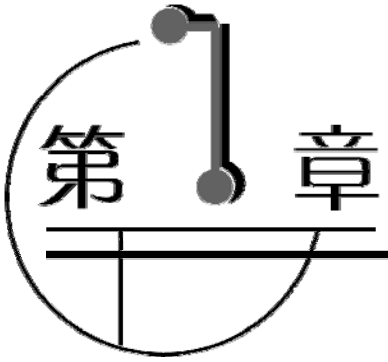


第1章 数据库系统概述

学习目标

数据库作为数据管理技术，是计算机科学的重要分支。在当今信息社会中，信息已经成为各行各业的重要财富和资源，对数据库的应用无处不在。因此，掌握数据库的基本知识和使用方法不仅是计算机科学与技术专业、信息管理专业学生的基本技能，也是非计算机专业学生应该具备的基本技能。本章主要介绍数据库系统的基本概念，包括数据库系统常用术语、数据库系统的体系结构、数据库管理系统、安全性和完整性、关系数据模型、关系运算、关系模式的规范化理论，以及数据库设计的方法与步骤等。

本章重点

- 数据与数据处理
- 数据库与数据库系统
- 关系数据模型
- 关系代数
- 规范化理论
- 数据库设计的方法和步骤

1.1 数据库的相关概念

数据库是信息系统的核心与基础，它提供了最基本、最准确、最全面的信息资源，对这些资源的管理和应用，已经成为人们科学决策的依据。数据库应用已遍及人们生活中的各个角落，如铁路及航空公司的售票系统、图书馆的图书借阅系统、学校的教学管理系统、超市售货系统和银行的业务系统等。数据库与人们的生活密不可分，几乎每个人的生活都离不开数据库。对于一个国家来说，数据库的建设规模、数据库信息量的大小和使用频率已成为衡量这个国家信息化发达程度的重要标志之一。而信息化对于加快国家产业结构调整、促进经济增长和提高人们生活质量具有明显的倍增效应和带动作用。

1.1.1 数据与数据处理

人们在现实中进行的各种活动都会产生相应的信息。例如，生产毛绒玩具的工厂，其用于生产的原材料的名称、库存量、单价、产地；生产出来的产品的名称、数量、单价；该工厂中职工的职称、编号、薪水、奖金等，所有这些都是信息。这些信息代表了所属实体的特定属性或状态，当把这些信息以文字记录下来时便形成数据。因此可以说，数据就是信息的载体。本节主要介绍信息、数据和数据处理的概念。

1. 信息与数据

信息与数据是两个密切相关的概念。信息是各种数据所包含的意义，数据则是负载信息的物理符号。例如，某个人的年龄、某个考生的考试成绩、某年度的国民生产总值等，都是信息。如果将这些信息用文字或其他符号记录下来，那么，这些文字或符号就是数据。同一数据在不同的场合具有不同的意义。例如，56 这个数字，既可以表示一个人的年龄，也可以表示水的温度，或者表示某个考生某科目的考试成绩。在许多场合下，对信息和数据的概念并不作严格的区分，可互换使用。例如，通常所说的“信息处理”和“数据处理”，这两个概念的意义是相同的。

信息是对现实世界事物存在方式或运动状态的反映。它已成为人类社会活动的一种重要资源，与能源、物质并称为人类社会活动的三大要素。一般来说，信息是一种被加工成特定形式的数据，这种数据形式对接收者来说是有意义的，而且对当前和将来的决策具有明显的或实际的价值。它具有如下特征。

- 信息可以被感知，不同的信息源有不同的感知方式。
- 信息的获取和传递不仅需要有载体，而且还消耗能量。
- 信息可以通过载体进行存储、压缩、加工、传递、共享、扩散、再生和增值等。

数据是将现实世界中的各种信息记录下来的、可以识别的符号。它是信息的载体，是信息的具体表现形式。在计算机内部，所有的数据均采用 0 和 1 进行编码。在数据库技术中，数据的含义很广泛，除了数字之外，文字、图形、图像、声音和视频等也视为数据，它们分别表示不同类型的信息。

另外，同一种信息可以用多种不同的数据形式进行表达，而信息的意义不随数据的表现形式的改变而改变。例如，要表示某只股票每天的收盘价格，既可以通过绘制曲线图表示，也可以通过绘制柱状图表示，还可以通过表格数据进行表示，而无论使用何种方式来表示，丝毫不会改变信息的含义。

例如，对数据可以做如此定义，描述事物的符号记录称为数据。在学校的学生档案中，可以记录学生的姓名、性别、出生日期、所在系、电话号码和入学时间等。按这个次序排列组合成如下所示的一条记录。

(赵智暄，女，1986-01-10，心理系，13831706516，2006)

这条记录中的信息就是数据。当然，数据可能会因为记录介质被破坏而丢失。例如，记录在纸上的数据，可能因为纸介质丢失、火灾而造成数据丢失；记录在计算机磁盘上的数据，可

能因为病毒、误操作、火灾等造成数据丢失。

2. 数据与信息的关系

数据与信息有着不可分割的联系。信息是被加工处理过的数据，数据和信息的关系是一种原料和成品之间的关系，如图 1-1 所示。

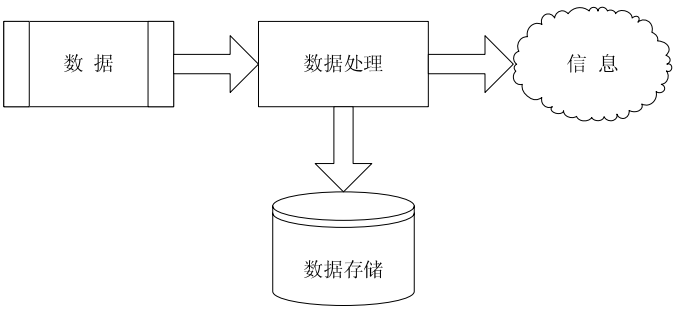


图 1-1　数据与信息的关系

数据和信息的关系主要表现在以下 4 个方面。

(1) 数据是信息的符号表示，或称载体。

(2) 信息是数据的内涵，是数据的语义解释。

(3) 数据是符号化的信息。

(4) 信息是语义化的数据。

3. 数据处理

数据处理是指对各种形式的数据进行收集、存储、加工和传播的一系列活动的总和。

进行数据处理的目的有两个：一是从大量的、原始的数据中抽取、整理出对人们有价值的信息，以作为行动和决策的依据；二是为了借助计算机，科学地保存和管理复杂的、大量的数据，以便人们能够方便而充分地利用这些宝贵的资源。

1.2　数据库

数据库(Database，简称 DB)就是数据的集合。例如，在日常生活中，人们用笔记本记录亲朋好友的联系方式，将他们的姓名、地址、电话等信息都记录下来。这个“通讯录”就是一个最简单的“数据库”，每个人的姓名、地址、电话等信息就是这个数据库中的“数据”。人们可以在笔记本这个“数据库”中添加新朋友的个人信息，由于某个朋友的电话变动也可以修改他的电话号码这个“数据”。使用笔记本这个“数据库”可以方便地查到某位亲朋好友的地址、邮编或电话号码等“数据”。

数据库是组织数据的一种手段，按照字面意思的理解，数据库就是存放数据的仓库。它是为了实现一定的目的按某种规则组织起来的“数据”的“集合”。在信息社会中，数据库的应用非常广泛，如银行业用数据库存储客户的信息、账户、贷款以及银行的交易记录；电信行业

用数据库存储用户的通话记录、套餐资费等信息。

在计算机领域，数据库是指长期存储在计算机内的、有组织的、可共享的、统一管理的相关数据的集合。

数据库中的数据不仅需要合理地存放，还要便于查找；数据库不仅可以供创建者本人使用，还可以供多个用户从不同的角度共享。即多个不同的用户可以根据不同的需求，使用不同的语言，同时存取数据库，甚至同时读取同一数据。

1.3 数据库技术的发展历程

从最早的商用计算机开始，数据处理就一直推动着计算机的发展。事实上，数据处理自动化早于计算机出现。Hollerith 发明的穿孔卡片，早在 20 世纪初就用来记录美国的人口普查数据，用机械系统来处理这些卡片并列出结果。穿孔卡片后来被广泛应用作为将数据输入计算机的一种手段。

按照年代来划分，数据库系统的发展可划分为以下几个阶段。

1. 20 世纪 50 年代至 60 年代早期

20 世纪 50 年代至 60 年代早期，磁带被用于数据存储。诸如工资单这样的数据处理已经自动化了，并且把数据存储在磁带上。数据处理包括从一个或多个磁盘上读取数据，并将数据写回到新的磁带上。数据也可以由一叠穿孔卡片输入，而输出到打印机上。例如，工资增长的处理是通过将增长表示到穿孔卡片上，在读入一叠穿孔卡片的同时要配有保存主要工资细节的磁带。工资的增加额将被加入到从主磁带读出的工资中，并被写到新的磁带上，新磁带将成为新的主磁带。

磁带(和卡片)都只能顺序读取，并且数据可以比内存大得多。因此，数据处理程序被迫用一种特定的顺序对来自磁带和卡片的数据进行读取和合并处理。

2. 20 世纪 60 年代末至 20 世纪 70 年代

20 世纪 60 年代末硬盘的广泛使用极大地改变了数据处理的情况，因为硬盘可以直接对数据进行访问。磁盘上数据的位置是无意义的，因为磁盘上的任何位置都可在几十毫秒内访问到，数据由此摆脱了顺序读取的限制。有了磁盘，就可以创建网状数据库和层次数据库，它们可以具有保存在磁盘上的如表和树等数据结构。程序员也可以创建和操作这些数据结构。

由 Codd 写的一篇具有里程碑意义的论文，定义了关系模型和在关系模型中用非过程化的方法来查询数据，关系数据库由此诞生。关系模型的简单性和能够对程序员隐藏所有细节的能力具有真正的诱惑力。

3. 20 世纪 80 年代

尽管关系模型在学术上很受重视，但是最初并没有实际的应用。因为它在性能上的不足，

所以关系型数据库在性能上还不能和当时已有的网状和层次数据库相提并论。这种情况直到 System R 的出现才得以改变，IBM 研究院的一个突破性项目开发了一种能够构造高效的关系型数据库系统的技术。Astrahan 和 Chamberlin 等人提供了很好的关于 System R 的综述。具有完全功能的 System R 原型诞生了 IBM 的第一个关系数据库产品 SQL/DS。最初的商用关系数据库系统，如 IBM 的 DB2、Oracle、Ingres 和 DEC 的 Rdb，在推动有效地处理陈述式查询技术上起到了主要作用。到了 20 世纪 80 年代早期，关系数据库已经可以在性能上和网状、层次数据库进行竞争了。关系数据库是如此简单易用，以至于最后它完全取代了网状和层次数据库。因为程序员在使用后者时，必须处理许多底层的实现问题，并且不得不将要做的查询任务编码成过程化的形式。更重要的是，在设计应用程序时还要时时考虑效率问题，而这需要付出很大的努力。相反，在关系数据库中，几乎所有的底层工作都由数据库自动来完成，使得程序员可以只考虑逻辑层的工作。因为关系模型在 20 世纪 80 年代已经取得了优势，所以它在数据模型中具有最高的统治地位。

另外，在 20 世纪 80 年代人们还对并行和分布式数据库进行了很多研究，同样在面向对象数据库方面也展开了初步的研究工作。

4. 20 世纪 90 年代初

SQL 语言主要是为了决策支持应用设计的，重在查询；而 20 世纪 80 年代主要的数据库是处理事务的应用，重在更新。决策支持和查询再度成为数据库的一个主要应用领域。分析大量数据的工具有了很大的发展。

在这个时期许多数据库厂商推出了并行数据库产品。数据库厂商还开始在其数据库中加入对象-关系的支持。

5. 20 世纪 90 年代末至今

随着互联网的兴起和发展，数据库比以前有了更加广泛的应用。现在数据库系统必须支持很高的事务处理速度，而且还要有很高的可靠性和 7×24 小时的可用性(一天 24 小时，一周 7 天都可用，也就是没有进行维护的停机时间)。另外，数据库系统还必须支持网络接口。

1.1.4　数据库系统

数据库系统是计算机化的记录保持系统，它的目的是存储和产生所需要的有用信息。这些有用的信息可以是使用该系统的个人或组织的有意义的任何事情，是对某个人或组织辅助决策过程中不可缺少的事情。

1. 数据库系统的组成

通常，一个数据库系统要包括以下 4 个主要部分：数据、用户、硬件和软件。

(1) 数据。数据是数据库系统的工作对象。为了区别输入、输出或中间数据，常把数据库

数据称为存储数据、工作数据或操作数据。它们是某特定应用环境中进行管理和决策所必需的信息。特定的应用环境，可以指一个公司、一个银行、一所医院和一所学校等。在这些应用环境中，各种不同的应用可通过访问其数据库获得必要的信息以辅助进行决策，决策完成后，再将决策结果存储在数据库中。

数据库中的存储数据是“集成的”和“共享的”。“集成”是指把某特定应用环境中的各种应用关联的数据及其数据间的联系全部集中地按照一定的结构形式进行存储，也就是把数据库看成若干个性质不同的数据文件的联合和统一的数据整体，并且在文件之间局部或全部消除了冗余，这使得数据库系统具有整体数据结构化和数据冗余小的特点；“共享”是指数据库中的数据块可以为多个不同的用户所共享，即多个不同的用户，使用多种不同的语言，为了不同的应用目的，而同时存取数据库，甚至同时存取同一数据块。共享实际上是基于数据库集成的。

(2) 用户。用户是指存储、维护和检索数据库中数据的人员。数据库系统中主要有 3 类用户：终端用户、应用程序员和数据库管理员。

- 终端用户：也称为最终用户，是指从计算机联机终端存储数据库的人员，还可以称为联机用户。这类用户使用数据库系统提供的终端命令语言、表格语言或菜单驱动等交互式对话方式来存取数据库中的数据。终端用户一般是不精通计算机和程序设计的各级管理人员、工程技术人员和各类科研人员。
- 应用程序员：也称为系统开发员，是指负责设计和编制应用程序的人员。这类用户通过设计和编写“使用及维护”数据库的应用程序来存取和维护数据库。这类用户通常使用 Access、SQL Server 或 Oracle 等数据库语言来设计和编写应用程序，以对数据库进行存取操作。
- 数据库管理员(DBA)：是指全面负责数据库系统的“管理、维护和正常使用”的人员，可以是一个人或一组人。而对于大型数据库系统来说，DBA 极为重要，通常设置有 DBA 办公室，应用程序员是 DBA 手下的工作人员。DBA 不仅要具有较高的技术专长，而且还要具备较深的资历，并具有了解和阐明管理要求的能力。DBA 的主要职责包括参与数据库设计的全过程；与用户、应用程序员、系统分析员紧密结合，设计数据库的结构和内容；决定数据库的存储和存取策略，使数据的存储空间利用率和存取效率均较优；定义数据的安全性和完整性；监督控制数据库的使用和运行，及时处理运行程序中出现的问题；改进和重新构建数据库系统等。

(3) 硬件。硬件是指存储数据库和运行数据库管理系统 DBMS 的硬件资源，包括物理存储数据库的磁盘、磁鼓、磁带或其他外存储器及其附属设备、控制器、I/O 通道、内存、CPU，以及外部设备等。数据库服务器的处理能力、存储能力、可靠性直接关系到整个系统的性能优劣，因此对服务器端硬件资源也有着较高的要求，应选用高可靠性、高可用性、高性价比的服务器。通常要求考虑以下问题。

- 具有足够大的内存，用于存放操作系统、DBMS 的核心模块、数据缓冲区和应用程序。
- 具有高速大容量的直接存取设备。一般数据库系统的数据量和数据的访问量都很大，因此需要容量大、速度快的存储系统存放数据，如采用高速大缓存硬盘，或者应用光

纤通道外接到外置的专用磁盘系统。

- 具有高速度 CPU，以拥有较短的系统响应时间。数据库服务器必须应对大量的查询并做出适当且及时的应答，因此要求处理能力强的 CPU 以满足较高的服务器处理速度和对客户的响应速率的要求。
- 有较高的数据传输能力，以提高数据传输率，保证足够的系统吞吐能力，否则，系统性能将形成瓶颈。
- 有足够的外存来进行数据备份。常配备磁盘阵列、磁带机或光盘机等存储设备。
- 高稳定性的系统。即数据库系统能够稳定持续运行，能提供长时间可靠稳定的服务。

(4) 软件。软件是指负责数据库存取、维护和管理的软件系统，通常叫作数据库管理系统(Database Management System，简称 DBMS)。数据库系统的各类用户对数据库的各种操作请求，都是由 DBMS 来完成的，它是数据库系统的核心软件。DBMS 提供一种超出硬件层之上的对数据库管理的功能，使数据库用户不受硬件层细节的影响。DBMS 是在操作系统支持下工作的。

2. 数据库系统的特点

数据库系统具有如下特点。

(1) 数据低冗余、共享性高。数据不再是面向某个应用程序，而是面向整个系统。当前所有用户可同时存取库中的数据，从而减少了数据冗余，节约存储空间，同时也避免了数据之间的不相容性和不一致性。

(2) 数据独立性提高。数据的独立性包括逻辑独立性和物理独立性。

- 数据的逻辑独立性是指当数据的总体逻辑结构改变时，数据的局部逻辑结构不变。由于应用程序是依据数据的局部逻辑结构编写的，所以，应用程序可不必修改，从而保证了数据与程序间的逻辑独立性。例如，在原有的记录类型之间增加新的联系，或在某些记录类型中增加新的数据项时，均可确保数据的逻辑独立性。
- 数据的物理独立性是指用户的应用程序与存储在磁盘上的数据库中数据是相互独立的，当数据的存储结构改变时，数据的逻辑结构不变，从而应用程序也不必改变。例如，改变存储设备和增加新的存储设备，或改变数据的存储组织方式，均可确保数据的物理独立性。

(3) 有统一的数据控制功能。数据库可以被多个用户所共享，当多个用户同时存取数据库中的数据时，为保证数据库中数据的正确性和有效性，数据库系统提供了以下 4 个方面的数据控制功能。

- 数据的安全性(security)控制：可防止不合法使用数据造成数据的泄露和破坏，保证数据的安全和机密。例如，系统提供口令检查或其他手段来验证用户身份，以防止非法用户使用系统；也可以对数据的存取权限进行限制，只有通过检查后才能执行相应的操作。
- 数据完整性(integrity)控制：系统通过设置一些完整性规则以确保数据的正确性、有效性和相容性。正确性是指数据的合法性，如代表年龄的整型数据，只能包含 0~9，不

能包含字母或特殊符号；有效性是指数据是否在其定义的有效范围内，如月份只能用1~12之间的数字来表示；相容性是指表示同一事实的两个数据应相同，否则就不相容，例如，一个人的性别不能既是男又是女。

- 并发(concurrency)控制：多用户同时存取或修改数据库时，防止相互干扰而提供给用户不正确的数据，并使数据库受到破坏。
- 数据恢复(recovery)：当数据库被破坏或数据不可靠时，系统有能力将数据库从错误状态恢复到最近某一时刻的正确状态。

1.1.5 数据库管理系统(DBMS)

数据库管理系统是位于用户和数据库之间的一个数据管理软件，它的主要任务是对数据库的建立、运用和维护进行统一管理、统一控制，即用户不能直接接触数据库，而只能通过DBMS来操纵数据库。

1. DBMS概述

数据库管理系统负责对数据库的存储进行管理、维护和使用。因此，DBMS是一种非常复杂的、综合性的、在数据库系统中对数据进行管理的大型系统软件。它是数据库系统的核心组成部分，在操作系统(OS)支持下工作。在确保数据安全可靠的同时，DBMS大大提高了用户使用“数据”的简明性和方便性，用户在数据库系统中的一切操作，包括数据定义、查询、更新和各种操作，都是通过DBMS完成的。

DBMS是数据库系统的核心部分，它把所有应用程序中使用的数据汇集在一起，并以记录为单位存储起来，便于应用程序查询和使用，如图1-2所示。

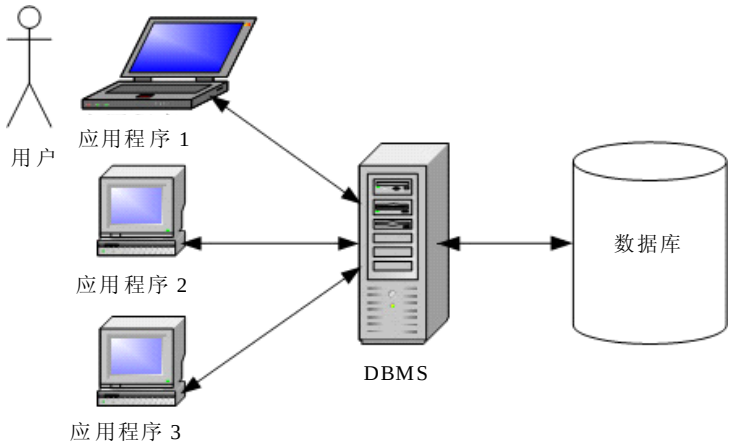


图1-2 DBMS、数据库及与用户之间的关系

常见的DBMS包括Access、Oracle、SQL Server、DB2、Sybase等。不同的数据库管理系统有不同的特点。Access相对于其他的一些数据库管理软件，如SQL Server、Oracle等来说，

操作相对简单，不需要用户具有高深的数据库知识，就能完成数据库所有的构造、检索、维护等功能。并且 Access 拥有简捷、美观的操作界面。

Access 属于小型桌面数据库管理系统，通常用于办公管理。它允许用户通过构建应用程序来收集数据，并可以通过多种方式对数据进行分类、筛选，将符合要求的数据供用户查看。用户可以通过显示在屏幕上的窗体来查看数据库中的数据，也可以通过报表将相关的数据打印出来，以便更详细地进行研究。

2. DBMS 的功能

DBMS 由于缺乏统一的标准，其性能、功能等许多方面随系统而异。一般情况下，大型系统功能比较强，小型系统功能较弱，同一类系统，性能也是有差异的。通常情况下，DBMS 提供了以下几个方面的功能。

- 数据库的建立和维护功能：包括数据库初始数据的装入，数据库的转储、恢复、重组织、系统性能监视、分析等功能。这些功能大都由 DBMS 的实用程序来完成。
- 数据库定义功能：DBMS 提供相应数据定义语言定义数据库结构，刻画数据库的框架，并被保存在数据字典中。数据字典是 DBMS 存取和管理数据的基本依据。
- 数据存取功能：DBMS 提供数据操纵语言实现对数据库数据的检索、插入、修改和删除等基本存取操作。
- 数据库运行管理功能：DBMS 提供数据控制功能，即数据的安全性、完整性和并发控制等，对数据库运行进行有效的控制和管理，以确保数据库数据正确有效和数据库系统的有效运行。
- 数据通信功能：DBMS 提供处理数据的传输功能，实现用户程序与 DBMS 之间的通信，这通常与操作系统协调完成。

3. DBMS 的组成

DBMS 大多是由许多系统程序所组成的一个集合。每个程序都有各自的功能，一个或几个程序一起协调完成 DBMS 的一件或几件工作任务。各种 DBMS 的组成因系统而异，一般来说，它由以下几个部分组成。

- 语言编译处理程序：主要包括数据描述语言翻译程序、数据操作语言处理程序、终端命令解释程序、数据库控制命令解释程序等。
- 系统运行控制程序：主要包括系统总控程序、存取控制程序、并发控制程序、完整性控制程序、保密性控制程序、数据存取与更新程序和通信控制程序等。
- 系统建立、维护程序：主要包括数据装入程序、数据库重组织程序、数据库系统恢复程序和性能监督程序等。
- 数据字典：通常是一系列表，它存储着数据库中有关信息的当前描述。它能帮助用户、数据库管理员和数据库管理系统本身使用和管理数据库。

1.1.6 数据库应用系统(DBAS)

数据库应用系统(Database Application System，简称 DBAS)是指在 DBMS 的基础上，针对一个实际问题开发出来的面向用户的系统。例如，网上银行就是一个数据库应用系统。用户通过登录网上银行，可以查询自己的账户余额，还可以进行转账汇款等操作。

1.2 数据库系统的体系结构

数据库系统的体系结构受数据库系统运行所在的计算机系统的影响很大，尤其是受计算机体系结构中的联网、并行和分布这些方面的影响。

- 计算机的联网可以使得某些任务在服务器系统上执行，而另一些任务在客户系统上执行。这种工作任务的划分引起了客户-服务器数据库系统的产生。
- 一个计算机系统中的并行处理能够加速数据库系统的活动，对事务做出更快速的响应，并且在单位时间中处理更多的事务。查询能够以一种充分利用计算机系统所提供的并行性的方式来处理。并行查询处理的需求引起了并行数据库系统的产生。
- 在一个组织机构的多个站点或部门间对数据进行分布，可以使得数据能够存放在产生它们或最需要它们的地方，而同时仍能被其他站点或其他部门访问。在不同站点上保存数据库的多个副本还使得大型组织机构甚至在一个站点受水灾、火灾、地震等自然灾害影响而遭到破坏的情况下仍能继续进行数据库操作。分布式数据库系统用来处理地理上或管理上分布在多个数据库系统中的数据。

本节将从传统的集中式系统结构开始介绍数据库系统的体系结构，包括客户-服务器系统、并行数据库系统和分布式数据库系统。

1.2.1 集中式体系结构

集中式数据库系统是运行在一台计算机上、不与其他计算机系统交互的数据库系统。这样的数据库系统范围很广，既包括运行在个人计算机上的单用户数据库系统，也包括运行在高端服务器系统上的高性能数据库系统。

现代通用的计算机系统包括一到多个 CPU，以及若干个设备控制器，它们通过公共总线连接在一起，但却提供对共享内存的访问，如图 1-3 所示。CPU 具有本地的高速缓冲存储器，用于存放存储器中部分数据的本地副本，从而加快对数据库的访问。每个设备控制器负责一种类型的设备(如磁盘驱动器、音频设备或视频播放设备)。CPU 和设备控制器可以并行地工作，它们竞争对于存储器的访问。

虽然当今通用的计算机系统有多个处理器，但它们具有粗粒度并行性，只有几个处理器(一般 2~4 个)，它们共享一个主存。在这样的机器上运行的数据库一般不试图将一个查询划分到多

个处理器上，而是在每个处理器上运行一个查询，从而使多个查询能并行地运行。因此，这样的系统能够提供较高的吞吐量，也就是说，尽管单个事务并没有运行得更快，但在每秒钟里能运行更多的事务了。

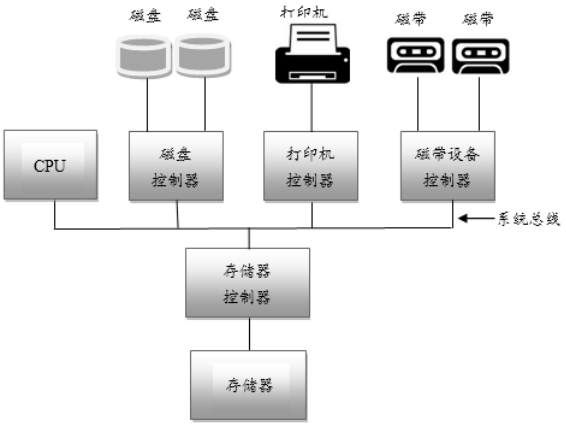


图 1-3　集中式计算机系统

1.2.2　客户-服务器结构

在客户-服务器结构中，数据库存放在服务器中，应用程序可以根据需要安排在服务器或客户工作站上，实现了客户端程序和服务器端程序的协同工作。这种结构解决了集中式结构和文件服务器结构的费用和性能问题。SQL Server 和 Oracle 都支持客户-服务器结构。

1.2.3　并行系统结构

并行系统提供并行地使用多个 CPU 和磁盘来提高处理速度和 I/O 速度。并行计算机正变得越来越普及，相应地使得并行数据库系统的研究变得更加重要。有些应用需要查询非常大型的数据库，有些应用需要在每秒钟处理很大数量的事务，这些应用的需求推动了并行数据库系统的发展。集中式数据库系统和客户-服务器数据库系统的能力不够强大，不足以处理这样的应用。

并行机器有若干种体系结构模式。如图 1-4 所示是其中比较重要的几种，图中的 M 表示存储器，P 表示处理器，圆柱体表示磁盘。

- 共享内存：所有的处理器共享一个公共的存储器。
- 共享磁盘：所有的处理器共享一组公共的磁盘，共享磁盘系统有时又称作群集。
- 无共享：各处理器既不共享公共的存储器，又不共享公共的磁盘。
- 层次的：这种模式是前几种体系结构的混合。

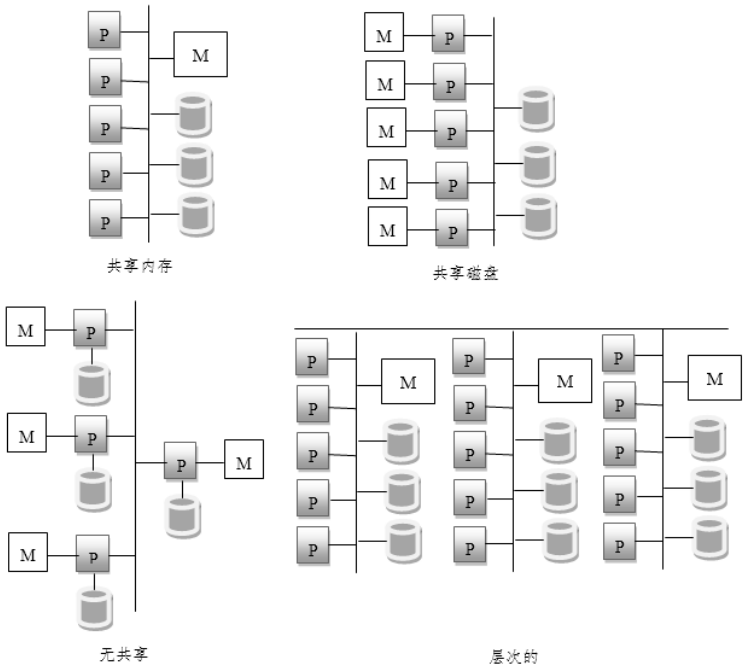


图 1-4　并行数据库体系结构

1.2.4　分布式系统结构

在分布式数据库系统中，数据库存储在几台计算机中。分布式系统中的计算机之间通过诸如高速网络或电话线等各种通信媒介互相通信。这些计算机不共享主存储器或磁盘。分布式系统中的计算机的规模和功能可大可小，小到工作站，大到大型机系统。

无共享并行数据库与分布式数据库之间的主要区别在于：分布式数据库一般是地理上分开的、分别管理的，并且是以较低的速度互相连接的。另一个主要的区别是：在分布式系统中，将事务区分为局部事务和全局事务。局部事务是仅访问在发起事务的站点上的数据的事务，而全局事务是需要访问发起事务的站点之外的某个站点或几个不同站点上的数据的事务。

建立分布式数据库系统有几个原因，主要包括共享数据、自治性和可用性。

- 共享数据：建立分布式数据库系统的主要优点就是它能提供一个环境，使一个站点上的用户可以访问存放在其他站点上的数据。例如，在一个分布式银行系统中，每一个支行存储与自己相关的数据，一个支行的用户可以访问另一个支行的数据。而如果没有这种能力，那么想要将资金从一个支行转到另一个支行的用户就必须求助于某种将已存在的系统相互关联起来的外部机制。
- 自治性：提供数据分布的方法来共享数据的主要优点在于，每个站点可以对局部存储的数据保持一定程度的控制。在集中式系统中，中心站点的数据库管理员对数据进行控制。在分布式系统中，每个站点的局部数据库管理员可以有不同程度的局部自治，

程度的大小依赖于分布式数据库系统的设计。

- 可用性：在分布式系统中，如果一个站点发生故障，其他的站点可能还能继续运行。特别是，如果数据项在几个站点上进行了备份，需要某一个特定数据项的事务可以在这几个站点中的任何一个上找到该数据项。

1.3　数据模型

计算机信息管理的对象是现实生活中的客观事物，但这些事物是无法直接送入计算机的，必须通过进一步整理和归类，进行信息的规范化，然后才能将规范信息数据化并送入计算机的数据库中保存起来。这一过程经历了 3 个领域——现实世界、信息世界和数据(机器)世界。

- 现实世界：存在于人脑之外的客观世界，包括事物和事物之间的联系。
- 信息世界：是现实世界在人们头脑中的反映。
- 数据(机器)世界：将信息世界中的实体进行数据化，事物和事物之间的联系用数据模型来描述。

在现实世界中，常常用模型来对某个对象进行抽象或描述，如飞机模型，它反映了该飞机的大小、外貌特征及其型号等；并可用文字语言来对该对象进行抽象或描述。

为了用计算机来处理现实世界的事物，首先需要将它们反映到人的大脑中，即首先需要把这些事务抽象为一种既不依赖于某一具体的计算机，又不受某一具体 DBMS 所左右的信息世界的概念模型，然后再把该概念模型转换为某一具体 DBMS 所支持的计算机世界的数据模型。

信息的 3 个世界及其关系如图 1-5 所示。

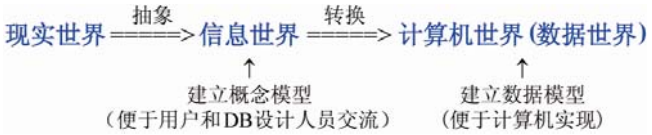


图 1-5　信息的 3 个世界及其关系

这个过程是通过研究“过程和对象”，然后建立相应的数据模型来实现的。在这两个转换过程中，需要建立两个模型即概念模型和逻辑数据模型。

1.3.1　概念模型

概念模型是对客观事物及其联系的抽象，用于信息世界的建模，它强调其语义表达能力，以及能够较方便、直接地表达应用中的各种语义知识。这类模型简单、清晰、易于被用户理解，是用户和数据库设计人员之间进行交流的语言。这种信息结构并不依赖于具体的计算机系统，不是某一个 DBMS 支持的数据模型，而是概念级的模型。

概念模型主要用来描述世界的概念化结构，它使数据库的设计人员在设计的初始阶段摆脱

计算机系统及 DBMS 的具体技术问题，集中精力分析数据以及数据之间的联系等，与具体的数据管理系统无关。概念数据模型必须换成逻辑数据模型，才能在 DBMS 中实现。

在概念模型中主要有以下几个基本术语。

1. 实体与实体集

实体是现实世界中可区别于其他对象的“事件”或物体。实体可以是人，也可以是物；可以指实际的对象，也可以指某些概念；还可以指事物与事物间的联系。例如，学生就是一个实体。

实体集是具有相同类型及共享相同性质(属性)的实体集合。例如，全班学生就是一个实体集。实体集不必互不相交。例如，可以定义学校所有学生的实体集 student 和所有教师的实体集 teacher，而一个 person 实体可以是 student 实体，也可以是 teacher 实体，甚至可能既是 student 实体又是 teacher 实体，也可以都不是。

2. 属性

实体通过一组属性来描述。属性是实体集中每个成员所具有的描述性性质。将一个属性赋予某实体集，表明数据库为实体集中每个实体存储相似信息，但每个实体在自己的相应属性上都有各自的值。一个实体可以由若干个属性来刻画，如学生实体包括学号、姓名、年龄、性别和班级等属性。

每个实体的每个属性都有一个值。例如，某个特定的 student 实体，其学号是 2013228649，姓名是许书伟，年龄是 22，性别是女。

3. 关键字和域

实体的某一属性或属性组合，其值能唯一标识出某一实体，称为关键字，也称码。例如，学号是学生实体集的关键字，由于姓名有相同的可能，故不能作为关键字。

每个属性都有一个可取值的集合，称为该属性的域，或者该属性的值集。例如，姓名的域为字符串集合，性别的域为“男”和“女”。

4. 联系

现实世界的事物之间总是存在某种联系，这种联系必然要在信息世界中加以反映。一般存在两种类型的联系：一是实体内部的联系，如组成实体的属性之间的联系；二是实体与实体之间的联系。

两个实体之间的联系又可以分为如下 3 类。

- 一对一联系(1:1)：例如，一个班级有一个班主任，而每个班主任只能在一个班任职。这样班级和班主任之间就具有一对一的联系。
- 一对多联系(1:n)：例如，一个班有多个学生，而每个学生只可以属于一个班，因此，在班级和学生之间就形成了一对多的联系。
- 多对多的联系(m:n)：例如，学校中的课程与学生之间就存在着多对多的联系。每个课

程可以供很多学生选修，而每个学生也可以选修多门课程。这种关系可以有很多种处理方法。

1.3.2　用 E-R 方法表示概念模型

概念模型的表示方法很多，其中最著名的是 E-R 方法(Entity-Relations，即实体-联系方法)，它用 E-R 图来描述现实世界的概念模型。E-R 图的主要成分是实体、联系和属性。E-R 图通用的表现规则如下。

- 矩形：表示实体集。
- 椭圆：表示属性。
- 菱形：用菱形表示实体间的联系，菱形框内写上联系名。用无向边分别把菱形与有关实体相连接，在无向边旁标上联系的类型。如果实体之间的联系也具有属性，则把属性和菱形也用无向边连上。
- 线段：将属性连接到实体集或将实体集连接到联系集。
- 双椭圆：表示多值属性。
- 虚椭圆：表示派生属性。
- 双线：表示一个实体全部参与到联系集中。
- 双矩形：表示弱实体集。

E-R 方法是抽象和描述现实世界的有力工具。用 E-R 图表示的概念模型与具体的 DBMS 所支持的数据模型独立，是各种数据模型的共同基础，因而比数据模型更一般、更抽象、更接近现实世界。

例如，要画出某个学校学生选课系统的 E-R 图，学校每学期开设若干课程供学生选择，每门课程可接受多个学生选修，每个学生可以选修多门课程，每门课程有一个教师讲授，每个教师可以讲授多门课程。

首先，确定实体集和联系。在本例中，可以将课程、学生和教师定义为实体，学生和课程之间是“选修”关系，课程和教师之间是“教授”关系。

接着，确定每个实体集的属性：【学生】实体的属性有学号、姓名、班级和性别；【课程】实体的属性有课程号、课程名和教科书；【教师】实体的属性有职工号、姓名和性别。在联系中反映出教师讲授的课程信息、每门课程上课的学生数以及学生选修的所有课程。最终得到的 E-R 图如图 1-6 所示。

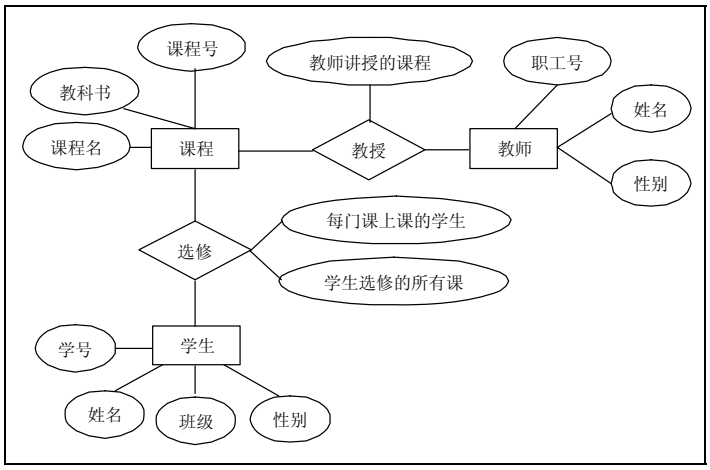


图 1-6　学生选课系统的 E-R 图

1.3.3　逻辑数据模型

数据库中的数据是结构化的，它是按某种数据模型来组织的。当前流行的基本数据模型有 3 类：关系模型、层次模型和网状模型。它们之间的根本区别在于数据之间联系的表示方式不同。关系模型用二维表来表示数据之间的联系；层次模型用树结构来表示数据之间的联系；网状模型用图结构来表示数据之间的联系。

层次模型和网状模型是早期的数据模型。通常把它们统称为格式化数据模型，因为它们是属于以“图论”为基础的表示方法。

按照这 3 类数据模型设计和实现的 DBMS 分别称为关系 DBMS、层次 DBMS 和网状 DBMS。相应地有关系(数据库)系统、层次(数据库)系统和网状(数据库)系统等简称。下面分别对这 3 种数据模型作一个简单的介绍。

1. 层次模型

层次数据模型是数据库系统最早使用的一种模型，它的数据结构是一棵有向树。层次结构模型具有如下特征。

- 有且仅有一个结点没有双亲，该结点是根结点。
- 其他结点有且仅有一个双亲。

在层次模型中，每个结点描述一个实体型，称为记录类型。一个记录类型可有许多记录值，简称记录。结点间的有向边表示记录之间的联系。如果要存取某一记录类型的记录，可以从根结点起，按照有向树层次逐层向下查找。查找路径就是存取路径。

层次模型结构清晰，各结点之间联系简单，只要知道每个结点的(除根结点以外)双亲结点，就可以得到整个模型结构。因此，画层次模型时可用无向边代替有向边。用层次模型模拟现实世界的层次结构的事物及其之间的联系是很自然的选择方式。例如，用层次模型表示“行政层

次结构”、“家族关系”等是很方便的。

层次模型的缺点是结构呆板，缺乏灵活性； 同一属性数据要存储多次，数据冗余大； 不适合于拓扑空间数据的组织；不能表示两个以上实体型之间的复杂联系和实体型之间的多对多联系。

美国 IBM 公司 1968 年研制成功的 IMS 数据库管理系统就是这种模型的典型代表。

2. 网状模型

如果取消层次模型的两个限制，即两个或两个以上的结点都可以有多个双亲，则“有向树”就变成了“有向图”。“有向图”结构描述了网状模型。网状模型具有如下特征。

- 可有一个以上的结点没有双亲。
- 至少有一个结点可以有多于一个双亲。

网状模型和层次模型在本质上是一样的。从逻辑上看，它们都是基本层次联系的集合，用结点表示实体，用有向边(箭头)表示实体间的联系；从物理上看，它们每一个节点都是一个存储记录，用链接指针来实现记录间的联系。当存储数据时这些指针就固定下来了，数据检索时必须考虑存取路径问题；数据更新时，涉及链接指针的调整，缺乏灵活性；系统扩充相当麻烦。网状模型中的指针更多，纵横交错，从而使数据结构更加复杂。

3. 关系模型

关系模型(Relational Model)是用二维表格结构来表示实体和实体之间联系的数据模型。关系模型的数据结构是一个“二维表框架”组成的集合，每个二维表又可称为关系。因此可以说，关系模型是“关系框架”组成的集合。

关系模型是使用最广泛的数据模型，目前大多数数据库管理系统都是关系型的，本书要介绍的 Access 就是一种关系数据库管理系统。

例如：对于某校学生、课程和成绩的管理，要用到如表 1-1 至表 1-3 所示的几个表格。如果要找到学生“栾鹏”的“高等数学”成绩，首先需在学生信息表中找到【姓名】为“栾鹏”的记录，记下他的学号 201021112，如表 1-1 所示。

表 1-1　学生信息表

学　号	姓　名	性　别	年　龄	院 系 ID	联 系 电 话
982111056	葛冰	女	29	9001	13831705804
201400021	赵智暄	女	13	9002	15910806516
201021112	栾鹏	男	35	7482	13681187162
201020202	李二丹	女	22	1801	—
201231008	闫伟峰	男	30	7012	13582107162

再到课程表中找到【课程名称】为“高等数学”的【课程号】1003，如表 1-2 所示。

表 1-2　课程表

课 程 号	课 程 名 称	学　　分	教 师 ID
1001	经济学原理	3	91001
1002	变态心理学	4	61001
1003	高等数学	6	81002

接着到成绩表中查找【课程号】为 1003，【学号】为 201021112 的对应成绩值，如表 1-3 所示。

表 1-3　学生成绩表

课 程 号	学　　号	成　　绩
1001	982111056	91
1003	201021112	61
1003	944114044	52
1001	981000021	82

通过上面的例子可以看出，关系模型中数据的逻辑结构就是一张二维表，它由行和列组成。一张二维表对应了一个关系，表中的一行即为一条记录，表中的一列即为记录的一个属性。

关系模型的优点是：结构特别灵活，满足所有布尔逻辑运算和数学运算规则形成的查询要求；能搜索、组合和比较不同类型的数据；增加和删除数据非常方便。

其缺点是：数据库大时，查找满足特定关系的数据较费时；对空间关系无法满足。

1.4　关系数据库

关系数据库是当今世界的主流数据库。本节主要介绍关系模型中的一些基本术语，以及关系模型的完整性约束。

在关系模型中，基本的数据结构是二维表，不是像层次或网状那样的链接指针。记录之间的联系是通过不同关系中的同名属性来体现的。例如，要查找某个教师讲授的课程，首先要在【教师】关系中根据【姓名】 找到对应的教师【编号】，然后根据【编号】的值在【课程】关系中找到对应的【课程名】即可。在查询过程中，同名属性教师【编号】起到了连接两个关系的纽带作用。由此可见，关系模型中的各个关系模式不应当孤立起来，不是随意拼凑的一堆二维表，它必须满足相应的要求。

1.4.1　关系模型中的基本术语

关系模型中经常用到的术语如下。

1. 关系

一个关系就是一张二维表。

2. 元组

二维表中的每一条记录就是一个元组，它是构成关系的一个个实体。可以说，“关系”是“元组”的集合，“元组”是属性值的集合，一个关系模型中的数据就是这样逐行逐列组织起来的。

3. 属性

二维表中的一列就是一个属性，又称为字段，第一行列出的是属性名(字段名)。

4. 域

域表示属性的取值范围。例如，【性别】属性只能取值为“男”或“女”。

5. 分量

分量是元组中的一个属性值。关系模型要求关系必须是规范化的，最基本的条件就是关系的每一个分量必须是一个不可分的数据项，即不允许表中还有表。

6. 关系描述

对关系的描述，一般表示如下。

关系名(属性 1，属性 2，……，属性 n)

例如，可以将学生关系描述如下。

学生(学号，姓名，性别，出生年月，籍贯，院系编号)

7. 候选关键字

关系中的一个或几个属性的集合，该属性集唯一标识一个元组，这个属性集合称为候选关键字。

8. 关系数据库

对应于一个关系模型的所有关系的集合称为关系数据库。

9. 主关键字

一个关系中有多个候选关键字，可以选择其中一个作为主关键字，也称为主码或主键。

10. 外关键字

如果一个属性组不是所在关系的关键字，但它是其他关系的关键字，则该属性组称为外关键字，也称为外码或外键。

11. 主属性

包含在任一候选关键字中的属性称为主属性，不包含在任何候选关键字中的属性称为非关键字属性。

例如，描述院系的关系模式如下。

院系(院系编号，院系名称)

其主键为【院系编号】，所以【学生】关系中的【院系编号】字段就是外键。

关系是一个二维表，但并不是所有的二维表都是关系。关系应该具有如下性质。

- 每一列中的分量是同一类型的数据。
- 不同的列要给予不同的属性名。
- 列的次序可以任意交换。
- 一个关系中的任意两个元组不能完全相同。
- 行的次序可以任意交换。
- 每一个分量必须是不可分的数据项。

1.4.2 关系数据库中表之间的关系

在关系数据库中，可以通过外部关键字来实现表与表之间的联系，公共字段是一个表的主键和另一个表的外键。如图 1-7 所示的【学生】表和【院系】表都包含【院系编号】属性，通过这个字段就可以在【院系】表和【学生】表之间建立联系，这个联系是一对多的联系，即一个院系中有多个学生。

学号	姓名	性别	出生年月	籍贯	院系编号
20060101	洪伟	女	1982-1-1	广西	01
20060102	许诺	女	1984-1-1	北京	02
20060103	丁丁	女	1983-1-1	江西	01
20060104	赵建	男	1978-1-1	河北	02

院系代码	院系名称
01	数学科学学院
02	计算机技术学院

图 1-7 【学生】表和【院系】表之间的联系

关系数据库中表之间的关系类型主要有三种：一对一、一对多、多对多。

1. 一对一

表之间的一对一关系意味着，对于第一个表中的每条记录，第二个表中有且仅有一条对应的记录。纯粹的一对一关系在关系数据库中并不常见。在大多数情况下，第二个表中包含的数据也包括在第一个表中。事实上，一般情况下会避免使用一对一关系，因为它违反了规范化理论。在规范化理论中，如果数据描述的是单个实体，则不应该将其拆分到多个表中。

一对一关系的一种常见情况是，数据在多个数据库之间传输或共享。

2. 一对多

一对多关系是关系数据库中比较常见的关系类型。在一对多关系中，第一个表(父表)中的每条记录与第二个表(子表)中的一条或多条记录相关。第二个表中的每条记录仅与第一个表中的一条记录相关。

3. 多对多

在多对多关系中，两个表中的每条记录可以与另一个表中的零条、一条或多条记录相关。例如，在学生选课系统中，每个学生可以选择多门课程，而每门课程也可以由多个学生选择。

1.4.3　关系模型的完整性约束

关系模型的完整性规则是对关系的某种约束条件，也就是说，关系的值随着时间变化应该满足一些约束条件。这些约束条件实际上是现实世界的要求。任何关系在任何时刻都要满足这些语义约束。

关系模型中有 3 类完整性约束：实体完整性、参照完整性和用户定义的完整性。其中，实体完整性和参照完整性是关系模型必须满足的完整性约束条件，被称作关系的两个不变性，应该由关系系统自动支持。用户定义的完整性是应用领域需要遵循的约束条件，体现了具体领域中的语义约束。

1. 实体完整性(Entity Integrity)

实体完整性规则为：如果属性(指一个或一组属性)A 是基本关系 R 的主属性，则 A 不能取空值。所谓空值就是“不知道”或“不存在”的值。例如，在【学生】关系中，【学号】这个属性为主键，则该字段不能取空值。

按照实体完整性规则的规定，基本关系的主键都不能取空值。如果主键由若干属性组成，则所有这些主属性都不能取空值。

对于实体完整性规则说明如下。

(1) 实体完整性规则是针对基本关系而言的。一个基本表通常对应现实世界的一个实体集。例如，【学生】关系对应于学生的集合。

(2) 现实世界中的实体是可区分的，即它们具有某种唯一性标识。例如，每个学生都是独

立的个体，是不一样的。

(3) 关系模型中以主键作为唯一性标识。

(4) 主键中的属性即主属性不能取空值。如果主属性取空值，就说明存在某个不可标识的实体，即存在不可区分的实体。这与(2)相矛盾，因此这个规则称为实体完整性。

2. 参照完整性(Referential Integrity)

参照完整性规则为：如果属性(或属性组)F 是基本关系 R 的外键，它与基本关系 S 的主键 Ks 相对应(基本关系 R 和 S 不一定是不同的关系)，则对于 R 中每个元组在 F 上的值必须为空或是等于 S 中某个元组的主键值。

现实世界中的实体之间往往存在某种联系，在关系模型中，实体和实体之间的联系都是用关系来描述的，这样就自然存在着关系和关系间的引用。例如，图 1-7 中的【学生】表和【院系】表，【学生】表中每条学生记录的【院系编号】在【院系】表中必须存在，即学生所属的院系必须是该学校中已存在的院系。

提示

除了不同关系之间存在参照完整性之外，同一个关系的内部也可能存在参照完整性。

3. 用户定义的完整性(User-defined Integrity)

任何关系数据库系统都应该支持实体完整性和参照完整性。这是关系模型所要求的。除此之外，不同的关系数据库系统根据其应用环境的不同，往往还需要一些特殊的约束条件。用户定义的完整性就是针对某一具体关系数据库的约束条件。它反映某一具体应用所涉及的数据必须满足的语义要求。例如，某个属性必须取唯一值、某个非主属性不能取空值、某个属性的取值范围在 0~100 之间(如学生的成绩)等。

关系模型应提供定义和检验这类完整性的机制，以便用统一的、系统的方法处理它们，而不要由应用程序承担这一功能。

满足完整性约束的关系数据库中的数据表之间不但具有独立性，而且若干个表之间又有一定的相关性，这一特点使其具有极大的优越性，主要表现在以下几点。

- 使数据具有最小的冗余度，支持复杂的数据结构。
- 数据具有共享性，能为多个用户服务。
- 具有高度的数据和程序的独立性，应用程序与数据的逻辑结构和数据的物理存储方式无关。

1.5 关系代数

关系代数是一种抽象的查询语言，它用关系的运算来表达查询。

任何一种运算都是将一定的运算符作用于一定的运算对象之上，从而得到预期的结果。所

以运算对象、运算符和运算结果是运算的 3 大要素。

关系代数的运算对象是关系，运算结果也是关系。关系代数用到的运算符包括 4 类，即集合运算符、专门的关系运算符、比较运算符和逻辑运算符，如表 1-4 所示。

表 1-4　关系代数用到的运算符

运 算 符		含　义	运 算 符		含　义
集合运算符	∪	并	比较运算符	>	大于
	−	差		<	小于
	∩	交		<>	不等于
	×	笛卡尔积		>=	大于等于
				<=	小于等于
专门的关系运算符	σ	选择	逻辑运算符	¬	非
	Π	投影		∧	与
	÷	除		∨	或
	⋈	连接			

按照运算符的不同，可以将关系代数的运算分为传统的集合运算和专门的关系运算两大类。其中，传统的集合运算将关系看成是元组的集合，其运算是从关系的“水平”方向即行的角度来进行的；而专门的关系运算同时涉及行和列。比较运算符和逻辑运算符则是用来辅助专门的关系运算符进行操作的。

关于关系代数的理论，在这仅作简单介绍，详细信息请参考专门的数据库理论书籍。

1.5.1　传统的集合运算

传统的集合运算都是二目运算，包括并、差、交和笛卡尔积这 4 种运算。

设关系 R 和关系 S 都具有 n 个属性，且相应的属性取自同一个域，t 是元组变量，$t \in R$ 表示 t 是 R 的一个元组，如图 1-8 所示。

R

A	B	C
a1	b1	c1
a1	b2	c2
a2	b2	c1

S

A	B	C
a1	b2	c2
a1	b3	c2
a2	b2	c1

图 1-8　关系 R 和关系 S

可以定义并、差、交、笛卡尔积运算如下。

1. 并

关系 R 和关系 S 的并运算记作如下形式。

$R \cup S = \{t \mid t \in R \vee t \in S\}$

其结果关系仍具有 n 个属性，由属于 R 或属于 S 的元组组成，结果如图 1-9 所示。

2. 差

关系 R 和关系 S 的差记作如下形式。

$R - S = \{t \mid t \in R \wedge t \notin S\}$

其结果关系仍具有 n 个属性，由属于 R 而不属于 S 的所有元组组成，结果如图 1-10 所示。

A	B	C
a1	b1	c1
a1	b2	c2
a2	b2	c1
a1	b3	c2

图 1-9　$R \cup S$

A	B	C
a1	b1	c1

图 1-10　R-S

3. 交

关系 R 和关系 S 的交记作如下形式。

$R \cap S = \{t \mid t \in R \wedge t \in S\}$

其结果关系仍具有 n 个属性，由既属于 R 又属于 S 的元组组成。关系的交也可用差来表示，即 $R \cap S = R - (R - S)$，结果如图 1-11 所示。

4. 笛卡尔积

严格地讲，在这里的笛卡尔积应该是广义的笛卡尔积，因为这里的笛卡尔积的元素是元组。

两个分别具有 n 和 m 个属性的关系 R 和 S 的笛卡尔积是一个(n+m)列的元组的集合。元组的前 n 列是关系 R 的一个元组，后 m 列是关系 S 的一个元组。若 R 有 $k1$ 个元组，S 有 $k2$ 个元组，则关系 R 和关系 S 的笛卡尔积有 $k1 \times k2$ 元组，记作如下形式。

$R \times S = \{t_r t_s \mid t_r \in R \wedge t_s \in S\}$

结果如图 1-12 所示。

A	B	C
a1	b2	c2
a2	b2	c1

图 1-11　$R \cap S$

R.A	R.B	R.C	S.A	S.B	S.C
a1	b1	c1	a1	b2	c2
a1	b1	c1	a1	b3	c2
a1	b1	c1	a2	b2	c1
a1	b2	c2	a1	b2	c2
a1	b2	c2	a1	b3	c2
a1	b2	c2	a2	b2	c1
a2	b2	c1	a1	b2	c2
a2	b2	c1	a1	b3	c2
a2	b2	c1	a2	b2	c1

图 1-12　$R \times S$

1.5.2　专门的关系运算

专门的关系运算包括选择、投影、连接、除运算等。下面简单介绍这些运算。

1. 选择

从一个关系中选出满足给定条件的记录的操作称为选择或筛选。选择运算是从行的角度进行的运算，选出满足条件的那些记录构成原关系的一个子集。其中，条件表达式中可以使用=、<>、>=、>、<和<=等比较运算符，多个条件之间可以使用 AND(∧)、OR(∨)和 NOT(¬)进行连接。选择操作记作如下形式。

$$\sigma_F(R)=\{t \mid t \in R \wedge F(t)='真'\}$$

其中，F 表示选择条件。

假设对于表 1-1 所示的学生信息表 Student，如果要查询年龄小于 25 岁的学生可以表示为 $\sigma_{年龄<25}(Student)$ 或 $\sigma_{4<25}(Student)$。

运算的结果为学生信息表 Student 中所有年龄小于 25 岁的记录。这里的 4 表示 Student 表的第 4 列。

2. 投影

从一个关系中选出若干指定字段的值的操作称为投影。投影是从列的角度进行的运算，所得到的字段个数通常比原关系少，或者字段的排列顺序不同。

投影操作记作如下形式。

$$\pi_A(R)=\{t[A] \mid t \in R\}$$

其中，A 为 R 中的属性列。

例如，查询学生的【姓名】和【联系电话】的操作如下。

$$\pi_{姓名,联系电话}(\text{Student}) \text{ 或 } \pi_{2,6}(\text{Student})$$

运算的结果为【姓名】和【联系电话】两列，以及这两列对应的所有数据组成的关系。

投影之后得到的关系不仅取消了原关系中的某些列，而且还可能取消原关系中的某些元组。因为取消了某些列之后，就可能出现重复行，应取消这些完全相同的行。

3. 连接

连接是把两个关系中的记录按一定条件横向结合，生成一个新的关系。最常用的连接运算是自然连接，它是利用两个关系中公用的字段，把该字段值相等的记录连接起来。

需要明确的是，选择和投影都属于单目运算。它们的操作对象只是一个关系，而连接则是双目运算，其操作对象是两个关系。

连接操作记作如下形式。

$$R \underset{A\theta B}{\infty} S=\{t_r t_s \mid t_r \in R \wedge t_s \in S \wedge t_r[A]=t_s[B]\}$$

系统在执行连接运算时，要进行大量的比较操作。不同关系中的公共字段或具有相同语义的字段是实现连接运算的“纽带”。例如，【学生信息表】和【学生成绩表】可以通过“Student.学号”和“Score.学号”作为连接的“纽带”。

4. 除

给定关系 $R(X,Y)$和 $S(Y,Z)$，其中 X、Y、Z 为属性组。R 中的 Y 和 S 中的 Y 可以有不同的属性名，但必须出自相同的域集。

那么 R 和 S 的除运算得到一个新的关系 $P(X)$，P 是 R 中满足下列条件的元组在 X 属性列上的投影：元组在 X 上分量值 x 的象集 Y_x 包含 S 在 Y 上投影的集合。

除运算记作如下形式。

$$R \div S = \{t_r[X] \mid t_r \in R \wedge \pi_y(S) \subseteq Y_x\}$$

其中，Y_x 为 x 在 R 中的象集，x=tr[X]。

提示

除操作是同时从行和列角度进行运算的。

1.6 规范化理论

为了使数据库设计的方法趋于完善，人们研究了规范化理论。目前，规范化理论的研究已经有了很大的发展。本节将主要介绍模式规范化在数据库设计过程中的必要性，及其规范化原理。

1.6.1 模式规范化的必要性

一般而言，关系数据库设计的目标是生成一组关系模式，使用户既无须存储不必要的重复信息，又可以方便地获取信息。方法之一是设计满足适当范式的模式。在学习范式前，首先来了解非规范化的表格。

当一个关系中的所有字段都是不可分割的数据项时，称该关系是规范化的。但是，当表格中有一个字段具有组合数据项时，即为不规范化的表，如图 1-13 所示。

编号	店名	季度营业额			
		第一季度	第二季度	第三季度	第四季度

图 1-13　字段含有组合数据项的不规范化表格

当表格中含有多值数据项时，该表格同样为不规范化的表格，如图 1-14 所示。

教师编号	教师姓名	年龄	教授科目
101	郑颖	35	数据挖掘
102	王海泉	36	计算机网络 网络协议与安全

← 多值数据

图 1-14　多值数据项的不规范化表格

6.2　规范化理论的基本概念

满足一定条件的关系模式称为范式(Normal Form，简称 NF)。在 1971 年至 1972 年，关系数据模型的创始人 E.F.Codd 系统地提出了第一范式(1NF)、第二范式(2NF)和第三范式(3NF)的概念。1974 年 Codd 和 Boyce 共同提出了 BCNF 范式，为第三范式的改进。一个低级范式的关系模式，通过投影分解的方法可转换成多个高一级范式的关系模式的集合，这个过程称为规范化。

1. 第一范式(1NF)

第一范式是最低的规范化要求，它要求关系满足一种最基本的条件。它与其他的范式不同，不需要诸如函数依赖之类的额外信息。

如果某个域的元素被认为是不可分的单元，那么这个域就是原子的(atomic)；如果一个关系模式的所有熟悉的域都是原子的，则称此关系模式属于第一范式(1NF)。

第一范式要求数据表不能存在重复的记录，即存在一个关键字，第二个要求是每个字段都已经分到最小不再可分，关系数据库的定义就决定了数据库满足这一条。主关键字应满足下面几个条件。

- 主关键字在表中是唯一的。
- 主关键字段不存在空值。
- 每条记录都必须有一个主关键字。
- 主关键字是关键字的最小子集。

满足第一范式的关系模式有许多不必要的重复值，并且增加了修改数据时疏漏的可能性，为了避免这种数据冗余和更新数据的疏漏，就引出了第二范式。

2. 第二范式(2NF)

如果一个关系属于第一范式(1NF)，且所有的非主关键字段都完全依赖于主关键字，则称为第二范式。

举个例子来说，有一个存储物品的关系有 5 个字段(物品 ID、仓库号、物品名称、物品数量、仓库地址)，这符合 1NF。其中，【物品 ID】和【仓库号】构成主关键字，但因为【仓库地址】只依赖于【仓库号】，即只依赖于主关键字的一部分，所以它不符合第二范式(2NF)。这样首先存在数据冗余，因为仓库数量可能不多；其次，在更改仓库地址时，如果漏改了某一条记录，则存在数据不一致性。最后，如果某个仓库的物品全部出库了，那么这个仓库地址就会丢失，所以这种关系不允许存在某个仓库中不放物品的情况。可以用投影分解的方法消除部分依赖的情况，

从而达到 2NF 的标准。方法是从关系中分解出新的二维表，使得每个二维表中所有的非关键字都完全依赖于各自的主关键字。可以作分解，将原来的一个表分解成两个表，如下。

物品(物品 ID，仓库号，物品名称，物品数量)

仓库(仓库号，仓库地址)

这样就完全符合第二范式(2NF)了。

3. 第三范式(3NF)

如果一个关系属于第二范式(2NF)，且每个非主关键字不传递依赖于主关键字，这种关系就是第三范式(3NF)。简而言之，从 2NF 中消除传递依赖，就是 3NF。例如，有一个关系(姓名，工资等级，工资额)，其中姓名是关键字，此关系符合 2NF，但是因为工资等级决定工资额，这就叫传递依赖，它不符合 3NF。同样可以使用投影分解的方法将上表分解成两个表：(姓名，工资等级)和(工资等级，工资额)。

上面提到了投影分解的方法，关系模式的规范化过程是通过投影分解来实现的。这种把低一级关系模式分解成若干个高一级关系模式的投影分解方法不是唯一的，应该在分解中满足 3 个条件。

- 无损连接分解，分解后不丢失信息。
- 分解后得到的每个关系都是高一级范式，不要同级甚至低级分解。
- 分解的个数最少，这就是完美要求，应该做到尽量少。

如图 1-15 所示满足第二范式，但是该关系中的字段仍然存在较高的数据冗余。

员工编号	姓名	部门编号	部门名称	办公室
1	黄鹂	XS	销售部	101
2	周音	XS	销售部	101
3	赵鹏程	CW	财务部	106
4	李厉	SH	策划部	103
5	张小城	XZ	行政部	105

图 1-15　满足第二范式的关系模式

转换为第三范式后的关系模式如图 1-16 所示。

员工编号	姓名	部门编号
1	黄鹂	XS
2	周音	XS
3	赵鹏程	CW
4	李厉	SH
5	张小城	XZ

部门编号	部门名称	办公室
XS	销售部	101
CW	财务部	106
SH	策划部	103
XZ	行政部	105

图 1-16　展开成第三范式关系

一般情况下，规范化到 3NF 就满足需要了，规范化程度更高的还有 BCNF、4NF、5NF。

规范化的基本思想是逐步消除数据依赖中不合适的部分，使模式中的各种关系模式达到某种程度的“分离”，即“一事一地”的模式设计原则。让一个关系描述一个概念、一个实体或者实体间的一种联系。如果多于一个概念，就把它分离出去。因此所谓规范化实质上是概念的单一化。

应该指出的是，规范化的优点是明显的，它避免了大量的数据冗余，节省了空间，保持了数据的一致性。如果完全达到 3NF，用户不会在超过两个以上的地方更改同一个值。而当记录会经常发生改变时，这个优点便很容易显现出来。但是，它最大的不利是，由于用户把信息放置在不同的表中，增加了操作的难度，同时把多个表连接在一起的时间花费也是巨大的，节省了时间必然付出了空间的代价；反之，节省了空间也必然要付出时间的代价，时间和空间在计算机领域中是一个矛盾统一体，它们是互相作用、对立统一的。因为表和表的连接操作的时间花费是很大的，从而降低了系统运行性能。

4. 第四范式(4NF)

第四范式的定义用到了多值依赖，多值依赖的定义如下：设 $R(U)$是属性集 U 上的一个关系模式。X、Y、Z 是 U 的子集，并且 $Z=U-X-Y$。关系模式 $R(U)$中多值依赖 $X\rightarrow\rightarrow Y$ 成立，当且仅当对 $R(U)$的任一关系 r，给定的一对(x, z)值有一组 Y 的值，这组值仅仅决定于 x 值而与 z 值无关。

若 $X\rightarrow\rightarrow Y$，而 $Z=\Phi$即 Z 为空，则称 $X\rightarrow\rightarrow Y$ 为平凡的多值依赖。

多值依赖具有以下性质。

- 多值依赖具有对称性，即若 $X\rightarrow\rightarrow Y$，则 $X\rightarrow\rightarrow Z$，其中 $Z=U-X-Y$。
- 多值依赖的传递性，即若 $X\rightarrow\rightarrow Y$，$Y\rightarrow\rightarrow Z$，则 $X\rightarrow\rightarrow Z-Y$。

函数依赖和多值依赖集为 D 的关系模式 $R(U)$，属于第四范式(4NF)的条件是对所有 D+中形如 $X\rightarrow\rightarrow Y$ 的多值依赖(X、Y 是 U 的子集)，至少有以下条件之一成立。

- $X\rightarrow\rightarrow Y$ 是一个平凡的多值依赖。
- X 是 R 的超码。

5. 其他范式

第四范式不是“最终”范式，正如前面提到的，多值依赖有助于理解并解决利用函数依赖无法理解的某些形式的信息重复。还有一些类型的概括多值依赖的约束称为连接依赖(Join Dependence)，由此引出的另外一种范式称为投影-连接范式(Project-Join Normal Form，简称 PJNF)，有的书中也将其称为第五范式(5NF)。

使用这些通用约束的一个实际问题是，难以用于推导。而且还没有形成一套具有保真性和完备性的推理规则用于约束的推导，因此很少被使用。

提示

有关范式的更多信息请读者参考相关书籍，本书不作过多介绍。

1.7 数据库语言

数据库系统提供两种不同类型的语言：一种是数据定义语言，用于定义数据库模式；另一种是数据操纵语言，用于表达数据库的查询和更新。而实际上，数据定义和数据操纵语言并不是两种分离的语言。相反，它们构成了单一的数据库语言，如广泛使用的 SQL 语言。

1.7.1 数据定义语言 DDL

数据库模式是通过一系列定义来说明的，这些定义由一种称为数据定义语言(Data-Definiton Language，DDL)的特殊语言来表达。例如，下面的 SQL 语句描述了 Student 表的定义。

```
Create table Student
(sno varchar(10),
sname varchar(50),
ssex varchar(4),
sage   integer,
sdeptno integer,
stelephone varchar(20))
```


1.7.2 数据操纵语言 DML

数据操纵语言(Data-Manipulation Language，简称 DML)使得用户可以访问或操纵那些按照某种特定数据模式组织起来的数据。数据操纵包括对存储在数据库中的信息进行检索，向数据库中插入新的信息，从数据库中删除信息和修改数据库中存储的信息。

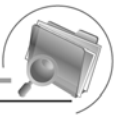
通常有以下两种基本的数据操纵语言。

- 过程化 DML：要求指定需要什么数据以及如何获得这些数据。
- 陈述式 DML：也称非过程化 DML，只要求用户指定需要什么数据，而不指明如何获得这些数据。

通常陈述式 DML 比过程化 DML 更易学易用。但是，由于不必指明如何获得数据，因此数据库系统会指出一种访问数据的高效路径。SQL 语言的 DML 部分是非过程化的。

查询是要求对信息进行检索的语句。DML 中涉及信息检索的部分称为查询语句。例如，下面的语句将从 Student 表中查询名为“赵智暄”的用户信息。

```
SELECT * FROM Student WHERE sname= '赵智暄';
```



1.8　数据库设计

数据库设计是指对于一个给定的应用系统，构造(设计)优化的数据库逻辑模式和物理结构。并据此建立数据库及其应用系统，使之能够有效地存储和管理数据。满足各种用户的应用需求，包括信息管理要求和数据操作要求。

- 信息管理要求是指在数据库中应该存储和管理哪些数据对象；
- 数据操作要求是指对数据对象需要进行哪些操作，如查询、增加、删除、修改和统计等操作。

1.8.1　数据库设计的目标

数据库设计的目标是为用户和各种应用系统提供一个信息基础设施和高效率的运行环境。高效率的运行环境包括数据库数据的存取效率、数据库存储空间的利用率，以及数据库系统运行管理的效率等。

数据库设计的过程是数据库应用系统从设计、实施到运行与维护的全过程。

1.8.2　数据库设计的特点

数据库设计和一般的软件系统的设计、开发和运行与维护有许多相同之处，更有其自身的一些特点。

1. 数据库建设的基本规律

“三分技术，七分管理，十二分基础数据”是数据库设计的特点之一。

在数据库建设中，不仅涉及技术，还涉及管理。要建设好一个数据库应用系统，开发技术固然重要，但是相比之下管理则更加重要。这里的管理不仅包括数据库建设作为一个大型的工程项目本身的项目管理，而且还包括该企业的业务管理。

“十二分基础数据”则强调了数据的收集、整理、组织和不断更新是数据库建设中的重要环节。人们往往忽视基础数据在数据库建设中的地位和作用。基础数据的收集、入库是数据库建立初期工作量最大、最烦琐、最细致的工作。在以后数据库运行过程中更需要不断地把新的数据加入到数据库中，使数据库成为一个“活库”，否则就成为“死库”。数据库一旦成了“死库”，系统也就失去了应用价值，原来的投资也就失败了。

2. 结构(数据)设计和行为(处理)设计相结合

数据库设计应该和应用系统相结合。也就是说，整个设计过程中要把数据库结构设计和对数据的处理设计密切结合起来。这是数据库设计的特点之二。

在早期的数据库应用系统开发过程中，常把数据库设计和应用系统的设计分离开来。由于数据库设计有它专门的技术和理论，因此，需要专门来讲解数据库设计。但这并不等于数据库设计和在数据库之上开发应用系统是相互分离的。相反，必须强调设计过程中数据库设计和应用系统的密切结合，并把它作为数据库设计的重要特点。

1 8.3 数据库设计的方法

大型数据库设计是涉及多学科的综合性技术，同时又是一项庞大的工程项目。它要求从事数据库设计的专业人员具备多方面的技术和知识。主要包括：计算机的基础知识、软件工程的原理和方法、程序设计的方法和技巧、数据库的基本知识、数据库设计技术、应用领域的知识等。这样才能设计出符合具体领域要求的数据库及其应用系统。

早期数据库设计主要采用手工与经验相结合的方法。设计的质量往往与设计人员的经验与水平有直接的关系。数据库设计是一种技艺，缺乏科学理论和工程方法的支持，设计质量难以保证。因此，人们努力探索，提出了各种数据库设计方法。其中，比较著名的有以下 4 种。

- 新奥尔良(New Orleans)方法：该方法把数据库设计分为若干阶段和步骤，并采用一些辅助手段实现每一过程。它运用软件工程的思想，按一定的设计规程用工程化方法设计数据库。新奥尔良方法属于规范化设计法。虽然从本质上看，它仍然是手工设计方法，但其基本思想是过程迭代和逐步求精。
- 基于 E-R 模型的数据库设计方法：该方法用 E-R 模型来设计数据库的概念模型，是数据库概念设计阶段广泛采用的方法。
- 3NF(第三范式)设计方法：该方法用关系数据理论为指导来设计数据库的逻辑模型，是设计关系数据库时在逻辑阶段可以采用的一种有效方法。
- ODL(Object Definition Language)方法：这是面向对象的数据库设计方法。该方法用面向对象的概念和术语来说明数据库结构。ODL 可以描述面向对象的数据库结构设计，可以直接转换为面向对象的数据库。

1 8.4 数据库设计的步骤

数据库设计是指对于一个给定的应用环境，构造最优的数据库模式，建立数据库及其应用系统，使之能够有效地存储数据，满足各种用户的应用需求。

数据库设计一般分为以下 6 个步骤。

1. 需求分析

进行数据库设计首先必须准确了解与分析用户需求，包括数据和处理。需求分析是整个设计过程的基础，是最困难、最耗时的一步。作为“地基”的需求分析是否做得充分与准确，决

定了在其上构建数据库大厦的速度与质量。需求分析做得不好，可能会导致整个数据库设计返工重做。

2. 概念结构设计

概念结构设计是整个数据库设计的关键，它通过对用户需求进行综合、归纳与抽象，形成一个独立于具体 DBMS 的概念模型。

概念模型是整个组织各个用户关心的信息结构。描述概念结构的有力工具是 E-R 图。数据库设计通常基于 E-R 模型来进行，然后转化成关系模型。

3. 逻辑结构设计

逻辑结构设计将概念结构转换为某个 DBMS 所支持的数据模型，并对其进行优化。

4. 物理结构设计

物理设计为逻辑数据模型选取一个最适合应用环境的物理结构，包括存储结构和存取方法等。物理结构设计通常分为以下两步。

(1) 确定数据库的物理结构：可分为确定数据的存取方法和数据的存储结构。

(2) 对物理结构进行评估：包括对时间效率、空间效率、维护开销和各种用户要求进行权衡，从多种设计方案中选择一个较优的方案。

5. 数据库实施

在数据库实施阶段，设计人员运用 DBMS 提供的数据库语言(如 SQL)及其宿主语言，根据逻辑设计和物理设计的结果建立数据库，编制与调试应用程序，组织数据入库，并进行调试运行。

(1) 定义数据库结构。确定了数据库的逻辑结构与物理结构后，就可以用所选用的 DBMS 提供的数据定义语言(DDL)来严格描述数据库结构了。

(2) 数据装载。数据库结构建立后，就可以向数据库中装载数据了。组织数据入库是数据库实施阶段最主要的工作。对于数据量不是很大的小型系统，可以用人工方式完成数据的入库，具体包括如下几个步骤。

① 筛选数据：需要装入数据库中的数据通常都分散在各个部门的数据文件或原始凭证中，所以首先必须把需要入库的数据筛选出来。

② 转换数据格式：筛选出来的需要入库的数据，其格式往往不符合数据库要求，还需要进行转换。这种转换有时可能很复杂。

③ 输入数据：将转换好的数据输入到计算机中。

④ 校验数据：检查输入的数据是否有误。

对于中大型系统，由于数据量大，用人工方式组织数据入库将会耗费大量人力物力，而且很难保证数据的准确性。因此，应设计一个数据输入子系统，由计算机辅助进行数据入库。

(3) 编制与调试应用程序。数据库应用程序的设计应该与数据设计并行进行。在数据库实施阶段，当数据库结构建立好后，就可以开始编制与调试数据库的应用程序。也就是说，编制与调试应用程序是与组织数据入库同步进行的。调试应用程序时，由于数据入库尚未完成，可先使用模拟数据。

(4) 数据库试运行。应用程序调试完成，并且已有少部分数据入库后，就可以开展数据库的试运行。数据库试运行也称为联合调试，其主要工作如下。

- 功能测试：即实际运行应用程序，执行对数据库的各种操作，测试应用程序的各种功能。
- 性能测试：即测量系统的性能指标，分析是否符合设计目标。

6. 数据库运行和维护

数据库应用系统经过试运行后即可投入正式运行。数据库投入运行标志着开发任务的基本完成和维护工作的开始，并不意味着设计过程的终结。由于应用环境在不断变化，数据库运行过程中物理存储也会不断变化，对数据库设计进行评价、调整、修改等维护工作是一个长期的任务，也是设计工作的继续和提高。

在数据库运行阶段，对数据库经常性的维护工作主要由 DBA 完成，包括以下内容。

(1) 数据库的转储和恢复。定期对数据库和日志文件进行备份，以保证一旦发生故障，能利用数据库备份及日志文件备份，尽快将数据库恢复到某种一致性状态，并尽可能减少对数据库的破坏。

(2) 数据库的安全性、完整性控制。DBA 必须对数据库安全性和完整性控制负起责任。根据用户的实际需要授予不同的操作权限。另外，由于应用环境的变化，数据库的完整性约束条件也会变化，也需要 DBA 不断修正，以满足用户要求。

(3) 数据库性能的监督、分析和改进。目前许多 DBMS 产品都提供了监测系统性能参数的工具，DBA 可以利用这些工具方便地得到系统运行过程中一系列性能参数的值。DBA 应该仔细分析这些数据，通过调整某些参数来进一步改进数据库性能。

(4) 数据库的重组和重构造。数据库运行一段时间后，由于记录不断被增、删、改，会使数据库的物理存储变坏，从而降低数据库存储空间的利用率和数据的存取效率，使数据库的性能下降。这时，DBA 就要对数据库进行重组织，或部分重组织(只对频繁增、删的表进行重组织)。数据库的重组织不会改变原设计的数据逻辑结构和物理结构，只是按原设计要求重新安排存储位置，回收垃圾，减少指针链，提高系统性能。DBMS 一般都提供了供重组数据库使用的实用程序，帮助 DBA 重新组织数据库。

数据库应用环境发生变化，会导致实体及实体间的联系也发生相应的变化。使原有的数据库设计不能很好地满足新的需求，从而不得不适当调整数据库的模式和内模式。这就是数据库的重构造。DBMS 都提供了修改数据库结构的功能。

重构造数据库的程度是有限的。如果应用变化太大，已无法通过重构数据库来满足新的需求；或重构数据库的代价太大时，则表明现有数据库应用系统的生命周期已经结束，应该重新设计新的数据库系统，开始新数据库应用系统的生命周期。

提示

设计一个完善的数据库应用系统是不可一蹴而就的，往往是上述的 6 个阶段的不断反复。

1.9　上机练习

本章的上机练习将设计一个网上购物系统的概念模型，并用 E-R 图表示各实体之间的关系。

(1) 在网上购物系统中包括如下实体：商品、购物车、用户等。

(2) 实体之间的关系如下：购物车中可以有多件商品，每类商品也可以被添加到不同的购物车中。购物车属于某个用户，每个用户可以购物多次，即产生多个购物车。

(3) 确定每个实体的属性：【商品】实体的属性有【商品编号】、【商品名称】、【类别】、【单价】、【库存数量】和【生产日期】；【购物车】实体的属性有【购物车编号】、【用户编号】和【购物车状态】；【用户】实体的属性有【用户编号】、【姓名】、【地址】、【生日】和【联系电话】。

(4) 最终得到的 E-R 图如图 1-17 所示。

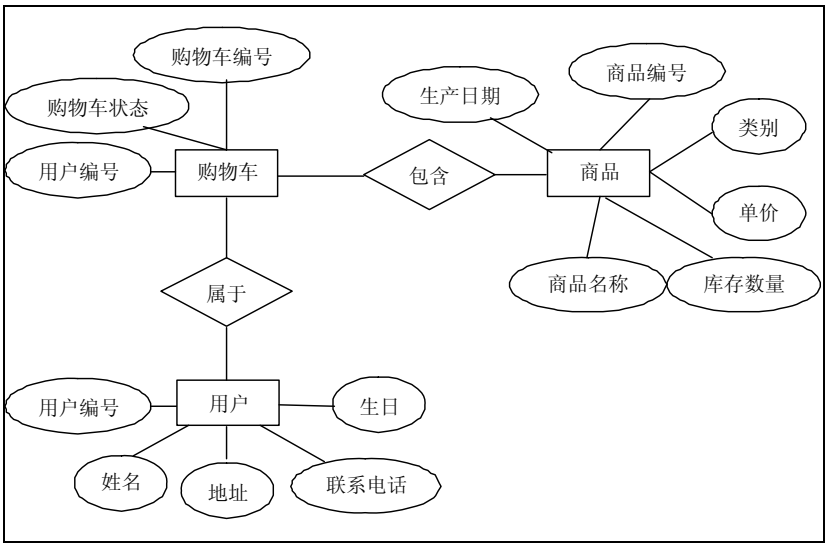


图 1-17　网上购物系统的 E-R 图

1.10 习题

1. 什么是数据库？什么是数据库系统？
2. 什么是数据库管理系统？它有哪些主要功能？
3. 常用的数据模型有哪几种？
4. 什么是关系模型？它是如何表示实体和实体之间的联系的？
5. 常用的关系运算有哪些？如何区分一元运算和二元运算？
6. 为什么要进行关系模式规范化？
7. 第三范式与第二范式相比有哪些改进？
8. 什么是数据操纵语言？它有什么作用？
9. 简述数据库设计的步骤。