 N. Hampton Lane
3. Ersetzen Sie den Namen »Jerry« aus dem letzten Datensatz durch »Sam«.
4. Löschen Sie sämtliche Datensätze der `Customers`-Tabelle, deren Vorname »Jerry« lautet.
5. Erstellen Sie eine SQL-Anweisung, die die Anzahl der Kunden zurückgibt, deren Name mit M beginnt.
6. Erstellen Sie eine SQL-Anweisung, die die Durchschnittsmenge der Produkte, die jeder Kunde kauft, zurückgibt (Hinweis: Benutzen Sie die `quantity`-Spalte)
7. Erstellen Sie eine SQL-Anweisung, die alle Kunden ausgibt, die im letzten Monat eine Bestellung aufgegeben haben. Sortieren Sie diese in aufsteigender alphabetischer Reihenfolge (A–Z) und gruppieren Sie die Datensätze nach Datum.



Interne MySQL- Funktionen

**Woche
2**

Fast alle Datenbanken stellen dem Benutzer eine Reihe geläufiger Funktionen zur Verfügung. Diese ermöglichen Administratoren oder anderen Benutzern, in einer Abfrage komplexe Aufgaben wie Zeichenketten-Bearbeitungen oder schwierige mathematische Berechnungen auszuführen. Dies gestattet dem Benutzer, von der leistungsstärkeren Prozessverarbeitung eines Datenbank-Servers zu profitieren. MySQL beinhaltet viele interne Funktionen. Heute lernen Sie:

- ▶ Programmieren auf der Server-Ebene und wie MySQL Ihnen dabei helfen kann
- ▶ Grundlegende Funktionen wie zum Beispiel numerische Operationen, Zeichenketten-Bearbeitungen und logische Bedingungen

Programmieren auf Server-Ebene

Momentan gibt es heftige Diskussionen darüber, an welcher Stelle die Programmausführung platziert werden soll. Soll die Applikation die gesamte schwere Prozessverarbeitung auf dem Benutzer-Desktop bewältigen oder soll das auf dem Datenbank-Server geschehen? Oder vielleicht an dritter Stelle, einem so genannten *Middle Tier*? Die Antwort auf diese Fragen ist nicht einfach. Sehr viel hängt von der Anwendung und dem Zweck ab, für den Sie diese einsetzen möchten. Eine herkömmliche webgestützte Anwendung hat andere Ansprüche als eine *Three-Tier*-Anwendung (Drei-Schichten-Anwendung). Die Ansprüche variieren vom vielbesuchten Online-Shop bis hin zur massiven Datenverarbeitung in einer Finanzwesen-Anwendung. Der Rest dieses Abschnitts stellt einige Beispiele vor und erläutert, welches der beste Ort für die Programmausführung ist und aus welchem Grund.

Das erste Beispiel ist eine webbasierte Anwendung wie beispielsweise die *Meet_A_Geek*-Webseite. Die gängigen Schritte bei einer webbasierten Anwendung sehen folgendermaßen aus:

1. Der Benutzer gibt die URL einer Webseite in seinen Browser ein.
2. Der Browser ermittelt den Server und befragt eine Webseite.
3. Der Web-Server beantwortet die Anfrage und schickt die Seite.
4. Der Browser erhält die Seite.
5. Der Benutzer durchforstet die Webseite, indem er die Abschnitte 1–4 so lange wiederholt, bis er auf eine Seite stößt, die Datenbankzugang erfordert (zum Beispiel ein Formular oder eine dynamisch generierte Seite).
6. Ein Programm oder Script läuft, das die angeforderten Daten vom Datenbank-Server erhält.
7. Die Seite wird dem Benutzer zurückgegeben.

Wie Sie sehen, gibt es viele Wege, die zur Datenbank oder zum Web-Server hin- und zurückführen. Stellen Sie sich eine stark besuchte Website vor – die Anfragen an den Server und seine Antworten könnten ins Unermessliche gehen.

Um den Server zu entlasten, könnten Sie die Programmausführung auf den Client abwälzen. Der Client ist der Rechner bzw. die Anwendung, welcher bzw. welche die Anfragen erhält oder versendet. Je mehr Logik (wie zum Beispiel Fehlerkorrektur oder Zeichenketten-Bearbeitung) Sie auf dem Client-Level anbringen können, desto besser für die Auslastung Ihres Servers.

Glauben Sie aber nicht, Sie könnten jetzt keine internen Funktionen auf dem Datenbank-Server benutzen – Seien Sie beruhigt, Sie können. Achten Sie lediglich auf die Größe Ihrer Datenbank und auf die Frequentierung Ihrer Seite. Wenn Sie eine kleine Seite mit recht wenig Daten besitzen, dann haben Sie so ziemlich freie Hand. Ist Ihre Seite aber stark frequentiert und muss sie viele Berechnungen ausführen oder werden viele Zeichenketten benötigt, wäre es vielleicht sinnvoller, diese Logik auf dem Client zu platzieren.

In einem Unternehmen kann das alles ganz anders aussehen. Denn auf Unternehmensebene greifen vielleicht mehrere Clients auf Ihre Datenbank via einer Anwendung zu. Diese Anwendung kann in Java, Visual Basic, C++ oder in einer anderen Programmiersprache geschrieben sein. Die Applikation kommuniziert mit einer mittleren Schicht, einem so genannten Middle-Tier, der die gesamte Geschäftslogik handhabt. Dieser verschafft sich Zugang zur Datenbank und gibt dem Client Anfragen zurück. Der Middle-Tier ist ebenfalls eine Applikation. In seiner Verantwortung liegt es, Anfragen zu verarbeiten und an die richtigen Stellen zu schicken.

Der Ablauf in einem Three-Tier-Design (Drei-Schichten-Modell) sieht folgendermaßen aus:

1. Ein Benutzer verwendet eine Front-End-Applikation und sendet eine Anfrage.
2. Die Anfrage wird auf Fehler überprüft und an den Middle-Tier geschickt.
3. Der Middle-Tier erhält die Anfrage, führt die notwendige Geschäftslogik aus und leitet sie an die Datenbank weiter.
4. Die Datenbank bearbeitet die Anfrage – sie gibt entweder ein Ergebnis aus einer `SELECT`-Anweisung zurück oder aktualisiert einige Datensätze. Anschließend schickt sie die Daten zum Middle-Tier zurück.
5. Der Middle-Tier führt die noch erforderliche Logik aus und gibt die Anfrage an den Client zurück.
6. Der Client erhält die Daten und übermittelt dem Benutzer die Ergebnisse.

Der größte Teil der Geschäftslogik wird also auf dem Middle-Tier platziert, während einfaches Parsen und Fehlerkorrekturen auf der Ebene der Front-End-Anwendung bewerkstelligt werden. Im Allgemeinen besteht die Aufgabe des Datenbank-Servers

lediglich darin, die angeforderten Daten zur Verfügung zu stellen oder die gespeicherten Daten zu bearbeiten. Bei einem so konstruierten Typ wird die Arbeit auf alle Schichten verteilt – jede Schicht erfüllt ihre Aufgaben. Die Front-End-Applikation ist in weiten Teilen ein einfaches Anzeigewerkzeug – eine Möglichkeit für den Benutzer, mit den Daten zu hantieren. Der Middle-Tier erledigt alle schwierigen Vorgänge und gibt Anfragen und Verbindungen an die Datenbank weiter. Die Datenbank macht das, was sie am besten kann – Daten zur Verfügung stellen.

In einer reinen Client-Server-Umgebung müssen beide Ihre Aufgaben erfüllen. Früher mussten die Server den Löwenanteil der logischen Vorgänge verarbeiten, da sie meist leistungstärker als die Client-Maschinen waren. Seit es günstige, leistungsstarke PCs gibt, kann ein größerer Teil der Logik auf der Client-Ebene ausgeführt werden. Heute verwenden die meisten großen Applikationen ein Drei- bzw. Multi-Schichten-Modell. Bei kleineren Applikationen kann allerdings auf die mittlere Schicht verzichtet werden.

Der allgemeine Ablauf in einer Client-Server-Applikation sieht folgendermaßen aus:

1. Der Benutzer verwendet eine grafische Applikation und sendet eine Anfrage. Die Applikation bearbeitet die ganze Logik, parst und korrigiert Fehler und schickt anschließend die fertige Anfrage an den Datenbank-Server.
2. Der Datenbank-Server erhält die Anfrage, führt sämtliche notwendigen Operationen durch und sendet die vervollständigte Anfrage zurück an den Client.
3. Der Client führt alle weiteren Operationen aus und übermittelt dem Benutzer die Daten.

Dieser Vorgang beinhaltet nicht so viele Schritte wie andere Applikationen. Zudem hat er seine Grenzen. Für kleine bis mittelgroße Applikationen besteht keine Notwendigkeit, so komplex wie andere Architekturen zu sein.

Der letzte und einfachste Architekturtyp ist der Bericht. Er kann von der Kommando-Zeile des MySQL-Monitors aus ausgeführt werden oder in einem einfachen Shell oder einem Perl-Script enthalten sein. Die einzige Aufgabe dieses Typs ist das Generieren eines Berichts. Der Bericht kann auch sehr komplex sein – er läuft im Normalfall folgendermaßen ab:

1. Der Benutzer stellt eine Verbindung zur Datenbank her und erlässt eine Anfrage oder eine Reihe von Anfragen.
2. Das Ergebnis wird auf dem Bildschirm oder in einer Datei ausgegeben.
3. Der Benutzer beendet die Verbindung zur Datenbank.

Dieser Programmtyp ist ein einfacher Bericht. Er sorgt sich nicht um eine grafische Schnittstelle. Die Logik – je nachdem wie komplex die Anfrage ist – kann fast ausschließlich auf dem Datenbank-Server mit den MySQL-internen Funktionen erledigt werden.

Beispiele hierfür sind einfache Batch-Jobs, die nach ihrer Beendigung Rechnungen ausdrucken, oder ein Bericht, der jeden Tag die Standardabweichung von Testergebnissen berechnet. Die Einsatzmöglichkeiten sind unbegrenzt. Allerdings benutzen diese Typen im Allgemeinen die Datenbank und nicht den Client für Ihre Logik.

Wie kann MySQL Ihnen helfen?

MySQL stellt viele Funktionen zur Verfügung, die Ihnen bei zahlreichen Aufgaben helfen können. Auf Anwendungsebene können diese Funktionen anstelle des Programmcodes oder in Verbindung mit ihm benutzt werden. Orientieren Sie sich hierbei an den vorhin aufgestellten Richtlinien.

Wegen der Schnelligkeit von MySQL kann es von Vorteil sein, wenn Sie einige der von Ihnen benötigten Funktionen auf die Server-Ebene auslagern. Schwindeln Sie ein wenig. Profitieren Sie von dem, was MySQL anzubieten hat. Sollen in Ihrer Anwendung Vor- und Nachname angezeigt werden, die aber in zwei verschiedenen Spalten gespeichert sind, benutzen Sie eine der internen Funktionen (`concat()`), die MySQL anbietet, und verbinden Sie die beiden Spalten, bevor der Name zu Ihrer Anwendung zurückgegeben wird. So sparen Sie nicht nur Zeit, sondern Sie ersparen sich auch jede Menge Kopfschmerzen.

Die grundlegenden MySQL-Funktionen

Die MySQL-Funktionen können in vier Typen aufgegliedert werden: numerische oder mathematische Funktionen, logische Funktionen, Zeichenketten-Funktionen sowie Datum- und Zeit-Funktionen. Heute werden Sie etwas über die ersten drei Typen erfahren. Die Datum- und Zeit-Funktionen werden wegen Ihrer großen Anzahl und Komplexität in einem gesonderten Kapitel, »MySQL und die Zeit«, an Tag 16 behandelt. In Anhang B, »Aktuelle MySQL-Funktionen«, finden Sie zudem eine Auflistung sämtlicher Funktionen einschließlich Kurzbeschreibung und Beispiel.

In der heutigen Lektion ist nur ein Teil der Funktionen aufgelistet. Lediglich die geläufigsten werden erklärt und anhand von Beispielen wird veranschaulicht, wie sie zu benutzen sind.

Nummerische Funktionen

Die numerischen Funktionen setzen sich aus verschiedenen mathematischen Operationen zusammen – diese reichen von komplex (Standardabweichung) bis einfach (Ad-

dition). In diesem Abschnitt werden Gebrauch und Syntax der leichteren Operationen erörtert. Wenn Sie den Dreh raus haben, können Sie einen Blick auf Anhang B werfen und sich an die komplexen Funktionen wagen.

Der Aufbau einer Funktion ist ziemlich einfach. Dem Funktionsnamen folgt sein Argument, das in Klammern () steht (siehe Abbildung 10.1). Zwischen dem Funktionsnamen und der Klammer ist keine Leerstelle (»«). Lassen Sie aus Versehen doch eine Leerstelle, so wird ein Fehler angezeigt, da MySQL davon ausgeht, dass Sie eine Spalte aus Ihrer Tabelle anstatt einer Funktion benennen möchten – das ist die Art und Weise, wie der Parser arbeitet. Eine Funktion kann ein Argument, mehrere Argumente oder gar keins haben. Die numerische Funktion kann nur numerische Typen und die Zeichenketten-Funktionen können nur Zeichen- oder Zeichenketten-Datentypen benutzen. Wenn Sie versuchen, etwas auszuführen, für was die Funktion nicht vorgesehen ist, wird ein Fehler angezeigt.

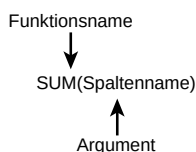


Abbildung 10.1:
Der Aufbau einer Funktion

Addition

Alle MySQL-Funktionen funktionieren in etwa gleich. Um die Additionsfunktion – wie auch jede andere Funktion, die diesen Zweck erfüllt – zu benutzen, müssen Sie diese aus einer SQL-Anweisung aufrufen. Sie wenden die Additionsfunktion folgendermaßen an:

```
SELECT (Spalte_1 + Spalte_2) as Total FROM Orders
```

Diese Anweisung gibt für jede Datensatzzeile der Tabelle die Summe der Werte aus `Spalte_1` und `Spalte_2` zurück. Stellen Sie sich zum Beispiel vor, Sie hätten zwei Datensatzzeilen in Ihrer Tabelle. Die erste Spalte enthält den Wert drei, die zweite in der gleichen Spalte den Wert vier. Die zweite Zeile Ihrer Tabelle enthält die Werte drei und drei. Das alles sieht dann folgendermaßen aus:

Spalte 1	Spalte 2
3	4
3	3

Wenn Sie jetzt die oben angegebene Query ausführen, erhalten Sie folgende Ergebnismenge:

```
Total
7
6
```


Jede Zeile wurde einzeln addiert. Um beide Zeilen der beiden Spalten zu addieren, benutzen Sie folgende Anweisung:

```
SELECT (SUM(Spalte_1) + SUM(Spalte_2)) AS Total FROM Orders
```

Ihre Ergebnismenge sieht folgendermaßen aus:

```
Total  
13
```

Die `SUM()`-Funktion rechnet die Werte der als Argument angegebenen Spalte zusammen. Sie wird in diesem Abschnitt behandelt, weil es sich um Addition handelt – wenn auch nur im weiteren Sinne.

Die `SUM()`-Funktion ist sehr praktisch, wenn Ihre Applikation oder Webseite die Gesamtsumme einer oder mehrerer Spalten aus Ihrer Tabelle benötigt. Finanztechnische Applikationen und Internetshopping sind zwei sehr gute Beispiele für einen möglichen Gebrauch dieser Funktion. Es ist ratsam, die Rückgabewerte Ihrer Funktion zu benennen – sonst müssen Sie auf Ihre Spalte anhand der Nummerierung statt anhand des Namens zugreifen, was ein wenig verwirrend sein kann.

Subtraktion

Subtraktion funktioniert ungefähr genauso wie Addition. Sie benutzen diese Funktion, um Spaltenwerte zu subtrahieren. Eine Anweisung, die eine Subtraktionsfunktion beinhaltet, sieht folgendermaßen aus:

```
SELECT (Spalte_2 - Spalte_1) AS Difference FROM Orders
```

Diese Anweisung gibt die Differenz der Spaltenwerte jeder Zeile einer Tabelle einzeln an. Es gibt keine Möglichkeit, ein Gesamtergebnis aller Werte zu erhalten, wie etwa bei der `SUM()`-Funktion. Sie können aber die beiden Spalten summieren und anschließend subtrahieren. Ihre Anweisung sieht folgendermaßen aus:

```
SELECT (SUM(Spalte_2) - SUM(Spalte_1)) AS Difference FROM Orders
```

Diese Anwendung führt zu dem gleichen Ergebnis.

Auch für Subtraktionsfunktionen sind finanztechnische Applikationen und Internetshopping gute Einsatzgebiete.

Multiplikation

Die Multiplikationsfunktion arbeitet genauso wie die vorherigen Funktionen. Das Multiplikationszeichen ist wie in den meisten Programmiersprachen der Asterisk (*). Eine Multiplikationsfunktion sieht zum Beispiel folgendermaßen aus:

```
SELECT (Spalte_1 * Spalte_2) AS Product FROM Orders
```

Diese Anweisung gibt für jede Zeile der Tabelle das Produkt der beiden Spaltenwerte zurück. Um das Produkt mehrerer Spalten zu erhalten, müssen Sie diese summieren und anschließend multiplizieren. Diese Anweisung sieht beispielsweise folgendermaßen aus:

```
SELECT SUM(Spalte_1 * Spalte_2) AS Product FROM Orders
```



Vergessen Sie nicht, dass MySQL sich an mathematische Regeln hält. Die Operationen in Klammern werden zuerst ausgeführt, dem folgten Multiplikation, Division, Addition und zuletzt Subtraktion. Denken Sie daran, wenn Sie Ihre Berechnungen ausführen. Ein Beispiel: $2+3*5 = 17$ und nicht 25.

Division

MySQL kann auch Divisionen ausführen, dabei wird wie bei den bereits vorgestellten Funktionen vorgegangen. Eine Divisionsfunktion sieht zum Beispiel folgendermaßen aus:

```
SELECT (Spalte_1 / Spalte_2) AS Result FROM Orders
```

Falls Sie versuchen, durch Null zu dividieren, wird ein NULL-Wert zurückgegeben. Bruchwerte werden als Dezimalbruch zurückgegeben.

Modulo

Die Modulo-Funktion gibt den Rest der Division von zwei Zahlen zurück. Teilen Sie zum Beispiel 14 durch 3, ist der Rest 2. Das ist es, was die MOD()-Funktion zurückgibt.

Die MOD()-Funktion erwartet zwei Argumente. Das erste ist der Zähler, das zweite der Nenner. Um das Beispiel von eben auszuführen, muss die Anweisung zum Beispiel folgendermaßen aussehen:

```
SELECT MOD(14, 3) as Result
```

Das Ergebnis lautet:

```
Result
2
```

Wie Sie sehen, unterscheidet sich diese Funktion von den anderen. Versuchen Sie, einen Wert durch Null zu teilen, (MOD(4,0)), gibt MySQL einen Null-Wert zurück. Wie bei anderen Programmiersprachen auch trennen Sie Argumente innerhalb einer Funktion durch Komma. Zur Ausführung dieser Funktion benutzt MySQL die Programmiersprache C. Wenn Sie wollen, können Sie Ihre eigenen Funktionen erstellen. Tag 18, »MySQL im Vergleich«, behandelt von Benutzern definierte Funktionen.

Dies waren die grundlegenden numerischen Funktionen. Wie Sie sehen, sind sie relativ einfach zu benutzen. Im Anhang B finden Sie weitere MySQL-Funktionen.

Logische und konditionale Funktionen

Dieses Kapitel behandelt logische und konditionale Funktionen und wie sie von MySQL eingesetzt werden.

IF(), IFNULL() und CASE

MySQL stellt Ihnen eine Möglichkeit zur Verfügung, um Ausdrücke konditional zu überprüfen. Es ist ein wenig schwierig, aber es funktioniert. Diese Funktionen haben drei Argumente. Ist das erste Argument wahr, wird das zweite zurückgegeben; ist es falsch, wird das dritte zurückgegeben. Vergleichen Sie beispielsweise Spalte 1 mit Spalte 2. Ist Spalte 1 größer als Spalte 2, sollte MySQL an Ihr Programm eine 1 zurückgeben. Ist Spalte 2 größer, sollte eine 0 zurückgegeben werden. Um dies auszuführen, benutzen Sie folgende Anweisung:

```
SELECT IF(Spalte1 > Spalte2, 1, 0) FROM Orders
```

Beinhaltet Spalte1 beispielsweise den Wert 12 und Spalte2 den Wert 3, so gibt diese Anweisung 1 (das zweite Argument) zurück. Beinhaltet Spalte1 den Wert 3 und Spalte2 den Wert 12, so gibt diese Anweisung 0 (das dritte Argument) zurück. Es kann vielleicht etwas dauern, bis Sie sich an diese Vorgehensweise gewöhnen.

Diese Funktion ist recht nützlich beim Vergleich von Spalten. Sie fragen sich vielleicht, weshalb Sie diese Funktion an Stelle der WHERE-Klausel benutzen sollten. Diese Funktion arbeitet einfach schneller als die WHERE-Klausel, bei deren Gebrauch MySQL die gesamte Anweisung parsen, einen Anweisungsplan generieren und die passenden Schlüssel auswählen muss, um anschließend die Anweisung auszuführen. Wenn Sie eine Funktion benutzen, wird kein Anweisungsplan erstellt und die Ergebnisse können sehr viel schneller zurückgegeben werden. Bei Vergleichen sollten Sie diese Funktionen und nicht die WHERE-Klausel benutzen.

Müssen Sie mehr als einen Wert vergleichen, stellt MySQL die CASE-Anweisung bereit, die der die CASE-Anweisung der Programmiersprache C sehr ähnelt. Die Syntax für diese Anweisung sieht folgendermaßen aus:

```
SELECT CASE Wert WHEN Vergleich THEN Ergebnis_A ELSE Ergebnis_B
```

Im letzten Beispiel könnten Sie die CASE- Anweisung folgendermaßen benutzen:

```
SELECT CASE 1 WHEN Spalte1 > Spalte2 THEN 0 ELSE 1
```

Die CASE- Anweisung ermöglicht Ihnen, mehrfache Vergleiche durchzuführen. Wollen Sie zum Beispiel die Werte von drei Spalten vergleichen und überprüfen, ob eine Spalte den Wert »Y« enthält, dann können Sie folgende Anweisung benutzen:

```
SELECT CASE 1 WHEN Spalte1 = "Y" THEN 1 WHEN Spalte2 = "Y" THEN 2  
WHEN Spalte3 = "Y" THEN 3 ELSE "NONE"
```

Diese Anweisung gibt die erste Spalte an, die den Wert `Y` enthält. Falls keine der beiden Spalten diesen Wert enthält, wird `NONE` zurückgegeben. Wie Sie sehen, ist dies eine sehr nützliche Funktion, wenn Sie mehrfache Vergleiche ausführen müssen. Da auch hier kein Anweisungsplan erstellt werden muss, wird der Vergleich auf diese Art schneller ausgeführt.

Die `IFNULL()`-Funktion ist eine weitere Funktion, mit der Sie Vergleiche durchführen können. Diese Funktion gibt das erste Argument zurück, wenn dieses nicht `NULL` ist; das zweite Argument wird zurückgegeben, wenn das erste `NULL` ist. Wenn Sie sich vergewissern wollen, dass Ihre Divisionsfunktion arbeitet und sie nicht aus Versehen durch `NULL` teilen, benutzen Sie folgende Anweisung:

```
SELECT INFULL((Spalte1 / Spalte2), 0))
```

Wenn `Spalte2` keinen `NULL`-Wert enthält, wird das Ergebnis der Division zurückgegeben. Falls der Wert von `Spalte2` doch `NULL` ist, so wird `0` zurückgegeben. Setzen Sie einfache Zahlen ein, um den Sachverhalt zu klären:

```
SELECT IFNULL((12 / 2), 0) as Result
```

Ihr Ergebnis lautet:

```
Result  
6
```

Wenn etwas nicht stimmen würde und Sie geben

```
SELECT IFNULL((12 / 0), 0) as Result
```

ein, ist das Ergebnis:

```
Result  
0
```

Haben Sie die `IFNULL()`-Funktion nicht benutzt und versucht, durch Null zu teilen, so wird ein `NULL`-Wert zurückgegeben. Wenn Sie den Fehler erkennen wollen, bevor es zu größeren Problemen kommt, benutzen Sie die `IFNULL()`-Funktion.

Die `IFNULL()`-Funktion ist eine gute Methode, um unerwartete Ergebnisse zu überprüfen. Sie ist sehr wichtig, wenn Sie nicht wissen, welche Werte in einer Spalte eingegeben werden.

AND, OR, NOT

MySQL enthält noch einige andere Operatoren. Diese geben eine `1` zurück, wenn die zu vergleichenden Werte gültig sind, und eine `0`, wenn diese falsch sind.

Die `AND`-Funktion bzw. der `AND`-Operator (je nachdem wie Sie es sehen) gibt einen wahren Wert zurück, wenn beide Werte gültig sind, und einen falschen Wert, wenn einer der Werte falsch oder `NULL` ist. Stellen Sie sich vor, Sie hätten folgende Anweisung:

```
SELECT 3 AND 4
```

Der zurückgegebene Wert wäre

1 or TRUE - both values are true (not 0 or NULL)

Was würde demnach die folgende Anweisung zurückgeben?

```
SELECT 0 AND 3
```

Haben Sie auf 0 getippt? Dann liegen Sie richtig. Beide Werte müssen gültig sein, damit der Ausdruck auch gültig ist.

Der AND-Operator kann ebenfalls, wie dies auch in anderen Programmiersprachen der Fall ist, mit && angegeben werden.

Der OR-Operator gibt einen gültigen Wert zurück, wenn einer der beiden zu vergleichenden Werte gültig ist. Zum Beispiel

```
SELECT 1 OR NULL
```

gibt den Wert 1 zurück und

```
SELECT NULL OR 0
```

gibt den Wert 0 zurück.

Der OR-Operator kann auch als || angegeben werden. Auch dieses Zeichen wird von den meisten Programmiersprachen benutzt.

Der NOT-Operator verneint die Werte des Arguments. Die Ausnahme ist NULL. NOT NULL wird als NULL zurückgegeben. Zum Beispiel

```
SELECT NOT (1 + 1)
```

gibt 0 zurück und

```
SELECT NOT (1 - 1)
```

gibt 1 zurück.

Der NOT-Operator kann auch mit Ausrufezeichen(!) angegeben werden.

Diese Operatoren können zur Ausführung logischer Berechnungen, bevor die Anfrage bei der WHERE-Klausel angelangt ist, angewandt werden. Dies macht Anfragen schneller und steigert die gesamte Performance.

Zeichenketten- und Zeichenfunktionen

Die Zeichenketten- und Zeichenfunktionen können Text bearbeiten, bevor er dem Client zurückgegeben wird. Auf diese Weise kann in Ihrer Anwendung Zeit und Verarbeitungsaufwand gespart werden.

LTRIM(), RTRIM(), TRIM()

Diese kleinen, nützlichen Funktionen entfernen überschüssige Leerzeichen am Anfang oder am Ende einer Zeichenkette. Sie sollten diese Funktionen bei allen Vergleichen von Zeichenketten benutzt werden. Die Zeichenkette " Hello" unterscheidet sich beispielsweise von der Zeichenkette "Hello". Benutzen Sie die LTRIM()-Funktion, um die führenden Leerzeichen zu entfernen, so dass bei einem neuen Vergleich LTRIM("Hello") gleich "Hello" ist.

Wie schon gesagt, beinhaltet die LTRIM()-Funktion ein Argument. Dieses ist die Zeichenkette, von der Sie alle führenden Leerzeichen entfernen wollen. Zum Beispiel gibt

```
SELECT LTRIM("  Mark")
"Mark"
```

zurück, aber

```
SELECT LTRIM("Mark  ")
```

gibt

```
"Mark  "
```

zurück.

Um die hinten stehenden Leerzeichen zu entfernen, müssen Sie die RTRIM()-Funktion anwenden. Sie funktioniert genauso wie ihr Gegenstück LTRIM() mit dem Unterschied, dass sie die hinteren und nicht die vorderen Leerzeichen entfernt. Zum Beispiel gibt

```
SELECT RTRIM("Mark  ")
"Mark"
```

zurück.

Die TRIM()-Funktion kann ein wenig mehr. Sie entfernt sowohl führende als auch folgende Leerzeichen. Zum Beispiel gibt

```
SELECT TRIM("  Mark  ")
"Mark"
```

zurück.

Die TRIM()-Funktion kann sowohl Leerzeichen als auch Zeichen oder Zeichengruppen entfernen. Zum Beispiel gibt

```
SELECT TRIM(TRAILING 'XXX' FROM "FileName.XXX")
"FileName"
```

zurück.

Ebenso können Sie führende Zeichen bestimmen. Zum Beispiel gibt

```
SELECT TRIM(LEADING 'XXX' FROM "FileName")  
"FileName"
```

zurück.

Möchten Sie vorne und hinten stehende Zeichen entfernen, benutzen Sie das Schlüsselwort `BOTH`. Zum Beispiel gibt

```
SELECT TRIM(BOTH 'XXX' FROM "XXXFileNameXXX")  
"FileName"
```

zurück.

SUBSTRING()

Die `SUBSTRING()`-Funktion beinhaltet zwei Argumente. Das erste ist eine Zeichenkette, das zweite eine Zahl. Diese Funktion gibt eine Zeichenkette ab der Stelle zurück, die im zweiten Argument angegeben ist. Wollen Sie zum Beispiel alles, was in `Hello World` nach `Hello` steht, zurückgeben, können Sie die `SUBSTRING()`-Funktion folgendermaßen verwenden:

```
SELECT SUBSTRING("Hello World", 7)
```

Diese Anweisung gibt `World` zurück.

Sie können diese Funktion vielfach einsetzen. Sie ist noch nützlicher, wenn sie mit anderen Funktionen, wie zum Beispiel der `INSTR()`-Funktion kombiniert wird. Genauereres dazu finden Sie in Anhang B.

LOCATE()

Die `LOCATE()`-Funktion beinhaltet drei Argumente. Das erste ist die Zeichenkette, die Sie finden wollen. Das zweite ist die Zeichenkette, die möglicherweise die gesuchte Zeichenkette beinhaltet, und das dritte und letzte gibt die Stelle an, von der aus Sie Ihre Suche beginnen möchten. Wollen Sie zum Beispiel das Wort »brown« in der Zeichenkette `"The quick brown fox jumped over the lazy dog"` finden, so benutzen Sie folgende Anweisung:

```
SELECT LOCATE("brown","The quick brown fox jumped over the lazy dog", 1);
```

Diese Anweisung gibt 10 zurück, da dies die erste Stelle ist, an der die gesuchte Zeichenkette gefunden wurde. Durch Kombination von Funktionen können Sie noch mehr aus ihnen herausholen. Sie können zum Beispiel die `SUBSTRING()`- mit der `LOCATE()`-Funktion verbinden, um eine Zeichenkette zurückzugeben, die in einem `BLOB`-Datentyp vorhanden sein könnte. Das sieht folgendermaßen aus:

```
SELECT SUBSTRING(Spalte1, LOCATE("Doktor", Spalte1), 1))
```

Die Ausgabe dieser Funktion ist alles, was in dieser Spalte nach »Doktor« steht. Nehmen wir beispielsweise an, dass Spalte1 den Wert "Dr.Ramirez" enthält und Sie lediglich seinen Nachnamen wissen wollen. Die Syntax hierfür sieht folgendermaßen aus:

```
SELECT SUBSTRING("Dr.Ramirez", (LOCATE(".", "Dr.Ramirez",1) + 2))
```

Die LOCATE()-Funktion rechnet bis 3. Zwischen dem Titel und dem Namen befindet sich eine Leerstelle. Die Funktion beginnt beim zweiten Argument zu zählen. Um lediglich den Nachnamen zu erhalten, müssen Sie dem Ergebnis der LOCATE()-Funktion 2 hinzufügen. Daraus ergibt sich die Anfangsstelle des Nachnamens.

Wie bereits erwähnt, können Sie durch das Kombinieren von Funktionen mehr aus diesen herausholen. Damit können Sie in Ihren Applikationen Zeit und Logik einsparen.

REPLACE()

Die REPLACE()-Funktion ersetzt eine Zeichenkette durch eine andere. Sie beinhaltet drei Argumente. Das erste ist die Haupt-Zeichenkette, das zweite die Zeichenkette, die ersetzt werden soll, und das dritte die Zeichenkette, die die vorherige ersetzt. Haben Sie zum Beispiel die Zeichenkette "FileName.XXX" und Sie wollen "XXX" durch "123" ersetzen, sieht Ihr Funktionsaufruf folgendermaßen aus:

```
SELECT REPLACE("FileName.xxx", "xxx", "123")
```

Das Ergebnis lautet:

```
"FileName.123"
```

REVERSE()

Die REVERSE()-Funktion dreht die als Argument gegebene Zeichenkette einfach um. Stellen Sie sich vor, Sie müssten die Zeichenkette "Hello" umdrehen. Ihre Anweisung lautet in diesem Fall:

```
SELECT REVERSE("Hello")
```

Das Ergebnis dieser Funktion lautet:

```
"o1leH"
```

UCASE(), LCASE()

Die UCASE()- und LCASE()-Funktionen geben bei jeder als Argument angegebenen Zeichenkette alles Großgeschriebene klein wieder und umgekehrt. Zum Beispiel:

```
SELECT UCASE("mark")
```

ergibt

"MARK"

und

```
SELECT LCASE("MARK")
```

gibt

"mark"

zurück.

Wie Sie sehen, besitzt MySQL viele nützliche und weniger nützliche Funktionen, um Zeichenketten zu bearbeiten. Diese können auf der Client-Ebene Zeit sparen und ermöglichen Ihnen, robuste Berichte zu erstellen, ohne dabei den Datenbank-Server zu verlassen.

Zusammenfassung

MySQL stellt Ihnen zahlreiche interne Funktionen bereit, die Ihnen, dem Datenbank-Verwalter oder Programmierer, ermöglichen, komplexe Aufgaben innerhalb der Datenbank auszuführen. Sie haben heute die Vor- und Nachteile kennen gelernt, die entstehen können, wenn die Logik auf dem Datenbank-Server platziert wird. Zudem wurden einige Richtlinien vorgestellt, mit deren Hilfe Sie bestimmen können, wo Sie Ihre Applikations-logik am besten platzieren können. Außerdem wurde auf die verschiedenen internen Funktionen von MySQL eingegangen.

Fragen und Antworten

- F** MySQL besitzt viele nützliche Funktionen. Was aber, wenn ich eine Funktion benötige, die MySQL nicht beinhaltet?
- A** *Da MySQL sehr flexibel ist, können Sie Ihre eigenen Funktionen erstellen. Diese so genannten benutzerdefinierten Funktionen arbeiten genauso wie die internen Funktionen von MySQL. Außerdem ist es möglich, Ihre Funktionen in die Applikationen zu rekompilieren, so dass Sie diese immer zur Verfügung haben, egal, wie oft Sie installieren. An Tag 18 erhalten Sie einen Überblick über benutzerdefinierte Funktionen sowie ein Beispiel, wie diese erstellt werden.*
- F** Wozu gibt es so viele Funktionen?
- A** *Dafür gibt es sehr viele verschiedene Gründe. Der wichtigste Grund besteht darin, dass MySQL einen Berichtsmechanismus für seine Daten bereitstellen*

sollte. Wie Sie wissen, kann das Erstellen von Berichten ziemlich schwierig werden – dafür werden unter anderem diese vielen Funktionen bereitgestellt. Ein weiterer Grund, weshalb MySQL so viele Funktionen beinhaltet, ist Kompatibilität. Sie werden in Ihrer alten Datenbank einige Abfragen haben, die interne Funktionen beinhalten. MySQL hat versucht, möglichst viele der geläufigsten internen Funktionen einzufügen, damit Sie beim Wechseln von Ihrer alten Datenbank auf MySQL möglichst wenig Probleme haben.

Übungen

1. Wenden Sie einige der heute gelernten Funktionen in Abfragen an. Benutzen Sie dabei die Meet_A_Geek-Datenbank.
2. Wie erhalten Sie die Gesamtzahl der am 1. März 2000 durchgeführten Transaktionen?



Sperren und Schlüssel in MySQL

**Woche
2**

Dieses Kapitel stellt die zwei Schlüsselkonzepte des Datenbank-Designs vor. Zum einen sind dies Sperren (Locks), die wichtig sind, um die Integrität der Daten in einer Multi-Threaded-Umgebung zu gewährleisten, zum anderen Schlüssel, die sowohl beim Entwurf, als auch bei der Performance-Verbesserung Ihrer Datenbank eine entscheidende Rolle spielen.

Heute lernen Sie Folgendes:

- ▶ was eine Sperre ist und welche Arten von Sperren MySQL unterstützt
- ▶ wie und wann man Sperren verwendet
- ▶ was Schlüssel sind
- ▶ die Konzepte von Primär-, eindeutigen und nicht- eindeutigen Schlüsseln
- ▶ die Bedeutung von Primärschlüsseln für die Datenbankstruktur
- ▶ die Vor- und Nachteile der Verwendung von Schlüsseln in Bezug auf Performance

Was ist eine Sperre?

Stellen Sie sich folgende Situation vor: In Ihrer Datenbank herrscht reger Betrieb, so rege, dass sie von vielen Benutzern nahezu gleichzeitig benutzt wird. Während einige Threads versuchen, Daten abzufragen, müssen andere Daten sowohl abfragen und bearbeiten als auch anschließend zurückschreiben.



Was ist ein Thread? Sie können sich einen *Thread* als eine Art Sub-Prozess des MySQL-Daemons vorstellen. Bei jeder Anfrage initiiert MySQL einen Thread. Mit Beendigung der Anfrage endet auch der Thread. Auf diese Weise können mehrere Personen gleichzeitig auf dieselbe Tabelle zugreifen, ohne sich dabei in die Quere zu kommen.

Obwohl Threads vom Prinzip her Unix-Prozessen gleichen, sind sie dennoch nicht ganz das Gleiche, da der MySQL-Daemon zwar ein einzelner Unix-Prozess ist, aber viele Threads beinhalten kann.

Sie haben zum Beispiel eine Tabelle, die den Inhalt eines Lagers enthält. Für jedes Produkt beinhaltet sie ein `qty_in_stock`-Feld (Artikelmenge). Bestellt nun ein Kunde einen Artikel, so überprüft die Server-Applikation, wie oft sich dieser Artikel im Lager befindet (vorausgesetzt die Anzahl ist größer als die Bestellung des Kunden), und führt dann die Bestellung aus. Schließlich ziehen Sie die verschickte Artikelmenge vom Lagerinhalt ab und aktualisieren Ihre Tabelle dementsprechend:

```
SELECT qty_in_stock FROM Gadgets WHERE gadget_id=some_gadget_id;
```

```
...
```

```
#An dieser Stelle befindet sich eine Überprüfungsroutine, die den
#Lagerbestand mit der Bestellmenge vergleicht und, wenn genug auf Lager
#ist, weitermacht ...
```

```
...
```

```
UPDATE Gadgets SET qty_in_stock=(qty_from_previous_statement-qty_purchased) WHERE
gadget_id=some_gadget_id;
```

Während dieser Programmcode ausgeführt wird, besteht immerhin die Möglichkeit, dass ein anderer Kunde zur gleichen Zeit das gleiche Produkt erwerben möchte. Ein zweiter Thread würde dann den gleichen Code ausführen, wobei jedoch die beiden Threads sämtliche Berechnungen durcheinander bringen und folglich falsche Daten in die Tabelle eingetragen würden (siehe Abbildung 11.1).

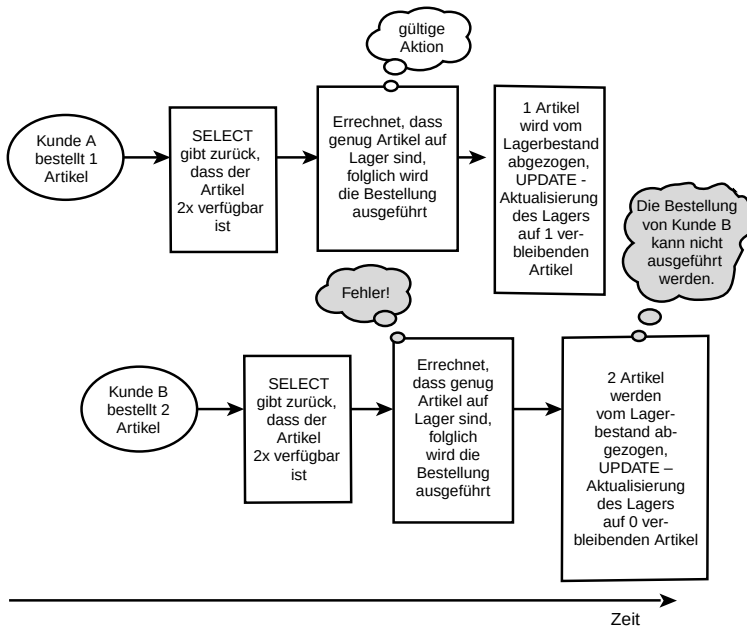


Abbildung 11.1:
Prozessablauf
ohne Sperren

Das kann natürlich zu Ärger mit dem Kunden führen. Sie versprechen ihm Ware, können sie ihm aber nicht geben, da das Lager bereits leer ist.

Bei solch einem Problem ist eine Sperre von großem Nutzen. Das Sperren sichert einem Thread den alleinigen Zugriff auf einige Tabellen, so dass es nicht zur Vermischung mit anderen Threads kommen kann. Ist die Aktion beendet, wird die Sperre aufgehoben, so dass auch andere Threads an die Daten gelangen können.

Zum Sperren Ihrer Tabelle benutzen Sie folgenden Programmcode:

```
LOCK TABLES Gadgets WRITE;
SELECT num_in_stock FROM Gadgets WHERE gadget_id=some_gadget_id;
...
#An dieser Stelle befindet sich eine Überprüfungsroutine, die den
#Lagerbestand mit der Bestellmenge vergleicht und, wenn genug auf Lager
#ist, weitermacht ...
...
UPDATE Gadgets SET num_in_stock=(num_from_previous_statement-num_purchased) WHERE
gadget_id=some_gadget_id;
UNLOCK TABLES;
```

Jetzt ist Ihr Geschäftsablauf etwas besser organisiert, da der Thread für den zweiten Kunden warten muss, bis der Thread für den ersten Kunden fertig ist (siehe Abbildung 11.2).

In diesem doch sehr einfachen Beispiel haben Sie lediglich eine Tabelle benutzt. In den meisten Fällen, wie beispielsweise bei einer Reihe von aufeinander bezogenen Anfragen, betrifft eine Anfrage mehr als nur eine Tabelle. In beiden Fällen müssen Sie die Tabelle/n, die Sie benutzen, sperren.

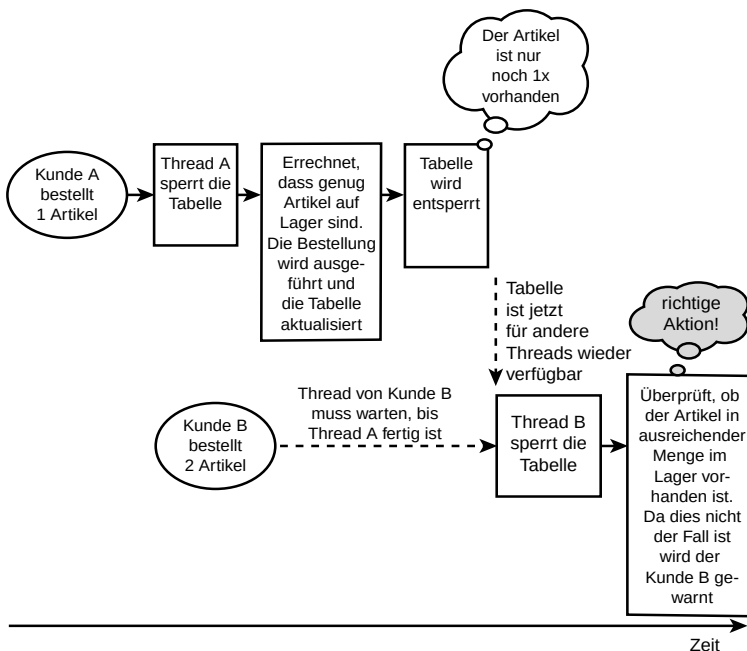


Abbildung 11.2:
Sicherer Prozess-
ablauf mit
Sperren

Stellen Sie sich Ihren Code ein bisschen anspruchsvoller vor. Sie wollen beispielsweise, diesen Verkauf samt Namen und Adresse des Kunden einer Transaktions-Tabelle hinzufügen.

Ihr Code könnte dann folgendermaßen aussehen:

```
LOCK TABLES gadgets WRITE, transactions WRITE, customer READ;
SELECT num_in_stock, price FROM gadgets WHERE gadget_id=some_gadget_id;
...
#An dieser Stelle befindet sich eine Überprüfungsroutine, die den
#Lagerbestand mit der Bestellmenge vergleicht und, wenn genug auf Lager
#ist, weitermacht ...
...
UPDATE gadgets SET num_in_stock=(num_from_previous_statement-num_purchased) WHERE
gadget_id=some_gadget_id;
SELECT cust_name, address, zip, from customer WHERE cust_id=this_customer_id;
INSERT INTO transactions VALUES (cust_name, address, zip, price, qty);
UNLOCK TABLES;
```

Beachten Sie, dass Sie bei den `gadgets`- und `transactions`-Tabellen `WRITE`-Sperren, bei der `Customers`-Tabelle hingegen `READ`-Sperren verwendet haben. Der Grund hierfür ist, dass Sie in die `Customers`-Tabelle nichts schreiben müssen. Obwohl Sie diese gesperrt haben, können andere Threads die `Customers`-Tabelle lesen.

Sie müssen sich darüber klar werden, dass es absolut notwendig ist, alle Tabelle zu sperren, die während der Ausführung des Programmcodes benötigt werden. Dadurch kann es niemals zu einem sogenannten *Deadlock* kommen. Eine *Deadlock*-Situation tritt ein, wenn ein Thread nur eine Tabelle aus einem Satz von Tabellen gesperrt hat und ein anderer Thread gleichzeitig eine andere dieser Tabellen sperrt. Beide Threads würden mit der Verarbeitung fortfahren. Nehmen wir jetzt an, dass Thread A Tabelle 1 und Thread B Tabelle 2 sperrt. Wenn jetzt aber Thread B ebenfalls Tabelle 1 (die er nicht sperren konnte) benutzen wollte, müsste er warten, bis Tabelle 1 von Thread A freigegeben wird. Es kann sein, dass dies niemals geschieht, weil Thread A vielleicht gerade pausiert und wartet, dass Thread B Tabelle 2 freigibt. So sieht eine *Deadlock*-Situation aus, die einer *Gridlock*-Situation in Ihrer Datenbank entspricht.

Der Gebrauch von Sperren in MySQL

Die Syntax für `LOCK TABLES` sieht folgendermaßen aus:

```
LOCK TABLES Tabellennamen1 (AS alias) {READ | [LOW_PRIORITY] WRITE} [,
Tabellennamen2 {READ | [LOW_PRIORITY] WRITE} ...]
```

Wie Sie sehen, kann die `LOCK TABLES`-Anweisung eine Liste von Tabellen entgegennehmen, einige mit `WRITE`-Sperren und einige mit `READ`-Sperren.

Es gibt einige einfache Regeln, wie MySQL `READ`- und `WRITE`-Sperren handhabt:

- ▶ Erhält ein Thread eine `READ`-Sperrung, kann dieser Thread aber auch alle anderen Threads, die entsprechenden Tabellen abfragen.
- ▶ Erhält ein Thread eine `WRITE`-Sperrung, ist dieser der einzige Thread, der Zugang zu den betroffenen Tabellen hat. Er kann darin lesen und schreiben. Die anderen Threads haben erst wieder Zugang zu den Tabellen, wenn die Sperrung aufgehoben ist.

Eine `LOCK TABLES`-Anweisung auszuführen, um eine Sperrung zu erhalten, bedeutet noch nicht, dass die Tabelle auch wirklich für diesen Thread gesperrt wird! Bevor einem Thread eine Sperrung zugestanden werden kann, gibt es einen Warteprozess.

In der Warteschleife

Wenn ein Thread eine Anfrage für ein `LOCK` stellt, muss er vielleicht warten, bis mögliche andere Sperren für die betreffenden Tabellen freigegeben sind. Es gibt eine `WRITE`- und eine `READ`-Warteschleife, die sehr ähnlich funktionieren.

Im Folgenden wird die Prioritätsfolge dargestellt:

Wenn eine `WRITE`-Sperrung erteilt wird:

- ▶ Ist die Tabelle momentan nicht gesperrt, wird die `WRITE`-Sperrung unverzüglich erteilt.
- ▶ Ist die Tabelle gesperrt, kommt der Antrag in die `WRITE`-Sperren-Warteschleife.

Wenn eine `READ`-Sperrung erteilt wird:

- ▶ Hat die Tabelle keine `WRITE`-Sperren, wird die `READ`-Sperrung sofort erteilt.
- ▶ Sonst wird der Antrag in die `READ`-Sperren-Warteschleife gestellt.

Immer wenn eine Sperrung freigegeben wird, haben Threads in der `WRITE`-Sperren-Warteschleife Vorrang gegenüber denen in der `READ`-Sperren-Warteschleife. Beantragt ein Thread also eine `WRITE`-Sperrung, wird er diese schnellstmöglich erhalten.

Dies geschieht nicht ohne Grund. Indem den Threads, die etwas in die Tabelle schreiben wollen, Vorrang gegeben wird, stellt MySQL sicher, dass die Datenbank so schnell wie möglich aktualisiert wird.

Einem Thread in der `READ`-Sperren-Warteschleife erteilt MySQL die Sperrung erst, wenn keine Threads mehr in der `WRITE`-Sperren-Warteschleife stehen. Es gibt allerdings auch Mittel und Wege, diese Standard-Vorgehensweise zu umgehen.

LOW_PRIORITY WRITE

Sie haben vielleicht eine Applikation, für die READ-Sperren dringender sind als WRITE-Sperren.

Indem Sie `LOW_PRIORITY WRITE` erlassen, wird das Warte-System auf den Kopf gestellt: Eine WRITE-Sperre muss warten, bis es keine READ-Sperren mehr gibt; erst dann wird eine WRITE-Sperre erteilt.

Wie dem auch sein, wenn Sie eine Applikation schreiben, mit der viele READs an die Datenbank gestellt werden, sollten Sie sicherstellen, dass genügend Zeit bleibt, um WRITES auszuführen. Seien Sie auf der Hut vor Performance-Einbußen, die entstehen können, weil ein WRITE einige Zeit warten muss, bis der Thread weiter ausgeführt werden kann.

SELECT HIGH_PRIORITY

`SELECT HIGH_PRIORITY` bietet eine weitere Möglichkeit, den Warteprozess zu manipulieren. Diese Anweisung ermöglicht, die Tabelle einzulesen, auch wenn eine WRITE-Sperre in der Warteschleife vorhanden ist.

Diese Methode sollte man nur bei SELECT-Anfragen, die auf einmal eingegeben werden müssen und die schnell zu vervollständigen sind, verwenden.

Das Freigeben von Tabellen

`UNLOCK TABLES` gibt alle vom momentanen Thread gesperrten Tabellen wieder frei. Tabellen werden auch freigegeben, wenn dieser gleiche Thread einen anderen LOCK-Befehl erlässt oder die Verbindung zum Server beendet wird. Sperren werden nicht wegen Zeitüberschreitungen aufgehoben.

Der Gebrauch von LOCK TABLES

Es kann viele Gründe für den Einsatz eines LOCK geben.

Ausführen verschiedener Tabellenoperationen

Wie Sie wissen, kann ein Thread einen LOCK erlassen, wenn er sicherstellen will, dass nichts von einer SELECT- bis zu einer UPDATE-Anweisung Zugang zu Daten in bestimmten Tabellen erhalten kann. Je mehr SQL-Anweisungen, die exklusiven Zugang zu bestimmten Tabellen erforderlich machen, auf einmal ausgeführt werden müssen, desto wichtiger ist es, LOCK TABLES zu benutzen. Dies ist wahrscheinlich der wichtigste Grund, weshalb man Sperren benutzen sollte.

Beachten Sie aber, dass einzelne `UPDATE`-Anweisungen *atomar* sind – das heißt, dass kein anderer Thread die Anweisung unterbrechen kann, ganz egal wie komplex sie ist. Besteht Ihre Anfrage also aus einer einzigen `UPDATE`-Anweisung, können Sie sich eine Sperre sparen, da eine einzelne Anweisung sowieso nicht unterbrochen werden kann. Auch wenn Ihre `UPDATE`-Anweisung 100.000 Datensätze betreffen würde, könnte kein anderer Thread Zugang zu diesen Datensätzen erhalten, bis sie nicht vollständig aktualisiert wurden.

Performance

Es kann sein, dass eine Applikation mehrere Anweisungen gegen eine Anzahl von Tabellen ausführen muss. In einem solchen Fall ist die Ausführung am schnellsten, wenn Sie die betroffenen Tabellen sperren. Dies garantiert den stabilen und ungestörten Zugang zu ihnen.

Die wohl größte Auswirkung haben Sperren, wenn Sie mehrere `INSERT`-Anweisungen in Ihrem Programmcode haben. Normalerweise wird der Index-Puffer einmal pro `INSERT`-Anweisung auf die Platte geschrieben. Verwenden Sie hingegen Sperren, wird diese Ausgabe auf die Platte verzögert, bis alle `INSERT`-Anweisungen abgeschlossen sind.

Alternativen zu fehlender transaktionaler Kontrolle

MySQL fehlt das, was man transaktionale Kontrolle nennt: die Fähigkeit, die verschiedenen Transaktionen, die in einem Datenbank-Management-System auftreten, zu verwalten.

Mit transaktionaler Kontrolle beeinflussen Änderungen an der Datenbank die Ziel-Tabellen nicht sofort, auch wenn es so scheint. Vielmehr werden diese Änderungen in einem temporären Puffer gehalten, bis Sie die Anweisung geben, die Daten endgültig in die Datenbank zu schreiben.

Transaktionale Anweisungen in *ANSI SQL* sind `COMMIT`, `ROLLBACK` und `SAVEPOINT`, die mit `INSERT`-, `UPDATE`- und `DELETE`-Anweisungen verwendet werden können (mehr Informationen hierzu erhalten Sie an Tag 17).

So kann `ROLLBACK` zum Beispiel eingesetzt werden, um Änderungen rückgängig zu machen, die nicht mit der `COMMIT`-Anweisung endgültig vollzogen wurden. Ebenso wird ein `ROLLBACK` ausgeführt, wenn Sie davon ausgehen müssen, dass ein anderer Thread Ihren Update-Anweisungen in die Quere gekommen ist.

Mit Sperren verfügen Sie über eine einfache Technik, um die fehlende transaktionale Kontrolle bei MySQL wieder wettzumachen. Nachdem Sie die entsprechenden Tabellen gesperrt haben, sollten Sie Ihre Daten auf jegliche Art von Verfälschung hin über-

prüfen, die Sie dazu veranlassen würden, einen *ROLLBACK* auszuführen. Wenn alles in Ordnung ist, können Sie Ihre Aktualisierungen vornehmen. Anschließend entsperren Sie die Tabellen wieder.

Das Sichern Ihrer Datenbank

Sie wollen vielleicht ein Backup Ihrer Datenbank durchführen und dabei sicherstellen, dass Sie eine konsistente und vollständige Kopie erhalten.

Konsistenz ist sehr wichtig. Sie wollen schließlich sicher sein, dass in dem Augenblick, in dem Sie Ihre Backup-Kopie erstellen, Tabellen nicht von einem Thread verändert werden und somit nur teilweise mit dem aktuellsten Stand gesichert sind.

In dieser Situation müssten Sie einen `READ LOCK` ausführen.

Was ist ein Schlüssel?

Wenden Sie einen Schlüssel in einer Datenbank-Tabelle an, so können Sie ganz schnell herausfinden, wo sich bestimmte Informationen befinden. Auch wenn Schlüssel für den Benutzer vielleicht keine Bedeutung haben, sind sie ein wichtiges strukturelles Element und können auf die Performance einer Datenbank beträchtlichen Einfluss haben.

Prinzipielles über Schlüssel

Stellen Sie sich vor, Sie hätten eine kleine Ansammlung von einfachen »nützlichen« Daten. Sie könnten damit eine `Customers`-Tabelle ähnlich einer altmodischen Kartei, wo auf jeder Karte der Kundename sowie andere Angaben stehen, erstellen. Suchen Sie einen Kunden, so müssen Sie jede Karte der Reihe nach durchgehen. Haben Sie die gewünschte Karte gefunden, lesen Sie diese »nützlichen« Informationen wie zum Beispiel den Namen, die Adresse oder die Telefonnummer des Kunden.

In der Regel ordnen Sie Ihre Kartei nach Nachnamen. Wissen Sie den Namen des Kunden, erleichtert dies die Suche. Was aber, wenn Sie jemanden an Hand eines anderen Kriteriums finden möchten?

Natürlich könnten Sie auch Ihre MySQL-Datenbank auf diese altmodische Art errichten. Sie würden dabei aber bald auf Probleme stoßen, beispielsweise wenn Sie alle Kunden aus einer bestimmten Gegend finden wollten. Ihre Datenbank müsste die Informationen über jeden Kunden der Reihe nach durchgehen – eine sehr ineffiziente Arbeitsweise.

Diese Vorgehensweise nennt man einen *Table-Scan*. Ein Table-Scan ist der zeitraubendste aller Datenbank-Prozesse.

Hier kann ein Schlüssel oder ein Index Abhilfe schaffen. Ein *Schlüssel* ist ein Feld in der Tabelle, das Ihnen hilft, Einträge in einer Tabelle auf sehr viel effizientere Art zu finden.

Wie Schlüssel funktionieren

Auch wenn er Teil der Tabelle ist, ist ein Schlüssel doch eine Art Extrazabelle in der Datenbank. Er nimmt physischen Platz auf der Festplatte (oder auf anderen Speichermedien) der Datenbank ein und kann genauso groß wie die Haupttabelle sein – theoretisch sogar größer.

Sie definieren Ihren Schlüssel so, dass er sich auf eine oder mehrere Spalten in einer bestimmten Tabelle bezieht. Da die Daten im Schlüssel auf den Daten in der Tabelle beruhen, können Sie einen Schlüssel löschen und wieder herstellen, ohne dabei Daten zu verlieren. Sie werden sehen, weshalb Sie dies machen sollten.

Angenommen, Sie wollen die Kunden anhand des Bundesstaates, in dem sie leben, auswählen. Erstellen Sie einen Schlüssel, wie in Abbildung 11.3 gezeigt. Wenn Sie den Schlüssel für die Staaten-Spalte erstellt haben, können Sie ganz einfach herausfinden, wo sich in Ihrer Tabelle die in Kalifornien lebenden Kunden befinden. Sie können Kunden also genauso gut nach Staat wie nach Namen ordnen.

Data	Location
CA	3
CA	67
FL	1
NY	4
...	
WI	2

Location	Name	State
1	Smith	FL
2	Lowe	WI
3	Shepher	CA
4	Reilly	NY
...		
67	Chapma	CA

Abbildung 11.3:
Zugriff auf die State-Spalte der
Customers-Tabelle mit einem
Schlüssel

Das Gute an Schlüsseln

Der richtige Gebrauch von Schlüsseln kann die Performance Ihrer Datenbank beträchtlich beeinflussen. Vergleichen Sie es mit einem Buch-Index – nur wenige Seiten ermöglichen Ihnen, einen Eintrag ganz schnell zu finden. Wenn Sie das ganze Werk durchsehen müssten, könnte dies unter Umständen eine halbe Ewigkeit dauern.

In den heutigen Datenbanken existieren Schlüssel, um beim Auslesen der Datenbank die Menge an benötigten Festplattenzugriffen zu minimieren. Auch die Methode, mittels eines Schlüssels Daten zu suchen und die Frage, ob die Daten dem, was Sie suchen, entsprechen, erfordern anspruchsvolle und geeignete Algorithmen.

Schlüssel in MySQL

MySQL unterstützt folgende Befehle, um Schlüssel für eine Tabelle zu erstellen.

```
ALTER TABLE Tabellenname ADD (KEY|INDEX) Indexname (Spaltenname[,...]);  
ALTER TABLE Tabellenname ADD UNIQUE Indexname (Spaltenname [...]);  
ALTER TABLE Tabellenname ADD PRIMARY KEY Indexname (Spaltenname [...]);
```

Beachten Sie, dass *KEY* und *INDEX* Synonyme sind.

Dies sind die bevorzugten Formate, bestehenden Tabellen Schlüssel hinzuzufügen. Wegen der Kompatibilität mit anderen Implementierungen von SQL unterstützt MySQL auch die Folgenden:

```
CREATE INDEX Indexname ON Tabellenname (Spaltenname [...]);  
CREATE UNIQUE INDEX [Indexname] ON Tabellenname (Spaltenname [...]);  
CREATE PRIMARY KEY ON Tabellenname (Spaltenname,...);
```

Sie können Schlüssel auch gleich beim Erstellen einer Tabelle definieren:

```
CREATE TABLE Tabellenname (Spaltenname Feldtyp [NULL|NOT NULL], KEY  
Spaltenindex(Spaltenname));
```

Für einen Primärschlüssel (mit mehreren Optionen) müssten Sie folgende Syntax verwenden:

```
CREATE TABLE Tabellenname (Spaltenname [NULL|NOT NULL] [DEFAULT Defaultwert]  
[AUTO_INCREMENT] [PRIMARY KEY] [Referenz-Definition] ...);
```

Sie sollten hier nur auf die syntaktischen Optionen der CREATE TABLE-Syntax achten, da Sie sich bei Ihren Beispielen auf die ALTER TABLE-Syntax konzentrieren werden.

Jetzt gehen wir näher auf die verschiedenen Schlüsseltypen, die von MySQL unterstützt werden, ein.

Schlüssel für einzelne Spalten

Die grundlegende Syntax für das Erstellen eines Schlüssels sieht folgendermaßen aus:

```
ALTER TABLE Tabellenname ADD KEY Indexname (Spaltenname[,...])
```

Um einen Schlüssel für die `lastname`-Spalte Ihrer `Customers`-Tabelle zu erstellen, benutzen Sie folgende Syntax:

```
ALTER TABLE Customers ADD KEY lastn_idx (lastname);
```

oder

```
ALTER TABLE Customers ADD INDEX lastn_idx (lastname);
```

Wenn Sie jetzt

```
DESC Customers;
```

eingeben, erhalten Sie folgende Tabellenbeschreibung:

```
mysql> DESC Customers;
```

Field	Type	Null	Key	Default	Extra
lastname	varchar(30)		MUL		
firstname	varchar(30)				
address	varchar(100)				
state	varchar(30)				
country	varchar(30)				

```
5 rows in set (0.00 sec)
```

Sie sehen, dass sich hier bei `lastname` unter `Key` etwas getan hat – Sie haben soeben einen Index für eine einzelne Spalte erstellt.

`MUL` in der `Key`-Spalte zeigt Ihnen an, dass es sich hierbei um einen nichteindeutigen Schlüssel handelt. Wenn Sie prüfen möchten, welche Schlüssel existieren, geben Sie Folgendes ein:

```
SHOW KEYS FROM Customers;
```

oder

```
SHOW INDEX FROM Customers;
```

Sie werden daraufhin Folgendes sehen:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part
customers	1	lastn_idx	1	lastname	A	NULL	NULL

```
1 row in set (0.01 sec)
```

Um den Index zu löschen, geben Sie Folgendes ein:

```
ALTER TABLE Customers DROP KEY lastn_idx;
```

oder

```
ALTER TABLE Customers DROP INDEX lastn_idx;
```

Schlüssel für mehrere Spalten

Sie können einen Schlüssel für mehrere Spalten, auch *composite index* genannt, aus mehr als einer Spalte erstellen. Zum Beispiel:

```
ALTER TABLE Customers ADD KEY comp_index (lastname, state, country);
```

Wenn der Schlüssel erstellt wird, arbeitet sich MySQL von links aus über die angegebenen Spalten vor und benutzt Teile der Spalten, um den Schlüssel zu erstellen. Die folgenden Spaltenreihen würden indiziert werden:

- ▶ Familienname, Bundesstaat, Land (lastname, state, country)
- ▶ Familienname, Bundesstaat, (lastname, state)
- ▶ Familienname (lastname)

Sie können sich den composite index von MySQL anzeigen lassen:

```
mysql> SHOW TABLES FROM Customers;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part
customers	1	comp_index	1	lastname	A	NULL	NULL
customers	1	comp_index	2	state	A	NULL	NULL
customers	1	comp_index	3	country	A	NULL	NULL

3 rows in set (0.00 sec)

Beim Erstellen von Schlüsseln für mehrere Spalten ist es von Vorteil, die einschränkendste Spalte an die erste Stelle zu setzen (im letzten Beispiel ist *lastname* einschränkender als *state*, das wiederum einschränkender als *country* ist). Bei einer Abfrage-Operation trägt dies zur Performance der Datenbank bei.

Eine *WHERE*-Klausel zu erstellen, indem Sie das Feld oder die Felder links des zusammengesetzten Schlüssels benutzen, ist am effektivsten, da damit sichergestellt wird, dass MySQL den Schlüssel bei seiner Arbeit benutzt.

Um den zusammengesetzten Schlüssel zu löschen, benutzen Sie die gleiche Syntax:

```
ALTER TABLE Customers DROP KEY comp_index;
```

Schlüssel für mehrere Spalten erzeugen mehr Overhead als Schlüssel für einzelne Spalten. Denken Sie daran, wenn Sie entscheiden müssen, ob Sie einen Schlüssel für nur eine Spalte oder einen für mehrere Spalten brauchen.

Teil-Schlüssel

Erstellen Sie einen Schlüssel für einen CHAR- oder VARCHAR-Spaltentyp, ist es möglich, die ersten paar Zeichen der Spalte zu indizieren. Sie verweisen dabei auf den ersten Teil oder das Präfix einer Spalte, indem Sie die Länge des Präfix zu dem Spaltennamen hinzufügen.

Sie können zum Beispiel einen Schlüssel für die ersten sechs Zeichen eines Kundenamens erstellen:

```
ALTER TABLE Customers ADD KEY lastn_idx (lastname(6))
```

Führen Sie dies gleichzeitig mit dem Erstellen einer Tabelle aus, müssen Sie Folgendes bestimmen (was die Syntax, die sich auf (lastname) bezieht, anzeigt):

```
CREATE TABLE Customers (lastname VARCHAR(30) NOT NULL, KEY lastname_idx (lastname(6),...))
```

Geben Sie

```
mysql> SHOW KEYS FROM Customers;
```

ein, erhalten Sie:

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part
customers	1	lastn_index	1	lastname	A	NULL	6

1 row in set (0.00 sec)



Für Spalten des BLOB- und TEXT-Datentyps haben Sie keine andere Wahl, als den Teilindex zu benutzen.

Zusammengesetzte Teilschlüssel

Für mehrere Spalten können Sie Teilschlüssel erstellen. Zum Beispiel:

```
ALTER TABLE Customers ADD KEY comp_index (lastname(6),state(2),country(3));
```

Nehmen wir jetzt an, dass die Daten in Ihrer Tabelle Folgendes mit einschließen:

```
mysql> SELECT * FROM Customers;
```

lastname	firstname	address	state	country
Shepherd	Tom	33 Madison Drive, Oakland	CA	USA
Chapman	Frederick	52 Ocean St, Sacramento	CA	USA
Lowe	Justin	3 Hill Walk, Twin Creeks	WI	USA
Spokes	Chris	Red Fern House, Bradwell	Oxford	UK

4 rows in set (0.00 sec)

Ihr Schlüssel enthält dann Folgendes:

Lastname	state	Country	comp_index
Shepherd	CA	USA	SephCAUSA
Chapman	CA	USA	ChapmCAUSA
Lowe	WI	USA	Lowe WIUSA
Spokes	Oxford	UK	SpokeOxUK

Tabelle 11.1: Zusammengesetzte Teilschlüssel

Beim Formulieren von Anfragen müssen Sie beachten, dass eine Anfrage

```
SELECT * FROM Customers WHERE lastname='Chapman';
```

den Index benutzt, wenn sie ausgeführt wird. Die Anfrage

```
SELECT * FROM Customers WHERE state='WI';
```

benutzt den Index jedoch nicht, da die Indizes von (lastname, state, country), (lastname, state) und (lastname) zwar existieren, die zweite Anfrage aber versucht, nur state zu benutzen. MySQL muss also einen zusammengesetzten Index mit dem angeforderten Feld ganz links finden, um diesen zu benutzen.

In etwa das Gleiche läuft bei der Anwendung eines Teilschlüssels ab. Die Anfrage

```
SELECT * FROM Customers WHERE lastname LIKE 'Chap%';
```

benutzt den Schlüssel, da sie nach einem entsprechenden Nachnamen sucht, indem sie die am weitesten links stehenden Zeichen vergleicht. Die Anfrage

```
SELECT * FROM Customers WHERE lastname LIKE '%hap%';
```

benutzt ihn jedoch nicht, da sie nach Übereinstimmungen vor und nach dem angegebenen Text sucht.

Ganz im Sinne der Performance verbraucht die Verwendung von Teilindizes weniger Platz im Schlüssel (und somit auf der Festplatte), als wenn Sie einen Index über das ganze Feld haben. Dies spart Platz und kann genauso effektiv sein. Die meisten Namen sind zum Beispiel nach sechs Zeichen eindeutig.

Eindeutige Schlüssel

Per definitionem gestattet ein eindeutiger Schlüssel in einer Spalte nur eindeutige Werte. Konstruieren Sie eine WHERE-Klausel, um auf einen bestimmten Datensatz zuzugreifen, befördert der eindeutige Schlüssel Sie zu einem, und nur einem, entsprechenden Datensatz.

Eindeutige Schlüssel werden nicht nur wegen der Performance benutzt, sondern auch um die Unversehrtheit der Daten zu gewährleisten. MySQL verhindert das Erstellen eines zweiten Datensatzes, der die gleichen Schlüsseldaten enthält.

Falls Sie versuchen, in eine Tabelle, in der eine Spalte eindeutig ist, die gleichen Daten noch einmal einzufügen, generiert MySQL einen Fehler. Das Gleiche passiert, wenn Sie versuchen, einen bestehenden Datensatz zu aktualisieren, um eine eindeutige Spalte auf einen Wert zu setzen, der schon vergeben ist.

Versuchen Sie darüber hinaus, eine Tabelle zu verändern und einer Spalte, in der Daten bereits als nichteindeutig definiert sind, einen eindeutigen Schlüssel hinzuzufügen, generiert MySQL ebenfalls einen Fehler.

Wenn Sie den Nachnamen der Kunden als eindeutigen Schlüssel benutzen möchten (in der Hoffnung, dass dieser auch eindeutig ist), wäre Folgendes möglich:

```
ALTER TABLE Customers ADD UNIQUE lastn_index (lastname);
```

Dies wäre jedoch sehr unpraktisch.

Sie haben die Möglichkeit, mehr als nur einen eindeutigen Schlüssel erstellen. Fügen Sie einen zweiten eindeutigen Index folgendermaßen hinzu:

```
ALTER TABLE Customers ADD UNIQUE address_index (address);
```

Haben Sie die vorangehenden Befehle in dieser Reihenfolge ausgegeben, müsste Ihre Tabelle folgendermaßen aussehen:

```
mysql> desc Customers;
```

Field	Type	Null	Key	Default	Extra
lastname	varchar(30)		PRI		
firstname	varchar(30)				
address	varchar(100)		UNI		
state	varchar(30)				
country	varchar(30)				

5 rows in set (0.00 sec)

Achten Sie darauf, wie MySQL die Schlüssel benannt hat – PRI für lastname und UNI für address. Es hat automatisch lastname zum Primärschlüssel, einer besonderen Form des eindeutigen Schlüssels, gemacht.

MySQL bestärkt Sie so sehr darin, Primärschlüssel zu erstellen, dass, wenn Sie lastn_index löschen würden, es automatisch aus address einen Primärschlüssel machen würde.

Eindeutige Schlüssel, besonders Primärschlüssel, sind sehr wichtig. Sie werden in Kürze ausführlicher behandelt.

Fremdschlüssel

Fremdschlüssel (FOREIGN KEYS) werden im Moment von MySQL nicht unterstützt. Ein Teil der Syntax ist trotzdem vorhanden, um die Portierung von Code von anderen Datenbanksystemen zu erleichtern. Wie dem auch sei, die Befehle funktionieren in der aktuellen Version nicht, und nur ein Teil der Syntax wird unterstützt.

Primärschlüssel

Ein Primärschlüssel gleicht im Prinzip einem eindeutigen Schlüssel; seine Daten müssen ebenfalls eindeutig sein, doch hat er Vorrang gegenüber einem eindeutigen (*unique*) Schlüssel. In jeder Tabelle kann es nur einen Primärschlüssel geben, dessen Feldwerte niemals *Null* sein können.

Ein Primärschlüssel wird im Allgemeinen als struktureller Link, der Beziehungen zwischen verschiedenen Tabellen definiert, benutzt. Wo auch immer Sie eine Verknüpfung zwischen Tabellen herstellen wollten, Sie würden den Primärschlüssel der Tabelle verwenden. Im Gegensatz dazu werden Schlüssel, die lediglich eindeutig sind, nur wegen der Performance hinzugefügt.

MySQL verlangt von Ihnen, dass Spalten, die als Primärschlüssel spezifiziert werden, das Attribut `NOT NULL` enthalten, auch wenn es eindeutigen (*unique*) Schlüsseln sonst erlaubt ist, `NULL`-Werte zu enthalten.

Primärschlüssel sind ein wichtiger Faktor beim Datenbank-Design. Für die Erleichterung bei der Verbindung von Tabellen und des gesamten Konzepts einer relationalen Datenbank ist er essentiell. Deshalb müssen Sie darauf achten, Ihren Primärschlüssel auf Informationen zu basieren, die immer eindeutig sind.

Im letzten Beispiel konnten Sie sehen, dass *lastname* nicht unbedingt eine gute Wahl für einen Primärschlüssel ist. Sie könnten jedoch einen komplizierteren Primärschlüssel erstellen, indem Sie einige Felder kombinieren, um etwas zusammenzubauen, das dann eindeutig ist.

Primärschlüssel für mehrere Spalten

Sie haben die Möglichkeit, einen Primärschlüssel zu erstellen, der als Schlüssel für mehrere Spalten dient.

Das können Sie nicht mit Ihrer `CREATE TABLE`-Anweisung bewerkstelligen, sondern Sie müssen dazu die `ALTER TABLE`-Syntax wie folgt verwenden:

```
ALTER TABLE Customers ADD PRIMARY KEY (lastname,firstname);
```

Gleichermaßen könnten Sie einen Teilindex einsetzen:

```
ALTER TABLE Customers ADD PRIMARY KEY (lastname(6),firstname(6));
```

Um einen Primärschlüssel zu entfernen, können Sie Folgendes benutzen:

```
ALTER TABLE Customers DROP PRIMARY KEY;
```

Synthetische Schlüssel

Neben Primärschlüsseln, die an Hand eindeutiger physischer Daten erstellt werden, können Sie auch welche »erfinden«. Diese nennt man *synthetic* oder *surrogate keys*.

Sie könnten beispielsweise für einen Schlüssel den exakten Zeitpunkt der Geburt die Sozialversicherungsnummer einer Person als Basis nehmen – immer in der Annahme, dass es keine zweite Person gibt, auf die diese Daten zutreffen.

Dies liefert allerdings keine hundertprozentige Gewissheit. Kaum jemand kennt den genauen Zeitpunkt seiner Geburt, und auch wenn man es auf die Sekunde genau belegen könnte, kann es doch jemanden geben, auf den das Gleiche zutrifft. Zudem vergeben nicht alle Länder Sozialversicherungsnummern. Hier und da ist es vielleicht illegal, diese ohne guten Grund zu speichern. Häufig ist es auch so, dass Kunden Ihre Nummer einfach nicht verraten wollen.

Es ist eigentlich nicht sehr schwer, einen eindeutigen Schlüssel an Hand von Daten aus einer bestimmten Betätigung zu erhalten. Wollte ich zum Beispiel eine Datenbank für meine riesige Fotosammlung erstellen, könnte ich wahrscheinlich Zeit als Faktor, der meine Daten eindeutig bestimmt, benutzen – natürlich vorausgesetzt ich zeichne die Zeit, zu der ich jedes Foto gemacht habe, auf und mache nicht schneller Fotos, als die Feinheit des Schlüsseltyps zulässt – vielleicht eins pro Sekunde.

Nehmen wir wieder die *Customers*-Tabelle mit einem zusammengesetzten eindeutigen Schlüssel, der zum Beispiel Nachname, Vorname und Adresse enthält. Das wäre besser, aber immer noch nicht ideal, da Leute Ihren Namen ändern oder umziehen. Ist das bei einem Kunden der Fall, können Sie beim nächsten Datenzugang mit dem Gebrauch Ihres eindeutigen Schlüssels Probleme haben, diesen in den Daten zu finden. Sie stehen dann vor der Wahl, den früheren Namen oder die alte Adresse als Schlüssel beizubehalten oder Ihren Schlüssel mit den neuen Daten zu aktualisieren.

Versuchen Sie ein Schlüsselfeld zu verändern, können Sie auf noch größere Probleme stoßen. Benutzen Sie den Schlüssel, um eine *Customers*- mit einer *Orders*-Tabelle zu verbinden, müssten Sie auch in der *Orders*-Tabelle das, was sich auf den Kunden bezieht, aktualisieren.

Dabei riskieren Sie, die Integrität der Verknüpfung zu verlieren. In einer relationalen Datenbank, in der wichtige Beziehungen von Daten verloren gehen, stellt dies ein schwerwiegendes Problem dar.

In diesem Fall wäre es besser, eine eindeutige Kunden-Identifikationsnummer, auch bekannt als *synthetic key*, zu erstellen. Ein synthetischer Schlüssel kann außerhalb der Datenbank total bedeutungslos sein. Er dient lediglich dazu, einen eindeutigen und einfachen Bezug für jeden Eintrag in der Tabelle bereitzustellen. Ein synthetischer Schlüssel verändert sich nicht, egal was sonst mit den Kunden geschieht.

Um ein synthetisches `customer_number`-Feld als Primärschlüssel für Ihre Kunden zu erstellen, können Sie es folgendermaßen definieren:

```
CREATE TABLE Customers (customer_number INT(8) PRIMARY KEY NOT NULL, last_name  
VARCHAR(30), ...)
```

Dies weist MySQL an, einen achtstelligen (ausreichend für 100.000.000 Kunden) Primärschlüssel des Typs `INTEGER` zu erstellen.

Die Auswahl der Schlüssel ist nicht ganz unbedeutend. Wenn Sie zu dem Entschluss gekommen sind, dass Sie in einer Tabelle einen Schlüssel brauchen, kann es natürlich verführerisch sein, überall synthetische Schlüssel zu erstellen. Sollte man dies machen?

Bevor Sie sich also für einen synthetischen Schlüssel entscheiden, sollten Sie erst einmal nach »natürlichen« Möglichkeiten geschaut haben, denn mit synthetischen Schlüsseln riskieren Sie, dass unnötigerweise das Design Ihrer Datenbank verkompliziert und die Performance verschlechtert wird.

Sinnvolle Auswahl der Schlüssel

Wann sollten Sie einen Schlüssel benutzen, und welchen Schlüssel sollen Sie benutzen? Wenn Sie in Ihrem Datenbank-Design Schlüssel richtig einsetzen, kann dies sowohl das Design als auch die Performance verbessern. Eine falsche Wahl kann sich nachteilig auswirken.

Die folgenden Abschnitte sind ein kurzer Exkurs in den Gebrauch und die Wahl von Schlüssel.

Wann sollen Schlüssel benutzt werden?

Ob Sie in einer Spalte einen Schlüssel benutzen sollten, hängt von der Art der Anfrage, die Sie ausführen wollen, ab.

Wann sollen Schlüssel benutzt werden?

- In *WHERE-Klauseln* – Wenn Sie eine Spalte regelmäßig für Auswahlkriterien benutzen, verbessert ein Schlüssel normalerweise die Performance. Je geringer die Anzahl der Datensätze in einer Tabelle, die von einer `SELECT...WHERE`-Anweisung zu-

rückgegeben werden, desto vorteilhafter ist ein Schlüssel. Besonders wenn bei einer Anfrage ein eindeutiges Ergebnis erwartet wird, sollte ein Schlüssel für die von der WHERE-Klausel betroffene(n) Spalte(n) erstellt werden.

- ▶ In ORDER BY-*und* GROUP BY-Klauseln – Das Sortieren von Daten ist eine aufwendige Sache. Da ein Schlüssel Ergebnisse automatisch in die richtige Reihenfolge bringt, sind bei Spalten, auf die Sie eine ORDER BY-oder GROUP BY-Klausel anwenden wollen, Schlüssel sehr empfehlenswert.
- ▶ Bei MIN()- *und* MAX()-Funktionen – Um die Minimal- und Maximalwerte in einer Spalte zu ermitteln, sind Schlüssel sehr gut geeignet.
- ▶ Bei Tabellen-Joins – Der Einsatz von Schlüssel unterstützt immer die Performance einer Datenbank, wenn dabei die indizierten Spalten zum Verbinden von Tabellen eingesetzt werden. Normalerweise sollten die meisten, wenn nicht sogar alle Spalten, die irgendwann in einer Tabellenverknüpfung benutzt werden, indiziert werden.

Wann sollen Schlüssel nicht benutzt werden?

Auch wenn Schlüssel bei SELECT-Operationen von großem Vorteil sein können, kann es zu Nachteilen kommen.

Beim Gebrauch von Tabellen in regelmäßigen Schreiboperationen – Immer wenn Sie in einer Tabelle eine Schreiboperation wie zum Beispiel eine INSERT-, UPDATE- oder DELETE-Anweisung ausführen, müssen sowohl die Haupttabelle als auch der Schlüssel in diese geschrieben werden. Hier bedeutet ein Schlüssel Overhead. In einem System, in dem Sie regelmäßig Schreiboperationen ausführen, müssen Sie diesen Overhead berücksichtigen und versuchen, ihn mit den Vorteilen, die bei Leseoperation entstehen, auszubalancieren.

Wenn eine SELECT-Anweisung eine große Menge an Datensätzen zurückgibt – Sie denken natürlich nicht im Traum daran, Wörter wie Er, Sie oder Es in einem Buchindex aufzulisten. Genauso wenig verwenden Sie einen Schlüssel für die »Kundenanrede«, da eine WHERE-Bedingung, die unter »Herr« und »Frau« auswählt, eine zu große Menge an Datensätzen ausgibt. In einem solchen Fall verschlechtert sich die Datenbank-Performance, da die Datenbank bei jeder gefundenen Übereinstimmung sowohl den Index als auch die Tabellendaten einlesen müsste. Selbst ein Table-Scan wäre schneller als die Ausführung solcher Lesevorgänge bei einer so hohen Menge an Datensätzen.

Kleine Tabellen – Es entstehen wenig Vorteile, wenn Sie Schlüssel bei kleinen Tabellen verwenden. Auch hier wäre ein Table-Scan genauso schnell.

In MySQL ist es nicht erlaubt, einen Schlüssel in Spalten mit NULL-Werten zu verwenden.

In MySQL kann eine Tabelle nicht mehr als 16 Schlüssel enthalten.

Wann sollen Primär- und eindeutige Schlüssel benutzt werden?

Im Allgemeinen sollte jede Tabelle in einer Datenbank, in der Sie auf einen Datensatz in eindeutiger Weise zugreifen möchten, einen Primärschlüssel enthalten.

Die AUTO_INCREMENT-Option

Nehmen Sie an, Sie verwenden zur eindeutigen Identifizierung eines Kunden in Ihrer Datenbank eine Kundennummer als synthetischen Schlüssel. Sie könnten Ihre CREATE TABLE-Anweisung wie folgt schreiben:

```
CREATE TABLE Customers (  
  customer_number INT(8) AUTO_INCREMENT PRIMARY KEY NOT NULL,  
  last_name VARCHAR(30),  
  ... )
```

Beachten Sie, dass Sie für die `customer_number` die `AUTO_INCREMENT`-Option angegeben haben. Dadurch wird die Nummer des Schlüssels bei jedem neuen Datensatzeintrag um eins erhöht, so dass keine identischen Schlüssel entstehen können.

Die `AUTO_INCREMENT`-Option kann in jeder Spalte, die vom Typ `INTEGER` ist, verwendet werden (das beinhaltet die Datentypen `TINYINT`, `SMALLINT`, `INT` und `BIGINT`). Wenn Sie während einer `INSERT`- oder `UPDATE`-Anweisung einen `NULL`- oder `0`-Wert in ein Feld dieses Typs einfügen, macht MySQL den Schlüsselwert des neuen Datensatzes automatisch um eins größer als den momentan größten Wert in dieser Spalte.

Der zusätzliche Gebrauch des `PRIMARY KEY`- und/oder `UNIQUE`-Attributs gewährleistet die Integrität Ihres Schlüssels.

Das Löschen eines Schlüssels

Wenn Ihre Datenbank viel Zeit für Leseoperationen verwendet und hin und wieder Aktualisierungen mit vielen Schreiboperationen erfolgen, könnte es günstig sein, Schlüssel zu verwenden, die vor der Aktualisierung gelöscht und sofort danach wieder eingefügt werden.

Sie löschen einen Schlüssel mit folgender Syntax:

```
ALTER TABLE Customers DROP INDEX lastn_idx;
...
(Ausführung der Schreib-Operationen)
...
CREATE INDEX lastn_idx ON Customers(lastname);
```

Unter Umständen möchten Sie einen Index löschen, um mit der Datenbank-Performance zu experimentieren. Ist ein Index gelöscht, kann sich die Performance verbessern oder verschlechtern.

Das Löschen eines Indexes entfernt keine Daten. Nichtsdestotrotz sollte mit dem `DROP`-Befehl immer sehr vorsichtig umgegangen werden, da eine `DROP TABLE`-Anweisung katastrophale Folgen haben kann.

Zusammenfassung

Dieses Kapitel hat Sie in die Funktion von Sperren eingeführt. Sie wissen jetzt, dass Sperren in einer Datenbank-Umgebung, die multi-threaded ist, möglicherweise benötigt werden, um einem Thread die Verarbeitung mehrfacher Anfragen zu ermöglichen, ohne dabei von einem anderen Thread gestört zu werden, bis die Anfrage beendet ist. Sie haben erfahren, wie `READ`-Sperren andere Threads davon abhalten, in die gesperrten Tabellen zu schreiben, während `WRITE`-Sperren anderen Threads ganz und gar den Zugang zu den Daten versperren.

Sie wissen nun, wie MySQL Sperren-Anfragen in eine Warteschlange stellt und Sperren nach bestimmten Regeln zuteilt.

Außerdem haben Sie gesehen, wie die Performance durch den Gebrauch von Sperren beeinflusst werden kann und wie Sperren eingesetzt werden können, um die fehlende transaktionale Kontrolle von MySQL auszugleichen.

Schlüssel bzw. Indizes sind noch wichtiger als Sperren und spielen nicht nur in MySQL, sondern in allen relationalen Datenbanken eine entscheidende Rolle. Die einfachste Aufgabe, die Schlüssel erfüllen, ist das effektive Suchen nach Daten. Primärschlüssel sind die Eckpfeiler des Datenbank-Designs.

Es gibt verschiedene Schlüssel: eindeutige oder nicht-eindeutige, für einzelne und mehrere Spalten. Sie haben gesehen, dass die gesamte oder nur ein Teil der Spalte indiziert werden kann.

Sie wissen jetzt auch, wie und ob Sie Schlüssel anwenden sollten. Sie können Schlüssel benutzen, um die Datenbankarbeit effizienter zu machen, um die Performance beim Auswählen von eindeutigen Ergebnissen zu unterstützen oder um Daten zu sor-

tieren bzw. zu gruppieren. Sie haben allerdings auch den unnötigen Einsatz von Schlüsseln kennen gelernt und wie es dabei beispielsweise bei aktualisierten Tabellen zur Verschlechterung der Performance kommen kann.

Eindeutige Schlüssel fördern nicht nur die Performance, sondern gewährleisten zudem die Integrität der Daten. Eindeutige Schlüssel können an Hand der vorhandenen Daten oder als synthetischer Schlüssel, unabhängig von diesen Daten, erstellt werden. Beim Erstellen von eindeutigen Schlüsseln stellt Ihnen MySQL bei der Definition einer numerischen Spalte die `AUTO_INCREMENT`-Option zur Seite, die Sie verwenden können, um eindeutige synthetische Schlüsselwerte zu erstellen.

Schließlich haben Sie gelernt, inwiefern Primärschlüssel eine Sonderform von eindeutigen Schlüsseln und wie wichtig sie für die Beziehungen zwischen allen Tabellen sind. Sie müssen eindeutig und dürfen nicht `NULL` sein, damit garantiert jeder Datensatz durch seinen Primärschlüssel identifiziert wird.

Fragen und Antworten

F Wann muss ich mich mit Sperren befassen?

A *Sperren sind von Bedeutung in einer Umgebung, die multi-threaded ist, in der Threads Daten in einer Datenbank aktualisieren, einfügen oder löschen. Gibt es gleichzeitig andere Threads, die aus der Datenbank lesen, sollte man den Gebrauch von Sperren in Betracht ziehen.*

Besonders wenn ein Thread mehrere SQL-Anweisungen ausführen muss, um seine Aufgabe zu erfüllen, sollten Sie eine Sperre verwenden, um die Integrität der Daten zu gewährleisten. Außerdem könnten Sie Sperren aus Gründen der Datenbank-Performance einsetzen.

F Worin besteht der Sinn eines nicht-eindeutigen Schlüssels?

A *Nicht-eindeutige Schlüssel können die Datenbank-Performance bei Leseoperationen verbessern. Auch wenn die daraus resultierenden Datensätze vom Schlüssel nicht eindeutig identifiziert werden können, kann viel schneller darauf zugegriffen werden. Auch das Sortieren und Gruppieren der Ergebnisse kann schneller abgewickelt werden.*

F Sind alle eindeutigen Schlüssel auch zwangsläufig Primärschlüssel?

A *Nein, in MySQL kann es nur einen Primärschlüssel, aber mehrere eindeutige Schlüssel pro Tabelle geben. Sowohl eindeutige als auch Primärschlüssel können die Datenauswahl an Hand einer `WHERE`-Klausel beschleunigen. Doch achten Sie darauf, eine Spalte als Primärschlüssel zu wählen, wenn es sich dabei um eine Spalte handelt, über die Sie mehrere Tabellen miteinander verknüpfen möchten.*

- F** Sollte jede Tabelle einen Primärschlüssel haben?
- A** *Nicht unbedingt. Im Allgemeinen sollten Tabellen einen Primärschlüssel haben, wenn Sie diese in einer Verknüpfungsoperation mit einer anderen Tabelle verwenden. Unter bestimmten Bedingungen sollten Sie allerdings den Gebrauch von Schlüsseln vermeiden.*
- F** Ich bin mir nicht immer sicher, wann ich Schlüssel verwenden soll und wann nicht. Woran kann ich erkennen, was das Richtige ist?
- A** *Auch wenn es einfach zu sagen ist, wann ein Primärschlüssel benutzt werden muss, so ist dies bei anderen Schlüsseln nicht immer so eindeutig. Es können Gründe dafür oder dagegen sprechen, unter Umständen sogar widersprüchlich sein. Lassen Sie in diesem Fall die Datenbank inklusive Schlüssel einfach laufen und messen Sie ihre Performance. Löschen Sie dann die Schlüssel und messen Sie die Performance noch einmal. Danach erstellen Sie die Schlüssel auf eine andere Art neu. Experimentieren Sie so lange, bis die Performance optimal ist. Wenn nötig, können Sie Schlüssel regelmäßig löschen und neu erstellen. Dies ist zum Beispiel bei einer Reihe von Schreiboperationen zu empfehlen.*

Übungen

1. Stellen Sie sich vor, Sie haben eine Tabelle namens `Orders`, und Sie möchten diese aktualisieren, während niemand drittes aus dieser lesend zugreifen kann. Sie haben zudem eine Tabelle `Products`, aus welcher Sie Daten abfragen wollen, in der Sie jedoch keine Veränderungen vornehmen möchten. Schreiben Sie die richtige Anweisung mit den richtigen Sperren (`LOCK TABLES` und `UNLOCK TABLES`).
2. Stellen Sie sich die folgenden Situationen vor. Wann würden Sie einen Schlüssel verwenden? Würden Sie einen eindeutigen oder einen nicht-eindeutigen Schlüssel verwenden?
 - ▶ Eine Tabelle mit allen Geburtstagen Ihrer Freunde. Ihr Computer greift einmal am Tag darauf zu, um zu sehen, ob jemand Geburtstag hat und, wenn dies der Fall ist, dieser Person eine E-Mail zu schicken.
 - ▶ Eine Tabelle mit Produkten, die über Ihren Online-Shop bestellt werden können. Doppelte Werte sind nicht erlaubt.
 - ▶ Eine Tabelle, die die Mehrwertsteuer aller verkauften Produkte speichert. In diese Tabelle wird jedes Mal, wenn ein Produkt verkauft wird, die jeweilige Summe geschrieben. Sie wird nur einmal im Quartal ausgelesen.



**Wie komme ich an
die Daten ran? –
DBIs**

**Woche
2**

Bis jetzt haben Sie mit Ihrer Datenbank nur über den MySQL-Monitor kommuniziert. Das ist ein bequemer und einfacher Weg, Ihre Datenbank zu erstellen und Daten zu verwalten. Es kann aber sein, dass er sich nicht immer für Ihre Bedürfnisse sowie die der Benutzer, die die Daten Tag ein Tag aus benutzen müssen, eignet. Machen Sie sich darüber keine Gedanken, es gibt mehr als nur den MySQL-Monitor.

MySQL liefert Ihnen eine ganze Reihe Programmier-Schnittstellen. Diese Schnittstellen oder APIs (Application Programming Interfaces) ermöglichen Ihnen, Applikationen, die MySQL benutzen, zu erstellen. Applikationen, die erstellt werden, um mit Datenbanken zu interagieren, werden im allgemeinen Front-End-Applikationen genannt. Diese stellen die Benutzer-Schnittstelle zur Verfügung. Die Datenbank-Engine hingegen wird im allgemeinen Back-End-Applikation genannt. MySQL ist eine sehr gute Back-End-Datenbank.

Heute erfahren Sie, wie Sie mittels Schnittstellen auf MySQL zugreifen können. Sie lernen Folgendes:

- ▶ wie diese Schnittstellen funktionieren
- ▶ die Grundlagen, die alle Schnittstellen teilen

Verbindung aufnehmen

Heutzutage ist es schwierig, eine Geschäftsapplikation zu erstellen, die ohne Datenbank auskommt. Es gibt sicherlich Hilfsprogramme, aber meistens ist es so, dass eine Applikation für die Geschäftswelt mit einer Datenbank kommunizieren muss. Um verstehen zu können, wie Applikationen und Datenbanken interagieren, müssen Sie wissen, wie Datenbank-Server funktionieren, und Sie müssen sich – zumindest etwas – mit Client/Server-Programmierung auskennen.

Wie Sie gelernt haben, sind Datenbank-Server Programme, die kontinuierlich im Hintergrund laufen. Je nach Betriebssystem heißen sie Services bzw. Daemons. Um von der Datenbank profitieren zu können, muss die Applikation zuerst einmal eine Verbindung zu diesem Server-Programm herstellen. Dies geschieht im Normalfall mittels einer Schnittstelle.

Eine *Schnittstelle* befindet sich zwischen Ihrem Programm und der Datenbank. Sie agiert als eine Art Dolmetscher, der die von Ihrem Programm gegebenen Befehle für die Datenbank übersetzt. Die Schnittstelle ist einfach eine Ansammlung von für den Programmierer zugänglichen Funktionen. Der Programmierer kann diese benutzen, um bestimmte Datenbank-Arbeiten zu verrichten, wie zum Beispiel die Verbindungsaufnahme zur Datenbank oder das Ausführen einer Anfrage.

Im Allgemeinen besteht die Schnittstelle aus zwei Teilen. Zum einen aus dem Programmcode Ihrer Applikation, der die Funktionsaufrufe und Variablen enthält. Dieser Code ist universell, das heißt, egal welche Datenbank Sie benutzen, die von Ihnen verwendeten Funktionen und Variablen bleiben immer die gleichen. Zum anderen aus dem Treiber. Dieser Teil der Schnittstelle leistet die Hauptarbeit. Er übersetzt die Funktionsaufrufe aus dem Programm-Code und interagiert daraufhin mit der Datenbank, um die angeforderten Ergebnisse zu erhalten. Diese Treiber werden von den Datenbank-Entwicklern hergestellt und sind spezifisch für die jeweilige Datenbank. Wenn Sie bisher Sybase benutzt haben und jetzt zu MySQL wechseln wollen, benötigen Sie den MySQL-Treiber, damit Ihr Programm-Code auch weiterhin funktioniert. Abbildung 12.1 illustriert den gesamten Ablauf.

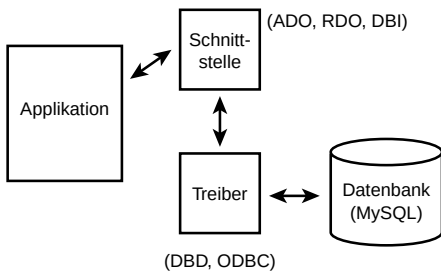


Abbildung 12.1:
Prozessablauf für Schnittstellen und Treiber

Einer der Vorteile, in dieser Weise auf eine Datenbank zuzugreifen, ist, dass Sie Ihren Programmcode nur einmal erstellen müssen. Egal welche Datenbank Sie als Back-End einsetzen, Ihr Programmcode bleibt der gleiche. Nehmen wir an, Sie haben eine Applikation entwickelt, die Sybase als Back-End und Perl DBI/DBD als Schnittstelle einsetzt. Nachdem Ihre Applikation erstellt und, der Programmcode getestet ist sowie Beta-Versionen rausgeschickt und, die Kunden zufrieden sind, entscheidet das Management sich zu einem Wechsel des Datenbank-Produkts. Sie haben von diesem unglaublichem MySQL gehört und beschließen, dieses zu verwenden. Jetzt stellt sich die Frage, wie viel Programmcode man abändern muss. Die Antwort: Fast gar keinen. Um Diskrepanzen zwischen den verschiedenen Datenbank-Produkten auszugleichen, müssen Sie vielleicht ein paar Abänderungen vornehmen – das ist aber auch schon alles. Bevor es Datenbank-unabhängige Schnittstellen gab, mussten die Programmierer die Funktionen, die mit den Datenbanken agierten, selbst codieren. Beim Wechsel auf ein anderes Datenbank-Produkt musste eine Schnittstelle komplett neu geschrieben werden – eine Menge Arbeit.

Für MySQL gibt es viele Schnittstellen – u.a. PHP, Perl, C++, Python oder ODBC. Sie müssen diese natürlich nicht benutzen; manchmal ist es sinnvoller, eine Schnittstelle selbst zu erstellen. Dafür gibt es in MySQL die C++-Quelldatei. Diese ermöglicht Ihnen, Ihre eigene Schnittstelle zu erstellen oder Ihre auf einem Ihnen zur Verfügung gestellten Programmcode basierende Schnittstelle zu verbessern.

Schnittstellen sind eine schnelle und saubere Methode, Applikationen zu erstellen, die mit Datenbanken kommunizieren. Sie fragen sich vielleicht, wie Applikationen mit Datenbanken kommunizieren. Stellen Sie sich die Schnittstelle als Kunden und den Treiber als Telefonisten vor. Ein Telefonist nimmt Anrufe entgegen und leitet sie weiter. Ruft ein Kunde bei der Vermittlungsstelle an und fragt nach einer bestimmten Person, kennt der Telefonist die Nummer, unter der diese Person erreicht werden kann, und verbindet den Kunden. Bei der Schnittstelle und dem Treiber läuft das genauso ab. Die Applikation möchte mittels Schnittstelle Verbindung zu einer Datenbank aufnehmen. Die Applikation liefert dann einige Informationen wie zum Beispiel die Adresse und den Namen der Datenbank, mit der sie kommunizieren möchte. Die Schnittstelle nimmt sich dieser Informationen an und »ruft den Treiber an«. Dieser erhält die Informationen und stellt die Verbindung her. Der Treiber weiß, mit welchem Port er »reden« und wie er mit der Datenbank kommunizieren muss. Der Port entspricht ungefähr der Telefonnummer der Person, mit der Sie sprechen möchten. Er ist ein Kommunikationskanal. Jetzt können Daten ungehindert zwischen Applikation und Datenbank ausgetauscht werden.

Verbindungsanforderungen

Da Sie jetzt in etwa wissen, wie Schnittstellen und Treiber funktionieren, führen wir nun eine richtige Verbindung durch. Sie wissen bereits, wie eine Verbindung aufgenommen wird, aber Sie kennen noch nicht die Voraussetzungen.

Eine Voraussetzung, um eine Verbindung zu einer Datenbank im Gegensatz zu einer physischen Verbindung herzustellen, ist der Treiber. Man muss den passenden Treiber installieren. Ohne Treiber kann die Applikation nicht mit der Datenbank kommunizieren. Vergewissern Sie sich, dass Sie den aktuellsten Treiber verwenden. Viele Hersteller aktualisieren ihre Treiber regelmäßig, meistens wenn ein neues Produkt für eine Datenbank auf den Markt kommt.

Eine weitere Voraussetzung ist die Kenntnis des Namens oder der Adresse des Servers. Denn wie wollen Sie jemanden anrufen, wenn Sie dessen Nummer nicht kennen? Oder wie wollen Sie jemanden besuchen, wenn Sie nicht wissen, wo er wohnt? Das Gleiche gilt für eine Datenbankverbindung. Ohne die IP-Adresse des Datenbank-Servers oder den Servernamen (für den Fall, dass Sie dynamische Adressierung verwenden) zu kennen, weiß der Treiber nicht, wohin er die Verbindung aufnehmen soll.

Außerdem benötigt der Treiber für den Zugriff auf eine Datenbank den Datenbanknamen. Auch hier gilt: Wie wollen Sie eine Verbindung herstellen, ohne zu wissen wohin? Wenn Sie die Datenbank nicht spezifizieren, weiß der Treiber nicht, wo er die Daten herbekommen soll.

Die letzte Voraussetzung, die bei jeder Datenbank-Verbindung benötigt wird, ist der Benutzername und ein Passwort. Der Datenbank-Server lässt nicht jeden auf die Datenbank zugreifen. Er prüft seine GRANT-Tabellen und sieht daran, wer auf welche Datenbank zugreifen darf. Stimmen Benutzername und Passwort nicht überein, wird keine Verbindung hergestellt.



Über Sicherheitsfaktoren können Sie an Tag 17 mehr erfahren.

Diese vier Anforderungen sind für fast alle Datenbank-Schnittstellen/-Treiber die gleichen. Wenn Sie ein Entwickler sind und damit beginnen möchten, Ihre erste Applikation zu schreiben, sollten Sie diese Erfordernisse kennen, bevor Sie weitermachen.

An die Daten rankommen

Das Konzept für den Zugriff auf Daten ist für jede Schnittstelle gleich. Es gibt eine Reihe von Schritten, die die gleichen sind, wenn Sie auf Daten zugreifen möchten, egal welche Schnittstelle oder Treiber Sie verwenden. Sie benötigen im Grunde immer die gleichen Informationen. Hier ist der allgemeine Ablauf:

1. Nehmen Sie Verbindung zur Datenbank auf.
2. Führen Sie eine Anfrage oder einen Befehl aus.
3. Haben Sie eine Anfrage ausgeführt, erhalten Sie danach einen Datensatz.
4. Bearbeiten Sie den Datensatz.
5. Beenden Sie die Verbindung (siehe Abbildung 12.2).

Dieser Kreislauf gilt für alle Schnittstellen, ganz gleich von welcher Programmiersprache er realisiert oder auf welcher Plattform er betrieben wird. Es wird kaum vorkommen, dass Sie von diesem natürlichen, logischen Kreislauf abweichen. In den folgenden Abschnitten werden die einzelnen Schritte genauer beschrieben.

Die Verbindung herstellen

Der erste Schritt dieses Prozesses besteht darin, eine Verbindung zur Datenbank herzustellen. Diese Verbindung stellt einen Verbindungszeiger zwischen Ihrer Applikation und der Datenbank her. Dieser Zeiger überträgt all Ihre SQL-Anweisungen von Ihrer Applikation auf die Datenbank und gibt Ihrer Applikation sämtliche Ergebnismengen der Datenbank zurück.

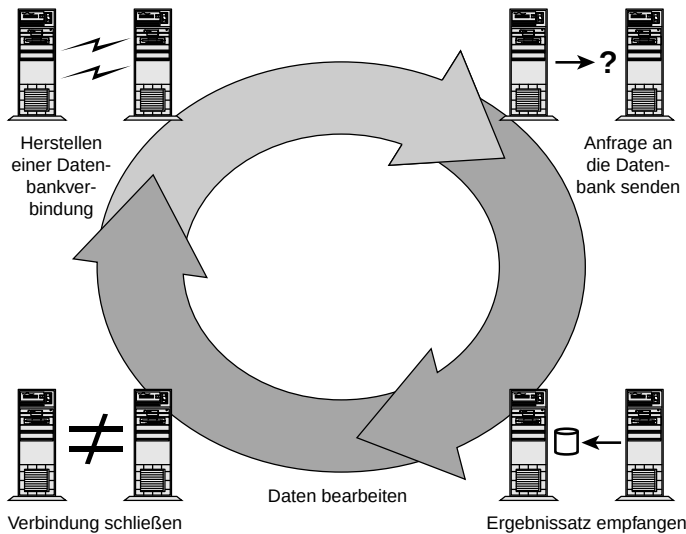


Abbildung 12.2:
Der allgemeine Schnittstellen-Kreislauf

Die Verbindung benötigt die vorher genannten Parameter: Die Datenbank, den Treiber, die IP-Adresse oder den Servernamen der Datenbank sowie den Benutzernamen und ein Passwort. Ohne diese Parameter ist es nicht möglich, eine Verbindung herzustellen. In den nächsten drei Tagen werden Sie diese Theorie in die Praxis umsetzen. Sie werden sehen, dass jede Schnittstelle zur Herstellung einer Verbindung die gleichen Parameter benutzt und einen Verbindungszeiger zwischen Ihrem Programm und der Datenbank erstellt.

Dieser Vorgang muss während einer Sitzung nur einmal ausgeführt werden. Nachdem Sie eine Verbindung hergestellt haben, können Sie diese für alle Ihre Anfragen verwenden. Wenn Sie eine neue Anfrage an die Datenbank ausführen, müssen Sie die Sitzung nicht erst beenden und wieder neu eröffnen. Hat Ihre Applikation den Verbindungszeiger erstellt, bleibt dieser bestehen, bis Ihre Applikation ihn explizit löscht oder er automatisch gelöscht wird, weil er nicht mehr gebraucht wird.

Anfragen an die Datenbank stellen

Im nächsten Schritt stellen Sie Anfragen an die Datenbank. An dieser Stelle betten Sie SQL-Anweisungen in Ihre Applikation ein und geben diese an die Datenbank weiter. Es kann sich hierbei um Befehle, die Datenstrukturen wie beispielsweise Tabellen erstellen, oder um `SELECT`-Anweisungen handeln. Das hängt ganz von den Anforderungen Ihres Programms ab. Sie können so ziemlich alles, was Sie in den MySQL-Monitor eingeben, auch in Ihr Programm einbetten. Das schließt `OPTIMIZE`-Anweisungen sowie die `FLUSH-Privilegien`-Anweisung mit ein. Sie können sogar die `LOAD DATA INFILE`-Anweisung benutzen. Dies ermöglicht Ihnen, sehr flexible und leistungsstarke Applikationen zu erstellen, die MySQL in seiner ganzen Spannweite nutzen.

Die meisten Schnittstellen unterscheiden zwischen dem Erstellen eines Datensatzes und dem Erlassen eines Befehls, der keine Daten zurückgibt. Die meisten Schnittstellen haben eine Ausführungsfunktion, die einfach die von Ihnen übergebenen SQL-Anweisungen ausführen. Das ist sehr praktisch bei `INSERT`-, `UPDATE`- und `DELETE`-Anweisungen, nicht aber bei einer `SELECT`-Anweisung. Denn wie wollen Sie die Ergebnisse bei der Ausführung einer `SELECT`-Anweisung erhalten? Wohin mit den Datensätzen? Deshalb stellen die Schnittstellen ein Objekt oder ein Array zur Verfügung, in denen die Datensätze gehalten werden. An den folgenden Tagen werden Sie mehr darüber erfahren, wie Sie Variablen von Ihrer Applikation an die Datenbank übertragen, um die benötigten Ergebnisse zu erhalten.

Datensätze erhalten und verarbeiten

Im dritten Schritt erhalten Sie die Datensätze. Ein *Ergebnissatz* ist das Resultat einer `SELECT`-Anfrage. Ein Ergebnissatz beinhaltet die gleichen Informationen wie die Ausführung einer `SELECT`-Anweisung im MySQL-Monitor. Der einzige Unterschied besteht darin, dass das Ergebnis nicht auf dem Bildschirm ausgegeben, sondern in eine Art Variable geladen wird, die Sie in Ihrer Applikation verwenden können. Bei diesen Variablen handelt es sich normalerweise um ein Array. Der Ergebnissatz besteht aus einer Reihe von Zeilen. Jede Zeile enthält die Informationen der Spalten, die in der `SELECT`-Anweisung erfragt wurden. Dies erstellt eine gitterartige Struktur, die der Ausgabe im MySQL-Monitor ähnelt. Um alle Datensätze aus diesem Ergebnissatz zu erhalten, müssen Sie die Daten aus einer Zeile herausziehen, zur nächsten Zeile gehen, wiederum die Daten herausziehen usw. Das ist wie beim Gehen – immer schön einen Fuß vor den anderen setzen.

Die Ergebnisse werden auf Basis eines Namen/Werte-Paars einer Spalte gespeichert. Beispielsweise gibt es eine `First_Name`-Spalte in der `Customers`-Tabelle. Je nach Schnittstelle, die Sie benutzt haben, können Sie sich in Ihrem Ergebnissatz mittels Spaltennamen auf die Spalte beziehen. Wenn Sie Perl benutzen, könnten Sie sich in einem Ergebnissatz folgendermaßen auf das `First_Name`-Feld beziehen:

```
$ref ->{'First_Name'}
```

Dies gibt den Wert zurück, der in der `First_Name`-Spalte für diese Zeile enthalten ist. Sie können sich auch mittels ihrer Indizes auf Spaltennamen beziehen. Falls die `First_Name`-Spalte die erste Spalte in Ihrer `SELECT`-Anfrage war, die benannt wurde, so können Sie sich auf sie beziehen, indem Sie ein `ADO-Recordset`-Objekt benutzen – wie im Folgenden gezeigt:

```
RstCustomers.Fields(0)
```

Was die Anzahl der Ergebnissätze betrifft, gibt es keinerlei Limit. Sie können für jede Anfrage, die Sie ausführen, einen neuen Ergebnissatz erstellen. Sie haben die Wahl –

beachten Sie aber, dass diese Variablentypen einen gewissen Speicherplatz benötigen. Bevor Sie einen Ergebnissatz mit 100 Objekten erstellen, sollten Sie an die Performance denken.

Nun da Sie den Ergebnissatz haben, können Sie mit den Daten machen, was immer Sie möchten. Sie können dem Benutzer die Daten anzeigen, sie in Grafiken und Diagrammen benutzen oder Berichte erstellen. Die Möglichkeiten sind unbegrenzt.

Die Verbindung beenden

Ihre Mutter pflegte wahrscheinlich immer zu sagen (zumindest hat meine es immer gesagt): »Räum Deine Spielsachen auf, wenn Du mit dem Spielen fertig bist!« Der gleiche Grundsatz gilt auch in der Programmierwelt. Wenn Sie Objekte erstellt haben und diese nicht länger zu gebrauchen sind, sollten Sie sie loswerden. Das Gleiche gilt auch für Verbindungen – wenn Sie diese nicht mehr brauchen, werden Sie sie los. Geben Sie die von diesen verbrauchten Ressourcen frei für andere Benutzer. Denn jede Verbindung verbraucht sowohl auf dem Client als auch auf dem Server Ressourcen. Das Aufräumen nach getaner Arbeit spart Ressourcen und verhindert Fehler.

Zusammenfassung

Dieser Tag hat Sie auf das, was noch kommt, vorbereitet. Ihnen wurden die Konzepte einer Schnittstelle und eines Treibers vorgestellt. Sie haben gelernt, dass diese beiden zusammenarbeiten, um Ihnen Zugang zu Ihrer Datenbank zu verschaffen. Sie konnten sehen, dass die Schnittstelle, unabhängig davon, welche Datenbank Sie benutzen, immer die gleichen Funktionen und Variablen zur Verfügung stellt. Der Treiber ist das Einzige, was verändert werden muss. Sie haben zudem die Vorteile dieser Methode kennen gelernt.

Außerdem haben Sie etwas über die Parameter, die für jede Datenbank benötigt werden, erfahren. Bevor eine Verbindung aufgenommen werden kann, müssen diese Parameter zur Verfügung stehen.

Es wurde auf die allgemeinen Schritte, die alle Schnittstellen bei der Arbeit mit Datenbanken gemein haben, eingegangen. Sie konnten sehen, dass dieser Zyklus logisch ist und dass er dem Programmierer große Flexibilität und Leistungsstärke ermöglicht. Darüber hinaus haben Sie erfahren, wie Sie SQL-Befehle in Ihrer Applikation einbetten können, um jegliche Aufgabe von Datenbank-Administration bis `SELECT`-Anweisungen erfüllen zu können.

Die Interaktion mit Ihrer Datenbank über eine Schnittstelle ist in der Geschäftswelt am wichtigsten. Ihre Datenbank ist nutzlos, wenn Sie keinen Zugang zu den Daten haben

oder diese nicht verarbeiten können. Schnittstellen bieten Programmierern, Web-Developern und Datenbank-Administratoren die Möglichkeit, Ihre Datenbanken programmatisch zu handhaben.

Fragen und Antworten

F Wie werden meine Anfragen ausgeführt?

A *Die SQL-Anweisungen, die Sie in Ihr Programm einbetten, werden an den Treiber übergeben. Der Treiber enthält alle Datenbank-spezifische Funktionen. Der Treiber ruft die Funktion auf, die die Anfrage gegen Ihre Datenbank ausführt. Das Ergebnis dieser Funktion ist Ihr Ergebnissatz.*

F Kann ich Datenbanken erstellen, indem ich Schnittstellen benutze?

A *Mit Schnittstellen können Sie so ziemlich alles machen, was Sie auch mit dem MySQL-Monitor machen können. Haben Sie die entsprechenden Genehmigungen, können Sie `CREATE`-, `DROP`-, `OPTIMIZE`-, `INSERT`-, `DELETE`- und `UPDATE`-Anweisungen ausführen. Es liegt in Ihrer Hand, administrative Aufgaben zu automatisieren oder Daten durch ein Programm, das Sie erstellen können, zu verarbeiten.*

F Welche Schnittstellen gibt es für MySQL?

A *MySQL besitzt viele Schnittstellen und Treiber. Einige wurden von Tcx, andere von Drittanbietern erstellt. Es gibt PHP-, Perl- (`DBD/DBI`-), `ODBC`- (`MyODBC`-), Java- (`JDBC`-), C/C++- und Python-Schnittstellen. Da MySQL `ODBC`-kompatibel ist, können Sie Tools wie VBScript, Visual Basic, Access, Crystal Reports oder jedes andere Tool, das eine `ODBC`-Datenquelle benutzen kann, verwenden.*

Übungen

1. Nennen Sie Vor- und Nachteile des Gebrauchs von Schnittstellen im Gegensatz zum MySQL-Monitor!
2. Welche sind die Grundvoraussetzungen für eine Datenbankverbindung?



**Wie benutze ich
MyODBC?**

**Woche
2**

Das MyODBC-Interface stellt nur eine Möglichkeit dar, wie Sie mit Ihren MySQL-Daten auf einer Windows-Plattform interagieren können. Diese Technologie wurde von Microsoft entwickelt und zu einem wesentlichen Bestandteil in vielen ihrer Applikationen gemacht.

Open Database Connectivity (ODBC) von Microsoft ermöglicht einem Benutzer, auf jede Datenbank mit den gleichen Befehlen zuzugreifen. Es ist eine Ebene zwischen dem Programm und der Datenbank. Jede Datenbank hat ihren eigenen ODBC-Treiber, der sich an den Standard-ODBC-Treiber von Microsoft hält. Wie schon gesagt, agiert dieser Treiber zwischen dem Programm und der Datenbank; Sie können sich ihn als Übersetzer vorstellen. In Ihren Programmen geben Sie Befehle, die daraufhin vom Treiber übersetzt und an die Datenbank weitergeleitet werden. Die Datenbank sendet dann den Ergebnissatz oder die Botschaft zurück zum Treiber, der diesen wiederum an die Applikation zurückschickt.

Diese Konstruktion ermöglicht dem Programmierer, ungeachtet der Datenbank, mit der er kommuniziert, den gleichen Programmcode zur Interaktion mit einer Datenbank zu benutzen. Das funktioniert so lange, wie sich der Programmierer beim Erlassen der Befehle an den ANSI-SQL-Standard hält. Benutzt er hingegen datenbankspezifische Befehle, sind diese nicht wiederverwendbar.

Erstellen Sie Programme für eine Windows-Plattform (das schließt Active Server Pages mit ein), werden Sie höchstwahrscheinlich ODBC zur Kommunikation mit Ihrer Datenbank verwenden. ODBC ist einfach zu benutzen und hält sich an die in der gestrigen Lektion vorgestellten Schritte.

In den heutigen Beispielen und Lektionen werden Sie ODBC einsetzen, um mit einer MySQL-Datenbank zu kommunizieren. Dazu werden Sie eine Active Server Page (ASP) mit Hilfe von VBScript erstellen. Heute werden Sie also Folgendes lernen:

- ▶ wie Sie einen MyODBC-Treiber bekommen und installieren
- ▶ wie Sie einen Datenquellennamen (DSN) einrichten
- ▶ wie Sie eine Verbindung zu einer MySQL-Datenbank mit Hilfe von ADO (ActiveX Data Objects) herstellen
- ▶ das Erstellen einer Active Server Page zum Zugriff auf eine MySQL-Datenbank

Woher bekomme ich einen Treiber?

Bevor Sie ODBC in Ihren Applikationen benutzen können, benötigen Sie zuerst den ODBC-Treiber für Ihre Datenbank. Microsoft installiert in seinen Applikationen automatisch mehrere seiner Treiber. Einige Datenbanken installieren den ODBC-Treiber

standardmäßig auch während ihrer Installation. Führen Sie die folgenden Schritte aus, und Sie werden sehen, welche Treiber momentan auf Ihrem Computer installiert sind.

1. Klicken Sie auf *Start, Einstellungen, Systemsteuerung*.
2. Sie sollten jetzt ein Icon namens »ODBC-Datenquellen (32Bit)« sehen (die Bezeichnung der Icons unterscheidet sich leicht bei den verschiedenen Betriebssystemen). Doppelklicken Sie auf dieses.
3. Der Datenquellen-Administrator wird angezeigt. Hier können Sie ODBC-Treiber anpassen oder neue hinzufügen sowie Datenquellennamen erstellen – dazu kommen wir gleich.
4. Klicken Sie auf die Registerkarte *Treiber*. Es erscheint eine Liste mit den momentan installierten Treibern. Wenn Sie der Lektion nicht schon vorgegriffen haben, sollten Sie hier noch keine MySQL-Treiber finden. Sie sollten einige andere, vor allem Microsoft-Treiber, vor sich haben.

Nun wissen Sie, welche Treiber momentan in Ihrem System installiert sind. Es ist jetzt an der Zeit, einen weiteren hinzuzufügen.

Auf der buchbegleitenden CD-ROM sollten Sie einen Ordner vorfinden, der alle Produkte enthält, die Sie von der MySQL-Webseite herunterladen können. Suchen Sie nach der ODBC-Zip-Datei. Diese sollte ungefähr den Namen `myodbc-2.50-36-win95.zip` (für Win95/98) bzw. `myodbc-2.50.26-nt.zip` (WinNT) tragen. Vergewissern Sie sich, dass Sie WinZip installiert haben. Doppelklicken Sie auf die Datei. Jetzt sollte sie sich selbst entpacken. Führen Sie die folgenden Schritte aus, um den MyODBC-Treiber auf Ihrem Rechner zu installieren.

1. Nach dem Entpacken sollte das Installationsprogramm laufen. Die erste Bildschirmausgabe sollte Abbildung 13.1 gleichen.
2. Klicken Sie auf *Continue*. Jetzt sollte das Treiber-Installations-Dialogfeld, wie in Abbildung 13.2 gezeigt, erscheinen. Klicken Sie auf *MySQL* und dann auf *OK*.
3. Die Programmdateien werden daraufhin auf Ihre Festplatte kopiert. Nach einigen Sekunden sollten Sie ein Dialogfeld wie in Abbildung 13.3 vor sich haben.
4. Dort klicken Sie auf *Close*. Den Datenquellennamen werden Sie erst später hinzufügen.

Glückwunsch, Sie haben den MyODBC-Treiber installiert.

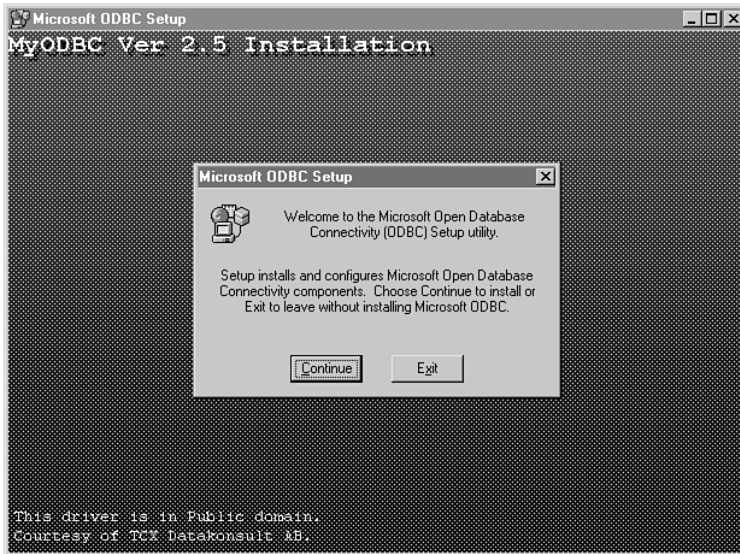


Abbildung 13.1:
Das MyODBC-
Installations-
programm



Abbildung 13.2:
Das MySQL-
Treiber-Installations-
Dialogfeld

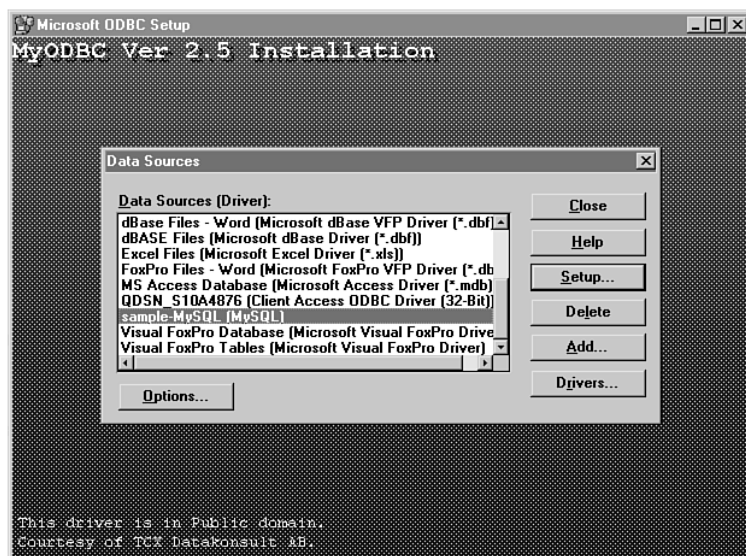


Abbildung 13.3:
Datenquellen-
Dialogfeld

Einrichten des Datenquellennamens (DSN)

Nun da der MyODBC-Treiber installiert ist, können Sie einen Datenquellennamen erstellen, den Ihre Applikation dann benutzen kann. Eine Datenquelle ist eine Datei, die sämtliche spezifischen Informationen, die Ihrer Datenbank dienlich sind, enthält – beispielsweise alle Datenbanknamen, alle Benutzernamen für eine bestimmte Datenbank oder sämtliche Informationen, die nötig sind, um eine Verbindung herzustellen. Auf diese Weise benötigt eine Applikation lediglich den DSN, um den ODBC-Treiber zu konfigurieren. Wie Sie beim Herstellen der Verbindung zu einer Datenbank um einen DSN herumkommen, werden Sie später lernen. Führen Sie jetzt folgende Schritte aus, um der Meet_A_Geek-Datenbank einen DSN hinzuzufügen.

1. Klicken Sie auf *Start, Einstellungen* und *Systemsteuerung* (Abbildung 13.4).
2. Doppelklicken Sie auf das ODBC-Datenquellen- (32Bit-) Icon. Dies öffnet den ODBC-Datenquellen-Administrator (siehe Abbildung 13.5).

Der Datenquellen-Administrator besteht aus sieben Registerkarten. Die erste, *Benutzer-DSN*, errichtet benutzerspezifische DSNs für den aktuellen Rechner. Ein Benutzer-DSN wird für eine bestimmte Person errichtet. Die zweite Registerkarte, *System-DSN*, erstellt eine Datenquelle für jeden, der den aktuellen Rechner benutzen kann. Die nächste, *Datei-DSN*, erstellt einen Datenquellennamen, der für jeden sichtbar ist und von jedem benutzt werden kann, der die richtigen Treiber installiert hat. Die nächste Registerkarte, *Treiber*, listet alle momentan auf Ihrem Rechner installierten

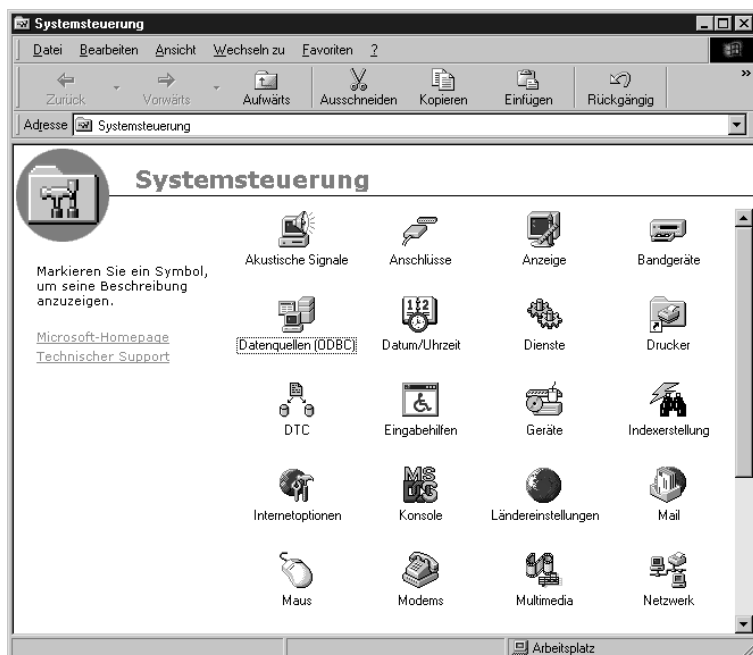


Abbildung 13.4:
Die System-
steuerung

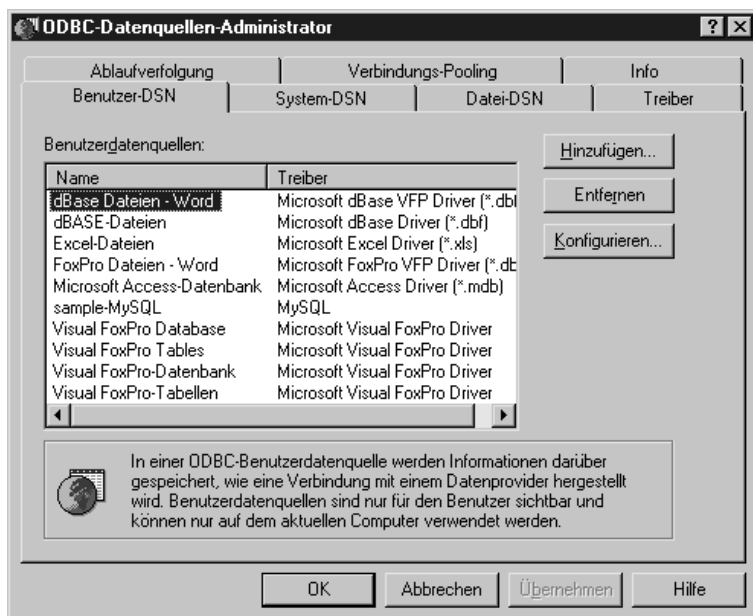


Abbildung 13.5:
ODBC-Datenquel-
len-Administrator

Treiber auf. Die fünfte Registerkarte, *Ablaufverfolgung*, aktiviert die Protokollierung für Ihre ODBC-Treiber; es werden sämtliche Verbindungen protokolliert. Microsoft Visual Studio benutzt dieses Feature, um Programme von Fehlern zu befreien. Die nächste ist die *Verbindungs-Pooling*-Registerkarte. Hinter ihr verbergen sich Optionen, die Ihrem ODBC-Treiber ermöglichen, offene Verbindungszeiger wiederzuverwenden. Dies spart Ihren Applikationen Zeit und Ressourcen. Die letzte Registerkarte ist *Info*. Hier finden Sie die grundlegenden ODBC DLLs (Dynamisch gelinkte Bibliotheken).

Da Sie eine Applikation erstellen werden, die von mehr als einer Person, die sich in den aktuellen Rechner einloggt, benutzt werden kann, werden Sie einen System-DSN für die Meet_A_Geek-Datenbank erstellen.

1. Klicken Sie auf die Registerkarte *System-DSN*. Falls Sie schon System-DSNs haben, sehen Sie diese hier (siehe Abbildung 13.6). Klicken Sie auf *Hinzufügen*, um einen neuen DSN zu erstellen.



Abbildung 13.6:
Die System-DSN-
Registerkarte

2. Jetzt sollte ein Dialogfeld wie in Abbildung 13.7 erscheinen. Es zeigt Ihnen eine Liste der erhältlichen Treiber an. Nach der Wahl des MySQL-Treibers klicken Sie auf *Fertig stellen*.

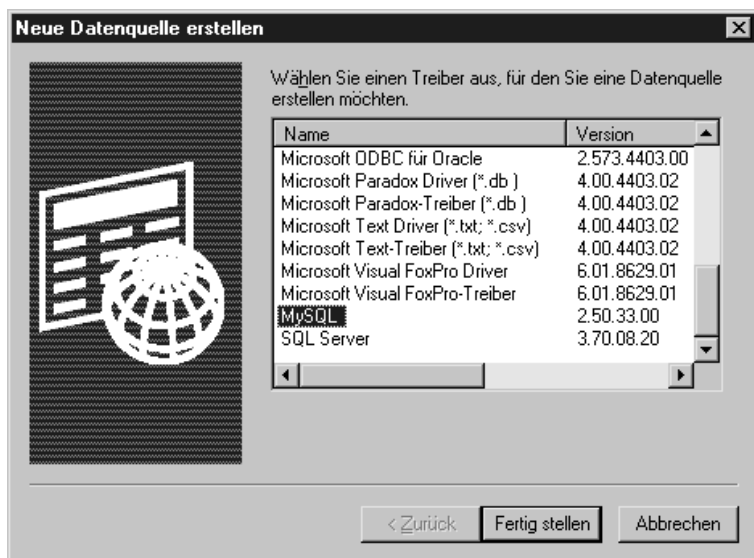


Abbildung 13.7:
Erstellen einer
neuen Daten-
quelle

- Das nächste Fenster ist das TcX-MySQL-Treiber-Konfigurationsfenster. Dieses Fenster enthält alle Informationen, die MyODBC benötigt, um mit der ausgewählten Datenbank eine korrekte Verbindung herzustellen. Geben Sie folgende Informationen in die vorgegebenen Felder ein:

Windows-DSN Name: Meet_A_Geek

MySQL host (name or IP): Den Namen oder die IP-Adresse Ihres MySQL-Servers

User: root

Password: Ihr root-Passwort

Port und SQL command: Lassen Sie diese Felder leer.

Das fertige Fenster sollte wie Abbildung 13.8 aussehen.

- Klicken Sie auf OK, um die DSN-Konfiguration abzuschließen. Sie werden zum ODBC-Datenquellen-Administrator zurückgeschickt und sollten dort Ihren neuen DSN in der Liste sehen. Klicken Sie wiederum auf OK, um den Administrator zu schließen. Sie haben jetzt Ihren ersten MySQL-System-DSN erstellt.

Beachten Sie, dass ein DSN eine einfache Datei ist, welche die vom Treiber benötigten Daten zur Verbindungsaufnahme mit einer Datenbank enthält. Ein DSN agiert genauso wie eine Konfigurationsdatei.

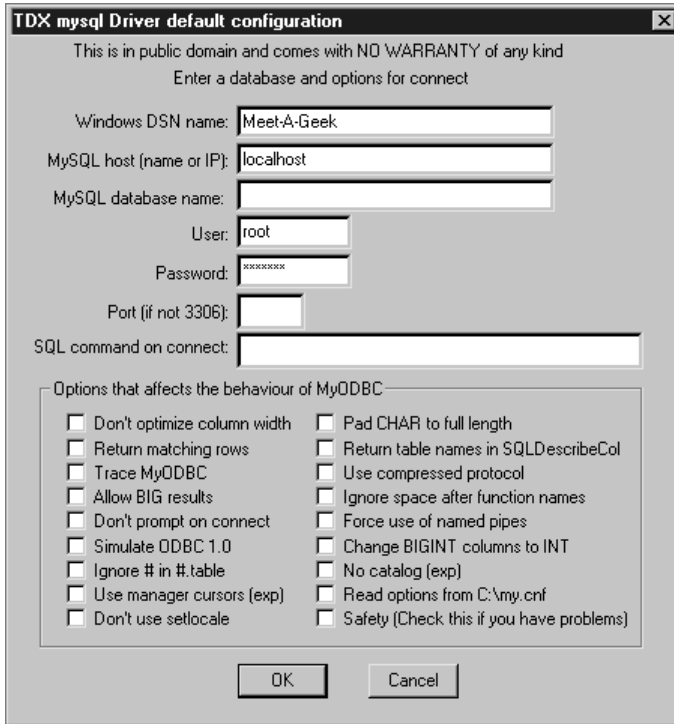


Abbildung 13.8:
Das Konfigurations-
fenster

Active Server Pages und VBScript

Beim Versuch, eine homogene Datenumgebung zu schaffen, hat Microsoft ODBC in die meisten seiner bekannten Programme miteingeschlossen. Beispielsweise kann jede Applikation in Microsoft Office eine Verbindung zu einer ODBC-Datenquelle herstellen. Da es so einfach zu benutzen und sehr sinnvoll ist, sind andere Firmen aufgesprungen und bieten nun ihrerseits Möglichkeiten zur Verbindungsaufnahme mit anderen Datenquellen mittels ODBC an.

Die Vorteile von ODBC sind der Entwicklergemeinschaft nicht entgangen. Software-Entwickler erkannten sehr schnell, welche negativen Folgen eine heterogene Datenumgebung hat, und Microsoft hat reagiert. Microsoft hat jede Menge Hilfsprogramme herausgebracht, die einen Entwickler, der ODBC zur Verbindungsaufnahme mit einer Datenbank benutzt, unterstützen. Diese sind u. a. ODBCdirect, Remote Data Objects (RDO), Data Access Objects (DAO) oder das etwas neuere ActiveX Data Objects (ADO). Sie gründen alle auf einer objektbasierten Annäherung an Daten.

Diese Programme können in den von Ihnen entwickelten Applikationen verwendet werden. Da die meisten, die dieses Buch lesen, MySQL wahrscheinlich in irgendeiner Form für den Datenbankzugang via Web benutzen, sind die Microsoft-Technologien, die Sie in den Beispielen einsetzen werden, Active Server Pages, VBScript und ADO. Diese sind leicht zu benutzen, wobei ADO beim Datenzugriff auf einer Windows-Plattform der Königsweg ist. Zuerst müssen Sie sich aber einige Grundlagen aneignen.

Active Server Pages

Um von diesen Technologien profitieren zu können, müssen Sie einen Web-Server samt installierter FrontPage-Erweiterungen benutzen. Diese sind auf der Microsoft-Webseite erhältlich. Sie laufen auf allen Plattformen, auch auf Linux. Ebenso muss auf dem Server, der diese Scripts ausführt, der MyODBC-Treiber installiert sein. Sie können auf dem Web-Server auch einen System-DSN für die Datenbank und somit DSN-Verbindungen erstellen.

Eine Active Server Page (ASP) ist eine Webseite, die aus Programmcode wie zum Beispiel VBScript und Standard-HTML-Tags besteht und in der Zeit zwischen der Anfrage nach einer ASP-Seite durch den Client (in diesem Fall dem Browser) und der Beantwortung dieser Anfrage durch den Server ausgeführt wird. Eine ASP übernimmt die Funktion eines Common-Gateway-Interface- (CGI-) Programms. CGI-Programme sind im Allgemeinen Programme, die bei Ihrer Ausführung HTML-Tags an den Webserver ausgeben. ASPs unterscheiden sich dadurch von ihnen, dass sie stark einer Webseite ähneln – mit dem Unterschied, dass sie Programmcode enthalten, der ausgeführt wird, bevor die Webseite angezeigt wird. Nehmen Sie zum Beispiel an, Sie tragen sich in ein Gästebuch auf einer Webseite ein, die ASP benutzt. Sie hinterlassen Ihre Beiträge und klicken anschließend auf *Absenden*. Dies ruft eine weitere ASP-gestützte Seite auf, die Ihren Kommentar in einer Datenbank speichert. Sie enthält zudem Code, der eine Danksagung ausgibt. Sehen Sie sich die folgenden Beispiele an, um ein besseres Verständnis für den ASP-Vorgang zu erlangen.

In der ersten Phase des Prozesses wird eine Webseite angezeigt, im Beispiel ein normales Formular eines Gästebuches (siehe Abbildung 13.9). Sie tragen einfach Ihre Beiträge ein und klicken auf *Absenden*. Der Browser schickt die im Gästebuch-Formular enthaltene Information als Variable an die Active Server Page. Diese wiederum fügt den Inhalt der Variablen in eine Datenbank ein. Danach wird Ihnen für Ihren Beitrag gedankt.



Abbildung 13.9:
Das Gästebuch-
Formular

Werfen Sie einen Blick auf das Listing 13.1.

Listing 13.1: Quellcode für Gaestebuch.asp

```
<%Language=VBScript%>
<HEAD>
</HEAD>
<%
    Dim Conn
    Dim Cmd

    Set Conn = Server.CreateObject("ADODB.Connection")
    Set Cmd = Server.CreateObject("ADODB.Command")

    Conn.CommandTimeout = 40
    Conn.ConnectionString = "server=10.1.1.50;db=ThorGB;" & _
        "uid=root;pwd=tacobell;driver=MySQL"
    Conn.Open

    Cmd.ActiveConnection = Conn
    Cmd.CommandType = 1
    Cmd.CommandText = "INSERT INTO GuestBook (Comments)" & _
        " VALUES(" & Request("Comments") & ")"
    Cmd.Execute

    Conn.Close
```

```
%>
```

```
<BODY>
```

```
<P align=center><FONT size=5><STRONG>Thank You For Signing Our Guest Book.</STRONG></FONT></P>
<P align=center><FONT size=5><STRONG>Come Back Real Soon!!</STRONG></FONT></P>
```

```
</BODY>
```

```
</HTML>
```

Wie Sie sehen, unterscheidet sich der Aufbau einer Active Server Page im Großen und Ganzen nicht von einer normalen Webseite. Der einzige Unterschied besteht im Programmcode zwischen den `<%%>`-Tags. Dieser Code wird auf dem Server, nicht auf dem Client, ausgeführt. Das bedeutet, dass dieser Code ausgeführt werden muss, bevor die Seite an den Browser geschickt werden kann. Die Ausgabe gleicht Abbildung 13.10. Wenn Sie sich den Quellcode der HTML-Ausgabeseite der letzten Seite aufrufen, sehen Sie Folgendes:

```
<HEAD>
```

```
</HEAD>
```

```
<BODY>
```

```
<P align=center><FONT size=5><STRONG>Thank You For Signing Our Guest Book.</STRONG></FONT></P>
<P align=center><FONT size=5><STRONG>Come Back Real Soon!!</STRONG></FONT></P>
```

```
</BODY>
```

```
</HTML>
```

Sie sehen, dass der Code zwischen den `<%%>`-Tags verschwunden ist. Das ist kein Fehler. Der Programmcode wurde lediglich auf dem Server ausgeführt und nicht an den Client geschickt. Mit Active Server Pages können Sie vieles machen – beispielsweise eine Seite an Hand des Inhalts einer Datenbank dynamisch generieren. Das wird aber erst etwas später behandelt.

Arbeiten mit VBScript

Nun da Sie ungefähr wissen, was eine Active Server Page ist und wie sie benutzt wird, betrachten Sie den Programmcode zwischen den `<%%>`-Tags einmal etwas genauer. Dieser in Active Server Pages benutzte Programmcode ist VBScript, ein Ableger der Visual-Basic-Programmiersprache. Da es die gleiche Syntax wie Visual Basic verwendet, ist VBScript sehr einfach zu benutzen. Darüber hinaus ist es genau so flexibel wie Visual Basic. VBScript behält die besten Aspekte von Visual Basic wie beispielsweise Fehlerbehandlung und Zeichenkettenbearbeitung bei.

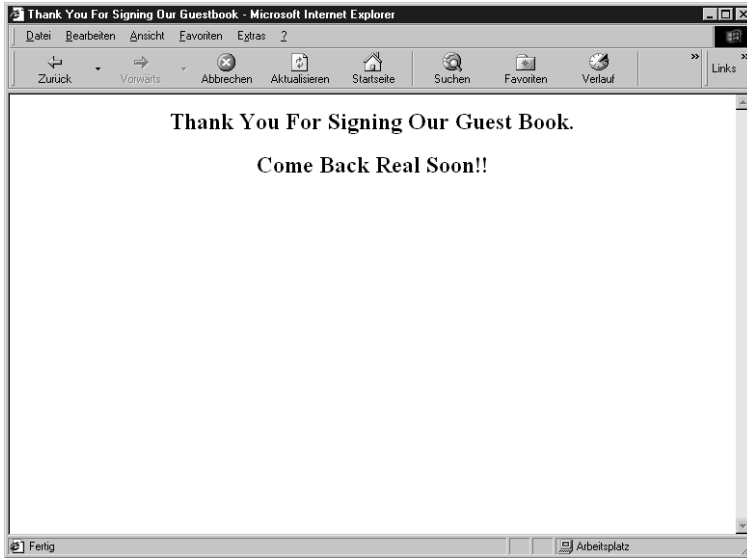


Abbildung 13.10:
Die Ausgabe von
Gaestebuch.asp



VBScript ist nicht die einzige Scriptsprache, die in Active Server Pages benutzt werden; andere sind zum Beispiel JScript, LotusScript oder PerlScript.

Es gibt viele Bücher über VBScript und Active Server Pages. Wenn Sie mehr darüber erfahren möchten, sollten Sie zumindest eines davon lesen (zum Beispiel *Jetzt lerne ich Active Server Pages*, 2. Auflage, erschienen bei Markt+Technik, 2000), da dieses Buch nur die Grundlagen abhandelt. Die reichen allerdings aus, um Ihnen zu zeigen, wie Sie eine Verbindung zu einer MySQL-Datenbank herstellen können.

VBScript-Deklarationen

Variablen können nur aus einem Typ bestehen – und zwar *Variant*. *Variant* ist ein Variablentyp, der jeden Wert enthalten kann – eine Zeichenkette, eine Zahl oder ein Objekt. VBScript stellt auch eine ganze Menge interner Funktionen zur Verfügung, die es von Visual Basic übernommen hat. Darunter sind u.a. die `MsgBox()`-, `CreateObject()`- und `IsNumeric()`-Funktionen.

Sie werden sehen, wie Sie VBScript einsetzen können, um eine Datenbank via Active Server Page zu manipulieren. VBScript beschränkt sich nicht auf ASPs. Microsoft hat diese Scriptsprache weiterentwickelt, so dass sie jetzt auf Kommandozeilenebene und auch für nicht- interaktive Administrationsaufgaben benutzt werden kann. Werfen wir jetzt einen Blick auf die grundlegenden Programmierarbeiten, die Sie mit VBScript ausführen können.

Der Programmfluss

VBScript stellt die gleichen Bedingungsanweisungen, die gleiche Programmflusskontrolle und Verarbeitung wie Visual Basic zur Verfügung. VBScript unterstützt beispielsweise Bedingungsanweisungen wie `If...Then...Else`. Die Syntax sieht im Allgemeinen folgendermaßen aus:

```
If expression Then
    Mach das
Else
    Mach dies
End If
```

Beachten Sie, dass Sie in VBScript (ebenso in Visual Basic) `Then` mit einschließen müssen. Um eine `If...Then`-Anweisung zu beenden, müssen Sie `End If` verwenden. Sehen Sie sich folgendes Beispiel an:

```
Dim vCount
Dim vLastName

If vCount > 1 Then
    Response.Write("Der Nachname ist " & vLastName)
Else
    Response.Write("<p></p>")
End If
```

In den ersten zwei Zeilen des Beispiels werden die Variablen deklariert. Der darauf folgende Block ist der `If...Then`-Abschnitt. Wenn der Ausdruck `vCount>1` WAHR ergibt, wird die Aussage auf der Webseite ausgegeben. Ergibt der Ausdruck FALSCH, wird nichts ausgegeben. Das ist ein einfaches Beispiel. Die Bedingungsanweisungen können auch komplexer sein, behalten aber ihre grundlegende Struktur bei.

Schleifenkontrolle

VBScript unterstützt ebenfalls `DoWhile`, `DoUntil`, `While` und `For...Next` als allgemeine Programmflusskontroll-Anweisungen. Eine Schleife benutzen Sie, um durch einen Ergebnissatz zu wandern oder eine Anweisung mehrere Male auszuführen, bevor Sie weitermachen. In VBScript kann eine typische Schleife folgendermaßen aussehen:

```
Do Until rstCustomers.EOF
    Response.Write("<p>" & rstCustomers("LastName") &
"</p>")
    RstCustomers.MoveNext
Loop
```

Was auch immer in diesem Beispiel zwischen den Schlüsselwörtern `Do` und `Loop` steht wird fortfahren, bis der Ausdruck `rstCusozomers.EOF` WAHR ergibt. Diese Art von Anweisung erweist sich als praktisch beim Wandern durch Ergebnissätze.

VBScript kann auch Rechnungen wie beispielsweise Additionen oder Subtraktionen ausführen. Es unterstützt zudem Vergleichsoperatoren. In einer Active Server Page komplexe Rechenvorgänge durchzuführen wird jedoch nicht empfohlen, da der ASP-Code auf dem Server ausgeführt wird, bevor er zum Client zurückgeschickt wird. Bei komplexen Rechenvorgängen könnte die Antwort auf die Anfrage des Client, auf sich warten lassen.

Kommentieren Ihres Programmcodes

In Ihren Server- Pages können Sie auch Kommentare anbringen. Ein Kommentar wird durch ein einfaches Anführungszeichen (') gekennzeichnet. In VBScript gibt es keine Block-Kommentare. Umfasst Ihr Kommentar mehr als eine Zeile, müssen Sie am Anfang jeder Zeile ein einfaches Anführungszeichen setzen.

```
' Dieses ist ein einzeiliger Kommentar.  
' Dieses  
' ist  
' ein  
' Kommentar über mehrere  
' Zeilen.
```

Die Tatsache, dass Sie von den internen VBScript-Funktionen profitieren können, ist ein weiteres Leistungsmerkmal von VBScript. Es unterstützt viele der Visual-Basic-Funktionen.

Generieren einer Webseite mit VBScript

Werfen Sie jetzt einen Blick auf ein VBScript-Beispiel, das zeigt, was Sie bisher gelernt haben. Der Ausschnitt des Programmcodes im Listing 13.2 zeigt, wie VBScript, wenn es in einer Active Server Page benutzt wird, eine Webseite generieren kann, die zwei Werte aus einer MySQL-Datenbank in einer Webseite anzeigt. Diese Webseite gleicht Abbildung 13.11. Beachten Sie, dass die Zeilenzahlen nur zur Orientierung dienen – sie sollten nicht in Ihrem Programmcode erscheinen.

Listing 13.2: Generieren einer Webseite

```
10 <%@ Language=VBScript %>  
20 <HTML>  
30 <HEAD>  
40 <META HTTP-EQUIV="Expires" CONTENT="0">  
50 </HEAD>  
60 <%  
70     ' Variablen Deklaration  
80     Dim rstCustomers 'ADO Recordset
```

```

90    Dim mConn          'ADO Connection
100   Dim mCmd           'ADO Command

110   Dim vRecordCount   'Counter
120   Dim vbbgcolor      'Back Ground Color

130   'Erstellen des ADO Objekts
140   Set mConn = Server.CreateObject("ADODB.Connection")
150   Set mCmd = Server.CreateObject("ADODB.Command")
160   Set rstCustomers = Server.CreateObject("ADODB.Recordset")

170   'Verbindungsparameter setzen
180   mConn.CommandTimeout = 40
190   mConn.CursorLocation = 1 'Client Side
200   mConn.ConnectionString = "server=10.1.1.50;driver=MySQL;"
    "db=Meet_A_Geek;uid=root;pwd=tacobell"
210   mConn.Open

220   mCmd.ActiveConnection = mConn
230   mCmd.CommandType = 1 'Text
240   mCmd.CommandText = "SELECT * FROM Customers"

250   'Öffnen des Ergebnissatzes, der auf der Anfrage von nCmd basiert
260   rstCustomers.Open mCmd,,1,1

270%>

280<BODY>

290 <P align=center><STRONG>Here is a list of our Customers 300 </STRONG>      </
P>
310 <P align=center>
320 <TABLE align=center border=1 cellPadding=1 cellSpacing=1
    width="100%">

330 <TR>
340 <TD align=middle bgColor=silver><STRONG>First
350     Name</STRONG></TD>
360 <TD align=middle bgColor=silver><STRONG>Last
370     Name</STRONG></TD>
380 <TD align=middle bgColor=silver><STRONG>State</STRONG></TD>
390 </TR>

400<%
410   ' Setzt den Counter auf 0
420   vRecordCount = 0

```

```

430 ' Wandert durch den Ergebnissatz
440 Do Until rstCustomers.EOF

450     ' Dieses erzeugt alternierende Zeilenfarben und erleichtert so das
    Lesen.
460     If vRecordCount Mod 2 > 0 Then
470         vbgcolor = "#COCOCO"
480     Else
490         vbgcolor = "#FFFFFF"
500     End If

510     'Gibt das Ergebnis in einer Tabelle aus.
520     Response.Write "<TR>"
530     Response.Write "<TD align=middle bgcolor =" & vbgcolor & ">"
        & rstCustomers.Fields("First_Name") & "</TD>"
540     Response.Write "<TD align=middle bgcolor =" & vbgcolor &
        ">" & rstCustomers.Fields("Last_Name") & "</TD>"
550     Response.Write "<TD align=middle bgcolor =" & vbgcolor &
        ">" & rstCustomers.Fields("State") & "</TD>"
560     Response.Write "</TR>"

570     'Bewegt den Zeiger zum nächsten Datensatz
580     rstCustomers.MoveNext

590     'Erhöht den Datensatzzähler
600     vRecordCount = vRecordCount + 1
610 Loop

620 rstCustomers.Close
630 mConn.Close
640 %>
650 </TABLE></P>
660 </BODY>
670 </HTML>

```

Die erste Textzeile übermittelt dem Server lediglich, dass es sich hierbei um eine Active Server Page handelt, die VBScript benutzt. Die nächsten zwei Zeilen sind einfache HTML-Tags, wobei die erste deklariert, dass es sich um eine HTML-Seite handelt, und die zweite den <HEAD>-Abschnitt startet. Die nächste Zeile ist ein META-Tag, das dem Browser übermittelt, diese Datei nicht im Browsercache zwischenspeichern. Besonders für das Anzeigen von Datenbankergebnissen ist dies eine wichtige Zeile.

Würde der Browser seinen Zwischenspeicher (*Cache*) benutzen, kann es sein, dass er alte Daten ausgibt. Und wie wir alle wissen, sind alte Daten nicht immer korrekte Daten. Um sicherzustellen, dass Sie die neuesten und besten Daten bekommen, benutzen Sie das META-Tag. Die nächste Zeile beendet Ihren <HEAD>-Abschnitt.

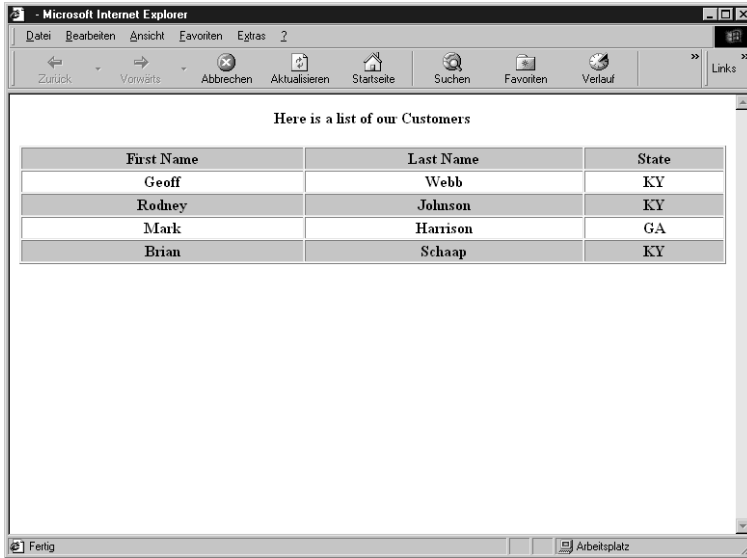


Abbildung 13.11:
Eine Webseite,
generiert mit
VBScript und
ASP

Die nächsten Zeilen werden auf dem Server ausgeführt, bevor sie zum Browser geschickt werden. Das wird durch `<%` kenntlich gemacht. Ein wichtiger Punkt ist, dass der `<HEAD>`-Abschnitt bereits zum Client geschickt wurde. Der von Ihnen erstellte Programmcode befindet sich zwischen `<HEAD>` und `<BODY>`, das heißt, der Client erhält sämtliche `<HEAD>`-Informationen, bevor der Code ausgeführt wird. In diesem Zwischenraum wird der Großteil des ASP-Programmierens bewerkstelligt. Auf diese Weise kann ein Teil der Informationen zum Client geschickt werden, während der Rest auf dem Server verarbeitet wird. Das ist eine Gefälligkeit für die Benutzer, damit sie wissen, dass etwas passiert. Die Variablen werden in den Zeilen 70 bis 120 deklariert.

Alle in VBScript deklarierten Variablen sind standardgemäß *Variant*-Typen. Wenn diese einen Wert zugewiesen bekommen, nehmen Sie den jeweils benötigten Typ automatisch an. Die ersten drei Variablen speichern Objekte, während `vRecordCount` eine Zahl und `vbgroundColor` eine Zeichenkette enthält.

Erstellen von ADO-Objekten

Die Zeilen 130 bis 160 sind ein bisschen interessanter.

An diesem Punkt erstellen Sie Ihre ADO-Objekte auf dem Server, was später in diesem Kapitel etwas ausführlicher behandelt wird. Sie setzen Ihre Variablen mit den Ergebnissen der `CreateObject`-Methode des `Server`-Objekts gleich. Sie fragen sich vielleicht, wo auf einmal das `Server`-Objekt hergekommen ist. Es ist ein eingebautes Objekt, das vom Internet Information Server (IIS) zur Verfügung gestellt wird. Das ist, was eine ASP-Seite zu einer ASP- und nicht zu einer Web-Seite macht. Es gibt sieben

eingebaute Objekte: das `Server`-Objekt, das `Application`-Objekt, das `Request`-Objekt, das `Response`-Objekt, das `Session`-Objekt, das `ObjectContext`-Objekt und das `ASPErr`-Objekt. Jedes von ihnen enthält eine Menge nützlicher Methoden und Eigenschaften. Dieses Buch behandelt allerdings nur diejenigen, die für unsere momentane Zielsetzung von Bedeutung sind.

Das Server-Objekt

Das `Server`-Objekt beinhaltet, wie Sie gesehen haben, eine sehr wichtige Methode – `CreateObject`. Diese Methode erstellt eine Instanz für ein benanntes Objekt auf dem Server. Vergessen Sie nicht, dass das Objekt, das Sie aufrufen, auf dem Server installiert und im Server-Register korrekt eingetragen sein muss. Falls nicht, werden Sie dieses Objekt nicht erstellen können. Das `Server`-Objekt enthält einige andere, kaum benutzte Methoden und ein Attribut. Diese sind hier nicht relevant und werden deshalb auch nicht behandelt.

Das Request-Objekt

Das `Request`-Objekt ist ein weiteres vom IIS zur Verfügung gestelltes, internes Objekt. Es beinhaltet viele häufig gebrauchte Methoden und Eigenschaften. Das `Request`-Objekt erhält die Werte, die in einer Anfrage vom Browser zum Server geschickt werden. In dem Gästebuch-Beispiel sendet ein Benutzer seine Kommentare. Diese werden zusammen mit einer Anfrage an den Web-Server geschickt. Um diese Kommentare zu erhalten, benutzen Sie das `Request`-Objekt. Durch die `QueryString`- oder `Form`-Attribute können Sie auf diese zugreifen. Die Werte werden in ein assoziatives Array oder eine Collection gespeichert, das heißt, sie werden in einem Name=Wert-Paar gespeichert. Wollen Sie auf diesen Wert zugreifen, benutzen Sie den Namen, der benutzt wurde, um den Wert in der Webseite zu erhalten. Im Gästebuch-Beispiel haben Sie die Kommentare über ein Formular zum Server geschickt. Die Kommentare wurden in der `textarea` namens `Comments` gespeichert. Wollen Sie diesen Wert von der Active Server Page zurückbekommen, könnten Sie die `QueryString`-Eigenschaft folgendermaßen benutzen:

```
Request.QueryString("Comments")
```

Der zurückgegebene Wert bildet den Inhalt der `textarea` namens `Comments`. Die `QueryString`-Eigenschaft enthält alle zwischen Webseiten gesendeten Anfragen – egal welchen Typs diese sind. Das bedeutet, dass die Informationen, die via HTML-Formularen oder Parametern in einer URL (`?Last=Newton&First=Steve`) geschickt werden, in einem `QueryString`-Attribut enthalten sind. Deshalb ist dieses auch das Standard-Attribut. In den meisten Fällen werden Sie ein `Request`-Objekt wie das folgende vorfinden:

```
Request("Comments")
```

und nicht das komplett ausgeschriebene

```
Request.QueryString("Comments")
```

Das Response-Objekt

Das `Response`-Objekt ist das Gegenstück zum `Request`-Objekt. Es schickt die Daten an den Client. Die am meisten verwendete Methode dieses Objekts ist die `Write`-Methode. Sie sendet die Ausgabe direkt an den Browser. Beachten Sie dabei, dass Sie eine Sprache benutzen, die der Browser versteht – das ist HTML. Deswegen muss Ihre Ausgabe korrekt in HTML-Tags formatiert sein. Falls nicht, kann es sein, dass kein Mensch Ihre Ausgabe jemals zu Gesicht bekommen wird.

Der nächste Teil des Listings richtet die Eigenschaften der ADO-Objekte, die sie vorhin erstellt haben, ein. Sie werden heute noch mehr über ADO erfahren. Im Moment bleiben Sie bei den Zeilen 280 bis 670 des Listing 13.1.

Der erste Punkt, den Sie sich hier merken müssen, ist, dass Sie VBScript in einer Active Server Page an jeder beliebigen Stelle platzieren können. Im Listing haben Sie etwas davon auch im `<BODY>`-Tag der Webseite. Dies ermöglicht Ihnen, den Programmcode von Client und Server zu mischen. Statisches kann immer mit einem normalen HTML-Programmcode erstellt werden, Dynamischeres hingegen mit VBScript.

Zeile 420, nach dem `<%>`-Tag, bringt Ihre Variable zurück auf 0. Danach benutzen Sie einen `DoWhile...Loop`-Block, um durch den Ergebnissatz zu wandern. Der Interpreter überprüft hier, ob er am Ende des Ergebnissatzes ist. Falls nicht, verarbeitet er die nächste Zeile, falls doch, beendet er die Schleife.

Innerhalb der Schleife (Zeilen 460 bis 500) befindet sich ein `If...Else`-Block. Hier ändern Sie die Farben der Zeilen in der Ausgabetablelle. Durch diesen kleinen Trick können Benutzer die Ergebnisse besser lesen. Sie teilen Ihren Zähler durch zwei. Setzen Sie die `Modulo()`-Funktion ein, um zu überprüfen, ob ein Rest bleibt. Wenn dem so ist, ist die Zeile ungerade, und sie wird silbern eingefärbt. Gibt es keinen Rest, ist die Zeile gerade, und sie bleibt weiß.

Der nächste Code-Block, die Zeilen 530 bis 550, geben die Werte Ihres Ergebnissatzes im Browser aus, wobei das Tabellenformat, mit dem Sie schon im statischen Teil des Quellcodes begonnen haben, beibehalten wird. Das muss auf diese Weise getan werden, weil Sie nicht wissen, wie viele Daten Sie haben. Daten sind ständig in Bewegung, deshalb können Sie nicht wissen, wie viele Datensätze Sie die ganze Zeit über haben. Denken Sie dabei an die Instandhaltung. Hätten Sie eine statische Webseite, die alle Ihre Kunden anzeigt, müssten Sie jedes Mal, wenn Sie einen neuen Datensatz hinzufügen oder einen Datensatz löschen, manuell eine neue Zeile einfügen – ein Alptraum! Das ist ein Grund, weshalb das Verschmelzen von Webseiten und Daten so beliebt ist.

Nun wandern Sie zum nächsten Datensatz und erhöhen die *Counter*-Variable um eins (Zeile 580). Gehen Sie nicht zum nächsten Datensatz, bleibt der Programmcode in einer Endlosschleife. Derselbe Inhalt würde wieder und wieder angezeigt werden. Das ist sehr frustrierend für einen Benutzer. Seien Sie also vorsichtig, wenn Sie im Web Schleifen einsetzen.

Am Ende des Programmcodes, nach der Schleife, werden Aufräumaktionen durchgeführt. Sie schließen erst Ihren Ergebnissatz und dann Ihre Verbindung. Daraufhin beenden Sie die serverseitige Ausführung der Webseite, und der Rest des statischen HTML wird ausgegeben.

Hoffentlich konnten Sie an Hand dieser Beispiele die Funktionsweise einer Active Server Page verstehen und wissen jetzt, wie diese Ihnen von Nutzen sein kann. Wie gesagt, dies war nur ein flüchtiger Blick auf die VBScript- und Active-Server-Pages-Technologie. Wenn Sie mehr als nur grundlegende Entwicklung betreiben wollen, müssen Sie mehr über dieses Thema lesen.

ActiveX Data Objects (ADO)

Wie bereits erwähnt, sind die ActiveX Data Objects (ADO) eine Gruppe von Objekten, die zur Kommunikation mit einer Datenbank benutzt werden. Unter diesen Objekten herrscht eine Hierarchie vor, die aber nicht zwingend ist. Das bedeutet, dass sie alle unabhängig voneinander erstellt werden können. Dies verschafft größere Flexibilität und einfacheren Gebrauch. Dieser Abschnitt wird einige der Methoden und Eigenschaften erläutern, die benutzt werden können, um von der MySQL ODBC API (*Application Programming Interface*) zu profitieren.

ADO besteht im Wesentlichen aus drei Objekten. Diese drei Objekte umfassen alles, was Sie mit einer Datenbank machen können. Es handelt sich hierbei, um das *Connection-Objekt*, das *Recordset-Objekt* und das *Command-Objekt*. Mit diesen drei Objekten können Sie sich mit jeder Datenbank verbinden und Ergebnisse abfragen und ausgeben.

Eigenschaften und Methoden des Connection-Objekts

Bei jeder API muss man zuerst eine Verbindung herstellen. ADO bildet da keine Ausnahme. Dazu verwendet es das *Connection-Objekt*, das viele Eigenschaften und Methoden beinhaltet, über die Tabelle 13.1 informiert.

Methoden oder Eigenschaften	Beschreibung
Attributes	Zeigt eine oder mehr Eigenschaften des <code>Connection</code> -Objekts an.
CommandTimeout	Zeigt an, wie lange Sie bei der Ausführung einer Anweisung warten, bevor diese beendet wird. Der Standardwert ist 30 Sekunden.
ConnectionString	Die wichtigste Eigenschaft des <code>Connection</code> -Objekts. Enthält alle nötigen Informationen, um eine Verbindung zu einer Datenbank herzustellen.
ConnectionTimeout	Zeigt an, wie lange bei einem Verbindungsaufbau gewartet wird, bis der Versuch abgebrochen und ein Fehler erzeugt wird.
CursorLocation	Zeigt an, wo der Cursor benutzt werden soll. Aus verschiedenen Gründen ist es besser, den Cursor auf dem Client einzusetzen. Ein Grund besteht darin, sicherzustellen, dass die gewünschten Operationen auch unterstützt werden.
DefaultDatabase	Wenn keine Datenbank im <code>ConnectionString</code> angegeben ist, wird die Verbindung hier nach einem Datenbanknamen suchen.
Open()	Öffnet eine Verbindung zu einer Datenbank mit den Parametern aus dem <code>ConnectionString</code> .
Provider	Gibt den Namen des Providers an, der in der Verbindung benutzt wird. Bei MySQL wäre dies der ODBC OLE DB-Provider.
State	Zeigt den aktuellen Status der Verbindung an.
Version	Zeigt die ADO-Version des momentan benutzten Objekts an.

Tabelle 13.1: Methoden und Eigenschaften des `Connection`-Objekts

Herstellen einer ADO-Verbindung

Um mittels VBScript und ADO eine Verbindung zu einer MySQL-Datenbank herzustellen, müssen mehrere Schritte ausgeführt werden:

1. Deklarieren Sie eine Variable, aus der dann das `Connection`-Objekt wird:

```
Dim mConn
```

2. Erstellen Sie das Objekt und setzen Sie Ihre Variable mit diesem gleich:

```
Set mConn = Server.CreateObject("ADODB.Connection")
```

3. Richten Sie die Eigenschaften des Connection-Objekts ein:

```
mConn.CommandTimeout = 40  
mConn.ConnectionTimeout = 40  
mConn.CursorLocation = 3
```

Eine kleine Nebenbemerkung: IIS 4 enthält einen Fehler. Der Interpreter arbeitet VBScript-Konstanten nicht richtig ab, so dass Sie statt der `adUseClient`-Konstante die Zahl 3 angeben müssen (diese wird normalerweise durch die Konstante repräsentiert). Das gleiche Problem tritt auf, wenn Sie die `open()`-Anweisung des `Recordset`-Objekts verwenden.

4. Erstellen Sie den ConnectionString:

```
mConn.ConnectionString="server=10.1.1.50;db=Meet_A_Geek;" &  
driver=MySQL;uid=root;pwd=tacobell"
```

Der Verbindungs-String besteht aus folgenden Informationen:

- ▶ *Server* – kann ein Name oder eine Adresse sein
- ▶ *db* – Der Name der Datenbank
- ▶ *Treiber* – Der Name des zu benutzenden ODBC-Treibers
- ▶ *uid* – Der Benutzername einer Person, der es erlaubt ist, eine Verbindung zu dieser Datenbank herzustellen.
- ▶ *pwd* – Das Passwort des uid

So lange die Komponenten des Verbindungs-Strings alle vorhanden sind, können sie in jeder Reihenfolge erscheinen. Verwenden Sie zur Verbindungsaufnahme eine Datenquelle, benötigen Sie in Ihrem Verbindungs-String lediglich den Namen der Datenquelle.

Hätten Sie eine Datenquelle namens »MeetAGeek«, könnten Sie den folgenden Verbindungs-String benutzen, solange der Datenquellename richtig konfiguriert ist:

```
mConn.ConnectionString = "DSN=MeetAGeek"
```

5. Öffnen Sie jetzt die Verbindung. Hierbei wird der Verbindungs-String an den Treiber übermittelt und die Verbindung zur Datenbank hergestellt. Benutzen Sie dazu folgende Syntax:

```
mConn.Open
```

Das ist alles, was Sie machen müssen. Wenn Sie diesen fünf einfachen Schritten folgen, können Sie eine Verbindung zu jeder MySQL-Datenbank oder zu jeder anderen Datenbank herstellen. Das ist das Schöne an dieser Technologie. Der hier benutzte Programmcode kann ebenso auf jede andere Datenbank angewandt werden – so lange Sie die richtigen Informationen angeben.

Es wird empfohlen, die Verbindung ohne einen DSN herzustellen. Auf diese Weise können Sie sich zusätzliche Arbeit sparen und sind nicht auf die Hilfe eines anderen angewiesen. Bei einer DSN-Verbindung muss jemand für Sie einen System-DSN auf dem Server erstellen. Lokal ist das nicht schwer, versuchen Sie es aber auf einem entfernten Server, werden Sie einige Probleme haben. Zudem verlassen Sie sich auf Informationen, die außerhalb Ihres Einflussbereichs liegen. Was, wenn der Server abstürzt und NT neu gestartet werden muss? Was geschieht dann mit all Ihren Programmen, die sich auf diesen DSN stützen? Was, wenn vergessen wird, Ihren DSN wieder einzurichten? Dann stecken Sie in der Klemme. Am besten, Sie benutzen Verbindungen ohne DSN. Auf diese Weise haben Sie alles selbst in der Hand.

Eigenschaften und Methoden des Command-Objekts

Ein weiteres wichtiges Objekt in der ADO-Objekt-Liste ist das `Command`-Objekt. Dieses Objekt ist vielfach einsetzbar. Es kann SQL-Anweisungen ausführen, die keinen Ergebnissatz zurückgeben; es kann auch eingesetzt werden, um die Struktur Ihrer Datenbank zu manipulieren, und es kann datenbankspezifische Anweisungen erlassen. Ebenso wie das `Connection`-Objekt besitzt das `Command`-Objekt viele Methoden und Eigenschaften. Diese sind mit einer kurzen Beschreibung in der unten stehenden Tabelle aufgelistet.

Methoden oder Eigenschaften	Beschreibung
<code>ActiveConnection</code>	Gibt die Verbindung an, die für alle Anweisungen benutzt wird. Diese kann auf ein offenes <code>Connection</code> -Objekt gesetzt werden.
<code>CommandText</code>	Ein String, der als Befehl agiert. Hierbei kann es sich um eine SQL-Anweisung oder in einigen Fällen auch um den Namen einer Tabelle handeln. So ziemlich alles, was Sie in den MySQL-Monitor eingeben können, kann auch hier angegeben werden.
<code>CommandTimeout</code>	Zeigt an, wie lange Sie bei der Ausführung einer Anweisung warten, bevor diese beendet wird.
<code>CommandType</code>	Zeigt an, welchen Befehlstyp Sie verwenden. Zwei Werte sind in MySQL möglich: Die Interpretation als Befehlszeile (1) oder als Tabellename (2). Die anderen Befehlstypen sind nicht auf MySQL anwendbar, da sie sich auf Merkmale beziehen, die MySQL nicht beinhaltet (um genau zu sein: Gespeicherte Prozeduren(4)).

Tabelle 13.2: Eigenschaften und Methoden des Command-Objekts

Methode oder Eigenschaft	Beschreibung
Execute()	Diese Methode veranlasst die Ausführung der Befehle aus dem <code>CommandText</code> .
Prepared	Ein Boolescher Wert (WAHR oder FALSCH), der anzeigt, ob vom Provider eine vorbereitete Anweisung aus dem <code>CommandText</code> vor der Ausführung erstellt werden soll oder nicht. Das kann die Performance einer Anfrage ab der zweiten Ausführung verbessern. MySQL unterstützt diese Option. #Nein, bei der ersten Anfrage wird kompiliert (und ausgeführt), ab der zweiten Anfrage wird das Kompilat ausgeführt.
State	Zeigt an, ob die Verbindung des <code>Command</code> -Objekts geöffnet oder geschlossen ist.

Tabelle 13.2: Eigenschaften und Methoden des `Command`-Objekts (Forts.)

Arbeiten mit dem `Command`-Objekt

Wie bereits gesagt, ist das `Command`-Objekt vielfach einsetzbar. Um das `Command`-Objekt zu benutzen, führen Sie die folgenden Schritte aus:

1. Deklarieren Sie zunächst eine Variable, die Ihr `Command`-Objekt enthält:

```
Dim mCmd
```
2. Benutzen Sie die `CreateObject`-Methode des `Server`-Objekts, um das `Command`-Objekt zu erstellen:

```
Set mCmd = Server.CreateObject("ADODB.Command")
```
3. Teilen Sie dem `Command`-Objekt mit, welche Verbindung benutzt werden soll:

```
mCmd.ActiveConnection = mConn
```
4. Richten Sie einige der Eigenschaften des neuen `Command`-Objekts ein. An dieser Stelle bestimmen Sie die Wartezeit und welchen Befehlstyp Sie erteilen wollen sowie die eigentliche Anweisung:

```
mCmd.CommandType = 1
mCmd.CommandTimeout = 40
mCmd.CommandText = "SELECT * FROM Customers"
```

5. Je nach Befehl können Sie die `Execute`-Methode benutzen, um Ihren Eingriff auszuführen. Wenn Sie beispielsweise einen Datensatz aus der `Customers`-Tabelle löschen wollten, könnten Sie folgendermaßen vorgehen (vorausgesetzt Sie haben bereits eine Verbindung zu der Datenbank hergestellt, die das `mConn`-Objekt benutzt):

```
Dim mCmd
Set mCmd = Server.CreateObject("ADODB.Command")
mCmd.ActiveConnection = mConn
mCmd.CommandType = 1
mCmd.CommandText = "DELETE FROM Customers WHERE Customer_ID=
1345"
mCmd.Execute
```

Wie Sie sehen können, beinhaltet das `Command`-Objekt, wie Sie Ihre Anweisungen der Datenbank übermitteln. Was wäre, wenn Sie die Ergebnisse einer Anfrage haben wollten? Wie würden Sie vorgehen, wenn Sie diese mittels eines Befehls erhalten wollten? Sie müssten dazu das `Recordset`-Objekt benutzen.

Eigenschaften und Methoden des Recordset-Objekts

Das `Recordset`-Objekt besitzt ebenso wie die anderen ADO-Objekte viele Eigenschaften und Methoden, die sich mit den von der Datenbank zurückgegebenen Daten befassen. Die Tabelle 13.3 enthält einige dieser Eigenschaften und Methoden.

Methoden oder Eigenschaft	Beschreibung
<code>AddNew()</code>	Erstellt einen neuen Datensatz für ein aktualisierbares <code>Recordset</code> -Objekt.
<code>BOF</code> , <code>EOF</code>	Gibt entweder <code>WAHR</code> oder <code>FALSCH</code> zurück, je nachdem, ob Sie sich am Anfang oder Ende des Ergebnissatzes befinden.
<code>Clone()</code>	Erstellt ein Duplikat eines <code>Recordset</code> -Objekts aus einem vorhandenen <code>Recordset</code> -Objekt.
<code>Close()</code>	Schließt ein geöffnetes <code>Recordset</code> -Objekt und alle abhängigen Objekte.
<code>CursorType</code>	Kennzeichnet den <code>Cursor</code> typ, der in einem <code>Recordset</code> -Objekt verwendet wird. <code>Cursor</code> typen werden später näher erläutert.
<code>Fields</code>	Enthält Informationen über jeden einzelnen Datensatz in einem <code>Recordset</code> -Objekt.

Tabelle 13.3: Methoden und Eigenschaften des `Recordset`-Objekts

Methoden oder Eigenschaft	Beschreibung
Find (Criteria, Skip, Dir, Start)	Durchsucht ein Recordset-Objekt nach den spezifizierten Kriterien.
LockType	Zeigt den Typ der Sperren an, die auf das Recordset-Objekt angewandt werden. Dies wird später näher erläutert.
Move()	Verschiebt die Position des aktuellen Datensatzes in einem Recordset-Objekt.
MoveNext()	Wechselt zum nächsten Datensatz in einem Recordset-Objekt.
MoveFirst()	Wechselt zum ersten Datensatz in einem Recordset-Objekt.
MoveLast()	Wechselt zum letzten Datensatz in einem Recordset-Objekt.
MovePrevious ()	Wechselt zum vorhergehenden Datensatz in einem Recordset-Objekt.
MaxRecords	Gibt die Höchstzahl der Datensätze an, die in einem Ergebnis enthalten sein können. Stellt die gleiche Funktionalität wie das LIMIT-Schlüsselwort in einer SQL-Anweisung zur Verfügung.
Open(Source, ActiveConnection, CursorType, LockType)	Öffnet das Recordset-Objekt aufgrund der zur Verfügung gestellten Kriterien.
RecordCount	Zeigt an, wie viele Datensätze im aktuellen Recordset-Objekt vorhanden sind.
Sort	Übermittelt dem Recordset-Objekt, welche Felder sortiert werden sollen und in welcher Reihenfolge (auf- oder absteigend). Entspricht dem Gebrauch einer ORDER BY-Klausel in einer SQL-Anweisung.
Source	Gibt die Datenquelle an. Das kann ein Command-Objekt, eine SQL-Anweisung oder ein Tabellename sein.
Format()	Speichert das Recordset-Objekt an einer spezifizierten Stelle im angegebenen Format. Das kann XML sein.
State	Zeigt an, ob ein Recordset-Objekt geöffnet oder geschlossen ist.
Status	Gibt eine Zahl zurück, die den Status des aktuellen Datensatzes in Bezug auf eine Operation anzeigt.
Update	Das Gegenstück zur addNew() -Methode. Speichert alle Änderungen, die Sie am aktuellen Datensatz eines Recordset-Objekts vornehmen, in der Datenbank.

Tabelle 13.3: Methoden und Eigenschaften des Recordset-Objekts (Forts.)

Arbeiten mit dem Recordset-Objekt

Sie benutzen das `Recordset`-Objekt am besten in Verbindung mit dem `Command`- und dem `Connection`-Objekt. Führen Sie folgende Schritte aus, um ein `Recordset`-Objekt zu öffnen:

1. Deklarieren Sie eine Variable, die das `Recordset`-Objekt enthält:

```
Dim rstCustomers
```

2. Setzen Sie diese Variable den Ergebnissen der `CreateObject`-Methode eines `Server`-Objekts gleich.

```
Set rstCustomers = Server.CreateObject("ADODB.Connection")
```

3. Öffnen Sie das `Recordset`-Objekt. Es gibt viele Möglichkeiten, dies zu tun. Sie alle erfordern, dass Sie eine Quelle angeben. Diese Quelle kann ein `Command`-Objekt, eine SQL-Anfrage oder ein Tabellennamen sein. Sie verwenden am besten ein `Command`-Objekt, das bereits eine aktive Verbindung eingerichtet hat:

```
rstCustomers.Open mCmd,,1, 1
```

Diese Anweisung öffnet ein auf dem `Command`-Objekt basierendes `Recordset`-Objekt. Die Angabe einer aktiven Verbindung als zweites Argument dieser Methode ist nicht nötig, da Sie ein `Command`-Objekt verwenden, das bereits eine aktive Verbindung eingerichtet hat. Die nächsten beiden Argumente geben den Cursortyp und den zu benutzenden Sperrentyp an. An Stelle der Microsoft-Konstante wird eine Zahl benutzt, da der IIS *buggy* ist, und bei der Verwendung der Konstante einen Fehler erzeugen würde.

Wenn alles richtig übergeben wurde und die SQL-Anweisung in Ihrem `Command`-Objekt gültig ist, wird ein `Recordset`-Objekt geöffnet, das die gewünschten Daten enthält. Wie Sie danach verfahren, liegt ganz bei Ihnen. Sie können Datensätze manipulieren, ausgeben, ändern oder hinzufügen.

Es gibt vier Parameter, die Sie benutzen können, wenn Sie ein `Recordset`-Objekt öffnen. Diese sind `Source`, `ActiveConnection`, `CursorType` und `LockType`. Es gibt zwar noch einen fünften Parameter, der allerdings nie verwendet wird, da er laut Microsoft Fehler produzieren kann.

`Source` ist der erste Parameter. Wie bereits erläutert, gibt es im Wesentlichen drei Typen von Quellen. Im Normalfall benutzen Sie das `Command`-Objekt als Ihre `Source`. Das ist die flexibelste Methode, ein `Recordset`-Objekt zu öffnen. Zudem können Sie auch eine SQL-Anweisung als Quelle verwenden. Die letzte Möglichkeit besteht darin, einen Tabellennamen als Quelle für Ihre Daten im `Recordset`-Objekt zu benutzen. Welche Möglichkeit Sie wählen, liegt bei Ihnen. Die erste Methode ist definitiv die beste, doch wenn Sie eine andere Methode wählen, sollten Sie nicht vergessen, eine `ActiveConnection` einzurichten

Die `ActiveConnection`, der zweite, fakultative Parameter in der `Open`-Methode eines `Recordset`-Objekts, ist ein für alle zugängliches `Connection`-Objekt. Benutzen Sie ein `Command`-Objekt, dessen aktive Verbindung bereits eingerichtet wurde, können Sie diese Option auslassen. Wenn Sie allerdings kein `Command`-Objekt verwenden, müssen Sie an dieser Stelle ein `Connection`-Objekt benutzen.

Der nächste Parameter `CursorType` definiert, welche Aktionen im `Recordset`-Objekt ausgeführt und wie die Daten im `Recordset`-Objekt angezeigt werden. Im Folgenden sehen Sie die Parameter, unter denen Sie auswählen können:

- ▶ *Keyset* – Daten, die seit dem Öffnen dieses Cursortyps hinzugefügt wurden, sind nicht sichtbar. Dieser Typ ermöglicht Ihnen, sich innerhalb des `Recordset`-Objekts vorwärts und rückwärts zu bewegen. Daten können sowohl hinzugefügt als auch gelöscht werden.
- ▶ *Dynamic* – Dies ist der flexibelste Cursor. Sie können sich im `Recordset`-Objekt frei nach hinten und nach vorne bewegen. Jegliche Datensätze, die seit dem Öffnen dieses Typs gelöscht oder hinzugefügt wurden, verändern das `Recordset`-Objekt. Darin spiegeln sich die Daten der Datenbank wider.
- ▶ *Static* – Dieser Typ kann nicht aktualisiert werden, noch sind jegliche Veränderungen, die nach dem Erstellen dieses `Recordset`-Objekts getätigt wurden, sichtbar. Dieser Typ wird im Allgemeinen bei Berichten oder statischen Ausgaben verwendet. Das Standard-`Recordset`-Objekt ist 0 – nur vorwärts. Dieses entspricht einem statischen Cursor – mit dem Unterschied, dass Sie sich innerhalb der Datensätze nur nach vorne bewegen können. Dieses `Recordset`-Objekt eignet sich besonders für das schnelle, einmalige Durchwandern Ihrer Daten.
- ▶ *Lock* – MySQL ermöglicht Ihnen im Gegensatz zu manch anderen Datenbanken, das Sperren mittels SQL-Anweisungen zu kontrollieren. Beim Versuch, die heterogene Umgebung zu beseitigen, hat Microsoft diese Option aufgenommen. Sie können bestimmen, welchen Sperrentyp Sie benutzen wollen. Der ODBC-Treiber erlässt die Befehle für Sie. Im Folgenden sind die verfügbaren Sperren aufgelistet:
 - ▶ *read only*: Die Daten sind schreibgeschützt. Eine Veränderung ist nicht möglich.
 - ▶ *pessimistic locking*: Der ODBC-Treiber unternimmt alles, um das richtige Bearbeiten der Daten sicherzustellen. Er sperrt die Tabelle, bevor eine Änderung vorgenommen wird, und hebt die Sperre wieder auf, wenn die Aktualisierung beendet ist.
 - ▶ *optimistic locking*: Der Provider sperrt die Tabelle nur, wenn die `Update`-Methode aufgerufen wird. *Pessimistic locking* ist die beste Wahl. Hier werden Tabellen gesperrt, wenn Sie sie benutzen. Falls Sie Bedarf an weiteren Sperren haben, erlassen Sie diese Anweisungen mit dem `Command`-Objekt.

Die Meet_A_Geek-Datenbank im Web

Nun, da Sie die ADO-Objekte kennen gelernt, und gesehen haben, was Sie damit bewerkstelligen können, werfen Sie einen Blick darauf, wie Sie diese in einer Alltagssituation anwenden können. Sie werden eine Reihe von ASP-Seiten erstellen, die es einem Anwender ermöglichen, eine persönliche Anzeige via Web aufzugeben. Die eingegebenen Daten auf der Webseite werden in einer MySQL-Datenbank gespeichert.

Erstellen der Webseite

Zuerst müssen Sie die Webseite entwerfen. Sie werden hier das gleiche Design benutzen, das Sie in Kapitel 3 »Das Entwerfen Ihrer ersten Datenbank« entworfen haben. Im Folgenden sehen Sie einen HTML-Teilausschnitt dieser Seite:

```
<form method="POST" action="PostAd.asp">
<p> To post your free ad, please answer the following questions.&nbsp;
  We will not give your name or email address to anyone without your
  consent.&nbsp;&nbsp;&nbsp;Items with a
  ( <font color="#FF0000">*</font><font color="#000000">)</font>
  are required fields.</p>
<P>Please provide the following contact information:
  <font color="#FF0000" size="4" face="Arial">*</font></P>
<BLOCKQUOTE>
<TABLE>
<TR>
<TD ALIGN="right">
<EM>First Name </EM></TD>
<TD>
<INPUT NAME="Contact_FirstName" SIZE=25>
</TD>
</TR>
<TR>
<TD ALIGN="right">
<EM>Last Name </EM></TD>
<TD>
<INPUT NAME="Contact_LastName" SIZE=25>
</TD>
</TR>
<TR>
<TD ALIGN="right">
<EM>Middle Initial </EM></TD>
<TD>
<INPUT NAME="Contact_MiddleInitial" SIZE=4 MAXLENGTH=1>
```

```

</TD>
</TR>
<TR>
<TD ALIGN="right">
<EM>Home Phone &nbsp;&nbsp;&nbsp;<br>
</EM><font size="2" face="Arial">(no cashes or spaces,&nbsp;&nbsp;&nbsp;<br>
include area-code)</font></TD>
<TD>
<INPUT NAME="Contact_HomePhone" SIZE=25 MAXLENGTH=10>
</TD>
</TR>
<TR>
<TD ALIGN="right">
<EM>E-mail</EM></TD>
<TD>
<INPUT NAME="Contact_Email" SIZE=25>
</TD>
</TR>
</TABLE>
</BLOCKQUOTE>
<P>
Please describe yourself:<font size="4" face="Arial">
<font color="#FF0000">*</font></font></P>
<BLOCKQUOTE>
<TABLE>
<TR>
<TD ALIGN="right">
<EM>Age</EM></TD>
<TD>

<INPUT NAME="Personal_Age" SIZE=3 MAXLENGTH=2>
</TD>
</TR>

```

----- Code geht hier noch weiter -----

Der Großteil des Programmcodes dieser Seite wurde hier ausgelassen, da ich Ihnen zeigen wollte, wie Daten von diesem Formular zum nächsten übergeben werden. Dies ist eine reguläre HTML-Seite. Sie enthält ein Formular und eine Reihe von Elementen innerhalb dieses Formulars. Die erste Zeile des Listings besteht aus dem `<FORM>`-Tag. Form-Tags kann eine `POST`- oder `GET`-Methode übergeben werden. Hier wird die `POST`-Methode verwendet. Die `POST`-Methode bewirkt, dass alle Elemente dieses Formulars in `Name=Wert`-Paaren an die nächste angeforderte Seite übergeben werden, die im `ACTION`-Attribut des `<FORM>`-Tags angegeben wird. In diesem Beispiel werden nach dem Drücken des »Submit«-Buttons alle `Name=Wert`-Paare an die Seite `PostAd.asp` übermittelt.

Der Rest des Programmcodes beinhaltet die Elemente des Formulars. Jedes Element wird durch ein `<input>`-Tag eingeleitet und hat einen eindeutigen Namen. Der Benutzer gibt zum Beispiel seinen Namen in das erste Feld des Formulars ein. Sie sehen im Quellcode, dass der Name dieses Elements `Contact_FirstName` ist. Was auch immer der Benutzer in dieses Feld eingibt, es ist der Wert, der in `Contact_FirstName` gehalten wird. Gibt der Benutzer also *Larry* in das Feld ein, würde das Name=Wert-Paar, das an den Server übergeben wird, `Contact_FirstName=Larry` sein.

Das Aufgeben einer Anzeige

Der interessantere Teil des Programmcodes spielt sich auf der nächsten Seite ab. Werfen Sie zuerst einen Blick auf den gesamten Programmcode in Listing 13.3. Anschließend wird der Code Zeile für Zeile erklärt.

Listing 13.3: Aufgeben einer Anzeige auf einer Webseite

```

10 <%@ Language=VBScript %>
20 <HTML>
30 <HEAD>
40 </HEAD>
50 <%
60     'Variablen Deklaration
70     Dim mCmd
80     Dim mConn
90     Dim strSQL

100     'Erstellen der ADO-Objekte
110     Set mCmd = Server.CreateObject("ADODB.Command")
120     Set mConn = Server.CreateObject("ADODB.Connection")

130     'Erstellen der Verbindung
140     mConn.CommandTimeout = 40
150     mConn.ConnectionTimeout = 40
160     mConn.ConnectionString = "server=10.1.1.50;driver=MySQL;" & _
        "db=Meet_A_Geek;uid=root;pwd=tacobell"
170     mConn.Open

180     'Erstellen des SQL-Strings
190     strSQL = "INSERT INTO Customers"
200     strSQL = strSQL & "(Customer_ID, First_Name, Last_Name, Address,"
210     strSQL = strSQL & " City, State, Zip, Email, Age, Gender, Race, "
220     strSQL = strSQL & "Eye_Color, Activity, Movie, Occupation.
        moker)"
230     strSQL = strSQL & " VALUES(NULL, " & _

```

```
240     Request("Contact_FirstName") & ", "
250     strSQL = strSQL & Request("Contact_LastName") & ", "
260     strSQL = strSQL & Request("Contact_MiddleInitial") & ", "
270     strSQL = strSQL & Request("Contact_Address") & ", "
280     strSQL = strSQL & Request("Contact_City") & ", "
290     strSQL = strSQL & Request("Contact_State") & ", "
300     strSQL = strSQL & Request("Contact_Zip") & ", "
310     strSQL = strSQL & Request("Contact_Email") & ", "
320     strSQL = strSQL & Request("Contact_Age") & ", "

330     'Geschlecht überprüfen
340     If Request("Contact_Gender") = "Male" Then
350         strSQL = strSQL & "M, "
360     Else
370         strSQL = strSQL & "F, "
380     End If

390     strSQL = strSQL & Request("Contact_EyeColor") & ", "
400     strSQL = strSQL & Request("Contact_Activity") & ", "
410     strSQL = strSQL & Request("Contact_Movie") & ", "
420     'Raucher (Ja/Nein)
430     If Request("Smoker") > 1 Then
440         strSQL = strSQL & "Y, "
450     Else
460         strSQL = strSQL & "N, "
470     End If
480

490     'Erstellen des Command-Objekts
500     mCmd.ActiveConnection = mConn
510     mCmd.CommandTimeout = 40
520     mCmd.CommandType = 1

530     'Dem Command-Object die SQL-Anweisung hinzufügen
540     mCmd.CommandText = strSQL

550     'Ausführung der Anweisung
560     mCmd.Execute
570 %>

580 <BODY>

590<P>Thank you, <%Response.Write Request("Contact_FirstName%> </P>

600 </BODY>
610 </HTML>
```

Die erste Zeile teilt dem Server mit, dass in dieser Serverpage, wie in den meisten ASPs, VBScript eingesetzt wird. Die Zeilen 10 bis 40 zeigen dem Browser an, dass es sich hierbei um ein HTML-Dokument handelt. Es folgt der <HEAD>-Abschnitt.

Erst im nächsten Abschnitt beginnt der aktive Teil der Server Page (Zeile 50). In Zeile 70, 80 und 90 werden die Variablen deklariert. In ASP ist es geläufig, die Variablen in dieser Weise zu benennen. Das `m` zeigt an, dass es sich hierbei um eine Klassen-Variable handelt. Alles, was hinter dem `m` steht, ist eine Abkürzung für den Objekt-Typ. Die dritte Variable enthält Ihre SQL-Anweisung. Da diese ziemlich lang wird, ist es am besten, die Anweisung in einer Variablen zu speichern, die Sie dann als `CommandText` übergeben.

Es werden Instanzen der Objekte erstellt, die den Variablen in den Zeilen 110 und 120 gleichgesetzt werden.

Der nächste Abschnitt des Programmcodes bestimmt die Eigenschaften des `Connection`-Objekts. Die Zeilen 140 und 150 legen die maximale Ausführungszeit des `Command`- und `Connection`-Objekts auf 40 Sekunden fest. Je nach Netzwerkgeschwindigkeit können diese Werte höher oder niedriger gewählt werden. Für den Internetgebrauch sollten diese Werte etwas höher sein als für den Intranetgebrauch. In Zeile 160 wird der `Connection`-String erstellt. Beachten Sie, dass hier kein DSN verwendet wird. Da es sich hierbei um eine Internetseite handelt, wollen Sie sich bei der Bereitstellung und Aufrechterhaltung eines DSN nicht auf Ihren ISP verlassen. Das ist der Grund dafür, dass Sie eine Verbindung ohne DSN benutzen. Wie Sie sehen, werden der Treibername, der Namen der Datenbank, der Benutzername und das Passwort bereitgestellt. Beachten Sie die Schreibweise des Treibers. Das ist einer der seltenen Fälle, in der Microsoft zwischen Groß- und Kleinschreibung unterscheidet. Der Treibername muss exakt mit dem Namen des installierten Treibers übereinstimmen (sowohl in Groß- und Kleinschreibung als auch bei der Zeichensetzung). Hier wird ein einfacher Treibername gewählt. Im Gegensatz zum Access-Treiber (Microsoft Access Driver (*.mdb)) erscheint MySQL als Treibername ziemlich einfach. In Zeile 170 wird dann schließlich die Verbindung hergestellt.

Würden Sie versuchen, eine Verbindung ohne Angabe der Variablen herzustellen, würden Sie einen Fehler erhalten. Diese Variablen müssen vor dem Öffnen der Verbindung definiert werden. Hierbei ist die Reihenfolge von Bedeutung.

Der folgende Abschnitt des Programmcodes (Zeilen 180 bis 480) erstellt den SQL-Anweisungs-String, bei dem es sich um eine gewöhnliche `INSERT`-Anweisung handelt. Richtig interessant wird es erst, wenn Sie die Werte für die einzelnen Spalten abfragen. Um das zu tun, verwenden Sie das `Request`-Objekt. Vergessen Sie nicht, dass wegen des `POST`-Attributs in Ihrem Formular alle Name=Wert-Paare bei der nächsten Seitenanfrage übermittelt werden. Alles, was Sie zu tun, ist, diese entgegenzunehmen. Sie haben zwei Auswahlfelder in Ihrem Formular – eines für das Geschlecht und eines, um zu erfahren, ob der Anwender Raucher ist (Zeile 330 bis 380 und 430 bis 480).

Sie benutzen diese Auswahlfelder, weil die Spalten in der Datenbank vom Typ `ENUM` sind und diese nur zwei verschiedene Werte entgegennehmen können. So stellen Sie schon im Formular sicher, dass Sie passende Werte erhalten.

Ab Zeile 500 beginnen Sie, das `Command`-Objekt zu erstellen. Sie legen die maximale Ausführungszeit in Zeile 510 und die Eigenschaften des Befehlstyps in Zeile 520 fest. Zudem bestimmen Sie die Verbindung für das `Command`-Objekt (Zeile 500). Danach wird der `CommandText` mit der SQL-Anweisung gleichgesetzt (Zeile 540) und die Anweisung ausgeführt (Zeile 560). Wenn keine Fehler auftreten, werden die Daten in die Datenbank eingefügt.

Der letzte Abschnitt (Zeile 580 bis 610) macht deutlich, dass Sie serverseitigen Programmcode an jeder Stelle der Webseite einfügen können. Es wird jedoch nicht empfohlen, dass Sie häufig zwischen server- und clientseitigem Script hin- und herwechseln. Das könnte nämlich die Anzeige der Seite im Browser verlangsamen. Für kleinere Seiten, die wenig anzuzeigen haben, ist es dennoch in Ordnung. In unserem Beispiel personalisieren Sie die Webseite dadurch, dass Sie dem Anwender eine kleine Danksagung unter Nennung seines Vornamens zurückschicken.

Diese Webseite hat Ihnen gezeigt, wie Sie einer MySQL-Datenbank Daten hinzufügen können, indem Sie ADO, VBScript und Active Server Pages verwenden. Aber was, wenn Sie ein paar Informationen anzeigen möchten? Für die Anzeige von Daten wird eine andere Methode verwendet. Sie müssen zusätzlich zu den `Command`- und `Connection`-Objekten ein `Recordset`-Objekt benutzen.

Ausgabe von Daten auf der Webseite

Jetzt werden Sie eine Webseite erstellen, die all Ihre Kunden anzeigt. Diese Seite kann entweder durch einen Hyperlink oder direkt aufgerufen werden. Wie schon beim vorherigen Beispiel wird der Programmcode in voller Länge in L dargestellt und anschließend Stück für Stück erläutert. Der folgende Programmcode wurde schon zuvor in diesem Kapitel gezeigt. Da Sie jetzt ein besseres Verständnis von ADO haben, werden wir diesen Code nochmals durchgehen.

```
10 <%@ Language=VBScript %>
20 <HTML>
30 <HEAD>
40 <META HTTP-EQUIV="Expires" CONTENT="0">
50 </HEAD>
60 <%
70     ' Variablen Deklaration
80     Dim rstCustomers 'ADO Recordset
90     Dim mConn        'ADO Connection
100    Dim mCmd          'ADO Command
```

```

110 Dim vRecordCount 'Counter
120 Dim vbbgcolor      'Back Ground Color

130 'Erstellen des ADO Objekts
140 Set mConn = Server.CreateObject("ADODB.Connection")
150 Set mCmd = Server.CreateObject("ADODB.Command")
160 Set rstCustomers = Server.CreateObject("ADODB.Recordset")

170 'Verbindungsparameter setzen
180 mConn.CommandTimeout = 40
190 mConn.CursorLocation = 1 'Client Side
200 mConn.ConnectionString = "server=10.1.1.50;driver=MySQL;"
    "db=Meet_A_Geek;uid=root;pwd=tacobell"
210 mConn.Open

220 mCmd.ActiveConnection = mConn
230 mCmd.CommandType = 1 'Text
240 mCmd.CommandText = "SELECT * FROM Customers"

250 'Öffnen des Ergebnissatzes, der auf der Anfrage von nCmd basiert
260 rstCustomers.Open mCmd,,1,1

270%>

280<BODY>

290 <P align=center><STRONG>Here is a list of our300 Customers</STRONG>      </P>
310 <P align=center>
320 <TABLE align=center border=1 cellPadding=1 cellSpacing=1
    idth="100%">

330 <TR>
340   <TD align=middle bgColor=silver><STRONG>First
350     Name</STRONG></TD>
360   <TD align=middle bgColor=silver><STRONG>Last
370     Name</STRONG></TD>
380   <TD align=middle bgColor=silver><STRONG>State</STRONG></TD>
390 </TR>

400<%
410 ' Setzt den Counter auf 0
420 vRecordCount = 0

430 ' Wandert durch den Ergebnissatz
440 Do Until rstCustomers.EOF

```



```

450      ' Dieses erzeugt alternierende Zeilenfarben und erleichtert so das
        Lesen.
460      If vRecordCount Mod 2 > 0 Then
470          vbgcolor = "#C0C0C0"
480      Else
490          vbgcolor = "#FFFFFF"
500      End If

510      'Gibt das Ergebnis in einer Tabelle aus.
520      Response.Write "<TR>"
530      Response.Write "<TD align=middle bgcolor =" & vbgcolor &
        ">" & rstCustomers.Fields("First_Name") & "</TD>"
540      Response.Write "<TD align=middle bgcolor =" & vbgcolor &
        ">" & rstCustomers.Fields("Last_Name") & "</TD>"
550      Response.Write "<TD align=middle bgcolor =" & vbgcolor &
        ">" & rstCustomers.Fields("State") & "</TD>"
560      Response.Write "</TR>"

570      'Bewegt den Zeiger zum nächsten Datensatz
580      rstCustomers.MoveNext

590      'Erhöht den Datensatzzähler
600      vRecordCount = vRecordCount + 1
610  Loop

620  rstCustomers.Close
630  mConn.Close
640 %>
650 </TABLE></P>
660 </BODY>
670 </HTML>

```

Das, was sich in den Zeilen 10 bis 50 (META-Tags und Header) abspielt, wurde schon zuvor erläutert. Also springen wir direkt in den Abschnitt, der sich mit den ADO-Objekten befasst.

Wie immer, deklarieren Sie die Variablen und erstellen die Objekte (Zeilen 70 bis 160).

Beachten Sie, dass Sie diesmal ein `Recordset`-Objekt erstellen. Es ist beim Programmieren üblich, das `Recordset`-Objekt nach den Daten zu benennen, die dieses enthält. Es erleichtert ihren Zweck zu erkennen, vor allem wenn mehrere `Recordset`-Objekte auf einmal benutzt werden.

In den Zeilen 170 bis 210 werden die Variablen Ihres `Connection`-Objekts definiert. Da Sie das schon kennen, gehen Sie zum `Command`-Objekt über. Diesmal werden Sie eine SQL-Anweisung ausführen, die ein Abfrageergebnis mit dem `Command`-Objekt zurück-

gibt. Das geschieht in den Zeilen 220 bis 260. Vergessen Sie nicht, dass Sie jede Anweisung, die Sie im MySQL-Monitor ausführen, auch im `CommandText` angeben können. In diesem Fall führen Sie eine `SELECT`-Anweisung aus, um alle Kunden der `Customers`-Tabelle nach Nachnamen sortiert zu erhalten.

Eine weitere Möglichkeit, diese Anfrage zu realisieren, besteht darin, den Befehlstyp *Tabellenname* (2) in dem `CommandType`-Attribut des `Command`-Objekts zu verwenden. Sie könnten statt einer 1 eine 2 verwenden, um anzuzeigen, dass eine Tabelle benutzt wird. Dann brauchen Sie lediglich den Tabellennamen im `CommandText` anzugeben:

```
CommandTyp = 2
```

```
CommandText = "Customers"
```

Dieses würde das gleiche Ergebnis wie die `SELECT *`-Anweisung ergeben.

Das zeigt Ihnen, wie flexibel ADO ist. Sie können einige Dinge auf unterschiedlichste Weise erledigen. Es hängt ganz von Ihren Vorlieben ab.

Der nächste Teil des Programmcodes öffnet das `Recordset`-Objekt (Zeile 260). `Source` ist der erste Parameter der `Open`-Methode. Als Quelle haben wurde das `Command`-Objekt ausgewählt. `ActiveConnection` stellt den zweiten Parameter dar. Da Sie ein `Command`-Objekt verwenden, das bereits eine `ActiveConnection` hergestellt hat, müssen Sie hier keine angeben. Der nächste Parameter ist der vom `Recordset`-Objekt verwendete `Cursor`-Typ. Da Daten lediglich angezeigt und nicht bearbeitet werden, wird der Standard – nur vorwärts – gewählt. `LockTyp` bildet den nächsten Parameter. Da ein statisches `Recordset`-Objekt benutzt wird, wird hier nur eine *Read only*-Sperre benötigt. Dieses ermöglicht auch anderen Benutzern Zugriff auf die Tabelle.

Zeilen 430 bis 610 behandeln das Durchwandern des `Recordset`-Objekts. Dieser Code ist ziemlich einfach. Um durch ein `Recordset`-Objekt zu wandern, was nichts anderes heißt, als einen Datensatz auf einmal zu bearbeiten, benutzen Sie eine `Do ... Until`-Schleife. Diese überprüft zunächst die Bedingungsanweisung. Ergibt die Bedingung *FALSCH*, wird weiter durch das `Recordset`-Objekt gewandert. Wenn nicht, geht es ohne Schleifendurchlauf weiter. Die Schleife enthält den Programmcode, der die Daten des `Recordset`-Objekts auf der Webseite ausgibt. Dafür benutzen Sie die `Fields`-Eigenschaft des `Recordset`-Objekts. Im Wesentlichen ist die `Fields`-Eigenschaft eine Ansammlung von `Name=Wert`-Paaren. Sie können sich auf einen Wert in einer Spalte beziehen, indem Sie den Spaltennamen so wie im Programmcode angegeben benutzen. Hier werden der Vor- und Nachname sowie der Bundesstaat, in dem dieser wohnt, angezeigt.

Der letzte Teil des Programmcodes (Zeilen 620 bis 630) schließt das `Recordset`-Objekt und danach die Verbindung.

Wie Sie sehen, können Sie Microsofts Technologie in MySQL auf vielfache Weise einsetzen. Es hängt ganz von Ihnen und Ihrem Vorstellungsvermögen ab. Dies verbindet zwei sehr unterschiedliche, aber fähige Technologien miteinander. Sie können einen MySQL-Server auf einer Linux-Plattform haben und über ADOs Verbindung mit diesem aufnehmen. Das ermöglicht Ihnen, die Vorteile von beiden zu nutzen. Sie erhalten eine stabile, verlässliche und schnelle Datenbank sowie eine Applikation, die auf dem meist-verbreiteten Desktop-Betriebssystem betrieben werden kann. Es ist alles da, Sie müssen nur herausfinden, wo Sie es am besten zusammenbringen.

Zusammenfassung

Heute haben Sie etwas über ODBC – Microsofts Antwort auf eine heterogene Umgebung – erfahren. Mit ODBC ist der Zugriff auf die Daten immer der gleiche, egal von welcher Datenbank die Daten abgerufen werden. Um aber von ODBC profitieren zu können, müssen Sie zusätzlich einige andere Tools benutzen. Die Tools, die Sie in den Beispielen verwendet haben, waren ActiveX Data Objects, VBScripts und Active Server Pages.

Sie haben gelernt, dass es in ADO im Wesentlichen drei Objekte gibt: das `Connection`-Objekt, das `Command`-Objekt und das `Recordset`-Objekt. Sie haben gesehen, wie diese miteinander zusammenarbeiten. Zudem haben Sie deren Methoden und Eigenschaften kennen gelernt.

Sie haben auch etwas über die Active-Server-Page-Technologie erfahren. Sie konnten sehen, wie eine ASP-Seite Daten verarbeiten und Seiten dynamisch erstellen kann.

Sie wissen jetzt, dass VBScript – ein Ableger von Visual Basic – eine einfach zu benutzende Sprache, ist und haben gesehen, wie Sie damit ADO-Objekte manipulieren und auf diese zugreifen können.

Schließlich haben Sie gesehen, wie man alle diese Technologien zusammen einsetzen kann – wie Sie zum Beispiel VBScript in einer ASP-Seite benutzen können, um über ADO eine Datenbank zu manipulieren.

Fragen und Antworten

F Kann ich mit Hilfe dieser Technologie einen Benutzer hinzufügen?

A *Mit ADO können Sie genau das Gleiche wie auch mit dem MySQL-Monitor ausführen. Sie können Benutzer hinzufügen, Berechtigungstabellen neu laden und sogar Ihre Tabellen optimieren.*

- F** Es scheint, dass Sie eine Menge Schnittstellen-Ebenen hinzufügen. Wie wirkt sich das auf die Performance aus?
- A** *Trotz der vielen hinzugefügten Ebenen ist ADO immer noch sehr schnell. Diese Lektion hat eine MySQL-Datenbank und eine Visual-Basic-Applikation über eine Anwählverbindung verwendet. Die Benutzer haben nicht einmal bemerkt, dass Sie auf eine entfernte Datenbank zugegriffen haben. So schnell ist ADO.*
- F** Kann ADO nur in ASP-Seiten verwendet werden?
- A** *Nein. ASPs sind nur eine Möglichkeit, ADOs einzusetzen. Visual Basic, Java und Visual C++ können alle ADO benutzen. Die Installation eines MyODBC-Treibers ist die einzige Voraussetzung. Ohne diesen können Sie nicht auf MySQL-Datenbanken zugreifen.*
- F** Warum sollte ich einen DSN verwenden?
- A** *Ein DSN stellt eine schnelle und unkomplizierte Möglichkeit dar, auf Ihre Datenbank zuzugreifen. Dieser ist für Desktop-Applikationen sehr gut geeignet. Auch andere Applikationen benötigen einen DSN, um auf ODBC-Datenquellen zuzugreifen. Um zum Beispiel irgendetwas von Microsoft Access nach MySQL zu importieren bzw. zu exportieren, benötigen Sie einen DSN. Des Weiteren können Sie die Datenbank wechseln, ohne Ihre Applikation neu erstellen zu müssen.*

Übungen

1. Stellen Sie eine Verbindung zu einer MySQL-Datenbank mit Hilfe von ADO her, und optimieren Sie Ihre Tabellen.
2. Erstellen Sie eine ASP-Seite, die es dem Benutzer ermöglicht, Daten aus der Customers-Tabelle zu bearbeiten.

A tilted, light gray calendar grid is visible in the background. The top row contains the letters 'M', 'D', 'M', 'D', and 'S' in white. Below this are several empty white squares representing days of the month.

Das Perl-Daten- bank-Interface

**Woche
2**

Die Programmiersprache Perl wurde im Jahre 1987 von Larry Wall entwickelt. Seitdem ist Perl zu einer der beliebtesten Programmiersprachen in der Computerindustrie geworden. Perl ist auf fast jeder Plattform (unter anderem auch unter Windows) einsetzbar. Sie kann sehr leicht portiert werden, manchmal ohne auch nur eine Zeile des Programmcodes zu verändern. Es ist eine sehr robuste Programmiersprache mit ausgesprochen großen Fähigkeiten zur Zeichenketten-Manipulation. Das macht Perl zu der bevorzugten Wahl sowohl bei CGI-Programmen als auch bei vielen anderen professionellen Applikationen.

Aufgrund der Popularität von Perl war es nur eine Frage der Zeit, bis die Anwender begannen, Perl zur Interaktion mit Datenbanken einzusetzen. Die ersten Versionen dieser Module waren allerdings im Vergleich zu denen, die heutzutage verfügbar sind, etwas schwerfällig. Damals wurden für jede Datenbank eigene spezifische Module entwickelt. Diese Module wurden nach den jeweiligen Datenbanken benannt wie etwa Sybperl und Oriperl. Sie waren damals ausreichend, passten aber nicht in das heterogene Datenmodell, das zum Standard für die Industrie wurde. Damit sind wir beim Perl-Datenbank-Interface, gewöhnlich als DBI bezeichnet. DBI stellte Programmierern eine Standardausstattung an Funktionen und Variablen zur Verfügung, um damit auf jede Datenbank zugreifen zu können – ähnlich wie ODBC, das Zugriff auf jede Datenbank unter Windows ermöglicht. Das Interface kommunizierte mit einem Treiber, der spezifisch für jede einzelne Datenbank war. Dieser Teil der Architektur wird auch DBD oder Datenbank-Treiber genannt. Diese beiden Teile boten zusammen eine Architektur, die einfach zu erlernen und anzuwenden war. Glücklicherweise hat MySQL einen DBD. Dies ermöglicht Ihnen den Zugriff auf eine MySQL-Datenbank via Perl.

Heute werde Sie Folgende lernen:

- ▶ wie Sie die DBI/DBD Treiber für MySQL installieren
- ▶ wie Sie die MySQL DBI/DBD verwenden
- ▶ wie Sie CGI-Programme erstellen, die mit einer MySQL-Datenbank kommunizieren können

Installation der Perl DBI und der MySQL DBD

Um von der Perl DBI profitieren zu können, müssen Sie erst die nötigen Module installieren. Dieser Vorgang ist relativ einfach, kann aber für Anfänger etwas einschüchternd wirken. Der erste Schritt besteht darin, die neueste Perl DBI zu besorgen. Sie finden diese an verschiedenen Orten. Das *Comprehensive Perl Archive Network* (CPAN) stellt immer die aktuellsten Versionen des Interface und ebenso die verschiedenen Datenbank-Treiber zur Verfügung. Auch auf der MySQL-Webseite finden Sie diese Treiber und das Interface. Hier befinden sich vielleicht nicht die aktuellsten Ver-

sionen – doch sind sie zumindest lauffähig. Die Perl DBI und die MySQL DBD sind zudem auf der diesem Buch beiliegenden CD-ROM enthalten. Dies erspart Ihnen die Mühe, sie sich aus dem Internet herunterladen zu müssen.

Perl besteht aus einer Reihe von Modulen. Jedes Modul erweitert die Fähigkeiten von MySQL. Es sind eine große Menge von Modulen für Perl verfügbar. Auch gibt es Module für CGI, Tk und verschiedene mathematische Funktionen sowie Module, die andere Programme wie zum Beispiel den Apache Web Server erweitern.

Die Installation von Modulen ist mit Perl wirklich einfach. Sie können es auf zwei verschiedene Weisen machen. Die erste Möglichkeit besteht darin, das Modul aus dem Internet zu laden und es wie jedes andere Programm zu installieren, indem Sie der Anleitung in der README-Datei folgen. Die andere Möglichkeit besteht darin, Perl selbst die Arbeit machen zu lassen.

Leider ist die Vorgehensweise der letztgenannten Möglichkeit von Plattform zu Plattform verschieden, obwohl sie im Grunde ähnlich ablaufen. Im nächsten Abschnitt wird die Installation von Perl unter Windows behandelt.

Das Herunterladen und Installieren von Modulen

Sie müssen die Windows-Version von Perl schon installiert haben, um die folgenden Schritte ausführen zu können. Sie finden die Software auf der Webseite von ActiveState (<http://www.activestate.com/>). Eine sehr nützliche FAQ (Frequently Asked Questions) finden Sie unter <http://www.perl.com>. Nachdem Sie Perl installiert haben, wechseln Sie in das Perl-Verzeichnis. Hier finden Sie ein Unterverzeichnis namens `bin`. Darin enthalten sind all die verschiedenen Hilfsprogramme, die bei Perl mitgeliefert werden. Suchen Sie nach dem PPM-Programm. Hierbei handelt es sich um den Perl Package Manager. Dieses kleine Programm findet Perl-Module im Internet, lädt diese auf die lokale Festplatte und installiert sie automatisch. Einen besseren Service kann man sich nicht vorstellen.

Um die DBI zu installieren, doppelklicken Sie auf die PPM-Datei. Daraufhin öffnet sich ein DOS-Fenster. In diesem Fenster (der MS-DOS-Eingabeaufforderung) werden Sie nun weiterarbeiten. Es ist ziemlich einfach, aber so ist Perl nun mal – Programme, die eine Aufgabe gut erledigen. Um eine Liste verfügbarer Befehle zu erhalten, geben Sie `help` ein. Wenn Sie eine detailliertere Beschreibung der einzelnen Befehle erhalten möchten, geben Sie `help` gefolgt von dem Befehl ein. Möchten Sie zum Beispiel mehr über den `set`-Befehl erfahren, geben Sie Folgendes ein:

```
PPM> help set
```

Diese Anweisung gibt eine Liste von Optionen für den Befehl `set` aus, unter denen Sie auswählen können. Sie werden jetzt die `install`-Anweisung ausführen. Damit veranlassen Sie den PPM, das spezifizierte Packet (Modul) zu installieren. Beachten Sie,

dass dieses Programm zwischen Groß- und Kleinschreibung unterscheidet. Daher müssen Sie alle Befehle und Paketnamen richtig eingeben, damit eine Anweisung auch wie gewünscht ausgeführt wird. Wenn Sie die DBI aus dem Internet installieren möchten, stellen Sie sicher, dass Sie mit dem Internet verbunden sind, und geben Sie folgende Anweisung ein:

```
PPM> install DBI
```

Der PPM sucht daraufhin im Internet nach dem aktuellsten DBI-Modul und beginnt, dieses herunterzuladen. Leider stellt der PPM keinen Indikator zur Verfügung, der Ihnen den Stand des aktuellen Prozesses anzeigt. Seien Sie daher geduldig. Es mag Ihnen vielleicht so erscheinen, als wenn der PPM nicht arbeiten würde – aber seien Sie sicher, er arbeitet.

Nachdem der PPM das Paket erfolgreich heruntergeladen hat, beginnt er mit der Installation. Sie werden eine Menge Daten über den Bildschirm (genauer: das DOS-Fenster) laufen sehen, was Ihnen anzeigt, dass der PPM arbeitet. Nachdem er den Prozess abgeschlossen hat, wird angezeigt, ob die Installation erfolgreich war oder nicht. Im Normalfall wurde die Installation erfolgreich ausgeführt. Sollte dies aber einmal nicht der Fall sein, so wiederholen Sie den Vorgang oder versuchen Sie, das Modul manuell zu installieren.

Wie Sie sehen, ist der PPM ein großartiges Programm, um Module unter Windows zu Ihrem Perl hinzuzufügen. Auf allen anderen Plattformen können Sie ähnliche Hilfsprogramme nutzen, die zumeist als Option des `Perl`-Befehls verfügbar sind. Sie können den gleichen Vorgang, den Sie eben mit dem PPM vollzogen haben, auch ausführen, indem Sie Folgendes auf Kommandozeilenebene eingeben (vorausgesetzt, Sie haben Perl installiert und die Pfadangaben richtig gesetzt):

```
%> Perl -MCPAN -e shell
```

Nachdem Sie diese Anweisung eingegeben haben, beginnen Sie – ähnlich wie beim PPM – eine interaktive Sitzung. Sie werden ein paar Fragen beantworten müssen, meistens zu den Pfaden, zu bestimmten Schlüsseldateien usw. Die Standardantworten sind in den meisten Fällen richtig. Nachdem Sie alle Fragen beantworten haben, geben Sie folgende Anweisung ein:

```
%> install DBI
```

Diese Anweisung installiert die DBI im richtigen Verzeichnis.

Installation von CD-ROM



Die folgenden Anweisungen gelten für die Installation unter Linux.

Wenn Sie die Distribution von der CD-ROM verwenden, müssen Sie Perl manuell installieren. Dazu führen Sie folgende Schritte aus:

1. Zunächst müssen Sie die CD-ROM mounten. Werfen Sie einen Blick in das Handbuch Ihres Betriebssystems, um herauszufinden, wie das geht.
2. Nachdem Sie die CD-ROM erfolgreich gemountet haben, müssen Sie die Dateien in ein Verzeichnis auf Ihrem System kopieren. Geben Sie Folgendes auf Kommandozeilen-Ebene ein:

```
%> cp DBI-1.11.tar.gz /home/perl/DBI-1.11.tar.gz
```

Verwenden Sie hierbei ein Verzeichnis, auf das Sie Zugriff haben. Im Allgemeinen ist das `home`-Verzeichnis eine gute Wahl.

3. Nachdem Sie die Datei kopiert haben, wechseln Sie in das angegebene Verzeichnis:

```
%> cd /home/perl
```

4. Daraufhin entpacken Sie die Datei:

```
%> gunzip DBI-1.11.tar.gz
```

```
%> tar -xvf DBI-1.11.tar
```

5. Nachdem Sie die Dateien erfolgreich entpackt haben, können Sie diese kompilieren. Wechseln Sie hierfür in das neu erstellte DBI-1.11-Verzeichnis:

```
%> cd DBI-1.11
```

6. Als Nächstes erstellen Sie die `make`-Datei:

```
%> perl Makefile.pl
```

Eine Menge Daten werden über den Bildschirm laufen, die Sie über den Zustand des Prozesses informieren. Nach Beendigung dieses Prozesses werden Sie informiert, ob der Kompilierungsvorgang erfolgreich war oder nicht. Ist dies der Fall, können Sie mit der Installation fortfahren.

7. Schließlich fügen Sie das Modul in die richtigen Verzeichnisse ein. Hierzu geben Sie folgende Anweisung ein:

```
%> make
```

Wiederum werden eine Menge Daten über den Bildschirm laufen. Wenn dieser Vorgang abgeschlossen ist, geben Sie Folgendes ein:

```
%> make test
```

Der Bildschirm füllt sich erneut mit einer Menge Daten. Ist der Prozess abgeschlossen, geben Sie folgende Anweisung ein:

```
%> make install
```

Nachdem diese Anweisung ausgeführt ist, haben Sie die Perl DBI erfolgreich installiert.

Installation der MySQL DBD

Im nächsten Schritt installieren Sie die MySQL DBD. Den Treiber finden Sie entweder bei CPAN oder auf der MySQL- Webseite. Der Einfachheit halber finden Sie ihn auch auf der beiliegenden CD-ROM. Um die DBD zu installieren, führen Sie folgende Schritte aus:

1. Mounten Sie die CD-ROM (wenn Sie dies noch nicht gemacht haben). Folgen Sie der Anleitung Ihres Betriebssystems, um diese Aufgabe zu erledigen.
2. Kopieren Sie die MySQL-Treiber in ein Verzeichnis, auf das Sie Zugriff haben. Der Dateiname des Treibers ist `Msql-MySQL-modules-1.2209.tar.gz`.
3. Nachdem Sie die Datei kopiert haben, entpacken Sie diese, wie Sie es auch mit der DBI getan haben. Ändern Sie lediglich die Dateinamen.
4. Wenn Sie die Datei entpackt haben, geben Sie folgende Anweisung ein:

```
%>perl Makefile.pl
```

Sie sollten nun mit folgenden Auswahlmöglichkeiten konfrontiert werden:

```
MySQL only
mSQL only (either of mSQL 1 or mSQL 2)
MySQL and mSQL (either of mSQL 1 or mSQL 2)
mSQL 1 and mSQL 2
MySQL, mSQL 1 and mSQL 2
```

Die Standardwahl ist Möglichkeit 3, die gleichzeitig auch die wahrscheinlich einfachste Möglichkeit darstellt. mSQL ist ein früher Vorgänger von MySQL. Sie richten keinen Schaden an, wenn Sie diesen Treiber ebenfalls installieren.

Nachdem Sie Ihre Wahl getroffen haben, konfrontiert Sie das Installationsprogramm mit ein paar weiteren Fragen. Die erste Frage ist, ob Sie irgendwelche vorher installierten Module behalten möchten. Wenn Sie andere Module installiert haben und diese immer noch benutzen, wählen Sie *Yes*. Ist das nicht der Fall, wählen Sie *No*. Anschließend werden Sie nach dem MySQL-Installationsverzeichnis gefragt. Haben Sie MySQL im Standardverzeichnis installiert (`/usr/local/mysql/`), können Sie einfach die Eingabetaste drücken. Die nächsten Fragen beziehen sich auf eine Datenbank, damit der Treiber sich selbst testen kann. Wenn Sie noch die *test*-Datenbank haben, drücken Sie einfach die Eingabetaste. Falls nicht, müssen Sie eine Datenbank angeben. Dabei müssen Sie ebenfalls einen Benutzernamen, das Passwort und einen Hostnamen angeben.

Nachdem Sie all diese Schritte ausgeführt haben und keine Fehler aufgetaucht sind, haben Sie den Treiber erfolgreich installiert und Sie können nun damit beginnen, Perl-Scripts für den Zugriff auf Ihre Datenbank zu schreiben.

Sie haben nun die Programme installiert, die Sie benötigen, um von dieser mächtigen Technologie zu profitieren. Ein warnendes Wort noch: Viele Bücher gehen nicht auf die Installation der verschiedenen Treiber auf einem System ein, da sie befürchten, nur die beschriebene Version verwenden zu dürfen. Das ist teilweise richtig. Die meisten Installationstechniken ändern sich nicht oft. Wenn sie sich ändern, dann im Allgemeinen nur, um den Prozess zu vereinfachen. Die beste Vorgehensweise ist, das `README`-Dokument zu lesen, bevor Sie irgendetwas installieren, um sicherzustellen, dass Sie sich an die aktuellsten Richtlinien halten.

Sie können auch den PPM verwenden, um den Treiber zu installieren. Starten Sie dafür wiederum den PPM und geben Sie folgende Anweisung in das sich öffnende DOS-Fenster ein:

```
PPM> install DBD-MySQL
```

Wenn der Treiber nicht unter diesem Namen gefunden wird, können Sie eine Suche nach dem richtigen Namen beginnen. Geben Sie dazu Folgendes ein:

```
PPM > search mysql
```

Diese Anweisung gibt die aktuellsten Treibernamen aus. Von hier aus brauchen Sie einfach nur `install Treibername` einzugeben und den PPM den Rest für Sie erledigen lassen.

Ihr erstes Perl-DBI/DBD-Script

Dieser Abschnitt geht davon aus, dass Sie bereits über grundlegendes Wissen über Perl verfügen. Ist das nicht der Fall, sollten Sie einige Bücher lesen, die Sie mit den nötigen detaillierten Informationen über diese Sprache versorgen. Aber auch wenn Sie Perl nicht kennen, sollten Sie weiterlesen, um zu sehen, wie nützlich und mächtig diese Programmiersprache ist.

Wie schon in den vorhergehenden Kapiteln besprochen, haben alle Interfaces die gleiche allgemeine Funktionalität. Das Perl DBI ist da nicht anders. Eine Verbindung muss aufgebaut, Anfragen an die Datenbank übergeben, Ergebnisse zurückgegeben und die Verbindung wieder geschlossen werden – das sind allgemeine Schritte, die alle Datenbank-Interfaces teilen.

Das Perl-DBI-Objekt

Die Perl DBI besteht aus einem Objekt, dem DBI-Objekt. Mit diesem Objekt können Sie zum Beispiel mit der `data_sources()`-Methode in Erfahrung bringen, welche Treiber zur Verfügung stehen, oder mit der `connect()`-Methode ein Datenbank-Handle erstellen. Die `connect()`-Methode gibt ein Datenbank-Handle zurück, durch das Sie Anweisungen an die Datenbank übergeben können. Ein Datenbank-Handle ist eine Art Verbindung. Um zum Beispiel einen Anruf zu tätigen, wählen Sie eine Nummer, um die gewünschte Verbindung herzustellen. Jemand am anderen Ende der Leitung antwortet, und die Konversation kann beginnen. Die Verbindung, die Sie zu der anderen Person haben, gleicht einer Verbindung mittels des DBI. Wenn alle Ihre Daten richtig sind, haben Sie eine gute Verbindung. Sie können eine Konversation führen und dann auflegen. Dieselben Vorgänge gibt es bei dem DBI. Das DBI kann viele Kennungen zu verschiedenen Datenbanken herstellen, ähnlich wie ein Telefonanbieter Sie mit einer Menge Leute via Telefon verbinden kann.

Das DBI ist ein sehr mächtiges Werkzeug, weil es so viele Verbindungen erstellen kann, wie Sie benötigen. Und nicht alle Verbindungen müssen dabei mit der gleichen Datenbank bestehen. Wenn Sie zum Beispiel Daten zwischen verschiedenen Datenbanken austauschen möchten, könnten Sie einfach mit dem DBI zwei Datenbank-Handle generieren – eins, das auf die eine Datenbank zeigt, das zweite, das auf die andere Datenbank zeigt. Das ermöglicht Ihnen, sehr mächtige Programme zu schreiben, mit denen Sie Daten transferieren können. Diese Art von Applikation ist beispielsweise sehr nützlich in einem Datencenter.

Verbindungsherstellung mit dem DBI

Es werden verschiedene Informationen benötigt, um eine Verbindung via DBI herzustellen. Sie benötigen den Namen des Treibers, den Namen der Datenbank, die Sie verwenden möchten, und den Benutzernamen samt Passwort. Das alles ist für Sie nicht neu, Sie haben diese Anforderungen schon zuvor gesehen. Die Syntax der Verbindungsanweisung sieht folgendermaßen aus:

```
$dbh = DBI -> connect(DBI:Treibername: database=Datenbankname, Benutzername,
Passwort);
```

Die Variable, die am Anfang der Anweisung steht, hält das Datenbank-Handle. Die gängige Bezeichnung dafür lautet `$dbh` oder `database_handle`. Sie können sie aber letztendlich nennen, wie Sie möchten. Es handelt sich dabei um eine ganz gewöhnliche Perl-Variable. Die Information nach dem Gleichheitszeichen ist das DBI-Objekt. Unter Verwendung der Pfeilsyntax von dem DBI-Objekt rufen Sie, um zu kennzeichnen, dass Sie eine Methode dieses Objekts aufrufen, die `connect()`-Methode mit folgenden Parametern auf:

1. DBI:*Treibername*: *database*=*Datenbankname*:

Die erste Hälfte dieser Anweisung beinhaltet den Treiber, den Sie verwenden möchten. Für MySQL ist dies `mysql`. Die zweite Hälfte, abgetrennt durch einen Doppelpunkt, enthält den Namen der Datenbank, die Sie benutzen möchten. Ein richtiges Argument für eine MySQL-Datenbank wäre hier

```
DBI:mysql:database=Meet_A_Geek
```

2. Das nächste Argument ist der Benutzername. Dabei muss es sich um einen gültigen Namen handeln, der benutzt werden kann, um eine Verbindung zu einem MySQL-Server aufzubauen.
3. Das dritte Argument ist das Passwort für den Benutzer, der im zweiten Argument genannt wird. Wenn der Benutzer kein Passwort besitzt (was ein großes Sicherheitsrisiko darstellt), bleibt dieses Argument leer. Sie würden es trotzdem mittels leerer Anführungszeichen ("") übergeben.

Um eine Verbindung zur `Meet_A_Geek`-Datenbank herzustellen, würden Sie folgende Anweisung angeben:

```
$dbh = DBI -> connect(DBI:mysql: database=Meet_A_Geek, "root", "tacobell");
```

Das Datenbank-Handle, im Allgemeinen als `$dbh` bezeichnet, hat viele Methoden, die mit ihm assoziiert werden können. Es können hier lediglich die gängigsten aufgelistet werden. Wenn Sie mehr über diese Methoden erfahren möchten, geben Sie `man DBI` auf Kommandozeilenebene ein. Im Folgenden sind einige Methoden angeführt:

4. `$dbh -> do(SQL-Anweisung)`

Diese Anweisung führt die in Klammern stehende SQL-Anweisung sofort aus. Diese Methode wird verwendet, wenn Sie Anweisungen wie `INSERT` oder `UPDATE` ausführen, die keinen Ergebnissatz zurückgeben. Sie können auch DCL-Anweisungen wie etwa `CREATE` oder `DROP` benutzen.

5. `$dbh -> prepare(SQL-Anweisung)`

Diese Anweisung generiert ein Anweisungs-Handle. Ein Anweisungs-Handle ist ein Datensatz-Objekt. Es verfügt über Methoden und Eigenschaften, die verwendet werden können, um die darin enthaltenen Daten zu beschreiben und zu manipulieren. Der Ergebnissatz, den es enthält, ist das Ergebnis der in Klammern angegebenen Anfrage.

6. `$dbh -> disconnect`

Diese Anweisung zerstört das Datenbank-Handle und schließt die Verbindung zur Datenbank.

Das Datenbank-Handle stellt noch weitere Methoden zur Verfügung, wie etwa die `table_info()`- oder `ping`-Methode. Da dieses Buch jedoch von MySQL und nicht von der Perl DBI handelt, kann diese hier auch nicht näher erläutert werden. Um mehr über diese DBI zu erfahren, werfen Sie einen Blick in die `man`-Seiten, auf die in diesem Kapitel bereits verwiesen wurde.

Erstellen eines Perl-Scripts

Jetzt können Sie Ihr erstes Perl-Script schreiben. Im Folgenden sehen Sie ein einfaches Script, das Verbindung mit der *Meet_A_Geek*-Datenbank aufnimmt und drei Datensätze einfügt:

```
#!/usr/bin/perl -w

use DBI;

$database = "Meet_A_Geek";
$driver = "DBI:mysql";

$dbh = DBI->connect($driver:database=$database, "root", "tacobell")
    or die "Verbindungsaufnahme nicht möglich";

# Einfügen der Werte
$dbh->do("INSERT INTO Customers (First_Name, Last_Name)
VALUES ('Renee', 'Robertson')");
$dbh->do("INSERT INTO Customers (First_Name, Last_Name)
VALUES ('Larry', 'Isacson')");
$dbh->do("INSERT INTO Customers (First_Name, Last_Name)
VALUES ('Mark', 'Harrison')");

# Verbindung zur Datenbank schließen
$dbh->disconnect;

exit;
```

Gehen Sie dieses Script Zeile für Zeile durch. Die erste Zeile enthält Informationen darüber, wo der Perl-Interpreter gefunden werden kann. Das ermöglicht Ihnen, das Programm auszuführen, ohne explizit den Perl-Interpreter aufzurufen. Die nächste Zeile weist den Perl-Interpreter an, das DBI-Modul zu verwenden. Wenn Sie diese Zeile vergessen, wird der Rest des Programmcodes nicht funktionieren. In den nächsten zwei Zeilen finden Sie Variablen, die für den Verbindungsaufbau benötigte Werte enthalten. Diese Technik wird Ihnen häufig begegnen. Sie ermöglicht Ihnen, die Daten schnell zu ändern oder den Verbindungs-String auf übergebenen Argumenten auf Kommandozeilen-ebene aufzubauen. Die `or die`-Anweisung stellt die minimalste Fehlerabfangroutine dar, die Sie benutzen können. Es gibt noch weitere Anweisungen,

die Sie verwenden können. Sie finden diese in den `man`-Seiten. Die nächsten Zeilen fügen Datensätze in die `Customers`-Tabelle der Datenbank ein. Danach wird die Verbindung beendet und das Handle zerstört. Die letzte Zeile des Scripts zeigt dem Interpreter an, dass das Ende des Scripts erreicht ist.

Wie Sie sehen, ist der Vorgang sehr einfach. In dem vorangegangenen Beispiel sahen Sie, wie zwei der drei Funktionen des Datenbank-Handles benutzt wurden. Werfen Sie nun einen Blick auf die dritte Funktion.

Die `prepare()`-Methode des Datenbank-Handles unterscheidet sich von den anderen beiden Funktionen des Datenbank-Handle-Objekts. Diese Methode gibt ein Anweisungs-Handle zurück, das eigene Methoden und Eigenschaften besitzt. Im Folgenden finden Sie einige dieser Methoden:

1. `execute()`:

Diese Methode führt die SQL-Anweisung aus, die in der `prepare()`-Methode des Datenbank-Handles übergeben wurde.

2. `fetchrow_hashref()`:

Diese Methode gibt ein assoziatives Array mit den Werten des Ergebnissatzes in einer `Name=Wert`-Paar-Relation zurück.

3. `finish()`:

Diese Anweisung zerstört das Anweisungs-Handle und gibt Ressourcen wieder frei.

4. `rows()`:

Diese Anweisung gibt die Anzahl der Datensätze zurück, die ein Ergebnis enthalten.

In Listing 14.1 finden Sie ein Beispiel, wie Sie die `prepare()`-Methode verwenden, um ein Anweisungs-Handle zu generieren.

Listing 14.1: Die `prepare()`-Methode und das Anweisungs-Handle

```
10 #!/usr/bin/perl -w

20 use DBI;

30 $DSN = "DBI:mysql:database=Meet_A_Geek";
40 $dbh = DBI->connect($DSN, "root", "tacobell")
50 or die "Can't connect";
60 $sth = $dbh->prepare("SELECT * FROM Customers WHERE State = 'KY'");
70 $sth->execute();
80 while($ref = $sth->fetchrow_hashref()){
```

```
90     print "$ref->{'First_Name'} $ref->{'Last_Name'}";
91 }

100 $sth->finish();
110 $dbh->disconnect;
120 exit;
```

Gehen Sie das Listing Zeile für Zeile durch. Die Zeilen 10 bis 50 haben Sie schon zuvor kennen gelernt. Diese Zeilen veranlassen das Script lediglich dazu, die DBI-Module zu benutzen, den Verbindungs-String zu erstellen und anschließend mit der Datenbank Verbindung aufzunehmen. In Zeile 60 beginnt es, interessanter zu werden. Hier setzen Sie die Variable `$sth` auf das Ergebnis der `prepare()`-Methode des Datenbank-Handles. Die `prepare()`-Methode überprüft die SQL-Anweisung und stellt sicher, dass es korrekt ist. Dann wird ein Anweisungs-Handle erstellt. Die SQL-Anweisung wird so lange gehalten, bis die `execute()`-Methode aufgerufen wird. Das erfolgt in Zeile 70.

Die `execute()`-Methode veranlasst, dass die Anfrage an die Datenbank gesendet und ein Ergebnissatz zurückgegeben wird. Alle Informationen aus dem Ergebnis liegen nun im Anweisungs-Handle ebenso wie etwa die Anzahl der zurückgegebenen Datensätzen oder die Informationen über die Daten selbst.

Nachdem die `execute()`-Methode ausgeführt wurde, können Sie eine der Methoden des Anweisungs-Handles benutzen, um die Daten zu verarbeiten. Mittels einer `while`-Schleife laufen Sie durch jeden Datenzeile und geben dabei die Daten aus den in der `print`-Anweisung benannten Spalten aus. Die Daten werden auf dem gleichen Weg aus der Datenbank abgefragt, wie sie in diese gespeichert werden.

In Zeile 100 wird das Anweisungs-Handle-Objekt zerstört und die belegten Ressourcen werden freigegeben. Die nächste Zeile zerstört das Datenbank-Handle und schließt die Verbindung. Das Script wird mit der `exit`-Anweisung beendet.

In diesem Beispiel haben Sie gesehen, wie Sie eine Verbindung zur Datenbank herstellen, eine Anfrage senden und die zurückgegebenen Ergebnisse manipulieren können. Denken Sie daran, dass Sie jedes Mal, wenn Sie mit einer Datenbank Verbindung aufnehmen wollen, folgende Schritte ausführen müssen:

1. Erstellen Sie einen Verbindungs-String, und benutzen Sie ihn, um mit der Datenbank Verbindung aufzunehmen. In Perl wird das unter Verwendung des DBI-Objekts gelöst:

```
$dbh = DBI->connect($driver:database=$database, "root", "tacobell");
```

2. Nach erfolgreicher Verbindungsaufnahme verwenden Sie das Datenbank-Handle (`$dbh`), um Anfragen an die Datenbank zu senden. Für Anfragen, die keinen Ergebnissatz zurückgeben, verwenden Sie die `do()`-Methode. Für Anfragen, die einen Ergebnissatz zurückgeben, verwenden Sie die `prepare()`-Anweisung.

```
$dbh->do("DELETE FROM Customers WHERE State='KY'");
```


3. Wenn Sie einen Ergebnissatz erwarten, verwenden Sie die `prepare()`-Anweisung:

```
$sth = $dbh -> prepare(SELECT * FROM Customers");
```

4. Bevor Sie das Ergebnis der Anfrage verarbeiten können, müssen Sie die Anfrage erst an die Datenbank senden. Dazu verwenden Sie die `execute()`-Methode des Anweisungs-Handles:

```
$sth -> execute();
```

5. Nachdem Sie die Anweisung ausgeführt haben, können Sie das Ergebnis mit der `fetchrow_hashref()`-Methode verarbeiten und sich die Datensätze ausgeben lassen. Sie müssen ein Schleifenkonstrukt verwenden, um alle Daten des Ergebnissatzes zu erhalten.

```
while($ref = $sth->fetchrow_hashref()) {  
    #Programmcode, der die Daten manipuliert  
    print "ref -> {,Last_Name'}";  
}
```

6. Nachdem Sie die Ergebnisse verarbeitet haben, räumen Sie auf, so dass Sie die System-Ressourcen wieder freigeben können. Hierzu verwenden Sie die `finish()`-Methode:

```
$sth -> finish();
```

7. Sie können nun fortfahren und mit dem bestehenden Datenbank-Handle weitere Anweisungen an die Datenbank geben. Sie können sogar viele Anweisungen mit einem Handle erzeugen. Wenn Sie Ihre Datenbanksitzung beenden wollen und alle Aufgaben erledigt haben, schließen Sie die Verbindung, indem Sie die `disconnect()`-Methode aufrufen:

```
$dbh -> disconnect();
```

Das ist schon alles. Wenn Sie es Schritt für Schritt betrachten, ist es ein einfacher Ablauf. Jetzt, wo Sie diesen Teil abgeschlossen haben, lesen Sie weiter, um zu sehen, wie Sie Ihr Wissen im Alltag anwenden können.

CGI, Perl, DBI und MySQL

Perl ist wegen seiner vielfältigen Fähigkeiten zur Zeichenketten-Manipulation gemeinsam mit der Common-Gateway-Interface (CGI-) Programmierung im Internet zu Hause. CGI ist die Umgebung, in der Programme, die Daten aus dem Internet zu Ihren Server senden oder von diesem empfangen, ablaufen. Diese Programme neigen im Allgemeinen dazu, selbst Schnittstellen zu weit größeren Applikationen und Datenbanken zu sein, die Daten von Ihrer Webseite benötigen, um ihre Aufgaben zu erfüllen. Ein Beispiel für CGI-Applikationen sind Online-Shops, Kataloge und jede Art von Web, das Webseiten dynamisch aufgrund der Kriterien eines Benutzers erstellen muss.

Sie fragen sich nun vielleicht, wie das alles zusammenpasst. Perl ist wahrscheinlich für die CGI-Programmierung am besten geeignet. Es ist sehr schnell und verfügt über starke Features zur Zeichenkettenmanipulation – zwei definitiv ausschlaggebende Punkte, wenn es um das Internet und HTML geht. Die DBI ist Perls Schnittstelle zu Datenbanken. Sie ist einfach zu verwenden und sehr mächtig. Und wenn es um Datenbanken für das Internet geht, kommt keine andere Datenbank auch nur in die Nähe von MySQLs Schnelligkeit und Benutzerfreundlichkeit. Mit dieser großartigen Kombination gibt es nichts, was Sie nicht tun könnten.

Der beste Weg, Ihnen zu zeigen, wie diese Dinge zusammenarbeiten, ist, mit einem Beispiel zu beginnen und alles Schritt für Schritt zu erläutern.

Datenverarbeitung auf einer Webseite

Sie werden jetzt eine Webseite erstellen, die Daten über ein Formular abfragt und diese anschließend in die Vendors-Tabelle der *Meet_A_Geek*-Datenbank einfügt. Werfen Sie zu Beginn einen Blick auf Abbildung 14.1, um zu sehen, wie das Formular, das Sie im Folgenden verwenden, aussieht. Das Formular enthält Abfragen zu allen Feldern, die Sie für die Datenbank benötigen. Im CGI-Programm werden die Informationen, die in diese Felder eingetragen wurden, samt der Aufforderung, die nächste Seite zu laden, an den Webserver übergeben. Der HTML-`<FORM>`-Tag hat zwei Attribute, die gesetzt werden müssen. Das erste ist das `Action`-Attribut, das dem Browser übermittelt, welches Programm aufgerufen werden soll, wenn der `submit`-Button gedrückt wurde. Das zweite ist das `Method`-Attribut. Dieses kann auf `GET` oder `POST` gesetzt werden. In allen hier aufgeführten Beispielen wird `POST` verwendet. Das ist die gängigste Methode für das Senden von Informationen.

Das folgende Listing zeigt den Programmcode für die `vendor.html`-Seite.

Listing 14.2: Abfrage der Verkäuferdaten mit einem Formular

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<title>Verkäufer</title>
</head>

<body>
<form method=post action= "/cgi-bin/AddVendor.pl">
<div align="left">
  <table border="0" width="824" height="27" cellspacing="0"
cellpadding="2">
    <tr>
      <td width="824" height="27" colspan="3">
```

```
<p align="center"><b>Verkäufer</b></td>  
</tr>  
<tr>  
    <td width="824" height="27" colspan="3">  
        <p align="center"><b>Um einen neuen Verkäufer anzulegen,  
            füllen Sie dieses Formular aus, und drücken Sie  
            abschließend den submit-Button. Alle Informationen sind  
            optional.</b></td>  
    </tr>  
<tr>  
    <td width="824" height="27" colspan="3"><b>  
<font color="#FFFFFF">.</font></b></td>  
    </tr>  
<tr>  
    <td width="824" height="27" colspan="3"  
bgcolor="#9C9C4E">&nbsp;</td>  
    </tr>  
<tr>  
    <td width="824" height="27" colspan="3"  
bgcolor="#9C9C4E"><b>Verkäufer-Name: </b>  
<input type="text" name="Vendor_Name" size="91"></td>  
    </tr>  
<tr>  
    <td width="824" height="27" colspan="3" bgcolor="#9C9C4E"><b>  
Adresse:&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~  
    </b><input type="text" name="Address" size="91"></td>  
    </tr>  
<tr>  
    <td width="824" height="27" colspan="3"  
bgcolor="#9C9C4E"><b>Adresse  
      2:&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~ </b>  
<input type="text" name="Address2" size="91"></td>  
    </tr>  
<tr>  
    <td width="824" height="27" colspan="3" bgcolor="#9C9C4E">  
<b>Telefon:&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~</b>  
&nbsp;&nbsp;&nbsp;&nbsp;&~<br>  
    <input type="text" name="Phone" size="32"></td>  
    </tr>  
<tr>  
    <td width="824" height="27" colspan="3"  
bgcolor="#9C9C4E">&nbsp;</td>  
    </tr>  
<tr>  
    <td width="245" height="27" bgcolor="#9C9C4E">  
<b>Stadt:&nbsp;&nbsp;&nbsp;&nbsp;&~</b>
```

[illegible]

In dem Formular sehen Sie eine Anzahl von Eingabefeldern. Diese Felder können die Eingabe des Benutzers entgegennehmen und sie an das CGI-Programm weitergeben. Beachten Sie, dass jedes Textfeld einen eigenen Namen zugewiesen bekommen hat. Dieser Name wird in Form eines Name/Wert-Paares an den Server übergeben. Auf diese Weise können Sie über die Namen später im Perl-Script auf die Eingabewerte des Benutzers zugreifen.

vendor - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Refresh Home Search Favorites History Mail Print Edit

Address C:\My Documents\My Webs\vendor.htm

Links Welcome to AOL.CDM Get Started with Java and AS-400 V4R3 - Java on AS-400 MSN Gaming Zone Yahoo! Finance - Mark

Vendor

To add a new vendor, simply fill out this form and click submit. All information is optional.

Vendor Name:

Address:

Address 2:

Phone:

City: State: Zip:

E-mail: URL: Contact:

Submit Reset

Done My Computer

Abbildung 14.1:
Datenabfrage mit
einem Formular

Modifizieren der Datenbank

In diesem Abschnitt werden Sie ein Perl-Skript erstellen, das die Eingabedaten aus dem Formular entgegennimmt und sie in die Datenbank einfügt. Listing 14.3 fügt die Verkäuferdaten in die Datenbank ein.

Listing 14.3: Perl-Skript, um Daten in die Datenbank einzufügen

```
#!/usr/bin/perl

use DBI;

#####
# Verbindungsroutine zur Meet_A_Geek-Datenbank #
#####

$database = "Meet_A_Geek";
$driver = "mysql";

$dsn = "DBI:$driver:database=$database;$options";
$dbh = DBI->connect($dsn,"root",'tacobell')
    or die "Error connecting to database";

%postInputs = readPostInput();
```

```

$Vendor_Name = $postInputs{'Vendor_Name'};
$Address = $postInputs{'Address'};
$Address2 = $postInputs{'Address2'};
$State = $postInputs{'State'};
$City = $postInputs{'City'};
$Zip = $postInputs{'Zip'};
$Phone = $postInputs{'Phone'};
$E_Mail = $postInputs{'E_Mail'};
$URL = $postInputs{'URL'};
$Contact = $postInputs{'Contact'};

$dbh->do("INSERT INTO Vendors
        VALUES(NULL, '$Vendor_Name', '$Address',
        '$Address2', '$City','$State', '$Zip', '$Phone', '$E_Mail',
        '$URL', '$Conact')");

$dbh->disconnect;

#####
# Routine, die die Daten des Formulars entgegennimmt#
# und die Werte in ein assoziatives Array überträgt. #
#####

sub readPostInput{
    my (%searchField, $buf, $pair, @pairs);

    if ($ENV{'REQUEST_METHOD'} eq 'POST'){
        read(STDIN, $buf, $ENV{'CONTENT_LENGTH'});
        @pairs = split(/&/, $buf);

        foreach $pair(@pairs){
            ($name, $val) = split(/=/,$pair);
            $val =~ tr/+// ;
            $val =~ s/%([a-fA-f0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
            $name =~ tr/+// ;
            $name =~ s/%([a-fA-f0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
            $searchField{$name} = $val;
        }
    }

    return (%searchField);
}

print "Content-Type: text/html\n\n";
print "<html><head><title>Datensatz hinzugefügt</title></head>";
print "<body>Der Verkäufer wurde hinzugefügt.</body>";
print "</html>";

```

Die erste Zeile des Listings gibt den Pfad zum Perl-Interpreter an. Die nächste Zeile veranlasst den Interpreter, das DBI-Modul zu verwenden. Es gibt auch ein CGI-Modul, das eine Menge Funktionalität bietet. Sie finden dieses auf der CPAN-Seite.

```
$database = "Meet_A_Geek";  
$driver = "mysql";  
  
$dsn = "DBI:$driver:database=$database;$options";  
$dbh = DBI->connect($dsn, "root", 'tacobell')  
    or die "Error connecting to database";
```

Dieses Programmstück erstellt den Verbindungs-String. Beachten Sie, dass die dritte Variable, \$DSN, verwendet wird, um den Treiber und den Datenbanknamen zu speichern. Es wurde auf diese Weise gelöst, um den Porgammcode lesbarer zu machen.

```
%postInputs = readPostInput();
```

Diese Zeile ruft die `readPostInput()`-Funktion auf. Sie analysiert die Name/Wert-Paare und erstellt einen Hash, der diese Werte beinhaltet.



Ein Hash oder auch assoziatives Array ist ein Array, das Daten als Name/Wert-Paar speichert.

```
$Vendor_Name = $postInputs{'Vendor_Name'};  
$Address = $postInputs{'Address'};  
$Address2 = $postInputs{'Address2'};  
$State = $postInputs{'State'};  
$City = $postInputs{'City'};  
$Zip = $postInputs{'Zip'};  
$Phone = $postInputs{'Phone'};  
$E_Mail = $postInputs{'E_Mail'};  
$URL = $postInputs{'URL'};  
$Contact = $postInputs{'Contact'};
```

Dieses Programmstück benutzt das assoziative Array, das aus dem `readPostInput()`-Funktionsaufruf resultiert, und weist den Variablen Werte zu. Dadurch ist der Programmcode lesbarer, und verständlicher zu machen. Sie hätten diese Schritte auch überspringen und direkt das assoziative Array verwenden können. Der einzige Nachteil dabei ist, dass Ihre SQL-Anweisung unglaublich lang und sehr schwer zu lesen wäre. Diese Schritte ersparen Ihnen später jede Menge Arbeit. Beachten Sie, dass die Namen im Array die gleichen Namen sind, die Sie zuvor im Formular der Webseite für die einzelnen Felder vergeben haben. Das ist sehr wichtig, da Perl zwischen Groß- und Kleinschreibung unterscheidet. Falls Sie sich hier verschreiben würden, könnte das später Stunden des Debuggings nach sich ziehen. Seien Sie also aufmerksam und lassen Sie sich Zeit.

```
$dbh->do("INSERT INTO Vendors
        VALUES(NULL, '$Vendor_Name', '$Address',
        '$Address2', '$City', '$State', '$Zip', '$Phone', '$E-Mail',
        '$URL', '$Conact')");
```

Dieser Teil beinhaltet die SQL-Anweisung und führt sie aus. Die Werte, die zuvor auf der Webseite eingegeben wurden, befinden sich anschließend in der Datenbank. Viel zu einfach, oder?

```
$dbh->disconnect();
```

Da Sie nun die Aufgabe erfüllt haben, ist es an der Zeit, aufzuräumen. Schließen Sie die Verbindung zur Datenbank und geben Sie die verwendeten Systemressourcen wieder frei.

```
sub readPostInput{
    my (%searchField, $buf, $pair, @pairs);

    if ($ENV{'REQUEST_METHOD'} eq 'POST'){
        read(STDIN, $buf, $ENV{'CONTENT_LENGTH'});
        @pairs = split(/&/, $buf);

        foreach $pair(@pairs){
            ($name, $val) = split(/=/,$pair);
            $val =~ tr/+/ /;
            $val =~ s/%([a-fA-f0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
            $name =~ tr/+/ /;
            $name =~ s/%([a-fA-f0-9][a-fA-F0-9])/pack("C", hex($1))/eg;
            $searchField{$name} = $val;
        }
    }

    return (%searchField);
}
```

Diese Funktion extrahiert die Name/Wert-Paare und erstellt das assoziative Array, das eine sehr praktische Funktion ist. Sie können sie bei jeder Ihrer anderen CGI-Applikationen ebenfalls benutzen. Das CGI-Modul, das bei CPAN verfügbar ist, nimmt Ihnen diese Arbeit ebenfalls ab.

```
print "Content-Type: text/html\n\n";
print "<html><head><title>Datensatz hinzugefügt</title></head>";
print "<body>Der Verkäufer wurde hinzugefügt.</body>";
print "</html>";
```

Das letzte Stück des Programmcodes sendet eine Bestätigungsmeldung an den Browser zurück. Mit dieser Bestätigungsmeldung teilen Sie dem Anwender mit, dass der Verkäufer erfolgreich in die Datenbank eingetragen wurde. Sie können an dieser Stelle auch Ihrer Kreativität freien Lauf lassen. Was Sie in diesem Schritt machen, ist, eine

Webseite an den Anwender zurückzusenden. Diese Seite können Sie wie jede andere Webseite gestalten und jede Technik einsetzen, die Sie sonst auch verwenden. JavaScript, DHTML und FLASH-Animationen sind nur einige Beispiele. Ihnen sind keine Grenzen gesetzt, da Sie eine HTML-Datei in ein Perl-Script einbetten. Es gibt nur einen Punkt, auf den Sie achten sollten. Die erste Zeile ist verpflichtend. Wenn Sie diese auslassen, wird die Seite nicht angezeigt. Die zwei Newlines (`\n`) sind erforderlich für den Inhalt, der im Browser angezeigt werden soll.

Das ist alles, was Sie machen müssen, um Datenbankzugriffe in Ihrer Webseite zu realisieren.

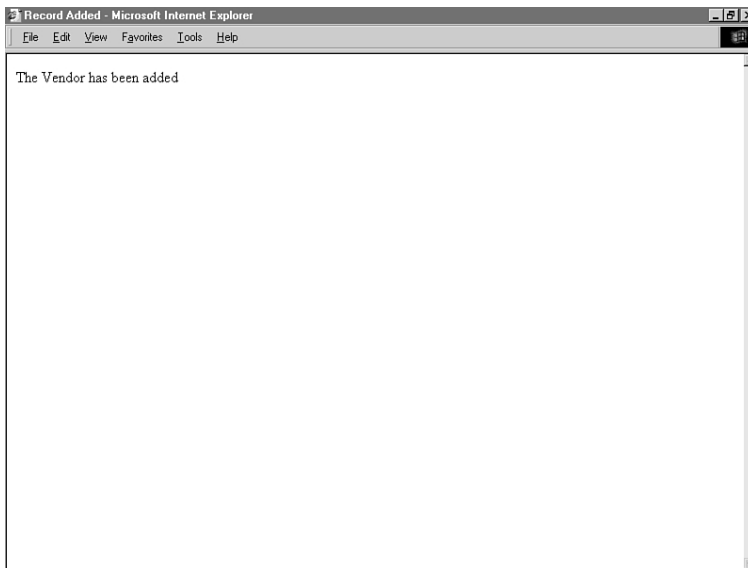


Abbildung 14.2:
Bestätigungsmel-
dung, die zum
Browser zurück-
gesendet wird

Lassen Sie uns einen zusammenfassenden Blick auf den CGI-Prozess werfen. Ein Anwender füllt ein Formular aus oder folgt einem Link, der Werte zusammen mit einer Anfrage an ein Perl-Script sendet. Das Script verarbeitet diese Werte. Es führt ebenfalls jede erforderliche Interaktion mit der Datenbank aus und sendet dann eine Rückmeldung in Form eines HTML-Dokuments zurück zum Browser.

Zusammenfassung

Heute haben Sie die Grundlagen des Perl-Datenbank-Interface in Verbindung mit MySQL kennengelernt. Sie haben nur die Spitze des Eisberges berührt, denn Sie können mit Perl und MySQL noch viel mehr machen. Sie können Perl verwenden, um

administrative Aufgaben einer MySQL-Datenbank zu automatisieren, oder auch, um Aufgaben wie die Instandhaltung oder Bearbeitung Ihrer Daten zu erledigen. Es sind Ihnen dabei keine Grenzen gesetzt.

Sie haben die drei Objekte, die Perl Ihnen zur Datenbank-Interaktion zur Verfügung stellt, kennen gelernt. Dabei handelt es sich um das DBI-Objekt, das Datenbank-Handle und das Anweisungs-Handle. Sie haben gesehen, wie diese Objekte eingesetzt werden und Ihnen eine mächtige und flexible Schnittstelle zu Ihrer Datenbank zur Verfügung stellen.

Zudem haben Sie heute gelernt, wie man einfache CGI-Perl-Programme erstellt, um Web/Datenbank-Interaktionen zu realisieren. Sie haben gesehen, wie Sie das DBI-Objekt verwenden können, um diese Aufgaben zu lösen.

Wenn es um Perl/CGI/Datenbank-Programmierung geht, gibt es eine vollkommen neue Welt zu entdecken. Heute haben Sie lediglich die Grundlagen kennen gelernt. Wenn Sie diese Programmiersprache vertiefen möchten, finden Sie bei Markt + Technik zum Beispiel die beiden Titel *Perl in 21 Tagen* von Laura Lemay und *Jetzt lerne ich CGI-Programmierung* von Rafe Colburn.

Fragen und Antworten

F Warum ist das Perl DBI so leistungstark?

A *Einer der Hauptgründe, weshalb das DBI für so leistungstark gehalten wird, ist sein einfaches, aber elegantes Design. So können Sie zum Beispiel ein Datenbank-Handle zu einer Oracle-Datenbank mit nur einer einzigen Zeile Programmcode erzeugen. In der nächsten Zeile könnten Sie ein Datenbank-Handle zu einer MySQL-Datenbank erstellen. An diesem Punkt könnten Sie zwei weitere Anweisungen erstellen, welche die Daten zwischen den beiden Datenbanken austauschen. All das können Sie mit etwa sechs Zeilen Programmcode realisieren. Einfacher geht's nicht.*

F Bin ich bei der Verwendung des Perl DBI auf einfache SQL-Anweisungen beschränkt?

A *Nein, Sie können mit dem Perl DBI jede Anweisung ausführen, die Sie auch mit dem MySQL-Monitor ausführen können, also auch beispielsweise Anweisungen wie `OPTIMIZE` und `LOAD DATA INFILE`. In der Tat verfügt das Datenbank-Handle über eine `func`-Methode, die Ihnen ermöglicht, Anweisungen wie `CREATE` Datenbank in einer Zeile zu übergeben. Die Syntax sieht folgendermaßen aus:*

```
$retVal = $dbh->func('createdb', Datenbankname, 'admin');
```

Weitere Anweisungen umfassen die *dropdb*-, *shutdown*- und *reload*-Funktionen. Diese folgen alle der gleichen Syntax. Die *shutdown*- und *reload*-Funktion benötigen das Datenbank-Argument, welches die *dropdb*- und *createdb*-Funktionen erwarten, nicht. Sie müssen exakt so wie im vorhergehenden Programmcode eingegeben werden.

Übungen

1. Erstellen Sie ein Perl-Script, das eine Anfrage vom Anwender entgegennimmt und Informationen aus der Datenbank wegen der übergebenen Werte anzeigt. Erstellen Sie zum Beispiel für die *Meet_A_Geek*-Datenbank eine Webseite, die einem Anwender ermöglicht, eine Kundenliste anzuzeigen, die auf dem Bundesstaat, in dem diese leben, basiert. Übergeben Sie diese Werte an ein Perl-Script, das diese in einer *SELECT*-Anweisung verwendet.
2. Erstellen Sie ein Perl-Script, das Benutzer zu der *mysql*-Datenbank hinzufügt (Hinweis: Verwenden Sie die *GRANT*-Anweisung).



Woche

Rückblick



Sie haben die zweite Woche mit MySQL hinter sich gebracht. Sie haben gelernt, wie Sie Daten in eine Datenbank einfügen und wie Sie MySQLs eigene Funktionen verwenden. Sie haben auch einige Programmier-Schnittstellen kennen gelernt und gesehen, was Sie mit diesen tun können.



Woche

Überblick



Nachdem Sie die 2. Woche mit MySQL erfolgreich hinter sich gebracht haben, sollten Sie in der Lage sein, mit Daten, die in einer MySQL-Datenbank gespeichert sind, zu arbeiten. Nun ist es Zeit, Sie in die administrativen Aufgaben und Problemstellungen von MySQL einzuführen.

Was Sie in dieser Woche lernen werden

Diese Woche werden Sie Ihre Kenntnisse in MySQL vervollständigen. Sie werden einiges über die Aufgaben eines MySQL-Administrators erfahren. Wie zuvor baut diese Woche auf das in den vorherigen Wochen Erlernte auf und nimmt Sie mit auf den Weg, ein wahrer MySQL DBA zu werden.

Wenn Sie diese Woche hinter sich gebracht haben, sollten Sie ein tiefgehendes Verständnis von MySQL haben. Mit Tag 15 starten Sie in die Woche und lernen, wie Sie mit dem letzten in diesem Buch vorgestellten Interface PHP auf einen MySQL-Server zugreifen. An Tag 16 werden Sie lernen, auf welche verschiedenen Weisen MySQL Zeitfunktionen unterstützt. Tag 17, »Datenbank-Sicherheit in MySQL«, bespricht alle Sicherheitsaspekte einer MySQL-Datenbank und am 18. Tag erhalten Sie einen Überblick über andere relationale Datenbank-Management-Systeme (RDBMS) und wie sich diese im Vergleich zu MySQL »schlagen«. Tag 19 behandelt administrative Aufgaben. Wie Sie auch das letzte Quäntchen Performance aus Ihrem RDBMS holen, erfahren Sie am 20. Tag, »MySQL optimieren«. Schließlich werden Sie am 21. und letzten Tag alles, was Sie zuvor gelernt haben, nutzen, um eine Webseite zu erstellen, die dynamische Seiten erzeugt und Informationen über die Besucher in einer Datenbank speichert.

A tilted, light gray calendar grid is visible in the background. The top row contains the letters 'M', 'D', 'M', 'D', 'I', and 'S' in white. Below this are several empty white squares representing days of the month.

MySQL und PHP

**Woche
3**

In diesem Kapitel werden Sie PHP, eine serverseitige Scriptsprache, die in HTML-Dateien eingebettet werden kann, kennen lernen.

Heute werden Sie

- ▶ PHPs grundlegende Sprachelemente kennen lernen
- ▶ PHP innerhalb von HTML-Dateien einbetten und auf einem Webserver ausführen
- ▶ PHP als Front-End für eine MySQL-Datenbank benutzen

Was ist PHP?

PHP steht für *PHP: Hypertext Preprocessor*. Es handelt sich um eine Sprache, die aus dem Open-Source-Umfeld kommt und sich stetig wachsender Beliebtheit erfreut. Sie steht damit in Konkurrenz zu Perl, ASP, JSP und zahlreichen anderen serverseitigen Scriptsprachen.

Serverseitige Ausführung

Ein Webserver, der nichts als HTML und Grafikdateien einsetzt, kann dem Besucher durchaus eine optisch interessante Webseite bieten. Zudem kann sie mit clientseitigen Skriptsprachen wie JavaScript, DHTML oder Java dynamischer gestaltet werden. Trotzdem wird der Seite ein wichtiger interaktiver Aspekt fehlen.

Egal wie geschickt Sie Ihre HTML-Seiten gestalten und mit clientseitigem Programmcode anfüllen, einige Aspekte der Interaktivität werden Ihnen dabei unzugänglich bleiben. Die Kommunikation mit dem Anwender wird sich stur in eine Richtung vollziehen: vom Server zum Client.

Mit einer wirklich dynamischen Webseite werden Sie diese Einschränkung überwinden. Sie können dem Anwender ermöglichen, mit einer großen Datenbank zu interagieren, Online-Shops zu durchforsten, Flüge online zu buchen und vieles mehr. Für solche Angebote benötigen Sie eine Programmiersprache, die serverseitig ausgeführt werden kann.

An dieser Stelle kommen *CGIs (Common Gateway Interfaces)* ins Spiel. Hierbei handelt es sich um serverseitig ausführbare Programme, deren Programmcode, wie der Name schon andeutet, auf dem Server ausgeführt wird und nicht auf dem Client.

Diese Vorgehensweise hat einige Vorteile. CGIs erlauben zum Beispiel die Interaktion mit großen Online-Datenbanken. Sie ermöglichen eine Zwei-Wege-Kommunikation zwischen Server und Client.

Ein weitere Vorteil ist, dass der Programmcode normalerweise unabhängig vom Client ist. Während clientseitige Programme oftmals für verschiedene Browser wie Netscape und Internet Explorer und deren verschiedenen Versionen modifiziert bzw. neu geschrieben werden müssen, laufen CGIs auf einem einzigen Rechner, dem Server. Durch diese Unabhängigkeit vom Client minimieren Sie Entwicklungskosten und stellen sicher, dass die größtmögliche Anzahl von Anwendern Ihre Seiten betrachten und nutzen kann.

PHP kommt ins Spiel

CGIs können in diversen Programmiersprachen geschrieben werden. Auf Unix-basierenden Systemen werden sie im Allgemeinen in Perl geschrieben. C, C++ und andere Sprachen können jedoch genauso verwendet werden. Neuere Sprachen sind hier zum Beispiel Microsofts ASP (Active Server Pages), Suns JSP (Java Server Pages) und Net-Objects, das eine serverseitige Lösung der vierten Generation ist. PHP wurde aus dem Open-Source-Lager hervorgebracht.

PHP-Scripts können in HTML-Dateien eingebettet werden. Wenn ein Webserver PHP unterstützt, sendet er bei der Anfrage nach einer HTML-Datei mit eingebettetem PHP die Datei nicht einfach an den Client zurück, sondern führt zuerst die PHP-Scripts innerhalb der Datei aus (PHP-Interpreter).

Solche Scripts können eine HTML-Datei »on-the-fly« erzeugen. Sie können jedoch auch Anweisungen zu Aktualisierungen von auf dem Server gespeicherten Dateien und Daten enthalten, E-Mails verschicken oder sogar Grafiken »on-the-fly« erzeugen.

Obwohl das Ergebnis für den Anwender sichtbar ist, weiß dieser nicht, dass die Seite extra für ihn erzeugt wurde! Der Anwender bekommt das PHP-Script nicht zu sehen und wird in den meisten Fällen auch nicht realisieren, dass es sich um eine dynamisch erzeugte Seite handelt.

PHP besitzt im Gegensatz zu traditionellen CGIs wie Perl den Vorteil, in HTML-Dateien eingebettet zu sein – das heißt PHP-Script muss nicht in einem CGI-BIN-Verzeichnis liegen, um ausgeführt zu werden. PHP besitzt zudem eine weit entwickelte Bibliothek von Funktionen und APIs, die dem Anwender ermöglichen, auf eine Vielzahl von Funktionalitäten zurückzugreifen.

Zum jetzigen Zeitpunkt ist PHP mittlerweile in den Versionen 3 und 4 verfügbar. Dieses einleitende Kapitel wird auf eine ausführliche Beleuchtung der Sprache verzichten und geht daher davon aus, dass die Unterschiede zwischen den beiden Versionen nicht relevant sind, um die grundlegenden Elemente der Sprache zu erlernen. Wenn immer möglich, sollten Sie jedoch die aktuelle 4er-Version verwenden, da mit dem Versionssprung einige wesentliche Verbesserungen wie zum Beispiel das Sessions-Management, verbesserte Implementierung der OOP (objektorientierte Programmierung) und eine deutliche Performance-Verbesserung hinzugekommen sind.

Die Installation von PHP

Zu Beginn eine gute Nachricht: PHP ist auf diversen Plattformen wie zum Beispiel Microsoft Windows, Unix und Linux, BeOS etc. lauffähig und auch auf den populärsten Webservern wie dem Apache und IIS (Microsofts Internet Information Server). Zudem ist es kostenlos erhältlich, egal ob Sie es im privaten, universitären oder kommerziellen Bereich verwenden möchten.

Aus Platzgründen kann in diesem Buch leider nicht auf die verschiedenen Installations- und Konfigurationsmöglichkeiten von PHP eingegangen werden. Ein paar Quellen, die Ihnen hier weiterhelfen, sollen Ihnen jedoch nicht vorenthalten werden:

- ▶ Die offizielle PHP-Webseite (<http://www.php.net/>): Hier können Sie die verschiedenen PHP-Distributionen herunterladen. Zudem finden Sie hier das offizielle PHP-Handbuch, in dem Sie auch Installations- und Konfigurationsanleitungen finden.
- ▶ Die offizielle ZEND-Webseite (<http://www.zend.com/>): Hier finden Sie neben den verschiedenen 4er-PHP-Distributionen auch Anleitungen und Zusatzsoftware wie den ZEND-Optimizer.
- ▶ Eine deutsche Anleitung zur Installation unter Linux finden Sie unter <http://www.baach.de/lamp/>
- ▶ Eine deutsche Anleitung und ein Forum zur Installation unter Windows finden Sie unter <http://www.dynamicwebpages.de/07.installation.php>.

PHP Grundlagen

Als erstes Beispiel betrachten wir eine einfache HTML-Webseite mit eingebettetem PHP-Programmcode:

```
<html>
<head>
<title>Meine erste PHP-Seite</title>
</head>
<body>
<?php
echo "Die Zeit ist jetzt: ".time();
?>
</body>
</html>
```

Speichern Sie dieses Script auf Ihrem Webserver als `showtime.php3` ab. (Beachten Sie, dass Sie in diesem Fall eine `php3`-Endung verwenden. Abhängig von der Konfiguration Ihres Webserver könnten Sie auch die Endung `.php`, `.phtml` oder sogar `.html` benutzen.)

Wenn Sie diese Seite nun mit einem Webbrowser aufrufen, sollten Sie eine ähnliche Ausgabe erhalten:

Die Zeit ist jetzt 951164380



Wenn Ihr Browser an dieser Stelle den HTML- und PHP-Code darstellt, dann haben Sie ein Konfigurationsproblem auf dem Server. Unter Umständen müssen Sie nur die Dateieindung ändern. Ändern Sie die Dateieindung in `.php`. Die möglichen Dateieindungen sind in der Konfigurationsdatei des Webserver vorgegeben (beim Apache Webserver z.B. in der `httpd.conf` bzw. `srm.conf`). Wenn dieses nicht zum Erfolg führt, sollten Sie eine der o.g. Quellen konsultieren.

Was ist hier passiert? Es handelt sich um eine ganz normale, einfache HTML-Seite, bis auf zwei wesentliche Unterschiede: Der eine ist die Dateieindung (`.php3`) und der andere die Zeile zwischen den `<?php` und `?>` Tags. Hierbei handelt es sich um spezielle Tags, die eine PHP-Code-Sektion in einer HTML-Datei kennzeichnen. Sie werden in Kürze ein wenig mehr über die richtige Einbettung von PHP in HTML-Seiten erfahren.

Die `echo()`-Funktion bewirkt in PHP die Ausgabe auf der Standardausgabe (in diesem Fall im Browser). In unserem Beispiel gibt `echo()` den Text in Anführungszeichen, gefolgt von der Systemzeit, die seit 1970 vergangen ist, in Sekunden an. Ein einfaches Script werden Sie sagen – aber dennoch, Sie haben soeben Ihre erste PHP-basierte dynamische Webseite programmiert!

Wie PHP arbeitet

Wenn eine Anfrage von einem Client-Browser an einen Webserver, der PHP unterstützt, geschickt wird, laufen auf dem Webserver folgende Vorgänge ab:

- ▶ Der Webserver erhält und liest die Anfrage des Client-Browsers.
- ▶ Er lokalisiert die angefragte Datei auf dem Server.
- ▶ Er führt im Gegensatz zu einer normalen HTML-Datei die Anweisungen des eingebetteten PHP-Programmcodes aus.
- ▶ Er sendet die erzeugte Datei an den Client-Browser zurück.

Jetzt, wo Sie einen ersten Einblick in die grundlegenden Prinzipien von PHP erhalten haben, werden Sie sich eingehender mit der Sprache selbst beschäftigen. Im weiteren Verlauf werden Sie lernen, wie Sie mit PHP Besuchern Ihrer Webseite Zugriff auf Ihre MySQL-Datenbank verschaffen.

Ein schneller Einstieg in PHP

Der folgende Teil soll Ihnen einen schnellen Einstieg in die Programmiersprache PHP ermöglichen. Wir gehen davon aus, dass Sie mit den elementarsten Prinzipien des Programmierens wie auch dem Schreiben einfacher HTML-Seiten vertraut sind. Sie werden hier nur die grundlegendsten Sprachelemente von PHP kennen lernen. Ziel ist es, Ihre Neugier zu wecken, nicht jedoch Ihnen eine umfassende Referenz zu bieten. Das ist in Hinblick auf die gebotene Kürze gar nicht möglich. Gleichwohl sollten Sie im Anschluss an diesen Abschnitt dazu in der Lage sein, gut durchdachte, dynamische Webseiten zu erstellen.

Allgemeines zur Syntax

PHP benötigt spezielle Tags, um zu kennzeichnen, wann eine HTML-Datei in den PHP-Modus wechselt. Sie können dazu eine der folgenden Möglichkeiten benutzen (eine `echo`-Anweisung ist jeweils als Beispiel angeführt):

```
<?php
echo " Hallo PHP! ";
?>
```

Dies ist eine XML-konforme Version. Möchten Sie PHP in XML-Dateien einsetzen, sollten Sie diese Variante unbedingt benutzen, die auch sonst zu empfehlen ist.

```
<?
echo " Hallo PHP! ";
?>
```

Dies ist eine Kurzform. Sie ist zwar die einfachste Version (weil kürzeste!), allerdings erfordert sie, dass in der PHP-Konfigurationsdatei der Parameter `Short tags` aktiviert ist (je nach Konfiguration ist dieser Parameter vielleicht schon aktiviert). Sie sollten diese Variante jedoch nicht benutzen, da sie nicht XML-konform ist.

```
<script language="PHP">
echo "Hallo PHP! ";
</script>
```

Dies ist eine Script-konforme Version. Diese Variante kann Ihnen bei einigen WYSIWYG-HTML-Editoren das Leben erleichtern, weil diese manchmal »ungewöhnliche« Tags nicht verarbeiten können.

```
<%
echo " Hallo PHP! ";
%>
```

Dies ist die ASP-konforme Version. Sie ist seit Version 3.0.4 verfügbar.

Eine wichtige Eigenschaft, die Sie beachten sollten, ist das Semikolon (;) am Ende jeder Anweisung. In PHP werden alle Anweisungen mit einem Semikolon abgeschlossen. PHP ist es daher egal, ob Sie mehrere Anweisungen in eine Zeile schreiben oder diese auf mehrere Zeilen verteilen, solange Sie diese jeweils mit einem Semikolon abschließen. Sie sollten sich jedoch an Programmierstandards halten und sich bemühen, »sauber« und übersichtlich zu programmieren.

Datentypen

PHP unterstützt Integer (Ganzzahlen), Doubles (Fließkommazahlen), Strings (Zeichenketten), Arrays (Datenfelder) und Objekte. PHP weist einer Variablen je nach Kontext automatisch einen Typ zu, das heißt der Entwickler muss sich praktischerweise nicht darum kümmern. Bei Bedarf kann einer Variablen selbstverständlich mittels der *Typenkonvertierung* ein bestimmter Typ zugewiesen werden. Oftmals besteht hierfür jedoch kein Bedarf.

Im Folgenden sehen Sie ein paar einfache Zuweisungen:

```
$nummer = 20; // Ganzzahliger Wert
$preis = 14.95 //Fließkommawert
$groesse = 'groß'; //String
$MeinArray[0] = "red"; // indiziertes Array, erstes Element (0)
$MeinArray["qty"] = 6; // assoziatives Array, element "qty"
```

Sie können entweder einfache (') oder doppelte Anführungszeichen (") für Zeichenketten-Werte benutzen. Sie erhalten jedoch, je nachdem welche Variante Sie wählen, verschiedene Resultate. Verwenden Sie doppelte Anführungszeichen, wird jede Variable innerhalb der Anführungszeichen ausgewertet. Verwenden Sie hingegen einfache Anführungszeichen, erfolgt keine Auswertung:

```
$var = "Der Preis ist: $preis";
echo $var;
```

ergibt:

```
Der Preis ist 14.95
```

Hingegen:

```
$var = „Der Preis ist $preis“;
echo $var;
```

ergibt:

```
Der Preis ist $preis
```

Wenn Sie Objekte verwenden wollen, benutzen Sie die `new`-Anweisung, um eine Instanz eines Objekts zu erzeugen und in Bezug zu einer Variablen zu setzen:

```
class myObject {
    function do_myObject ($p) {
        echo "Das ist myObject mit Parameter $p.";
    }
}
$var = new myObject;
$var -> do_myObject(5);
```

Dieses Listing wird folgende Ausgabe produzieren:

Das ist myObject mit Parameter 5.

Falls Sie mit der objektorientierten Programmierung (OOP) nicht vertraut sind, sollten Sie sich kundig machen, indem Sie die entsprechende Literatur konsultieren.

In PHP können Variablen auch als Boolesche Datentypen behandelt werden (WAHR oder FALSCH). Ist eine Variable undefiniert, NULL oder 0 (null), gibt sie FALSCH, enthält sie dagegen einen Wert, gibt sie WAHR zurück. Variablen auf WAHR oder FALSCH hin zu überprüfen ist sinnvoll in Bedingungsanweisungen und Schleifen, wie Sie später im Abschnitt »Kontrollstrukturen« lernen werden.

Variablen

Variablen, egal welchen Datentyps, steht immer ein \$-Zeichen voran und sie sind alle *case-sensitive* (d.h. es wird zwischen Groß- und Kleinschreibung unterschieden).

Standardmäßig kann auf Variablen von überall aus dem Script zugegriffen werden, auch wenn sich diese in eingebundenen Scripts befinden (Sie werden später mehr über das Einbinden von Scripts mittels der `INCLUDE`-Anweisung in andere Scripts erfahren). In diesem Sinne ist ihr Gültigkeitsbereich global.

Wenn Sie jedoch Variablen in benutzerdefinierten Funktionen verwenden, sind diese nur in der Funktion sichtbar und auch nur hier gültig. Von außerhalb der Funktion kann auf diese Variablen nicht zugegriffen werden.

Sie haben schon einige Beispiele gesehen, in denen mit benutzerdefinierten Variablen gearbeitet wurde. PHP stellt jedoch auch eine beachtliche Anzahl von vordefinierten Variablen zur Verfügung.

Der einfachste Weg, diese Variablen für Ihr System aufzulisten, ist, die Funktion `phpinfo()` auszuführen:

```
<?php
phpinfo();
?>
```


Diese Funktion ist wohl eine der »ausgabefreudigsten« Funktionen von PHP. In Ihrer Ausgabe finden sich eine Menge Variablen und Informationen, die in Bezug zum Webserver stehen. Sie können einige dieser Variablen auch getrennt aufrufen:

```
<?php
echo "$HTTP_USER_AGENT\n";
?>
```

wird die folgende oder eine ähnliche Ausgabe produzieren:

```
Mozilla/4.7 (Macintosh; I; PPC)
```

Solche Informationen können sinnvoll sein, wenn Sie wissen möchten, welche Art von Client-Rechner auf Ihre Webseite zugreift, woher dieser kam (`$HTTP_REFERER`), welches Script aufgerufen wurde (`$PHP_SELF`), welche Anfragemethode verwendet wurde (`$REQUEST_METHOD`) oder wie die Anfrage lautete (`$QUERY_STRING`) und vieles mehr. Wenn Sie mit HTML-Formularen arbeiten, können diese Variablen ebenfalls sehr nützlich sein.

Variablen aus HTML-Formularen

PHP macht Ihnen das Leben einfach, wenn es um die Verarbeitung von Eingabedaten aus Formularen geht. Formulare benutzen die `name=wert`-Struktur und alles, was Sie in einem Formular mit `name=` definieren, erzeugt nach dem Absenden automatisch eine Variable in PHP mit diesem Namen und dem dazugehörigen Wert.

Betrachten Sie zum Beispiel folgendes HTML-Formular:

```
<form action="formular.php" method="post">
Geben Sie Ihren Namen ein: <input type="text" name="benutzername">
Welche Größe möchten Sie bestellen?
<select name="groesse">
<option value="s">Klein
<option value="m">Mittel
<option value="l">Groß
</select>
<input type="submit" name="submit" value="Abschicken">
</form>
```

Wenn Sie dieses Script aufrufen, ausfüllen und abschicken, generiert PHP automatisch die Variablen `$benutzername` und `$groesse`. Diese enthalten die Werte, die der Benutzer eingegeben hat. Sie können auf diese Weise also dynamisch generierte Ausgaben erzeugen:

```
<?php
echo "Sie gaben $benutzername als Namen an, und bestellten die Größe $groesse";
?>
```

Sie können aus einem HTML-Formular sogar ein vollständiges Array mit Daten erhalten. Wenn Sie ein `select`-Tag mit dem Attribut `multiple` versehen und dem Namen des Tags eckige Klammern (`[]`) hinzufügen, erhalten Sie nach dem Abschicken ein Array in PHP, das die entsprechenden Daten enthält:

```
Welche Farben möchten Sie?
<select multiple name="farben[]">
<option value="R">Rot
<option value="G">Grün
<option value="B">Blau
<option value="Y">Gelb
</select>
```

Auf diesem Weg haben Sie in dem HTML-Formular durch die Art der Deklaration der Variablen `$farben[]` nach dem Abschicken ein Array `$farben[]` erzeugt und können dieses nun weiterverwenden.

Ausdrücke

In PHP erfolgen Wertzuweisungen von Variablen mit dem Zuweisungsoperator `=` (Gleichheitszeichen):

```
$groesse = 10;
```

setzt den Wert der Variablen `$groesse` auf die Ganzzahl 10.

Sie können auch mehrere Zuweisungen auf einen Schlag machen

```
$linkeFarbe = rechteFarbe = „braun“;
```

wodurch beide Variablen auf den gleichen Wert (genauer: den gleichen String) gesetzt werden.

Neben dem Zuweisen von Werten können Sie Variablen auch mittels Vergleichsoperatoren vergleichen:

- ▶ `==` auf Gleichheit überprüfen
- ▶ `!=` auf Ungleichheit überprüfen
- ▶ `<` kleiner als
- ▶ `<=` kleiner oder gleich als
- ▶ `>` größer als
- ▶ `>=` größer oder gleich als



Verwechseln Sie nicht `=` mit `==` ! Bei folgendem Beispiel

```
if (name = 'John') { echo "Hallo John!"; }
```

werden Sie immer `Hallo John` als Ausgabe bekommen. Warum? Weil `=` ein Zuweisungsoperator und kein Vergleichsoperator ist. Wenn Sie der Variablen `$name` einmal einen Wert zugewiesen haben, ergibt diese immer `WAHR` (weil die Variable nicht leer ist), und die `if`-Bedingung führt die `echo`-Anweisung aus.

```
if ( $name == 'John') { echo "Hallo John!"; }
```

vergleicht die Variable `$name` mit dem String `John` und entscheidet dann, je nachdem ob der Vergleich `WAHR` oder `FALSCH` ergibt, ob die `echo`-Anweisung ausgeführt wird.

PHP bietet mehrere abkürzende Schreibweisen (*Shortcuts*), um einer Variablen Werte zuzuweisen. Im Folgenden sehen Sie einige Beispiele:

- ▶ `$n++` und `++$n` erhöhen (inkrementieren) den Wert von `$n` um eins
- ▶ `$n--` und `--$n` erniedrigt (dekrementieren) den Wert von `$n` um eins
- ▶ `$n += 6` erhöht `$n` um den Wert 6
- ▶ `$n -= 2.5` erniedrigt `$n` um den Wert 2.5
- ▶ `$n /= 10` teilt `$n` durch 10
- ▶ `$text .= "mit bestem Gruß"` hängt den Text an das Ende des Strings `$text` an



Die Zuweisung `$n++` unterscheidet sich von der Zuweisung `++$n`. Wenn `++` (oder `--`) vor der Variablen steht, erfolgt die Erhöhung bzw. Erniedrigung des Wertes vor jeder anderen Aktion, wie z.B. der Rückgabe des Wertes. Wenn das `++` (oder `--`) hinter der Variablen steht, erfolgt die Erhöhung bzw. Erniedrigung nachdem der Wert zurückgegeben wurde. Das folgende Beispiel soll das verdeutlichen:

```
$n = 5; echo $n++;
```

gibt 5 zurück und erhöht im Anschluss daran den Wert von `$n` um eins auf 6, während

```
$n = 5; echo ++$n;
```

zuerst `$n` um den Wert eins auf 6 erhöht und diesen danach erst zurückgibt.

Operatoren

PHP stellt Ihnen die gängigen arithmetischen Operatoren zur Verfügung:

- ▶ `+` für die Addition (`$x + $y`)
- ▶ `-` für die Subtraktion (`$x - $y`)

- ▶ * für die Multiplikation (jährlich = monatlich * 12)
- ▶ / für die Division (\$x / 10)
- ▶ % für die Modulo-Operation, das heißt den Rest, der bei einer Division übrig bleibt (\$x % \$y ergibt den Rest, der bei der Division von \$x / \$y bleibt).

Kontrollstrukturen: if

PHP bietet mehrere Möglichkeiten, um den Ablauf in Ihrem Script zu steuern. Eine der wichtigsten ist die `if()`-Bedingung für das bedingte Ausführen von Anweisungen und Ausdrücken:

```
if (Ausdruck_A) {
    Anweisung A;
}
elseif (Ausdruck_B) {
    Anweisung B;
}
else {
    Anweisung C;
}
```

Eine `elseif()`-Bedingung überprüft einen *Ausdruck* und führt gegebenenfalls eine *Anweisung* aus. *else*-Anweisungen sind hingegen optional. Im einfachsten Fall benötigen Sie lediglich ein `if` und den in Klammern gesetzten, zu überprüfenden ersten *Ausdruck*:

```
if ( $preis > 100 ) echo "Das ist aber teuer!";
```

Dies ist die korrekte Syntax. Die Nachricht wird nur ausgegeben, wenn `$preis` größer als 100 ist.

Möchten Sie mehrere Anweisungen beim Eintreten einer Bedingung ausführen, müssen Sie diese in geschweifte Klammern setzen:

```
if ( $preis > 100 ) {
    echo "Das ist aber teuer!";
    echo "Vielleicht ein anderes Mal.";
}
```

Sie können auch eine `if() {} else {}`-Konstruktion benutzen, so dass die *Anweisung* in den geschweiften Klammern der *else*-Anweisung ausgeführt wird, wenn der *Ausdruck* der ersten Bedingung **FALSCH** ist (denken Sie daran, dass die geschweiften Klammern weggelassen werden können, wenn Sie nur eine *Anweisung* ausführen möchten).

Die komplexeste Form, um `if()`-Bedingungen zu nutzen, ist `if() {} elseif () {} else {}`. Sie können beliebig viele `elseif()`-Bedingungen verwenden. Wenn die erste `if()`-Bedingung nicht erfüllt ist, wird zunächst jede `elseif()`-Bedingung nacheinander ausprobiert.

Wenn eine der `elseif()`-Bedingungen WAHR ist, wird die dazugehörige *Anweisung* ausgeführt und das Script beendet. Wenn keine der Bedingungen WAHR ist, wird die *Anweisung* der `else`-Anweisung ausgeführt. Ist keine `else`-Anweisung vorhanden, wird das Script beendet, ohne dass eine *Anweisung* ausgeführt wird.

Kontrollstrukturen: while

Das einfachste Schleifenkonstrukt, das PHP zur Verfügung stellt, ist die `while`-Schleife. Die Syntax einer `while`-Schleife sieht folgendermaßen aus:

```
while (Ausdruck) {  
    Anweisungen;  
}
```

Beim Eintritt in die Schleife wird zunächst überprüft, ob der *Ausdruck* gültig (WAHR) ist. Solange dies der Fall ist, werden die *Anweisungen* innerhalb der Schleife ausgeführt. Geschweifte Klammern um die *Anweisungen* sind nur erforderlich, wenn es sich um mehr als eine *Anweisung* handelt.

Eine `while`-Schleife ähnelt einer `if`-Kontrollstruktur ohne `elseif`- bzw. `else`-Optionen. Ähnlich wie `if` werden die *Anweisungen* innerhalb der Schleife nur ausgeführt, wenn der *Ausdruck* WAHR ist. Im Unterschied zu `if` springt `while` jedoch wieder an den Schleifenanfang zurück und überprüft den *Ausdruck* so lange weiter, bis er irgendwann FALSCH ergibt. Dann wird die Schleife verlassen und der restliche Programmcode ausgeführt.

Ein einfaches Beispiel, in dem von 3 auf 0 heruntergezählt wird:

```
<?php  
$n = 4;  
echo "Countdown mit einer WHILE-Schleife:<br>\n";  
while (--$n)  
    echo "$n ...";  
?>  
Countdown mit einer WHILE-Schleife:  
3...2...1...
```

Was ist hier passiert? Jedes Mal wenn die Ausführung in die `while`-Schleife eintritt, wird die Variable `$n` um eins dekrementiert und dann auf seinen Wert (WAHR/FALSCH) überprüft. Beim ersten Durchgang ergibt `$n` 3, was in Hinsicht auf die `while`-Logik WAHR ist. Daher wird die `echo`-Anweisung ausgeführt.

Beim vierten Durchgang der Schleife hat `$n` den Wert 1, und eine weitere Dekrementierung um eins ergibt 0. Das bedeutet, `FALSCH` und die `echo`-Anweisung werden demzufolge nicht ausgeführt. Die Schleife wird beendet.

Eine weitere Form der `while`-Schleife ist `do...while`:

```
do {
    Anweisungen;
} while (Ausdruck)
```

Im Gegensatz zur ersten Form der `while`-Schleife garantiert diese Form die Ausführung der *Anweisungen* zumindest einmal. Der *Ausdruck* wird erst am Ende der Schleife überprüft. Ist dieser `WAHR`, wird der Schleifendurchlauf wiederholt. Die Schleife läuft so lange, bis der *Ausdruck* `FALSCH` ergibt, und gibt dann die Kontrolle an das aufrufende Script zurück.

Kontrollstrukturen: for

Eine komplexere Möglichkeit einer Schleife bietet `for`. Der Syntax einer `for`-Schleife lautet:

```
for (Ausdruck1 ; Ausdruck 2; Ausdruck 3;) {
    Anweisungen;
}
```

Hierbei ist die Bedeutung der *Ausdrücke* wichtig:

- ▶ *Ausdruck 1* wird unabhängig von einer Bedingung am Anfang der Schleife ausgeführt. Im Allgemeinen wird hier eine Zählvariable auf einen bestimmten Wert gesetzt.
- ▶ *Ausdruck 2* enthält die Bedingung, das heißt sie wird bei jedem Durchlauf der Schleife ausgeführt und auf `WAHR` bzw. `FALSCH` überprüft. Ist das Ergebnis der Bewertung `WAHR`, wird die *Anweisung* ausgeführt, ist es `FALSCH`, wird die Schleife verlassen.
- ▶ *Ausdruck 3* wird im Anschluss an die *Anweisungen* innerhalb der Schleife ausgeführt. Hierbei handelt es sich in der Regel wiederum um eine Zählvariable, die um einen bestimmten Wert erhöht bzw. erniedrigt wird.

Auf folgende Weise könnten Sie den Countdown aus dem vorherigen Beispiel mit einer `for`-Schleife realisieren:

```
<?php
echo "Countdown mit einer for-Schleife";
for ($n = 4; --$n; $n) {
    echo "$n...";
}
?>
```

Die `for`-Schleife produziert genau das Gleiche wie die `while`-Schleife. Die Schleife beginnt, indem `$n` auf den Wert 4 gesetzt wird. Der zweite Ausdruck dekrementiert `$n` um eins und setzt die Variable auf 3. Diese Bedingung ergibt `WAHR`, folglich wird die *Anweisung* innerhalb der Schleife ausgeführt.

(Wie schon zuvor in anderen Beispielen gezeigt, können Sie auf die geschweiften Klammern (`{}`) verzichten, wenn es sich um nur eine Anweisung handelt.)

Am Ende der Schleife wird `$n` auf `WAHR` oder `FALSCH` überprüft. Solange `$n` nicht gleich 0 ist, ergibt die Überprüfung `WAHR`. Erst beim vierten Durchgang ergibt diese Überprüfung `FALSCH`, und die Schleife wird verlassen.

Sie hätten die `for`-Schleife auch folgendermaßen eingeben können:

```
for ($n=3; $n; $n--)
```

Bei dieser Variante wird `$n` am Anfang auf den Wert 3 gesetzt und bei jedem Durchgang mit dem zweiten *Ausdruck* auf `WAHR` oder `FALSCH` überprüft. Am Ende der Schleife wird `$n` jeweils um eins dekrementiert. Beim vierten Durchlauf hat `$n` den Wert 0, und die Schleife wird verlassen, weil der zweite *Ausdruck* `FALSCH` ergibt.

Optionale Befehle für Schleifen: `break` und `continue`

Sie können den Befehl `break` überall in einer `for`- bzw. `while`-Schleife einsetzen, um die Schleife geradewegs zu verlassen.

```
do {  
    Anweisungen;  
    if (Ausdruck) {  
        Anweisungen;  
        break;  
    }  
} while (Ausdruck)  
// Ausführung geht hier weiter
```

`break` ist eine Möglichkeit, um eine Schleife auf »unelegante« Art und Weise zu verlassen. Es ermöglicht Ihnen, die Schleife an jedem beliebigen Punkt zu verlassen und hinter das Ende der Schleife zu springen.

Die `continue`-Anweisung ist der `break`-Anweisung ähnlich, jedoch im Vergleich weniger drastisch; `continue` beendet die Ausführung der Anweisungen innerhalb der Schleife, jedoch nicht die Schleife selbst. Die Ausführung springt an das Ende der Schleife, jedoch nicht notwendigerweise aus ihr heraus. Es wird versucht, die Schleife wie gewohnt auszuführen, sofern die Bedingungsanweisung erfüllt ist.

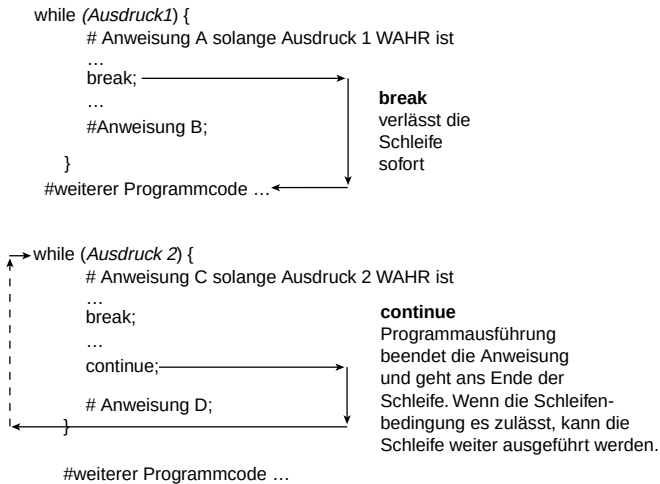


Abbildung 15.1:
Eine Schleife mit *break*
oder *continue* verlassen

Kontrollstrukturen: `include()` und `require()`

PHP ermöglicht es, mit den `include()`- und `require()`-Funktionen ein weiteres Script oder Programmschnipsel in ein anderes zu integrieren. Folgendes Beispiel zeigt die Syntax der Funktionen:

```
include("datei2.php3");
```

bzw.

```
require("datei2.php3");
```

Im Wesentlichen bezwecken beide Funktionen das Gleiche, ihre Vorgehensweise ist jedoch verschieden. `require` ersetzt zwar vollständig die `require`-Anweisung im Code durch die in Klammern stehende Datei, kann aber nicht in Schleifen benutzt werden. Mit anderen Worten, wenn Sie eine Anweisung mehr als einmal ausführen und dabei jeweils eine andere Datei laden möchten, wird das nicht funktionieren. Hier benötigen Sie die `include`-Anweisung. `include` ist flexibler als `require` und wird verwendet, wenn Sie komplexere Einbindungsanweisungen benötigen.

`include` und `require` können beim Erstellen einer Webseite sehr hilfreich sein, auch wenn Sie sonst kein PHP benutzen. Möchten Sie zum Beispiel einen allgemeinen Kopf-, Navigations- und Fußteil für Ihre Webseite verwenden, die auf allen (Unter-)Seiten gleich sind, könnten Ihre Webseiten wie folgt aussehen:

```

<html>
<head>
<title>Meine Webseite - Seite Eins</title>
<?php

```



```
include("etwas_javascript.html");  
include("einige_bodytags.html");  
include("mein_kopfteil.html");  
include("navigationsleiste.html");  
?>
```

```
<p>Hier steht der Inhalt</p>
```

```
<?php  
include("mein_dokumentenfuss.html");  
?>
```

Das ist Ihre ganze Seite! Für jede Ihrer Seiten müssen Sie nur noch den individuellen, zentralen HTML-Teil erstellen. Alle eingebetteten Dateien müssen nur einmal erstellt werden und können auf der gesamten Webseite benutzt werden.

Funktionen

PHP ermöglicht Ihnen, benutzerdefinierte Funktionen zu erstellen. Dabei handelt es sich um Programmblöcke, die, einmal erstellt, von überall her im Script aufgerufen und verwendet werden können – sogar von einem externen PHP-Script.

Neben den benutzerdefinierten Funktionen bietet PHP eine große Anzahl an »Built-in«-Funktionen (also in PHP vorhandene Funktionen). Diese stehen für eine große Anzahl von Applikationen zur Verfügung. Eine knappe Auflistung (an späterer Stelle in diesem Kapitel) erklärt die Funktionen, die für die MySQL-API zur Verfügung stehen.

Wenn Sie eine benutzerdefinierte Funktion erstellen, spezifizieren Sie darin, was sie tun und was sie (wenn überhaupt) zurückgeben soll. Sie können der Funktion optional Parameter übergeben und angeben, wie die Funktion diese verwenden soll.

In PHP werden Funktionen auf folgende Weise definiert:

```
function funktionsname (parameter) {  
    // Funktionsanweisungen;  
    return variablen; //optional  
}
```

Jetzt können Sie die Funktion folgendermaßen aufrufen:

```
funktionsname (Parameter);
```

Möchten Sie beispielsweise einen gewissen Prozentsatz zu einem Preis hinzuaddieren (z.B. die Mehrwertsteuer), so erstellen Sie eine Funktion, der zwei Parameter übergeben werden: der Preis und der Prozentsatz.

```
function plus_Prozent ($preis, $prozent) {
    $neuerPreis = $preis * (1 + ($prozent / 100));
    return $neuerPreis;
}
```

So rufen Sie die Funktion auf:

```
echo "Der neue Preis ist ".plus_prozent(150, 25);
```

Das Script würde Folgendes ausgeben:

```
Der neue Preis ist 187.5
```

Arrays

Das Kennntnis von Arrays und ihrer Funktionsweise in PHP ist äußerst wichtig, um zu verstehen, wie PHP mit einer MySQL-Datenbank kommuniziert.

In PHP enthalten numerisch indizierte Arrays Elemente, die mit dem »nullten« Element beginnen (`$meinArray[0]`). PHP unterstützt neben diesen auch assoziative Arrays, die Sie etwas später kennen lernen werden. Die Anzahl der Elemente eines Arrays erhalten Sie z.B. mit der Funktion `count()` (`count($meinArray)`).

Numerisch indizierten Arrays können auf verschiedene Arten Daten zugewiesen werden:

```
$meinArray[] = ,rot';
$meinArray[] = ,grün';
$meinArray[] = ,blau';
```

erzielt das gleiche Ergebnis wie

```
$meinArray[0] = ,rot';
$meinArray[1] = ,grün';
$meinArray[2] = ,blau';
```

Den Inhalt eines Arrays zeigen Sie am bequemsten mit einer einfachen `for`-Schleife an:

```
for ($i=0; $i < count($meinArray); $i++) {
    echo "Element $i ist ".$meinArray[$i]."<br>\n";
}
```

Diese Schleife produziert folgende Ausgabe:

```
Element 0 ist rot
Element 1 ist grün
Element 2 ist blau
```

Sie können auch assoziative Arrays verwenden:

```
$meinArray["R"] = ,rot';  
$meinArray["G"] = ,grün';  
$meinArray["B"] = ,blau';
```

Ein solches Array können Sie dann zum Beispiel folgendermaßen durchlaufen:

```
while (list($bez, $farbe) = each($meinArray)) {  
    echo $Element $bez ist $farbe<br>\n";  
}
```

Im obigen Beispiel sind einige neue Funktionen eingeführt worden. Einmal wird dort die Funktion `list()` vorgestellt, ein Sprachkonstrukt, das einer Gruppe von Variablen Werte in einer Operation zuweist (in diesem Fall besteht das Array aus einem Paar von Schlüsseln und Werten). Zum anderen taucht die Funktion `each()` auf, die zum Durchlaufen von Arrays dient.



In PHP haben Arrays einen internen Zeiger. Sie können diesen Zeiger benutzen, um zu verfolgen, wo Sie sich in einem Array befinden. Das ist praktisch, weil Sie dadurch nicht gezwungen sind, in einfachen Fällen eine separate Hilfsvariable zu benutzen. Es existieren entsprechende Funktionen – `each()`, `next()`, `prev()` und `array_walk()` – die es Ihnen ermöglichen, Arrays Element für Element zu durchlaufen. Hierbei wird der Zeiger jedes Mal auf »unsichtbare« Weise weiterbewegt.

`each()` benutzt den Namen eines Arrays als Argument, gibt den Schlüssel und Wert des aktuellen Elements (in Hinsicht auf den internen Zeiger) zurück und bewegt den Zeiger um eins weiter.

`list($schlüssel, $wert)` oder eine ähnlich Anweisung, die Sie zuvor benutzt haben, sind ein bequemer Weg, um ein Schlüssel=Wert-Paar zu erhalten.

In dieser Einleitung haben Sie nur eindimensionale Arrays kennen gelernt. Trotzdem sollten Sie wissen, dass PHP auch mehrdimensionale Arrays unterstützt. PHP stellt Ihnen auch eine große Anzahl an Funktionen für die Arbeit mit Arrays wie zum Beispiel der Sortierung zur Verfügung. Es lohnt sich, andere PHP-Ressourcen für weitere Informationen zu erforschen.

PHP trifft MySQL

PHP verfügt über einige sehr nützliche Funktionen, um mit einer MySQL-Datenbank zu kommunizieren. Betrachten Sie, was passiert, wenn ein Client-Rechner eine Anfrage, in der die Interaktion mit einer MySQL-Datenbank erforderlich ist, an einen Webserver richtet, der PHP unterstützt:

- ▶ Der Server empfängt und liest die Anfrage vom Client-Rechner.
- ▶ Der Server lokalisiert die angeforderte Seite auf dem Webserver.
- ▶ Der Server führt alle Anweisungen des eingebetteten PHP-Programmcodes aus.
- ▶ PHP richtet die Anfrage via MySQL-API an den MySQL-Datenbank-Server und stellt die Ergebnisse zusammen.
- ▶ Der Webserver schickt die Ergebnisseite zurück zum Client-Rechner.

PHP stellt eine bemerkenswerte Anzahl an Funktionen zur Verfügung, um mit dem MySQL-Datenbank-Server zu kommunizieren. Sie können, wie Sie gleich sehen werden, mit einem Bruchteil dieser Funktionen bereits mächtige Applikationen erstellen. Die folgende Liste aller Funktionen soll Ihnen hierbei als Referenz dienen:

Funktionsname	Verwendung
<code>mysql_affected_rows()</code>	Liefert die Anzahl der betroffenen Datensätze aus einer vorhergehenden MySQL-Anweisung (<i>INSERT/ UPDATE/ DELETE</i>) zurück.
<code>mysql_change_user()</code>	Ändert die Identität des Benutzers einer aktiven Verbindung
<code>mysql_close()</code>	Schliesst eine MySQL-Verbindung
<code>mysql_connect()</code>	Öffnet eine Verbindung zum MySQL-Server
<code>mysql_create_db()</code>	Erstellt eine MySQL-Datenbank
<code>mysql_data_seek()</code>	Bewegt den internen Datensatzzeiger
<code>mysql_db_query()</code>	Schickt eine MySQL-Anfrage an eine genannte MySQL-Datenbank
<code>mysql_drop_db()</code>	Löscht eine MySQL-Datenbank
<code>mysql_errno()</code>	Liefert die MySQL-Fehlernummer der zuletzt ausgeführten Operation
<code>mysql_error()</code>	Liefert den MySQL-Fehlertext der zuletzt ausgeführten Operation
<code>mysql_fetch_array()</code>	Liefert ein Ergebnis als assoziatives Array
<code>mysql_fetch_fields()</code>	Liefert Feldinformationen aus einem Ergebnis und gibt diese als Objekt zurück
<code>mysql_fetch_length()</code>	Liefert die Länge aller Felder eines Ergebnisses
<code>mysql_fetch_object()</code>	Liefert ein Ergebnis als Objekt zurück
<code>mysql_fetch_rows()</code>	Liefert ein Ergebnis als numerisch indiziertes Array
<code>mysql_field_flags()</code>	Liefert Flags, die mit einem gegebenen Feld zusammenhängen
<code>mysql_field_len()</code>	Liefert die maximale Länge eines Feldes aus einem Anfrageergebnis
<code>mysql_field_name()</code>	Liefert den Namen eines Feldes aus einem Anfrageergebnis
<code>mysql_field_seek()</code>	Setzt den Feldzeiger auf ein durch <i>field_offset</i> spezifiziertes Feld

Tabelle 15.1: PHPs MySQL-Funktionen

Funktionsname	Verwendung
<code>mysql_field_table()</code>	Liefert den Namen einer Tabelle, die das spezifizierte Feld enthält
<code>mysql_field_type()</code>	Liefert den Typ eines Feldes aus einem Anfrageergebnis
<code>mysql_free_result()</code>	Gibt den von einem Anfrageergebnis belegten Speicher wieder frei
<code>mysql_insert_id()</code>	Liefert die ID der zuletzt ausgeführten <i>INSERT</i> -Anweisung
<code>mysql_list_dbs()</code>	Liefert eine Liste verfügbarer Datenbanken auf dem Server
<code>mysql_list_fields()</code>	Liefert alle Felder einer Tabelle
<code>mysql_list_tables()</code>	Liefert eine Liste aller Tabellen einer gegebenen Datenbank
<code>mysql_num_fields()</code>	Liefert die Anzahl der Felder in einem Ergebnis
<code>mysql_num_rows()</code>	Liefert die Anzahl der Datensätze im Ergebnis zurück
<code>mysql_pconnect()</code>	Öffnet eine persistente Verbindung zum MySQL-Server
<code>mysql_query()</code>	Schickt eine MySQL-Anfrage an einen schon selektierten MySQL-Server
<code>mysql_result()</code>	Liefert ein Ergebnis von einer Anfrage
<code>mysql_select_db()</code>	Wählt eine MySQL-Datenbank aus
<code>mysql_tablename()</code>	Liefert den Namen einer Tabelle

Tabelle 15.1: PHPs MySQL-Funktionen (Forts.)

Verbindung zu einem MySQL-Server herstellen

Um eine Verbindung zu einem MySQL-Server herzustellen, benutzen Sie die Funktion `mysql_connect()`:

```
int mysql_connect ([string hostname[:port][:path/to/socket]] [, string
Benutzername] [,Passwort])
```

Diese Funktion gibt nach erfolgreicher Verbindungsaufnahme einen positiven Integerwert (Verbindungskennung) bzw. `FALSCH` im Fehlerfall zurück. Mit der Funktion `mysql_pconnect()` können Sie eine persistente Verbindung zum MySQL-Server aufbauen.

```
$server = "localhost";
$benutzer = "root";
$passwort = "pass";
$link_id = mysql_connect ($server, $benutzer, $passwort);
echo "Verbindungskennung ist ".$link_id;
```

Sind alle Parameter richtig gesetzt, sollten Sie folgende Ausgabe erhalten:

```
Verbindungskennung ist 1
```

Glückwunsch! Die zurückgegebene Verbindungskennung signalisiert Ihnen, dass die Verbindungsaufnahme mit dem MySQL-Server unter Benutzung der richtigen Angaben von Server (\$server), Benutzer (\$benutzer) und Passwort (\$password) erfolgreich war.

Das Erstellen einer Datenbank

Mit der Funktion `mysql_create_db()` können Sie eine Datenbank erstellen:

```
Int mysql_create_db (string Datenbankname [, int Verbindungskennung]);
```

Die Angabe der Verbindungskennung ist optional. Wird diese nicht angegeben, so wird die aktuell aktive Verbindung benutzt. Diese Funktion gibt einen Integerwert zurück – im Erfolgsfall positiv (WAHR), andernfalls FALSCH. Sie könnten auf folgende Weise eine Datenbank erzeugen:

```
$create_id = mysql_create_db("MyDB");  
echo "Create-ID ist ".$create_id."<br>\n";
```

Wenn Sie dieses Script ausführen, sollten Sie folgende Ausgabe erhalten:

```
Create-ID ist 1
```

Sie haben jetzt erfolgreich eine Datenbank mit dem Namen `MyDB` auf dem Server erstellt.

Grundlagen von PHP-Datenbankabfragen

Möchten Sie eine Tabelle in der Datenbank erstellen, können Sie von den Funktionen `mysql_db_query()` und `mysql_query()` Gebrauch machen. Mit diesen Funktionen können Sie jede beliebige Anweisung an eine Datenbank schicken. Die Syntax lautet:

```
int mysql_db_query (string Datenbank, string Anweisung [, int Verbindungskennung]);  
int mysql_query (string Anweisungen [, int Verbindungskennung]);
```

Wenn Sie diese Funktionen verwenden, liefern diese jeweils einen Integerwert als Ergebniskennung zurück, der auf das Abfrageergebnis zeigt. Tritt ein Fehler auf, geben die Funktionen 0 oder FALSCH zurück.

Die Verbindungskennung ist optional; wird sie nicht angegeben, wird die aktuell aktive Verbindung benutzt.

`mysql_db_query()` ermöglicht Ihnen, eine Datenbank zusammen mit dem Aufruf anzugeben, während `mysql_query()` die aktuell ausgewählte Datenbank verwendet. Eine Datenbank können Sie folgendermaßen auswählen (selektieren):

```
int mysql_select_db (string Datenbankname [, int Verbindungskennung]);
```

Betrachten Sie das folgende Beispiel, in dem eine Datenbank ausgewählt und anschließend eine Tabelle erstellt wird:

```
$select = mysql_select_db("MyDB");
$sql = "CREATE TABLE fotos (
    num INTEGER NOT NULL PRIMARY KEY,
    aufnahme_datum DATE,
    beschreibung VARCHAR (200)";
$result = mysql_query ($sql);
echo "Das Ergebnis der Erstellung der Tabelle ist $result<br>/n";
```

Wenn alles planmäßig gelaufen ist, sollten Sie folgende Ausgabe erhalten haben:

Das Ergebnis der Erstellung der Tabelle ist 1

Fehlerbehandlung

Wie erfahren Sie aber von einem eventuell aufgetretenen Fehler in der Ausführung? Um eine Fehlermeldung zu erhalten und diesen auf angemessene Art auswerten zu können, ändern Sie die letzte Zeile des obigen Scripts auf folgende Weise ab:

```
if ($result) {
    echo "Das Ergebnis der Erstellung der Tabelle ist $result<br>/n";
}
else {
    echo "Fehler: ".$mysql_errno()."; Fehlerbeschreibung: ".$mysql_error();
}
```

Wenn Sie dieses Script nun ausführen (und die Tabelle `fotos` schon existiert), erhalten Sie, bedingt durch die Verwendung der Funktionen `mysql_errno()` und `mysql_error()` folgende Ausgabe:

Fehler: 1050; Fehlerbeschreibung: Table 'fotos' already exists

Um die Fehlerbehandlung etwas cleverer zu handhaben, können Sie am Anfang des Scripts eine benutzerdefinierte Fehlerabfangfunktion erstellen:

```
function error_report() {
    echo "Fehler: ".$mysql_errno()."; Fehlerbeschreibung: ".$mysql_error()."<br>/n";
}
```

Wann immer Sie nun eine Fehlerbehandlungsroutine brauchen, rufen Sie einfach die Funktion auf:

```
error_report();
```

Anweisungen zum Einfügen von Daten

Im folgenden Abschnitt erfahren Sie, wie Sie Daten in eine Tabelle einfügen können. Sie können die gleiche `mysql_query()`-Funktion benutzen wie im vorherigen Beispiel:

```
$sql = "INSERT INTO fotos VALUES
      (10000, '99-07-15', 'Am Strand von Tabago')";
$result = mysql_query($sql);
if ($result) {
    echo "Daten hinzugefügt<br>\n";
}
else {
    error_report();
}
```

Bei INSERT-, UPDATE- oder DELETE-Anweisungen ist das Ergebnis der `mysql_query()`-Funktion immer ein Integerwert: positiv im Erfolgsfall, andernfalls FALSCH.

Ausführen von SELECT-Anweisungen und Weiterverarbeitung des Ergebnisses

Gehen wir einmal davon aus, dass Sie inzwischen mehr (womöglich eigene) Daten in die Tabelle `fotos` eingefügt haben (Sie können das nun selber!); dann sollten Sie jetzt einen Schritt weiter gehen und lernen, wie Sie diese Daten abfragen und weiterverarbeiten können.

Stellen Sie sich vor, Sie möchten aus der Tabelle alle Datensätze ausgegeben bekommen, in deren Beschreibung das Wort `Strand` vorkommt. Folgendes kleines Script führt Sie zum Erfolg:

```
$sql = "SELECT num, aufnahme_datum, beschreibung
      FROM fotos WHERE beschreibung LIKE ,%Strand%'";
$result = mysql_query($sql);
if ($result) {
    while ($row = mysql_fetch_array($result)) {
        echo "Datum: ".$row[aufnahme_datum]. " ".$row[beschreibung]. "<br>\n";
    }
}
else {
    error_report();
}
```

Wenn alles glatt gegangen ist, erhalten Sie die folgende (oder eine ähnliche) Ausgabe:

```
Datum: 1999-07-15 Am Strand von Tabago
Datum: 2000-02-01 Der Strand in Mauritius
```


Nun kommt das Ganze endlich in Bewegung! Überlegen Sie, was hier genau passiert ist:

Zunächst wird die Variable `$sql`, die die SQL-Anweisung enthält, gesetzt. Danach wird die Funktion `mysql_query()` mit dem Parameter `$sql` aufgerufen, wodurch die Anweisung ausgeführt und die Ergebniskennung (Zeiger auf das Ergebnis) in der Variablen `$result` gehalten wird. Bei `$result` handelt es sich also um einen *Zeiger* auf das Ergebnis.

Wenn der MySQL-Server einen Fehler zurückgibt, ist `$result` `FALSCH`, und die Fehlerfunktion (`error_report()`) wird aufgerufen, um diesen auszuwerten. (Bedenken Sie: Wenn die Anweisung kein Ergebnis zurückliefert, weil keine passenden Datensätze gefunden wurden, wird auch kein Fehler ausgegeben.)

Vorausgesetzt es ist kein Fehler aufgetreten, wird in die `if`-Anweisung verzweigt (`$result` ist in diesem Fall `WAHR`) und die Funktion `mysql_fetch_array()` aufgerufen.

Die Syntax von `mysql_fetch_array()` lautet:

```
array mysql_fetch_array (int Ergebniskennung [, int Ergebnistyp]);
```

Diese Funktion liefert ein Array zurück, das die Daten der angefragten Felder enthält, bzw. `FALSCH`, wenn keine Daten zurückgeliefert wurden. Auf die im Array gespeicherten Daten kann auf zwei verschiedene Arten zugegriffen werden: zum einen mittels eines numerischen Indexes (`$row[2]`) und zum anderen über den Feldnamen als assoziativen Index (`$row["beschreibung"]`).

Der optionale Parameter *Ergebnistyp* kann die Werte `MYSQL_ASSOC`, `MYSQL_NUM` oder `MYSQL_BOTH` annehmen (Sie werden diese jedoch kaum in einfachen Scripts wie den hier vorgestellten verwenden wollen).

Vergessen Sie nicht, dass in PHP jedes Array einen internen Zeiger besitzt. Beim ersten Aufruf der Funktion `mysql_fetch_array()` steht dieser auf der *nullten* Position (Zeile 0). Jedes Mal, wenn die Funktion erneut aufgerufen wird, bewegt sich dieser Zeiger praktischerweise um eine Zeile vorwärts.

Auf diese Art ist der Zeiger in der `while`-Schleife eingebettet, ohne dass Sie explizit darauf verweisen müssen. Bei jedem Aufruf der Funktion `mysql_fetch_array()` und dem Gleichsetzen des Arrays `$row` mit dem Ergebnis der Funktion bewegt sich der Zeiger um eine Zeile weiter durch das Array, das aus dem Abfrageergebnis resultiert. Dies geschieht so lange, bis das Ende des Arrays erreicht ist, an dem der Zeiger nicht mehr weiter vorwärts bewegt werden kann und `FALSCH` zurückgegeben wird. `$row` wird `FALSCH`, die `while`-Schleife wird verlassen und die Ausgabe ist fertig.

Diese grundlegende Funktionalität ist zwar bescheiden, aber sie genügt völlig, um die meisten Anfragen an eine Datenbank zu richten. Diese wenigen hier vorgestellten PHP-Funktionen sind ausreichend, um den Besuchern Ihrer datenbankgestützten Webseite ein wirklich dynamisches Erlebnis zu bieten.

Zusammenfassung

Heute haben Sie einen kleinen Ausflug in die Welt von PHP unternommen. Sie haben gelernt, wie PHP-Scripts in HTML-Seiten eingebettet werden können, und gesehen, wie man einfache PHP-Programme schreibt.

Sie haben erfahren, wie man die Möglichkeiten der PHP-Kontrollstrukturen wie zum Beispiel `includes`, `ifs`, `while` und `for`-Schleifen nutzt. Auch wenn Sie PHP nur in rudimentären Grundzügen kennen gelernt haben, sind Sie nun in der Lage, mit diesen grundlegenden Kenntnissen durchaus mächtige Scripts zu schreiben.

Zudem wurde eingegangen auf PHPs MySQL-API und deren zahlreiche Funktionen, die der Kommunikation mit einem MySQL-Server dienen. Sie konnten sehen, wie man mit einem MySQL-Server Verbindung aufnimmt und Datenbanken und Tabellen erstellt, Daten einfügt sowie Fehler abfängt.

Schließlich haben Sie gelernt, wie man via PHP die SQL-Anweisung `SELECT` verwendet. Ergebnisse werden als Arrays zurückgeliefert. Zu guter Letzt haben Sie erfahren, wie diese Daten weiterverarbeitet werden, damit sie dem Benutzer auf eine sinnvolle Art und Weise präsentiert werden.

Fragen & Antworten

F Wo kann ich mehr über PHP erfahren?

A Folgende Webseiten seien Ihnen ans Herz gelegt:

<http://www.php-builder.com/> (englischsprachige PHP-Ressource, in der Sie viele Tutorials finden.)

<http://www.php-center.de/> (deutschsprachige Portalseite zum Thema PHP)

<http://www.dynamicwebpages.de/> (deutschsprachiger PHP-Knotenpunkt; hier finden Sie u. a. viele Tutorials, Links, Anleitungen und das deutschsprachige PHP-Handbuch)

<http://www.php-homepage.de/> (deutschsprachige PHP-Ressource mit Links, Foren und vielem mehr)

<http://www.koehntopp.de/php/> (deutschsprachige PHP-FAQ)

Sie werden zudem einen Webserver benötigen, auf dem Sie PHP installieren können. Eine Übersicht zu Installationsanleitungen für diverse Plattformen wie Windows oder Linux finden Sie unter <http://www.dynamicwebpages.de/07.installation.php>.

- F** Warum sollte ich PHP und nicht eine andere Sprache verwenden?
- A** *Es gibt diverse serverseitige Sprachen und Werkzeuge auf dem Markt. PHP hat jedoch den Vorteil, dass es kostenlos und schnell ist und zudem auf den meisten Plattformen installiert werden kann.*
- PHP ist außerdem leicht zu erlernen, und Sie müssen lediglich etwas PHP-Code in eine Webseite einfügen, um ein PHP-Script zu erstellen.*

Übungen

1. Schreiben Sie ein PHP-Script, das eine `SELECT`-Anweisung auf eine Tabelle `Produkte` ausführt. Die Tabelle besteht aus den Feldern `Name` und `Preis`. Ihre Abfrage sollte alle Produkte ausgeben, deren Preis niedriger als 50 DM ist. Sie sollten an eine Fehlerabfangroutine denken, die, falls es nötig sein sollte, die benutzerdefinierte Funktion `error_report()` aufruft.
2. Schreiben Sie eine PHP-Funktion, um Verbindung zu einem MySQL-Server auf `localhost` (Webserver) aufzunehmen. Die Funktion soll die Verbindungskennung an das aufrufende Programm zurückgeben.



MySQL und die Zeit

**Woche
3**

Heute werden Sie sich näher mit Zeit- und Datumsfunktionen von MySQL beschäftigen. Sie werden verschiedene Formate und Funktionen zur Anzeige und Berechnung von Zeit und Datum, die MySQL Ihnen bietet, kennen lernen.

Die Handhabung des Datums in MySQL

MySQL verfügt über eine Vielzahl von Datentypen, um Datums- und Zeitinformationen zu verarbeiten. Im Allgemeinen akzeptiert MySQL diverse Formate, wenn Daten in ein Feld eingegeben werden. Dennoch ist die Datums- und Zeitausgabe standardisiert, sie erscheint immer in einem vorgegebenen Format.

Alle Zeit- und Datumstypen haben einen gültigen Wertebereich und einem »NULL«-Wert, auf den ein Feld gesetzt wird, falls Sie versuchen, dort einen ungültigen Wert einzufügen. Zeitformate besitzen eine intuitive Ordnung, wie sie auch im richtigen Leben vorkommt. Von links nach rechts gelesen: Stunden, gefolgt von Minuten und Sekunden.

Ein Datum wird hingegen immer mit der Jahreszahl links, gefolgt von Monat und Tag angezeigt (niemals Tag/Monat/Jahr).

Wenn MySQL Datum und Zeit ausgibt, haben Sie normalerweise die Möglichkeit, es anzuweisen, den gewünschten Wert als String (Zeichenkette) oder numerischen Wert auszugeben, auch wenn beide Ausgabearten die gleiche Information liefern. Das verwendete Format wird von dem Umfeld abhängen, in dem Sie die Informationen benötigen.

MySQL ist sehr flexibel, wenn es um das Akzeptieren von Zeit- und Datumseingaben geht: Das Datum 2001-05-12 ist für MySQL zum Beispiel genauso gültig wie 2001/05/12, 1+5+12 oder sogar 20010512. Sie können eine Vielzahl von Trennzeichen zwischen den einzelnen Datumsteilen verwenden – oder auch gar keine. Sie können sogar führende Nullen für Zahlen, die kleiner als 10 sind, weglassen.

MySQL überprüft nur teilweise, ob ein eingegebenes Datum auch wirklich gültig ist. Es überprüft zum Beispiel, dass sich Tage im Bereich von 1-31 und Monate im Bereich von 1-12 bewegen. Ob ein Datum wirklich existiert, wird jedoch nicht geprüft. So wird zum Beispiel der 30. Februar als gültiges Datum akzeptiert, auch wenn es diesen real nicht gibt. Dies macht MySQL effizienter, während es Werte entgegennimmt, indem es Ihrer Applikation die Verantwortung überträgt, zu prüfen, ob ein Datum wirklich gültig.

Datumsbezogene Datentypen

MySQL verfügt über eine Vielzahl von datumsbezogenen Datentypen, die Ihnen das Leben erleichtern, wenn Sie mit Zeit- und Datumsinformationen arbeiten müssen. Es bietet Ihnen `DATETIME`-, `DATE`-, `TIME`-, `YEAR`- und `TIMESTAMP`-Typen. Werfen Sie einen schnellen Blick auf die unten stehende Tabelle:

Datentyp	Standardformat	Null-Wert
<code>DATETIME</code>	<code>YYYY-MM-DD HH:MM:SS</code>	<code>0000-00-00 00:00:00</code>
<code>DATE</code>	<code>YYYY-MM-DD</code>	<code>0000-00-00</code>
<code>TIME</code>	<code>HH:MM:SS</code>	<code>00:00:00</code>
<code>YEAR</code>	<code>YYYY</code>	<code>0000</code>
<code>TIMESTAMP</code>	Unterschiedlich (vergleichen Sie hierzu die <code>TIMESTAMP</code> -Sektion in einem späteren Abschnitt)	<code>0000000000000000</code> (im längsten Fall)

Tabelle 16.1: Datentypen für Zeit und Datum

DATETIME

Das `DATETIME`-Format enthält die kompletten Datums- und Zeitinformationen eines Ereignisses bis auf die Sekunde genau.

Der Wertebereich von `DATETIME` liegt zwischen `1000-01-01 00:00:00` und `9999-12-31 23:59:59`.

DATE

Das `DATE`-Format speichert die Datumsinformation mit Tagesgenauigkeit also ohne Zeitinformation. MySQL ist bei seiner Speicherung Jahr-2000-kompatibel (Y2K).

Der Wertebereich von `DATE` liegt zwischen `1000-01-01` und `9999-12-31`.

TIME

Der `TIME`-Datentyp speichert die Zeit eines Ereignisses unabhängig von einem bestimmten Datum.

Der Wertebereich von `TIME` befindet sich nicht nur zwischen `00:00:00` und `23:59:59`, sondern er liegt tatsächlich zwischen `-838:59:59` und `838:59:59`. Dadurch haben Sie die Möglichkeit, nicht nur die Zeit eines Tages anzugeben, sondern auch die vergangene Zeit bzw. die Zeitdifferenz zwischen zwei Ereignissen.

Wie schon bei `DATE` und anderen Datumstypen hat MySQL einen ziemlich lockeren Umgang bei der Akzeptanz von gültigen Eingabewerten, wenn es darum geht, welche Trennzeichen Sie zwischen den einzelnen Teilangaben einer Zeitanzeige wie Stunde, Minute und Sekunde verwenden. Folglich ist 08:32:30 das Gleiche wie 08-32-20 oder sogar (ohne führende Nullen) 8.32.20.

Sie können `TIME` auch komplett ohne Trennzeichen angeben, zum Beispiel 083220 oder 83220. Aber seien Sie vorsichtig und werden Sie nicht zu minimalistisch: MySQL liest `TIME` von rechts beginnend ein, verlangt also, dass Sekunden angegeben sind, jedoch nicht notwendigerweise auch Stunden. Demzufolge werden 8:32 oder 0832 als 00:08:32 (acht Minuten und 32 Sekunden nach Mitternacht) verstanden und nicht als 8:32 a.m (acht Uhr zweiundreißig).



Gültige Zeitangaben, die außerhalb des gültigen Wertebereichs liegen, werden an der -838:59:59 und 838:59:59-Grenze abgeschnitten und nicht auf 00:00:00 gesetzt. Hingegen werden ungültige Zeitangaben wie z.B. Minuten oder Sekunden über 59 auf 00:00:00 gesetzt. Das heißt, wenn Sie eine Zeit mit dem Wert 00:00:00 sehen, welches für sich genommen ein gültiger Wert ist, können Sie niemals wissen, ob versucht wurde, eine gültige Zeit einzugeben oder nicht.

YEAR

Der Datentyp `YEAR` speichert das Jahr eines Ereignisses und bedarf nur eines Byte an Speicherplatz.

Der Wertebereich von `YEAR` liegt zwischen 1901 und 2155. Ein Jahr kann entweder als String oder als numerischer Wert spezifiziert werden.

Werte für den Datentyp `YEAR` können entweder zwei- oder vierstellig angegeben werden, jedoch werden zweistellige Angaben in vierstellige umgewandelt. Jahre können als numerischer Wert (im Bereich von 1 bis 99) oder als Zeichenkette (im Bereich von ,0' oder ,00' bis ,99') angegeben werden.

Zweistellige Jahresangaben, die im Bereich von 1 bis 69 (oder 0 bis 69) angegeben werden, werden als Jahre von 2001 bis 2061 (bzw. 2000 bis 2069) interpretiert. Jahresangaben im Bereich von 70 bis 99 werden als Jahre von 1970 bis 1999 interpretiert.

TIMESTAMP

`TIMESTAMP` ist ein sehr nützliches Feldformat, wobei das betreffende Feld immer auf das aktuelle Datum und die aktuelle Zeit gesetzt wird, wenn in der entsprechenden Zeile der Tabelle eine Eingabe oder Aktualisierung erfolgt ist. Dieser Datentyp bietet Ihnen demnach ein »zuletzt aktualisiert«-Feld, ohne das Sie selbst nach jeder Änderung diese Information neu setzen müssten.

`TIMESTAMP` kann nur Zeit- und Datumsinformationen zwischen 1970 und 2037 verwalten, und das nur bis zu einer Genauigkeit von einer Sekunde.

Obwohl ein `TIMESTAMP`-Feld intern immer gleich verwaltet wird, kann es eine Vielzahl von externen Formaten annehmen. So kann es extern aus einer Zahlenlänge zwischen 2 und 14 Stellen bestehen. Die nachfolgende Tabelle illustriert alle Formate, die `TIMESTAMP` Ihnen bietet:

Datentyp	Erscheinungsbild
<code>TIMESTAMP(14)</code>	YYYYMMDDHHMMSS
<code>TIMESTAMP(12)</code>	YYMMDDHHMMSS
<code>TIMESTAMP(10)</code>	YYMMDDHHMM
<code>TIMESTAMP(8)</code>	YYYYMMDD
<code>TIMESTAMP(6)</code>	YYMMDD
<code>TIMESTAMP(4)</code>	YYMM
<code>TIMESTAMP(2)</code>	YY

Tabelle 16.2: `TIMESTAMP`-Anzeigeformate

Werfen Sie einen Blick auf Tabelle 16.2. Wenn eine Aktualisierung am 12. Mai 2000 um 9:30 Uhr erfolgt, so wird ein `TIMESTAMP(14)` auf 20000512093000 gesetzt, während ein `TIMESTAMP(8)` auf 20000512 gesetzt wird, weil dieser keine Zeitinformaten enthält.



`TIMESTAMP`-Felder werden auf die aktuelle Zeit gesetzt, solange Sie diese nicht explizit mit eigenen Werten füllen. Mit anderen Worten: Sie müssen bei `INSERT`- oder `UPDATE`-Anweisungen den Wert nicht explizit angeben, oder Sie setzen den Wert auf `NULL`. Jede dieser Möglichkeiten wird das `TIMESTAMP`-Feld auf die aktuelle Zeit setzen. Das Gleiche erreichen Sie, wenn Sie das Feld auf `NOW()` setzen. `NOW()` ist eine Funktion, die das aktuelle Datum und die aktuelle Zeit zurückgibt. Mit dieser Funktion werden Sie später noch nähere Bekanntschaft machen.

Sie können eine gewünschte Zeitinformaten in ein `TIMESTAMP`-Feld schreiben, indem Sie diese explizit angeben. Sie können den Wert eines `TIMESTAMP`-Feldes auf jeden gewünschten Wert setzen, solange Sie sich dabei im gültigen Wertebereich bewegen (man könnte annehmen, dass dieses den wahren Zweck eines `TIMESTAMP`-Feldes verfehlt!). Werte außerhalb des Wertebereichs werden ignoriert und durch `NULL`-Werte ersetzt.

Ebenso wie bei den `DATETIME`-, `DATE`- und `TIME`-Datentypen ist MySQL auch beim `TIMESTAMP`-Datentyp sehr flexibel, wenn es um gültige Eingabeformate geht. So akzeptiert es zum Beispiel 2000-05-12, 2000+05+12 usw. und erlaubt Ihnen auf diese Weise, jedes von Ihnen gewünschte Trennzeichen zu verwenden oder auch keines wie bei 20000512.



Wenn Sie ein Feld mit dem Datentyp `TIMESTAMP` erstellen, sollten die einzugebenden Werte eher als numerischer Wert als als Zeichenkette spezifiziert werden, und 14, 12, 8 oder 6 Zeichen lang sein. Definieren Sie hingegen ein Datum als Zeichenkette, sollte es zumindest 6 Zeichen lang sein. Der Grund dafür ist, dass ein Datum wie `98-12` oder `98`, welches Sie für Dezember 98 oder das Jahr 1998 halten, von MySQL mit Null-Werten für Tag bzw. Monat und Tag bedacht wird. Das würde dazu führen, dass das von Ihnen eingegebene Datum ungültig und dem Feld ein `NULL`-Wert zugewiesen würde. Daher sollten Sie nicht vergessen ein Datum immer mit Jahr, Monat und Tag zu definieren, auch wenn Sie nur an dem Wert des Jahres oder des Jahres und des Monats interessiert sind.

Alle `TIMESTAMP`-Felder haben den gleichen Speicherplatzverbrauch von 4 Byte. Sie können ein `TIMESTAMP`-Feld daher mit `ALTER TABLE` verändern, ohne dass dies einen Datenverlust nach sich zieht. Wenn Sie die Länge eines Feldes erweitern, wird es mit Daten aus dem zuvor versteckten Teil des Feldes gefüllt. Und wenn Sie ein Feld verkürzen, gehen die Informationen nicht verloren; das Feld wird lediglich weniger Informationen anzeigen, wenn Sie das nächste Mal darauf zugreifen.



Ein `TIMESTAMP`-Feld wird, wenn Sie es mit `NOW()` oder `NULL` belegen, von MySQL immer auf die aktuelle Zeit gesetzt. Wenn Sie das Feld in einer `INSERT`- oder `UPDATE`-Anweisung jedoch nicht explizit erwähnen, wird nur das erste `TIMESTAMP`-Feld der Tabelle aktualisiert und auf die aktuelle Zeit gesetzt. Nachfolgende `TIMESTAMP`-Felder werden mit `NULL`-Werten gefüllt (welches an sich ein gültiger Wert ist), wenn Sie hier keinen anderen Wert angeben.

Die Übertragung von Daten zwischen Datentypen

MySQL ermöglicht den Datentransfer zwischen verschiedenen Datentypen. Sie könnten zum Beispiel ein `DATETIME`-Feld haben, das den genauen Zeitpunkt einer Kundentransaktion gespeichert hat. Später möchten Sie diese Daten in eine andere Tabelle übertragen, sind aber nur am Kaufdatum interessiert.

Wenn Sie Daten aus einer `DATETIME`- oder `TIMESTAMP`-Spalte haben und diese einer `DATE`-Spalte zuweisen wollen, wird die Zeitinformation verloren gehen, weil der `DATE`-Typ keinen Speicherplatz für die Zeitinformation hat.

Möchten Sie im Gegensatz dazu einen `DATE`-Wert in eine `DATETIME`- oder `TIMESTAMP`-Spalte einfügen, so wird MySQL `00:00:00` für die Zeitinformation einfügen, da diese Daten zuvor nicht enthalten waren.



Beachten Sie, dass die verschiedenen Feldtypen unterschiedliche Wertebereiche haben, und die betreffende Spalte in der Lage sein muss, die gewünschten Werte aufzunehmen. Andernfalls gehen diese Informationen

verloren. Der Versuch, ein Datum außerhalb des für ein `TIMESTAMP`-Feld gültigen Wertebereiches zwischen 1970 und 2037 in ein solches einzufügen, führt dazu, dass dieses mit einem `NULL`-Wert versehen wird.

Interne MySQL-Datums-Funktionen

MySQL bietet Ihnen eine Fülle von Datums- und Zeitfunktionen, die Sie im Zusammenhang mit `SELECT`-Anweisungen und `WHERE`-Bedingungen benutzen können.

So gibt zum Beispiel die Funktion `MONTHNAME()` den Namen des Monats aus einem Datumsfeld zurück. Sie können diese Funktion in einer `SELECT`-Anweisung verwenden; `SELECT MONTHNAME(20000105)` gibt beispielsweise Januar zurück. In ähnlicher Weise könnten Sie folgende Anweisung ausführen

```
mysql> SELECT mod_date FROM orders WHERE MONTHNAME(mod_date)='January';
```

die folgende Ausgabe produziert:

```
+-----+
| mod_date |
+-----+
| 20000104 |
| 20000117 |
| 20000117 |
+-----+
3 rows in set (0.06 sec)
```

Wenn einer internen Funktion von MySQL ein Datum übergeben wird, so muss dieses Datum die Form `YYYYMMDD` (bei numerischen Werten) bzw. `YYYY-MM-DD` (bei Zeichenketten) haben. Dies sollten Sie auf keinen Fall vergessen.

Sie werden im Folgenden einen näheren Blick auf die verfügbaren Funktionen werfen.

Funktionen zur Ausgabe von Tagesinformationen

MySQL stellt Ihnen vier Funktionen zur Konvertierung eines Datums in Tagen zur Verfügung:

`DAYOFYEAR(Datum)`

`DAYOFMONTH(Datum)`

`DAYOFWEEK(Datum)`

`WEEKDAY(Datum)`

`DAYOFYEAR(Datum)` gibt Ihnen den Tag eines Jahres für ein der Funktion übergebenes Datum im numerischen Format zurück; mit anderen Worten erhalten Sie die Anzahl der Tage, die seit dem 1. Januar des betreffenden Jahres vergangen sind.

Der 1. Februar 2000

```
mysql> SELECT DAYOFYEAR(20000201);
```

ergibt z.B.:

```
+-----+
| DAYOFYEAR(20000201) |
+-----+
|                    32 |
+-----+
```

Auf ähnliche Weise können Sie der Funktion `DAYOFYEAR` auch ein Datum als Zeichenkettenparameter übergeben:

```
mysql> SELECT DAYOFYEAR(,2000/02/01');
```

```
+-----+
| DAYOFYEAR(,2000/02/01') |
+-----+
|                    32 |
+-----+
```

`DAYOFMONTH(Datum)` gibt den Tag des Monats für ein der Funktion übergebenes Datum zurück.

Wenig überraschend ergibt die Abfrage für den 1. Februar 2000

```
mysql> SELECT DAYOFMONTH(20000201);
```

```
+-----+
| DAYOFMONTH(20000201) |
+-----+
|                    1 |
+-----+
```

oder wenn das Datum als Zeichenkette übergeben wird:

```
mysql> SELECT DAYOFMONTH('2000-02-01');
```

```
+-----+
| DAYOFMONTH(,2000-02-01') |
+-----+
|                    1 |
+-----+
```

`DAYOFWEEK(Datum)` gibt den Tag der Woche für ein gegebenes Datum zurück, beginnend mit 1 für Sonntag, 2 für Montag, 3 für Dienstag usw.

So ergibt zum Beispiel die Anweisung

```
mysql> SELECT DAYOFWEEK('0-2-1');
```

die folgende Ausgabe:

```
+-----+
| DAYOFWEEK('0-2-1') |
+-----+
|                    3 |
+-----+
1 row in set (0.22 sec)
```

WEEKDAY(Datum) funktioniert hingegen etwas anders. Sie gibt den Wochentag-Index für ein gegebenes Datum zurück (0=Montag, 1=Dienstag, 2=Mittwoch usw.):

```
mysql> SELECT WEEKDAY('0-2-1');
```

```
+-----+
| WEEKDAY('0-2-1') |
+-----+
|                  1 |
+-----+
1 row in set (0.11 sec)
```

Ausgabe von Namen für Monate und Tage

Es gibt zwei Funktionen, um Monats- und Tagesnamen für ein bestimmtes Datum ausgeben zu lassen:

MONTHNAME(Datum)

DAYNAME(Datum)

MONTHNAME(Datum) kann verwendet werden, um den Namen eines Monats für ein als Zeichenkette übergebendes Datum ausgeben zu lassen.

```
mysql> SELECT MONTHNAME('0-2-1');
```

gibt Folgendes aus:

```
+-----+
| MONTHNAME('0-2-1') |
+-----+
| February            |
+-----+
1 row in set (2.03 sec)
```

Die Funktion DAYNAME(Datum) arbeitet ähnlich:

```
mysql> SELECT DAYNAME('2000-01-01');
```

und produziert folgende Ausgabe:

```
+-----+
| DAYNAME('2000-01-01') |
+-----+
| Saturday              |
+-----+
1 row in set (0.06 sec)
```

Beachten Sie allerdings, dass es sich bei `Datum` um ein gültiges Datum handeln muss. Andernfalls wird die Funktion fehlschlagen und `NULL` zurückgeben:

```
mysql> SELECT MONTHNAME('9');
+-----+
| MONTHNAME('9') |
+-----+
| NULL           |
+-----+
1 row in set (0.00 sec)
```

Ausgabe von Jahr, Quartal, Monat und Woche

MySQL bietet eine Auswahl an Funktionen, mit denen Sie numerische Daten wie Jahr, Quartal, Monat oder Woche aus einem Datum extrahieren können:

```
YEAR(Datum)
QUARTER(Datum)
MONTH(Datum)
WEEK(Datum [, Wochenstarttag])
YEARWEEK(Datum[,Wochenstarttag])
```

`YEAR(Datum)` gibt das vierstellige numerische Jahr eines der Funktion übergebenen Datums zurück:

```
mysql> SELECT YEAR('01/12/25 11:00:00');
+-----+
| YEAR('01/12/25 11:00:00') |
+-----+
| 2001 |
+-----+
1 row in set (2.09 sec)
```

`QUARTER(Datum)` gibt das Quartal von 1 bis 4 aus:

```
mysql> SELECT QUARTER('01/12/25 11:00:00');
```

```
+-----+
| QUARTER('01/12/25 11:00:00') |
+-----+
|                               4 |
+-----+
```

```
1 row in set (0.05 sec)
```

MONTH(Datum) gibt den Monat von 1 bis 12 aus:

```
mysql> SELECT MONTH('01/12/25 11:00:00');
```

```
+-----+
| MONTH('01/12/25 11:00:00') |
+-----+
|                               12 |
+-----+
```

```
1 row in set (0.00 sec)
```

Die Funktion WEEK(Datum) gibt bei Übergabe nur eines Parameters (Datum) die Wochenzahl des Jahres von 0 bis 53 aus:

```
mysql> SELECT WEEK('2001-12-26');
```

```
+-----+
| WEEK('2001-12-26') |
+-----+
|                     52 |
+-----+
```

```
1 row in set (2.14 sec)
```

Ohne Angabe des zweiten optionalen Parameters geht MySQL davon aus, dass Sonntag der erste Tag der Woche ist und am Anfang des Jahres alle Tage vor dem Wochenstarttag in Woche 0 liegen. So gibt WEEK(,2000-01-01') 0 zurück, da es sich beim 1. Januar 2000 um einen Samstag handelt.

Dieses Verhalten können Sie mittels des zweiten Parameters, *Wochenstarttag*, überschreiben. Anstatt die Woche mit dem Sonntag beginnen zu lassen (Sonntag=0), können Sie einen anderen Wochentag als Starttag bestimmen.

Ohne den zweiten Parameter gibt WEEK(,2000-01-09') 2 zurück, da es sich hier um einen Sonntag handelt (Woche 2 des Jahres).

Geben Sie als zweiten Parameter jedoch 1 an, gibt WEEK(,2000-01-09', 1) 1 zurück, da Sie MySQL angewiesen haben, den Montag als Wochenstarttag zu definieren.

YEARWEEK(Datum [, Wochenstarttag]) arbeitet sehr ähnlich, doch wird die Wochenzahl hinten an die Jahreszahl gehängt. In dem vorhergehenden Beispiel würde YEARWEEK(,2000-01-09', 1) 200001 zurückgeben.

Das alles klingt möglicherweise etwas akademisch, aber international gesehen haben auch nicht alle Kalender die gleichen Konventionen.

Aktuelles Datum und Zeit

Es gibt verschiedene Funktionen in MySQL, um das aktuelle Datum bzw. die aktuelle Zeit auszugeben.

`NOW()` / `SYSDATE()` / `CURRENT_TIMESTAMP()`

`CURDATE()` / `CURRENT_DATE()`

`CURTIME()` / `CURRENT_TIME()`

Um die vollständige Datums- und Zeitangabe im `DATETIME`-Format zu erhalten, sollten Sie `NOW()` benutzen. Dies ist der einfachste Weg.

Die Ausgabe kann entweder als Zeichenkette zurückgegeben werden:

```
mysql> SELECT NOW();
+-----+
| NOW() |
+-----+
| 2000-10-06 19:39:01 |
+-----+
1 row in set (0.11 sec)
```

oder im numerischen Format:

```
mysql> SELECT NOW()+0;
+-----+
| NOW()+0 |
+-----+
| 20001006194956 |
+-----+
1 row in set (0.00 sec)
```

`SYSDATE()` und `CURRENT_TIMESTAMP()` geben genau das Gleiche wie `NOW()` aus.

Möchten Sie lediglich das aktuelle Datum ausgegeben bekommen, verwenden Sie `CURDATE()` oder `CURRENT_DATE()`, die identisch sind.

```
mysql> SELECT CURDATE();
+-----+
| CURDATE() |
+-----+
| 2000-10-06 |
+-----+
1 row in set (0.11 sec)
```

Sind Sie lediglich an der aktuellen Uhrzeit interessiert, benutzen Sie `CURTIME()` bzw. `CURRENT_TIME()`. Beide Funktionen sind äquivalent und geben eine Uhrzeit wie zum Beispiel 19:57:43 zurück (bzw. 195743 in einem numerischen Kontext).

Formatierung von Datum und Zeit

DATE_FORMAT(Datum, Format)

TIME_FORMAT(Zeit, Format)

Die Datums- und Zeitformat-Funktionen sind sehr nützlich, da sie Ihnen ermöglichen, ein Datum bzw. eine Zeitangabe in fast jeder gewünschten Weise darzustellen. Neben der Zeit bzw. dem Datum übergeben Sie der Funktion ein Formatzeichen als Parameter. Diese sind in der folgenden Tabelle aufgeführt:

Format Parameter	Ausgabeformat
%r	12-Stunden Anzeige (<i>hh:mm:ss (AM PM)</i>)
%T	24-Stunden-Anzeige (<i>hh:mm:ss</i>)
%Y	Nummerisches Jahr, 4-stellig
%y	Nummerisches Jahr, 2-stellig
%m	Nummerischer Monat mit führender Null (01, 02, ..., 12)
%c	Nummerischer Monat ohne führende Null (1, 2, ..., 12)
%M	Monatsname (Januar, Februar, ...)
%b	Abgekürzter Monatsname (Jan, Feb, ...)
%D	Tag des Monats mit englischem Suffix (1st, 2nd, 3rd, ...)
%d	Tag des Monats mit führender Null (01, 02, 03, ..., 31)
%e	Tag des Monats ohne führende Null (1, 2, 3, ..., 31)
%W	Wochentagsname (Sonntag, Montag, ...)
%a	Abgekürzter Wochentagsname (So, Mo, ...)
%H	Stunde (00, 01, ..., 23)
%k	Stunde (0, 1, ..., 23)
%h	Stunde (01, 02, ..., 12)
%I	Stunde (01, 02, ..., 12)
%l	Stunde (1, 2, ..., 12)
%i	Minuten (00, 01, ..., 59)
%S	Sekunden (00, 01, ..., 59)
%s	Sekunden (0, 1, ..., 59)
%p	AM oder PM
%U	Wochennummer des Jahres (0–53), Sonntag als erster Tag der Woche
%u	Wochennummer des Jahres (0–53), Montag als erster Tag der Woche

Tabelle 16.3: DATE_FORMAT- und TIME-FORMAT-Ausgaben

Format Parameter Ausgabeformat

%X & %V	Jahr- und Wochennummer, Sonntag als erster Tag der Woche
%x & %v	Jahr- und Wochennummer, Montag als erster Tag der Woche
%j	Tag des Jahres mit führender Null (001, 002, ..., 366)
%W	Wochentagsnummer (Sonntag=1, Montag=2, ...)
%%	Prozentzeichen

Tabelle 16.3: DATE_FORMAT- und TIME_FORMAT-Ausgaben (Forts.)

Einige Beispiele zu den Formatfunktionen:

- ▶ SELECT DATE_FORMAT('1999-12-31 23:00:00', '%r on %W') gibt 11:00:00 PM on Friday aus.
- ▶ SELECT DATE_FORMAT('2001-11-05', '%D %M') gibt 5th November aus.
- ▶ SELECT TIME_FORMAT(NOW(), '%H:%i:%s') könnte etwas ähnliches wie 20:39:35 ergeben.



DATE_FORMAT() ist die flexiblere der beiden Funktionen. Ihr können entweder Datums-, Zeit- oder Datums- und Zeit-Informationen übergeben werden. TIME_FORMAT() kann nur Zeitinformationen verarbeiten. Andernfalls gibt die Funktion ein NULL-Ergebnis zurück.

Ausgabe von Stunden, Minuten und Sekunden

HOUR(Zeit)
 MINUTE(Zeit)
 SECOND(Zeit)

Diese Funktionen können verwendet werden, um aus einer gegebenen Zeit auf einfache Art und Weise die Stunde, Minute oder Sekunde zu extrahieren.

HOUR() gibt eine Zahl im Bereich von 0 bis 23 zurück. HOUR(18:10:27) gibt zum Beispiel 18 zurück. Sie können auch das Folgende ausprobieren:

```
mysql> SELECT HOUR(NOW());
+-----+
| HOUR(NOW()) |
+-----+
|          20 |
+-----+
1 row in set (0.06 sec)
```

MINUTE() und SECOND() funktionieren auf die gleiche Weise und geben beide einen Wert zwischen 0 und 59 zurück.

Ausgabe der Tage seit 1 n. Chr.

TO_DAYS(Datum)
FROM_DAYS(Zahl)

TO_DAYS(Datum) erwartet ein Datum als Übergabeparameter und gibt die Anzahl der Tage seit dem ersten Tag des Jahres 1 n. Chr. zurück (ein Jahr 0 existiert nicht!). Folglich gibt TO_DAYS(,2000-10-06') die Zahl 730764 zurück.

Den gegenteiligen Effekt erreichen Sie mit FROM_DATE(Zahl). SELECT FROM_DAYS(1000000) würde 2737-11-28 als den millionsten Tag nach dem 1. Tag des Jahres 1 ausgeben.

Diese Funktionen sollten erst auf Daten nach 1582 angewandt werden, in dem der gregorianische Kalender eingeführt wurde.

Ausgabe der Sekunden seit Tagesanfang

TIME_TO_SEC(Zeit)
SEC_TO_TIME(Sekunden)

TIME_TO_SEC(zeit) gibt die Zeit in Sekunden zurück, die seit Anfang des Tages vergangen sind.

So gibt zum Beispiel TIME_TO_SEC(,01.00.00') 3600 zurück. Folgende Anweisung:

```
mysql> SELECT TIME_TO_SEC(110);
```

ergibt

```
+-----+
| TIME_TO_SEC(110) |
+-----+
|                70 |
+-----+
1 row in set (0.00 sec)
```

da 110 (numerisch) von MySQL als 1 Minute und 10 Sekunden interpretiert wird.

SEC_TO_TIME(Sekunden) macht genau das Gegenteil. Die Funktion konvertiert die Sekunden, die seit Anfang des Tages vergangen sind, in das HH:MM:SS bzw. HHMMSS-Format. Folgende Anweisung:

```
mysql> SELECT SEC_TO_TIME(70);
```

ergibt

```
+-----+
| SEC_TO_TIME(70) |
+-----+
| 00:01:10        |
+-----+
1 row in set (0.06 sec)
```

bzw. wenn es in einem numerischen Kontext verwendet wird, ergibt

```
mysql> SELECT SEC_TO_TIME(70)+0;
```

folgende Ausgabe

```
+-----+
| SEC_TO_TIME(70)+0 |
+-----+
|                  110 |
+-----+
1 row in set (0.05 sec)
```

Der Unix-Zeitstempel

UNIX_TIMESTAMP([Datum])

FROM_UNIXTIME(UNIX_TIMESTAMP[, Format])

Der UNIX_TIMESTAMP enthält die Anzahl der Sekunden seit dem 1. Januar 1970. UNIX_TIMESTAMP() ohne den optionalen Parameter *Datum* gibt den aktuellen Unix-Zeitstempel zurück (UNIX_TIMESTAMP() könnte zum Beispiel 970863031 zurückgeben).

Mit dem Parameter *Datum* im DATE-, DATETIME- oder TIMESTAMP-Format oder in numerischer Form, YYYYMMDD oder YYYYMMDD, gibt die Funktion den Unix-Zeitstempel für dieses Datum zurück:

```
mysql> SELECT UNIX_TIMESTAMP('2020-10-04 22:23:00');
```

```
+-----+
| UNIX_TIMESTAMP('2020-10-04 22:23:00') |
+-----+
|                  1601842980 |
+-----+
1 row in set (0.00 sec)
```

FROM_UNIXTIME() hat genau den gegenteiligen Effekt. Wird diese Funktion mit UNIX_TIMESTAMP als grundlegender Parameter verwendet, gibt sie Datum und Zeit entweder im YYYY-MM-DD HH:MM:SS- oder im YYYYMMDDHHMMSS-Format zurück, abhängig davon, ob sie in einer Zeichenkette oder einem numerischen Kontext verwendet wurde.

So würde folgende Anweisung:

```
mysql> SELECT FROM_UNIXTIME(1601842980)*1;
```

eine numerische Datums- und Zeitinformation ausgeben:

```
+-----+
| FROM_UNIXTIME(1601842980)*1 |
+-----+
|                20201004222300 |
+-----+
1 row in set (0.05 sec)
```

Sie können dieser Funktion optional einen Format-Parameter übergeben, der die Ausgabe nach den Regeln der `FORMAT_DATE()`-Funktion formatiert (siehe Tabelle 16.3). So können Sie zum Beispiel den Tag, auf den die milliardste Sekunde der Unix-Epoche fällt, ausgeben lassen:

```
mysql> SELECT FROM_UNIXTIME(1000000000, '%W, %M');
+-----+
| FROM_UNIXTIME(1000000000, '%W, %M') |
+-----+
| Sunday, September                    |
+-----+
1 row in set (0.05 sec)
```

Datums- und Zeitberechnungen

MySQL bietet mehrere Wege zur Datums- und Zeitberechnung. Folgende Funktionen stehen hierfür zur Verfügung:

```
DATE_ADD(Datum, INTERVALL Ausdruck Typ)
ADDDATE(Datum, INTERVALL Ausdruck Typ)
DATE_SUB(Datum, INTERVALL Ausdruck Typ)
SUBDATE(Datum, INTERVALL Ausdruck Typ)
PERIOD_ADD(Periode, Monate)
PERIOD_DIFF(Periode 1, Periode 2)
```

`DATE_ADD()` nimmt ein Datum, addiert ein Zeitintervall, das durch Ausdruck gegeben ist, und gibt entsprechend das Datum und die Zeit zurück.



Diese Funktionen sind neu seit MySQL 3.22. In MySQL 3.23 haben Sie die Möglichkeit, statt `DATE_ADD()` und `DATE_SUB()` + bzw. - zu verwenden. Neu ist auch die Funktion `EXTRACT (Typ FROM Datum)`. Diese wird verwendet, um Datums- bzw. Zeitinformationen zu extrahieren. Parameter, die Sie für *Typ* verwenden können, finden Sie in der folgenden Tabelle:

Typangabe	Bedeutung und Format
SECOND	Sekunden
MINUTE	Minuten
HOURL	Stunden
DAY	Tage
MONTH	Monate
YEAR	Jahre
MINUTE_SECOND	"Minuten:Sekunden"
HOURL_MINUTE	"Stunden:Minuten"
DAY_HOURL	"Tage Stunden"
YEAR_MONTH	"Jahre-Monate"
HOURL_SECOND	"Stunden:Minuten:Sekunden"
DAY_MINUTE	"Tage Stunden:Minuten"
DAY_SECOND	"Tage Stunden:Minuten:Sekunden"

Tabelle 16.4: Werte für Ausdruck bei Zeit- und Datumsberechnung

Betrachten Sie nun einige Beispiele. Nehmen Sie zunächst den 4. Juli 1980 und addieren Sie 15 Jahre hinzu:

```
mysql> SELECT DATE_ADD('1980-07-04', INTERVAL 15 YEAR);
+-----+
| DATE_ADD('1980-07-04', INTERVAL 15 YEAR) |
+-----+
| 1995-07-04                               |
+-----+
1 row in set (0.11 sec)
```

Jetzt addieren Sie in einem weiteren Beispiel 6 Monate zum letzten Weihnachten:

```
mysql> SELECT DATE_ADD('1999-12-24', INTERVAL 6 MONTH);
+-----+
| DATE_ADD('1999-12-24', INTERVAL 6 MONTH) |
+-----+
| 2000-06-24                               |
+-----+
1 row in set (0.06 sec)
```

Experimentieren Sie nun ein wenig mit der Zeitmanipulation. Addieren Sie 72 Stunden zu Montagmorgen, 9 Uhr:

```
mysql> SELECT DATE_ADD('2000-10-02 09:00:00', INTERVAL 72 HOUR);
```

```
+-----+
| DATE_ADD('2000-10-02 09:00:00', INTERVAL 72 HOUR) |
+-----+
| 2000-10-05 09:00:00                               |
+-----+
```

```
1 row in set (0.00 sec)
```

Finden Sie heraus, welches Datum vor 12 Tagen war? Benutzen Sie hierfür die Funktion `DATE_SUB()`:

```
mysql> SELECT DATE_SUB(NOW(),INTERVAL 12 DAY);
```

```
+-----+
| DATE_SUB(NOW(),INTERVAL 12 DAY) |
+-----+
| 2000-09-24 23:33:29             |
+-----+
```

```
1 row in set (0.06 sec)
```

Sie können auch zusammengesetzte Intervalle verwenden; subtrahieren Sie zum Beispiel 10 Jahre und 4 Monate vom kommenden ersten Dezember:

```
mysql> SELECT DATE_SUB(20001201, INTERVAL "10 4"YEAR_MONTH);
```

produziert

```
+-----+
| DATE_SUB(20001201, INTERVAL "10 4"YEAR_MONTH) |
+-----+
| 1990-08-01                                     |
+-----+
```

```
1 row in set (0.00 sec)
```



Wenn Sie vergessen, die volle Länge eines Feldes wie erwartet anzugeben, geht MySQL davon aus, dass Sie die am weitesten links stehenden Informationen ausgelassen haben. Demzufolge gibt `DATE_SUB(20001201, INTERVAL "10" YEAR_MONTH)` `2000-02-01` aus, da es 10 Monate subtrahiert und nicht 10 Jahre.

`PERIOD_ADD(Zeitraum, Monate)` nimmt einen *Zeitraum* im Format `YYMM` oder `YYYYMM` und addiert eine gewisse Anzahl von *Monaten* zu diesem hinzu (ein *Zeitraum* im Format `YYMM` wird von MySQL als abgekürzte vierstellige Jahresangabe bewertet und dementsprechend konvertiert). Sie können zum Beispiel 12 Monate zum 8. Juni hinzudieren:

```
mysql> SELECT PERIOD_ADD(0806,12);
```

```
+-----+
| PERIOD_ADD(0806,12) |
+-----+
|          200906 |
+-----+
1 row in set (0.00 sec)
```

Sie können dieselbe Funktion auch zur Subtraktion verwenden; subtrahieren Sie zum Beispiel 2 Monate vom April 2000:

```
mysql> SELECT PERIOD_ADD(200004, -2);
```

```
+-----+
| PERIOD_ADD(200004, -2) |
+-----+
|          200002 |
+-----+
1 row in set (0.00 sec)
```



PERIOD_ADD() arbeitet mit Jahren und Monaten, übernimmt jedoch keinen Datums-Parameter. Wenn Sie versuchen, der Funktion ein Datum im DATE-Format zu übergeben, erhalten Sie ein ungültiges Ergebnis.

Um die Differenz zwischen zwei Zeiträumen zu erhalten, verwenden Sie **PERIOD_DIFF(Zeitraum 1, Zeitraum 2)**. Versuchen Sie zu berechnen, wie viele Monate zwischen dem Januar 1980 und dem Juni 2000 liegen:

```
mysql> SELECT PERIOD_DIFF(200006, 198001);
```

```
+-----+
| PERIOD_DIFF(200006, 198001) |
+-----+
|                245 |
+-----+
1 row in set (0.00 sec)
```

Genau wie **PERIOD_ADD()** erwartet auch **PERIOD_DIFF()** Zeitangaben im YYYYMM- bzw. YYYYMM-Format und nicht im normalen DATE-Format.

Zusammenfassung

In diesem Kapitel haben Sie verschiedene Formate kennen gelernt, die MySQL benutzt, um Datums- und Zeitausgaben zu formatieren. MySQL ist sehr flexibel, wenn es um das Akzeptieren von Datums- und Zeiteingaben geht. Es ermöglicht eine Reihe von Trennzeichen und Feldlängen und interpretiert Ihre Eingaben nach einem festen

Regelwerk. Wie auch immer – MySQL gibt die gleichen Daten in einem aussagekräftigen Format aus und passt Zeichenketten wie auch numerische Formate an, je nachdem wie Sie diese Werte in dem Ergebnis benutzen.

Sie haben verschiedene Formate kennen gelernt:

- ▶ DATETIME: für kombinierte Datums- und Zeitinformationen
- ▶ DATE: für Datumsinformationen
- ▶ TIME: für Zeitinformationen
- ▶ YEAR: für Jahresinformationen
- ▶ TIMESTAMP: speichert Datum und Zeit und setzt sich selbst automatisch auf den Erstellungs- bzw. letzten Modifikationszeitpunkt eines Datensatzes

Zudem wurde eine Reihe von Funktionen vorgestellt, die sowohl numerische als auch textliche Informationen aus einer Datums- bzw. Zeitangabe extrahieren. Sie haben gelernt, wie man zeitliche arithmetische Operationen anwenden kann, indem Sie MySQLs Werkzeuge zur Addition bzw. Subtraktion für Zeitintervalle verwenden. Letztendlich haben Sie erfahren, wie man die Differenz zwischen zwei Zeitpunkten ausrechnet.

Fragen und Antworten

- F** Welches ist der beste Weg, um die aktuelle Systemzeit zu erhalten und diese in meinen Datensatz einzufügen?
- A** *Die aktuelle Systemzeit wird am einfachsten mit der Funktion `NOW()` ermittelt, die sowohl das Datum als auch die Zeit ausgibt. Sie können diese Information in jede Ihrer Datums- und Zeit-Felder Ihrer Tabelle einfügen. Sie können jedoch auch eine Spalte mit dem `TIMESTAMP`-Datentyp belegen, was dazu führt, dass dieses Feld bei jeder Daten-Aktualisierung oder -eingabe in dieser Zeile aktualisiert wird (vorausgesetzt, Sie schreiben keine anderen Werte in dieses Feld, da dies dazu führen würde, dass die explizit angegebenen Daten dort eingefügt werden).*
- F** Wie kann ich Zeit und Datumsinformationen in einer benutzerfreundlichen Art und Weise darstellen?
- A** *Benutzen Sie die `DATE_FORMAT()`- und `TIME_FORMAT()`-Funktion. Diese Funktionen bieten Ihnen große Flexibilität bei der Formatierung von Datums- und Zeitinformationen nach den Bedürfnissen des Benutzers.*

- F** Welches ist das beste Format, um Datums- bzw. Zeitinformationen in MySQL einzugeben?
- A** *MySQL bietet auch hier größtmögliche Freiheit und kann Zeit- und Datumsinformationen in vielfachen Formaten akzeptieren. Als Trennzeichen akzeptiert MySQL sowohl Punkte als auch (Schräg-)Striche oder verzichtet sogar komplett auf diese. Um eine klare Linie beizubehalten, sollten Sie vielleicht Ihre Daten im gleichen Format eingeben, wie MySQL sie auch ausgibt (YYYY-MM-DD für Datumsangaben bzw. HH:MM:SS für Zeitangaben), wenn Daten in diesem Format zur Verfügung stehen.*

Übungen

1. Schreiben Sie eine `SELECT`-Anweisung, welche die aktuelle Zeit im Format »Stunde:Minute am/pm Wochentag nth Monat Jahr« ausgibt.
2. Schreiben Sie eine `SELECT`-Anweisung, um herauszufinden, auf welchen Wochentag der 13. Oktober 2010 fällt.
3. Schreiben Sie eine `SELECT`-Anweisung, die 1000 Tage auf den 1. April 2000 addiert und das Datum zurückgibt. Auf welchen Wochentag wird der folgende Tag fallen?



Datenbank- sicherheit in MySQL

**Woche
3**

Datenbanksicherheit ist eine wesentliche Komponente eines jeden Datenbanksystems. Gewisse Sicherheitsmaßnahmen sind notwendig, um Ihre Daten nicht nur vor potentiellen Hackern, sondern auch vor Benutzern zu schützen. Denn manchmal löschen Benutzer in Ihrer Unwissenheit Datensätze, die sie eigentlich nicht löschen wollten. Um solchen »Unfällen« vorzubeugen, können Sie ein Sicherheitslevel einrichten, um Benutzer vom Löschen, Aktualisieren oder Hinzufügen von Datensätzen abzuhalten.

Sicherheit spielt eine wichtige Rolle in jeder Applikation, auf die mehrere Benutzer zur gleichen Zeit Zugriff haben. MySQL handhabt Sicherheitsaspekte sehr gut und zählt heutzutage zu den sichersten Datenbanken auf dem Markt. Heute werden Sie Folgendes erfahren:

- ▶ wie Sicherheit in MySQL implementiert ist
- ▶ welche Ebenen der Kontrolle und welche Kontrollmechanismen es gibt
- ▶ wie Sie Benutzer und deren Privilegien hinzufügen bzw. editieren

Wie Sicherheit in MySQL implementiert ist

Das MySQL-Sicherheitssystem ist sehr flexibel. Mit diesem können Sie einem potentiellen Benutzer auf verschiedenen Ebenen Zugang zu Ihren Datenbanken gewähren – von der Möglichkeit, sich von einem spezifischen Rechner als ein spezifischer Benutzer einzuloggen, bis hin zu vollem Administrator-Zugang von überall her. Es ist Ihre Entscheidung, wie stark Ihre Datenbank gesichert sein soll. Dieses Kapitel bemüht sich, Ihnen einige Richtlinien und Ideen über Datenbanksicherheit an die Hand zu geben.

MySQL verwaltet alle Berechtigungen und Privilegien in der *mysql*-Datenbank. Diese ist eine von zwei Datenbanken, die automatisch bei der Installation von MySQL erstellt werden (die zweite ist die *test*-Datenbank). Die einzigen Benutzer, die Zugang zu dieser Datenbank haben sollten, sind die Administratoren. Die Datenbank unterscheidet sich nicht von jeder anderen MySQL Datenbank. Die Datendateien sind in *data*, einem Verzeichnis unter dem *mysql*-Eltern-Verzeichnis abgelegt, in dem auch alle anderen Dateien gespeichert sind. Die Tabellen dieser Datenbank sind

```
user
db
host
func
columns_priv
tables_priv
```

Sie können Anfragen an diese Datenbank richten, wie Sie sie auch an jede andere Datenbank richten würden. Diese Tabellen sind zusammengekommen auch als `GRANT`-Tabellen bekannt. Jede Spalte in einer solchen `GRANT`-Tabelle spiegelt wider, welche Berechtigungen ein Benutzer hat, indem diese entweder mit einem `Y` (das heißt der Benutzer ist berechtigt, eine bestimmte Aktion auszuführen) oder einem `N` (keine Berechtigung) belegt sind. Zum Beispiel hat ein Benutzer mit `DELETE`-Privilegien in der `user`-Tabelle ein `Y` in der `DELETE_PRIV`-Spalte.

Die user-Tabelle

Die `user`-Tabelle beinhaltet sämtliche Berechtigungen für alle Benutzer, die Zugang zu MySQL haben. Sie können hier alle Berechtigungen für Benutzer setzen. Diese Tabelle besteht aus den folgenden Spalten:

- ▶ `Host`: Ist der Name des Rechners des Benutzers. In MySQL können Sie den Zugang eines Benutzers von dem Rechner (IP-Adresse) abhängig machen, von dem aus dieser zugreift.
- ▶ `User`: der Benutzername, der benutzt wird, um Zugang zu MySQL zu erlangen.
- ▶ `password`: das Passwort des Benutzers.
- ▶ `Select_priv`: berechtigt den Benutzer, `SELECT`-Anweisungen auszuführen.
- ▶ `Insert_priv`: berechtigt den Benutzer, mittels `INSERT`-Anweisungen Daten in die Datenbank einzufügen.
- ▶ `Update_priv`: berechtigt den Benutzer, mittels `UPDATE`-Anweisungen Daten in der Datenbank zu editieren.
- ▶ `Delete_priv`: berechtigt den Benutzer, mittels `DELETE`-Anweisungen Daten aus der Datenbank zu löschen.
- ▶ `Create_priv`: berechtigt den Benutzer, Tabellen und Datenbanken auf dem MySQL-Server hinzuzufügen.
- ▶ `Drop_priv`: berechtigt den Benutzer, Tabellen und Datenbanken auf einem MySQL-Server zu löschen. Diese Berechtigung sollte nur mit Vorsicht vergeben werden, da hiermit großer Schaden angerichtet werden kann.
- ▶ `Reload_priv`: gestattet dem Benutzer, die `GRANT`-Tabellen mittels der `FLUSH`-Anweisung zu erneuern.
- ▶ `Shutdown_priv`: berechtigt den Benutzer, den MySQL-Server herunterzufahren.

- ▶ `Process_priv`: berechtigt den Benutzer, mittels `mysqladmin processlist`- bzw. `SHOW PROCESSLIST`-Anweisung Einsicht in die MySQL-Prozessliste zu nehmen. Zudem kann der Benutzer auch Prozesse beenden.
- ▶ `File_priv`: erlaubt dem Benutzer, Dateien auf dem MySQL-Server zu lesen und zu schreiben. Sie sollten bei der Vergabe dieser Berechtigung besonders vorsichtig sein. Denn falls diese nicht richtig angewendet wird, könnte beispielsweise jemand, der Ihnen schaden will, Systemdateien auf dem Server überschreiben. Dieses ist einer der Gründe, weshalb MySQL niemals als Root-Benutzer laufen sollte.
- ▶ `Grant_priv`: berechtigt den Benutzer, anderen Benutzern Berechtigungen zu erteilen. Dieses Privileg sollte aus ersichtlichem Grund nur den Administratoren gewährt werden.
- ▶ `References_priv`: wird bisher für nichts verwendet.
- ▶ `Index_priv`: berechtigt den Benutzer, Indizes in Tabellen zu erstellen und Indizes zu löschen. `Create_priv` und `Drop_priv` betreffen diese Berechtigung in keiner Weise. Auch wenn ein Benutzer schon `CREATE`- und `DROP`-Berechtigungen hat, muss er die `Index_priv`-Berechtigung haben, um Indizes zu erstellen bzw. zu löschen.
- ▶ `Alter_priv`: berechtigt den Benutzer, die Struktur einer Tabelle zu ändern. Diese Berechtigung umfasst nicht die Berechtigung, Indizes in Tabellen zu erstellen. Der Benutzer muss auch diese Berechtigungen haben, um Tabellen verändern zu können.

Wenn Sie einem Benutzer auf dieser Ebene Berechtigungen einräumen, so hat dieser globalen Zugang zu der Datenbank. Das bedeutet, dass ein Benutzer, dem zum Beispiel `DELETE`-Rechte in der `user`-Tabelle eingeräumt wurden, in allen Datenbanken auf dem MySQL-Server Datensätze löschen kann. Es gibt jedoch Situationen, in denen dies nicht immer sinnvoll ist. Stellen Sie sich vor, Sie sind der Administrator eines MySQL-Servers mit zwei Datenbanken: einer für die Buchhaltung und einer für die Angestellten. Die Buchhaltungs-Datenbank enthält alle Tabellen und Daten, die mit den Geschäftsvorgängen zusammenhängen, wie zum Beispiel Konten für Ein- und Ausgänge und die Gehaltsliste. Die Personal-Datenbank enthält alle Informationen über die Angestellten. In einer solchen Konstellation würden Sie den Benutzern der Buchhaltungs-Datenbank die Berechtigung geben, eigene Datensätze zu löschen, jedoch nicht die Berechtigung, auch Datensätze in der Personal-Datenbank zu löschen. Dennoch würden Sie den Benutzern der Buchhaltungs-Datenbank Rechte einräumen wollen, die Personal-Datenbank abzufragen und Datensätze anzuzeigen. Wenn Sie den Benutzern der Buchhaltungs-Datenbank `DELETE`-Berechtigungen in der `user`-Tabelle einräumen würden, hätten diese aber die Möglichkeit, auch Datensätze der Personal-Datenbank zu löschen. Wie können Sie dieses Problem nun umgehen? Lesen Sie weiter.

Die db-Tabelle

Die `db`-Tabellen enthält die Berechtigungen für alle Datenbanken auf Ihrem MySQL-Server. Berechtigungen, die an dieser Stelle vergeben werden, sind nur für eine bestimmte Datenbank gültig. Demnach würden Sie in dem vorhergehenden Beispiel dem Benutzer `DELETE`-Berechtigungen auf der Datenbank-Ebene und nicht auf Benutzer-Ebene erteilen.

Die `db`-Tabelle hat bis auf einige Ausnahmen fast die gleichen Spalten wie die `user`-Tabelle. Da diese Tabelle Berechtigungen auf der Datenbank-Ebene erteilt, gibt es dort keine Administrator-Privilegien wie `Reload_priv`, `Shutdown_priv`, `Process_priv` und `File_priv`. Diese Operationen stehen in keiner Beziehung zu den Anweisungen, die an eine Datenbank gerichtet werden können, und sind daher nur in der `user`-Tabelle zu finden. Die einzig neue Spalte in der `db`-Tabelle ist `Db`. Diese enthält den Namen der Datenbank, für welche Sie die entsprechenden Rechte vergeben.

Die host-Tabelle

Die `host`-Tabelle kontrolliert gemeinsam mit der `db`-Tabelle den Zugang zum MySQL-Server, indem sie die Anzahl der Hosts (Rechner), die auf den MySQL-Server zugreifen dürfen, beschränkt. Die Tabelle hat die gleichen Spalten wie die `db`-Tabelle.

Die `columns_priv`- und `tables_priv`-Tabellen

Die `columns_priv`- und `tables_priv`-Tabellen verwalten die Berechtigungen für Tabellen und Spalten einer Datenbank. In MySQL können Sie die Berechtigungen eines Benutzers bis auf die Spalten genau festlegen. Diese Tabellen enthalten folgende Spalten:

- ▶ `Host`: der Host (Rechner), von dem ein Benutzer auf die Datenbank zugreift.
- ▶ `Db`: die Datenbank, welche die Tabellen enthält, für die Sie Berechtigungen vergeben möchten.
- ▶ `User`: der Benutzername der Person, der Sie Berechtigung erteilen möchten.
- ▶ `Table_name`: der Tabellename in der Datenbank, zu welcher Sie Berechtigungen erteilen möchten. Achten Sie auf Groß- und Kleinschreibung!
- ▶ `Column_priv`: diese Spalte kontrolliert in beiden Tabellen den Zugang, den ein Benutzer hat. Sie kann folgende Werte enthalten: `SELECT`, `INSERT`, `UPDATE` und `REFERENCES`. Bei der Vergabe mehrerer Rechte werden diese durch Komma getrennt.
- ▶ `Timestamp`: diese Spalte enthält den Zeitstempel, anhand dessen man den Zeitpunkt der letzten Änderung verfolgen kann.

`Grantor` und `Table_priv` sind die einzigen beiden Spalten, die in der `tables_priv`-Tabelle und nicht in der `columns_priv`-Tabelle erscheinen. Die `Grantor`-Spalte enthält den Namen der Person, welche die Berechtigungen vergibt. Die `Table_priv`-Spalte beinhaltet die Berechtigungen für die betreffende Tabelle. Sie kann folgende Werte enthalten: `SELECT`, `INSERT`, `UPDATE`, `CREATE`, `DROP`, `GRANT`, `REFERENCES`, `INDEX` und `ALTER`.

Die `columns_priv`-Tabelle enthält nur eine Spalte – die `Column_name`-Spalte – die nicht in beiden Tabellen vorkommt. Diese Spalte enthält den Namen der Spalte, die durch die Berechtigungen in der `Column_priv`-Spalte betroffen ist.

Sie mögen sich fragen, wie das funktionieren soll. Es funktioniert ähnlich wie bei den `db`- und `user`-Tabellen. Wenn Sie einem Benutzer `SELECT`-Rechte für alle Spalten einer Tabelle einräumen möchten, können Sie dies in der `tables_priv`-Tabelle tun. Möchten Sie jedoch, dass ein Benutzer bestimmte Rechte nicht erhält, so tun Sie dies auf der Spalten-Ebene.

Einrichtung von Sicherheitskontrollen

Werfen Sie nun einen Blick auf das Ganze, um ein besseres Verständnis zu erlangen. In diesem Abschnitt wird näher auf das vorhergehende Beispiel der Buchhaltungs-Datenbank und Personal-Datenbank eingegangen. Das Szenario: Sie sind der Datenbank-Administrator für alle Datenbanken in Ihrer Firma. In jeder Abteilung der Firma haben Sie einen Superuser ernannt. Diese Menschen helfen Ihnen bei den täglichen administrativen Aufgaben. Daneben hat jede Abteilung natürlich eine Reihe von Angestellten mit verschiedenen Zugangsanforderungen an die einzelnen Datenbanken. Um alles so sicher wie möglich zu halten, erteilen Sie den Angestellten nur die Rechte, die sie für Ihre Arbeit benötigen.

Bill ist der Superuser im Bereich Buchhaltung und Julie der Superuser im Bereich Personalwesen. Um der Firmenpolitik treu zu bleiben, geben Sie Bill und Julie nur die administrativen Rechte für ihre jeweiligen Datenbanken. Demzufolge haben sie Einträge in der `user`-Tabelle, die ihren Status widerspiegeln. Da sie die Berechtigungen auf die jeweilige Datenbank beschränken möchten, werden sie den beiden keine Rechte in dieser Tabelle einräumen. Hier werden nur der `Host`, `Benutzername` und das `Passwort` eingetragen. Die Berechtigungen tragen Sie alle in die `db`-Tabelle ein. Auf diese Weise können Sie ihren Zugang beschränken. Auf der anderen Seite sind Sie der Datenbank-Administrator und benötigen globalen Zugang. Sie sollten die einzige Person in der `user`-Tabelle sein, die sämtliche Privilegien enthält. Durch die Beschränkung der Rechtevergabe erhalten Sie eine sicherere Server-Umgebung.

Da Sie die Rechte von Bill und Julie beschränkt haben, sind Sie der einzige, der neue Benutzer anlegen kann und diese Rechte vergeben kann. Julie hat einen neuen Angestellten in der Personal-Datenbank. Dieser soll zwar alle Datensätze der Tabelle `Lohn_Gehalt` abrufen können, jedoch nur einige Spalten dieser Tabelle editieren dür-

fen. Dafür geben Sie der neuen Angestellten, Sheila, einen Eintrag in der `user`-Tabelle, damit sie mit der Datenbank überhaupt Verbindung aufnehmen kann. Schließlich vergeben Sie `SELECT`-Rechte auf alle Tabellen der Personal-Datenbank. Um Sheilas `UPDATE`-Rechte zu beschränken, müssen Sie diese Rechte in der `tables_priv`- und `column_priv`-Tabelle festlegen.

Kontrollebenen

Sie werden in Kürze lernen, wie man Benutzer und deren Rechte einrichtet. Zu diesem Zeitpunkt reicht es, wenn Sie wissen, dass MySQL über ein hierarchisches Sicherheitskonzept verfügt. Wenn ein Benutzer mit einer MySQL-Datenbank Verbindung aufnehmen will, überprüft MySQL zunächst, ob ein passender Eintrag für `Host`, `Benutzername` und `Passwort` in der `user`-Tabelle vorhanden ist. Ist dies der Fall, erhält der Benutzer Zugang zum System. Wenn der Benutzer eine Anfrage an eine Datenbank stellt, überprüft MySQL zunächst wiederum in der `user`-Tabelle, ob der Benutzer die erforderlichen Rechte besitzt. Hat der Benutzer in dieser Tabelle keine Rechte, überprüft MySQL die `db`-Tabelle. Wiederum sucht MySQL nach einer Übereinstimmung von `Host`, `Benutzername` und `Passwort`. Findet es eine Übereinstimmung, überprüft es die Rechte des Benutzers. Sind hier die erforderlichen Rechte nicht eingetragen, um die Anfrage auszuführen, sucht MySQL in den `tables_priv`- und `columns_priv`-Tabellen nach den entsprechenden Rechten, um die Anfrage auszuführen. Kann MySQL keine Berechtigungen finden, gibt es eine Fehlermeldung aus. All das passiert jedes Mal, wenn eine Anfrage an MySQL geschickt wird.

Wie Sie sehen, gibt es zwei Kontrollpunkte in MySQL. Der eine Kontrollpunkt liegt bei der Verifikation der Verbindung, der andere bei der Verifikation der Anfrage. Diese Kontrollpunkte bieten eine sicherere Umgebung für Ihre Datenbank. Ein Benutzer, der mit der Datenbank Verbindung aufnehmen kann, ist noch lange nicht dazu berechtigt, alle Operationen auszuführen. Das sind recht gute Sicherheitsmaßnahmen gegen Möchtegern-Hacker, die Ihren Geschäftsbetrieb stören wollen, und nicht versierte Benutzer, die ungewollt Schaden anrichten könnten.

Verbindungsverifikation

Die Verifikation der Verbindung erfolgt in dem Moment, in dem Sie versuchen, sich mit der Datenbank zu verbinden. Jede Verbindung zu MySQL erfordert einen `Benutzernamen`, `Hostnamen` und ein `Passwort`. Der `Benutzername` ist der Name der Person, die versucht, eine Verbindung zum MySQL-Server aufzunehmen. Das `Passwort` ist eine zusätzliche Sicherheitsmaßnahme. Damit soll sichergestellt werden, dass die Person auch wirklich der Benutzer ist, für den sie sich ausgibt. Der `Hostname` ist der Name des Rechners, von dem aus der Benutzer versucht, eine Verbindung aufzubauen. My-

SQL kann nicht nur einem Benutzer die Verbindungsaufnahme verweigern, sondern auch einem bestimmtem Rechner. Sie können Verbindungen von bestimmten Domains erlauben bzw. verbieten – ganz wie es Ihnen beliebt. Damit machen Sie Ihre Datenbank zu einem sehr sicheren Platz für Ihre Daten.

Der Verifikationsprozess der Verbindung ist ziemlich einfach. MySQL überprüft die eingehende Anfrage mit den in der `user`-Tabelle vorhandenen Informationen zu Benutzername, Passwort und Hostname. Wenn MySQL eine Übereinstimmung findet, wird dem Benutzer die Verbindungsaufnahme gestattet. Findet MySQL hingegen keine Übereinstimmung in der `user`-Tabelle, verbietet es die Verbindungsaufnahme.

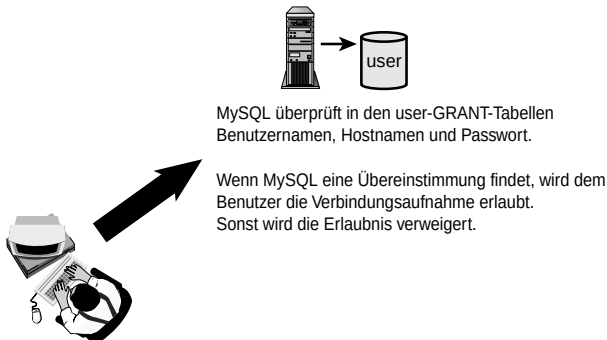


*Abbildung 17.1:
Der Verifikations-
prozess beim Aufbau
einer Verbindung mit
MySQL*

Benutzer versucht, eine Verbindung zu MySQL herzustellen

Anfrageverifikation

Diese Verifikation erfolgt jedes Mal, wenn ein Benutzer eine Anfrage an die Datenbank stellt. Jede Anweisung, die nach der Verbindungsaufnahme erfolgt, durchläuft den gleichen Verifikationsprozess. Dieses Vorgehen stellt sicher, dass die Benutzer ständig in dem Rahmen arbeiten, der Ihnen durch die Rechtevergabe gesteckt wurde. Zudem ist es ein zusätzlicher Sicherheitsmechanismus, der böswillige Personen davon abhält, Ihre Daten zu stehlen, auch wenn es diesen gelungen ist, Verbindung mit der Datenbank aufzunehmen.

Der Ablauf der Verifikation ist wiederum ziemlich einfach. Immer wenn eine Anfrage an die Datenbank gestellt wird, überprüft MySQL die Berechtigungen des Benutzers in der `user`-Tabelle. Hat der Benutzer auf dieser Ebene Berechtigungen, darf er alles in jeder Datenbank im MySQL RDBMS tun, was ihm beliebt. Daher sollten Berechtigungen auf dieser Ebene eher spärlich vergeben werden. Findet MySQL keine entsprechenden Berechtigungen in der `user`-Tabelle, sucht es diese in der `db`-Tabelle.

Die `db`-Tabelle ist die nächste Ebene im MySQL-Sicherheitskonzept. Berechtigungen, die hier vergeben werden, sind nur für eine bestimmte Datenbank gültig. `SELECT`-Rechte, die auf dieser Ebene vergeben wurden, erlauben dem Benutzer, sämtliche Daten al-

ler Tabellen einer Datenbank einzusehen. Dies ist im Allgemeinen in Ordnung. Möchten Sie die Berechtigungen jedoch noch weiter einschränken, so können Sie dies in der `tables_priv`-Tabelle tun.

Die `tables_priv`-Tabelle ist die nächste Tabelle, in der MySQL bei einer Anfrage nachsieht, ob der Benutzer die nötigen Rechte besitzt. Werden die benötigten Rechte hier gefunden, führt MySQL die Anfrage aus. Werden sie nicht gefunden, hat MySQL noch einen weiteren Ort, an dem es diese überprüfen kann: die `columns_priv`-Tabelle.

Die `columns_priv`-Tabelle ist der letzte Platz, in dem MySQL nachsieht und überprüft, ob ein Benutzer die benötigten Rechte für eine Aktion hat oder nicht. Sind die Berechtigungen auch in dieser Tabelle nicht vorhanden, gibt MySQL eine Fehlermeldung zurück, dass die Anfrage abgelehnt wurde. Das alles geschieht so schnell, dass die Performance kaum beeinträchtigt wird und Ihnen somit ein hoher Sicherheitslevel ohne Performanceverlust ermöglicht wird. Die folgende Abbildung illustriert den Verifikationsprozess:

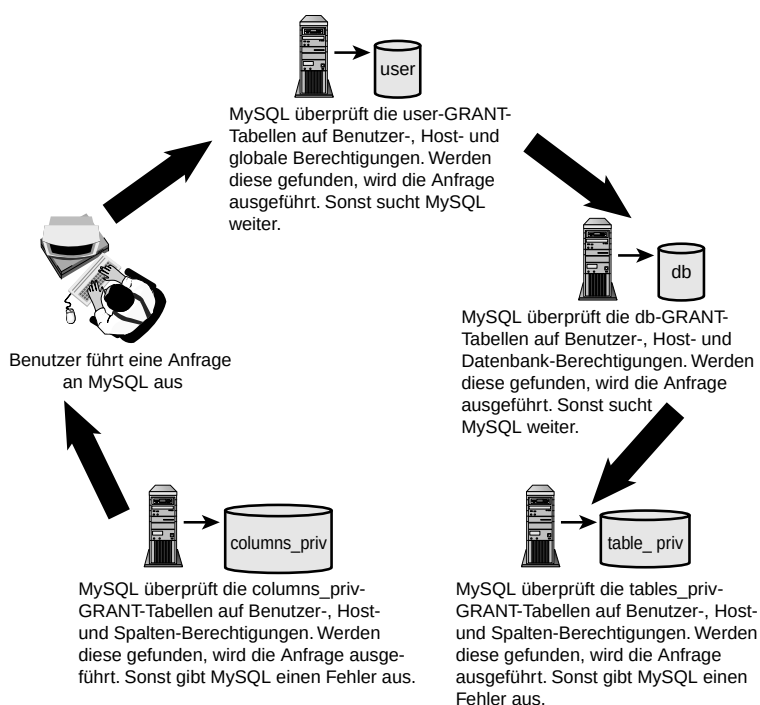


Abbildung 17.2:
Der Verifikations-
prozess bei einer
Anfrage an
MySQL

Vergabe der Benutzerrechte

Sie werden sich fragen, ob Sie sich Gedanken machen müssen über die Art und Weise, wie MySQL Sicherheit implementiert. Die einfache Antwort lautet: Nein. MySQL verfügt über Werkzeuge, die Ihnen diese Aufgabe abnehmen. Aber Sie würden nie ein guter DBA (Datenbank-Administrator) sein, wenn Sie nicht wüssten, wie MySQL Sicherheit implementiert. Das wird Ihnen auch weiterhelfen, Sicherheitslöcher aufzuspüren, falls Sie jemals welche haben sollten.

Es gibt verschiedene Möglichkeiten, um Ihre Datenbank zu schützen. MySQL versorgt Sie mit einer großen Anzahl an Werkzeugen, die Sie bei der Sicherheitskontrolle Ihrer Datenbank unterstützen. Der folgende Abschnitt beginnt mit dem schwierigsten Weg und arbeitet sich in umgekehrter Reihenfolge vorwärts.

Änderungen an der GRANT-Tabelle

Die wohl schwierigste Methode besteht darin, die *GRANT*-Tabellen selbst per Hand zu bearbeiten und zu ändern. Indem Sie zunächst ein Verständnis von Aufbau und Funktionsweise der *GRANT*-Tabellen erhalten, werden Sie die später vorgestellten Methoden hierzu um so mehr zu schätzen wissen. Von Zeit zu Zeit kann es auch einmal nötig sein, die *GRANT*-Tabellen direkt per Hand zu editieren. Dieser Abschnitt wird Ihnen zeigen, wie Sie vorgehen müssen. Da es sich bei diesen um gewöhnliche MySQL-Tabellen handelt, arbeiten die Anweisungen *INSERT*, *DELETE*, *UPDATE* und *SELECT* auch so, wie Sie es gewohnt sind. Starten Sie, indem Sie einen Benutzer hinzufügen.

Um einen neuen Benutzer anzulegen, müssen Sie sich zunächst mit dem MySQL-Server verbinden und die *mysql*-Datenbank benutzen. Hierzu führen Sie folgende Schritte aus:

- ▶ Wechseln Sie in das *MySQL*-Installationsverzeichnis: `cd /usr/local/mysql` (Linux) bzw. `cd c:\mysql\bin` (Windows)
- ▶ Stellen Sie sicher, dass der *mysqld*-Daemon läuft: `bin/mysqladmin -p ping` (Linux) bzw. `mysqladmin -p ping` (Windows)
- ▶ Sie werden nach dem *Root*-Passwort gefragt. Nachdem Sie dieses korrekt eingegeben haben, erhalten Sie bei Erfolg die Meldung »MySQL is alive«. Starten Sie nun den *MySQL*-Monitor mit `mysql -u root -p Passwort -h localhost`. Geben Sie für Passwort Ihr *Root*-Passwort an. Sie können nun auf die *mysql*-Datenbank mit `use mysql`; zugreifen.

Die *mysql*-Datenbank sollte nun die aktive Datenbank sein. Daraufhin fügen Sie eine neue Zeile in die *user*-Tabelle ein. Erinnern Sie sich, das es sich hierbei um die Tabelle

handelt, in der MySQL nachsieht, ob ein Anwender die richtigen Zugangsberechtigungen hat. Um einen Benutzer ohne Rechte einzufügen, geben Sie Folgendes ein:

```
mysql> INSERT INTO user (Host, User, password)
-> VALUES
-> ("localhost","Scott",PASSWORD("Dinosaurier"));
Query OK, 1 row affected (0.06 sec)
```

Diese Anweisung legt einen neuen Benutzer in der `user`-Tabelle an. Der Benutzer ist Scott, der nur von dem Datenbankserver (`localhost`) aus Verbindung zur Datenbank aufnehmen kann. Scott hat nicht die Möglichkeit, von außerhalb auf den MySQL-Server zuzugreifen. Die Anweisung vergibt dem Benutzer Scott auch das Passwort »Dinosaurier«. In MySQL müssen Sie bei Passwörtern auf die Groß- und Kleinschreibung achten. Die `PASSWORD()`-Funktion verschlüsselt das Passwort und speichert es so verschlüsselt in der Tabelle ab. Dieses verhindert, dass neugierige Augen alle Passwörter aus der Datenbank ausspähen können. Einmal mit dem MySQL-Server verbunden, kann Scott allerdings keine weiteren Aktionen mehr vornehmen, da ihm bis jetzt weder globale noch lokale Rechte eingeräumt wurden.

Fernzugriff auf eine Datenbank

Wenn Sie Scott erlauben wollen, von einem fremden Rechner aus auf den MySQL-Server zuzugreifen, müssen Sie den Hostnamen in der `user`-Tabelle ändern, um diesem Umstand Rechnung zu tragen. Hostnamen können entweder IP-Adressen oder Domainnamen sein. Ein Hostname könnte zum Beispiel `10.1.1.50` oder `staff.jsmc.org` sein. Dieses würde Scott ermöglichen, von einem entfernten Rechner in der `staff.jsmc.org`-Domain bzw. der IP-Adresse `10.1.1.50` aus auf den MySQL-Server zuzugreifen. Soll Scott von jedem beliebigen Rechner aus auf den MySQL-Server zugreifen können, können Sie dem Rechnung tragen, indem Sie das Wildcard-Zeichen (Platzhalter) `%` verwenden. Sie könnten die Spalte für Hostname auch leer lassen, was zum gleichen Resultat wie das Verwenden einer Wildcard führen würde. Eine weitere Option, die sich Ihnen bietet, ist, dass Sie Scott den Zugriff von überall her aus der `jsmc.org`-Domain erlauben. Um das zu tun, benutzen Sie die Wildcard auf folgende Weise:

```
mysql> INSERT INTO user (Host, User, password)
-> VALUES
-> ("%jsmc.org","Scott",PASSWORD("Dinosaurier"));
Query OK, 1 row affected (0.06 sec)
```

Nun ist Scott berechtigt, von überall aus der `jsmc.org`-Domain auf den MySQL-Server zuzugreifen. Ein wichtiger Punkt sollte hier jedoch nicht vergessen werden. Auch wenn das Wildcard-Zeichen `%` für *jeden* Rechner steht, bedeutet es genau genommen: von *jedem entfernten* Rechner. Ein Benutzer, der sich auf dem lokalen Rech-

ner einloggt – also auf dem Rechner, auf dem der MySQL-Server läuft –, benötigt einen `localhost`-Eintrag. Wenn dieser Eintrag fehlt, kann der Benutzer sich nicht vom lokalen Rechner aus einloggen.

Vergabe von Berechtigungen

Nachdem Sie nun einen neuen Benutzer angelegt haben, wird es Zeit, diesem auch gewisse Benutzerrechte zu vergeben. Um Scott den Zugriff auf irgendeine Datenbank, die im MySQL-RDBMS vorhanden ist, zu gewähren, müssen Sie einen Eintrag in der `db`-Tabelle vornehmen. Erinnern Sie sich, das war die Tabelle, in der alle relevanten Rechte für eine individuelle Datenbank vergeben wurden. Wenn Sie möchten, dass Scott `SELECT`-Rechte in der `Meet_A_Geek`-Datenbank besitzt, machen Sie folgenden Eintrag:

```
mysql> INSERT INTO db (Host, User, Db, Select_priv)
-> VALUES
-> ("localhost", "Scott", "Meet_a_Geek", "Y");
Query OK, 1 row affected (2.14 sec)
```

Diese Anweisung berechtigt Scott nur vom lokalen Rechner aus, die `Meet_A_Geek`-Datenbank abzufragen (`SELECT`-Rechte). Er hätte keinen Zugriff auf die Datenbank, wenn er versuchen würde, von einem anderem Rechner oder einer anderen Domain aus auf diese zuzugreifen. Die gleichen Regeln über die Verwendung von Wildcards gelten auch hier. Dieses zeigt beispielhaft, wie MySQL ein Sicherheitssystem zur Verfügung stellt, das Ihnen kein anderes RDBMS auf dem Markt geben kann.

Wenn Scott Zugang zu mehreren Datenbanken auf dem MySQL-Server benötigen würde, müsste er mehr als einen Eintrag in dieser Tabelle haben, welche ihm explizit die gewünschten Rechte einräumen würden.

Löschen von Berechtigungen

Wenn Sie die Rechte eines Benutzers wieder beschneiden möchten, verwenden Sie die `UPDATE`-Anweisung. Möchten Sie Scott zum Beispiel die `SELECT`-Rechte wegnehmen, und ihm dafür `UPDATE`-Rechte geben, führen Sie folgende Anweisung aus:

```
mysql> UPDATE db SET SELECT_priv = "N", Update_priv = "Y"
-> WHERE User = "Scott"
-> AND Host = "localhost"
-> AND Db = "Meet_A_GEEK";
Query OK, 1 row affected (0.17 sec)
Rows matched: 1 Changed: 1 Warnings: 0
```

Diese Anweisung nimmt Scott seine `SELECT`-Rechte und gibt ihm dafür `UPDATE`-Rechte. Jetzt, wo Sie Scott Rechte für die Datenbank vergeben haben, sind Sie fertig – richtig?

Nein, nicht wirklich. Wenn der `mysqld`-Daemon gestartet wird, lädt MySQL die `GRANT`-Tabellen in den Speicher, um die Performance zu beschleunigen. Das Problem besteht nun darin, dass Änderungen, die an den `GRANT`-Tabellen vorgenommen werden, nicht im Speicher aktualisiert werden. Demnach wird ein Benutzer, den Sie gerade erst eingerichtet haben, so lange nicht auf die Datenbank zugreifen können, bis die `GRANT`-Tabellen im Speicher aktualisiert wurden. Um diese Tabellen zu erneuern, müssen Sie diese »flushen«. Dieses geschieht mit der `FLUSH`-Anweisung. Um diese Anweisung ausführen zu können, benötigen Sie selbstverständlich die `FLUSH`- bzw. `RELOAD`-Rechte. Eine andere Möglichkeit, die `GRANT`-Tabellen zu »flushen« (neu zu laden), besteht darin, das Werkzeug `mysqladmin` mit der `RELOAD`-Option zu verwenden. Das alles dauert nur einen kurzen Augenblick, und anschließend sind alle Änderungen, die Sie vorgenommen haben, aktualisiert.

Die GRANT-Anweisung

Wenn Sie den direkten Weg der Rechtevergabe in den `GRANT`-Tabellen scheuen, gibt es für Sie eine weitere Möglichkeit, Benutzerrechte zu verwalten: Die `GRANT`-Anweisung, welche die folgende Syntax hat:

```
GRANT [privileges] (Spaltennamen)
ON Datenbankname.Tabellenname
TO Benutzername@Hostname
IDENTIFIED BY "password"
[WITH GRANT OPTIONS]
```

Das wirkt auf den ersten Blick vielleicht etwas kompliziert, ist aber in Wirklichkeit ziemlich einfach. Nach der Anweisung `GRANT` führen Sie alle Rechte auf, die Sie dem neuen Benutzer erteilen möchten. Im Folgenden finden Sie eine Auflistung aller Rechte, die Sie einem Benutzer geben können:

- ▶ `ALL`: erteilt dem Benutzer alle verfügbaren Rechte
- ▶ `ALTER`: erlaubt dem Benutzer, die Struktur von Tabellen und Spalten sowie Indizes zu ändern.
- ▶ `CREATE`: gestattet dem Benutzer, Tabellen und Datenbanken zu erstellen.
- ▶ `DELETE`: erlaubt dem Benutzer, Datensätze von Tabellen zu löschen.
- ▶ `DROP`: erlaubt dem Benutzer, Tabellen und Datenbanken zu löschen.
- ▶ `FILE`: erlaubt dem Benutzer, Dateien auf dem Server zu schreiben und zu lesen.
- ▶ `INDEX`: befähigt den Benutzer, Indizes zu erstellen bzw. zu löschen.
- ▶ `INSERT`: erlaubt dem Benutzer, Datensätze zu einer Datenbank hinzuzufügen.

- ▶ PROCESS: gestattet dem Benutzer, MySQL-System-Prozesse einzusehen und zu beenden.
- ▶ REFERENCES: im Moment ohne Bedeutung.
- ▶ RELOAD: erlaubt dem Benutzer, die *FLUSH*-Anweisung auszuführen.
- ▶ SELECT: befähigt den Benutzer, SELECT-Abfragen auszuführen.
- ▶ SHUTDOWN: erlaubt dem Benutzer, den MySQL-Server herunterzufahren.
- ▶ UPDATE: erlaubt dem Benutzer, existierende Datensätze in der Datenbank zu editieren.
- ▶ USAGE: erlaubt dem Benutzer, mit dem MySQL-Server Verbindung aufzunehmen. Ein auf diese Weise angelegter Benutzer hat weiter keine Rechte.

Der nächste Teil der Anweisung ist optional. Er ermöglicht Ihnen, die Spalten, für die Sie Berechtigungen vergeben möchten, anzugeben. Achten Sie bei der Eingabe der Spaltennamen auf Groß- und Kleinschreibung, da MySQL zwischen Groß- und Kleinschreibung unterscheidet.

Nach dem Schlüsselwort *ON* führen Sie die Datenbank und die Tabelle(n) auf, für welche Sie die gewünschten Rechte vergeben möchten. Um alle Tabellen einer Datenbank anzugeben, können Sie das Sternchen (Asterisk) benutzen:

```
Meet_A_Geek.*
```

Wenn Sie die Rechte global vergeben möchten, verwenden Sie zwei Sternchen:

```
*,*
```

Der nun folgende Teil der *GRANT*-Anweisung ist die *TO*-Klausel. Hier geben Sie den Benutzernamen und Hostnamen des neuen Benutzers an. An dieser Stelle können Sie wiederum Wildcards verwenden. Wenn Sie Scott Rechte vergeben möchten, geben Sie Folgendes ein:

```
"Scott@localhost"  
"Scott@%jsmc.org"  
"Scott@"
```

Der nächste Teil der Anweisung ist das Passwort. Sie brauchen hier nicht die *PASSWORD()*-Funktion zu verwenden, MySQL macht das automatisch für Sie.

Der letzte Teil der Anweisung ist komplett optional. Es gibt dem neuen Benutzer die Möglichkeit, auch *GRANT*-Rechte zu erhalten.

Betrachten Sie nun einige Beispiele. Kommen wir wieder zurück auf unseren Freund Scott. Zunächst geben Sie ihm die Erlaubnis, Verbindung mit dem MySQL-Server aufzunehmen, ohne ihm irgendwelche weiteren Rechte einzuräumen. Dazu benutzen Sie folgende Anweisung:


```
mysql> GRANT USAGE ON *.*  
-> TO "Scott@%"  
-> IDENTIFIED BY "Dinosaurier";  
Query OK, 0 rows affected (0.22 sec)
```

Diese Anweisung gestattet Scott, von jedem Rechner aus auf den MySQL-Server mit dem Passwort »Dinosaurier« zuzugreifen. Wenn Sie nun die GRANT-Tabellen betrachten, werden Sie feststellen, dass Scott nur einen Eintrag in der user-Tabelle hat. Sie sehen zudem, dass das Passwort verschlüsselt ist. Das war doch ganz einfach.

Vergeben Sie nun SELECT-Rechte an Scott für alle Tabellen der Meet_A_Geek-Datenbank:

```
mysql> GRANT SELECT  
-> ON Meet_A_Geek.*  
-> TO "Scott@%"  
-> IDENTIFIED BY "Dinosaurier";  
Query OK, 0 rows affected (0.16 sec)
```

Diese Anweisung gestattet Scott, von jedem Rechner aus auf die MySQL-Datenbank Meet_A_Geek zuzugreifen und dort auf alle Tabellen SELECT-Abfragen durchzuführen. Beschränken Sie nun Scotts SELECT-Rechte auf die Customers-Tabelle in der Meet_A_Geek-Datenbank. Dazu geben Sie folgende Anweisung ein:

```
mysql> GRANT SELECT  
-> ON Meet_A_Geek.Customers  
-> TO "Scott@%"  
-> IDENTIFIED BY "Dinosaurier";
```

Beachten Sie, wie die Punktnotation (.) benutzt wird. Der erste Teil besteht aus dem Namen der Datenbank, der zweite Teil aus dem Namen der Tabelle. Das ist Standard – auch für andere Datenbanksysteme.

Geben Sie Scott nun UPDATE-Rechte für die First_Name- und Last_Name-Spalten der Customers-Tabelle. Führen Sie folgende Anweisung aus:

```
mysql> GRANT UPDATE (First_Name, Last_Name)  
-> ON Meet_A_Geek.Customers  
-> TO "Scott@localhost"  
-> IDENTIFIED BY "Dinosaurier";
```

Diese Anweisung beschränkt Scotts Berechtigungen darauf, die Spalten First_Name und Last_Name der Customers-Tabelle in der Meet_A_Geek-Datenbank zu editieren. Wie Sie sehen, ist diese Anweisung eine einfache Möglichkeit, ein Benutzerkonto zu verwalten.

Es gibt noch eine weitere Möglichkeit, um in MySQL Berechtigungen an Benutzer zu vergeben. Diese Befehle führen Sie weg vom MySQL-Monitor und in das mysql-Verzeichnis. In dem Unterverzeichnis bin (in dem sich alle MySQL-Werkzeuge befinden) findet

sich ein Programm namens `mysql_setpermissions`. Dabei handelt es sich um ein Perl-Script, das Sie durch den Vergabevorgang von Rechten in Ihrer Datenbank führt. Um dieses Werkzeug zu verwenden, wechseln Sie in das MySQL-Installationsverzeichnis. Der `mysqld`-Daemon muss laufen. Zunächst starten Sie das Werkzeug wie folgt:

```
/usr/local/mysql> bin/mysql_setpermission
```

Sie sollten diese Ausgabe erhalten:

```
>Passwort for user to connect to MySQL:
```

Nachdem Sie das Root-Passwort erfolgreich eingegeben haben, sollten Sie den »Willkommen-Bildschirm« mit folgenden Menüpunkten sehen können:

1. Set password for user.
 2. Add a database + user privilege for that database -
User can do all except all admin functions.
 3. Add user privilege for an existing database -
User can do all except admin functions.
 4. Add user privilege for an existing database -
User can do all except admin functions + no create/drop.
 5. Add user privilege for an existing database -
User can do only selects (no update, delete, insert, and so on).
- Exit this programm.

Wie Sie sehen, ist dieses Werkzeug in seinen Möglichkeiten sehr beschränkt. Doch wenn Sie etwas brauchen, was schnell und einfach zu bedienen ist, dann können Sie dieses kleine Programm benutzen. Zurück zu Scott. Fügen Sie ihn als neuen Benutzer in eine existierende Datenbank ein, indem Sie Option 3 verwenden.

Wenn Sie Option 3 ausgewählt haben, wird Ihnen eine Liste von vorhandenen Datenbanken angezeigt, aus denen Sie auswählen können. Wählen Sie die `Meet_A_Geek`-Datenbank aus. Das Programm fragt Sie nun nach einem Benutzernamen. Geben Sie `Scott` ein. Anschließend fragt Sie das Programm, ob Sie ein Passwort für Scott vergeben möchten. Um Ihre Datenbank zu schützen, sollten Benutzer immer ein Passwort haben. Geben Sie daher `Y` ein. Nun fragt Sie das Programm nach einem Passwort. Geben Sie das Wort `Dinosaurier` oder was immer Sie wollen, ein. Nachdem Sie die Eingabetaste gedrückt haben, werden Sie ein zweites Mal aufgefordert, das Passwort einzugeben. Haben Sie das getan, werden Sie nach einem Hostnamen gefragt. Sie können einen Hostnamen, einen Platzhalter oder eine Kombination aus beidem angeben. Geben Sie `%` ein. Das Programm wird Sie nun fragen, ob Sie einen weiteren Hostnamen unter diesem Benutzer anlegen möchten. Fürs erste geben Sie `NO` ein. Das Programm zeigt Ihnen nun Ihre Angaben und fragt Sie nochmals, ob alles richtig ist. Ist dies der Fall, geben Sie `Y` ein. Sie haben damit für die `Meet_A_Geek`-Datenbank einen neuen Benutzer namens `Scott` mit allen Benutzerrechten – außer den Administrator-Rechten – angelegt, der von jedem Rechner aus Verbindung mit der Datenbank auf-

nehmen kann. Dieses kleine Programm kann Ihnen eine Menge Zeit sparen, wenn Sie einmal auf die Schnelle einen neuen Benutzer anlegen müssen.

Jetzt wissen Sie, wie man neue Benutzer anlegt. Aber wie werden Sie diese wieder los?

Löschen von Benutzern und Benutzerrechten

Sie haben im vorherigen Abschnitt ausführlich Bekanntschaft mit dem Hinzufügen von Benutzern und Benutzerrechten gemacht. Nun ist es an der Zeit, Ihnen zu zeigen, wie man diese wieder entfernt. MySQL bietet Ihnen hier nur eine Möglichkeit, um einen Benutzer zu löschen: Sie müssen die GRANT-Tabellen per Hand editieren und gegen jede Tabelle der mysql-Datenbank eine DELETE-Anweisung ausführen. Um dies sicher und richtig zu machen, müssen Sie den zu löschenden Benutzer und Hostnamen in einer WHERE-Bedingung genau spezifizieren. Folgende Anweisungen würden Sie verwenden, um Scott aus den GRANT-Tabellen zu entfernen:

```
DELETE FROM user WHERE User = "Scott" AND Host = "%";  
DELETE FROM db WHERE User = "Scott" AND Host = "%";
```

Diese Anweisungen würden Scotts Eintrag in der user- und db-Tabelle löschen, in denen der Hostname der Platzhalter % ist. Es könnten jedoch noch andere Einträge von Scott in den Tabellen vorhanden sein, daher müssen Sie diese manuell säubern. Um zu überprüfen, ob Scott noch irgendwelche weiteren Einträge in den Tabellen hat, könnten Sie folgende Anweisung ausführen:

```
SELECT FROM user WHERE user = "Scott";  
SELECT FROM db WHERE user = "Scott";
```

Sie können Benutzerrechte auf zwei Arten widerrufen. Die eine Möglichkeit besteht darin, dass Sie verschiedene SQL-Anweisungen gegen die Tabellen ausführen. Zum anderen können Sie die REVOKE-Anweisung benutzen, die sehr zu empfehlen ist, wenn es darum geht, jemandem Rechte wieder zu entziehen, weil sie wenig fehleranfällig ist.

Die Syntax von REVOKE ist der GRANT-Anweisung ziemlich ähnlich:

```
REVOKE [privileges] (Spalten)  
ON Datenbankname.Tabellenname FROM Benutzername@Hostname
```

Die Rechte, die Sie einem Benutzer entziehen können, sind die gleichen Rechte, die wir schon bei der GRANT-Anweisung vorgestellt haben. Mit der REVOKE-Anweisung können Sie entweder einen Teil oder alle Rechte eines Benutzers löschen. Alles andere in der REVOKE-Anweisung muss mit der korrespondierenden GRANT-Anweisung übereinstimmen. Wenn Sie nun die REVOKE-Anweisung genauer betrachten, sollte Ihnen alles ziemlich bekannt vorkommen:

Nach `REVOKE` führen Sie alle Rechte eines Benutzers auf, die diesem entzogen werden sollen. Anschließend können Sie optional Spaltennamen angeben, zu denen Rechte erteilt wurden. Danach erscheint das Schlüsselwort `ON`, dem der Name der Datenbank und der Name der Tabelle folgt, für die Sie die Rechte entziehen. Sie können auch alle Datenbanken und alle Tabellen mit der `.*`-Syntax angeben. Anschließend werden nach dem Schlüsselwort `FROM` Benutzer- und Hostname angegeben. Schauen Sie sich nun einige Beispiele an:

Um Scotts Rechte komplett zu löschen, können Sie folgende Anweisung benutzen:

```
mysql> REVOKE ALL ON *.* FROM "Scott@";
```

Damit entziehen Sie Scott sämtliche Benutzerrechte, die er in allen Datenbanken mit dem Platzhalter `%` als Hostnamen hatte. Wurde Scott jemals die Berechtigung erteilt, vom lokalen Server aus auf MySQL zuzugreifen, müssen Sie folgende Anweisung ausführen, um dies rückgängig zu machen:

```
mysql> REVOKE ALL ON *.* FROM "Scott@localhost";
```

Um spezifische Rechte zu entziehen, benutzen Sie `REVOKE` ähnlich der `GRANT`-Anweisung. Entziehen Sie Scott beispielsweise die `UPDATE`-Rechte für die Spalten `Last_Name` und `First_Name` der `Customers`-Tabelle:

```
mysql> REVOKE UPDATE (First_Name, Last_Name)
-> ON Meet_A_Geek.Customers
-> FROM "Scott@%.jsmc.org";
```

Wie Sie sehen, ist die `REVOKE`-Anweisung der `GRANT`-Anweisung in Syntax und Verwendung sehr ähnlich. Der Hauptunterschied liegt darin, dass Sie mit der einen Anweisung Benutzerrechte zuteilen, während Sie diese mit der anderen Anweisung wieder entziehen.

Wie Sie geeignete Sicherheitsmaßnahmen ergreifen

Als Administrator einer MySQL-Datenbank sind sie verantwortlich für die Sicherheit und Integrität der Daten – das heißt Sie müssen für eine einwandfreie Datenbank sorgen. Die Informationen in der Datenbank sind einzig und allein in Ihrer Verantwortung. Alles liegt in Ihren Händen. Wenn die Daten also beschädigt werden, weil ein unzufriedener Angestellter in Ihre Datenbank einbricht und Datenzeile für Datenzeile löscht, wird jeder auf Sie verweisen. Vielleicht sind die Daten in Ihren Händen sogar hoch klassifiziert oder streng vertraulich. Es liegt in Ihrer Verantwortung, dass es auch so bleibt. Das ist eine große Verantwortung, aber wenn Sie MySQL benutzen, lehnen Sie sich beruhigt zurück, da Sie den Schritt zu einem sehr sicheren Datenbanksystem getan haben.

Es gibt ein paar zusätzliche Schritte, die Sie unternehmen können, um die Umgebung des Systems noch sicherer zu machen. Einer der ersten Schritte besteht darin, `mysqld` niemals als Root des Betriebssystems laufen zu lassen. Der Hauptgrund hierfür ist, dass jeder, der `FILE`-Privilegien in MySQL besitzt, Zugriff auf alle Dateien auf dem Server hat, wenn MySQL als Root läuft. Statt MySQL als Root zu benutzen, sollten Sie es als `mysqladmin`- oder `mysqlgroup`-Benutzer laufen lassen. Auf diese Weise stellen Sie sicher, dass Benutzer, die Schaden zufügen möchten, mit `FILE`-Privilegien keine Systemdateien überschreiben können.

Zudem können Sie das `data`-Unterverzeichnis im `mysql`-Verzeichnis sichern. Alle Dateien in diesem Verzeichnis sollten dem Benutzer gehören, unter dem auch der `mysqld`-Daemon läuft. Sie sollten auch nur für diesen Benutzer schreib- und lesbar sein. Das stellt sicher, dass kein anderer Benutzer auf diese Dateien Zugriff erhält.

Sie können außerdem, wenn Sie Hostnamen definieren, die Verwendung von Platzhaltern (%) beschränken. Sie sollten, wenn möglich, immer IP-Adressen verwenden. Dadurch können Sie mithelfen, das System sicherer zu machen.

Wenn es um Sicherheit geht, ist es immer am sinnvollsten, das System am Anfang so streng wie möglich zu konfigurieren und dieses erst im weiteren Verlauf zu lockern. Wenn Sie neuen Benutzern Rechte erteilen, dann nur so viele, wie sie benötigen, um Ihre Arbeit zu erledigen. Mit einer strengen Sicherheitspolitik helfen Sie, Ihre Datenbank vor feindlichen Attacken zu bewahren.

Zusammenfassung

Heute haben Sie eine Menge über Sicherheit gelernt. Sie haben erfahren, wie MySQL Sicherheit in einer hierarchischen Weise implementiert und dass es mehrere Ebenen der Kontrolle sowie zwei Verifikationspunkte – die Zugangs- und Anfragenverifikation – gibt. Sie konnten auch sehen, wie diese zweiseitige Sicherung Ihnen helfen kann, Sicherheitslöcher zu vermeiden.

Sie haben zudem gelernt, wie man Benutzer zu einer MySQL-Datenbank hinzufügt. Sie haben gesehen, dass MySQL hierfür mehrere Varianten zur Verfügung stellt. Sie können Benutzer hinzufügen, indem Sie die `GRANT`-Tabellen direkt per SQL-Anweisungen editieren oder die `GRANT`-Anweisung im MySQL-Monitor benutzen. Darüber hinaus wurde auf das MySQL-Werkzeug `mysql_setpermission` eingegangen, das Ihnen erlaubt, auf einfache und schnelle Weise neue Benutzer und Rechte anzulegen.

Des Weiteren haben Sie heute erfahren, wie Benutzer einer Datenbank wieder gelöscht werden können. Sie haben gelernt, dass die einzige Möglichkeit, einen Benutzer komplett aus der Datenbank zu löschen, darin besteht, diesen direkt mit der

SQL-DELETE-Anweisung aus den GRANT-Tabellen zu entfernen. Außerdem haben Sie erfahren, dass Sie einige bzw. alle Rechte eines Benutzers mit der REVOKE-Anweisung entziehen können.

Abschließend wurden einige Möglichkeiten vorgestellt, wie Sie Ihre Datenbank gegen böswillige Attacken schützen können. Letztendlich liegt die Verantwortung für die Daten beim DBA. Sie sind sowohl für die Datenbank als auch für die darin liegenden Daten verantwortlich. Es liegt bei Ihnen und Ihrem Geschäft, für eine strenge Sicherheitspolitik zu sorgen – egal, welche Art von Daten Sie verwalten.

Fragen und Antworten

F Gibt es noch andere Vorsichtsmaßnahmen, um die Sicherheit meiner Datenbank zu gewährleisten?

A *Es gibt tausend Möglichkeiten, wie Sie Ihre Datenbank sicher machen können. Dieses Kapitel hat Ihnen gezeigt, was Sie intern mit MySQL selbst machen können. Es wurde nicht darauf eingegangen, wie Sie Ihren Server sicherer machen können. Firewalls und Proxy-Server sind nur zwei Wege, um Hacker abzuschrecken. Service abzuschotten und Sicherheitspatches zu installieren, wenn sie verfügbar sind, ist ein anderer Weg, um die Sicherheit des Systems zu gewährleisten. Leider gehen diese Themen weit über dieses Buch hinaus. Sie sollten sich nach entsprechenden Büchern, die sich mit der Sicherheit von Betriebssystemen befassen, umschauen.*

F Ich habe einen Benutzer komplett aus der Datenbank gelöscht. Trotzdem hat dieser Benutzer immer noch Zugriff. Warum?

A *Es gibt viele Möglichkeiten für dieses Problem. Die nahe liegendste ist, dass Sie einen anonymen Benutzer in Ihrer Benutzer-Datenbank haben. Ein anonym Benutzer ist ein Benutzer, der einen leeren Eintrag in der User-Spalte der user-Tabelle hat. Das erlaubt jedem, mit Ihrer Datenbank Verbindung aufzunehmen. Um zu testen, ob das Ihr Problem ist, führen Sie folgende Anfrage aus:*

```
SELECT * FROM user WHERE User = "";
```

Dies kann ein großes Sicherheitsloch sein, ohne dass Sie es bemerken.

F Ich habe eine Internet-Applikation geschrieben, die mit einer MySQL-Datenbank interagiert. Ich möchte zwar, dass Benutzer sich in mein System einloggen können, aber ich möchte dafür nicht unzählige Benutzerkonten erstellen. Was soll ich tun?

- A** *Es ist üblich, eine user-Tabelle innerhalb Ihrer eigenen Datenbank zu erstellen. Sie können Benutzer in dieser Datenbank anstatt in der MySQL eigenen user-Tabelle anlegen. Sie können dann die Zugangsberechtigung überprüfen, indem Sie Anfragen auf die gleiche Art wie MySQL an diese Tabelle richten.*

Übungen

1. Fügen Sie einige Benutzer zu der Meet_A_Geek-Datenbank hinzu, indem Sie sich der Techniken bedienen, die Sie in diesem Kapitel erlernt haben. Fügen Sie einen Benutzer mit der GRANT-Anweisung, einen mit dem mysql_setpermission-Programm und einen, indem Sie die GRANT-Tabellen direkt editieren, hinzu.
2. Entziehen Sie den Benutzern, die Sie in Übung 1 erstellt haben, mit der REVOKE-Anweisung alle Rechte.
3. Löschen Sie die Benutzer wieder aus der Datenbank.

A tilted, light gray calendar grid is visible in the background, showing days of the week (M, D, M, D, F, S) and empty date boxes.

MySQL im Vergleich

**Woche
3**

Über die Open-Source-Software wurde heftig diskutiert. Viele Leute denken: »Was nichts kostet, taugt auch nichts.« Sie verstehen nicht, dass ein Open-Source-Produkt es mit den etablierten Firmen aufnehmen kann. Die meisten Leute vertrauen lieber auf »Markenprodukte«, auch wenn diese schlechte Software produzieren. Mit Linux kam die Wende. Es hat der Open-Source-Software Legitimation verschafft und den Weg geebnet für Produkte wie MySQL. Dieses Kapitel vergleicht MySQL mit all seinen Stärken und Schwächen mit anderen Datenbanken der gleichen Größenordnung.

Heute erfahren Sie etwas über:

- ▶ Schnelligkeit
- ▶ so genannte *Workarounds* in MySQL für gespeicherte Prozeduren
- ▶ Transaktionen in MySQL und anderen Datenbanken
- ▶ Beschränkungen von MySQL

Schnelligkeit muss sein

Es wurde notwendig, immer größere Datenmengen zu speichern und zugänglich zu machen, und ebenso wuchs auch der Bedarf an Datenbanken, die Anfragen schnell und effektiv verarbeiten können. Der Wettlauf um die schnellste Datenbank hatte begonnen. Gleichzeitig wuchs aber auch die Nachfrage nach mehr Funktionalität. Datenbank-Administratoren benötigten immer mehr Zusatzprogramme und effektivere Methoden, um mit den gespeicherten Datenmassen fertig zu werden. Das Problem war folgendes: Je mehr Funktionalität und Features einer Datenbank hinzugefügt wurden, desto langsamer wurde sie. MySQL hat sehr darauf geachtet, die größtmögliche Anzahl an Features bereitzustellen, ohne jedoch an Schnelligkeit einzubüßen.

Wie schnell ist MySQL? TcX hat eine Webseite (<http://www.tcx.com/>) eingerichtet, auf der sie die Unterschiede zwischen MySQL und anderen Datenbanken einsehen können. Diese Webseite benutzt die Ergebnisse von *crashme*, einem Programm, das die Funktionalität aller großen relationalen Datenbanksysteme auflistet und vergleicht. TcX bietet dabei auch einen Link auf eine Seite an, auf der die Schnelligkeit von MySQL und anderen Datenbanken auf verschiedenen Plattformen unter Verwendung des gleichen Interface getestet wurde.

MySQL ist in nahezu jeder Kategorie am schnellsten. Im Vergleich mit DB2, Informix, Sybase Enterprise 11.5 und Microsoft SQL-Server – alle unter Windows NT und mit ODBC als gemeinsamer Schnittstelle getestet – übertrifft MySQL die anderen Datenbanken bei fast jedem Test. Ein Test beinhaltete beispielsweise die Ausführung von 20.000 `SELECT`-Anweisungen mit unterschiedlichem Schwierigkeitsgrad. MySQL er-

füllte diese Aufgabe fast doppelt so schnell wie der Microsoft SQL-Server und fünfmal so schnell wie Informix. Den Statistikern zufolge ist MySQL im Schnitt um 40 % schneller als anderen Datenbanken dieser Größenordnung.

Wie Sie alle wissen, können Statistiken manipuliert sein. Am besten probieren Sie MySQL also selbst aus. Falls Sie jemals eine »Marken-Datenbank« benutzt haben, werden Sie schwer beeindruckt sein von dem, was MySQL leisten kann. Falls nicht, werden Sie auch ohne Vergleichsmöglichkeit über MySQLs Schnelligkeit erstaunt sein.

Aus diesem Grund eignet sich MySQL auch besonders gut für das Internet. Es übertrifft die Schnelligkeitsanforderungen für das Web bei weitem. Anfragen werden schneller ausgeführt und Ergebnisse schneller zurückgegeben. Da MySQL sich so gut für das Internet eignet, ist es auch gut in Unternehmens-Applikationen zu gebrauchen.

Wie bereits gesagt, nimmt bei steigender Funktionalität die Schnelligkeit ab. Die Entwickler von MySQL haben ein optimales Gleichgewicht zwischen Funktionsumfang und Schnelligkeit hergestellt. Sie konnten natürlich nicht alle Features, die in anderen Datenbanken dieser Größenordnung enthalten sind, einbringen, haben aber die wichtigsten ausgewählt und die weniger wichtigen unter den Tisch fallen lassen. Bald wird MySQL selbst diese weniger wichtigen Funktionen beinhalten, ohne großartig an Schnelligkeit zu verlieren. Die Entwickler überlassen die Entscheidung, ob sie ein Feature-Set auf höherem Niveau benutzen und somit etwas Geschwindigkeit opfern wollen, den Datenbank-Administratoren. Den Administratoren die Wahl zu lassen, ist von großem Vorteil; es überlässt somit versierten Benutzern, die Ihre Bedürfnisse genau kennen, die Entscheidung und nicht den Herstellern.

Datenbanken und Transaktionen

Einige Datenbanken, wie zum Beispiel Microsoft SQL-Server und Sybase Adaptive Server benutzen eine Form von SQL namens Transact-SQL oder abgekürzt T-SQL, welche, im Gegensatz zur gewöhnlichen SQL, die Fähigkeit besitzt, Programme zu erstellen. Diese Programme sind nicht kompiliert und auch nicht alleine lauffähig; sie enthalten aber die gleichen Kontrollstrukturen wie die meisten anderen Programmiersprachen, wie zum Beispiel Schleifen und IF-Anweisungen. Ein T-SQL-Befehl kann beispielsweise folgendermaßen aussehen:

```
BEGIN TRAN update_orders
UPDATE Orders
SET Order_Shipped = 'Y'
WHERE Order_ID = 1454
IF @@ERROR = 0
BEGIN
    COMMIT TRAN update_orders
```

```
END
ELSE
BEGIN
    ROLLBACK TRAN update_orders
    PRINT "Error encountered. Transaction failed"
END
```

Wie Sie sehen, sind diese Anweisungen etwas komplexer als die SQL-Anweisungen, die Sie bisher kennen gelernt haben. Die Fähigkeit, komplexe Vorgänge auf einem Datenbank-Server auszuführen, eröffnet Ihnen zahlreiche Möglichkeiten.

Diese Möglichkeiten sind innerhalb eines Feature-Sets für diese Datenbanken hinzugefügt worden. Sie umfassen gespeicherte Prozeduren, die Fähigkeit, Transaktionen rückgängig zu machen (*Rollbacks*), sowie *Cursor*. Diese Merkmale werden Sie in der aktuellen Version von MySQL nicht finden. Der Grund, weshalb diese noch nicht in MySQL implementiert sind, und die entsprechenden Lösungsansätze, mit denen diese Funktionalität dennoch realisiert werden kann (für diejenigen, die sich trotzdem durchschlagen wollen), werden in den nächsten Abschnitten behandelt.

BEGIN TRAN und ROLLBACK TRAN

Datenbanken sind auf Transaktionen aufgebaut. Eine *Transaktion* ist eine Aktion, die auf einer Datenbank durchgeführt werden kann. Jede INSERT-, UPDATE- oder DELETE-, aber auch CREATE- und DROP-Anweisung wird als Transaktion angesehen.

Jeder T-SQL-Befehl wird, bevor er ausgeführt wird, vom Datenbanksystem bearbeitet. Dies ermöglicht der Datenbank-Engine, zuerst alle Befehle einzulesen, um dann einen effizienteren Anweisungsplan zu entwickeln. Erst dann werden diese ausgeführt. Da die Datenbank-Engine alle Anweisungen zuerst bearbeitet, bevor sie diese ausführt, kann sie zwei sehr nützliche Features einsetzen – BEGIN TRAN und ROLLBACK TRAN.

Sie können eine Reihe von Anweisungen geben und falls etwas nicht so funktioniert, wie es eigentlich sollte, können Sie alles rückgängig machen, so dass die Transaktion nicht ausgeführt wird. Werfen Sie einen Blick auf folgendes Beispiel:

```
BEGIN TRAN
DELETE FROM Customers
WHERE Customer_ID = 13
ROLLBACK TRAN
```

Die Datenbank-Engine erstellt einen Ausführungsplan für diese Reihe von Anweisungen. Da hier aber ROLLBACK und nicht COMMIT eingegeben wurde, wird die DELETE-Anweisung nicht ausgeführt. Der Server übermittelt Ihnen, wie viele Datensätze betroffen gewesen wären, falls Sie diese Transaktion durchgeführt hätten – mehr aber auch nicht. Wie Sie sehen, ist ROLLBACK ein ziemlich nützliches Feature, mit dem Sie bedenkenlos Ihre SQL-Anweisungen ausprobieren können. Sie können überprüfen, wie lan-

ge eine umfangreiche Anfrage dauert, ohne dabei Ihre Daten zu gefährden. Das ist eine tolle Sache, geht aber zu Lasten der Geschwindigkeit. Da der Server jede gegebene Anweisung bearbeiten muss, verwendet er Zeit darauf, die Anfrage zu überprüfen und zu bearbeiten, anstatt sie direkt auszuführen, und büßt somit an Performance ein.

MySQL ist kein transaktionales Datenbanksystem. Es benutzt auch nicht T-SQL. Schon gar nicht können Sie eine Anweisung wie im obigen Beispiel geben und erwarten, dass sie funktioniert. Auf den ersten Blick scheint dies ein Nachteil zu sein, und Sie fragen sich vielleicht, warum MySQL dieses Feature nicht unterstützt. Die Antwort ist ganz einfach: MySQL ist auf Schnelligkeit und einfachen Gebrauch ausgelegt. Der Server einer transaktionalen Datenbank benötigt viel Zeit, um jede Anweisung zu verarbeiten, Zeit, die verwendet werden könnte, um Ihre Anweisungen auszuführen. Das ist ein wichtiger Grund, weshalb MySQL T-SQL nicht verwendet. Ein weiterer Grund, weshalb MySQL die *COMMIT*-/ *ROLLBACK*-Fähigkeit nicht unterstützt, besteht darin, dass dies gegen die Philosophie des einfachen Gebrauchs und der niedrigen Kosten verstoßen würde. In einem transaktionalen System wird jede Transaktion in eine spezielle Datei, dem *Transaktionsprotokoll*, geschrieben. Sie kann somit, falls nötig, rückgängig gemacht werden. Die Größe dieser Datei wird vom Administrator festgelegt. Wenn dieser die Datei für die auftretenden Transaktionen zu klein gestaltet, werden, sobald die Datei voll ist, keine weiteren Transaktionen durchgeführt. Das heißt, niemand kann mehr Datensätze in einer Produktionsdatenbank einfügen, aktualisieren oder löschen – was einer Katastrophe gleichkäme. Dieses Transaktionsprotokoll stellt einen wunden Punkt für die meisten Administratoren transaktionaler Datenbanken dar. Sie widmen dieser Datei aus gutem Grund ihr besonderes Augenmerk. MySQL besitzt auch eine Art Protokollierungs-Feature (Logging-Mechanismus). Dieses Feature führt nicht zu solchen Problemen, die ein Transaktionsprotokoll einer transaktionalen Datenbank verursachen kann.

Hoffentlich verstehen Sie jetzt, weshalb TcX sich gegen die Implementierung dieses Merkmals entschieden hat. Es geht nicht nur zu Lasten der Geschwindigkeit, sondern verursacht auch einen höheren Zeitaufwand bei der Administration. Seit der Version 3.23.15 bietet MySQL mit der Berkeley DB dennoch eine Möglichkeit an, Transaktionen zu nutzen. Eine ausführliche Beschreibung würde jedoch den Rahmen dieses Buches sprengen. Daher verweisen wir Sie an dieser Stelle auf die offizielle MySQL-Dokumentation, die Sie unter <http://www.mysql.com/documentation/> finden.

Gespeicherte Prozeduren

Gespeicherte Prozeduren sind ein weiteres Feature, das von T-SQL profitiert. Es handelt sich dabei um eine Reihe von T-SQL-Befehlen, die kompiliert in der Datenbank gespeichert werden. Dabei wird ein Anweisungsplan dieser Befehlskette gespeichert, so dass zukünftige Aufrufe schneller ausgeführt werden können, ohne erst den Anweisungsplan zu erstellen. Intern werden sie genauso wie jedes andere Datenbank-Objekt (Datenbank, Tabelle) behandelt.

Gespeicherte Prozeduren bieten viele Vorteile; beispielsweise passen sie sehr gut in das Konzept modularer Programmierung, in dem sie programmiert wurden, um eine bestimmte Aufgabe zu erfüllen. Sie eignen sich außerdem hervorragend zur Wiederverwendung. Ein weiterer Vorteil besteht darin, dass sie Parameter übernehmen und weitergeben können. Dies ermöglicht dem Programmierer, mehrere gespeicherte Prozeduren miteinander zu verbinden und eine Art Batch-Programm zu erstellen. Das Verlagern von Sicherheitsaspekten stellt einen weiteren Vorzug dar. Ein Benutzer, der keine expliziten Berechtigungen für eine bestimmte Tabelle hat, könnte diese Tabelle trotzdem über gespeicherte Prozeduren verwenden. Dies ist eine feine Sache, wenn Sie den Zugang des Benutzers begrenzen, diesem aber dennoch eine Möglichkeit bieten möchten, Daten in einer von Ihnen gesicherten Umgebung zu manipulieren.

Gespeicherte Prozeduren können sehr komplex werden. T-SQL stellt die meisten Funktionalitäten, die auch andere Programmiersprachen anbieten, zur Verfügung. In ihnen kann der Programmierer Variablen deklarieren, Schleifen erstellen und Bedingungsanweisungen ausführen – ganz zu schweigen von den eigentlichen Funktionen, die der Datenbank-Server bietet. Dies ermöglicht dem Programmierer, eine Menge Programmierlogik auf dem Datenbank-Server zu realisieren.

Einer der Hauptvorteile von gespeicherten Prozeduren besteht darin, dass sie schneller ausgeführt werden als normale Anweisungen, da der Anweisungsplan bereits erstellt und gespeichert ist. Der beste Weg ist also schon ermittelt. Es ist in etwa so, als wenn Ihr ganzer Tag schon im Voraus geplant wäre und Sie nur noch diesem Plan folgen müssten. Das wäre um einiges schneller, als wenn Sie erst jede einzelne Aktion planen müssten, bevor Sie diese ausführen. Auch im Quellcode ist eine gespeicherte Prozedur übersichtlicher als eine Reihe von SQL-Anweisungen.

Gespeicherte Prozeduren haben keine wirklichen Nachteile. In der Tat sind sie das bevorzugte Mittel, um T-SQL-Anweisungen auf einem transaktionalen Datenbanksystem auszuführen. Der einzige kleine Nachteil besteht darin, dass Sie diese im Voraus erstellen müssen. Ist eine Datenbank sehr schnell – so schnell, dass das Ergebnis einer Reihe von Anweisungen, die von einem Remote-Rechner ausgeführt werden, schneller zurückgegeben wird als das einer gespeicherten Prozedur – warum sollten Sie diese dann überhaupt brauchen? Verständlicherweise ist es praktisch, eine Anweisungsausführung auf dem Server zu haben und lediglich die Ergebnisse zurückzugeben. Aber lohnt es sich, dafür die Schnelligkeit von MySQL zu opfern?

Möglicherweise beinhaltet eine zukünftige Version von MySQL eine Option für gespeicherte Prozeduren. Vergessen Sie jedoch nicht, je mehr Funktionalität hinzugefügt wird, desto langsamer wird MySQL. MySQL ist auch ohne gespeicherte Prozeduren schnell genug, aber manchmal hat ihre Verwendung einfach einen Sinn. Für diese Fälle gibt es einen *Workaround*.

Workarounds für gespeicherte Prozeduren

Den einzig möglichen Lösungsansatz, um die Funktionalität von gespeicherten Prozeduren zu realisieren, bieten derzeit benutzerdefinierte Funktionen (so genannte UDFs, englisch für *user-defined-functions*). Diese Funktionen werden in C bzw. C++ erstellt. Sie sind den eigentlichen Funktionen von MySQL sehr ähnlich und werden auch auf die gleiche Weise in MySQL verwendet.

Das Erstellen von UDFs ist eine relativ schwierige Angelegenheit. Jede UDF sollte als Funktion in ihrem C++-Programm erscheinen. Jede UDF kann eine optionale `init`- und `deinit`-Funktion haben. Diese funktionieren wie Konstruktoren und Destruktoren. Sie können die Argumente in der `init`-Funktion überprüfen und eine Aufräumaktion am Ende mit der `deinit`-Funktion durchführen.

Listing 18.1 ist ein Beispiel für eine UDF, die eine Zahl verdoppelt, die der Funktion als Argument übergeben wurde. Ihre UDF wird wahrscheinlich ein wenig komplexer sein. Dieses Beispiel ist absichtlich einfach gehalten worden, damit Sie sehen können, wie eine UDF aufgebaut ist:

Listing 18.1: Benutzerdefinierte Funktionen in C

```
#ifdef STANDARD
#include <stdio.h>
#include <string.h>
#else
#include <global.h>
#include <my_sys.h>
#endif

#include <mysql.h>
#include <mctype.h>
#include <mstring.h>

extern "C" {
my_bool Double_Proc_init(UDF_INIT *initid,
UDF_ARGS *args, char *message);
void Double_Proc_deinit(UDF_INIT *initid);
long long Double_Proc(UDF_INIT *initid, UDF_ARGS *args,
char *is_null, char *error);
}

my_bool Double_Proc_init(UDF_INIT *initid, UDF_ARGS *args, char
*message)
```

```
{
    if (args->arg_count != 1 || args->arg_type[0] != INT_RESULT)
    {
        strcpy(message,"Wrong arguments to Double_Proc");
        return 1;
    }
    return 0;
}

void Double_Proc_deinit(UDF_INIT *initid)
{
}

long long Double_Proc(UDF_INIT *initid, UDF_ARGS *args,
                     char *is_null, char *error)
{
    int ResultSet;
    ResultSet = args * 2;

    Return ResultSet;
}
```

Alle UDFs müssen die Bibliotheken aus dem Beispiel enthalten. Nach den include-Anweisungen sehen Sie die Prototypen der Funktion. Als Rückgabewert kommen nur drei Typen für Ihre Funktion in Frage. Diese sind REAL, INTEGER und STRING. Diese Typen sind äquivalent mit double, long und char* in C++. Möchten Sie, dass Ihre Funktion einen SQL-STRING-Typ zurückgibt, so deklarieren Sie den Prototyp als char*. Möchten Sie einen SQL-INTEGER-Typ zurückbekommen, müssen Sie den Prototyp als long und für einen SQL-REAL-Typ müssen Sie diesen als double deklarieren.

Vergessen Sie nicht, dass Ihre UDF konform mit dem in Listing 18.1 gezeigten Layout sein muss. Die init- und deinit-Funktionen müssen ebenfalls, wie erläutert, benannt werden. Wenn Sie sich nicht an die Namenskonventionen, die in Listing 18.1 gezeigt werden, halten, wird Ihre UDF nicht funktionieren. So wird im Listing zum Beispiel die Funktion Double_Proc aufgerufen. Die init-Funktion muss daher Double_Proc_init heißen. Wenn die Funktion einen anderen Namen hätte, können Sie Double_Proc durch diesen Namen ersetzen, müssten aber das _init beibehalten. Die gleichen Namenskonventionen gelten auch für die deinit-Funktion.

Die Deklaration der Funktionsargumente sowie Argumententyp und -anzahl müssen mit denen in Listing 18.1 übereinstimmen. Gehen Sie daher das Listing Schritt für Schritt ganz genau durch.

Der besten Weg, um mehr über den Einsatz und Aufbau von UDFs zu erfahren, ist, den MySQL-Quellcode von der MySQL-Webseite herunterzuladen und zu entpacken. Suchen Sie in dem Verzeichnis nach einem SQL-Verzeichnis. Hier finden Sie ein Bei-

spiel für eine UDF namens `udf_example.cc`. Sie können diese Datei als Vorlage für Ihre eigenen UDFs verwenden. So können Sie sicherstellen, dass Sie das richtige Format und die richtigen Deklarationen verwenden.

Wichtig ist auch, dass Sie mehrere UDFs in einer Datei haben können. Sie können in der Tat so viele UDFs erstellen, wie Sie Lust haben – das im MySQL Quellcode mitgelieferte Beispiel macht dies deutlich.

Nachdem Sie Ihre UDF erstellt haben, müssen Sie diese als gemeinsam nutzbares Objekt kompilieren (das bedeutet, dass die von Ihnen verwendete Plattform diese Möglichkeit unterstützen muss). Um Ihre UDF zu kompilieren, geben Sie folgende Befehle ein:

```
gcc -shared -o Double_Proc.o Double_proc.cc
```

Nachdem Sie Ihre Prozedur erfolgreich kompiliert haben, müssen Sie diese in das Verzeichnis verschieben, in dem das System nach gemeinsam genutzten Dateien sucht. Unter Red Hat sollte es das `/usr/lib`-Verzeichnis sein. Kopieren Sie die Datei in dieses Verzeichnis. Jetzt stoppen und starten Sie den `mysqld`-Daemon neu.

Anschließend müssen Sie MySQL die neue Funktion bekannt machen. Das machen Sie, indem Sie die `CREATE FUNCTION`-Anweisung verwenden. Die `CREATE FUNCTION`-Anweisung hat folgende Syntax:

```
CREATE FUNCTION Funktionsname  
RETURNS Rückgabe-Typ  
SONAME shared_library_name
```

Nach dem `CREATE FUNCTION`-Schlüsselwort geben Sie den Funktionsnamen Ihrer UDF an. In unserem Beispiel wäre dies `Double_Proc`. Danach geben Sie den Rückgabebetyp Ihrer UDF an. Dieser kann `REAL`, `INTEGER` oder `STRING` sein. Er sollte mit dem Rückgabebetyp Ihrer UDF übereinstimmen. Der letzte Parameter der Anweisung ist der Name Ihrer kompilierten Prozedur. Das ist die Datei, die Sie in das `/usr/lib`-Verzeichnis verschoben haben. Um bei unserem Beispiel zu bleiben, würden Sie einen Eintrag in der `func`-Tabelle der `mysql`-Datenbank wie folgt erstellen:

```
CREATE FUNCTION Double_Proc  
RETURNS INTEGER  
SONAME Double_Proc.so
```

Wenn Sie eine Funktion aus dieser Tabelle löschen möchten, verwenden Sie die `DROP FUNCTION`-Anweisung. Diese Anweisung hat die folgende Syntax:

```
DROP FUNCTION Funktionsname
```

Nach dem `DROP FUNCTION`-Schlüsselwort geben Sie den Namen der zu löschenden Funktion an. Bei dem vorangegangenen Beispiel können Sie die Funktion `Double_Proc` mit folgender Anweisung wieder löschen:

```
DROP FUNCTION Double_Proc
```

Die CREATE- und DROP-Funktionen fügen Einträge in die func-Tabelle der mysql GRANT-Tabellen hinzu bzw. löschen diese. MySQL lädt diese Funktionen jedes Mal, wenn es neu gestartet wird, aus dieser Tabelle.

Diese Funktionen sind in allen Datenbanken verfügbar. Sie stehen nicht nur einer bestimmten Datenbank zur Verfügung. Sie können von jedem verwendet werden, der SELECT-Berechtigungen für eine Datenbank hat.

Wie Sie sehen, sind UDFs gespeicherten Prozeduren sehr ähnlich, stellen jedoch nicht die gleiche Funktionalität wie diese zur Verfügung. Das Schöne an UDFs ist, dass sie keine Auswirkungen auf die Schnelligkeit von MySQL haben. Der Nachteil ist allerdings, dass Sie nicht wirklich SQL-Anweisungen in diese einbetten können, was bei gespeicherten Prozeduren jedoch möglich ist. Vorgänge wie das Überprüfen von Daten, die Berechnung komplexer Anweisungen oder die Durchführung sehr spezielle Manipulationen können Sie alle mit solchen UDFs erledigen.

Zukünftige Versionen von MySQL werden eine *PROCEDURE*-Anweisung zur Verfügung stellen, die annähernd die gleiche Funktionalität bietet wie gespeicherte Prozeduren. Dadurch wird es möglich, die Daten zu manipulieren, bevor Sie zum Client zurückgegeben werden.

Da MySQL so schnell ist, gibt es wenige Gründe für den Einsatz von gespeicherten Prozeduren. Unter Verwendung der verschiedenen APIs können Sie Funktionen innerhalb dieser APIs erstellen, welche die gleichen Aufgaben wie gespeicherte Prozeduren erfüllen. Unter Verwendung von Visual Basic und dem ODBC-Interface können Sie zum Beispiel eine Funktion erstellen, welche die Aktualisierung einer ganzen Tabelle bewirkt. Werfen Sie einen Blick auf das folgende Beispiel:

```
Option Explicit
Dim mConn As New ADODB.Connection
Dim mCmd as New ADODB.Command

Public Sub UpdateCustomers(DateOrdered)
With mConn
    .ConnectionString = "Driver=MySQL;Server=10.1.2.23;" & _
        db=Meet_A_Geek;uid=sa;pwd=taco"
    .Open
End With

With mCmd
    .ActiveConnection = mConn
    .CommandType = adCmdText
    .CommandText = "UPDATE Customers SET Active = 'N' " & _
        WHERE Last_Order < " & DateOrdered
    .Execute
End With
End Sub
```

Dieser Codeschnipsel aktualisiert die Customers-Tabelle, indem er alle Kunden auf inaktiv setzt, wenn deren letzte Bestellung zeitlich vor einem der Funktion in DateOrdered übergebenen Datum liegt. Dieser Programmcode kann wiederverwendet werden. Die Person, die diese Funktion ausführt, benötigt keine UPDATE-Rechte, da Sie einen anderen Benutzer in der Verbindungszeile angeben können, der UPDATE-Rechte hat. Da es sich hierbei um eine Programmiersprache handelt, können Sie auch von der Logik und den Funktionen dieser Sprache profitieren. Sie könnten sogar ein COM-Objekt mit diesem Programmcode erstellen und es auf dem Server laufen lassen. So haben Sie alle Vorteile einer gespeicherten Prozedur, ohne auch nur ein kleines Stück Geschwindigkeit dafür zu opfern.

Cursor

Cursor ermöglichen Ihnen, immer nur einen Datensatz aus einem Ergebnissatz zu manipulieren. Dies entspricht im Großen und Ganzen der Art und Weise, wie Sie unter Verwendung einer API durch einen Ergebnissatz gehen. Cursor ermöglichen dem Programmierer oder DBA, komplexe Aufgaben innerhalb der Datenbank auf dem Server auszuführen. Das ist eine schöne Eigenschaft, aber für Cursor etwas schwierig. Sie müssen in der Regel mehrere Schritte wie Deklaration, Öffnen, Holen eines Datensatzes und Schließen eines Cursors ausführen. Im Folgenden sehen Sie ein Beispiel, wie Cursor im Normalfall verwendet werden:

```
DECLARE @Cust_ID VARCHAR(20)
DECLARE customers_c CURSOR
FOR SELECT Customer_ID FROM Customers WHERE State= "KY"
OPEN customers_c
FETCH customers_c INTO @Cust_ID
WHILE @@FETCH_STATUS=0
BEGIN
    UPDATE Orders SET Sales_Tax = .06 WHERE Customer_ID = @Cust_ID
    FETCH customers_c INTO @Cust_ID
END
CLOSE customers_c
DEALLOCATE customers_c
```

Dieses Beispiel zeigt nicht die optimale Verwendung eines Cursors, sondern es zeigt Ihnen, was ein Cursor ist und wie er verwendet werden kann. In der ersten Zeile wird eine Variable @Cust_ID deklariert, welche die Werte eines Datensatzes von einem Cursor hält. In der nächsten Zeile wird der Cursor deklariert. In der dritten Zeile sehen Sie die Anfrage, aus welcher der Cursor gemacht wird. Der Cursor wird das Ergebnis dieser Anfrage beinhalten. Die nächste Zeile öffnet den Cursor oder – mit anderen Worten – sie führt die Anweisung aus. In der folgenden Zeile wird der erste Datensatz aus dem Ergebnis im Cursor in die Variable @Cust_ID geladen. Die nächsten Anweisungen führen eine Schleife aus, die durch den Ergebnissatz, der im Cursor liegt, läuft. Diese

Schleife aktualisiert – basierend auf den Werten im Cursor – die `Orders`-Tabelle. Das geschieht so lange, bis der ganze Ergebnissatz durchlaufen wurde. Danach wird der Cursor geschlossen und dealloziert.

Der größte Nachteil bei der Verwendung von Cursor, ist der Performance-Verlust. Cursor beanspruchen viele Ressourcen, um ihre Aufgaben zu erledigen. Sie werden normalerweise lediglich als letzte Möglichkeit verwendet.

MySQL stellt eben aus diesem Grunde keine Cursor zur Verfügung. Sie sind zu langsam. Der einzige Weg, um solche Aufgaben zu erledigen, besteht darin, Algorithmen in der API zu erstellen. Das ist schneller und einfacher zu lesen und zu verfolgen. Die MySQL-Entwickler haben gut daran getan, alle unnötigen Funktionen auszulassen, denn diese hätten MySQL von dem weggeführt, was es primär ausmacht: Schnelligkeit.

Trigger

Ein Trigger ist eine gespeicherte Prozedur, die ausgeführt wird, wenn eine bestimmte Aktion wie zum Beispiel eine `INSERT`-, `UPDATE`- oder `DELETE`-Anweisung gegen die Datenbank ausgeführt wurde. Hierzu ein Beispiel: jedes Mal, wenn ein Benutzer einen Datensatz aus der `Customers`-Tabelle löscht, möchten Sie diesen Datensatz automatisch in einer `Customers_Archiv`-Tabelle archivieren. Anstatt der Applikation, die den Datensatz löscht, zusätzlichen Programmcode hinzuzufügen oder eine gespeicherte Prozedur manuell in der Applikation aufzurufen, könnten Sie einen Trigger benutzen, der diese Aufgabe für Sie übernimmt. Immer wenn ein Datensatz aus der `Customers`-Tabelle gelöscht würde, würde der Trigger gefeuert werden und den Datensatz in die Tabelle `Customers_Archiv` einfügen.

Trigger haben viele Vorteile. Sie können helfen, die Integrität einer Datenbank sicherzustellen, indem Sie Beziehungen der Tabellen verwalten. Da sie bei einer bestimmten Aktion gefeuert werden, können sie zum Beispiel Aufräumarbeiten automatisch übernehmen und somit dem DBA helfen, die Tabellen in Ordnung zu halten. In einer 1:1-Relation zum Beispiel könnte ein Trigger angewiesen werden – für den Fall, dass ein Datensatz aus einer Tabelle gelöscht wird – den korrespondierenden Datensatz aus der anderen Tabelle automatisch zu entfernen. In einer 1:n-Relation könnte ein Trigger gefeuert werden, wenn ein Datensatz aus einer »Eltern«-Tabelle gelöscht wird, um alle korrespondierenden Datensätze der »Kinder«-Tabellen zu löschen. Das kann eine Menge Arbeit und Extra-Programmcode sparen.

Ein anderes, praktisches Merkmal von Triggern ist der Dominoeffekt. Dieser tritt auf, wenn ein Trigger einer Tabelle gefeuert wird und damit das Feuern eines anderen Triggers einer anderen Tabelle auslöst und so weiter. Das kann sehr nützlich, aber auch sehr gefährlich sein. Nehmen Sie einmal an, Sie möchten alle Daten eines Kunden aus der `Meet_A_Geek`-Datenbank löschen, wenn dieser seine Mitgliedschaft been-

det. Anstatt eine Menge Programmcode zu schreiben, um eine Aufräumaktion auszuführen, könnten Sie Trigger benutzen, die diese Aufgabe für Sie ausführen. Sie könnten einen Trigger erstellen, der alle Bestellungen des Kunden aus der *Orders*-Tabelle löscht, wenn dieser Kunde aus der *Customers*-Tabelle gelöscht wird. Sie könnten auch einen Trigger in der *Orders*-Tabelle haben, der alle Transaktionen mit den entsprechenden Bestellnummern löscht, wann immer eine Bestellung gelöscht wird. Wenn Sie also einen Kunden löschen, werden zwei Trigger gefeuert. Einer löscht alle korrespondierenden Bestellungen aus der *Orders*-Tabelle, der andere löscht alle dazugehörigen Transaktionen in der *Transaction*-Tabelle. Das spart Ihnen eine Menge Zeit. Wenn Sie jedoch den Überblick über Ihre Trigger verlieren, kann es zu unerwünschtem Datenverlust führen. Seien Sie also vorsichtig.

Der Nachteil bei der Verwendung von Triggern ist, dass Sie das System langsamer machen. Jedes Mal, wenn eine Anweisung ausgeführt wird, muss die Datenbank zuerst überprüfen, ob ein Trigger für diese Aktion existiert. Wenn dies der Fall ist, wird die entsprechende Aktion ausgeführt. Das kann zu Lasten anderer Aktionen gehen, was der Hauptgrund dafür ist, dass MySQL keine Trigger unterstützt.

In der Tat gibt es keine Workarounds für die Verwendung von Triggern in MySQL. Wenn Ihr Applikations-Code und Ihr Datenbank-Design gut sind, ist der Bedarf an Triggern minimal. Der beste Ratschlag ist, dass Sie die Datenbankintegrität in Ihrem Applikations-Code sicherstellen. Räumen Sie hinter sich auf, und alles ist in Ordnung.

SQL-bedingte Beschränkungen (Constraints)

Viele Datenbanken erlassen Beschränkungen, um die Datenbankintegrität sicherzustellen. Eine Beschränkung ermöglicht zum Beispiel, Relationalität zu fordern oder Daten in ein bestimmtes Format zu zwingen. Es gibt verschiedene Formen von Beschränkungen wie zum Beispiel *CHECK*, *FOREIGN KEYS* oder *UNIQUE* – nur um einige zu nennen. Sie alle haben den gleichen Zweck: das Sicherstellen der Datenbankintegrität.

Es gibt viele Gründe, weshalb man solche Beschränkungen einsetzt. Ein Vorteil ist, dass es keine Möglichkeit gibt, die Regeln einer Relation zwischen zwei Tabellen zu brechen. Wenn ein Datensatz in einer 1:1-Relation gelöscht wird, muss der korrespondierende Datensatz auch gelöscht werden. Wenn ein Datensatz in einer Tabelle eingefügt wird, muss auch der anderen Tabelle ein Datensatz hinzugefügt werden. Das ist eine gute Möglichkeit, um den Programmcode von der Client-Seite auf den Server zu verlagern.

Ein weiterer Vorteil ist, dass die Integrität der Daten sichergestellt werden kann. Wenn eine alphanumerische Spalte dazu bestimmt ist, nur alphabetische Zeichen aufzunehmen, kann eine *CHECK*-Einschränkung dazu verwendet werden, sicherzustellen,

dass niemals Zahlen in diese Spalte eingetragen werden. Dies bewirkt wiederum, dass Sie diese Überprüfung nicht in Ihrer Applikation ausführen müssen, sondern dass sie innerhalb der Datenbank auf dem Server erfolgt.

Der größte Nachteil dieser bedingten Beschränkungen besteht darin, dass sie eine Menge Systemressourcen verschlingen – besonders beim Einfügen und Aktualisieren von Datensätzen. Das System wird langsamer, muss die Beschränkungen überprüfen und kann die Anweisungen erst dann ausführen.

Einen weiteren Nachteil stellen die Frustration und der Ärger dar, den solche SQL-bedingten Beschränkungen hervorrufen können. Diese Beschränkungen zu umgehen, kann gerade bei `DELETE`-Anweisungen echte Kopfschmerzen verursachen. Diese Kopfschmerzen werden umso größer, je schlechter das Design der Datenbank ist.

MySQL unterstützt SQL-bedingte Beschränkungen nicht. Diese machen es nur langsamer, was gegen den primären Grund, MySQL zu verwenden, spricht. Die Verantwortung liegt in den Händen der Programmierer und Administratoren. Es ist Ihre Angelegenheit, die Integrität der Daten und die Beziehungen zwischen den Tabellen sicherzustellen. Wenn Sie gute Datenbank-Applikationen schreiben und das Design der Datenbank gut und einfach zu verstehen ist, ist der Bedarf für Beschränkungen ebenso wie für Trigger oder jedes andere Extramerkmale minimal.

Zusammenfassung

Es gibt eine Menge schöner Features für Datenbanken. Gespeicherte Prozeduren, Trigger und Cursor stellen Datenbank-Administratoren und Entwicklern mehr Funktionalität zur Verfügung. Doch sie haben ihren Preis. Der Preis, den Sie zahlen, heißt Schnelligkeit und Komplexität. MySQL kann alle Aufgaben, die mit diesen Features gelöst werden können, auch erledigen – vielleicht nicht auf die gleiche Art und Weise, aber Sie erhalten aufgrund der Schnelligkeit und Flexibilität von MySQL das gleiche Ergebnis.

Wenn Sie eine dieser Zusatzfunktionen für Ihre Datenbank einsetzen möchten, bedenken Sie, weshalb diese Funktionen erstellt wurden. Gespeicherte Prozeduren werden wegen Geschwindigkeitsvorteilen gegenüber normalen SQL-Anweisungen verwendet. Da Sie aber schon mit MySQL über eine so hohe Geschwindigkeit verfügen, weshalb sollten Sie gespeicherte Prozeduren einsetzen?

Ein anderer Punkt, den Sie beachten sollten, ist, dass das Vorhandensein von mehr Funktionalität dazu führt, dass die Leute eher fauler und nachlässiger werden. Programmierer sind im Allgemeinen schon faule Menschen (oder wie sie es sehen: effizient denkend) und versuchen daher, die Dinge immer noch einfacher zu machen. Diese Features geben einem Programmierer eine Entschuldigung dafür an die Hand,

nachlässiger zu programmieren und weniger Zeit auf die Datenbank-Entwurfsphase zu verwenden. Das bedeutet nicht, dass alle Programmierer, die diese Funktionen nutzen, nachlässig und faul sind – es bedeutet nur, dass sie jetzt eine Entschuldigung dafür haben, Funktionen für Aufgaben einzusetzen, die auch durch ein gutes Datenbank-Design und eine anständige Programmiertechnik erledigt werden könnten.

MySQL legt die Verantwortung in die Hand der Entwickler. Entwerfen Sie ein gutes Design und programmieren Sie Ihre Applikation richtig, und Sie werden keine gespeicherten Prozeduren oder eines der anderen Features benötigen, die heute diskutiert wurden, da MySQL Sie mit der Schnelligkeit versorgt, an der es den anderen Datenbanken mangelt und die deswegen diese Features implementieren mussten.

Fragen und Antworten

- F** Warum sollte ich eine benutzerdefinierte Funktion (UDF) schreiben?
- A** *Wie heute erklärt wurde, können UDFs helfen, den Bedarf an gespeicherten Prozeduren zu decken. Sie können in einer UDF vordefinierte Routinen erstellen. Eine UDF kann ebenso sehr komplexe mathematische Berechnungen oder Zeichenketten-Manipulationen ausführen (Sie haben Zugriff auf alle C++-Bibliotheken, bevor das Ergebnis zum Benutzer zurückgegeben wird). UDFs sind sehr schnell. Es handelt sich dabei um kompilierte und nicht um interpretierte oder gepackte Programme wie SQL oder Perl.*
- F** Wenn MySQL so viele Merkmale fehlen, warum soll ich es dann benutzen?
- A** *Sie betrachten momentan nur das Negative. MySQL fehlen vielleicht einige Merkmale, die andere Datenbanken haben; aber betrachten Sie, was MySQL alles besitzt, was anderen Datenbanken fehlt:*
- ▶ MySQL kann auf Kommandozeilen-Ebene ausgeführt werden. Das bedeutet, dass Sie eine MySQL-Datenbank von überall her warten können, ohne dabei andere Programme wie PC Anywhere oder VNC einsetzen zu müssen.
 - ▶ MySQL ist extrem schnell – fast doppelt so schnell wie all seine Mitbewerber. Wer braucht schon Extrafunktionalitäten, wenn die Datenbank so schnell ist?
 - ▶ Der Quellcode von MySQL steht zur Verfügung. Für welche andere Datenbank gilt das noch?

Übungen

1. Begeben Sie sich auf die *crashme*-Webseite (<http://www.mysql.com/information/crash-me.php>) und vergleichen Sie MySQL mit der Datenbank Ihrer Wahl.
2. Erstellen Sie eine benutzerdefinierte Funktion ähnlich derjenigen, die am heutigen Tag vorgestellt wurde.



MySQL administrieren

**Woche
3**

In diesem Kapitel werden Sie erfahren, wie MySQL administriert wird. Die Administration von MySQL umfasst ein breites Aufgabenfeld, wie zum Beispiel das Erstellen von Backups und das Wiederherstellen von Daten und beschädigten Datenbanken.

Sie werden einerseits lernen, wie Sie das MySQL-Werkzeug `mysqladmin` für die allgemeine MySQL-Administration verwenden, und andererseits, wie Sie sich MySQLs interner Funktionen bedienen, um auf administrative Daten mittels SQL-Anweisungen zuzugreifen.

Datensicherung

Datensicherung (Erstellen eines Backups) ist in jedem Datenbanksystem eine der wichtigsten administrativen Aufgaben. Die Gründe hierfür sind vielfältig – sie reichen von Feuer und Diebstahl bis hin zu versehentlicher Beschädigung der Daten durch einen Anwender. Was auch immer das zu vermeidende Risiko ist, Datensicherung zielt auf die Minimierung geschäftlicher Risiken. Ebenso wichtig wie die Sicherungsdateien, ist die Weise, wie effizient Sie Daten wiederherstellen können, wenn eine Katastrophe einmal eingetreten ist; das heißt, wie schnell Sie, mit einem Minimum an Ausfallszeit, den normalen Geschäftsbetrieb wiederherstellen können.

MySQL stellt Ihnen zwei nützliche Werkzeuge zur Verfügung, um Daten zu sichern und wiederherstellen: `mysqldump` und `mysqlimport`.

Mit diesen beiden Werkzeugen können Sie Teile oder eine komplette Datenbank sichern und an der gleichen oder an anderer Stelle wiederherstellen.

Speichern von Dateien

`mysqldump` ist ein nützliches Werkzeug, um eine MySQL-Datenbank zu sichern. Es kann sowohl zur Sicherung wie auch zum Abspeichern einer Datenbank eingesetzt werden, so dass diese an einen anderen Ort verlagert werden kann. `mysqldump` können Sie entweder direkt auf Kommandozeilenebene aufrufen oder es mittels eines Zeitplans wie *CRON* unter Unix oder Linux oder dem *At*-Kommando unter Windows NT/2000 verwenden.

Die Syntax für `mysqldump` ist die folgende:

```
mysqldump [Optionen] Datenbank [Tabellen]
```

Wenn Sie `mysqldump`, ohne Optionen oder Tabellen zu spezifizieren, aufrufen, wird das Programm die komplette Datenbank abspeichern (dumpen).

Wenn Sie `mysqldump` ausführen, erstellt es eine Datei, die voll mit `INSERT`- und `CREATE`-Anweisungen ist, die als Instruktionen zur Wiederherstellung der Tabellen und Daten dienen.

Haben Sie die Dump-Datei erstellt, können Sie diese entweder für Sicherungsvorhaben verwenden oder sie an einen Ort senden, an den die Datenbank verschoben werden soll (zum Beispiel bei einem Serverwechsel). In Kürze werden Sie erfahren, wie Sie eine solche Datei mit dem `mysqlimport`-Programm verwenden, um eine Datenbank zu importieren. Zunächst sollten Sie jedoch etwas über die nützlichsten Optionen des `mysqldump`-Programms erfahren:

Option	Bedeutung
<code>--add-locks</code>	Fügt <code>LOCK TABLES</code> -Anweisungen am Anfang und <code>UNLOCK TABLES</code> -Anweisungen am Ende des Tabellen-Dumps ein. Dieses sorgt dafür, dass Tabellen während des Imports gesperrt werden, und führt somit zu einem Geschwindigkeits-zuwachs während des Wiederherstellungsprozesses.
<code>--add-drop-table</code>	Fügt eine <code>DROP-TABLE</code> -Anweisung vor jeder <code>CREATE-TABLE</code> -Anweisung ein. Wenn schon eine Kopie der Datenbank am Zielort besteht, stellt diese Option sicher, dass keine Duplikate entstehen und die Tabellenspezifikation alle vorherigen Spezifikationen ersetzt.
<code>-e, --extended-insert</code>	Veranlasst <code>mysqldump</code> , mehrzeilige <code>INSERT</code> -Syntax zu verwenden. Dieses soll <code>INSERT</code> -Anweisungen kompakter und schneller machen.
<code>-q, --quick</code>	Gibt das Ergebnis des Dumps direkt auf der Standardausgabe aus. Sie brauchen diese Option unter Umständen, wenn Sie knappe Systemressourcen haben.
<code>--opt</code>	Diese Option umfasst die Optionen <code>--add-locks</code> , <code>--add-drop-tables</code> , <code>--extended-insert</code> und <code>--quick</code> . Dieses ist ein bequemer Weg, die schnellstmögliche Dump- und Import-Datei zu erstellen.
<code>-F, --flush-logs</code>	Flusht die MySQL-Server-Logdateien, bevor die Dump-Erstellung gestartet wird.
<code>-h=Hostname, --host=Hostname</code>	Bewirkt, dass der Dump auf <i>Hostname</i> statt auf <code>localhost</code> (welches die Standardeinstellung ist) erstellt wird.
<code>-l, --lock-tables</code>	Veranlasst, dass die Tabellen vor dem Dump gesperrt werden. Wenn komplexere Aktualisierungen in Ihrem System vorkommen, stellt diese Option sicher, dass Sie einen vollständigen Dump erhalten, der nicht nur teilweise aktuell ist.
<code>-t, --no-create-info</code>	Mit dieser Option werden nur die Daten gesichert. <code>CREATE-TABLE</code> -Anweisungen werden ausgelassen.
<code>-d, --no-data</code>	Lässt die eigentlichen Daten weg und sichert nur die Tabellendefinitionen. Diese Option ist nützlich, wenn Sie nur das Design Ihrer Datenbank sichern möchten, jedoch nicht den Inhalt.

Tabelle 19.1: `mysqldump`-Optionen

Option	Bedeutung
<code>--tab=Pfad, -T=Pfad</code>	Speichert den Dump in Dateien Ihres Dateisystems, die mit <i>Pfad</i> spezifiziert sind. <code>mysqldump</code> erstellt für jede Tabelle eine Datei namens <i>Tabellenname.sql</i> , die die <code>CREATE TABLE</code> -Anweisung enthält, und eine Datei namens <i>Tabellenname.txt</i> , die den Inhalt der Tabelle enthält. Die Daten werden in einem Format gespeichert, welches Tabs als Trennzeichen verwendet (wenn <code>--tab</code> angegeben wurde). Andernfalls werden die Daten in einem Format gespeichert, das mit der <code>--fields-terminated-by</code> -Option angegeben wird (bedenken Sie, dass dies nur funktioniert, wenn <code>mysqldump</code> auf dem gleichen Rechner wie der <code>mysql</code> -Daemon läuft).
<code>--fields-terminated-by='Trennzeichen'</code>	Gibt das <i>Trennzeichen</i> an, mit dem die Felder von- einander abgetrennt werden. Wird kein <i>Trennzeichen</i> angegeben, wird als Standardwert das Tabzeichen verwendet (<code>\t</code> = Tab). Wird bei der <code>-T</code> -Option verwendet.
<code>--fields-enclosed-by='Trennzeichen'</code>	Gibt das <i>Trennzeichen</i> an, das benutzt wird, um jedes Feld einzuschließen. Wird bei der <code>-T</code> -Option verwendet.
<code>--fields-optionally-enclosed-by='Trennzeichen'</code>	Gibt das <i>Trennzeichen</i> an, das benutzt wird, um <code>CHAR</code> - und <code>VARCHAR</code> -Felder einzuschließen. Wird bei der <code>-T</code> -Option verwendet.
<code>--fields-escaped-by='Escape-Zeichen'</code>	Gibt das <i>Escape-Zeichen</i> an, welches vor spezielle Zeichen gesetzt wird. Standard ist <code>\\</code> (führt zu einem Backslash). Wird bei der <code>-T</code> -Option verwendet.
<code>--lines-terminated-by='Trennzeichen'</code>	Gibt das <i>Zeilen-Trennzeichen</i> an. Standard ist <code>\n</code> (newline). Vergleichen Sie hierzu auch Tag 8, »Das Füllen der Datenbank mit Daten«, dessen Trennzeichenspezifikationen mit diesen hier übereinstimmen.
<code>-u Benutzername, --user=Benutzername</code>	Ermöglicht Ihnen, einen <i>MySQL-Benutzernamen</i> anzugeben, um mit dem Server in Verbindung zu treten. Andernfalls wird Ihr Unix-Login-Name benutzt.
<code>-p[Passwort], --password[=Passwort]</code>	Übergibt Ihr <i>Passwort</i> bei der Verbindungsaufnahme mit dem <i>MySQL-Server</i> . Beachten Sie, dass Sie kein Leerzeichen zwischen <code>-p</code> und Ihrem <i>Passwort</i> einfügen dürfen. Wenn Sie nur <code>-p</code> oder <code>--password</code> benutzen, fordert <i>MySQL</i> Sie auf, Ihr <i>Passwort</i> an der Konsole einzugeben (verhindert somit also, dass es auf dem Bildschirm sichtbar wird).

Tabelle 19.1: `mysqldump`-Optionen (Forts.)

Option	Bedeutung
<code>-w='Bedingung',</code> <code>--where='Bedingung'</code>	Gibt nur Datensätze in den Dump aus, die der <i>WHERE-Bedingung</i> entsprechen.
<code>-?, --help</code>	Zeigt eine Liste aller Optionen des <code>mysqldump</code> -Programms an und beendet es anschließend wieder.

Tabelle 19.1: `mysqldump`-Optionen (Forts.)



`mysqldump` bietet noch weitere Optionen, wie z.B. Optionen für das Debugging und die Kompression. Wir haben hier nur die nützlichsten aufgeführt. Führen Sie `mysqldump --help` aus, um eine vollständige Liste aller Optionen zu erhalten.



Wenn Sie Berechtigungsprobleme beim Ausführen von `mysqldump` haben, die sich entweder im `Errcode: 13` oder `Errcode: 2` äußern, haben Sie wahrscheinlich eine falsch konfigurierte `UMASK`, wenn der `mysql`-Daemon gestartet wird.

Der Standardeintrag für die `UMASK` ist für gewöhnlich `0660` (oktal). Wie auch immer, Sie können diesen Fehler beseitigen, indem Sie den `mysql`-Daemon mit einer korrekten `UMASK` neu starten:

```
Shellprompt> mysqladmin shutdown
Shellprompt> UMASK=384 #das ist 0600 oktal
Shellprompt> export UMASK
Shellprompt> safe mysqld&
```

Sie sollten jetzt das `mysqldump`-Programm normal benutzen können.

Im Folgenden sehen Sie ein Beispiel, wie Sie `mysqldump` benutzen können. Stellen Sie sich vor, Sie möchten eine Sicherungskopie einer vollständigen Datenbank namens `book` erstellen. Geben Sie Folgendes ein:

```
mysqldump --opt book > /Pfad/nach/book.sql
```

Diese Anweisung würde eine Datei mit folgendem oder ähnlichem Inhalt erzeugen (rufen Sie sich die `Customers`-Tabelle von Tag 11 in Erinnerung):

```
# MySQL dump 8.8
#
# Host: localhost    Database: book
#-----
# Server version 3.23.22-beta-debug
#
# Table structure for table 'customers'
#
DROP TABLE IF EXISTS customers;
CREATE TABLE customers (
```

```

lastname varchar(30) DEFAULT '' NOT NULL,
firstname varchar(30) DEFAULT '' NOT NULL,
address varchar(100) DEFAULT '' NOT NULL,
state varchar(30) DEFAULT '' NOT NULL,
country varchar(30) DEFAULT '' NOT NULL,
KEY lastn_index (lastname(6),state(2),country(3))
);

#
# Dumping data for table 'customers'
#

LOCK TABLES customers WRITE;
INSERT INTO customers VALUES ('Shepherd','Tom','33 Madison Drive,
Oakland','CA','USA'),('Chapman','Frederick','52 Ocean St,
Sacramento','CA','USA'),('Lowe','Justin','3 Hill Walk, Twin
Creeks','WI','USA'),('Spokes','Chris','Red Fern House,
Bradwell','Oxford','UK');
UNLOCK TABLES;
```

Beachten Sie die DROP TABLE-, LOCK TABLES- und UNLOCK TABLES-Anweisungen sowie den Umstand, dass die INSERT-Anweisung eine lange Zeile ist; das sind die Auswirkungen, wenn man die --opt-Option bei dem mysqldump-Programm benutzt.

Alternativ möchten Sie Ihre Datenbank vielleicht in verschiedene voneinander getrennte Dateien abspeichern: eine Datei für die SQL-Anweisungen und eine für die tabseparierten Daten. Geben Sie in diesem Fall Folgendes ein:

```
mysqldump --tab=/root book
```

Diese Anweisung erzeugt zwei Dateien pro Tabelle, eine mit den SQL-CREATE-Anweisungen (customers.sql):

```

# MySQL dump 8.8
#
# Host: localhost    Database: book
#-----
# Server version 3.23.22-beta-debug
#
# Table structure for table 'customers'
#
CREATE TABLE customers (
  lastname varchar(30) DEFAULT '' NOT NULL,
  firstname varchar(30) DEFAULT '' NOT NULL,
  address varchar(100) DEFAULT '' NOT NULL,
  state varchar(30) DEFAULT '' NOT NULL,
```

```
country varchar(30) DEFAULT '' NOT NULL,  
KEY lastn_index (lastname(6),state(2),country(3))  
);
```

Und eine mit den Tabulator-getrennten Daten (customers.txt):

Shepherd	Tom	33 Madison Drive, Oakland	CA	USA
Chapman	Frederick	52 Ocean St, Sacramento	CA	USA
Lowe	Justin	3 Hill Walk, Twin Creeks	WI	USA
Spokes	Chris	Red Fern House, Bradwell	Oxford	UK

Welche Dateien wichtig sind

Ihr gesamtes Datenbank-Design und die zugehörige Geschäftsapplikation bestimmen, welche Teile der Datenbank die wichtigsten sind und wie sie dementsprechend mit Backups oder Datenspeicherung verfahren.

Auf einem System beispielsweise, von dem zum größten Teil Daten abgefragt werden, kann es genügen, diese Daten einmal alle paar Wochen zu sichern. Sie würden wahrscheinlich eine `mysqldump --opt`-Anweisung ausführen, die alle Informationen (Tabellen und Daten) sichern würde, damit Sie die Datenbank später, wenn es nötig ist, wiederherstellen können.

Ein anderes Szenario ist eine »Master-Slave«- geteilte Datenbank, die auf zwei verschiedene Orte verteilt ist. Im Master-Ort liegen die Daten, während am Slave-Ort (vielleicht einem Webserver) das Abfrage-System für den Anwender liegt (um zum Beispiel einen Produktkatalog zu durchsuchen). In diesem Beispiel besteht nicht nur die Notwendigkeit für ein Backup, sondern auch für das Kopieren des Systems auf regulärer Basis von einem Ort zum anderen.

Im zweiten Fall werden Sie `mysqldump` wahrscheinlich viel häufiger aufrufen, vielleicht mit folgender Anweisung, mit der Sie einen anderen MySQL-Server mit Daten füllen:

```
mysqldump --opt Datenbankname -hostname="Datenbankname_auf_anderem_Host"
```

Diese Beispiele führten sowohl das Backup vollständiger Tabellen als auch das von reinen Daten auf. Während ein vollständiges Backup sehr geeignet ist, um Ihre Datenbank wieder vollständig in Stand zu setzen, sind diese Prozeduren jedoch nicht immer die praktikabelsten.

Vollständige Backups werden oftmals auf Bändern, CD-ROMs oder anderen Speichermedien gesichert, die beiseite gelegt werden können. Das regelmäßige Sichern von großen Datenmengen benötigt aber auch viel Speicherplatz, und das Wiederherstellen nach einem geringfügigen Unfall kann unnötig verkompliziert werden. An dieser Stelle ist die inkrementelle Datensicherung sinnvoll.

Ein inkrementelles Backup sichert alle Änderungen, die seit dem letzten vollständigen Backup erfolgt sind. Wenn ein vollständiges Backup einmal pro Woche benötigt wird, kann ein inkrementelles Backup einmal am Tag durchgeführt werden, wobei die Daten auf der lokalen Festplatte gespeichert werden.

In Kürze werden Sie einen näheren Blick auf inkrementelle Datensicherung durch Protokollieren von Aktualisierungen in der Datenbank werfen.

Import und Wiederherstellen von Daten

Das `mysqlimport`-Programm kann eine große Anzahl an Datenformaten einlesen, darunter auch Tabulator-getrennte, und es fügt diese Daten in eine spezifizierte Datenbank ein. Die Syntax ist die folgende:

```
mysqlimport [Optionen] Datenbank Dateiname1 Dateiname2 ...
```

Diese Anweisung kann benutzt werden, um Daten-Dumps, die mit `mysqldump` erzeugt wurden, einzulesen.

Genau wie das `mysqladmin`-Programm wird auch `mysqlimport` von der Kommandozeile aus gestartet. Es bewerkstelligt in der Tat das Gleiche wie die SQL-Anweisung `LOAD DATA INFILE`.

Die spezifizierten Dateinamen (egal in welcher Anzahl) müssen mit den zu erstellenden Tabellennamen übereinstimmen. `mysqlimport` entfernt alle Dateiendungen von den Dateinamen und fügt die Aktualisierungen in die Tabelle mit entsprechendem Namen ein.

Sie können `mysqlimport` mit folgenden Optionen verwenden:

Option	Bedeutung
<code>-d, --delete</code>	Leert die Tabellen, bevor die <i>INSERT</i> -Anweisungen, die in der Datei spezifiziert sind, ausgeführt werden.
<code>-h Hostname, --host Hostname</code>	Importiert die Daten auf den angegebenen <i>Hostnamen</i> statt auf den Standardrechner <i>localhost</i> .
<code>-l, --lock-tables</code>	Sperrt alle Tabellen, bevor die <i>INSERT</i> -Anweisungen ausgeführt werden.
<code>-u Benutzername, --user Benutzername</code>	Übergibt den MySQL-Benutzernamen, wenn mit dem Server Verbindung aufgenommen wird.
<code>-p [Passwort], --password[=Passwort]</code>	Wird normalerweise in Zusammenhang mit der Option <code>-u</code> verwendet und übergibt dann das zugehörige <i>Passwort</i> bei der Verbindungsaufnahme. Beachten Sie, dass keine Leerzeichen zwischen <code>-p</code> und dem <i>Passwort</i> stehen dürfen. Wenn Sie kein <i>Passwort</i> angeben, werden Sie von MySQL auf der Konsole dazu aufgefordert.

Tabelle 19.2: `mysqlimport`-Optionen

Option	Bedeutung
-r, --replace und -i, --ignore	Ermöglicht Ihnen die Kontrolle über Datensätze, die einzigartige Schlüsselwerte haben («unique keys»). -r oder --replace bewirkt, dass importierte Datensätze bestehende Datensätze überschreibt, wenn Sie die gleichen einzigartigen Schlüsselwerte haben. -i oder --ignore bewirkt, dass solche Datensätze übergangen, und nicht importiert werden. Wenn keine dieser beiden Optionen angegeben ist und ein solcher doppelter Datensatz auftaucht, wird eine Fehlermeldung ausgegeben und der Import abgebrochen (beachten Sie, dass, wenn Sie eine DROP TABLE-Anweisung zu Beginn des Imports ausführen, das Auftauchen einer solchen Situation sehr unwahrscheinlich ist).
-s, --silent	Unterdrückt MySQLs Nachrichten und gibt nur Fehlermeldungen an der Standardausgabe aus.
--fields-terminated-by="Trennzeichen"	Gibt das Zeichen an, welches als Trenner nach jedem Feld benutzt wird. Standard ist hier der Tabulator \t.
--fields-enclosed-by="Trennzeichen"	Spezifiziert das Trennzeichen, welches jedes Feld umschließt.
--fields-optionally-enclosed-by="Trennzeichen"	Spezifiziert das Trennzeichen, welches CHAR- oder VARCHAR-Felder umschließt.
--fields-escaped-by="Escape-Zeichen"	Gibt das Escape-Zeichen an, welches vor spezielle Zeichen gesetzt wird. Standard ist \ (führt zu einem Backslash).
--lines-terminated-by='Trennzeichen'	Gibt das Zeilen-Trennzeichen an. Standard ist \n (newline).
?, --help	Zeigt eine Liste aller Optionen des mysqlimport-Programms an und beendet es anschließend wieder.

Tabelle 19.2: mysqlimport-Optionen (Forts.)



Beachten Sie, dass diese Tabelle keine ausführliche Liste aller Optionen des mysqlimport-Programms ist. Geben Sie `mysqlimport --help` für eine vollständige Liste aller Optionen ein.

Auf folgende Weise könnten Sie zum Beispiel die Daten der Customers-Tabelle in die Beispiel-Datenbank `book` einfügen:

```
mysqlimport -u root -pPasswort book customers.txt
```

Diese Anweisung importiert die Daten aus der Datei `customers.txt` in die `book.Customers`-Tabelle und produziert folgende kurze Ausgabe, um zu bestätigen, was ausgeführt wurde:

```
book.Customers: Records: 4 Deleted: 0 Skipped: 0 Warnings: 0
```

Protokollieren von Transaktionen

MySQL bietet einige handliche Wege, um Transaktionen in Ihrer Datenbank zu protokollieren. Lesen Sie weiter, um zu sehen, *weshalb* und *wie* Sie so etwas ausführen sollten.

Protokollieren von Aktualisierungen, um Backup-Strategien zu unterstützen

Wie schon im vorherigen Abschnitt angedeutet, brauchen Sie in bestimmten Situationen eine Backup-Strategie, die aus einer Kombination aus vollständiger und inkrementeller Datensicherung besteht.

Jedes inkrementelle (tägliche) Backup könnte dabei auf der lokalen Festplatte gespeichert werden, während das (wöchentliche) vollständige Backup auf ein anderes Speichermedium transferiert würde. Sie sollten jedes der täglichen, inkrementellen Backups behalten (anstatt die Datei vom Vortag jeweils zu überschreiben). Wenn nun die Notwendigkeit einer Wiederherstellung einer Datenbank eintreten sollte, würde ein Administrator zunächst das aktuellste vollständige Backup zurückspielen und anschließend die Änderungen aus dem aktuellsten inkrementellen (täglichen) Backup.

Diese Strategie führt dazu, dass die täglichen Backups nie älter als einen Tag sind und somit der Speicherplatzverbrauch auf ein Minimum reduziert wird und zusätzlich eine vollständige Sicherung einmal in der Woche erfolgt.

Obwohl Sie `mysqldump` für das vollständige Backup verwenden, würden Sie für ein inkrementelles Backup ein anderes Werkzeug benutzen. Wenn Sie den `mysqld`-Daemon mit der `--log-update`-Option starten, veranlassen Sie MySQL, jede Veränderung in der Datenbank zu protokollieren. MySQL erstellt also ein so genanntes Update-Log in Form von SQL-Anweisungen, das benutzt werden kann, um alle Änderungen an der Datenbank zu replizieren.

Diese Logs (Protokolle) sind als *Hostname.n* benannt, wobei *n* jedes Mal, wenn eine neue Log-Datei gestartet wird, um eins erhöht wird. Eine neue Log-Datei wird immer dann gestartet, wenn Sie eine der folgenden Anweisungen ausführen:

- ▶ `mysqladmin refresh`
- ▶ `mysqladmin flush-logs`
- ▶ `mysqldump --flush-logs` mit Optionen
- ▶ `FLUSH LOGS`
- ▶ Serverstart oder Neustart

Anhand des Update-Logs, das eine Liste von SQL-Anweisungen beinhaltet (ähnlich einem MySQL-Dump), kann der Administrator auch nur einzelne Änderungen wiederherstellen.

Folgendes ist bei einer solchen Backup-Strategie die Abfolge der Ereignisse:

1. Starten Sie den MySQL-Server mit der Protokollierungsoption: `safe mysqld -log-update &`
2. Führen Sie das `mysqldump`-Programm aus, um (wöchentlich) ein vollständiges Backup zu erhalten. Geben Sie die `--flush-logs`-Option an, damit die Log-Dateien automatisch geleert werden, wenn ein Dump erstellt wird.
3. Die Log-Datei protokolliert jetzt jede Änderung, die in der Datenbank und ihren Daten stattfindet. Zu jeder Zeit existiert eine (oder mehrere) komplette Log-Datei seit dem letzten vollständigen Dump. Für noch mehr Sicherheit können Sie diese Dateien auch auf einem externen Speichermedium ablegen.

Weitere Protokollierungsarten

Sie haben schon gesehen, wie MySQL alle Änderungen, wie `UPDATES`, `INSERTs` oder `DELETES` an einer Datenbank protokollieren kann, wenn Sie den `mysqld`-Daemon mit der `--log-update`-Option starten. Es erstellt dann eine Textdatei, die jede SQL-Anweisung, die an die Datenbank gegeben wurde, enthält.

Sie können den `mysqld`-Daemon auch mit der `--log`-Option starten, die MySQL veranlasst, ein »main-activity«-Protokoll zu generieren. Dieses protokolliert fast alles – `SELECT`-Anfragen ebenso wie Aktualisierungen. Die Log-Datei wird normalerweise in `/var/lib/mysql/Hostname.log` gespeichert, wobei `Hostname` der Name des Rechners ist (der Speicherort kann bei älteren Systemen auch `/usr/local/var` sein).

Die `--log`-Option beim Starten des `mysqld`-Daemons kann nützlich sein, wenn Sie Fehler (Bugs) in Ihrer Applikation diagnostizieren möchten. Wenn Sie Anzeichen dafür haben, dass ungültige Anfragen an die Datenbank gestellt werden, können Sie die Log-Datei durchforsten und nachsehen, welche Anfrage eine Applikation ausgeführt hat und wann das geschehen ist.

Wenn Sie wirklich detaillierte Informationen über das, was MySQL bewerkstelligt, benötigen (zum Beispiel für das Debugging von System-Problemen auf tieferer Ebene), können Sie den `mysqld`-Daemon mit der Option `--debug` starten. Es gibt zahlreiche Unteroptionen für den Debugging-Modus, mit denen Sie Aktivitäten von MySQL sehr genau verfolgen können.

Um das Protokollieren auszuschalten, fahren Sie den `mysqld`-Daemon herunter und starten ihn ohne (Protokollierungs-)Optionen erneut (bzw. benutzen `safe mysqld`).

Verschieben einer MySQL-Datenbank

Es gibt verschiedene Gründe, eine MySQL-Datenbank an einen anderen Ort zu verlagern: wenn Sie ein System-Upgrade durchführen, Ihre Datenbank einfach auf ein anderes System transferieren wollen oder sogar eine »Master«-Kopie Ihrer Datenbank an einem sicheren Ort haben und von dieser periodisch eine Kopie ziehen möchten, um sie zum Beispiel auf Ihrem Webserver oder einem anderem System verfügbar zu machen. Was immer auch der Grund ist: die Vorgehensweise ist immer die gleiche und umfasst diese drei grundlegenden Schritte: Speichern, Transferieren und Wiederherstellen.

Sie haben bereits eine Methode, um Ihre Datenbank zu sichern, kennen gelernt – das `mysqldump`-Programm. Dieses ist die bevorzugte Methode. Es ist auch möglich, eine Datenbank direkt aus dem Dateisystem heraus zu sichern, indem Sie die `.frm`-, `.ISD`- und `.ISM`-Dateien bearbeiten. Diese Methode stellt jedoch die Datenkonsistenz nicht sicher und ist daher nicht zu empfehlen.

Ihre Datenbank kann mit `tar` auf Bändern gesichert und mit `ZIP` komprimiert werden, um den Transfer zu beschleunigen.

Auf dem Rechner, auf dem die Datenbank wiederhergestellt werden soll, `unzip`-en und `untar`-en Sie die Datei wieder. Danach sind Sie bereit, die Datenbank wiederherzustellen.

Wenn Sie den MySQL-Dump mit `mysqldump --tab` oder ähnlichen Optionen erstellt haben, haben Sie nun eine Anzahl von Dateien mit SQL-Anweisungen und Daten. Alternativ dazu haben Sie, wenn Sie den Dump mit `mysqldump --opt` erstellt haben, eine einzige große Datei, sowohl mit `SQL-CREATE`-Anweisungen als auch den Daten in Form von `INSERT`-Anweisungen. Um Ihre Datenbank unter Unix oder Linux wiederherzustellen, führen Sie folgende Anweisung aus:

```
cat /home/mydata/dump.txt | /usr/local/mysql/bin/mysql -u Benutzername -pPasswort
                                Datenbankname
```

Der `cat`-Befehl zusammen mit `|` (pipe) ist eine sehr mächtige Kombination. In nur einer einzigen Zeile veranlassen Sie `cat`, die Datei zu lesen und diese nach `mysql` zu schicken, was durch die Pfadangabe auf dem System spezifiziert ist. In der Regel müssen Sie die Parameter `-u` und `-p` sowie den Namen der Zieldatenbank angeben.

Systemeigene Funktionen von MySQL

MySQL verfügt über eine Anzahl an Systemfunktionen, die Zugang zu administrativen Aspekten Ihrer Datenbank bieten, welche Sie innerhalb Ihrer Datenbank aufrufen können.

Die Befehle sind in ihrem Umfang relativ breit gefächert und reichen von Datenbank- und Benutzer-relevanten Informationen bis hin zu Encoding- und Encryption-Methoden.

DATABASE()

`DATABASE()` gibt den Namen der aktuellen Datenbank zurück, mit welcher Sie in Verbindung stehen. Wenn Ihre Datenbank zum Beispiel *mydb* heißt, würde

```
mysql> SELECT DATABASE();
```

folgende Ausgabe produzieren:

```
+-----+
| DATABASE() |
+-----+
| mydb       |
+-----+
1 row in set (0.16 sec)
```

USER(), SYSTEMUSER(), SESSION_USER()

Diese Funktionen geben den Namen des aktuellen Datenbank-Benutzers zurück, einschließlich des Hostnamens. In der Regel geben die Funktionen etwas, das ungefähr folgendermaßen aussieht, zurück:

```
tony@localhost
```

VERSION()

`VERSION()` gibt die verwendete Version von MySQL zurück.

PASSWORD(String)

Die `PASSWORD(String)`-Funktion verschlüsselt einen übergebenen String und gibt das verschlüsselte Ergebnis zurück, wie es in der `password`-Spalte der MySQL-user-Tabelle vorkommt.

Zum Beispiel gibt

```
mysql> SELECT PASSWORD("meinPasswort");
```

Folgendes aus:

```
+-----+
| PASSWORD("meinPasswort") |
+-----+
| 5e20fda637d6669f        |
+-----+
1 row in set (2.26 sec)
```

Beachten Sie, dass `PASSWORD()` nicht notwendigerweise genauso arbeitet wie `ENCRYPT()`, welches die `Unix-crypt()`-Funktion verwendet. Ein durch `PASSWORD()` verschlüsselter String kann sich von dem gleichen String unterscheiden, wenn er mit `ENCRYPT()` verschlüsselt wurde.

ENCRYPT(string[,salt])

`ENCRYPT()` verschlüsselt einen übergebenen String mit einem optionalen zwei Zeichen langen *Salt* (spätere Versionen von MySQL akzeptieren längere *Salts*).

Verschlüsselung konvertiert einen Text in ein für den Menschen nicht lesbares Format. Daher sollten Sie private Informationen auch so verschlüsseln, dass sie nicht von anderen gelesen werden können. Es gibt keine Möglichkeit, diese Daten wieder zu entschlüsseln.

Der *Salt* »würzt« den Verschlüsselungs-Algorithmus. Wird dieser nicht angegeben, wird ein Zufalls-*Salt* verwendet, und das Verschlüsselungsergebnis ist nicht vorhersehbar. Wenn ein *Salt* jedoch angegeben wird, wird das Ergebnis jedes Mal das gleiche sein, wenn ein *String* mit diesem *Salt* verschlüsselt wird.

MySQL benutzt den `Unix-crypt()`-Systemaufruf. Ist die `crypt()`-Funktion auf Ihrem System nicht verfügbar, wird `NULL` als Ergebnis zurückgegeben.

Betrachten Sie nun, wie `ENCRYPT()` sich ohne Angabe eines *Salt* verhält:

```
mysql> SELECT ENCRYPT("Etwas_ganz_geheim");
+-----+
| ENCRYPT("Etwas_ganz_geheim") |
+-----+
| g/14mcWjhTRpI              |
+-----+
1 row in set (0.16 sec)
```

Führen Sie nun genau die gleiche Anweisung noch einmal aus:

```
mysql> SELECT ENCRYPT("Etwas_ganz_geheim");
```

Dieses Mal ist das Ergebnis ein anderes:

```
+-----+
| ENCRYPT("Etwas_ganz_geheimes") |
+-----+
| 11uEXTreCBuKE |
+-----+
1 row in set (0.01 sec)
```

Wenn Sie einen *Salt* angeben, können Sie das Ergebnis vorhersagbar machen. Es gibt zwar keine Entschlüsselungsmethode, aber Sie können überprüfen, ob ein verschlüsselter String mit einem unverschlüsselten, der gegen dieses getestet wird, übereinstimmt. Haben Sie zum Beispiel ein verschlüsseltes Passwort in Ihrer Datenbank, ohne allerdings davon eine »nur Text«-Version zu besitzen, können Sie immer noch ausprobieren, ob ein Test-Passwort mit dem abgespeicherten übereinstimmt.

Bleiben wir bei dem zweiten Beispiel. Benutzen Sie die ersten zwei Zeichen des verschlüsselten Strings. Verschlüsseln Sie den String erneut (woraus sich das neue Test-Passwort ergeben kann):

```
mysql> SELECT ENCRYPT("Etwas_ganz_geheimes", "11");
```

Sie haben »11« von dem verschlüsselten Text genommen und erhalten folgende Ausgabe:

```
+-----+
| ENCRYPT("Etwas_ganz_geheimes", "11") |
+-----+
| 11uEXTreCBuKE |
+-----+
1 row in set (0.02 sec)
```

Wenn Sie die beiden Ergebnisse nun vergleichen, werden Sie feststellen, dass Sie übereinstimmen.

ENCODE(string, password), DECODE(encoded_string, password)

Die `ENCODE()`-Funktion verschlüsselt einen übergebenen *string* mit *password* als Zugangs-Passwort und gibt einen binären String zurück.

`DECODE()` bewirkt das Gegenteil; es nimmt den binären *encoded_string* und entschlüsselt ihn mit *password*.

Ein Beispiel zur Verschlüsselung mit `ENCODE()`:

```
mysql> SELECT ENCODE("Mein_String","Mein_Passwort");
```

```
+-----+
| ENCODE("Mein_String","Mein_Passwort") |
+-----+
| ¡[Ê $·!>-¶s |
+-----+
```

```
1 row in set (0.00 sec)
```

Ein Beispiel zur Entschlüsselung mit `DECODE()`:

```
mysql> SELECT DECODE(ENCODE("Mein_String","Mein_Passwort"),"Mein_Passwort");
```

```
+-----+
| DECODE(ENCODE("Mein_String","Mein_Passwort"),"Mein_Passwort") |
+-----+
| Mein_String |
+-----+
```

```
1 row in set (0.00 sec)
```

Das Entschlüsseln mit einem falschen Passwort gibt statt eines Fehlers ein falsches Ergebnis zurück. Zum Beispiel gibt

```
mysql> SELECT DECODE(ENCODE("Mein_String","Mein_Passwort"),"Falsches_Passwort");
```

folgende Ausgabe zurück:

```
+-----+
| DECODE(ENCODE("Mein_String","Mein_Passwort"),"Falsches_Passwort") |
+-----+
| 0YÀ-ÃÖT_5*E |
+-----+
```

```
1 row in set (0.00 sec)
```

Beachten Sie, dass das Ergebnis von `ENCODE()`, da es binär vorliegt, in einem `BLOB`-Datentyp gespeichert werden muss.

MD5(string)

`MD5()` gibt eine MD5-Checksum für einen gegebenen String zurück. Diese ist eine 32 Zeichen lange hexadezimale Nummer.

LAST_INSERT_ID([Ausdruck])

Diese Funktion gibt den letzten Wert, der in eine `AUTO_INCREMENT`-Spalte durch automatische Generierung eingefügt wurde, zurück. Sie bezieht sich nur auf die Werte, die innerhalb der aktuellen Verbindung erstellt wurden.

Datenbank-Management mit *mysqladmin*

Das *mysqladmin*-Programm wird benutzt, um eine große Anzahl an administrativen Aufgaben in einer MySQL-Datenbank zu bearbeiten. Es wird im Allgemeinen auf Kommandozeilenebene ausgeführt.

Wenn Sie das Programm aufrufen, sollten Sie *mysqladmin* einige Optionen und Befehle übergeben, die es anweisen, was es wie ausführen soll.

Wie *mysqladmin* benutzt wird

Die Syntax für das *mysqladmin*-Programm lautet wie folgt:

```
mysqladmin [Optionen] Befehl1 [Befehls_Option1] Befehl2 [Befehls_Option2]
```

Beachten Sie, dass Sie in einigen Fällen *mysqladmin* den Pfad zum `mysql/bin`-Verzeichnis voranstellen müssen.

Ebenso müssen Sie, nachdem Sie Benutzer und Benutzerrechte für eine Datenbank vergeben haben, die `-p`- und/oder `-u`-Option beim Aufruf von *mysqladmin* verwenden, wenn Ihr MySQL-Benutzername sich von Ihrem Unix-Namen unterscheidet. Dies zeigt MySQL an, dass Sie einen Benutzernamen und ein Passwort anbieten; tun Sie dies nicht, obwohl es von MySQL verlangt wird, erhalten Sie eine Fehlermeldung.

Zusammenfassung: Sie können folgende Befehle mit *mysqladmin* verwenden:

Befehl	Bedeutung
create <i>Datenbankname</i>	Erstellt eine neue Datenbank mit dem angegebenen Namen.
drop <i>Datenbankname</i>	Löscht die spezifizierte Datenbank.
Status	Gibt eine kurze Statusmeldung des Servers zurück.
Version	Gibt Informationen über die Version des Servers zurück.
extended-status	Gibt eine ausführliche Statusmeldung des Servers zurück.
Variables	Zeigt verfügbare Variablen an.
Processlist	Zeigt eine Liste aktiver Threads auf dem Server an. Nützlich, um zu sehen, wie beschäftigt der Server ist.
flush-hosts	Leert alle Hosts aus dem Cache.
flush-logs	Leert alle Log-Dateien.
flush-tables	Leert alle Tabellen.
flush-privileges	Lädt die GRANT-Tabellen neu.

Tabelle 19.3: *mysqladmin*-Befehle

Befehl	Bedeutung
kill <i>id1</i> , <i>id2</i>	Beendet MySQL-Threads.
password <i>neues_Passwort</i>	Ändert das alte Passwort in das neue Passwort.
Ping	Sendet ein »ping«-Signal zum <code>mysqld</code> -Daemon, um zu überprüfen, ob er noch läuft.
Reload	Lädt die GRANT-Tabellen neu.
Refresh	Lädt die GRANT-Tabellen neu und schließt offene Log-Dateien.
Shutdown	Führt den MySQL-Server herunter.

Tabelle 19.3: `mysqladmin`-Befehle (Forts.)

Neben den Befehlen können `mysqladmin` auch Optionen übergeben werden, die das Programm anweisen, wie es die Befehle auszuführen hat.

Optionen	Bedeutung
-#, --debug=	Gibt in eine Debug-Log-Datei aus; oftmals ist dies »d:t:o, Dateiname«.
-f, --force	Das Löschen einer Datenbank kann eine Warnung ausgeben, genauso wie einige andere Befehle; --force veranlasst, dass die Anweisung weiter ausgeführt wird, wenn eine solche Warnung auftritt.
-C, --compress	Benutzt Komprimierung bei Server/Client-Protokollen.
-h, --host= <i>Hostname</i>	Verbindet mit einem spezifizierten Host.
-p, --password[= <i>Passwort</i>]	Gibt ein <i>Passwort</i> an, das bei der Verbindungsaufnahme mit dem Server verwendet wird. Wenn Sie das <i>Passwort</i> nicht angeben, werden Sie an der Kommandozeile dazu aufgefordert.
-P, --port[= <i>Portnummer</i>]	<i>Portnummer</i> , die für die Verbindung verwendet wird.
-i, --sleep= <i>x</i>	Veranlasst <code>mysqladmin</code> , einen Befehl immer wieder auszuführen, mit einer Pause von <i>x</i> Sekunden.
-s, --silent	Veranlasst <code>mysqladmin</code> , sich ohne Ausgabe zu beenden, wenn ein Verbindungsaufbau nicht möglich war.
-S, --socket= <i>Socket</i>	<i>Socket</i> -Datei, welche für die Verbindungsaufnahme verwendet wird.
-t, --timeout= <i>x</i>	Spezifiziert den Timeout in Sekunden für die Verbindungsaufnahme mit dem MySQL-Server.

Tabelle 19.4: `mysqladmin`-Optionen

Optionen	Bedeutung
-u, --user= <i>Benutzername</i>	Gibt den <i>Benutzernamen</i> an, wenn dieser von dem aktuellem Systembenutzer abweicht.
-V, --version	Gibt Versionsinformationen aus und beendet das Programm.
-w, --wait[= <i>x</i>]	Wartet und versucht <i>x</i> -mal erneut, wenn die Datenbankverbindung nicht vorhanden ist.
-, --help	Zeigt alle Befehle und Optionen und beendet das Programm.

Tabelle 19.4: mysqladmin-Optionen (Forts.)

Da einige dieser Optionen ziemlich kompliziert sind, sollten Sie sie genauer betrachten.

mysqladmin create Datenbankname

Wenn Sie MySQL gerade erst installiert haben und/oder eine neue Datenbank erstellen möchten, können Sie das mit `mysqladmin create` tun.

Betrachten Sie dieses Beispiel, indem Sie sich als `root` auf dem MySQL-Server einloggen und eine neue Datenbank für Ihre Fotografien erstellen:

```
$> mysqladmin -u root -p create photo_db;
Enter password: *****
Database "photo_db" created.
```

Zu diesem Zeitpunkt ist die MySQL-Datenbank nichts weiter als ein leeres Verzeichnis – `mysqladmin` hat lediglich ein `photo_db`-Verzeichnis auf dem Server erstellt.

mysqladmin drop Datenbankname

Sollten Sie jemals eine Datenbank löschen wollen, können Sie dies mit dem `mysqladmin drop`-Befehl tun.

Stellen Sie sich vor, Sie wollten die gerade eben erstellte Datenbank `photo_db` wieder loswerden:

```
$> mysqladmin -u root -p drop photo_db;
Enter password: *****
Dropping the database is potentially a very bad thing to do.
Any data stored in the database will be destroyed.
```

```
Do you really want to drop the 'photo_db;' database [y/N]
y
Database "photo_db" dropped
```

Im Gegensatz zu `create` hat `mysqladmin` nun das Verzeichnis, das die `photo_db`-Datenbank und ihren gesamten Inhalt enthält, gelöscht.

Status- und Versionsinformationen

`mysqladmin` bietet eine Anzahl von Möglichkeiten, um sich Status- und Versionsinformationen über den MySQL-Server anzeigen zu lassen:

- ▶ `mysqladmin status`
- ▶ `mysqladmin version`
- ▶ `mysqladmin extended-status`
- ▶ `mysqladmin variables`

Die erste Möglichkeit stellt eine einzeilige Statusmeldung über den MySQL-Server zur Verfügung (zur besseren Veranschaulichung ist die Ausgabe hier auf mehrere Zeilen verteilt):

```
user% mysqladmin status
```

```
Uptime: 4257009
Threads: 10
Questions: 4550528
Slow queries: 14
Opens: 408
Flush tables: 1
Open tables: 59
```

Betrachten Sie nun, was die einzelnen Angaben bedeuten:

- ▶ **Uptime:** Die Zeit in Sekunden, die der MySQL-Server schon läuft.
- ▶ **Threads:** die Anzahl der Threads bzw. Clients, die in diesem Moment aktiv sind. Diese ist immer mindestens 1, da der Wert den Thread, der den `mysqladmin`-Befehl enthält, auch zählt.
- ▶ **Questions:** die Anzahl der Anfragen, die der `mysqld`-Daemon erhalten hat, seitdem der Server läuft.
- ▶ **Slow queries:** die Anzahl der Anfragen, deren Beantwortung mehr als eine bestimmte Zeit gedauert hat. Diese Zeit wird mit `long_query_time` spezifiziert, die mit dem `mysqladmin extended-status`-Befehl angezeigt werden kann.
- ▶ **Opens:** die Anzahl der Tabellen, die seit dem Start des MySQL-Servers geöffnet wurden.
- ▶ **Flush tables:** die Anzahl der `flush`-, `reload`- und `refresh`-Anweisungen, die seit dem Start des Servers ausgeführt wurden.
- ▶ **Open tables:** die Anzahl der Tabellen, die aktuell geöffnet sind.

Bei einigen Versionen erhalten Sie noch zusätzlich folgende Informationen:

- ▶ Queries per second avg: Anzahl der Anfragen pro Sekunde im Schnitt.
- ▶ Memory in use: aktueller Speicherplatzverbrauch.
- ▶ Max memory used: maximaler Speicherplatzverbrauch seit dem Start des MySQL-Servers.

Werfen Sie nun einen Blick auf eine etwas benutzerfreundlichere Anweisung, `mysqladmin version`:

```
% mysqladmin version

/usr/local/bin/mysqladmin Ver 8.0 Distrib 3.22.32, for unknown-freebsdelf3.3 on i386
TCX Datakonsult AB, by Monty

Server version      3.22.32
Protocol version    10
Connection          Localhost via UNIX socket
UNIX socket         /tmp/mysql.sock
Uptime:             49 days 6 hours 27 min 59 sec

Threads: 10
Questions: 4550463
Slow queries: 14
Opens: 408
Flush tables: 1
Open tables: 59
```

Viele dieser Informationen, vor allem die letzten Zeilen, entsprechen denen, die Sie schon von der `status`-Ausgabe her kennen. Der einzige Unterschied besteht darin, dass Uptime nun in Stunden, Minuten und Sekunden angegeben wird. Es werden auch noch andere nützliche Informationen angezeigt.

Die Ausgabe gibt Informationen über die Version des laufenden MySQL-Servers sowie über das Kommunikations-Protokoll aus. Sie könnten diese Informationen für das Debugging benötigen, wenn Sie eine Software benutzen, die das MySQL-Kommunikations-Protokoll direkt verwendet.

Die Ausgabe enthält auch Informationen darüber, wie man mit MySQL verbunden ist – in diesem Fall mittels eines Unix-Sockets, da der Zugriff auf den Server via SSH erfolgte.

Daher wird auch der Unix-Socket spezifiziert. Wenn Sie mit MySQL via TCP/IP verbunden sind (zum Beispiel auf Windows), würden Sie statt dessen die TCP-Port-Nummer angezeigt bekommen.

Es gibt zwei weitere Befehle, `mysqladmin extended-status` und `variables`, die noch detaillierte Informationen über den Server-Status ausgeben.

Thread-Informationen

Sie können sehen, was MySQL aktuell ausführt, wenn Sie `mysqladmin` mit dem Befehl `processlist` ausführen:

```
%> mysqladmin -u root -p processlist
Enter password: *****
```

Id	User	Host	db	Command	Time	State	Info
13	root	localhost		Query	0		show processlist

Beachten Sie, dass ein MySQL-Prozess, wie er in der Prozessliste gezeigt wird, sich von einem Unix-Prozess unterscheidet. Wenn Sie unter Unix das `ps`-Kommando ausführen, scheint es vielleicht so, als ob nur ein einziger MySQL-Prozess am Laufen ist, doch tatsächlich hat MySQL selbst doch viele Threads laufen, wie die `mysqladmin processlist` anzeigt.

Die Liste enthält ein `Id` für jeden Thread, den Benutzer, der dieses Thread initiiert hat und den Host, von welchem aus der Benutzer in Verbindung steht. `db` ist der Name der Datenbank, mit der ein Benutzer verbunden ist.

`Command` gibt Informationen über den Typ des Befehls der von einem Thread ausgeführt wird. Es kann einer der folgenden Typen sein:

- ▶ *Sleep*: Der Thread wartet auf eine Benutzereingabe. Das ist der normale Zustand eines Threads.
- ▶ *Processes*: Der Thread betrachtet gerade die andern Threads.
- ▶ *Connect*: Der Thread empfängt gerade eine eingehende Verbindung von einem Client.
- ▶ *Init DB*: Der Thread initialisiert gerade die Datenbank und bereitet sie für die Nutzung vor (beachten Sie, dass ein Thread zwischen verschiedenen Datenbanken wechseln kann).
- ▶ *Query*: Der Thread führt gerade eine Anfrage aus. Sie sehen diesen Typ selten, da eine Anfrage in der Regel nur eine sehr kurze Zeit braucht.
- ▶ *Field list*: Der Thread generiert gerade eine Liste aller Felder einer Tabelle.
- ▶ *Create DB*: Der Thread erstellt gerade eine neue Datenbank.
- ▶ *Drop DB*: Der Thread löscht gerade eine Datenbank.

- ▶ *Reload*: Der Thread lädt gerade die MySQL-Zugangstabellen.
- ▶ *Refresh*: Der Thread setzt gerade die Log-Dateien zurück und leert den Cache.
- ▶ *Statistics*: Der Thread generiert gerade Statistiken.
- ▶ *Kill*: Der Thread beendet gerade einen anderen Thread.
- ▶ *Shutdown*: Der Thread ist dabei, alle anderen Threads zu löschen, und bereitet den Server für das Herunterfahren vor.
- ▶ *Quit*: Der Thread ist gerade dabei, beendet zu werden.

Reparieren einer beschädigten Datenbank

Eine vermutete Beschädigung an einer Datenbank muss nicht immer gleich eine Katastrophe sein. MySQL hat sich dieses Problems angenommen und stellt einige Werkzeuge zur Verfügung, um beschädigte Tabellen wieder zu reparieren.

Dies sind im Wesentlichen die zwei Programme `isamchk` und `myisamchk`.

Bevor Sie lernen, wie und wann diese Werkzeuge eingesetzt werden, werfen Sie einen kurzen Blick auf die Art, wie MySQL Daten speichert.

MySQLs Datenstruktur

Zunächst wechseln Sie in das `data`-Verzeichnis der kleinen `book`-Datenbank und betrachten, welche Dateien hier vorhanden sind:

```
% ls -la
total 28
drwx----- 2 mysqlmysql4096 Feb 14 13 :15 .
drwxr-xr-x11 mysqlroot4096 Feb 15 13 :24 ..
-rw-rw---- 1 mysqlmysql11015 Feb 14 15 :16 customers.ISD
-rw-rw---- 1 mysqlmysql12048 Feb 14 15 :16 customers.ISM
-rw-rw---- 1 mysqlmysql18682 Jan 31 19 :13 customers.frm
```

Es gibt nur eine Tabelle in der `book`-Datenbank: die `Customers`-Tabelle. Wie Sie sehen, hat `Customers` drei Dateien.

Die `.frm`-Datei ist eine Formatdatei, die die Datenstruktur selbst beinhaltet. Versuchen Sie gar nicht erst, diese Datei anzusehen – das Format ist nicht lesbar.

Die `.ISD`-Datei beinhaltet die aktuellen Daten, während die `.ISM`-Datei Informationen zu Schlüssel und anderen internen Referenzen beinhaltet. Das ist die Datei, um welche sich die Programme `isamchk` und `myisamchk` am meisten kümmern, wenn es ums Reparieren geht.

Führen Sie `isamchk -d` aus, um Informationen über die `Customers`-Tabelle angezeigt zu bekommen:

```
% isamchk -d customers
ISAM file: customers
Data Records: 15
Deleted blocks: 5
Recordlength: 220
Record Format: Packed
table description:
Key   StartLenIndex Type
1     1 6 multip. text
      1612      text
      1913      text
```



Beachten Sie, dass Sie sich zum Ausführen des Programms im richtigen Verzeichnis befinden müssen, wie z.B. `/var/lib/mysql/Datenbankname/`, oder explizit den vollständigen Pfad zu den Tabellen angeben müssen. Sie können auch die `*`-Wildcard einsetzen, um alle Tabellen einer Datenbank zu überprüfen.

Diese Ausgabe zeigt Ihnen an, dass die Tabelle 15 Datensätze (Records) und fünf gelöschte Blöcke hat. Die letzte Angabe ist verschwendeter Platz auf Ihrer Festplatte.

Festplattenplatz wiedergewinnen

Sie können den im vorherigen Beispiel angezeigten »verschwendeten« Platz auf Ihrer Festplatte wiedergewinnen, wenn Sie `isamchk -r` ausführen.

```
% isamchk -r customers
- recovering ISAM-table 'customers.ISM'
Data records: 15
- Fixing index 1
```

Die Anweisung überprüft die Tabelle und erstellt diese neu, ohne unnötig Platz zu verschwenden. In dem konkreten Beispiel können Sie sehen, wenn Sie unter Linux `ls -l` ausführen, dass die Datei `customers.ISD` nun lediglich noch 756 Byte statt 1015 Byte wie vor dem Ausführen von `isamchk -r` benötigt.

Es ist anzuraten, `isamchk -d` regelmäßig auf schnell wachsenden Datenbanken auszuführen, um im Auge zu behalten, wie viel Platz in den Tabellen noch zur Verfügung steht. Dies ist besonders wichtig, wenn Ihre Applikation eine große Anzahl von `DELETE`-Anweisungen ausführt, die im Regelfall große Löcher zwischen Ihren Daten auf der Festplatte reißt. Wann immer die Angabe zu den *deleted Blocks* zu hoch oder auch ein signifikanter Teil der *Data records* zu hoch wird, sollten Sie `isamchk -r` ausführen, um den verschwendeten Platz auf Ihrer Festplatte zurückzugewinnen.



Wenn ein Client während der Ausführung von `isamchk` auf Ihre Datenbank zugreift, könnte das `isamchk` in die Irre führen, da das Programm annehmen könnte, dass die Tabellen beschädigt sind, obwohl das nicht der Fall ist. Jede laufende Änderung kann von `isamchk` als Beschädigung gedeutet werden. Daher kann es sehr gefährlich sein, `isamchk` in einem solchen Zeitpunkt auszuführen.

Um dieses zu vermeiden, sollten Sie den `mysqld`-Daemon vor dem Ausführen von `isamchk` herunterfahren. Alternativ können Sie auch sicherstellen, dass niemand in dem Moment Zugriff auf eine der Tabellen hat, und führen einfach `mysqladmin --flush-tables` vor `isamchk` aus.

Wenn Sie vorhaben, Tabellen zu reparieren oder zu optimieren, ist es sehr wichtig, dass Sie den `mysqld`-Daemon herunterfahren, bevor sie `isamchk` ausführen. Wenn Sie den `mysqld`-Daemon nicht für eine Weile herunterfahren können, sollten Sie zumindest `mysqladmin --flush-tables` ausführen und Anwender davon abhalten, auf die Datenbank zuzugreifen, bis die Reparaturen bzw. Optimierungen abgeschlossen sind.

Reparieren beschädigter Tabellen mit `isamchk`

Die allgemeine Syntax von `isamchk` lautet:

```
isamchk [Optionen] Tabellename
```

Während `isamchk` auf `.ism`- bzw. `.isd`-Dateien Anwendung findet, bewirkt das Programm `myisamchk` dasselbe auf den neueren `MyISAM`-Tabellentypen mit den `.myi`- und `.myd`-Endungen.

Folgende Optionen stehen zur Verfügung:

Option	Bedeutung
<code>-a, --analyze</code>	Analysiert die Verteilung von Schlüsseln. Nützlich, um einige JOIN-Operationen zu beschleunigen.
<code>-#</code> =debug-Optionen, <code>--debug=debug_Optionen</code>	Erstellt ein Protokoll für das Debugging, oftmals in der Form <code>d:t:o,Dateiname</code> , das in <i>Dateiname</i> gespeichert wird.
<code>-d, --description</code>	Gibt einige beschreibende Informationen über den Status einer Tabelle aus.
<code>-e, --extended-check</code>	Überprüft eine Datei sehr genau. In der Regel ist diese Option nicht nötig, da <code>isamchk</code> die meisten Fehler auch ohne sie findet.
<code>-f, --force</code>	Überschreibt temporäre Dateien.
<code>-i, --information</code>	Zeigt statistische Informationen über eine Tabelle an.

Tabelle 19.5: `isamchk`-Optionen

Option	Bedeutung
-k=x, --keys-used=x	Wird mit der -r-Option verwendet und weist isamchk an, die ersten x Schlüssel vor dem Reparieren zu löschen.
-l, --no-symlinks	Weist isamchk an, keinen symbolischen Links zu folgen (standardmäßig würde isamchk das tun).
-q, --quick	Wird in Zusammenhang mit der -r-Option verwendet, um die Reparatur zu beschleunigen. Repariert nur .isd-Dateien und keine Daten-dateien, solange kein optionales zweites -q angegeben ist. In einem solchen Fall werden auch die Datendateien repariert.
-r, --recover	Führt die Wiederherstellung aus. Diese Option wird die meisten Probleme und Fehler beseitigen, mit Ausnahme von <i>unique-key</i> -Verletzungen.
-o, --safe_recover	Verwendet die alte Recover-Methode. Diese ist langsamer als -r.
-s, --silent	Gibt nur Fehlermeldungen aus. Geben Sie -ss an, um auch diese zu unterdrücken.
-v, --verbose	Gibt mehr Informationen aus. Geben Sie -vv für noch mehr Informationen an.
-S, --sort-index	Sort <i>index</i> blockt im Zuge der Geschwindigkeitsverbesserung in Applikationen das Ausführen von vielen <i>read next</i> -Operationen.
-R=index, --sort-records=index	Sortiert aktuelle Datensätze nach einem übergebenden <i>index</i> . Dies führt bei einigen Anfragen zu Geschwindigkeitsvorteilen.
-u, --unpack	Entpackt eine Datei, die mit <i>pack_isam</i> gepackt wurde.
-V, --version	Gibt die Versionsnummer aus und beendet das Programm.
-W, --wait	Wartet mit dem Ausführen, wenn eine Tabelle gesperrt ist.
?, --help	Zeigt eine Liste aller isamchk-Optionen an.

Tabelle 19.5: isamchk-Optionen (Forts.)

Im Folgenden werden Sie einige dieser Optionen näher betrachten.

Schnelle Reparaturen

Um eine schnelle Reparatur an einer oder mehreren Tabellen durchzuführen, benutzen Sie folgende Syntax:

```
isamchk -rq Tabellenname
```

Das folgende Beispiel soll das veranschaulichen:

```
% isamchk -rq customers
- check delete-chain
```

```
- recovering ISAM-table 'customers.ISM'  
Data records: 15  
- Fixing index 1
```

Diese Anweisung bewirkt eine schnelle und effektive Reparatur. Es handelt sich dabei um eine schnelle Überprüfung, da das Programm nur die `.ism`-Datei bearbeitet und nicht die `.isd`- (Daten-) Datei.

Sie können ein zweites `-q` hinzufügen, um `isamchk` anzuweisen, auch die `.isd`- (Daten-) Datei zu überprüfen.

Um eine extensivere Überprüfung der Dateien zu veranlassen, verwenden Sie

```
isamchk -e Tabellename
```

Bei dieser Anweisung überprüft das Programm sowohl die `.ism`- als auch die `.isd`-Datei. Es überprüft die Dateien zusätzlich noch viel genauer auf mögliche Beschädigungen. Diese Anweisung wird die meisten Fehler beseitigen, jedoch bei ersten Fehlern die Überprüfung beenden.

```
isamchk -ev Tabellename
```

ist eine »wortreichere« Form der vorhergehenden Anweisung – mit dem Unterschied, dass Sie die Überprüfung der Dateien auch beim Auftreten von ernststen Fehlern nicht abbricht, und – wenn nötig – alle störenden Daten löscht. Aus diesem Grund ist es ratsam, ein Backup Ihrer Tabellen zu erstellen, bevor Sie `isamchk` zu Reparaturzwecken ausführen. Es ist auch eine gute Idee, die Anweisungen in der gezeigten Reihenfolge jeweils für sich zu starten, um sich progressiv in die Daten vorzuarbeiten und ein Gefühl für den Umfang der Beschädigung zu erhalten, bevor Sie eine umfangreiche Reparatur beginnen.

Probleme mit Schlüsseln beheben

Obwohl Schlüssel (keys) und Indizes normalerweise die Performance der Datenbank erhöhen sollen, gibt es Situationen, in denen Sie der Performance eher schaden, wie zum Beispiel bei `INSERT`- oder `UPDATE`-Anweisungen.

Wenn Sie `isamchk` bei beschädigten Schlüsseln ausführen, kann es sein, dass das Programm annimmt, dass die Daten selbst beschädigt sind, und diese löscht.

Daher müssen Sie die Schlüssel temporär entfernen, die Tabelle reparieren und anschließend die Schlüssel wieder einfügen. Die folgende Anweisung

```
isamchk -rqk=0
```

bewirkt genau dies, indem die Schlüssel entfernt (auf 0 setzen) und eine schnelle Überprüfung inklusive Reparatur durchgeführt wird. Die Anweisung

```
% isamchk -rqk=0 *.ISM
```

produziert zum Beispiel:

```
-check delete-chain
-recovering ISAM-table 'customers.ISM'
Data records: 15
```

Nachdem diese Anweisung ausgeführt wurde und Sie mit dem Ergebnis zufrieden sind, führen Sie

```
isamchk -rq
```

aus, um die Schlüssel wiederherzustellen.



Nachdem Sie mit dem Programm `isamchk` eine Reparatur ausgeführt haben, kann es den Anschein haben, dass Ihre Applikation nicht die reparierten Tabellen verwendet. Diese Situation tritt häufig dann auf, wenn es Ihnen nicht möglich war, während der Reparatur den `mysqld`-Daemon herunterzufahren. Führen Sie daher einfach `mysqladmin reload` aus, wenn ein solcher Fall eintritt.

Zusammenfassung

Heute haben Sie verschieden Aspekte der MySQL-Administration kennen gelernt:

- ▶ Wie Sie ein Backup Ihrer Datenbank in verschiedenen Formaten mit dem Programm `mysqldump` und seinen umfangreichen Optionen erstellen.
- ▶ Wie Sie eine Datenbank wiederherstellen, indem Sie einerseits das Programm `mysqlimport` zum Einlesen von Daten-Dateien benutzen und andererseits die Unix-Kommandos `cat` und `pipe` verwenden, um SQL-Strukturen und Daten direkt an MySQL zu übergeben.
- ▶ Wie Sie sich die verschiedenen Typen von Protokollierungsmöglichkeiten von MySQL zu Nutze machen: Aktualisierungs-Log-Dateien, um Änderungen an der Datenbank zu protokollieren, und »Full Activity«-Log-Dateien, um MySQL auf einer sehr detaillierten Ebene zu beobachten. Aktualisierungs-Log-Dateien sind nützlich bei inkrementeller Datensicherung, während vollständige Log-Dateien nützlich sind, um Applikations- und Systemproblemen auf die Spur zu kommen.
- ▶ Wie Sie Systemfunktionen verwenden, um Zugriff auf administrative Informationen Ihrer MySQL-Datenbank zu erlangen.
- ▶ Wie Sie das `mysqladmin`-Programm mit seinen umfangreichen Optionen und Befehlen für das Erstellen und Löschen von Datenbanken, das Ausgeben des Server-Status und von Thread-Informationen, dem Leeren der Log-Dateien oder anderen administrativen Funktionen verwenden.

- Wie die Dateistruktur von MySQL aussieht, wie Sie verschwendeten Festplattenplatz wieder freigeben und wie Sie eine beschädigte Tabelle überprüfen und wiederherstellen.

Mit diesem Wissen sind Sie auf dem bestem Weg zu effektiven Backups Ihrer Datenbank. Sie sind nun in der Lage, Systeminformationen abzufragen und die meisten Probleme, die auftreten können, selbst zu beheben.

Fragen und Antworten

- F** Ich möchte lediglich ein ganz einfaches Backup meiner kompletten Datenbank erstellen. Wie mache ich das?

- A** Benutzen Sie die folgende Anweisung:

```
mysqldump -opt Datenbankname > /Pfad/zum/mein_dump.sql
```

Diese Anweisung erstellt ein relativ kompaktes und schnelles Backup der kompletten Datenbank mit SQL-Anweisungen für das Erstellen der Tabellen und das Einfügen von Daten. Das Backup wird als eine große Datei (mein_dump.sql) abgespeichert, die Sie an einem sicheren Ort abspeichern können.

- F** Wie kann ich mit einem solchen Backup meine Datenbank wiederherstellen?

- A** Geben Sie in der (Unix-)Konsole Folgendes ein:

```
cat /home/mydata/mein_dump.sql | /Pfad/zu/mysql -u Benutzername -p  
Datenbankname
```

Diese Anweisung arbeitet die ganze Datei ab, löscht existierende Tabellen und stellt die komplette Datenbank auf dem Rechner wieder her.

- F** Was ist `mysqladmin`?

- A** `mysqladmin` ist ein nützliches Werkzeug für das Erstellen und Löschen von Datenbanken, das Herunterfahren des MySQL-Servers, die Überwachung von (Client-)Threads-Aktivitäten und vielem mehr. Es kann Log- und Host-Dateien leeren, Zugangspasswörter ändern und noch viele weitere Aufgaben ausführen.

Übungen

1. Schreiben Sie eine Anweisung (Syntax), um eine Tabelle namens *Meine_Tabelle* auf gelöschte Blöcke zu untersuchen.
2. Geben Sie die Syntax an, um alle aktiven Threads eines MySQL-Servers anzuzeigen.
3. Schreiben Sie einen Funktionsaufruf für den verschlüsselten String *my string* mit dem *Salt* in `ht1QbxA2IJPhU`.



MySQL optimieren

**Woche
3**

Nun, da Sie Ihre Datenbank aufgesetzt haben und diese seit einigen Wochen erfolgreich in Betrieb haben, sollte alles gut laufen. Doch plötzlich kommt ein verärrgerter Programmierer auf Sie zu und sagt Ihnen, Ihre Datenbank sei langsam. Es gebe eine große Meute von Anwendern, die über ihn herfielen, weil jede noch so einfache Anfrage eine halbe Ewigkeit dauere. Sie sagen ihm, das liege an seinem Programmcode. Und er besteht vehement darauf, dass die geringe Performance durch Ihre Datenbank und nicht durch seinen Programmcode verursacht werde. Sie erwidern, dass Sie ja schließlich MySQL benutzen und es kein schnelleres RDBMS auf dem Markt gebe. Nach ein paar Minuten weicht der Schmerz über das verletzte Ego, und Sie beide beschließen, sich zusammzusetzen und das Problem genauer zu analysieren. Sie finden heraus, dass es sich um eine Kombination aus beiden Problemen handelt, schlechte SQL-Anweisungen auf der Seite des Programmierers und einige Design- und Performance-Probleme auf Ihrer Seite.

Wenn Ihnen dieses Szenario bekannt vorkommt oder auch, wenn solche Probleme für Sie noch in weiter Ferne liegen, wird Ihnen dieser Tag weiterhelfen und erläutern, wie Sie Performance-Probleme aufspüren und was Sie dagegen tun können. MySQL ist schnell – um nicht zu sagen sehr schnell – aber es gibt einige Dinge, die Sie tun können, um Standardfehler zu vermeiden. Das können Design-Probleme sein oder auch einfach nur schlechte SQL-Anweisungen. Dieser Tag wird Ihnen helfen, Performance-Probleme zu lokalisieren und zu lösen.

Heute werden Sie Folgendes lernen:

- ▶ wie man die Performance einer Datenbank »tunen« (verbessern) kann
- ▶ wie man bessere SQL-Anweisungen schreibt
- ▶ wie man ungenutzten Speicherplatz wieder freigibt und die Datenbank säubert

Performance-Tuning

Der erste Ort, den Sie überprüfen sollten, wenn Performance-Probleme auftauchen, ist das System selbst. Auf welchem Typ von Computer läuft Ihr Datenbank-Server? Wie viel Speicher (RAM) hat der Computer? Wie hoch ist die Geschwindigkeit des Prozessors? Das alles sind Fragen, die sich stellen sollten, bevor Sie Ihre Datenbank in der Produktion (zum Beispiel auf einem Webserver) einsetzen.

Das Beste, was einem Datenbank-Server passieren kann, ist, auf einem topaktuellen System mit tonnenweise Speicher zu laufen, das einzig und allein für den Datenbank-Server verwendet wird. Das ist aber auch eine ideale Situation, und Sie werden nicht immer das bekommen, was Sie sich wünschen. MySQL ist eine andere Art von Datenbank. Es läuft auf fast jeder Plattform – auf einigen besser und auf einigen schlechter.

MySQL wurde im Wesentlichen auf Intel-Computern mit Linux als Betriebssystem entwickelt. Daher ist Linux wohl die Plattform, auf der MySQL am besten läuft. Da diese Computer Linux einsetzen, muss die Hardware nicht so stark aus dem High-End-Bereich stammen, wie es unter Windows NT der Fall wäre. Dieser Umstand macht MySQL noch attraktiver: eine hochperformante Datenbank, die auch auf älteren Rechnern läuft. MySQLs Performance auf Windows-Plattformen ist nicht so gut wie auf anderen Plattformen. Das hat viele Gründe: Speicherzuweisung auf der Betriebssystemebene ist einer der Hauptgründe.

MySQL ist *multithread-fähig*. Das bedeutet, dass jedes Mal, wenn eine Verbindung zum MySQL-Server hergestellt wird, eine neuer Thread gestartet wird. Jeder Thread benötigt Speicherressourcen. Das Zwischenspeichern von Ergebnissen aus Anfragen benötigt diese ebenfalls. Daher gilt, je mehr Speicher, desto besser. Das erhöht im Allgemeinen die Performance.

Ein anderer Punkt, der für die Performance von Bedeutung ist, ist die Festplatte. Eine schnellere Festplatte produziert schnellere Ergebnisse. Wenn Sie eine viel beanspruchte Tabelle haben, könnten Sie diese Tabelle auf eine separate Festplatte auslagern. Das trägt immens zur Beschleunigung bei. Das Aufsetzen einer »multiple-disk«-Umgebung geht jedoch weit über das Ziel dieses Buches hinaus. Die MySQL-Online-Dokumentation enthält alle Informationen, die Sie für ein solches Vorhaben benötigen.

Konfigurieren der Systemvariablen

Nachdem Sie die Hardware auf die Datenbank-Ansprüche abgestimmt haben, sollten Sie sich daran machen, das aktuelle Datenbanksystem selbst anzupassen. Es gibt eine Reihe von Variablen, die das Verhalten von MySQL determinieren. Um Ihre aktuelle Konfiguration einzusehen, führen Sie folgende Schritte aus:

- ▶ Begeben Sie sich in das `mysql`-Verzeichnis und stellen Sie sicher, dass der `mysqld`-Daemon läuft.
- ▶ Geben Sie die folgende Anweisung ein: `bin/mysqladmin -u root -p variables`
- ▶ Sie werden nach dem Root-Passwort gefragt. Geben Sie es ein, und Sie sollten die folgende oder eine ähnliche Ausgabe erhalten (hier eine Ausgabe auf einem Windows-System):

Variable_name	Value
ansi_mode	OFF
back_log	50
basedir	c:\mysql\
character_set	latin1

character_sets	latin1	
connect_timeout	5	
concurrent_insert	OFF	
datadir	c:\mysql\data\	
delay_key_write	ON	
delayed_insert_limit	100	
delayed_insert_timeout	300	
delayed_queue_size	1000	
join_buffer_size	131072	
flush	OFF	
flush_time	1800	
init_file		
interactive_timeout	28800	
key_buffer_size	8388572	
language	c:\mysql\share\english\	
log	OFF	
log_update	OFF	
log_bin	OFF	
log_slave_updates	OFF	
long_query_time	10	
low_priority_updates	OFF	
lower_case_table_names	1	
max_allowed_packet	1048576	
max_connections	100	
max_connect_errors	10	
max_delayed_threads	20	
max_heap_table_size	16777216	
max_join_size	4294967295	
max_sort_length	1024	
max_tmp_tables	32	
max_write_lock_count	4294967295	
myisam_sort_buffer_size	8388608	
net_buffer_length	16384	
net_retry_count	10	
pid_file	c:\mysql\data\user.pid	
port	3306	
protocol_version	10	
record_buffer	131072	
skip_locking	ON	
skip_networking	OFF	
skip_show_database	OFF	
slow_launch_time	2	
socket	MySQL	
sort_buffer	2097116	
table_cache	64	
table_type	MYISAM	

thread_stack	65536	
thread_cache_size	0	
timezone	(MEZ) - Mitteleurop. Sommerzeit	
tmp_table_size	1048576	
tmpdir	c:\windows\temp\	
version	3.23.22-beta-debug	
wait_timeout	28800	
+-----+-----+-----+		

Um einen dieser Werte zu ändern, benutzen Sie die folgende Syntax

```
safe_mysqld -O variable = Wert
```

wobei *variable* eine der Systemvariablen ist und *Wert* ein logischer Wert für diese Variable. Änderungen sind nicht beständig. Das bedeutet, wenn Sie Ihre Session beenden und den MySQL-Server erneut ohne jegliche Optionen starten, so wird MySQL wieder die Standardeinstellungen verwenden. Keine Ihrer vorhergehenden Änderungen werden mehr vorhanden sein. Um Ihre Änderungen dauerhaft zu machen, müssen Sie Ihr Start-up-Script ändern (unter Windows: C:\WINDOWS\my.ini bzw. c:\my.cnf).

Die meisten hier aufgelisteten Variablen erklären sich von selbst. Die Folgenden sind die wichtigsten Variablen, mit denen Sie Ihr System optimieren können:

- ▶ **back_log:** Erhöhen Sie diesen Wert, wenn Sie eine große Anzahl simultaner Verbindungen erwarten. Dieser Wert kontrolliert, wie viele Verbindungen MySQL hält, während es neue Threads startet.
- ▶ **delayed_queue_size:** Dieser Wert kontrolliert die Anzahl der Reihen, die in Warteschleifen stehen können, wenn eine `INSERT DELAYED`-Anweisung verwendet wurde. Erhöhen Sie diesen Wert, wenn eine hohe Anzahl solcher `INSERTs` zu erwarten ist.
- ▶ **flush_time:** Dieser Wert gibt an, wie viele Sekunden MySQL wartet, bis es den Inhalt des Zwischenspeichers auf die Festplatte schreibt.
- ▶ **table_cache:** Dieser Wert gibt die Anzahl aller offenen Tabellen für alle Threads an. Erhöhen Sie diesen Wert, erhöhen Sie also die Anzahl der Tabellen, die zur gleichen Zeit geöffnet sein können. Dieses verringert die »Kosten« und kann die Dinge beschleunigen. Trotzdem kann das Betriebssystem Grenzen haben, die möglicherweise die Anzahl der simultan geöffneten Dateien einschränken.
- ▶ **wait_timeout:** Setzt die Zeitspanne, die eine inaktive Verbindung aufrechterhalten wird. Ein kleinerer Wert kann die Geschwindigkeit verbessern.
- ▶ **buffers:** Das Erhöhen der Werte eines jeden Buffers wird zur Performance-Verbesserung führen. Ein zu hoher Wert kann aber auch schädlich sein. Die Werte dieser Einstellungen sollten auf den real vorhandenen Speicherressourcen basieren.



Wenn Sie eine große Anzahl an Tabellen und viel Speicher haben, können Sie MySQL beschleunigen, indem Sie die Buffer-Größe gleich der Größe des Speichers setzen, z.B. `buffer = 32 M`;

mysqld-Daemon und seine Parameter

`mysqld` ist das Herz von MySQL. Hierbei handelt es sich um das eigentliche Serverprogramm. Im Endeffekt ist es der `mysqld`-Daemon, an den Sie bei MySQL denken. Der Datenbank-Server kann mit einigen Optionen gestartet werden, die Ihnen bei der Optimierung helfen. Im Folgenden finden Sie eine Liste der wichtigeren Parameter, die Sie `mysqld` übergeben können.

Parameter	Bedeutung
<code>-b, --base-dir</code> = <i>vollständiger Pfad</i>	Dieser Parameter weist dem MySQL den Weg zum <code>mysql</code> -Installations-Verzeichnis. Dieses gibt dem Server an, wo er alle benötigten Dateien finden kann.
<code>--big-tables</code>	Wenn MySQL mit dieser Option gestartet wird, erlaubt es große Ergebnis-Sets. Das verhindert Fehler bezüglich voller Tabellen.
<code>-h</code> oder <code>--datadir</code> = <i>vollständiger Pfad</i>	Dieser Parameter gibt MySQL den Pfad zum <code>data</code> -Verzeichnis an. Setzen Sie diese Option, wenn Sie das Verzeichnis an einen anderen Ort verlagert haben.
<code>--enable-locking</code>	Diese Option erlaubt das Sperren des Systems. MySQL unterstützt dieses nicht länger standardmäßig.
<code>-l, --log=[Dateiname]</code>	Veranlasst MySQL, Systemmeldungen in diese Datei zu protokollieren. Wenn kein <code>Dateiname</code> angegeben ist, werden diese in die <code>mysql-log</code> -Datei geschrieben.
<code>--log-update=Dateiname</code>	Veranlasst MySQL, alle Änderungen an der Datenbank zu protokollieren. Dieses ist dann ein sehr nützliches Werkzeug, um Backups Ihrer Datenbank zu erstellen. Diese Option wurde näher an Tag 19 besprochen »MySQL administrieren«.
<code>--log-long-format=Dateiname</code>	Diese Option veranlasst MySQL, mehr Informationen zu protokollieren als die <code>--log-update</code> -Option. Für nähere Informationen sehen Sie wiederum bei Tag 19 nach.
<code>--low-priority-insert</code>	Diese Option veranlasst MySQL, <code>SELECT</code> -Anweisungen vor <code>INSERT</code> -Anweisungen auszuführen.

Tabelle 20.1: `mysqld`-Parameter

Parameter	Bedeutung
<code>--skip-grant-tables</code>	Dieses ist eine handliche Option für den Fall, dass Sie das Root-Passwort vergessen haben. Diese Option lädt keine GRANT-Tabellen beim Start, und somit sind keine Passwörter noch benutzerdefinierte Funktionen mehr wirksam.
<code>--skip-name-resolve</code>	Diese Option kann die Dinge ein wenig beschleunigen. Sie veranlasst MySQL IP-Adressen zu verwenden, anstatt IP-Adressen in Namen umzuwandeln.
<code>-V, --version</code>	Dies gibt Informationen über die Version der aktuellen MySQL-Installation aus.

Tabelle 20.1: *mysqld-Parameter (Forts.)*

Eine Möglichkeit, MySQL zu beschleunigen, besteht darin, die Protokollierung auszuschalten – was allerdings nicht zu empfehlen ist.. Wenn Ihre Datenbank abstürzen sollte, gäbe es keinen Weg mehr, die Daten wiederherzustellen. Trotzdem würde das Ausschalten der Protokollierung einen Geschwindigkeitsvorteil bringen. Sie selbst müssen hier abwägen.

Ein weiterer kleiner Performance-Gewinn kann durch die Option `--skip-name-resolve` erzielt werden. Das kann ein klein wenig Zeit sparen und sollte keine negativen Effekte auf Ihre Datenbank-Operationen haben.

Erstellen von besseren SQL-Anweisungen

Als Nächstes sollten Sie einen Blick auf Ihr Datenbank-Design und Ihre SQL-Anweisungen, die Sie benutzen, um Zugang zu Ihren Daten zu erlangen, werfen. Man könnte ein ganzes Buch über die besten Techniken schreiben, aber das würde über das Ziel dieses Buches weit hinausgehen. Dieses Buch wird einige grundlegende Schritte behandeln, die Sie unternehmen können, um Ihre Datenbank zu beschleunigen.

Die Struktur Ihrer Datenbank spielt eine wesentliche Rolle für Ihre Performance. Wenn Ihre Tabellen nicht richtig konstruiert wurden, ist es wahrscheinlich, dass Sie schnell mit Problemen konfrontiert werden. Im Folgenden werden einige allgemeine Regeln für »schnelle Tabellen« aufgeführt:

- ▶ *Benutzen Sie den kleinstmöglichen Datentyp:* Je kleiner der Datentyp, desto weniger Plattenplatz und Speicher wird verbraucht. Je weniger Speicher verbraucht wird, desto schneller können Daten angefordert und benutzt werden.
- ▶ *»Zwingen Sie Werte in Spalten«, indem Sie diese auf NOTNULL setzen:* Das spart etwas Platz und beschleunigt die Dinge.

- ▶ *Versuchen Sie variable Längen für Spalten zu vermeiden:* Wenn Sie müssen, dann müssen Sie. Aber wenn Sie auch ohne diese auskommen, dann tun Sie es auch. Dieses kann die Performance wirklich beeinflussen.
- ▶ *Vermeiden Sie den Gebrauch zu vieler Indizes:* Indizes beschleunigen zwar `SELECT`-Anweisungen, verlangsamen jedoch `INSERT`- und `UPDATE`-Anweisungen. Haben Sie zu viele Indizes, verlangsamt sich alles. Um hierzu weitere Informationen zu erhalten, schauen Sie sich Tag 6 an.
- ▶ *Wählen Sie die Tabellentypen mit Blick auf die Performance:* Es gibt vier Tabellentypen in MySQL (nicht zahlende Benutzer haben nur drei). Diese sind `static` (Standard), `dynamic`, `heap` und `compressed`. Die `static`-Tabelle wird standardmäßig erstellt. Dies ist die schnellste aller normalen Tabellen und kann nur in der Länge nichtvariable Spalten fassen. Wenn eine Tabelle nur eine einzige Spalte mit variabler Länge hat, wird der Typ automatisch in `dynamic` geändert. Diese Tabellen sind langsamer, da sie mehr Informationen enthalten (jede Zeile enthält Informationen über die Länge). Die `heap`-Tabellen sind extrem schnell bei kleinen und mittleren Tabellen, da diese nur im Speicher existieren. Dieser Typ von Tabellen ist großartig für `JOINS` und Ähnliches. `compressed`-Tabellen sind nur lesbar. Dieser Typ benötigt weniger Plattenplatz und ist sehr schnell. Dieser Typ steht nur zahlenden MySQL-Nutzern mit ausführlichem E-Mail-Support zur Verfügung.
- ▶ *Benutzen Sie Standards in Spalten:* Dies verringert die Ausführungszeit von `INSERT`-Anweisungen und erhöht die Performance.

In einem weiteren Schritt sollten Sie einen Blick auf die SQL-Anweisungen werfen, mit denen Sie die Daten in Ihrer Datenbank manipulieren. Im Folgenden werden einige allgemeine Regeln für die Verbesserung der allgemeinen Performance aufgeführt:

- ▶ Schreiben Sie Ihre Datenbankabfragen so, dass diese nach Möglichkeit Nutzen aus Indizes ziehen können. Dafür sind Indizes gemacht. Zudem können sie wesentlichen Einfluss auf die Performance von Abfragen haben.
- ▶ Benutzen Sie die `LIMIT`-Anweisung in Ihren Anfragen. Dies veranlasst MySQL, in einigen Fällen Indizes zu verwenden, obwohl es das normalerweise nicht tun würde.
- ▶ Vermeiden Sie, wenn immer möglich, unnötige Zeichensetzungen wie etwa Klammern. MySQL muss solche Zeichensetzungen erst herausziehen, bevor es die Anfrage bearbeiten kann. Wenn keine solche Zeichensetzungen vorhanden sind, kann MySQL Anfragen viel schneller ausführen.
- ▶ Sicherheit steht in direkter proportionaler Verbindung zur Geschwindigkeit, mit der Anfragen in MySQL ausgeführt werden. Je höher die Sicherheitsstufe, desto mehr Aufwand entsteht. Aber opfern Sie Sicherheit niemals zu Gunsten der Performance. Setzen Sie Sicherheitsoptionen mit Bedacht dort ein, wo sie sinnvoll sind. Sie könnten zum Beispiel nur einen Benutzer für eine vielbenutzte Daten-

bank erstellen. Bestimmen Sie Ihren Sicherheitslevel also eher mit Ihrer Applikation und nicht auf der Server-Ebene. Sie könnten zum Beispiel eine Tabelle mit Benutzernamen und Passwörtern erstellen und diese von der Applikation überprüfen lassen, um festzustellen, ob ein Benutzer Zugangsberechtigung zu Ihrer Datenbank besitzt oder nicht. Wenn der Benutzer diese Überprüfung besteht, verwenden Sie den Benutzer aus den GRANT-Tabellen für den Rest seiner Datenbankzugriffe.

- Benutzen Sie `SELECTCOUNT(*)`, um alle Datensätze (Zeilen) einer Tabelle zu zählen. Diese Anweisung ist sehr schnell, da Sie die entsprechenden Informationen aus dem Tabellenkopf erfährt.

Dies sind nur einige allgemeine Hilfestellungen, um Ihre Anweisungen zu beschleunigen. Die besten Ergebnisse erzielen Sie allerdings mit einer Optimierung der Anweisungen selbst.

Erstellen besserer WHERE-Klauseln

Der beste Weg, um Ihre WHERE-Klausel zu formulieren, ist, zunächst die vorhandenen Indizes Ihrer Tabellen zu betrachten. Hat die Tabelle einen Index, dann benutzen Sie diesen; das wird Ihre Anfragezeiten merkbar beschleunigen. Auch wenn Sie die WHERE-Bedingung nicht neu erstellen können, erstellen Sie Indizes.

Beachten Sie, dass bedingte Anweisungen mit numerischen Werten schneller ausgeführt werden als mit String- oder Zeichen-Werten. Wenn immer möglich, vergleichen Sie Zahlen.

Die LIKE-Anweisung kann Anfragen ebenfalls negativ beeinflussen. Wenn bei Vergleichen die zu vergleichende Konstante zum Beispiel keine Wildcards (Platzhalter) an erster Stelle hat, werden unter Umständen Indizes verwendet. Folgende Anweisung würde den Index benutzen, wenn die Spalte *Last_Name* einen hätte:

```
SELECT State FROM Customers WHERE Last_Name LIKE "Rober%";
```

Bei der folgenden Anweisung würde ein Index nicht verwendet werden:

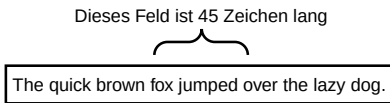
```
SELECT State FROM Customers WHERE Last_Name LIKE "%ober%";
```

Ungenutzten Speicherplatz freigeben

Wird ein Datensatz aus einer Datei gelöscht oder ändert sich eine Spalte, die zuvor Felder variabler Länge hatte, entstehen ungenutzte Stellen in der Datei, die diese Daten speichert.

Vor der UPDATE-Anweisung

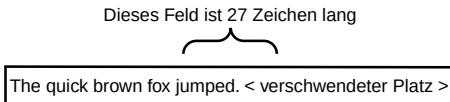
Dieses Feld ist 45 Zeichen lang



The quick brown fox jumped over the lazy dog.

Nach der UPDATE-Anweisung

Dieses Feld ist 27 Zeichen lang



The quick brown fox jumped. < verschwendeter Platz >

Abbildung 20.1:
Eine Spalte mit variabler Länge vor und
nach einem UPDATE

Dieser »tote« Platz kann sich anhäufen, Plattenplatz verbrauchen und Datenbank-Anfragen verlangsamen. Es gibt allerdings auch keine Möglichkeit, diesen Effekt komplett zu vermeiden (außer zum Beispiel keine Felder mit variabler Länge zu verwenden).

Ein anderes Problem, das sich negativ auf die Performance der Datenbank auswirken kann, ist die Anordnung der Schlüssel. Stellen Sie sich vor, Sie haben einen Index auf einer eindeutigen `AUTO_INCREMENT`-Spalte. Sie haben viele Datensätze eingefügt und einige gelöscht, und die Datensatznummern sind vielleicht völlig aus der Reihenfolge gekommen. Das kann die Index-Performance stark beeinträchtigen: Stellen Sie sich nur einmal vor, wie jemand etwas in einer Datei finden soll, wenn die Daten nicht mehr in alphabetischer Reihenfolge sind.

Glücklicherweise haben die Entwickler von MySQL dieses Problem vorhergesehen und ein Werkzeug zur Verfügung gestellt, um dieses Problem zu beheben. Dieses kleine Programm heißt `myisamchk`. Sie finden es im `bin`-Verzeichnis des `mysql`-Installationsverzeichnisses.

Das `myisamchk`-Programm kann, ähnlich der `OPTIMIZE`-Anweisung, dazu verwendet werden, Tabellen und Schlüssel zu optimieren, nur dass es viel schneller ist. Es kann auch helfen, beschädigte Tabellen zu reparieren. Die Syntax ist wie folgt:

```
myisamchk Optionen Tabellename
```

wobei *Tabellename* der Name der Tabelle oder in Zusammenhang mit Wildcards von Gruppen von Tabellen ist. Um das Programm auf alle Tabellen in Ihrem `data`-Verzeichnis anzuwenden, führen Sie die folgende Anweisung aus:

```
/bin/myisamchk /usr/local/mysql/data/*/*.myi
```

Diese Anweisung überprüft alle (Daten-)Dateien im `data`-Verzeichnis. Dem `myisamchk`-Programm können folgende Optionen übergeben werden:

Option	Bedeutung
-a, -analyze	Überprüft die Pfade von Tabellen, um zu sehen, ob JOINS beschleunigt werden können.
-d, -description	Gibt einige Informationen über die Tabelle aus, wie etwa Datensatz-Anzahl, -Format und -größe, gelöschte Blöcke und Schlüsselinformationen.
-f, -force	Überschreibt automatisch alle temporären Dateien.
-i, -information	Stellt detaillierte Informationen über die Tabelle zur Verfügung. Dazu gehören Datensatz-Länge, Platzverbrauch und Größe des verlorenen Plattenplatzes. Das sind einige sehr nützliche Informationen, wenn Sie das Programm zu Reparaturzwecken einsetzen müssen.
-q, -quick	Veranlasst das Programm, eine schnellere Reparatur durchzuführen. Die Originaldatei wird dabei nicht angerührt. Um diese Datei zu benutzen, fügen Sie ein weiteres -q (<code>myisamcheck -qq</code>) hinzu.
-r, -recover	Veranlasst das Programm, jede beschädigte Datei zu reparieren. Zusätzlich säubert es die Dateien und gibt verlorenen Platz wieder frei.
-o, -safe-recovery	Hierbei handelt es sich um die langsamere Variante von -r. Es kann manchmal Beschädigungen beheben, die -r nicht beheben kann.
-S, -sort-index	Diese Option veranlasst <code>myisamchk</code> , den Index von oben nach unten zu sortieren. Das führt zu schnelleren Anfragen, wenn Indizes verwendet werden.
-u, -unpack	Dekomprimiert eine Datei, die mit <code>myisampack</code> komprimiert wurde.
-v	Gibt detailliertere Informationen aus (<i>verbose</i>).
-w	Veranlasst das Programm, zu warten, bis eine Tabelle wieder entsperrt ist, bevor es Reparaturen oder Sortieroperationen vornimmt.

Tabelle 20.2: `myisamchk`-Optionen

Sie können auch den Speicherplatz angeben, der von den Arbeiten von `myisamchk` verwendet werden darf. Je mehr Speicher, desto schneller die Anwendung.

Die Wartung Ihrer Datenbank ist ein Muss, um gute Performance zu erzielen. Sie sollten eine Routine schreiben, die Ordnung in Ihren Tabellen hält. Ein Shell-Script in Verbindung mit einem `cron`-Job auf Unix-Plattformen bzw. ein Batch-Job oder Perl-Script auf Windows-Systemen kann Ihnen diese Arbeit abnehmen. Die meisten Probleme können schon, bevor Sie große Auswirkungen auf Ihr Produktionssystem zeigen, behoben werden.

Um verlorenen Speicherplatz wieder freizugeben und Ihre Tabellen auf Fehler zu überprüfen, geben Sie folgende Anweisung ein:

```
%> bin/myisamchk -r /usr/local/mysql/data/*/*.myi
```

Diese Anweisung gibt den verlorenen Speicherplatz wieder frei und überprüft und repariert Ihre Tabellen. Wenn Sie die Indizes in Ihren Tabellen wieder in die richtige Reihenfolge bringen möchten, führen Sie folgende Anweisung aus:

```
%> bin/myisamchk -S /usr/local/mysql/data/*/*.myi
```

Alle Ihre Indizes werden so wieder in die richtige Reihenfolge gebracht. Diese Operation kann durchaus etwas länger dauern. Daher sollten Sie den Speicher, den die Anwendung hierfür verwenden darf, erhöhen. Das tun Sie, indem Sie jede andere Anwendung, die gerade auf Ihrem System läuft, schließen und den `-o`-Schalter des `myisamchk`-Programms benutzen.

Das `myisamchk`-Programm kann benutzt werden, um die Performance Ihrer Datenbank zu erhöhen. Indem es verlorenen Platz wieder freigibt, schafft es Platz für kleinere Datensätze, was das Hauptziel bei der Optimierung ist. Kleiner ist besser.

Kompilieren und Komprimieren

Wenn alles Bisherige keinen Erfolg gebracht hat, dann ist es jetzt Zeit, um zu »härten« Mitteln zu greifen. Zunächst sollten Sie in Erwägung ziehen, den Quellcode auf Ihrem System mit einem anderen Compiler neu zu kompilieren. Die RPM-Versionen von MySQL werden mit einem schnellen Compiler mit allen Optimierungsoptionen kompiliert. Sie könnten auch auf eine andere Plattform, wie etwa SUN-Systeme, wechseln, um hier von einer größeren Performance und mehreren Prozessoren zu profitieren. Sie können die Geschwindigkeit Ihrer Anwendung auch erhöhen, indem Sie einen Compiler verwenden, der schnellere Anwendungen für dieses System erstellt. Einige Leute berichten davon, dass alleine diese Maßnahme Ihnen eine Performancesteigerung von 20 bis 30 % gebracht hat.

Wenn Sie MySQL nicht selbst neu kompilieren wollen bzw. können, dann sollten Sie den E-Mail-Support kaufen, um an das `mysampack`-Programm zu kommen. Wenn Sie dieses auf Tabellen anwenden, von denen nur gelesen wird, werden Sie eine signifikante Geschwindigkeitssteigerung bemerken.

Zusammenfassung

Optimierung ist wie Schwarze Magie. Sie brauchen eine Menge Praxis und Übung, um zu erfahren, wie alles läuft. Und auch dann kann es noch in »Trial-and-Error« enden. Die Punkte, die heute angesprochen wurden, sowie die allgemeinen Regeln und Richtlinien sollen Ihnen nur einen Startpunkt bieten, von dem aus Sie Ihr »Abenteuer Performance« beginnen können.

Im Folgenden erhalten Sie nochmals einen kurzen Überblick über die Schritte, die Sie zur Performance-Steigerung Ihrer Datenbank unternehmen können:

1. Überprüfen Sie die Hardware, auf der Ihr Datenbank-Server laufen soll. Genügt diese den Ansprüchen und erwarteten Anforderungen?
2. Überprüfen Sie das Betriebssystem. Einige sind besser als andere. SUNs OS und Linux sind großartig für Multi-Prozessoren-Einsatz. Auf der anderen Seite ist Windows NT einfacher zu bedienen. Wägen Sie die Vor- und Nachteile gegeneinander ab, und entscheiden Sie dann, welches die beste Plattform für Ihre Bedürfnisse ist.
3. Überprüfen Sie Ihr Tabellen-Design. Ist es solide? Sind die Tabellen in der 3. Normalform? Sind die Spaltentypen so klein, wie sie sein könnten? Kommen auch Spalten variabler Länge vor – und wenn ja, brauchen Sie diese wirklich?
4. Betrachten Sie die Anweisungen, die an die Datenbank gestellt werden. Verwenden Sie Indizes? Sind die `WHERE`-Bedingungen kurz und bündig?
5. Werden Ihre Tabellen gewartet? Wann haben Sie das letzte Mal das `myisamchk`-Programm ausgeführt? Haben Sie einen `cron`-Job, der diese Aufgaben automatisch ausführt?
6. Wenn alles andere erfolglos bleibt, kompilieren Sie den Quellcode mit einem besseren Compiler. Das kann helfen. Ziehen Sie auch in Betracht, das `myisampack`-Programm zu erwerben. Das kann Ihnen auch helfen, Ihre Performance zu verbessern.

Es gibt viele Wege, um die Performance einer Datenbank zu steigern. Leider wird es etwas Zeit dauern, bis Sie herausgefunden haben, was bei Ihnen hilft. Hoffentlich hat Ihnen der heutige Tag einige Einblicke in die Kunst der Datenbank-Optimierung verschafft.

Fragen und Antworten

F Wie kann ich einen `cron`-Job aufsetzen, der meine Tabellen automatisch wartet?

A *Im Allgemeinen müssen Sie, um den `cron`-Job zu nutzen, einen Eintrag in der `cron`-Datei vornehmen. Diese Datei beinhaltet alle Informationen für den `cron`-Job. Der folgende Eintrag in Ihrer `cron`-Datei sollte Ihre Tabellen jede Woche überprüfen und ggf. reparieren:*

```
45 0 * * 0 /usr/local/mysql/bin/myisamchk -r /usr/local/mysql/data/*/*.*mi
```

Konsultieren Sie Ihre System-Dokumentation, um genau zu überprüfen, was für einen `cron`-Job-Eintrag benötigt wird.

F Was macht die OPTIMIZE-Anweisung?

A Die OPTIMIZE-Anweisung ist dem `myisamchk`-Programm sehr ähnlich – mit dem Unterschied, dass es langsamer ist und nicht über eine solche Funktionalität verfügt. Auf der anderen Seite kann es mit einem ODBC-Interface oder jedem anderem Interface benutzt werden, da MySQL die OPTIMIZE-Anweisung wie jede andere SQL-Anweisung behandelt. Die Syntax der OPTIMIZE-Anweisung ist sieht folgendermaßen aus

```
OPTIMIZE Tabellename
```

wobei *Tabellename* der Name der Tabelle ist, die Sie optimieren möchten. Diese Anweisung gibt zwar verlorenen Speicher wieder frei, repariert jedoch keine beschädigten Tabellen.

Übungen

1. Führen Sie `myisamchk` mit dem `-recover`-Schalter auf alle Tabellen der `Meet_A_Geek`-Datenbank aus.
2. Bringen Sie die Voraussetzungen in Erfahrung, die ein Server haben muss, damit die `Meet_A_Geek`-Datenbank dort performant laufen kann. Beachten Sie dabei, dass die Datenbank rund 10.000-mal am Tag via Internet aufgerufen werden wird. Beachten Sie auch, dass MySQL am besten auf Unix-basierten Systemen läuft.



**Ein Projekt wird
verwirklicht**

**Woche
3**

In den letzten drei Wochen haben Sie eine Menge gelernt. Heute werden Sie die Schritte rekapitulieren, die Sie getan haben, um an diesen Punkt zu kommen. Jede Applikation bzw. jedes Projekt kann in mehrere Schritte aufgeteilt werden, die unternommen werden müssen, um eine bestimmte Aufgabe zu erfüllen. Diese Aufgabe kann das Erstellen einer neuen Datenbank-Applikation einer Webseite sein. Die Schritte, die unternommen werden müssen, um zum Erfolg zu kommen, sind:

1. Definition der Problemstellung
2. Analyse der Anforderungen
3. Planung der Struktur/Architektur
4. Programmierung
5. Testen
6. Implementieren

Dieser Tag wird jeden Schritt des Entwicklungsprozesses gehen und ihn auf die Meet_A_Geek-Datenbank anwenden. Am Ende des Tages sollten Sie wissen, wie Sie Ihr jetzt erlerntes Wissen anwenden können.

Erstellen der Meet_A_Geek-Datenbank

Bevor Sie mit dem Projekt beginnen, sollten Sie sich erst einmal die Zeit nehmen, dieses genau zu definieren und alle Erfordernisse zu analysieren, anstatt sofort mit der Programmierung loszulegen. Um herauszufinden, was für Ansprüche ein Projekt an Sie stellt, müssen Sie es zunächst genau definieren. Sie können ein Projekt definieren, indem Sie sich ein paar einfache Frage beantworten: Was soll das System leisten? Welches System soll durch dieses ersetzt werden? Ein gut umrissenes Projekt ist der erste Schritt in Richtung Erfolg. Wenn Sie es hingegen niemals genau bestimmt haben, woher wollen Sie wissen, ob Sie erfolgreich waren oder nicht? Für das Meet_A_Geek-Projekt ist die Definition ziemlich einfach: Erstellen Sie ein Web, mit dem Anwender (Kunden) einfach und schnell über das Netz Anzeigen platzieren und auf Anzeigen antworten können. Wenn ein Anwender auf eine Anzeige eines anderen Mitglieds antwortet, soll er die Möglichkeit haben, ein Geschenk aus dem eigenem Web-Shop zu bestellen und es zu versenden.

Während der ersten Schritte in Ihrem Projekt legen Sie fest, welche Ziele Sie erreichen möchten und welche Arbeiten dafür zu erfüllen sind. Während dieser Schritte sollten Sie auch den Erfolg oder Misserfolg des Projekts definieren. Wenn Sie der Projektleiter sind, liegt das in Ihrer Verantwortung. Nachdem Sie diese Fragen beantwortet haben, sind Sie bereit, den nächsten Schritt zu tun.

Im zweiten Schritt des Entwicklungsprozesses sollten Sie die Voraussetzungen des Projekts ermitteln. Sie müssen sich mit potentiellen Anwendern des Systems unterhalten, um zu klären, was die Applikation erfüllen soll. Sie müssen herausfinden, welche Informationen für den Anwender wichtig sind, welche Regeln auf die Daten angewandt werden müssen, um zu sehen, ob diese gültig sind oder nicht, und welche Art von Berichten generiert werden muss. Es ist diese Phase des Projekts, in der Sie den Geschäftsprozess, die Geschäftsregeln und die Geschäftsobjekte bestimmen. Das ist der erste Schritt im Entwicklungsprozess der Datenbank, über den Sie am dritten Tag, »Planung Ihrer ersten Datenbank« mehr gelernt haben. In dieser Phase sollten Sie beginnen, nach geeigneten Geschäftsobjekten zu suchen, die später zu Tabellen in Ihrer Datenbank werden.

Anforderungsphase

Diese Phase fordert von Ihnen, dass Sie sich mit den verschiedenen Ebenen des Managements unterhalten, um ein vollständiges Verständnis von dem zu erhalten, was das Projekt für Anforderungen erfüllen soll und was die Applikation können muss, um die Geschäftsziele zu erreichen. Von hier an sollten Sie sich Gedanken darüber machen, was Sie alles aufbringen müssen: welches Equipment benötigt wird, welche Plattform Sie einsetzen möchten und wie die Datenbank aussehen soll.

Zu diesem Zeitpunkt des Projekts können Sie folgende Punkte festhalten:

1. Ein potentieller Kunde kann entweder auf eine Anzeige antworten oder selbst eine erstellen.
2. Kunden müssen Mitglieder werden, bevor Sie auf Anzeigen antworten können.
3. Anschließend an die Suche in der Datenbank nach potentiellen Verabredungen kann der Kunde entweder E-Mails oder auch Geschenke als Antwort auf die Anzeige verschicken.
4. Alle Transaktionen müssen verfolgt werden.
5. Mitglieder müssen sich authentifizieren, bevor Sie in der Datenbank suchen können.
6. Die Produkte, die als Geschenke im Shop angeboten werden, müssen in derselben Datenbank gehalten werden.

Planungsphase

Nun, wo die Anforderungen an die Datenbank geklärt sind, können Sie mit der Planungsphase der Datenbank beginnen. Das ist die wichtigste Phase des ganzen Prozesses. Hier wird der Grundstein für das gesamte Projekt gelegt. Ebenfalls wird hier die Meet_A_Geek-Datenbank gestaltet und ein Flussdiagramm für die Webseite entworfen.

Auf den bisherigen Ergebnissen basierend wurde das Flussdiagramm aus der folgenden Abbildung für die Webseite erstellt.

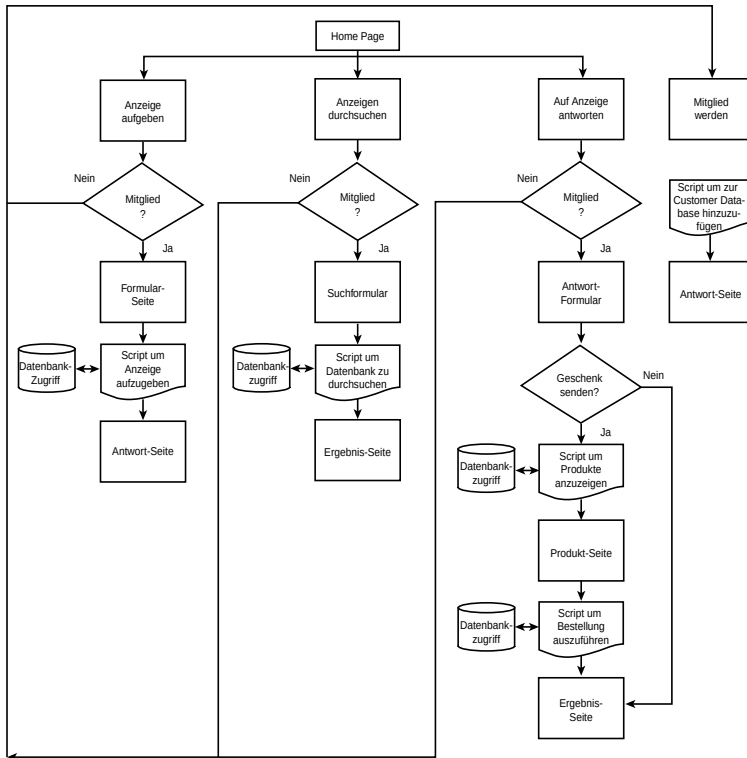


Abbildung 21.1:
Flussdiagramm
der
Meet_A_Geek-
Datenbank

Beachten Sie, wie das Flussdiagramm der Webseite hilft, die Ablauf-Regeln, die zuvor in der Anforderungsphase bestimmt wurden, zu überprüfen, im Besonderen die Regel, die besagt, dass ein Anwender keine Suche über die Datenbank oder Antwort auf eine Anzeige ausführen kann, solange er kein Mitglied ist. Diese Regel wird durch die Seitenplanung bestärkt.

Datenmodellierung

Nachdem Sie den grundlegenden Ablauf der Seite geplant haben, können Sie nun damit beginnen, die Datenbank zu entwerfen. Bedenken Sie, dass die Datenbank der Motor ist, der die Applikation antreibt. Wenn die Datenbank nicht einwandfrei entworfen wurde, wird sie auch nicht einwandfrei laufen. Eine schlecht entwickelte Datenbank ist nicht flexibel und kann sich somit nicht an veränderte Geschäftsanforderungen anpassen. Zeit, die Sie hier investieren, ist gut investierte Zeit.

Aus der Anforderungsphase ergeben sich folgende Geschäftsobjekte:

1. Kunden (Customers)
2. Anzeigen (Ads)
3. Bestellungen (Orders)
4. Produkte (Products)

In dieser Phase des Entwicklungsprozesses entwerfen Sie die Objekte Ihrer Datenbank. Das hilft Ihnen, die Spalten Ihrer Datenbank herauszufinden. Sie haben Näheres über diesen Prozess an Tag 3 gelesen. Indem Sie sich Fragen über die Charakteristika der Objekte gestellt haben, fanden Sie eine geeignete Beschreibung der Geschäftsobjekte heraus (Tabelle 21.1 bis Tabelle 21.4). An dieser Stelle ist es sinnvoll, die Datentypen dieser Charakteristika festzulegen. Es spart Ihnen später Zeit. Beachten Sie, dass der Spaltentyp Ihnen helfen kann, Ihre Ablaufregeln zu bekräftigen.

Charakteristika	Datentyp
First_Name	VARCHAR(15), NOT NULL
Last_Name	VARCHAR(20), NOT NULL
Middle_Initial	CHAR(1)
Address	VARCHAR(50), NOT NULL
City	VARCHAR(30), NOT NULL
State	CHAR(2), NOT NULL
Zip	VARCHAR(15), NOT NULL
Email	VARCHAR(30), NOT NULL
Age	INTEGER
Gender	ENUM('M' , 'F') DEFAULT 'F'
Race	VARCHAR(10)
Eye_Color	VARCHAR(15)
Hair_Color	VARCHAR(15)
Fav_Activity	VARCHAR(30)
Fav_Movie	VARCHAR(30)
Occupation	VARCHAR(50)
Smoker	ENUM('Y' , 'N') DEFAULT 'N'

Tabelle 21.1: Die Customers-Tabelle

Charakteristika	Datentyp
Ad_Text	TEXT, NOT NULL
Date_Posted	DATETIME, NOT NULL

Tabelle 21.2: Die Ads-Tabelle

Charakteristika	Datentyp
Name	VARCHAR(30), NOT NULL
Description	VARCHAR(255), NOT NULL
Price	FLOAT(10,2), NOT NULL
Manufacturer	VARCHAR(50), NOT NULL
Picture	VARCHAR(50), NOT NULL

Tabelle 21.3: Die Products-Tabelle

Charakteristika	Datentyp
Order_Date	DATETIME, NOT NULL
Quantity	INTEGER, NOT NULL
Product	VARCHAR(30), NOT NULL
Customer	VARCHAR(30), NOT NULL
Amount_Due	FLOAT(10,2), NOT NULL
Payment_Received	ENUM('Y' , 'N')

Tabelle 21.4: Die Orders-Tabelle

Jetzt, wo Sie die Tabellen beschrieben haben und die Ablaufregeln bestimmt sind, wird es Zeit, Ihre Objekte in Tabellen umzuwandeln, indem Sie Schlüssel zu jeder Tabelle hinzufügen und Datennormalisierung betreiben.

Normalisierung der Datenbank

Erstellen Sie einen eindeutigen Schlüssel in der Customers-Tabelle und nennen Sie diesen `Customer_ID`. Eine Person kann nur eine Mitgliedschaft haben, daher muss der Schlüssel eindeutig sein. Fügen Sie nun auch den anderen Tabellen eindeutige Schlüssel hinzu und benennen Sie diese nach den Tabellennamen: `Ad_ID`, `Order_ID` und `Product_ID`.

Der nächste Schritt wäre normalerweise die Normalisierung der Datenbank. Da diese Tabellen aber schon normalisiert sind, können Sie diesen Schritt in unserem Beispiel

bei Seite lassen. Die einzige Änderung, die Sie vornehmen müssen, ist, dass Sie die Products- und Customers-Spalten in der Orders-Tabelle in Product_ID und Customer_ID umbenennen.



Um das Beispiel einfach zu halten, wurde diese Datenbank nicht vollständig normalisiert. Die Datenbank könnte noch weiter aufgeteilt werden. Sie könnten z.B. noch eine Manufacturer-Tabelle und eine OrderMaster- und OrderDetail-Tabelle erstellen.

Jetzt müssen Sie die Beziehungen (Relationen), die zwischen den Tabellen bestehen, festlegen. Sie können eine 1:n-Beziehung zwischen der Customers- und der Ads-Tabelle bestimmen. Fügen Sie dazu eine Customer_ID-Spalte in die Ads-Tabelle ein. Auch zwischen der Customers- und Orders-Tabelle und der Products- und Orders-Tabelle besteht jeweils eine 1:n-Beziehung. Da Sie die entsprechenden Spalten in den Tabellen schon haben, müssen Sie diese auch nicht mehr hinzufügen.

Es scheint, dass Sie nunmehr alle Probleme beseitigt haben. Erstellen Sie nun mit einem Modellierungsprogramm ein Modell Ihrer Datenbank. Die folgende Abbildung zeigt eine Microsoft-Access-Präsentation der Datenbank und ihrer Relationen.

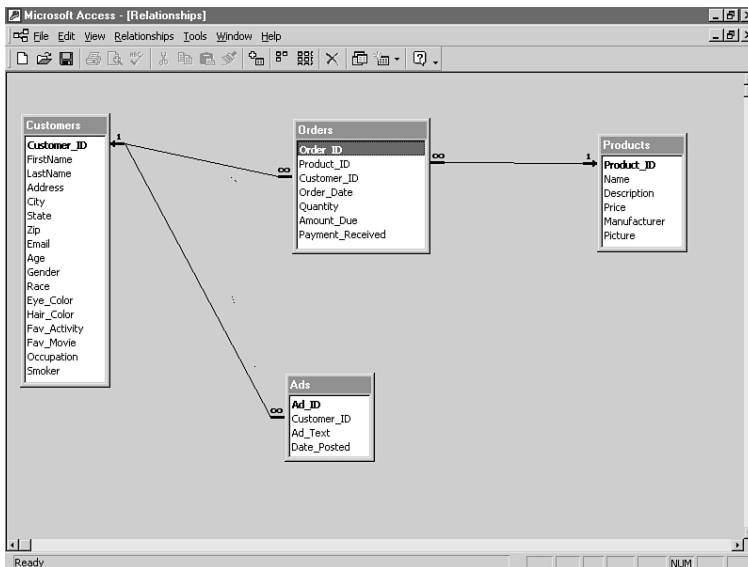


Abbildung 21.2:
Modell der
Meet_A_Geek-
Datenbank

Wenn Sie in einem Team arbeiten, dann ist es immer eine gute Idee, das Team einen Blick auf das Datenbank-Modell werfen zu lassen. Manchmal findet jemand einen Fehler, den Sie selbst übersehen haben. Teilen Sie Ihre Entwicklung mit anderen und lassen Sie andere Ihre Entwicklung kritisieren. Das kann die Struktur Ihrer Datenbank nur verbessern. Wenn Sie mit der Struktur zufrieden sind, ist es an der Zeit, die Datenbank zu erstellen.

Erstellen der Datenbank

Um Ihre Datenbank zu erstellen, muss der `mysqld`-Daemon laufen. Sie benötigen zur Erstellung die `CREATE`-Berechtigung. Wenn diese beiden Voraussetzungen erfüllt sind, können Sie die Datenbank auf zwei verschiedene Arten erstellen: entweder mit dem `mysqladmin`-Programm oder mit der `SQL-CREATE`-Anweisung. Es ist Ihre Entscheidung.

Um die Datenbank mit dem `mysqladmin`-Programm zu erstellen, geben Sie folgende Anweisung an der Konsole ein:

```
%> bin/mysqladmin -p CREATE DATABASE Meet_A_Geek
```

Um die Datenbank mit einer `SQL`-Anweisung zu erstellen, benutzen Sie den `MySQL-Monitor` und geben folgendes ein:

```
mysql> CREATE DATABASE Meet_A_Geek;  
Query OK, 1 row affected (0.17 sec)
```

Erstellen Sie nun die Tabellen und Spalten. Es gibt verschiedene Wege, auf denen Sie dies tun können. Der beste Weg ist, eine Textdatei mit den `SQL`-Anweisungen zu erstellen und diese einfach dem `mysql`-Programm zu übergeben. Auf diese Weise haben Sie immer eine Kopie der Struktur Ihrer Tabellen – nur für den Fall, dass Ihre Datenbank irgendwann einmal so beschädigt wird, dass Sie wieder von Null anfangen müssen. Wenn Sie diesen Weg nicht wählen möchten, können Sie die `SQL`-Anweisungen auch direkt in den `MySQL-Monitor` eingeben. Sie haben an Tag 4 »Das Erstellen Ihrer ersten Datenbank« mehr darüber gelernt. Um Ihr Gedächtnis aufzufrischen, finden Sie im Folgenden die Anweisungen, um die `Orders`-Tabelle zu erstellen:

```
mysql> CREATE TABLE Orders(  
-> Order_ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,  
-> Customer_ID INT NOT NULL,  
-> Product_ID INT NOT NULL,  
-> Order_Date DATETIME NOT NULL,  
-> Quantity INT NOT NULL,  
-> Amount_Due FLOAT(10,2) NOT NULL,  
-> Payment_Received ENUM("Y","N"));
```

Beachten Sie, dass Sie auch zu diesem Zeitpunkt schon Indizes erstellen können. Erinnern Sie sich: Indizes können die Datenbank-Performance bei `SELECT`-Anweisungen immens steigern. Betrachten Sie Ihre Tabellen und wählen Sie mögliche Kandidaten für einen Index aus. `MySQL` indiziert automatisch die Primärschlüssel (Primary Keys) einer Tabelle – also hat die Tabelle, die wahrscheinlich am ehesten einen Index benötigt, die `Ads`-Tabelle, schon einen Index. Die `Ads`-Tabelle ist ein guter Kandidat, weil die Anwender Ihrer Webseite diese Tabelle wohl am häufigsten durchsuchen werden. Sie sollten einen Index auf das Textfeld setzen, um die Performance zu steigern, wenn es nötig wird. Für den Moment sollten Sie alles so belassen, wie es ist, und erst mal abwarten, wie sich die Performance der Datenbank entwickeln wird.

Importieren existierender Daten

Nachdem Sie jetzt die Datenbank erstellt haben, können Sie damit beginnen, diese mit Daten zu füllen. Unter Umständen hat Ihr Kunde bereits Daten in einer Access-Datenbank (oder einer ähnlichen) und möchte diese nun in die neue Datenbank portieren. Wie Sie Daten in eine Datenbank einfügen, wurde ausführlich an Tag 8 diskutiert. Heute wird kurz vorgestellt, wie man Daten importiert.

Wenn Sie noch Access als Datenbank benutzen und jetzt den Schritt in Richtung MySQL gewagt haben, dann haben Sie Glück. Da MySQL ODBC-kompatibel ist, können Sie möglicherweise die Daten relativ einfach in Ihre MySQL-Datenbank importieren – vorausgesetzt Sie haben exakt die gleiche Struktur in beiden Datenbanken. Dazu benutzen Sie einfach das Exportmenü in der Haupteiste. Dies führt Sie durch den Exportvorgang.

Eine weitere Möglichkeit, Daten in eine MySQL-Datenbank zu importieren, ist die Perl DBI. Schreiben Sie ein einfaches Script, das die Daten aus der einen Datenbank auswählt und in die andere Datenbank einfügt.

Mit der `LOAD DATA INFILE`-Anweisung können Sie ebenfalls Daten in Ihre Datenbank importieren. Diese Anweisung können Sie im MySQL-Monitor ausführen. Sie erlaubt Ihnen, formatierte Daten aus einer Datei in eine MySQL-Datenbank zu importieren. Um Daten aus einer Komma-separierten Datei in eine Datenbank zu importieren, führen Sie folgende Anweisung aus:

```
LOAD DATA INFILE './home/olddata.txt'
INTO TABLE Customers IGNORE
FIELDS
TERMINATED BY ',';
```

Diese Anweisung importiert die Datei *olddata.txt* aus dem `home`-Verzeichnis des MySQL-Server-Rechners. Sie lädt alle Daten aus der Datei in die `Customers`-Tabelle und ignoriert dabei alle doppelten Einträge. Der letzte Teil der Anweisung beschreibt, wie die Daten in der Datei getrennt sind.

Manchmal werden Sie Daten vor dem Import bearbeiten müssen, damit diese in die neue Struktur passen. Das können Sie mit einem eigenem Script (PHP/Perl) tun oder mit einem der Programme, die Sie auf der CD-ROM finden. Beachten Sie, dass MySQL Datums-Informationen anders speichert als viele andere Datenbanken. Seien Sie also vorsichtig, wenn Sie Datumsinformationen importieren.

Sicherheitsaspekte

Bevor Sie anfangen, Scripts zu erstellen, um auf die Datenbank zuzugreifen, müssen Sie einen Blick auf einige Sicherheitsaspekte werfen. Wie wollen Sie den Zugang zur Datenbank beschränken bzw. überhaupt handhaben? Um es nicht zu kompliziert zu machen, sollten Sie die MySQL-GRANT-Tabellen benutzen, um den Datenbankzugang zu regeln.

Das Script, das neue Kunden in die Tabellen einfügen soll, wird auf die Datenbank als Benutzer *NewCustomer* zugreifen und nur INSERT-Berechtigungen für die *Customers*-Tabelle haben. Nachdem ein Kunde der *Customers*-Tabelle hinzugefügt wurde, bekommt er einen Benutzernamen und ein Passwort, die ihm ermöglichen, die Datenbank zu durchsuchen, Anzeigen zu erstellen und eigene Anzeigen zu editieren. Der Benutzername ist der Name des Kunden; Vorname und Name sind mit einem Unterstrich verbunden. Das Passwort ist die E-Mail-Adresse. Alle Kunden (Benutzer: *AllCustomers*) haben INSERT-Rechte für die *Ads*-Tabelle, UPDATE-Rechte für einige Spalten der *Customers*-Tabelle und SELECT-Rechte für alle beiden Tabellen. Dieser Benutzer hat auch INSERT-Rechte für die *Orders*-Tabelle, damit der Kunde Geschenke bestellen und versenden kann. Die folgende Tabelle gibt einen Überblick über die Benutzer und ihre Rechte.

Benutzer	Tabelle	Berechtigungen
NewCustomer	Customers	Nur INSERT
AllCustomers	Customers	UPDATE, SELECT
	Orders	INSERT
	Ads	INSERT, UPDATE, DELETE
Staff	Alle Tabellen	Alle Berechtigungen

Tabelle 21.5: Überblick über Benutzer und Berechtigungen der Meet_A_Geek-Datenbank

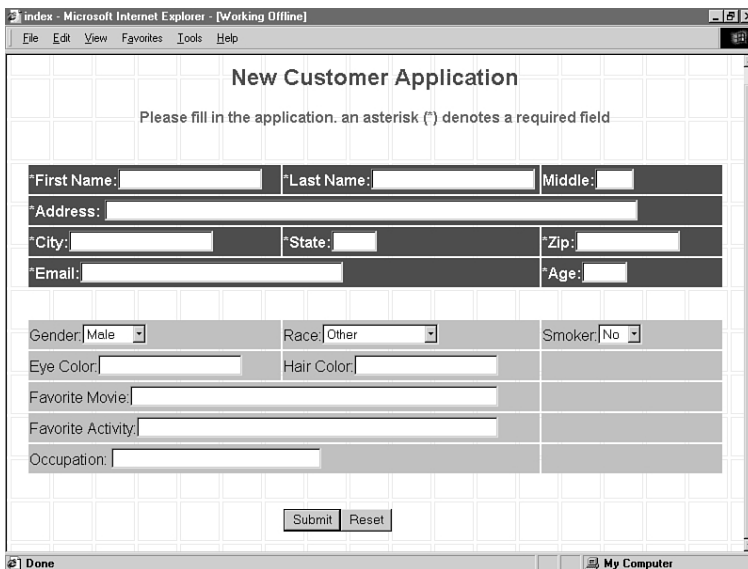
Erstellen der Webseiten und Scripts

Sie sind nun in die vierte Phase des Entwicklungsprozesses eingetreten, dem eigentlichen Erstellen der Datenbank-Applikation. Das ist die längste Phase des Prozesses, aber auch die lohnendste. Hier setzen Sie alles um, was Sie vorher nur geplant haben. Wenn Sie einen Blick auf das Flussdiagramm werfen, können Sie sehen, dass fünf Scripts entwickelt und geschrieben werden müssen, um auf die eine oder andere Weise auf die Datenbank zuzugreifen. Wenn Sie eine Webseite wie diese erstellen, würden Sie normalerweise eine Technologie wählen und bei dieser bleiben. Sie würden zum Beispiel nicht entscheiden, Perl-CGI-Scripts für das eine und Active Server Pages für

das andere Script zu benutzen. Die Technologien überlappen einander. Es würde einfach keinen Sinn ergeben. In diesem Beispiel werden Perl-Scripts verwendet, um zu zeigen, wie einfach Sie mit Perl auf MySQL zugreifen können.

Erstellen der Webseite

Zunächst müssen Sie eine Webseite erstellen, welche die benötigten Daten an ein Perl-Script schicken kann. Anschließend erstellen Sie das Script, mit dem Sie neue Kunden zu Ihrer Datenbank hinzufügen können. Auch hier werden Sie ein Perl-Script verwenden, um diese Aufgabe zu erfüllen. Andere Mitglieder Ihres Teams sind für diesen Teil verantwortlich. Sie haben nach harter Arbeit eine Webseite entwickelt, die Sie in der folgenden Abbildung sehen können.



The screenshot shows a web browser window titled 'index - Microsoft Internet Explorer - [Working Offline]'. The page content is a form titled 'New Customer Application'. Below the title is a note: 'Please fill in the application. an asterisk (*) denotes a required field'. The form contains several input fields and dropdown menus. The fields are: First Name (required), Last Name (required), Middle, Address (required), City (required), State (required), Zip (required), Email (required), Age (required), Gender (dropdown with 'Male' selected), Race (dropdown with 'Other' selected), Smoker (dropdown with 'No' selected), Eye Color, Hair Color, Favorite Movie, Favorite Activity, and Occupation. At the bottom of the form are two buttons: 'Submit' and 'Reset'.

Abbildung 21.3:
Die »Neue Kunden anlegen«-
Applikation der
Meet_A_Geek-
Datenbank

Listing 21.1: HTML-Seite, um neue Benutzer in der Datenbank anzulegen

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<title>New Customer Application</title>
</head>
<body bgcolor="#FFFFFF" text="#000000" link="#999999"
vlink="#990000" alink="#666666">
<div align="center">
```

```
<center>
<table border="0" width="741" height="31">
  <tr>
    <td width="741" height="51" colspan="3">
      <font face="Arial,Helvetica,Geneva" color="#990033" size="5">
        <p align="center">
          <b>New Customer Application</b></font>
        </td>
      </tr>
      <tr>
        <td width="741" height="31" colspan="3">
          <font face="Arial,Helvetica,Geneva" color="#666666">
            <p align="center"><b>
              Please fill in the application. An asterix(*) denotes a
              required field.
            </b></font>
          </td>
        </tr>
        <form action="../cgi-bin/AddCustomer.pl" method = POST>
          <tr>
            <td width="741" height="31" colspan="3">
              <font face="Arial,Helvetica,Geneva"></font>
            </td>
          </tr>
          <tr>
            <td width="262" height="31" bordercolor="#990033" bgcolor="#990033">
              <font face="Arial, Helvetica, Geneva" color="#FFFFFF">
                <b>*First Name:</b>
                <input type="text" name="FirstName" size="20">
              </font>
            </td>
            <td width="268" height="31" bordercolor="#990033" bgcolor="#990033">
              <font face="Arial, Helvetica, Geneva" color="#FFFFFF">
                <b>*Last Name:</b>
                <input type="text" name="LastName" size="23">
              </font>
            </td>
            <td width="188" height="31" bordercolor="#990033" bgcolor="#990033">
              <font face="Arial, Helvetica, Geneva" color="#FFFFFF">
                <b>Middle:</b>
                <input type="text" name="Middle" size="4">
              </font>
            </td>
          </tr>
          <tr>
            <td width="741" height="31" colspan="3" bordercolor="#990033">
```



```

        bgcolor="#990033">
        <font face="Arial, Helvetica, Geneva" color="#FFFFFF">
        <b>*Address:</b>
        <input type="text" name="Address" size="79">
        </font>
        </td>
    </tr>
    <tr>
        <td width="264" height="31" bordercolor="#990033" bgcolor="#990033">
        <font face="Arial, Helvetica, Geneva" color="#FFFFFF">
        <b>*City:</b>
        <input type="text" name="City" size="20">
        </font>
        </td>
        <td width="267" height="31" bordercolor="#990033" bgcolor="#990033">
        <font face="Arial, Helvetica, Geneva" color="#FFFFFF">
        <b>*State:</b>
        <input type="text" name="State" size="5">
        </font>
        </td>
        <td width="187" height="31" bordercolor="#990033" bgcolor="#990033">
        <font face="Arial, Helvetica, Geneva" color="#FFFFFF">
        <b>*Zip:</b>
        <input type="text" name="Zip" size="14">
        </font>
        </td>
    </tr>
    <tr>
        <td width="538" height="31" colspan="2" bordercolor="#990033"
        bgcolor="#990033">
        <font face="Arial, Helvetica, Geneva" color="#FFFFFF">
        <b>*Email:</b>
        <input type="text" name="Email" size="38">
        </font>
        </td>
        <td width="187" height="31" bordercolor="#990033" bgcolor="#990033">
        <font face="Arial, Helvetica, Geneva" color="#FFFFFF">
        <b>*Age:</b>
        <input type="text" name="Age" size="5">
        </font>
        </td>
    </tr>
    <tr>
        <td width="725" height="31" colspan="3">
        <font face="Arial, Helvetica, Geneva"></font>
        </td>
    </tr>

```

```

</tr>
<tr>
<td width="265" height="31" bgcolor="#C0C0C0">
<font face="Arial, Helvetica, Geneva">
Gender:
<select size="1" name="Gender">
  <option value="Male" selected>Male</option>
  <option value="Female">Female</option>
</select>
</font>
</td>
<td width="267" height="31" bgcolor="#C0C0C0">
  <font face="Arial, Helvetica, Geneva">
    Race:
    <select size="1" name="Race">
      <option value="Caucasian">Caucasian</option>
      <option value="African American">African American</option>
      <option value="American Indian">American Indian</option>
      <option value="Hispanic">Hispanic</option>
      <option value="Asian">Asian</option>
      <option value="Other" selected>Andere</option>
    </select>
  </font>
</td>
<td width="187" height="31" bgcolor="#C0C0C0">
  <font face="Arial, Helvetica, Geneva">
    Smoker:
    <select size="1" name="Smoker">
      <option value="Yes">Yes</option>
      <option value="No" selected>No</option>
    </select>
  </font>
</td>
</tr>
<tr>
  <td width="265" height="31" bgcolor="#C0C0C0">
    <font face="Arial, Helvetica, Geneva">
      Eye Color:
      <input type="text" name="EyeColor" size="20">
    </font>
  </td>
  <td width="267" height="31" bgcolor="#C0C0C0">
    <font face="Arial, Helvetica, Geneva">
      Hair Color:
      <input type="text" name="HairColor" size="20">
    </font>
  </td>

```

```
<td width="187" height="31" bgcolor="#C0C0C0">
<font face="Arial, Helvetica, Geneva">&nbsp;</font>
</td>
</tr>
<tr>
<td width="532" height="31" colspan="2" bgcolor="#C0C0C0">
<font face="Arial, Helvetica, Geneva">
Favorite Movie:
<input type="text" name="FavMovie" size="54">
</font>
</td>
<td width="187" height="31" bgcolor="#C0C0C0">
<font face="Arial, Helvetica, Geneva">&nbsp;</font>
</td>
</tr>
<tr>
<td width="532" height="31" colspan="2" bgcolor="#C0C0C0">
<font face="Arial, Helvetica, Geneva">
Favorite Activity:
<input type="text" name="FavActivity" size="53">
</font>
</td>
<td width="187" height="31" bgcolor="#C0C0C0">
<font face="Arial, Helvetica, Geneva">&nbsp;</font>
</td>
</tr>
<tr>
<td width="532" height="31" colspan="2" bgcolor="#C0C0C0">
<font face="Arial, Helvetica, Geneva">
Occupation:
<input type="text" name="Occupation" size="30">
</font>
</td>
<td width="187" height="31" bgcolor="#C0C0C0">
<font face="Arial, Helvetica, Geneva">&nbsp;</font>
</td>
</tr>
<tr>
<td width="532" height="31" colspan="2">
<font face="Arial, Helvetica, Geneva">&nbsp;</font>
</td>
<td width="187" height="31">
<font face="Arial, Helvetica, Geneva">&nbsp;</font>
</td>
</tr>
<tr>
```

```

        <td width="265" height="31">
        <font face="Arial, Helvetica, Geneva">&nbsp;</font>
        </td>
        <td width="267" height="31">
        <font face="Arial, Helvetica, Geneva">
        <input type="submit" value="Submit" name="cmdSubmit">
        <input type="reset" value="Reset" name="cmdReset">
        </font>
        </td>
        <td width="187" height="31">
        <font face="Arial, Helvetica, Geneva">&nbsp;</font>
        </td>
    </tr>
</form>
</table>
<font face="Arial, Helvetica, Geneva">
</center>
</div>
</font>
</body>
</html>

```

Nachdem Sie diese Seite aufgerufen haben, sind das Erste, was Sie beachten sollten, die Namen der `<input>`-Felder. Das sind die Namen, die Sie benötigen, wenn Sie auf die Werte, die von dieser Seite aus dem CGI-Script übergeben werden, um anschließend in die Datenbank eingetragen zu werden, zugreifen möchten. Ein anderer Punkt, den Sie beachten sollten, ist das `<form>`-Tag. Stellen Sie sicher, dass Sie hier auf den richtigen Ort des Perl-CGI-Scripts verweisen und dass die `POST`-Methode verwendet wird.

Erstellen des Scripts

Nun ist es Zeit, das Perl-Script zu erstellen. Da es sich bei der Anwendung um eine Produktionsapplikation handelt, müssen Sie sicherstellen, dass die eingegebenen Daten auch richtig sind (zumindest syntaktisch). Daher müssen Sie die Daten in Ihrem Script überprüfen. Das könnte auch clientseitig erfolgen, aber da Sie ein Server-Script schreiben, sollte die Überprüfung auch hier auf dem Server erfolgen. Das Listing 21.2 ist ein solches serverseitiges Perl-Script.

Listing 21.2: Script zum Anlegen neuer Benutzer

```

#!/usr/local/bin/perl
use DBI;

&GetFormInput;

```

```
## Die Zwischen-Variablen machen Ihr Skript lesbarer, sind  
## aber weniger effizient und nicht wirklich noetig.
```

```
$FirstName = $field{'FirstName'} ;  
$LastName = $field{'LastName'} ;  
$Middle = $field{'Middle'} ;  
$Address = $field{'Address'} ;  
$City = $field{'City'} ;  
$State = $field{'State'} ;  
$Zip = $field{'Zip'} ;  
$Email = $field{'Email'} ;  
$Age = $field{'Age'} ;  
$Gender = $field{'Gender'} ;  
$Race = $field{'Race'} ;  
$Smoker = $field{'Smoker'} ;  
$EyeColor = $field{'EyeColor'} ;  
$HairColor = $field{'HairColor'} ;  
$FavMovie = $field{'FavMovie'} ;  
$FavActivity = $field{'FavActivity'} ;  
$Occupation = $field{'Occupation'} ;
```

```
$message = "" ;  
$found_err = "" ;
```

```
## Ueberprueft die Werte auf Fehler
```

```
$errmsg = "<p>Field'First Name' must be filled.</p>\n" ;
```

```
if ($FirstName eq "") {  
    $message = $message.$errmsg ;  
    $found_err = 1 ; }
```

```
$errmsg = "<p>Field 'Last Name' must be filled.</p>\n" ;
```

```
if ($LastName eq "") {  
    $message = $message.$errmsg ;  
    $found_err = 1 ; }
```

```
$errmsg = "<p>Field 'Address' must be filled.</p>\n" ;
```

```
if ($Address eq "") {  
    $message = $message.$errmsg ;  
    $found_err = 1 ; }
```

```
$errmsg = "<p>Field 'City' must be filled.</p>\n" ;
if ($City eq "") {
    $message = $message.$errmsg ;
    $found_err = 1 ; }

$errmsg = "<p>Field 'State' must be filled.</p>\n" ;

if ($State eq "") {
    $message = $message.$errmsg ;
    $found_err = 1 ; }

$errmsg = "<p>Field 'Zip' must be filled.</p>\n" ;

if ($Zip eq "") {
    $message = $message.$errmsg ;
    $found_err = 1 ; }

$errmsg = "<p>Please enter a valid email address</p>\n" ;

if ($Email !~ /.+\.+\.+\.+\/) {
    $message = $message.$errmsg ;
    $found_err = 1 ; }

$errmsg = "<p>The value in field 'Age' is not valid</p>\n" ;

if ($Age =~ /\D/) {
    $message = $message.$errmsg ;
    $found_err = 1 ; }

elseif ($Age < 0) {
    $message = $message.$errmsg ;
    $found_err = 1 ; }

if ($found_err) {
    &PrintError;
}

# Fuegt die Daten nur dann in die Datenbank ein, wenn sie alle
# Tests bestanden haben.

$DSN = "DBI:mysql:database=Meet_A_Geek";
```

```
my $dbh = DBI->connect($DSN,"NewCustomer","mdew34")
or die "Error conneting to database";

$dbh->do("INSERT INTO Customers
(First_Name, Last_Name, Middle_Initial,
    Address, City, State, Zip, Email, Age,
    Gender, Race, Eye_Color,
    Hair_Color, Fav_Activity, Fav_Movie,
    Occupation, Smoker)
VALUES('$FirstName', '$LastName', '$Middle',
    '$Address','$City', '$State', '$Zip', '$Email', $Age,
    '$Gender','$Race', '$EyeColor', '$HairColor',
    '$FavActivity','$FavMovie', '$Occupation', '$Smoker')");

## Erstellt den Benutzer in der MySQL-Datenbank

$UserName = $FirstName."_".$LastName."@";

$dbh->do("GRANT INSERT, DELETE, SELECT ON Meet_A_Geek.Ads
    TO $UserName IDENTIFIED BY '$Email'");

$dbh->do("GRANT INSERT ON Meet_A_Geek.Orders
    TO $UserName IDENTIFIED BY '$Email'");

$dbi->do("GRANT
    UPDATE (Fav_Movie, Fav_Activity, Occupation, Smoker)
    ON Meet_A_Geek.Customers
    TO $UserName IDENTIFIED BY '$Email'");

$dbi->do("GRANT SELECT ON Meet_A_Geek.Customers
    TO $UserName IDENTIFIED BY '$Email'");

$dbi->do("FLUSH PRIVILEGES");

$dbh->disconnect;

## Wenn alles geklappt hat, wird der Benutzer an diese Adresse
## weitergeleitet.

print "Location: http://CustomerAdd.html\nURI:";
print "http://CustomerAdd.html\n\n" ;

## Wenn ein Uebertragungsfehler vorliegt, wird dem
## Benutzer hierdurch eine Fehlermeldung gesandt.
```

```

sub PrintError {
print "Content-type: text/html\n\n";
print $message ;

exit 0 ;
return 1 ;
}

## Routine, die Namen/Wert-Paare aus der Form extrahiert.
sub GetFormInput {

    (*fval) = @_ if @_ ;

    local ($buf);
    if ($ENV{'REQUEST_METHOD'} eq 'POST') {
        read(STDIN,$buf,$ENV{'CONTENT_LENGTH'});
    }
    else {
        $buf=$ENV{'QUERY_STRING'};
    }
    if ($buf eq "") {
        return 0 ;
    }
    else {
        @fval=split(/&/,$buf);
        foreach $i (0 .. $#fval){
            ($name,$val)=split (/=/, $fval[$i],2);
            $val=~tr/+/ /;
            $val=~ s/%(..)/pack("c",hex($1))/ge;
            $name=~tr/+/ /;
            $name=~ s/%(..)/pack("c",hex($1))/ge;

            if (!defined($field{$name})) {
                $field{$name}=$val;
            }
            else {
                $field{$name} .= ",$val";
            }

        }
    }
}

return 1;
}

```


Dieses Script zeigt, wie man Werte, bevor sie in die Datenbank eingefügt werden, überprüfen kann. Wenn Sie ordentliche Daten aus Ihrer Datenbank herausbekommen möchten, dann müssen Sie auch ordentliche Daten einfügen. Zusätzlich wird in dem Script mit der `GRANT`-Anweisung der Benutzer erstellt und es werden Rechte an ihn vergeben. Bevor Sie vom Server getrennt werden, wird zudem die `FLUSH`-Anweisung ausgeführt, damit der neue Benutzer sofort Zugriff auf die Datenbank erhält. Alles andere, bis auf den Teil zur Fehlerüberprüfung und zur Benutzeranlage, sollte Ihnen schon bekannt vorkommen.

Als Benutzer in das System einloggen

Der nächste Punkt, den Sie in Angriff nehmen müssen, ist der Benutzer-Login. Die Herausforderung ist hier, den Benutzernamen und das Passwort zu erhalten und diese Identität irgendwie über die komplette Benutzer-Sitzung festzuhalten. Es gibt mehrere Wege, um diese Aufgabe zu meistern. Einer ist die Verwendung von Cookies, um die Werte von Seite zu Seite zu übergeben. Aber es gibt ein paar Schönheitsfehler bei dieser Variante. Was passiert, wenn der Benutzer Cookies deaktiviert hat? Dann sind Sie aufgeschmissen. Einen anderen Weg, um einen Benutzer auf der Webseite zu verfolgen und benutzerrelevante Daten über die gesamte Benutzersitzung zu erhalten, sind Sessions. Diese werden zum Beispiel von PHP ab Version 4 zur Verfügung gestellt und stellen die beste Variante dar. In unserem Beispiel werden Sie trotzdem mit `<hidden>`-Feldern arbeiten, um es nicht zu kompliziert zu machen. Da Anwender sich erst einloggen müssen, um auf eine bestimmte Seite zuzugreifen, könnten Sie die Seite dynamisch generieren und die Informationen in `<hidden>`-Feldern übergeben. Klickt ein Benutzer zum Beispiel den »Anzeige aufgeben«-Link an, wird er auf eine Login-Webseite geführt, auf der er seinen Benutzernamen und sein Passwort eingeben muss. Ein Perl-Script vergleicht anschließend die Daten mit denen in der MySQL-Datenbank und erstellt bei Erfolg die »Anzeige aufgeben«-Seite. In dieser HTML-Seite könnten Sie zwei `<hidden>`-Felder haben, von denen das eine den Benutzernamen und das andere das Passwort enthalten könnte. Wenn der Benutzer seine Anzeige fertig hat und auf »Abschicken« drückt, werden diese Daten an den Webserver geschickt. Die Informationen in den `<hidden>`-Felder dienen dazu, Zugang zur MySQL-Datenbank zu erhalten. Das scheint jetzt vielleicht alles ein wenig verwirrend, aber ein Blick auf den Quellcode wird Ihnen helfen, die Dinge zu verstehen.

Ein Anwender möchte eine Anzeige aufgeben. Gemäß Ihren Ablaufregeln muss dieser Anwender erst ein Mitglied werden. Um die Identität zu überprüfen, muss sich der Anwender zunächst einloggen. Das geschieht über die `LogIn`-Webseite, die Sie in der folgenden Abbildung sehen.

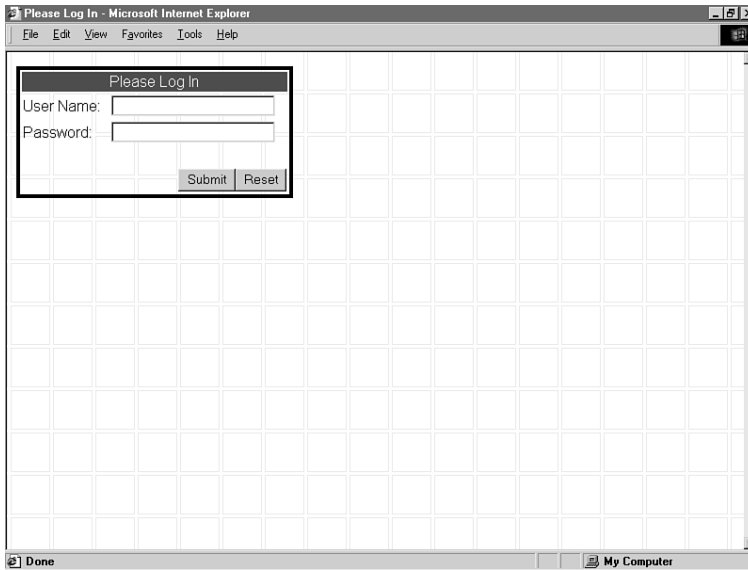


Abbildung 21.4:
Das Login-
Formular

Listing 21.3: HTML-Seite für den Benutzer-Login

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<title>Log In</title>
</head>

<body>
<div align="left">
<form action="../cgi-bin/login.pl" method=post>
  <table border="0" width="294" height="21"
style="border-style:solid; border-color:#000000">
    <tr>
      <td width="294" height="21" bgcolor="#990033" colspan="2">
        <p align="center"><font color="#FFFFFF">
          Please Log in</font>
        </td>
      </tr>
      <tr>
        <td width="91" height="21">
          User Name:
        </td>
        <td width="187" height="21">
          <input type="text" name="Username" size="23">

```

[illegible]

Nach dem Drücken des »Einloggen«-Buttons wird das Script `Login.pl` aufgerufen. Dieses Script nimmt die Werte aus dem Benutzernamen- und Passwort-Feld und überprüft sie auf Gültigkeit gegen die MySQL-Datenbank. Wenn eine Übereinstimmung gefunden wird (also die Werte richtig sind), wird die angeforderte Seite generiert. Wurde keine Übereinstimmung gefunden, wird der Benutzer erneut aufgefordert, seine Daten einzugeben.

Listing 21.4: Script für den Benutzer-Login

```
#!/usr/local/bin/perl
&GetFormInput;
```

```
## Die Zwischen-Variablen machen Ihr Skript lesbarer, sind
## aber weniger effizient und nicht wirklich noetig.
```

```
$Username = $field{'Username'} ;
$Password = $field{'Password'} ;
```

```
$message = "" ;
$found_err = "" ;

$errmsg = "<p>Field 'User Name' must be filled in.</p>\n" ;

if ($Username eq "") {
    $message = $message.$errmsg ;
    $found_err = 1 ; }

$errmsg = "<p>Field 'Password' must be filled in.</p>\n" ;

if ($Password eq "") {
    $message = $message.$errmsg ;
    $found_err = 1 ; }

if ($found_err) {
    &PrintError; }

$DSN = "DBI:mysql:database=Meet_A_Geek";

$message = "Invalid Log In - Try Again";

my $dbh = DBI->connect($DSN, $Username, $Password)
or &PrintError);

$dbh->disconnect;

print "Content-type: text/html\n\n";
print "<html>\n" ;
print "\n" ;
print "<head>\n" ;
print "<title>Post Ad</title>\n" ;
print "</head>\n" ;
print "\n" ;
print '<body bgcolor="#FFFFFF" text="#000000" link="#999999";
print ' vlink="#990000" alink="#666666">';
print '<font face="Arial, Helvetica, Geneva">'. "\n" ;
print '<form action="../cgi-bin/PostAd" method=post>'. "\n" ;
print '<div align="left">'. "\n" ;
print '<input type="hidden" name="Username" value="$Username">'. "\n" ;
print '<input type="hidden" name="Password" value="$Password">'. "\n" ;
print ' </font>';
print '<table border="0" width="498" height="30" ' ;
print 'style="border-style: solid; border-color: #000000">'. "\n" ;
```

```

print "    <tr>\n" ;
print '        <td width="498" height="20">';
print '<font face="Arial, Helvetica, Geneva">'. "\n" ;
print '<p align="center">';
print '<bEnter the following information to '. "\n";
print ' Post an Ad</b>';
print '</font></td>'. "\n" ;
print "    </tr>\n" ;
print "    <tr>\n" ;
print '        <td width="498" height="20">';
print '<font face="Arial, Helvetica, Geneva">';
print '</font></td>'. "\n" ;
print "    </tr>\n" ;
print "    <tr>\n" ;
print '        <td width="498" height="20">';
print '<font face="Arial, Helvetica, Geneva">';
print '<b><font size="2">Use this box to form a'. "\n" ;
print 'message for your ad. You can say anything you like.';
print '</font></b></font></td>\n" ;
print "    </tr>\n" ;
print "    <tr>\n" ;
print '        <td width="498" height="132" valign="top">';
print '<font face="Arial, Helvetica, Geneva">';
print '<textarea rows="6" name="S1" cols="61"></textarea>';
print '</font></td>'. "\n" ;
print "    </tr>\n" ;
print "    <tr>\n" ;
print '        <td width="498" height="30" valign="top">';
print '<font face="Arial, Helvetica, Geneva">'. "\n" ;
print '        <p align="right">';
print '<input type="submit" value="Submit">';
print '<input type="reset" value="Reset">';
print '</font></td>'. "\n" ;
print "    </tr>\n" ;
print '    </table>';
print '<font face="Arial, Helvetica, Geneva">'. "\n" ;
print "</div>\n" ;
print "</form>\n" ;
print "</font></body>\n" ;
print "\n" ;
print "</html>\n" ;

sub PrintError {
print "Content-type: text/html\n\n";
print "<html>\n" ;
print "\n" ;

```

[illegible]

```

print "    </tr>\n" ;
print "    <tr>\n" ;
print '        <td width="91" height="21">';
print '<font face="Arial, Helvetica, Geneva">';
print '</font></td>'. "\n" ;
print '        <td width="187" height="21">';
print '<font face="Arial, Helvetica, Geneva">';
print '</font></td>'. "\n" ;
print "    </tr>\n" ;
print "    <tr>\n" ;
print '        <td width="91" height="21">';
print '<font face="Arial, Helvetica, Geneva">';
print '</font></td>'. "\n" ;
print '        <td width="187" height="21">';
print '<font face="Arial, Helvetica, Geneva">'. "\n" ;
print '            <p align="right">'
print '<input type="submit" value="Submit" name="B1">';
print '<input type="reset" value="Reset" name="B2">';
print '</font></td>'. "\n" ;
print "    </tr>\n" ;
print ' </table>'
print '<font face="Arial, Helvetica, Geneva">'. "\n" ;
print "</form>\n" ;
print "</div>\n" ;
print "\n" ;
print "</font></body>\n" ;
print "\n" ;
print "</html>\n" ;

```

```

exit 0 ;
return 1 ;
}

```

```

sub GetFormInput {

    (*fval) = @_ if @_ ;

    local ($buf);
    if ($ENV{'REQUEST_METHOD'} eq 'POST') {
        read(STDIN,$buf,$ENV{'CONTENT_LENGTH'});
    }
    else {
        $buf=$ENV{'QUERY_STRING'};
    }
    if ($buf eq "") {
        return 0 ;
    }
}

```

```

else {
    @fval=split(/&/,$buf);
    foreach $i (0 .. $#fval){
        ($name,$val)=split (/=/,$fval[$i],2);
        $val=~tr/+// ;
        $val=~ s/%(..)/pack("c",hex($1))/ge;
        $name=~tr/+// ;
        $name=~ s/%(..)/pack("c",hex($1))/ge;

        if (!defined($field{$name})) {
            $field{$name}=$val;
        }
        else {
            $field{$name} .= ",$val";
        }
    }
}
return 1;
}

```

Dieses Script überprüft zunächst, ob die Einträge leer sind oder nicht. Wenn Sie leer sind, wird die `PrintError`-Subroutine aufgerufen. Diese Routine generiert die Login-Seite nochmals, diesmal jedoch mit einer Fehlermeldung. Sind die Einträge nicht leer, wird versucht, mit den übergebenen Werten (Benutzername und Passwort) eine Verbindung zur MySQL-Datenbank herzustellen. War der Verbindungsaufbau erfolgreich, werden die `$Username`- und `$Password`-Variablen in die `<hidden>`-Felder eingefügt und die `PostAd.html`-Seite wird generiert. Ist kein Verbindungsaufbau zustande gekommen, wird die Subroutine `PrintError` aufgerufen. Dies ist ein kleiner Trick, um eine sichere Umgebung für Ihre Datenbank zu erzeugen und den Benutzer so wenig wie möglich zu belästigen.

Anzeigen abfragen

Betrachten Sie nun das Suchformular. Dieses Formular ermöglicht dem Benutzer, die Datenbank nach Anzeigen und Bekanntschaften zu einem passenden Datum zu durchsuchen. Der Benutzer hat die Möglichkeit, verschiedene Optionen in dem Formular zu wählen, die als Grundlage für die Suche dienen. Aus ihnen wird später die SQL-Anweisung generiert. Die Ergebnisseite zeigt daraufhin alle Übereinstimmungen in der Datenbank. Werfen Sie einen Blick auf den HTML-Code in Listing 21.5:

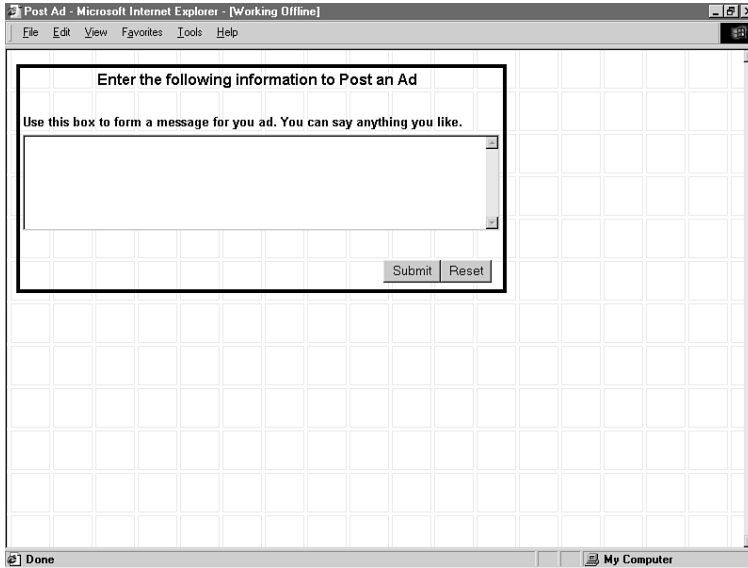


Abbildung 21.5:
Die »Anzeigen
aufgeben«-Appli-
kation der
Meet_A_Geek-
Datenbank

Listing 21.5: Suchformular

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">

<html>
<head>
  <title>Search Ads</title>
</head>

<body>

<div align="center">
  <center>
<form action="../cgi-bin/SearchAd.pl" method=post>
  <table border="0" width="521" height="25"
style="border-style: solid; border-color: #000000">
    <tr>
      <td width="521" height="25" bgcolor="#990033">
        <p align="center"><font color="#FFFFFF">
Select from the following
choices to find a perfect mate!!</font></td>
      </tr>
      <tr>
        <input type="hidden" name="Username" value="">
        <input type="hidden" name="Password" value="">
```

```

        <td width="521" height="25"></td>
    </tr>
    <tr>
        <td width="521" height="25">I am looking for a
<select size="1" name="Gender">
    <option value="M">Man</option>
    <option value="F" selected>Woman</option>
</select> between the ages of <select size="1" name="LAge">
    <option value="18">18</option>
    <option value="F">19</option>
    <option value="18" selected>Any</option>
</select>and <select size="1" name="UAge">
    <option value="1" selected>Any</option>
    <option value="18">18</option>
    <option value="19">19</option>
</select>.</td>
    </tr>
    <tr>
        <td width="521" height="25">My perfect mate will have
<select size="1" name="Hair">
    <option value="Brown">Brown</option>
    <option selected>Any Color</option>
    <option value="Black">Black</option>
    <option value="Blonde">Blonde</option>
</select>hair.&nbsp;   My perfect mate will also </td>
    </tr>
    <tr>
        <td width="521" height="25">have <select size="1" name="Eyes">
    <option value="Brown">Brown</option>
    <option selected>Any Color</option>
    <option value="Blue">Blue</option>
    <option value="Green">Green</option>
</select> eyes. They <select size="1" name="Smoke">
    <option value="Y">Y</option>
    <option selected value="N">N</option>
</select> smoke and they are of the</td>
    </tr>
    <tr>
        <td width="521" height="25"><select size="1" name="Race">
    <option selected>Any</option>
    <option value="Caucasian">Caucasian</option>
    <option value="African American">African American</option>
    <option value="American Indian">American Indian</option>
    <option value="Hispanic">Hispanic</option>
    <option value="Asian">Asian</option>
    <option value="Other">Andere</option>
</select> ethnic group. </td>

```

```

        </tr>
        <tr>
            <td width="521" height="25"></td>
        </tr>
        <tr>
            <td width="521" height="25">
Enter a phrase that describe the person you're
looking for. For </td>
        </tr>
        <tr>
            <td width="521" height="25">
example- Likes the outdoors, Athletic, Fireside chats, etc.</td>
        </tr>
        <tr>
            <td width="521" height="25">
<input type="text" name="Phrase" size="69"></td>
        </tr>
        <tr>
            <td width="521" height="25"></td>
        </tr>
    </center>
    <tr>
        <td width="521" height="25">
            <p align="right"><input type="submit" value="Submit"
name="B1"><input type="reset" value="Reset" name="B2"></td>
        </tr>
    </table>
</div>
</body>

</html>

```

Listing 21.6 zeigt das komplette Script für die Suche über die Datenbank:

Listing 21.6: Script zur Auswertung des Suchformulars

```

#!/usr/local/bin/perl

&GetFormInput;

## Diese Zwischen-Variablen machen Ihr Skript lesbarer, sind
## aber weniger effizient und nicht wirklich noetig.

$Username = $field{'Username'} ;
$Password = $field{'Password'} ;
$Gender = $field{'Gender'} ;
$LAge = $field{'LAge'} ;

```

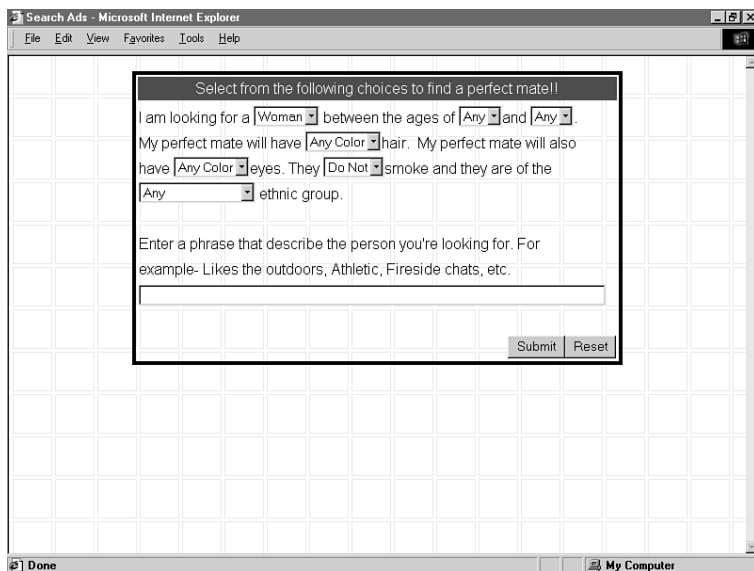


Abbildung 21.6:
Das Such-
formular

```
$UAge = $field('UAge') ;
$Hair = $field('Hair') ;
$Eyes = $field('Eyes') ;
$Smoke = $field('Smoke') ;
$Race = $field('Race') ;
$Phrase = $field('Phrase') ;

$message = "" ;
$found_err = "" ;

$errmsg = "<p>Field 'User Name' must be filled in.</p>\n" ;

if ($Username eq "") {
    $message = $message.$errmsg ;
    $found_err = 1 ; }

$errmsg = "<p>Field 'Password' must be filled in.</p>\n" ;

if ($Password eq "") {
    $message = $message.$errmsg ;
    $found_err = 1 ; }

if ($found_err) {
```

```
&PrintError; }

## Zusammensetzen der SQL-Anweisung

$SQL = "SELECT * FROM Customers as C, Ads as A
        WHERE A.Customer_ID = C.Customer_ID ";

$Build = "AND C.Gender = 'M' ";

if ($Gender eq "M"){
    $SQL = $SQL.$Build;}

$Build = "AND C.Gender = 'F' ";

if ($Gender eq "F"){
    $SQL = $SQL.$Build;}

$Build = "AND C.Age >= $LAge AND C.Age <= $UAge ";
$SQL = $SQL.$Build;

$Build = "AND C.Hair_Color ='$Hair' ";
if ($Hair){
    $SQL = $SQL.$Build;}

$Build = "AND C.Eye_Color ='$Eyes' ";
if ($Eyes){
    $SQL = $SQL.$Build;}

$Build = "AND C.Smoke ='$Smoke' ";
$SQL = $SQL.$Build;

$Build = "AND C.Race = '$Race' ";
if ($Race){
    $SQL = $SQL.$Build;}

$Build = "AND A.Ad_Text LIKE '%$Phrase%' "
if ($Phrase){
    $SQL = $SQL.$Build;}

## Verbindung mit der Datenbank aufnehmen

$DSN = "DSN:mysql:database=Meet_A_Geek";

$message = "Error Connecting to Database";

my $dbh = DBI->connect($DSN, $Username, $Password)
```

```

        or &PrintError;
my $sth = $dbh->prepare($SQL);
$sth->execute();

print "Content-type: text/html\n\n";
print "<html>\n" ;
print "\n" ;
print "<head>\n" ;
print "<title>Your Search produced the '";
print 'Following Results</title>\n" ;
print "</head>\n" ;
print "\n" ;
print '<body bgcolor="#FFFFFF" text="#000000" link="#999999" ' ;
print 'vlink="#990000" alink="#666666">';
print '<font face="Arial, Helvetica, Geneva">'. "\n" ;
print "\n" ;
print '<div align="center">'. "\n" ;
print "  <center>\n" ;
print '  </font>';
print '<table border="1" width="508" height="17" ' ;
print 'bordercolordark="#666666" bordercolorlight="#CCCCCC">'. "\n" ;
print "    <tr>\n" ;
print '      <td width="507" height="17" bgcolor="#990033" ' ;
print 'colspan="5">';
print '<font face="Arial, Helvetica, Geneva">'. "\n" ;
print '      <p align="center"><font color="#FFFFFF">';
print '<b>Yours Search produced the ' ". "\n" ;
print "      Following Results:</b></font>\n";
print "</font></td>\n" ;
print "    </tr>\n" ;
print "    <tr>\n" ;
print '      <td width="507" height="17" ' ;
print 'bgcolor="#FFFFFF" colspan="5">';
print '<font face="Arial, Helvetica, Geneva">&nbsp;';
print '</font></td>'. "\n" ;
print "    </tr>\n" ;
print "    <tr>\n" ;
print '      <td width="186" height="17" bgcolor="#FFFFFF">';
print '<font face="Arial, Helvetica, Geneva">';
print '<font color="#666666"><b>Name</b></font>';
print '</font></td>'. "\n" ;
print '      <td width="52" height="17" bgcolor="#FFFFFF">';
print '<font face="Arial, Helvetica, Geneva">';
print '<font color="#666666"><b>Age</b></font>';
print '</font></td>'. "\n" ;
print '      <td width="98" height="17" bgcolor="#FFFFFF">';

```

```

print '<font face="Arial, Helvetica, Geneva">';
print '<font color="#666666"><b>City</b></font>';
print '</font></td>'. "\n" ;
print '      <td width="41" height="17" bgcolor="#FFFFFF">';
print '<font face="Arial, Helvetica, Geneva">';
print '<font color="#666666"><b>State</b></font>';
print '</font></td>'. "\n" ;
print '      <td width="95" height="17" bgcolor="#FFFFFF">';
print '<font face="Arial, Helvetica, Geneva">';
print '<font color="#666666"><b>View'. "\n" ;
print "          </b></font></font></td>\n" ;
print "      </tr>\n" ;

```

Das folgende Stueck Code produziert das Anzeige-Muster.
 ## Es benutzt die Mod-Funktion, um zu testen, ob eine
 ## Zeile gerade oder ungerade ist, und passt die Farben
 ## dementsprechend an.

```

while($ref = $sth->fetchrow_hashref()){

    if($x % 2 > 0){
        $bg = "#666666";
        $ft = "#FFFFFF";
    }
    else{
        $bg = "#FFFFFF";
        $ft = "#000000";
    }

    print "      <tr>\n" ;
    print '          <td width="186" height="17" bgcolor=$bg>';
    print '<font face="Arial, Helvetica, Geneva">';
    print '<font color=$ft><b>$ref->{'First_Name'} $ref->{'Last_Name'}';
    print '</b></font>';
    print '</font></td>'. "\n" ;
    print '      <td width="52" height="17" bgcolor=$bg>';
    print '<font face="Arial, Helvetica, Geneva">';
    print '<font color=$ft>$ref->('Age')</font>';
    print '</font></td>'. "\n" ;
    print '      <td width="98" height="17" bgcolor=$bg>';
    print '<font face="Arial, Helvetica, Geneva">';
    print '<font color=$ft><b>$ref->('City')</b></font>';
    print '</font></td>'. "\n" ;
    print '      <td width="41" height="17" bgcolor=$bg>';
    print '<font face="Arial, Helvetica, Geneva">';
    print '<font color=$ft><b>$ref->('State')</b></font>';
    print '</font></td>'. "\n" ;

```

```
print '      <td width="95" height="17" bgcolor=$bg>';
print '<font face="Arial, Helvetica, Geneva">'. "\n" ;
print '      <p align="center"><font color="#000000">';
print '<b><a href="../../cgi-bin/viewad.pl?ref->{'Customer_ID'}&';
print 'Username=$Username&Password=$Password">Ad</a>';
print '</b></font></font></td>'. "\n" ;
print "      </tr>\n" ;
```

```
$x = $x +1;
```

```
}
```

```
$sth->finish();
```

```
$dbh->disconnect();
```

```
print "      <tr>\n" ;
print '      <td width="186" height="17" bgcolor="#FFFFFF">';
print '<font face="Arial, Helvetica, Geneva">&nbsp;';
print '</font></td>'. "\n" ;
print '      <td width="52" height="17" bgcolor="#FFFFFF">';
print '<font face="Arial, Helvetica, Geneva">&nbsp;';
print '</font></td>'. "\n" ;
print '      <td width="98" height="17" bgcolor="#FFFFFF">';
print '<font face="Arial, Helvetica, Geneva">&nbsp;';
print '</font></td>'. "\n" ;
print '      <td width="41" height="17" bgcolor="#FFFFFF">';
print '<font face="Arial, Helvetica, Geneva">&nbsp;';
print '</font></td>'. "\n" ;
print '      <td width="95" height="17" bgcolor="#FFFFFF">';
print '<font face="Arial, Helvetica, Geneva">&nbsp;';
print '</font></td>'. "\n" ;
print "      </tr>\n" ;
print '    </table>';
print '<font face="Arial, Helvetica, Geneva">'. "\n" ;
print "    </center>\n" ;
print "</div>\n" ;
print "\n" ;
print "</font></body>\n" ;
print "\n" ;
print "</html>\n" ;
```

```
## Wenn ein Fehler auftritt, wird der Benutzer zurueck zur
## Login-Seite geleitet.
```

```
sub PrintError {
```


[illegible]

```
print '      <td width="187" height="21">';
print '<font face="Arial, Helvetica, Geneva">';
print '<input type="text" name="Password" size="23">';
print '</font></td>'. "\n" ;
print "      </tr>\n" ;
print "      <tr>\n" ;
print '      <td width="91" height="21">';
print '<font face="Arial, Helvetica, Geneva">';
print '</font></td>'. "\n" ;
print '      <td width="187" height="21">';
print '<font face="Arial, Helvetica, Geneva">';
print '</font></td>'. "\n" ;
print "      </tr>\n" ;
print "      <tr>\n" ;
print '      <td width="91" height="21">';
print '<font face="Arial, Helvetica, Geneva">';
print '</font></td>'. "\n" ;
print '      <td width="187" height="21">';
print '<font face="Arial, Helvetica, Geneva">'. "\n" ;
print '      <p align="right">';
print '<input type="submit" value="Submit" name="B1">';
print '<input type="reset" value="Reset" name="B2">';
print '</font></td>'. "\n" ;
print "      </tr>\n" ;
print '    </table>';
print '<font face="Arial, Helvetica, Geneva">'. "\n" ;
print "</form>\n" ;
print "</div>\n" ;
print "\n" ;
print "</font></body>\n" ;
print "\n" ;
print "</html>\n" ;
```

```
exit 0 ;
return 1 ;
}
```

```
sub GetFormInput {

    (*fval) = @_ if @_ ;

    local ($buf);
    if ($ENV{'REQUEST_METHOD'} eq 'POST') {
        read(STDIN,$buf,$ENV{'CONTENT_LENGTH'});
    }
    else {
```

```

        $buf=$ENV{'QUERY_STRING'};
    }
    if ($buf eq "") {
        return 0 ;
    }
    else {
        @fval=split(/&/,$buf);
        foreach $i (0 .. $#fval){
            ($name,$val)=split (/=/,$fval[$i],2);
            $val=~tr/+/ /;
            $val=~ s/%(..)/pack("c",hex($1))/ge;
            $name=~tr/+/ /;
            $name=~ s/%(..)/pack("c",hex($1))/ge;

            if (!defined($field{$name})) {
                $field{$name}=$val;
            }
            else {
                $field{$name} .= ",$val";
            }

        }
    }
    return 1;
}

```

Dieses Listing beinhaltet eine Menge Programmcode. Der interessanteste Teil ist das Zusammensetzen der SQL-Anweisung. Diese Anweisung wird aufgrund der Optionen erstellt, die der Benutzer zuvor ausgewählt hat. Einfache if-Anweisungen kontrollieren, was in die Anweisung kommt. Die SQL-Anweisung selbst ist eine einfache SELECT-Anweisung, die Informationen aus der `Customers`-Tabelle in Verknüpfung mit der `Ads`-Tabelle abfragt. Das ist sinnvoll, da Sie nur auf diese Weise Daten von Kunden erhalten, die auch Anzeigen aufgegeben haben. Das Ergebnis der Abfrage wird in der Variablen `$sth` gespeichert. Wenn kein Fehler aufgetreten ist, generiert das Script die Ergebnisseite. Diese Seite enthält eine Tabelle mit Daten und alternierenden Zeilenfarben. Das wird durch die Überprüfung der Zeilen auf »gerade« bzw. »ungerade« ermöglicht. Wenn die Zeile eine gerade Zeilennummer hat, wird die Hintergrundfarbe auf Dunkelgrau und die Schriftfarbe auf Weiß gesetzt. Ist die Zeilennummer ungerade, so ist die Hintergrundfarbe Weiß und die Schriftfarbe Schwarz. Dieser kleine Trick ist einfach zu realisieren und verbessert die Seite, die an den Benutzer zurückgesandt wird.

Die Zellen der Tabelle sind mit den Werten aus dem Ergebnis der Datenbankabfrage gefüllt. Die `while`-Schleife ermöglicht, durch das Ergebnis zu »gehen« und die Werte, die aus der SELECT-Anweisung resultieren, auf der Seite auszugeben.

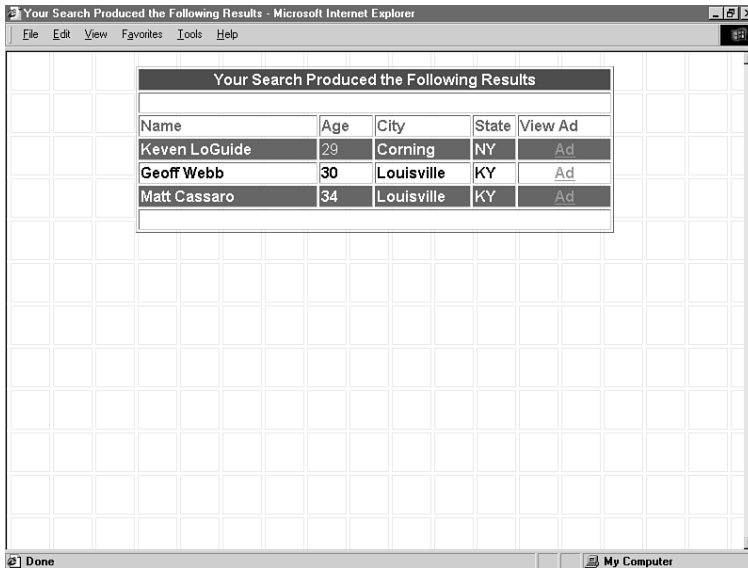


Abbildung 21.7:
Suchergebnis-
anzeige

Testen und Online stellen

Der letzte Schritt im Entwicklungsprozess besteht darin, Ihre Programme zu testen und sie anschließend online zu stellen. Wenn Sie Ihren Programmcode komplett durchgetestet haben, geben Sie ihn an jemand anderen weiter. Sie können darauf wetten, dass derjenige in neun von zehn Fällen Fehler finden wird, die Ihnen zuvor nicht aufgefallen waren. Der Programmierer selbst ist selten ein guter Tester. Er weiß, wie seine Applikation laufen soll, und daher wird sie auch daraufhin geprüft. Er probiert keine unerwarteten Dinge. Der Benutzer hingegen schon. Testen Sie Ihre Scripts sehr genau, bevor Sie diese online stellen. Testen Sie Ihre Scripts und Webseiten auch auf verschiedenen Browsern (Internet Explorer und Netscape). Sie werden überrascht sein, dass einige Seiten auf einem Browser perfekt und auf einem anderen schrecklich aussehen. Abschließend testen Sie Ihre Ausgaben auf verschiedenen Bildschirmauflösungen. Diese bestimmen oftmals zum großen Teil, wie eine Webseite aussieht.

Im letzten Schritt stellen Sie Ihre Applikation online. Hier spielen Sie Ihre Daten auf einen Produktionsserver und machen die Seite der Öffentlichkeit zugänglich – Ihr ultimatives Ziel. Die ganze Arbeit und all die Stunden der Entwicklung zielen alle auf diesen einen Moment ab. Daher sollten Sie an dieser Stelle keine Überraschungen mehr erleben. Sie sollten vorab wissen, ob Ihr Projekt ein Erfolg oder Misserfolg wird. Wenn Sie dem hier geschilderten Entwicklungsprozess gefolgt sind, können Sie schon fast sicher sein, dass Ihr Projekt ein Erfolg wird.

Zusammenfassung

Heute haben Sie einiges über den Entwicklungsprozess und die Schritte, die darin enthalten sind, erfahren. Sie haben gelernt, wie die Datenbank-Planung mit dem Entwicklungsprozess zusammenhängt und wie Ihnen dies hilft, Ihr Ziel zu erreichen. Sie haben auch gelernt, wie wichtig die Planung ist. Mit einer guten Planung ist es ein einfacher Weg von der Idee zum fertigen System.

Sie haben heute auch die Schlüsselereignisse der letzten drei Wochen rekapituliert. Sie haben einen Rückblick auf das Erstellen von Datenbanken, Tabellen und Spalten geworfen. Zudem haben Sie gesehen, wie Daten in eine bestehende Datenbank importiert werden. Auch wurde auf Sicherheitsaspekte und ihre Implementierung eingegangen. Schließlich haben Sie eine Reihe von Webseiten und Perl-Scripts erstellt, die Ihre Pläne Wirklichkeit werden ließen.

Fragen und Antworten

- F** Die `Products`-Tabelle hat eine Bild-Spalte. Wie werden Bilder in MySQL gespeichert?
- A** *MySQL kann Bilder als `BLOBs` speichern. In den Newsgroups wird immer wieder heiß diskutiert, ob es besser ist, ein Bild direkt in der Datenbank oder lediglich einen Link auf das Verzeichnis zu speichern, in dem sich das Bild befindet. Das Hauptargument für die letztgenannte Variante ist, dass der Datentyp `BLOB` ineffizient ist. Er verbraucht zu viel Platz und kann nicht optimiert werden (zum Beispiel mit der `OPTIMIZE`-Anweisung). Wie dem auch sei – das Dateisystem ist wahrscheinlich der beste Platz, um ein Bild abzuspeichern.*
- F** Wie oft sollte meine Datenbank optimiert werden?
- A** *Sie sollten Ihre Datenbank regelmäßig optimieren, besonders, wenn Ihre Datenbank noch neu ist. Die `OPTIMIZE`-Anweisung wird vor allem benötigt, wenn eine große Anzahl an Datensätzen in eine Datenbank eingefügt bzw. aus dieser gelöscht wird.*

Übungen

1. Zwei Scripts aus dem Flussdiagramm haben Sie noch nicht erstellt: ein Script, um die Einträge der `Products`-Tabelle anzuzeigen, und eines, um Bestellungen aufzugeben. Erstellen Sie diese beiden Scripts und verwenden Sie dabei Perl und die DBI/DBD.
2. Erstellen Sie die Scripts aus der vorherigen Übung. Benutzen Sie jedoch dieses Mal ADO und ODBC.

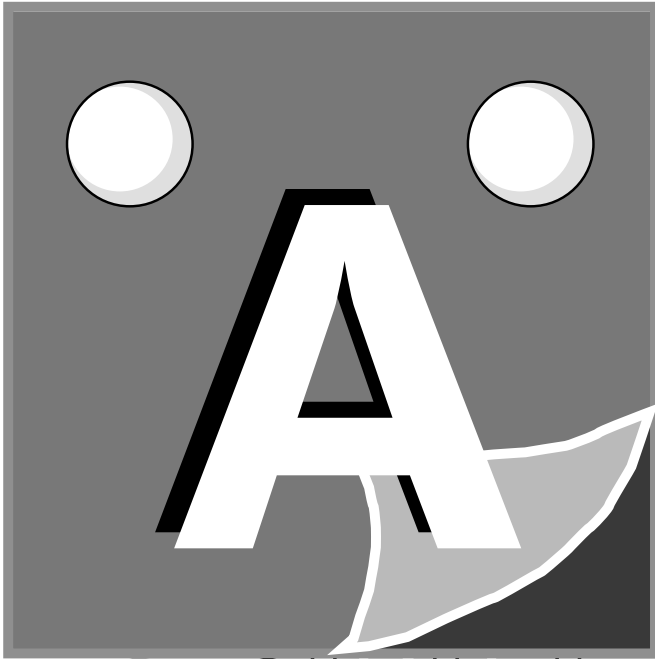


Woche

Rückblick



Sie haben die dritte und letzte Woche mit dem Erlernen von MySQL abgeschlossen. Sie haben am Anfang der Woche einen Blick auf oft verwendete Benutzer-Interfaces geworfen und in der Mitte der Woche eine Menge Aufgaben eines Datenbank-Administrators kennen gelernt. Sie haben die Woche, mit einem vollständigen Rückblick auf das, was Sie bisher gelernt haben, beendet, indem Sie eine Webseite, die MySQL als Datenbank verwendet, von Grund auf erstellt haben. Sie verfügen nun über das Grundwissen, um MySQL-Datenbanken für richtige Produktionen zu erstellen.



SQL-Syntax und Befehle





Dieser Anhang stellt Ihnen eine Übersicht über SQL-Anweisungen zur Verfügung. Die Anweisungen sind in alphabetischer Reihenfolge aufgeführt. Einer Anweisung folgt eine Liste möglicher Parameter. Optionale Parameter sind in eckige Klammern [] eingeschlossen. Beispiele werden an entsprechender Stelle gegeben.

ALTER [IGNORE] TABLE *Tabellenname* Spezifikation[, Spezifikation]

Folgende Spezifikationen können Sie angeben

- ▶ `ADD [COLUMN] Spaltenname (Spaltendefinition) [FIRST oder AFTER Spaltenname]`
- ▶ `ADD INDEX [Indexname] (Spalten-Liste)`
- ▶ `ADD PRIMARY KEY (Spalten-Liste)`
- ▶ `ADD UNIQUE [Indexname] (Spalten-Liste)`
- ▶ `ALTER [COLUMN] Spaltenname {SET DEFAULT Defaultwert oder DROP DEFAULT}`
- ▶ `CHANGE [COLUMN] Alter_Spaltenname CREATE-Definition`
- ▶ `DROP [COLUMN] Spaltenname`
- ▶ `DROP PRIMARY KEY`
- ▶ `DROP INDEX Indexname`
- ▶ `MODIFY [COLUMN] CREATE-Definition`
- ▶ `RENAME [AS] Neuer_Tabellenname`

Eine Menge Informationen, aber alles halb so schlimm. Das `IGNORE`-Schlüsselwort veranlasst MySQL-Zeilen mit gleichen Werten in eindeutigen Schlüsseln zu löschen; andernfalls passiert nichts. Jede der vorhergehenden Spezifikationen kann in einer `ALTER TABLE`-Anweisung verwendet werden.

Beispiele:

```
ALTER TABLE Customers ADD COLUMN Account_Number INT
ALTER TABLE Customers ADD INDEX (Customer_ID)
ALTER TABLE Customers ADD PRIMARY KEY (Customer_ID)
ALTER TABLE Customers ADD UNIQUE (Customer_ID)
ALTER TABLE Customers CHANGE Customer_ID Customer_Number INT
ALTER TABLE Customers DROP Customer_ID
ALTER TABLE Customers DROP PRIMARY KEY
```

Im letzten Beispiel ist die Angabe eines Spaltennamens nicht erforderlich, da eine Tabelle nur einen Primärschlüssel haben kann.



```
ALTER TABLE Customers DROP INDEX Customer_ID
ALTER TABLE Customers MODIFY First_Name varchar(100)
ALTER TABLE Customers RENAME Customer
```

CREATE DATABASE database name

Diese einfache Anweisung erstellt eine Datenbank. Der Datenbankname muss gültig sein, sonst wird eine Fehlermeldung generiert.

Beispiel:

```
CREATE DATABASE Meet_A_Geek
```

CREATE [AGGREGATE] FUNCTION Funktionsname, RETURNS {STRING|REAL|INTEGER} SONAME shared_library_name

Diese Funktion veranlasst, dass eine von Ihnen erstellte Funktion in die `func`-Tabelle der `mysql`-Datenbank geladen wird. *Funktionsname* ist der Name der Funktion, mit der Sie diese innerhalb einer SQL-Anweisung aufrufen möchten. Sie müssen den Rückgabewert Ihrer Funktion hinter dem Schlüsselwort `RETURNS` angeben. Sie haben dabei die Wahl zwischen `STRING`, `REAL` oder `INTEGER`. `SONAME` verweist auf den Bibliotheksnamen der Funktion.

Wenn Sie das Schlüsselwort `AGGREGATE` verwenden, behandelt MySQL die Funktion als Aggregatsfunktion, also ähnlich wie `SUM()`, `AVG()` oder `MAX()`.

CREATE [UNIQUE] INDEX Indexname ON Tabellename (Spalten-Liste)

Diese Anweisung erstellt einen Index in (einer) übergebenen Spalte(n) einer spezifizierten Tabelle.

Beispiel:

```
CREATE INDEX idx_cust_ID ON Customers (Customer_ID)
```

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] Tabellename Tabellendefinition [Tabellenoptionen] [[IGNORE or REPLACE] Select-Anweisung]

Diese Anweisung erstellt eine Tabelle. Die *Tabellendefinition* besteht aus einer Liste von Spaltennamen und Typen.

`CREATE TABLE` hat folgende optionale Parameter:



- ▶ `CREATE TEMPORARY TABLE Temp_Customers (Cust_ID INT):` Erstellt eine temporäre Tabelle, die automatisch wieder gelöscht wird, wenn die Verbindung, mit der sie erstellt wird, beendet wird.
- ▶ `CREATE TABLE IF NOT EXISTS Customers (Cust_ID INT):` Erstellt nur dann die angegebene Tabelle, wenn diese noch nicht existiert.
- ▶ `TYPE = {ISAM, MYISAM oder HEAP}:` Bestimmt den Typ, von dem die Tabelle sein soll.

Die *Tabellenoptionen* können folgende Werte annehmen:

- ▶ `ISAM` ist der original Tabellentyp.
- ▶ `MYISAM` ist ein neuerer Speichertyp. Dieser Tabellentyp ist Standard.
- ▶ `HEAP`-Tabellen werden nur im Speicher abgelegt. Sie unterstehen folgenden Einschränkungen:
 - ▶ `HEAP`-Tabellen unterstützen keine `AUTO_INCREMENT`-Spalten.
 - ▶ Nur `=` oder `<=>` kann bei Indizes verwendet werden.
 - ▶ `HEAP`-Tabellen benutzen ein festes Format für die Länge von Datensätzen.
 - ▶ `HEAP`-Tabellen unterstützen keine `TEXT`- oder `BLOB`-Datentypen.
 - ▶ Sie können nichteindeutige Schlüssel in einer `HEAP`-Tabelle haben (nicht so gewöhnlich bei derartigen Tabellen).
 - ▶ **Beispiel:** `CREATE TABLE Customer(Cust_ID INT) TYPE=HEAP`
- ▶ `AUTO_INCREMENT = X:` Legt den Startpunkt einer `AUTO_INCREMENT`-Spalte fest. **Beispiel:** `CREATE TABLE Customers (Cust_ID INT AUTO_INCREMENT) AUTO_INCREMENT = 90000`
- ▶ `AVG_ROW_LENGTH = X:` Wird selten verwendet. Bestimmt die Länge von Tabellenzeilen bei Spalten variabler Länge.
- ▶ `CHECKSUM = {0 oder 1}:` Ermöglicht dem `myisamchk`-Programm, beschädigte Tabellen einfacher zu reparieren. Geht zu Lasten der Performance.
- ▶ `COMMENT = "Kommentar":` Ermöglicht Ihnen, einen Kommentar zu einer Tabelle hinzuzufügen. Dieser darf maximal 60 Zeichen lang sein.
- ▶ `MAX_ROWS = X:` Legt die maximal Anzahl an Zeilen (Datensätzen) in einer Tabelle fest. Kann zur Verbesserung der Performance beitragen.
- ▶ `MIN_ROWS = X:` Legt die minimale Anzahl an Zeilen (Datensätzen) in einer Tabelle fest. Kann zur Verbesserung der Performance beitragen.
- ▶ `PACK_KEYS = {0 oder 1}:` Wenn diese Option auf 1 gesetzt wird, werden Ihre Indizes kleiner und schneller. `UPDATE`-Anweisungen brauchen dann jedoch länger.



- ▶ `PASSWORD = "Passwort"`: Schützt die `.frm`-Datei mit einem Passwort.
- ▶ `DELAY_KEY_WRITE = {0 oder 1}`: Wenn Sie diese Option auf 1 setzen, werden alle `UPDATE`-Anweisungen erst ausgeführt, wenn die Tabelle nicht in Gebrauch ist.
- ▶ `ROW_FORMAT = {DEFAULT, DYNAMIC, STATIC oder COMPRESSED}`: Bestimmt, wie Datensätze in einer Tabelle gespeichert werden.

Beispiel einer `CREATE TABLE`-Anweisung:

```
CREATE TABLE Customers
(Customer_ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
First_Name VARCHAR(20) NOT NULL,
Last_Name VARCHAR(35) NOT NULL,
Address VARCHAR(100) NOT NULL);
```

DELETE [LOW PRIORITY] FROM Tabellennamen WHERE Bedingungen [LIMIT X]

Die `DELETE`-Anweisung löscht Werte aus einer spezifizierten Tabelle auf der Grundlage der Bedingungen der `WHERE`-Klausel. Die `DELETE`-Anweisung hat folgende optionale Parameter:

- ▶ `LOW_PRIORITY`: Verzögert das Löschen, bis die Tabelle nicht mehr in Gebrauch ist.
- ▶ `LIMIT X`: Beschränkt die Anzahl der zu löschenden Datensätze auf X.

Beispiel:

```
DELETE FROM Customers WHERE Customer_ID = 3
```

Um alle Datensätze einer Tabelle zu löschen, ohne dabei die Tabelle selbst zu entfernen, verwenden Sie die folgende Anweisung:

```
DELETE FROM Customers
```

DESCRIBE Tabellennamen [Spalten-Name]

Diese Anweisung gibt detaillierte Informationen zu den Spalten einer Tabelle aus.

Um eine Beschreibung aller Spalten einer Tabelle zu erhalten, verwenden Sie die folgende Anweisung:

```
DESCRIBE Customers
```

Um eine Beschreibung einer bestimmten Spalte zu erhalten, benutzen Sie:

```
DESCRIBE Customers Customer_ID
```



DROP DATABASE [IF EXISTS] Datenbankname

Die `DROP DATABASE`-Anweisung löscht eine Datenbank, ohne zuvor noch eine Warnung oder Bestätigungsfrage auszugeben. Seien Sie also vorsichtig, wenn Sie diese Anweisung verwenden.

Beispiel:

```
DROP DATABASE Meet_A_Geek
```

DROP FUNCTION Funktionsname

Löscht eine benutzerdefinierte Funktion aus der `func`-Tabelle der `mysql`-Datenbank. Vergleichen Sie hierzu die `CREATE`-Funktion.

DROP INDEX Indexname ON Tabellename

Diese Anweisung löscht einen angegebenen Index aus einer Tabelle.

Beispiel:

```
DROP INDEX idx_cust_id ON Customers
```

DROP TABLE [IF EXISTS] Tabellename [, Tabellename] ...

Diese Anweisung löscht eine spezifizierte Tabelle. Seien Sie vorsichtig mit allen `DELETE`- oder `DROP`-Anweisungen. Sie geben keine Warnungen aus.

Beispiel:

```
DROP TABLE Customers
```

EXPLAIN {SELECT-Anweisung oder Tabellename}

In Zusammenhang mit einer `SELECT`-Anweisung werden detaillierte Informationen darüber ausgegeben, wie MySQL die `SELECT`-Anweisung ausführen würde. Bei Angabe eines Tabellennamens bewirkt die Anweisung das Gleiche wie die `SHOW`-Anweisung. Vergleichen Sie diese Anweisung für mehr Informationen.

Beispiel:

```
EXPLAIN SELECT C.First_Name  
FROM Customers AS C, Orders as O  
WHERE C.Customer_ID = O.Customer_ID
```



FLUSH *Flush-Option*[, *Flush-Option*] ...

Diese Anweisung leert den Cache von MySQL. Mögliche Optionen sind:

- ▶ **HOSTS:** Leert die HOST-Cache-Tabelle. Benutzen Sie diese Option, wenn sich eine Ihrer HOST-IP-Adressen oder die lokale IP-Adresse ändert.
- ▶ **LOGS:** Schließt alle offenen LOG-Dateien.
- ▶ **PRIVILEGES:** Lädt die GRANT-Tabellen neu. Benutzen Sie diese Option, wenn Sie einen neuen Benutzer angelegt haben oder die Informationen in der `mysql`-Datenbank geändert haben.
- ▶ **TABLES:** Schließt alle offenen Tabellen.
- ▶ **STATUS:** Setzt die Systemstatus-Variable zurück auf 0.

Beispiel:

```
FLUSH PRIVILEGES
```

Sie können auch mehrere Optionen auf einmal angeben:

```
FLUSH PRIVILEGES, TABLES
```

GRANT *Privilegien* ON Datenbank-Objekt TO Benutzername IDENTIFIED BY Passwort [WITH GRANT OPTION]

Mit dieser Anweisung können Sie Benutzerrechte (Privilegien) vergeben. Dies ist die empfohlene Methode, um neue Benutzer anzulegen. Der Administrator kann eines, einige oder alle der folgenden Rechte vergeben:

- ▶ ALL PRIVILEGES
- ▶ ALTER
- ▶ CREATE
- ▶ DELETE
- ▶ DROP
- ▶ FILE
- ▶ INDEX
- ▶ INSERT
- ▶ PROCESS
- ▶ REFERENCES
- ▶ RELOAD



- ▶ SELECT
- ▶ SHUTDOWN
- ▶ UPDATE
- ▶ USAGE

Das *Datenbank-Objekt* kann eine Spalte, Tabelle oder Datenbank sein.

Das *Passwort* wird automatisch verschlüsselt. Verwenden Sie daher nicht die `password()`-Funktion. Vergessen Sie nicht, anschließend die `GRANT`-Tabellen mit der `FLUSH`-Anweisung neu zu laden.

Wenn Sie die `WITH GRANT OPTION`-Option verwenden, vergeben Sie an den spezifizierten Benutzer die Möglichkeit, selbst Berechtigungen an andere Benutzer zu vergeben.

Bei der Angabe von *Hosts* (Rechnern) können Sie Platzhalter (Wildcards) verwenden.

Die `IDENTIFIED BY`-Klausel ist optional, wird aber dringend empfohlen, um die Sicherheit Ihrer Datenbank zu gewährleisten.

Beispiel:

Um einen Benutzer mit alle Berechtigungen für alle Datenbanken einschließlich der Fähigkeit, anderen Benutzern Rechte zu vergeben, zu versorgen, verwenden Sie die folgende Anweisung:

```
GRANT ALL PRIVILEGES ON *.*  
TO mark@localhost  
IDENTIFIED BY "dinosaur"  
WITH GRANT OPTION
```

Um einem Benutzer `SELECT`- und `UPDATE`-Rechte für eine bestimmte Datenbank zu vergeben, benutzen Sie folgende Anweisung:

```
GRANT SELECT, UPDATE ON Meet_A_Geek.*  
TO sydney@localhost  
IDENTIFIED BY "toys"
```

Um einen Benutzer ohne jegliche Rechte zu erstellen, benutzen Sie diese Anweisung:

```
GRANT USAGE ON *.* TO user@localhost
```




INSERT [LOW_PRIORITY oder DELAYED] [IGNORE] [INTO] *Tabellenname* [(*Spalten-Liste*)] VALUES(*Werte-Liste*), INSERT [LOW_PRIORITY oder DELAYED] [IGNORE] [INTO] *Tabellenname* [(*Spalten-Liste*)] SELECT ..., INSERT [LOW_PRIORITY oder DELAYED] [IGNORE] [INTO] *Tabellenname* [(*Spalten-Liste*)] SET *Spaltenname* = *Ausdruck* [, *Spaltenname* = *Ausdruck*] ...

Diese Anweisungen fügen neue Datensätze (Zeilen) in eine bestehende Tabelle ein. Wenn keine *Spaltennamen* verwendet werden, müssen die Werte der *Werte-Liste* exakt in Typ, Position und Anzahl mit denen der Tabelle übereinstimmen. Werden *Spaltennamen* verwendet, müssen die Werte der *Werte-Liste* exakt in Typ, Position und Anzahl mit diesen übereinstimmen.

Wenn Sie die *SELECT*-Klausel benutzen, muss das Ergebnis des *SELECTs* in Typ, Position und Anzahl der Spalten in der Tabelle übereinstimmen. Wenn Sie eine *Spalten-Liste* angeben haben, müssen die Ergebnisse mit dieser übereinstimmen.

Die optionale *LOW_PRIORITY*-Klausel veranlasst, dass mit dem Einfügen der Daten so lange gewartet wird, bis die Tabelle nicht mehr in Gebrauch ist.

Die Option *DELAYED* bewirkt, dass das Einfügen der Daten verzögert wird, bis die Tabelle frei ist. Mehrere *INSERT*-Anweisungen werden dabei gebündelt und auf einmal in die Datenbank geschrieben. Das verbessert die Performance.

Beispiel:

```
INSERT INTO Customers
(Customer_ID, First_Name, Last_Name)
VALUES (NULL, 'Scott', 'Johnson')
```

KILL thread_ID

Diese Anweisung beendet den Prozess, der durch die *thread_ID* identifiziert wird. Dieses kann auch mit dem *mysqladmin*-Programm erfolgen. Die *thread_IDs* können mit der *SHOW PROCESSLIST*-Anweisung angezeigt werden.

Beispiel:

```
KILL 18
```

LOAD DATA [LOW_PRIORITY] [LOCAL] INFILE '*Dateiname*' [IGNORE oder REPLACE] INTO *Tabellenname*, [FIELDS TERMINATED BY *Zeichen*] [ENCLOSED BY *Zeichen*], [ESCAPED BY *Zeichen*] [LINES TERMINATED BY *Zeichen*],[IGNORE X LINES] [(*Spalten-Liste*)]

Diese Anweisung lädt eine Textdatei namens *Dateiname* in eine Tabelle.



Wenn das optionale Schlüsselwort `LOCAL` verwendet wird, sucht MySQL auf dem Client-Rechner nach der Datei. Wird es nicht angegeben, wird die Datei vom Server geladen.

Die Angabe des *Dateinamens* muss mit einem vollständigen Pfad erfolgen.

Die Tabelle muss die richtige Anzahl an Spalten mit den richtigen Datentypen enthalten. Ist das nicht der Fall, wird ein Fehler ausgegeben.

Beispiel:

Das folgende Beispiel lädt eine CSV-Datei (Komma-separierte-Datei) namens `accessdata.txt` in die `Customers`-Tabelle. Die Felder sind mit Anführungszeichen eingeschlossen.

```
LOAD DATA INFILE '/home/mark/accessdata.txt' INTO Customers
FIELDS TERMINATED BY "," ENCLOSED BY ""
```

LOCK TABLES Tabellename {READ oder [LOW_PRIORITY] WRITE} [,Tabellename {READ oder [LOW_PRIORITY] WRITE}] ...

Diese Anweisung sperrt eine Tabelle für den aktuellen Prozess. Wenn Sie Tabellen sperren, ist es wichtig, dass Sie alle Tabellen sperren, die Sie verwenden möchten.

Die Sperre auf eine Tabelle bleibt so lange bestehen, bis diese entweder mit der `UNLOCK`-Anweisung entsperrt, oder der sperrende Thread beendet wird. Vergleichen Sie auch `UNLOCK`.

Beispiel:

```
LOCK TABLES Customers READ, Orders WRITE
SELECT C.* FROM Customer AS C, Orders AS O WHERE O.Customer_ID = C.Customer_ID
UNLOCK TABLES
```

In diesem Beispiel musste sowohl die `Customers`- als auch die `Orders`-Tabelle gesperrt werden, da die `SELECT`-Anweisung beide Tabellen verwendet.

OPTIMIZE TABLE Tabellename

Diese Anweisung gibt den verlorenen Plattenplatz wieder frei, der durch gelöschte Datensätze entsteht.

Sie sollte verwendet werden, wenn eine große Anzahl an Datensätzen gelöscht oder Felder variabler Länge strukturell verändert wurden.

Die Tabelle kann, bis die Operation abgeschlossen ist, nur gelesen werden.



Beispiel:

```
OPTIMIZE TABLE Customers
```

REPLACE [LOW_PRIORITY oder DELAYED] [INTO] *Tabellenname* [(*Spaltenliste*)] VALUES(*Werte-Liste*), REPLACE [LOW_PRIORITY oder DELAYED] [INTO] *Tabellenname* [(*Spaltenliste*)] SELECT ..., REPLACE [LOW_PRIORITY oder DELAYED] [INTO] *Tabellenname* SET *Spaltenname* = *Ausdruck* [, *Spaltenname* = *Ausdruck*] ...

Diese Anweisung ist identisch mit der INSERT-Anweisung und verfügt über dieselben Optionen. Der einzige Unterschied besteht darin, dass diese Anweisung einen Datensatz zuerst löscht, bevor ein neuer Datensatz eingefügt wird, wenn der alte Datensatz den gleichen Wert in einem eindeutigen Index hat.

Beispiel:

```
REPLACE INTO Customers (Customer_ID, First_Name) VALUES(12, "Jason")
```

REVOKE *Privilegien-Liste* ON *Datenbank-Objekt* FROM *Benutzername*

Die REVOKE-Anweisung entzieht einem mit *Benutzernamen* spezifizierten Benutzer die in der *Privilegien-Liste* aufgezählten Berechtigungen für ein *Datenbank-Objekt* (Spalte, Tabelle oder Datenbank), die an diesen mit der GRANT-Anweisung vergeben wurden.

Damit die Änderungen wirksam werden, müssen die GRANT-Tabellen mit der FLUSH-Anweisung neu geladen werden.

Vergleichen Sie die GRANT-Anweisung für eine Liste der Privilegien.

Beispiel:

Das folgende Beispiel entzieht dem Benutzer `mark@localhost` alle Rechte. Der Benutzer wird aber nicht aus den MySQL-GRANT-Tabellen gelöscht.

```
REVOKE ALL PRIVILEGES ON *.* FROM mark@localhost
```



SELECT [DISTINCT] *Spalten-Liste* [INTO OUTFILE '*Dateiname*' *Export-Optionen*] FROM *Tabellen-Liste* [{CROSS oder INNER oder STRAIGHT oder LEFT oder NATURAL} JOIN *Tabellenname* [ON *Join-Bedingung*]] [WHERE *Kriterien*][GROUP BY *Spaltennamen*][HAVING *Kriterien*][ORDER BY *Spaltenname*][ASC oder DESC][LIMIT[*Offset*,] *Zeilenanzahl*][PROCEDURE *Prozedurname*]

Diese Anweisung gibt einen Ergebnissatz auf Grundlage der spezifizierten Kriterien zurück.

Das DISTINCT-Schlüsselwort gibt nur eindeutige Werte einer Spalte zurück.

Die Klauseln hinter der WHERE-Klausel stellen zusätzliche Filtermöglichkeiten zur Verfügung.

Beispiel:

```
SELECT * FROM Customers WHERE Last_Name  
LIKE "Paul%"  
ORDER BY Last_Name
```

SHOW Parameter

Die SHOW-Anweisung gibt einen Ergebnissatz auf der Grundlage der spezifizierten Parameter zurück. Mögliche Parameter sind:

COLUMNS FROM *Tabellenname* [FROM *Datenbankname*] oder COLUMNS FROM *Datenbankname*.*Tabellenname*: Stellt detaillierte Informationen basierend auf dem Tabellennamen zur Verfügung.

DATABASES: Zeigt eine Liste vorhandener Datenbanken an.

GRANTS FOR *Benutzername*: Zeigt die Rechte des spezifizierten Benutzers an.

INDEX FROM *Tabellenname*: Gibt Informationen über alle Indizes einer Tabelle zurück.

PROCESSLIST: Zeigt eine Liste aller aktuellen Prozesse und IDs an.

STATUS: Zeigt die Server-Status-Variablen an.

TABLES FROM *Datenbankname*: Zeigt eine Liste aller Tabellen einer Datenbank an.

TABLE STATUS FROM *Datenbankname*: Zeigt detaillierte Informationen über alle Aktionen an, die in der spezifizierten Tabelle vorgenommen wurden.

VARIABLES: Gibt eine Liste von Systemvariablen aus.

Beispiel:

```
SHOW COLUMNS FROM Meet_A_Geek.Customers  
SHOW PROCESSLIST
```



UNLOCK TABLES *Tabellenname* oder UNLOCK TABLE

Diese Anweisung gibt alle Sperren, die von einem Client gehalten werden, wieder frei.

UPDATE [LOW_PRIORITY] *Tabellenname* SET *Spaltenname* = Wert [WHERE *Kriterien*] [LIMIT X]

Die UPDATE-Anweisung ermöglicht dem Benutzer, Werte in einer Datenbank zu editieren.

Das optionale LOW_PRIORITY-Schlüsselwort bewirkt, dass Aktualisierungen nur ausgeführt werden, wenn die Tabelle nicht in Gebrauch ist.

Beispiel:

```
UPDATE Customers SET First_Name = "Geoff" WHERE Customer_ID = 12
```

USE *Datenbankname*

Die USE-Anweisung macht die spezifizierte Datenbank zur aktiven Datenbank. Auf andere Datenbanken kann immer noch zugegriffen werden, wenn Sie die Punktnotation verwenden (*Datenbankname.Tabellenname*).

Beispiel:

```
USE Meet_A_Geek
```




Aktuelle MySQL- Funktionen



Dieser Anhang enthält eine Liste aktueller MySQL-Funktionen, die innerhalb von SQL-SELECT-Anweisungen verwendet werden können. Die Funktionen sind in alphabetischer Reihenfolge aufgelistet, gefolgt von einer kurzen Beschreibung und einem Beispiel. Zwischen Funktionsname und Klammern darf kein Leerzeichen sein.

ABS(x)

Gibt den absoluten Wert von *x* zurück.

Beispiel:

```
SELECT ABS(-2)
```

Ergebnis:

2

ACOS(x)

Gibt den `arc`-Kosinus von *x* zurück. Gibt `NULL` zurück, wenn *x* nicht zwischen -1 und 1 liegt.

Beispiel:

```
SELECT ACOS(-0.653644)
```

Ergebnis:

2.2.83186

ADDDATE(Datum, INTERVALL Intervall-Ausdruck)

Gibt das Ergebnis aus dem Datum plus dem Intervall zurück. Gibt `NULL` zurück, wenn das übergebene Datum kein gültiges Datum ist. Diese Funktion ist identisch mit der `DATE_ADD()`-Funktion.

Mögliche Intervalle:

YEAR, DAY, HOUR, MINUTE, SECOND

Beispiele:

```
SELECT ADDDATE("1969-04-29"], INTERVAL 1 YEAR)
SELECT ADDDATE("1969-04-29", INTERVAL 1 DAY)
SELECT ADDDATE("1969-04-29", INTERVAL 40 DAY)
```

Ergebnisse:

1970-04-29
1969-04-30
1969-06-08

**ASIN(x)**

Gibt den `arc`-Sinus von `x` zurück. Gibt `NULL` zurück, wenn `x` nicht zwischen -1 und 1 liegt.

Beispiel:

```
SELECT ASIN(0.987)
```

Ergebnis:

```
1.409376
```

ASCII(x)

Gibt den ASCII-Code des ersten Zeichens einer Zeichenkette (`x`).

Beispiel:

```
SELECT ("A");
```

Ergebnis:

```
65
```

Beispiel:

```
SELECT ("Abc");
```

Ergebnis:

```
65
```

ATAN(x)

Gibt den `arc`-Tangens von `x` zurück.

Beispiel:

```
SELECT ATAN(3)
```

Ergebnis:

```
1.249046
```

ATAN2(x,y)

Gibt den `arc`-Tangens auf der Grundlage der übergebenen Werte zurück, um das Quadrat auszurechnen.

Beispiel:

```
SELECT ATAN2(2,2)  
SELECT ATAN2(-2,2)
```



Ergebnis:

.785398

-.785398

AVG(*Ausdruck*)

Diese Funktion gibt den Durchschnitt der Zahlen an, die in dem *Ausdruck* zur Verfügung gestellt werden. Nur Werte, die nicht NULL sind, werden berücksichtigt.

Beispiel:

```
SELECT AVG(Price) FROM Customers
```

Ergebnis:

23

(vorausgesetzt, Price ist eine Spalte in einer Tabelle)

BENCHMARK(*x*, *Ausdruck*)

Führt den übergebenen *Ausdruck* *x*-mal aus. Der Rückgabewert der Funktion ist immer 0. Die wichtige Information ist die vergangene Zeit, die am Ende angegeben wird. Diese Funktion ermöglicht Ihnen zu entscheiden, wie schnell Ihr Server Anfragen bearbeitet.

Beispiel:

```
SELECT BENCHMARK(20000, SELECT * FROM Customers)
```

Ergebnis:

Das Ergebnis hängt von bestimmten Faktoren wie der Server-Auslastung ab und variiert daher. Das Ergebnis ist immer 0 mit Angabe der verbrauchten Zeit.

BIN(*x*)

Gibt den binären Wert von *x* zurück, wo *x* ein BIGINT ist. Der Rückgabebetyp ist ein String.

Beispiel:

```
SELECT BIN(3)
```

Ergebnis:

11



CEILING(x)

Gibt den kleinsten Integer-Wert zurück, der nicht kleiner als *x* ist. Der Rückgabewert wird in einen `BIGINT` konvertiert. Vergleichen Sie die Funktion `FLOOR()`.

Beispiel:

```
SELECT CEILING(4.56)
```

Ergebnis:

5

Beispiel:

```
SELECT CEILING(-4.56)
```

Ergebnis:

-4

CHAR(x, y, z, ...)

Gibt einen String mit Werten zurück, die auf der ASCII-Codierung der übergebenen Werte (*x*, *y*, *z*, ...) basiert. `NULL`-Werte werden übergangen.

Beispiel:

```
SELECT CHAR(65, 78, 87, 100)
```

Ergebnis:

ANWd

CHARACTER_LENGTH, CHAR_LENGTH

Vergleichen Sie die Funktion `LENGTH()`.

COALESCE(Ausdruck1, Ausdruck2, ...)

Gibt den ersten Wert der übergebenen Liste zurück, der nicht `NULL` ist.

Beispiel:

```
SELECT COALESCE(NULL, 345, 56)
```

Ergebnis:

345



CONCAT(x, y, z, ...)

Gibt als Ergebnis eine Kombination aller Argumente in einem String zurück. Wenn ein NULL-Wert unter den Argumenten ist, wird NULL zurückgegeben.

Beispiel:

```
SELECT CONCAT("Paul", " ", "du Bois")
```

Ergebnis:

Paul du Bois

Beispiel:

```
SELECT CONCAT("SO", "NULL", "0987")
```

Ergebnis:

NULL

CONV(x, von-Basis, nach-Basis)

Konvertiert eine Zahl x von einer Basis zu einer anderen. Der Rückgabetyt ist ein String.

Beispiel:

```
SELECT CONV(3, 10, 2)
```

Ergebnis:

11

COS(x)

Gibt den Kosinus von x zurück, wobei x in RAD gegeben ist.

Beispiel:

```
SELECT COS(4)
```

Ergebnis:

-0.653644

COT(x)

Gibt den Co-Tangens von x zurück.

Beispiel:

```
SELECT COT(5)
```



Ergebnis:

-0.29581292

COUNT(x)

Gibt die Anzahl aller Werte eines Ergebnisses zurück, die nicht `NULL` sind. Wenn ein `*` benutzt wird, gibt die Funktion die Anzahl aller Zeilen (Datensätze) eines Ergebnisses zurück.

Beispiel:

```
SELECT COUNT(Smokers) as Smokers FROM Customers WHERE Smoker="Y"  
SELECT COUNT(*) FROM Customers
```

Ergebnisse:

129

300

CURDATE() und CURRENT_DATE()

Gibt das aktuelle Systemdatum zurück.

Beispiel:

```
SELECT CURDATE()
```

Ergebnis:

2000-10-19

CURTIME() und CURRENT_TIME()

Gibt die aktuelle Systemzeit zurück.

Beispiel:

```
SELECT CURTIME()
```

Ergebnis:

23:21:53

CURRENT_TIMESTAMP()

Gibt die aktuelle Zeit und das aktuelle Datum zurück. Vergleichen Sie hierzu auch `NOW()`, `SYSDATE()` und `CURTIME()`.



Beispiel:

```
SELECT CURRENT_TIMESTAMP()
```

Ergebnis:

2000-10-19 23:23:09

DATABASE()

Gibt den Namen der aktiven Datenbank zurück.

Beispiel:

```
SELECT DATABASE()
```

Ergebnis:

Meet_A_Geek

DATE_ADD(Datum, INTERVAL Wert Typ)

Addiert den *Wert* zu dem *Datum* hinzu. `INTERVAL` ist ein Schlüsselwort und muss in der Funktion verwendet werden. Der Typ kann einer der Typen sein, die in Tabelle B.1 aufgelistet sind.

Typ	Erwarteter Wert
SECOND	Sekunden
MINUTE	Minuten
HOURL	Stunden
DAY	Tage
MONTH	Monate
YEAR	Jahre
MINUTE_SECOND	"Minuten:Sekunden"
HOURL_SECOND	"Stunden:Minuten:Sekunden"
HOURL_MINUTE	"Stunden:Minuten"
DAY_SECOND	"Tage Stunden:Minuten:Sekunden"
DAY_MINUTE	"Tage Stunden:Minuten"
DAY_HOURL	"Tage Stunden"
YEAR_MONTH	"Jahre Monate"

Tabelle B.1: Zeit- und Datumstypen



Wenn Sie einen Wert aus der ersten Spalte verwenden, muss dieser mit dem in der zweiten Spalte angegebenen Format übereinstimmen.

Siehe auch `ADDDATE()` und `DATE_SUB()`.

Beispiel:

```
SELECT DATE_ADD("2000-01-27", INTERVAL 4 DAY)
```

Ergebnis:

2000-01-31

Beispiel:

```
SELECT DATE_ADD("2000-04-23 23:59:59", INTERVAL "1:1" MINUTE_SECOND)
```

Ergebnis:

2000-04-24 00:01:00

DATE_FORMAT(*Datum*, *Formatzeichen*)

Formatiert das übergebene *Datum* nach dem Format, das durch das *Formatzeichen* bestimmt wird. Es können mehr als nur ein *Formatzeichen* benutzt werden, um ein bestimmtes Format zu beschreiben. Tabelle B.2 listet die Formatzeichen auf.

Formatzeichen	Bedeutung
%M	Vollständiger Monatsname
%m	Nummerische Monatsanzeige
%b	Abgekürzter Monatsname
%W	Vollständiger Wochentagsname
%a	Abgekürzter Wochentagsname
%D	Tag des Monats
%d	Tag des Monats (numerisch)
%Y	Jahr 4-stellig
%y	Jahr 2-stellig
%j	Tag des Jahres – numerisch
%r	Zeit – 12-Stunden-Anzeige
%T	Zeit – 24-Stunden-Anzeige
%H	Stunden 00-23
%h	Stunden 01-12

Tabelle B.2: Formatzeichen



Formatzeichen	Bedeutung
%i	Minuten 00-59
%S	Sekunden

Tabelle B.2: Formatzeichen (Forts.)

Beispiel:

```
SELECT DATE_FORMAT("2000-02-27", "%M %D %Y")
```

Ergebnis:

February 27th 2000

DATE_SUB(Datum, INTERVAL Wert Typ) und SUBDATE(Datum, INTERVAL Wert Typ)

Subtrahiert den *Wert* von dem übergebenen *Datum*. *INTERVAL* ist ein Schlüsselwort und muss in der Funktion verwendet werden. Vergleichen Sie `ADDDATE()` für Typen und deren Werte.

Beispiel:

```
SELECT DATE_SUB("2000-01-01", INTERVAL 1 DAY)
```

Ergebnis:

1999-12-31

Beispiel:

```
SELECT DATE_SUB("2000-03-10 00:00:00", INTERVAL 1 MINUTE)
```

Ergebnis:

2000-03-09 23:59:00

DAYNAME(Datum)

Gibt den Tagesnamen des spezifizierten *Datums* aus.

Beispiel:

```
SELECT DAYNAME("2000-01-27")
```

Ergebnis:

Thursday

**DAYOFMONTH(*Datum*)**

Gibt den Tag des Monats eines spezifizierten *Datums* aus.

Beispiel:

```
SELECT DAYOFMONTH("2000-01-27")
```

Ergebnis:

27

DAYOFWEEK(*Datum*)

Gibt den Index des Wochentages eines spezifizierten *Datums* aus. Die folgende Liste enthält die korrekten Indizes für die jeweiligen Tage:

1 = Sonntag

2 = Montag

3 = Dienstag

4 = Mittwoch

5 = Donnerstag

6 = Freitag

7 = Samstag

Beispiel:

```
SELECT DAYOFWEEK("2000-01-27")
```

Ergebnis:

5

(korrespondiert mit dem Wochentag Donnerstag)

DAYOFYEAR(*Datum*)

Gibt den Tag des Jahres für ein spezifiziertes *Datum* aus (julianisches Datum).

Beispiel:

```
SELECT DAYOFYEAR("2000-01-27")
```

Ergebnis:

27



DECODE(*Binärer-String*, *Passwort*)

Gibt das entschlüsselte Ergebnis des *binären Strings* zurück. Das *Passwort* muss das gleiche sein wie bei der Verschlüsselung.

DEGREES(*x*)

Gibt den Wert von *x* in DEG an, wobei *x* in RAD vorliegen muss. Siehe auch `RADIANS()`.

Beispiel:

```
SELECT DEGREES(4.345)
```

Ergebnis:

```
248.950162
```

ELT(*X*, *a*, *b*, *c*, ...)

Gibt den String (*a*, *b*, *c*, ...) zurück, der in der Position mit *X* übereinstimmt. Siehe auch `FIELD()`.

Beispiel:

```
SELECT ELT(3, "Egon", "Hartmut", "Simone", "Uwe")
```

Ergebnis:

```
Simone
```

ENCODE(*Wort*, *Passwort*)

Gibt einen verschlüsselten binären String von *Wort* zurück. Die Verschlüsselung basiert auf dem *Passwort*. Siehe auch `DECODE()`.

Beispiel:

```
SELECT ENCODE("tacobell", "taco")
```

Ergebnis:

```
_pÖM
```

(Binärer, verschlüsselter String von »tacobell«).

**ENCRYPT(Wort [, Salt])**

Gibt einen verschlüsselten String zurück. Die Verschlüsselung erfolgt unter Verwendung der Unix-crypt()-Funktion. Der optionale *Salt* kann ein Zwei-Zeichen-String sein.

Beispiel:

```
SELECT ENCRYPT("tacobell","sa")
```

Ergebnis:

```
SaAUT2FASzS2s
```

EXP(x)

Gibt das Ergebnis von *x* zur Basis *e* zurück.

Beispiel:

```
SELECT EXP(5)
```

Ergebnis:

```
148.413159
```

EXTRACT(Wert FROM Datum)

Extrahiert einen formatierten Wert aus einem gegebenen Datum, wobei der Wert durch eine der folgenden Formate bestimmt wird:

```
SECOND
```

```
MINUTE
```

```
HOURL
```

```
DAY
```

```
MONTH
```

```
YEAR
```

```
MINUTE_SECOND
```

```
HOURL_SECOND
```

```
HOURL_MINUTE
```

```
DAY_SECOND
```

```
DAY_MINUTE
```



DAY_HOUR

YEAR_MONTH

Siehe auch `ADDDATE()` und `SUB_DATE()`.

Beispiel:

```
SELECT EXTRACT(DAY FROM "2000-01-27")
```

Ergebnis:

27

Beispiel:

```
SELECT EXTRACT(DAY_SECOND FROM "2000-01-27 12:01:45")
```

Ergebnis:

27120145

FIELD(x, a, b, c ...)

Gibt die Position in der Argumenten-Liste zurück, an der *x* und der String übereinstimmen. Siehe auch `ELT()`.

Beispiel:

```
SELECT FIELD("Rasmus", "Andi", "Zeev", "Thies", "Rasmus")
```

Ergebnis:

4

FIND_IN_SET(x, String-Liste)

Gibt die Position in der *String-Liste* zurück, an der *x* gefunden wurde. Wird *x* nicht gefunden, wird 0 zurückgegeben. Die *String-Liste* ist eine Liste von Strings, die mit einem Komma voneinander getrennt sind.

Beispiel:

```
SELECT FIND_IN_SET("Egon", "Egon, Rasmus, Zeev, Andi")
```

Ergebnis:

1

FLOOR(x)

Gibt den größten Integer-Wert, der nicht größer als *x* ist, zurück. Der zurückgegebene Wert wird in einen `BIGINT` konvertiert. Siehe auch `CEILING()`.



Beispiel:

```
SELECT FLOOR(4.56)
```

Ergebnis:

4

Beispiel:

```
SELECT FLOOR(-4.56)
```

Ergebnis:

-5

FORMAT(Num, Dec)

Gibt die eine auf *Dec*-Dezimalstellen gerundete Zahl *Num* zurück. Ist *Dec* gleich 0, wird keine Nachkommastelle angezeigt.

Beispiel:

```
SELECT FORMAT(12345.45, 1)
```

Ergebnis:

12,345.5

FROM_DAYS(Tage)

Wird der Funktion eine Tagesnummer, die aus einem Aufruf der Funktion `TO_DAYS()` beruht, übergeben, gibt diese ein entsprechendes Datum aus.

Beispiel:

```
SELECT FROM_DAYS(694734)
```

Ergebnis:

1902-02-13

FROM_UNIXTIME(Unix-Zeit [, Formatzeichen])

Gibt ein Datum auf der Grundlage der übergebenen Unix-Zeit zurück. Der optionale Parameter stellt Formatierungsmöglichkeiten zur Verfügung.

Beispiel:

```
SELECT FROM_UNIXTIME(951631200, "%M %D %Y")
```



Ergebnis:

February 27 2000

GREATEST(x, y, ...)

Gibt den größten Wert aus der übergebenen Werte-Liste aus. Verwendet die gleichen Regeln wie LEAST() für den Vergleich. Siehe auch LEAST().

Beispiel:

```
SELECT GREATEST(45, 222, 3, 99)
```

Ergebnis:

222

Beispiel:

```
SELECT GREATEST("A", "B", "C")
```

Ergebnis:

C

Beispiel:

```
SELECT GREATEST("Mark", "Sam", "Ken")
```

Ergebnis:

Sam

HEX(x)

Gibt den hexadezimalen Wert von x zurück, wobei x ein BIGINT ist. Der Rückgabebetyp ist ein String.

Beispiel:

```
SELECT HEX(15)
```

Ergebnis:

F

HOUR(Zeit)

Gibt die Stunde aus einer übergebenen Zeit aus.

Beispiel:

```
SELECT HOUR("11:45:01")
```



Ergebnis:

11

IF(*Ausdruck1*, *Ausdruck2*, *Ausdruck3*)

Ergibt *Ausdruck1* WAHR, wird *Ausdruck2* zurückgegeben. Ist *Ausdruck1* FALSCH, wird *Ausdruck3* zurückgegeben.

Beispiel:

```
SELECT IF(1, "This", "That")
SELECT IF(0, "This", "That")
```

Ergebnis:

This

That

IFNULL(*Ausdruck1*, *Ausdruck2*)

Ist *Ausdruck1* NULL, wird *Ausdruck2* zurückgegeben. Ist *Ausdruck1* nicht NULL, wird *Ausdruck1* zurückgegeben.

Beispiel:

```
SELECT IFNULL(Favorite_Movie, "Keiner") FROM Customers WHERE Customer_ID = 5567
```

Ergebnis:

Ist die Spalte *Favorite_Movie* NULL, wird das Wort »Keiner« zurückgegeben. Ist *Favorite_Movie* nicht NULL, wird der Wert der Spalte zurückgegeben. Dieses ist eine brauchbare Funktion in Fällen, in denen Sie etwas in Ihrem Ergebnis zurückerhalten müssen.

ISNULL(*Ausdruck*)

Gibt 1 zurück, wenn der Ausdruck NULL ist. Andernfalls gibt die Funktion 0 zurück.

Beispiel:

```
SELECT ISNULL(Favorite_Movie) FROM Customers WHERE Customer_ID=2322
```

Ergebnis:

1, wenn die Spalte *Favorite_Movie* NULL ist, andernfalls 0.



INSERT(*x, y, z, i*)

Gibt den String *x* zurück, wobei *j* die Zeichen, die mit *y* beginnen auf einer Länge von *z* ersetzt.

Beispiel:

```
SELECT INSERT("Database", 5, 4, "ware")
```

Ergebnis:

Dataware

INSTR(*x, y*)

Gibt den Wert der Position aus, an der String *y* in String *x* vorkommt. Siehe auch `LOCATE()` und `POSITION()`.

Beispiel:

```
SELECT INSTR("Mark", "M")
```

Ergebnis:

1

LAST_INSERT_ID()

Gibt die letzte Folgenummer eines in einer `AUTO_INCREMENT`-Spalte eingefügten Datensatzes aus. Diese Nummer wird abhängig von der Verbindung gespeichert. Wenn also zwei Datensätze von zwei verschiedenen Verbindungen zur gleichen Zeit eingefügt werden, gibt die `LAST_INSERT_ID()`-Funktion für beide Verbindungen unterschiedliche Werte zurück, nämlich für jede Verbindung den jeweiligen Wert.

Beispiel:

```
SELECT LAST_INSERT_ID()
```

Ergebnis:

15

LCASE(*x*)

Gibt den String *x* zurück und konvertiert alle Buchstaben in Kleinbuchstaben.

Beispiel:

```
SELECT LCASE("Simone")
```




Ergebnis:

simone

LEAST(x, y, z, ...)

Gibt den kleinsten Wert aus der übergebenen Werte-Liste zurück. Folgende Regeln gelten bei dem Vergleich:

- ▶ Wenn alle Werte Integer-Werte (Ganzzahlen) sind oder der Vergleich in einem solchem Kontext stattfindet, werden sie als Integer verglichen.
- ▶ Wenn alle Werte Fließkommazahlen sind oder der Vergleich in einem solchem Kontext stattfindet, werden sie als Fließkommazahlen verglichen.
- ▶ Andernfalls werden die übergebenen Argumente als Strings verglichen. Dieser Vergleich beachtet so lange keine Unterschiede von Groß- und Kleinschreibung, bis einer der Werte ein Binärwert ist.

Beispiel:

```
SELECT LEAST(9, 45, 12, 34, 6)
```

Ergebnis:

6

Beispiel:

```
SELECT LEAST("A", "B", "C")
```

Ergebnis:

A

Beispiel:

```
SELECT LEAST("Mark", "Sam", "Ken")
```

Ergebnis:

Ken

LEFT(x, y)

Gibt die *y* Zeichen von *x* zurück. Beginnt auf der linken Seite, bis es den Wert von *y* erreicht hat. Siehe auch `RIGHT()`.

Beispiel:

```
SELECT LEFT("Datenbank", 5)
```



Ergebnis:

Daten

LENGTH(x)

Gibt die Länge von x zurück.

Beispiel:

```
SELECT LENGTH("Der schnelle Fuchs")
```

Ergebnis:

18

LOAD_FILE(Dateiname)

Öffnet die Datei und gibt den Inhalt als String zurück. Die Datei muss auf dem Server liegen und der Anwender dieser Funktion FILE-Rechte haben.

Beispiel:

```
UPDATE Products SET Picture = LOAD_FILE("/home/pictures/flowers.gif") WHERE  
Product_ID = 12
```

Ergebnis:

Die Anweisung lädt das Bild `flowers.gif` in die `Picture`-Spalte der `Products`-Tabelle. Die Spalte muss entweder vom Typ `BLOB` oder `TEXT` sein.

LOCATE(x, y, z)

Gibt die Position des Strings x in String y ausgehend von z an. Gibt 0 zurück, wenn keine Übereinstimmung gefunden wurde.

Beispiel:

```
SELECT LOCATE("M", "Mark", 2)
```

Ergebnis:

0

Beispiel:

```
SELECT LOCATE("M", "Mark", 1)
```

Ergebnis:

1

**LOG(x)**

Gibt den Logarithmus naturalis von x zurück.

Beispiel:

```
SELECT LOG(5)
```

Ergebnis:

```
1.609438
```

Beispiel:

```
SELECT LOG(-5)
```

Ergebnis:

```
NULL
```

LOG10(x)

Gibt den Logarithmus von x zurück.

Beispiel:

```
SELECT LOG10(1000)
```

Ergebnis:

```
3
```

Beispiel:

```
SELECT LOG10(-1000)
```

Ergebnis:

```
NULL
```

LOWER(x)

Gibt den String x zurück und konvertiert diesen in Kleinbuchstaben. Siehe auch `LCASE()`, `UCASE()` und `UPPER()`.

Beispiel:

```
SELECT LOWER("Mark")
```

Ergebnis:

```
Mark
```



LPAD(x, y, z)

Gibt den String *x* zurück und füllt diesen mit String *z* auf, bis String *x* die Länge von *y* hat. Siehe auch `RPAD()`.

Beispiel:

```
SELECT LPAD("Mark", 8, "OK")
```

Ergebnis:

```
OKOKMark
```

LTRIM(x)

Gibt *x* ohne führende Leerstellen zurück. Siehe auch `RTRIM()` und `TRIM()`.

Beispiel:

```
SELECT LTRIM("   Mark")
```

Ergebnis:

```
Mark
```

(ohne führende Leerstellen)

MAX(Ausdruck)

Gibt den größten Wert des übergebenen *Ausdrucks* zurück.

Beispiel:

```
SELECT MAX(Customer_ID) FROM Customers
```

Ergebnis:

```
387
```

MID(x, y, z)

Gibt den *z*-stelligen String *x* beginnend bei Position *y* zurück.

Beispiel:

```
SELECT MID("Datenbank", 5, 4)
```

Ergebnis:

```
bank
```

**MIN(*Ausdruck*)**

Gibt den kleinsten Wert des übergebenen *Ausdrucks* zurück.

Beispiel:

```
SELECT MIN(Customer_ID) FROM Customers
```

Ergebnis:

1

MINUTE(*Zeit*)

Extrahiert die Minuten aus einer übergebenen *Zeit*.

Beispiel:

```
SELECT MINUTE("11:45:01")
```

Ergebnis:

45

MOD(*x*, *y*)

Gibt den Rest aus der Division von *x* geteilt durch *y* zurück (Modulo). Sie können auch das %-Symbol (Modulo-Operator) verwenden.

Beispiel:

```
SELECT MOD(13,2)
```

Ergebnis:

1

Beispiel:

```
SELECT 19 % 7
```

Ergebnis:

5

MONTH(*Datum*)

Gibt den Monatsindex für ein bestimmtes *Datum* zurück.

Beispiel:

```
SELECT MONTH("2000-01-27")
```



Ergebnis:

1

MONTHNAME(*Datum*)

Gibt den Monatsnamen für ein bestimmtes *Datum* zurück.

Beispiel:

```
SELECT MONTHNAME("2000-05-10")
```

Ergebnis:

May

NOW()

Gibt die aktuelle Zeit und das aktuelle Datum zurück. Siehe auch `CURTIME()`, `CURRENT_TIMESTAMP()` und `SYSDATE()`.

Beispiel:

```
SELECT NOW()
```

Ergebnis:

2000-10-20 14:58:37

OCT(x)

Gibt den oktalen Wert von *x* zurück, wobei *x* ein `BIGINT` ist. Der Rückgabetyyp ist ein String.

Beispiel:

```
SELECT OCT(10)
```

Ergebnis:

12

OCTET_LENGTH()

Siehe `LENGTH()`.

PASSWORD(*Passwort*)

Verschlüsselt das übergebene *Passwort*.



Beispiel:

```
SELECT PASSWORD("tacobe11")
```

Ergebnis:

```
35decd2c4ab0f0c1
```

PERIOD_ADD(x, y)

Gibt das Ergebnis der Addition von x plus Periode y zurück.

Beispiel:

```
SELECT PERIOD_ADD(9910, 4)
```

Ergebnis:

```
200002
```

PERIOD_DIFF(x, y)

Gibt die Anzahl der Monate zwischen x und y zurück. x und y sollten dabei in der Form **YYMM** oder **CCYYMM** vorliegen.

Beispiel:

```
SELECT PERIOD_DIFF(200010, 199804)
```

Ergebnis:

```
30
```

PI()

Gibt den Wert von π zurück.

Beispiel:

```
SELECT PI()
```

Ergebnis:

```
3.141593
```

POSITION(x, y)

Gibt die Position des ersten Vorkommens von x in y zurück. Gibt 0 zurück, wenn x nicht in y gefunden wurde.



Beispiel:

```
SELECT LOCATE("uch", "Der schnelle Fuchs")
```

Ergebnis:

15

Beispiel:

```
SELECT POSITION("uch", "Der schnelle Fuchs")
```

Ergebnis:

15

POW(x, y) und POWER(x, y)

Gibt das Ergebnis von x potenziert mit y zurück.

Beispiel:

```
SELECT POWER(2,3)
```

Ergebnis:

8

Beispiel:

```
SELECT POW(9,2)
```

Ergebnis:

81

QUARTER(Datum)

Gibt das Jahresquartal eines übergebenen *Datums* zurück.

Beispiel:

```
SELECT QUARTER("2000-06-22")
```

Ergebnis:

2

RADIANS(x)

Wandelt x von DEG in RAD um.



Beispiel:

```
SELECT RADIANS(248)
```

Ergebnis:

4.328417

RAND() und RAND(Anlage)

Gibt eine Zufalls-Fließkommazahl im Bereich von 0 bis 1 zurück. Die Anlage wird zur Generierung der Zufallszahl verwendet, wenn sie angegeben wird. Die gleiche Anlage gibt die gleiche Serie von Zufallszahlen zurück. Dadurch ist die Ausgabe nicht mehr zufällig, sondern vorhersehbar.

Beispiel:

```
SELECT RAND()
```

Ergebnis:

0.16592424657748

Beispiel:

```
SELECT RAND()
```

Ergebnis:

0.60488347113587

Beispiel:

```
SELECT RAND(3)
```

Ergebnis:

0.18109050875631

Beispiel:

```
SELECT RAND(3)
```

Ergebnis:

0.18109050875631

REPEAT(x, y)

Gibt den String x y -mal aus. Wenn y kleiner als 0 ist, wird ein leerer String zurückgegeben.



Beispiel:

```
SELECT REPEAT("Simone", 4)
```

Ergebnis:

```
SimoneSimoneSimoneSimone
```

REPLACE(x, y, z)

Gibt den String *x* zurück und ersetzt alle Vorkommnisse von *y* durch *z*.

Beispiel:

```
SELECT REPLACE("MySQL ist toll.", "toll", "genial")
```

Ergebnis:

```
MySQL ist genial
```

REVERSE(x)

Gibt den String *x* in umgekehrter Reihenfolge zurück.

Beispiel:

```
SELECT REVERSE("MySQL")
```

Ergebnis:

```
LQSyM
```

RIGHT(String, x)

Gibt die am weitesten rechts stehenden Zeichen eines *Strings* mit *x* Länge zurück. Wenn der *String* NULL ist, wird NULL zurückgegeben.

Beispiel:

```
SELECT ("MySQL", 2)
```

Ergebnis:

```
QL
```

ROUND(x) und ROUND(x, y)

Gibt den Wert *x*, gerundet auf den nächstliegenden Integer-Wert, zurück. Wird der Funktion ein zweiter Wert, *y*, übergeben, bestimmt dieser die Anzahl der Nachkommastellen, auf die gerundet wird.



Beispiel:

```
SELECT ROUND(5.374)
```

Ergebnis:

5

Beispiel:

```
SELECT ROUND(-5.374)
```

Ergebnis:

-5

Beispiel:

```
SELECT ROUND(4.345, 1)
```

Ergebnis:

4.3

RPAD(x, y, z)

Gibt den String *x* zurück und füllt ihn mit *z* auf, bis er mit der Länge von *y* übereinstimmt. Siehe auch `LPAD()`.

Beispiel:

```
SELECT RPAD("Mark", 8, "OK")
```

Ergebnis:

MarkOKOK

RTRIM(x)

Gibt *x* zurück und schneidet alle folgenden Leerzeichen ab.

Beispiel:

```
SELECT RTRIM("Mark   ");
```

Ergebnis:

Mark

(ohne Leerzeichen nach Mark)



SECOND(Zeit)

Extrahiert die Sekunden aus einer übergebenen Zeit.

Beispiel:

```
SELECT SECOND("11:45:01")
```

Ergebnis:

1

SEC_TO_TIME(Sekunden)

Gibt die Zeit im hh:mm:ss-Format auf Grundlage der übergebenen *Sekunden* zurück.

Beispiel:

```
SELECT SEC_TO_TIME(56789)
```

Ergebnis:

15:46:29

SESSION_USER()

Gibt den Benutzer der aktuellen Verbindung zurück.

Beispiel:

```
SELECT SESSION_USER()
```

Ergebnis:

mark@localhost

SIGN(x)

Gibt das Vorzeichen von x zurück. Wenn x negativ ist, wird -1 zurückgegeben. Ist x gleich 0, wird 0 zurückgegeben. Ist x positiv, wird 1 zurückgegeben.

Beispiel:

```
SELECT SIGN(-14)
```

Ergebnis:

-1

Beispiel:

```
SELECT SIGN(0)
```



Ergebnis:

0

Beispiel:

```
SELECT SIGN(14)
```

Ergebnis:

1

SIN(x)

Gibt den Sinus von x zurück, wobei x in RAD angegeben ist.

Beispiel:

```
SELECT SIN(7)
```

Ergebnis:

0.656987

SOUNDEX(x)

Gibt einen `SOUNDEX`-String von x zurück.

Beispiel:

```
SELECT SOUNDEX("George")
```

Ergebnis:

G620

SPACE(x)

Gibt einen String mit x Leerzeichen zurück.

Beispiel:

```
SELECT SPACE(12)
```

Ergebnis:

» »

SQRT(x)

Gibt die Quadratwurzel von x zurück.



Beispiel:

```
SELECT SQRT(9)
```

Ergebnis:

3

Beispiel:

```
SELECT SQRT(-16)
```

Ergebnis:

NULL

STRCMP(*String1*, *String2*)

Gibt 1 zurück, wenn *String1* identisch mit *String2* ist. Sind sie verschieden, gibt die Funktion 0 zurück. Ist einer der beiden *Strings* NULL, gibt die Funktion NULL zurück.

Beispiel:

```
SELECT STRCMP("Simone", "Simone")
```

Ergebnis:

1

Beispiel:

```
SELECT STRCMP("Tamara", "Simone")
```

Ergebnis:

0

Beispiel:

```
SELECT STRCMP("Simone", "simone")
```

Ergebnis:

0

STD(*Ausdruck*) oder STDDEV(*Ausdruck*)

Gibt die Standardabweichung für einen gegebenen Ausdruck zurück. Nur Werte, die nicht NULL sind, werden berücksichtigt, um diesen Wert zu errechnen.

Beispiel:

```
SELECT STD(Quantity) FROM Orders
```



Ergebnis:

2.435

SUM(Ausdruck)

Gibt die Summe für einen gegebenen Ausdruck zurück. Nur Werte, die nicht NULL sind, werden berücksichtigt, um diesen Wert zu errechnen.

Beispiel:

```
SELECT SUM(Cost) FROM Orders
```

Ergebnis:

10234.34

SUBSTRING_INDEX(x, y, z)

Gibt einen Teilstring von *x* zurück. Wenn *z* positiv ist, sucht die Funktion das *z*-te Vorkommen von *y*, von links ausgehend, und gibt dann alles, was zur linken Seite steht, zurück. Wenn *z* negativ ist, sucht die Funktion das *z*-te Vorkommen von *y*, von rechts ausgehend, und gibt dann alles, was zur rechten Seite steht, zurück. Wird *y* gar nicht im String gefunden, wird der komplette String zurückgegeben. Wenn eines der Argumente NULL ist, wird NULL zurückgegeben.

Beispiel:

```
SELECT SUBSTRING_INDEX("mysql.3-23-26.Linux.tar.gz", ".", 3)
```

Ergebnis:

mysql.3-23-26.Linux

Beispiel:

```
SELECT SUBSTRING_INDEX("mysql.3-23-26.Linux.tar.gz", ".", -3)
```

Ergebnis:

Linux.tar.gz

Beispiel:

```
SELECT SUBSTRING_INDEX("mysql.3-23-26.Linux.tar.gz", "", 3)
```

Ergebnis:

mysql.3-23-26.Linux.tar.gz



SUBDATE()

Siehe `DATE_SUB()`.

SYSDATE()

Gibt die aktuelle Zeit und das aktuelle Datum zurück. Siehe auch `CURTIME()`, `NOW()` und `CURRENT_TIMESTAMP()`.

Beispiel:

```
SELECT SYSDATE()
```

Ergebnis:

```
2000-10-20 16:14:04
```

SYSTEM_USER()

Gibt den Benutzer der aktuellen Verbindung aus.

Beispiel:

```
SELECT SYSTEM_USER()
```

Ergebnis:

```
mark@localhost
```

TAN(x)

Gibt den Tangens von x zurück, wobei x in RAD liegt.

Beispiel:

```
SELECT TAN(12)
```

Ergebnis:

```
-0.635860
```

TIME_FORMAT(Zeit, Formatzeichen)

Formatiert eine übergebene *Zeit* auf der Grundlage der übergebenen *Formatzeichen*. Als Formatzeichen können alle zeitbezogenen Zeichen aus Tabelle B.2 verwendet werden. Siehe auch `DATE_FORMAT()`.

Beispiel:

```
SELECT TIME_FORMAT("2000-01-23 00:34:33", "%H %i")
```




Ergebnis:

00 34

TIME_TO_SEC(*Zeit*)

Gibt die Anzahl der Sekunden auf der Grundlage einer übergebenen *Zeit* zurück.

Beispiel:

```
SELECT TIME_TO_SEC("15:26:29")
```

Ergebnis:

55589

TO_DAYS(*Datum*)

Gibt die Anzahl der Tage vom Jahr 0 ausgehend bis zum übergebenen *Datum* zurück.
Siehe auch `FROM_DAYS()`.

Beispiel:

```
SELECT TO_DAYS("1946-08-17")
```

Ergebnis:

710990

TRIM([[*BOTH* oder *LEADING* oder *TRAILING*] [*x*] FROM *y*])

Gibt den String *y* zurück, wobei der spezifizierte String *x* entweder vom Anfang, vom Ende oder sowohl vom Anfang als auch Ende entfernt wird. Wenn keine Optionen angegeben werden, werden Leerzeichen vom Anfang und Ende des Strings *y* entfernt.
Siehe auch `LTRIM()` und `RTRIM()`.

Beispiel:

```
SELECT TRIM(" Simone ")
```

Ergebnis:

Simone

(ohne voran- bzw. nachgestellte Leerzeichen)

Beispiel:

```
SELECT TRIM(LEADING "S" FROM "Simone")
```



Ergebnis:

imone

Beispiel:

```
SELECT TRIM(BOTH "X" FROM "XXXFILE.XXX")
```

Ergebnis:

FILE.

TRUNCATE(x, y)

Beschneidet *x* auf *y* Nachkommastellen. Wenn *y* gleich 0 ist, werden keine Nachkommastellen angezeigt.

Beispiel:

```
SELECT TRUNCATE(3.4567, 2)
```

Ergebnis:

3.45

Beispiel:

```
SELECT TRUNCATE(3.4567, 0)
```

Ergebnis:

3

UCASE(x)

Gibt den String *x* zurück und konvertiert alle Buchstaben in Großbuchstaben. Siehe auch `UPPER()`, `LOWER()` und `LCASE()`.

Beispiel:

```
SELECT UCASE("sam i am")
```

Ergebnis:

SAM I AM

UNIX_TIMESTAMP([Datum])

Ohne Angabe eines optionalen *Datums*, wird der Unix-Zeitstempel zurückgegeben. Wenn ein *Datum* übergeben wird, wird es als Unix-Zeitstempel ausgegeben.



Beispiel:

```
SELECT UNIX_TIMESTAMP()
```

Ergebnis:

```
972052901
```

Beispiel:

```
SELECT UNIX_TIMESTAMP("2000-10-20")
```

Ergebnis:

```
971992800
```

UPPER(x)

Gibt den String *x* zurück und konvertiert alle Buchstaben in Großbuchstaben. Siehe auch `UCASE()`, `LOWER()` und `LCASE()`.

Beispiel:

```
SELECT UPPER("sam i am")
```

Ergebnis:

```
SAM I AM
```

USER()

Gibt den Namen des Benutzers der aktuellen Verbindung aus.

Beispiel:

```
SELECT USER()
```

Ergebnis:

```
mark@localhost
```

VERSION()

Gibt die Version des MySQL-Servers als String aus.

Beispiel:

```
SELECT VERSION()
```

Ergebnis:

```
3.22.23b
```



WEEK(Datum [, Start])

Gibt die Wochennummer für ein bestimmtes Datum aus. Wird der optionale Parameter *Start* angegeben, können Sie mit diesem bestimmen, an welchem Tag die Woche beginnt. Verwenden Sie 0, beginnt die Woche mit Sonntag. Verwenden Sie 1, beginnt sie am Montag.

Beispiel:

```
SELECT WEEK("2000-01-02")
```

Ergebnis:

1

Beispiel:

```
SELECT WEEK("2000-01-02", 1)
```

Ergebnis:

0

WEEKDAY(Datum)

Gibt den Wochentagsindex für ein bestimmtes Datum aus. Die folgende Liste enthält eine Übersicht über alle Indizes mit den dazugehörigen Wochentagen:

0 = Montag

1 = Dienstag

2 = Mittwoch

3 = Donnerstag

4 = Freitag

5 = Samstag

6 = Sonntag

Beispiel:

```
SELECT WEEKDAY("2000-01-27")
```

Ergebnis:

3



YEAR(*Datum*)

Extrahiert das Jahr aus einem übergebenen *Datum*.

Beispiel:

```
SELECT YEAR("00-01-27")
```

Ergebnis:

2000



Lösungen zu den Übungen



Tag 1

Übung 1

Vergleichen Sie die Preise einiger anderer Datenbanken mit der gleichen Ausstattung wie MySQL – das sind zum Beispiel SQL Server, Oracle, Adaptive Server und DB2. Überprüfen Sie nun, wieviel Sie mit MySQL sparen können.

Sie werden sehen, dass MySQL preislich weit unter den Mitbewerbern liegt. Zum jetzigen Zeitpunkt kostet Microsofts SQL-Server \$28.999,00 US-\$ für die Enterprise-Edition mit unlimitierter Anzahl von Clients. Die Enterprise-Edition von Sybases Adaptive Server liegt für eine 10-Benutzer-Lizenz bei \$1.650,00 US-\$ für die Windows NT-Plattform. Die Linux-Version kostet immerhin noch \$995 US-\$. MySQL ist kostenlos, und das sowohl für Linux als auch für Windows.

Übung 2

Besuchen Sie Websites und testen Sie Produkte, die MySQL benutzen. (Einige befinden sich auf der beigefügten CD-ROM.) MySQL in Aktion zu sehen kann Ihre Meinung über Open-Source-Produkte wirklich verändern.

Support Wizard ist ein gutes Beispiel. Das Programm benutzt MySQL sehr effizient.

Tag 2

Übung 1

Lassen Sie alle in der Datenbank befindlichen Daten erscheinen. Benutzen Sie dazu die Kommandozeile.

Sie können dieses mit folgender Anweisung erreichen:

```
SELECT * FROM mysql;
```

Sie können ebenfalls die SHOW-Anweisung verwenden. Diese zeigt alle Tabellen in einer Datenbank an:

```
SHOW TABLES FROM database_name
```




Übung 2

Überprüfen Sie, ob der `mysqld`-Daemon läuft. Wenden Sie zwei verschiedene Methoden an.

Eine Möglichkeit besteht darin, das `mysqladmin`-Programm zu verwenden:

```
bin/mysqladmin -p ping
```

Der andere Weg ist, die aktuellen Prozesse anzeigen zu lassen. Unter Linux bzw. Unix-Systemen verwenden Sie hierfür den Befehl `grep`. Unter Windows drücken Sie die Tastenkombination `Ctrl` + `Alt` + `Delete`.

```
ps -aux | grep mysqld
```

Übung 3

Verbinden Sie sich mit einem entfernten Rechner und fahren Sie den MySQL-Server hoch und wieder runter.

Starten Sie ein Telnet-Sitzung und verbinden Sie sich mit dem Rechner, auf dem der MySQL-Server läuft. Nach erfolgreichem Verbindungsaufbau wechseln Sie in das `mysql`-Installationsverzeichnis (`/usr/local/mysql` im Normalfall) und führen den `mysql-shutdown`-Befehl aus. Um MySQL wieder zu starten, geben Sie `safe_mysqld&` auf der Kommandozeilenebene ein.

Übung 4

Benutzen Sie den MySQL-Monitor auf einem entferntem Rechner.

Diese Aufgabe kann ebenfalls in einer Telnet-Sitzung ausgeführt werden. Starten Sie dabei in der Kommandozeile den MySQL-Monitor, indem Sie sich in das `/bin`-Verzeichnis von MySQL begeben und `mysql` eingeben.

Tag 3

Übung 1

Im `Meet_A_Geek`-Projekt haben Sie mehrere Geschäftsobjekte definiert. Können Sie weitere Objekte bestimmen?

Aufgrund von Gesprächen mit dem Kunden können Sie weitere Geschäftsobjekte definieren. In Frage kämen z.B. ein Lieferanten-Objekt, ein Werbungs-Objekt, ein Kupon-Objekt oder sogar ein Kreditkarten-Check-Objekt.



Übung 2

Erstellen der noch fehlenden Geschäftsregeln für das Meet_A_Geek-Projekt.

Zusätzliche Geschäftsregeln basieren wiederum auf Gesprächen mit dem Kunden bzw. einem tieferem Studium des Geschäftsablaufs. Eine Regel könnte sein, dass eine Transaktion erst ausgeführt werden kann, wenn eine Bezahlungsart ausgewählt wurde. Eine andere Regel wäre, dass ein Produkt nicht versendet werden kann, wenn es nicht im Lager vorhanden ist.

Tag 4

Übung 1

Erstellen und löschen Sie Datenbanken, indem Sie das `mysqladmin`-Programm sowie den Monitor benutzen.

Um eine Datenbank zu erstellen, benutzen Sie die folgende Anweisung:

```
bin/mysqladmin -p CREATE DATABASE TestDB
```

Um eine Datenbank zu löschen, benutzen Sie die folgende Anweisung:

```
bin/mysqladmin -p DROP DATABASE TestDB
```

Um die gleichen Ergebnisse mit dem MySQL-Monitor zu erzielen, verwenden Sie folgende Anweisungen:

```
CREATE DATABASE TestDB;
```

```
DROP DATABASE TestDB;
```

Übung 2

Fügen Sie der Datenbank einige Benutzer hinzu und versuchen Sie, diese Konten zu benutzen.

```
GRANT ALL ON *.* TO mark@% IDENTIFIED BY "thor2000";
```



Tag 5

Übung 1

Erläutern Sie die Vorteile der Normalisierung.

Normalisierung hilft dabei, eine flexible und effiziente Datenbank zu erstellen, die dazu beiträgt, auf einfache Weise abfragbar und manipulierbar zu sein.

Übung 2

Finden Sie Bereiche heraus, die nicht unbedingt normalisiert werden müssen.

Positionen wie etwa zwei Spalten, welche die Adresse und den ZIP-Code beinhalten, müssen unter Umständen nicht normalisiert werden. Auch bei einer Datenbank, die nur eine kleine Menge von Daten beinhaltet, kann eine Tabelle viel effizienter sein.

Tag 6

Übung 1

Erstellen Sie das Schema für das Meet_A_Geek-Projekt. Bauen Sie das Schema auf dem Entwurf, der am dritten Tag entwickelt wurde, auf.

Die folgenden Anweisungen könnten in einer Textdatei gespeichert und dann dem mysql-Befehl übergeben werden:

```
CREATE DATABASE Meet_A_Geek;
USE Meet_A_Geek;
CREATE TABLE Customers
(Customer_ID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    First_Name VARCHAR(15) NOT NULL,
    Last_Name VARCHAR(30) NOT NULL,
    Address VARCHAR(100) NOT NULL,
    City VARCHAR(30) NOT NULL,
    State VARCHAR(20) NOT NULL,
    Zip VARCHAR(15) NOT NULL,
    Email VARCHAR(20) NOT NULL,
    Age TINYINT NOT NULL,
    Gender CHAR(1) NOT NULL,
    RACE VARCHAR(15) NOT NULL,
    Eye_Color VARCHAR(20),
```



```
Hair_Color VARCHAR(20),  
Activity VARCHAR(100),  
Movie VARCHAR(50),  
Occupation VARCHAR(50),  
Smoker ENUM('Y', 'N') DEFAULT "N");
```

```
CREATE TABLE Shippers  
(Shipper_ID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,  
    Name VARCHAR(100) NOT NULL,  
    Location VARCHAR(100) NOT NULL,  
    Shipping_Method VARCHAR(40) NOT NULL,  
    Active CHAR(1) NOT NULL);
```

```
CREATE TABLE Orders  
(Order_ID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,  
    Shipper_ID INT NOT NULL,  
    Transaction_ID INT NOT NULL,  
    Customer_ID INT NOT NULL,  
    Order_Date DATETIME NOT NULL,  
    Customer_Message VARCHAR(255),  
    Amount_Due FLOAT(5,2) NOT NULL);
```

```
CREATE TABLE ProductionOrders  
(ProductionOrders_ID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,  
    Product_ID INT NOT NULL,  
    Order_ID INT NOT NULL);
```

```
CREATE TABLE Transactions  
(Transaction_ID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,  
    Transaction_Date DATETIME NOT NULL,  
    Amount_Paid FLOAT(5,2) NOT NULL,  
    Ship_Date DATETIME NOT NULL);
```

```
CREATE TABLE Products  
(Product_ID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,  
    Name VARCHAR(100) NOT NULL,  
    Description VARCHAR(255) NOT NULL,  
    Price FLOAT(5,2) NOT NULL,  
    Manufacturer VARCHAR(255) NOT NULL,  
    Picture VARCHAR(100));
```



Übung 2

Erstellen Sie Indizes für die `Meet_A_Geek`-Datenbank. Bestimmen Sie zudem, welche Spalten indiziert werden sollen, indem Sie das Gelernte anwenden.

Einen Index könnten Sie auf folgende Spalten setzen:

- ▶ `Customer_ID` in der `Customer`-Tabelle
- ▶ `Shipper_ID` in der `Shippers`-Tabelle
- ▶ `Product_ID` und `Order_ID` in der `ProductionOrders`-Tabelle

Tag 7

Übung 1

Verwenden Sie das hier Gelernte und definieren Sie im `Meet_A_Geek`-Projekt alle Datentypen der Spalten neu.

Siehe Übung 1 des vorhergehenden Tages.

Übung 2

Üben Sie das Einfügen von Datensätzen in eine Tabelle, indem Sie diese in Spalten mit den Datentypen `SET` und `ENUM` einsetzen.

Hier ein paar Beispiele:

```
INSERT INTO Customers (Smoker) VALUES (,Y');
INSERT INTO Customers (Smoker) VALUES (NULL);
```

Das erste Beispiel fügt ein `Y` in die Tabelle ein. Das zweite Beispiel fügt den Standardwert ein, welcher in dieser Datenbank `N` ist.

Tag 8

Übung 1

Benutzen Sie den `mysqlimport`-Befehl samt den passenden Optionen, um folgende Aufgabe zu lösen.



Sie müssen ein Spreadsheet aus der Versandabteilung importieren. Die Versandabteilung stellt Ihnen dafür ein Worksheet zur Verfügung, der alle ausgegangenen Bestellungen beinhaltet. Die Felder sind durch Slashes (/) voneinander getrennt und die Daten mit einfachen Anführungszeichen umschlossen. Alle Daten auf dem Sheet sind neue Daten, alte Daten wurden nicht aktualisiert. Importieren Sie die Datei.

Verwenden Sie folgende Anweisung:

```
bin/mysqlimport -p -i -fields-enclosed-by=' ' -fields terminated-by=/'
```

Übung 2

Wenden Sie den `mysqldump`-Befehl im folgenden Szenario richtig an:

Der Chef möchte den Buchhaltern einen auf der `Orders`-Tabelle basierenden Bericht übergeben. Die Buchhalter benötigen die Anzahl und den Preis jedes bestellten Artikels, um den Umsatz zu berechnen. Dafür brauchen Sie nicht die DDL, jedoch eine durch Komma abgegrenzte Datei. Erstellen Sie die Datei.

Verwenden Sie folgende Anweisung:

```
bin/mysqldump -p -t --fields-terminated -by=,
Meet_A_Geek Orders >> Report.txt
```

Tag 9

Übung 1

Erstellen Sie aus folgenden Beispielen SQL-Anweisungen:

- Geben Sie alle Datensätze der Kunden aus der `Customers`-Tabelle aus, die Bestellungen aufgegeben haben.

```
SELECT * FROM Customers WHERE Customer_ID = Orders.Customer_ID
```

- Geben Sie alle Spediteure, die im letzten Monat beauftragt wurden, aus.

```
SELECT Shipper_Name FROM Shipper as S, Orders as O
WHERE O.Order_Date >= '2000-01-01'
AND O.Order_Date <= '2001-01-31'
AND O.Shipper_ID = S.Shipper_ID
```

- Geben Sie alle Kunden aus.

```
SELECT * FROM Customers
```



Übung 2

Fügen Sie folgenden Kunden in die Customers-Tabelle ein:

- ▶ Vorname: Jerry
- ▶ Nachname: Maxon
- ▶ Adresse: 123 N. Hampton Lane

```
INSERT INTO Customers (First_Name, Last_Name, Address1)
VALUES('Brent', 'Jacobs', '123 N. Hampton Lane')
```

Übung 3

Ersetzen Sie den Namen »Jerry« aus dem letzten Datensatz durch »Sam«.

```
UPDATE Customers SET First_Name = 'Sam'
WHERE Customer_ID = LAST_INSERT_ID()
```

Übung 4

Löschen Sie sämtliche Datensätze der Customers-Tabelle, deren Vorname »Jerry« lautet.

```
DELETE FROM Customers WHERE First_Name = 'Jerry'
```

Übung 5

Erstellen Sie eine SQL-Anweisung, die die Anzahl der Kunden zurückgibt, deren Name mit M beginnt.

```
SELECT COUNT(Customer_ID) as Cust_Count
FROM Customers
WHERE First_Name LIKE "M%"
```

Übung 6

Erstellen Sie eine SQL-Anweisung, die die Durchschnittsmenge der Produkte, die jeder Kunde kauft, zurückgibt (Hinweis: Benutzen Sie die quantity-Spalte).

```
SELECT AVG(Quantity) FROM Orders
```



Übung 7

Erstellen Sie eine SQL-Anweisung, die alle Kunden ausgibt, die im letzten Monat eine Bestellung aufgegeben haben. Sortieren Sie diese in aufsteigender alphabetischer Reihenfolge (A-Z) und gruppieren Sie die Datensätze nach Datum.

```
SELECT C.* FROM Orders as O, Customers as C
WHERE O.Order_Date >='2000-01-01'
AND O.Order_Date <= '2000-01-31'
AND C.Customer_ID = O.Customer_ID
ORDER BY C.Last_Name
GROUP BY O.Order_Date
```

Tag 10

Übung 1

Wenden Sie einige der heute gelernten Funktionen in Abfragen an. Benutzen Sie dabei die Meet_A_Geek- Datenbank.

Hier einige Beispiele:

```
SELECT SUM(Amount_Paid) AS Total FROM Transactions;
SELECT MAX(Customer_ID) AS Top FROM Customers;
```

Übung 2

Wie erhalten Sie die Gesamtzahl der am 1. März 2000 durchgeführten Transaktionen?

```
SELECT COUNT(Transaction_ID)
FROM Transactions
WHERE Transaction_Date = '20000301';
```

Tag 11

Übung 1

Stellen Sie sich vor, Sie haben eine Tabelle namens `Orders`, und Sie möchten diese aktualisieren, während niemand drittes aus dieser lesend zugreifen kann. Sie haben zu-



dem eine Tabelle `Products`, aus welcher Sie Daten abfragen, in der Sie jedoch keine Veränderungen vornehmen möchten. Schreiben Sie die richtige Anweisung mit den richtigen Sperren (`LOCK TABLES` und `UNLOCK TABLES`).

```
LOCK TABLES orders WRITE, products READ;
...
UNLOCK TABLES;
```

Übung 2

Stellen Sie sich die folgenden Situationen vor. In welchem würden Sie einen Schlüssel verwenden? Würden Sie einen eindeutigen oder einen nicht eindeutigen Schlüssel verwenden?

- ▶ Eine Tabelle mit allen Geburtstagen Ihrer Freunde. Ihr Computer greift einmal am Tag darauf zu, um zu sehen, ob jemand Geburtstag hat, und wenn das der Fall ist, dieser Person eine E-Mail zu schicken.
Einen nicht eindeutigen Schlüssel.
- ▶ Eine Tabelle mit Produkten, die über Ihren Online-Shop bestellt werden können. Doppelte Werte sind nicht erlaubt.
Einen eindeutigen Schlüssel.
- ▶ Eine Tabelle, die die Mehrwertsteuer aller verkauften Produkte speichert. In diese Tabelle wird jedes Mal, wenn ein Produkt verkauft wird, die jeweilige Summe geschrieben. Sie wird nur einmal im Quartal ausgelesen.
Keinen Schlüssel.

Tag 12

Übung 1

Nennen Sie Vor- und Nachteile des Gebrauchs von Schnittstellen im Gegensatz zum MySQL-Monitor!

- ▶ Vorteile:
Schnittstellen ermöglichen Ihnen, komplexe Datenbank-Operationen über eine Programmiersprache auszuführen.
Schnittstellen ermöglichen Ihnen, in die Datenbank eingegebene Daten zu kontrollieren und Fehlerüberprüfungen durchzuführen.



► Nachteile:

Sie müssen eine Programmiersprache wie PHP oder Perl beherrschen, um die Schnittstelle zu verwenden.

Die Entwicklung dauert länger.

Übung 2

Welche sind die Grundvoraussetzungen für eine Datenbankverbindung?

Servername oder IP-Adresse, Datenbankname, Benutzername und Passwort.

Tag 13

Übung 1

Stellen Sie eine Verbindung zu einer MySQL-Datenbank mit Hilfe von ADO her, und optimieren Sie Ihre Tabellen.

```
Dim mConn as Connection  
Dim mCmd as Command
```

```
With mConn  
    .ConnectionString = "Server=10.1.1.20;driver=MySQL;" &  
    "db=Meet_A_Geek;uid=mark;pwd=my2000"  
    .Open  
End With
```

```
With mCmd  
    .ActiveConnection=mConn  
    .CommandType = adCmdText  
    .CommandText = "OPTIMIZE TABLE Customers"  
    .Execute , , adExecuteNoRecords  
End With  
mConn.Close
```

Übung 2

Erstellen Sie eine ASP-Seite, die es dem Benutzer ermöglicht, Daten aus der Customers-Tabelle zu bearbeiten.

Das Script finden Sie auf der CD-Rom im Anhang-C-Verzeichnis.



Tag 14

Übung 1

Erstellen Sie ein Perl-Script, welches eine Anfrage vom Anwender entgegennimmt und Informationen aus der Datenbank aufgrund der übergebenen Werte anzeigt. Erstellen Sie z.B. für die Meet_A_Geek-Datenbank eine Webseite, die es einem Anwender ermöglicht, eine Liste der Kunden, basierend auf dem Bundesstaat, in dem diese leben, anzuzeigen. Übergeben Sie diese Werte an ein Perl-Script, welches diese in einer SELECT-Anweisung verwendet.

Die Scripts finden Sie auf der CD-ROM.

Übung 2

Erstellen Sie ein Perl-Script, welches Benutzer zu der mysql-Datenbank hinzufügt. (Hinweis: Verwenden Sie die GRANT-Anweisung.)

```
#!/usr/local/bin
use DBI;
$DSN = "DBI:mysql:Meet_A_Geek:10.1.1.50";
$dbh = DBI->connect($DSN,"mark","tacobell");
$dbh->do("GRANT ALL ON *.* TO Joe@% IDENTIFIED BY 'thecircus'");
$dbh->disconnect();
```

Tag 15

Übung 1

Schreiben Sie ein PHP-Script, das eine SELECT-Anweisung auf eine Tabelle Produkte ausführt. Die Tabelle besteht aus den Feldern Name und Preis. Ihre Abfrage sollte alle Produkte ausgeben, deren Preis niedriger als 50 DM ist. Sie sollten an eine Fehlerabfangroutine denken, die, wenn nötig, die benutzerdefinierte Funktion error_report() aufruft.

```
<?php
function error_report () {
    echo "Error: ".mysql_errno()."; error description:
    ".mysql_error()."<br>\n";
}
```



```
$sql = "SELECT name, price FROM products WHERE price < 50";
if ($result = mysql_query ($sql)) {
    while ($row = mysql_fetch_array ($result)) {
        echo "Product: ".$row[name]. " price ".$row[price]. "<br>\n";
    }
} else error_report ();
?>
```

Übung 2

Schreiben Sie eine PHP-Funktion, um Verbindung zu einem MySQL-Server auf localhost (Webserver) aufzunehmen. Die Funktion soll die Verbindungskennung an das aufrufende Programm zurückgeben.

```
<?php
function connect2database($host, $user, $pass, $database) {
    $vk = mysql_pconnect($host, $user, $pass)
        or die ("Verbindungsaufnahme fehlgeschlagen!");
    mysql_select_db($database, $vk)
        or die ("Verbindungsaufnahme mit Datenbank fehlgeschlagen!");
    return($vk);
}

$vk = connect2database("localhost", "Fred", "geheim", "DB");
?>
```

Tag 16

Übung 1

Schreiben Sie eine SELECT-Anweisung, welche die aktuelle Zeit im Format »Stunde:Minute am/pm Wochentag nth Monat Jahr« ausgibt.

```
SELECT DATE_FORMAT(NOW(), '%r %W %D %M %Y');
```

Übung 2

Schreiben Sie eine SELECT-Anweisung, um herauszufinden, auf welchen Wochetag der 13. Oktober 2010 fällt.

```
SELECT DAYNAME('2010-10-13');
```



Übung 3

Schreiben Sie eine `SELECT`-Anweisung, die 1000 Tage auf den 1. April 2000 addiert und das Datum zurückgibt. Auf welchen Wochentag wird der folgende Tag fallen?

```
SELECT DATE_ADD('2000-04-01',INTERVAL 1000 DAY);
```

Dies ist ein Freitag. Sie können das zweite Ergebnis mit dieser Anweisung erhalten:

```
DAYNAME(DATE_ADD('2000-04-01',INTERVAL 1000 DAY)).
```

Tag 17

Übung 1

Fügen Sie einige Benutzer zu der `Meet_A_Geek`-Datenbank hinzu, indem Sie sich der Techniken bedienen, die Sie in diesem Kapitel erlernt haben. Fügen Sie einen Benutzer mit der `GRANT`-Anweisung, einen mit dem `mysql_setpermission`-Programm und einen, indem Sie die `GRANT`-Tabellen direkt editieren, hinzu.

Unter Verwendung der `GRANT`-Anweisung:

```
GRANT ALL ON *.* TO Paul@% IDENTIFIED BY "horatio";
```

Verwenden Sie das `mysql_setpermission`-Programm. Wenn Sie nicht weiterkommen, verwenden Sie die `--help`-Option.

Um die `GRANT`-Tabellen zu ändern, müssen Sie die richtigen Rechte besitzen. Um einen Benutzer hinzuzufügen, geben Sie folgende Anweisung ein:

```
INSERT INTO user VALUES("%","mark", password("cougar"), "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y", "Y");
```

Um dem Benutzer Zugang zu einer bestimmten Datenbank zu erteilen, müssen Sie die `db`-Tabelle editieren.

Übung 2

Entziehen Sie den Benutzern, die Sie in Übung 1 erstellt haben, mit der `REVOKE`-Anweisung alle Rechte.

```
REVOKE ALL ON *.* TO Paul@%;
```



Übung 3

Löschen Sie die Benutzer wieder aus der Datenbank.

```
DELETE FROM user WHERE User="Paul " AND Host="%";
```

Tag 18

Übung 1

Begeben Sie sich auf die `crashme`-Webseite (<http://www.mysql.com/information/crash-me.php>), und vergleichen Sie MySQL mit der Datenbank Ihrer Wahl.

Auf der Seite finden Sie eine Auswahl von Datenbanken, mit denen Sie MySQL vergleichen können.

Übung 2

Erstellen Sie eine benutzerdefinierte Funktion ähnlich derjenigen, die am heutigen Tag vorgestellt wurde.

Um Ihre neue Funktion zu implementieren, müssen Sie diese in die `func`-Tabelle der `mysql`-Datenbank eintragen. Das machen Sie mit der `CREATE-FUNCTION`-Anweisung. Die Syntax ist wie folgt:

```
CREATE FUNCTION Funktionsname RETURNS Rückgabetyt SONAME "Datei.so"
```

Nachdem Sie die Funktion in die `func`-Tabelle eingetragen und die `GRANT`-Tabellen neu geladen haben, sollten Sie Ihre Funktion wie jede anderer MySQL-Funktion benutzen können.

Tag 19

Übung 1

Schreiben Sie eine Anweisung (Syntax), um eine Tabelle namens »Meine_Tabelle« auf gelöschte Blöcke zu untersuchen.

```
isamchk -r Meine_Tabelle
```



Übung 2

Geben Sie die Syntax an, um alle aktiven Threads eines MySQL-Servers anzuzeigen.

```
mysqladmin -u username -p processlist
```

Übung 3

Schreiben Sie einen Funktionsaufruf für den verschlüsselten String *my string* mit dem *Salt* in *ht1QbxA2IJPhU*.

```
SELECT ENCRYPT('my string','ht');
```

Tag 20

Übung 1

Führen Sie *myisamchk* mit dem *-recover*-Schalter auf alle Tabellen der *Meet_A_Geek*-Datenbank aus.

```
bin/myisamchk -r *.MYI
```

Übung 2

Bringen Sie die Voraussetzungen in Erfahrung, die ein Server haben muss, damit die *Meet_A_Geek*-Datenbank dort performant laufen kann. Beachten Sie dabei, dass die Datenbank rund 10.000-mal am Tag via Internet aufgerufen werden wird. Beachten Sie auch, dass MySQL am besten auf Unix-basierten Systemen läuft.

MySQL läuft am besten unter Unix-basierten Systemen. Für die *Meet_A_Geek*-Datenbank wäre ein »High-End«-Unix-Server die erste Wahl. Trotzdem ist das natürlich eine sehr subjektive Einschätzung. Die Datenbank würde wahrscheinlich auch auf einem Intel-Rechner mit Linux gut laufen.



Tag 21

Übung 1

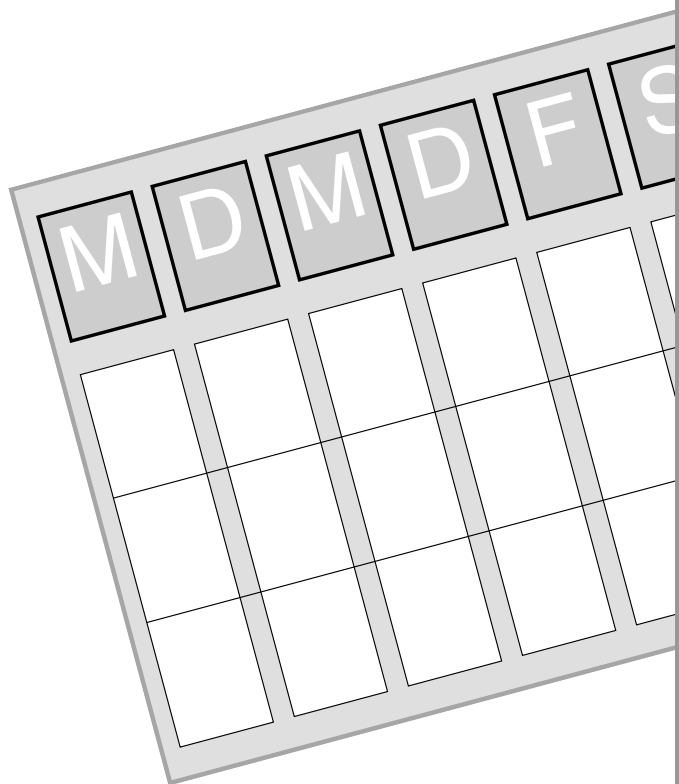
Zwei Scripts aus dem Flussdiagramm haben Sie noch nicht erstellt. Ein Script, um die Einträge der `Products`-Tabelle anzuzeigen, und eines, um Bestellungen aufzugeben. Erstellen Sie diese beiden Scripts, und verwenden Sie dabei Perl und die DBI/DBD.

Die beiden Scripts finden Sie auf der CD-ROM.

Übung 2

Erstellen Sie die Scripts aus der vorherigen Übung. Benutzen Sie jedoch dieses Mal ADO und ODBC.

Die Scripts finden Sie auf der CD-ROM.



Stichwortverzeichnis



- (Subtraktions-Operator) 307
- ! (Ausrufezeichen) 189
- #, --debug= (Option, mysqladmin) 402
- \$dbh
- , disconnect 277
- , do (SQL-Anweisung) 277
- , prepare (SQL-Anweisung) 277
- % (Modulo-Operator) 308
- % (Prozentzeichen) 357
- & (Kaufmanns-Und) 42
- && (doppeltes Kaufmanns-Und) 189
- ' (einfache Anführungszeichen) 243, 303
- () (Klammern) 184
- (Parameter, mysqld) 420
-) 45, 79, 96, 100, 303
- * (Asterisk) 158, 170, 185
- * (Multiplikations-Operator) 308
- .* (Syntax) 364
- + (Additions-Operator) 307
- , (Komma) 186
- . (Punkt) 361
- / (Divisions-Operator) 308
- <% Tag 246
- <input>-Tag 260
- = (Gleichheitszeichen) 170, 306
- , --help, mysqldump 148
- ?, --help (Option, isamchk) 410
- , --help (Option, mysqladmin) 403
- , --help (Option, mysqldump) 389
- , --help (Option, mysqlimport) 131, 393
- { } (geschweifte Klammern) 311
- | | (Pipe Symbole) 189

Numerics

- 1.NF (1. Normalform) 85
- 1:1-Beziehungen 57
- 1:n-Beziehungen 59
- 2.NF (2. Normalform) 85
- 3.NF (3. Normalform) 85
- 4.NF (4. Normalform) 86
- 5.NF (4. Normalform) 86

A

- a, --analyze (Option, isamchk) 409
- a, --analyze (Option, myisamchk) 425
- ABS(x)-Funktion 488
- abschließen 96

Access

- , Datenimport aus 136
- , DSN (Datenquellennamen) 136
- , exportsql.txt, Export-Tool 144
- , GUI (grafische Benutzeroberfläche) 142
- , importsql, Import-Tool 144
- , ODBC (Open Data Base Connectivity) 136
- , Telnet-Sitzung 142
- ACOS(x)-Funktion 488
- Active Server Pages siehe ASPs
- ActiveConnection (Recordset-Objekt) 257
- ActiveState-Webseite 271
- ActiveX Data Objects siehe ADO
- ADDDATE-Funktion 341, 488
- add-drop-table (Option, mysqldump) 146, 387
- Addition
 - , Funktion 184
 - , mathematische Funktionen 184
 - , Operator 307
- add-locks (Option, mysqldump) 146, 387
- ADO
 - , (ActiveX Data Objects) 237, 249
 - , Active Server Pages 268
 - , Command-Objekt 249
 - , Command-Objekt, ActiveConnection 252
 - , Command-Objekt, CommandText 252
 - , Command-Objekt, CommandTimeout 252
 - , Command-Objekt, CommandType 252
 - , Command-Objekt, Execute 253
 - , Command-Objekt, Methoden und Eigenschaften 252
 - , Command-Objekt, Prepared 253
 - , Command-Objekt, State 253
 - , Command-Objekt, Verwendung 253
 - , Connection-Objekt 249
 - , Connection-Objekt, Attributes 250
 - , Connection-Objekt, CommandTimeout 250
 - , Connection-Objekt, ConnectionString 250
 - , Connection-Objekt, ConnectionTimeout 250
 - , Connection-Objekt, CursorLocation 250
 - , Connection-Objekt, DefaultDatabase 250
 - , Connection-Objekt, Methoden und Eigenschaften 249
 - , Connection-Objekt, Open() 250

- , Connection-Objekt, Provider 250
- , Connection-Objekt, State 250
- , Connection-Objekt, Version 250
- , Datenbank-Verbindung 250
- , Erstellen von ADO-Objekten 246
- , Meet_A_Geek-Datenbank 258
- , Meet_A_Geek-Datenbank, Aufgeben einer Anzeige 260
- , Meet_A_Geek-Datenbank, Ausgabe von Daten auf der Webseite 263
- , Meet_A_Geek-Datenbank, Erstellen der Webseite 258
- , Performance 268
- , Recordset-Objekt 249
- , Recordset-Objekt, ActiveConnection 257
- , Recordset-Objekt, AddNew() 254
- , Recordset-Objekt, BOF, EOF 254
- , Recordset-Objekt, Clone() 254
- , Recordset-Objekt, Close() 254
- , Recordset-Objekt, CursorType 254, 257
- , Recordset-Objekt, Eigenschaften und Methoden 254
- , Recordset-Objekt, Fields 254
- , Recordset-Objekt, Find (Criteria, Skip, Dir, Start) 255
- , Recordset-Objekt, Format 255
- , Recordset-Objekt, LockType 255
- , Recordset-Objekt, MaxRecords 255
- , Recordset-Objekt, Move() 255
- , Recordset-Objekt, MoveFirst() 255
- , Recordset-Objekt, MoveLast() 255
- , Recordset-Objekt, MoveNext() 255
- , Recordset-Objekt, MovePrevious() 255
- , Recordset-Objekt, Open (Source, ActiveConnection, CursorType, LockType) 255
- , Recordset-Objekt, RecordCount 255
- , Recordset-Objekt, Sort 255
- , Recordset-Objekt, Source 255
- , Recordset-Objekt, State 255
- , Recordset-Objekt, Status 255
- , Recordset-Objekt, Update 255
- , Recordset-Objekt, Verwendung 256
- Ads-Tabelle, Meet_A_Geek-Datenbank 434
- Ändern von Tabellen 101
- Aggregat-Funktionen 171
- , AVG() 173
- , COUNT() 171
- , COUNT(DISTINCT) 172
- , GROUP BY-Klausel 174
- , MAX() 172
- , MIN() 173
- , STD() 173
- , SUM() 173
- , WHERE-Klausel 174
- Alias, Tabellen 159
- ALL (Berechtigung, Privileg) 359
- Alpha-Version 36
- ALTER (Berechtigung, Privileg) 359
- ALTER TABLE
 - , Befehl 102f., 120, 474
 - , Syntax 211
- ALTER, reservierte Wörter 156
- Alter_priv (Berechtigung, Privileg) 350
- Alter_priv, Spalte 350
- American National Standards Institute siehe ANSI
- AND
 - , Funktion 188
 - , reservierte Wörter 156
- Anfragen
 - , Ausführung 227
 - , Datenbank 224
- anonyme Benutzer 366
- ANSI
 - , ANSI92, ANSI-92-JOIN 168
 - , ANSI92, SQL (Structured Query Language) 167
 - , SQL92-Standard 32
- Anweisungen
 - , ALTER TABLE 474
 - , CASE 187
 - , COMMIT 202
 - , CREATE DATABASE 475
 - , CREATE FUNCTION 377, 475
 - , CREATE INDEX 475
 - , CREATE TABLE 475
 - , DELETE 166, 477
 - , DESCRIBE 477
 - , DROP DATABASE 478
 - , DROP FUNCTION 377, 478
 - , DROP INDEX 478
 - , DROP TABLE 478
 - , EXPLAIN 478
 - , FLUSH 479



- , GRANT 359f., 479
- , INSERT 162, 481
- , INSERT INTO 162
- , INSERT INTO...SELECT 163
- , KILL 481
- , LIKE 423
- , LOAD DATA INFILE 134, 224, 392, 437, 481
- , LOCK TABLES 199, 482
- , new 303
- , OPTIMIZE 167, 224, 428, 469
- , OPTIMIZE TABLE 482
- , REPLACE 176, 483
- , REVOKE 363, 483
- , ROLLBACK 202
- , SAVEPOINT 202
- , SELECT 160, 484
- , SELECT INTO 162
- , SELECT INTO OUTFILE 149
- , SET 164
- , SHOW 484
- , UNLOCK TABLES 201, 485
- , UPDATE 164, 485
- , USE 133, 485
- Apache
 - , httpd.conf-Datei 301
 - , srm.conf-Datei 301
- API (Application Programming Interface) 33
- Applikationen
 - , Client-Server, Datenfluss 182
 - , Front-End 220
 - , Middle Tier 181
 - , Schnittstelle (Interface) 220ff.
 - , SQL-Anweisungen 223
 - , Three-Tier 180f.
 - , Unternehmen, Datawarehousing 29
 - , Unternehmen, Flexibilität 29
 - , Unternehmen, Internet 29
 - , Unternehmen, Multi-User-Systeme 28
 - , Unternehmen, Sicherheit 29
 - , webgestützte Anwendung, Programmierlogik 180
- Argument, HIGH_PRIORITY 161
- arithmetische Operatoren
 - , Addition (+) 307
 - , Division (/) 308
 - , Modulo (%) 308
 - , Subtraktion (-) 307
 - , Subtraktion (*) 308
- array_walk()-Funktion (PHP) 315
- Arrays 314
 - , array_walk()-Funktion 315
 - , assoziative 314
 - , count()-Funktion 314
 - , each()-Funktion 315
 - , Hash (assoziatives Array) 287
 - , list()-Funktion 315
 - , next()-Funktion 315
 - , numerisch indizierte 314
 - , PHP (PHP Hypertext Preprocessor) 314
 - , prev()-Funktion 315
- AS, reservierte Wörter 156
- ASCII(x)-Funktion 489
- ASIN(x)-Funktion 489
- ASPs 238
 - , ADO 268
 - , CGI (Common Gateway Interface) 238
 - , FrontPage-Erweiterungen 238
 - , VBScript 240
- Assistenten, Export-Text-Assistent 139
- Asterisk (*) 158, 170, 185
- ATAN(x)-Funktion 489
- ATAN2(x,y)-Funktion 489
- Aufbau 184
- Ausrufezeichen (!) 189
- AUTO_INCREMENT 113
 - , Schlüssel 215
- , Spaltenattribut 113f., 119, 161
- AVG()-Funktion 173
- AVG(Ausdruck)-Funktion 490
- B**
 - b, --base-dir(Parameter, mysqld) 420
 - back_log 419
 - Back-End-Datenbank 220
 - Backup
 - , Dateien in Master-Slave-Datenbanken 391
 - , Daten 386
 - , Daten, Aktualisierungsprotokoll 394
 - , Daten, Datei 386
 - , Daten, Protokollierungsarten 395

- , Datenbank 203, 412
- , Tabellen 203
- , volles oder inkrementelles 394
- Batchmodus 132
- BCP siehe Bulk Copy Protocol
- Befehle
 - , ALTER TABLE 102f., 474
 - , break, for- oder while- Schleife 311
 - , COMMIT 202
 - , continue, for- oder while- Schleife 311
 - , CREATE 72
 - , CREATE (mysqladmin) 76
 - , CREATE DATABASE 72, 475
 - , CREATE Datenbankname (mysqladmin) 401, 403
 - , CREATE FUNCTION 377, 475
 - , CREATE INDEX 475
 - , CREATE TABLE 475
 - , DELETE 166, 477
 - , DESCRIBE 102, 477
 - , DROP 75
 - , DROP (mysqladmin) 76
 - , DROP DATABASE 75, 478
 - , DROP Datenbankname (mysqladmin) 401, 403
 - , DROP FUNCTION 377, 478
 - , DROP INDEX 478
 - , DROP TABLE 478
 - , EXPLAIN 478
 - , extended-status (mysqladmin) 401
 - , FLUSH 479
 - , flush-hosts (mysqladmin) 401
 - , flush-logs (mysqladmin) 401
 - , flush-privileges (mysqladmin) 401
 - , flush-tables (mysqladmin) 401
 - , GRANT 359f., 479
 - , INSERT 162, 481
 - , INSERT INTO 162
 - , INSERT INTO...SELECT 163
 - , KILL 481
 - , kill 43
 - , kill id1, id2 (mysqladmin) 402
 - , LIKE 423
 - , LOAD DATA INFILE 134f., 224, 392, 437, 481
 - , LOCK TABLES 199, 482
 - , mysql (MySQL-Monitor) 75
 - , mysqladmin 76
 - , mysqladmin, Datenbank erstellen 436
 - , mysqladmin, shutdown 43
 - , mysqld 41
 - , mysqlimport 130
 - , OPTIMIZE 167, 224, 428, 469
 - , OPTIMIZE TABLE 482
 - , password neues_Passwort (mysqladmin) 402
 - , Ping (mysqladmin) 402
 - , processlist (mysqladmin) 401, 406
 - , Refresh (mysqladmin) 402
 - , Reload (mysqladmin) 402
 - , REPLACE 176, 483
 - , REVOKE 363, 483
 - , ROLLBACK 202
 - , safe_mysql & 42
 - , SAVEPOINT 202
 - , SELECT 160, 484
 - , SELECT INTO 162
 - , SELECT INTO OUTFILE 149
 - , SET 164
 - , SHOW 484
 - , shutdown (mysqladmin) 402
 - , Status (mysqladmin) 401
 - , UNLOCK TABLES 201, 485
 - , UPDATE 164, 485
 - , USE 74, 133, 485
 - , Variablen (mysqladmin) 401
 - , Version (mysqladmin) 401
 - BEGIN TRAN 372
 - BENCHMARK (x, Ausdruck)-Funktion 490
 - Benutzer
 - , anonym 366
 - , Datenbanken 366
 - , GRANT-Anweisung, hinzufügen 357
 - , hinzufügen 78
 - , löschen 363
 - , mit ADO hinzufügen 267
 - , Name 223
 - , Rechte entziehen 363
 - , Unternehmen 28
 - Benutzerrechte
 - , Änderungen 356
 - , GRANT-Anweisung 359



- , Löschen 358, 363
- , Vergabe 356, 358
- Berechtigungen
 - , ALL 359
 - , ALTER 359
 - , Alter_priv 350
 - , CREATE 359
 - , Create_priv 349
 - , DELETE 350, 359
 - , Delete_priv 349
 - , DROP 359
 - , Drop_priv 349
 - , FILE 359
 - , File_priv 350
 - , Grant_priv 350
 - , Host 349
 - , INDEX 359
 - , Index_priv 350
 - , INSERT 359
 - , Insert_priv 349
 - , password 349
 - , PROCESS 360
 - , Process_priv 350
 - , REFERENCES 360
 - , References_priv 350
 - , RELOAD 360
 - , Reload_priv 349
 - , SELECT 360
 - , Select_priv 349
 - , SHUTDOWN 360
 - , Shutdown_priv 349
 - , UPDATE 360
 - , Update_priv 349
 - , USAGE 360
 - , User 349
- beschädigte Daten mit isamchk reparieren
 - , Schlüssel 411
 - , Tabellen 409
- beschädigte Datenbank, Reparatur 407
 - , Datenstruktur 407
 - , Festplattenplatz wiedergewinnen 408
 - , isamchk 409
- Beseitigung
 - , Nicht-Schlüssel Daten 89
 - , Teilabhängigkeiten 88
- Bewegung, Open Source 26
- Beziehungen
 - , 1 57
 - , 1:n 59
 - , m:n 60
- Beziehungen, zwischen Tabellen
 - , definieren 67
 - , eine-zu-vielen 59
 - , eins-zu-eins 57
 - , festlegen 56
 - , viele-zu-vielen 60
- BIGINT-Datentyp 111f.
 - big-tables (Parameter, mysqld) 420
- Bilder, speichern 121, 469
- BIN(x)-Funktion 490
- Binär-Versionen, MySQL 36
- BINARY, Spaltenattribut 119
- bin-Verzeichnis 39
- BLOB (Binary Large Object)-Datentyp 115f.
- Boyce-Codd-Normalform 91
- break-Befehl, in for- oder while-Schleife 311
- Browser 289
- buffers 419
- Bulk Copy Protocol (BCP) 128
- C**
 - c, --complete_insert (Option, mysqldump) 146
 - C, --compress (Option, mysqladmin) 402
 - C/C++-Programmiersprache 375
 - , UDF (User Defined Function) 375
 - CASE (Anweisung, Syntax) 187
 - CD-ROM
 - , DBMS (Datenbank Management System) 26
 - , Perl DBI, Installation 272
 - CEILING(x)-Funktion 491
 - CGI
 - , Perl 270f., 281f., 284, 287ff.
 - , Perl DBI 270f., 281, 287
 - , PHP 298f.
 - ChangeLog-Datei 39
 - CHAR(X)-Datentyp 115
 - CHAR(x, y, z, ...)-Funktion 491

- CHARACTER_LENGTH-,
 CHAR_LENGTH- Funktion 491
- Client
 - , Fat Client 31
 - , Thin Client 31
- Client/Server
 - , Applikation, Datenfluss 182
 - , Paradigma 30
- COALESCE (Ausdruck1, Ausdruck2, ...)
 - Funktion 491
- Codd, E.F. 30
- Column_priv-Spalte 351
- columns_priv-Tabelle 348, 351, 355
- Command-Objekt
 - , ActiveConnection (ADO) 252
 - , CommandText (ADO) 252
 - , CommandTimeout (ADO) 252
 - , CommandType (ADO) 252
 - , Eigenschaften 252
 - , Execute() (ADO) 253
 - , Methoden 252
 - , Prepared (ADO) 253
 - , State (ADO) 253
 - , Verwendung 253
- COMMIT-Befehl 202
- Common Gateway Interface (CGI) siehe CGI
- Composite-Index
 - , Schlüssel für mehrere Spalten 207
 - , Teil-Schlüssel 208
 - , zusammengesetzte Teil-Schlüssel 208
- Comprehensive Perl Archive Network siehe
 CPAN
- concat()-Funktion 160
- CONCAT(x, y, z, ...)-Funktion 492
- configure
 - , Befehl 40
 - , Datei 39
- Connect (processlist-Befehl) 406
- Connection-Objekt
 - , Attributes (ADO) 250
 - , CommandTimeout (ADO) 250
 - , ConnectionString (ADO) 250
 - , ConnectionTimeout (ADO) 250
 - , CursorLocation (ADO) 250
 - , DefaultDatabase (ADO) 250
 - , Eigenschaften (ADO) 249
 - , Methoden (ADO) 249
 - , Open() (ADO) 250
 - , Provider (ADO) 250
 - , State (ADO) 250
 - , Version (ADO) 250
 - , Verwendung 250
- Constraints (Beschränkungen) 381
- continue, in for- oder while-Schleifen 311
- CONV(x, von-Basis, nach-Basis)-Funktion
 492
- COS(x)-Funktion 492
- COT(x)-Funktion 492
- COUNT()-Funktion 171
- count()-Funktion (PHP) 314
- COUNT(DISTINCT)-Funktion 172
- COUNT(x)-Funktion 493
- CPAN (Comprehensive Perl Archive Net-
 work) 270
- crash_me-Tool 40
- CREATE
 - , (Berechtigung, Privileg) 359
 - , Befehl 72
 - , Befehl (mysqladmin) 76
 - , DATABASE Befehl 72, 475
 - , Datenbankname Befehl (mysqladmin) 401,
 403
 - , DB (processlist-Befehl) 406
 - , FUNCTION-Befehl 475
 - , FUNCTION-Syntax 377
 - , INDEX Befehl 475
 - , reservierte Wörter 156
 - , TABLE-Befehl 475
 - , TABLE-Syntax 205
- Create_priv (Berechtigung, Privileg) 349
- Create_priv, Spalte 349
- cron-Datei 427
- cron-Job 427
- CROSS JOIN 169
 - , reservierte Wörter 156
- CSV-Datei 129
 - , Import 131
- CURDATE()-Funktion 336, 493
- CURRENT_DATE()-Funktion 336, 493
- CURRENT_TIME()-Funktion 336, 493
- CURRENT_TIMESTAMP()-Funktion 336,
 493



Cursor 379f.
CursorType-Parameter (Recordset-Objekt) 257
CURTIME()-Funktion 493
Customers-Normalisierung 86
Customers-Tabelle
–, Charakteristika 433
–, Datentypen 67
–, Entwurf 433
–, Erstellen 96
–, Kopie erstellen 99
–, Meet_A_Geek_Datenbank 433
 Webseiten erstellen 439

D
–d, --delete (Option, mysqlimport) 130, 392
–d, --description (Option, isamchk) 409
–d, --description (Option, myisamchk) 425
–d, --no-data (Option, mysqldump) 147, 387
Daemon 41, 133
–, Datenbank-Server 220
–, mysqld 42, 73, 96
–, Threads 196
Data Access Objects (DAO) 237
Data Creation Language (DCL) 155
Data Manipulation Language (DML) 155
data_sources()-Methode 276
DATABASE()-Funktion 397, 494
--datadir=vollständiger Pfad (Parameter, mysqld) 420
data-Verzeichnis 39
Datawarehousing, Unternehmen 29
DATE_ADD (Datum, INTERVALL Ausdruck Typ)-Funktion 341
DATE_ADD (Datum, INTERVALL Wert Typ)-Funktion 494
DATE_FORMAT (Datum, Format)-Funktion 337
DATE_FORMAT (Datum, Formatzeichen)-Funktion 495
DATE_SUB (Datum, INTERVALL Ausdruck Typ)-Funktion 341
DATE_SUB (Datum, INTERVALL Wert Typ) und SUBDATE (Datum, INTERVALL Wert Typ)-Funktion 496
DATE-Datentyp 327

Dateien
–, ChangeLog 39
–, configure 39
–, INSTALL-BINARY 39
–, manual.htm 40
–, manual.txt 40
–, manual_toc.htm 40
–, mysql_for_dummies 40
–, mysqld.exe 43
–, PUBLIC 39
–, README 39
–l, --log= 420
mysqld, -l, --log= 420

Daten
–, Ausgabe, auf Meet_A_Geek-Datenbank-Webseiten 263
–, Backup 386
–, DELETE-Anweisung 166
–, Export 151
–, Export, Methoden 145
–, Export, mysqldump-Utility 145, 149
–, Export, Optionen 145
–, Export, SELECT INTO OUTFILE-Anweisung 149
–, Import 437
–, Import, aus Access 136
–, Import, Bearbeitung mit SQL (Structured Query Language) 158
–, Import, Bulk Copy Protocol (BCP) 128
–, Import, CSV-Datei 129
–, Import, Datei-Konvertierungsvorgang 128
–, Import, INSERT-Anweisung 162
–, Import, LOAD DATA INFILE-Anweisung 134
–, Import, Methoden 128
–, Import, mysqlimport-Programm 130, 392
–, Import, REPLACE-Anweisung 176
–, Import, SELECT-Anweisung 160
–, Import, Speichern in Dateien 49
–, Import, UPDATE-Anweisung 164
–, Import, XML (Extensible Markup Language) 129
–, Wiederherstellen, mysqlimport-Programm 392

Datenbank Management System (DBMS) 26

Datenbanken

- , ADO (ActiveX Data Objects), Command-Objekt 252
- , ADO (ActiveX Data Objects), Connection-Objekt, Methoden oder Eigenschaften 249
- , ADO (ActiveX Data Objects), Meet_A_Geek 258ff., 262f., 265ff.
- , ADO (ActiveX Data Objects), Recordset-Objekt 254
- , ADO (ActiveX Data Objects), Verbindung zu 249
- , Ads-Tabelle 434
- , Aggregat-Funktionen 171, 174
- , Anfragen ausführen 227
- , anonyme Benutzer 366
- , Anweisungen, Syntaxregeln 158
- , ASPs (Active Server Pages) 237
- , auf Bändern gesichert 396
- , Benutzer 78
- , Benutzerrechte 438
- , Beschädigungen, reparieren 407
- , Beschränkungen (Constraints) 381
- , Bilder speichern 121
- , Client/Server-Paradigma siehe Spalten, Indizes, Meet_A_Geek-Datenbank, Tabellen
- , CREATE-Befehl 72
- , Customers, Schema 86
- , Customers-Tabelle 433
- , Datenbank-Anfragen 224
- , Datensätze erhalten 225
- , Datenwiederholung, aussondern 87
- , DBD (Datenbank-Treiber) 270
- , DBI (Datenbank Interface) 269
- , db-Tabelle 78
- , Definition 27
- , DELETE-Befehl 166
- , Design 52
- , Design, Geschäftsablauf definieren 53
- , Design, Geschäftsobjekte definieren 54
- , Design, Geschäftsregeln definieren 55
- , Design, Modellierung 55
- , Design, Tabellen-Beziehungen 56f., 61, 67
- , DSN 233
- , Ergebnissatz 225
- , erstellen 61, 75, 80, 430, 436
- , erstellen, Funktionen 316
- , erstellen, mit Schnittstellen 227
- , Front-End Applikation 220
- , GRANT-Tabellen 438
- , Handles, \$dbh 277
- , Handles, erstellen 276
- , host-Tabelle 78
- , Import 437
- , Import-Werkzeuge 144
- , INSERT Befehl 162
- , Joins 167
- , Joins, CROSS JOIN 169
- , Joins, INNER JOIN 169
- , Joins, LEFT JOIN 170
- , Joins, NATURAL LEFT JOIN 171
- , Joins, RIGHT JOIN 176
- , löschen 75, 77
- , löschen, DROP Befehl 75
- , löschen, Funktionen 316
- , Master, Dateien, sichern 391
- , Modell 435
- , MyODBC-Treiber, Installation 230
- , MySQL, Monitor 155
- , MySQL, RDBMS 81
- , mysqladmin 401
- , mysqladmin, Datenbank erstellen 403
- , mysqladmin, Datenbank löschen 403
- , mysqladmin, MySQL-Server 404
- , mysql-Datenbank 48
- , Name, Schnittstellen-Verbindungsanforderungen 222
- , Normalisierung siehe Normalisierung von Datenbanken 84
- , ODBC (Open Database Connectivity) 230
- , Optimierung, Überblick 426
- , OPTIMIZE-Befehl 469
- , Orders-Tabelle 434
- , Performancesteigerung 421
- , Products-Tabelle 434
- , RDBMS (Relationale Datenbank-Management-Systeme) 110
- , relationale, Codd, E.F. 30
- , relationale, Definition 30
- , REPLACE-Anweisung 176
- , sample_db, MySQL-Monitor-Befehle 73
- , Schemata, importieren 107
- , Schemata, mit Scripts erstellen 100



- , Schnelligkeit, MySQL im Vergleich 370
- , Schnittstellen siehe Schnittstellen
- , Scripts erstellen 438
- , SELECT-Befehle 160
- , Server 45, 220, 317
- , Sicherheit siehe Sicherheit
- , Sicherheitsaspekte 438
- , sichern 203, 412f.
- , siehe Spalten, Indizes,
Meet_A_Geek_Datenbank, Tabellen 27
- , Slave, Dateien, sichern 391
- , sortieren 174
- , Sperren 196f., 199f.
- , SQL (Structured Query Language) 154
- , SQL (Structured Query Language), Komponenten 155
- , SQL (Structured Query Language), Syntax 156
- , Struktur, Anzeige 47
- , symbolische Links 73
- , Table-Scan 204
- , Three-Tier-Design 181
- , transaktionale Kontrolle, Alternativen 202
- , Treiber 136
- , T-SQL (Transact-SQL) 371
- , UPDATE-Befehl 164
- , user-Tabelle 78
- , VBScript 237, 240
- , VBScript, ASPs (Active Server Pages) 241
- , VBScript, Funktionen 241
- , VBScript, Kommentieren des Programm-codes 243
- , VBScript, Programmfluss 242
- , VBScript, Schleifenkontrolle 242
- , VBScript, Varianten 241
- , VBScript, Webseite 243
- , Verbindung 223, 226
- , verschieben 396
- , verwalten 28, 401
- , Webseiten erstellen 438
- Datenbank-Handle (\$dbh) 280
- Datenbank-Interface siehe DBI
- Datenbank-Management-System (DBMS) 41
- Datenbank-Tabellen
 - , Schlüssel, AUTO_INCREMENT 215
 - , Schlüssel, Befehle zur Erstellung 205
 - , Schlüssel, definieren 205
 - , Schlüssel, eindeutige 209, 215
 - , Schlüssel, Fremd- 211
 - , Schlüssel, für einzelne Spalten 205
 - , Schlüssel, für mehrere Spalten 207
 - , Schlüssel, Gebrauch 218
 - , Schlüssel, löschen 215
 - , Schlüssel, nicht-eindeutig 217
 - , Schlüssel, Primär- 211, 215
 - , Schlüssel, Prinzipien 203
 - , Schlüssel, sinnvolle Auswahl 213
 - , Schlüssel, Teil- 208
 - , Schlüssel, Vorteile 204
 - , Schlüssel, wann benutzen 213
 - , Schlüssel, wann nicht benutzen 214
 - , Schlüssel, zusammengesetzte Teil- 208
 - , Sperren 217
- Datenbank-Treiber siehe MySQL-DBD
- Datenfluss, Client-Server-Applikation 182
- Datenquellen-Administrator 231, 233
- Datenquellen-Dialogfeld 233
- Datenquellenennamen 136, 268
 - , erstellen 233
 - , Meet_A_Geek-Datenbank, hinzufügen 233
- Datenstruktur 407
- Datentypen
 - , BIGINT 111f.
 - , BLOB 115f.
 - , CHAR(X) 115
 - , DATE 327
 - , DATETIME 327
 - , DECIMAL (M, D) 111f.
 - , DOUBLE (M, D) 111f.
 - , ENUM 117
 - , FLOAT (M, D) 111f.
 - , INT 111f.
 - , LONGBLOB 115
 - , LONGTEXT 115
 - , MEDIUMBLOB 115
 - , MEDIUMINT 111f.
 - , MEDIUMTEXT 115
 - , SET 117
 - , SMALLINT 111f.
 - , TEXT 115f.
 - , TIME 327
 - , TIMESTAMP 327f.
 - , TINYBLOB 115
 - , TINYINT 111f.

- , TINYTEXT 115
- , VARCHAR(X) 115
- , YEAR 327f.
- Datenzugang, Indizes, Anfragen 104
- DATETIME-Datentyp 327
- Datum
 - , TIMESTAMP-Datentyp, Felder 329
 - , Typen 494
 - , Werte 499
- DAYNAME (Datum)-Funktion 333, 496
- DAYOFMONTH (Datum)-Funktion 331, 497
- DAYOFWEEK (Datum)-Funktion 331, 497
- DAYOFYEAR (Datum)-Funktion 331, 497
- DBD (Datenbank-Treiber) siehe MySQL-DBD
- dbh (Datenbank-Handle) 280
- DBI, (Datenbank Interface) 269
 - CGI (Common Gateway Interface) 281
- , Perl, Objekt 276
- , Perl, Verbindung zu Datenbank 276
- DBMS, siehe Datenbank Management System
- Db-Spalte 351
- db-Tabelle 78, 348, 351, 354
- DCL (Data Creation Language) 155
- debug (mysqld) 395
- DECIMAL (M, D) Datentyp 111f.
- DECODE (Binärer-String, Passwort)-Funktion 498
- DECODE (encoded_string, password)-Funktion 399
- DEFAULT, Spaltenattribut 119
- Definitionen
 - , Geschäftsablauf 53
 - , Geschäftsobjekte 54
 - , Geschäftsregeln 55
 - , Schlüssel 205
 - , Datenbanken 27
 - , PHP-Funktionen 313
 - , relationale Datenbanken 30
 - , Tabellen-Beziehungen 67
- DEGREES(x)-Funktion 498
- delayed_insert-Option, mysqldump) 146
- delayed_queue_size 419
- DELETE
 - , (Berechtigung, Privileg) 350, 359
 - , Anweisung 166
 - , Befehl 166, 477
 - , Berechtigung 350
 - , reservierte Wörter 156
 - , Tabellen 159
- Delete_priv (Berechtigung, Privileg) 349
- Delete_priv, Spalte 349
- DESCRIBE-Befehl 102, 477
- Design
 - , Datenbanken- 52
 - , Datenbanken, erstellen 61
 - , Datenbanken, Geschäftsablauf definieren 53
 - , Datenbanken, Geschäftsobjekte definieren 54
 - , Datenbanken, Geschäftsregeln definieren 55
 - , Datenbanken, Meet_A_Geek 62, 64f., 67
 - , Datenbanken, Modellierung 55
 - , Datenbanken, Tabellen-Beziehungen 57, 60f., 67
 - , Meet_A_Geek-Datenbank, erstellen 431
 - , Meet_A_Geek-Datenbank, normalisieren 434
 - , Three-Tier 181
 - , Webseiten, Meet_A_Geek-Datenbank 258
- Dialogfeld
 - , Datenquellen- 231, 233
 - , Exportieren einer Anfrage 138
 - , Treiber-Installations- 231f.
- disconnect-Anweisung 277
- Division
 - , Funktion 186
 - , Operator 308
- DML (Data Manipulation Language) 155
- do (SQL-Anweisung)> 277
- do()-Methode 280
- Domain Key-Normalform 92
- DOUBLE (M, D)-Datentyp 111f.
- Download
 - , MySQL 36
 - , Perl DBI (Datenbank-Interface) 271
- DROP
 - , (Berechtigung, Privileg) 359
 - , Befehl 75
 - , Befehl (mysqladmin) 76
 - , DATABASE-Befehl 75, 478
 - , Datenbanken löschen 75
 - , Datenbankname-Befehl 401



- , Datenbankname-Befehl (mysqladmin) 403
- , FUNCTION-Befehl 377, 478
- , FUNCTION-Syntax 377
- , Groß- und Kleinschreibung 77
- , INDEX-Befehl 478
- , reservierte Wörter 156
- , TABLE-Befehl 478
- Drop DB, (processlist-Befehl) 406
- Drop_priv (Berechtigung, Privileg) 349
- Drop_priv, Spalte 349
- DSN siehe Datenquellennamen
- dumme Terminals 30
- Dynamic (CursorType-Parameter) 257

E

- e, --extended-check (Option, isamchk) 409
- e, --extended-insert (Option, mysqldump) 146, 387
- each()-Funktion (PHP) 315
- Ebenen, Schnittstellen (ADO) 268
- echo()-Funktion (PHP) 301
- Eigenschaften
 - , Command-Objekt 252
 - , Connection-Objekt 249
 - , QueryString 247
 - , Recordset-Objekt 254
- eindeutige Schlüssel 209, 215
- eingeben 79
- ELT(X, a, b, c, ...)-Funktion 498
- enable-locking (Parameter, mysqld) 420
- ENCLOSED BY, Schlüsselwort 135
- ENCODE(string, password)-Funktion 399
- ENCODE(Wort, Passwort)-Funktion 498
- ENCRYPT (Wort- [, Salt])-Funktion 499
- ENCRYPT(string[,salt])-Funktion 398
- entfernen, gelöschte Datensätze 167
- ENUM-Datentyp 117
- Errcode
 - , 13 389
 - , 2 389
- Erstellen der Webseite, Skripte erstellen 439
- ESCAPED BY, Schlüsselwort 135
- execute() 279
- EXP(x)-Funktion 499
- EXPLAIN-Befehl 478
- Export
 - , Daten 150

- , Methoden 145
- , mysqldump 145
- , mysqldump-Optionen 146
- , Werkzeuge 144
- Exportieren einer Anfrage-Dialogfeld 138
- exportsql.txt, Datei 144
- Export-Text-Assistent 139
- extended-status, Befehl 401
- Extensible Markup Language siehe XML
- EXTRACT (Wert FROM Datum)-Funktion 499

F

- F, --flush-logs (Optionen, mysqldump) 147, 387
- f, --force (Option, isamchk) 409
- f, --force (Option, myisamchk) 425
- f, --force (Option, mysqladmin) 402
- f, --force (Option, mysqldump) 147
- f, --force (Option, mysqlimport) 130
- FAQ, Perl-Webseite 271
- Fat Client 31
- Fehlerbehandlung (PHP) 319
- Festplattenplatz wiedergewinnen 408
- fetchrow_hashref() 279
- Field list (processlist-Befehl) 406
- FIELD(x, a, b, c ...)-Funktion 500
- fields-enclosed-by=Trennzeichen (Option, mysqldump) 147, 388
- fields-enclosed-by=Trennzeichen (Option, mysqlimport) 131, 148, 393
- fields-escaped-by=Escape-Zeichen (Option, mysqldump) 148, 388
- fields-escaped-by=Escape-Zeichen (Option, mysqlimport) 131, 148, 393
- fields-optionally-enclosed-by=Trennzeichen (Option, mysqldump) 147, 388
- fields-optionally-enclosed-by=Trennzeichen (Option, mysqlimport) 131, 148, 393
- fields-terminated-by=Trennzeichen (Option, mysqldump) 147, 388
- fields-terminated-by=Trennzeichen (Option, mysqlimport) 131, 148, 393
- FILE (Berechtigung, Privileg) 359
- File-priv (Berechtigung, Privileg) 350
- File-priv, Spalte 350
- FIND_IN_SET(x, String-Liste)-Funktion 500

- finish() 279
- Flexibilität 28
- FLOAT (M, D)-Datentyp 111f.
- FLOOR(x)-Funktion 500
- FLUSH
 - , (Berechtigung, Privileg) 224
 - , Befehl 479
 - , GRANT-Tabellen neu laden 359
- Flush tables, mysqladmin Statusbericht 404
- flush_time 419
- flush-hosts-Befehl 401
- flush-logs-Befehl 401
- flush-privileges-Befehl 401
- flush-tables-Befehl 401
- Flussdiagramm, Meet_A_Geek-Datenbank 432
- FORMAT(Num, Dec)-Funktion 501
- Formatzeichen 495
- FORM-Tag 282
 - , Eigenschaften 247
 - , GET-Method 282
 - , POST-Method 282
- for-Schleifen 310
- Fremdschlüssel 211
- Frequently Asked Questions siehe FAQ
- FROM
 - , reservierte Wörter 156
 - , Schlüsselwort 168, 364
- FROM_DAYS(Tage)-Funktion 501
- FROM_DAYS(Zahl)-Funktion 339
- FROM_UNIXTIME (UNIX_TIMESTAMP-[Format])Funktionen 340
- FROM_UNIXTIME (Unix-Zeit-[Formatzeichen])Funktionen 501
- Front-End-Applikation 220
- FrontPage-Erweiterungen 238
- full (Option, mysqldump) 147
- FULL JOIN, reservierte Wörter 156
- func-Tabelle 348
- Funktion
 - , TRIM([BOTH oder LEADING oder TRAILING] [x] FROM y)) 521
 - , WEEK(Datum [, Start]) 334, 524
- Funktionen 341
 - , ABS(x) 488
 - , ACOS(x) 488
 - , ADDDATE(Datum, INTERVALL Ausdruck Typ) 341
 - , ADDDATE(Datum, INTERVALL Interval-Ausdruck) 488
 - , Aggregat 158, 171
 - , AND 188
 - , array_walk() (PHP) 315
 - , ASCII(x) 489
 - , ASIN(x) 489
 - , ATAN(x) 489
 - , ATAN2(x,y) 489
 - , Aufbau 184
 - , AVG (Ausdruck) 490
 - , AVG() 173
 - , BENCHMARK(x, Ausdruck) 490
 - , BIN(x) 490
 - , CASE 187
 - , CEILING(x) 491
 - , CHAR(x, y, z, ...) 491
 - , CHARACTER_LENGTH, CHAR_LENGTH 491
 - , COALESCE(Ausdruck1, Ausdruck2, ...) 491
 - , concat() 160
 - , CONCAT(x, y, z, ...) 492
 - , CONV(x, von-Basis, nach-Basis) 492
 - , COS(x) 492
 - , COT(x) 492
 - , COUNT() 171
 - , count() (PHP) 314
 - , COUNT(DISTINCT) 172
 - , COUNT(x) 493
 - , CURDATE() 336, 493
 - , CURRENT_DATE() 336, 493
 - , CURRENT_TIME() 336, 493
 - , CURRENT_TIMESTAMP() 336, 493
 - , CURTIME() 336, 493
 - , DATABASE() 397, 494
 - , DATE_ADD(Datum, INTERVALL Ausdruck Typ) 341
 - , DATE_ADD(Datum, INTERVALL Wert Typ) 494
 - , DATE_FORMAT(Datum, Format) 337
 - , DATE_FORMAT(Datum, Formatzeichen) 495



- , DATE_SUB(Datum, INTERVALL Wert Typ) und SUBDATE(Datum, INTERVALL Wert Typ) 496
- , DAYNAME(Datum) 333, 496
- , DAYOFMONTH(Datum) 331, 497
- , DAYOFWEEK(Datum) 331, 497
- , DAYOFYEAR(Datum) 331, 497
- , DECODE(binärer-String, Passwort) 498
- , DECODE(kodierter String, Passwort) 399
- , DEGREES(x) 498
- , each() (PHP) 315
- , echo() (PHP) 301
- , ELT(X, a, b, c, ...) 498
- , ENCODE(String, Passwort) 399
- , ENCODE(Wort, Passwort) 498
- , ENCRYPT(string[,salt]) 398
- , EXP(x) 499
- , EXTRACT(Wert FROM Datum) 499
- , FIELD(x, a, b, c ...) 500
- , FIND_IN_SET(x, String-Liste) 500
- , FLOOR(x) 500
- , FORMAT(Num, Dec) 501
- , FROM_DAYS(Tage) 501
- , FROM_DAYS(Zahl) 339
- , GREATEST(x, y, ...) 502
- , HEX(x) 502
- , HOUR(Zeit) 338, 502
- , IF() 187
- , IF(Ausdruck1, Ausdruck2, Ausdruck3) 503
- , IFNULL() 188
- , IFNULL(Ausdruck1, Ausdruck2) 503
- , include() (PHP) 312
- , INSERT(x, y, z, j) 504
- , INSTR(x, y) 504
- , ISNULL(Ausdruck) 503
- , LAST_INSERT_ID() 114
- , LAST_INSERT_ID() 504
- , LAST_INSERT_ID([Ausdruck]) 400
- , LCASE(x) 504
- , LEAST(x, y, z, ...) 505
- , LEFT(x, y) 505
- , LENGTH(x) 506
- , list() (PHP) 315
- , LOAD_FILE(Dateiname) 506
- , LOCATE() 191
- , LOCATE(x, y, z) 506
- , LOG(x) 507
- , LOG10(x) 507
- , LOWER(x) 507
- , LPAD(x, y, z) 508
- , LTRIM() 190
- , LTRIM(x) 508
- , MAX() 172
- , MAX(Ausdruck) 508
- , MD5(string) 400
- , MID(x, y, z) 508
- , MIN() 173
- , MIN(Ausdruck) 509
- , MINUTE(Zeit) 338, 509
- , MOD(x, y) 509
- , MONTH(Datum) 334, 509
- , MONTHNAME(Datum) 333, 510
- , mysql_affected_rows() (PHP) 316
- , mysql_close() (PHP) 316
- , mysql_connect() (PHP) 316
- , mysql_create_db() (PHP) 316
- , mysql_data_seek() (PHP) 316
- , mysql_db_query() (PHP) 316
- , mysql_drop_db() (PHP) 316
- , mysql_errno() (PHP) 316
- , mysql_error() (PHP) 316
- , mysql_fetch_array() (PHP) 316
- , mysql_fetch_fields() (PHP) 316
- , mysql_fetch_length() (PHP) 316
- , mysql_fetch_object() (PHP) 316
- , mysql_fetch_rows() (PHP) 316
- , mysql_field_flags() (PHP) 316
- , mysql_field_len() (PHP) 316
- , mysql_field_name() (PHP) 316
- , mysql_field_seek() (PHP) 316
- , mysql_field_table() (PHP) 317
- , mysql_field_type() (PHP) 317
- , mysql_free_result() (PHP) 317
- , mysql_insert_id() (PHP) 317
- , mysql_list_dbs() (PHP) 317
- , mysql_list_fields() (PHP) 317
- , mysql_list_tables() (PHP) 317
- , mysql_num_fields() (PHP) 317
- , mysql_num_rows() (PHP) 317
- , mysql_pconnect() (PHP) 317
- , mysql_query() (PHP) 317
- , mysql_result() (PHP) 317
- , mysql_select_db() (PHP) 317
- , mysql_tablename() (PHP) 317

- , next() (PHP) 315
 - , NOT 188
 - , NOW() 336, 510
 - , numerische 183
 - , numerische, Addition 184
 - , numerische, Division 186
 - , numerische, MOD() 186
 - , numerische, Modulo 186
 - , numerische, Multiplikation 185
 - , numerische, Subtraktion 185
 - , OCT(x) 510
 - , OCTET_LENGTH() 510
 - , OR 188
 - , PASSWORD() 79
 - , PASSWORD(Passwort) 510
 - , PASSWORD(String) 397
 - , PERIOD_ADD(Periode, Monate) 341
 - , PERIOD_ADD(x, y) 511
 - , PERIOD_DIFF(Periode 1, Periode 2) 341
 - , PERIOD_DIFF(x, y) 511
 - , PI() 511
 - , POSITION(x, y) 511
 - , POW(x, y) 512
 - , POWER(x, y) 512
 - , prev() (PHP) 315
 - , QUARTER(Datum) 334, 512
 - , RADIANS(x) 512
 - , RAND() und RAND(Anlage) 513
 - , REPEAT(x, y) 513
 - , REPLACE() 192
 - , REPLACE(x, y, z) 514
 - , require() (PHP) 312
 - , REVERSE() 192
 - , REVERSE(x) 514
 - , RIGHT(String, x) 514
 - , ROUND(x) 514
 - , RPAD(x, y, z) 515
 - , RTRIM() 190
 - , RTRIM(x) 515
 - , SEC_TO_TIME(Sekunden) 339, 516
 - , SECOND(Zeit) 338, 516
 - , SESSION_USER() 397, 516
 - , SIGN(x) 516
 - , SIN(x) 517
 - , SOUNDEX(x) 517
 - , SPACE(x) 517
 - , SQRT(x) 517
 - , STD() 173
 - , STD(Ausdruck) oder STDDEV(Ausdruck) 518
 - , STRCMP(String1, String2) 518
 - , SUBDATE() 520
 - , SUBDATE(Datum, INTERVALL Ausdruck Typ) 341
 - , SUBSTRING() 191
 - , SUBSTRING_INDEX(x, y, z) 519
 - , SUM() 173
 - , SUM(Ausdruck) 519
 - , SYSDATE() 336, 520
 - , SYSTEM_USER() 520
 - , SYSTEMUSER() 397
 - , TAN(x) 520
 - , TIME_FORMAT(Zeit, Format) 337
 - , TIME_FORMAT(Zeit, Formatzeichen) 520
 - , TIME_TO_SEC(Zeit) 339, 521
 - , TO_DAYS(Datum) 339, 521
 - , TRIM() 190
 - , TRUNCATE(x, y) 522
 - , UCASE() 192
 - , UCASE(x) 522
 - , UPPER(x) 523
 - , USER() 397, 523
 - , VBScript 241
 - , VERSION() 397, 523
 - , WEEKDAY(Datum) 331, 524
 - , YEAR(Datum) 334, 525
 - , YEARWEEK(Datum[, Wochenstarttag]) 334
 - , Zeichenketten 189
- G**
- gelöschte Datensätze entfernen 167
 - Geschäfts-
 - , Ablauf, definieren 53
 - , Objekte, definieren 54
 - , Objekte, Meet_A_Geek 64
 - , Regeln 55, 66
 - Geschichte von MySQL 26
 - gespeicherte Prozeduren 373
 - GET-Methode 259
 - Gleichheitszeichen (=) 170, 306
 - Gleichungen, SELECT Anweisung 160
 - grafische Benutzeroberfläche (GUI) 31, 142
 - GRANT
 - , Anweisung 359f.



- , Befehl 359f., 479
- , Schlüsselwort 359
- , Syntax 359
- Grant_priv (Berechtigung, Privileg) 350
- Grant_priv, Spalte 350
- GRANT-Tabellen 348
- , Änderungen 356
- , Benutzer hinzufügen 357
- , Benutzer manuell hinzufügen 361
- , erstellen 40
- , manuell bearbeiten 362
- , neuladen 359
- GREATEST(x, y, ...)-Funktion 502
- Größe, Tabellen 108
- Groß- und Kleinschreibung
- , mysqladmin 77
- , MySQL-Monitor 73
- GROUP BY
- , Klausel 174, 214
- , reservierte Wörter 156
- GUI siehe grafische Benutzeroberfläche

H

- h (Parameter, mysqld) 420
- h Hostname, --host Hostname (Option, mysqlimport) 130, 392
- h, --host=Hostname (Option, mysqladmin) 402
- h=Hostname, --host=Hostname (Optionen, mysqldump) 387
- h=Hostname, --host=Hostname(Option, mysqldump) 147
- Handle
- , Anweisung, \$dbh 277
- , Anweisung, erstellen mit prepare() 279
- , Datenbank, \$dbh 277, 280
- , Datenbank, erstellen 276
- , Datenbank, Methode 276
- Hash, assoziatives Array 287
- herunterladbare Produkte, von einer MySQL-Webseite 231
- HEX(x)-Funktion 502
- HIGH_PRIORITY-Argument 161
- Hilfssystem, online 32
- Host (Berechtigung, Privileg) 349
- Host, Spalte 349, 351

- Hostname.n 394
- host-Tabelle 78, 348, 351
- HOURL(Zeit)-Funktion 338, 502
- HTML 129
- , <FORM>-Tag 259
- , <HEAD> 262
- , <input>-Tag 260
- HTML- und PHP-Code im Web-Browser 301
- Hypertext Markup Language siehe HTML

I

- i, --ignore (Option, mysqlimport) 131, 393
- i, --information (Option, isamchk) 409
- i, --information (Option, myisamchk) 425
- i, --sleep=x (Option, mysqladmin) 402
- IBM, SQL (Structured Query Language) 154
- IF NOT EXISTS, Parameter 99
- IF(Ausdruck1, Ausdruck2, Ausdruck3)-Funktionen 187, 503
- If...Then...Else-Anweisung 242
- If...Then...Else-Syntax 242
- if-Anweisung 308
- IFNULL()-Funktion 188
- IFNULL(Ausdruck1, Ausdruck2)-Funktion 503
- IIS 246
- Import
- , aus Access 136
- , Bulk Copy Protocol (BCP) 128
- , CSV-Datei 129, 131
- , Dateien 392
- , Daten 437
- , Daten, Datei-Konvertierungs-Vorgang 128
- , Datenbank-Schemata 108
- , mysqlimport-Programm 130, 392
- , Werkzeuge 144
- importsql, Datei 144
- Import-Werkzeuge für Datenbanken 144
- include()-Funktion (PHP) 312
- include-Verzeichnis 39
- INDEX (Berechtigung, Privileg) 359
- Index_priv (Berechtigung, Privileg) 350
- Index_priv, Spalte 350
- Indizes
- , erstellen 105
- , Gebrauch 104

- , Index löschen 106
- , löschen 106
- , Sinn 105
- Init DB (processlist-Befehl) 406
- inkrementelles Backup 394
- INNER JOIN 169
- INSERT Befehl
 - , Anweisung, Sperren 202
 - , reservierte Wörter 156
- INSERT INTO...SELECT-Befehl 163
- INSERT INTO-Befehl 162
- INSERT(x, y, z, j)-Funktion 504
- Insert_priv (Berechtigung, Privileg) 349
- Insert_priv, Spalte 349
- INSERT-Befehl 162, 481
 - , (Berechtigung, Privileg) 359
 - , Anweisung, DELAYED (verzögerte) 164
- Installation
 - , MyODBC, Dialogbox 231
 - , MyODBC, Treiber 231
 - , MySQL, Linux 37
 - , MySQL, Windows 38
 - , Perl, DBD 270, 272, 274
 - , Perl, DBI 270, 272
 - , PHP 300
 - , XML 129
- INSTALL-BINARY-Datei 39
- INSTR(x, y)-Funktion 504
- INT-Datentyp 111f.
- Interfaces siehe Schnittstellen
- interne Funktionen 179ff., 183, 193f.
- interne Zeiger, Arrays 315
- Internet 27
 - , Datenbankserver 29
 - , Thin Client 31
 - , Unternehmen 29
- Internet Information Server siehe IIS
- Internet Service Provider siehe ISP
- Interpreter
 - , Perl 278f., 287
 - , PHP 299
- INTO
 - , reservierte Wörter 156
 - , Schlüsselwort 162
- intrinsische Funktionen 79, 179

- isamchk 407
 - , #-debug-Optionen, --debug=debug_Optionen 409
 - , ?, --help 410
 - , -a, --analyze 409
 - , -d, --description 409
 - , -e, --extended-check 409
 - , -f, --force 409
 - , -i, --information 409
 - , -k=x, --keys-used=x 410
 - , -l, --no-symlinks 410
 - , -o, --safe_recover 410
 - , -q, --quick 410
 - , -r, --recover 410
 - , -R=index, --sort-records=index 410
 - , -s, --silent 410
 - , -S, --sort-index 410
 - , -u, --unpack 410
 - , -v, --verbose 410
 - , -V, --version 410
 - , -W, --wait 410
- ISNULL(Ausdruck)-Funktion 503
- ISP (Internet Service Provider) 26

J

- JOIN, reservierte Wörter 156
- Joins 167
 - , ANSI 92 167
 - , CROSS JOIN 169
 - , INNER JOIN 169
 - , Karthesischer Join 159, 169
 - , LEFT JOIN 170
 - , NATURAL LEFT JOIN 171
 - , RIGHT JOIN 176
 - , SQL (Structured Query Language) 167
 - , Tabellen 167ff., 176, 214

K

- k=x, --keys-used=x (Option, isamchk) 410
- Karthesischer Join 159, 169
- Kaufmanns-Und (&) 42
- Keyset (CursorType-Parameter) 257
- Kill (processlist-Befehl) 407
- kill id1, id2, Befehl 402
- KILL-Befehl 481



- kill-Befehl 43
- Klammern
 - , () 184
 - , eckige 306, 474
 - , geschweifte 311
- Klauseln
 - , GROUP BY-Klausel, Aggregat-Funktionen 174
 - , GROUP BY-Klausel, Schlüssel 214
 - , ORDER BY-Klausel 175
 - , ORDER BY-Klausel, Schlüssel 214
 - , WHERE-Klausel 157, 159
 - , WHERE-Klausel, Aggregat-Funktionen 174
 - , WHERE-Klausel, Funktionen 187
 - , WHERE-Klausel, Optimierung 423
 - , WHERE-Klausel, Schlüssel 213
- Komma (,) 186
- Kommentieren des Programmcodes (VB-Script) 243
- Komponenten von SQL 155
- konditionale
 - , Anweisungen 423
 - , Funktionen 187f.
- Konfiguration, Anzeige 417
- Konkatenation von Strings 160
- Kontrollen, Sicherheit 352
- Kontrollstrukturen (PHP)
 - , for 310
 - , if 308
 - , include 312
 - , require 312
 - , while 309
- Kopie, einer Tabelle erstellen 99
- L**
 - L, –local (Option, mysqlimport) 130
 - l, –lock-tables (Option, mysqldump) 147, 387
 - l, –lock-tables (Option, mysqlimport) 130, 392
 - l, –no-symlinks (Option, isamchk) 410
 - LAN (Local Area Network) 30
 - LAST_INSERT_ID()-Funktion 114
 - LAST_INSERT_ID()-Funktion 504
 - Layer siehe Ebenen
 - LCASE(x)-Funktion 504
 - LEAST(x, y, z, ...)-Funktion 505
 - LEFT JOIN 170
 - , reservierte Wörter 156
 - LEFT(x, y)-Funktion 505
 - LENGTH(x)-Funktion 506
 - lib-Verzeichnis 39
 - LIKE
 - , Befehl 423
 - , reservierte Wörter 156
 - , Schlüsselwort 423
 - LIMIT
 - , Anweisung 157, 166
 - , reservierte Wörter 156
 - , Schlüsselwort 157, 165f.
 - lines-terminated-by=Trennzeichen (Option, mysqldump) 148, 388
 - lines-terminated-by=Trennzeichen (Option, mysqlimport) 131, 148, 393
 - Link, symbolisch 73
 - Linux
 - , Entwicklung 26
 - , MySQL, Installation 417
 - , MySQL, Performance 417
 - , MySQL, Installation 37
 - , Perl, Installation 272
 - list()-Funktion (PHP) 315
 - Lizensierung
 - , MySQL 36
 - , Regeln 36
 - LOAD DATA INFILE-Befehl 134f., 224, 392, 437, 481
 - LOAD_FILE(Dateiname)-Funktion 506
 - Local Area Network siehe LAN
 - localhost 78, 80
 - LOCATE()-Funktion 191
 - LOCATE(x, y, z)-Funktion 506
 - Lock (CursorType-Parameter) 257
 - LOCK TABLES-Befehl 482
 - , Syntax 199
 - Locks siehe Sperren
 - Löschen
 - , Datenbanken, DROP-Befehl 75
 - , Datenbanken, mysqladmin 75, 77

- , Indizes 106
- , Schlüssel 215
- , Spalten 103
- , Tabellen 103
- LOG(x)-Funktion 507
- LOG10(x)-Funktion 507
- Logging siehe Protokollierung
- Logik
 - , Datenbank-Normalisierung 84
 - , Programmierlogik, Berichte erstellen 183
 - , Programmierlogik, Platz 180
 - , Programmierlogik, Three-Tier 181
- logische Funktionen 187
- log-long-format=Dateiname (Parameter, mysqld) 420
- Logs siehe Protokollierung
- log-update=Dateiname (Parameter, mysqld) 420
- LOB-Datentyp 115
- LONGTEXT-Datentyp 115
- LOW_PRIORITY WRITE-Sperren 201
- LOWER(x)-Funktion 507
- low-priority-insert(Parameter, mysqld) 420
- LPAD(x, y, z)-Funktion 508
- LTRIM()-Funktion 190
- LTRIM(x)-Funktion 508

M

- m:n-Beziehungen 60
- Management, Datenbanken 401
- Management-Systeme (Datenbanken) 26
- manual.htm-Datei 40
- manual.txt-Datei 40
- manual_toc.htm-Datei 40
- Master-Slave, Datenbanken 391
- mathematische Funktionen 183
 - , Addition 184
 - , Division 186
 - , mathematische Regeln 186
 - , Modulo 186
 - , Multiplikation 185
 - , Subtraktion 185
- Max memory used, mysqladmin-Statusbericht 405
- MAX()-Funktion 172, 214
- MAX(Ausdruck)-Funktion 508
- MD5(string)-Funktion 400
- MEDIUMBLOB-Datentyp 115
- MEDIUMINT-Datentyp 111f.
- MEDIUMTEXT-Datentyp 115
- Meet_A_Geek-Datenbank 62, 258, 430
 - , Ads-Tabelle 434
 - , Benutzerrechte 438
 - , Beziehungen 67
 - , Customers, Erstellen der Webseite 439
 - , Customers, Erstellen des Scripts 444
 - , Customers-Tabelle 433
 - , DSN erstellen 235
 - , Entwicklungsprozess 469
 - , Erstellung der Datenbank 436
 - , Erstellung, Anforderungs-Phase 431
 - , Erstellung, Datenmodellierung 432
 - , Erstellung, Normalisierung der Datenbank 434
 - , Erstellung, Planungs-Phase 431
 - , Flussdiagramm 432
 - , Geschäftsablauf 62
 - , Geschäftsobjekte 62
 - , Geschäftsregeln 66
 - , Import von Daten 437
 - , Modell 435
 - , Orders-Tabelle 434
 - , Products-Tabelle 434
 - , Sicherheitsaspekte 438
 - , System-DSN erstellen 233
 - , Testen 468
 - , Webseite, Ausgabe der Daten 263, 266
 - , Webseite, Benutzer hinzufügen 439
 - , Webseite, Benutzer hinzufügen (PerlScript) 444
 - , Webseite, Benutzer-Login 449
 - , Webseite, Erstellen 439
 - , Webseite, Login.pl 451
 - , Webseite, postAd.asp 258ff.
 - , Webseite, PostAd.html 456
 - , Webseite, searchAds.pl 457, 459
 - , Webseite, Such-Formular 456
- Memory in use, mysqladmin-Statusbericht 405
- Merkmale von MySQL 32f.
- Methoden
 - , AddNew() 254



- , Anweisungs-Handle 279
- , Clone() 254
- , Close() 254
- , Command-Objekt 252
- , Connection-Objekt 249
- , CreateObject 246
- , data_sources 276
- , disconnect 277
- , do 277, 280
- , Execute() 253
- , execute() 279
- , Export 145
- , fetchrow_hashref() 279
- , finish() 279
- , GET 259
- , Import 128
- , Open() 255
- , POST 259
- , prepare 277
- , prepare() 279
- , Recordset-Objekt 254
- , rows() 279
- , Verbindung, DBI 276
- , Write 248
- Microsoft Access siehe Access
- MID(x, y, z)-Funktion 508
- Middle Tier-Design 181
- MIN()-Funktion 173, 214
- MIN(Ausdruck)-Funktion 509
- MINUTE(Zeit)-Funktion 338, 509
- Mirror Sites (MySQL-Download) 37
- MOD(x, y)-Funktion 509
- Modell, Meet_A_Geek-Datenbank 435
- Modellierung einer Datenbank 55
- Module, Perl 271
- Modulo
 - , Funktion 186
 - , Operator 308
- MONTH(Datum)-Funktion 334, 509
- MONTHNAME(Datum)-Funktion 333, 510
- Multiplikation
 - , Funktion 185
 - , Operator 308
- Multi-Threaded Server 32
- Multi-Thread-System 417
- myasamchk 407
- myisamchk 424
 - , -a, -analyze 425
 - , -d, -description 425
 - , -f, -force 425
 - , -i, -information 425
 - , -o, -safe-recovery 425
 - , -q, -quick 425
 - , -r, -recover 425
 - , -S, -sort-index 425
 - , -u, -unpack 425
 - , -v 425
 - , -w 425
- myisampack 426
- MyODBC 230
 - , Installation 231f.
 - , ODBC 230
 - , TcX MySQL-Treiber 236
 - , Treiber 230
- MySQL
 - , Befehle 81
 - , Dateien 39
 - , Datenbanken 47, 78, 348, 396
 - , Download 36
 - , fehlende Merkmale 383
 - , Geschichte 26
 - , Installation, Linux 37
 - , Installation, Windows 38
 - , Kommandozeile 44
 - , Linux 42
 - , Lizenzierung 36
 - , Merkmale 32
 - , RDBMS 81
 - , Server 42, 404
 - , SQL (Structured Query Language) 27
 - , verfügbare Schnittstellen 227
 - , Verzeichnis 96
 - , Webseite 36, 231
 - , Windows 43
- mysql.server-Script 43
- mysql_affected_rows()-Funktion (PHP) 316
- mysql_bench-Verzeichnis 40
- mysql_change_user()-Funktion (PHP) 316
- mysql_close()-Funktion (PHP) 316
- mysql_connect()-Funktion (PHP) 316
- mysql_create_db()-Funktion (PHP) 316
- mysql_data_seek()-Funktion (PHP) 316

- mysql_db_query()-Funktion (PHP) 316
- mysql_drop_db()-Funktion (PHP) 316
- mysql_erro()-Funktion (PHP) 316
- mysql_error()-Funktion (PHP) 316
- mysql_fetch_array()-Funktion (PHP) 316
- mysql_fetch_fields()-Funktion (PHP) 316
- mysql_fetch_length()-Funktion (PHP) 316
- mysql_fetch_object()-Funktion (PHP) 316
- mysql_fetch_rows()-Funktion (PHP) 316
- mysql_field_flags()-Funktion (PHP) 316
- mysql_field_len()-Funktion (PHP) 316
- mysql_field_name()-Funktion (PHP) 316
- mysql_field_seek()-Funktion (PHP) 316
- mysql_field_table()-Funktion (PHP) 317
- mysql_field_type()-Funktion (PHP) 317
- mysql_for_dummies-Datei 40
- mysql_free_result()-Funktion (PHP) 317
- mysql_insert_id()-Funktion (PHP) 317
- mysql_list_dbs()-Funktion (PHP) 317
- mysql_list_fields()-Funktion (PHP) 317
- mysql_list_tables()-Funktion (PHP) 317
- mysql_num_fields()-Funktion (PHP) 317
- mysql_num_rows()-Funktion (PHP) 317
- mysql_pconnect()-Funktion (PHP) 317
- mysql_query()-Funktion (PHP) 317
- mysql_result()-Funktion (PHP) 317
- mysql_select_db()-Funktion (PHP) 317
- mysql_setpermissions 362
- mysql_tablename()-Funktion (PHP) 317
- mysqladmin 402
 - , #, --debug= 402
 - , ?, --help 403
 - , Befehl 76
 - , -C, --compress 402
 - , create Datenbankname 401, 403
 - , Datenbank erstellen 436
 - , drop Datenbankname 401, 403
 - , extended-status 401, 404
 - , -f, --force 402
 - , flush-hosts 401
 - , flush-logs 401
 - , flush-privileges 401
 - , flush-tables 401
 - , -h, --host=Hostname 402
 - , -i, --sleep=x 402
 - , kill id1, id2 402
 - , Optionen 401
 - , -p, --password[=Passwort] 402
 - , -P, --port[=Portnummer] 402
 - , password neues_Passwort 402
 - , Ping 402
 - , processlist 401, 406
 - , Refresh 402
 - , Reload 402
 - , -s, --silent 402
 - , -S, --socket=Socket 402
 - , shutdown 43, 402
 - , status 401, 404
 - , Syntax 401
 - , -t, --timeout=x 402
 - , -u, --user=Benutzername 403
 - , -V, --version 403
 - , variables 401, 404
 - , version 401, 404
 - , -w, --wait[=x] 403
- mysql-Befehl (MySQL-Monitor) 75
- mysqld
 - , -b, --base-dir 420
 - , Befehl 42
 - , --big-tables 420
 - , Daemon 42
 - , --datadir=vollständiger Pfad 420
 - , --enable-locking 420
 - , -h 420
 - , --log-long-format=Dateiname 420
 - , --log-update=Dateiname 420
 - , --low-priority-insert 420
 - , --skip-grant-tables 421
 - , --skip-name-resolve 421
 - , -V, --version 421
- mysqld.exe, Datei 43
- MySQL-DBD (Datenbank-Treiber) 270, 274
- mysqldump 146, 396
 - , ?, --help 148, 389
 - , --add-drop-table 146, 387
 - , --add-locks 146, 387
 - , -c, --complete_insert 146
 - , -d, --no-data 147, 387
 - , Dateien speichern 386
 - , --delayed_insert 146
 - , -e, --extended-insert 146, 387
 - , Errcode 13 oder 2 389



- , -F, --flush-logs 147, 387
 - , -f, --force 147
 - , --fields-enclosed-by=Trennzeichen 147, 388
 - , --fields-escaped-by=Escape-Zeichen 148, 388
 - , --fields-optionally-enclosed-by=Trennzeichen 147, 388
 - , --fields-terminated-by=Trennzeichen 147, 388
 - , --full 147
 - , -h=Hostname, --host=Hostname 147, 387
 - , -l, --lock-tables 147, 387
 - , --lines-terminated-by=Trennzeichen 148, 388
 - , Methoden 145
 - , --opt 146, 387
 - , Optionen 146, 387
 - , -p[Passwort], --password[=Passwort] 148, 388
 - , -q, --quick 146, 387
 - , -t, --no-create-info 147, 387
 - , --tab=Pfad, -T=Pfad 147, 388
 - , -u Benutzername, --user=Benutzername 148, 388
 - , -w=Bedingung, --where=Bedingung 148, 389
 - mysqlimport 130, 392
 - , --?, --help 131, 393
 - , -d, --delete 130, 392
 - , Daten, Import oder Wiederherstellen 392
 - , -f, --force 130
 - , --fields-enclosed-by=Trennzeichen 131, 148, 393
 - , --fields-escaped-by=Escape-Zeichen 131, 148, 393
 - , --fields-optionally-enclosed-by=Trennzeichen 131, 148, 393
 - , --fields-terminated-by=Trennzeichen 131, 148, 393
 - , -h Hostname, --host Hostname 130, 392
 - , -i, --ignore 131, 393
 - , Import im Batchmodus 132
 - , Import von CSV-Dateien 131
 - , -L, --local 130
 - , -l, --lock-tables 130, 392
 - , --lines-terminated-by=Trennzeichen' 131, 148, 393
 - , Optionen 130, 148
 - , -p [Passwort], --password[=Passwort] 130, 392
 - , -r, --replace 131, 393
 - , -s, --silent 131, 393
 - , Syntax 392
 - , -u Benutzername, --user Benutzername 130, 392
 - MySQL-Monitor 155
 - , Befehle 73
 - , Starten 96
 - mysql-Verzeichnis 44
- ## N
- NATURAL LEFT JOIN 171
 - neun Grade der Normalisierung 86
 - new-Anweisung, Objekte 303
 - next()-Funktion (PHP) 315
 - nicht-eindeutiger Schlüssel 217
 - Nicht-Schlüsseldaten beseitigen 89
 - Normalformen
 - , 1. Normalform (1.NF) 86
 - , 2. Normalform (2.NF) 88
 - , 3. Normalform (3.NF) 89
 - , 4. Normalform(4NF) 91
 - , 5. Normalform(5NF) 91
 - , Boyce-Codd-Normalform 86, 91
 - , Domain Key 86
 - , Domain Key Normalform 91
 - , Join Projection 86
 - , Over-Strong Join Projection 86
 - , Over-Strong Join Projection Normalform 91
 - , Strong Join Projection 86
 - , Strong Join Projection Normalform 91
 - Normalisierung 84
 - , Customers-Datenbank 86
 - , Grenzen 90
 - , Logik 84
 - , Meet_A_Geek-Datenbank 434
 - , neun Grade 86
 - , Tabellen, Nicht-Schlüsseldaten 89

- , Tabellen, Spalten 84
- , Tabellen, Teilabhängigkeiten 88
- , wiederholende Daten beseitigen 86
- NOT NULL, Spaltenattribut 119
- NOT-Funktion 188
- NOW()-Funktion 336, 510
- NULL, Spaltenattribut 119
- numerische Datentypen
- , AUTO_INCREMENT 113, 119
- , BINARY 119
- , DEFAULT 119
- , NOT NULL 119
- , NULL 119
- , PRIMARY KEY 119
- , UNIQUE 119
- , UNSIGNED 119
- , ZEROFILL 115, 119
- numerische Funktionen 183
- , Addition 184
- , Division 186
- , MOD() 186
- , Modulo 186
- , Multiplikation 185
- , Subtraktion 185

O

- o, –safe_recover (Option, isamchk) 410
- o, –safe_recovery (Option, myisamchk) 425
- Objekte
- , Command-Objekt 252
- , Command-Objekt, Eigenschaften 252
- , Command-Objekt, Methoden 252
- , Command-Objekt, Verwendung 253
- , Connection-Objekt 249
- , Connection-Objekt, Eigenschaften 249
- , Connection-Objekt, Methoden 249
- , DAO (Data Access Objekt) 237
- , Geschäfts-, definieren 54
- , Geschäfts-, Meet_A_Geek 62
- , new-Anweisung 303
- , Perl DBI 276
- , RDO (Remote Data Objects) 237
- , Recordset-Objekt 254
- , Recordset-Objekt, CursorType 254
- , Recordset-Objekt, Eigenschaften 254
- , Recordset-Objekt, Methoden 254
- , Recordset-Objekt, Source 255
- , Recordset-Objekt, Verwendung 256
- , Request-Objekt 247
- , Response-Objekt 248
- , Server-Objekt 247
- OCT(x)-Funktion 510
- OCTET_LENGTH()-Funktion 510
- ODBC 230
- , ADO (ActiveX Data Objects) 237
- , ADO (ActiveX Data Objects), Command-Objekt 252
- , ADO (ActiveX Data Objects), Connection-Objekt 249
- , ADO (ActiveX Data Objects), Meet_A_Geek-Datenbank 258
- , ADO (ActiveX Data Objects), Performance 268
- , ADO (ActiveX Data Objects), Recordset-Objekt 254
- , ASP (Active Server Pages) 238, 241, 246
- , DAO (Data Access Objects) 237
- , Direct 237
- , DSN (Data Source Name) 233
- , MyODBC, Installation 229f., 232
- , ODBCDirect 237
- , RDO (Remote Data Objects) 237
- , Treiber 230
- , VBScript 237, 240
- , VBScript, Active Server Page (ASP) 240
- , VBScript, Funktionen 241
- , VBScript, Generieren einer Webseite 243
- , VBScript, Kommentieren des Programm-codes 243
- , VBScript, Programmfluss 242
- , VBScript, Schleifenkontrolle 242
- , VBScript, Variant 241
- , Zip-Datei 231
- ON
- , reservierte Wörter 156
- , Schlüsselwort 360f., 364
- Open Database Connectivity siehe ODBC
- Open Source
- , Bewegung 26
- , Software 370
- Open tables, mysqladmin-Statusbericht 404
- open-Befehl 251



- Opens, mysqladmin-Statusbericht 404
- Operatoren
 - , AND 188
 - , arithmetische 307
 - , NOT 188
 - , OR 188
- opt (Optionen, mysqldump) 146, 387
- Optimierung, Übersicht 426
- optimistic locking 257
- OPTIMIZE Befehl 167, 224, 428, 469
- OPTIMIZE TABLE Befehl 482
- Optionen
 - , #, --debug= (mysqladmin) 402
 - , #=debug-Optionen, --debug=debug_Optionen (isamchk) 409
 - , ?, --help (isamchk) 410
 - , ?, --help (mysqladmin) 403
 - , ?, --help (mysqldump) 148, 389
 - , ?, --help (mysqlimport) 131, 393
 - , -a, --analyze (isamchk) 409
 - , -a, --analyze (myisamchk) 425
 - , --add-drop-table (mysqldump) 146, 387
 - , --add-locks (mysqldump) 146, 387
 - , -c, --complete_insert (mysqldump) 146
 - , -C, --compress (mysqladmin) 402
 - , -d, --delete (mysqlimport) 130, 392
 - , -d, --description (isamchk) 409
 - , -d, --description (myisamchk) 425
 - , -d, --no-data (mysqldump) 147, 387
 - , --delayed_insert (mysqldump) 146
 - , -e, --extended-check (isamchk) 409
 - , -e, --extended-insert (mysqldump) 146, 387
 - , -F, --flush-logs (mysqldump) 147, 387
 - , -f, --force (isamchk) 409
 - , -f, --force (myisamchk) 425
 - , -f, --force (mysqladmin) 402
 - , -f, --force (mysqldump) 147
 - , -f, --force (mysqlimport) 130
 - , --fields-enclosed-by=Trennzeichen (mysqldump) 147, 388
 - , --fields-enclosed-by=Trennzeichen (mysqlimport) 131, 148, 393
 - , --fields-escaped-by=Escape-Zeichen (mysqldump) 148, 388
 - , --fields-escaped-by=Escape-Zeichen (mysqlimport) 131, 148, 393
 - , --fields-optionally-enclosed-by=Trennzeichen (mysqldump) 147, 388
 - , --fields-optionally-enclosed-by=Trennzeichen (mysqlimport) 131, 148, 393
 - , --fields-terminated-by=Trennzeichen (mysqldump) 147, 388
 - , --fields-terminated-by=Trennzeichen (mysqlimport) 131, 148, 393
 - , --full (mysqldump) 147
 - , -h Hostname, --host Hostname (mysqlimport) 130, 392
 - , -h, --host=Hostname (mysqladmin) 402
 - , -h=Hostname, --host=Hostname (mysqldump) 147, 387
 - , -i, --ignore (mysqlimport) 131, 393
 - , -i, --information (isamchk) 409
 - , -i, --information (myisamchk) 425
 - , -i, --sleep=x (mysqladmin) 402
 - , -k=x, --keys-used=x (isamchk) 410
 - , -L, --local (mysqlimport) 130
 - , -l, --lock-tables (mysqldump) 147, 387
 - , -l, --lock-tables (mysqlimport) 130, 392
 - , -l, --no-symlinks (isamchk) 410
 - , --lines-terminated-by=Trennzeichen (mysqldump) 148, 388
 - , --lines-terminated-by=Trennzeichen (mysqlimport) 131, 148, 393
 - , -o, --safe_recover (isamchk) 410
 - , -o, --safe-recovery (myisamchk) 425
 - , --opt (mysqldump) 146, 387
 - , -p, --password[=Passwort] (mysqladmin) 402
 - , -P, --port[=Portnummer] (mysqladmin) 402
 - , -q, --quick (isamchk) 410
 - , -q, --quick (myisamchk) 425
 - , -q, --quick (mysqldump) 146, 387
 - , -r, --recover (isamchk) 410
 - , -r, --recover (myisamchk) 425
 - , -r, --replace (mysqlimport) 131, 393
 - , -R=index, --sort-records=index (isamchk) 410
 - , -s, --silent (isamchk) 410
 - , -s, --silent (mysqladmin) 402
 - , -s, --silent (mysqlimport) 131, 393
 - , -S, --socket=Socket (mysqladmin) 402
 - , -S, --sort-index (isamchk) 410
 - , -S, --sort-index (myisamchk) 425

- , -t, --no-create-info (mysqldump) 147, 387
- , -t, --timeout=x (mysqladmin) 402
- , --tab=Pfad, -T=Pfad (mysqldump) 147, 388
- , -u Benutzername, --user Benutzername (mysqlimport) 130, 392
- , -u Benutzername, --user=Benutzername (mysqldump) 148, 388
- , -u, --unpack (isamchk) 410
- , -u, --unpack (myisamchk) 425
- , -u, --user=Benutzername (mysqladmin) 403
- , -v (myisamchk) 425
- , -v, --verbose (isamchk) 410
- , -V, --version (isamchk) 410
- , -V, --version (mysqladmin) 403
- , -w (myisamchk) 425
- , -W, --wait (isamchk) 410
- , -w, --wait[=x] (mysqladmin) 403
- , -w=Bedingung, --where=Bedingung (mysqldump) 148, 389
- OR
- , Funktion 188
- , reservierte Wörter 156
- Oracle, SQL (Structured Query Language) 154
- ORDER BY
- , Klausel 174, 214
- , reservierte Wörter 156
- Orders-Tabelle, Meet_A_Geek-Datenbank 434
- P**
- p [Passwort], -password[=Passwort] (Option, mysqlimport) 130, 392
- p, -password[=Passwort] (Option, mysqladmin) 402
- P, --port[=Portnummer] (Option, mysqladmin) 402
- p[Passwort], -password[=Passwort] (Option, mysqldump) 388
- p[Passwort], -password[=Passwort](Option, mysqldump) 148
- Paradigma, Client/Server siehe Spalten, Indizes, Meet_A_Geek-Datenbank, Tabellen
- Parameter
- , -b, --base-dir (mysqld) 420
- , --big-tables (mysqld) 420
- , --datadir=vollständiger Pfad (mysqld) 420
- , --enable-locking (mysqld) 420
- , -h (mysqld) 420
- , IF NOT EXISTS 99
- , -l, --log=[Dateiname] (mysqld) 420
- , --log-long-format=Dateiname (mysqld) 420
- , --log-update=Dateiname (mysqld) 420
- , --low-priority-insert (mysqld) 420
- , --skip-grant-tables (mysqld) 421
- , --skip-name-resolve (mysqld) 421
- , -V, --version (mysqld) 421
- password (Berechtigung, Privileg) 349
- password neues_Passwort Befehl 402
- PASSWORD()-Funktion 79
- PASSWORD(Passwort)-Funktion 510
- PASSWORD(String)-Funktion 397
- password, Spalte 349
- Passwort 40
- , ändern 40
- , root 41, 72, 96
- , Verbindungsanforderungen 223
- , vergessen 48
- Performance
- , Festplatte 417
- , kompilieren und komprimieren 426
- , Linux 417
- , Multithreaded 417
- , mysqld-Daemon 420
- , Optimierung 427
- , Speicher 417
- , Struktur der SQL-Anweisungen 421
- , System-Variablen 417
- , Tuning 416
- PERIOD_ADD(Periode, Monate)-Funktion 341
- PERIOD_ADD(x, y)-Funktion 511
- PERIOD_DIFF(Periode 1, Periode 2)-Funktion 341
- PERIOD_DIFF(x, y)-Funktion 511
- Perl
- , CGI (Common Gateway Interface) 287, 290
- , CPAN (Comprehensive Perl Archive Network) 270
- , DBD 270



- , DBI 269
- , DBI, CD-ROM 272
- , Download 271
- , Installation 270
- , Interpreter 278
- , Module 271
- , Objekt 276
- , Script 278
- , Verbindung zu Datenbank 276
- , Webseite, FAQs 271
- Perl Package Manager (PPM) 271
- pessimistic locking 257
- PHP 297
 - , Arbeitsweise 301
 - , Arrays 314
 - , Ausdrücke 306
 - , CGI (Common Gateway Interface) 298
 - , Datenbanken, Abfragen 320
 - , Datenbanken, Anfragen 320
 - , Datenbanken, Erstellen 318
 - , Download 300
 - , echo() 301
 - , for-Schleife 310
 - , for-Schleife, break 311
 - , for-Schleife, continue 311
 - , Funktionen 313
 - , if-Anweisung 308
 - , include()-Funktion 312
 - , Installation 300, 322
 - , Kontrollstrukturen, for 310
 - , Kontrollstrukturen, if 308
 - , Kontrollstrukturen, include() 312
 - , Kontrollstrukturen, require() 312
 - , Kontrollstrukturen, while 309
 - , mysql_affected_rows() 316
 - , mysql_close() 316
 - , mysql_connect() 316
 - , mysql_create_db() 316
 - , mysql_data_seek() 316
 - , mysql_db_query() 316
 - , mysql_drop_db() 316
 - , mysql_errno() 316
 - , mysql_error() 316
 - , mysql_fetch_array() 316
 - , mysql_fetch_fields() 316
 - , mysql_fetch_length() 316
 - , mysql_fetch_object() 316
 - , mysql_fetch_rows() 316
 - , mysql_field_flags() 316
 - , mysql_field_len() 316
 - , mysql_field_name() 316
 - , mysql_field_seek() 316
 - , mysql_field_table() 317
 - , mysql_field_type() 317
 - , mysql_free_result() 317
 - , mysql_insert_id() 317
 - , mysql_list_dbs() 317
 - , mysql_list_fields() 317
 - , mysql_list_tables() 317
 - , mysql_num_fields() 317
 - , mysql_num_rows() 317
 - , mysql_pconnect() 317
 - , mysql_query() 317
 - , mysql_result() 317
 - , mysql_select_db() 317
 - , mysql_tablename() 317
 - , Operatoren, arithmetische 307
 - , phpinfo()-Funktion 304
 - , require()-Funktion 312
 - , serverseitig 298
 - , Syntax 302
 - , Variablen 304
 - , Variablen, aus HTML-Formularen 305
 - , Variablen, Datentypen 303
 - , Variablen, Zuweisen von Werten 306
 - , Verbindung zu Datenbank 317
 - , Vorteile 323
 - , Webseiten 300, 322
 - , while-Schleifen 309
 - , while-Schleifen, break 311
 - , while-Schleifen, continue 311
- phpinfo() 304
- PI()-Funktion 511
- Ping-Befehl 402
- Pipe-Symbole (| |) 189
- Plattformen
 - , MySQL kompilieren 426
 - , MySQL, Binärdistribution 36
 - , MySQL, Sourcedistribution 36
 - , Portierungen 32
- Portierungen, MySQL 32
- POSITION(x, y)-Funktion 511

- PostAd.asp 259
 - POST-Methode 259
 - POW(x, y)-Funktion 512
 - POWER(x, y)-Funktion 512
 - PPM siehe Perl Package Manager
 - Präfix, Teilschlüssel 208
 - prepare() 279
 - prev()-Funktion (PHP) 315
 - Primärschlüssel 211, 215
 - , mehrere Spalten 211
 - , synthetische 213
 - , zusammengesetzte Spalten 211
 - PRIMARY KEY
 - , löschen 106
 - , Spaltenattribut 119
 - , Syntax 205
 - print-Befehl (Perl) 280
 - Priorität, Dateien 391
 - Privilegien
 - , ALL 359
 - , ALTER 359
 - , Alter_priv 350
 - , CREATE 359
 - , Create_priv 349
 - , DELETE 350, 359
 - , Delete_priv 349
 - , DROP 359
 - , Drop_priv 349
 - , FILE 359
 - , File_priv 350
 - , Grant_priv 350
 - , Host 349
 - , INDEX 359
 - , Index_priv 350
 - , INSERT 359
 - , Insert_priv 349
 - , password 349
 - , PROCESS 360
 - , Process_priv 350
 - , REFERENCES 360
 - , References_priv 350
 - , RELOAD 360
 - , Reload_priv 349
 - , SELECT 360
 - , Select_priv 349
 - , SHUTDOWN 360
 - , Shutdown_priv 349
 - , UPDATE 360
 - , Update_priv 349
 - , USAGE 360
 - , User 349
 - PROCESS (Berechtigung, Privileg) 360
 - Process_priv (Berechtigung, Privileg) 350
 - Process_priv, Spalte 350
 - Processes (processlist-Befehl) 406
 - processlist-Befehl 401
 - processlist-Befehl (mysqldadmin) 406
 - Products-Tabelle, Meet_A_Geek-Datenbank 434
 - Produkte, zum Herunterladen von einer My SQL Webseite 231
 - Programmcode, PostAd.asp 259
 - Programmfluss, VBScript 242
 - Programmieren, Server-Ebene 180
 - Programmierlogik
 - , Client-Server 182
 - , Middle-Tier Design 181
 - , Ort 180
 - , Three-Tier Design 180
 - Protokollierung
 - , Aktualisierung 394
 - , Arten 395
 - , Backups 394
 - , Transaktionen 394
 - Prozentzeichen (%) 357
 - PUBLIC-Datei 39
- ## Q
- q, –quick (Option, isamchk) 410
 - q, –quick (Option, myisamchk) 425
 - q, –quick (Optionen, mysqldump) 387
 - q, –quick (Option, mysqldump) 146
 - QUARTER(Datum)-Funktion 334, 512
 - Quellcode-Versionen, MySQL 36
 - Queries per second avg, mysqldadmin-Statusbericht 405
 - Queries siehe Anfragen
 - Query (processlist-Befehl) 406
 - QueryString, Eigenschaft 247
 - Questions, mysqldadmin-Statusbericht 404
 - Quit (processlist-Befehl) 407



R

- r, --recover (Option, isamchk) 410
- r, --recover (Option, myisamchk) 425
- r, --replace (Option, mysqlimport) 131, 393
- R=index, --sort-records=index (Option, isamchk) 410
- RADIANS(x)-Funktion 512
- RAND()- und RAND(Anlage)-Funktion 513
- RDBMS 110
- RDO (Remote Data Objects) 237
- Read Only-Sperren 257
- README-Datei 39
- READ-Sperre 200
 - , Regeln, Befehle 200
- Recordset-Objekt
 - , AddNew() (ADO) 254
 - , BOF, EOF (ADO) 254
 - , Clone() (ADO) 254
 - , Close() (ADO) 254
 - , CursorType (ADO) 254
 - , Eigenschaften 254
 - , Fields (ADO) 254
 - , Find (Criteria, Skip, Dir, Start) (ADO) 255
 - , Format() (ADO) 255
 - , LockType (ADO) 255
 - , MaxRecords (ADO) 255
 - , Methoden 254
 - , Move() (ADO) 255
 - , MoveFirst() (ADO) 255
 - , MoveLast() (ADO) 255
 - , MoveNext() (ADO) 255
 - , MovePrevious() (ADO) 255
 - , Open(Source, ActiveConnection, CursorType, LockType) (ADO) 255
 - , RecordCount (ADO) 255
 - , Sort (ADO) 255
 - , Source (ADO) 255
 - , State (ADO) 255
 - , Status (ADO) 255
 - , Update (ADO) 255
 - , Verwendung 256
- REFERENCES (Berechtigung, Privileg) 360
- References_priv (Berechtigung, Privileg) 350
- References_priv, Spalte 350
- Refresh (processlist-Befehl) 407
- Refresh-Befehl 402

Regeln

- , Geschäfts-, definieren 55
- , Geschäfts-, Meet_A_Geek 66
- , READ-Sperren 200
- , WRITE-Sperren 200
- relationale Datenbanken 154
 - , Codd, E.F. 30
 - , Definition 30
- Relationale Datenbank-Management-Systeme
 - siehe RDBMS
- Relationen siehe Beziehungen
- RELOAD (Berechtigung, Privileg) 360
- Reload (processlist-Befehl) 407
- Reload_priv (Berechtigung, Privileg) 349
- Reload_priv, Spalte 349
- Reload-Befehl 402
- Remote Data Objects siehe RDO
- Reparatur
 - , Datenbanken 407
 - , Datenstruktur 407
 - , Festplattenplatz wiedergewinnen 408
 - , Schlüssel (isamchk) 411
 - , Tabellen 409
- REPEAT(x, y)-Funktion 513
- REPLACE()-Funktion 192
- REPLACE(x, y, z)-Funktion 514
- REPLACE-Befehl 176, 483
- Request-Objekt 247
- require()-Funktion (PHP) 312
- reservierte Wörter 156
- reservierte Wörter (MySQL)
 - , ALTER 156
 - , AND 156
 - , AS 156
 - , CREATE 156
 - , CROSS JOIN 156
 - , DELETE 156
 - , DROP 156
 - , FROM 156
 - , FULL JOIN 156
 - , GROUP BY 156
 - , INSERT 156
 - , INTO 156
 - , JOIN 156
 - , LEFT JOIN 156
 - , LIKE 156

- , LIMIT 156
- , ON 156
- , OR 156
- , ORDER BY 156
- , RIGHT JOIN 156
- , SELECT 156
- , SET 156
- , UPDATE 156
- , WHERE 156
- Response-Objekt 248
- REVERSE()-Funktion 192
- REVERSE(x)-Funktion 514
- REVOKE, Schlüsselwort 363
- REVOKE-Befehl 363, 483
- RIGHT JOIN, reservierte Wörter 156
- RIGHT(String, x)-Funktion 514
- ROLLBACK TRAN 372
- ROLLBACK-Befehl 202
- ROUND(x)-Funktion 514
- rows() 279
- RPAD(x, y, z)-Funktion 515
- RTRIM()-Funktion 190
- RTRIM(x)-Funktion 515

S

- s, --silent (Option, isamchk) 410
- s, --silent (Option, mysqladmin) 402
- s, --silent (Option, mysqlimport) 131, 393
- S, --socket=Socket (Option, mysqladmin) 402
- S, --sort-index (Option, isamchk) 410
- S, --sort-index (Option, myisamchk) 425
- safe_mysql &-Befehl 42
- SAVEPOINT-Befehl 202
- Schema für Datenbanken
 - , Customers-Tabelle 86
 - , importieren 107
 - , mit Script erstellen 100
- Schleifen
 - , PHP, break 311
 - , PHP, continue 311
 - , PHP, for 310
 - , PHP, for, Ausdrücke 310
 - , PHP, while 309
 - , PHP, while, Ausdruck 309
 - , VBScript 242
- Indizes
 - Schlüssel 204f.
 - , ALTER TABLE-Syntax 211
 - , AUTO_INCREMENT 215
 - , Befehle zur Erstellung 205
 - , Datenbank-Tabellen 203
 - , definieren 205
 - , eindeutige 209, 215
 - , Feld 57, 68
 - , Fremd- 211
 - , für einzelne Spalten 205
 - , für mehrere Spalte 207
 - , für mehrere Spalten 207
 - , Gebrauch 218
 - , GROUP BY-Klausel 214
 - , hinzufügen 87
 - , Löschen, Syntax 216
 - , MAX()-Funktion 214
 - , Min()-Funktion 214
 - , nicht-eindeutig 217
 - , Optimierung 424
 - , ORDER BY-Klauseln 214
 - , Präfix einer Spalte 208
 - , Primär- 211, 215
 - , Primär-, mehrere 211
 - , Primär-, Syntax 205
 - , PRIMARY KEY löschen, Syntax 106
 - , Prinzipien 203
 - , Reparatur mit isamchk 411
 - , sinnvolle Auswahl 213
 - , synthetische 212
 - , Tabellen-Joins 214
 - , Table-Scan 204
 - , Teil- 208
 - , Vorteile 204
 - , wann benutzen 213
 - , wann nicht benutzen 214
 - , WHERE-Klausel 213
 - , zusammengesetzte Teil- 208
- Schlüsselwörter
 - , GRANT 359
 - , AND 159
 - , AS 159
 - , ENCLOSED BY 135
 - , ESCAPED BY 135
 - , FROM 168, 364



- , INTO 162
- , JOIN 168
- , LIKE 423
- , LIMIT 165f.
- , ON 360f., 364
- , OR 159
- , REVOKE 363
- , TERMINATED BY 135
- , USING 170
- , VALUES 162
- Schlüssel, siehe auch Felder
- Schnelligkeit, MySQL, Datenbanken, -vergleich 370
- Schnittstellen 220, 226
 - , Applikationen 220ff.
 - , Datenbanken erstellen 227
 - , Datenzugriff 223
 - , Datenzugriff, Abfragen 224
 - , Datenzugriff, Anfragen 224
 - , Datenzugriff, verarbeiten 225
 - , Datenzugriff, Verbindung 223, 226
 - , MySQL, verfügbare 227
 - , Treiber 221
 - , Verbindungen 220ff.
 - , Verbindungsanforderungen 222
- Scripts, Perl DBI/DBD, Erstellung 275, 278
- scripts-Verzeichnis 40
- SEC_TO_TIME(Sekunden)-Funktion 339, 516
- SECOND(Zeit) Funktion 338, 516
- SELECT
 - , Anweisungen 225
 - , Anweisungen, Funktionen 488
 - , Befehl 160, 484
 - , Berechtigung, Privileg 360
 - , Gleichungen 160
 - , reservierte Wörter 156
 - , Syntax 160
 - , Tabellen 160
- SELECT HIGH_PRIORITY-Sperren 201
- SELECT INTO OUTFILE-Befehl 149
- SELECT INTO-Befehl 162
- Select_priv (Berechtigung, Privileg) 349
- Select_priv, Spalte 349
- Semikolon (45, 79, 96, 100, 303
- Server
 - , Client/Server-Paradigma siehe Spalten, Indizes, Meet_A_Geek-Datenbank, Tabellen
 - , Datenbanken 44f.
 - , Fat Client siehe Spalten, Indizes, Meet_A_Geek-Datenbank, Tabellen
 - , Fernwartung 44
 - , Multi-Threaded siehe Spalten, Indizes, Meet_A_Geek-Datenbank, Tabellen
 - , MySQL 44
 - , MySQL, starten 41
 - , MySQL, Statusinformationen 404
 - , MySQL, stoppen 41
 - , MySQL, Threadinformationen 406
 - , MySQL, Versionsinformationen 404
 - , MySQL-Monitor 48
 - , Thin-Client siehe Spalten, Indizes, Meet_A_Geek-Datenbank, Tabellen
 - , Verbindungsanforderungen 222
 - , Verbindungsaufnahme mit PHP 317
- Server-Objekt 247
- Services 41, 220
- SESSION_USER() Funktion 397, 516
- SET
 - , Befehl 164
 - , Datentyp 117
 - , reservierte Wörter 156
- share-Verzeichnis 40
- Shortcuts 307
- SHOW-Befehl 484
- shutdown, mysqladmin 43
- SHUTDOWN (Berechtigung, Privileg) 360
- Shutdown (processlist-Befehl) 407
- Shutdown_priv (Berechtigung, Privileg) 349
- Shutdown_priv, Spalte 349
- Shutdown-Befehl 402
- Sicherheit
 - , Anfrage-Verifikation 354
 - , Benutzerrechte, Änderungen 356
 - , Benutzerrechte, GRANT-Anweisung 359
 - , Benutzerrechte, Löschen 358
 - , Benutzerrechte, Vergabe 358
 - , columns_priv-Tabelle 351
 - , db-Tabelle 351
 - , host-Tabelle 351

- , Kontrollebenen 353
- , Löschen von Benutzern und Benutzerrechten 363
- , mysql-Datenbank 348
- , Sicherheitsaspekte 438
- , tables_priv-Tabelle 351
- , user-Tabelle 349
- , Verbindungs-Verifikation 353
- , Vergabe der Benutzerrechte 356
- , Vorsichtsmaßnahmen 366
- SIGN(x)-Funktion 516
- SIN(x)-Funktion 517
- skip-grant-tables(Parameter, mysqld) 421
- skip-name-resolve(Parameter, mysqld) 421
- Slave-Datenbanken, Backup 391
- Sleep (processlist-Befehl) 406
- Slow queries, mysqladmin-Statusbericht 404
- SMALLINT-Datentyp 111f.
- Sortieren 174
 - , Daten 175
 - , Spalten 174
- SOUNDEX(x)-Funktion 517
- Source, Parameter (RecordsetObject) 256
- SPACE(x)-Funktion 517
- Spalten
 - , Attribut, AUTO_INCREMENT 113, 119
 - , Attribut, BINARY 119
 - , Attribut, DEFAULT 119
 - , Attribut, NOT NULL 119
 - , Attribut, NULL 119
 - , Attribut, PRIMARY KEY 119
 - , Attribut, UNIQUE 119
 - , Attribut, UNSIGNED 119
 - , Attribut, ZEROFILL 115, 119
 - , Datentypen 109ff., 115, 117
 - , Durchschnitt 173
 - , hinzufügen 102
 - , Index erstellen 105
 - , löschen 102f.
 - , Namen ändern 101
 - , Normalisierung 84
 - , Sortieren 174
 - , SQL-Anweisungen 157
 - , Standard-Abweichung 173
 - , Summe 173
- , Typ ändern 102
- , user-Tabelle 349f.
- , variabler Länge 423
- Speichern
 - , Bilder in Datenbanken 469
 - , Daten in Dateien 49
 - , gespeicherte Prozeduren 373
 - , UDFs (User Defined Functions) 375
- Sperren 195ff., 217
 - , Anfragen-Warteschleife 200
 - , Datenbank-Backup 203
 - , Deadlock 199
 - , INSERT 202
 - , LOCK TABLES 199
 - , LOW_PRIORITY WRITE 201
 - , READ 199
 - , READ LOCK 203
 - , SELECT HIGH_PRIORITY 201
 - , Syntax 198
 - , transaktionale Kontrolle 202
 - , UNLOCK TABLES 201
 - , UPDATE 202
 - , WRITE 199
- SQL (Structured Query Language) 44
 - , Abgrenzung 27
 - , Aggregat-Funktionen 171
 - , ANSI 92 167
 - , ANSI SQL92 32
 - , Anweisungen, optimierte 421
 - , Befehle 155, 225
 - , Daten, Bearbeitung 158
 - , IBM 154
 - , INSERT 162
 - , Joins 167
 - , Joins, CROSS JOIN 169
 - , Joins, INNER JOIN 169
 - , Joins, LEFT JOIN 170
 - , Joins, NATURAL LEFT JOIN 171
 - , Joins, RIGHT JOIN 176
 - , Komponenten 155
 - , Oracle 154
 - , relationale Datenbank 154
 - , REPLACE 176
 - , reservierte Wörter 155
 - , SELECT-Anweisung 160



- , Sortieren 174
- , Syntax 158
- , UPDATE 164
- SQRT(x)-Funktion 517
- Standard-Abweichung (STD) 173
- Standardstruktur, Tabellen in Datenbanken 48
- Starten
 - , MySQL-Monitor 96
 - , MySQL-Server 41
- Startoptionen, MySQL 49
- Static (CursorType-Parameter) 257
- Statistics (processlist-Befehl) 407
- Status-Befehl 401
- STD()-Funktion 173
- STD(Ausdruck)- oder STDDEV(Ausdruck)-Funktion 518
- Stoppen, MySQL-Server 41, 43
- STRCMP(String1, String2)-Funktion 518
- Strukturen
 - , Daten 407
 - , Datenbanken, Anzeige 47
 - , Tabellen, Standard 48
- SUBDATE()-Funktion 520
- SUBDATE(Datum, INTERVALL Ausdruck Typ)-Funktion 341
- Subselect 162
- SUBSTRING()-Funktion 191
- SUBSTRING_INDEX(x, y, z)-Funktion 519
- Subtraktion
 - , Funktion 185
 - , Operator 307
- SUM()-Funktion 173
- SUM(Ausdruck)-Funktion 519
- support-files-Verzeichnis 39
- Surrogate Keys siehe synthetische Schlüssel
- symbolische Links 73
- Syntax
 - , ALTER TABLE 211
 - , CASE 187
 - , CREATE FUNCTION 377
 - , CREATE TABLE 205
 - , GRANT 359
 - , If...Then...Else 242
 - , LOCK TABLES 199
 - , mysqladmin 401
 - , mysqlimport 392
 - , Schlüssel für einzelne Spalten 206
 - , Schlüssel löschen 216
 - , SELECT 160
 - synthetische Primärschlüssel 213
 - synthetische Schlüssel 212
 - SYSDATE()-Funktion 336, 520
 - SYSTEM_USER()-Funktion 520
 - System-DSN, Meet_A_Geek-Datenbank, erstellen 233
 - System-Funktionen 397
 - , DATABASE() 397
 - , DECODE(encoded_string, password) 399
 - , ENCODE(string, password) 399
 - , ENCRYPT(string[,salt]) 398
 - , MD5(string) 400
 - , PASSWORD(String) 397
 - , SESSION_USER() 397
 - , SYSTEMUSER() 397
 - , USER() 397
 - , VERSION() 397
 - SYSTEMUSER()-Funktion 397
 - System-Variablen
 - , back_log 419
 - , buffers 419
 - , delayed_queue_size 419
 - , flush_time 419
 - , table_cache 419
 - , wait_timeout 419
 - System-Variablen, konfigurieren 419
- T**
 - t, --no-create-info (Optionen, mysqldump) 147, 387
 - t, --timeout=x (Option, mysqladmin) 402
 - tab=Pfad, -T=Pfad (Option, mysqldump) 147, 388
 - Tabellen
 - , Ads-Tabelle, Meet_A_Geek-Datenbank 434
 - , ändern 101
 - , Aktualisierung mit UDFs 379
 - , Alias 159
 - , Backup 203
 - , Bericht, Flush tables 404
 - , Bericht, Open tables 404

- , Beziehungen siehe Beziehungen
- , Customers-Tabelle, Meet_A_Geek-Datenbank 433
- , db 78, 351, 354
- , DELETE-Anweisung 159
- , Erstellen 96, 102
- , Felder, Datentypen 61
- , Felder, Schlüssel 61
- , GRANT- 349
- , host 78
- , IF NOT EXISTS, Parameter 99
- , Indizes 104f.
- , Indizes, erstellen 105
- , Indizes, löschen 106
- , Joins 214
- , Karthesischer Join 159
- , löschen 103
- , maximale Datengröße 108
- , Meet_A_Geek-Datenbank 432ff.
- , mehrere Operationen 201
- , mysql-Datenbank 78, 348
- , Namenskonventionen 99
- , Nicht-Schlüssel-Daten beseitigen 89
- , Normalisierung 84
- , Optimierung 424
- , Orders-Tabelle, Meet_A_Geek-Datenbank 434
- , Performancesteigerung 421
- , Products-Tabelle, Meet_A_Geek-Datenbank 434
- , READ-Sperre 200
- , Reparieren mit isamchk 407
- , Schlüssel siehe Schlüssel
- , SELECT INTO-Befehl 162
- , SELECT-Befehl 159
- , Spalte, Alter_priv 350
- , Spalte, Column_name- 352
- , Spalte, Create_priv 349
- , Spalte, Db- 351
- , Spalte, Delete_priv 349
- , Spalte, Drop_priv 349
- , Spalte, File_priv 350
- , Spalte, Grant_priv 350
- , Spalte, Grantor- 352
- , Spalte, Host 349
- , Spalte, Host- 351
- , Spalte, Index_priv 350
- , Spalte, Insert_priv 349
- , Spalte, password 349
- , Spalte, Process_priv 350
- , Spalte, References_priv 350
- , Spalte, Reload_priv 349
- , Spalte, Select_priv 349
- , Spalte, Shutdown_priv 349
- , Spalte, Table_name- 351
- , Spalte, Table_priv- 352
- , Spalte, Timestamp- 351
- , Spalte, Update_priv 349
- , Spalte, User 349
- , Spalte, User- 351
- , Spalten siehe Spalten
- , Sperren 199, 217
- , Sperren (LOCK TABLES) 199
- , Standardstruktur 48
- , Tabelle, columns_priv- 351
- , Tabelle, db- 351
- , Tabelle, host- 351
- , Tabelle, tables_priv- 351
- , Tabelle, user- 349
- , table_cache, Systemvariable 419
- , Teilabhängigkeiten beseitigen 88
- , temporäre, einsehen 98
- , temporäre, erstellen 98
- , transaktionale Kontrolle, Alternativen 202
- , Umbenennung 102
- , UNLOCK TABLES, Sperren 201
- , UPDATE, Sperren 201
- , UPDATE-Befehl 159
- , user 78
- , wiederholende Daten beseitigen 86
- , WRITE-Sperre 200
- table_cache 419
- Table_name-Spalte 351
- tables_priv-Tabelle 348, 351, 355
- Table-Scan 204
- TAN(x)-Funktion 520
- tar, packen, sichern 396
- TcX 26
- , Consultingfirma 26
- , MySQL-Treiber 236
- , Webseite 370
- Teilabhängigkeiten 88



Teil-Schlüssel 208
Telnet-Sitzung 143
temporäre Tabellen 98
–, bearbeiten 98
–, einsehen 98
TERMINATED BY, Schlüsselwort 135
tests-Verzeichnis 39
TEXT-Datentyp 115f.
Thin Client 31
Threads
–, Definition 196
–, MySQL, Informationen 406
–, mysqladmin Statusbericht 404
Three-Tier-Design 181
TIME_FORMAT(Zeit, Format)-Funktion 337
TIME_FORMAT(Zeit, Formatzeichen)-Funktion 520
TIME_TO_SEC(Zeit)-Funktion 339, 521
TIME-Datentyp 327
TIMESTAMP-Datentyp 327f.
Timestamp-Spalte 351
TINYBLOB-Datentyp 115
TINYINT-Datentyp 111f.
TINYTEXT-Datentyp 115
TO_DAYS(Datum)-Funktion 339, 521
Tools, crash_me 40
transaktionale
–, Anweisungen 202
–, Datenbanken siehe T-SQL
–, Kontrolle, Alternativen 202
Treiber
–, Datenbanken 136
–, MyODBC, Installation 231
–, MySQL DBD (Datenbank-Treiber) 274
–, ODBC (Open Database Connectivity) 136, 230
–, Schnittstellen 221
–, Schnittstellen-Verbindungsanforderungen 222
Trigger 380
TRIM()-Funktion 190
TRIM([[[BOTH oder LEADING oder TRAILING] [x] FROM y])]-Funktion 521
TRUNCATE(x, y)-Funktion 522
T-SQL (Transact SQL) 371
–, BEGIN TRAN 372

–, Cursor 379
–, gespeicherte Prozeduren 373
–, ROLLBACK TRAN 372
–, Transaktions-Protokoll 373
–, Trigger 380
–, T-SQL-Befehle 371
Tuning, Performance 416
–, mysqld-Daemon-Parameter 420
–, System-Variablen Einstellungen 417
Typenkonvertierung (PHP) 303

U

–u Benutzername, --user Benutzername (Option, mysqlimport) 130, 392
–u Benutzername, --user=Benutzername (Option, mysqldump) 148, 388
–u, --unpack (Option, isamchk) 410
–u, --unpack (Option, myisamchk) 425
–u, --user=Benutzername (Option, mysqladmin) 403
UCASE()-Funktion 192
UCASE(x)-Funktion 522
UDFs (User Defined Functions) 375
Umbenennung
–, Spaltennamen 101
–, Tabellennamen 102
ungenutzten Speicherplatz freigeben 423
UNIQUE, Spaltenattribut 119, 120
UNLOCK TABLES Befehl 201, 485
UNSIGNED-Spaltenattribut 113, 119
Unternehmen
–, Datawarehousing 29
–, Flexibilität 29
–, Internet 29
–, Multi-User-Systeme 28
–, Sicherheit 29
UPDATE
–, (Berechtigung, Privileg) 360
–, Befehl 164, 485
–, reservierte Wörter 156
Update_priv (Berechtigung, Privileg) 349
Update_priv, Spalte 349
Update-Protokolle 394
–, wiederherstellen 395
UPPER(x)-Funktion 523
Uptime, mysqladmin Statusbericht 404

USAGE (Berechtigung, Privileg) 360

USE-Befehl 74, 133, 485

User (Berechtigung, Privileg) 349

User Defined Functions siehe UDFs

USER()-Funktion 397, 523

User-Spalte 349, 351

user-Tabelle 78, 348f.

USING, Schlüsselwort 170

Utilities

–, isamchk 407

–, myasamchk 407

–, mysql_setpermission 362

–, mysqladmin 401ff.

–, mysqldump 386f.

–, mysqlimport 392

V

-v (Option, myisamchk) 425

-v, --verbose (Option, isamchk) 410

-V, --version (Option, isamchk) 410

-V, --version (Option, mysqladmin) 403

-V, --version (Parameter, mysqld) 421

VALUES, Schlüsselwort 162

VARCHAR(X) Datentyp 115

variable Länge, Felder 423

Variablen

–, back_log 419

–, buffers 419

–, Datentypen (PHP) 303

–, delayed_queue_size 419

–, flush_time 419

–, PHP 304

–, phpinfo(), Systemvariablen 304

–, Systemvariablen, Konfiguration 417

–, table_cache 419

–, Typenkonvertierung (PHP) 303

–, Variables, Befehl 401

–, wait_timeout 419

–, Wertzuweisungen (PHP) 306

Variables-Befehl 401

VBScript 237, 240

–, Active Server Page 240f.

–, Funktionen 241

–, Generieren einer Webseite 243

–, Kommentieren des Programmcodes 243

–, Programmfluss 242

–, Schleifenkontrolle 242

–, Variant 241

Verben

–, SQL-Syntax, Anweisungen 156

–, Tabellen Alias 159

Verbindung

–, ADO, zu Datenbanken 250

–, Anforderungen 222

–, Daten abfragen 223

–, Herstellen 220, 223

–, Perl DBI, zu Datenbank 276

–, PHP, zu Datenbank 317

–, Schließen 226

Vergleich

–, MySQL und andere Datenbanken 370

–, MySQL und SQL 27

vermischte Datentypen 117

VERSION()-Funktion 397, 523

Version, MySQL 404

Version-Befehl 401

verteilte Systeme 32

Verzeichnisse 43

–, /mysql, Kommandozeilen-Interface 44

–, /usr/local/mysql 40

–, bin 39

–, data 39

–, include 39

–, lib 39

–, mysql 42, 96

–, mysql_bench 40

–, scripts 40

–, share 40

–, support-files 39

–, tests 39

vollständiges Backup 394

W

-w (Option, myisamchk) 425

-W, --wait (Option, isamchk) 410

-w, --wait[=x] (Option, mysqladmin) 403

-w=Bedingung, --where=Bedingung (Option, mysqldump) 148, 389

wait_timeout 419

Wall, Larry 270

Warteschleifen 200

–, READ-Sperren 200



- , WRITE-Sperren 200
- webbasierte Anwendung, Programmierlogik 180
- Web-Browser, HTML- und PHP-Code 301
- Webseite
 - , ActiveState 271
 - , ADO-Objekte 246f.
 - , Anzeigen aufgeben 457ff., 461ff.
 - , Anzeigen von Kundendaten 263ff.
 - , Benutzer-Login 450, 452ff.
 - , Bestätigungsmeldung an Browser 289
 - , Code generieren 243f.
 - , Daten 282ff.
 - , erstellen 439ff., 444
 - , generieren mit VBScript 243, 245
 - , LAMP (Linux Apache MySQL PHP) 300
 - , Login.html 449f.
 - , Login.pl-Skript 451ff.
 - , Meet_A_Geek 62, 431
 - , Meet_A_Geek Datenbank 260
 - , Meet_A_Geek-Datenbank, Anzeige der Daten 263ff.
 - , Meet_A_Geek-Datenbank, Design 258f.
 - , MySQL 36f., 231
 - , MySQL, TcX 370
 - , Perl 271
 - , Perl-Script 444ff., 449
 - , PHP (Hypertext Preprocessor) 300f., 322
 - , PHP, ZEND 300
 - , PostAd.asp 261
 - , PostAd.asp, Code 260
 - , Programmierlogik 180
 - , Request-Objekt 247
 - , Response-Objekt 248f.
 - , SearchAd.pl-Skript 459, 461ff.
 - , Server-Objekt 247
 - , Such-Formular 456ff., 461ff.
 - , Testen des Programmcodes 468
 - , vendor.html-Seite 282, 284
 - , WAMP (Windows Apache MySQL PHP) 300
- Webseiten online stellen 468
- WEEKDAY(Datum)-Funktion 331, 524
- Werkzeuge
 - , Datenbank-Import 144
 - , Utilities 362
- Wertebereiche numerischer Datentypen 112
- WHERE 156
 - , reservierte Wörter 156
 - , SQL 156
- WHERE-Klausel 157, 159
 - , Datums- und Zeitfunktionen 331
 - , Erstellung 423
 - , Funktionen 187
 - , Schlüssel 207, 213
 - , SQL 154
- while-Schleifen 309, 311
 - , break 311
 - , continue 311
- Wiederherstellen, Daten, mysqlimport-Programm 392
- Windows
 - , MS-DOS-Eingabeaufforderung 271
 - , MySQL 38, 43
 - , MySQL, ODBC 230
- WinZip 38, 231
- Workarounds, gespeicherte Prozeduren 375, 377, 378, 379
- Write LOCK 200
- WRITE-Sperre 199
 - , Regeln, Befehle 200
- X**
- XML (Extensible Markup Language) 129
- Y**
- YEAR Datentyp 327f.
- YEAR Datum-Funktion 334, 525
- YEARWEEK(Datum[,Wochenstarttag]) Funktion 334
- Z**
- Zeichen
 - , alphanumerische 110
 - , Datentypen 115
 - , Funktionen 189ff.
- Zeichenketten-Datentypen 115
- Zeichenketten-Funktionen 189
- Zeiger, Arrays 315
- ZEROFILL, Spaltenattribut 113, 115, 119
- Zip-Datei, ODBC (Open Database Connectivity) 231