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)，例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，在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、基本技术和方法，对于理工类专业的学生，特别是其中的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，以便为本章和后续章节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门知识，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一般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长期以来一直是一个极具挑战性的课题。

本书将软件工程思想与数据库设计技术相结合，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，并将其分成用户需求分析、数据库设计、数据库实现、数据库运行与维护 4 个时期，以及用户需求

分析、概念结构设计、逻辑结构设计、物理结构设计、数据库结构创建、数据库应用行为设计与实现、数据库运行与维护共 7 个阶段。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。

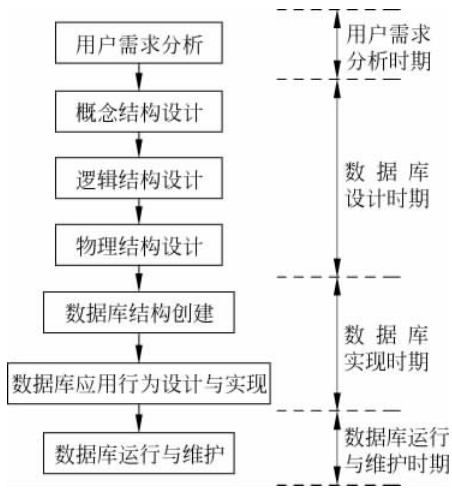

图 3.1 数据库应用系统的生命周期及各个时期对应的阶段划分

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。所以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、数据库实现和数据库运行与维护 4 个时期。

1. 用户需求分析时期

用户需求分析是指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

数据库设计是指设计数据库的结构特性,也即为某一用户组织的数据管理应用需求构造出最优的数据库逻辑结构和物理结构。数据库设计主要包括数据库的概念结构设计、逻辑结构设计和物理结构设计 3 个阶段。

概念结构设计和逻辑结构设计主要有两种方法:一种方法是用需求分析阶段得到的描述用户数据管理需求的各种数据表格作为数据库的概念结构模型,然后以这些表格的表头为关系框架将其变换成一组关系模式,再对其进行规范化设计和优化后,形成数据库的逻辑

结构模型。另一种方法是先将现实世界中的事物及它们之间的联系用 E-R 图表示，得到数据库的概念结构模型，然后再将用 E-R 图表示的概念结构转换成一组关系模式，在进行必要的规范化设计和优化后，形成数据库的逻辑结构模型。相比之下，由于用 E-R 图方法得到的关系模式比较规范，所以规范化设计比较简单，但完成一个大型用户组织信息管理功能的数据库应用系统设计时涉及的（由大量子 E-R 图构成）总体 E-R 图也是一件比较复杂的工作。而由数据表格得到的关系模式通常很不规范，所以规范化设计比较复杂，但在需求分析阶段收集各种现成的信息管理表格却是一件比较容易的事情。

数据库的物理结构是指数据库在物理存储设备上的存储结构和存取方法。因此，需要根据所选用的 RDBMS 的数据库存储方法，为逻辑结构设计阶段设计好的逻辑数据库模型和应用要求，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。

3. 数据库实现时期

数据库实现是指依据数据库物理结构设计阶段设计好的数据库物理模型，创建数据库的物理存储结构，编程设计实现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型和行为特性。数据库实现主要包括数据库物理存储结构创建、数据库的子模式设计、数据库应用需求功能模块和行为特性的编程设计及实现、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（一般简称用户）、应用程序员、系统分析员和数据库管理员（DataBase Administrator，DBA）。不同的人员所看到的数据库是有区别的，也即数据库应用系统具有不同的视图级别。显然，了解这些人员在数据库应用系统生命周期各个阶段的作用，对于数据库应用系统的设计和实现，以及对其后的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。

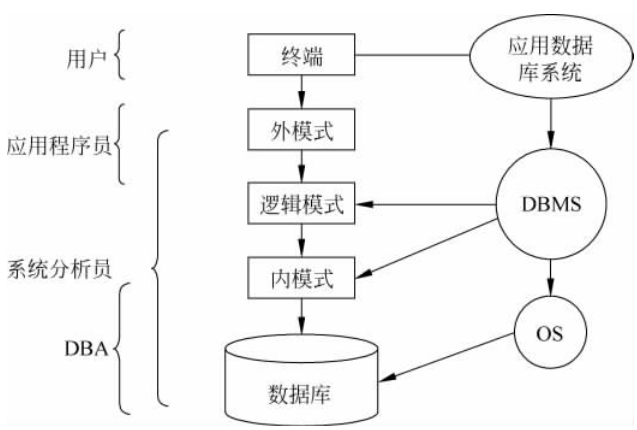

图 3.2　数据库应用系统的视图级别

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。他们一般是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,由于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。但在许多情况下,应用程序员与系统分析员一起完成数据库外模式,甚至包括模式和存储模式的设计,并根据外模式和部分逻辑模式编写应用程序。

(3) 系统分析员。是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。以使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,例如收集碎片等。

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,目的是理清不同人员的职责,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地位。

3.2 用户需求分析

围绕数据库应用系统设计而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。

3.2.1 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

了解用户组织中各个业务的活动内容,建立描述用户业务处理及其信息流动过程的数

据流图(详见 3.2.2 节),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,需要详细了解各个用户(科员或部门)当前业务活动、业务流程和业务处理中各个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;梳理各个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

这一步首先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。接着,通过进一步的梳理和分类,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、出生年月等有关反映学生自然情况的数据信息作为一类。然后,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。最后,还要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个业务部门的信息处理要求,也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,进而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、各个部门的职责及其相互关系、各个部门的

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。例如如图 3.3 所示的大学教务部门的组织机构层次方框图。

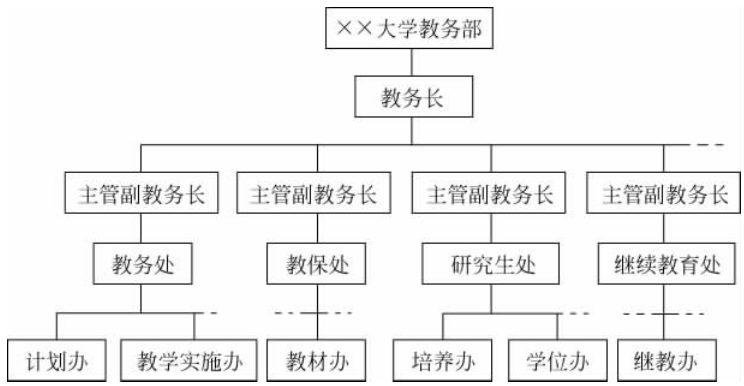

图 3.3 组织机构层次方框图示例

然后,根据各个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、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,就可以最终形成完整的系统需求分析说明书。通常情况下还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。

3.2.2 数据流图及用户业务处理逻辑描述

本节的数据流图和 3.2.3 节的数据字典内容,是有关数据需求的描述方法。

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。

数据流图一般用如图 3.4 所示的 4 种基本符号表示。

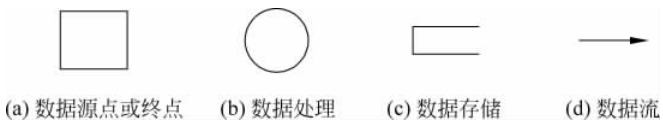

图 3.4　数据流图的基本符号

1. 数据源点或终点

数据的源点是指数据的起源处，数据的终点是指数据的目的地。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，例如作者、出版社、书库管理员等。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，数据库也可以看作是一个文件。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

数据流是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，如图 3.5 所示。

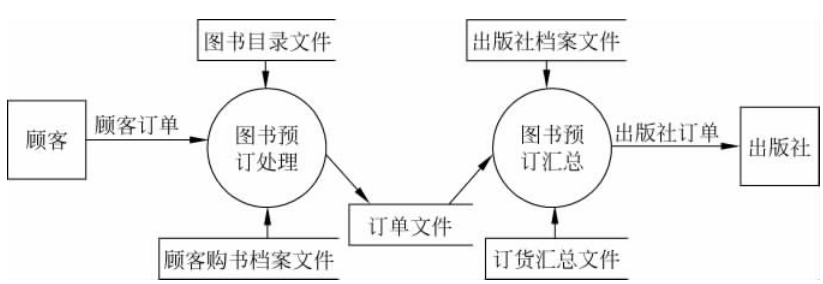

图 3.5　描述图书预订系统处理逻辑的数据流图

3.2.3 数据字典

数据流图表示了数据与处理的关系,但在数据流图中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4个部分。

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、数据表中各个属性(字段)的编号、名称、数据类型、数据长度、取值范围、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	类 型	长 度	字 段 名
1	课程代号	char	7	KCDH
2	课程名称	vchar	50	KCMC
3	课程类型	vchar	8	KCLX
4	学时	numeric	2	XS
5	学分	numeric	1	XF
6	任课教研室	vchar	8	RKJYS

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动方式书写,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。

3.2.4　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的最基本的信息管理要求，一般来说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（例如，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）各种管理信息的报表输出和打印功能

2. 企业网站的功能需求

一般来说，各种网站实质上是一个网络信息管理系统，所以一个功能较强，且具有动态信息管理功能的网站一般应具有下列一些功能模块。

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个办公室。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，目前大多数都是采用 Internet 网络，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个办公室。

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 8 章）等。由于这部分内容已超出了本课程的内容范围，所以不再赘述。

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都采用 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++ Builer)、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，在出现不可抗拒的故障的情况下，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，以便最小程度地减少数据的丢失和破坏。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。一般的方法是系统管理员具有对数据库的一切操作和管理权限；给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；给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10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。这些内容已超出了本课程的内容范围，不再赘述。

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，也称为概念模型。概念结构设计阶段的主要任务是根据用户需求分析阶段形成的系统需求说明书，把用户的信息需求抽象为独立于具体机器、独立于具体DBMS的信息结构。作者以为，概念结构有两种设计方法，一种是属性表概念结构设计方法，即用一组属性表表示数据库概念结构的方法；另一种是实体-联系模型（Entity Relationship Model，E-R模型）概念结构设计方法，也即用实体-联系图（Entity Relationship Diagram，E-R图）表示数据库概念结构的方法。

3.3.1　属性表概念结构设计方法

属性表概念结构设计方法的基本思路是：期望建成的数据库应用系统中的关系表，尽可能地与用户组织进行日常信息管理时用到的各种表格相一致。所以采用的所谓设计方法是，直接把用户组织所用的各种表格进行收集、整理和归纳，把每一个表格看作成是描述其相应数据与信息特性的一个子概念结构（模型），用户组织实施信息管理涉及的所有表格（包括现有的表格和经过梳理新构建的表格），就构成了该用户组织的概念结构（模型）。

例如，表3.2给出了一个简化了的大学学生学籍表，可看作是大学教学信息管理系统中的一个子概念结构（模型）。

表 3.2 大学学生学籍表

学号		姓名		性别		照片
出生日期		籍贯		入学日期		
父亲姓名		母亲姓名		家庭联系电话		
家庭住址				邮政编码		
所学专业				专业代码		
主要经历	起始年月	学习/工作单位		职业	担任职务	备注
入学信息	准考证号		语文成绩		数学成绩	
	总成绩		英语成绩		综合成绩	

E-R 模型是一种典型的用于描述用户组织数据管理应用需求中的数据及其之间联系的图示(E-R 图)方法。由于这种用 E-R 图描述的概念结构简单、直观,易于用户理解,更能体现其语义观点和语义表达能力,所以在数据库应用系统的概念结构设计中得到了广泛的应用。由于 E-R 图数据库概念结构设计方法涉及较多的内容,所以下面分成多节来分别介绍构建 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。

3.3.2 实体与实体集

1. 实体与实体集的概念

实体(Entity)是指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,一位学生,一把椅。

(4) 实体可以是无形的对象。例如,专业,爱和恨这样的概念。

具有相同属性的实体可构成一个实体集。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

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”、“出生年月”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,“出生年月”取值为 1996-12-14。

属性的值域(Domain)是指属性的取值范围。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) “出生年月”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。

3.3.3　实体集之间的联系及联系集

1. 实体集之间的联系

实体集之间有意义的相互作用称为实体集之间的联系(Relationship)。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

如果实体集 E_1 中的每一个实体(至少有一个)至多与实体集 E_2 中的一个实体有联系；反之，实体集 E_2 中的每一个实体至多与实体集 E_1 中的一个实体有联系，则称 E_1 和 E_2 是一对一联系(One-to-one Relationship)，记为“1∶1”。图 3.6(a)描述了一对一联系的概念，图 3.6(b)是一对一联系的符号表示。

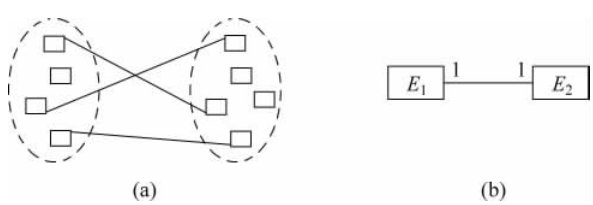

图 3.6　一对一联系

例如，学校与校长之间、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

如果实体集 E_1 中至少有一个实体与实体集 E_2 中的一个以上的实体有联系；反之，实体集 E_2 中的每一个实体至多与实体集 E_1 中的一个实体有联系，则称 E_1 和 E_2 是一对多联系(One-to-many Relationship)，记为“1∶N”，如图 3.7 所示。

例如,校长与教师、系与学生之间都是一对多联系。

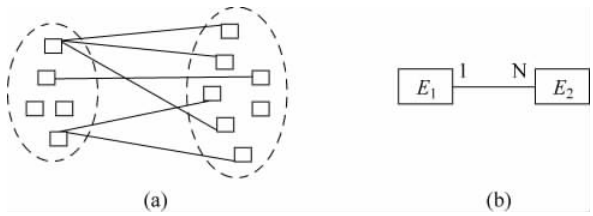

图 3.7 一对多联系

3) 多对多联系

如果实体集 E_1 中至少有一个实体与实体集 E_2 中的一个以上的实体有联系;反之,实体集 E_2 中至少有一个实体与实体集 E_1 中的一个以上的实体有联系,则称 E_1 和 E_2 是多对多联系(Many-to-many Relationship),记为"$M:N$",如图 3.8 所示。

例如,学生与课程、教师与学生之间都是多对多联系。

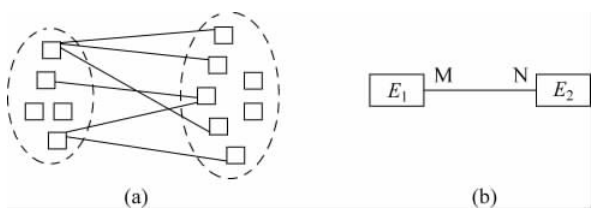

图 3.8 多对多联系

2. 联系集

具有相同属性的联系可构成联系集(Relationship Set),并把能够唯一地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。联系集的标识码一般由被它联系的两个实体集的标识码组成。

3.3.4 E-R 图设计方法

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。有时为了直观起见,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($1:1$、$1:N$ 或 $M:N$)。

下面通过对于每个学生来说,是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,也同时给出了 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）一所大学有多个专业，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，并只属于某一个班级。

（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（M∶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（M∶N 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（M∶N 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（M∶N 联系）的“讲授”联系集来联系。

并在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。

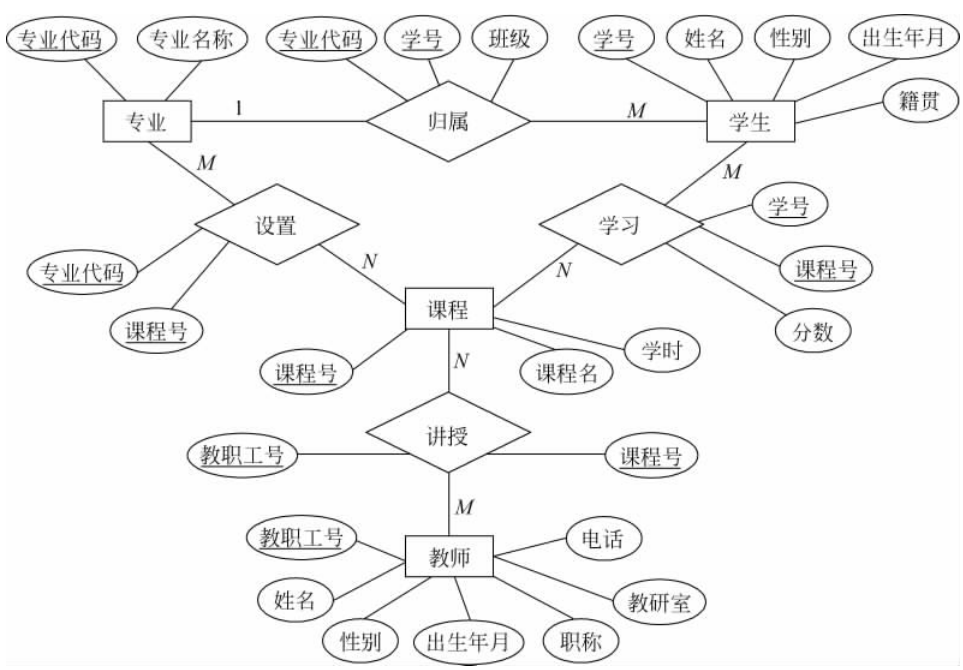

图 3.9　大学教学信息管理 E-R 图

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为学号和课程号;讲授联系集的标识码为教职工号和课程号。

3.3.5 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 图实际上是一种比较规范的实体-联系图。由于现实中数据之间的联系十分复杂,所以在 E-R 图设计中,还会遇到一些特殊的情况。

1. 递归联系

在本节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∶*N* 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。

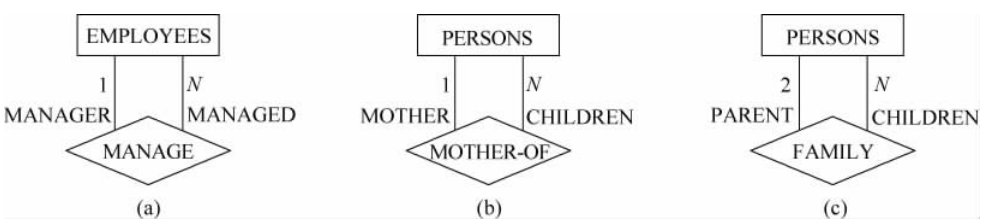

图 3.10 递归联系的 E-R 图表示

2. 冗余联系

冗余联系是指在设计 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。

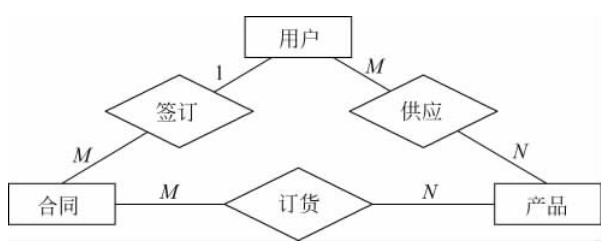

图 3.11 冗余联系示例

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 模型描述的方法。

例 3.4　在如图 3.12 所示的航空公司 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中的成员之一。

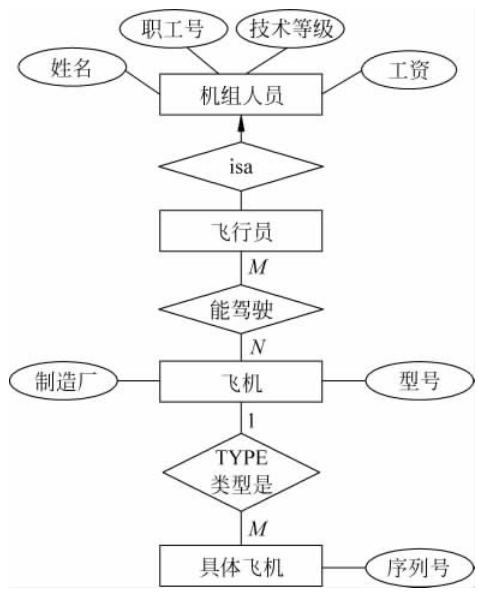

图 3.12　航空公司飞行员和机组人员的 isa 联系 E-R 图

其中：

(1) 实体集飞行员没有属性，利用“飞行员 isa 机组人员”联系可标识每个飞行员。把飞行员分离出来作为单独的实体集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，这种联系是多对多联系。如果对非飞行员的机组人员也保存这样的信息，则可能会造成数据库空间的浪费。

(2) 实体集具体飞机是指航线上航行的每架被编了号码的飞机，是具体的飞行体，这一架可区别于其他一架。

(3) 具体飞机与飞机之间存在 TYPE 联系。具体飞机有唯一的序列号，每次出航对应一架具体飞机。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 联系是由具体飞机至飞机的多对一联系。

由上述描述分析可知，图 3.10 描述的经理(MANAGER)实体集和职工(EMPLOYEES)实体集之间的联系可以用“MANAGERS isa EMPLOYEES”描述，即经理管理职工，但经理也是职工中的一员。同理有“MOTHERS isa PERSON”。在实际的 E-R 图设计中，也可以根据需要将递归联系设计成 isa 联系。图 3.10(a)和图 3.10(b)对应的 isa 联系的 E-R 图如图 3.13(a)和图 3.13(b)所示。

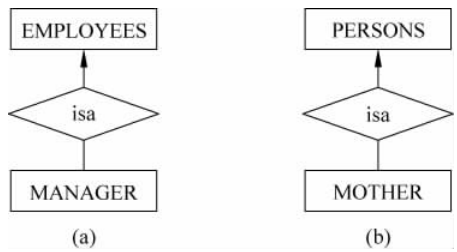

图 3.13　图 3.10(a)和(b)的递归联系的 isa 联系 E-R 图表示

4. 弱实体与弱联系

如果某实体集 E_2 的存在依赖于另一个实体集 E_1 的存在，并且这两个实体集之间的联系是用 E_1 来标识的，那么就把实体集 E_2 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　在如图 3.14 所示的例子中，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就是如此)，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。

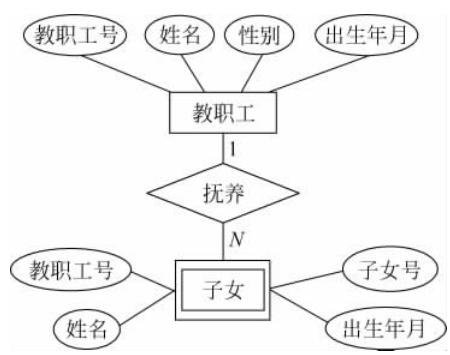

图 3.14　弱实体示例

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(即“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。

3.3.6　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：第一步是设计分 E-R 图，第二步是把各分 E-R 图合并成总体 E-R 图，第三步是对总体 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，一个部门通常又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就至少包括教务管理、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、招生和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 图的设计是指将设计好的各个分 E-R 图进行集成，最终形成一个完整的、能支持各个局部概念结构的数据库概念结构的过程。由于面向各个局部应用的分 E-R 图可能是由不同的设计人员设计的，因此在将各个分 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 图都由同一个人设计，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，仍会存在一定的不一致或矛盾。所以总体 E-R 图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。

同名异义是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。例如在一个大学的教学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。

异名同义是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一般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

属性特征冲突是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一般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，而且在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

结构冲突是指同一意义的对象（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，当然有时也要考虑各属性排列顺序上的不一致。

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起来。

（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 E_1 和 E_2 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，以便使其一致起来。

（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是(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个方面的综合上,考虑将该对象确定为实体还是确定为属性。

3. 总体 E-R 图的优化

由于在设计面向各个局部应用的分 E-R 图时,可能对全局应用考虑不周,因而有时会使得到的总体 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信息冗余,导致在不同的关系模式中存在冗余的属性(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。

由于在面向各个局部应用时对全局应用考虑不周,设计出的各分 E-R 图中可能存在着某些冗余,以至于使得到的总体 E-R 图存在冗余数据和冗余联系。冗余数据是指可由基本数据导出的数据,冗余联系是指可由实体间的其他联系导出的联系。显然,冗余信息的存在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 3.6 在如图 3.15 所示的 E-R 图中,就课程实体集来说,它应该有课程号、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 3.15 的学习联系中消去学分属性,由此就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)

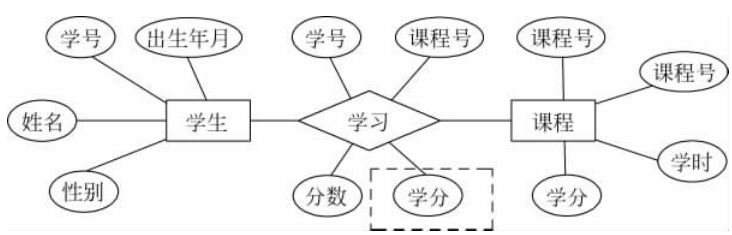

图 3.15 消除 E-R 图中的冗余信息示例

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,就是按照一定的规则,将概念结构设计阶段设计好的独立于任何 DBMS 数据模

型的信息结构，转换为由已选用好的 RDBMS 产品所支持的一组关系模式，并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和处理，在此基础上，还要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，对这组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，也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) 将由属性表或 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　属性表表示的概念结构向关系模型的转换

将由属性表表示的概念结构转换成关系模型的基本思路是，把用属性表表示的每一个子概念结构转换成一个或多个关系模式。转换规则如下。

(1) 把属性表中具有单值属性的所有属性转换成一个关系模式(简称其为主关系模式)，并确定该关系模式的主键。

(2) 把属性表中的那些由多值属性组成的子表分别转换成一个关系模式(简称为子关系模式)，并在各子关系模式中加入主关系模式的主键属性，同时进一步确定各子关系模式的主键。

例 3.7　将用表 3.2 表示的大学学生学籍子概念结构转换成关系模式。

按照上面的转换规则，转换成的主关系模式为

学生学籍关系(学号，姓名，性别，出生日期，籍贯，入学日期，父亲姓名，
母亲姓名，家庭联系电话，家庭住址，邮政编码，所学专业，
专业代码，照片，准考证号，总成绩，语文成绩，数学成绩，
英语成绩，综合成绩)

学生学籍关系的主键由“学号”一个属性组成。

进一步分析可知，在表 3.2 中有一个子表，转换成的子关系模式为

学生主要经历关系(学号，起始年月，学习或工作单位，职业，担任职务，备注)

显然，学生主要经历关系的主键应由“学号”和“起始年月”两个属性组成。

需要注意的是，按照上述转换规则可以实现由同一属性表(子概念结构)转换成的各关系模式之间的“连接运算”，对于不同的属性表转换成的关系模式之间的“连接”问题，需要在进行关系模式规范化后，在关系模式优化阶段由数据库设计者根据系统的可能查询需求分析确定。由于篇幅所限不再赘述。

3.4.2　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的转换和一对一联系向关系模型的转换 3 种情况。

1. 多对多联系向关系模型的转换

转换规则：当两个实体集间的联系为 $M:N$ 联系时，每一个实体集用一个单独的关系

模式表示,该关系模式的属性用相应实体集的属性表示,关系的键用相应实体集的标识码表示。联系集也用一个单独的关系模式表示,该关系模式的属性用该联系集的属性表示,关系的键用该联系集的标识码表示。

例 3.8 按照上述转换规则,可将图 3.9 中的 3 组 $M:N$ 联系转换成如图 3.16 所示的 7 个关系模型。

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 3.16 图 3.9 的 E-R 图中的 $M:N$ 联系的关系模式

2. 一对多联系向关系模型的转换

转换规则:当两个实体集间的联系为 $1:N$ 联系时,联系两个实体集的联系集不再单独设置为关系,两个实体集的转换采用如下策略。

(1) 将位于联系集 1 端和 N 端的实体集按“多对多联系向关系模型的转换”中所述的转换方式分别转换成一个关系模式,并将 1 端实体集的标识码和联系集的非标识码属性加入到 N 端实体集所转换成的关系模式中(即,将其作为 N 端实体集所转换成的关系模式的属性)。

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”加入到学生关系模式中。从而得到对应于图 3.9 中的 $1:N$ 联系的关系模式如图 3.17 所示。显然,学生关系模式应该选用图 3.17 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 3.17 图 3.9 的 E-R 图中的 $1:N$ 联系的关系模式

(2) 对于像“一般”意义上的实体集和“具体”意义上的实体集这样的两个实体集之间的 $1:N$ 联系,将位于 1 端的“一般”意义上的实体集和位于 N 端的“具体”意义上的实体集按“多对多联系向关系模型的转换”中所述的转换方式分别转换成一个关系模式,并将联系它们的联系集看作一个属性加入到位于 N 端的实体集对应的关系模式中。

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 $1:M$。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

转换规则：当两个实体集间的联系为 1∶1 联系时,联系两个实体集的联系不再单独设置为关系,将位于联系集两端的实体集按“多对多联系向关系模型的转换”中所述的转换方式分别转换成一个关系模式,并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非标识码属性。

例 3.11　公司实体集与公司总裁实体集之间存在 1∶1 联系,其 E-R 图如图 3.18 所示。在将其转换成关系模型时,公司实体集和公司总裁实体集分别被转换成一个关系模式,并按其查询习惯在公司关系模式中加入另一个关系模式(公司总裁关系模式)的总裁姓名(此处认为公司总裁姓名不重名,是公司总裁关系模式的主键)和任职年月(是任职联系集的非标识码属性)。转换成的关系模式如下所示：

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

同理也可以转换成如下的关系模式：

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)

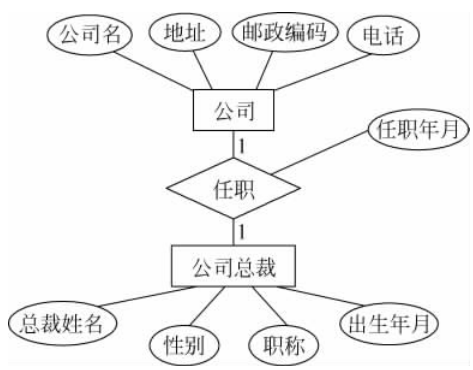

图 3.18　公司总裁与公司的一对一联系

下面对上述转换原则中的有关概念做进一步的强调和解释。

(1) E-R 图中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。

(2) 在 1∶N 联系的转换规则中,“一般”意义上的实体集是指传统意义上的实体集概念,例如“飞机”实体集；而“具体”意义上的实体集是指由“一般”意义上的实体集中的某些具体的个体组成的特殊实体集,例如由一些具有特殊特性的个体飞机组成的“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。

例如,对于如图 3.9 所示的大学教学信息管理 E-R 图,按照上述原则转换成的关系模式如下。

- 学生关系模式：S(S#,SNAME,SSEX,SBIRTHIN,PLACEOFB,SCODE#,CLASS)
- 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#)

另外，在这一步的设计中也存在部分优化的问题，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就可以删除掉。

3.4.3 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

在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。由于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，也即判断其是否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，也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，最终实现使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 6 章详细介绍。

2. 关系模式的优化

当关系模式的规范化程度越高时，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查询效能。

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#)
```

但进一步分析可知，一个学校的专业数相对于开设的课程数来说要少得多。因此，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。所以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后,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

数据库是存储在物理设备上的。因此,在设计好数据库的逻辑结构后,就需要根据所选用的关系 DBMS 软件的数据库物理结构设计要求和应用需求,在物理存储设备上为逻辑结构设计阶段设计好的数据库逻辑结构,选取和设计一个占用较少存储空间、具有尽可能高的查询效率和较低维护代价的数据库物理结构;并对设计好的物理结构从时间和空间效率方面进行评价,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。所进行的这些选取和设计工作即为数据库的物理设计。

数据库物理结构的设计过程及其与逻辑设计和数据库实现阶段的关系如图 3.19 所示。

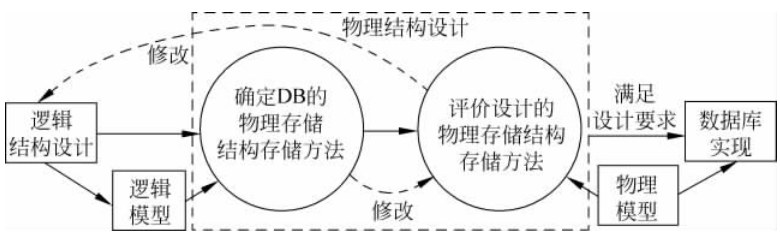

图 3.19　数据库物理结构的设计过程

为了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 的存储组织方式、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的索引和聚簇(也称聚集)的建立等。

下面主要从数据库的物理文件与数据表、数据库物理文件的存储位置、索引技术(数据表的存储结构与组织方式 1)、数据聚簇(数据表的存储结构与组织方式 2)、数据库物理结构评价 5 个方面,说明数据库物理结构设计中涉及的有关问题。

3.5.1　数据库的物理文件与数据表

1. 数据库的物理文件

数据库中的所有的数据信息和控制信息都是存放在物理文件中的。总体上来说,数据

库的物理文件主要包括数据库数据文件、控制文件和日志文件3类。

1) 数据库数据文件

数据库数据文件即数据库中存放用户数据的文件。数据库数据文件一般由多个数据文件组成,其中的主数据文件用来存储数据库的启动信息和部分数据,是数据库的起点,并指向数据库中的其他次数据文件。

2) 控制文件

控制文件存放与数据库所有文件相关的控制信息,用于实现数据库的安全性和完整性控制,决定恢复数据时使用哪些重做日志等。每个数据库必须至少有一个控制文件,通常使用两个或两个以上的控制文件。

3) 日志文件

日志文件是一个系统文件,用于记录用户对数据库进行的输入、删除和修改等数据库更新的信息。例如,从事更新的用户信息;开始更新时间,更新结束时间;更新前的数据(当为修改操作时),更新后的数据;更新的数据(当为输入或删除操作时)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时,就可在系统恢复机制的控制下实现对数据库中数据的恢复。

每个数据库要求至少有一个日志文件,通常使用两个或两个以上的日志文件。

2. 数据表

数据表是关系数据库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,当数据库文件多于一个时,每一个数据表及其数据只能存储在其中的一个数据库文件中。

数据表是SQL Server数据库中的最重要的数据库对象之一。

3.5.2 数据库物理文件的存储位置

数据库物理文件的存储位置与计算机系统的硬件环境(例如,配置的硬盘个数)、应用需求(例如,是单机数据库系统还是需要远程访问的网络数据库系统)、系统存储空间的分布和存储时间及存储空间利用率等有关。

对于一个小型数据库来说,可以将数据库系统软件、用户数据库数据文件、日志文件和控制文件放置在一个磁盘上的不同的文件夹上,即可降低存储访问的冲突和提高数据安全性。

对于一个大型或较大型数据库来说,一般都有一个主数据文件和若干个次数据文件。由于主数据文件中主要存放的是数据库的启动信息、数据表目录、指向数据库的其他次数据文件信息,以及部分数据信息;其他次数据文件中存放的是数据信息。所以,通过把主数据文件和次数据文件存储在不同的驱动器上,就可以减少小磁盘访问的竞争。

每个数据库至少应拥有一个日志文件,但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、有比较长的恢复时段和恢复还原的更快速度,每个数据库应至少配置两个或3个日志文件。数据库将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。当最后一个联机日志文件写满后,数据库将用新的事务的记录覆盖重

写第一个日志文件的内容。也即又在第一个日志文件开始写事务记录，并如此循环。另外，由于日志文件保留了数据库中当前事务的信息，所以除非数据库在备份之前关闭，否则就无法从备份中恢复它们。所以，需要把这些日志文件与数据文件分开存放在不同的磁盘驱动器中，以避免出现相互间的性能影响。

为了保证数据库的安全性和控制信息的不丢失，对于一个大型或较大型数据库来说，需要配置有两个或 3 个控制文件，通过把多个控制文件分别存储在不同的驱动器上，就可以提高系统的安全性。相对于其他数据库文件来说，控制文件只有很少的 I/O 操作。

依据上述的分析，表 3.3 给出了一个小巧而优秀的 4 磁盘规划设计方案，这些磁盘可以是逻辑磁盘，也可以是物理磁盘。

表 3.3　四磁盘规划设计方案

磁　盘	存 放 内 容
1	DBMS 系统软件；应用软件
2	控制文件 1；主数据文件
3	控制文件 2；次数据文件
4	控制文件 3；日志文件 1、2、3

3.5.3　索引技术(数据表的存储结构与组织方式 1)

关系数据库是以表的形式组织数据的，各个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，几万个记录、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的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和为多个表建立聚簇。

在关系型数据库系统中，索引为实现数据库信息的快速查询提供了技术支持。

1. 索引的概念

索引是现实生活中最常用的快速检索技术，例如词典中的索引和图书资料库中的索引等。在数据库中，索引是对数据库表中一列或多列的值进行排序的一种数据结构，由给定的一个或一组数据项(主键或非主键)组成。

设 $k_i(i=1,2,\cdots,n)$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_{k_i}的地址为 $A(R_{k_i})$，则$(k_i,A(R_{k_i}))$称为一个索引项，由多个索引项组成的如图 3.20 所示的表形式的数据结构称为索引。

K_1	K_2	K_3	$\cdots$	k_n
$A(R_{k_1})$	$A(R_{k_2})$	$A(R_{k_3})$	$\cdots$	$A(R_{k_n})$

图 3.20　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 3.21 是一种学生关系及其索引的图示表示方式。

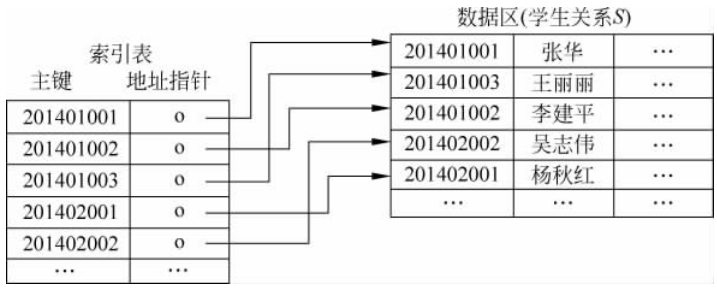

图 3.21 学生关系索引

由图 3.21 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,进而直接找到要查询的记录。由于索引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。

2. 线性索引

线性索引是一种按照索引项中数据项的值排序的索引方式,其中数据项一般为主键。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,每一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 3.21 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,修改关系中的记录和删除关系中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,所以实现对关系中记录删除、插入、修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个块中,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 3.22 是一个稀疏索引的例子。

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除、插入、修改,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

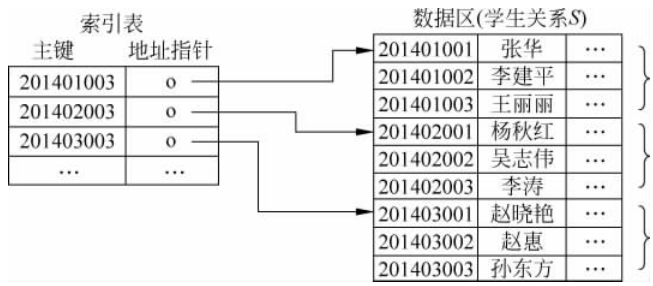

图 3.22　学生关系的稀疏索引方式

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为止。这种多级索引如图 3.23 所示，是一棵多级索引树。其中假设每个块可以存放 3 个索引项。

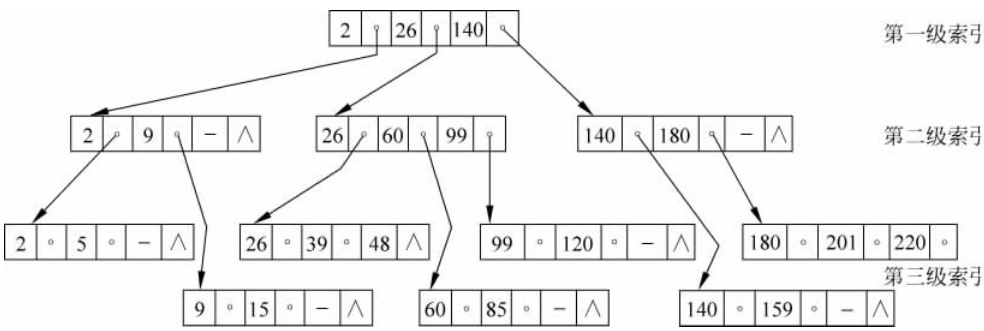

图 3.23　多级索引

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(平衡树)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，简称为索引存储块或索引块。叶结点是存放记录索引项的存储块，简称为记录索引块或叶块，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，有如下的 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棵子树，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$\lceil m/2 \rceil$棵子树($\lceil \rceil$为向上取整符号,例如,$\lceil 3/2 \rceil = 2$)。

(4) 从根结点到叶结点的每一条路径长度相等,也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,每个索引块的第一个索引项不包含主键值,但它包含着所有比第二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 个数据记录的主键值,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定义为

k_1	k_2	k_3	…	k_n

也即,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。本书在 B—树和 B+树的图示中,仍采用了这种方法。

(3) 一般假设每一个索引块能容纳的索引项数是个奇数,且 $m=2d-1 \geqslant 3$;每一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 3.24 是图 3.23 中多级索引结构的 B—树表示方法,该 B—树是一个 3 阶 B—树。

由图 3.24 可知,B—树中的主键值分布在各个索引层上。根结点和内结点中的索引项有两个作用:一是标识搜索的路径,起路标作用;二是标识主键值所属数据记录的位置,由其主键值即可指出该主键值所属记录的位置。

4. SQL Server 的索引结构

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

1) 聚簇索引

聚簇索引是一种对表在物理数据页中的数据,按建立索引的列值(由一个或多个列组成

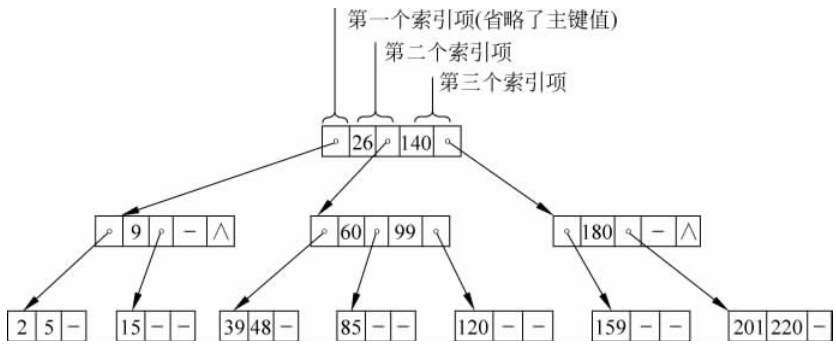

图 3.24　图 3.23 中多级索引的 B—树

的主键)进行(升序)排序,并按排序结果重新把数据记录物理地顺序存放在一起的索引结构。也就是说,聚簇索引中索引行的顺序与表中数据记录的物理顺序相同。

聚簇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,数据页是聚簇索引的最底层(叶子结点)。也就是说,根页和非叶级页存放的是索引数据,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。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。聚簇索引的结构如图 3.25 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 3.24 的 B—树结构。

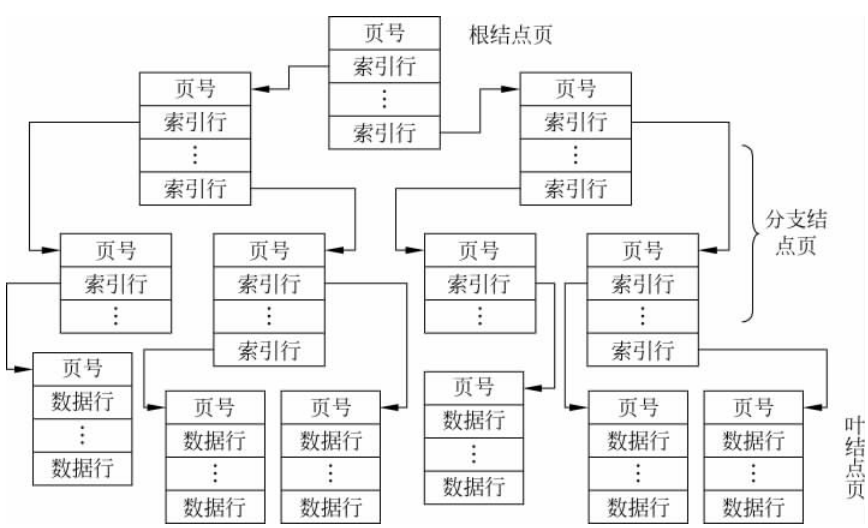

图 3.25　聚簇索引结构示意图

2) 非聚簇索引

非聚簇索引是一种如图 3.25 所示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)。也就是说,非聚簇索引仍包含按升序排列的列值,但它并不对表在物理

数据页中的数据记录的物理顺序进行重新排序,所以索引顺序与表中数据记录的物理顺序并不相同,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。也即,非聚簇索引的最底层的叶级索引页中的行指向的是数据页中的数据记录。由于在采用非聚簇索引时,表中数据在各数据页中的组织采用堆(Heap)方式组织,所以非聚簇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。聚簇索引的结构如图 3.26 所示,其中上部是 B-树形式的非聚簇索引,下部为按堆结构组织的数据页。

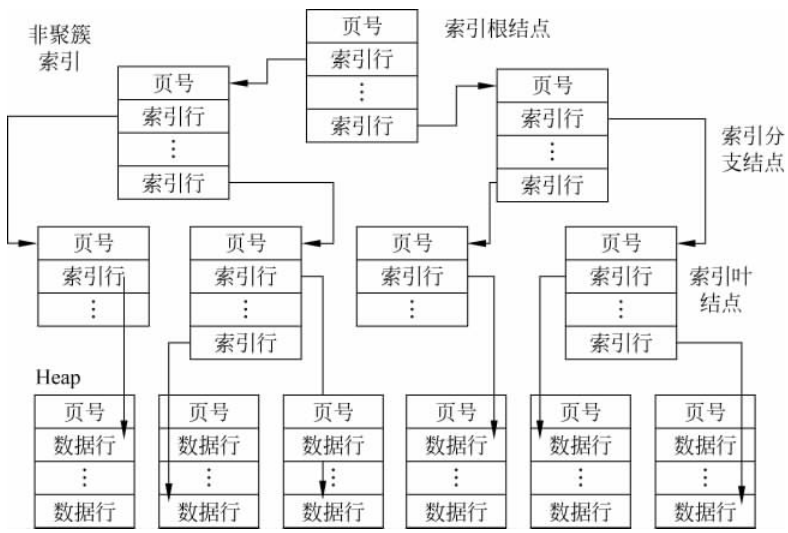

图 3.26 非聚簇索引结构示意图

在 SQL Server 中,由于一个表中的记录只能以一种物理顺序存放,因此每个表只能有一个聚簇索引。但一个表可以有不止一个非聚簇索引,最多可以建立 249 个非聚簇索引,每一个非聚簇索引提供一种访问数据的不同排序顺序。

聚簇索引和非聚簇索引可通过企业管理器创建,也可使用 Transcaction-SQL 语句创建。聚簇索引一般比非聚簇索引的查询速度快很多,因为对于聚簇索引,一旦具有第一个索引值的记录被找到,具有连续索引值的记录也一定物理地紧跟其后;由于非聚簇索引只对表进行逻辑排序,所以查询速度不如聚簇索引快。

3.5.4 数据聚簇(数据表的存储结构与组织方式 2)

数据聚簇是数据库物理结构设计中提高查询性能的另一种有效方法。

数据聚簇的基本思想是,使那些经常在一起进行连接查询的表的数据在物理介质上尽量临近存放,也即把它们聚集地存放在一起。其基本实现方法是:进行连接查询的几个表必定存在公共字段,这些公共字段或者是某个表的主键,或者是某个表的外键,所以就可利用这些公共字段,把相关表中主键与外键相同的记录临近存放,把多个表的数据存储到同一物理块上。这样在进行连接查询操作时,进行连接运算的几个表的数据就在同一个物理块中同时调到内存中,从而可以减少存储管理中的页面调进调出次数和搜索时间。另外,聚簇方式在存储几个表的数据记录时,值相同的主键与外键对多表只存放一次,所以聚簇还可以

减少存储代价。为了便于理解，下面以图示的方式说明聚簇的实现思想。

例 3.12　设有如下的学生关系模式 S1 和专业关系模式 SS，其对应的具体关系如图 3.27 所示。

学生关系模式：S1(S#，SNAME，SSEX，SBIRTHIN，SCODE#)
专业关系模式：SS(SCODE#，SSNAME)

学生关系 S1

学号	姓名	性别	出生年月	专业代码
201401001	张华	男	1996-12-14	S0401
201401002	李建平	男	1996-08-20	S0401
201401003	王丽丽	女	1997-02-02	S0401
201402001	杨秋红	女	1997-05-09	S0402
201402002	吴志伟	男	1996-06-30	S0402
201402003	李涛	男	1997-06-25	S0402
201403001	赵晓艳	女	1996-03-11	S0403

专业关系 SS

专业代码	专业名称
S0401	计算机科学与技术
S0402	指挥信息系统工程
S0403	网络工程
S0404	信息安全

图 3.27　关系模式 S1 和 SS 的具体关系

显然，专业代码既是专业关系 SS 的主键(SCODE#)，又是学生关系 S1 的外键(SCODE#)。利用专业代码 SCODE# 这个公共字段建立的表 SS 和表 S1 的聚簇的逻辑表示如图 3.28 所示。

图 3.28　关系 S1 和关系 SS 的聚簇数据逻辑结构示例

由图 3.28 可见，两个关系在公共属性 SCODE# 上值相等的记录被临近地聚集在一起，而且两个关系的公共属性 SCODE# 的值只存放了一次。显然，两个关系中数据的这种存储方式特别便于连接操作时数据的存取。

3.5.5　数据库物理结构评价

在数据库的物理结构设计后，还需要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。

3.6 数据库实现技术简介

数据库实现主要包括数据库物理存储结构创建、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. 数据库物理结构创建

数据库物理结构创建就是用 RDBMS 提供的 DDL 语言创建数据库,确定数据库的存储分配模式;创建关系模型中的每一个关系模式,包括创建表(Create Table)、创建索引(Create Index)、创建聚簇(Create Cluster)等。

创建数据库时一般需要确定数据库的名称、所有者、大小以及存储该数据库的文件或文件组。创建数据库还涉及数据库存储结构中的有关存储参数的设置问题。合理的参数设置可以提高数据库的存储效率和查询效率,不合理的参数设置不仅会降低数据库的存储效率和查询效率,而且当数据库运行到一定程度时还可能使数据库崩溃,所以,应当引起高度的重视。详细的参数设计问题已经超出了本书的内容范围,故不再赘述。SQL Server 2012 的数据表创建和索引创建分别见 4.4 节和 4.6 节。

2. 数据库的子模式设计

数据库的子模式设计就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),以便应用程序员为面向各个用户的数据库应用程序的设计提供方便。

3. 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,利用所选支持 RDBMS 的主语言,编程设计实现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,也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# 编制应用程序时,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,又便于底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。

当用 VB. NET 或 C# 这样的主语言编制应用程序时,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。当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是十分有益的。

基于 SQL Server 2012 Standard 标准版数据库管理系统软件、Visual Studio 2010 集成环境中的 Visual Basic. 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 9 章。

4. 装入实际数据进行系统试运行

实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。在这两个阶段的工作中，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，甚至于重构。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，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。

当系统试运行结束后，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，然后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

在一个软件系统投入运行后，不论是从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。

软件维护是指在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的进一步诊断和修改过程，称为改正性维护。例如，给软件打补丁实质上可看作是一种改正性维护。

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是一种适应性维护。

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了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,所以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。

习题 3

3-1 解释下列术语。

(1) 应用程序员　　(2) 系统分析员

(3) 数据库管理员(DBA)　　(4) 数据流图

(5) 数据字典　　(6) 数据库概念结构

(7) 一对多联系(1∶N)　　(8) 多对多联系(M∶N)

(9) 线性索引　　(10) 稠密索引

(11) 稀疏索引　　(12) 索引存储块

(13) 记录索引块　　(14) 改正性维护

(15) 适应性维护　　(16) 完善性维护

3-2　什么是数据库应用系统设计?

3-3　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?

3-4　分别简述数据库应用系统生命周期中的 7 个阶段的主要任务。

3-5　数据流图的用途和作用是什么?

3-6　用户需求分析的具体步骤是什么?

3-7　简述属性表概念结构设计方法的基本思想。

3-8　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) 当两个实体集间的联系为 M∶N 联系时;

(2) 当两个实体集间的联系为 1∶N 联系时;

(3) 当两个实体集间的联系为 1∶1 联系时。

3-9　将图 3.12 的实体-联系图转换成关系模型。

3-10　建立索引的目的是什么?

3-11　数据聚簇的基本思想是什么?

3-12　简述数据库的数据文件、控制文件和日志文件的功用。

3-13　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。