第3章 数据库应用系统设计方法

数据库应用系统设计是指以计算机为开发和应用平台，以操作系统(OS)、某一商用数据库管理系统(DBMS)软件及开发工具、某一程序语言等为软件环境，采用数据库设计技术完成某一特定应用领域或部门的信息/数据管理功能的数据库应用系统的设计实现过程。这样的数据库应用系统可能是一个信息管理系统(或管理信息系统)，如教学信息管理系统、旅馆信息管理系统等；也可能是一个装备制造或工业控制软件系统中的数据库管理子系统/模块，例如在电厂生产监控管理系统中，由于涉及对数据库中数据和实时采集到的数据的综合处理与分析，所以必须有数据库管理子系统的支持。显然，这些系统中都涉及数据库的建立和对数据库中数据的运用。

掌握数据库应用系统设计实现的基本理论、基本技术和基本方法，对于理工学科门类和管理学科门类的本科生，特别是其中的计算机类和信息类专业的学生，把握各自专业领域计算机信息系统和装备制造系统中的控制软件设计中的共性问题，提高计算机的应用水平和开发能力等都具有十分重要的意义。

3.1 数据库应用系统设计概述

本节介绍数据库应用系统的生命周期和基本的设计方法和步骤，以便为后续内容的学习理清思路。

3.1.1 数据库应用系统的生命周期

数据库应用系统的设计是一个比较复杂的软件设计问题。

(1) 一个数据库应用系统首先是一个应用软件系统，所以其设计过程总体上应遵循软件生命周期的阶段划分原则和设计方法。

(2) 数据库应用系统的设计又涉及数据库的逻辑组织、物理组织、查询策略与控制机制等专业知识，所以又有自己独具特色的设计要求。

(3) 一个数据库应用系统的设计要求设计人员应具有一定的关于用户组织的业务知识或实践经验，而实际的数据库设计人员大多缺乏应用领域的业务知识，所以数据库应用系统的设计一直是一个极具挑战性的课题。

吸收软件工程思想并结合数据库设计技术的个性特点，一般把数据库应用系统从开始设计时的需求分析，到被新的系统取代而停止使用的整个时期，称为数据库应用系统的生命周期。早期由于数据库管理系统软件价格昂贵，且处在我国改革开放以前及改革开放初期的企事业单位经济实力也有限，建立基于数据库的信息管理系统对许多单位来说都会涉及

一个相对较大的经济投入。所以当时“数据库设计规划”被列为数据库应用系统设计的一个重要时期/阶段。随着计算机应用的普及和我国经济实力的不断壮大,基于数据库技术对各类信息的计算机管理已经成为信息化社会的基本条件,因此数据库设计规划目前不再是数据库应用系统生命周期中的一个时期/阶段。同理,随着数据库管理系统软件功能的不断扩充和技术的成熟,数据库应用系统的生命周期中各设计阶段的分工也在微调。

基于以上考虑,本书从第4版起,将数据库应用系统的生命周期划分成4个时期、6个阶段;4个时期分别是用户需求分析时期、数据库设计时期、数据库实现时期、数据库运行与维护时期;6个阶段分别是用户需求分析阶段、概念结构设计阶段、逻辑结构设计阶段、物理结构设计阶段、数据库应用行为设计与实现阶段、数据库运行与维护阶段。数据库应用系统的生命周期及各个时期对应的阶段划分如图3.1所示。

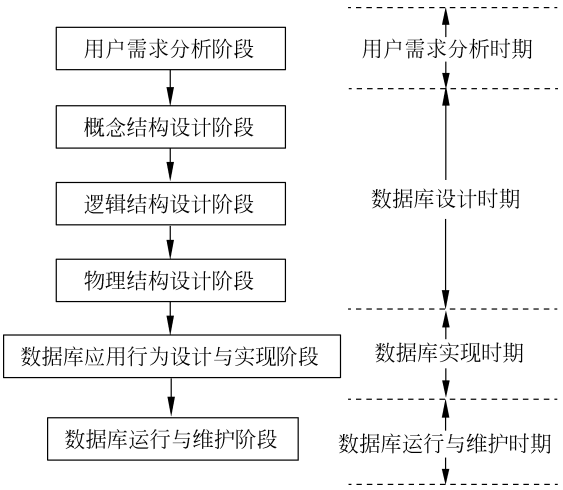

图3.1 数据库应用系统的生命周期及各个时期对应的阶段划分

3.1.2 数据库应用系统设计方法

从理论上讲,数据库应用系统的设计涉及数据库应用系统生命周期的各个时期和阶段,但在数据库应用系统正式投入运行和使用后的漫长阶段中,除了设计者必要的改正性维护(修改软件投入运行后发现的某些软件设计错误)、适应性维护(为了适应变化了的软件和硬件环境而进行的修改软件活动)、完善性维护(修改某些设计时考虑不周的情况和因增加新功能而修改软件)外,基本上都属于用户对该系统的使用、管理和维护问题。因此,数据库应用系统的设计方法主要涉及用户需求分析、数据库设计、数据库实现、数据库运行与维护4个时期。

1. 用户需求分析时期

用户需求分析时期的主要工作包括分析用户对数据管理的功能需求、应用需求和安全性需求,是进行数据库应用系统设计的基础。用户需求分析的结果能否准确反映用户对数据管理应用需求的实际要求,将直接影响以后各个阶段的设计工作,并事关整个数据库应用系统设计的成败。

2. 数据库设计时期

数据库设计时期的主要工作是针对某一用户组织的数据管理应用需求,设计(构造)最

优的数据库概念结构、逻辑结构和物理结构，使之能有效地存储和管理数据，以满足用户的信息管理和数据操作需求。

3. 数据库实现时期

数据库实现时期的主要工作是依据数据库应用系统的功能需求和数据库物理结构设计阶段设计好的存储结构和存取方法，创建数据库、创建数据库表和设计数据库的子模式，给已经创建好的数据表装入实验数据，设计编程实现能够满足该用户组织中各个用户对数据库应用需求的功能模块及其行为特性，装入实际数据进行系统试验性运行等。

4. 数据库运行与维护时期

数据库应用系统运行与维护时期的主要工作包括：必要的改正性维护、适应性维护、完善性维护；数据库转储备份与恢复及故障维护；数据库运行性能的检测与改善等。

3.1.3 数据库应用系统研发、管理和使用人员视图级别

数据库应用系统的使用、开发和管理人员主要有数据库应用系统用户（简称用户）、应用程序员、系统分析员和数据库管理员（Database Administrator，DBA）。不同人员所看到的数据库是有区别的，即数据库应用系统具有不同的视图级别。了解这些人员在数据库应用系统生命周期中各阶段的作用，对于数据库应用系统的设计和实现及管理使用都是有意义的。图 3.2 给出了数据库应用系统的视图级别模型。

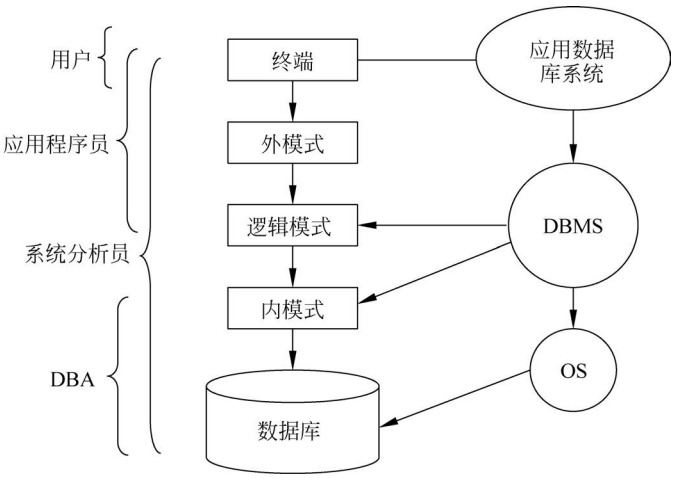

图 3.2　数据库应用系统的视图级别模型

图 3.2 形象地描述了在数据库使用、开发和管理过程中，不同人员看到的数据库和所处的角色。

（1）用户，仅指使用数据库应用系统的人员。一般指从事某一具体领域工作的业务管理人员，只需熟练掌握该数据库应用系统的使用方法，而无须了解该数据库应用系统的有关设计问题。但在比较小型的数据库应用系统中，一般不配置专门的数据库管理员，所以用户同时要承担 DBA 的职责。

（2）应用程序员，仅指那些在数据库应用系统设计中专门从事应用程序编写的程序员。一般情况下，应用程序员编程需要知道（看到）数据库的逻辑模式、外模式和自己编写的程序模块的输出结果界面（终端）。但在一些情况下，部分应用程序员也会与系统分析员一起完

成数据库外模式、逻辑模式和存储模式的设计,并根据外模式、逻辑模式和输出结果界面要求编写应用程序。

(3) 系统分析员,指在数据库应用系统设计中负责系统需求分析,承担数据库应用系统的软、硬件配置,参与数据库各级模式设计的工作人员。所以系统分析员实质上是指那些负责数据库应用系统设计的总体设计人员或总体设计师。在一个数据库应用系统的设计中,要求系统分析员熟悉计算机系统的软、硬件知识;熟悉数据库应用系统的设计技术;熟悉系统设计中所用的数据库软件等。

(4) 数据库管理员(DBA),仅指在数据库系统运行过程中监管系统运行情况,改进系统性能和存储空间管理,负责数据库的备份与恢复等工作的系统管理人员。数据库管理员的职责如下。

① 数据库运行情况的监控。确保数据库始终处于正常运行状态。

② 数据库数据的备份。即按要求定期备份数据库。

③ 数据库的恢复。即当数据库在运行过程中遇到硬件或软件故障时,负责恢复数据库软件的正常运行,恢复数据库中数据的正确性。

④ 数据库的存储空间管理与维护。根据数据库存储空间的变化情况进行存储空间分配;根据存储效率情况对存储空间进行整理,如收集碎片等。

需要说明的是,这里给出数据库应用系统研发、管理和使用过程中的人员视图级别,目的是厘清不同人员的职责,以便于更好地理解数据库应用系统设计过程中各类人员的作用和地位。

3.2 用户需求分析

围绕数据库应用系统设计进行的用户需求分析,包括管理的数据需求、系统功能需求、系统环境配置及安全性需求。

用户需求分析阶段的主要任务是了解用户组织的机构,建立用户组织的结构层次方框图;分析用户的业务活动,建立用户的数据管理业务数据流图;收集所需数据,整理数据库中的信息内容;分析用户的数据处理要求和数据安全性与完整性要求;确定系统功能和软硬环境配置,最终形成系统需求分析说明书。

数据流图和数据字典是描述用户需求的图形和表格表示手段,因而在系统地介绍用户需求过程之前,先介绍数据流图和数据字典。

3.2.1 数据流图

数据流图(Data Flow Diagram,DFD)是一种用于描绘系统逻辑模型的图形工具,是逻辑系统的图形表示。数据流图只关心系统需要完成的基本逻辑功能,而无须考虑这些逻辑功能的实现问题,所以数据流图中没有任何具体的物理元素,只从数据传递和处理的角度反映信息在系统中的流动情况。

数据流图一般用图3.3中的4种基本符号表示。

1. 数据源点或终点

数据的源点指数据的起源处;数据的终点指数据的目的地。数据的源点和终点分别对应于外部对象,这些外部对象是存在于系统之外的人、事物或组织,如作者、出版社、书库管

图 3.3　数据流图的基本符号

理员等。数据的源点或终点用方框表示，对外部对象的命名写在方框内。

2. 数据处理

数据处理是对数据流图中的数据进行特定加工的过程。一个处理可以是一个程序、一组程序或一个程序模块，也可以是某个人工处理过程。对数据的处理用圆圈表示，表示每个处理功能或作用的名称一般写在圆圈内。

3. 数据存储

数据存储代表待处理的处于静态状态的数据存放的场所。一个数据存储可以是一个文件、文件的一部分、一个数据库、数据库中的一个记录等。文件可以是磁盘、磁带、纸张或其他存储载体或介质上的文件，数据库也可以看作一个文件。数据存储用右开口的长方形表示。数据存储的名称写在右开口的长方形中。

4. 数据流

数据流指数据流图中数据的流动情况，用单向箭头表示数据流图中由它连接的两个符号间的数据流动，单向箭头的指向即为数据的流动方向。除流向或流出数据存储的数据流可以不命名外（因为其含义已经表示了对文件的存入或读取操作），一般都要给出流动的数据的命名，并写在相应单向箭头的旁边。

【例 3.1】 用数据流图描述图书预订系统的业务处理逻辑。答案如图 3.4 所示。

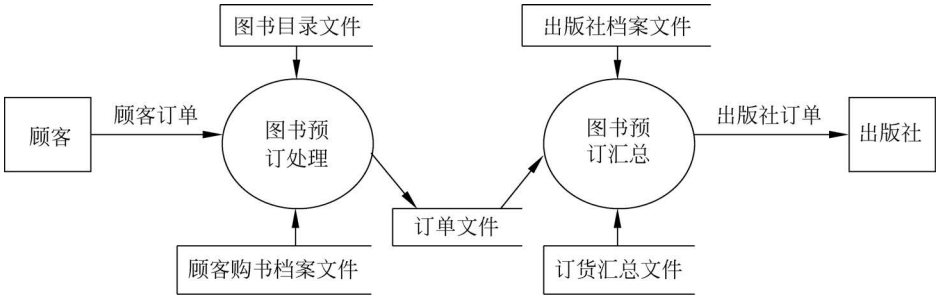

图 3.4　描述图书预订系统处理逻辑的数据流图

3.2.2　数据字典

数据流图表示了数据与处理的关系，但无法表达出每个数据和处理的具体含义和详细描述信息。数据字典（Data Dictionary，DD）用于详细地给出数据流图中所有数据的定义和描述信息，是描述和定义数据流图中所有数据的集合。数据字典通常包括以下四部分。

1. 数据项

数据项是不可再分的数据单位，是组成数据流的基本元素。数据项的定义和描述信息主要有数据项名、别名、含义、类型、长度、取值范围、使用频率、使用方式、与其他数据项的逻辑关系等。其中，“取值范围”和“与其他数据项的逻辑关系”是定义数据完整性的约束条件。

2. 数据流

数据流表示数据处理过程中的输入或输出的数据,可以是数据项,也可以是由数据项组成的某种数据结构的数据单位。对数据流的定义和描述信息主要有数据流名、含义、组成数据流的数据项或数据结构、数据流的来源或去向、数据流的流量等。

3. 数据表

数据表是信息管理中最常见的数据格式,许多数据表与数据库逻辑设计后的关系模式有一定的对应关系。对数据表的描述信息主要有数据表名、数据表中各字段序号、字段名、数据类型、字段长度、字段名含义(备注)、数据表的所有者等。这些信息为数据库逻辑模式中相关信息的确定奠定了良好的基础。

表3.1是进行大学教学信息管理数据库应用系统的数据需求分析时,建立的数据字典中的一个典型的数据表。

表3.1 课程数据表

序 号	中文名称	数据类型	字段长度	备 注
1	课程代号	字符串	7	课程代号
2	课程名称	字符串	20	课程名称
3	课程类型	字符串	8	课程类型
4	学时	整数	3	学时
5	学分	整数	1	学分
6	任课教研室	字符串	12	任课教研室

4. 处理

处理表示一个处理所要完成的工作或功能。对处理的定义和描述信息主要包括处理的名称、处理的定义或描述、流入和流出处理的数据流、执行频次等。

数据字典没有统一的格式,可以按照自己对各条目内涵的理解设计一套通俗易懂的图表或文档格式。数据字典的编制可以用手工方式书写,也可以借助文字处理软件在计算机上实现。数据字典在用户需求分析阶段配合画初步的数据流图时初步建立,并伴随数据库设计过程和设计方案的不断改进和完善而不断修改、充实和完善。

3.2.3 用户需求分析过程

1. 分析用户的业务活动,建立用户业务数据流图

了解用户组织中各业务的活动内容,建立描述用户业务处理及其信息流动过程的数据流图,是用计算机自动或部分自动地实现满足用户业务流程要求的信息化管理的关键步骤。所以,需要详细了解各用户(科员或部门)当前业务活动、业务流程和业务处理中各环节之间的信息流动顺序、处理顺序、需要的信息存储支持和处理的结果信息存储需求;梳理各业务活动和业务流程中的输入信息和输出信息,及其与中间信息的关系。

分析的基本方法是和用户(主管领导、科室业务人员)进行个别询问和座谈交流;查阅各部门的业务处理记录和档案资料;视情进行必要的跟班作业;在此基础上对在以往的业务活动中或模棱两可,或互相推诿的问题进行问卷调查,在广泛征求各方面意见的基础上给出合理的业务处理流程,并以数据流图的形式给出描述用户业务处理过程及其数据详细流动情况的系统逻辑模型。

本部分需求分析的结果是建立描述用户业务信息流动和处理逻辑的数据流图。

2. 整理用户业务信息流动和处理的数据信息，建立描述数据信息的数据字典

(1) 要收集和整理描述用户业务信息流动和业务处理要求所涉及的各种数据信息，包括各种账表、单据、报表、合同和档案中的数据描述要求，从各种规章、制度和业务处理文件中抽取出来的数据描述信息。

(2) 通过进一步梳理和分类，标注出各个数据所相关的业务范围或相关部门。例如，可将教学管理数据库中的学生的学号、姓名、性别、出生日期等有关反映学生自然情况的数据信息作为一类。

(3) 确定每类信息中数据元素的确切的名称、类型、长度、取值范围和应用特征，例如该数据元素是否可为空值(NULL)等。

(4) 进一步弄清每类信息允许哪些用户执行哪些操作及操作的频度等。

本部分工作的基本方法是从涉及的各种文档资料中收集、分析和整理建立数据库所需的数据信息，并通过必要的个别交谈咨询和问卷调研等措施完善收集和整理的数据信息。本部分工作的结果主要是完成数据字典的编制。

3. 分析用户的信息处理要求，确定系统的信息管理和系统处理功能

宏观上来说，数据库应用系统的基本功能就是实现对要管理的业务数据信息的录入、删除、修改、查询和报表输出打印等，但不同的应用领域和管理层对信息的管理和处理都有各自不同的特定要求。因此，这一步是在前几步工作的基础上，进一步了解和细化用户组织中各业务部门的信息处理要求，即他们希望从数据库中获得哪些信息，要获取的信息包括哪些具体内容；希望数据库应用系统完成什么样的处理功能，对数据的处理方式和响应时间有什么样的要求。在此基础上，根据计算机目前的处理能力来确定系统应实现的功能和所应具有的性能。

本步工作的基本方法包括个别交谈询问和集体座谈交流，查阅业务处理记录和档案资料，进行必要的跟班作业。分析的结果是以文档形式描述的系统功能需求列表、系统性能要求列表和必要的辅助说明信息。

4. 调研用户组织的机构组成和地理分布，初步确定系统的软硬环境配置方案

本部分主要是通过对用户组织的机构组成、各部门的职责及其相互关系、各部门的规模和地理分布范围等信息的调研了解，为系统网络环境及体系结构、系统硬件环境及性能要求、系统的软件环境配置及开发工具需求等的确定提供依据。

调研了解的基本方法是与该用户组织中的有关领导和业务主管进行座谈，索取和收集相关的文档资料。在调研了解的基础上，勾画出一张能够比较全面反映该用户组织机构及其相互关系的组织机构层次方框图，如图 3.5 所示的大学教务部门的组织机构层次方框图。

根据各部门的地理分布情况和初步的投资意向，初步确定系统的网络拓扑结构和网络软硬件配置；根据整个系统的信息处理需求，确定系统相关的硬件和软件配置，给出相应的开发平台与开发工具需求。同时，也要根据用户组织的性质和信息安全要求，确定相应的信息安全措施，例如网络防火墙的配置、数据库加密措施、信息传输中的安全措施等。

5. 收集、整理需要进行管理的具体数据

通过这一步的工作，一方面可通过现有的实际数据及其组织格式的分析，进一步完善数据字典中有关数据信息的描述；另一方面，可进一步为数据库概念结构和数据库逻辑结构的设计提供参考；同时，也可为后续的系统实现验证和实际数据的录入奠定基础。本部分

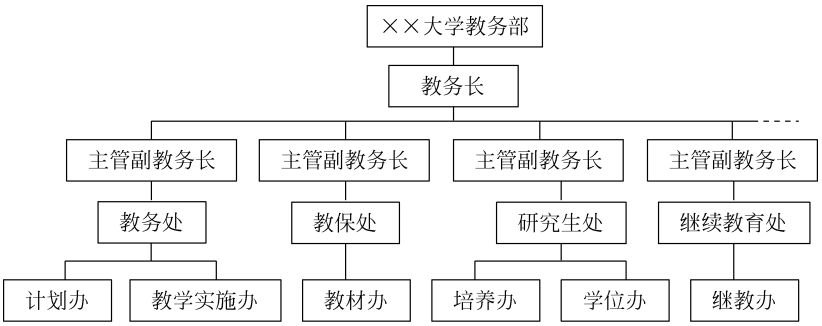

图 3.5　组织机构层次方框图示例

的工作可能一直延续到系统设计实现完成为止。

通过以上用户需求分析过程,可以最终形成完整的系统需求分析说明书。通常情况下还需要对需求分析说明书进行评审,根据评审意见进一步修改需求分析说明书。

3.2.4　数据库应用系统的功能需求

数据库应用系统的功能包括信息管理功能、信息处理功能和辅助决策功能。由于用户组织的层次、规模及应用领域的不同,不同的数据库应用系统的功能一般都有较大的区别。因此,要根据用户组织的特点、应用背景及决策需求,合理梳理功能需求类别,确定数据库应用系统的功能模块。下面通过两个实例说明不同数据库应用系统的功能需求。

1. 教学信息管理数据库应用系统的功能需求

实现教学信息管理是高等院校数据库管理系统最基本的信息管理要求,一般至少应具有下列主要功能模块。

1) 学生信息管理

学生基本情况的录入、修改、删除和灵活多样的查询功能;学生选课信息的录入、修改、删除和查询功能;学生课程考试成绩的录入、修改、删除和灵活多样的查询功能(如统计、排名等);学生留级、休学等特殊信息的管理功能。

2) 课程信息管理

课程基本信息的录入、修改、删除和查询功能;专业选课信息及选课要求信息的查询功能;课程安排及其主讲教师信息的查询功能等。

3) 院系及专业信息管理

院系机构设置和专业设置信息的管理,包括相关的录入、修改、删除和查询功能。

4) 常用教学信息报表输出

根据各种信息汇总要求,设置信息报表布局格式,将通过多表查询获得的各项数据填入表中,形成制式的报表并输出。

2. 企业网站的功能需求

各种网站实质上是一个网络信息管理系统,因此,一个功能较强且具有动态信息管理功能的网站一般应包含下列功能模块。

1) 新闻发布

管理员可对新闻进行增加、修改、删除,可上传相关图片等。

2）产品展示

用于展示企业的产品性能、指标和用途等信息。管理员可对产品进行增加、修改、删除，可设定打折额度及期限，可上传和更换产品图片等。

3）用户管理

支持客户通过网站进行注册。企业可通过客户信息的查询等，掌握客户的基本情况、产品需求情况和客户等级，并为客户提供必要的信息服务。

4）需求调查与信息反馈

支持客户通过该模块提出自己的产品需求，或反馈所购产品的质量信息等；可实现对客户需求和反馈信息的浏览、处理和答复。

5）网上购物

相当于电子商务中的在线销售模块。支持客户资料的管理、商品信息的管理、订单处理等；支持商品的进、销、存；支持客户的回访、意见受理等功能。

6）人才招聘

支持企业的人才招聘。应聘者可通过该模块进行注册，填写个人详细信息，并根据企业招聘要求选取合适的工作需求；企业管理人员可对应聘人员的信息进行浏览、筛选和招聘后进行删除等。

7）企业论坛

用于企业内部或对外进行交流。可自行定制论坛版块，管理员或领导可向全体人员发公开信，所有发布资料均自动记录在数据库中，以便后期查询及汇总。

3.2.5 数据库应用系统环境配置与安全性需求

1. 数据库应用系统的环境配置

在用户需求分析中，通过对用户组织的信息管理及应用功能的需求分析，可以初步获知要建立的系统的复杂程度。通过对用户组织机构组成的调研和分析，可以获知用户组织的规模、管理层次及其相互间的业务关系。通过用户组织各分机构的地理分布，例如，该用户组织位于一个办公室内，仅有比较单一的业务；该用户组织位于一个建筑物内；该用户组织位于一个占面积几千亩地的大院内；该用户组织的分支机构位于一个城市的不同街区；该用户组织的分支机构位于不同的城市等，就可获知该用户组织机构的地理分布情况。以此为基础就可确定要建的数据库应用系统的环境配置。

1）单机数据库应用系统还是基于网络的数据库应用系统

对于位于一个办公室内，且仅有比较单一业务的应用来说，一般采用在单个计算机上运行的单用户数据库应用系统。这类系统不需要考虑进行外部连接的网络环境。对于位于一个建筑物内的用户组织，一般采用局域网络环境，网络服务器和数据库服务器等设在信息中心，用户应用的客户端或浏览器计算机直接连接到各办公室。对于分支机构位于一个城市的不同街区，或位于不同城市的用户组织来说，一般采用广域网络环境，目前大多数采用Internet，同样将网络服务器和数据库服务器等设在信息中心，用户应用的浏览器计算机直接连接到各办公室。

对于基于网络的数据库应用系统来说，涉及系统网络拓扑结构和路由连接方式的确定；网络服务器和数据库服务器及网络等设备的选择；数据库应用模式的选择（采用C/S模式

还是采用B/S模式,详见第7章)等。

2) 采用的操作系统和数据库软件

目前,单机数据库应用系统和基于网络的数据库应用系统中的用户端计算机上基本采用Windows操作系统。数据库管理系统软件一般根据系统规模、用户要求和用户应用背景等确定,例如系统规模比较小、应用功能不太复杂的系统可以采用Access数据库;中等规模和中等复杂度的系统可以采用SQL Server数据库;有些军用系统则要求用Oracle数据库。开发数据库应用程序的主语言主要有CB(C++Builder)、VC(Visual C++)、VB(Visual Basic)等,大多采用开发人员熟悉的语言。有关开发辅助工具的选择,主要取决于系统规模、经费投入大小和开发人员的需要和习惯等。

2. 系统的安全性需求

系统的安全性需求因系统应用背景的不同一般差别较大。最基本的系统安全性需求包括以下两方面。

1) 数据库数据的备份

数据库数据的备份是指,人为定期地或系统自动定期地将整个数据库(文件)或数据库中的数据复制到别的存储介质上并放在别处的过程。这样,出现不可抗拒的故障时,就可利用备份的数据库数据,将系统中的数据库恢复到最后一次备份时的状态,并利用数据库管理系统的恢复机制和系统日志文件等,恢复最后一次备份时至系统遭受破坏这一期间的数据库信息,以便将数据的丢失和破坏减少到最低限度。有关概念详见第10章。

2) 用户操作权限的授权和控制

用户操作权限的授权和控制是指,按不同用户对数据库中数据操作应用涉及的范围授予不同的操作权限。一般系统管理员具有对数据库的一切操作和管理权限;具有局部数据管理权的用户仅授予他/她所涉及的那部分数据的数据更新(插入、删除和修改)权限,可以授予他/她所涉及的全部业务数据的查询权限;没有局部数据管理权限的其他用户仅授予他/她所涉及的业务数据的查询权限。有关概念详见第9章。

进一步的信息安全措施还有数据库加密等。当然,网络防火墙的配置、信息传输中的安全措施等都会对数据库应用系统的安全起到进一步的保护作用。

3.3 数据库概念结构设计

概念结构是一种能反映用户观点并更接近于现实世界的数据模型,也称概念模型。数据库的概念结构一般是指用来表示用户组织中所涉及的所有对象和事物(即实体),以及它们之间联系的一组实体-联系模型(Entity Relationship Model,E-R模型)。因此,概念结构设计的目标就是从需求分析中找到实体(集)、确认实体的属性、确认实体之间的联系,设计出反映用户组织信息管理需求的E-R模型,并进而确定信息的正确性和完整性。由于E-R模型是一种图示表示形式,所以E-R模型也称为实体-联系图(Entity Relationship Diagram,E-R图)。

E-R图数据库概念结构设计方法涉及较多内容,下面分成多节分别介绍构建E-R图时涉及的实体集与联系集、实体集之间的联系,以及E-R图的概念结构设计方法。

3.3.1 实体与实体集

1. 实体与实体集的概念

实体(Entity)是存在于用户组织中的抽象的但有意义的“事物”,是用户组织中独立的客体。

(1) 实体可以是具体的对象。例如,一所学校、一个班。

(2) 实体可以是抽象的对象。例如,一次考试、爱和恨这样的概念。

(3) 实体可以是有形的对象。例如,一个学生、一把椅子。

(4) 实体可以是无形的对象。例如,专业、年级。

实体集是具有相同类型及相同属性的一组实体构成的集合。例如,一个专业的学生可构成一个学生实体集;所有的课程构成一个课程实体集;所有的福特汽车可构成一个福特汽车实体集。

2. 属性与实体集的标识码

实体集通过一组属性表示。属性(Attribute)是指实体集中所有实体所具有的共同特征。例如,不同专业的学生实体集的属性都包括“学号”“姓名”“性别”“出生日期”等。

属性的值指属性的具体取值。例如,学生张华,其“姓名”取值为张华,“学号”取值为201401001,“性别”取值为男,“出生日期”取值为1996-12-14。

属性的值域(Domain)指属性的取值范围。例如,大学生的“性别”的值域为{男,女}。属性的值域可以是整数的集合、实数的集合、字符串的集合,或其他类型的值的集合。

当某实体集的每个属性使它的值域中的一个值同该实体集中的一个实体相联系时,就具体地描述了该实体集中的某个特定的实体。例如,设学生实体集的属性分别具有如下值域。

(1) “姓名”的值域为{张华,李建平,王丽丽,…… }

(2) “学号”的值域为{201401001,201401002,201401003,…… }

(3) “性别”的值域为{男,女}

(4) “出生日期”的值域为{1996-12-14,1996-08-20,1997-02-02,…… }

则当该学生实体集的每个属性使它的值域中的一个值同该实体集中的“张华”这个特定实体相联系时,学生张华的属性值集合{张华,201401001,男,1996-12-14,……}就具体地描述了该学生实体集中的一个特定实体——张华的特征。用来表示一个特定实体的属性值集合称为实体记录,例如{张华,201401001,男,1996-12-14,…… }是一个实体记录。实体记录集即为同类实体的实体记录的集合。

一般地,把能够唯一地标识实体集中的每个实体的一个或一组属性称为实体集的标识码(Identification Key),并用实体集的标识码标识实体集中的不同实体。

3.3.2 实体集之间的联系及联系集

1. 实体集之间的联系

实体集之间的联系(Relationship)指实体集之间有意义的相互作用及对应关系。实体集间的联系有“一对一联系”“一对多联系”和“多对多联系”3种。

1) 一对一联系

实体集 E1 中的每个实体(至少有一个)至多与实体集 E2 中的一个实体有联系；反之，实体集 E2 中的每个实体至多与实体集 E1 中的一个实体有联系，则称 E1 和 E2 是一对一联系(One-to-one Relationship)，记为 1∶1。图 3.6 中左面的图描述一对一联系的概念，右面的图是一对一联系的符号表示。

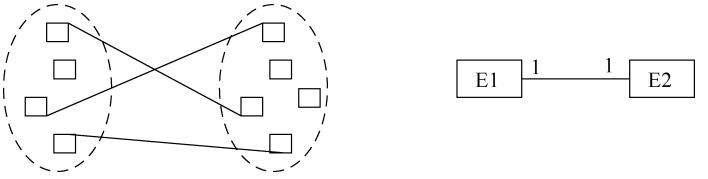

图 3.6　一对一联系

例如，住院的病人与床位之间、飞机的座位与乘客之间都是一对一联系。

2) 一对多联系

实体集 E1 中至少有一个实体与实体集 E2 中的一个以上的实体有联系；反之，实体集 E2 中的每一个实体至多与实体集 E1 中的一个实体有联系，则称 E1 和 E2 是一对多联系(One-to-many Relationship)，记为 1∶N 或 1∶*，如图 3.7 所示。

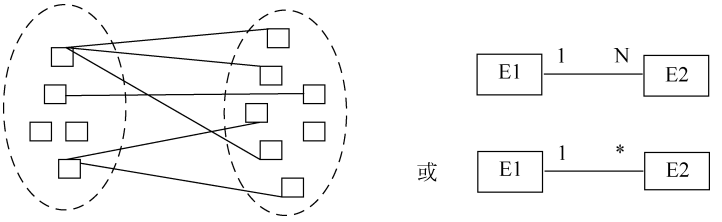

图 3.7　一对多联系

例如，系与学生之间是一对多联系。

3) 多对多联系

实体集 E1 中至少有一个实体与实体集 E2 中的一个以上的实体有联系；反之，实体集 E2 中至少有一个实体与实体集 E1 中的一个以上的实体有联系，则称 E1 和 E2 是多对多联系(Many-to-many Relationship)，记为 M∶N 或 *∶*，如图 3.8 所示。

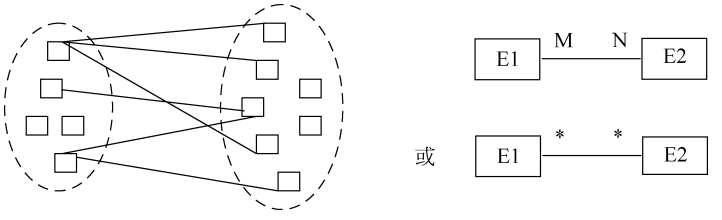

图 3.8　多对多联系

例如，学生与课程、教师与学生之间都是多对多联系。

2. 联系集

联系集(Relationship Set)指具有相同属性的联系构成的集合。同理，把能够唯一标识联系集中的每个联系的一个或一组属性称为联系集的标识码。联系集的标识码由被它联系

的两个实体集的标识码组成。

3.3.3 E-R 图及设计方法

E-R 图是一种以直观图示的方式描述实体(集)及其之间联系的语义模型,是一种十分有效的数据库概念结构描述工具。

在 E-R 图中,每个实体集用一个矩形框表示,并将实体集的名字记入矩形框中;每个联系集用一个菱形框表示,并将联系集的名字记入菱形框中;每个属性用一个椭圆形框表示,并将属性的名字记入椭圆形框中。有时为了直观,在标识码属性名下面画一条横线;用一条直线表示一个实体集与一个联系集之间的联系,并在直线的端部标注联系的种类(1∶1、1∶N、M∶N 或 1∶1、1∶∗、∗∶∗)。

下面通过对于每个学生来说最有亲身体会的大学教学信息管理的例子来说明,如何用 E-R 图描述一所大学的教学信息管理情况,同时给出了 E-R 图的设计方法。

【例 3.2】 仅以专业、课程、学生和教师为客体,设计反映一所大学的教学信息管理情况的 E-R 图。

分析:我国的大专院校虽然管理体制各具特色,但就其专业、课程、学生和教师之间的关系来说,可描述如下。

(1) 一个大学有多个专业,每个专业用唯一的专业代码和专业名称标识。

(2) 每个专业设置有多门课程,某些课程可被多个专业设置。

(3) 一个专业有多个学生;一个学生只能属于某一专业,一个学生并属于某一个班级。

(4) 一个学生必须学习多门课程,多个学生可以同时学习同一门课程。

(5) 每位教师可以主讲多门课程,绝大多数课程由多位教师主讲。

由上述分析可知,专业、课程、学生和教师均可分别看作实体,并对应有专业实体集、学生实体集、课程实体集和教师实体集。同时:

(6) 学生实体集与专业实体集之间的联系可用多对一(∗∶1 联系)的“归属”联系集来联系,且每个学生属于“归属”联系集的某个班级。

(7) 专业实体集与课程实体集之间的联系可用多对多(∗∶∗联系)的“设置”联系集来联系。

(8) 学生实体集和课程实体集之间的联系可用多对多(∗∶∗联系)的“学习”联系集来联系,且通过该课程学习后具有学习成绩“分数”。

(9) 教师实体集和课程实体集之间的联系可用多对多(∗∶∗联系)的“讲授”联系集来联系。

进一步分析每个实体集应具有的基本属性的基础上,可以得到如图 3.9 所示的 E-R 图。

由图 3.9 可知,“专业代码”是专业实体集的标识码,“学号”是学生实体集的标识码,“课程号”是课程实体集的标识码,“教职工号”是教师实体集的标识码。联系集的标识码由被它联系的两个实体集的标识码组成。所以,归属联系集的标识码为“学号”和“专业代码”;设置联系集的标识码是“专业代码”和“课程号”;学习联系集的标识码是“学号”和“课程号”;讲授联系集的标识码为“教职工号”和“课程号”。

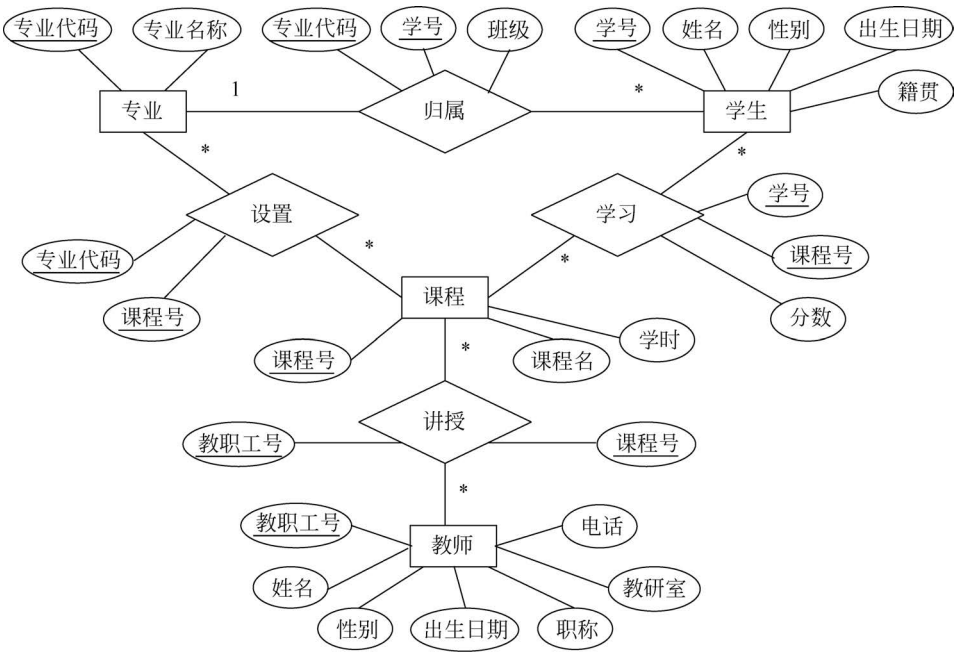

图 3.9　大学教学信息管理 E-R 图

3.3.4　实体-联系模型设计中的一些特殊情况

如图 3.9 所示的 E-R 图是一种比较规范的实体-联系图。现实中数据之间的联系十分复杂,在 E-R 图设计中还会遇到一些特殊情况。

1. 递归联系

之前介绍的联系是指不同实体集之间的联系。在实际应用中,有时还需要对"同一实体集"中的不同实体之间的联系进行模型化,这种联系称为递归联系(Recursion Relationship)。图 3.10(a)给出的递归联系:联系集是 MANAGE(管理),每个联系的一方是 MANAGER(经理),另一方是 MANAGED(被管理的人)。由于经理也是职工中的一员,所以实质上描述的是同一实体集中不同子集之间的联系。体现在图中就是同一实体集框对同一联系集框有两根连线,而且连线旁边标记了实体介入联系的方法。图 3.10(a)中的 1 : * 联系表示的含义是:一个 MANAGER(经理)可以管理多个 MANAGED(被管理的人),每个被管理人最多只能有一个直接管理人。图 3.10(b)和图 3.10(c)分别是其他两个递归联系的例子。

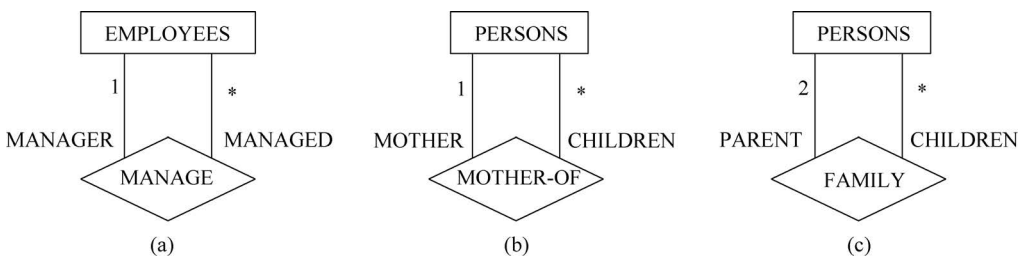

图 3.10　递归联系的 E-R 图表示

2. 冗余联系

冗余联系指在设计 E-R 图中建立的多余的联系。

【例 3.3】 在如图 3.11 所示的 E-R 图中，产品实体集与用户实体集之间的联系就是冗余联系。因为，某产品被供应给了哪些用户的信息，可以通过用户、合同、产品这 3 个实体集及它们相互之间存在的“签订”和“订货”两个联系来导出。

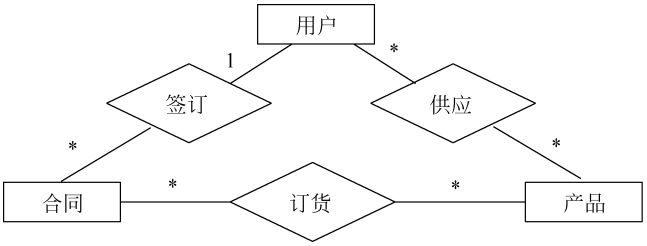

图 3.11 冗余联系示例

3. isa 联系

在 isa 联系中，A isa B 表示实体集 A 包含在实体集 B 中，A 是 B 中的一种特殊的群体。所以，isa 联系实质上是对同一实体集中的实体之间的联系进行模型化。下面通过实例说明 isa 联系如何用 E-R 模型描述的。

【例 3.4】 在图 3.12 的航空公司 E-R 图描述中，飞行员和机组人员之间存在 isa 联系，即飞行员也是机组人员的成员之一。

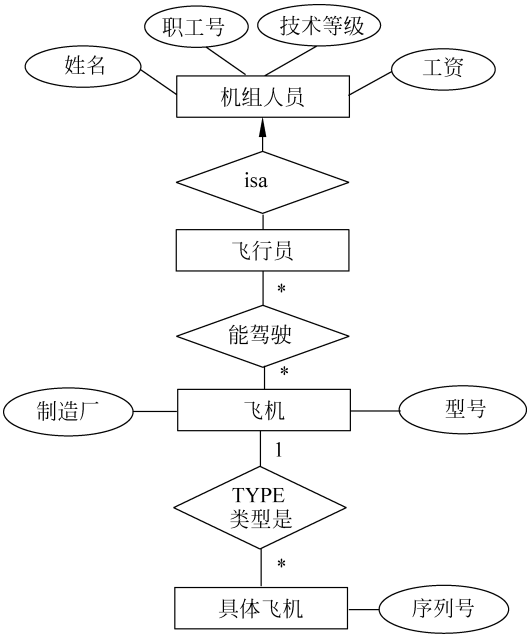

图 3.12 航空公司飞行员和机组人员的 isa 联系 E-R 图

其中：

(1) 把飞行员分离出来作为单独的实体集，一是为了使他同飞机实体集建立“能驾驶”联系，以表示每个飞行员能驾驶哪几种飞机；二是除了用机组人员的属性标识飞行员外，还

可以再给飞行员实体集补充别的属性(例如,用于记录飞行员的训练及健康等信息)。

(2) 飞机实体集与具体飞机实体集之间存在的 TYPE(类型是)联系是一种“具体意义上的联系”。具体飞机有唯一的序列号,每次出航对应一具体飞机。同一航线上同一型号的飞机,如波音 747,可能有许多架,它们具有不同的序列号。这就使多架具体飞机对应一种型号的飞机,而一种型号的飞机对应许多架具体飞机,所以 TYPE 联系是具体飞机实体集与飞机实体集之间的多对一联系。

4. 弱实体与弱联系

如果某实体集 E2 的存在依赖于另一个实体集 E1 的存在,并且这两个实体集之间的联系是用 E1 来标识的,那么就把实体集 E2 称为弱实体(Weak Entity)。在 E-R 图中用双矩形框表示弱实体。如果由联系集所联系的某些实体集是由其他联系集来标识的,那么就把这种联系称为弱联系(Weak Relationship)。

【例 3.5】 在图 3.13 的例子中,子女依赖于教职工的存在而存在(例如,在大学的附属医院中,子女随父母享受部分医疗待遇的情况),并且联系“抚养”是用来标识子女的,子女实体集是由“子女号”和抚养他(她)的“教职工号”来标识的,因此子女实体集是职工实体集的弱实体。

同理,子女实体集是由它的子女号,及它和一个教职工实体集的联系集(“抚养”联系集)来标识的(由教职工号体现),则该教职工所抚养的子女实体集和其他实体集之间的任何联系(集)都将导致弱联系(集),即对于和某弱实体集相联系的其他实体集来说,它们与弱实体集之间的联系都是弱联系。

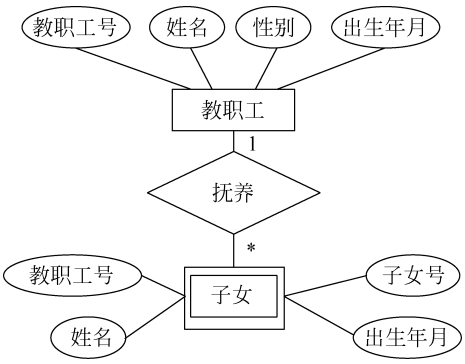

图 3.13 弱实体示例

5. 组合实体集(聚集)

在有银行贷款支持的购房交易中,由第三方(银行)提供担保的客户与项目之间存在的订购关系。逻辑上可以用如图 3.14(a)所示的 E-R 图描述该担保-订购关系。

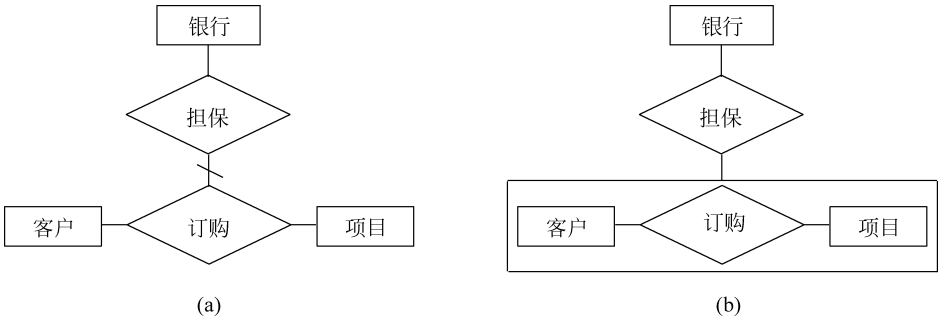

图 3.14 担保-订购关系的 E-R 图

显然,图 3.14(a)的 E-R 图违反了“联系集是实体集与实体集之间的联系”的 E-R 图设计约定。解决该问题的方案是:可以将该联系集(订购)和被其联系的实体集(客户和项目)组合(聚集)成高层实体集(组合实体集),并约定组合实体集也可以被看成一个实体集,并由

某一联系集把它和其他实体集联系起来。这样，图 3.14(a)的 E-R 图就可以表示成图 3.14(b)的 E-R 图了。把这种由两个实体集及一个联系集组合在一起形成的高层实体集称为组合实体集。

【例 3.6】 某海外代购公司为扩展公司业务，需要开发一个信息化管理系统。请根据其管理需求和涉及的实体集及其相互之间的联系描述，完成该系统的 E-R 图设计(本例题选自 2018 年上半年全国软件设计师综合技能考试试题二)。

该系统的管理需求和涉及的实体集及其相互之间的联系描述如下。

(1) 公司员工信息包括工号、身份证号、姓名、性别和一个手机号，工号唯一标识每一位员工。员工分为代购员和配送员。

(2) 商品信息包括商品条码、商品名称、所在超市名称、采购价格、销售价格和商品介绍，系统内部用商品条码唯一标识每种商品。一种商品只在一家超市代购。

(3) 顾客信息包括顾客真实姓名、身份证号(缴税用)、一个手机号和一个收货地址，系统自动生成唯一的顾客编号。

(4) 托运公司信息包括托运公司名称、电话和地址，系统自动生成唯一的托运公司编号。

(5) 顾客登录系统后，可以下订单购买商品；订单支付成功后，系统记录唯一的支付凭证编号。顾客需要在订单里指定运送方式(空运或海运)。

(6) 代购员根据顾客的订单在超市采购对应商品，一个订单所含的多个商品可能由多名代购员从不同超市采购。

(7) 采购完的商品交由配送员根据顾客订单组合装箱，然后交给托运公司运送。托运公司按顾客订单核对商品名称和数量，然后按顾客的地址进行运送。

根据以上的相关描述，可得到如图 3.15 所示的 E-R 图。其中：

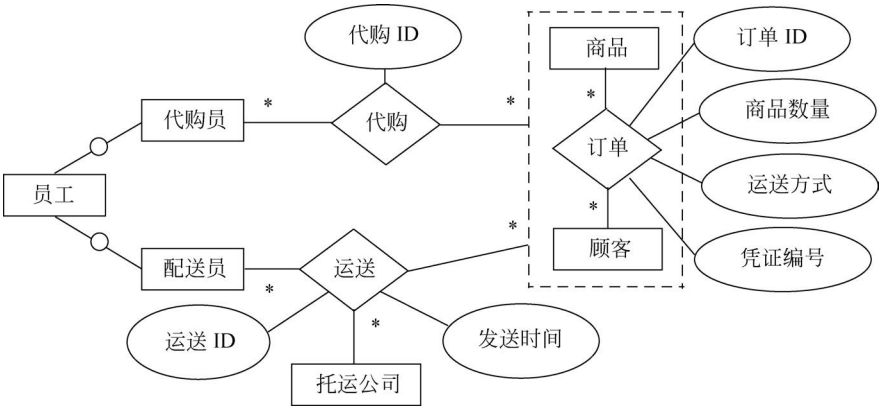

图 3.15　组合实体集 E-R 图示例

(1) 代购员实体集、配送员实体集与员工实体集之间是一种递归联系，即代购员和配送员都是员工，换句话说，只要在员工实体集中增加一个工种属性或标识码特征不同就可以区分他们是代购员还是配送员，所以不需要再为代购员和配送员单独设置实体集。为便于理解，图 3.16 从左至右表示了递归联系表示形式的 E-R 图的简化表示过程。

(2) “商品”“顾客”及其联系“订单”三者作为一个整体(组合实体集)，一是与“代购员”构成多对多联系；二是与“配送员”构成多对多联系；三是与“托运公司”构成多对多联系。

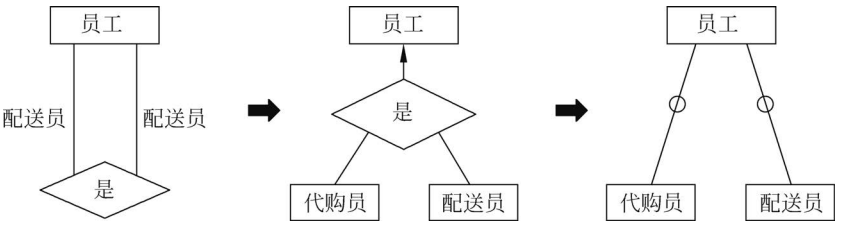

图 3.16　递归联系 E-R 图的简化过程

(3) 对该 E-R 图的进一步理解,详见例 3.12。

3.3.5　基于 E-R 图的概念结构设计步骤和方法

用 E-R 图描述的概念结构的设计分为 3 个步骤:第 1 步是设计分 E-R 图,第 2 步是把各分 E-R 图合并成总体 E-R 图,第 3 步是对总体 E-R 图进行优化。

1. 分 E-R 图的设计

一个数据库应用系统往往涉及多个部门,一个部门通常包括多个数据库用户,不同用户对数据库具有不同的数据观点,因而不同用户对数据库数据的需求也不相同。分 E-R 图的设计是从不同用户或用户组的数据观点出发,将数据库应用系统划分成多个不同的局部应用,通过确定不同用户组所属部门管理业务的数据描述及数据之间的联系,设计出符合不同用户组或不同部门数据管理需求的局部概念结构。

例如,一个大学的教学管理数据库应用系统至少包括教务管理、研究生管理、科研管理、后勤管理等不同部门。由包括“培养”“计划”“招生”“学科建设”用户组组成的研究生管理,就需要设计一个分 E-R 图。

2. 总体 E-R 图的设计

总体 E-R 图的设计指将设计好的各分 E-R 图进行集成,最终形成一个完整的、能支持各局部概念结构的数据库概念结构的过程。由于面向各局部应用的分 E-R 图可能是由不同的设计人员设计的,因此在将各分 E-R 图进行综合集成时,必定存在不一致的地方,甚至还可能存在有矛盾的地方。即使整个系统的各分 E-R 图由同一个人设计,由于系统的复杂性和面向某一局部应用的设计时对全局应用考虑不周,仍可能存在一定的不一致或矛盾。所以,总体 E-R 图的设计关键是消除各分 E-R 图之间存在的不一致和矛盾之处。

1) 命名冲突的消除

命名冲突包括同名异义和异名同义两种情况。

同名异义指在不同的局部应用中的意义不同的对象(实体集、联系集或属性)采用了相同的名称。例如,一个大学的教学信息管理数据库应用系统设计中,如果在教务管理子系统和研究生管理子系统中都采用“学生”对本科生和研究生命名,就属于同名异义的情况。因为在教务管理子系统中的“学生”指本科生,而在研究生管理子系统中的“学生”指硕士研究生或博士研究生。

异名同义指同一意义的对象在不同的局部应用中采用了不同的名称。

消除属性命名冲突的方法是通过讨论和协商解决,或通过行政手段解决。对实体集和联系集的命名冲突要通过认真分析研究,给出合理的解决办法。

2）属性特征冲突的消除

属性特征冲突指同一意义的属性在不同局部应用的分 E-R 图中采用了不同的数据类型、不同的数据长度、不同的数据取值范围、不同的度量单位等而产生的冲突。最常见的现象是某些局部应用采用整数表示人员的年龄，而另一些局部应用则以出生日期表示人员的年龄。

消除属性特征冲突的方法是在照顾全局应用和遵守惯例的基础上协商解决。例如在对人员年龄的表示上，不仅目前习惯上用出生日期表示，而且用出生日期表示后无须每年修改，所以应统一改为用出生日期表示。

3）结构冲突的消除

结构冲突指同一意义的对象（实体集、联系集或属性）在不同局部应用的分 E-R 图中采用了不同的结构特征。一般包括以下 3 种情况。

（1）同一实体集在不同局部应用的分 E-R 图中所包含的属性不一致，主要指属性个数不同，有时也要考虑各属性排列顺序的不一致。

消除这类冲突的办法是综合不同局部应用对该实体集各属性的观察角度和应用需求，进行必要的归并、取舍和调整使其一致。

（2）表示同一意义的实体集间的联系在不同局部应用的分 E-R 图中采用了不同的联系类型。例如，实体 E1 和 E2 在一个局部应用的分 E-R 图中是一对多联系，而在另一个局部应用的分 E-R 图中是多对多联系。

消除这类冲突的办法是分析不同局部应用对该联系的观察角度和应用语义，进行必要的综合或调整，以便使其一致。

（3）表示同一意义的对象（实体集、属性）在不同局部应用的分 E-R 图中具有不同的抽象。例如教职工的配偶在某一局部应用中抽象为实体，而在另一个局部应用中却看成（教职工的）属性。

消除这类冲突的办法是从实体与属性的确定原则和从不同局部应用的观察角度与应用语义这两方面的综合上，考虑将该对象确定为实体还是属性。

3. 总体 E-R 图的优化

在设计面向各局部应用的分 E-R 图时，可能对全局应用考虑不周，因而会使得到的总体 E-R 图存在冗余数据（可由基本数据导出的数据）和冗余联系（可由实体间的其他联系导出的联系），进而就会在由 E-R 图得到的不同关系模式之间产生冗余信息，导致在不同的关系模式中存在冗余属性（字段）。因而会破坏数据库的完整性，给数据库的维护带来困难，所以必须对集成后的总 E-R 图进行优化。

【例 3.7】 在图 3.17 的 E-R 图中，就课程实体集来说，应该有课程号、课程名、学时、学分 4 个属性；就学生学习某一门课程来说，学习联系集应该有学习该课程的学生的学号、标识该课程的课程号、学生学习该课程的分数、该课程的学分 4 个属性。但在两个关系模式：

- 课程关系模式：C(C#,CNAME,CLASSH,C_NUMBER)
- 学习关系模式：SC(S#,C#,GRADE, C_NUMBER)

中同时存在学分（C_NUMBER）属性，显然是一种属性冗余的情况。分析可知，某个学生学习某门课程的学分可从课程的学分属性中导出，所以应当从图 3.17 的学习联系中消去学分属性，由此可得到没有冗余属性的关系模式：

- 课程关系模式：C(C#,CNAME,CLASSH,C_NUMBER)
- 学习关系模式：SC(S#,C#,GRADE)

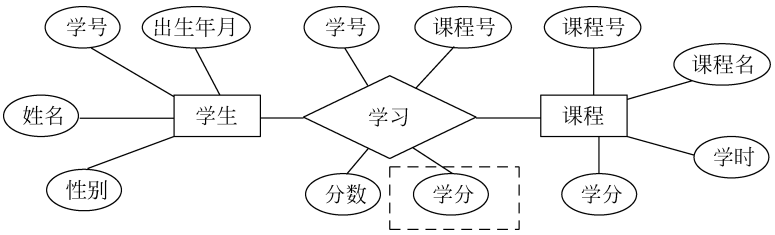

图 3.17　消除 E-R 图中的冗余信息示例

3.4　数据库逻辑结构设计

逻辑结构是一种由具体的 DBMS 支持的数据模型。关系数据库逻辑结构设计阶段的主要任务，就是把用 E-R 模型表示的数据库的概念结构转换成关系数据库管理系统所支持的逻辑数据模式；利用关系数据库的规范化理论对这组关系模式进行规范化设计；并根据数据库的完整性和一致性要求以及系统查询效率要求，对得到的关系模式进行必要的优化处理，从而得出满足所有数据要求的关系数据模型，即数据库的逻辑模式。

数据库的逻辑结构设计一般分成以下 3 步。

(1) 将由 E-R 图表示的概念结构转换成关系模型。

(2) 利用规范化理论对关系模型进行规范化设计和处理。

(3) 对关系模型进行优化处理。

3.4.1　E-R 图表示的概念结构向关系模型的转换

由 E-R 图表示的概念结构向关系模型的转换分为“多对多联系向关系模型的转换”“一对多联系向关系模型转换”“一对一联系向关系模型转换”3 种情况。

1. 多对多联系向关系模型的转换规则

(1) 把位于联系集两端的两个实体集分别转换成一个单独的关系模式，关系模式的属性用与之对应的实体集的属性表示，关系模式的主键用与之对应的实体集的标识码表示。

(2) 将联系集转换成一个关系模式，关系模式的属性用该联系集联系的两个实体集的标识码属性和该联系集的私有属性表示，关系模式的主键用该联系集联系的两个实体集的标识码表示。

【例 3.8】 按照上述转换规则，可将图 3.9 中的 3 组 * ： * 联系转换成如图 3.18 所示的 7 个关系模型。

2. 一对多联系向关系模型的转换

1) 常规的一对多联系向关系模型的转换规则

(1) 联系集不需要转换成单独的关系模式。

(2) 把位于联系集两端的两个实体集分别转换成一个单独的关系模式，关系模式的属性用与之对应的实体集的属性表示，关系模式的主键用与之对应的实体集的标识码表示。

(3) 把位于 1 端实体集的标识码属性和联系集的私有属性添加到位于多端的关系模式中。

学生关系模式：S（S#，SNAME，SSEX，SBIRTHIN，PLACEOFB）
专业关系模式：SS（SCODE#，SSNAME）
课程关系模式：C（C#，CNAME，CLASSH）
设置关系模式：CS（SCODE#，C#）
学习关系模式：SC（S#，C#，GRADE）
教师关系模式：T（T#，TNAME，TSEX，TBIRTHIN，TITLEOF，TRSECTION，TEL）
讲授关系模式：TEACH（T#，C#）

图 3.18　图 3.9 的 E-R 图中的 * ： * 联系的关系模式

【例 3.9】 在图 3.9 中，一对多联系集“归属”不再设置对应的关系。将位于 1 端的“专业”实体集转换成一个关系模式，将位于 N 端的“学生”实体集转换成一个关系模式，并将位于 1 端的“专业”实体集的标识码“专业代码”和联系集“归属”的非标识码属性“班级”加入学生关系模式中。从而得到对应图 3.9 中的 1 ： N 联系的关系模式，如图 3.19 所示。显然，学生关系模式应该选用图 3.19 中的格式。

学生关系模式：S（S#，SNAME，SSEX，SBIRTHIN，PLACEOFB，SCODE#，CLASS）
专业关系模式：SS（SCODE#，SSNAME）

图 3.19　图 3.9 的 E-R 图中的 1 ： N 联系的关系模式

2）非常规的一对多联系向关系模型的转换规则

（1）联系集不需要转换成单独的关系模式。

（2）把位于联系集两端的两个实体集分别转换成一个单独的关系模式，关系模式的属性用与之对应的实体集的属性表示，关系模式的主键用与之对应的实体集的标识码表示。

（3）把联系集看成一个属性，添加到位于 *（多）端的关系模式中。

【例 3.10】 在图 3.12 的航空公司飞行员和机组人员的 E-R 图中，“飞机”实体集是“一般”意义上的飞机（抽象的飞机），“具体飞机”实体集是由一些具体飞机组成的实体集，“飞机”与“具体飞机”之间的联系是 1 ： * 。所以没有必要将联系集 TYPE 转换成一个关系，而只要将“飞机”实体集和“具体飞机”实体集分别转换成一个关系模式，并在“具体飞机”关系模式的属性表中（仅有序列号一个属性）再加上 TYPE 属性。

3. 一对一联系向关系模型的转换

当两个实体集间的联系为常规的 1 ： 1 联系时，联系集不再转换成单独的关系模式，两个实体集的转换策略如下。

（1）联系集不需要转换成单独的关系模式。

（2）把位于联系集两端的两个实体集分别转换成一个单独的关系模式，关系模式的属性用与之对应的实体集的属性表示，关系模式的主键用与之对应的实体集的标识码表示。

（3）在转换成的两个关系模式中的任意一个关系模式的属性中加入另一个关系模式的主键属性和联系集的私有属性，这样就得到两组转换成的关系模式。

（4）选择两组关系模式中数据冗余小、信息完整和更符合实际的一组关系模式，作为最终的关系模式。

【例 3.11】 公司实体集与公司总裁实体集之间存在着 1 ： 1 联系，其 E-R 图如图 3.20 所示。

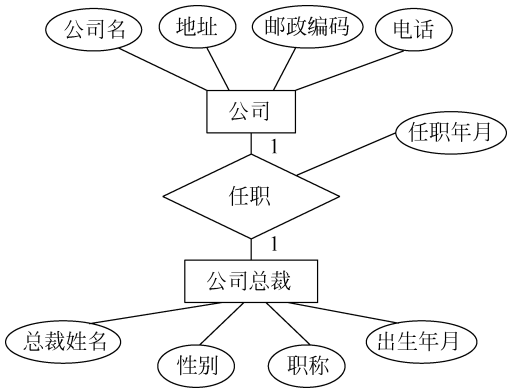

图 3.20　公司总裁与公司的一对一联系

第 1 步：转换

按“一对一联系”的转换规则，将公司实体集和公司总裁实体集分别转换成一个关系模式，并在“公司关系”中加入另一个关系模式(“公司总裁关系”)的总裁姓名(此处认为公司总裁姓名不重名，是“公司总裁关系”的主键)和任职年月(任职联系集的私有属性)。转换成的关系模式即为第一组关系模式。

第一组关系模式：

公司关系(公司名，地址，邮政编码，电话，总裁姓名，任职年月)
公司总裁关系(总裁姓名，性别，出生年月，职称)

同理，也可以转换成如下的第二组关系模式。

第二组关系模式：

公司关系(公司名，地址，邮政编码，电话)
公司总裁关系(总裁姓名，性别，出生年月，职称，任职年月，公司名)

第 2 步：优化选择

下面对转换成的两组关系模式进行分析，最终选出数据冗余小、信息完整和更符合实际的一组关系模式。

从本例来说，若第一组关系模式中的“公司关系”的主键为{公司名}，每当公司更换总裁时，就要修改“公司关系”中的“总裁姓名”和“任职年月”两个属性的值，且“公司总裁关系”中的记录不能反映出历届总裁的“任职年月”信息，所以一旦“公司关系”中的“总裁姓名”和“任职年月”两个属性的值被修改为现任总裁信息时，历任总裁的任职信息就丢失了。即使将“公司关系”的主键改为{公司名，任职年月}，该关系可以保存历任总裁的信息；但由于“总裁姓名”和“任职年月”不是“公司关系”的固有和自然属性，所以也有点不太符合实际。

而在第二组关系模式中的“公司关系”中，记录的都是公司的自然信息，与谁任总裁无关，且其“公司总裁关系”中的记录可反映出历届总裁及其任职时间。所以，从信息完整性和更符合实际角度出发，最终应该选择第二组关系模式作为本例的正确答案。即，本例的转换结果是：

公司关系模式(公司名，地址，邮政编码，电话)
公司总裁关系模式(总裁姓名，性别，出生年月，职称，任职年月，公司名)

4. 具有组合实体集的联系集向关系模型转换的方法

分析图 3.14 可知，在进行 E-R 图向关系模式的转换中，组合实体集内的实体集和联系

集已经按其联系类型(一般都是 * ： * 联系)被转换成相应的关系模式了。设置“客户-订购-项目”组合实体集的目的主要是强调联系集“担保”的存在。

具有组合实体集的联系集向关系模型转换的规则如下。

(1) 把位于联系集某一端的实体集按照前述规则转换成一个关系模式；把位于联系集某一端的组合实体集按照前述规则转换成若干关系模式。

(2) 如果是一对一联系或一对多联系，联系集不需要转换成一个单独的关系模式。

(3) 如果是多对多联系，把该联系集转换成一个关系模式，关系模式的属性由该联系集的私有属性、被联系的实体集的标识码属性，以及来自组合实体集中的联系集的全部或部分主码属性组成。

下面以例 3.6 的 E-R 图向关系模式的转化为例来进一步说明。

【例 3.12】 例 3.6 中某海外代购公司信息管理数据库的 E-R 图向关系模式的转换。

根据例 3.6 的描述和图 3.15 的 E-R 图分析可知：

(1) 对多对多联系“商品-订单-顾客”的转换。

商品(商品条码,商品名称,所在超市名称,采购价格,销售价格,商品介绍)
顾客(顾客编号,顾客姓名,身份证号,手机号,收货地址)
订单(订单 ID,顾客编号,商品条码,商品数量,运送方式,支付凭证编号)

(2) 对多对多联系“代购员(员工)-代购-顾客的商品订单”的转换。

员工(工号,身份证号,姓名,性别,手机号)
代购(代购 ID,(代购员)工号,订单 ID)

其中，用“工号”的编排特征区分其代购员的身份。

(3) 对多对多联系“配送员(员工)-运送-托运公司”的转换，以及多对多联系“托运公司-运送-顾客的商品订单”的转换。

托运公司(托运公司编号,托运公司名称,电话,地址)
运送(运送 ID,(配送员)工号,托运公司编号,订单 ID,发运时间)

“员工”关系模式如(2)中所示。同理，用“工号”的编排特征区分其配送员的身份。

5. E-R 图向关系模型转换中的相关问题

通过以上 4 类转换规则及转换实例说明，还需要综合考虑以下问题。

(1) 一般情况下，E-R 图中的多对多联系集的主码是由被联系集联系的两个实体集的主码组成，但在某些应用中，联系集自身的 ID 号码单独也可以构成其主码。例 3.6 和例 3.12 中的转换实例就说明了这一点。

(2) 在 1 ： N 联系的转换规则中，“具体意义上的联系集”联系的实体集一个是某一实体集，另一个是该实体集中的某些具体的个体组成的特殊实体集，例如由一些具有不同序列号的个体飞机组成的“具体飞机”实体集。这种实体集与实体集之间的联系一般是名为“类型是”的特殊联系，且将一些具有特殊特性的个体组成“具体”的实体集是为了强调这些个体的特殊性，以便单独对其进行描述和处理。

(3) 对于那些既存在与某个实体集的“一对一联系”，又存在与某个实体集的“多对多联系”的实体集(例如图 3.9 中的“学生”实体集)，应选择属性多的那个关系模式作为其最终的关系模式。例如，在例 3.8 中，转换成的学生关系模式为：

S(S#,SNAME,SSEX,SBIRTHIN,PLACEOFB)

而在例3.9中,转换成的学生关系模式为:

S(S#,SNAME,SSEX,SBIRTHIN,PLACEOFB,SCODE#,CLASS)

所以,最终的学生关系模式应该是后者而不是前者。

(4) 对于那些是"三元联系"的联系集来说,转换成的关系模式中应该包括其可能涉及的所有实体集的标识码属性。例如,在例3.12中"运送"联系是三元联系,所以运送关系模式除了它的私有属性"运送ID"和"发送时间"外,还包括员工实体集的(配送员)工号、托运公司实体集的托运公司编号、顾客-订单-商品组合实体集的订单ID。"运送"联系的最终关系模式为:

运送(运送ID,(配送员)工号,托运公司编号,订单ID,发运时间)

(5) 在进一步的关系模式优化中,可能涉及对某些关系模式的进一步简化,一种最典型的情况是:若某个联系集只包括与它相联系的两个实体集的主码属性而无自己独有的属性,则由该联系集所转换成的关系模式有时可以删除掉。

3.4.2 关系数据库模式的规范化设计及优化

1. 关系数据库模式的规范化设计

经过数据库逻辑结构设计阶段后,就会得到用于描述用户组织数据管理需求的一组关系模式。关系模式的合理与否直接与关系数据库的查询性能有关,因此需要按照某种规则或标准来衡量得到的这组关系模式的合理性,即判断其是不是规范化的关系模式。

具体来说,所谓关系模式的规范化设计,就是按照不同的范式(标准)对关系模式进行分解,用一组等价的关系子模式来代替原有的关系模式,即通过将一个关系模式不断地分解成多个关系子模式和建立模式之间的关联来消除数据依赖中不合理的部分,最终实现一个关系仅描述一个实体或者实体间的一种联系的目的。

目前遵循的主要范式包括1NF、2NF、3NF、BCNF等几种,3NF、BCNF应用最为广泛,一般推荐以3NF为标准。通过对关系模式的进一步规范化设计,可以有效地消除数据冗余,理顺数据的从属关系,保持数据库的完整性,增强数据库的稳定性、伸缩性、适应性。

这里只需要对关系模式规范化的目的和用途有初步了解,有关关系模式规范化设计的理论和方法将在第5章详细介绍。

2. 关系模式的优化

关系模式的规范化程度越高,关系模式分解得就越彻底,也就是说关系模式分解得就越"碎",这样在实际应用中必然会出现过多的连接运算而降低系统的询效效能。

另外,数据库逻辑设计的结果有时也不是唯一的,为了进一步提高数据库应用系统的性能,还应该结合应用需求适当地修改、调整数据模式的结构。

关系数据模式的优化就是通过对照需求分析阶段得到的信息处理需求,进一步分析得到的关系模式是否符合处理要求和系统查询效率,并从最常用的查询要求中找到最常需要进行的连接运算及相关的关系模式,进而从查询效率角度出发对某些模式进行合并或分解。

举例来说,假设下面是一个通过概念结构设计和逻辑结构设计得到的关系模式:

SSC(SCODE#,SSNAME,C#)

进一步分析可知，虽然一个学校的专业数相对于开设的课程数来说要少很多，但如果按照关系模式 SSC(SCODE#,SSNAME,C#)存储专业名称和开设的课程(课程号)，专业名称就会出现大量的冗余存储情况。因此，从减少冗余等角度考虑，应该将关系模式 SSC 分解成两个关系模式：

```
SS(SCODE#,SSNAME)
CS(SCODE#,C#)
```

当然，某些情况下也会涉及合并关系模式的情况。

通过本阶段的数据库逻辑结构设计，就可得到所要设计的数据库应用系统的数据库逻辑结构。

3.5 数据库物理结构设计

数据库的物理结构指数据库在物理存储设备上的文件组织方式、存储结构和存取方法。因此，数据库物理结构设计就是要建立数据库文件，并根据应用要求和所选用的 DBMS 的数据库存储结构及存储方法，为逻辑结构设计阶段设计好的逻辑数据库模型选择和设计其在物理存储设备上的物理存储结构和存取方法。基于 SQL Server 数据库管理系统开发数据库应用系统时，数据库的物理结构设计内容主要包括：创建数据库数据文件和日志文件，在数据库数据文件中创建二维表形式的关系模式(以下简称数据表、关系表或表，是关系模式在关系数据库中的存在形式)，为数据表创建索引，对数据库的物理结构进行评价等。

要完成数据库的物理设计工作，设计人员必须了解所选用的关系 DBMS 的数据库文件的存储组织方式；了解数据库(表)中数据的存储结构和存取方法；了解数据库应用系统对处理频率和响应时间的要求；了解外存设备的特性等。一些大型的关系 DBMS 系统软件，例如 Oracle 数据库等，会涉及复杂的数据库物理存储组织等问题；但许多关系 DBMS 系统软件屏蔽了大量的内部物理结构，留给用户参与设计的主要是表和索引的建立等。

3.5.1 SQL Server 的数据文件及存储结构

本节以一个实际的商用数据管理系统——SQL Server 为例，介绍物理数据库设计的相关问题。首先介绍 SQL Server 的数据库文件及存储结构，然后介绍 SQL Server 的数据库，最后介绍 SQL Server 数据库的数据文件和日志文件的创建方法。

1. SQL Server 的数据库文件

SQL Server 为每个用户组织的数据库应用系统都建有一组相应的数据库文件，这组数据库文件包括数据文件和日志文件两种。

1) 数据文件

数据文件(Data File)指存储用户的数据库对象(如表、视图、索引等；详见 3.5.2 节)和用户数据的文件。对于一个数据量不是很大的数据库应用系统，有且仅有一个数据文件，即主数据文件。对于一个数据量很大的大型数据库应用系统，可以有一个主数据文件和多个次数据文件，主数据文件用于存储数据库的启动信息和部分数据，并指向数据库中的其他次数据文件。主数据文件的扩展名是.mdf；次数据文件的扩展名是.ndf。

2) 日志文件

日志文件(Log File)是一个系统文件,其功能是用于记录用户在数据库中插入新数据、删除原数据、修改原数据时的各种相关的更新信息,这些更新信息也称为恢复数据库所需的日志信息。日志文件的扩展名是.ldf。

数据库的更新信息一般包括:从事更新的用户标识信息;开始更新时间,结束更新时间;当为修改操作时,未修改前的数据和修改后的数据;当为插入操作时,新插入的数据;当为删除操作时,删除前的原数据等。这样当数据库运行过程中由于某种原因造成程序的中止运行、磁盘损坏、系统掉电等使数据库中信息不安全和不一致的情况时,就可在系统恢复机制的控制下,利用日志文件中的信息实现对数据库中部分数据的恢复。

每个数据库至少应有一个日志文件。对于一个大型或较大型数据库来说,为了保障充足的日志信息、比较长的恢复时段和更快的恢复速度,每个数据库应至少配置两个或3个日志文件。数据库以连续方式写入一个日志文件,直到写满为止,然后再从第二个日志文件开始写起。当最后一个日志文件写满后,数据库系统会用新的更新信息记录覆盖(重写)第一个日志文件的内容。即又在第一个日志文件开始写数据库的更新信息记录,并如此循环。

SQL Server的数据库中的数据文件和日志文件组合在一起,形成了SQL Server数据库的物理表象。

2. SQL Server数据库的存储结构

SQL Server数据库文件的存储结构如图3.21所示。SQL Server的数据库文件由数据文件和日志文件组成;每个SQL Server文件由多个盘区(extent)组成,每个盘区由8个连续的页面组成。SQL Server利用盘区和页面数据结构给数据库对象分配存储空间。

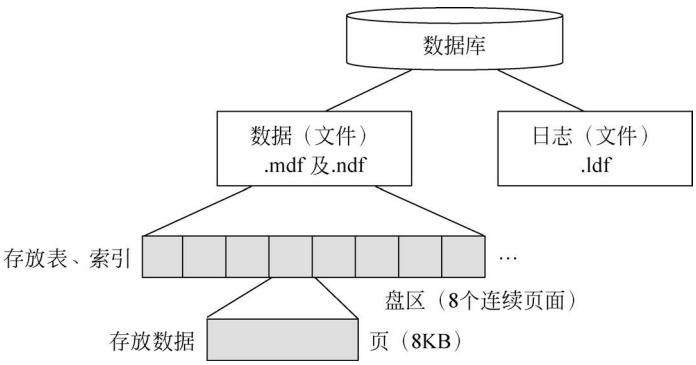

图3.21 SQL Server数据库的存储结构

1) 盘区

每个盘区是由8个连续页组成的数据结构,大小为8KB×8=64KB。当创建一个数据库对象(例如,一个表、一个索引)时,SQL Server自动以盘区为单位给它们分配存储空间。每个盘区只能包含一个数据库对象,每个数据对象可以占用多个盘区。

2) 页面

每个页面大小为8KB(8192字节),页面的开始部分是96字节的页头信息,用于存储该页的页编号、页类型、该页可用的空间数量、占用该页的数据库对象的对象标识等系统信息;剩余的8096字节用于存放该页的数据库对象的数据信息。

SQL Server 2012的页面分为数据页、索引页、文本页、图像页等8种。数据是按行存储

在数据页中的，数据页中行的最大长度为 8096 字节，行不能跨页存储。

3.5.2 SQL Server 2012 的数据库

数据库是以数据库文件形式存在的，数据库文件是数据库的物理表象。SQL Server 2012 的数据库分为系统数据库和用户数据库两类。

1. 系统数据库

系统数据库指存储 SQL Server 的有关系统信息的数据库。该类数据库存储的是 SQL Server 专用的、用于管理自身和管理用户数据库的数据。安装 SQL Server 2012 时，系统会自动创建 4 个系统数据库，分别是 master、model、tempdb 和 msdb。

(1) master 数据库。master 数据库存储的信息包括所有可用的数据库，及为每一个数据库分配的空间、使用中的进程、用户账户、活动的锁、系统错误等信息和系统存储过程等。

master 数据库和它的事务日志存储在文件 master. mdf 和 mastlog. ldf 中，由于这个数据库是系统用数据库，所以不允许用户修改它。

(2) msdb 数据库。msdb 数据库由 SQL Server Agent 服务使用，用于管理警报和任务。它还存储 SQL Server 管理数据库的每一次备份和恢复的历史信息。msdb 数据库存储在 msdbdata. mdf 中，其事务日志存储在 msdblog. ldf 中。

(3) model 数据库。model 数据库的作用是为新的数据库充当模板，用户新建的数据库是 model 的副本。用户可以对 model 数据库进行修改，包括添加用户定义的数据类型、规则、默认值和存储过程，对 model 数据库的任何修改都会自动反映到新建的数据库中。model 数据库存储在 model. mdf 中，其事务日志存储在 modellog. ldf 中。

(4) tempdb 数据库。tempdb 数据库是为所有 SQL Server 数据库和数据库用户共享的数据库。它用于存储临时信息，任何因用户行为而创建的临时表都会在该用户与 SQL Server 断开连接时删除；所有在 tempdb 中创建的临时表都会在 SQL Server 停止和重启时删除。tempdb 数据库存储在 tempdb. mdf 和 templog. ldf 中。

2. 用户数据库

用户数据库指存储用户的数据库对象、存储用户的数据和存储用户更新数据库时的日志信息的数据库。数据库是以数据库文件形式存在的，数据库文件是数据库的物理表象。用户的数据库文件包括数据文件和日志文件。如前所述，存储用户的数据库对象和数据的数据库文件称为数据文件，例如 JXGL. mdf。存储用户更新数据库时更新前后的相关数据和更新时间等信息的数据库文件称为日志文件，例如 JXGL_log. ldf。

在 SQL Server 中，用户可通过 SQL Server Management Studio 或使用 SQL 语言创建自己的数据库(文件)。用户创建了自己的数据库后，可在创建的数据库中进一步创建数据库对象。数据库对象是数据库的重要组成部分，也是数据库编程的主要对象。SQL Server 2012 中具有代表性的数据库对象主要如下所述。

(1) 表。即关系表，是关系模式在数据库中的物理表象，用于存放相应关系模式的数据。

(2) 视图。是一种虚拟的关系表，也称虚表；是一种用于查看数据库中一个或多个表中的部分字段的数据的一种方法。

(3) 索引。是对数据库表中一列或多列的值进行排序的一种结构，使用索引可加快访问数据库表中特定信息的速度，将在 3.5.5 节介绍。

(4) 存储过程。是一组预先编译好的能实现特定数据操作功能的SQL语句集,用于简化相关或重复的数据库操作,将在第6章介绍。

(5) 函数。将一个或多个T-SQL命令语句创建成函数,以便能够重复使用这些函数,将在第6章介绍。

(6) 触发器。是一种特殊的存储过程,在用户向表中插入、更新或删除数据时自动执行;用于进行数据一致性验证等,将在第9章介绍。

(7) 用户。数据库的用户,即允许访问数据库的用户列表。

3.5.3 使用SQL Server Management Studio创建数据库的方法

本书以SQL Server 2012数据库管理系统软件环境为依托,以大学教学信息管理数据库应用系统中的数据库创建为例,说明使用SQL Server Management Studio创建数据库的方法。

【例3.13】 使用SQL Server Management Studio工具,为图1.8和图1.11所示的大学教学信息管理数据库应用系统创建数据库,数据库的名称为JXGL。

(1) 单击"开始→所有程序→Microsoft SQL Server 2012→SQL Server Management Studio"菜单命令,启动SQL Server Management Studio工具。

(2) 在启动后弹出的"连接到服务器"对话框(如图1.17所示)中,单击"连接"按钮,连接数据库服务器。

(3) 在成功启动后的SQL Server Management Studio工具主界面中,打开"对象资源管理器"(有时"对象资源管理器"已经自动打开,如图1.18所示)。在"对象资源管理器"中右击"数据库"对象,在弹出的快捷菜单上选择"新建数据库"。系统将显示"新建数据库"对话框,如图3.22所示。

图3.22 "新建数据库"对话框

(4) 单击"新建数据库"对话框中的"常规"选项卡,进行数据库属性设置。设置过程如下。

① 在"数据库名称"文本框中输入要建立的数据库的名称"JXGL"。注意,数据库的名称必须遵循SQL Server的命名规范,并且不能与已有的数据库名重复。系统会自动为该数

据库建立两个数据库文件：数据文件 JXGL. mdf 和日志文件 JXGL_log. ldf，默认存储在“安装目录\Microsoft SQL Server\MSSQL1. MSSQLSERVER\MSSQL\DATA”目录下。

② 在“数据库文件”列表中的“逻辑名称”栏可以根据需求修改数据文件和日志文件的逻辑名称。如果确需修改，作为初学者，建议按①的方法在“数据库名称(N)”栏重新输入新的名称。

③ 根据需求可修改数据文件和日志文件的初始大小和自动增长方式。若要修改自动增长方式，则需单击“自动增长”栏中相应文件的[...]按钮，在弹出的“更改 JXGL 的自动增长设置”对话框中进行相应的设置，如图 3.23 所示。建议初学者不要尝试做这一步。

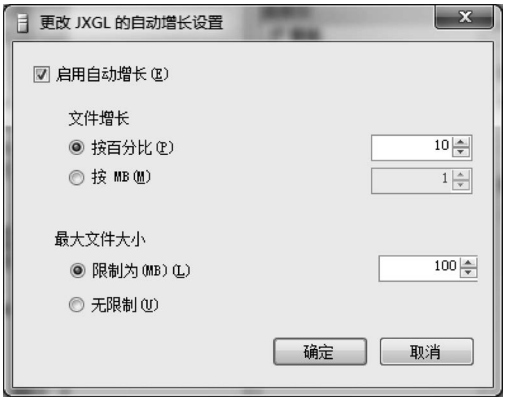

图 3.23 “更改 JXGL 的自动增长设置”对话框

④ 若要修改存储数据库文件的存储路径，先要在某硬盘分区(例如 D 盘)中建好目录，例如 D:\JXGL，然后单击“路径”栏中相应文件的[...]按钮，在弹出的“选择文件夹(S):”对话框中选择建好的存储路径 D:\JXGL，“选择文件夹(S):”中的“所选路径(P):”就设置成 D:\JXGL 了，接着单击“确定”按钮即可。需要说明的是，因为“选择文件夹(S):”对话框中的“所选路径(P):”中的内容只能从路径目录中选择，不能修改，也不接收新输入字符，所以必须先建好存储路径。

⑤ “添加(A)”是指新添加一个次数据文件或次日志文件。作为初学者，建议不要，也没必要尝试做这一步。

(5) 在图 3.22 的窗体界面中，单击“确定”按钮，SQL Server 即完成数据库的创建。此时或在下次启动 SQL Server Management Studio 后，就可在“对象资源管理器”中看到新建的 JXGL 数据库了。

3.5.4 数据表及其创建与修改

数据表是 SQL Server 2012 中最基本的数据库对象。

1. 数据表的概念

数据表即关系表，简称表，是关系数据库中组织数据的基本单位；是关系模式在数据库中的物理表象；是数据库中存储各特定主题数据的地方；也是数据管理应用中从数据库中查询数据的“数据之源”。数据表及该表所属的数据是存储在数据库文件中的(如例 3.13 中的 JXGL. mdf)，当数据库文件多于一个时，每个数据表及其数据只能存储在其中的一个数据库文件中。

2. 数据表的数据类型及约束

下面以图1.11给出的本书教学案例中的课程关系模式

C(C#,CNAME,CLASSH)

为例,说明与该关系模式对应的课程信息表涉及的相关概念。

在创建数据表时,需要给出表名、各属性(字段)名、各属性的数据类型和与该数据类型相适应的数据长度、是否为主键属性、该属性值是否允许空值(null)等。其中,字段类型、字段长度、是否主键属性、是否允许空值等都属于该属性的约束信息。主键属性的属性值约定不能为空值。课程关系表的属性、属性值及约束如表3.2所示。

表3.2 课程关系表及其约束

字段名称	字段类型	字段长度	主键属性	允许空值	字段含义
C#	char	7	是	否	课程号
CNAME	varchar	20	否	否	课程名
CLASSH	int		否	是	学时

对表3.2中的数据说明如下。

(1) char为字符型数据类型,学号字段C#对应的char(7)表示学号的字符长度约定为7个字符的固定长度,当不足7个字符时,系统自动用空白字符补充。

(2) varchar也是字符型数据类型,课程名字段对应的varchar(20)表示课程名的最大长度是20个字符,但当课程名实际长度较短时,不需要补充字符。例如,"数据结构"的实际长度为8个字符,采用varchar数据类型,既考虑到了存储较长的课程名的需要,在存储较短的课程名时也不会浪费存储空间。

(3) int为整型数据类型,长度由系统约定,用户无须指定其长度。

(4) 主键属性为"是"时,表示该属性为主键属性,在创建表时要对该属性设置主键约束( )。

(5) 字段"允许空值"约束为"是"时,在创建表时要将该属性的"允许null值"设置成"是"或"null";字段"允许空值"约束为"否"时,在创建表时要将该属性的"允许null值"设置成"否"或"not null"。

3. 创建数据表的方法

建立SQL Server数据库文件(空数据库)后,接下来就可以在该数据库文件中创建数据表了。在SQL Server中创建表的方法有两种:一种是利用SQL Server Management Studio工具创建表;另一种是利用SQL语句创建表(详见第4章)。下面介绍利用SQL Server Management Studio工具创建表的方法。

【例3.14】 利用表设计器在大学教学信息管理数据库JXGL中创建"课程表"C。

1) 进入"表设计器"对话框

(1) 启动SQL Server Management Studio工具,并连接数据库服务器。在SQL Server Management Studio工具的主界面中的"对象资源管理器"中展开"数据库",进而展开之前创建的JXGL。主界面中的资源管理器部分如图3.24所示。

(2) 右击JXGL下的"表"对象,在弹出的快捷菜单中选择"新建表"命令,如图3.25所示,系统会弹出"表设计器"对话框,如图3.26所示。

图 3.24　展开“数据库”和展开 JXGL

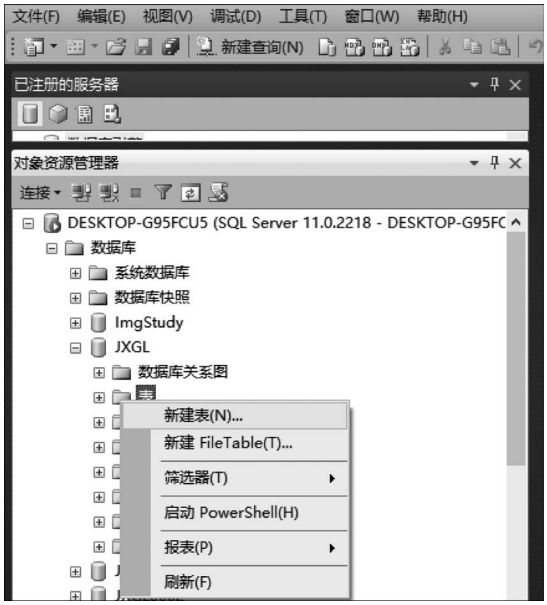

图 3.25　选择“新建表”菜单命令

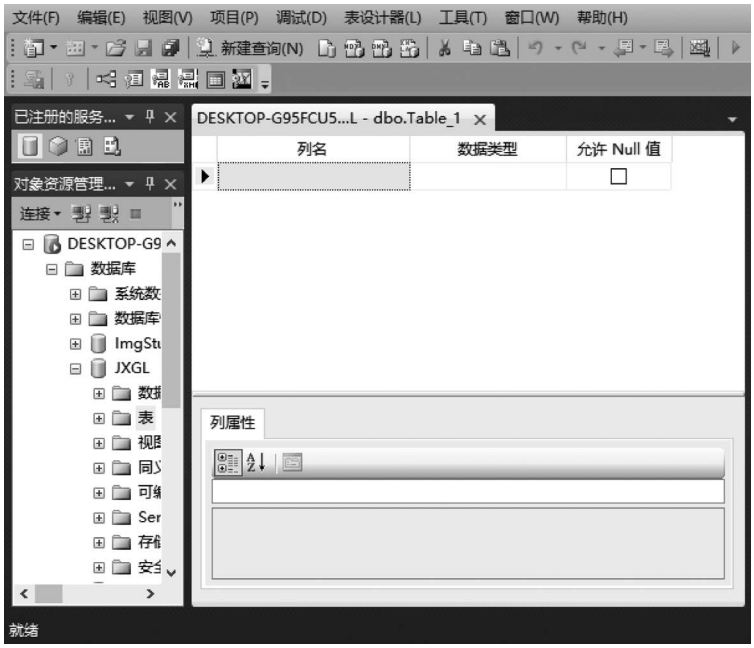

图 3.26　“表设计器”对话框

2）“课程表”结构的建立

在“表设计器”中依次输入已经设计好的“课程表”C 的各属性信息，包括列名、数据类型、是否允许为空值等，如图 3.27 所示。

3）为“课程表”C 创建主键

（1）在“表设计器”中右击 C#字段，在弹出的快捷菜单中选择“设置主键”命令，如图 3.28 所示。

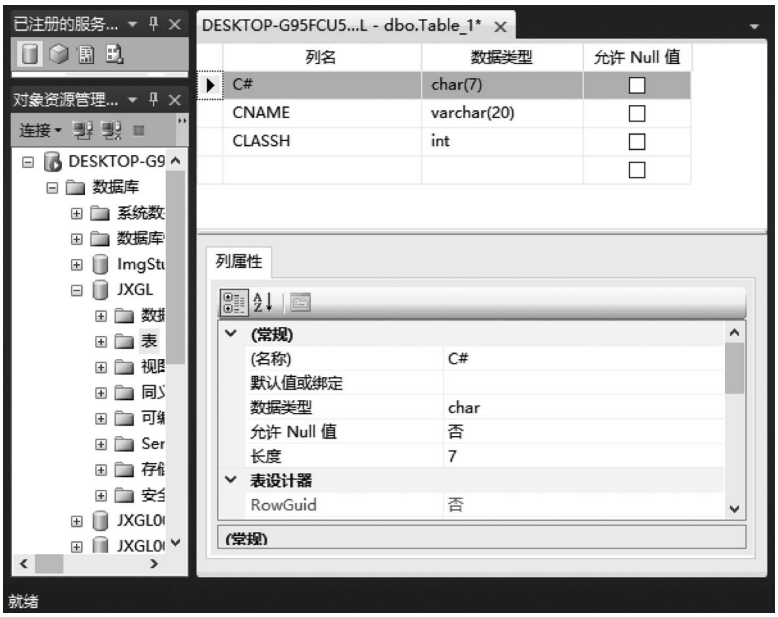

图 3.27　设计好的“课程表”C

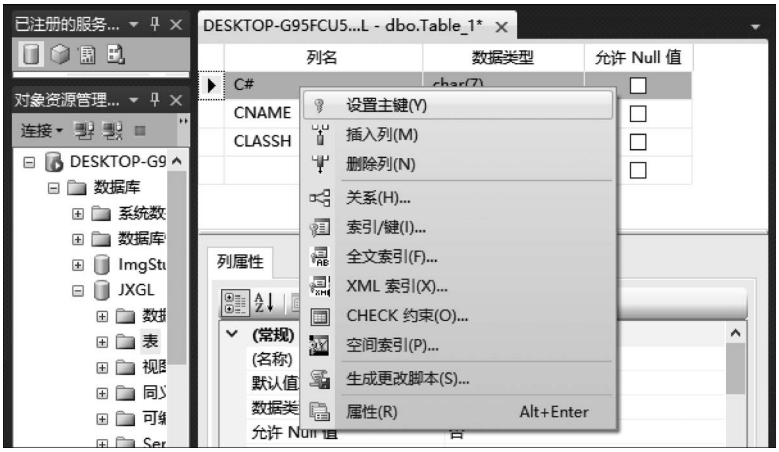

图 3.28　“设置主键”的过程——弹出设置主键快捷菜单

(2) 将课程号字段设置为主键,如图 3.29 所示。

4) 保存创建的表

(1) 在“文件”菜单中选择“保存”菜单命令,弹出“选择名称”对话框,如图 3.30 所示,接着为该表输入名称 C,单击“确定”按钮,完成学生表 C 的创建。

(2) 此时,或在下次启动 SQL Server Management Studio 后,在“对象资源管理器”的 JXGL 数据库下的“表”对象中,用户可以查看到刚刚建立的数据表 C。

4. 表结构的修改方法

创建表之后,由于某种原因需要对表结构、约束或其他列的属性进行修改。对一个已存在的表可以进行的修改操作包括以下几方面。

(1) 更改表名。

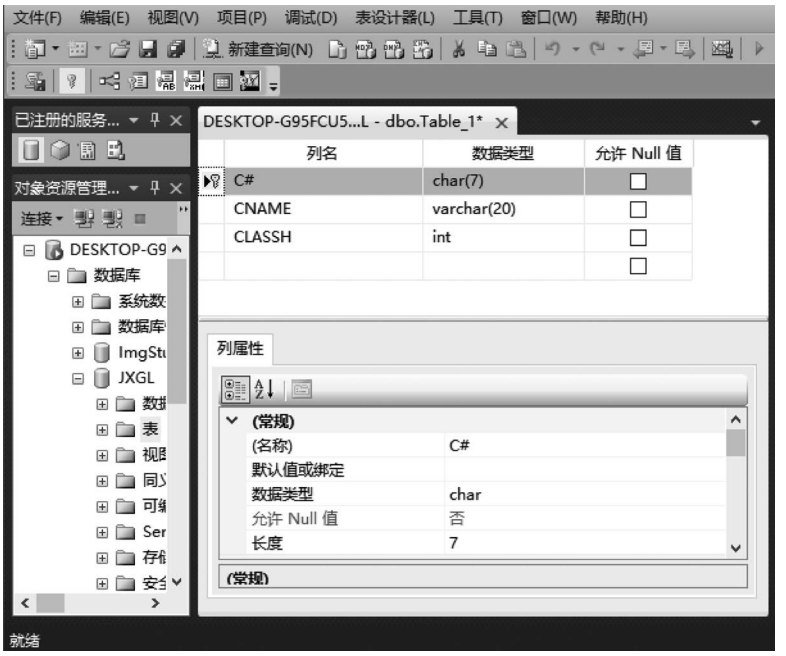

图 3.29 “设置主键”的过程——S# 为主键

图 3.30 “选择名称”对话框

(2) 添加新的列。

(3) 删除已有的列。

(4) 修改已有列的属性(列名、数据类型、长度、默认值以及约束)。

第 1 项修改操作在 SQL Server Management Studio 工具的“对象资源管理器”中实施。只需右击要修改的表,在弹出的快捷菜单中选择“重命名”命令,即可修改表名。重命名一个表将导致引用该表的其他数据库对象(例如存储过程、视图、触发器等)无效,因此要慎重。

后 3 项修改操作均在“表设计器”对话框中进行,右击要修改的表,在弹出的快捷菜单中选择“修改”命令,即可弹出要修改的表,用户可对数据表的各列及其属性进行添加、删除和修改。

5. 表的删除方法

对于数据库中不再需要的表,可以将其删除。另外,考虑到有时修改已经建立好的表也比较麻烦或有一定的限制,也可以将其删除再重新创建。

当要删除一个表时,首先在 SQL Server Management Studio 工具的数据库窗口的“对象资源管理器”中选中要删除的表,然后在要删除的表上右击,在弹出菜单中选择“删除”选项。另一种方法是选择 SQL Server Management Studio 主窗口的“编辑”→“删除”命令。

同修改表一样,当删除一个表时,该表的结构定义、表中的所有数据以及表的索引、触发器、约束等会都从数据库中永久删除,因此执行删除表的操作时要谨慎。

6. 表中数据的插入和更新

创建表的目的就是用表来存储数据和方便数据的管理。下面通过示例介绍表中数据的输入(添加)和更新方法。

1) 添加数据记录

【例 3.15】 向已经建立的课程关系表 C 中输入图 1.8 中的课程关系的数据记录。

(1) 展开 SQL Server Management Studio 左边的"对象资源管理器",在 JXGL 数据库的表目录中右击课程表 C,在弹出的快捷菜单中选择"编辑前 200 行(E)"选项,此时就打开了待输入数据的课程表 C,如图 3.31 所示。

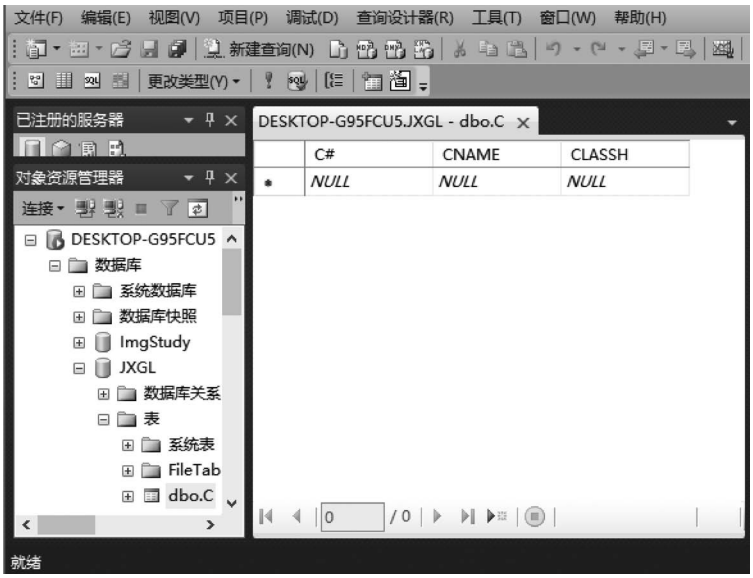

图 3.31 没有数据记录的数据表显示示例

(2) 将光标定位到"行编辑器"中有"*"的一行,即可输入要添加的记录。采用同样方法可完成图 1.8 中的课程关系的所有数据记录的输入,结果如图 3.32 所示。

(3) 所有数据记录输入完毕后,在"文件"菜单中选择"全部保存"命令即可。

2) 修改数据记录

如果之前添加的数据记录出错了或过时了,就需要对其进行修改。修改数据记录步骤如下。

(1) 将光标定位到需要修改的数据记录。

(2) 直接对需要修改的字段进行修改。

(3) 关闭数据表视图,系统自动保存表中数据。

3) 删除数据记录

当之前添加的数据记录不便于修改或不再需要时,就需要将其从表中删除。

【例 3.16】 删除大学教学信息管理数据库 JXGL 中课程关系表 C 中的"通信原理"课程。

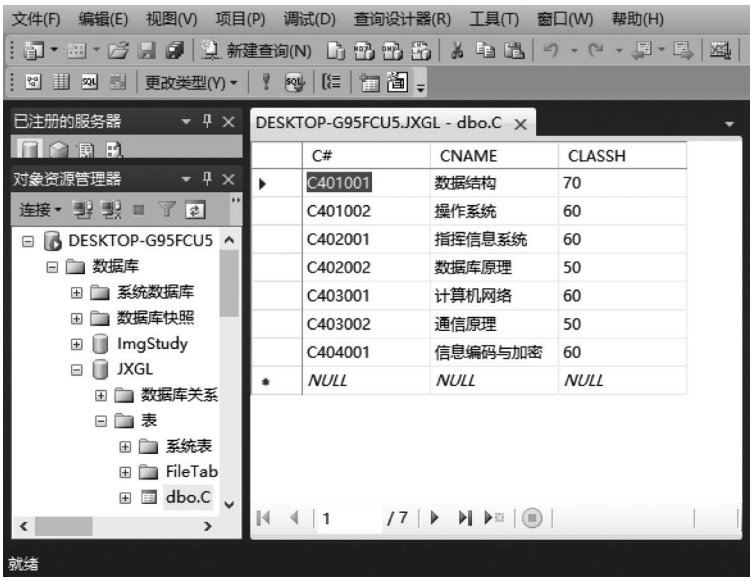

图 3.32　在表 C 中输入数据记录的数据视图

(1) 右击拟删除的数据记录的“行定位器”,弹出快捷菜单,选择“删除”选项,如图 3.33 所示。

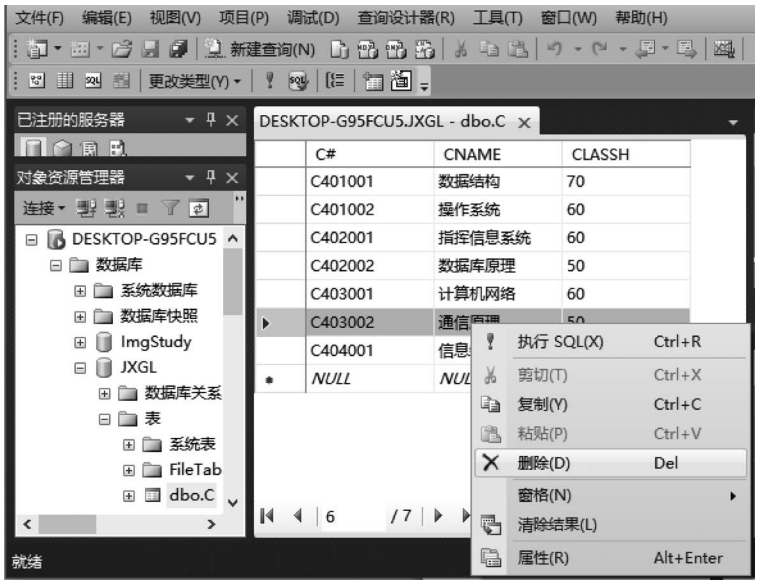

图 3.33　弹出快捷菜单选择“删除”选项

(2) 选择“删除”命令,弹出如图 3.34 所示的确认删除记录提示对话框,单击对话框中的“是”按钮,即可删除该记录。

4) 浏览表中的数据记录

用户通过 SQL Server Management Studio 工具可以方便地浏览数据表的所有记录。

【例 3.17】　浏览大学教学信息管理数据库 JXGL 中课程关系表 C 的所有课程记录。

(1) 启动 SQL Server Management Studio 工具,在“对象资源管理器”中选择“数据库”

对象中的JXGL数据库。

(2) 在JXGL数据库下的“表”对象中右击课程表C,在弹出的快捷菜单中选择“编辑前200行”命令,如图3.35所示。

图3.34 删除记录提示对话框

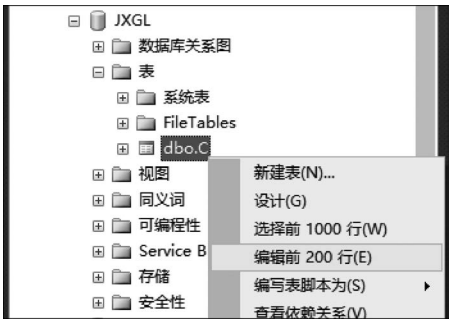

图3.35 表编辑快捷菜单

(3) 单击该选项后,在“数据视图”中将显示课程表C的所有记录,如图3.32所示。

3.5.5 索引技术

关系数据库是以表的形式组织数据的;各关系表中的数据是存储在数据库的数据文件中的;表中数据记录的查询是通过顺序扫描表来实现的。当一个表中的记录数很大(例如,几十万个记录,甚至几百万个记录),极端情况下且查询的内容是表中的最后一个记录时,则需要扫描完整个表后才能找到该记录,显然查询效率比较低。因此,提高表中数据记录的查询速度和减少访问时间,就成为数据库物理结构设计中一个非常重要的问题。目前提高表中数据记录查询速度的主要方法是为表建立索引。

1. 索引的概念

在关系数据库中,索引是一种物理地对数据库表中一列或多列的值进行排序的存储结构,是将一个关键字与它对应的数据记录相关联的过程。所以,一个索引由若干索引项构成,每个索引项至少包含一个关键字和其对应的数据记录在存储器中的位置的信息。索引技术是组织大型数据库以及磁盘文件的重要技术。

设k_i(i=1,2,…,n)为某一关系表中按其某种逻辑顺序排列的主键值,其对应的记录R_{ki}的地址为$A(R_{ki})$,则$(k_i, A(R_{ki}))$称为一个索引项,由多个索引项组成的如图3.36所示的表形式的数据结构称为索引,有时也称索引表。

k_1	k_2	k_3	…	k_n
$A(R_{k1})$	$A(R_{k2})$	$A(R_{k3})$	…	$A(R_{kn})$

图3.36 索引结构

可见,索引的实质就是按照记录的主键值将记录进行分类,并建立主键值到记录位置的地址指针。图3.37是一种学生关系及其索引的图示表示方式。

由图3.37可知,只要给出索引的主键,就可以在索引表中查到相应记录的地址指针,并进而直接找到要查询的记录。由于索引表比它的关系表小得多,所以利用索引查找要比直接在关系表上查找快得多。

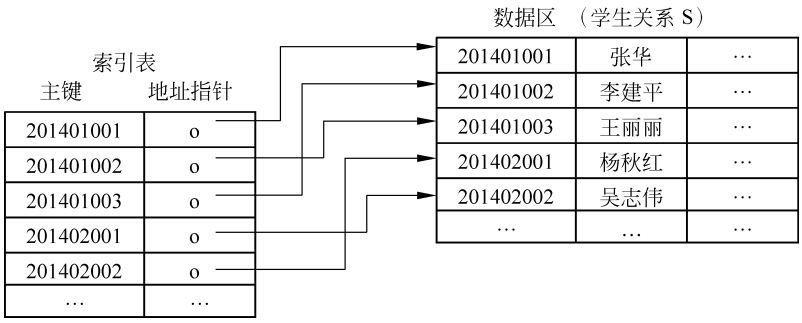

图 3.37　学生关系索引

2. 线性索引

线性索引是一种将索引项集合组织为线性结构的索引方式，或者说是使用单一下标按照顺序向下遍历每一行的索引项来引用表中数据记录的索引项组织方法，所以线性索引也称为索引表。线性索引可分为稠密索引和稀疏索引两种。

1）稠密索引

在稠密索引(Dense Index)方式中，按主键值的排序建立索引项，每个索引项包含一个主键值和一个由该主键值标识的记录的地址指针，所以每个索引项对应一个记录，记录的存放顺序是任意的。由于索引项的个数与记录的个数相等，也就是说索引项较多，所以称为稠密索引。图 3.37 的学生关系索引即是一个稠密索引的例子。

引入索引机制后，向关系表中插入记录，修改关系表中的记录和删除关系中表的记录就要通过索引来实现。对于稠密索引来说，由于数据记录的存放顺序是任意的，所以实现对关系表中记录删、插、改的关键是索引表中主键值的查找问题。由于按主键值排序的稠密索引表相当一个顺序文件，主键值的查找可以用顺序文件的查找方法，即当主键值不大时采用顺序扫描方式；当主键值较大时采用二分查找等查找方式。

稠密索引的优点是查找、更新数据记录方便，存取速度快，记录无须顺序排列。缺点是索引项多，索引表大，空间代价大。

2）稀疏索引

在稀疏索引(Sparse Index)方式中，所有数据记录按主键值顺序存放在若干块中，每个块的最大主键值(即该块最后一个数据记录的主键值)和该块的起始地址组成一个索引项，索引项按主键值顺序排列组成索引表。由于是每个块只有一个索引项，也就是说索引项较少，所以称为稀疏索引。图 3.38 所示是一个稀疏索引的例子。

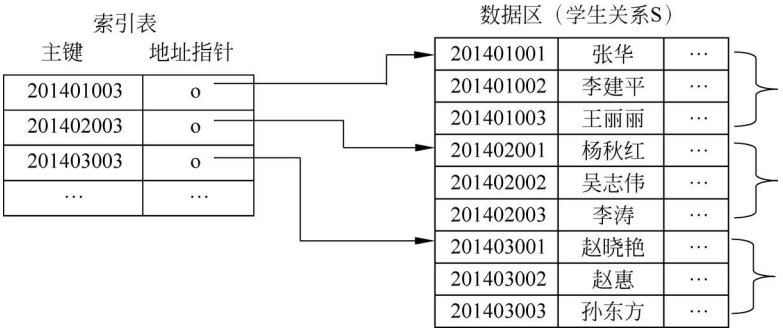

图 3.38　学生关系的稀疏索引方式

稀疏索引由于索引项较少,因而节省存储空间;但由于稀疏索引方式不仅索引表中的主键值是按序存放的,而且各个块中的数据记录也是按主键值的顺序存放的。与此同时,在各个块的存储组织中,对于同一系统来说块的大小一般还是相对固定的。所以实现对关系表中记录的删、插、改,特别是插入操作,是十分麻烦的。在插入操作较多的应用中采用稀疏索引方式是不太适宜的。

3. B－树

在稀疏索引方式中,当索引项很多时,可以将索引分块,建立高一级的索引;进一步,还可以建立更高一级的索引,……,直至最高一级的索引只占一个块。这种多级索引如图3.39所示,是一棵多级索引树,图中假设每个块可以存放3个索引项。

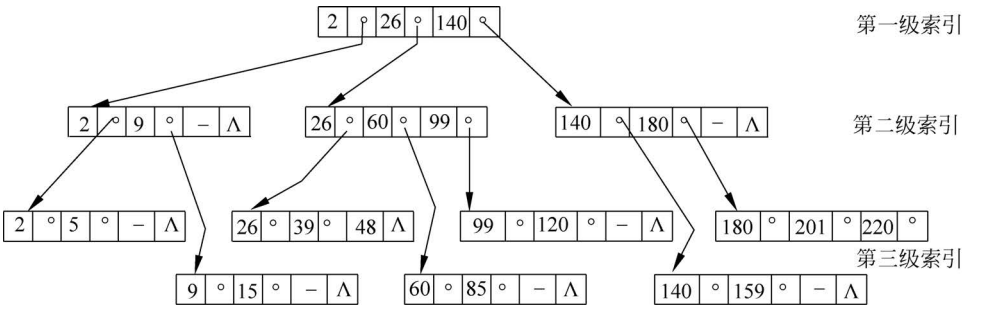

图3.39　多级索引

当在多级索引上进行插入,使得第一级索引增长到一块容纳不下时,就可以再加一级索引,新加的一级索引是原来第一级索引的索引。反之,在多级索引上进行删除操作会减少索引的级数,于是就产生了B－树(Balance－树)的概念。B－树的表述涉及下列一些术语。

(1) 结点。在B－树中,将根结点、叶结点和内结点(B－树中除根结点和叶结点以外的结点)统称为结点。根结点和内结点是存放索引项的存储块,简称索引存储块或索引块。叶结点是存放记录索引项的存储块,简称记录索引块或叶块,每个记录索引项包含关系中一个记录的主键和它的地址指针。

(2) 子树。结点中每个地址指针指向一棵子树,即结点中的每个分支称为一棵子树。

(3) B－树的深度。每棵B－树所包含的层数,包括叶结点,称为B－树的深度。

(4) B－树的阶数。B－树的结点中最多的指针数称为B－树的阶数。

在上述术语的基础上,有如下B－树定义。

满足如下条件的B－树称为一棵m阶B－树(m为不小于3的正整数)。

(1) 根结点或者至少有两个子树,或者本身为叶结点。

(2) 每个结点最多有m棵子树。

(3) 每个内结点至少有⌈m/2⌉棵子树(⌈ ⌉为向上取整符号,例如,⌈3/2⌉=2)。

(4) 从根结点到叶结点的每一条路径长度相等,即树中所有叶结点处于同一层次上。

在此定义基础上同时约定:

(1) 除叶结点之外的所有其他结点的索引块最多可存放m－1个主键值和m个地址指针,其格式为:

p_0	k_1	p_1	k_2	p_2	…	k_{m-1}	p_{m-1}

其中，k_i（$1 \leqslant i \leqslant m-1$）为主键值，$p_i$（$0 \leqslant i \leqslant m-1$）为指向第 i 个子树的地址指针。为了节省空间，每个索引块的第 1 个索引项不包含主键值，但它包含着所有比第 2 个索引项的主键值小的所有可能的数据记录。

（2）叶结点上不包含数据记录本身，而是由记录索引项组成的记录索引块，每个记录索引项包含有主键值和地址指针。每个叶结点中的记录索引项按其主键值大小从左到右顺序排列。每个叶结点最多可存放 n 个记录索引项（n 为不小于 3 的正整数），其格式应为：

k_1	p_1	k_2	p_2	…	k_n	p_n

叶结点到数据记录之间的索引可以是稠密索引：每个记录索引项的地址指针指向一个数据记录，这时，k_i（$1 \leqslant i \leqslant n$）为第 i 数据记录的主键值，$p_i$ 为指向第 i 个数据记录的地址指针。也可以是稀疏索引：每个记录索引项的地址指针指向包含该记录索引项的主键值所在块的起始地址，这时，k_i（$1 \leqslant i \leqslant n$）为第 i 个记录块的最大主键值，$p_i$ 为指向第 i 个记录块的起始地址指针。

通常，为了表述方便，许多文献将叶结点的格式简记为：

k_1	k_2	k_3	…	k_n

即，省略了记录索引项的地址指针。但应注意，这仅仅是为了便于描述，在记录索引项中必须要有地址指针。

（3）一般假设每个索引块能容纳的索引项数是个奇数，且 $m=2d-1 \geqslant 3$；每个记录索引块能容纳的记录索引项也是个奇数，且 $n=2e-1 \geqslant 3$。这里，d 和 e 是大于等于 2 的正整数。

图 3.40 是图 3.39 中多级索引结构的 B－树表示方法，该 B－树是一个 3 阶 B－树。

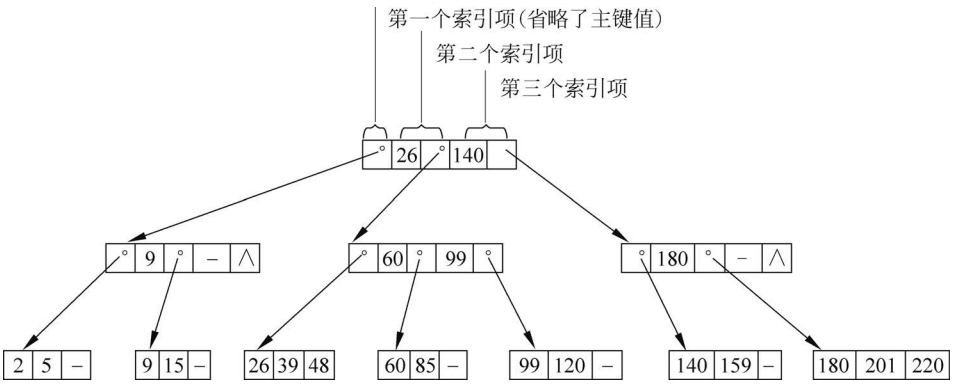

图 3.40　图 3.39 中多级索引的 B－树

4. SQL Server 的索引及其结构

SQL Server 有两种类型的索引：聚集索引（Clustered Index，也称聚类索引、聚簇索引）和非聚集索引（Nonclustered Index，也称非聚类索引、非聚簇索引）。

1）聚集索引

聚集索引是按照数据行（记录）键值的排列顺序在表中存储对应的数据记录，因而表中数据行的物理顺序与索引顺序一致，检索效率比较高。但由于聚集索引为了保证数据行在表中按键值的排列顺序存储，每当有新记录要插入表中时，都涉及表中所有数据行或部分数

据行的重新排列,所以它所需要的空间也就特别大。

聚集索引的结构类似于一种树状结构,由根结点和非叶级结点的索引页和叶级结点的数据页组成,数据页是聚集索引的最底层(叶子结点)。也就是说,根页和非叶级页存放的是索引数据(索引行,由主键值和地址指针组成),叶级页存放的是表中的数据记录。根结点中的每一行指向分支结点,分支结点中的行又指向其他可能的分支结点,最后一级分支结点中的行最终指向叶子结点,最底层的叶子结点为实际的数据页。进一步讲,根结点和除最后一级分支结点的非叶级结点索引页中的每一行中,存放的是指向下一级索引页的指针(页号)及其中的一个主键值;最后一级索引页结点中的行中存放的是主键值和指向包含该主键值所在记录的叶级数据页的指针。进一步讲,SQL Server 中的聚集索引为稀疏索引,存放数据记录的块的大小是 SQL Server 的一个页的大小(8KB)。

聚集索引的结构如图 3.41 所示,其中由根页结点至最后一级分支索引页结点组成的索引结构是一种类似于图 3.40 所示的 B—树结构。

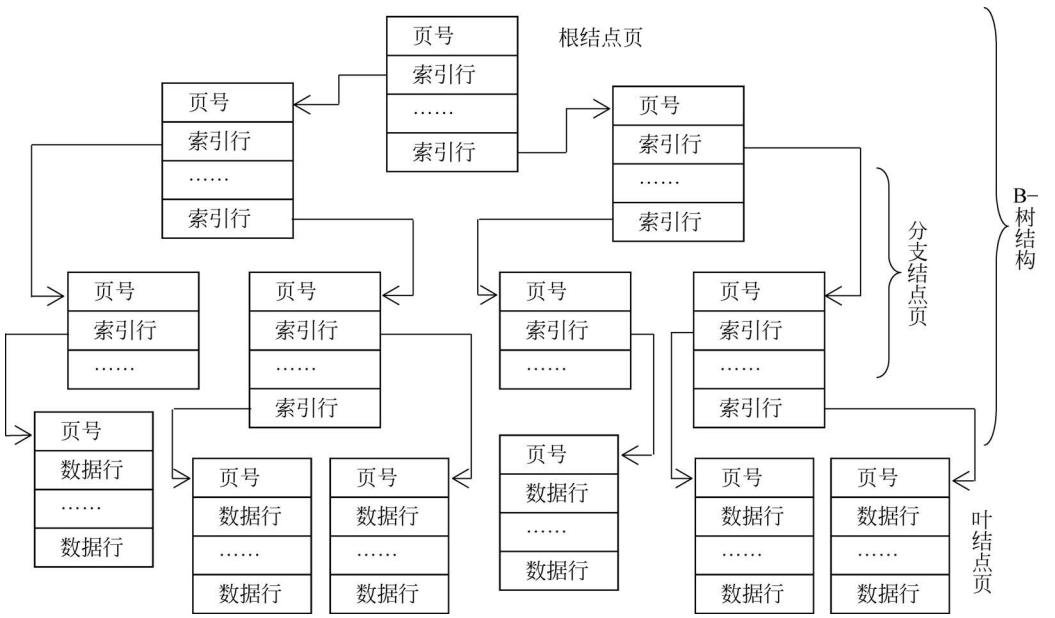

图 3.41　聚集索引的结构示意图

2) 非聚集索引

非聚集索引是一种由根页结点至最后一级分支索引页结点组成的索引结构(B—树结构),是完全独立于数据行的。也就是说,非聚集索引仍包含有按升序排列的列键值,但索引顺序与表中数据记录的物理顺序并不相同,即它并不对表中物理数据页中的数据记录的物理顺序进行重新排序;且叶级索引页中包含的是数据记录的主键值及其指向该数据记录的指针,而不是数据记录本身。

采用非聚集索引时,表中数据在各数据页中的组织采用堆(Heap)方式组织,所以非聚集索引最底层的叶级索引页的各行中的指向数据记录的指针值到各数据页号之间的映射,是一种 Hash 函数映射。

聚集索引的结构如图 3.42 所示,其中上部是 B—树形式的非聚集索引,下部为按堆结构组织的数据页。

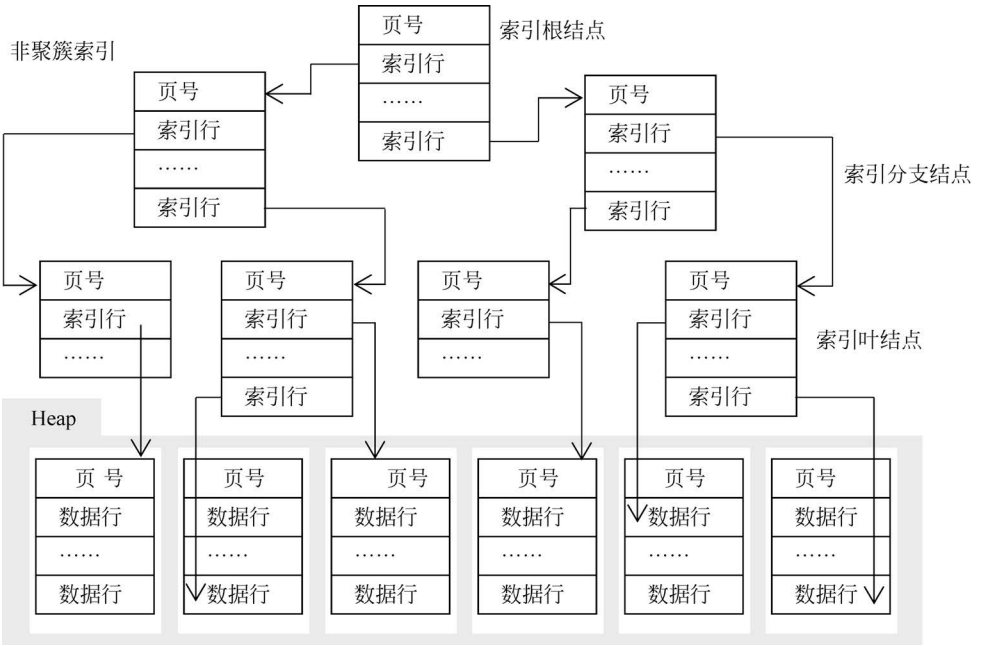

图 3.42　非聚集索引的结构示意图

在 SQL Server 中，聚集索引要求表中的数据记录只能以一种排序方式存储在磁盘上，所以一个表只能有一个聚集索引，但一个表可以有多个非聚簇索引。

3）唯一索引及约束

唯一索引与其他索引唯一不同的是，唯一索引不允许索引键中存在两个相同的值。因为索引中每一个条目都与表中的行对应。唯一索引不允许重复值被插入索引，也就保证了对应的行不允许被插入索引所在的表。显然，唯一索引保障了主键和候选键功能的实现。

在为表声明主键或唯一约束时，SQL Server 会自动创建与之对应的唯一索引。当创建主键约束时，如果表上还没有创建聚集索引，则 SQL Server 将自动在创建主键约束的列或组合上创建聚集唯一索引，主键列不允许为空值。

5. 索引的规划

索引的规划是数据库物理设计的基本问题。索引规划首先是确定需要为哪些关系表建立索引，接着涉及选择建立索引的属性（字段）、建立索引的原则和顺序。

1）选择索引字段

索引是建立在关系的字段上的，所以为表建立索引需要确定以下 3 个问题。

（1）在哪些字段建立索引。

（2）在哪些字段建立组合索引。

（3）要将哪些索引要设计为唯一索引。

2）确认在哪个或哪些字段上建立索引或建立组合索引的一般规则

（1）如果一个（或一组）字段经常出现在选择或连接查询条件中，则考虑在这个（或这组）字段上建立索引（或组合索引）。

（2）如果一个字段经常作为最大值和最小值等聚集函数的参数，则考虑在这个字段上建立索引。

3) 聚集索引和非聚集索引的创建顺序

如果在一个表中既要创建聚集索引,又要创建非聚集索引时,由于创建聚集索引会改变数据记录的物理存放顺序,所以应先创建聚集索引再创建非聚集索引。

4) 索引规划方案的评价

系统对索引的维护是要付出一定的开销的。在一个关系上建立的索引数过多会带来较多的额外开销,降低系统查询速度。

在对每个关系确定了要建立多少个索引后,就形成了索引规划方案。这时就要计算每个索引规划方案对应的系统代价,即各事务运行开销的总和。通过对多个索引规划方案的系统运行代价进行比较,从中选出最佳方案。

6. 索引的创建

完成数据表的索引规划后,就可以利用 SQL Server Management Studio 工具或 T-SQL 语句创建索引了。

索引只有在数据表中有巨大级别的数据量时,才会体现出索引在提升查询效率方面的作用。作为数据库原理课程的初学者,开始设计实现的数据库应用系统中的数据量一般都比较小,因此没有必要为其中的数据表建立索引。但从本书内容的完整性上考虑,下面将给出 SQL Server 创建索引的方法。

1) 创建索引

【例 3.18】 使用 SQL Server Management Studio,为大学教学信息管理数据库中的学生关系 S 在学生姓名 SNAME 列上建立非聚集索引,要求按升序排列。

(1) 启动 SQL Server Management Studio 工具,打开大学教学信息管理数据库 JXGL。

(2) 在"表"对象中选择学生关系表 S,右击选择"修改"菜单命令,打开"表设计器"对话框。

(3) 在"表设计器"界面上右击,弹出快捷菜单,如图 3.43 所示;选择"索引/键"菜单项,系统显示"索引/键"对话框,如图 3.44 所示。

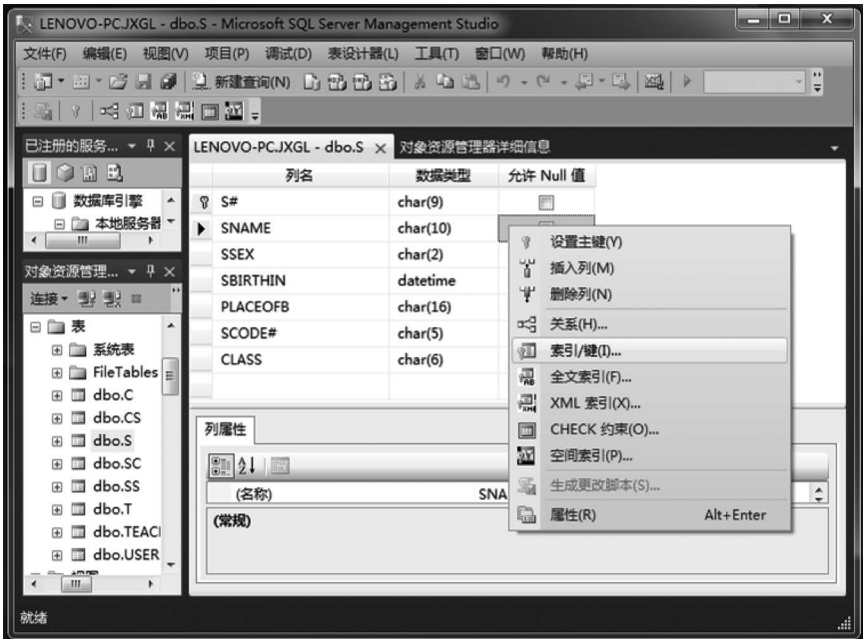

图 3.43 弹出快捷菜单

图 3.44 “索引/键”对话框

(4) 单击“添加”按钮。“选定的主/唯一键或索引”列表将显示新索引的系统分配名称 IX_S,如图 3.45 所示,然后依次设置索引的相关属性,步骤如下。

图 3.45 正在编辑的“索引/键”对话框

① 在“常规”选项中,选择“类型”选项,从属性右侧的下拉列表中选择“索引”项。

② 如图 3.46 所示,在“列”选项中单击[...]按钮,弹出“索引列”对话框(见图 3.47),选择要进行索引的列——学生姓名 SNAME,指定排序顺序为升序(ASC)。最多可选择 16 列,为获得最佳的性能,一般为每个索引选择一列或两列。

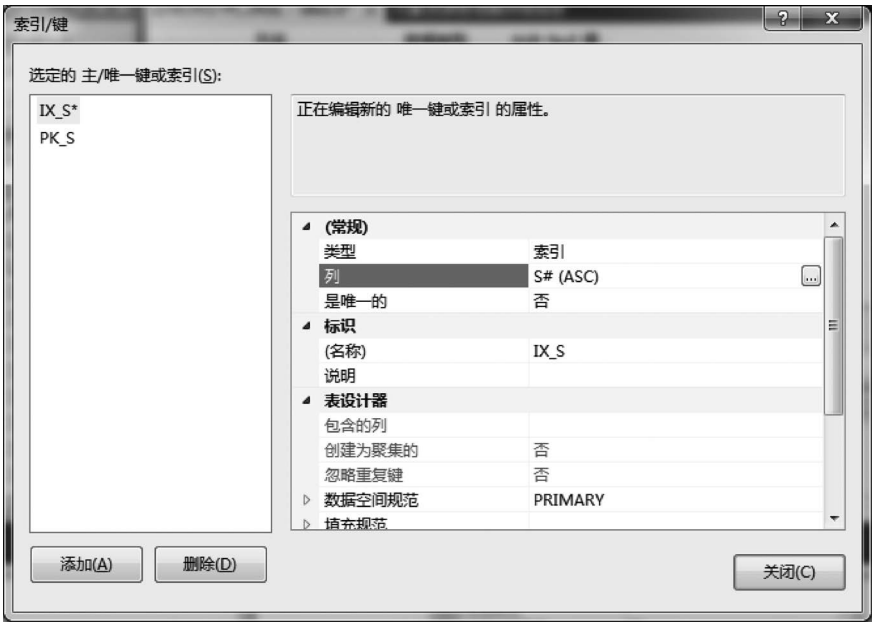

图 3.46　在“索引/键”对话框中编辑“列”选项

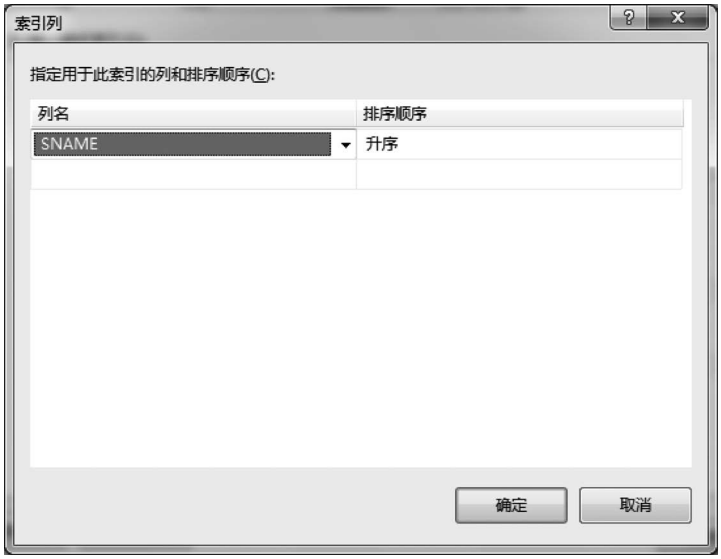

图 3.47　“索引列”对话框

③ 如图 3.48 所示,单击“创建为聚集的”选项,从属性右侧的下拉列表中选择“否”项。

(5) 设置完成后,单击“关闭”按钮,关闭“索引/键”对话框。

(6) 在“文件”菜单中,选择“保存”菜单命令,即可完成设置。

2) 查看索引

【例 3.19】 在大学教学信息管理数据库中,查看学生关系 S 中的索引。

(1) 启动 SQL Server Management Studio 工具,打开大学教学信息管理数据库 JXGL。

(2) 在“表”对象中选择学生关系表 S,展开“索引”目录,如图 3.49 所示。

图 3.48　编辑“创建为聚集的”选项

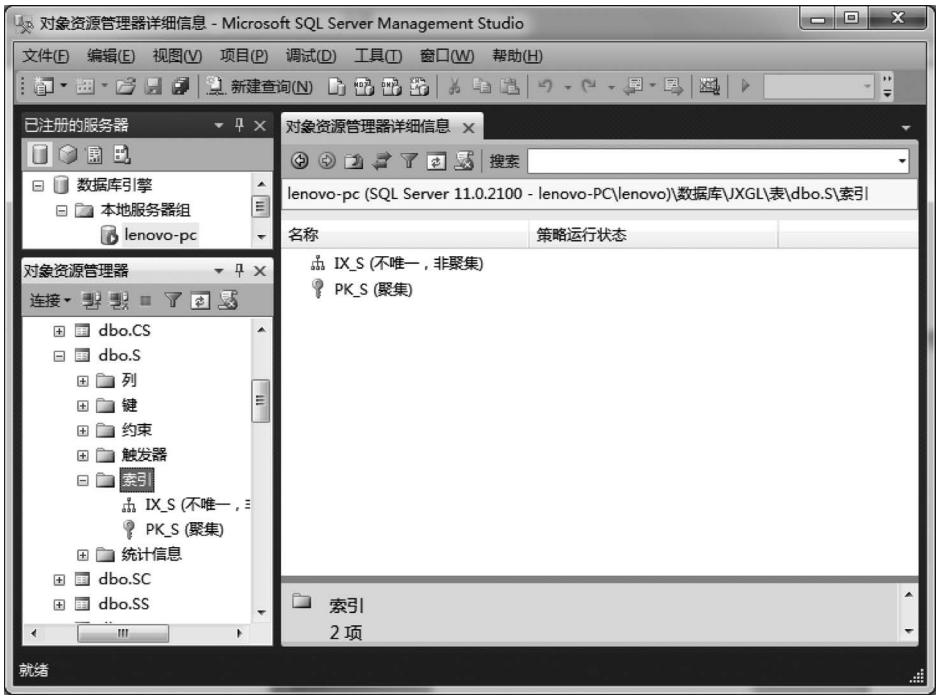

图 3.49　展开学生关系表 S 的“索引”目录

(3) 右击要查看的“索引”项，选择“属性”菜单命令，弹出“索引属性”对话框，如图 3.50 所示。用户可以在该对话框中查看索引的各种属性。

3) 删除索引

【例 3.20】 在大学教学信息管理数据库中，删除学生关系 S 中的索引。

(1) 启动 SQL Server Management Studio 工具，打开大学教学信息管理数据库 JXGL。

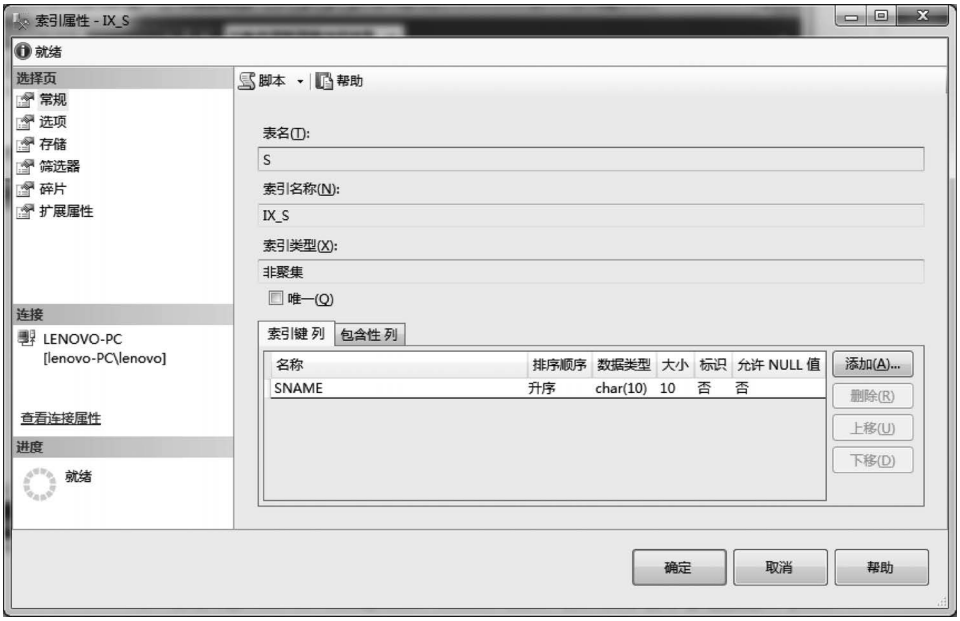

图 3.50 “索引属性”对话框

(2) 在“表”对象中选择学生关系表 S,展开“索引”目录,选中想删除的索引,右击选择“删除”命令。

(3) 在“删除对象”对话框中,如图 3.51 所示,单击“确定”按钮,完成删除。

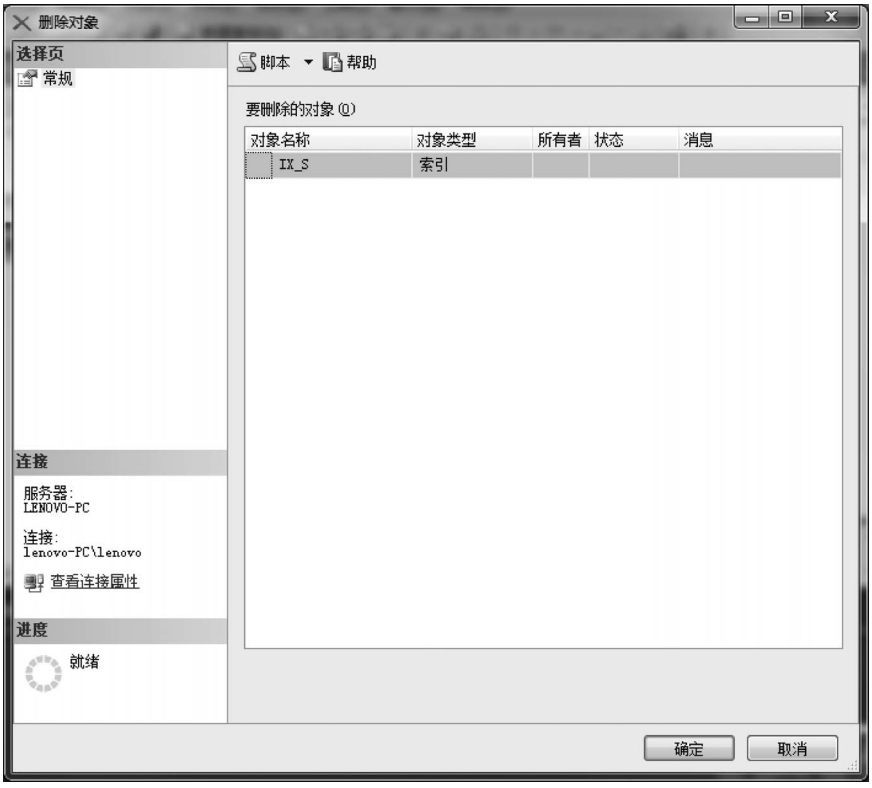

图 3.51 “删除对象”对话框

3.5.6 数据库物理结构评价

在数据库的物理结构设计后，还需对设计好的物理结构进行评价，以便确定是否对其逻辑结构或物理结构进行进一步的优化设计。

评价方法就是对数据库物理设计过程中设计好的存储结构和存储方式，从时间效率、空间效率、维护代价和各种用户要求等方面进行定量估算和权衡。由于评价中要定量地估算各种方案的存储空间、存取时间和维护代价等，所以数据库物理结构的评价方法依赖于所选用的 DBMS。在定量分析和评价过程中由于进行性能比较可能会产生多种方案，数据库设计人员必须对这些方案进行细致的权衡、比较和分析，从中选择一个较优的方案作为数据库的物理结构。如果该结构不符合用户需求，则需要修改设计。

数据库物理结构设计及其评价后的修改过程，以及与逻辑设计和数据库实现阶段的关系如图 3.52 所示。

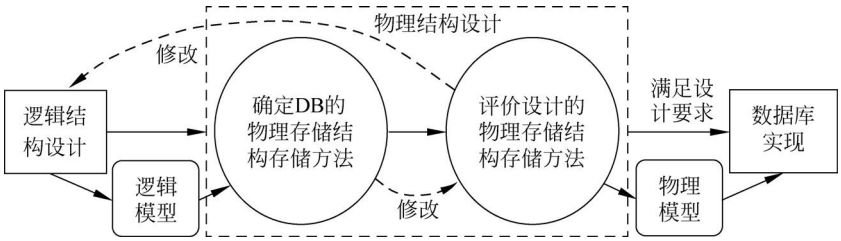

图 3.52 数据库物理结构的设计过程

3.6 数据库实现技术简介

数据库实现主要包括数据库的子模式设计、数据库应用行为设计与实现、装入实际数据进行系统试运行等。

1. 数据库的子模式设计

数据库的子模式设计是根据数据库应用系统面对的不同用户，及他们分别看到的数据库局部逻辑结构，由系统分析员创建必要的用户子模式(用户视图)，以便为应用程序员面向各用户的数据库应用程序查询模块的设计提供方便。

2. 数据库应用行为设计与实现

数据库应用行为设计与实现是指在给数据库表录入和加载实验验证数据的基础上，利用所选用的 RDBMS 的主语言，编程设计能够满足该用户组织中各种用户对数据库应用需求的功能模块和行为特性，即进行应用程序的设计、调试和实现。

数据库应用程序的设计包括功能模块设计、统计分析设计、特殊功能要求设计、用户接口和系统界面设计等。一般来说，当纯粹利用数据库管理系统提供的开发工具开发应用程序时，实现比较方便、快捷，但界面格式相对单调，统计分析功能较弱，用户的一些特殊功能实现起来相对也困难一些。当利用某种主语言，例如 VB. NET 或 C＃编写应用程序时，实现方式比较灵活，便于实现功能比较强大的统计分析和用户的特殊功能要求。VB. NET 和 C＃都是一种既提供了比较好的用户界面设计功能，又便于低层统计分析和特殊功能实现的语言环境，是一种比较适合于数据库应用程序开发的主语言环境。

用 VB.NET 或 C# 这样的主语言编写应用程序时,程序调试方式与用其进行一般的软件系统设计时的软件调试方式基本类似。用数据库管理系统提供的开发工具开发应用程序时,熟练掌握该工具软件的应用特性及其环境参数的设置,对于应用程序的调试十分有益。

基于 SQL Server 2012 标准版数据库管理系统软件、Visual Studio 2010 集成环境中的 VB.NET 7.0 程序设计语言和.NET Framework 的 ADO.NET 2.0 数据库访问对象模型的数据库应用程序设计内容详见第 8 章。

3. 装入实际数据进行系统试运行

试验数据的加载和应用程序的设计与调试过程,实际上就是对建立的数据库应用系统的初步试运行过程,是对数据库应用系统的功能和性能进行测试和评价的过程。在这两阶段的工作中,实际上在不停地对发现的某些概念结构设计和物理结构设计中存在的缺陷和问题进行着某种程度的扩充、修改,甚至重构。但从总体上讲,数据库中的实验数据还比较少,还不便于发现特别是系统性能方面的深层次问题。所以在应用程序设计完成后,要删除掉数据库中的所有临时实验数据,正式录入和加载实际数据,并进入系统的试运行期。

在系统试运行期,要通过反复执行数据库的各种操作,测试系统是否满足用户的功能要求,并对数据库和应用程序进行必要的修改,直至达到系统的功能和设计要求。

在系统试运行期,还要特别注意系统的性能测试、评价和修改完善。一般来说,数据库应用系统的性能主要包括大型查询的响应时间、事务的更新时间开销、大型报表的生成时间开销、数据库的存储空间开销、物理数据库的存储性能和效率等。对于系统性能方面发现的问题要进行必要的专门测试实验,找出设计上的漏洞,及时进行完善和修改。

当数据库试运行结束后,就标志着数据库实现时期的工作已经结束,接下来就可以进入数据库应用系统生命周期的最后一个时期,即数据库应用系统运行与系统维护时期。

3.7 数据库应用系统运行与系统维护

下面先介绍数据库应用系统运行与系统维护过程中涉及的几个概念,再介绍数据库应用系统运行与维护时期的主要工作。

3.7.1 软件维护

一个软件系统投入运行后,系统所实现的功能的正确性、系统运行的可靠性、用户对该软件使用的方便性和系统功能的扩充等方面,往往会存在某些错误或不尽如人意的地方,所以必然涉及对投入运行后的软件系统的维护问题。

软件维护指一个软件交付使用之后,为了改正错误或满足新的需求而维护软件的过程。常用的软件维护包括改正性维护、适应性维护和完善性维护。

1. 改正性维护

软件投入运行后,对发现的某些软件错误地进一步诊断和修改的过程,称为改正性维护。例如,给软件打补丁实质上可看作一种改正性维护。

2. 适应性维护

把为了适应变化了的软件和硬件环境而进行的修改软件的活动,称为适应性维护。例如,软件的升级实质上可看作一种适应性维护。

3. 完善性维护

针对用户对投入运行后的软件提出的增加新功能或修改已有功能的建议而进行的修改和维护软件的活动，称为完善性维护。

3.7.2 运行与维护时期的主要工作

数据库应用系统运行与维护时期的主要工作如下。

1. 必要的改正性维护、适应性维护和完善性维护

数据库应用系统在投入实际应用后，和其他软件系统一样还会发现某些软件错误，因此必然要进行改正性维护；随着时间的推移，会涉及软件进一步适应变化的软件和硬件环境的问题，所以也涉及某种程度的适应性维护；也会发现有某些设计时考虑不周的情况，因而也要进行必要的完善性维护。

2. 数据库备份与恢复及故障维护

数据库在运行过程中，可能因意外的事故、硬件或软件故障、计算机病毒或“黑客”攻击等各种原因造成数据库故障或信息丢失，因此要定期地做好数据库备份工作。对于大型或较大型的数据库系统来说，必须每日进行数据库备份；对于一般的小型数据库来说，也要根据转储备份计划对数据库进行定期备份，以保证数据库发生故障时能尽快将数据库恢复到最近的一致性状态。

当数据库出现故障和信息被破坏或丢失时，要及时做好数据库系统的故障排除和数据恢复工作，确保数据库的正常运行。

3. 数据库运行性能的检测与改善

数据库性能检测就是利用数据库管理系统提供的系统性能参数检测工具，经常性地查看数据库运行过程中物理性能参数的变化情况，检测和分析数据库存储空间的应用情况。特别是，数据库经过较长时间的运行后，会因记录的不断插入、删除和修改产生大量空间碎片，造成数据库运行性能的急剧下降，所以要视情况对空间碎片进行整理，对数据库的存储空间状况及响应时间进行分析评价，结合用户反应确定改进措施，以保证数据库始终运行于最佳状态。

扫一扫 作业　扫一扫 自测题

习　题　3

3-1 解释下列术语。

（1）数据库概念结构　　（2）数据库逻辑结构

（3）一对多联系(1∶*)　　（4）多对多联系(*∶*)

（5）稀疏索引　　（6）稠密索引

（7）数据文件　　（8）用户数据库

3-2 数据库应用系统的生命周期分为哪几个时期？每个时期又分别分为哪几个阶段？

3-3 分别简述数据库应用系统生命周期中的 6 个阶段的主要任务是什么？

3-4 何为数据流图？数据流图与程序流程图有哪些区别？

3-5 简述在下列情况下，实体—联系模型向关系模型转换的一般原则。

（1）当两个实体集之间的联系为 M∶N 联系时。

(2) 当两个实体集之间的联系为1∶N联系时。

(3) 当两个实体集之间的联系为1∶1联系时。

3-6 在图3.53所示的E-R图中,给出了职工、商店、商品实体集及联系集的属性信息等。请给出从该E-R图可以转换成的关系模式,并指出每个关系模式的主键属性。

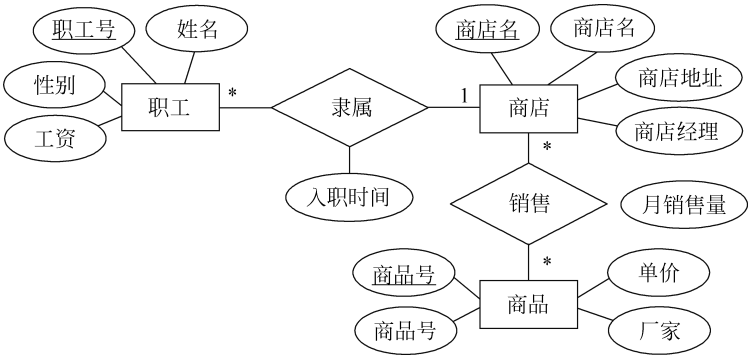

图3.53 习题3-6图

3-7 设一个图书管理系统涉及的人员有读者(主要属性有借书证号、姓名、身份证号、工作单位、读者类型、办证日期、借阅限量数等)、图书管理员(主要属性有管理员号、管理员姓名、身份证号、职称等)、图书(主要属性有图书号、书名、作者、出版社、出版日期、单价、库存数量等)。要求:

(1) 请设计反映读者借阅图书和管理员管理图书业务的E-R图。

(2) 将设计的E-R模型转换成关系模型。

3-8 图3.54给出的是一张交通违章处罚通知书。要求:

交通违章通知书: 编号: TZ11201

司机姓名: XXX 驾驶执照: XXXXXX 地址: XXXXXXXXXX 邮编: XXXXXX 电话: XXXXXXXX
机动车牌照: XXXXXX 型号: XXXXXX 制造厂: XXXXXX 生产日期: XXXXXX
违章日期: XXXXXX 时间: XXXXXX 地点: XXXXXX 违章记载: XXXXXXXX
处罚方式: 警告: 罚款: 暂扣驾驶执照:
警察签字: XXX 警察编号: XXX
被处罚人签字: XXX

图3.54 交通违章通知书图示

(1) 请根据这张通知书所提供的信息设计一个 E-R 模型(违章处罚通知书或只有警告处罚一种情况,或既有警告处罚也有罚款处罚两种情况),图上可省略属性标注。

(2) 将设计的 E-R 模型转换成关系模型(列出所有属性),并标出每一个关系模式的主码和外码(如果有)。

3-9 简述数据库的数据文件和日志文件的功能和作用。

3-10 SQL Server 2012 包括哪两类数据库? 其作用是什么?

3-11 完成与本章内容相关的数据库应用系统课程设计部分内容,主要包括用户数据需求、功能需求和安全性需求分析;基于 E-R 图的数据库应用系统概念结构设计;以 E-R 图向关系模型转换为主要内容的数据库应用系统逻辑结构设计;与 SQL Server 2012 性能相适应的物理结构设计。