

高等学校教材



# 数据库系统原理 及其应用

庄成三 洪 玫 杨秋辉 编著  
史济民 主审

TP311.13



电子工业出版社  
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

URL: <http://www.phei.com.cn>

高等学校教材

# 数据库系统原理及其应用

庄成三 洪 玫 杨秋辉 编著  
史济民 主审

电子工业出版社

Publishing House of Electronics Industry

北京·BEIJING

## 内 容 简 介

本书共 13 章, 主要内容包括数据库的基本概念、数据模型、关系数据语言、物理数据组织、数据控制和事务管理、分布式数据库、应用系统的体系结构、开发工具、开发方法和数据库管理等, 还包括 Web 平台上数据库应用系统开发、多媒体数据库、数据仓库、数据挖掘等最新技术。本书理论体系完整, 强调应用, 取材新颖实用, 既可作为大专院校的教材, 也可作为工程技术人员的参考书。

未经许可, 不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有, 翻版必究。

## 图书在版编目(CIP)数据

数据库系统原理及其应用/庄成三等编著/史济民主审.-北京: 电子工业出版社, 2000. 6

ISBN 7-5053-5178-8

I. 数… II. 庄… III. 数据库系统-基本知识 IV. TP311.13

中国版本图书馆 CIP 数据核字(2000)第 03646 号

书 名: 数据库系统原理及其应用

编 著 者: 庄成三 洪 玫 杨秋辉

主 审 者: 史济民

责任编辑: 应月燕

特约编辑: 于 平

排版制作: 电子工业出版社计算机排版室

印 刷 者:

装 订 者: 北京李史山胶印厂

出版发行: 电子工业出版社 URL: <http://www.phei.com.cn>

北京市海淀区万寿路 173 信箱 邮编 100036

经 销: 各地新华书店

开 本: 787×1092 1/16 印张: 20.25 字数: 518 千字

版 次: 2000 年 6 月第 1 版 2001 年 1 月第 2 次印刷

书 号: ISBN 7-5053-5178-8

G·416

印 数: 5 000 册 定价: 29.00 元

凡购买电子工业出版社的图书, 如有缺页、倒页、脱页、所附磁盘或光盘有问题者, 请向购买书店调换;  
若书店售缺, 请与本社发行部联系调换。电话 68279077

## 前 言

数据库是计算机科学的重要分支,是计算机应用十分重要的基础领域。因此,数据库系统课程也就成为高校计算机教学中比较重要的主干课程,它不仅是高校的计算机专业学生的必修课程,也是许多非计算机类专业学生的必修或选修课程,该课程还作为许多在职人员计算机业务进修或培训的科目。

数据库是计算机学科的一个面向应用的分支学科,对于绝大多数学生来说,学习数据库课程的主要目的是为了应用现有的数据库管理系统软件(DBMS)和数据库应用系统的开发技术及工具,解决在实际工作中的各类计算机应用问题,比如实现本企业或组织的信息管理系统,开发网络上的以数据库为基础的应用软件,或管理大中型的数据库系统等。这就要求数据库系统课程的教学内容要面向应用,要根据数据库技术发展情况及时更新,使学生除了掌握必要的系统的数据库知识外,重点应放在培养学生研制、开发和管理数据库应用系统的能力。即使针对 DBMS 研制或数据库新技术新方法的研究人员,了解现在实用的数据库管理系统的知识和数据库应用的知识也是十分重要的,因为数据库是应用推动发展的学科,新技术、新思想大多是从应用中产生的。

然而,国内现有的数据库教材中,内容的安排大都存在一些问题,一类教材太强调理论,重点放在数据库的系统理论和数据库设计原理上,缺少对数据库应用方面的知识、方法和工具的介绍,内容抽象而比较脱离实际,属于“经典理论”,学生工作后感到学了的知识用不上,要用的又未学;而另一类教材又忽略理论知识,缺乏系统性,直接针对具体一种数据库管理系统(DBMS),介绍其使用方法,太大众化,达不到对计算机专业学生的要求。我们认为,对于高校计算机专业用的教材,应该兼顾数据库理论基础知识和应用开发能力。

随着计算机应用的推广和应用水平的提高,特别是 Internet 和 WWW 技术的飞速发展,数据库应用技术也有很大的发展,比如多媒体数据库技术,Web 环境下的数据库应用技术,数据库应用的开发工具等都得到迅速的发展。现有的数据库系统教材大都在内容上跟不上数据库技术的这一发展,缺乏这些方面的新内容,即使某些教材介绍了一些新的应用知识,但都显得零散而不系统,不能满足实际教学需要。

鉴于以上原因,我们感到数据库系统课程的教材需要强化应用环节、更新内容,以反映数据库应用的当前情况,这就是我们编写本书的目的。本书集数据库基础理论、数据库管理系统(DBMS)和数据库应用技术为一体,以关系数据库系统为核心,全面、系统地阐述了数据库系统及其与应用有关的基本概念、基本理论和方法技术。本书吸取了国内外同类教材的优点,特别加强了数据库应用方面的内容,增加了许多反映当前数据库系统发展的最新水平和新趋势的内容,力图使学生掌握数据库系统的基本理论和技术,了解数据库系统发展的新技术新趋势,能胜任应用数据库技术以及市面上可用的工具软件设计开发各类数据库应用系统或维护管理大中型数据库系统的工作,或至少为此打下良好的基础。换句话说,学生毕业后面对开发各类数据库应用系统或维护管理大中型数据库系统的工作时本书学到的知识大多能用上。为此,在这本新教材中,我们对数据库的‘经典’的内容进行了取舍,传统的三种数据库类型中,层次数据库和网状数据库国内很少有应用,只保留了它们的数据模型的内容;关系数据库系统总的

说来加强了,也根据应用的需要,各部分有详有略,陈旧的或实际中很少用的内容都不要,或仅略做介绍;增加了数据库应用方面的内容和数据库的新技术与新领域的知识,与已有各类数据库教材相比,内容有很大的改变。

本书可作为计算机或相关专业本科的数据库系统课程的教材,参考学时数为 60~68 学时,面向对象数据库及 SQL3 的内容大多可略去,第 9 章后的内容可选择部分讲解,余下自学。本书也可作为计算机专业的研究生课程的教材。本书经适当取舍后,也可供各类数据库技术培训班和专科学校使用。另外,本书便于自学,可作为从事计算机和数据库工作的科研与应用技术人员的业务参考书。

本书编者在高校从事了 15 年的计算机专业研究生、本科生数据库系统课程教学,多年从事数据库及数据库管理系统的研究、数据库应用系统的开发,指导数据库方向的研究生论文以及本科生毕业设计,从事在职人员计算机技术培训,积累了大量的经验,也收集了许多资料,包括使用多年的内部交流讲义、国内外优秀教材、国内外科技文献和论文、Internet 上的最新技术资料、科研项目设计开发文档和技术报告等,这些工作为我们编写本教材打下了基础。由于我们水平有限,而数据库技术的发展又非常迅速,因此书中难免有些缺点和错误,殷切希望广大读者批评指正。

本书由庄成三拟出提纲,规划各章节的内容;然后由洪玫编写大部分章节的初稿,杨秋辉编写了第 12 章、第 13 章的 13.2 和 13.3;庄成三再对全书进行了修改和统稿,补充了许多内容,不少章节又做了很大的修改。史济民教授作为主审审核了全书。在本书的编写过程中,得到许多同行、同事的大力支持和鼓励,本校 96 级部分研究生提供了许多翻译资料并进行了书稿的录入工作,在此表示衷心的感谢!

庄成三 洪 玫 杨秋辉  
1999 年 12 月于四川大学

# 目 录

|                                |        |
|--------------------------------|--------|
| 第 1 章 导论 .....                 | ( 1 )  |
| 1.1 数据库管理系统的特点 .....           | ( 1 )  |
| 1.2 数据库系统的三级模式和数据独立性 .....     | ( 4 )  |
| 1.2.1 数据库系统的三级模式 .....         | ( 5 )  |
| 1.2.2 数据独立性 .....              | ( 7 )  |
| 1.3 数据语言 .....                 | ( 7 )  |
| 1.3.1 数据定义语言(DDL) .....        | ( 8 )  |
| 1.3.2 数据操纵语言(DML) .....        | ( 8 )  |
| 1.3.3 数据控制语言(DCL) .....        | ( 8 )  |
| 1.3.4 宿主语言 .....               | ( 9 )  |
| 1.4 数据库管理系统的结构 .....           | ( 9 )  |
| 1.4.1 数据库管理系统的组成 .....         | ( 9 )  |
| 1.4.2 数据库管理系统的进程结构 .....       | ( 10 ) |
| 1.5 数据库系统的发展历史 .....           | ( 11 ) |
| 1.6 数据库技术的新进展 .....            | ( 13 ) |
| 1.6.1 面向对象数据库系统 .....          | ( 13 ) |
| 1.6.2 模糊数据库与演绎数据库 .....        | ( 14 ) |
| 1.6.3 分布式数据库与联邦数据库 .....       | ( 15 ) |
| 1.6.4 多媒体数据库 .....             | ( 16 ) |
| 1.6.5 Web 数据库 .....            | ( 16 ) |
| 1.6.6 并行数据库 .....              | ( 17 ) |
| 1.6.7 数据仓库、数据挖掘与联机分析 .....     | ( 17 ) |
| 1.6.8 数据库应用系统体系结构的发展 .....     | ( 18 ) |
| 习题 .....                       | ( 18 ) |
| 第 2 章 数据模型 .....               | ( 20 ) |
| 2.1 数据模型的概念 .....              | ( 20 ) |
| 2.2 实体-联系(E-R)概念模型 .....       | ( 21 ) |
| 2.3 关系数据模型 .....               | ( 25 ) |
| 2.3.1 关系数据模型的基本概念 .....        | ( 25 ) |
| 2.3.2 将 E-R 模型转换成关系数据模式 .....  | ( 28 ) |
| 2.3.3 在关系数据模型上的操作 .....        | ( 29 ) |
| 2.3.4 关系数据模型的完整性约束 .....       | ( 35 ) |
| 2.4 网状数据模型 .....               | ( 36 ) |
| 2.4.1 网状数据模型的数据结构 .....        | ( 36 ) |
| 2.4.2 将 E-R 模型转化为网状数据模式 .....  | ( 38 ) |
| 2.4.3 网状数据模型的数据约束 .....        | ( 40 ) |
| 2.4.4 在网状数据模型上的操作 .....        | ( 41 ) |
| 2.4.5 网状数据模型与关系数据模型的比较 .....   | ( 41 ) |
| 2.5 层次数据模型 .....               | ( 42 ) |
| 2.5.1 层次数据模型的数据结构 .....        | ( 42 ) |
| 2.5.2 由 E-R 模型向层次数据模式的转换 ..... | ( 44 ) |
| 2.5.3 层次数据模型的数据约束 .....        | ( 45 ) |

|                                    |       |
|------------------------------------|-------|
| 2.5.4 在层次数据模型上的操作 .....            | (45)  |
| 2.5.5 层次数据模型与网状数据模型的比较 .....       | (46)  |
| 2.6 面向对象数据模型 .....                 | (46)  |
| 2.6.1 面向对象模型的基本概念 .....            | (47)  |
| 2.6.2 将 E-R 模型转换为面向对象数据模型 .....    | (48)  |
| 2.7 逻辑数据模型 .....                   | (49)  |
| 2.7.1 逻辑规则的语义解释 .....              | (49)  |
| 2.7.2 Datalog 数据模型 .....           | (51)  |
| 习题 .....                           | (53)  |
| <b>第 3 章 数据语言</b> .....            | (55)  |
| 3.1 数据语言的特点 .....                  | (55)  |
| 3.2 QBE 语言 .....                   | (57)  |
| 3.2.1 QBE 的数据查询 .....              | (58)  |
| 3.2.2 QBE 的数据更新 .....              | (61)  |
| 3.2.3 QBE 的数据定义 .....              | (62)  |
| 3.2.4 QBE 的完整性 .....               | (63)  |
| 3.3 SQL 的数据查询 .....                | (64)  |
| 3.3.1 SQL 语言简介 .....               | (64)  |
| 3.3.2 SQL 的查询语句 .....              | (66)  |
| 3.3.3 SQL 查询的方法和例子 .....           | (67)  |
| 3.4 SQL 的数据更新 .....                | (74)  |
| 3.5 SQL 的数据定义和数据库一览表 .....         | (76)  |
| 3.6 嵌入式 SQL .....                  | (79)  |
| 3.7 存储过程和数据库触发器 .....              | (85)  |
| 3.8 SQL3 .....                     | (88)  |
| 习题 .....                           | (96)  |
| <b>第 4 章 关系数据库设计</b> .....         | (99)  |
| 4.1 关系数据库模式可能出现的异常 .....           | (99)  |
| 4.2 关系模式的函数依赖 .....                | (102) |
| 4.2.1 函数依赖的一般概念 .....              | (102) |
| 4.2.2 Armstrong 公理系统 .....         | (104) |
| 4.3 多值依赖和连接依赖 .....                | (105) |
| 4.4 关系的规范化 .....                   | (106) |
| 4.4.1 范式及其类型 .....                 | (107) |
| 4.4.2 关系规范化的基本原则 .....             | (111) |
| 4.5 在实际数据库设计中关系规范化的应用 .....        | (113) |
| 4.6 面向对象模型和关系数据模型的相互转换 .....       | (115) |
| 习题 .....                           | (118) |
| <b>第 5 章 物理数据组织</b> .....          | (119) |
| 5.1 物理数据模型 .....                   | (119) |
| 5.2 堆组织 .....                      | (123) |
| 5.3 哈希文件 .....                     | (124) |
| 5.4 索引文件 .....                     | (126) |
| 5.4.1 稀疏索引 .....                   | (126) |
| 5.4.2 稠密索引 .....                   | (129) |
| 5.4.3 动态索引及 B <sup>+</sup> 树 ..... | (130) |
| 5.5 嵌套记录结构 .....                   | (133) |
| 5.6 次索引 .....                      | (136) |

|                                 |       |
|---------------------------------|-------|
| 5.7 关系数据库系统常用结构 .....           | (137) |
| 5.8 面向对象数据库物理存储结构 .....         | (138) |
| 习题 .....                        | (142) |
| <b>第 6 章 数据库保护</b> .....        | (144) |
| 6.1 数据库的完整性 .....               | (144) |
| 6.2 商品化 DBMS 完整性维护实例 .....      | (146) |
| 6.3 数据库的安全性 .....               | (147) |
| 6.4 商品化 DBMS 数据安全措施实例 .....     | (150) |
| 6.5 数据库应用系统的安全措施 .....          | (152) |
| 习题 .....                        | (153) |
| <b>第 7 章 事务管理</b> .....         | (155) |
| 7.1 并发存取可能出现的异常 .....           | (155) |
| 7.2 事务管理的基本概念 .....             | (156) |
| 7.3 两阶段加锁协议 .....               | (160) |
| 7.4 层次结构的加锁方法 .....             | (165) |
| 7.5 事务故障处理和恢复方法 .....           | (168) |
| 7.6 乐观和保守的协议 .....              | (171) |
| 7.7 时印并发控制方法 .....              | (172) |
| 7.8 商品化 DBMS 并发控制和事务管理的特点 ..... | (175) |
| 7.8.1 Oracle 事务管理的特点 .....      | (175) |
| 7.8.2 Sybase 并发控制的特点 .....      | (180) |
| 习题 .....                        | (182) |
| <b>第 8 章 分布式数据库系统</b> .....     | (183) |
| 8.1 分布式数据库系统的一般概念 .....         | (183) |
| 8.2 分布式数据库设计 .....              | (185) |
| 8.3 分布式事务 .....                 | (188) |
| 8.4 分布式加锁 .....                 | (190) |
| 8.5 分布式事务的交付 .....              | (195) |
| 8.6 基于时印的分布式并发控制 .....          | (199) |
| 8.7 乐观的并发控制方法 .....             | (200) |
| 8.8 C/S 结构中的并发控制 .....          | (202) |
| 8.9 基于版本的并发控制 .....             | (203) |
| 8.10 复制数据库技术 .....              | (205) |
| 8.11 联邦数据库系统简介 .....            | (205) |
| 8.12 多数据库系统简介 .....             | (207) |
| 习题 .....                        | (209) |
| <b>第 9 章 数据库应用系统的开发环境</b> ..... | (211) |
| 9.1 数据库应用系统的结构 .....            | (211) |
| 9.2 商品化 DBMS 的特点和编程接口 .....     | (214) |
| 9.2.1 Oracle .....              | (215) |
| 9.2.2 Sybase .....              | (220) |
| 9.2.3 Informix 的特点 .....        | (227) |
| 9.2.4 Ontos 的数据语言 .....         | (228) |
| 9.2.5 O2 的数据语言 .....            | (231) |
| 9.2.6 ODBC 和 JDBC .....         | (232) |
| 9.3 典型的数据库应用系统开发环境 .....        | (236) |
| 9.3.1 PowerBuilder .....        | (236) |

|                                         |              |
|-----------------------------------------|--------------|
| 9.3.2 Delphi .....                      | (240)        |
| 9.3.3 Visual Basic 简介 .....             | (244)        |
| 9.3.4 Developer/2000 简介 .....           | (245)        |
| 9.3.5 Informix-NewEra 简介 .....          | (246)        |
| 习题 .....                                | (247)        |
| <b>第 10 章 基于 Web 的数据库应用系统开发技术 .....</b> | <b>(249)</b> |
| 10.1 Internet 和 Web 的基本概念 .....         | (249)        |
| 10.2 Web 作为数据库应用系统的开发平台 .....           | (251)        |
| 10.2.1 对 Web-DBMS 集成技术的要求 .....         | (251)        |
| 10.2.2 Web-DBMS 方法的特点 .....             | (251)        |
| 10.2.3 基于 Web 的数据库应用系统体系结构 .....        | (253)        |
| 10.3 公共网关接口(CGI) .....                  | (254)        |
| 10.3.1 CGI 的一般概念 .....                  | (254)        |
| 10.3.2 CGI 程序的工作方式 .....                | (255)        |
| 10.3.3 CGI 的特点 .....                    | (257)        |
| 10.4 应用编程接口(API) .....                  | (257)        |
| 10.4.1 ISAPI .....                      | (257)        |
| 10.4.2 Internet 数据库连接器(IDC) .....       | (259)        |
| 10.5 Java Applet 简介 .....               | (261)        |
| 10.6 Microsoft ActiveX 平台 .....         | (261)        |
| 10.7 扩展标记语言(XML)简介 .....                | (266)        |
| 习题 .....                                | (268)        |
| <b>第 11 章 数据库应用系统开发方法 .....</b>         | <b>(269)</b> |
| 11.1 概述 .....                           | (269)        |
| 11.2 数据库应用系统的开发方法和技术 .....              | (270)        |
| 11.3 系统分析 .....                         | (275)        |
| 11.4 系统设计 .....                         | (280)        |
| 11.5 原型设计和生成 .....                      | (283)        |
| 习题 .....                                | (284)        |
| <b>第 12 章 数据库管理 .....</b>               | <b>(285)</b> |
| 12.1 DBA 的作用和职责 .....                   | (285)        |
| 12.2 商品化 DBMS 的数据库管理工具 .....            | (288)        |
| 习题 .....                                | (293)        |
| <b>第 13 章 数据库应用发展的新技术 .....</b>         | <b>(295)</b> |
| 13.1 多媒体数据库技术 .....                     | (295)        |
| 13.2 数据仓库 .....                         | (299)        |
| 13.3 联机分析处理(OLAP)和数据挖掘 .....            | (307)        |
| 习题 .....                                | (312)        |
| <b>参考文献 .....</b>                       | <b>(314)</b> |

# 第1章 导 论

一个完整的数据库系统由数据库、数据库管理系统、数据库管理员和应用程序等四部分组成,如图 1-1 所示。其中数据库是相关数据的集合,以一定的组织形式存储在永久存储介质上;数据库管理系统(Database Management System,简称 DBMS)是管理数据库的软件,实现在数据库上的各种操作,它是数据库系统的核心;数据库管理员(Database Administrator,简称 DBA)是一个或一组人员,是数据库的高级用户,他完成数据库的规划、设计、维护和管理等工作;应用程序是各种用户使用数据库的界面,它在 DBMS 支持下为用户提供各类服务。

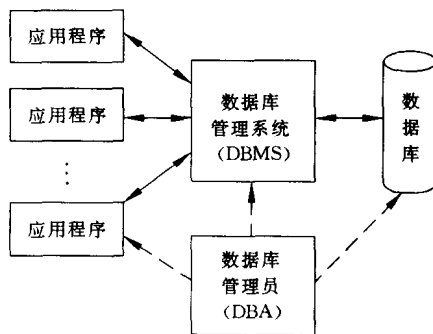


图 1-1 数据库系统的组成

数据库管理系统是一类非常重要的软件系统。目前在世界上有许多计算机都正运行着各种 DBMS,管理着许多数据库系统,这些数据库系统又支持着更大数量的应用系统。像其他重要的软件系统(如编译系统和操作系统)一样,围绕着数据库管理系统和数据库系统已发展起来一套完整的概念、理论、方法和技术,无论是学习和理解怎样有效地运用 DBMS 管理数据,还是设计和实现 DBMS,这些概念、理论、方法和技术都是非常有用的。在本书里,我们将学习这些概念、理论、方法和技术,以及应用这些技术和方法开发和维护数据库应用系统。

## 1.1 数据库管理系统的特点

数据库管理系统有两个最基本的特征,这两个特征使它区别于其他类型的程序设计系统:

- 管理长期应用的数据的能力。
- 高效地存取大量数据的能力。

第一是指有一个数据库永久存在,这个数据库的内容是 DBMS 存取和管理的数据。第二是将数据库管理系统(DBMS)和文件系统区分开,文件系统也可管理要长期应用的数据,但一般并不提供对任意部分数据快速存取的功能。文件系统用的是简单的存取技术(比如线性扫描数据),它仅仅适用于数据量较小的情况,当数据量很大时,必须用 DBMS。

DBMS 上述的两个性质是最基本的。除此之外,大多数商品化的 DBMS 中还有若干其他功能,下面列出其中比较重要的:

- 支持至少一种数据模型,数据模型是数据的数学抽象描述,用户可用这个模型来描述数据的结构和数据上的操作。
- 支持某些高级语言,提供高级语言用户接口,允许用户定义数据结构,存取数据和操纵数据。
- 事务管理,即提供多用户并发存取数据库的能力,特别是要保证在多个用户同时存取数据库时系统给出正确的结果。

- 存取控制,即限制未授权用户非法存取数据和检查并自动维护数据有效的能力。
- 可恢复性,即从系统故障中恢复系统的能力。

通常,DBMS 有数据目录管理功能,即提供对数据定义信息(数据目录)的管理能力。很多 DBMS,如关系型 DBMS,提供了说明性的查询语言,并能对查询进行优化处理。

### (1) 数据模型

每个数据库管理系统至少支持一种数据模型,这种模型允许用户用更易理解的术语准确地描述数据。通常可以在几个不同的抽象级别上描述数据。在抽象级别的最底层,数据可看作是由文件组成的。

**[例 1.1]** 一个公司通常保存有关于它的雇员的文件。每个雇员的记录可能有他的姓名、雇员 ID 号、工资、居住地址等信息。假设在一个记录中仅仅保存雇员的姓名和他的经理。这个记录的结构如下:

```
record
    name:char[30];    /* 姓名
    manager:char[30]; /* 经理
end
```

该文件本身是记录的序列,公司的每个雇员在文件中有一条记录。

在我们将要讨论的关系数据模型中,一个这样的记录文件就是一个关系,这关系可被描述为:

```
EMPLOYEES(NAME,MANAGER)    /* 雇员(姓名,经理) */
```

这里,EMPLOYEES 是关系名,这个关系对应了在例 1.1 中叙述过的文件,NAME 和 MANAGER 是字段名。在关系理论中字段被称为属性。

在这非正式导论性的章节里,我们有时将“文件”(File)和“关系”(Relation)作为同义词用,当我们学习数据库系统的更多的知识后,就会清楚它们是不同的概念。关系是文件的抽象,在这抽象中,字段的数据类型一般不被关心,记录的次序也没指定,在关系中记录被叫做元组。于是,一个文件是一个记录的表,而一个关系是一个元组的集合。

### (2) 高效的文件存取

存储文件的能力并不是 DBMS 独有的能力,任何操作系统的文件系统都有这种能力。当我们存取文件的数据时,DBMS 的优点就显示出来了。例如,我们希望找到雇员“王红”的经理。如果这公司有几千雇员,搜索整个文件找到有 NAME=“王红”的记录代价非常大。DBMS 帮助我们建立了“索引文件”或“索引”,无论这文件多大,索引允许我们基本上用一步就能存取“王红”的记录。类似地,插入新记录或删除记录也能在很短的、基本上和文件尺寸无关的恒定时间内完成。一种常用的索引结构是以 NAME 作关键字的哈希(Hash)表。在第 5 章将讨论哈希和其他索引结构。

数据库管理系统帮助我们做的另一件事是在文件中导航,那就是组合两个或更多文件以得到我们需要的信息。下面的例子说明了导航。

**[例 1.2]** 假设在一个雇员记录中我们存储他工作的部门而不是他的经理。在另一个称为 Department(部门)的文件中的记录联系各部门的名称和他的经理。再用关系来描述这些数据,我们有:

```
EMPLOYEES(NAME, DEPT)    /* 雇员(姓名,部门) */
```

DEPARTMENTS(DEPT, MANAGER) /\* 部门(部门,经理) \*/

现在,如想找到“王红”的经理,我们必须利用两个文件中 DEPT 字段相等的条件,从 EMPLOYEES 导航到 DEPARTMENTS。即是说,我们首先在 EMPLOYEES 文件中找到 NAME = “王红”的记录,从这个记录得到 DEPT 属性的值,比如是“广告部”。然后,我们在 DEPARTMENTS 文件中找到有 DEPT = “广告部”的记录,再找到它的 MANAGER 的值,比如是“赵亮”。如果建立了正确的索引,我们能在很短的时间内完成查询。

### (3) 查询语言

为了使存取数据更方便,DBMS 提供了查询语言或数据操纵语言来表示数据上的操作。不同的查询语言设计数据处理程序的方法不同,特别是与存取数据有关的很多细节不相同。相比较而言,支持关系数据模型的数据库管理系统的查询语言简单规范,程序设计容易,涉及的细节较少。

[例 1.3] 在例 1.2 中讨论的查询可用 SQL 语言写出来:

```
• SELECT MANAGER      /* 告诉数据库管理系统打印出经理作为答案 */
• FROM EMPLOYEES,DEPARTMENTS /* 查看 EMPLOYEES 和 DEPARTMENTS 关系
                               */
• WHERE EMPLOYEES.NAME = “王红” /* 指明雇员名字是“王红” */
    AND EMPLOYEES.DEPT = DEPARTMENTS.DEPT; /* 找到部门文件中部门
    (DEPT)字段(在 DEPARTMENTS 关系中)与名叫王红这个雇员的部门
    (DEPT)字段(在 EMPLOYEES 关系中)值相同的元组,该元组的经理
    (MANAGER)字段的值即是这雇员的经理。 */
```

另外,我们也可以用简化型的网状模型查询语言写出同一查询:

```
• EMPLOYEES.NAME := “王红”
• FIND EMPLOYEES RECORD BY CALC-KEY /* 告诉 DBMS 在 EMPLOYEES 中找王
  红的记录 */
• FIND OWNER OF CURRENT EMP-DEPT SET /* 通过联系雇员到他的部门的系结构
  EMP-DEPT,找到雇员属籍的部门(“系”和“属籍”是 DBTG 数据模型的技术术语) */
• FIND FIRST MANAGER RECORD IN CURRENT DEPT-MGR SET /* 另一个系结构
  DEPT-MGR 联系部门到他的经理,找到部门属籍的经理 */
• PRINT MANAGER.NAME /* 此打印操作并不是查询语言的一部分,而是周围的“宿主语
  言”的一部分,宿主语言是普通的程序设计语言。 */
```

比较以上两种语言可以看出,在网状模型的查询语言中,导航被显式地写出来了,这要求程序设计员做额外的工作,即要描述怎样从一个记录到达下一个记录,而在关系模型的 SQL 语言中,仅仅说明答案应满足的条件即可。SQL 语言的这种“说明性”是支持 SQL 语言的关系数据库管理系统越来越流行的重要原因之一。

### (4) 事务管理

事务是在数据库上操作的过程。数据库管理系统的又一重要特点是具有同时管理一大批事务的能力。数据库系统往往是多机网络系统,时常有许多分散的计算机同时在数据库上操作。银行应用的数据库系统就是典型的例子,这种数据库被成百上千个自动储蓄机和银行分支机构的雇员在同一时间存取。航空公司订票系统是另一例子。有时两个存取彼此互不干涉,例如,任意数量的事务能在同一时刻读同一帐号的金额而不会出错。但是当你正在银行存取你的

工资支票时,你的妻子同时从一个自动储蓄机取钱,两个事务就同时执行了。同时执行且没有协同的两个事务,其运行结果是不能预测的,此时可能发生错误。所以,修改数据项的事务必须同时封锁其他试图读或写这个数据项的事务。数据库管理系统提供了某种形式的并发控制机制,以阻止多个事务不协同地存取同一数据项。

当数据库被分散到许多不同的计算机系统上时,允许重复的数据存储在不同的计算机上,以实现快速的局部存取和防止一个计算机发生故障时数据遭到破坏,此时的情况更复杂。对分布式操作有用的某些存取控制技术将在第 8 章讲述。

#### (5) 数据的安全性

数据库管理系统不仅必须防止系统失效时丢失数据,而且必须防止未授权的存取操作,比如只有具有某种手续的用户有对雇员工资文件的存取权。数据库管理系统必须能将各个用户同他们能存取的文件、文件的字段和其他特权联系起来。于是,数据库管理系统必须为每个用户维护一个表,这个表记录了用户在每个对象上所有的存取权限。比如一个用户可以读某文件,但不能在文件中插入、删除数据;另一用户或许根本不能看这文件;第三个用户可以读、写或随意修改这个文件等等。

为了满足多用户的需求和系统数据安全性的要求,数据库管理系统提供了视图设施,使用户在数据库中仅看他所关心的数据,而不是数据库的全体。视图允许我们创建想象的对象,这种对象是以一种精确的方式从实际对象(文件或关系)中定义出来的。

[例 1.4] 对于 EMPLOYEES 文件,假设我们希望多数用户有除 SALARY 字段外其余字段的存取权,在 SQL 语言中可定义一视图 SAFE-EMPS:

```
CREATE VIEW SAFE-EMPS AS
SELECT NAME,DEPT,ADDRESS FROM EMPLOYEES;
```

视图 SAFE-EMPS 由 EMPLOYEES 的 NAME、DEPT 和 ADDRESS 等字段组成,即可看作是关系:SAFE-EMPS(NAME,DEPT,ADDRESS)。视图 SAFE-EMPS 是一逻辑文件,在物理上并不存在,但可对它进行查询,比如可通过 SQL 语句查询王红的部门:

```
SELECT DEPT FROM SAFE-EMPS WHERE NAME="王红";
```

多数用户只允许存取视图 SAFE-EMPS,而不允许存取关系 EMPLOYEES。具有存取工资特权的用户被给予了读 EMPLOYEES 关系的特权,其中一部分有修改 EMPLOYEES 关系的特权,可以修改工资。

## 1.2 数据库系统的三级模式和数据独立性

当设计数据库时,我们对数据库的结构感兴趣;当应用数据库时,我们关心的是数据库中存在的**数据**。**注意**,数据库中的数据经常变化,而数据库的结构在一定时间范围内不会改变。

我们称数据库的当前内容是数据库的实例,也叫数据库的外延(Extension)或数据库的状态。数据库的结构指数据库管理的数据实体的类型、特征、实体间的联系的表示,也称为数据库模式(Schema)或数据库的内涵。比如我们定义一个学生数据库的模式如表 1.1 所示。而表 1.2 是这个数据库模式的一个实例。

数据库中结构的定义可以在多个抽象级别进行,形成多个级别的数据库模式,这些模式的实例就是相应的抽象数据库,下面具体介绍三级模式。

表 1.1

| 数据名称 | 数据类型   | 数据宽度 | 小数点位数 |
|------|--------|------|-------|
| 学号   | CHAR   | 12   |       |
| 姓名   | CHAR   | 8    |       |
| 年龄   | NUMBER | 2    |       |
| 性别   | CHAR   | 4    |       |
| 入学成绩 | NUMBER | 5    | 1     |

表 1.2

|        |    |    |   |       |
|--------|----|----|---|-------|
| 950100 | 张山 | 20 | 男 | 550.0 |
| 950120 | 李静 | 18 | 女 | 600.0 |
| 950125 | 赵易 | 19 | 男 | 580.0 |
| 950128 | 王安 | 20 | 男 | 610.0 |

### 1.2.1 数据库系统的三级模式

在计算机与最终用户之间,数据有许多级抽象,图 1-2 说明了数据库系统的三级抽象模式:外模式、逻辑模式、内模式(或物理模式、存储模式),这是数据库系统的体系结构或总框架。在数据库系统领域,一般不必深入到二进位或字节的级别看待数据,而是从文件级(本书称为物理级)开始,因为数据库系统往往是建立在文件系统基础之上的。三级抽象模式在数据库系统中都存储于数据目录(系统数据字典)中,是数据目录的最基本的内容,DBMS 通过数据目录来管理和访问数据模式。

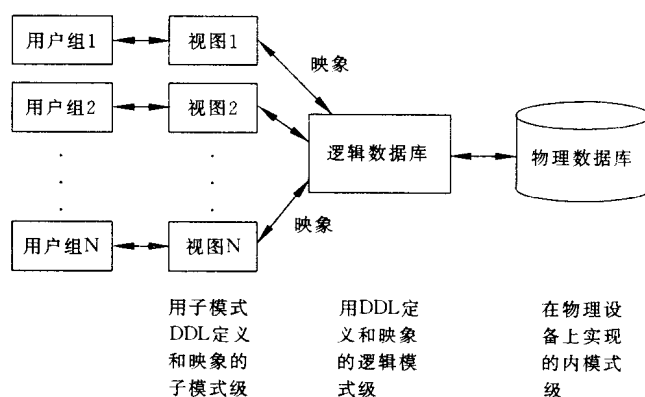


图 1-2 数据库系统的三级模式

#### 1. 物理数据库级

文件、文件索引以及其他存储结构的集合叫做物理数据库。物理数据库永久保存在次级存储设备上,比如磁盘。多个不同的物理数据库可以被同一数据库管理系统管理。物理数据库的结构称为物理模式或内模式(Internal Schema)、存储模式,是数据在数据库系统的内部表示,即对数据的物理结构和存储方式的描述,是低级描述,一般由 DBMS 提供的工具或语言完成。用户可以在多大程度上直接定义内模式,在具体的 DBMS 中各不相同,当前的许多 DBMS 产品自身可以完成大部分内模式的定义工作,几乎不需用户介入。第 5 章将叙述物理数据库系统中运用的主要数据结构。

#### 2. 逻辑数据库级

逻辑数据库是对现实世界中数据库用户的数据的抽象。逻辑数据库的结构称为逻辑模式(Logical Schema)或模式,是数据库中全体数据的逻辑结构和特性的描述,是所有用户的公共

数据视图。数据库管理系统提供模式数据定义语言 DDL 来描述逻辑模式,即严格地定义数据的名称、特征、相互关系、约束等。逻辑模式的物理实现可以通过运行 DDL 程序完成。逻辑模式的基础是数据模型,DDL 允许我们用“数据模型”的术语描述逻辑模式,例如,在关系数据模型中,关系被组织成表格形式,它的列是关系的属性,它的行是关系的元组,关系元组类似于文件的记录。在第 2 章我们将介绍关于数据模型的概念。

数据库管理系统允许企业将它的所有信息文件汇集到一起,用统一的方式,即逻辑模式描述的方式看待。这对一个企业或组织并不是一个简单的任务,因为企业或组织内同种类型的信息一般被保存在不同地点,同类信息运用的格式也经常不同。

[例 1.5] 一个公司的各子公司都保存有关于雇员和他们所在的部门的信息。但是一个子公司可能将雇员名字作为一个整体存储,而另一子公司可能将雇员的姓和名分别存储。从一种格式转换成另一种格式或许并不难,但是它必须在建立一个统一的逻辑数据库前被完成。

数据结构的差别可能就更难被一致化了。在一个子公司中可能每个雇员有一个记录,记录中包含这雇员所在的部门;另一个子公司可能在文件中列出所有部门,每个部门的记录后面有一个该部门雇员记录的表,表中一个雇员有一个记录。这两者不同之处是:前者的数据库中,如果某部门没有雇员,则部门信息也就没有了,但后者不会出现这种情况。如果这两个子公司的部门完全相同,每个子公司有可能有无雇员的部门,根据这两个子公司的数据库结构,“列出所有部门”的查询将有不同的答案,为了建立逻辑模式,必须事先达成统一结构的协议,这个过程叫做数据库集成。

### 3. 视图级

视图是数据库个别用户看到的数据视图,是逻辑数据库的派生。视图的结构称为子模式或外模式(External Schema)、用户模式,是逻辑模式的子集或变形,它是与某一应用有关的数据的逻辑表示。不同的用户需求不同,看待数据的方式也可以不同,对数据保密的要求也可以不同,使用的程序设计语言也可以不同,因此不同用户的子模式的描述可以是不同的。即使对逻辑模式中的同一数据,在子模式中的结构、类型、长度、保密级别等都可能不同。许多数据库管理系统提供了建立视图的方法(用子模式数据定义语言)、表示视图上的查询和操作的方法(用子模式数据操纵语言)。从某种意义上说,构造视图是数据库集成的逆过程。对组成逻辑模式的每一组数据,我们可构造一个仅包含某些数据的视图。视图对加强数据库的安全性也很重要。它可将用户对数据存取限制在视图的范围内,例 1.4 说明了视图的这种应用。

民航的售票系统是视图的一个应用例子。系统中包括了一组处理航班和旅客的程序,这些程序和使用程序的人员并不必知道诸如人事档案、丢失的行李、飞行员的航行分配等其他数据库中的信息。调度员可能需要知道关于航班、飞机和人事档案的某些信息(如哪些飞行员有资格驾驶 747),但不必知道雇员的工资、旅客等信息。于是,可以为订票部门建立一个数据库视图,为调度办公室建立另一个完全不同的视图。

视图往往对应一个小的逻辑数据库,一般它同逻辑数据库可看成在同一抽象级别。然而,因为视图处理的数据并不实际存储在数据库中,而仅可以从逻辑数据库中构造出来,视图比逻辑模式的抽象级别更高。例如,人事部门可能有一个视图,它包括每个雇员的年龄。但是年龄一般并不包括在逻辑数据库中,因为每天都可能有雇员的年龄改变了,所以逻辑数据库仅包括雇员的出生年月日。当一用户希望通过访问视图得到一个雇员的年龄时,DBMS 翻译这个要求成为“当前日期减去出生年月日”,再从物理数据库上取出相应的数据完成计算。

[例 1.6] 我们用有关程序语言中的例子来说明三级抽象模式之间的差别以及模式和实例的关系。

(1) 用程序语言定义一个二维数组:  $\text{INTEGER ARRAY } A[1 \sim n, 1 \sim m]$

这相当于数据库系统中的逻辑模式定义。

(2) 设置每个数组元素在物理设备上的存储位置为:

$$A[i, j] \text{ 存于 } A_0 + 4(m(i-1) + j - 1)$$

即从  $A_0$  位置开始, 一个数组元素占 4 个单元, 这相当于物理模式定义。

(3) 再定义一个一维数组:

$$F(i) = \sum_{j=1}^m A[i, j]$$

即数组  $F$  的每一元素由数组  $A$  的一行元素相加得到, 这相当于子模式定义。

(4) 假设数组  $A$  的  $m=n=3$ , 并给出  $A$  的具体元素, 这是一个魔方矩阵, 即逻辑模式的实例(逻辑数据库):

$$\begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix}$$

(5) 物理模式的实例(物理数据库)是连续存储的 9 个数, 从位置  $A_0$  开始存储, 按次序是 8, 1, 6, 3, 5, 7, 4, 9, 2。

(6) 子模式的实例(视图)是函数  $f(1)=f(2)=f(3)=15$ 。

### 1.2.2 数据独立性

图 1-2 中, 在从视图到物理数据库的三级抽象中提供了两级映像, 即子模式到逻辑模式的映像和逻辑模式到物理模式的映像, 这两级映像保证了数据库系统的数据独立性。数据独立性包含物理独立性和逻辑独立性两层意义。

很明显, 在数据库系统中, 当其物理模式发生改变时, 可以通过重定义逻辑模式到物理模式的映像, 使逻辑模式保持不变, 从而子模式不变, 则建立在子模式上的应用程序就不变, 即应用程序不因为数据库数据的物理组织改变而改变, 尽管此时可能会影响应用程序的执行效率, 而且逻辑模式的实现方法要改变, 这种独立性叫数据物理独立性。比如, 无论数组的物理实现是行为主(一行一行地存)或列为主(一列一列地存), 对数组的引用都能正确地工作。物理数据独立性的好处是: 我们可以在系统运行中调节物理数据库以改善系统的效率, 而不影响应用程序的运行。

子模式和逻辑模式之间提供了又一种独立性, 叫数据逻辑独立性。当数据库系统投入使用后, 可能有必要修改逻辑模式。例如, 要加入新的数据实体类型或向现存的实体类中增加别的信息。许多类型的逻辑模式修改能在不影响现存子模式的情况下完成。比如可以重定义子模式到逻辑模式的映像, 以保证现存子模式不变, 则应用程序就不用修改。有一种例外, 就是当对应某子模式的逻辑模式的信息被删除了, 这个子模式不能通过重定义逻辑模式到子模式的映射来实现, 此时应用程序会受到影响, 这是不可能避免的。

## 1.3 数据语言

在普通的程序设计语言中, 说明语句和执行语句都是同一种语言, 然而在数据库的语言中

则常要区分数据说明语言和数据操纵语言。因为在普通程序设计中,数据仅仅当程序运行时才存在,而在数据库系统中,数据一经定义就长期存在,所以分开定义数据和操纵数据的语言是有道理的。

### 1.3.1 数据定义语言(DDL)

数据库模式用数据定义语言来描述。逻辑模式用模式 DDL 定义,它以某种数据模型为基础描述数据实体和实体间的联系。子模式用子模式 DDL 定义,它类似于模式 DDL,子模式 DDL 有时可用与模式 DDL 不同的数据模型,而且可以有几种不同的子模式 DDL,定义不同的数据模式。物理模式用内模式 DDL 定义,而物理数据库的详细设计由数据库管理系统的实用程序完成。

[例 1.7] 下面的 SQL 数据定义语句用关系模型描述了航空公司的运行航班实体。

```
CREATE TABLE FLIGHTS
( NUMBER: INT PRIMARY KEY,
  DATE: CHAR(6),
  SEATS: INT,
  FROM: CHAR(3),
  TO: CHAR(3));      /* 定义逻辑模式,建立基表 */
CREATE INDEX IX FOR FLIGHTS ON NUMBER;      /* 定义物理模式,建立索引 */
```

该段代码描述了关系和它的属性,规定它们在物理级用整数和定长字符串实现并建立了按航班号的索引,索引的目的是为了使查找信息更快,不必搜索整个航班文件。

当设计数据库和修改数据库设计时运用数据定义语言。不能用数据定义语言查询或修改数据库中的数据。

### 1.3.2 数据操纵语言(DML)

在数据库上操作数据需要特殊的语言,称为数据操纵语言或简称查询语言,用这种语言可以实现数据库数据的插入、删除、修改、查询、统计等操作,比如下面的这些操作:

- ① 从数据库中检索 7 月 24 日的 999 航线上可用的座位数。
- ② 将 8 月 31 日的 123 航班的可用座位减少 4。
- ③ 找到 8 月 20 日从 ORD(芝加哥的 O'Hara 机场)到 JFK(纽约的肯尼迪机场)的所有航班。
- ④ 增加 456 航班入数据库,它有 100 个座位,在 8 月 21 日从 ORD 到 JFK。

操作①和③是查询数据库的例子,可用类似例 1.3 的程序来实现。操作②是更新数据库的例子,可用下列 SQL 语句行实现:

```
UPDATE FLIGHTS SET SEATS=SEAT-4
WHERE NUMBER=123 AND DATE="AUG31";
```

操作④是插入记录到数据库的例子,可用下列 SQL 语句实现:

```
INSERT INTO FLIGHTS VALUES(456,"AUG21",100,"ORD","JFK");
```

### 1.3.3 数据控制语言(DCL)

这是数据库中实现数据访问权限控制以及事务管理的一类语言,比如限制用户对某些数