



随书附赠 **DVD**
与书中所有章节一一对应的 **PPT** 文档
超过 **14** 小时的多媒体语音精彩视频



QQ技术群：21948169

MySQL 数据库应用

从入门到精通 **第二版**

王飞飞 崔洋 贺亚茹 编著

本书技术支持论坛
<http://www.rzchina.net>

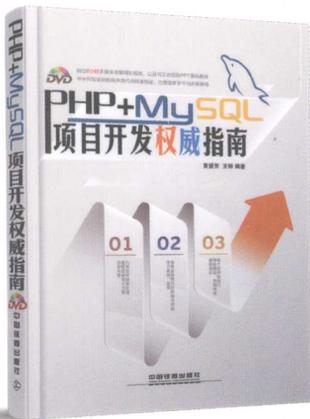
秉承理论学习与实际开发相结合的原则，力求实现所有技术点和经典实例的完美搭配，旨在帮助MySQL数据库初学者轻松入门，并迅速达到熟练水平。



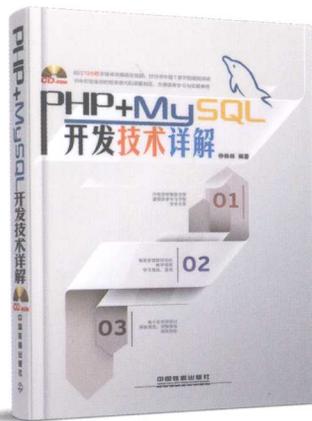
中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE



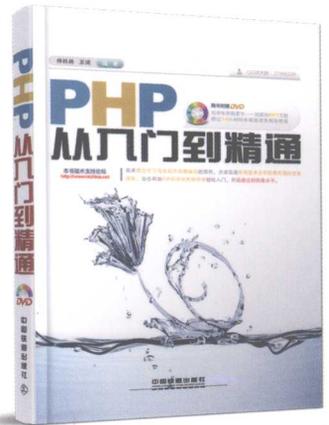
MySQL数据库应用 从入门到精通 **第二版**



书名:《PHP+MySQL 项目开发权威指南》
定价: 59.80元 (附赠光盘)



书名:《PHP+MySQL 开发技术详解》
定价: 69.80元 (附赠光盘)



书名:《PHP从入门到精通》
定价: 59.80元 (附赠光盘)

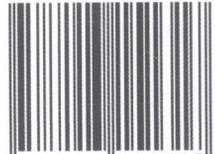
上架建议: 计算机/数据库/My



中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

地址: 北京市西城区右安门西街8号
邮编: 100054
网址: <http://www.tdpress.com>

ISBN 978-7-113-15131-7



9 787113 151317 >

定价: 59.80元 (附赠光盘)

014037233

TP311.138SQ
507-2



MySQL 数据库应用

从入门到精通 第二版

王飞飞 崔洋 贺亚茹 编著



TP311.138SQ
507-2



北航 C1725487

中国铁道出版社
CHINA RAILWAY PUBLISHING HOUSE

P

内 容 简 介

最适合的往往是最实用的，就像我们要讲述的 MySQL 一样，它的功能不是最全的，架构也不是最完善的。但是其体积小、速度快、总体拥有成本低，尤其是它具备开放源码的优势，迅速成为中小型企业和网站的首选数据库。

本书共分为 3 篇。其中第一篇为 MySQL 数据库基础篇，内容包括数据库所涉及的基本概念、MySQL 数据库的安装与配置。第二篇为 MySQL 数据库操作和应用篇，内容包括操作数据库对象、操作表对象、操作索引对象、操作视图对象、操作触发器对象和操作数据。第三篇为 MySQL 数据库管理篇，内容包括 MySQL 数据库的用户管理和权限管理、MySQL 数据库的日志管理、MySQL 数据库的性能优化和 PowerDesigner 数据库设计软件。

秉承注重理论与实际开发相结合的原则，书中的每个技术点都配备了与此相对应的实例，旨在帮助 MySQL 数据库初学者快速入门，同时也适合 MySQL 数据库管理员和想全面学习 MySQL 数据库技术以提升应用水平的人员使用。

图书在版编目 (CIP) 数据

MySQL 数据库应用从入门到精通 / 王飞飞, 崔洋, 贺亚茹编著. — 2 版. — 北京: 中国铁道出版社, 2014. 4
ISBN 978-7-113-15131-7

I. ①M… II. ①王… ②崔… ③贺… III. ①关系数据库系统 IV. ①TP311.138

中国版本图书馆 CIP 数据核字 (2014) 第 021338 号

书 名: MySQL 数据库应用从入门到精通 (第 2 版)

作 者: 王飞飞 崔洋 贺亚茹 编著

策 划: 荆 波

责任编辑: 张 丹

责任印制: 赵星辰

读者热线电话: 010-63560056

特邀编辑: 马寒梅

封面设计: 多宝格·付巍

出版发行: 中国铁道出版社 (北京市西城区右安门西街 8 号 邮政编码: 100054)

印 刷: 三河市兴达印务有限公司

版 次: 2014 年 4 月第 2 版 2014 年 4 月第 1 次印刷

开 本: 787mm×1092mm 1/16 印张: 29.75 字数: 692 千

书 号: ISBN 978-7-113-15131-7

定 价: 59.80 元 (附赠光盘)

版权所有 侵权必究

凡购买铁道版图书，如有印制质量问题，请与本社读者服务部联系调换。电话：(010) 51873174

打击盗版举报电话：(010) 51873659

前言

Foreword

MySQL 的优势

最适合的往往是最实用的，就像我们要讲述的 MySQL 一样，它的功能不是最全的，架构也不是最完善的。但是其体积小、速度快、总体拥有成本低，尤其是它具备开放源代码的优势，迅速成为中小型企业网站的首选数据库。

在数据库世界里，关于 MySQL 数据库的新闻一直不断，始终萦绕在程序员的耳边。2008 年，SUN 公司以 10 亿美元收购了 MySQL 数据库，标志着该数据库已经成为世界上的主流数据库之一。2010 年，Oracle 公司收购了 SUN 公司，标志着该数据库成为 Oracle 公司的主流数据库产品。随着 MySQL 数据库的逐渐成熟，全球规模最大的网络搜索引擎公司 Google 决定使用该 MySQL 数据库，国内很多大型的公司也开始使用 MySQL 数据库，例如网易、新浪等。这就给 MySQL 数据库带来了前所未有的机遇，同时也出现了学习 MySQL 数据库的高潮。

与同类书相比本书有何特色

本书以“数据库基本概念→MySQL 数据库环境搭建→利用 SQL 语句操作数据库对象→MySQL 数据库高级管理”为主线，辅以开发项目时遇到的常用 SQL 语句操作，让读者在学习关于 MySQL 数据库和 SQL 语句基础知识的同时，能更快速地适应数据库的工作。

在学习 MySQL 数据库软件的同时，希望读者能牢记：动手才是硬道理。结合这一主题，本书最后提供了两个非常实用的案例，供读者学习和研究。

本书从 MySQL 数据库的环境配置和 SQL 语句的基本语法出发，详细讲解了 MySQL 数据库的各种基础操作和如何利用 SQL 语句来操作数据库对象，同时也给出了极具代表性和实用性的应用示例。

本书的特点主要体现在以下几个方面。

- 以现实职场中经典数据库操作和完整系统的项目为背景，结合当前最主流的版本 MySQL 软件的基本语法知识，组织和编写全书的内容。
- 采用以实例驱动模式为指引，即不仅每章都是一个完整的实例，而且各章实例所涉及的知识还涵盖了 MySQL 软件的各方面。通过对本书实例的剖析，读者不仅能够深刻体会到数据库和 MySQL 软件的各种知识点特性，而且在具体开发应用时能够“游刃有余”。
- 从数据库的基础概念开始讲解，逐步深入到 MySQL 软件的基础操作和软件的高级操作、管理和应用。内容难度从易到难，讲解由浅入深，使学习循序渐进。
- 每段代码都经过详细步骤来演示，并指明了难点和核心要点，使读者能够明确重点。在具体讲解时，还穿插了大量的使用技巧，以便让读者能够体验实际操作 MySQL 软件的技巧。

- 随书附带的光盘中，包括了各章节的 SQL 语句，这些 SQL 语句代码不但具有一定的实用性，更可贵的是具有一定的通用性。同时还配有大量的视频讲解，对于初学者来说，视频讲解具有非常直观的辅助学习的作用。
- 注重实际应用，提高实战水平。本书不仅介绍了 SQL 语句设计，还讲解了用 PowerDesigner 软件进行数据库设计的知识。

第 2 版改版说明：

《MySQL 数据库应用从入门到精通》第 1 版出版之后，在很短的时间内得到了读者的肯定和市场认可，我们也很高兴该书的策划想法能顺利通过市场检验。第 1 版图书出版至今已经一年有余了，一年来我们细细的审阅过该书的每个章节，根据 MySQL 数据库技术的发展和作者经验的增加，逐一完善了书中许多欠缺的地方，如下所示。

- 第 1 章中增加了 MySQL 数据库系统的历史，发展以及开源文化介绍。
- 第 1 版中第 4.2 节数字类型介绍比较粗略，本次改版细化并补充实例。
- 第 11.5 节子查询中补充了 EXISTS 查询关键字的相关内容。
- 第 17（第 1 版中第 16 章）章日志管理中补充了慢查询日志的相关内容，让日志分类更完整。
- 第 1 版中缺少游标和事务管理的基础技术点，本次改版特意增加了该部分内容（第 16 章）。
- 根据新版本的变化，第 1 版光盘中的讲解视频和 PPT 文件重新整理归纳，并根据读者来信反映，做到了讲解更为细致。
- 根据热心读者反映的意见和建议，全书通审并修订了很多细小的错误。

本书内容及知识体系

本书分为 3 篇，共 19 章，结合目前最主流的软件环境 MySQL 5.5.x，全方位地介绍了关于数据库的基本概念和 MySQL 软件的各种操作，从数据库的基础知识、MySQL 软件的基础操作和标准 SQL 语句讲起，再进一步详细介绍了关于 MySQL 软件的高级操作。

第 1 篇 MySQL 数据库基础篇（第 1~2 章）

本篇主要介绍了 MySQL 软件涉及的基础概念和该软件的安装过程。首先介绍了数据库涉及的基础概念，分别为数据管理技术发展阶段、数据库技术经历阶段、数据库管理系统提供的功能、SQL 语言和常见数据库管理系统；然后详细讲解了 MySQL 软件的安装和配置过程。

第 2 篇 MySQL 数据库操作和应用篇（第 3~15 章）

本篇主要介绍了 MySQL 数据库对各类对象的基本操作和应用，其中前者主要包含数据库对象操作、表对象操作、索引对象操作、视图对象操作、触发器对象操作和数据操作。在具体介绍操作数据时，详细介绍了数据的插入、更新和删除操作，而对于数据查询操作则会从单表查询和多表查询两方面进行介绍，单表查询主要包含简单数据记录查询、条件数据查询、排序数据查询结果、限制数据查询数量、统计函数和分组数据查询；多表查询主要包含内连接查询、外连接查询、合并查询数据记录和子查询。

第 3 篇 MySQL 数据库管理篇（第 16~19 章）

本篇主要介绍了 MySQL 数据库的高级管理，包含 MySQL 数据库的用户管理和权限管理、MySQL 数据库的日志管理和 MySQL 数据库的性能优化，最后介绍了用 PowerDesigner 软件进行的数据库设计。

物超所值的附赠光盘

为了方便读者阅读本书，本书附带 DVD 光盘。内容如下：

- 本书所有实例的源代码。
- 本书每章内容的多媒体语音教学视频。
- 本书每章内容的 PPT 文件。

适合阅读本书的读者

- 希望使用 MySQL 数据库的新手。
- 迫切希望提高 MySQL 数据库使用技能和水平的程序人员。
- 具备一定的编程经验但是数据库操作技巧不丰富的工程师。
- 希望了解和使用 MySQL 5 服务器、SQLyog 客户端软件和 PowerDesigner 设计软件的人员。

阅读本书的建议

- 没有数据库基础知识的读者，建议从第 1 章开始按顺序阅读并演练每一个实例。
- 有一定 SQL 语言基础的读者，可以根据实际情况有重点地选择所需阅读章节和案例。
- 对于每一个章节，先自己思考一下所需要掌握的知识点，然后再阅读，这样学习效果会更好。
- 可以先将书中的每个知识点和案例阅读一遍，然后结合光盘中提供的多媒体教学视频再理解一遍，这样学习起来就更加容易，理解也会更加深刻。

编 者

2014 年 2 月

目 录

Contents

第 1 篇 MySQL 数据库基础篇

第 1 章 数据库概述

| | |
|------------------------------|---|
| 1.1 关于数据库的基本概念 | 2 |
| 1.1.1 数据管理技术的发展阶段 | 2 |
| 1.1.2 数据库系统阶段涉及的概念 | 3 |
| 1.1.3 数据库技术经历的阶段 | 3 |
| 1.1.4 数据库管理系统提供的功能 | 4 |
| 1.1.5 什么是 SQL | 4 |
| 1.2 MySQL 数据库管理系统 | 5 |
| 1.2.1 MySQL 与开源文化 | 5 |
| 1.2.2 MySQL 发展历史 | 5 |
| 1.2.3 常见数据库管理系统 | 7 |
| 1.2.4 为什么要使用 MySQL 数据库 | 7 |
| 1.3 小结 | 8 |

第 2 章 MySQL 安装和配置

| | |
|---|----|
| 2.1 下载和安装 MySQL 软件 | 9 |
| 2.1.1 基于客户端——服务器 (C/S) 的数据库管理系统 | 9 |
| 2.1.2 MySQL 的各种版本 | 10 |
| 2.1.3 下载 MySQL 软件 | 10 |
| 2.1.4 安装 MySQL 软件 | 12 |
| 2.1.5 图形化配置数据库服务器 MySQL | 14 |
| 2.1.6 手工配置数据库服务器 MySQL | 17 |
| 2.1.7 卸载 MySQL 软件 | 20 |
| 2.2 关于 MySQL 软件的常用操作 | 21 |
| 2.2.1 通过图形界面启动和关闭 MySQL 服务 | 21 |
| 2.2.2 通过 DOS 窗口启动和关闭 MySQL 服务 | 23 |
| 2.3 MySQL 官方客户端软件 | 24 |
| 2.3.1 命令行客户端软件——MySQL Command Line Client | 25 |
| 2.3.2 通过 DOS 窗口连接 MySQL 软件 | 26 |

MySQL 数据库应用从入门到精通

| | | |
|-------|----------------------------------|----|
| 2.3.3 | 下载 MySQL-Workbench 客户端软件 | 28 |
| 2.3.4 | 安装 MySQL-Workbench 客户端软件 | 29 |
| 2.3.5 | 使用 MySQL-Workbench 客户端软件 | 30 |
| 2.4 | MySQL 常用图形化管理软件——SQLyog 软件 | 32 |
| 2.4.1 | 下载 SQLyog 软件 | 32 |
| 2.4.2 | 安装 SQLyog 客户端软件 | 33 |
| 2.4.3 | 通过 SQLyog 客户端软件登录 MySQL 软件 | 35 |
| 2.5 | 使用免安装的 MySQL 软件 | 35 |
| 2.6 | 小结 | 37 |

第 2 篇 MySQL 数据库操作和应用篇

第 3 章 MySQL 数据库基本操作

| | | |
|-------|-------------------------|----|
| 3.1 | 数据库和数据库对象 | 40 |
| 3.2 | 数据库相关操作——创建数据库 | 41 |
| 3.2.1 | 创建数据库的语法形式 | 41 |
| 3.2.2 | 通过工具来创建数据库 | 42 |
| 3.3 | 数据库相关操作——查看和选择数据库 | 44 |
| 3.3.1 | 查看数据库 | 44 |
| 3.3.2 | 选择数据库 | 45 |
| 3.4 | 数据库相关操作——删除数据库 | 46 |
| 3.4.1 | 删除数据库的语法形式 | 46 |
| 3.4.2 | 通过工具来删除数据库 | 47 |
| 3.5 | 小结 | 49 |

第 4 章 MySQL 数据库中存储引擎和数据类型

| | | |
|-------|--------------------------|----|
| 4.1 | 认识存储引擎 | 50 |
| 4.1.1 | MySQL 5.5 所支持的存储引擎 | 50 |
| 4.1.2 | 操作默认存储引擎 | 53 |
| 4.1.3 | 选择存储引擎 | 55 |
| 4.2 | 数据类型 | 56 |
| 4.2.1 | 整数类型 | 56 |
| 4.2.2 | 浮点数类型、定点数类型和位类型 | 59 |
| 4.2.3 | 日期和时间类型 | 60 |
| 4.2.4 | 字符串类型 | 62 |
| 4.3 | 小结 | 64 |

第5章 表的操作

| | |
|--|-----|
| 5.1 表的基本概念..... | 65 |
| 5.2 创建表..... | 66 |
| 5.2.1 创建表的语法形式..... | 66 |
| 5.2.2 通过 SQLyog 客户端软件来创建表..... | 67 |
| 5.3 查看表结构..... | 69 |
| 5.3.1 DESCRIBE 语句查看表定义..... | 69 |
| 5.3.2 SHOW CREATE TABLE 语句查看表详细定义..... | 70 |
| 5.3.3 通过 SQLyog 软件来查看表信息..... | 70 |
| 5.4 删除表..... | 71 |
| 5.4.1 删除表的语法形式..... | 72 |
| 5.4.2 通过 SQLyog 软件删除表..... | 72 |
| 5.5 修改表..... | 73 |
| 5.5.1 修改表名..... | 73 |
| 5.5.2 增加字段..... | 74 |
| 5.5.3 删除字段..... | 78 |
| 5.5.4 修改字段..... | 79 |
| 5.6 通过 SQLyog 客户端软件修改表..... | 83 |
| 5.6.1 修改表名..... | 84 |
| 5.6.2 增加字段..... | 84 |
| 5.6.3 删除字段..... | 87 |
| 5.6.4 修改字段..... | 89 |
| 5.7 操作表的约束..... | 93 |
| 5.7.1 MySQL 支持的完整性约束..... | 93 |
| 5.7.2 设置非空约束 (NOT NULL, NK)..... | 94 |
| 5.7.3 设置字段的默认值 (DEFAULT)..... | 95 |
| 5.7.4 设置唯一约束 (UNIQUE, UK)..... | 96 |
| 5.7.5 设置主键约束 (PRIMARY KEY, PK)..... | 97 |
| 5.7.6 设置字段值自动增加 (AUTO_INCREMENT)..... | 100 |
| 5.7.7 设置外键约束 (FOREIGN KEY, FK)..... | 102 |
| 5.8 小结..... | 104 |

第6章 索引的操作

| | |
|------------------|-----|
| 6.1 为什么使用索引..... | 105 |
| 6.2 创建和查看索引..... | 106 |

| | | |
|-------|----------------------------|-----|
| 6.2.1 | 创建和查看普通索引 | 106 |
| 6.2.2 | 创建和查看唯一索引 | 110 |
| 6.2.3 | 创建和查看全文索引 | 114 |
| 6.2.4 | 创建和查看多列索引 | 118 |
| 6.2.5 | 通过 SQLyog 客户端软件来创建索引 | 121 |
| 6.3 | 删除索引 | 125 |
| 6.3.1 | 删除索引的语法形式 | 125 |
| 6.3.2 | 通过 SQLyog 软件删除索引 | 127 |
| 6.4 | 小结 | 128 |

第 7 章 视图的操作

| | | |
|-------|-------------------------------------|-----|
| 7.1 | 为什么使用视图 | 129 |
| 7.2 | 创建视图 | 130 |
| 7.2.1 | 创建视图的语法形式 | 130 |
| 7.2.2 | 通过 SQLyog 软件创建视图 | 131 |
| 7.2.3 | 创建各种视图 | 132 |
| 7.3 | 查看视图 | 136 |
| 7.3.1 | SHOW TABLES 语句查看视图名 | 136 |
| 7.3.2 | SHOW TABLE STATUS 语句查看视图详细信息 | 136 |
| 7.3.3 | SHOW CREATE VIEW 语句查看视图定义信息 | 138 |
| 7.3.4 | DESCRIBE DESC 语句查看视图设计信息 | 139 |
| 7.3.5 | 通过系统表查看视图信息 | 139 |
| 7.3.6 | SQLyog 查看视图信息 | 140 |
| 7.4 | 删除视图 | 142 |
| 7.4.1 | 删除视图的语法形式 | 142 |
| 7.4.2 | 通过 SQLyog 软件删除视图 | 143 |
| 7.5 | 修改视图 | 144 |
| 7.5.1 | CREATE OR REPLACE VIEW 语句修改视图 | 144 |
| 7.5.2 | ALTER 语句修改视图 | 146 |
| 7.5.3 | 通过 SQLyog 软件修改视图 | 147 |
| 7.6 | 利用视图操作基本表 | 149 |
| 7.6.1 | 检索（查询）数据 | 149 |
| 7.6.2 | 利用视图操作基本表数据 | 150 |
| 7.7 | 小结 | 151 |

第 8 章 触发器的操作

| | | |
|-----|----------------|-----|
| 8.1 | 为什么使用触发器 | 152 |
|-----|----------------|-----|

| | | |
|-------|-------------------------------|-----|
| 8.2 | 创建触发器..... | 153 |
| 8.2.1 | 创建有一条执行语句的触发器..... | 153 |
| 8.2.2 | 创建包含多条执行语句的触发器..... | 155 |
| 8.2.3 | 通过 SQLyog 客户端软件来创建触发器..... | 157 |
| 8.3 | 查看触发器..... | 158 |
| 8.3.1 | 通过 SHOW TRIGGERS 语句查看触发器..... | 158 |
| 8.3.2 | 通过查看系统表 triggers 实现查看触发器..... | 160 |
| 8.3.3 | 通过 SQLyog 客户端软件来查看触发器..... | 162 |
| 8.4 | 删除触发器..... | 163 |
| 8.4.1 | 通过 DROP TRIGGER 语句删除触发器..... | 163 |
| 8.4.2 | 通过工具来删除触发器..... | 164 |
| 8.5 | 小结..... | 165 |

第 9 章 数据的操作

| | | |
|-------|------------------|-----|
| 9.1 | 插入数据记录..... | 166 |
| 9.1.1 | 插入完整数据记录..... | 166 |
| 9.1.2 | 插入数据记录一部分..... | 168 |
| 9.1.3 | 插入多条数据记录..... | 171 |
| 9.1.4 | 插入查询结果..... | 174 |
| 9.1.5 | 通过工具来插入数据记录..... | 176 |
| 9.2 | 更新数据记录..... | 178 |
| 9.2.1 | 更新特定数据记录..... | 179 |
| 9.2.2 | 更新所有数据记录..... | 180 |
| 9.2.3 | 通过工具来更新数据记录..... | 182 |
| 9.3 | 删除数据记录..... | 184 |
| 9.3.1 | 删除特定数据记录..... | 184 |
| 9.3.2 | 删除所有数据记录..... | 185 |
| 9.3.3 | 通过工具来删除数据记录..... | 187 |
| 9.4 | 小结..... | 189 |

第 10 章 单表数据记录查询

| | | |
|--------|-------------------------|-----|
| 10.1 | 简单数据记录查询..... | 190 |
| 10.1.1 | 简单数据查询..... | 191 |
| 10.1.2 | 避免重复数据查询——DISTINCT..... | 194 |
| 10.1.3 | 实现数学四则运算数据查询..... | 196 |
| 10.1.4 | 设置显示格式数据查询..... | 197 |

MySQL 数据库应用从入门到精通

| | | |
|--------|----------------------------------|-----|
| 10.2 | 条件数据记录查询 | 198 |
| 10.2.1 | 带关系运算符和逻辑运算符的条件数据查询 | 199 |
| 10.2.2 | 带 BETWEEN AND 关键字的范围查询 | 201 |
| 10.2.3 | 带 IS NULL 关键字的空值查询 | 203 |
| 10.2.4 | 带 IN 关键字的集合查询 | 204 |
| 10.2.5 | 带 LIKE 关键字的模糊查询 | 207 |
| 10.3 | 排序数据记录查询 | 213 |
| 10.3.1 | 按照单字段排序 | 213 |
| 10.3.2 | 按照多字段排序 | 215 |
| 10.4 | 限制数据记录查询数量 | 217 |
| 10.4.1 | 不指定初始位置 | 217 |
| 10.4.2 | 指定初始位置 | 219 |
| 10.5 | 统计函数和分组数据记录查询 | 221 |
| 10.5.1 | MySQL 支持的统计函数 | 222 |
| 10.5.2 | 关于统计函数注意点 | 227 |
| 10.5.3 | 分组数据查询——简单分组查询 | 228 |
| 10.5.4 | 分组数据查询——实现统计功能分组查询 | 230 |
| 10.5.5 | 分组数据查询——实现多个字段分组查询 | 231 |
| 10.5.6 | 分组数据查询——实现 HAVING 子句限定分组查询 | 234 |
| 10.6 | 小结 | 235 |

第 11 章 多表数据记录查询

| | | |
|--------|--------------------------------|-----|
| 11.1 | 关系数据操作 | 237 |
| 11.1.1 | 并 (UNION) | 237 |
| 11.1.2 | 笛卡儿积 (CARTESIAN PRODUCT) | 238 |
| 11.1.3 | 内连接 (INNER JOIN) | 239 |
| 11.1.4 | 外连接 (OUTER JOIN) | 241 |
| 11.2 | 内连接查询 | 243 |
| 11.2.1 | 自连接 | 244 |
| 11.2.2 | 等值连接 | 246 |
| 11.2.3 | 不等连接 | 250 |
| 11.3 | 外连接查询 | 252 |
| 11.3.1 | 左外连接 | 253 |
| 11.3.2 | 右外连接 | 254 |
| 11.4 | 合并查询数据记录 | 256 |
| 11.5 | 子查询 | 258 |

| | | |
|--------|-------------------------|-----|
| 11.5.1 | 为什么使用子查询 | 259 |
| 11.5.2 | 返回结果为单行单列和单行多列子查询 | 259 |
| 11.5.3 | 返回结果为多行单列子查询 | 262 |
| 11.5.4 | 返回结果为多行多列子查询 | 268 |
| 11.6 | 小结 | 270 |

第 12 章 使用 MySQL 运算符

| | | |
|--------|-------------------|-----|
| 12.1 | 为什么要使用运算符 | 271 |
| 12.2 | 使用算术运算符 | 271 |
| 12.3 | 使用比较运算符 | 273 |
| 12.3.1 | 常用比较运算符 | 274 |
| 12.3.2 | 实现特殊功能比较运算符 | 276 |
| 12.4 | 使用逻辑运算符 | 280 |
| 12.5 | 使用位运算符 | 283 |
| 12.6 | 小结 | 286 |

第 13 章 使用 MySQL 常用函数

| | | |
|--------|--|-----|
| 13.1 | 使用字符串函数 | 287 |
| 13.1.1 | 合并字符串函数 CONCAT()和 CONCAT_WS() | 288 |
| 13.1.2 | 比较字符串大小函数 STRCMP() | 290 |
| 13.1.3 | 获取字符串长度函数 LENGTH()和字符数函数 CHAR_LENGTH() | 290 |
| 13.1.4 | 实现字母大小写转换函数 UPPER()和字符数函数 LOWER() | 292 |
| 13.1.5 | 查找字符串 | 293 |
| 13.1.6 | 从现有字符串中截取子字符串 | 296 |
| 13.1.7 | 去除字符串的首尾空格 | 297 |
| 13.1.8 | 替换字符串 | 299 |
| 13.2 | 使用数值函数 | 301 |
| 13.2.1 | 获取随机数 | 301 |
| 13.2.2 | 获取整数的函数 | 302 |
| 13.2.3 | 截取数值函数 | 303 |
| 13.2.4 | 四舍五入函数 | 303 |
| 13.3 | 使用日期和时间函数 | 304 |
| 13.3.1 | 获取当前日期和时间的函数 | 304 |
| 13.3.2 | 通过各种方式显示日期和时间 | 306 |
| 13.3.3 | 获取日期和时间各部分值 | 308 |
| 13.3.4 | 计算日期和时间的函数 | 311 |

MySQL 数据库应用从入门到精通

| | |
|---|-----|
| 13.4 使用系统信息函数 | 314 |
| 13.4.1 获取 MySQL 系统信息 | 315 |
| 13.4.2 获取 AUTO_INCREMENT 约束的最后 ID 值 | 315 |
| 13.4.3 其他函数 | 316 |
| 13.5 小结 | 317 |

第 14 章 存储过程和函数的操作

| | |
|---|-----|
| 14.1 为什么使用存储过程和函数 | 318 |
| 14.2 创建存储过程和函数 | 319 |
| 14.2.1 创建存储过程语法形式 | 319 |
| 14.2.2 创建函数语法形式 | 320 |
| 14.2.3 创建简单的存储过程和函数 | 321 |
| 14.2.4 通过工具来创建存储过程和函数 | 322 |
| 14.3 关于存储过程和函数的表达式 | 326 |
| 14.3.1 操作变量 | 326 |
| 14.3.2 操作条件 | 327 |
| 14.3.3 使用游标 | 328 |
| 14.3.4 使用流程控制 | 332 |
| 14.4 查看存储过程和函数 | 333 |
| 14.4.1 通过 SHOW PROCEDURE STATUS 语句查看存储过程状态信息 | 334 |
| 14.4.2 通过 SHOW FUNCTION STATUS 语句查看函数状态信息 | 335 |
| 14.4.3 通过查看系统表 information_schema.routines 实现查看存储过程和函数的信息 | 336 |
| 14.4.4 通过 SHOW CREATE PROCEDURE 语句查看存储过程定义信息 | 339 |
| 14.4.5 通过 SHOW CREATE FUNCTION 语句查看函数定义信息 | 340 |
| 14.4.6 通过工具来查看存储过程和函数 | 341 |
| 14.5 修改存储过程和函数 | 343 |
| 14.5.1 修改存储过程 | 343 |
| 14.5.2 修改函数 | 344 |
| 14.6 删除存储过程和函数 | 345 |
| 14.6.1 通过 DROP 语句删除存储过程 | 345 |
| 14.6.2 通过 DROP FUNCTION 语句删除函数 | 346 |
| 14.6.3 通过工具来删除存储过程和函数 | 346 |
| 14.7 小结 | 349 |

第 15 章 MySQL 事务

| | |
|-----------------|-----|
| 15.1 事务概述 | 350 |
|-----------------|-----|

| | | |
|--------|----------------------------------|-----|
| 15.2 | MySQL 事务控制语句 | 351 |
| 15.3 | MySQL 事务隔离级别 | 353 |
| 15.3.1 | READ-UNCOMMITTED (读取未提交内容) | 354 |
| 15.3.2 | READ-COMMITTED (读取提交内容) | 355 |
| 15.3.3 | REPEATABLE-READ (可重读) | 356 |
| 15.3.4 | Serializable (可串行化) | 357 |
| 15.4 | InnoDB 锁机制 | 360 |
| 15.4.1 | 锁的类型 | 360 |
| 15.4.2 | 锁粒度 | 361 |
| 15.5 | 小结 | 364 |

第 3 篇 MySQL 数据库管理篇

第 16 章 MySQL 安全性机制

| | | |
|--------|----------------------------------|-----|
| 16.1 | MySQL 软件所提供的权限 | 366 |
| 16.1.1 | 系统表 mysql.user | 366 |
| 16.1.2 | 系统表 mysql.db 和 mysql.host | 368 |
| 16.1.3 | 其他权限表 | 369 |
| 16.2 | MySQL 软件所提供的用户机制 | 372 |
| 16.2.1 | 登录和退出 MySQL 软件的完整命令 | 373 |
| 16.2.2 | 创建普通用户账户 | 375 |
| 16.2.3 | 利用拥有超级权限用户 root 修改用户账户密码 | 378 |
| 16.2.4 | 利用拥有超级权限用户 root 修改普通用户账户密码 | 382 |
| 16.2.5 | 删除普通用户账户 | 388 |
| 16.3 | 权限管理 | 390 |
| 16.3.1 | 对用户进行授权 | 390 |
| 16.3.2 | 查看用户所拥有权限 | 393 |
| 16.3.3 | 收回用户所拥有权限 | 394 |
| 16.4 | 小结 | 397 |

第 17 章 MySQL 日志管理

| | | |
|--------|----------------------|-----|
| 17.1 | MySQL 软件所支持的日志 | 398 |
| 17.2 | 操作二进制日志 | 399 |
| 17.2.1 | 启动二进制日志 | 399 |
| 17.2.2 | 查看二进制日志 | 400 |
| 17.2.3 | 停止二进制日志 | 402 |

MySQL 数据库应用从入门到精通

| | | |
|--------|----------------|-----|
| 17.2.4 | 删除二进制日志 | 402 |
| 17.3 | 操作错误日志 | 404 |
| 17.3.1 | 启动错误日志 | 404 |
| 17.3.2 | 查看错误日志 | 405 |
| 17.3.3 | 删除错误日志 | 405 |
| 17.4 | 通用查询日志 | 405 |
| 17.4.1 | 启动通用查询日志 | 406 |
| 17.4.2 | 查看通用查询日志 | 406 |
| 17.4.3 | 停止通用查询日志 | 407 |
| 17.4.4 | 删除通用查询日志 | 407 |
| 17.5 | 慢查询日志 | 408 |
| 17.5.1 | 启动慢查询日志 | 408 |
| 17.5.2 | 查看慢查询日志 | 409 |
| 17.5.3 | 分析慢查询日志 | 409 |
| 17.5.4 | 停止慢查询日志 | 410 |
| 17.5.5 | 删除慢查询日志 | 410 |
| 17.6 | 小结 | 410 |

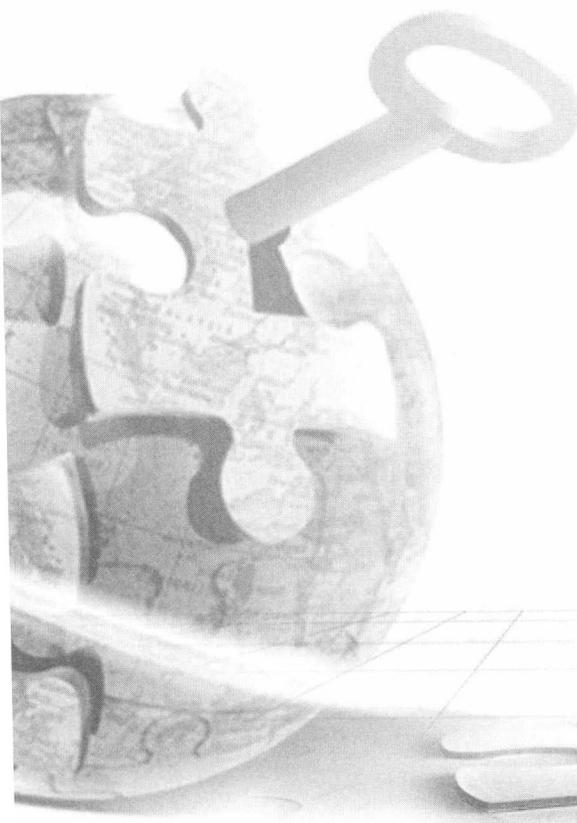
第 18 章 MySQL 数据库维护和性能提高

| | | |
|--------|------------------------------|-----|
| 18.1 | MySQL 数据库维护 | 411 |
| 18.1.1 | 通过复制数据文件实现数据备份 | 411 |
| 18.1.2 | 通过命令 mysqldump 实现数据备份 | 412 |
| 18.1.3 | 通过复制数据文件实现数据还原 | 417 |
| 18.1.4 | 通过命令 mysql 实现数据还原 | 417 |
| 18.1.5 | 实现数据库表导出到文本文件 | 420 |
| 18.1.6 | 实现文本文件导入到数据库表 | 424 |
| 18.1.7 | 数据库迁移 | 427 |
| 18.2 | 通过 SQLyog 客户端软件实现数据库维护 | 428 |
| 18.2.1 | 通过 SQLyog 客户端软件进行备份操作 | 428 |
| 18.2.2 | 通过 SQLyog 客户端软件进行还原操作 | 432 |
| 18.2.3 | 通过 SQLyog 客户端软件进行导出操作 | 433 |
| 18.2.4 | 通过 SQLyog 客户端软件进行导入操作 | 435 |
| 18.3 | MySQL 数据库性能优化 | 436 |
| 18.4 | 小结 | 437 |

第 19 章 使用 PowerDesigner 进行数据库设计

| | |
|--|-----|
| 19.1 下载和安装 PowerDesigner 软件 | 438 |
| 19.1.1 下载 PowerDesigner 软件 | 438 |
| 19.1.2 安装 PowerDesigner 软件 | 440 |
| 19.2 使用 PowerDesigner 软件进行数据库设计 | 442 |
| 19.2.1 利用 PowerDesigner 软件设计概念数据模型 | 442 |
| 19.2.2 利用 PowerDesigner 软件设计物理数据模型 | 451 |
| 19.2.3 利用 PowerDesigner 软件创建数据库脚本 | 452 |
| 19.3 小结 | 455 |

第 1 篇 MySQL 数据库基础篇



本篇主要介绍了 MySQL 数据库涉及的基础概念和安装过程。首先介绍了数据库涉及的基础概念，分别为数据管理技术发展阶段、数据库技术经历阶段、数据库管理系统提供的功能、SQL 语言和常见数据库管理系统；然后详细讲解了 MySQL 数据库的安装和配置过程。

第 1 章 数据库概述

数据管理技术经历多年的发展，已经发展到数据库系统阶段，在该阶段会把数据存储到数据库（DataBase，DB）中，即数据库相当于存储数据仓库。为了便于用户组织和管理数据，其还专门提供了数据库管理系统（DataBase Management System，DBMS），可以有效管理存储在数据库中的数据。本书所要讲解的 MySQL 软件，就是一种非常优秀的数据库管理系统。

通过本章的学习，可以掌握如下内容：

- 数据管理技术；
- 数据库相关概念和知识；
- MySQL 数据库基础概念和知识。

1.1 关于数据库的基本概念

在目前阶段，如果要存储和管理数据，则离不开数据库。当数据存储到数据库后，就会通过数据库管理系统对这些数据进行组织和管理。本节将详细介绍学习数据库的基本概念。

1.1.1 数据管理技术的发展阶段

所谓数据管理，是指对各种数据进行分类、组织、编码、存储、检索和维护。发展到现在，数据管理技术经历了三个阶段，分别为人工管理阶段、文件系统阶段和数据库系统阶段。

1. 人工管理阶段

20 世纪 50 年代中期以前，由于计算机中硬件还没有像现在这样的磁盘、软件没有专门管理数据的软件，所以计算机只局限于科学技术方面，数据则由计算和处理它的程序自行携带。该时期被称为人工管理阶段。

人工管理阶段的特点如下：

- 数据不能长期保存。
- 程序本身管理数据。
- 数据不能共享。
- 数据不具有独立性。

2. 文件系统阶段

随着技术的发展，在 20 世纪 50 年代后期到 20 世纪 60 年代中期，计算机不仅应用于科学技术，而且开始用于管理。在该时期由于计算机硬件出现了磁盘，计算机软件出现了高级语言和操作系统，因此程序和数据有了一定独立性，出现了程序文件和数据文件，这就是所谓的文件系统阶段。

文件系统阶段的特点如下：

- 数据可以长期保存。
- 数据由文件系统来管理。
- 数据冗余大，共享性差。
- 数据独立性差。

3. 数据库系统阶段

随着网络技术的发展，计算机软/硬件功能的进步，在 20 世纪 60 年代后期，计算机可以管理规模巨大的数据，这时如果计算机还使用文件系统来管理数据，则远远不能满足当时各种应用需求，于是出现了数据库技术，特别是关系型数据库技术。该阶段就是所谓的数据库系统阶段。

数据库系统阶段的特点如下：

- 数据实现结构化。
- 数据实现了共享性。
- 数据独立性强。
- 数据粒度变小。

1.1.2 数据库系统阶段涉及的概念

到目前为止，处理数据的技术仍然处于数据库系统阶段。在该阶段处理数据时，经常会涉及各种概念：数据库、数据库管理系统和数据库系统。同时如果想完全掌握数据库系统阶段的数据处理技术，也必须熟悉和掌握这些概念。

数据库（DataBase，DB）是指长期保存在计算机的存储设备上，按照一定规则组织起来，可以被各种用户或应用共享的数据集合。

数据库管理系统（DataBase Management System，DBMS）是指一种操作和管理数据库的大型软件，用于建立、使用和维护数据库，对数据库进行统一管理 and 控制，以保证数据库的安全性和完整性。用户通过数据库管理系统访问数据库中的数据。当前比较流行和常用的数据库管理系统有 Oracle、MySQL、SQL Server 和 DB2 等。

数据库系统（DataBase System，DBS）是指在计算机系统中引入数据库后的系统，通常由计算机硬件、软件、数据库管理系统和数据管理员组成。

在通常情况下，经常会用数据库来表示它们使用的数据库软件。这经常会引起混淆，确切地说，数据库软件应该为数据库管理系统，数据库是通过数据库管理系统创建和操作的容器。

1.1.3 数据库技术经历的阶段

在数据库系统管理数据阶段，随着时间的推移，又经历了三个技术阶段，分别为：层次数据库和网状数据库技术阶段、关系数据库技术阶段和后关系数据库技术阶段。本节将详细介绍这些数据库技术。

1. 层次数据库和网状数据库技术阶段

层次数据库代表为 IMS，1968 年 IBM 为阿波罗 11 号飞船顺利登月提供了重要保障。网状数据库代表为 IDS，其为 1961 年美国通用电气公司使用的数据库。该阶段的数据库技术，支持三级模式结构，用指针来表示数据之间的联系；数据定义语言和数据操纵语言相对独立。

2. 关系数据库技术阶段

关系数据库技术的出现，可以说是数据库系统管理数据的一个里程碑。该技术具有严格的数学理论，概念简单清晰，便于使用，受到众多数据库管理系统厂家的追捧。关系数据库技术的代表数据库管理系统为 Oracle、DB2、SQL Server、MySQL、SyBase 和 InFormix 等。

3. 后关系数据库技术阶段

由于关系数据库在数据模型、性能、扩展伸缩性上存在问题，所以出现了一些后关系数据库的技术，例如，面向对象数据库技术（ORDBMS）和结构化数据库技术（NOSQL）。

对于 ORDBMS 技术，虽然其尚未完全成熟，但是能很好地支持对数据和对象的管理，能够很好地和面向对象设计技术相融合。而 NOSQL 技术，则打破了关系型数据库与 ACID 理论相互统一的局面。利用 NOSQL 技术存储数据时，不需要固定的表结构，因此在大型数据存取上具备关系型数据库无法比拟的性能优势。

1.1.4 数据库管理系统提供的功能

数据库管理系统提供许多功能，可以通过 SQL（结构化查询语言）来定义和操作数据，维护数据的完整性和安全性，以及进行各种数据库的管理等。那么数据库管理系统所支持的 SQL 提供哪些功能呢？

1. 数据定义语言（Data Definition Language, DDL）

数据库管理系统提供了数据定义语言定义数据库涉及各种对象，定义数据的完整性约束、保密限制等约束。

2. 数据操作语言（Data Manipulation Language, DML）

数据库管理系统提供了数据操作语言实现对数据的操作。基本的数据操作有两类：检索（查询）和更新（插入、删除和更新）。

3. 数据控制语言（Data Control Language, DCL）

数据库管理系统提供了数据控制语言实现对数据库的控制，包含数据完整性控制、数据安全性控制和数据库的恢复等。

1.1.5 什么是 SQL

1.1.4 节所涉及的 SQL，其发音为字母 S-Q-L 或 sequel，是 Structure Query Language（结构化查询语言）的缩写，是目前广泛使用的关系数据库标准语言。该语言由 IBM 在 20 世纪 70 年代开发出来，被作为 IBM 关系数据库 System R 的原型关系语言，实现关系数据库中信息的检索。

由于 SQL 简单易学、功能丰富和使用灵活，因此受到众多人的追捧。经过不断的发展、完善和扩充，SQL 被美国国家标准局（ANSI）确定为关系型数据库语言的美国标准，后来又被国际标准化组织（ISO）采纳为关系数据库语言的国际标准。各种 SQL 标准的出台，使得所有数据库生产厂家

都推出了各自支持 SQL 的数据库管理系统，而本书所介绍的 MySQL 数据库也实现了对 SQL 的支持。

SQL 具有数据库管理系统的所有功能，主要功能如下：数据定义语言、数据操作语言和数据控制语言。

SQL 具有如下优点：

- SQL 不是某个特定数据库供应商专有的语言。几乎所有重要的数据库管理系统都支持 SQL，所以只要学习了 SQL 就能与所有数据库进行交互。
- SQL 简单易学，该语言的语句都是由描述性很强的英语单词组成，而且这些单词的数目不多。
- SQL 高度非过程化，即用 SQL 进行数据库操作，只需指出“做什么”，无须指明“怎么做”，存取路径的选择和操作的执行由数据库管理系统自动完成。

1.2 MySQL 数据库管理系统

随着时间的推移，开源数据库管理系统逐渐流行起来。开源数据库管理系统之所以能在中低端应用中占据很大的市场份额，是因为开源数据库具有免费使用、配置简单、稳定性好、性能优良的特点。本书所介绍的 MySQL 数据库管理系统正是开源数据库中的杰出代表，为了便于讲解，后面将用 MySQL 代替 MySQL 数据库管理系统。

1.2.1 MySQL 与开源文化

所谓“开源”，就是开放资源（Open Source）的意思。不过在程序界更多人习惯理解为“开放源代码”的意思。开放源代码运动起源于自由软件和黑客文化，最早来自于 1997 年在加州召开的一个研讨会，参加研讨会的有一些黑客和程序员，也有来自 Linux 国际协会的人员。在此会议上通过了一个新的术语“开源”。1998 年 2 月，网景公司正式宣布其发布的 Navigator 浏览器的源代码，这一事件成为开源软件发展历史的转折点。

开源即是自由的化身，提倡一种公开的、自由的精神。软件开源的发展历程，为软件行业及非软件行业带来了巨大的参考价值。虽然获取开放软件的源码是免费的，但是对源码的使用、修改却需要遵循该开源软件所作的许可声明。开源软件常用的许可证方式包括 BSD（Berkeley Software Distribution）、Apache Licence、GPL（General Public License）等，其中 GNU 的 GPL 为最常见的许可证之一，为许多开源软件所采用。

在计算机发展的早期阶段，软件几乎都是开放的，任何人使用软件的同时都可以查看软件的源代码，或者根据自己的需要去修改它。在程序员的社团中大家互相分享软件，共同提高知识水平。这种自由的风气给大家带来了欢乐，也带来了进步。在开源文化的强力带动下，产生了强大的开源操作系统 Linux，其他还有 Apache 服务器、Perl 程序语言、MySQL 数据库、Mozilla 浏览器等。

1.2.2 MySQL 发展历史

MySQL 的历史最早可以追溯到 1979 年，Monty Widenius 用 BASIC 设计了一个报表工具，过了不久，又将此工具使用 C 语言重写，移植到 UNIX 平台，当时只是一个底层的面向报表的存储引擎。

这个工具叫作 Unireg。

1985 年，三个瑞典人 David Axmark、Allan Larsson 和 Michael Widenius 成立了一家公司，这就是 MySQLAB 的前身，这个公司最初并不是为了开发数据库产品，而是在实现想法的过程中需要一个数据库并希望能够使用开源的产品。但在当时并没有一个合适的选择。因此自己设计了一个利用索引顺序存取数据的方法，也就是 ISAM (Indexed Sequential Access Method) 存储引擎核心算法的前身。此软件以创始人之一 Michael Widenius 女儿 My 的名字命名。MySQL 的 Logo 为海豚标志，如图 1.1 所示，海豚代表了速度、动力、精确等 MySQL 所拥有的特性。Logo 中海豚名字叫“sakila”，是由来自非洲斯威士兰的开源软件开发者 Ambrose Twebaze 提供的。根据 Ambrose 所说，Sakila 来自一种叫 SiSwati 的斯威士兰方言，也是在 Ambrose 的家乡乌干达附近坦桑尼亚的 Arusha 的一个小镇的名字。



图 1.1 MySQL 数据库 Logo

MySQL 是一款免费开源、小型、关系型数据库管理系统。随着该数据库功能的不断完善、性能的不提高，可靠性不断增强。2000 年 4 月，MySQL 对旧的存储引擎进行了整理，命名为 MyISAM。2001 年，支持事务处理和行级锁存储引擎 InnoDB 被集成到 MySQL 发行版中，该版本集成了 MyISAM 与 InnoDB 存储引擎，MySQL 与 InnoDB 的正式结合版本是 4.0。2004 年 10 月，发布了经典的 4.1 版本。2005 年 10 月，发布了里程碑的一个版本，MySQL 5.0，在 5.0 中加入了游标，存储过程，触发器，视图和事务的支持。在 5.0 之后的版本里，MySQL 明确地表现出迈向高性能数据库的发展步伐。MySQL 公司于 2008 年 1 月 16 号被 SUN 公司收购，而在 2009 年 SUN 又被 Oracle 收购。MySQL 的发展前途一片光明。

MySQL 虽然是免费的，但与其他商业数据库一样，具有数据库系统的通用性，提供了数据的存取、增加、修改、删除或更加复杂的数据操作。同时 MySQL 是关系型的数据库系统，支持标准的结构化查询语言，同时 MySQL 为客户端提供了不同的程序接口和链接库，如 C、C++、Java、PHP 等。目前 MySQL 被广泛地应用在 Internet 上的中小型网站中。由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，因此许多中小型网站为了降低网站总体拥有成本而选择了 MySQL 作为网站数据库。

目前 MySQL 可以下载的最新版本为 MySQL 5.6 版本。在最新的 5.6 版本中，数据库的可扩展性、集成度以及查询性能都得到提升。新增功能包括实现全文搜索，开发者可以通过 InnoDB 存储引擎列表进行索引和搜索基于文本的信息；InnoDB 重写日志文件容量也增至 2TB，能够提升写密集型应用程序的负载性能；加速 MySQL 复制；提供新的编程接口，使用户可以将 MySQL 与新的和原有的应用程序以及数据存储无缝集成。MySQL5.1 是当前稳定并且使用广泛的发布系列。只针对漏洞修复重新发布，没有增加会影响稳定性的新功能。MySQL4.x 是旧的稳定发布系列。目前只有少量用户使用。

1.2.3 常见数据库管理系统

目前市场上比较流行的数据库管理系统产品主要是 Oracle、IBM、Microsoft 和 Sybase 等公司的产品，下面对常用的几种系统进行简要的介绍。

1. Oracle 数据库管理系统

Oracle 数据库管理系统被认为是业界目前比较成功的关系型数据库管理系统，由世界第二大软件供应商 Oracle 公司于 1983 年推出。Oracle 的数据库产品被认为是运行稳定、功能齐全、性能超群的贵族产品。这主要是因为该数据库产品在技术方面的遥遥领先，还有就是其着重于大型的企业数据库领域应用。该数据库管理系统的 logo 如图 1.2 所示。



图 1.2 Oracle 数据库 logo

2. DB2 数据库管理系统

DB2 数据库管理系统是一款支持多媒体、Web 关系型数据库管理系统，其功能不仅可以满足大中型公司的需要，而且可以灵活地服务于中小型电子商务解决方案。据统计，目前 DB2 数据库管理系统用户超过 6 000 万，分布于约 40 万家公司。

3. SQL Server 数据库管理系统

SQL Server 数据库管理系统是一款功能比较全面，效率比较高，可以作为大中型企业或单位的数据库管理系统，由世界第一大软件供应商 Microsoft 公司推出。该数据库管理系统继承了 Microsoft 软件产品的界面友好、易学易用的特点，与其他大型数据库管理系统产品相比，在操作性和交互性方面独树一帜。

4. PostgreSQL 数据库管理系统

PostgreSQL 数据库管理系统是一款最富特色的自由数据库管理系统，甚至也可以说是最强大的自由软件数据库管理系统。该数据库管理系统支持了目前世界上最丰富的数据类型，是自由软件数据库管理系统中唯一支持事务、子查询、多版本并行控制系统、数据完整性检查等特性的自由软件，该数据库管理系统的 logo，如图 1.3 所示。



图 1.3 PostgreSQL 数据库 logo

1.2.4 为什么要使用 MySQL 数据库

在关于数据库的开源软件中，最杰出的代表软件为 MySQL 数据库管理系统和 PostgreSQL 数据

库管理系统，那么为什么 MySQL 数据库管理系统独占鳌头，受到众多程序员的追捧呢？

为了弄清楚原因，需要通过两种数据库软件的追求目标来说起。根据专门机构的调查研究显示，许多数据库管理系统提供的功能特性，只有 40% 的功能被使用，而一些复杂的高级功能特性不仅会增加系统的复杂性，而且往往还会引起系统的性能问题。PostgreSQL 数据库管理系统是加州大学伯利分校以教学为目的开发的数据库系统，以追求功能实现的“完美”为首要目标。而 MySQL 数据库管理系统的开发者，在性能与标准的取舍上，一直坚持性能优先的原则，从不为了追求标准的符合性而牺牲性能。这就决定了 MySQL 数据库管理系统在性能方面远远优于 PostgreSQL 数据库管理系统，成为互联网行业非常流行的数据库软件之一，因为 Web 应用往往需要支持大量的数据和并发请求，性能常常是首要考虑因素。

1.3 小 结

本章主要介绍数据库的相关概念，分为数据库的基本概念和 MySQL 数据库管理系统。前者详细介绍了数据管理技术的发展阶段、数据库技术经历的阶段、数据库管理系统提供的功能和数据库管理系统所支持的语言——SQL。后者主要介绍了 MySQL 数据库管理系统，分别通过 MySQL 概念、常见数据库管理系统和为什么要使用 MySQL 数据库三方面进行讲解。

通过对本章的学习，读者不仅会掌握数据库的基本概念，而且还会对 MySQL 数据库管理系统有一定认识。

第 2 章 MySQL 安装和配置

MySQL 原来隶属于 MySQLAB 公司，该公司总部位于瑞典，其中公司名中的“AB”是瑞典语“aktiebolag”（股份公司）的首字母缩写。在 2008 年 1 月 16 日，MySQL 被 SUN 公司收购，而 2009 年 SUN 公司被世界第二大软件供应商 Oracle 公司收购。成为 Oracle 数据库的有益补充。

随着 MySQL 功能的不断完善，该数据库管理系统几乎支持所有的操作系统，同时也支持许多新的特性，这些都使得 MySQL 非常迅猛的发展，目前已经广泛应用在各个行业中。

通过本章的学习，可以掌握如下内容：

- 下载、安装和卸载 MySQL 软件
- 通过各种方式配置 MySQL 软件
- 启动和关闭服务
- 熟练使用 MySQL 客户端软件

2.1 下载和安装 MySQL 软件

对于不同的操作系统，MySQL 提供了相应的版本。本章将以 Windows 平台下的图形化安装包和免安装包为例，详细讲解 MySQL 的下载、安装、配置等过程。本节的测试环境是 32 位的 Windows 系统。

2.1.1 基于客户端——服务器（C/S）的数据库管理系统

到目前为止，市场上几乎所有的数据库管理系统都是基于客户端—服务器模式。基于该模式的数据库管理系统分为两个不同的部分，分别为服务器软件和客户端软件。服务器软件是负责所有数据访问和处理的一个软件，而关于数据添加、删除等所有请求都来自于客户端软件。

注意：客户端软件和服务器软件可能安装在两台计算机或一台计算机上。不管这套软件在不在相同的计算机上，客户端软件和服务器软件都可以进行通信，从而实现数据的相关操作。

本书所介绍的 MySQL 也是基于 C/S 模式，那么搭建 MySQL 环境需要哪些服务器软件和客户端软件呢？

- 服务器端软件为 MySQL 数据库管理系统，可以在本地计算机上或者具有访问权限的远程服务器上安装该软件。

MySQL 数据库应用从入门到精通

- 客户端软件为可以操作 MySQL 服务器的软件。

2.1.2 MySQL 的各种版本

目前 MySQL 数据库按照用户群分为社区版（Community）和企业版（Enterprise），这两个版本的重要区别为：社区版可以自由下载而且完全免费，但是官方不提供任何技术支持，适用于大多数普通用户；企业版不仅不能免费下载而且还收费，但是该版本提供了更多的功能，可以享受完备的技术支持，适用于对数据库的功能和可靠性要求比较高的企业客户。

MySQL 版本更新非常快，现在主推（GA）的社区版本为 5.5.21。从 MySQL 版本 5 开始，开始支持触发器、视图、存储过程等数据库对象。常见的软件版本有 GA、RC、Alpha 和 Bean，它们的含义分别如下：

- GA（General Availability）：官方推崇广泛使用的版本。
- RC（Release Candidate）：候选版本意思，该版本是最接近正式版的版本。
- Alpha 和 Bean 都属于测试版本，其中 Alpha 是指内测版本，Bean 是指公测版本。

2.1.3 下载 MySQL 软件

MySQL 软件是完全网络化的跨平台关系型数据库系统，目前最新的版本为 MySQL5.5.21，可以通过下面的步骤来实现该平台的下载，具体如下：

- （1）首先访问下载 MySQL 的官方网站（<http://mysql.com/>），如图 2.1 所示。



图 2.1 MySQL 软件首页

- （2）打开 MySQL 软件首页后，单击 Downloads(GA)导航栏，就会进入关于 MySQL 产品的页面，如图 2.2 所示。在该页面中单击社区版（MySQL Community）中的 Download 超链接，进入下载页面。



图 2.2 下载社区版

(3) 首先在“Select Platform”下拉菜单中选择“Microsoft Windows”平台，然后单击“Windows (x86, 32-bit), MSI Installer”选项右边的 Download 按钮，下载 32 位的 MySQL 安装软件，如图 2.3 所示。

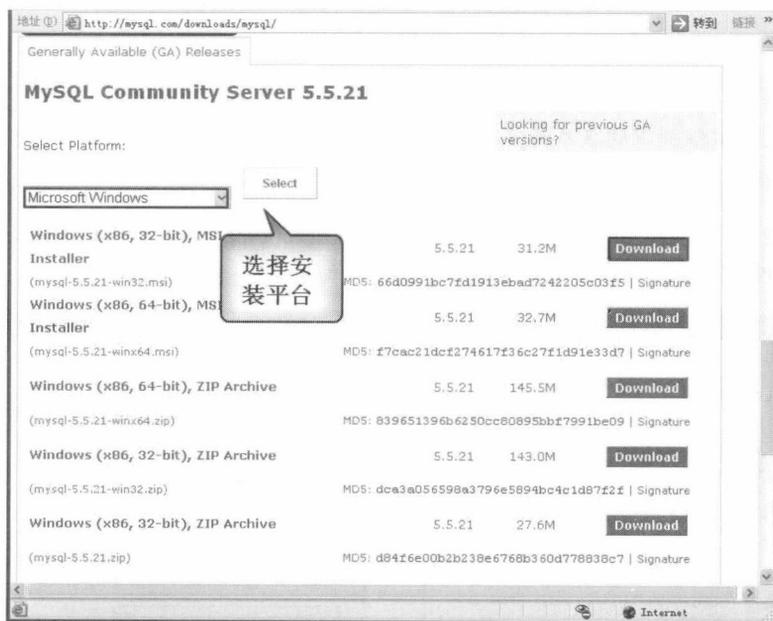


图 2.3 选择 MySQL 5.5.21

(4) 选择相应平台和版本的 MySQL 软件后，单击 Download 按钮后，进入选择镜像网站的页面，在该页面中可以选择任意一个网站进行下载，如图 2.4 所示。

下载完 MySQL 5.5.21 后，可安装该数据库软件。

MySQL 数据库应用从入门到精通

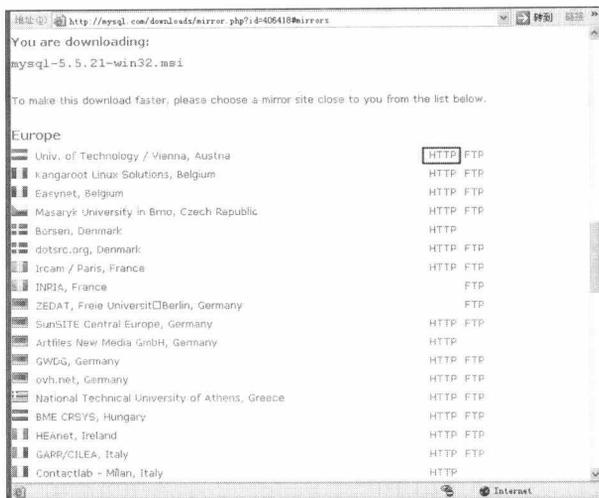


图 2.4 选择镜像网站

2.1.4 安装 MySQL 软件

2.1.3 节介绍了下载数据库 MySQL 软件的详细步骤，下载完数据库 MySQL 软件的安装程序后，即可开始安装该数据库。具体的安装步骤如下：

(1) 双击 MySQL 安装程序 (mysql-5.5.21-win32.msi)，接着使用 Windows Installer 开始安装过程，如图 2.5 所示。

注意：在安装文件 mysql-5.5.21-win32.msi 中，第一个“5”表示主版本号，第二个“5”表示发行级别，“21”表示该级别下的版本号，“win32”表示运行在 32 位的 Windows 操作系统下，“msi”表示安装文件的格式。

(2) 单击 Next 按钮后，在出现的对话框中选择“I accept the terms in the License Agreement”选项接受协议，如图 2.6 所示，然后单击 Next 按钮进入“Choose Setup Type”对话框。

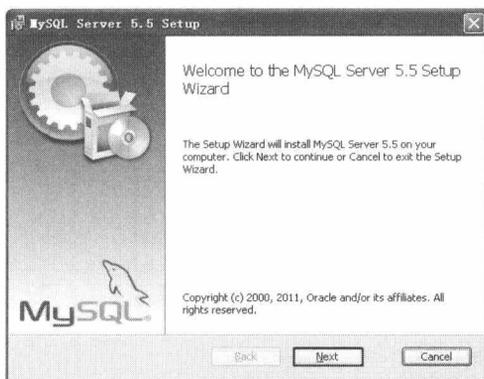


图 2.5 MySQL 欢迎界面

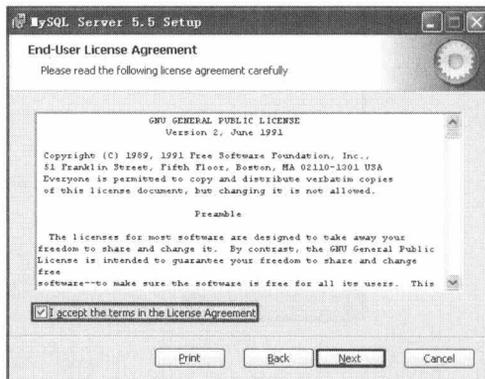


图 2.6 接受许可协议对话框

(3) 单击 Typical 按钮，如图 2.7 所示，进入“Ready to install MySQL Server 5.5”对话框，关于 MySQL 安装类型如下：

- Typical: 默认的安装类型。
- Complete: 完全的安装类型。
- Custom: 自定义的安装类型。

(4) 在“Ready to install MySQL Server 5.5”对话框中确认一下安装的信息，单击 Install 按钮，开始对 MySQL 软件的安装，如图 2.8 所示。

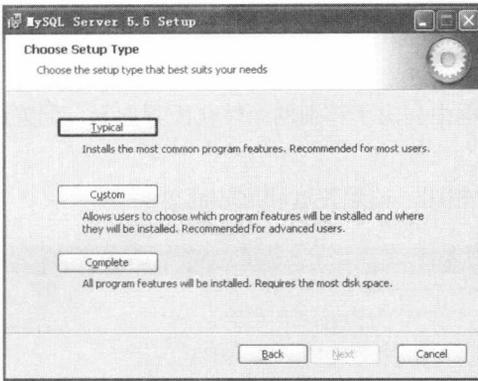


图 2.7 MySQL 安装类型对话框

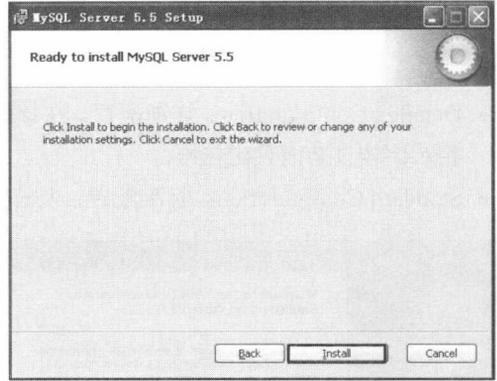


图 2.8 安装信息的确认

(5) 安装完成后，单击 Next 按钮，进入一系列说明界面，如图 2.9 和图 2.10 所示。

(6) 最后进入安装完成的界面，在该对话框中会询问是否现在进行配置，如图 2.11 所示。如果不想马上配置，可以取消选中 Launch the MySQL Instance Configuration Wizard 复选框，然后单击 Finish 按钮，完成对 MySQL 软件的安装。

至此，即可成功安装 MySQL 5.5.21 软件。

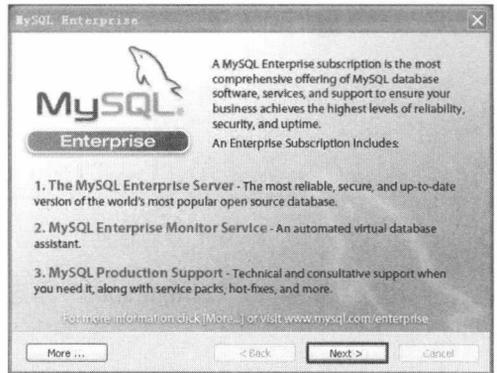


图 2.9 说明界面 1

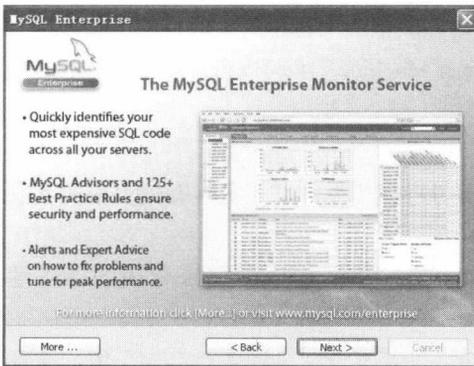


图 2.10 说明界面 2

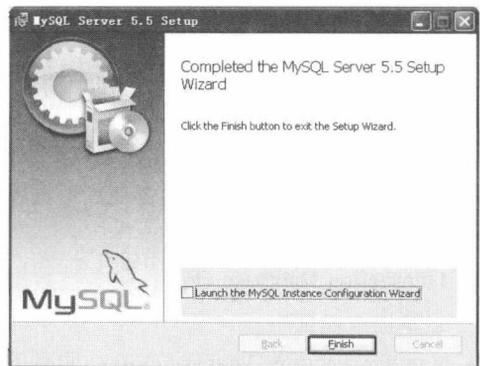


图 2.11 安装成功

2.1.5 图形化配置数据库服务器 MySQL

安装完 MySQL 后，默认会提供一个图形化的实例配置向导，可以帮助 MySQL 用户逐步进行实例参数的设置，具体步骤如下：

(1) 单击“开始”→“程序”→“MySQL”→“MySQL Server 5.5”→“MySQL Server Instance Configuration Wizard”菜单，进入图形化实例配置向导的欢迎界面，如图 2.12 所示。

(2) 单击 Next 按钮，进入选择配置类型界面，如图 2.13 所示。MySQL 提供了以下两种配置类型。

- **Detailed Configuration:** 详细配置，在该配置过程中列出了详细的个性化配置向导，配置过程相对比较复杂而且比较慢。
- **Standard Configuration:** 标准配置，与详细配置相比，该配置过程比较简单。

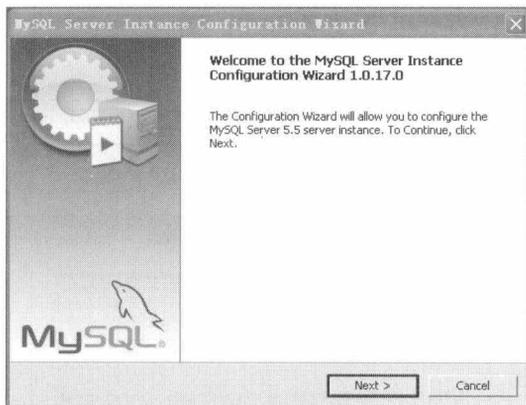


图 2.12 MySQL 欢迎界面

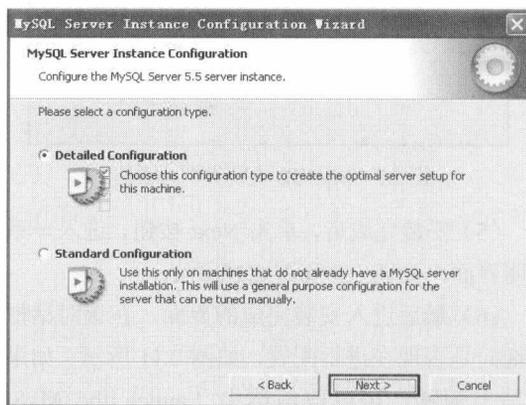


图 2.13 选择配置类型界面

(3) 选择“Detailed Configuration”单选按钮，然后单击 Next 按钮，进入选择应用类型界面，如图 2.14 所示。

MySQL 提供了以下 3 种应用类型供选择。

- **Developer Machine:** 开发机，该类型应用将会使用最小数量的内存。
- **Server Machine:** 服务器，该类型应用将会使用中等大小的内存。
- **Dedicated MySQL Server Machine:** 专用服务器，该类型应用将使用当前可用的最大内存。

(4) 在 MySQL 选择应用类型界面中选择“Developer Machine”单选按钮，然后单击 Next 按钮，进入选择用途类型界面，如图 2.15 所示。

MySQL 提供了以下 3 种用途类型数据库供选择。

- **Multifunctional Database:** 多功能数据库，对事务性存储引擎和非事务性存储引擎的存取速度都很快。
- **Transaction Database Only:** 事务性数据库，主要优化了事务性存储引擎，但是非事务性存储引擎也可以使用。
- **Non-Transaction Database Only:** 非事务性数据库，主要优化了非事务性存储引擎，注意事务性存储引擎不可以使用。

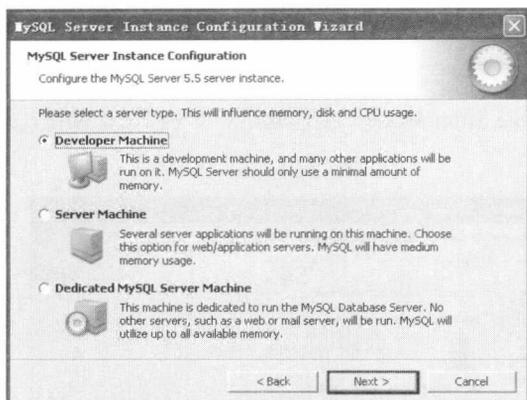


图 2.14 选择应用类型界面

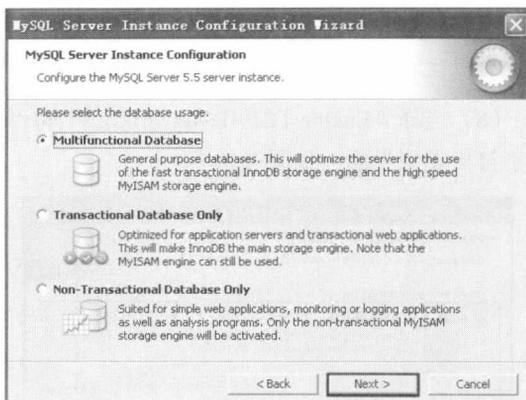


图 2.15 选择用途类型界面

(5) 选择“Multifunctional Database”单选按钮，然后单击 Next 按钮，进入 InnoDB 数据文件目录配置界面，如图 2.16 所示。

InnoDB 数据文件在数据库第一次启动时创建，默认创建在 MySQL 的安装目录下。用户可以根据实际的硬盘空间状况进行路径的选择。

(6) 在 InnoDB 数据文件目录配置界面中保留默认值，然后单击 Next 按钮，进入并发连接设置界面，如图 2.17 所示。

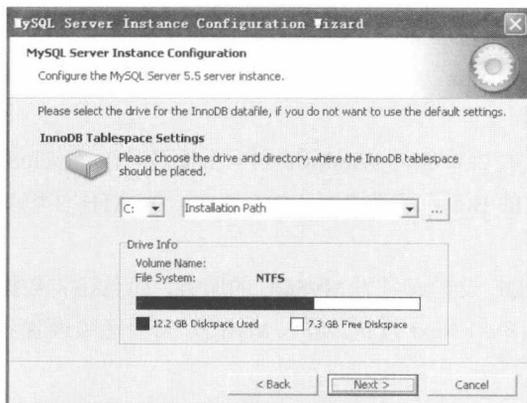


图 2.16 数据文件目录配置界面

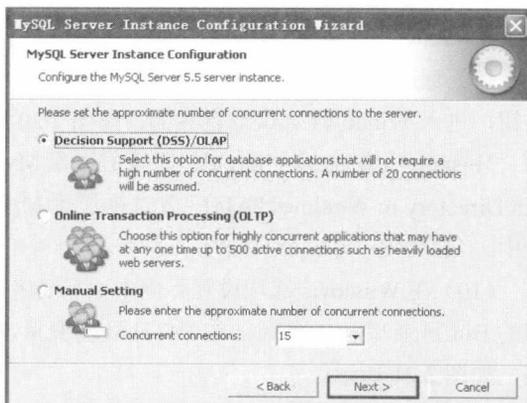


图 2.17 并发连接设置界面

MySQL 提供了以下 3 种类型并发处理供选择。

- Decision Support(DSS)/OLAP: 决策支持系统，设置并发连接数为 20。
- Online Transaction Processing(OLTP): 在线事物系统，设置并发连接数为 500。
- Manual Setting: 手工设置，可以手工设置并发连接数。

(7) 选择“Decision Support(DSS)/OLAP”单选按钮，然后单击 Next 按钮，进入网络选项设置界面，如图 2.18 所示。

“Enable TCP/IP Networking”复选框表示是否启动 TCP/IP 连接，而“Enable Strict Mode”复选框表示是否采用严格模式来启动服务。

MySQL 数据库应用从入门到精通

注意：如果 MySQL 安装在服务器上，一定要选择“Add firewall exception for this port”复选框，这样在同一网络内的用户可以访问该端口。

(8) 选择“Enable TCP/IP Networking”和“Enable Strict Mode”复选框后，单击 Next 按钮，进入字符集设置界面，如图 2.19 所示。

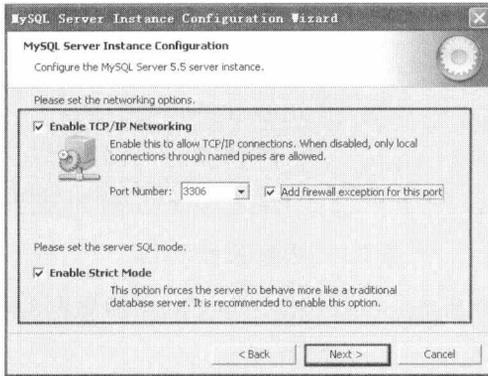


图 2.18 网络选项设置界面

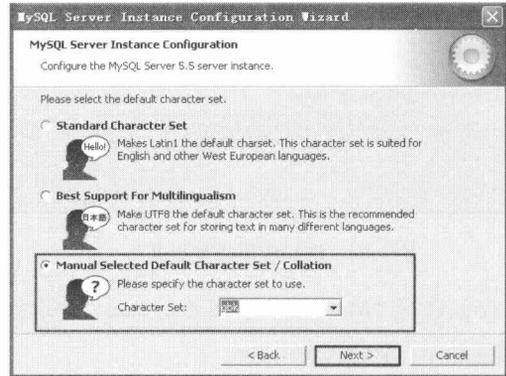


图 2.19 字符集设置界面

MySQL 提供了以下 3 种方式来设置字符集。

- Standard Character Set: 标准字符集，MySQL 提供的标准字符集默认为 Latin1。
- Best Support For Multilingualism: 支持多国语言最好的字符集，默认值为 UTF8。
- Manual Selected Default Character Set/Collation: 手动设置，可以手动设置字符集。

(9) 通过 Manual Selected Default Character Set/Collation 方式设置字符集为 gbk，然后单击 Next 按钮，进入 Windows 选项设置界面，如图 2.20 所示。

“Install As Windows Service”复选框设置 MySQL 是否将作为 Windows 的一个服务，而“Include Bin Directory in Windows PATH”复选框设置 MySQL 的 Bin 目录是否写入 Windows 的 PATH 环境变量中。

(10) 在 Windows 选项设置界面中，不仅将 MySQL 设置成名为 MySQL 的服务，而且还将数据库的 Bin 目录写入 Windows 的 PATH 环境变量，然后单击 Next 按钮，进入 MySQL 安全选项设置界面，如图 2.21 所示。



图 2.20 Windows 选项设置界面

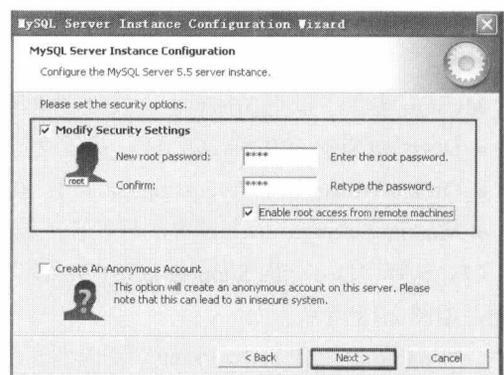


图 2.21 安全选项设置界面

在安全选项设置界面中有两个安全设置复选框，“Modify Security Settings”复选框确定是否修改默认用户 root 的密码，“Create An Anonymous Account”复选框确定是否创建一个匿名用户，在具体开发时，建议不要创建匿名用户，因为这样会给系统带来安全漏洞。

注意：如果 MySQL 安装在服务器上，需要选择“Enable root access from remote machines”复选框来设置可以让远程计算机通过用户 root 登录 MySQL。

(11) 在安全选项设置界面中设置用户 root 的密码为 root，单击 Next 按钮，进入准备执行界面，如图 2.22 所示。

(12) 确认设置没有问题后，单击 Execute 按钮后开始执行。执行成功后的界面如图 2.23 所示。最后单击 Finish 按钮，结束 MySQL 的全部配置。

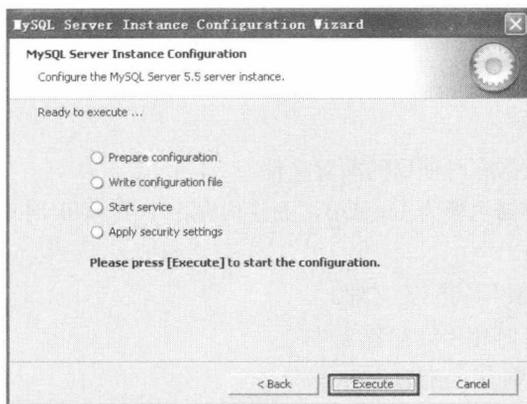


图 2.22 准备执行界面

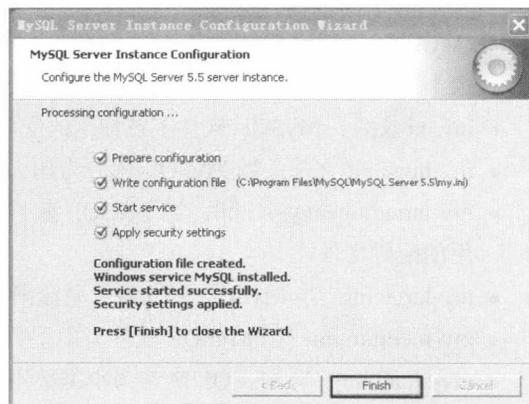


图 2.23 执行成功界面

至此，就可以成功配置 MySQL 5.5.21 数据库服务器。

2.1.6 手动配置数据库服务器 MySQL

通过图形化的实例配置向导来实现参数的设置，虽然简洁高效，但是实现起来不够灵活。本节将介绍一种比较灵活的配置方式——手动修改 MySQL 软件配置文件。

在手动修改 MySQL 配置文件之前，需要对 MySQL 安装后的文件有所了解。在 2.1.4 节中安装 MySQL 软件时，选择的是 Typical 类型，该类型下 MySQL 软件的安装文件在“C:\Program Files\MySQL\MySQL Server 5.5”目录下，而 MySQL 软件的数据库文件则安装在“C:\Documents and Settings\All Users\Application Data\MySQL\MySQL Server 5.5\data”目录下。

MySQL 软件的安装文件夹下会存在许多文件夹和文件，具体内容如图 2.24 所示，其中 4 个文件夹比较重要，其作用分别为：

- bin 文件夹，该文件夹下存放着可执行文件。
- include 文件夹，该文件夹下存放着头文件。
- lib 文件夹，该文件夹下存放着库文件。
- share 文件夹，该文件夹下存放着字符集、语言等信息。

MySQL 数据库应用从入门到精通



图 2.24 MySQL 软件安装文件

在 MySQL 软件安装文件夹下除了文件之外，还存在许多扩展名为.ini 的文件。不同名字的.ini 文件代表不同的意思。各个.ini 文件的含义如下。

- my.ini 文件：MySQL 软件正在使用的配置文件。
- my-huge.ini 文件：当 MySQL 软件为超大型数据库时使用的配置文件。
- my-innodb-heavy-4G.ini：当 MySQL 软件的存储引擎为 InnoDB，而且内存不小于 4GB 时使用的配置文件。
- my-large.ini：当 MySQL 软件为大型数据库时使用的配置文件。
- my-medium.ini：当 MySQL 软件为中型数据库时使用的配置文件。
- my-small.ini：当 MySQL 软件为小型数据库时使用的配置文件。
- my-template.ini：配置文件模板。

了解了关于 MySQL 软件的安装文件夹和文件后，只需修改 my.ini 配置文件中的内容即可实现修改数据库实例的参数，关于 my.ini 文件的内容如下：

```
# MySQL Server Instance Configuration File
# -----
# Generated by the MySQL Server Instance Configuration Wizard
#通过使用 MySQL 服务配置向导生成。
# Installation Instructions

#客户端参数配置
# CLIENT SECTION
# -----
[client]
#客户端通过 3306 端口号连接 MySQL 数据库服务器，如果需要修改端口号，可以直接修改。
Port=3306
[mysql]
#客户端的默认字符集。
default-character-set=gbk

#服务器端参数配置
# SERVER SECTION
# -----
[mysqld]
```

```
#服务器端的端口号。
port=3306
#MySQL 数据库服务器的安装目录
basedir="C:/Program Files/MySQL/MySQL Server 5.5/"
#MySQL 数据库数据文件的目录
datadir="C:/Documents and Settings/All Users/Application Data/MySQL/MySQL Server
5.5/Data/"
#MySQL 软件端的字符集
character-set-server=gbk
#MySQL 软件的存储引擎
default-storage-engine=INNODB
# Set the SQL mode to strict
#MySQL 软件的 SQL 模式
sql-mode="STRICT_TRANS_TABLES,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION"
#MySQL 软件的最大连接数
max_connections=100
#MySQL 软件的查询缓存
query_cache_size=0
# MySQL 软件中可以打开表的总数。
table_cache=256
# MySQL 软件内存中可以存储临时表的最大值。
tmp_table_size=9M
# MySQL 软件中可以保留的客户端连接线程数。
thread_cache_size=8
# MySQL 软件重建索引时允许的最大临时文件的大小。
myisam_max_sort_file_size=100G
# MySQL 软件重建索引时允许的最大缓存大小。
myisam_sort_buffer_size=17M
# MySQL 软件中最大关键字缓存大小。
key_buffer_size=10M
# MySQL 软件全扫描 MyISAM 表时的缓存大小。
read_buffer_size=64K
# MySQL 软件可以插入排序好数据的缓存大小
read_rnd_buffer_size=256K
# MySQL 软件用户排序时缓存大小
sort_buffer_size=256K

#关于 INNODB 存储引擎参数设置
*** INNODB Specific options ***
# 附加内存池大小
innodb_additional_mem_pool_size=2M
# 关于提交日志的时机
innodb_flush_log_at_trx_commit=1
# 存储日志数据的缓存区的大小
innodb_log_buffer_size=1M
# 缓存池中缓存区的大小
innodb_buffer_pool_size=17M
# 日记文件的大小
innodb_log_file_size=10M
# 允许线程的最大数
innodb_thread_concurrency=8
```

MySQL 数据库应用从入门到精通

上述文件是去除大量注释后的 my.ini 文件。通过修改 MySQL 配置文件来手动配置数据库服务器 MySQL 时，经常会修改的参数如下。

- default-character-set: 客户端字符集。
- character-set-server: 服务器端字符集。
- port: 客户端和服务器的端口号。
- default-storage-engine: MySQL 软件的存储引擎。

注意：如果修改了 MySQL 软件的配置文件，则必须重新启动 MySQL 服务，修改的内容才能生效。

与图形化配置向导相比，手动配置 MySQL 服务器的方式更简单、更灵活和更有效。

2.1.7 卸载 MySQL 软件

既然可以安装和配置 MySQL，那么有时还需要卸载 MySQL 软件。可是查看“开始”→“程序”→“MySQL”→“MySQL Server 5.5”菜单中却发现没有专门卸载 MySQL 的菜单命令，那么如何卸载 MySQL 软件呢？

实现卸载 MySQL 软件的具体步骤如下：

(1) 双击 MySQL 安装程序 (mysql-5.5.21-win32.msi)，使用 Windows Installer 开始卸载过程，如图 2.25 所示。

(2) 单击 Next 按钮后，在出现的对话框中单击 Remove 按钮进行卸载操作，该对话框中相关按钮的含义如下，如图 2.26 所示，进入“Ready to remove MySQL Server 5.5”对话框：

- Change: 修改 MySQL 数据库功能操作。
- Repair: 修复 MySQL 数据库操作。
- Remove: 卸载 MySQL 数据库操作。

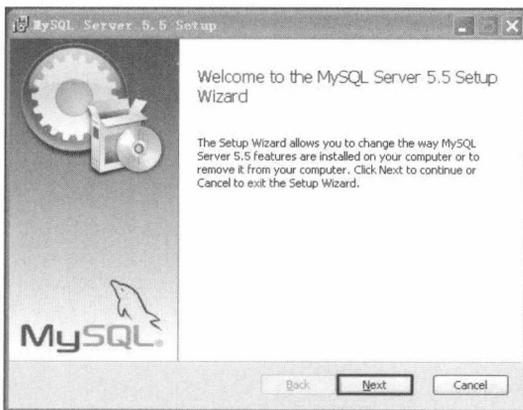


图 2.25 卸载欢迎界面

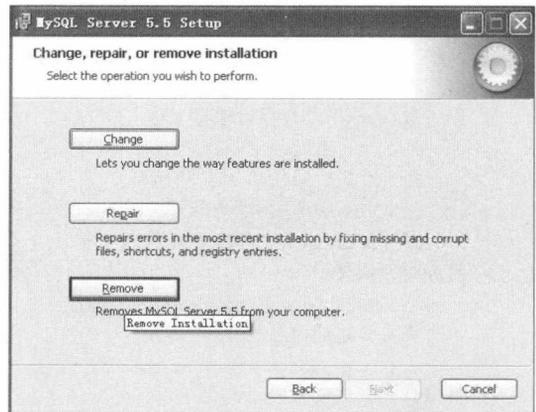


图 2.26 选择卸载 MySQL 操作

(3) 在“Ready to remove MySQL Server 5.5”对话框中确认卸载的信息，单击 Remove 按钮开始对 MySQL 软件的卸载，如图 2.27 所示。卸载完后出现确认卸载完成的对话框，在该对话框中单击 Finish 按钮，完成对 MySQL 软件的卸载，如图 2.28 所示。

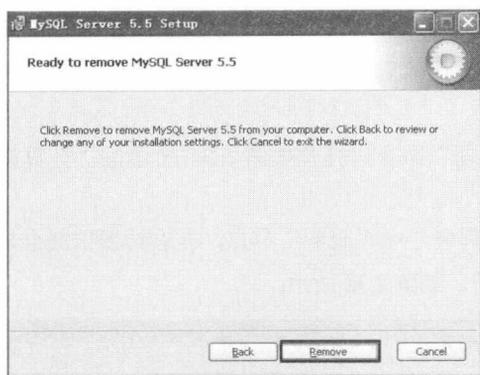


图 2.27 卸载信息确认

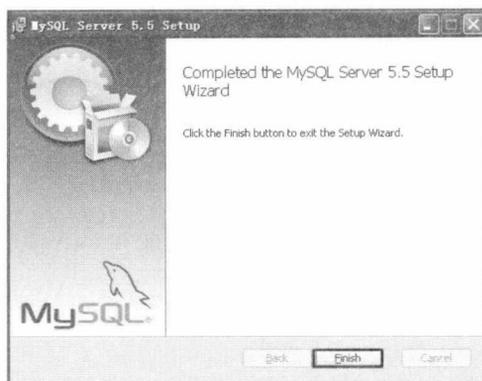


图 2.28 卸载成功

如果按照上述步骤执行，MySQL 软件即可成功卸载。这时就可以重新安装 MySQL 软件，不过在重新安装之前，一定要把卸载 MySQL 软件时没有删除干净的 MySQL 文件删除掉，即需要删除路径为“C:\Program Files\MySQL”的文件。

2.2 关于 MySQL 软件的常用操作

2.1 节介绍了如何下载、安装、配置和卸载 MySQL，本节将详细介绍 MySQL 软件的常用操作：启动和关闭 MySQL 服务。所谓 MySQL 服务是指一系列关于 MySQL 软件后台进程，只有启动了 MySQL 服务后，才可以连接 MySQL 软件进行操作。

2.2.1 通过图形界面启动和关闭 MySQL 服务

经常使用 Windows 操作系统的用户都知道，Windows 系统拥有许多服务，如果想启动和关闭服务，可以在服务窗口（见图 2.29）中实现。对于 MySQL 数据库服务器，只有启动 MySQL 服务，客户端才能登录到 MySQL 数据库。本节将详细介绍如何启动 MySQL 服务。

在 2.1.5 节配置 MySQL 的过程中，已经设置 MySQL 服务作为 Windows 系统的服务，服务名为 MySQL。同时选择“Launch the MySQL Server automatically”复选框，使 MySQL 服务为自动启动类型，如图 2.30 所示。

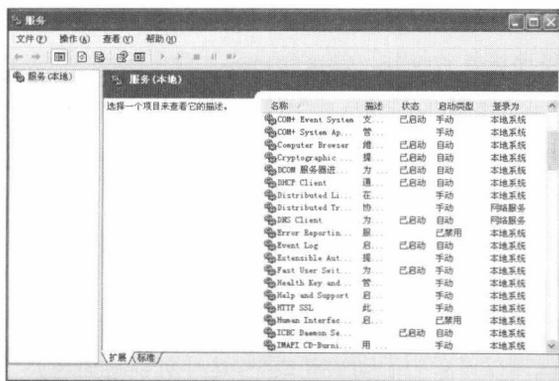


图 2.29 服务窗口

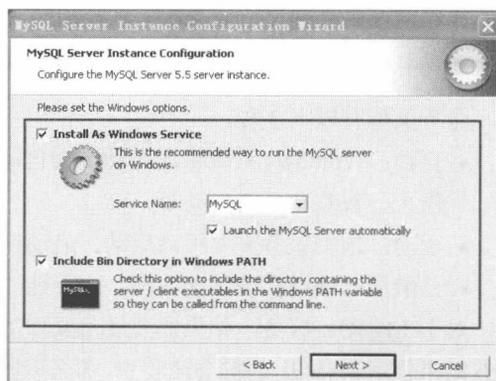


图 2.30 设置 MySQL 服务

MySQL 数据库应用从入门到精通

由于 Windows 系统的资源是非常有限，而每运行一个服务都会消耗一定资源，因此对于一些经常不用的服务最好关闭，当需要使用时再开启。那么如何启动和关闭 MySQL 服务呢？具体步骤如下：

(1) 右击“我的电脑”，在弹出的菜单中选择“管理”命令，打开如图 2.31 所示的“计算机管理”窗口。

(2) 选择“计算机管理(本地)”→“服务和应用程序”→“服务”节点，在右边窗口就会显示 Windows 系统的所有服务，其中包含名为 MySQL 的服务，如图 2.32 所示。

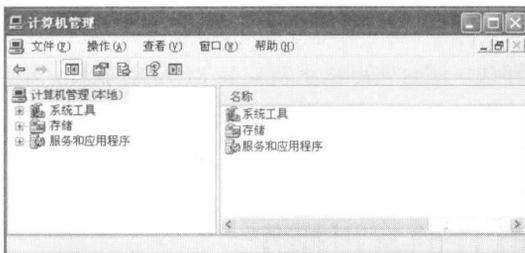


图 2.31 “计算机管理”窗口

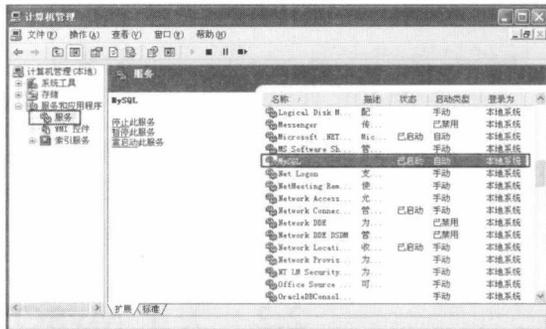


图 2.32 MySQL 服务

注意：对于 Windows 系统的服务也可以在“控制面板”中设置，具体菜单命令为“开始”→“控制面板”→“管理工具”→“服务”。

(3) 查看 MySQL 服务可以发现该服务已经处于启动状态，并且该服务的类型为“自动”。如果想修改 MySQL 服务的状态，可以单击“计算机管理”工具栏中的相应按钮。其中  按钮可以实现启动服务功能、 按钮可以实现停止服务功能、 按钮可以实现暂停服务功能、 按钮可以实现重新启动服务功能，如图 2.33 所示。

(4) 对于 MySQL 服务，由于不是系统自带的服务，所以除了需要在需要时启动外，还需要设置启动类型为手动类型。在具体设置时，需要右击 MySQL 服务，然后选择“属性”命令，弹出“MySQL 的属性(本地计算机)”对话框，如图 2.34 所示。在该对话框中不仅可以设置服务类型，而且还可以操作服务。

注意：如果修改了“MySQL 的属性”对话框中的相应内容，一定要先单击“应用”按钮，然后才可以单击“确定”按钮关闭该对话框。只有经过这样的操作，修改的内容才能生效。

服务类型有以下 3 种。

- 自动：Windows 系统每次启动都会自动启动该类型服务，服务启动后可以手动将状态修改为停止、暂停和重新启动等。
- 手动：该类服务需要手动启动，启动后可以改变服务状态，如停止、暂停等。
- 已禁用：该类服务不能启动，也不能改变服务状态。

对于 MySQL 服务，如果需要经常操作 MySQL 软件，那么该服务最好设置为自动启动类型。当然如果使用 MySQL 的机会比较少，那么该服务就可以设置为手动启动类型，这样可以避免 MySQL 服务长时间占用系统资源。



图 2.33 操作服务状态

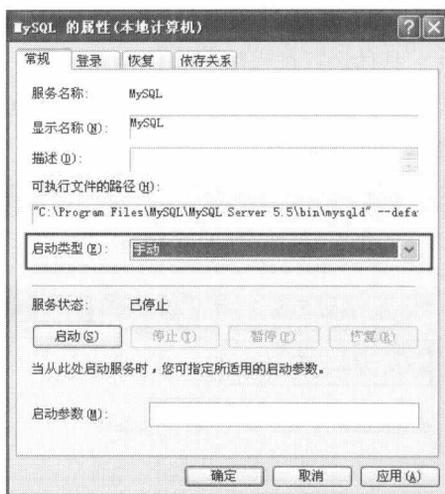


图 2.34 设置 MySQL 服务类型

2.2.2 通过 DOS 窗口启动和关闭 MySQL 服务

经常使用 Windows 操作系统的用户都知道，除了可以通过图形界面来启动和关闭服务外，还可以通过 DOS 窗口来启动和关闭服务。同理，关于 MySQL 软件的服务 MySQL，也可以通过 DOS 窗口来启动和关闭，具体步骤如下：

(1) 选择“开始”→“运行”命令，打开“运行”对话框，然后在“打开”文本框中输入“cmd”，打开 DOS 窗口，如图 2.35 所示。

(2) 在 DOS 窗口中，如果想查看 Windows 系统已经启动的服务，可以通过如下命令来实现：

```
net start
```

具体运行过程如图 2.36 所示，可以发现关于 MySQL 软件的服务 MySQL 已经启动。

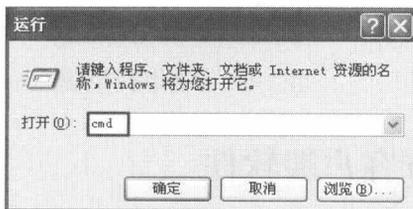


图 2.35 运行对话框



图 2.36 查看已经启动的服务

(3) 如果 MySQL 软件的服务 MySQL 已经启动，可以通过命令来关闭 MySQL 服务，具体命令内容如下：

MySQL 数据库应用从入门到精通

```
net stop MySQL
```

具体运行过程如图 2.37 所示。

(4) 如果 MySQL 软件的服务 MySQL 是处于关闭状态，可以通过命令来启动 MySQL 服务，具体命令内容如下：

```
net start MySQL
```

具体运行过程如图 2.38 所示。

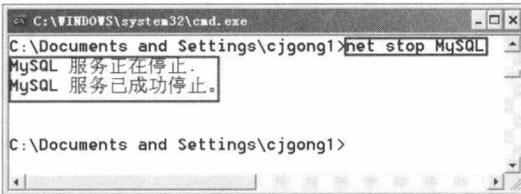


图 2.37 关闭 MySQL 服务

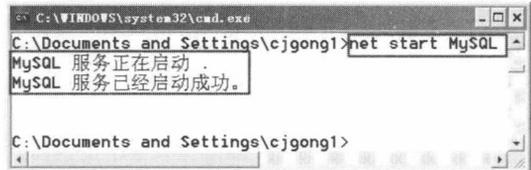


图 2.38 启动 MySQL 服务

当 MySQL 软件的服务启动后，除了可以在图形界面的服务窗口和 DOS 窗口中查看外，还可以通过 Windows 系统的任务管理器来查看。当打开任务管理器，如果存在“mysql.exe”进程则表示 MySQL 软件的服务启动，如图 2.39 所示。

注意：如果要打开任务管理器有两种方式，分别为通过使用快捷键【Ctrl+Alt+Delete】来实现或右击桌面的任务栏，然后选择“任务管理器 (K)”命令来实现，如图 2.40 所示。



图 2.39 任务管理器

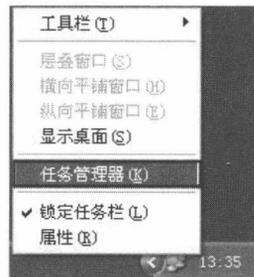


图 2.40 打开任务管理器

2.3 MySQL 官方客户端软件

由于 MySQL 软件是基于 C/S 模式的数据库管理系统，因此在日常各种工作中，可以通过各种客户端软件来与 MySQL 数据库管理系统相关联。针对 MySQL 软件，MySQL 公司开发了众多的客户端软件来帮助用户管理 MySQL 软件，最著名的就是“MySQL Command Line Client”和 MySQL-Workbench 客户端软件。

2.3.1 命令行客户端软件——MySQL Command Line Client

当 MySQL 软件安装完后，一般都会安装一个简单命令行实用程序（MySQL Command Line Client）。该客户端没有下拉菜单、没有流行的用户界面、不支持鼠标等任何类似的东西。

当选择“开始”→“程序”→“MySQL”→“MySQL Server 5.5”→“MySQL 5.5 Command Line Client”菜单命令时，就会打开“MySQL Command Line Client”程序，如图 2.41 所示。

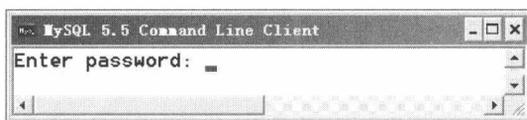


图 2.41 运行“MySQL Command Line Client”程序

输入正确的密码后，登录到 MySQL 软件。如果要使用该种方式登录 MySQL 软件，需要保证 MySQL 软件的服务处于启动状态。

当通过 MySQL Command Line Client 程序登录到服务器后（见图 2.42），首先会输出一段欢迎内容和一个“mysql>”命令提示符。在欢迎内容中主要介绍了如下几部分内容。

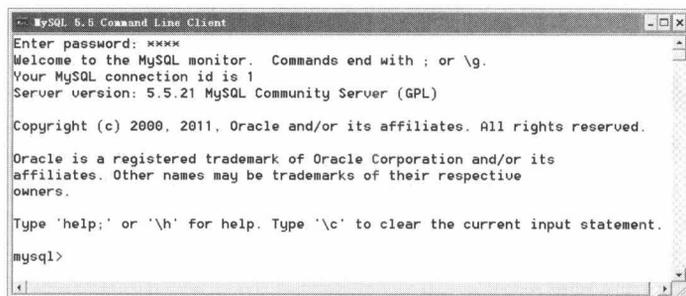


图 2.42 登录到 MySQL 数据库

- Commands end with , or \g: 命令的结束符，用“;”符号或者“\g”符号结束，同时还可以通过“\G”符号来结束。
- Your MySQL connection id is 1: 其中 id 表示客户端的连接 ID，该数据记录了 MySQL 服务到目前为止的连接次数，每次新连接都会自动加 1。由于数据库服务是新安装的，所以 ID 值为 1。
- Server version: 5.5.21: MySQL 软件的版本。
- MySQL Community Server (GPL): 表示 MySQL 软件是社区版。
- Type 'help;' or 'h' for help: 表示输入“help;”或“h”命令可以查看帮助信息。
- Type 'c' to clear the current input statement: 表示输入“c”命令可以清除前面的命令。

如果想通过“MySQL Command Line Client”程序来操作 MySQL 软件，只需在“mysql>”命令提示符后输入相应内容，同时并以分号(;)或\g 来结束，最后按【Enter】键即可操作 MySQL 软件。

“MySQL Command Line Client”程序是众多 MySQL 客户端软件中使用最多的工具之一，它可以快速地登录和操作 MySQL 软件，本书中使用的绝大多数实例都是由该客户端软件运行。

MySQL 数据库应用从入门到精通

2.3.2 通过 DOS 窗口连接 MySQL 软件

在 Windows 系统中还可以通过 DOS 窗口来登录 MySQL 软件，以解决没有安装“MySQL Command Line Client”的情况。该种方式登录 MySQL 软件的具体步骤如下：

(1) 选择“开始”→“运行”命令，打开“运行”对话框，然后在“打开”文本框中输入“cmd”，打开 DOS 窗口，如图 2.43 所示。

(2) 在 DOS 窗口中，可以通过 mysql 命令来登录 MySQL 软件，具体命令内容如下：

```
mysql -h 127.0.0.1 -u root -p
```

【代码说明】上述命令中，mysql 是登录 MySQL 软件的命令，-h 表示需要登录 MySQL 软件的 IP 地址，-u 表示登录 MySQL 软件的用户名，-p 表示登录 MySQL 软件的密码。

【运行效果】

具体运行过程如图 2.44 所示，当提示输入密码时，输入正确密码则会输出一段欢迎内容和一个“mysql>”命令提示符，这时就表示登录到 MySQL 软件。

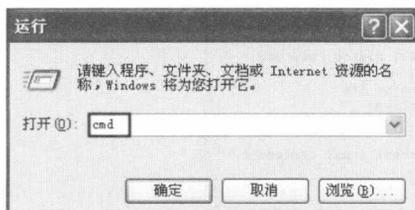


图 2.43 “运行”对话框

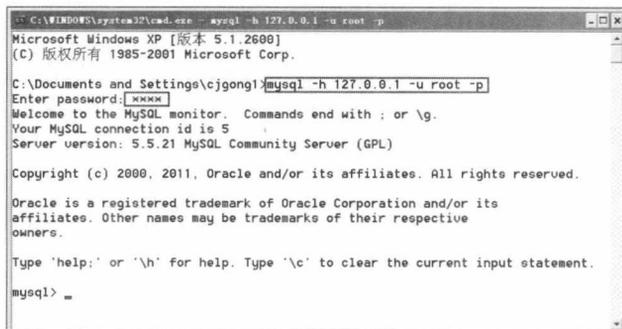


图 2.44 登录 MySQL 软件

通过上述步骤就可以成功连接到 MySQL 软件。

当在 DOS 窗口中执行“mysql -h 127.0.0.1 -u root -p”命令时，有时会出现如图 2.45 所示的错误。之所以会出现上述错误，是由于在配置 MySQL 软件时，在如图 2.46 所示的界面中没有选择“Include Bin Directory in Windows PATH”复选框。

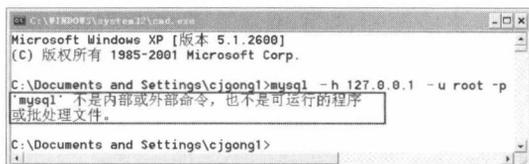


图 2.45 出现错误

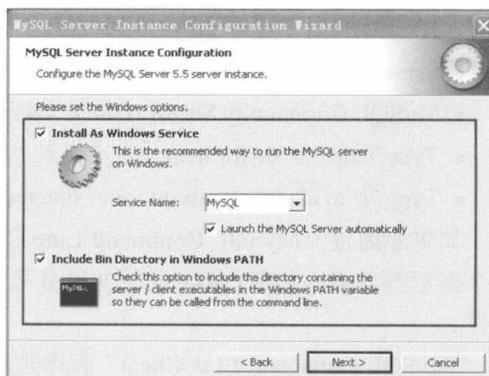


图 2.46 设置 MySQL 命令

为了解决上述错误，除了可以重新配置 MySQL 软件外，还可以通过配置环境变量 path 来实现，具体步骤如下：

(1) 右击“我的电脑”，在弹出的菜单中选择“属性”命令，打开“系统属性”对话框，如图 2.47 所示。

(2) 在“系统属性”对话框中选择“高级”选项卡，然后单击“环境变量”按钮（见图 2.48），则会打开“环境变量”对话框。

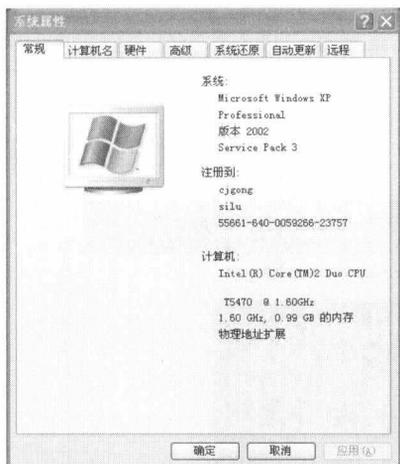


图 2.47 “系统属性”对话框

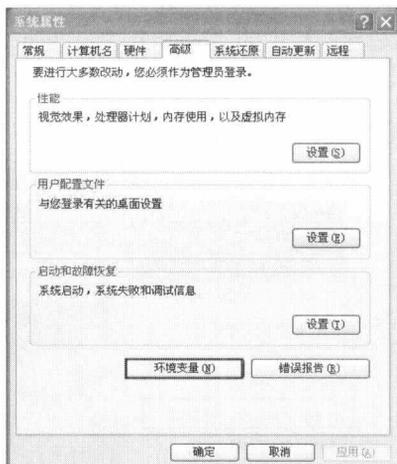


图 2.48 打开环境变量

(3) 在“环境变量”对话框中，在“系统变量”选项选中“Path”环境变量（见图 2.49），然后单击“编辑”按钮，则可以打开关于名为 Path 的“编辑系统变量”对话框，在该对话框的“变量值”文本框中添加 MySQL 软件的目录“C:\Program Files\MySQL\MySQL Server 5.5\bin”，目录之间用“;”符号分割，如图 2.50 所示。

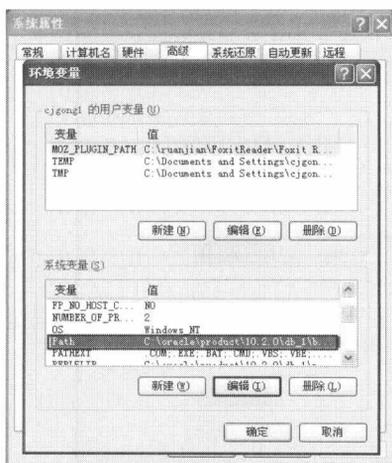


图 2.49 打开“编辑系统变量”

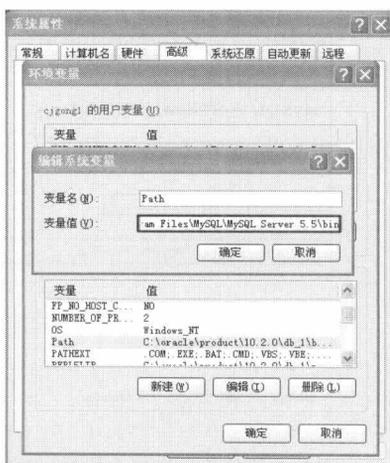


图 2.50 设置环境变量 Path

经过上述步骤修改完成后，如果在 DOS 窗口中再次执行“mysql -h 127.0.0.1 -u root -p”命令，则不会再次出现错误。

2.3.3 下载 MySQL-Workbench 客户端软件

如果想熟练使用“MySQL Command Line Client”客户端软件，必须要对相关命令非常熟悉，这对于初级 MySQL 用户来说比较困难。于是 MySQL 官方公司专门开发了图形化客户端软件 MySQL-Workbench。本节将详细介绍如何下载图形化客户端软件 MySQL-Workbench，具体步骤如下：

(1) 打开 MySQL 首页后，单击 Downloads(GA)导航栏，进入关于 MySQL 产品页面，在该页面中单击左边导航栏中的“MySQL Workbench (GUI Tool)”超链接，进入该软件下载页面，如图 2.51 所示。

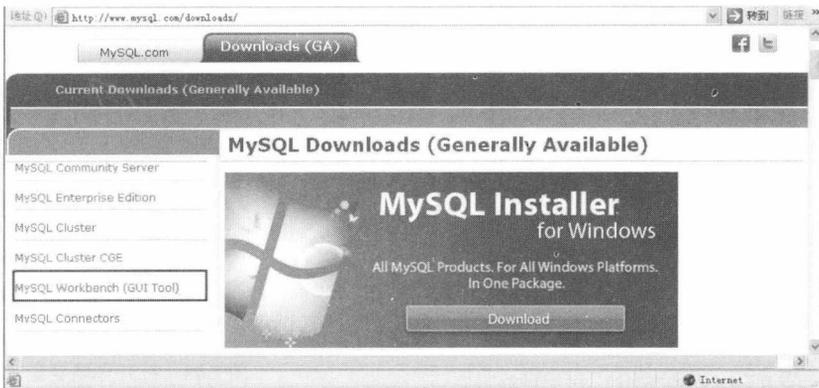


图 2.51 下载 MySQL Workbench

(2) 在“Select platform”下拉菜单中选择“Microsoft Windows”平台，然后单击“Windows (x86, 32-bit), MSI Installer”选项右边的 Download 按钮，下载 32 位的 MySQL Workbench 安装软件，如图 2.52 所示。

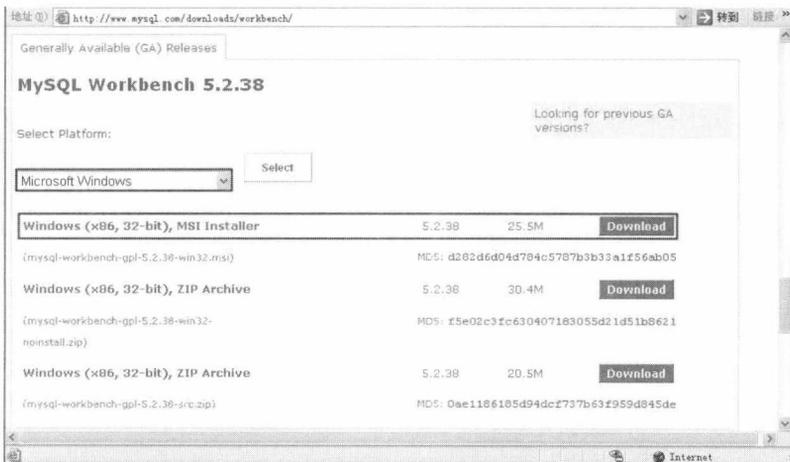


图 2.52 选择 MySQL Workbench 5.2.38

(3) 选择相应平台和版本的 MySQL Workbench 软件后，单击 Download 按钮，进入选择镜像网站的页面，在该页面中可以选择任意一个网站进行下载，如图 2.53 所示。



图 2.53 选择镜像网站

下载完 MySQL Workbench 后，即可安装该客户端软件。

2.3.4 安装 MySQL-Workbench 客户端软件

2.3.3 节介绍了如何下载客户端软件 MySQL-Workbench，下载完 MySQL-Workbench 安装程序后即可开始安装该软件。具体的安装步骤如下：

(1) 双击 MySQL-Workbench 安装程序 (mysql-workbench-gpl-5.2.38-win32.msi)，使用 Windows Installer 开始安装过程，进入欢迎界面，如图 2.54 所示。

(2) 单击 Next 按钮后，弹出“Destination Folder”对话框，单击 Change 按钮，设置 MySQL-Workbench 安装路径，然后单击 Next 按钮，进入选择安装类型对话框，如图 2.55 所示。

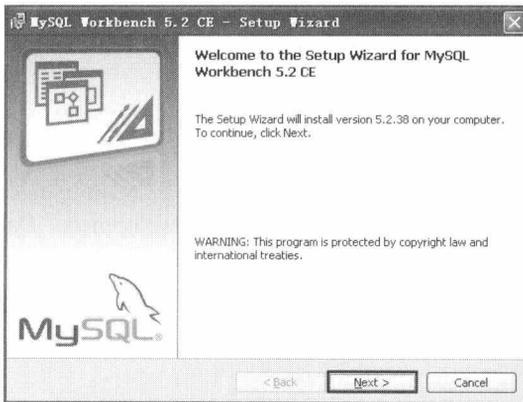


图 2.54 欢迎界面

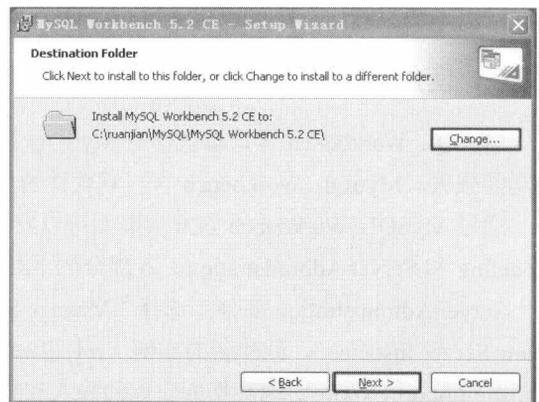


图 2.55 设置安装路径

MySQL 数据库应用从入门到精通

(3) 选择 Complete 单选按钮进行全部安装 (见图 2.56)，然后单击 Next 按钮，进入确认信息对话框，关于 MySQL 安装类型如下：

- Complete: 完全的安装类型。
- Custom: 自定义的安装类型。

(4) 在信息确认对话框中确认一下安装的信息，如果没有错误，单击 Install 按钮，对 MySQL-Workbench 软件进行安装，如图 2.57 所示。

(5) 最后进入安装完成的界面，在该对话框中会询问是否现在启动 MySQL-Workbench 软件，如果不想现在启动，可以取消选中“Launch MySQL Workbench now”复选框，然后单击 Finish 按钮，完成对 MySQL-Workbench 软件的安装，如图 2.58 所示。

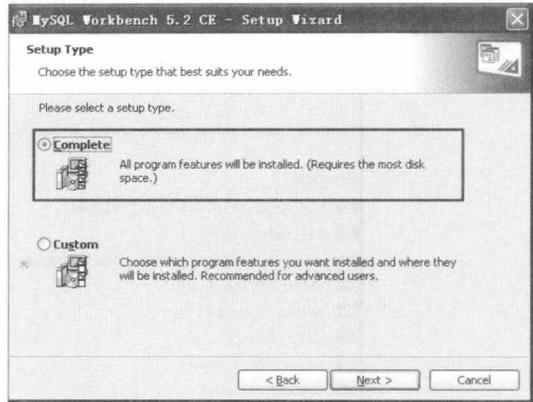


图 2.56 选择安装类型

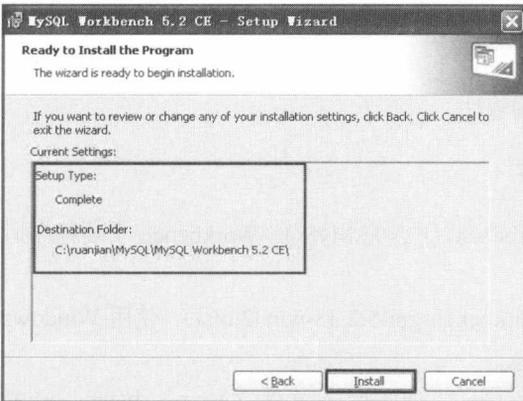


图 2.57 确认信息界面



图 2.58 安装成功

经过上述步骤后，即可成功安装 MySQL Workbench 客户端软件。

2.3.5 使用 MySQL-Workbench 客户端软件

MySQL_Workbench 客户端软件是 MySQL 官方提供的图形管理工具，该工具不仅简洁实用，而且功能强大。MySQL_Workbench 客户端软件的界面如图 2.59 所示。

通过 MySQL_Workbench 客户端界面可以发现该软件包含 3 部分功能：SQL Development、Data Modeling 和 Server Administration。各部分的主要功能如下。

Server Administration 部分：选择“Manage Server Instance”，配置 MySQL 软件运行，然后选择“New Server Instance”，创建新的实例 (mysql@localhost)，主要配置界面如图 2.60 所示。最后选择 Manage Import/Export 实现数据库实例的导入和导出。

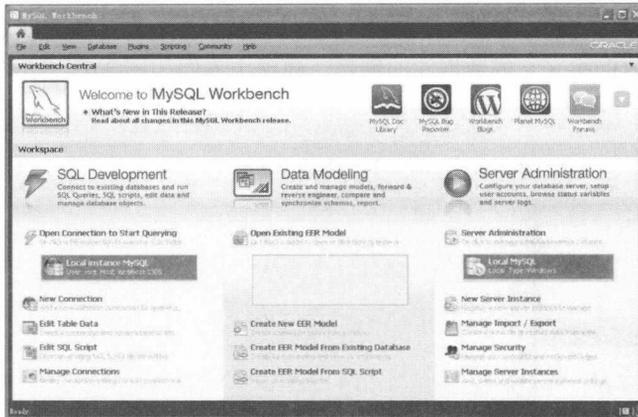


图 2.59 MySQL_Workbench 客户端软件的界面

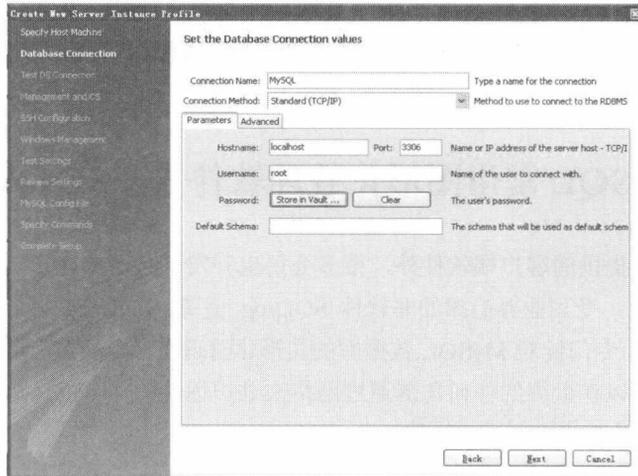


图 2.60 设置连接配置界面

SQL Development 部分: 选择 New Connection, 创建链接, 然后选择 Edit Table Data 查询数据等, 查询数据主界面如图 2.61 所示。

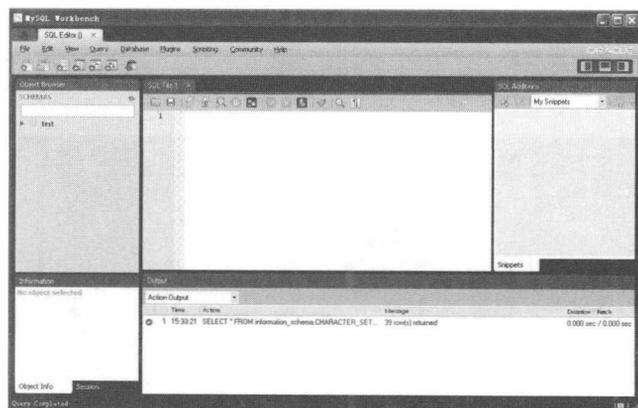


图 2.61 查询数据主界面

MySQL 数据库应用从入门到精通

Data Modeling 部分：主要功能为从建模开始来设计数据库，在建模完成后导出为 SQL 语句，然后再将 SQL 导入到数据库来完成数据库的创建，建模主界面如图 2.62 所示。

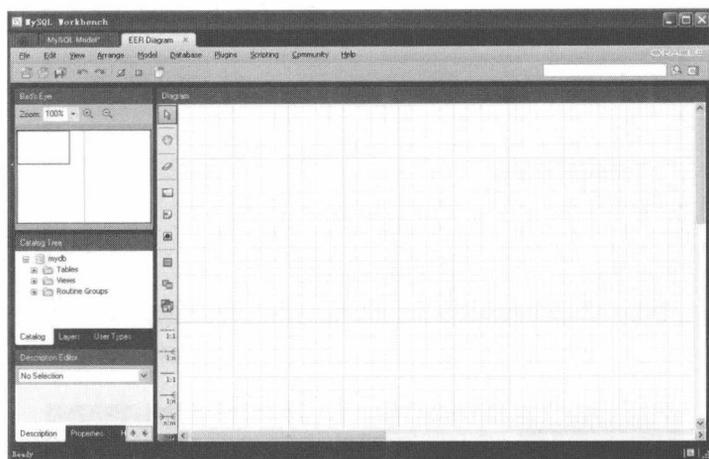


图 2.62 建模主界面

2.4 MySQL 常用图形化管理软件——SQLyog 软件

除了 MySQL 官方提供的客户端软件外，很多公司也开发了自己的客户端软件。众多的第三方图形化 MySQL 工具中，受到业界追捧的非软件 SQLyog 莫属。SQLyog 客户端软件由世界著名的 Webyog 公司开发研制，专门针对 MySQL 数据库的图形化管理工具。该客户端软件的突出特点是简洁高效、功能强大，可以在世界的任何角落通过网络来维护远端的 MySQL 数据库。本书所使用的图形化 MySQL 客户端软件为 SQLyog 软件。

2.4.1 下载 SQLyog 软件

作为一款最受欢迎的图形化 MySQL 客户端软件 SQLyog，其最新的版本为 SQLyog-9.6.3-0，可以通过下面的步骤来实现该平台的下载，具体如下：

(1) 首先访问下载 SQLyog 的官方网站 (<http://www.webyog.com/cn/>)，如图 2.63 所示。

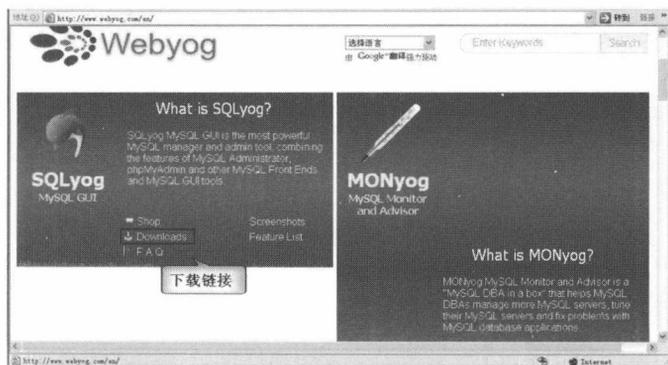


图 2.63 SQLyog 首页

(2) 打开 SQLyog 首页后，单击 Downloads 超链接，进入下载 SQLyog 产品页面，如图 2.64 所示。单击“GA (Stable) 9.6.3-0 Trial”超链接，实现 SQLyog 软件的下载。



图 2.64 下载 SQLyog 软件

在下载 SQLyog 产品页面中，关于“SQLyog MySQL GUI”的下载类型有两种，分别为 GA (Stable) 9.6.3-0 Trial 和 GA (Stable) 9.6.3-0 Full，其中前者表示试用版本，可以免费使用 30 天，而后者如果使用，需要输入相应信息。

下载完 SQLyog-9.6.3-0Trial 后，即可安装 SQLyog 客户端软件。

2.4.2 安装 SQLyog 客户端软件

2.4.1 节介绍了如何下载客户端软件 SQLyog，下载完安装程序后，开始安装该软件。具体的安装步骤如下所示：

(1) 双击 SQLyog 安装程序 (SQLyog-9.6.3-0Trial.exe)，使用 Windows Installer 开始安装，弹出“Please select a language”对话框，如图 2.65 所示。为了便于使用，在该对话框中选择“Chinese(Simplified)”，单击 OK 按钮，进入欢迎界面，如图 2.66 所示。



图 2.65 选择语言

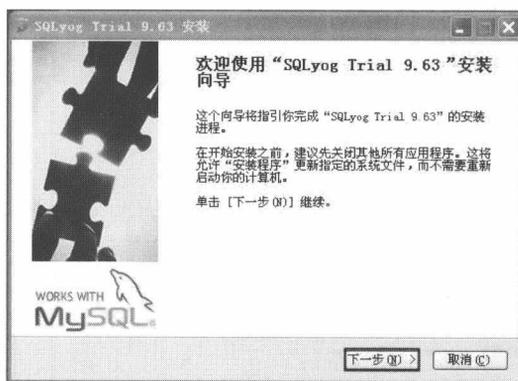


图 2.66 欢迎界面

(2) 单击“下一步”按钮，进入“许可证协议”对话框，如图 2.67 所示，选择“我接受‘许可证协议’中的条款”单选按钮，然后单击“下一步”按钮，进入“选择组件”对话框。

MySQL 数据库应用从入门到精通

(3) 选择相应的组件，如图 2.68 所示，然后单击“下一步”按钮，进入“选择安装位置”对话框。

(4) 设置安装目录为“C:\ruanjian\SQLyog”，如图 2.69 所示，然后单击“安装”按钮进行自动安装，如图 2.70 所示。

(5) 自动安装成功后，单击“下一步”按钮，进入安装完成对话框。在该对话框中会询问是否现在启动 SQLyog 软件，如果不想现在启动，可以取消选中“运行 SQLyog Trial 9.63”复选框，然后单击“完成”按钮，完成对 SQLyog 客户端软件的安装，如图 2.71 所示。

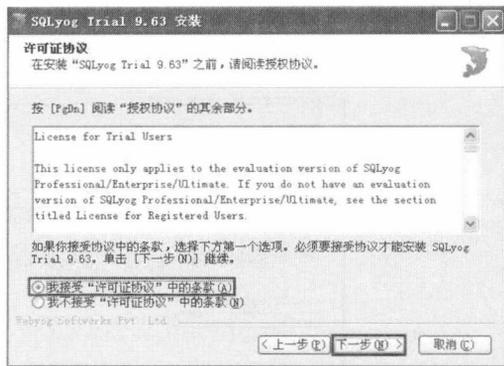


图 2.67 接受许可证协议

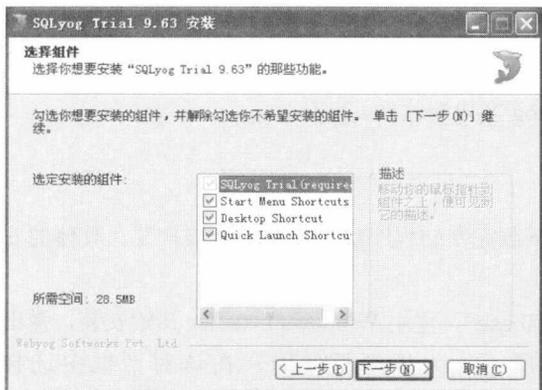


图 2.68 选择组件

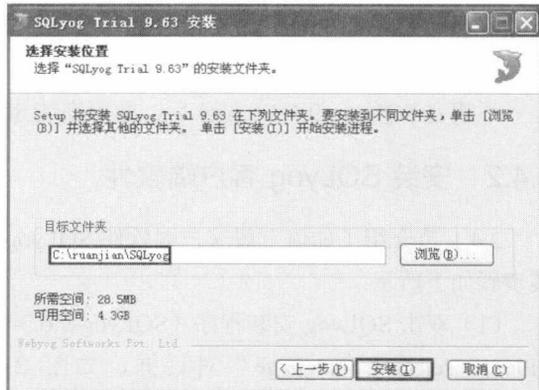


图 2.69 选择安装位置

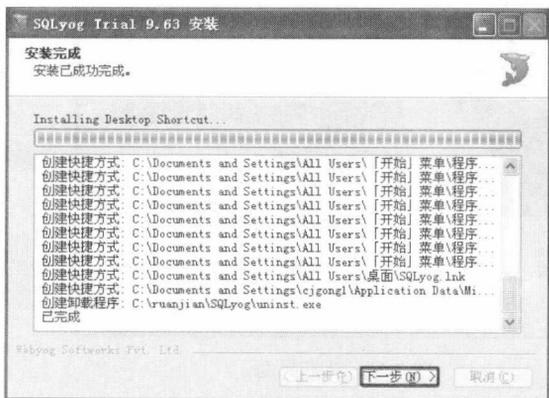


图 2.70 进行安装

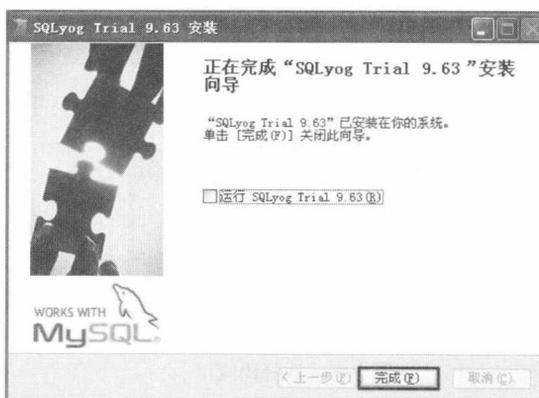


图 2.71 完成安装

经过上述步骤，已经成功安装 SQLyog 客户端软件。

2.4.3 通过 SQLyog 客户端软件登录 MySQL 软件

SQLyog 客户端软件的功能非常全面，包含数据库的管理、数据对象管理、数据导入和导出、数据管理，下面将对几个重要的管理功能进行详细介绍。

双击 SQLyog 快捷方式，打开关于该客户端软件的登录界面，正确输入服务器的地址、用户名、密码、端口，如图 2.72 所示，单击“连接”按钮，即可连接和登录数据库服务器，进入 SQLyog 软件的操作界面，如图 2.73 所示。



图 2.72 登录界面

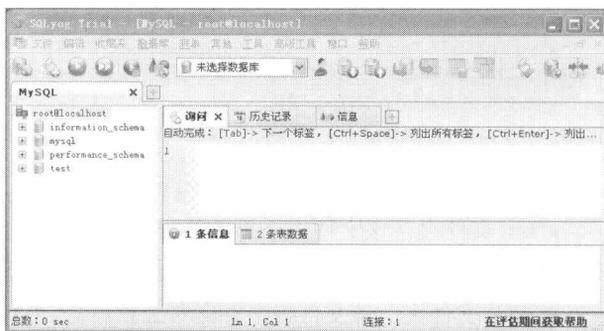


图 2.73 操作界面

通过上述步骤，即可成功登录 MySQL 软件。

2.5 使用免安装的 MySQL 软件

通过 MySQL 安装程序 (mysql-5.5.21-win32.msi) 来安装 MySQL 软件，虽然简洁高效，但是实现起来不够灵活。本节将介绍一种比较灵活的安装方式——通过免安装的 MySQL 软件包来实现。

1. 下载免安装版 MySQL 软件包

在具体使用免安装版的 MySQL 软件包之前，首先到网站“<http://www.mysql.com/downloads/mysql/>”去下载，在该页面中选择免安装版“Windows (x86, 32-bit), ZIP Archive”，然后单击 Download 超链接即可实现软件包的下载，如图 2.74 所示。

2. 解压免安装版 MySQL 软件包

解压 mysql-5.5.21-win32.zip 软件包到 Windows 系统的 C 盘里，该软件包解压后的默认文件名为“mysql-5.5.21-win32”。为了便于使用，修改解压后的文件名为 mysql。该文件夹下的内容如图 2.75 所示。

MySQL 数据库应用从入门到精通

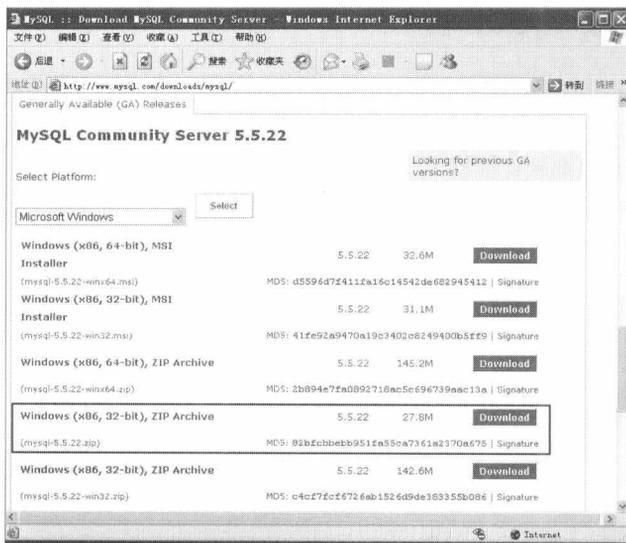


图 2.74 下载免安装版软件

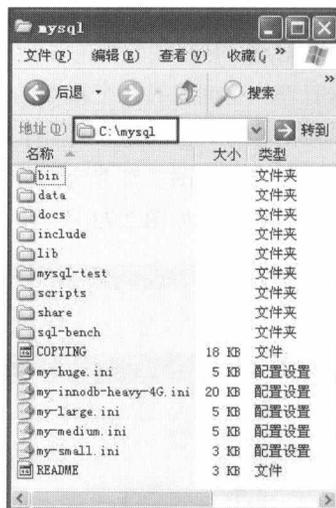


图 2.75 mysql 文件夹内容

mysql 文件夹下各个文件的作用如下。

- bin 文件夹：该文件夹下存放着可执行文件。
- data 文件夹：该文件夹存储着日志文件和数据库。
- docs 文件夹：该文件夹下存储着版权信息、MySQL 的更新日志和安装信息等文档。
- include 文件夹：该文件夹下存放着头文件。
- lib 文件夹：该文件夹下存放着库文件。
- mysql-test 文件夹：该文件夹存储着与测试有关的文件。
- scripts 文件夹：该文件夹存储着用 Perl 语言编写的实用工具脚本。
- share 文件夹：该文件夹下存放着字符集、语言等信息。
- sql-bench 文件夹：该文件夹下存储着多种数据库之间性能比较的信息和基准程序。

3. 创建和修改 my.ini 文件

在 MySQL 软件安装文件夹下除了程序文件外，还存在许多扩展名为.ini 的文件，但是没有名为 my.ini 的文件。由于现在计算机的内存一般为 2GB，远远大于 256MB，因此选择 my-large.ini 文件为模板，复制该文件到 C:\mysql 文件夹里同时修改文件名为 my.ini。

创建好 my.ini 文件后，还需要对该文件进行相应的设置，首先在该文件的“[mysqld]”组中加入如下两条记录：

```
basedir="c:\\mysql\\"
datadir="c:\\mysql\\data\\"
```

【代码说明】

上述两条记录中，basedir 参数表示 MySQL 软件的安装路径，datadir 参数表示 MySQL 数据的文件。然后再添加一个组和一条记录，具体内容如下：

```
[WindowsMySQLServer]
Server="C:\\mysql\\bin\\mysqld.exe"
```

【代码说明】

上述内容中，组名 WindowsMySQLServer 表示 Windows 操作系统中名为 MySQL 的服务，Server 参数表示 MySQL 软件端程序，该参数的值为 MySQL 软件可执行文件，即为 "C:\\mysql\\bin\\mysqld.exe"。

注意：不同版本的 MySQL 软件，其服务端的程序是不一样的。版本为 5.5.21 的服务器可执行文件为 mysqld.exe。

当修改完 my.ini 文件后，需要进行保存，才能关闭 my.ini 文件。这样配置文件 my.ini 才能生效。

4. 设置 MySQL 服务为 Windows 系统服务

为了便于操作，最好把 MySQL 软件可执行文件添加到 Windows 系统的服务里，具体设置方式如下，在“运行窗口”中执行如下命令（见图 2.76）。单击“确定”按钮后，如果出现一个 DOS 窗口一闪而过，则说明这个命令已经执行成功。

5. 启动服务

为了能够连接到 MySQL 软件进行操作，需要启动 MySQL 服务，具体命令如下：

```
net start MySQL
```

具体执行过程如图 2.77 所示。

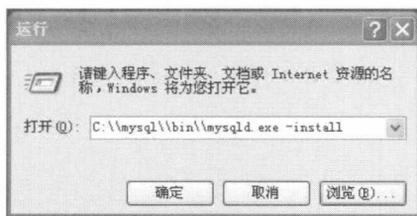


图 2.76 设置服务

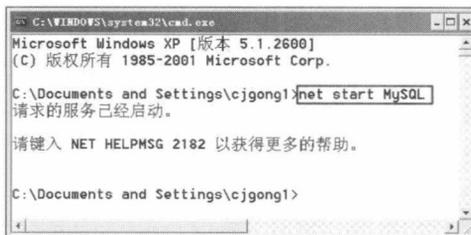


图 2.77 启动服务

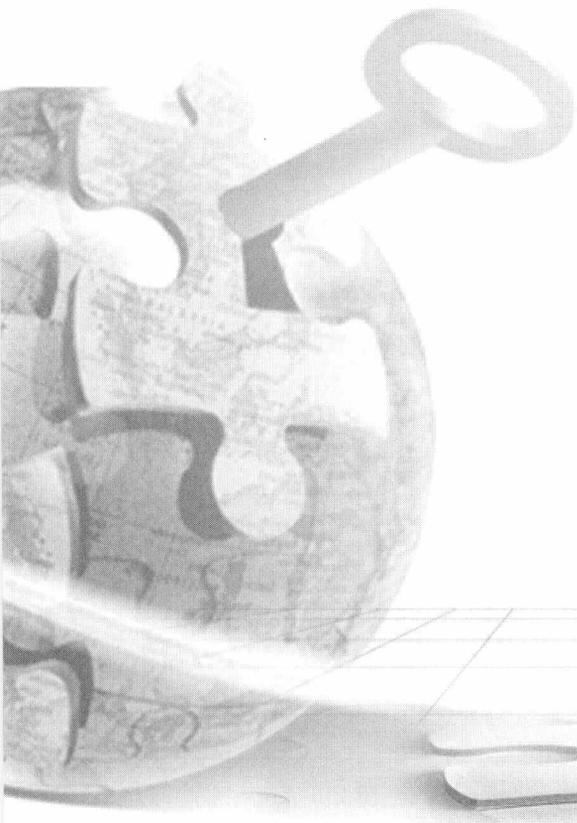
通过上述步骤，MySQL 软件安装和配置完毕。

2.6 小 结

本章主要介绍 Windows 系统下 MySQL 软件的安装、配置和常见操作，主要从下载和安装 MySQL 软件、MySQL 软件的常用操作和 MySQL 软件的客户端软件三方面介绍。其中前者详细介绍了 MySQL 安装软件的相关概念与 MySQL 软件的下载、安装、配置和卸载操作；接着通过图形化界面和 DOS 窗口两种方式来介绍 MySQL 软件的常用操作；最后详细介绍了 MySQL 官方客户端软件“MySQL Command Line Client”和“MySQL-Workbench”与常用第三方图形化客户端软件 SQLyog 的下载和操作。

通过对本章的学习，读者不仅能够正确搭建 MySQL 软件环境，而且还会熟悉操作 MySQL 客户端软件。

第2篇 MySQL 数据库操作和应用篇



本篇主要介绍了 MySQL 数据库对各类对象的基本操作和应用，其中前者主要包含数据库对象操作、表对象操作、索引对象操作、视图对象操作、触发器对象操作和数据操作。在具体介绍操作数据时，详细介绍了数据的插入、更新和删除操作。而对于数据查询操作则会从单表查询和多表查询两方面进行介绍，单表查询主要包含简单数据记录查询、条件数据查询、排序数据查询结果、限制数据查询数量、统计函数和分组数据查询；多表查询主要包含内连接查询、外连接查询、合并查询数据记录和子查询。

第 3 章 MySQL 数据库基本操作

数据库这个概念的用法很多，但针对本书而言，数据库是一种可以通过某种方式存储数据库对象的容器。简而言之，其实数据库就是一个拥有特定排放顺序的文件柜，而数据库对象则是存放在文件柜中的各种文件。

通过本章的学习，可以掌握如下内容：

- 数据库和数据库对象基本概念；
- 创建数据库；
- 查看和选择数据库；
- 删除数据库。

3.1 数据库和数据库对象

对于 MySQL 服务器，当连接上 MySQL 服务器后，即可操作数据库中存储到数据库对象里的数据。上述描述中涉及了几个概念：数据库、数据库对象和数据。在具体介绍数据库操作之前，首先需要了解这些概念。

查看帮助文档发现，数据库是存储数据库对象的容器。在 MySQL 软件中，数据库可以分为系统数据库和用户数据库两大类。

1. 系统数据库

系统数据库是指安装完 MySQL 服务器后，会附带的一些数据库，如图 3.1 所示。系统数据库会记录一些必需的信息，用户不能直接修改这些系统数据库。各个系统数据库的作用如下：

- **information_schema**：主要存储系统中的一些数据库对象信息，如用户表信息、列信息、权限信息、字符集信息和分区信息等。
- **performance_schema**：主要存储数据库服务器性能参数。
- **mysql**：主要存储系统的用户权限信息。
- **test**：该数据库为 MySQL 数据库管理系统自动创建的测试数据库，任何用户都可以使用。

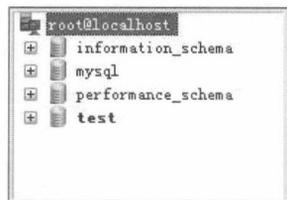


图 3.1 系统数据库

2. 用户数据库

用户数据库是用户根据实际需求创建的数据库，如图 3.2 所示。其中 `userdatabase` 数据库属于用户数据库。

既然数据库是存储数据库对象的容器，那么什么是数据库对象呢？数据库可以存储哪些数据库对象呢？所谓数据库对象是指存储、管理和使用数据的不同结构形式，主要包含表、视图、存储过程、函数、触发器和事件等。

在 SQLyog 客户端软件的“对象资源管理器”中，每个数据库节点下都拥有一个树形路径结构，如图 3.3 所示。其实树形路径结构中的每个具体子节点都是数据库对象。关于数据库对象，后面章节将逐步介绍。

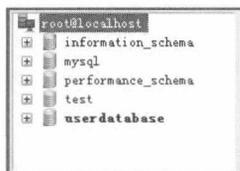


图 3.2 用户数据库



图 3.3 数据库对象

3.2 数据库相关操作——创建数据库

数据库的操作包括创建数据库、查看数据库、选择数据库，以及删除数据库。本节将详细介绍如何创建数据库。创建数据库，实际上就是在数据库服务器中划分一块空间，用来存储相应的数据库对象。

3.2.1 创建数据库的语法形式

查看帮助文档发现，在 MySQL 中创建数据库通过 SQL 语句 `CREATE DATABASE` 来实现，其语法形式如下：

```
create Database database_name
```

上述语句中 `database_name` 参数表示所要创建的数据库名字，在具体创建数据库时，数据库名不能与已经存在的数据库名重名。除了上述要求外，推荐数据库名命名（标识符）规则如下：

- 由字母、数字、下划线、@、#和\$符号组成，其中字母可以是英文字符 `a~z` 或 `A~Z`，也可以是其他语言的字母字符。
- 首字母不能是数字和\$符号。
- 标识符不允许是 MySQL 的保留字。
- 不允许有空格和特殊字符。
- 长度小于 128 位。

下面将通过一个具体的实例来说明如何创建视图。

【实例 3-1】 执行 SQL 语句 `CREATE DATABASE`，在数据库管理系统中创建名为 `databasetest`

MySQL 数据库应用从入门到精通

的数据库。具体 SQL 语句如下：

```
create Database databasetest;
```

【代码说明】上述 SQL 语句中，数据库名的标识符一定要有实际意义。

【运行效果】执行上面的 SQL 语句，其结果如图 3.4 所示。

```
mysql> #创建数据库#
mysql> CREATE DATABASE databasetest;
Query OK, 1 row affected (0.00 sec)

mysql>
```

图 3.4 创建数据库

通过执行查询结果可以发现，执行完 SQL 语句后，下面有一行提示“Query OK, 1 row affected (0.00 sec)”，这段提示可以分为 3 部分，含义如下。

- “Query OK”：表示 SQL 语句执行成功。
- “1 row affected”：表示操作只影响了数据库中一行的记录。
- “0.00 sec”：表示操作执行的时间。

注意：查看帮助文档发现，创建数据库的 SQL 语句不属于查询操作，那么为什么显示结果却是“Query OK”呢？

上述问题是 MySQL 软件的一个特点，即所有 SQL 语句中的 DDL 和 DML（不包含 SELECT）语句执行成功后都会显示“Query OK”。

3.2.2 通过工具来创建数据库

通过 MySQL 数据库服务器自带的工具“MySQL Command Line Client”来创建数据库，虽然高效、灵活，但是对于初级用户比较困难，需要掌握 SQL 语句。在具体实践中，用户可以通过客户端软件 SQLyog 来创建数据库。

下面将通过一个具体的实例来说明如何通过客户端软件 SQLyog 创建数据库。

【实例 3-2】与实例 3-1 一样，在数据库管理系统中创建名为 databasetest 的数据库。

(1) 首先连接数据库服务器，在“对象资源管理器”窗口中将显示 MySQL 数据库管理系统中所有的数据库，如图 3.5 所示。

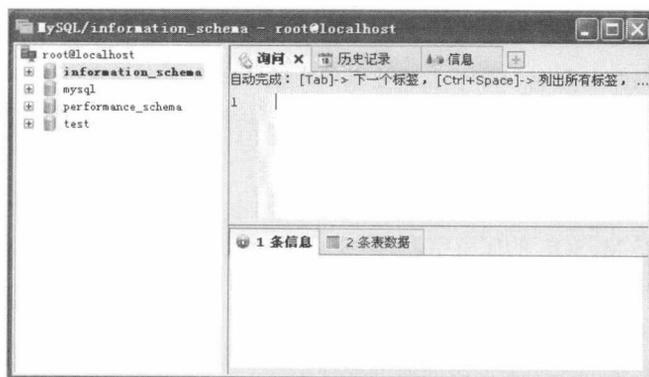


图 3.5 连接 MySQL 数据库管理系统

(2) 右击“对象资源管理器”窗口中空白处，在弹出的菜单中选择“创建数据库”命令，如图3.6所示。



图 3.6 创建数据库命令

(3) 弹出“创建数据库”对话框，在“数据库名称”文本框中输入 databasetest，然后单击“创建”按钮，创建数据库 databasetest，具体设置信息如图3.7所示。

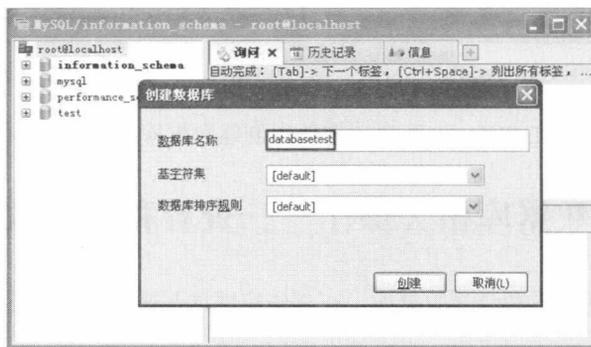


图 3.7 创建数据库

在“创建数据库”对话框中，除了必须输入数据库的标识符外，还可以设置该数据库的字符集和数据库排序规则，关于字符集和排序规则，后面章节将逐步介绍。

(4) 当数据库 databasetest 创建成功后，“对象资源管理器”就会显示出名为 databasetest 的数据库，如图3.8所示。

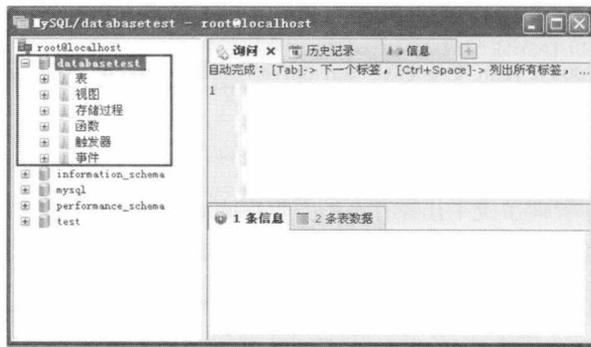


图 3.8 数据库创建成功

MySQL 数据库应用从入门到精通

对于 SQLyog 工具，除了可以通过以上步骤（向导方式）创建数据库外，还可以在“询问”窗口中输入 SQL 语句来实现，具体步骤如下：

(1) 首先在“询问”窗口中输入创建名为 databasetest 数据库的 SQL，然后单击工具栏中的“执行查询”（）按钮，进行创建数据库，如图 3.9 所示。

(2) 当数据库创建成功后，不仅会在“信息”窗口显示相关信息，而且当单击工具栏中的“刷新对象浏览器”（）按钮，会在“对象资源管理器”窗口中显示新建的数据库，如图 3.10 所示。



图 3.9 执行查询 SQL 语句



图 3.10 数据库创建成功

通过上述步骤，则可以在 SQLyog 客户端软件中创建数据库 databasetest 成功。

3.3 数据库相关操作——查看和选择数据库

本节将详细介绍如何查看和选择数据库，在具体实现这些操作之前，首先需要确定所操作的数据库对象已经存在。

3.3.1 查看数据库

对于初级用户，当创建数据库时，经常会发生如图 3.11 所示的错误。之所以不能正确创建数据库“databasetest”，是因为该数据库已经存在。因此对于有经验的用户，在创建数据库之前，需要查看数据库管理系统中是否已经存在该名字数据库。

```
mysql> #创建一个已经存在的数据库#
mysql> CREATE DATABASE databasetest;
ERROR 1007 (HY000): Can't create database 'databasetest'; database exists
mysql> _
```

图 3.11 创建数据库出错

那么如何查看数据库管理系统中已经存在的数据库呢？查看帮助文档可以发现，在 MySQL 中查看已经存在的数据库通过 SQL 语句“SHOW DATABASES”来实现，其语法形式如下：

```
SHOW DATABASES;
```

上述 SQL 语句主要用来实现显示 MySQL 软件中所有的数据库。

执行上面的 SQL 语句，其结果如图 3.12 所示。

通过执行 SQL 结果可以发现，执行完“SHOW DATABASES”语句后，会显示一个列表。在该列表中，除了会显示用户数据库 databasetest 外，还有另外 4 个系统数据库。

对于客户端软件 SQLyog，如果想查看数据库管理系统中已经存在的数据库，除了可以在“询问”窗口中执行“SHOW DATABASES”语句外，还可以通过单击工具栏中的“刷新对象浏览器”按钮，这时就会在“对象资源管理器”窗口中显示出所有的数据库，如图 3.13 所示。



图 3.12 通过 SQL 语句显示数据库

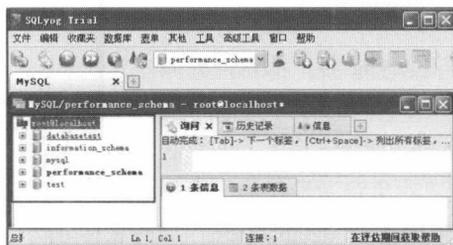


图 3.13 通过客户端软件显示数据库

3.3.2 选择数据库

既然数据库是数据库对象的容器，而在数据库管理系统中一般又会存在许多数据库，那么在操作数据库对象之前，首先需要确定是哪一个数据库。即在对数据库对象进行操作时，需要先选择一个数据库。

查看帮助文档可以发现，在 MySQL 中选择数据库通过 SQL 语句 USE 来实现，其语法形式如下：

```
USE database_name
```

上述语句中，database_name 参数表示所要选择的数据库名字。

在具体选择数据库之前，首先需要查看数据库管理系统中已经存在的数据库，然后才能从这些已经存在的数据库中进行选择，如果选择一个不存在的数据库，则会出现如图 3.14 所示的错误。

【实例 3-3】 执行 SQL 语句 USE，选择名为 databasetest 的数据库。具体 SQL 语句如下：

```
SHOW DATABASES;
USE databasetest;
```

【代码说明】 上述 SQL 语句中，首先查看 MySQL 软件中所有的数据库，然后才选择数据库 databasetest。

【运行效果】 执行上面的查询语句，其结果如图 3.15 所示。

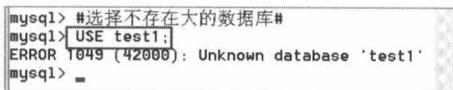


图 3.14 选择不存在的数据库



图 3.15 选择数据库

MySQL 数据库应用从入门到精通

注意：在执行选择数据库语句时，如果出现“Database changed”提示，则表示选择数据库成功。

对于客户端软件 SQLyog，如果想选择数据库管理系统中已经存在的数据库，可以在“询问”窗口中执行 USE 语句，在执行前 SQLyog 客户端软件界面如图 3.16 所示，执行后 SQLyog 客户端软件界面如图 3.17 所示。除了上述方法外，还可以在“对象资源管理器”窗口中单击所要选的数据库 databasetest，如图 3.18 所示。

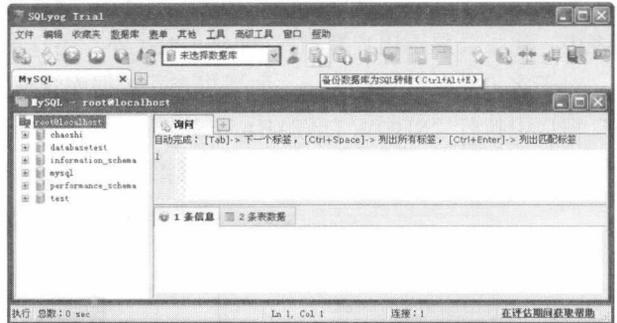


图 3.16 未选择数据库

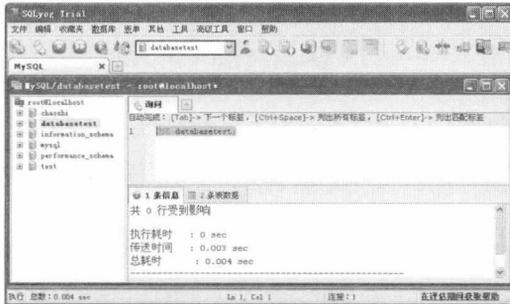


图 3.17 选择数据库



图 3.18 选择数据库

3.4 数据库相关操作——删除数据库

本节将详细介绍如何删除数据库，在具体实现该操作之前，首先需要确定所操作的数据库对象已经存在。

3.4.1 删除数据库的语法形式

查看帮助文档可以发现，在 MySQL 中删除数据库通过 SQL 语句 DROP DATABASE 来实现，其语法形式如下：

```
DROP DATABASE database_name
```

上述语句中，database_name 参数表示所要删除的数据库名字。

【实例 3-4】 执行 SQL 语句 DROP DATABASE 删除数据库，即首先创建一个名为 databasetest1 的数据库，然后再删除 databasetest1 数据库。具体步骤如下。

(1) 创建数据库 databasetest1，具体语句如下：

```
CREATE DATABASES databasetest1;
```

【运行效果】 执行上面的 SQL 查询语句，其结果如图 3.19 所示。

(2) 查看 MySQL 数据库管理系统中，是否已经存在名为 databasetest1 的数据库，具体语句如下：

```
SHOW DATABASES;
```

【运行效果】执行上面的 SQL 查询语句，其结果如图 3.20 所示。

```
mysql> #创建数据库#
mysql> CREATE DATABASE databasetest1;
Query OK, 1 row affected (0.00 sec)
mysql> _
```

图 3.19 创建数据库

```
mysql> #查询数据库#
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| chaoshi |
| databasetest |
| databasetest1 |
| mysql |
| performance_schema |
| test |
+-----+
7 rows in set (0.00 sec)
mysql> _
```

图 3.20 查看数据库

(3) 根据查询数据库的结果可以发现 databasetest1 数据库已经创建成功，接着通过 DROP DATABASE 删除数据库，具体语句如下：

```
DROP DATABASE databasetest1;
```

【运行效果】执行上面的查询语句，其结果如图 3.21 所示。

根据执行结果，数据库 databasetest1 已经删除成功。这时如果再次查询数据库，则数据库列表中不会显示出数据库 databasetest1，执行结果如图 3.22 所示。

```
mysql> #删除数据库#
mysql> DROP DATABASE databasetest1;
Query OK, 0 rows affected (0.02 sec)
mysql> _
```

图 3.21 实现删除数据库

```
mysql> #显示数据库#
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| chaoshi |
| databasetest |
| mysql |
| performance_schema |
| test |
+-----+
6 rows in set (0.00 sec)
```

图 3.22 实现删除数据库

注意：数据库删除后，该数据库容器里的数据库对象也会全部删除，所以删除数据库之前一定要仔细、小心。

3.4.2 通过工具来删除数据库

通过 MySQL 数据库服务器自带的工具“MySQL Command Line Client”来删除数据库，虽然高效、灵活，但是对于初级用户比较困难，这需要掌握 SQL 语句。在具体实践中，用户可以通过客户端软件 SQLyog 来删除数据库。

下面将通过一个具体的实例来说明如何通过客户端软件 SQLyog 删除数据库。

MySQL 数据库应用从入门到精通

【实例 3-5】与实例 3-4 一样，删除数据库管理系统中名为 databasetest1 的数据库。

(1) 连接数据库服务器，在“对象资源管理器”窗口中将显示 MySQL 数据库管理系统中所有的数据库，如图 3.23 所示。

(2) 右击“对象资源管理器”窗口中空白处，在弹出的菜单中选择“创建数据库”命令，如图 3.24 所示。



图 3.23 连接 MySQL 数据库管理系统

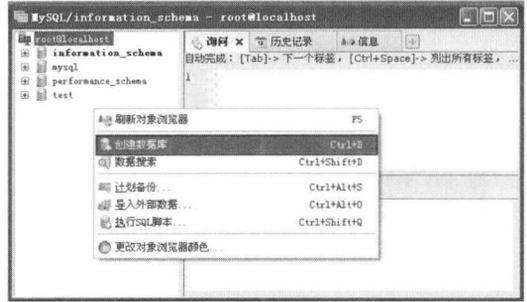


图 3.24 选择创建数据库命令

(3) 弹出“创建数据库”对话框，在“数据库名称”文本框中输入 databasetest，然后单击“创建”按钮，创建数据库 databasetest1，具体设置信息如图 3.25 所示。这时如果单击工具栏中的“刷新对象浏览器”按钮，会在“对象资源管理器”窗口中显示出新建的数据库，如图 3.26 所示。

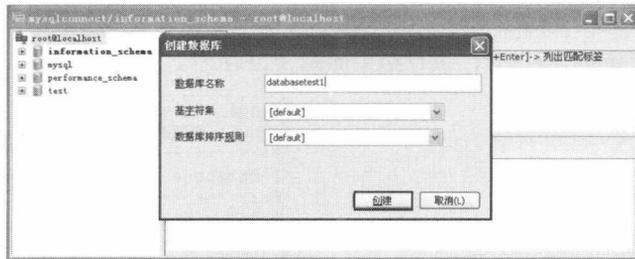


图 3.25 创建数据库



图 3.26 创建数据库成功

(4) 如果要删除刚创建的数据库 databasetest1，只要右击“对象资源管理器”窗口中的数据库 databasetest1，然后在弹出的菜单中选择“更多数据库操作”→“删除数据库”命令，如图 3.27 所示。



图 3.27 选择“删除数据库”命令

(5) 在弹出的确认删除对话框中，单击“是”按钮（见图 3.28）后，这时“对象资源管理器”窗口中就没有了数据库 databasetest1，如图 3.29 所示。

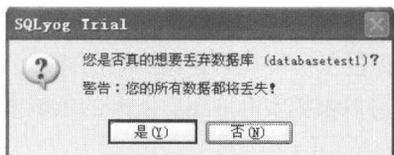


图 3.28 确认删除对话框



图 3.29 删除数据库 databasetest1 成功

通过上述步骤，即可在 SQLyog 客户端软件中删除数据库 databasetest1。

3.5 小 结

本章主要介绍了在 MySQL 软件中如何操作数据库，主要从数据库相关概念和数据库相关操作两方面进行了讲解。其中前者主要介绍了数据库和数据库对象相关概念，后者详细介绍了如何通过 SQL 语句和客户端软件 SQLyog 这两种方式来创建数据库、查看数据库、选择数据库和删除数据库。

通过对本章的学习，读者不仅对数据库有了一个全新的认识，而且还掌握了 MySQL 软件中数据库的各种操作。

第 4 章 MySQL 数据库中存储引擎和数据类型

与其他数据库软件不同，MySQL 数据库软件提供了一个名为存储引擎的概念。由于存储引擎是以插件的形式被 MySQL 数据库软件引入，所以可以根据实际应用、实际的领域来选择相应的存储引擎。

在 MySQL 数据库软件中，虽然通过存储引擎决定数据库对象表的类型，但是如果创建表，还需要了解数据类型，因为其决定了表中可以存储数据的类型。

通过本章的学习，可以掌握如下内容：

- MySQL 软件所支持的存储引擎；
- MySQL 软件所支持的数据类型。

4.1 认识存储引擎

存储引擎是 MySQL 数据库管理系统的一个重要特征，在具体开发时，为了提高 MySQL 数据库管理系统的使用效率和灵活性，可以根据实际需要来选择存储引擎。因为存储引擎指定了表的类型，即如何存储和索引数据、是否支持事务等，同时存储引擎也决定了表在计算机中的存储方式。本节将详细介绍 MySQL 5.5 所支持的存储引擎，以及如何选择合适的存储引擎。

4.1.1 MySQL 5.5 所支持的存储引擎

用户在选择存储引擎之前，首先需要确定数据库管理系统支持哪些存储引擎。查看帮助文档发现，在 MySQL 数据库管理系统中查看支持的存储引擎通过 SQL 语句 SHOW ENGINES 来实现，其语法形式如下：

```
SHOW ENGINES
```

在上述语句中可以实现查看当前 MySQL 数据库管理系统所支持的存储引擎。

【实例 4-1】执行 SQL 语句 SHOW ENGINES，查看 MySQL 5.5 所支持的存储引擎和默认存储引擎。具体步骤如下：

(1) 执行 SQL 语句 SHOW ENGINES，查看存储引擎，具体 SQL 语句如下：

```
SHOW ENGINES \G
```

【代码说明】上述 SQL 语句中，主要用来实现显示存储引擎。

【运行效果】执行上面的 SQL 语句，其结果如图 4.1 所示。

```
mysql> #查看存储引擎#
mysql> SHOW ENGINES \G
***** 1. row *****
Engine: FEDERATED
Support: NO
Comment: Federated MySQL storage engine
Transactions: NULL
XA: NULL
Savepoints: NULL
***** 2. row *****
Engine: MRG_MYISAM
Support: YES
Comment: Collection of identical MyISAM tables
Transactions: NO
XA: NO
Savepoints: NO
***** 3. row *****
Engine: MyISAM
Support: YES
Comment: MyISAM storage engine
Transactions: NO
XA: NO
Savepoints: NO
***** 4. row *****
Engine: BLACKHOLE
Support: YES
Comment: /dev/null storage engine (anything you write to it disappears)
Transactions: NO
XA: NO
Savepoints: NO
***** 5. row *****
Engine: CSV
Support: YES
Comment: CSV storage engine
Transactions: NO
XA: NO
Savepoints: NO
***** 6. row *****
Engine: MEMORY
Support: YES
Comment: Hash based, stored in memory, useful for temporary tables
Transactions: NO
XA: NO
Savepoints: NO
***** 7. row *****
Engine: ARCHIVE
Support: YES
Comment: Archive storage engine
Transactions: NO
XA: NO
Savepoints: NO
***** 8. row *****
Engine: InnoDB
Support: DEFAULT
Comment: Supports transactions, row-level locking, and foreign keys
Transactions: YES
XA: YES
Savepoints: YES
***** 9. row *****
Engine: PERFORMANCE_SCHEMA
Support: YES
Comment: Performance Schema
Transactions: NO
XA: NO
Savepoints: NO
9 rows in set (0.09 sec)

mysql>
```

图 4.1 查询存储引擎

执行结果显示，MySQL 5.5 支持 9 种存储引擎，分别为 FEDERATED、MRG_MYISAM、MyISAM、BLACKHOLE、CSV、MEMORY、ARCHIVE、InnoDB 和 PERFORMANCE_SCHEMA。其中 Engine 参数表示存储引擎名称；Support 参数表示 MySQL 数据库管理系统是否支持该存储引擎，其中值 YES 表示支持，值 NO 表示不支持，值 DEFAULT 表示该存储引擎是数据库管理系统默认支持的存储引擎；Comment 参数表示关于存储引擎的评论；Transactions 参数表示存储引擎是否支持事务，其中值 YES 表示支持，而值 NO 表示不支持；XA 参数表示存储引擎所支持的分布式是否符合 XA 规范，其中值 YES 表示支持，而值 NO 表示不支持；Savepoints 参数表示存储引擎是否支持事务处理中的保存点，其中值 YES 表示支持，而值 NO 表示不支持。

MySQL 数据库应用从入门到精通

注意：通过执行结果可以发现，MySQL 5.5 数据库管理系默认的存储引擎为 InnoDB 存储引擎。

(2) 在具体执行 SQL 语句中，可以用“;”、“\g”和“\G”符号表示语句结束。其中前两个符号的作用一样，而最后一个符号除了表示语句结束外，还可以使得结果显示的更加美观。执行 SQL 语句 SHOW ENGINES，以“;”或者“\g”作为结束符号查看存储引擎，具体 SQL 语句如下：

```
SHOW ENGINES;
```

或者

```
SHOW ENGINES\g
```

上述命令中，虽然语句内容一致，但是前者以“;”符号结束，而后者通过“\g”符号结束。执行上面的 SQL 语句，其结果如图 4.2 和图 4.3 所示。

```
mysql> #查看存储引擎#
mysql> SHOW ENGINES;
+-----+-----+-----+-----+
| Engine | Support | Comment |
+-----+-----+-----+-----+
| FEDERATED | NO | Federated MySQL storage engine |
| MRG_MYISAM | YES | Collection of identical MyISAM tables |
| MyISAM | YES | MyISAM storage engine |
| BLACKHOLE | YES | /dev/null storage engine (anything you write to it disappears) |
| CSU | YES | CSU storage engine |
| MEMORY | YES | Hash based, stored in memory, useful for temporary tables |
| ARCHIVE | YES | Archive storage engine |
| InnoDB | DEFAULT | Supports transactions, row-level locking, and foreign keys |
| PERFORMANCE_SCHEMA | YES | Performance Schema |
| ISAM | NO | Obsolete storage engine |
+-----+-----+-----+-----+
9 rows in set (0.00 sec)

mysql>
```

图 4.2 显示效果不友好

```
mysql> #查看存储引擎#
mysql> SHOW ENGINES\g
+-----+-----+-----+
| Engine | Support | Comment |
+-----+-----+-----+
| MyISAM | YES | Default engine as of MySQL 3.23 with great performance |
| MEMORY | YES | Hash based, stored in memory, useful for temporary tables |
| InnoDB | DEFAULT | Supports transactions, row-level locking, and foreign keys |
| BerkeleyDB | NO | Supports transactions and page-level locking |
| BLACKHOLE | NO | /dev/null storage engine (anything you write to it disappears) |
| EXAMPLE | NO | Example storage engine |
| ARCHIVE | YES | Archive storage engine |
| CSU | NO | CSU storage engine |
| ndbcluster | NO | Clustered, fault-tolerant, memory-based tables |
| FEDERATED | NO | Federated MySQL storage engine |
| MRG_MYISAM | YES | Collection of identical MyISAM tables |
| binlog | DISABLED | This is a meta storage engine to represent the binlog in a transaction |
| ISAM | NO | Obsolete storage engine |
+-----+-----+-----+
13 rows in set (0.00 sec)

mysql>
```

图 4.3 显示效果不友好

执行结果显示，MySQL 5.5 支持 9 种存储引擎，但是显示效果不便于用户查看，所以以“;”和“\g”符号结束方式不常用。

(3) 在 MySQL 数据库管理系统中，除了可以通过 SQL 语句 SHOW ENGINES 查看所支持的存储引擎外，还可以通过 SQL 语句 SHOW VARIABLES 来查看所支持的存储引擎，具体 SQL 语句如下：

```
SHOW VARIABLES LIKE 'have%';
```

执行上面的 SQL 语句，其结果如图 4.4 所示。

```
mysql> #查看存储引擎#
mysql> SHOW VARIABLES LIKE 'have%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| have_compress | YES   |
| have_crypt    | NO    |
| have_csu      | YES   |
| have_dynamic_loading | YES   |
| have_geometry | YES   |
| have_innodb   | YES   |
| have_ndbcluster | NO    |
| have_openssl  | DISABLED |
| have_partitioning | YES   |
| have_profiling | YES   |
| have_query_cache | YES   |
| have_rtree_keys | YES   |
| have_ssl      | DISABLED |
| have_symlink  | YES   |
+-----+-----+
14 rows in set (0.00 sec)

mysql> _
```

图 4.4 查看存储引擎

在显示结果中，Variable_name 参数表示存储引擎的名字；Value 参数表示 MySQL 数据库管理系统是否支持存储引擎，其中值 YES 表示支持，值 NO 表示不支持，值 DISABLED 表示支持但是还没开启。

4.1.2 操作默认存储引擎

查看帮助文档可以发现，安装版 MySQL 5.5 数据库管理系统的默认存储引擎为 InnoDB，免安装版 MySQL 5.5 数据库管理系统的默认存储引擎为 MyISAM。在数据库管理系统中，可以修改默认存储引擎吗？本节将详细介绍关于默认存储引擎的操作。

1. 查询默认存储引擎

如果需要操作默认存储引擎，首先需要查看默认存储引擎。那么如何查看默认存储引擎呢？查看帮助文档发现，可以通过执行 SQL 语句 SHOW VARIABLES 来查看默认的存储引擎，具体 SQL 语句如下：

```
SHOW VARIABLES LIKE 'storage_engine%';
```

上述命令中，设置关键字 LIKE 的关键字为“storage_engine%”，表示查询默认存储引擎。执行上面的 SQL 语句，其结果如图 4.5 所示。

```
mysql> #查看默认存储引擎#
mysql> SHOW VARIABLES LIKE 'storage_engine%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| storage_engine | InnoDB |
+-----+-----+
1 row in set (0.00 sec)

mysql>
```

图 4.5 查看默认存储引擎

执行结果显示，InnoDB 存储引擎为默认存储引擎。

2. 修改默认存储引擎

在 MySQL 数据库管理系统中，如果需要修改默认存储引擎，可以通过两种方式来实现。一种方式为向导方式，另一种方式为手动修改配置文件方式。

当通过向导方式修改默认存储引擎时，选择“开始”→“程序”→“MySQL”→“MySQL Server 5.5”→“MySQL Server Instance Configuration Wizard”菜单命令，进入图形化实例配置向导的欢迎界面。

在图形化实例配置向导中，连续单击 Next 按钮，进入“MySQL 选择用途类型”界面，如图 4.6 所示。在该界面中如果选择“Multifunctional Database”单选按钮，则 MySQL 数据库管理系统的默认存储引擎为 InnoDB。如果选择“Non-Transaction Database Only”单选按钮，则 MySQL 数据库管理系统的默认存储引擎为 MyISAM。

通过手动方式来修改默认存储引擎时，需要修改 MySQL 数据库管理系统的配置文件 my.ini，具体步骤如下：

(1) 首先打开 my.ini 配置文件，关于 “[mysqld]” 组的内容如下：

```
#服务器端参数配置
# SERVER SECTION
.....
[mysqld]
#服务器端的端口号。
port=3306
#MySQL 数据库服务器的安装目录
basedir="C:/Program Files/MySQL/MySQL Server 5.5/"
#MySQL 数据库数据文件的目录
datadir="C:/Documents and Settings/All Users/Application Data/MySQL/MySQL Server
5.5/Data/"
#MySQL 服务器端的字符集
character-set-server=gbk
#MySQL 服务器的存储引擎
default-storage-engine=INNODB
.....
```

如果想修改默认存储引擎，只需修改[mysqld]组中的 default-storage-engine 参数。即如果想设置默认存储引擎为 MyISAM，只需修改成“default-storage-engine=MyISAM”即可实现。

注意：如果想使修改后的参数生效，须重新启动 MySQL 服务。

(2) 重启 MySQL 服务后，这时再次执行 SQL 语句 SHOW VARIABLES 来查看默认的存储引擎，具体 SQL 语句如下：

SHOW VARIABLES LIKE 'storage_engine%';

【运行效果】执行上面的 SQL 语句，其结果如图 4.7 所示。

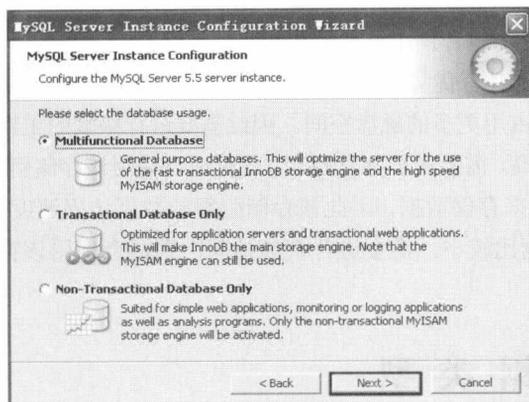


图 4.6 选择用途类型界面

```
mysql> #查看存储引擎#
mysql> SHOW VARIABLES LIKE 'storage_engine%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| storage_engine | MyISAM |
+-----+-----+
1 row in set (0.03 sec)

mysql>
```

图 4.7 查看存储引擎

执行结果显示，默认存储引擎已经修改为 MyISAM。

4.1.3 选择存储引擎

在具体使用 MySQL 数据库管理系统时，选择一个合适的存储引擎是一个非常复杂的问题。因为每种存储引擎都有自己的特性、优势和应用场合，所以不能随便选择存储引擎。为了能够正确地选择存储引擎，必须掌握各种存储引擎的特性。

下面重点介绍几种常用的存储引擎，它们对各种特性的支持如表 4.1 所示。

表 4.1 存储引擎特性

| 特 性 | MyISAM | InnoDB | MEMORY |
|---------|--------|--------|--------|
| 存储限制 | 有 | 64TB | 有 |
| 事务安全 | 不支持 | 支持 | 不支持 |
| 锁机制 | 表锁 | 行锁 | 表锁 |
| B 树索引 | 支持 | 支持 | 支持 |
| 哈希索引 | 不支持 | 不支持 | 支持 |
| 全文索引 | 支持 | 不支持 | 不支持 |
| 集群索引 | 不支持 | 支持 | 不支持 |
| 数据缓存 | | 支持 | 支持 |
| 索引缓存 | 支持 | 支持 | 支持 |
| 数据可压缩 | 支持 | 不支持 | 不支持 |
| 空间使用 | 低 | 高 | N/A |
| 内存使用 | 低 | 高 | 中等 |
| 批量插入的速度 | 高 | 低 | 高 |
| 支持外键 | 不支持 | 支持 | 不支持 |

MySQL 数据库应用从入门到精通

表 4.1 主要介绍了 MyISAM、InnoDB 和 MEMORY 三种存储引擎特性的对比，接下来将详细介绍这 3 个存储引擎的应用场合并给出相应的建议。

- MyISAM 存储引擎：由于该存储引擎不支持事务、也不支持外键，所以访问速度比较快。因此对事务完整性没有要求并以访问为主的应用适合使用该存储引擎。
- InnoDB 存储引擎：由于该存储引擎在事务上具有优势，即支持具有提交、回滚和崩溃恢复能力的事务安装，所以比 MyISAM 存储引擎占用更多的磁盘空间。因此需要进行频繁的更新、删除操作，同时还对事务的完整性要求比较高，需要实现并发控制，此时适合使用该存储引擎。
- MEMORY 存储引擎：该存储引擎使用内存来存储数据，因此该存储引擎的数据访问速度快，但是安全上没有保障。如果应用中涉及数据比较小，需要进行快速访问，则适合使用该存储引擎。

4.2 数据类型

在 MySQL 数据库管理系统中，可以通过存储引擎来决定表的类型，即其决定了表的存储方式。同时 MySQL 数据库管理系统也提供了数据类型决定表存储数据的类型。查看帮助文档可以发现，MySQL 数据库管理系统提供了整数类型、浮点数类型、定点数类型和位类型、日期和时间类型、字符串类型。

4.2.1 整数类型

MySQL 数据库管理系统除了支持标准 SQL 中的所有整数类型（SMALLINT 和 INT）外，还进行了相应扩展。扩展后增加了 TINYINT、MEDIUMINT 和 BIGINT 这三个整数类型。

下面通过表 4.2 展示各种整数类型的特性，其中 INT 与 INTEGER 这两个整数类型是同名词（可以相互替换），具体内容如下。

表 4.2 整数类型

| 整数类型 | 字节 | 最小值 | 最大值 |
|---------------|----|----------------------------------|---|
| TINYINT | 1 | 有符号-128 无符号 0 | 有符号 127 无符号 255 |
| SMALLINT | 2 | 有符号-32768 无符号 0 | 有符号 32767 无符号 65535 |
| MEDIUMINT | 3 | 有符号-8388608 无符号 0 | 有符号 8388607 无符号 1677215 |
| INT 和 INTEGER | 4 | 有符号-2147483648 无符号 0 | 有符号 2147483647 无符号 4294967295 |
| BIGINT | 8 | 有符号-9223372036854775808 无符号 0 | 有符号 9223372036854775807 无符号 18446744073709551615 |

表 4.2 中内容显示，TINYINT 类型占用字节数最小，只需 1 字节，因此该类型的取值范围最小。BIGINT 类型占用字节数最大，需要 8 个字节，因此该类型的取值范围最大。

注意：为什么要了解整数类型所占的字节数？因为根据数据类型所占的字节数可以算出该类型的取值范围。

在计算机中所有的内容都存储为不同组合的二进制码（0 和 1），整数类型数据也不例外，只不过整数是有符号数（正负数），因此其左边的第一位为符号位（0 为正数，1 为负数）。例如，TINYINT 类型占 1 字节（1 字节=8 位），所以该类型数据的最大值二进制如图 4.8 所示，最小值二进制如图 4.9 所示。

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|---|---|---|---|---|---|---|---|

图 4.8 正整数最大值二进制表示

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|

图 4.9 负整数最小值二进制表示

对于图 4.8 所示的二进制数，转换成十进制数为 2^7-1 ，即 127。对于图 4.9 所示的二进制数，转换成十进制数为 -2^7 ，即 -128。

在具体使用 MySQL 数据库管理系统时，如果需要存储整数类型数据，则可以选择 TINYINT、SMALLINT、MEDIUMINT、INT、INTEGER 和 BIGINT 类型，至于选择这些类型中的哪一个，首先需要判断存储整数数据的取值范围，当不超过 255 时，那选择 TINYINT 类型就足够了。虽然 BIGINT 类型的取值范围最大，最常用的整数类型却是 INT 类型。

如果无法区分各个整数类型的表示范围，可以通过查看 MySQL 的系统帮助查看相关信息。查看系统帮助的方法如实例 4-1 所示。

【实例 4-1】查看系统帮助

```
mysql> HELP contents;
You asked for HELP about HELP category: "Contents"
For more information, type 'HELP <item>', where <item> is one of the following
categories:
  #部分结果省略
  Data Types
  Functions

mysql> HELP Data Types;
You asked for HELP about HELP category: "Data Types"
For more information, type 'HELP <item>', where <item> is one of the following
topics:
#部分结果省略
  BIGINT
  DOUBLE
  INT
  SMALLINT
  VARCHAR
  YEAR DATA TYPE

mysql> HELP INT;
Name: 'INT'
```

MySQL 数据库应用从入门到精通

Description:

INT[(M)] [UNSIGNED] [ZEROFILL]

A normal-size integer. The signed range is -2147483648 to 2147483647.

The unsigned range is 0 to 4294967295.

使用命令“HELP contents”可以查看 MySQL 帮助文档支持的目录列表，然后根据需要查看的条目选择查看。输入“HELP INT”可以查看 INT 类型的帮助，如上述实例所示，INT 类型的表示范围为-2 147 483 648~2 147 483 647。

整数类型的使用比较简单，实例 4-2 所示为 INT 类型的创建和使用方法。

【实例 4-2】INT 的创建和使用

```
mysql> INSERT INTO int_test VALUES
-> (0),
-> (-1),
-> (1.1),
-> (1234567890),
-> (12345678901),
-> (-12345678901);
Query OK, 6 rows affected, 2 warnings (0.00 sec)
Records: 6 Duplicates: 0 Warnings: 2

mysql> SHOW WARNINGS;
+-----+-----+-----+-----+
| Level | Code | Message |
+-----+-----+-----+-----+
| Warning | 1264 | Out of range value adjusted for column 'id' at row 5 |
| Warning | 1264 | Out of range value adjusted for column 'id' at row 6 |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> SELECT * FROM int_test;
+-----+
| id |
+-----+
| 0 |
| -1 |
| 1 |
| 1234567890 |
| 2147483647 |
| -2147483648 |
+-----+
6 rows in set (0.00 sec)
```

上述实例中首先创建了一个含有 INT 类型字段的表，然后使用 INSERT 语句进行插入操作。当插入的数值在 INT 类型的表示范围内时，INT 数值可以正常插入，但如果插入的为其他数据类型（如浮点数）或者超过了 INT 表示的范围，此时会将插入的值截断并显示警告信息。

表 4.6 日期和时间类型

| 日期和时间类型 | 字节 | 最小值 | 最大值 |
|-----------|----|---------------------|---------------------|
| DATE | 4 | 1000-01-01 | 9999-12-31 |
| DATETIME | 8 | 1000-01-01 00:00:00 | 9999-12-31 23:59:59 |
| TIMESTAMP | 4 | 19700101080001 | 2038 年的某个时刻 |
| TIME | 3 | -835:59:59 | 838:59:59 |
| YEAR | 1 | 1901 | 2155 |

表 4.6 中内容显示，每种日期和时间数据类型都有一个取值范围，如果插入的值超过了该类型的取值范围，则会插入默认值。

在具体应用中，各种日期和时间类型的应用场合如下：

- 如果要表示年月日，一般会使用 DATE 类型。
- 如果要表示年月日时分秒，一般会使用 DATETIME 类型。
- 如果需要经常插入或者更新日期为当前系统时间，一般会使用 TIMESTAMP 类型。
- 如果要表示时分秒，一般会使用 TIME 类型。
- 如果要表示年份，一般会使用 YEAR 类型。因为该类型比 DATE 类型占用更少的空间。

在具体使用 MySQL 数据库管理系统时，要根据实际应用来选择满足需求的最小存储的日期类型。例如，如果应用只需存储“年份”，则可以选择存储字节为 1 的 YEAR 类型。如果要存储年月日时分秒，并且年份的取值可能比较久远，最好使用 DATETIME 类型，而不是 TIMESTAMP 类型，因为前者比后者所表示的日期范围要长一些。如果存储的日期需要让不同时区的用户使用，则可以使用 TIMESTAMP 类型，因为只有该类型日期能够与实际时区相对应。

日期和时间类型的使用方法如实例 4-5 所示。

【实例 4-5】日期和时间类型的使用方法

```
mysql> CREATE TABLE d_test(
-> f_date DATE,
-> f_datetime DATETIME,
-> f_timestamp TIMESTAMP,
-> f_time TIME,
-> f_year YEAR);
Query OK, 0 rows affected (0.05 sec)

mysql> SELECT CURDATE(),NOW(), NOW(),time(NOW()), YEAR(NOW()) \G
***** 1. row *****
CURDATE(): 2013-09-02
NOW(): 2013-09-02 16:28:09
NOW(): 2013-09-02 16:28:09
time(NOW()): 16:28:09
YEAR(NOW()): 2013
1 row in set (0.00 sec)

mysql> INSERT INTO d_test VALUES( CURDATE(),NOW(), NOW(),time(NOW()), YEAR(NOW())
);
Query OK, 1 row affected (0.00 sec)
```

MySQL 数据库应用从入门到精通

```
mysql> SELECT * FROM d_test \G
***** 1. row *****
  f_date: 2013-09-02
  f_datetime: 2013-09-02 16:28:28
  f_timestamp: 2013-09-02 16:28:28
  f_time: 16:28:28
  f_year: 2013
1 row in set (0.00 sec)
```

上述实例首先创建了一个包含日期和时间类型的表，使用了 SELECT 查看相关函数的输出以便对比，然后使用 INSERT 语句插入相关数值，通过此实例读者可以了解日期和时间类型的使用方法。

4.2.4 字符串类型

MySQL 数据库管理系统中有多种表示字符串的数据类型，各种版本有微小的差异，下面通过表 4.7 展示 MySQL 5.5 数据库管理系统所支持的 CHAR 系列字符串类型的特性，具体内容如下。

表 4.7 CHAR 系列字符串类型

| CHAR 系列字符串类型 | 字节 | 描述 |
|--------------|----|--------------------|
| CHAR(M) | M | M 为 0~255 之间的整数 |
| VARCHAR(M) | M | M 为 0~65 535 之间的整数 |

表 4.7 中内容显示，字符串类型 CHAR 的字节数是 M，例如 CHAR(4)的数据类型为 CHAR，其最大长度为 4 个字节。VARCHAR 类型的长度是可变的，其长度的范围为 0~65 535。

在具体使用 MySQL 数据库管理系统时，如果需要存储少量字符串，则可以选择 CHAR 和 VARCHAR 类型，至于是选择这两个类型中的哪一个，则需要判断所存储字符串长度是否经常变化，如果经常发生变化，则可以选择 VARCHAR 类型，否则选择 CHAR 类型。

下面通过表 4.8 展示 MySQL 5.5 数据库管理系统所支持的 TEXT 系列类型字符串的特性，具体内容如下。

表 4.8 TEXT 系列字符串类型

| TEXT 系列字符串类型 | 字节 | 描述 |
|--------------|-----------------|-------------|
| TINYTEXT | 0~255 | 值的长度为+2 个字节 |
| TEXT | 0~65 535 | 值的长度为+2 个字节 |
| MEDIUMTEXT | 0~167 772 150 | 值的长度为+3 个字节 |
| LONGTEXT | 0~4 294 967 295 | 值的长度为+4 个字节 |

表 4.8 中内容显示，TEXT 系列中的各种字符串类型允许的长度和存储字节不同，其中 TINYTEXT 字符串类型允许存储字符串长度最小，LONGTEXT 字符串类型允许存储字符串长度最大。

在具体使用 MySQL 数据库管理系统时，如果需要存储大量字符串（存储文章内容的纯文本），则可以选择 TEXT 系列字符串类型。至于是选择这些类型中的哪一个，则需要判断所存储字符串长度，根据存储字符串的长度来决定是选择允许长度最小的 TINYTEXT 字符串类型，还是选择允许长度最大的 LONGTEXT 字符串类型。

下面通过表 4.9 展示 MySQL 5.5 数据库管理系统所支持的 BINARY 系列字符串类型的特性，具体内容如下。

表 4.9 BINARY 系列字符串类型

| BINARY 系列字符串类型 | 字节 | 描述 |
|----------------|----|-----------|
| BINARY(M) | M | 允许长度为 0~M |
| VARBINARY(M) | M | 允许长度为 0~M |

表 4.9 内容中的两个类型，与 CHAR 系列字符串类型中 CHAR 和 VARCHAR 非常类似，不同的是，前者可以存储二进制数据（例如图片、音乐或者视频文件），而后者只能存储字符数据。

在具体使用 MySQL 数据库管理系统时，如果需要存储少量二进制数据，则可以选择 BINARY 和 VARBINARY 类型。至于是选择这两个类型中的哪一个，则需要判断所存储二进制数据长度是否经常变化。如果经常发生变化则可以选择 VARBINARY 类型，否则选择 BINARY 类型。

下面通过表 4.10 展示 MySQL 5.5 数据库管理系统所支持的 BLOB 系列字符串类型的特性，具体内容如下。

表 4.10 BLOB 系列字符串类型

| BLOB 系列字符串类型 | 字节 |
|--------------|-------------------|
| TINYBLOB | 0~255 |
| BLOB | 0~2 ¹⁶ |
| MEDIUMBLOB | 0~2 ²⁴ |
| LOB | 0~2 ³² |

表 4.10 内容中的四个类型，与 TEXT 系列字符串类型非常类似，不同的是，前者可以存储二进制数据（例如图片、音乐或者视频文件），而后者只能存储字符数据。

在具体使用 MySQL 数据库管理系统时，如果需要存储大量二进制数据（存储电影等视频文件），则可以选择 BLOB 系列字符串类型。至于是选择这些类型中的哪一个，则需要判断所存储二进制数据长度，根据存储二进制数据的长度来决定是选择允许长度最小的 TINYBLOB 字符串类型，还是选择允许长度最大的 LOBBLOB 字符串类型。

字符串类型使用方法如实例 4-6 所示。

【实例 4-6】字符串类型使用方法

```
mysql> CREATE TABLE user(
-> id INT,
-> name VARCHAR(20));
Query OK, 0 rows affected (0.11 sec)

mysql> INSERT INTO user VALUES(1,'bob'),
-> (2,'petter'),
-> (3,"a123456789123456789123"
->);
Query OK, 3 rows affected, 1 warning (0.00 sec)
Records: 3 Duplicates: 0 Warnings: 1
```

MySQL 数据库应用从入门到精通

```
mysql> SHOW WARNINGS;
+-----+-----+-----+
| Level | Code | Message |
+-----+-----+-----+
| Warning | 1265 | Data truncated for column 'name' at row 3 |
+-----+-----+-----+
1 row in set (0.00 sec)

mysql> SELECT * FROM user ;
+-----+-----+-----+
| id | name |
+-----+-----+-----+
| 1 | bob |
| 2 | petter |
| 3 | a1234567891234567891 |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

上述实例创建了一个包含 VARCHAR 类型的表，长度为 20，然后进行数据插入。注意如果插入的字符串的长度超过字符串定义的长度，字符串会被截断并显示警告信息。

4.3 小 结

本章主要介绍在 MySQL 软件中创建数据库对象表之前，需要掌握的一些概念，主要从存储引擎和数据类型两方面介绍。前者决定了数据库对象表的类型，主要介绍了 MySQL 数据库所支持的存储引擎、存储引擎的常用操作和选择存储引擎的策略；后者决定了数据库对象表中存储数据类型，主要介绍了整数类型、浮点数类型、定点数类型、位类型、日期和时间类型、字符串类型。

通过对本章的学习，读者不仅掌握了 MySQL 数据库所支持的各种存储引擎，而且还了解了 MySQL 数据库所支持的各种数据类型，为下一章操作数据库对象表做准备。

第 5 章 表的操作

在 MySQL 数据库中，表是一种很重要的数据库对象，是组成数据库的基本元素，由若干个字段组成，主要用来实现存储数据记录。表的操作包含创建表、查看表、删除表和修改表，这些操作是数据库对象的表管理中最基本、最重要的操作。

通过本节的学习，可以掌握在数据库中操作表，内容包括：

- 表的相关概念；
- 表的基本操作：创建、查看、更新和删除；
- 表的使用策略。

5.1 表的基本概念

表是包含数据库中所有数据的数据库对象。数据在表中的组织方式与在电子表格中相似，都是按行和列的格式组织的。其中每一行代表一条唯一的记录，每一列代表记录中的一个字段，如图 5.1 所示。

| COLLATION_NAME | CHARACTER SET NAME | ID | IS DEFAULT | IS COMPILED | SORTLEN |
|-------------------|--------------------|----|------------|-------------|---------|
| big5_chinese_ci | big5 | | | Yes | 1 |
| big5_bin | big5 | | | Yes | 1 |
| dec8_swedish_ci | dec8 | | | Yes | 1 |
| dec8_bin | dec8 | 69 | | Yes | 1 |
| cp850_general_ci | cp850 | 4 | Yes | Yes | 1 |
| cp850_bin | cp850 | 80 | | Yes | 1 |
| hp8_english_ci | hp8 | 6 | Yes | Yes | 1 |
| hp8_bin | hp8 | 72 | | Yes | 1 |
| koi8r_general_ci | koi8r | 7 | Yes | Yes | 1 |
| koi8r_bin | koi8r | 74 | | Yes | 1 |
| latin1_german_ci | latin1 | 5 | | Yes | 1 |
| latin1_swedish_ci | latin1 | 8 | Yes | Yes | 1 |
| latin1_danish_ci | latin1 | 15 | | Yes | 1 |
| latin1_german2_ci | latin1 | 31 | | Yes | 1 |
| latin1_bin | latin1 | 47 | | Yes | 1 |

图 5.1 表

表中的数据库对象包含列、索引和触发器如图 5.2 所示。

- 列 (Columns)：也称属性列，在具体创建表时，必须指定列的名字和数据类型。
- 索引 (Indexes)：是指根据指定的数据库表列建立起来的顺序，提供了快速访问数据的途径

MySQL 数据库应用从入门到精通

且可监督表的数据，使其索引所指向的列中的数据不重复，后面章节将详细介绍。

- 触发器 (Triggers)：是指用户定义的事务命令的集合，当对一个表中的数据进行插入、更新或删除时这组命令就会自动执行，可以用来确保数据的完整性和安全性，后面章节将详细介绍。

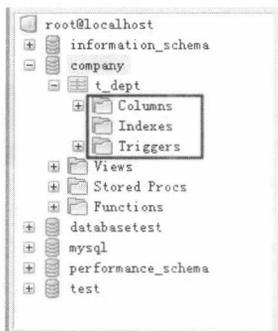


图 5.2 表中数据库对象

5.2 创建表

表的操作包括创建表、查看表、删除表和修改表。本节将详细介绍如何创建表。所谓创建表就是在数据库中建立新表，该操作是进行其他表操作的基础。

5.2.1 创建表的语法形式

查看帮助文档发现，在 MySQL 数据库管理系统中创建表通过 SQL 语句 CREATE TABLE 来实现，其语法形式如下：

```
CREATE TABLE table_name (
    属性名 数据类型,
    属性名 数据类型,
    .
    .
    .
    属性名 数据类型
)
```

上述语句中 table_name 参数表示所要创建的表名字，表名紧跟在关键字 CREATE TABLE 后面。表的具体内容定义在圆括号之中，各列之间用逗号分隔。其中“属性名”参数表示表字段的名称，“数据类型”参数指定字段的数据类型，例如，如果列中存储的为数字，则相应的数据类型为数值类型。在具体创建数据库时，表名不能与已经存在的表对象重名，其命名规则与数据库命名（标识符）规则一致。

【实例 5-1】执行 SQL 语句 CREATE TABLE，在数据库 company 中创建名为 t_dept 的表。具体步骤如下：

- (1) 首先执行 SQL 语句 CREATE DATABASE，创建数据库 company，具体 SQL 语句如下：

```
CREATE DATABASE company;
USE company;
```

【代码说明】在上述 SQL 语句中，首先创建数据库 company，然后选择该数据库。

【运行效果】执行上面的 SQL 语句，其结果如图 5.3 所示。

(2) 执行 SQL 语句 CREATE TABLE，创建表 t_dept，具体 SQL 语句如下：

```
CREATE TABLE t_dept (
    deptno INT,
    dname VARCHAR(20),
    loc VARCHAR(40)
);
```

【代码说明】上述 SQL 语句中创建了表 t_dept。对于表名标识符，不能是 MySQL 的关键字，如 CREATE、USE 等，建议表名标识符为 t_xxx 或 tab_xxx。表 t_dept 中有 3 个字段，分别为 INT 和 VARCHAR 类型，各属性之间用“，”符号隔开，最后一个属性后却不需要“，”符号。

【运行效果】执行上面的 SQL 语句，其结果如图 5.4 所示。

```
mysql> #创建数据库#
mysql> CREATE DATABASE company;
Query OK, 1 row affected (0.11 sec)

mysql> #选择数据库#
mysql> USE company;
Database changed
mysql> _
```

图 5.3 创建和选择数据库

```
mysql> #创建表格#
mysql> CREATE TABLE t_dept(
-> deptno INT,
-> dname VARCHAR(20),
-> loc VARCHAR(40)
-> );
Query OK, 0 rows affected (0.23 sec)

mysql>
```

图 5.4 创建表格 t_dept

注意：在创建表之前，需要选择数据库。如果没有选择数据库，创建表时就会出现“No database selected”错误，如图 5.5 所示。在创建表时，如果数据库中已经存在该表，则会出现“Table 't_dept' already exists”错误，如图 5.6 所示。

```
mysql> #创建表格#
mysql> CREATE TABLE t_dept<
-> deptno INT,
-> dname VARCHAR(20),
-> loc VARCHAR(40)
-> >;
ERROR 1046 (3D000): No database selected
```

图 5.5 没选择数据库错误

```
mysql> #创建表格#
mysql> CREATE TABLE t_dept<
-> deptno INT,
-> dname VARCHAR(20),
-> loc VARCHAR(40)
-> >;
ERROR 1050 (42S01): Table 't_dept' already exists
```

图 5.6 表已经存在错误

通过上述步骤，可以在数据库 company 中成功创建表对象 t_dept。

5.2.2 通过 SQLyog 客户端软件来创建表

在学习 MySQL 数据库阶段，可以通过 MySQL 数据库服务器自带的工具“MySQL Command Line Client”来创建表，该工具可以帮助大家尽快掌握关于创建表的语法。但是在数据库开发阶段，用户一般通过客户端软件 SQLyog 来创建表。

下面将通过一个具体的实例来说明如何通过 MySQL 客户端软件 SQLyog 创建表。

【实例 5-2】与实例 5-1 一样，在数据库 company 中创建名为 t_dept 的表。具体步骤如下：

(1) 首先连接数据库管理系统，然后右击“对象资源管理器”窗口中空白处，在弹出的快捷菜单中选择“创建数据库”命令，打开“创建数据库”对话框，“创建数据库”对话框的具体信息如图 5.7 所示。

MySQL 数据库应用从入门到精通

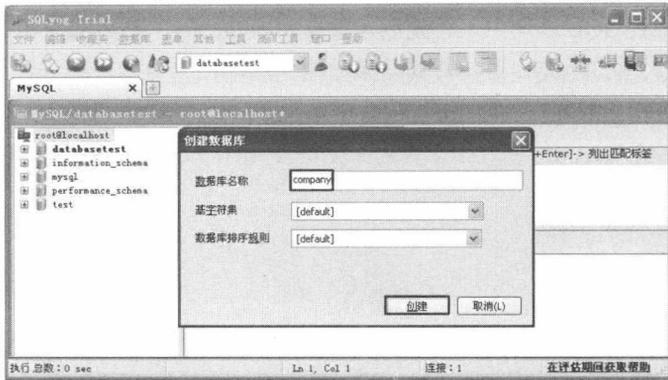


图 5.7 “创建数据库”对话框

(2) 在“对象资源管理器”窗口中，右击 company 数据库，然后选择“创建”→“表”命令，如图 5.8 所示。

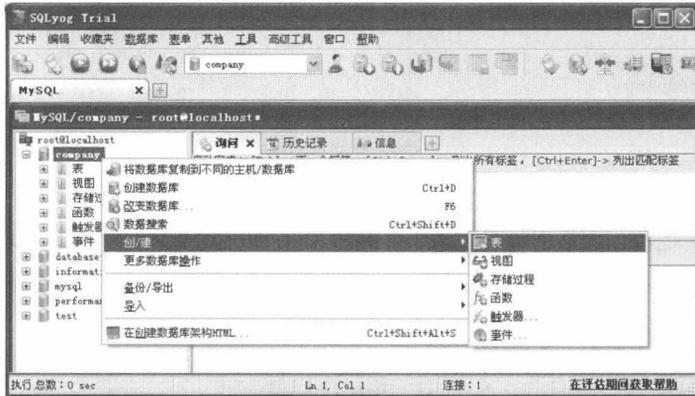


图 5.8 “表”命令

(3) 在打开的“新表”窗口中，设置相应信息，具体内容如图 5.9 所示。在“表名称”文本框中输入表的名称，其中“列选项卡”中的“列名”列设置字段名，“数据类型”列设置字段的类型，“长度”列设置类型的宽度。



图 5.9 表的详细信息

(4) 在“新表”窗口中单击“保存”按钮，实现创建表 t_dept。在“对象资源管理器”窗口中选择 company 数据库，然后单击“刷新”按钮，则会在“表”节点显示表 t_dept，如图 5.10 所示。

通过上述步骤，可以在数据库 company 中成功创建表对象 t_dept。对于 SQLyog 工具，除了可以通过以上步骤（向导方式）创建表外，还可以在“询问”窗口中输入创建表的 SQL 语句，然后单击工具栏中的“执行查询”按钮，可以实现表的创建，如图 5.11 所示。



图 5.10 创建表成功

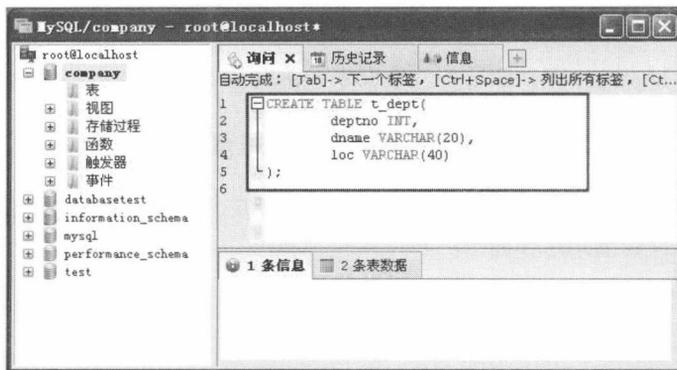


图 5.11 输入 SQL 语句

5.3 查看表结构

当创建完后，经常需要查看表信息。那么如何在 MySQL 软件中查看表信息呢？查看帮助文档，可以发现许多实现查看表信息的语句，例如 DESCRIBE、SHOW CREATE TABLE 等。为了便于讲解，本节将通过各种语句来查看数据库 company 中名为 t_dept 的表对象信息。

5.3.1 DESCRIBE 语句查看表定义

创建完表，如果需要查看一下表的定义，可以通过执行 SQL 语句 DESCRIBE 来实现，其语法形式如下：

```
DESCRIBE table_name
```

上述语句中 table_name 参数表示所要查看表对象定义信息的名字。

【实例 5-3】 执行 SQL 语句 DESCRIBE，查看数据库 company 中创建名为 t_dept 表时的定义信息。具体步骤如下：

(1) 执行 SQL 语句 USE，选择数据库 company，具体 SQL 语句如下：

```
USE company;
```

【运行效果】 执行上面的 SQL 语句，其结果如图 5.12 所示。

(2) 执行 SQL 语句 DESCRIBE，查看表 t_dept 定义信息，具体 SQL 语句如下：

```
DESCRIBE t_dept;
```

【运行效果】 执行上面的 SQL 语句，其结果如图 5.13 所示。

```
mysql> #选择数据库#
mysql> USE company;
```

图 5.12 选择数据库

```
mysql> #查看表定义#
mysql> DESCRIBE t_dept;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| deptno | int(11)       | YES  |     | NULL    |       |
| dname  | varchar(20)   | YES  |     | NULL    |       |
| loc    | varchar(40)   | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)

mysql>
```

图 5.13 查看表定义信息

通过上述步骤，即可查看数据库 company 中表对象 t_dept 的定义信息。

5.3.2 SHOW CREATE TABLE 语句查看表详细定义

创建完表，如果需要查看表结构的详细定义，可以通过执行 SQL 语句 SHOW CREATE TABLE 来实现，其语法形式如下：

```
SHOW CREATE TABLE table_name;
```

上述语句中 table_name 参数表示所要查看表定义的名字。

【实例 5-4】执行 SQL 语句 SHOW CREATE TABLE，查看数据库 company 中名为 t_dept 表的详细信息。具体步骤如下：

(1) 执行 SQL 语句 USE，选择数据库 company，具体 SQL 语句如下：

```
USE company;
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.14 所示。

(2) 执行 SQL 语句 SHOW CREATE TABLE，查看表 t_dept 定义，具体 SQL 语句如下：

```
SHOW CREATE TABLE t_dept \G
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.15 所示。

```
mysql> #选择数据库#
mysql> USE company;
```

图 5.14 选择数据库

```
mysql> #查看表详细定义#
mysql> SHOW CREATE TABLE t_dept \G
+-----+-----+-----+-----+-----+-----+
| Table: t_dept
| Create Table: CREATE TABLE `t_dept` (
|   `deptno` int(11) DEFAULT NULL,
|   `dname` varchar(20) DEFAULT NULL,
|   `loc` varchar(40) DEFAULT NULL
| ) ENGINE=InnoDB DEFAULT CHARSET=gbk
| row in set (0.00 sec)

mysql>
```

图 5.15 查看表详细定义

注意：在显示表详细定义信息时，可以使用“;”、“\g”和“\G”符号来结束。为了让结果显示的更加美观，便于用户查看，最好使用“\G”符号来结束。

通过上述步骤，即可查看数据库 company 中表对象 t_dept 的详细信息。

5.3.3 通过 SQLyog 软件来查看表信息

为了能够尽快掌握查看表的各种语句，需要尽可能多地使用 MySQL 数据库服务器自带的工具“MySQL Command Line Client”来查看表信息，但是客户端软件 SQLyog 可以更容易、更简单地查看表对象的各种信息。

在客户端软件 SQLyog 中，不仅可以在“询问”窗口中运行各种查看表语句来查看表对象，而

且还可以通过查看表对象来查看表的各种信息，具体步骤如下：

(1) 连接 MySQL 服务器，然后在“对象资源管理器”窗口中，选中表对象 t_dept，如图 5.16 所示。

(2) 在“信息”窗口就会显示表对象 t_dept 的具体信息，在该窗口中可以通过两种方式来显示，分别为 HTML 和“文本/详细”方式。HTML 显示方式如图 5.17 所示，“文本/详细”显示方式如图 5.18 所示。



图 5.16 选择表对象 t_dept

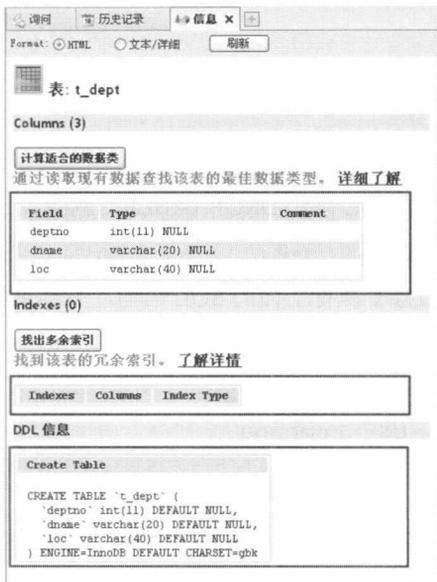


图 5.17 HTML 显示方式

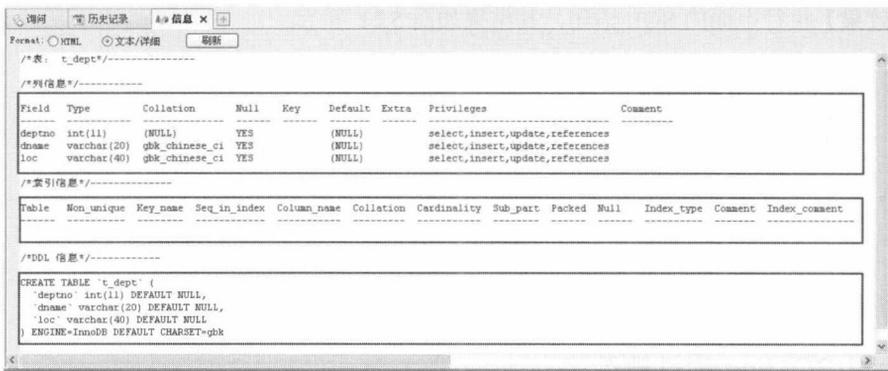


图 5.18 “文本/详细”显示方式

5.4 删除表

表的操作包括创建表、查看表、删除表和修改表。所谓删除表就是指删除数据库中已经存在的表。在具体删除表时，会直接删除表中所保存的所有数据，因此在删除表时应该非常小心。

5.4.1 删除表的语法形式

查看帮助文档发现，在 MySQL 数据库管理系统中删除表通过 SQL 语句 DROP TABLE 来实现，其语法形式如下：

```
DROP TABLE table_name
```

上述语句中 table_name 参数表示所要删除的表名字，所要删除的表必须是数据库中已经存在的表。

【实例 5-5】执行 SQL 语句 DROP TABLE，删除数据库 company 中名为 t_dept 的表。具体步骤如下：

(1) 执行 SQL 语句 USE，选择数据库 company，具体 SQL 语句如下：

```
USE company;
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.19 所示。

(2) 执行 SQL 语句 DROP TABLE，删除表 t_dept，具体 SQL 语句如下：

```
DROP TABLE t_dept;
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.20 所示。

```
mysql> #选择数据库#
mysql> USE company;
```

图 5.19 选择数据库

```
mysql> #删除表#
mysql> DROP TABLE t_dept;
Query OK, 0 rows affected (0.09 sec)

mysql> _
```

图 5.20 删除表

(3) 为了检验数据库 company 中是否还存在 t_dept 表，执行 SQL 语句 DESCRIBE，具体 SQL 语句内容如下：

```
DESCRIBE t_dept;
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.21 所示。

```
mysql> #查看表#
mysql> DESCRIBE t_dept;
ERROR 1146 (42S02): Table 'company.t_dept' doesn't exist
mysql> _
```

图 5.21 查看表

执行结果显示，t_dept 表已经不存在，则表示成功删除 t_dept 表。

5.4.2 通过 SQLyog 软件删除表

在客户端软件 SQLyog 中，不仅可以通过在“询问”窗口中执行 DROP TABLE 语句来删除表，而且还可以通过向导来实现，具体步骤如下：

(1) 在“对象资源管理器”窗口中，单击数据库 company 中表节点前的加号，然后右击“t_dept”节点，从弹出的快捷菜单中选择“更多表操作”→“从数据库删除表”命令，如图 5.22 所示。

(2) 弹出对话框来确认是否删除表，如图 5.23 所示。单击“是”按钮后，“对象资源管理器”窗口数据库 company 中“表节点”里就没有任何表对象，如图 5.24 所示。

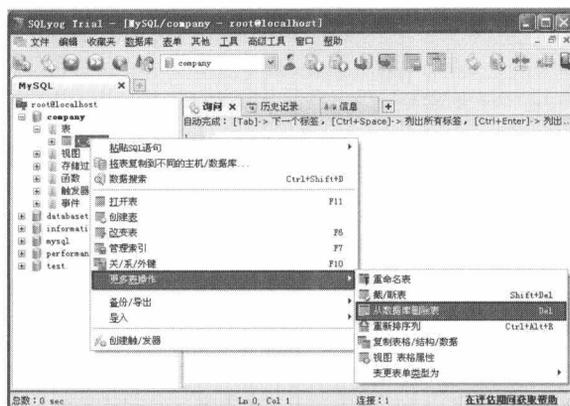


图 5.22 选择删除表命令

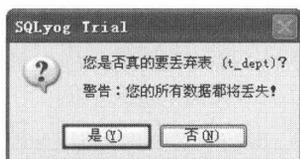


图 5.23 查看表对象图



图 5.24 删除表成功

通过上述步骤，即可成功删除数据库 company 中的表对象 t_dept。

5.5 修改表

对于已经创建好的表，当使用一段时间后，就需要进行一些结构上的修改，即表的修改操作。该操作的解决方案是先将表删除，然后再按照新的表定义重建表。但是这种解决方案有问题，即如果表中已经存在大量数据，那么重建表后还需要做许多额外工作，例如数据的重载等。为了解决上述问题，MySQL 数据库提供“ALTER TABLE”语句实现修改表结构。

5.5.1 修改表名

在数据库中，可以通过表名来区分不同的表，因为表名在数据库中是唯一的，不能重复。查看帮助文档发现，在 MySQL 数据库管理系统中修改表名可以通过 SQL 语句 ALTER TABLE 来实现，其语法形式如下：

```
ALTER TABLE old_table_name RENAME [TO] new_table_name
```

在上述语句中，old_table_name 参数表示所要修改表的名字，new_table_name 参数为修改后的新名字。所要操作的表对象必须在数据库中已经存在。

【实例 5-6】 执行 SQL 语句 ALTER TABLE，修改数据库 company 中 t_dept 表的名称为 tab_dept。具体步骤如下：

(1) 执行 SQL 语句 USE，选择数据库 company，具体 SQL 语句如下：

```
USE company;
```

【运行效果】 执行上面的 SQL 语句，其结果如图 5.25 所示。

(2) 执行 SQL 语句 ALTER TABLE，修改表 t_dept 的名字为 tab_dept，具体 SQL 语句如下：

```
ALTER TABLE t_dept
    RENAME tab_dept;
```

【运行效果】 执行上面的 SQL 语句，其结果如图 5.26 所示。

```
mysql> #选择数据库#
mysql> USE company;
```

图 5.25 选择数据库

```
mysql> #修改表的名字#
mysql> ALTER TABLE t_dept
-> RENAME tab_dept;
Query OK, 0 rows affected (0.20 sec)

mysql>
```

图 5.26 修改表的名字

(3) 为了检验数据库 company 中是否已经修改 t_dept 表为 tab_dept 表，执行 SQL 语句 DESC，具体 SQL 语句内容如下：

```
DESC t_dept;
```

和

```
DESC tab_dept;
```

【代码说明】 上述两句 SQL 内容，主要用来实现查看表对象 t_dept 和 tab_dept。

【运行效果】 执行上面的 SQL 语句，其结果如图 5.27 所示。

执行结果显示，t_dept 表已经不存在，已经修改名为 tab_dept 的表，并且 tab_dept 表的结构与 t_dept 表的结构完全一致。

```
mysql> #查看表#
mysql> DESC t_dept;
ERROR 1146 (42S02): Table 'company.t_dept' doesn't exist
mysql> DESC tab_dept;
+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+
| deptno | int(11)       | YES  |     | NULL    |      |
| dname  | varchar(20)   | YES  |     | NULL    |      |
| loc   | varchar(40)   | YES  |     | NULL    |      |
+-----+
3 rows in set (0.02 sec)

mysql> _
```

图 5.27 查看表信息

5.5.2 增加字段

对于表，可以看成是由列和行来构成的，其中“列”经常被称为字段。根据创建表的语法可以发现，字段是由字段名和数据类型进行定义的。本节将详细介绍如何为一个已经存在的表增加字段。

1. 在表的最后一个位置增加字段

查看帮助文档发现，在 MySQL 数据库管理系统中增加字段通过 SQL 语句 ALTER TABLE 来实现，其语法形式如下：

```
ALTER TABLE table_name
    ADD 属性名 属性类型
```

在上述语句中，参数 table_name 表示所要修改表的名字，“属性名”参数为所要增加字段的名称

称，“属性类型”为所要增加字段能存储数据的数据类型。如果该语句执行成功，字段将增加到所有字段的最后一个位置。

【实例 5-7】 执行 SQL 语句 ALTER TABLE，为数据库 company 中 t_dept 表增加一个名为 descri，类型为 VARCHAR 的字段，所增加字段在表中所有字段的最后一个位置。具体步骤如下：

(1) 执行 SQL 语句 USE，选择数据库 company，具体 SQL 语句如下：

```
USE company;
```

然后查看已经存在表 t_dept 的定义信息，具体 SQL 语句如下：

```
DESC t_dept;
```

【运行效果】 执行上面的 SQL 语句，其结果如图 5.28 和图 5.29 所示。

```
mysql> #选择数据库#
mysql> USE company;
```

图 5.28 选择数据库

```
mysql> #查看表#
mysql> DESC t_dept:
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| deptno | int(11) | YES | | NULL | |
| dname | varchar(20) | YES | | NULL | |
| loc | varchar(40) | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

图 5.29 查看表定义

(2) 执行 SQL 语句 ALTER TABLE，增加一个名为 descri 的字段，具体 SQL 语句如下：

```
ALTER TABLE t_dept
ADD descri VARCHAR(20);
```

【运行效果】 执行上面的 SQL 语句，其结果如图 5.30 所示。

(3) 为了检验数据库 company 中的 t_dept 表是否添加 descri 字段，执行 SQL 语句 DESC，具体 SQL 语句内容如下：

```
DESC t_dept;
```

【运行效果】 执行上面的 SQL 语句，其结果如图 5.31 所示。

```
mysql> #增加字段#
mysql> ALTER TABLE t_dept
-> ADD descri VARCHAR(20);
Query OK, 0 rows affected (0.70 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql>
```

图 5.30 添加字段

```
mysql> #查看表#
mysql> DESC t_dept:
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| deptno | int(11) | YES | | NULL | |
| dname | varchar(20) | YES | | NULL | |
| loc | varchar(40) | YES | | NULL | |
| descri | varchar(20) | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

图 5.31 查看表信息

执行结果显示，与图 5.29 相比，表 t_dept 中已经增加了一个名为 descri 的字段，并且该字段还在表的最后一个位置，即增加字段成功。

2. 在表的第一个位置增加字段

通过 SQL 语句 ALTER TABLE 来实现增加字段时，如果不想让所增加的字段在所有字段的最后一个位置，可以通过 FIRST 关键字使所增加的字段在表中所有字段的第一个位置，具体的 SQL 语

MySQL 数据库应用从入门到精通

句语法形式如下：

```
ALTER TABLE table_name
    ADD 属性名 属性类型 FIRST;
```

在上述语句中，多了一个关键字 FIRST，表示所有增加的字段在所有字段之前，即在表中第一个位置。

【实例 5-8】 执行 SQL 语句 ALTER TABLE，为数据库 company 中 t_dept 表中的第一位置，增加一个名称为 descri，类型为 VARCHAR 的字段，所增加字段在表所有字段的第一个位置。具体步骤如下：

(1) 执行 SQL 语句 USE，选择数据库 company，具体 SQL 语句如下：

```
USE company;
```

然后查看已经存在表 t_dept 的定义信息，具体 SQL 语句如下：

```
DESC t_dept;
```

【运行效果】 执行上面的 SQL 语句，其结果如图 5.32 和图 5.33 所示。

```
mysql> #选择数据库#
mysql> USE company;
```

图 5.32 选择数据库

```
mysql> #查看表#
mysql> DESC t_dept:
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| deptno | int(11)       | YES  |     | NULL    |       |
| dname  | varchar(20)   | YES  |     | NULL    |       |
| loc    | varchar(40)   | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

图 5.33 查看表结构

(2) 执行 SQL 语句 ALTER TABLE，增加一个名为 descri 的字段，具体 SQL 语句如下：

```
ALTER TABLE t_dept
    ADD descri VARCHAR(20) FIRST;
```

【运行效果】 执行上面的 SQL 语句，其结果如图 5.34 所示。

(3) 为了检验数据库 company 中的 t_dept 表是否添加 descri 字段，执行 SQL 语句 DESCRIBE，具体 SQL 语句内容如下：

```
DESCRIBE t_dept;
```

【运行效果】 执行上面的 SQL 语句，其结果如图 5.35 所示。

```
mysql> #添加字段#
mysql> ALTER TABLE t_dept
-> ADD descri VARCHAR(20) FIRST;
Query OK, 0 rows affected (0.50 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql>
```

图 5.34 添加字段

```
mysql> #查看表#
mysql> DESC t_dept:
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| descri | varchar(20)   | YES  |     | NULL    |       |
| deptno | int(11)       | YES  |     | NULL    |       |
| dname  | varchar(20)   | YES  |     | NULL    |       |
| loc    | varchar(40)   | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)

mysql> _
```

图 5.35 查看表信息

执行结果显示，与图 5.33 相比，表 t_dept 中已经增加了一个名为 descri 的字段，并且该字段还在表的第一个位置，即增加字段成功。

3. 在表的指定字段之后增加字段

通过 SQL 语句 ALTER TABLE 来实现增加字段时，除了可以在表的第一个位置或最后一个位置增加字段外，还可以通过关键字 AFTER 在指定的字段之后添加字段，具体的 SQL 语句语法形式如下：

```
ALTER TABLE table name
    ADD 属性名 属性类型
    AFTER 属性名
```

在上述语句中，多了一个关键字 AFTER，表示所有增加的字段在该关键字所指定字段之后。

【实例 5-9】执行 SQL 语句 ALTER TABLE，为数据库 company 中 t_dept 表增加一个名称为 descri，类型为 VARCHAR 的字段，所增加字段在 deptno 字段之后位置。具体步骤如下：

(1) 执行 SQL 语句 USE，选择数据库 company，具体 SQL 语句如下：

```
USE company;
```

然后查看已经存在表 t_dept 的定义信息，具体 SQL 语句如下：

```
DESC t_dept;
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.36 和图 5.37 所示。

```
mysql> #选择数据库#
mysql> USE company;
```

图 5.36 选择数据库

```
mysql> #查看表#
mysql> DESC t_dept:
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| deptno | int(11) | YES | | NULL | |
| dname | varchar(20) | YES | | NULL | |
| loc | varchar(40) | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

图 5.37 查看表

(2) 执行 SQL 语句 ALTER TABLE，增加一个名为 descri 的字段，具体 SQL 语句如下：

```
ALTER TABLE t_dept
    ADD descri VARCHAR(20)
    AFTER deptno;
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.38 所示。

(3) 为了检验数据库 company 中的 t_dept 表是否添加 descri 字段，执行 SQL 语句 DESC，具体 SQL 语句内容如下：

```
DESC t_dept;
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.39 所示。

```
mysql> #添加字段#
mysql> ALTER TABLE t_dept
-> ADD descri UARCHAR(20)
-> AFTER deptno;
Query OK, 0 rows affected (0.44 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> _
```

图 5.38 添加字段

```
mysql> #查看表#
mysql> DESC t_dept:
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| deptno | int(11) | YES | | NULL | |
| descri | varchar(20) | YES | | NULL | |
| dname | varchar(20) | YES | | NULL | |
| loc | varchar(40) | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.01 sec)

mysql>
```

图 5.39 查看表信息

MySQL 数据库应用从入门到精通

执行结果显示，与图 5.37 相比，表 `t_dept` 中已经增加了一个名为 `descri` 的字段，并且该字段还在表 `deptno` 字段之后的位置，即增加字段成功。

5.5.3 删除字段

对于表，既然可以在修改表时进行字段的增加操作，那么也可以在修改表时进行字段的删除。所谓删除字段是指删除已经在表中定义好的某个字段。即在创建好的表格中，发现某个字段需要删除，查看帮助文档发现，在 MySQL 数据库管理系统中删除字段通过 SQL 语句 `ALTER TABLE` 来实现，其语法形式如下：

```
ALTER TABLE table_name
DROP 属性名
```

上述语句中 `table_name` 参数表示所要修改表的名字，“属性名”参数表示所要删除的字段名。

【实例 5-10】执行 SQL 语句 `ALTER TABLE`，为数据库 `company` 中 `t_dept` 表删除名为 `deptno` 的字段。具体步骤如下：

(1) 执行 SQL 语句 `USE`，选择数据库 `company`，具体 SQL 语句如下：

```
USE company;
```

然后查看已经存在表 `t_dept` 的定义信息，具体 SQL 语句如下：

```
DESC t_dept;
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.40 和图 5.41 所示。

```
mysql> #选择数据库#
mysql> USE company;
```

图 5.40 选择数据库

```
mysql> #查看表#
mysql> DESC t_dept;
+-----+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| deptno | int(11)   | YES  |     | NULL    |       |
| dname  | varchar(20) | YES  |     | NULL    |       |
| loc    | varchar(40) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

图 5.41 查看表

(2) 执行 SQL 语句 `ALTER TABLE`，删除名为 `deptno` 的字段，具体 SQL 语句如下：

```
ALTER TABLE t_dept
DROP deptno;
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.42 所示。

(3) 为了检验数据库 `company` 中的 `t_dept` 表是否删除 `deptno` 字段，执行 SQL 语句 `DESCRIBE`，具体 SQL 语句内容如下：

```
DESCRIBE t_dept;
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.43 所示。

执行结果显示，与图 5.41 相比，表 `t_dept` 中已经删除了一个名为 `deptno` 的字段。

```
mysql> #删除字段deptno#
mysql> ALTER TABLE t_dept
-> DROP deptno;
Query OK, 0 rows affected (0.44 sec)
Records: 0 Duplicates: 0 Warnings: 0
mysql>
```

图 5.42 删除字段

```
mysql> #查看表#
mysql> DESC t_dept;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| dname | varchar(20)   | YES  |     | NULL    |      |
| loc   | varchar(40)   | YES  |     | NULL    |      |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.02 sec)

mysql>
```

图 5.43 查看表信息

5.5.4 修改字段

根据创建表的语法可以发现，字段是由字段名和数据类型来进行定义的，如果来实现修改字段，除了可以修改字段名外，还可以实现修改字段所能存储的数据类型。由于一个表中会拥有许多字段，因此还可以实现修改字段的顺序。本节将详细介绍如何修改字段。

1. 修改字段的数据类型

查看帮助文档发现，在 MySQL 数据库管理系统中修改字段类型通过 SQL 语句 ALTER TABLE 来实现，其语法形式如下：

```
ALTER TABLE table_name
MODIFY 属性名 数据类型
```

上述语句中 table_name 参数表示所要修改表的名字，“属性名”参数为所要修改的字段名，“数据类型”为修改后的数据类型。

【实例 5-11】执行 SQL 语句 ALTER TABLE，在数据库 company 的表 t_dept 中，将 deptno 字段的数据类型由原来的 INT(11)类型修改为 VARCHAR(20)类型。具体步骤如下：

(1) 执行 SQL 语句 USE，选择数据库 company，具体 SQL 语句如下：

```
USE company;
```

然后查看已经存在表 t_dept 的定义信息，具体 SQL 语句如下：

```
DESC t_dept;
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.44 和图 5.45 所示。

```
mysql> #选择数据库#
mysql> USE company;
```

图 5.44 选择数据库

```
mysql> #查看表#
mysql> DESC t_dept;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| deptno | int(11)       | YES  |     | NULL    |      |
| dname  | varchar(20)   | YES  |     | NULL    |      |
| loc    | varchar(40)   | YES  |     | NULL    |      |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

图 5.45 查看表

(2) 执行 SQL 语句 ALTER TABLE，修改 deptno 字段的类型为 VARCHAR(20)，具体 SQL 语句如下：

```
ALTER TABLE t_dept
MODIFY deptno VARCHAR(20);
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.46 所示。

MySQL 数据库应用从入门到精通

(3) 为了检验数据库 company 的 t_dept 表中，字段 deptno 的类型是否修改为 VARCHAR(20)，执行 SQL 语句 DESC，具体 SQL 语句内容如下：

```
DESC t_dept;
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.47 所示。

```
mysql> #修改字段类型#
mysql> ALTER TABLE t_dept
-> MODIFY deptno VARCHAR(20);
Query OK, 0 rows affected (0.38 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql>
```

图 5.46 修改字段类型

```
mysql> #查看表#
mysql> DESC t_dept;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| deptno | varchar(20)  | YES  |     | NULL    |       |
| dname  | varchar(20)  | YES  |     | NULL    |       |
| loc    | varchar(40)  | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> _
```

图 5.47 查看表信息

执行结果显示，与图 5.44 相比，表 t_dept 中字段 deptno 的类型已经修改成 VARCHAR(20) 类型。

2. 修改字段的名字

查看帮助文档发现，在 MySQL 数据库管理系统中修改字段名称通过 SQL 语句 ALTER TABLE 来实现，其语法形式如下：

```
ALTER TABLE table_name
CHANGE 旧属性名 新属性名 旧数据类型
```

上述语句中 table_name 参数表示所要修改表的名字，“旧属性名”参数表示所要修改的字段名，“新属性名”参数表示所要修改成的字段名。

【实例 5-12】执行 SQL 语句 ALTER TABLE，在数据库 company 的表 t_dept 中，将名为 loc 的字段修改为 location。具体步骤如下：

(1) 执行 SQL 语句 USE，选择数据库 company，具体 SQL 语句如下：

```
USE company;
```

然后查看已经存在表 t_dept 的定义信息，具体 SQL 语句如下：

```
DESC t_dept;
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.48 和图 5.49 所示。

```
mysql> #选择数据库#
mysql> USE company;
```

图 5.48 选择数据库

```
mysql> #查看表#
mysql> DESC t_dept;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| deptno | int(11)       | YES  |     | NULL    |       |
| dname  | varchar(20)  | YES  |     | NULL    |       |
| loc    | varchar(40)  | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

图 5.49 查看表

(2) 执行 SQL 语句 ALTER TABLE，修改字段 loc 的名字为 location，具体 SQL 语句如下：

```
ALTER TABLE t_dept
CHANGE loc location VARCHAR(40);
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.50 所示。

(3) 为了检验数据库 company 的表 t_dept 中，字段 loc 是否已修改为 location，执行 SQL 语句 DESC，具体 SQL 语句内容如下：

```
DESC t_dept;
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.51 所示。

```
mysql> #修改字段名称#
mysql> ALTER TABLE t_dept
-> CHANGE loc location VARCHAR(40);
Query OK, 0 rows affected (0.55 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

图 5.50 修改字段名称

```
mysql> #查看表#
mysql> DESC t_dept;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| deptno | varchar(20) | YES  |     | NULL    |       |
| dname  | varchar(20) | YES  |     | NULL    |       |
| location | varchar(40) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

图 5.51 查看表信息

执行结果显示，与图 5.49 相比，表 t_dept 中已经不存在字段 loc，而该字段已经修改成为名为 location 的字段。

3. 同时修改字段的名称和属性

通过关键字 MODIFY 可以修改字段的数据类型，通过关键字 CHANGE 可以修改字段的名称，那么有没有一个关键字能够同时修改字段的名称和数据类型呢？查看帮助文档发现，在 MySQL 数据库管理系统中同时修改字段名称和数据类型，通过 SQL 语句 ALTER TABLE 来实现，其语法形式如下：

```
ALTER TABLE table_name
CHANGE 旧属性名 新属性名 新数据类型
```

上述语句中“新属性名”参数表示所要修改成的字段名，“新数据类型”为所要修改成的数据类型。

【实例 5-13】执行 SQL 语句 ALTER TABLE，在数据库 company 的表 t_dept 中，将名为 loc 的字段修改为 location，数据类型由原来的 VARCHAR(40)修改为 VARCHAR(20)。具体步骤如下：

(1) 执行 SQL 语句 USE，选择数据库 company，具体 SQL 语句如下：

```
USE company;
```

然后查看已经存在表 t_dept 的定义信息，具体 SQL 语句如下：

```
DESC t_dept;
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.52 和图 5.53 所示。

```
mysql> #选择数据库#
mysql> USE company;
```

图 5.52 选择数据库

```
mysql> #查看表#
mysql> DESC t_dept;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| deptno | int(11)       | YES  |     | NULL    |       |
| dname  | varchar(20)  | YES  |     | NULL    |       |
| loc    | varchar(40)  | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

图 5.53 查看表

(2) 执行 SQL 语句 ALTER TABLE，修改名为 deptno 的字段，具体 SQL 语句如下：

MySQL 数据库应用从入门到精通

```
ALTER TABLE t_dept
CHANGE loc location VARCHAR(20);
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.54 所示。

(3) 为了检验数据库 company 的 t_dept 表中，字段 loc 的名字是否已修改为 location，数据类型是否修改为 VARCHAR(20)，执行 SQL 语句 DESC，具体 SQL 语句内容如下：

```
DESC t_dept;
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.55 所示。

```
mysql> #修改字段名字和数据类型#
mysql> ALTER TABLE t_dept
-> CHANGE loc location VARCHAR(20);
Query OK, 0 rows affected (0.28 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

图 5.54 修改字段名字和数据类型

```
mysql> #查看表#
mysql> DESC t_dept:
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| deptno | varchar(20)   | YES  |     | NULL    |       |
| dname  | varchar(20)   | YES  |     | NULL    |       |
| location | varchar(20)  | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.02 sec)
```

图 5.55 查看表信息

执行结果显示，与图 5.53 相比，表 t_dept 中已经不存在字段 loc，而该字段已修改成名为 location 的字段，同时数据类型已经修改为 VARCHAR(20)。

4. 修改字段的顺序

查看帮助文档发现，在 MySQL 数据库管理系统中修改字段名称顺序，通过 SQL 语句 ALTER TABLE 来实现，其语法形式如下：

```
ALTER TABLE table_name
MODIFY 属性名 1 数据类型 FIRST|AFTER 属性名 2
```

上述语句中，table_name 参数表示所要修改表的名字，“属性名 1”参数表示所要调整顺序的字段名，“FIRST”参数表示将字段调整到表的第一个位置，“AFTER 属性名 2”参数表示将字段调整到属性名 2 字段位置之后。

注意：属性名 1 和属性名 2 必须是表中已经存在的字段名。

【实例 5-14】执行 SQL 语句 ALTER TABLE，在数据库 company 的表 t_dept 中，首先将名为 loc 的字段调整到表的第一个位置，然后把字段 deptno 调整到字段 dname 字段之后。具体步骤如下：

(1) 执行 SQL 语句 USE，选择数据库 company，具体 SQL 语句如下：

```
USE company;
```

然后查看已经存在表 t_dept 的定义信息，具体 SQL 语句如下：

```
DESC t_dept;
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.56 和图 5.57 所示。

```
mysql> #选择数据库#
mysql> USE company;
```

图 5.56 选择数据库

```
mysql> #查看表#
mysql> DESC t_dept:
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| deptno | int(11)       | YES  |     | NULL    |       |
| dname  | varchar(20)   | YES  |     | NULL    |       |
| loc    | varchar(40)   | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

图 5.57 查看表

(2) 执行 SQL 语句 ALTER TABLE，将字段 loc 调整到表的第一个位置，具体 SQL 语句如下：

```
ALTER TABLE t_dept
MODIFY loc VARCHAR(40) FIRST;
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.58 所示。

(3) 为了检验数据库 company 的 t_dept 表中，字段 loc 的位置是否已经为第一个位置，执行 SQL 语句 DESC，具体 SQL 语句内容如下：

```
DESC t_dept;
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.59 所示。

```
mysql> #查询表#
mysql> DESC t_dept:
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| loc   | varchar(40)  | YES  |     | NULL    |       |
| deptno | int(11)      | YES  |     | NULL    |       |
| dname | varchar(20)  | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> _
```

图 5.58 调整字段 loc 为第一个位置

```
mysql> #查询表#
mysql> DESC t_dept:
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| loc   | varchar(40)  | YES  |     | NULL    |       |
| deptno | int(11)      | YES  |     | NULL    |       |
| dname | varchar(20)  | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> _
```

图 5.59 查询表信息

执行结果显示，与图 5.53 相比，表 t_dept 中字段 loc 已经调整到表中第一个位置。

(4) 执行 SQL 语句 ALTER TABLE，将字段 deptno 调整到字段 dname 之后位置，具体 SQL 语句如下：

```
ALTER TABLE t_dept
MODIFY deptno INT(11) AFTER dname;
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.60 所示。

(5) 为了检验数据库 company 的 t_dept 表中，字段 deptno 是否调整到指定位置，执行 SQL 语句 DESC，具体 SQL 语句内容如下：

```
DESC t_dept;
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.61 所示。

```
mysql> #调整字段的位置#
mysql> ALTER TABLE t_dept
-> MODIFY deptno INT(11) AFTER dname;
Query OK, 0 rows affected (0.33 sec)
Records: 0 Duplicates: 0 Warnings: 0

mysql> _
```

图 5.60 调整字段 deptno 到指定位置

```
mysql> #查询表#
mysql> DESC t_dept:
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| loc   | varchar(40)  | YES  |     | NULL    |       |
| dname | varchar(20)  | YES  |     | NULL    |       |
| deptno | int(11)      | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> _
```

图 5.61 查询表信息

执行结果显示，与图 5.57 相比，表 t_dept 中已经调整字段 deptno 到字段 dname 字段之后位置。

5.6 通过 SQLyog 客户端软件修改表

在学习 MySQL 数据库阶段，可以通过 MySQL 数据库服务器自带的工具“MySQL Command Line Client”来修改表，该工具可以帮助大家尽快掌握关于修改表的语法。但是在数据库开发阶段，用户

MySQL 数据库应用从入门到精通

一般通过客户端软件 SQLyog 来修改表，即通过 SQL 语句实现修改表的功能，这完全可以通过工具向导来实现。

5.6.1 修改表名

在客户端软件 SQLyog 中，不仅可以通过在“询问”窗口中运行修改表名的 SQL 语句，而且还可以通过工具向导来实现修改表名操作，具体步骤如下：

(1) 在“对象资源管理器”窗口中，右击选中表对象 t_dept，然后在弹出的快捷菜单中选择“更多表操作”→“重命名表”命令，如图 5.62 所示。



图 5.62 选择“重命名表”命令

(2) 在弹出的“对象资源管理器”窗口中，关于表 t_dept 的名称就处于可编辑状态（见图 5.63），修改名称为 tab_dept 后，单击【Enter】键即可实现表名称的修改。最后“对象资源管理器”中的对象如图 5.64 所示。



图 5.63 处于可编辑状态



图 5.64 修改表名为 tab_dept

5.6.2 增加字段

在客户端软件 SQLyog 中，不仅可以通过在“询问”窗口中运行添加字段的 SQL 语句，而且还可以通过工具向导来实现添加字段操作，具体步骤如下：

(1) 在“对象资源管理器”窗口中，右击选中表对象 t_dept，然后在弹出的快捷菜单中选择“改变表”命令，如图 5.65 所示。

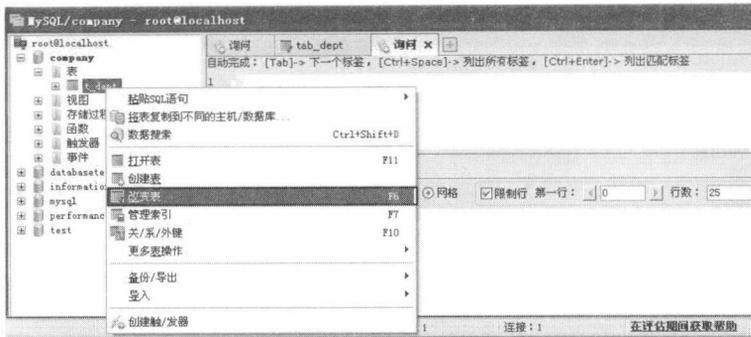


图 5.65 选择“改变表”命令

(2) 弹出 `t_dept` 的窗口 (见图 5.66), 如果想添加一个数据类型为 `VARCHAR(20)` 的字段 `descri`, 可以在“列选项卡”中的最后一行填写, 详细信息如图 5.67 所示。



图 5.66 表 `t_dept` 窗口



图 5.67 添加空白行

(3) 当确认所填写的字段信息无误后, 单击“保存”按钮, 弹出确认对话框, 如图 5.68 所示, 在该对话框中单击“确定”按钮即可实现字段的添加。

最后, 在添加字段成功后, 打开表 `t_dept` 后, 就会发现所添加的字段 `descri`, 如图 5.69 所示。

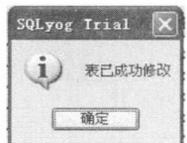


图 5.68 添加字段成功

MySQL 数据库应用从入门到精通



图 5.69 添加字段成功

通过 SQL 语句增加字段时，默认将字段增加到表的最后位置，同时还可以将字段增加到表的第一位置或表中指定的任意位置。在客户端软件 SQLyog 中，如果想将字段添加到表的第一位置，需要经过如下步骤来实现。

首先在 t_dept 窗口的“列选项卡”中，选择第一行（见图 5.70），然后单击“插入新行”按钮，即可在第一行的上面添加一个空白行（见图 5.71），最后添加相应的字段信息就可以实现在表的第一位置添加字段。

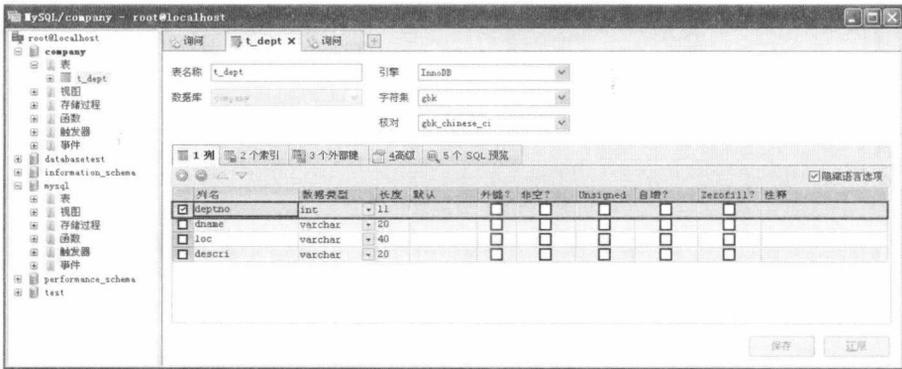


图 5.70 选择第一行



图 5.71 增加新空白行

在客户端软件 SQLyog 中，如果想将字段添加到指定的位置，需要经过如下步骤来实现。

首先在 t_dept 窗口的“列选项卡”中，选择要指定的行（列名为 dname 的行，见图 5.72），然后单击“插入新行”按钮，即可在该行的上面添加一个空白行（见图 5.73），最后添加相应的字段信息就可以实现在指定的位置上添加字段。

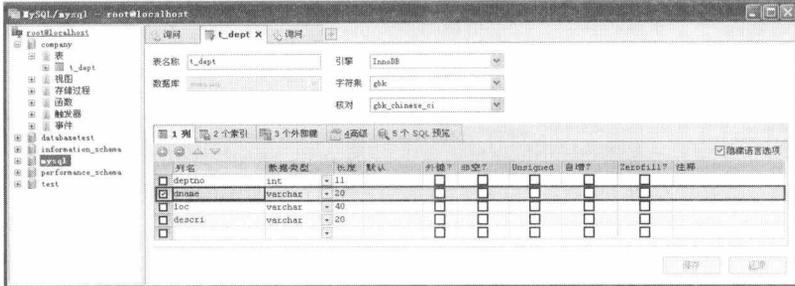


图 5.72 选择指定位置

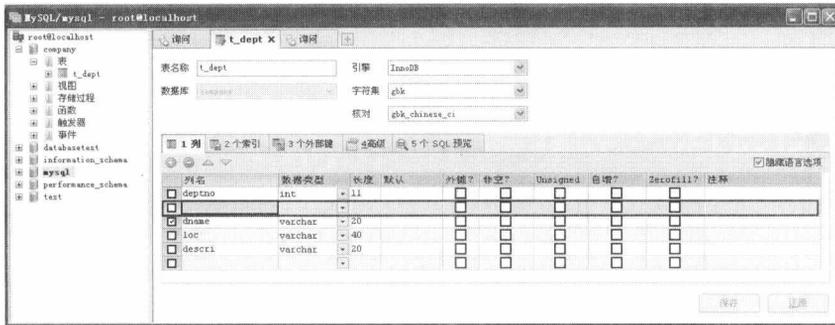


图 5.73 增加新空白行

通过上述步骤，即可将字段增加到表的第一位置或表中指定的任意位置。

5.6.3 删除字段

在客户端软件 SQLyog 中，不仅可以通过在“询问”窗口中运行删除字段的 SQL 语句，而且还可以通过工具向导来实现删除字段操作，具体步骤如下：

(1) 在“对象资源管理器”窗口中，右击选中表对象 t_dept，在弹出的快捷菜单中选择“改变表”命令，如图 5.74 所示。

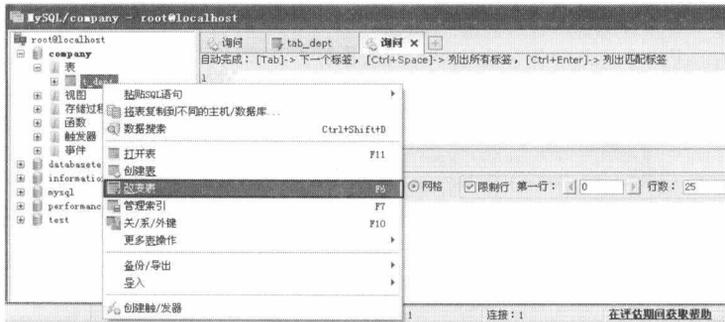


图 5.74 选择“改变表”命令

MySQL 数据库应用从入门到精通

(2) 弹出关于 t_dept 的窗口，如图 5.75 所示，如果想删除字段 deptno，可以在“列选项卡”中选择字段 deptno，如图 5.76 所示。



图 5.75 表 t_dept 窗口



图 5.76 选择字段 deptno

(3) 单击“删除选定行”按钮 ，弹出确认对话框，在该对话框中单击“是”按钮，即可实现字段的删除，如图 5.77 所示。

(4) 单击“保存”按钮保存当前的操作，最后 t_dept 表的字段如图 5.78 所示。

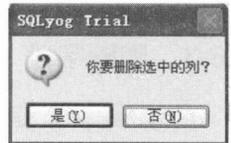


图 5.77 确认删除列

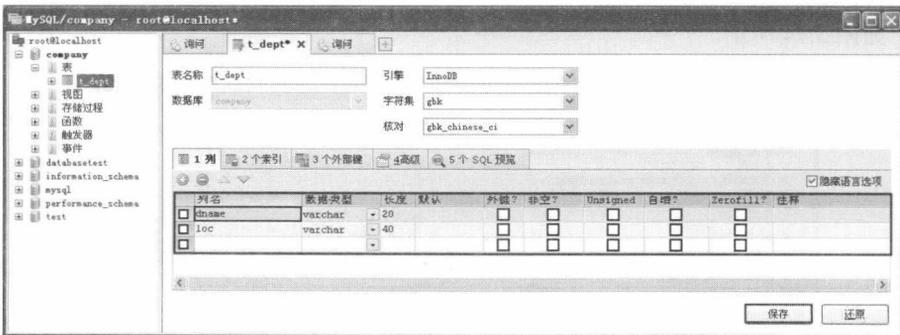


图 5.78 删除字段 deptno

通过上述步骤，可以成功删除字段 deptno。

5.6.4 修改字段

在 MySQL 软件中，可以通过相关 SQL 语句修改字段的名称、修改字段的数据类型和修改字段的顺序，同样在客户端软件 SQLyog 中，也可以通过向导来实现这些操作。

1. 修改字段的名称和数据类型

【实例 5-15】在客户端软件 SQLyog 中，在数据库 company 的表 t_dept 中，将名为 loc 的字段修改为 location，数据类型由原来的 VARCHAR(40)修改为 VARCHAR(20)。具体步骤如下：

(1) 在“对象资源管理器”窗口中，右击选中表对象 t_dept，然后在出现的命令中选择“改变表”命令如图 5.79 所示，弹出关于 t_dept 的窗口，如图 5.80 所示。



图 5.79 选择“改变表”操作

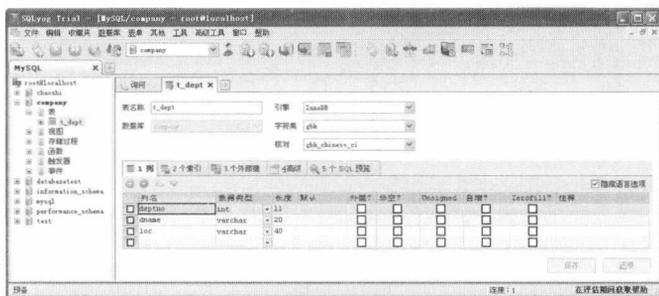


图 5.80 表 t_dept 窗口

(2) 在 t_dept 的窗口中，如果想修改字段 loc 为 location，只需在“列名”单元格上双击，该单元格就会处于编辑状态，如图 5.81 所示，然后直接修改字段名为 location，最后单击【Enter】键即可实现字段名的修改，如图 5.82 所示。

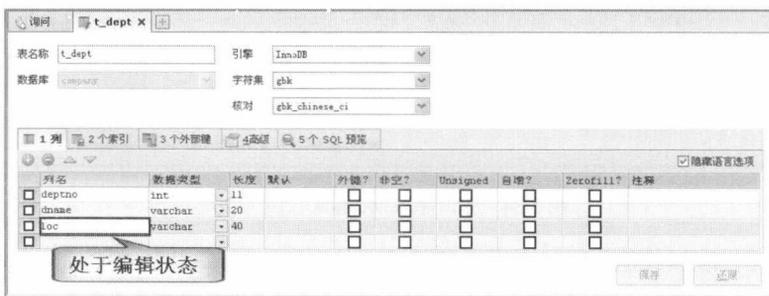


图 5.81 字段 loc 列名处于编辑状态

MySQL 数据库应用从入门到精通

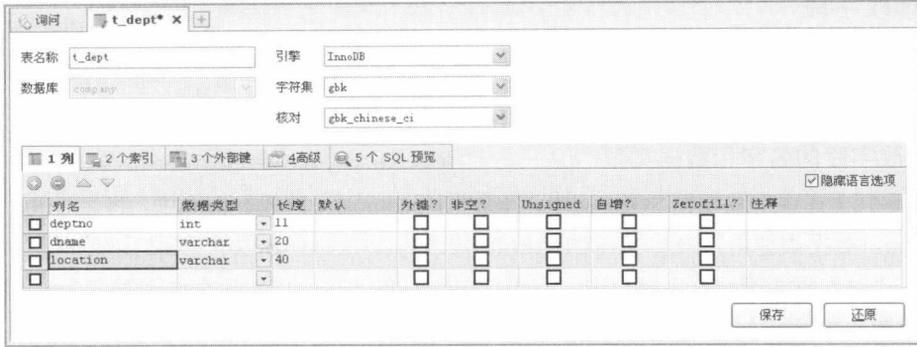


图 5.82 修改字段 loc 列名为 location

(3) 在 `t_dept` 的窗口中，如果想修改字段 `location` 的类型为 `VARCHAR(20)`，只需在“长度”单元格上双击，该单元格就会处于编辑状态，如图 5.83 所示，然后直接修改长度名为 20，最后单击【Enter】键即可实现字段类型的修改，如图 5.84 所示。

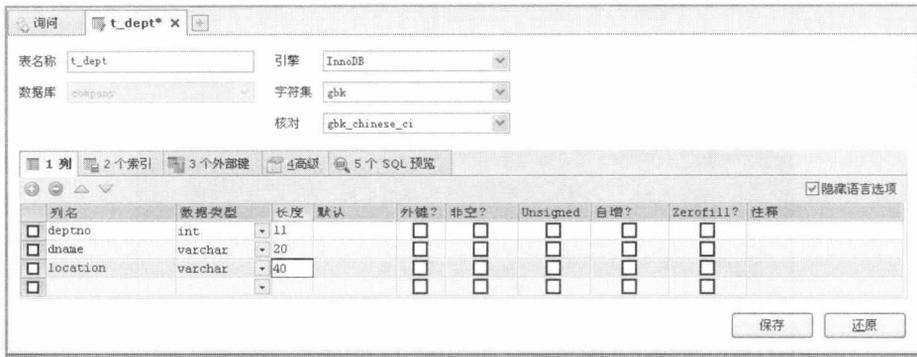


图 5.83 字段 location 长度处于编辑状态



图 5.84 修改字段 location 长度为 20

(4) 如果想使修改后的值生效，只要单击“保存”按钮，就会出现确认对话框，在该对话框中单击“确定”按钮，即可实现字段名字和类型的修改，如图 5.85 所示。



图 5.85 使修改生效

2. 修改字段的顺序

【实例 5-16】在客户端软件 SQLyog 中，在数据库 company 的表 t_dept 中，首先将名为 loc 的字段调整到表的第一个位置，然后把字段 deptno 调整到字段 dname 字段之后。具体步骤如下：

(1) 在“对象资源管理器”窗口中，右击选中表对象 t_dept，然后在弹出的快捷菜单中选择“改变表”命令，如图 5.86 所示，弹出关于 t_dept 的窗口，如图 5.87 所示。



图 5.86 选择“改变表”操作



图 5.87 表 t_dept 窗口

(2) 在 t_dept 的窗口中，如果需要将名为 loc 的字段调整到表的第一个位置，首先选择字段 loc，然后单击“上移”按钮（）两次，如图 5.88 所示，即可将该字段移动到表的第一个位置，如图 5.89 所示。

MySQL 数据库应用从入门到精通



图 5.88 单击“上移”按钮



图 5.89 调整顺序后的效果

(3) 在 t_dept 的窗口中，如果需要将字段 deptno 调整到字段 dname 字段之后，首先选择字段 deptno，然后单击“下移”按钮(▼)一次，如图 5.90 所示，即可将该字段移动到字段 dname 字段之后，如图 5.91 所示。

(4) 如果想使修改后的值生效，需要单击“保存”按钮，弹出确认对话框，在该对话框中单击“确定”按钮，即可使调整后的顺序生效。



图 5.90 单击“下移”按钮

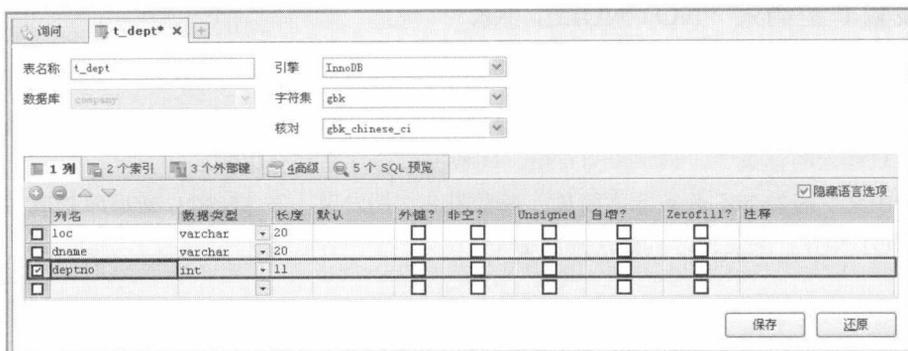


图 5.91 调整顺序后的效果

5.7 操作表的约束

对于已经创建好的表，虽然字段的数据类型决定了所能存储的数据类型，但是表中所存储的数据是否合法并没有进行检查。在具体使用 MySQL 软件时，如果想针对表中的数据做一些完整性检查操作，可以通过表的约束来完成。本节将详细介绍关于表的约束。

5.7.1 MySQL 支持的完整性约束

所谓完整性是指数据的准确性和一致性，而完整性检查就是指检查数据的准确性和一致性。MySQL 数据库管理系统提供了一致机制来检查数据库表中的数据是否满足规定的条件，以保证数据库表中数据的准确性和一致性，这种机制就是约束。

查看帮助文档，可以发现 MySQL 数据库管理系统除了支持标准 SQL 的完整性约束外，还进行了相应扩展。扩展后增加 AUTO_INCREMENT 约束。

表 5.1 所示为 MySQL 软件所支持的完整性约束。

表 5.1 完整性约束

| 完整性约束关键字 | 含 义 |
|------------------|-------------------------|
| NOT NULL | 约束字段的值不能为空 |
| DEFAULT | 设置字段的默认值 |
| UNIQUE KEY (UK) | 约束字段的值是唯一 |
| PRIMARY KEY (PK) | 约束字段为表的主键，可以作为该表记录的唯一标识 |
| AUTO_INCREMENT | 约束字段的值为自动增加 |
| FOREIGN KEY (FK) | 约束字段为表的外键 |

表 5.1 显示的完整性约束中，MySQL 数据库管理系统不支持 check 约束，即可以使用 check 约束但是却没有任何效果。根据约束数据列限制，约束可分为：单列约束，即每个约束只约束一列数据；多列约束，即每个约束可以约束多列数据。

5.7.2 设置非空约束 (NOT NULL, NK)

当数据库表中的某个字段上的内容不希望设置为 NULL 时，则可以使用 NK 约束进行设置。即 NK 约束在创建数据库表时为某些字段加上“NOT NULL”约束条件，保证所有记录中该字段都有值。如果用户插入的记录中，该字段为空值，则数据库管理系统会报错。

设置表中某字段的 NK 约束非常简单，查看帮助文档发现，在 MySQL 数据库管理系统中通过 SQL 语句 NOT NULL 来实现，其语法形式如下：

```
CREATE TABLE table_name (
    属性名 数据类型 NOT NULL,
    .....
);
```

在上述语句中，属性名参数表示所要设置非空约束的字段名字。

【实例 5-17】执行 SQL 语句 NOT NULL，在数据库 company 中创建表 t_dept 时，设置 deptno 字段为 NK 约束。具体步骤如下：

(1) 执行 SQL 语句 CREATE DATABASE，创建数据库 company，具体 SQL 语句如下：

```
CREATE DATABASE company;
USE company;
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.92 所示。

(2) 执行 SQL 语句 CREATE TABLE，创建表 t_dept，具体 SQL 语句如下：

```
CREATE TABLE t_dept(
    deptno INT(20)NOT NULL,
    dname VARCHAR(20),
    loc VARCHAR(40)
);
```

【代码说明】在上述语句中创建表 t_dept 时，通过 SQL 语句 NOT NULL 设置字段 deptno 为 NK 约束。

【运行效果】执行上面的 SQL 语句，其结果如图 5.93 所示。

```
mysql> #创建数据库#
mysql> CREATE DATABASE company;
Query OK, 1 row affected (0.11 sec)

mysql> #选择数据库#
mysql> USE company;
Database changed
mysql> _
```

图 5.92 创建和选择数据库

```
mysql> #创建表#
mysql> CREATE TABLE t_dept(
-> deptno INT(20) NOT NULL,
-> dname VARCHAR(20),
-> loc VARCHAR(40)
-> );
Query OK, 0 rows affected (0.14 sec)

mysql>
```

图 5.93 创建表格 t_dept

(3) 为了检验数据库 company 中的 t_dept 表中，字段 deptno 是否被设置为 NK 约束，执行 SQL 语句 DESC，具体 SQL 语句内容如下：

```
DESC t_dept;
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.94 所示。

执行结果显示，表 t_dept 中字段 deptno 已经被设置为 NOT NULL 约束，如果用户插入的记录

中，该字段为空值，则数据库管理系统会报如下所示的错误。

```
ERROR 1048 (23000): Column 'deptno' cannot be null
```

```
mysql> #查询表#
mysql> DESC t_dept;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| deptno | int(20) | NO   |     | NULL    |       |
| dname  | varchar(20) | YES |     | NULL    |       |
| loc    | varchar(40) | YES |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.09 sec)
```

图 5.94 查看表信息

5.7.3 设置字段的默认值 (DEFAULT)

当为数据库表中插入一条新记录时，如果没有为某个字段赋值，那么数据库系统会自动为这个字段插入默认值。为了达到这种效果，可以通过 SQL 语句关键字 DEFAULT 来设置。

设置数据库表中某字段的默认值非常简单，查看帮助文档发现，在 MySQL 数据库管理系统中通过 SQL 语句 DEFAULT 来实现，其语法形式如下：

```
CREATE TABLE table_name (
    属性名 数据类型 DEFAULT 默认值,
    ....
);
```

在上述语句中，属性名参数表示所要设置默认值的字段名字，“默认值”为该字段的默认值。

【实例 5-18】执行 SQL 语句 DEFAULT，在数据库 company 中创建表 t_dept 时，设置 dname 字段的默认值为 cjangong。具体步骤如下：

(1) 执行 SQL 语句 CREATE DATABASE，创建数据库 company，具体 SQL 语句如下：

```
CREATE DATABASE company;
USE company;
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.95 所示。

(2) 执行 SQL 语句 CREATE TABLE，创建表 t_dept，具体 SQL 语句如下：

```
CREATE TABLE t_dept(
    deptno INT NOT NULL,
    dname VARCHAR(20) DEFAULT 'cjangong',
    loc VARCHAR(40)
);
```

【代码说明】在上述语句中创建表 t_dept 时，通过 SQL 语句 DEFAULT 设置字段 dname 的默认值为 cjangong。

【运行效果】执行上面的 SQL 语句，其结果如图 5.96 所示。

```
mysql> #创建数据库#
mysql> CREATE DATABASE company;
Query OK, 1 row affected (0.11 sec)

mysql> #选择数据库#
mysql> USE company;
Database changed
mysql> _
```

图 5.95 创建和选择数据库

```
mysql> #创建表#
mysql> CREATE TABLE t_dept(
-> deptno INT NOT NULL,
-> dname VARCHAR(20) DEFAULT 'cjangong',
-> loc VARCHAR(40)
-> );
Query OK, 0 rows affected (0.14 sec)
mysql>
```

图 5.96 创建表格 t_dept

MySQL 数据库应用从入门到精通

(3) 为了检验数据库 company 的 t_dept 表中，字段 dname 是否被设置了默认值，执行 SQL 语句 DESC，具体 SQL 语句内容如下：

```
DESC t_dept;
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.97 所示。

```
mysql> #查询表#
mysql> DESC t_dept:
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| deptno | int(11)       | NO   |     | NULL    |       |
| dname  | varchar(20)   | YES  |     | cjqong  |       |
| loc    | varchar(40)   | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)

mysql>
```

图 5.97 查看表信息

执行结果显示，表 t_dept 中字段 dname 已被设置了默认值，如果用户插入的新记录中，该字段为空值，则数据库管理系统会自动插入值 cjqong。

5.7.4 设置唯一约束（UNIQUE，UK）

当数据库表中的某个字段上的内容不允许重复时，则可以使用 UK 约束进行设置。即 UK 约束在创建数据库表时为某些字段加上“UNIQUE”约束条件，保证所有记录中该字段上的值不重复。如果用户插入的记录中，该字段上的值与其他记录里该字段上的值重复，则数据库管理系统会报错。

设置表中某字段的 UK 约束非常简单，查看帮助文档发现，在 MySQL 数据库管理系统中通过 SQL 语句 UNIQUE 来实现，其语法形式如下：

```
CREATE TABLE table_name (
    属性名 数据类型 UNIQUE L,
    .....
);
```

上述语句中，属性名参数表示所要设置唯一约束的字段名字。

【实例 5-19】执行 SQL 语句 UNIQUE，在数据库 company 中创建表 t_dept 时，设置 dname 字段为 UK 约束。具体步骤如下：

(1) 执行 SQL 语句 CREATE DATABASE，创建数据库 company，具体 SQL 语句如下：

```
CREATE DATABASE company;
USE company;
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.98 所示。

(2) 执行 SQL 语句 CREATE TABLE，创建表 t_dept，具体 SQL 语句如下：

```
CREATE TABLE t_dept(
    deptno INT,
    dname VARCHAR(20) UNIQUE,
    loc VARCHAR(40)
);
```

【代码说明】在上述语句中创建表 t_dept 时，通过 SQL 语句 NOT NULL 设置字段 dname 为 UK 约束。

【运行效果】执行上面的 SQL 语句，其结果如图 5.99 所示。

```
mysql> #创建数据库#
mysql> CREATE DATABASE company;
Query OK, 1 row affected (0.11 sec)

mysql> #选择数据库#
mysql> USE company;
Database changed
mysql>
```

图 5.98 创建和选择数据库

```
mysql> #创建表#
mysql> CREATE TABLE t_dept(
-> deptno INT,
-> dname VARCHAR(20) UNIQUE,
-> loc VARCHAR(40)
-> );
Query OK, 0 rows affected (0.14 sec)
```

图 5.99 创建表格 t_dept

(3) 为了检验数据库 company 中的 t_dept 表中，字段 dname 是否被设置为 NK 约束，执行 SQL 语句 DESC，具体 SQL 语句内容如下：

```
DESC t_dept;
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.100 所示。

执行结果显示，表 t_dept 中字段 dname 已经被设置为 NOT NULL 约束，如果用户插入的记录中，该字段上有重复值，则数据库管理系统会报如下所示的错误。

```
ERROR 1062 (23000): Duplicate entry 'c' for key 'dname'
```

(4) 如果想给字段 dname 上的 UK 约束设置一个名字，可以执行 SQL 语句 CONSTRAINT，创建表 t_dept，具体 SQL 语句如下：

```
CREATE TABLE t_dept(
deptno INT,
dname VARCHAR(20),
loc VARCHAR(40),
CONSTRAINT uk_dname UNIQUE(dname)
);
```

【代码说明】在上述语句中，通过关键字 CONSTRAINT 设置唯一约束的标识符。

【运行效果】执行上面的 SQL 语句，其结果如图 5.101 所示。

```
mysql> #查询表#
mysql> DESC t_dept;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| deptno | int(11) | YES | NULL | NULL | |
| dname | varchar(20) | YES | UNI | NULL | |
| loc | varchar(40) | YES | NULL | NULL | |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)

mysql>
```

图 5.100 查看表信息

```
mysql> #创建表#
mysql> CREATE TABLE t_dept(
-> deptno INT,
-> dname VARCHAR(20),
-> loc VARCHAR(40),
-> CONSTRAINT uk_dname UNIQUE(dname)
-> );
Query OK, 0 rows affected (0.13 sec)

mysql>
```

图 5.101 设置约束标识符

注意：在为约束设置标识符时，推荐使用“约束缩写_字段名”，因此设置为 uk_dname。

5.7.5 设置主键约束 (PRIMARY KEY, PK)

当想用数据库表中的某个字段来唯一标识所有记录时，则可以使用 PK 约束进行设置。即 PK 约束在创建数据库表时为某些字段加上“PRIMARY KEY”约束条件，则该字段可以唯一地标识所有记录。

在数据库表中之所以设置主键，是为了便于数据库管理系统快速地查找到表中的记录。在具体设置主键约束时，必须要满足主键字段的值是唯一、非空的。由于主键可以是单一字段，也可以是

MySQL 数据库应用从入门到精通

多个字段，因此分为单字段主键和多字段主键。

1. 单字段主键

设置表中某字段的 PK 约束非常简单，查看帮助文档发现，在 MySQL 数据库管理系统中通过 SQL 语句 PRIMARY KEY 来实现，其语法形式如下：

```
CREATE TABLE table_name (
    属性名 数据类型 PRIMARY KEY,
    .....
);
```

上述语句中，属性名参数表示所要设置 PK 约束的字段名字。

【实例 5-20】执行 SQL 语句 UNIQUE，在数据库 company 中创建表 t_dept 时，设置 deptno 字段为 PK 约束。具体步骤如下：

(1) 执行 SQL 语句 CREATE DATABASE，创建数据库 company，具体 SQL 语句如下：

```
CREATE DATABASE company;
USE company;
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.102 所示。

(2) 执行 SQL 语句 CREATE TABLE，创建表 t_dept，具体 SQL 语句如下：

```
CREATE TABLE t_dept(
    deptno INT PRIMARY KEY,
    dname VARCHAR(20),
    loc VARCHAR(40)
);
```

【代码说明】上述语句中创建了表 t_dept 时，通过 SQL 语句 PRIMARY KEY 设置字段 deptno 为 PK 约束。

【运行效果】执行上面的 SQL 语句，其结果如图 5.103 所示。

```
mysql> #创建数据库#
mysql> CREATE DATABASE company;
Query OK, 1 row affected (0.11 sec)

mysql> #选择数据库#
mysql> USE company;
Database changed
mysql> =
```

图 5.102 创建和选择数据库

```
mysql> #创建表#
mysql> CREATE TABLE t_dept(
-> deptno INT PRIMARY KEY,
-> dname VARCHAR(20),
-> loc VARCHAR(40)
-> );
Query OK, 0 rows affected (0.14 sec)

mysql> =
```

图 5.103 创建表格 t_dept

(3) 为了检验数据库 company 中的 t_dept 表中，字段 dname 是否被设置为 NK 约束，执行 SQL 语句 DESC，具体 SQL 语句内容如下：

```
DESC t_dept;
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.104 所示。

执行结果显示，表 t_dept 中字段 deptno 已经被设置为 PK 约束，同时可以发现主键约束相当于非空约束加上唯一约束。如果用户插入的记录中，该字段上有重复值，则数据库管理系统会报如下所示的错误。

```
ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY'
```

如果用户插入的记录中，该字段上的值为 NULL，则数据库管理系统会报如下所示的错误。

```
ERROR 1048 (23000): Column 'deptno' cannot be null
```

(4) 如果想给字段 deptno 上的 PK 约束设置一个名字，可以执行 SQL 语句 CONSTRAINT，创建表 t_dept，具体 SQL 语句如下：

```
CREATE TABLE t_dept(
    deptno INT,
    dname VARCHAR(20),
    loc VARCHAR(40),
    CONSTRAINT pk_dname PRIMARY KEY(dname)
);
```

【代码说明】在上述语句中，通过关键字 CONSTRAINT 设置主键约束的标识符。

【运行效果】执行上面的 SQL 语句，其结果如图 5.105 所示。

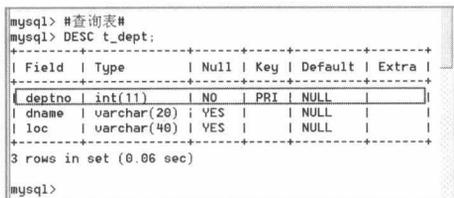


图 5.104 查看表信息

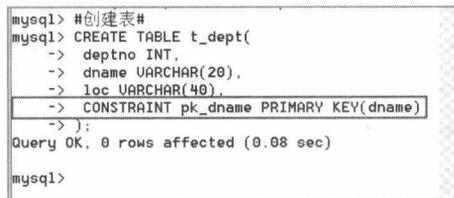


图 5.105 设置约束标识符

注意：在为约束设置标识符时，推荐使用“约束缩写_字段名”，因此设置主键名为 pk_dname 等。

2. 多字段主键

当主键由多个字段组合而成时，则需要通过 SQL 语句 CONSTRAINT 来实现，其语法形式如下：

```
CREATE TABLE table_name (
    属性名 数据类型,
    .....
    【CONSTRAINT 约束名】 PRIMARY KEY(属性名, 属性名.....)
);
```

在上述语句中，在字段定义完之后统一设置主键，PRIMARY KEY 关键字括号中的字段可以有多个，需要通过逗号分割，用来实现设置多字段主键。

【实例 5-21】执行 SQL 语句 CONSTRAINT，在数据库 company 中创建表 t_dept 时，设置 deptno 和 dname 字段为 PK 约束。具体步骤如下：

(1) 执行 SQL 语句 CREATE DATABASE，创建数据库 company，具体 SQL 语句如下：

```
CREATE DATABASE company;
USE company;
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.106 所示。

(2) 执行 SQL 语句 CREATE TABLE，创建表 t_dept，具体 SQL 语句如下：

```
CREATE TABLE t_dept(
    deptno INT,
```

MySQL 数据库应用从入门到精通

```

    dname VARCHAR(20),
    loc VARCHAR(40),
    CONSTRAINT pk_dname_deptno PRIMARY KEY(deptno, dname)
);

```

【代码说明】在上述语句中创建表 t_dept 时，通过 SQL 语句 PRIMARY KEY 设置字段 deptno 和 dname 为 PK 约束。

【运行效果】执行上面的 SQL 语句，其结果如图 5.107 所示。

```

mysql> #创建数据库#
mysql> CREATE DATABASE company;
Query OK, 1 row affected (0.11 sec)

mysql> #选择数据库#
mysql> USE company;
Database changed
mysql> _

```

图 5.106 创建和选择数据库

```

mysql> #创建表#
mysql> CREATE TABLE t_dept(
  -> deptno INT,
  -> dname VARCHAR(20),
  -> loc VARCHAR(40),
  -> CONSTRAINT pk_dname_deptno PRIMARY KEY(dname,deptno)
  -> );
Query OK, 0 rows affected (0.16 sec)
mysql> _

```

图 5.107 创建表格 t_dept

(3) 为了检验数据库 company 中的 t_dept 表中，字段 dname 是否被设置为 PK 约束，执行 SQL 语句 DESC，具体 SQL 语句内容如下：

```
DESC t_dept;
```

【运行效果】执行上面的 SQL 语句，其结果如图 5.108 所示。

```

mysql> #查询表#
mysql> DESC t_dept;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| deptno | int(11)       | NO   | PRI | 0       |       |
| dname  | varchar(20)  | NO   | PRI |         |       |
| loc    | varchar(40)  | YES  |     |         |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>

```

图 5.108 查看表信息

执行结果显示，表 t_dept 中字段 deptno 和 dname 已经被设置为联合主键。

5.7.6 设置字段值自动增加 (AUTO_INCREMENT)

AUTO_INCREMENT 是 MySQL 唯一扩展的完整性约束，当为数据库表中插入新记录时，字段上的值会自动生成唯一的 ID。在具体设置 AUTO_INCREMENT 约束时，一个数据库表中只能有一个字段使用该约束，该字段的数据类型必须是整数类型。由于设置 AUTO_INCREMENT 约束后的字段会生成唯一的 ID，所以该字段也经常会被设置成 PK 主键。

设置表中某字段值的自动增加约束非常简单，查看帮助文档发现，在 MySQL 数据库管理系统中通过 SQL 语句 AUTO_INCREMENT 来实现，其语法形式如下：

```

CREATE TABLE table_name (
    属性名 数据类型 AUTO_INCREMENT,
    ....
);

```