目录

第 Ⅰ 篇 基础篇

本篇介绍数据库的基本概念和基础知识，它是读者即进一步学习后续章节的基础。本篇由下列 7 章组成：

第 1 章，数据库概述。介绍了文件管理数据与数据库管理数据的本质区别，数据独立性的含义以及数据库系统的组成。

第 2 章，数据模型与数据库系统体系结构。介绍了数据库技术发展过程中所使用过的数据模型，数据独立性的概念。本章介绍的知识是读者进一步学习后边的关系数据库及相关知识的基础。

第 3 章，关系数据库。介绍了关系数据库采用的数据模型的特点，同时介绍了关系数据库基于的理论基础——关系代数和关系演算。读者在学习完本章和第 5 章的数据操作语句之后，可以对关系代数、关系演算、SQL 查询语句之间的功能及表达方法进行比较。本章介绍的关系代数也是学习第 14 章查询优化的基础。

第 4 章，SQL 语言基础及数据定义功能。在 SQL 语言部分介绍了常用的数据类型，由于不同的数据库管理系统提供的数据类型不完全相同，因此本章主要介绍的是 SQL Server 数据库管理系统提供的数据类型，这部分内容是定义关系表的基础。在数据定义功能部分介绍了架构和基本表的概念和定义语句，同时介绍了数据完整性约束的定义方法。

第 5 章介绍了 SQL 的数据操作语句。主要包括查询、添加、删除和更改数据的 SQL 语句，同时介绍了一些高级查询功能，包括 CASE 表达式、嵌套子查询和相关子查询等。这章使用第 4 章建立的数据表，运用实际的数据，通过描述问题的分析思路以及用图示的方法展示查询语句的执行结果，使读者能够准确理解和掌握查询语句的功能。

第 6 章，索引和视图。在索引部分，除了介绍索引的概念的定义方法外，还用图示的方法详细讲述了索引的构建过程以及利用索引的查找过程，使读者能够从系统内部了解索引的作用。在视图部分，介绍了视图的概念和定义语句，并简单介绍了物化视图的概念和作用。

第 7 章，触发器和存储过程。触发器用于实现复杂的完整性约束和业务规则，本章介绍了触发器的概念和使用方法。存储过程是一段封装好的代码块，这个代码块可供应用程序调用使用，存储过程提供了代码共享的功能。

第 1 章 数据库概述

数据库是管理数据的一种技术，现在数据库技术已经被广泛应用到我们日常生活中的方方面面。本章首先介绍数据管理技术发展的过程，然后介绍使用数据库技术管理数据的特点和好处。

1.1 概述

随着信息管理水平的不断提高， 应用范围的日益扩大，信息已成为企业的重要财富和资源，同时，作为管理信息的数据库技术也得到了很大的发展，其应用领域也越来越广泛。人们在不知不觉中扩展着对数据库的使用，比如信用卡购物，飞机、火车订票系统、商场的进销存、图书馆对书籍及借阅的管理等，无一不使用了数据库技术。从小型事务处理到大型信息系统，从联机事务处理到联机分析处理，从一般企业管理到计算机辅助设计与制造（CAD/CAM）、地理信息系统等，数据库技术已经渗透到我们日常生活中的方方面面，数据库中信息量的大小以及使用的程度已经成为衡量企业的信息化程度的重要标志。

数据库是数据管理的最新技术，其主要研究内容是如何对数据进行科学的管理，以提供可共享、安全、可靠的数据。数据库技术一般包含数据管理和数据处理两部分。

数据库系统本质上是一个用计算机存储数据的系统，数据库本身可以看作是一个电子文件 柜，但它的功能不仅仅只是保存数据，而且还提供了对数据进行各种管理和处理的功能，比如安全管理、数据共享的管理、数据查询处理等等。

本章将介绍数据库的基本概念，包括数据管理的发展过程、数据库系统的组成等。读者可从本章了解为什么要学习数据库技术，并为后续章节的学习做好准备。

1.2 一些基本概念

在系统地介绍数据库技术之前，首先介绍数据库中最常用的一些术语和基本概念。

一． 数据（Data）

数据是数据库中存储的基本对象。早期的计算机系统主要应用于科学计算领域，处理的数据基本是数值型数据，因此数据在人们头脑中的直觉反应就是数字，但数字只是数据的一种最简单的形式，是对数据的传统和狭义的理解。目前计算机的应用范围已十分广泛，因此数据种类也更加丰富，比如，文本、图形、图像、音频、视频、商品销售情况等都是数据。

可以将数据定义为：数据是描述事物的符号记录。描述事物的符号可以是数字，也可以是文字、图形、图像、声音、语言等，数据有多种表现形式，它们都可以经过数字化后保存在计算机中。

数据的表现形式并不一定能完全表达其内容，有些还需要经过解释才能明确其表达的含义，比如 20，当解释其代表人的年龄时就是 20 岁，当解释其代表商品价格时，就是 20 元。因此，数据和数据的解释是不可分的。数据的解释是对数据演绎的说明，数据的含义称为数据的语义。因此数据和数据的语义是不可分的。

在日常生活中，人们一般直接用自然语言来描述事物，例如描述一门课程的信息：数据库系统基础，4 个学分，第 5 学期开设。但在计算机中经常按如下形式描述：

（数据库系统基础，4，5）

即把课程名、学分、开课学期信息组织在一起，形成一个记录，这个记录就是描述课程的数据。这样的数据是有结构的。记录是计算机表示和存储数据的一种格式或方法。

二．数据库（Database，简称 DB）

数据库，顾名思义，就是存放数据的仓库，只是这个仓库是存储在计算机存储设备上的，而且是按一定的格式存储的。

人们在收集并抽取出一个应用所需要的大量数据之后，就希望将这些数据保存起来，以供进一步从中得到有价值的信息，并进行相应的加工和处理。在科学技术飞速发展的今天，人们对数据的需求越来越多，数据量也越来越大。最早人们把数据存放在文件柜里，现在人们可以借助计算机和数据库技术来科学地保存和管理大量的复杂数据，以便能方便而充分地利用宝贵的数据资源。

严格地讲，数据库是长期存储在计算机中的有组织的、可共享的大量数据的集合。数据库中的数据按一定的数据模型组织、描述和存储，具有较小的数据冗余、较高的数据独立性和易扩展性，并可为多种用户共享。

概况起来，数据库数据具有永久存储、有组织和可共享三个基本特点。

三．数据库管理系统（Database Management System，简称 DBMS）

在了解了数据和数据库的基本概念之后，下一个需要了解的就是如何科学有效地组织和存储数据，如何从大量的数据中快速地获得所需的数据以及如何对数据进行维护，这些都是数据库管理系统要完成的任务。数据库管理系统是一个专门用于实现对数据进行管理和维护的系统软件。

数据库管理系统位于用户应用程序与操作系统软件之间，如图 1-1 所示。数据库管理系统与操作系统一样都是计算机的基础软件，同时也是一个非常复杂的大型系统软件，其主要功能包括如下几个方面：

图 1-1 数据库管理系统在计算机系统中的位置

1．数据库的建立与维护功能

包括创建数据库及对数据库空间的维护，数据库的转储与恢复功能，数据库的重组功能，数据库的性能监视与调整功能等。这些功能一般是通过数据库管理系统中提供的一些实用工具实现的。

2．数据定义功能

包括定义数据库中的对象，比如表、视图、存储过程等。这些功能的实现一般是通过数据库管理系统提供的数据定义语言（Data Definition Language，DDL）实现的。

3．数据组织、存储和管理功能

为提高数据的存取效率，数据库管理系统需要对数据进行分类存储和管理。数据库中的数据包括数据字典、用户数据和存取路径数据等。数据库管理系统要确定这些数据的存储结构、存取方式以及存储位置，以及如何实现数据之间的关联。确定数据的组织和存储的主要目的是提高存储空间利用率和存取效率。一般的数据库管理系统都会根据数据的具体组织和存储方式提供多种数据存取方法，比如索引查找、Hash 查找、顺序查找等。

4．数据操作功能

包括对数据库数据的查询、插入、删除和更改操作，这些操作一般是通过数据库管理系统提供的数据操作语言（Data Manipulation Language，DML）实现的。

5．事务的管理和运行功能

数据库中的数据是可供多个用户同时使用的共享数据，为保证数据能够安全、可靠地运

行，数据库管理系统提供了事务管理功能，这些功能保证数据能够并发使用并且不会产生相互干扰的情况，而且在数据库发生故障时能够对数据库进行正确的恢复。

6. 其他功能

包括与其他软件的网络通讯功能、不同数据库管理系统间的数据传输以及互访问功能等。

四．数据库系统（Database System，DBS）

数据库系统是指在计算机中引入数据库后的系统，一般由数据库、数据库管理系统（及相关的实用工具）、应用程序、数据库管理员组成。为保证数据库中的数据能够正常、高效地运行，除了数据库管理系统软件之外，还需要一个（或一些）专门人员来对数据库进行维护，这个专门人员就称为数据库管理员（Database Administrator，DBA）。我们将在 1.5 节详细介绍数据库系统的组成。

一般在不引起混淆的情况下，常常把数据库系统简称为数据库。

1.3 数据管理技术的发展

数据库技术是应数据管理任务的需要而产生和发展的。数据管理是指对数据进行分类、组织、编码、存储、检索和维护，它是数据处理的核心，而数据处理则是指对各种数据的收集、存储、加工和传播等一系列活动的总和。

自计算机产生之后，人们就希望用它来帮助我们对数据进行存储和管理。最初对数据的管理是以文件方式进行的，也就是用户通过编写应用程序来实现对数据的存储和管理。后来，随着数据量越来越大，人们对数据的要求越来越多，希望达到的目的也越来越复杂，文件管理方式已经很难满足人们对数据的需求，由此产生了数据库技术，也就是用数据库来存储和管理数据。数据管理技术的发展因此也就经历了文件管理和数据库管理两个阶段。

本节将介绍文件管理和数据库管理在管理数据上的主要差别。

1.3.1 文件管理

理解今日数据库特征的最好办法是了解在数据库技术产生之前，人们是如何通过文件的方式对数据进行管理的。

20 世纪 50 年代后期到 60 年代中期，计算机的硬件方面已经有了磁盘等直接存取的存储设备，软件方面，操作系统中已经有了专门的数据管理软件，一般称为文件管理系统。文件管理系统把数据组织成相互独立的数据文件，利用“按文件名访问，按记录进行存取”的管理技术，可以对文件中的数据进行修改、插入和删除等操作。

在出现程序设计语言之后，开发人员不但可以创建自己的文件并将数据保存在自己定义的文件中，而且还可以编写应用程序来处理文件中的数据，即编写应用程序来定义文件的结构，实现对文件内容的插入、删除、修改和查询操作。当然，真正实现磁盘文件的物理存取操作的还是操作系统中的文件管理系统，应用程序只是告诉文件管理系统对哪个文件的哪些数据进行哪些操作。我们将由开发人员定义存储数据的文件及文件结构，并借助文件管理系统的功能编写访问这些文件的应用程序，以实现对用户数据的处理的方式称为**文件管理**，在本章后面的讨论中，为描述简单我们将忽略操作系统中的文件管理系统，假定应用程序是直接对磁盘文件进行操作。

用户通过编写应用程序来管理存储在自定义文件中的数据的操作模式如图 1-2 所示。

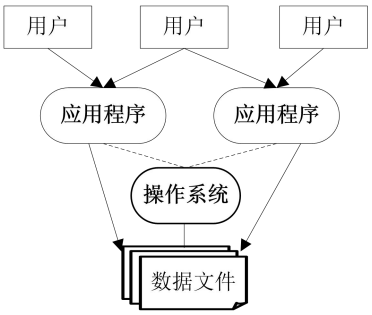


图 1-2　用文件存储数据的操作模式

假设某学校要用文件的方式保存学生及其选课的数据，并针对这些数据文件构建对学生及选课情况进行管理的系统。此系统主要实现两部分功能：学生基本信息管理和学生选课情况管理。假设教务部门管理学生选课情况，各系管理自己的学生基本信息。学生基本信息管理只涉及学生的基本信息数据，假设这些数据保存在 F1 文件中；学生选课情况管理涉及学生的部分基本信息、课程基本信息和学生选课信息，假设文件 F2 和 F3 分别保存课程基本信息和学生选课信息的数据。

设 A1 为实现“学生基本信息管理”功能的应用程序，A2 为实现“学生选课管理”功能的应用程序。图 1-3 所示为用文件存储并管理数据的两个系统的实现示例（图中省略了操作系统部分）。

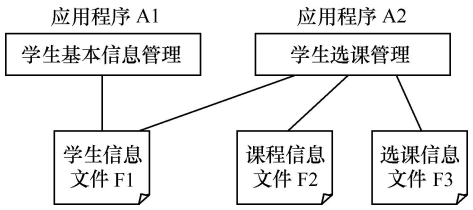


图 1-3　用文件存储数据的实现示例

假设文件 F1、F2 和 F3 分别包含如下信息：

F1 文件：　学号、姓名、性别、出生日期、联系电话、所在系、专业、班号。

F2 文件：　课程号、课程名、授课学期、学分、课程性质。

F3 文件：　学号、姓名、所在系、专业、课程号、课程名、修课类型、修课时间、考试成绩。

我们将文件中所包含的每一个子项称为文件结构中的“字段”或“列”，将每一行数据称为一个“记录”。

“学生选课管理”的处理过程大致为：在学生选课管理中，若有学生选课，则先查 F1 文件，判断有无此学生；若有则再访问 F2 文件，判断其所选的课程是否存在；若一切符合规则，就将学生选课信息写到 F3 文件中。

这看似很好，但仔细分析一下，就会发现用文件方式管理数据有如下缺点。

（1）编写应用程序不方便。应用程序编写者必须清楚地了解所用文件的逻辑及物理结构，如文件中包含多少个字段，每个字段的数据类型，采用何种逻辑结构和物理存储结构。操作系统只提供了打开、关闭、读、写等几个底层的文件操作命令，而对文件的查询、修改等操作都必须在应用程序中编程实现。这样就容易造成各应用程序在功能上的重复，比如图 1-3 中的“学生基本信息管理”和“学生选课管理”都要对 F1 文件进行操作，而共享这两个功能相同的操作却很难。构。

（2）数据冗余不可避免。由于 A2 应用程序需要在学生选课信息文件（F3 文件）中包含学生的一些基本信息，比如学号、姓名、所在系、专业等，而这些信息同样包含在学生信息文件（F1 文件）中，因此 F3 文件和 F1 文件中存在重复数据，从而造成数据的重复，称为数据冗余。

数据冗余所带来的问题不仅仅是存储空间的浪费（其实，随着计算机硬件技术的飞速发展，存储容量不断扩大，空间问题已经不是我们关注的主要问题），更为严重的是造成了数据的不一致（inconsistency）。例如，某个学生所学的专业发生了变化，我们一般只会想到

在 F1 文件中进行修改，而往往忘记了在 F3 文件中应做同样的修改。由此就造成了同一名学生在 F1 文件和 F3 文件中的“专业”不一样，也就是数据不一致。当发生数据不一致时，人们不能判定哪个数据是正确的，尤其是当系统中存在多处数据冗余时，更是如此。这样数据就失去了其可信性。

文件本身并不具备维护数据一致性的功能，这些功能完全要由用户（应用程序开发者）负责维护。这在简单的系统中还可以勉强应对，但在复杂的系统中，若让应用程序开发者来保证数据的一致性，几乎是不可能的。

（3）应用程序依赖性。就文件管理而言，应用程序对数据的操作依赖于存储数据的文件的结构。定义文件和记录的结构通常是应用程序代码的一部分，如 C 程序的 struct。文件结构的每一次修改，比如添加字段、删除字段，甚至修改字段的长度（如电话号码从 7 位扩到 8 位），都将导致应用程序的修改，因为在打开文件进行数据读取时，必须将文件记录中不同字段的值对应到应用程序的变量中。随着应用环境和需求的变化，修改文件的结构不可避免，这些都需要在应用程序中做相应的修改，而（频繁）修改应用程序是很麻烦的。人们首先要熟悉原有程序，修改后还需要对程序进行测试、安装等；甚至修改了文件的存储位置或者文件名，也需要对应用程序进行修改，这显然给程序的维护带来很多麻烦。

所有这些都是由于应用程序对文件的结构以及文件的物理特性过分依赖造成的，换句话说，用文件管理数据时，其数据独立性（data independence）很差。

（4）不支持对文件的并发访问。在现代计算机系统中，为了有效利用计算机资源，一般都允许同时运行多个应用程序（尤其是在现在的多任务操作系统环境中）。文件最初是作为程序的附属数据出现的，它一般不支持多个应用程序同时对同一个文件进行访问。回忆一下，某个用户打开了一个 Word 文件，当第二个用户在第一个用户未关闭此文件前打开此文件时，会得到什么信息呢？他只能以只读方式打开此文件，而不能在第一个用户打开的同时对此文件进行修改。再回忆一下，如果用某种程序设计语言编写一个对某文件中内容进行修改的程序，其过程是先以写的方式打开文件，然后修改其内容，最后再关闭文件。在关闭文件之前，不管是在其他的程序中，还是在同一个程序中都不允许再次打开此文件，这就是文件管理方式不支持并发访问的含义。

对于以数据为中心的系统来说，必须要支持多个用户对数据的并发访问，否则就不会有我们现在这么多的火车或飞机的订票点，也不会有这么多的银行营业网点。

（5）数据间联系弱。当用文件管理数据时，文件与文件之间是彼此独立、毫不相干的，文件之间的联系必须通过程序来实现。比如对上述的 F1 文件和 F3 文件，F3 文件中的学号、姓名等学生的基本信息必须是 F1 文件中已经存在的（即选课的学生必须是已经存在的学生）；同样，F3 文件中的课程号等与课程有关的基本信息也必须存在于 F2 文件中（即学生选的课程也必须是已经存在的课程）。这些数据之间的联系是实际应用当中所要求的很自然的联系，但文件本身不具备自动实现这些联系的功能，我们必须通过编写应用程序，即手工地建立这些联系。这不但增加了编写代码的工作量和复杂度，而且当联系很复杂时，也难以保证其正确性。因此，用文件管理数据时很难反映现实世界事物间客观存在的联系。

（6）难以满足不同用户对数据的需求。不同的用户（数据使用者）关注的数据往往不同。例如，对于学生基本信息，对负责分配学生宿舍的部门可能只关心学生的学号、姓名、性别和班号，而对教务部门可能关心的是学号、姓名、所在系和专业。

若多个不同用户希望看到的是学生的不同基本信息，那么就需要为每个用户建立一个文件，这势必造成很多的数据冗余。我们希望的是，用户关心哪些信息就为他生成哪些信息，对用户不关心的数据将其屏蔽，使用户感觉不到其他信息的存在。

可能还会有一些用户，其所需要的信息来自于多个不同的文件，例如，假设各班班主任关心的是：班号、学号、姓名、课程名、学分、考试成绩等。这些信息涉及了三个文件：从

F1 文件中得到“班号”，从 F2 文件中得到“学分”，从 F3 文件中得到“考试成绩”；而“学号”、“姓名”可以从 F1 文件或 F3 文件中得到，“课程名”可以从 F2 文件或 F3 文件中得到。在生成结果数据时，必须对从三个文件中读取的数据进行比较，然后组合成一行有意义的数据。比如，将从 F1 文件中读取的学号与从 F3 文件中读取的学号进行比较，学号相同时，才可以将 F1 文件中的“班号”与 F3 文件中的当前记录所对应的学号和姓名组合起来，之后，还需要将组合结果与 F2 文件中的内容进行比较，找出课程号相同的课程的学分，再与已有的结果组合起来。然后再从组合后的数据中提取出用户需要的信息。如果数据量很大，涉及的文件比较多时，我们可以想象这个过程有多复杂。因此，这种复杂信息的查询，在按文件管理数据的方式中是很难处理的。

（7）无安全控制功能。在文件管理方式中，很难控制某个人对文件能够进行的操作，比如只允许某个人查询和修改数据，但不能删除数据，或者对文件中的某个或者某些字段不能修改等。而在实际应用中，数据的安全性是非常重要且不可忽视的。比如，在学生选课管理中，我们不允许学生修改其考试成绩，但允许他们查询自己的考试成绩。在银行系统中，更是不允许一般用户修改其存款数额。

人们对数据需求的增加，迫切需要对数据进行有效、科学、正确、方便的管理。针对文件管理方式的这些缺陷，人们逐步开发出了以统一管理和共享数据为主要特征的数据库管理系统。

1.3.2　数据库管理

20 世纪 60 年代后期以来，计算机管理数据的规模越来越大，应用范围越来越广泛，数据量急剧增加，同时多种应用同时共享数据集合的要求也越来越强烈。

随着大容量磁盘的出现，硬件价格的不断下降，软件价格的不断上升，编制和维护系统软件和应用程序的成本相应的不断增加。在数据处理方式上，对联机实时处理的需求越来越多，同时开始提出和考虑分布式处理技术。在这种背景下，以文件方式管理数据已经不能满足应用的需求，于是出现了新的管理数据的技术——数据库技术，同时出现了统一管理数据的专门软件——数据库管理系统。

从 1.3.1 节的介绍我们可以看到，在数据库管理系统出现之前，人们对数据的操作是通过直接针对数据文件编写应用程序实现的，这种模式会产生很多问题。在有了数据库管理系统之后，人们对数据的操作全部是通过数据库管理系统实现的，而且应用程序的编写也不再直接针对存放数据的文件。有了数据库技术和数据库管理系统之后，人们对数据的操作模式发生了根本的变化，如图 1-4 所示。

比较图 1-2 和图 1-4，可以看到主要区别有两个：第一个是在操作系统和用户应用程序之间增加了一个系统软件——数据库管理系统，使得用户对数据的操作都是通过数据库管理系统实现的；第二个是有了数据库管理系统之后，用户不再需要有数据文件的概念，即不再需要知道数据文件的逻辑和物理结构及物理存储位置，而只需要知道存放数据的场所——数据库即可。

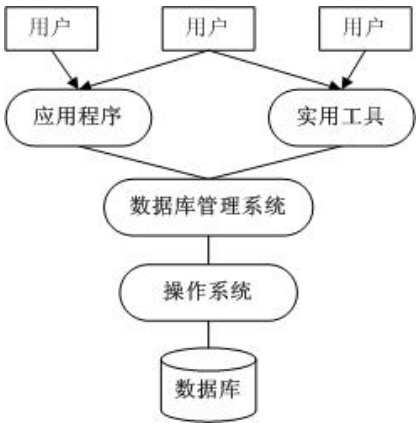


图 1-4　用数据库进行管理的操作模

从本质上讲，即使在有了数据库技术之后，数据最终还是以文件的形式存储在磁盘上的（这点我们将在本书附录 A 中的创建数据库部分可以体会到），只是这时对物理数据文件的存取和管理是由数据库管理系统统一实现的，而不再是每个用户通过编写应用程序实现。数据库和数据文件既有区别又有联系，它们之间的关系类似于单位的名称和地址之间的关系。单位地址代表了单位的实际存在位置，单位名称是单位的逻辑代表。而且一

个数据库可以包含多个数据文件，就像一个单位可以有多个不同地址一样（就像我们现在的很多大学，都是一个学校有多个校址），每个数据文件存储数据库的部分数据。不管一个数据库包含多少个数据文件，对用户来说他只针对数据库进行操作，而无需对数据文件进行操作。这种模式极大地简化了用户对数据的访问。

在有了数据库技术之后，用户只需要知道存放所需数据的数据库名，就可以对数据库对应的数据文件中的数据进行操作。将对数据库的操作转换为对物理数据文件的操作是由数据库管理系统自动实现的，用户不需要知道，也不需要干预。

对于 1.3.1 中列举的学生基本信息管理和学生选课管理两个子系统，如果使用数据库技术来实现，其实现方式如图 1-5 所示。

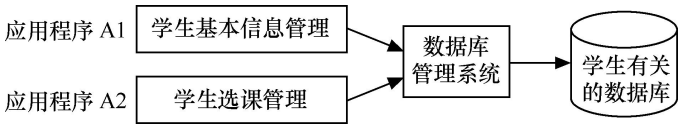


图 1-5 用数据库存储数据的实现示例

与用文件管理数据相比，用数据库技术管理数据具有以下特点：

（1）相互关联的数据集合。在用数据库技术管理数据时，所有相关的数据都被存储在一个数据库中，它们作为一个整体定义，因此可以很方便地表达数据之间的关联关系。比如学生基本信息中的“学号”与学生选课管理中的“学号”，这两个学号之间是有关联关系的，即学生选课中的“学号”的取值范围在学生基本信息的“学号”取值范围内。在关系数据库中，数据之间的关联关系是通过参照完整性实现的。

（2）较少的数据冗余。由于数据是被统一管理的，因此可以从全局着眼，对数据进行最合理的组织。例如，将 1.3.1 节中文件 F1、F2 和 F3 的重复数据挑选出来，进行合理的管理，这样就可以形成如下所示的几部分信息：

学生基本信息：学号、姓名、性别、出生日期、联系电话、所在系、专业、班号。

课程基本信息：课程号、课程名、授课学期、学分、课程性质。

学生选课信息：学号、课程号、修课类型、修课时间、考试成绩。

在关系数据库中，可以将每一类信息存储在一个表中（关系数据库的概念将在后边介绍），重复的信息只存储一份，当在学生选课中需要学生的姓名等其他信息时，根据学生选课中的学号，可以很容易地在学生基本信息中找到此学号对应的姓名等信息。因此，消除数据的重复存储不影响对信息的提取，同时还可以避免由于数据重复存储而造成的数据不一致问题。比如，当某个学生所学的专业发生变化时，只需在“学生基本信息”一个地方进行修改即可。

同 1.3.1 节中的问题一样，当所需的信息来自不同地方，比如（班号，学号，姓名，课程名，学分，考试成绩）信息，这些信息需要从 3 个地方（关系数据库为 3 张表）得到，这种情况下，也需要对信息进行适当的组合，即学生选课中的学号只能与学生基本信息中学号相同的信息组合在一起，同样，学生选课中的课程号也必须与课程基本信息中课程号相同的信息组合在一起。过去在文件管理方式中，这个工作是由开发者编程实现的，而现在有了数据库管理系统，这些繁琐的工作完全交给了数据库管理系统来完成。

因此，在用数据库技术管理数据的系统中，避免数据冗余不会增加开发者的负担。在关系数据库中，避免数据冗余是通过关系规范化理论实现的。

（3）程序与数据相互独立。在数据库中，组成数据的数据项以及数据的存储格式等信息都与数据存储在一起，它们通过 DBMS 而不是应用程序来操作和管理，应用程序不再需要处理文件和记录的格式。

程序与数据相互独立有两方面的含义。一方面是当数据的存储方式发生变化时（这里包括逻辑存储方式和物理存储方式），比如从链表结构改为散列结构，或者是顺序存储和非顺

序存储之间的转换，应用程序不必作任何修改。另一方面是当数据所包含的数据项发生变化时，比如增加或减少了一些数据项，如果应用程序与这些修改的数据项无关，则不用修改应用程序。这些变化都将由 DBMS 负责维护。大多数情况下，应用程序并不知道也不需要知道数据存储方式或数据项已经发生了变化。

在关系数据库中，数据库管理系统通过将数据划分为三个层次来自动保证程序与数据相互独立。我们将在第 2 章详细介绍数据的三个层次，也称为三级模式结构。

（4）保证数据的安全和可靠。数据库技术能够保证数据库中的数据是安全的和可靠的。它的安全控制机制可以有效地防止数据库中的数据被非法使用和非法修改；其完整的备份和恢复机制可以保证当数据遭到破坏时（由软件或硬件故障引起的）能够很快地将数据库恢复到正确的状态，并使数据不丢失或只有很少的丢失，从而保证系统能够连续、可靠地运行。保证数据的安全是通过数据库管理系统的安全控制机制实现的，保证数据的可靠是通过数据库管理系统的备份和恢复机制实现的。

（5）最大限度地保证数据的正确性。数据的正确性也称为数据的完整性，它是指存储到数据库中的数据必须符合现实世界的实际情况，比如人的性别只能是“男”和“女”，人的年龄应该在 0 到 150 之间（假设没有年龄超过 150 岁的人）。如果在性别中输入了其他值，或者将一个负数输入到年龄中，在现实世界中显然是不对的。数据的正确性是通过在数据库中建立完整性约束来实现的。当建立好保证数据正确的约束之后，如果有不符合约束的数据要存储到数据库中，数据库管理系统能主动拒绝这些数据。

（6）数据可以共享并能保证数据的一致性。数据库中的数据可以被多个用户共享，即允许多个用户同时操作相同的数据。当然，这个特点是针对支持多用户的大型数据库管理系统而言的，对于只支持单用户的小型数据库管理系统（比如 Access），在任何时候最多只允许一个用户访问数据库，因此不存在共享的问题。

多用户共享问题是数据库管理系统内部解决的问题，它对用户是不可见的。这就要求数据库管理系统能够对多个用户进行协调，保证多个用户之间对相同数据的操作不会产生矛盾和冲突，即在多个用户同时操作相同数据时，能够保证数据的一致性和正确性。设想一下火车订票系统，如果多个订票点同时对某一天的同一车次火车进行订票，那么必须保证不同订票点订出票的座位不能重复。

数据可共享并能保证共享数据的一致性是由数据库管理系统的并发控制机制实现的。

到今天，数据库技术已经发展成为一门比较成熟的技术，通过上述讨论，我们可以概括出数据库具备如下特征：

数据库是相互关联的数据的集合，它用综合的方法组织数据，具有较小的数据冗余，可供多个用户共享，具有较高的数据独立性，具有安全控制机制，能够保证数据的安全、可靠，允许并发地使用数据库，能有效、及时地处理数据，并能保证数据的一致性和正确性。

需要强调的是，所有这些特征并不是数据库中的数据固有的，而是靠数据库管理系统提供和保证的。

1.4　数据独立性

数据独立性是指应用程序不会因数据的物理表示方式和访问技术的改变而改变，即应用程序不依赖于任何特定的物理表示方式和访问技术，它包含两个方面：物理独立性和逻辑独立性。物理独立性是指当数据的存储位置或存储结构发生变化时，不影响应用程序的特性；逻辑独立性是指当表达现实世界的信息内容发生变化时，比如增加一些列、删除无用列等，也不影响应用程序的特性。要准确理解数据独立性的含义，可先了解下什么是非数据独立性。

在数据库技术出现之前，也就是在使用文件管理数据的时候，实现的应用程序常常是数据依赖的，也就是说数据的物理存储方式和有关的存取技术都要在应用程序中考虑，而且，有关物理存储的知识和访问技术直接体现在应用程序的代码中。例如，如果数据文件使用了索引，那么应用程序必须知道有索引存在，也要知道数据是按索引排序的，这样应用程序的内部结构就是基于这些知识而设计的。一旦数据的物理存储方式改变了，就会对应用程序产生很大的影响。例如，如果改变了数据的排序方式，则应用程序不得不做很大的修改。而且在这种情况下，应用程序修改的部分恰恰是与数据管理密切联系的部分，而与应用程序最初要解决的问题毫不相干。

在用数据库技术管理数据的方式中，可以尽量避免应用程序对数据的依赖，这有如下两种情况：

- 不同的用户关心的数据并不完全相同，即使对同样的数据不同用户的需求也不尽相同。比如前边的学生基本信息数据，包括：学号、姓名、性别、出生日期、联系电话、所在系、专业、班号，分配宿舍的部门可能只需要：学号、姓名、班号，性别，教务部门可能只需要：学号、姓名、所在系、专业和班号。好的实现方法应根据全体用户对数据的需求存储一套完整的数据，而且只编写一个针对全体用户的公共数据的应用程序，但能够按每个用户的具体要求只展示其需要的数据，而且当公共数据发生变化时（比如增加新数据），可以不修改应用程序，每个不需要这些变化数据的用户也不需要知道有这些变化。这种独立性（逻辑独立性）在文件管理方式下是很难实现的。
- 随着科学技术的进步以及应用业务的变化，有时必须要改变数据的物理存储方式和存取方法以适应技术发展及需求变化。比如改变数据的存储位置或存储结构（就像一个单位可以搬到新的地址，或者是调整单位各科室的布局）以提高数据的访问效率。理想情况下，这些变化不应该影响应用程序（物理独立性）。这在文件管理方式下也是很难实现的。

因此，数据独立性的提出是一种客观应用的要求。数据库技术的出现正好克服了应用程序对数据的物理表示和访问技术的依赖。

1.5　数据库系统的组成

我们在 1.1 节简单介绍了数据库系统的组成，数据库系统是基于数据库的计算机应用系统，一般包括数据库、数据库管理系统（及相应的实用工具）、应用程序和数据库管理员四个部分，如图 1-6 所示。数据库是数据的汇集场所，它以一定的组织形式保存在存储介质上；数据库管理系统是管理数据库的系统软件，它可以实现数据库系统的各种功能；应用程序专指访问数据库数据的程序，数据库管理员负责整个数据库系统的正常运行。

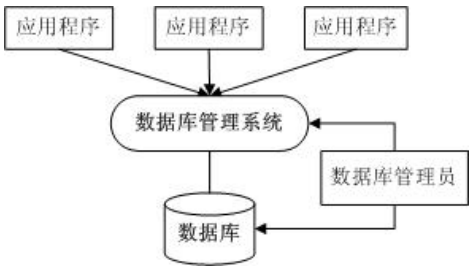


图 1-6　数据库系统组成示意图

任何程序的运行和存储都需要占用硬件资源，下面就从硬件、软件和人员几个方面简要介绍数据库系统包含的主要内容。

1．硬件

由于数据库中的数据量一般都比较大，而且 DBMS 由于丰富的功能而使得自身的规模也很大（SQL Server 2008 的完整安装大致需要 2GB 的硬盘空间和至少 512MB 以上的内存），因此整个数据库系统对硬件资源的要求很高。必须要有足够大的内存，来运行操作系统、数据库管理系统和应用程序，而且还要有足够大的硬盘空间来存放数据库数据以及相应的系统

软件和应用程序。

2．软件

数据库系统的软件主要包括：

- 数据库管理系统。是整个数据库系统的核心，是建立、使用和维护数据库的系统软件。
- 支持数据库管理系统运行的操作系统。数据库管理系统中的很多底层操作是靠操作系统完成的，数据库中的安全控制等功能通常也是与操作系统共同实现的。因此，数据库管理系统要和操作系统协同工作来完成很多功能。不同的数据库管理系统需要的操作系统平台不尽相同，比如 SQL Server 只支持在 Windows 平台上运行，而 Oracle 有支持 Windows 平台和 Linux 平台的不同版本。
- 以数据库管理系统为核心的实用工具，这些实用工具一般是数据库厂商提供的随数据库管理系统软件一起发行的。

3．人员

数据库系统中包含的人员主要有：数据库管理员、系统分析人员、数据库设计人员、应用程序编程人员和最终用户。

- 数据库管理员负责维护整个系统的正常运行，负责保证数据库的安全和可靠。
- 系统分析人员主要负责应用系统的需求分析和规范说明，这些人员要和最终用户以及数据库管理员配合，以确定系统的软、硬件配置，并参与数据库应用系统的概要设计。
- 数据库设计人员主要负责确定数据库数据，设计数据库结构等。数据库设计人员也必须参与用户需求调查和系统分析。在很多情况下，数据库设计人员就由数据库管理员担任。
- 应用程序编程人员负责设计和编写访问数据库的应用系统的程序，并对程序进行调试和安装。
- 最终用户是数据库应用程序的使用者，他们是通过应用程序提供的人机交互界面来操作数据库中数据的人员。

习　题

一、选择题

1. 下列关于用文件管理数据的说法，错误的是

A．用文件管理数据，难以提供应用程序对数据的独立性

B．当存储数据的文件名发生变化时，必须修改访问数据文件的应用程序

C．用文件存储数据的方式难以实现数据访问的安全控制

D．将相关的数据存储在一个文件中，有利于用户对数据进行分类，因此也可以加快用户操作数据的效率

2. 下列说法中，不属于数据库管理系统特征的是

A．提供了应用程序和数据的独立性

B．所有的数据作为一个整体考虑，因此是相互关联的数据的集合

C．用户访问数据时，需要知道存储数据的文件的物理信息

D．能保证数据库数据的可靠性，即使在存储数据的硬盘出现故障时，也能防止数据丢失

3. 数据库管理系统是数据库系统的核心，它负责有效地组织、存储和管理数据，它位于用

户和操作系统之间，属于

A．系统软件　　B．工具软件

C．应用软件　　D．数据软件

4. 数据库系统是由若干部分组成的。下列不属于数据库系统组成部分的是

A．数据库　　B．操作系统

C．应用程序　　D．数据库管理系统

5. 下列关于数据库技术的描述，错误的是

A．数据库中不但需要保存数据，而且还需要保存数据之间的关联关系

B．数据库中的数据具有较小的数据冗余

C．数据库中数据存储结构的变化不会影响到应用程序

D．由于数据库是存储在磁盘上的，因此用户在访问数据库时需要知道其存储位置

二、简答题

1. 试说明数据、数据库、数据库管理系统和数据库系统的概念。
2. 数据管理技术的发展主要经历了哪几个阶段？
3. 文件管理方式在管理数据方面有哪些缺陷？
4. 与文件管理相比，数据库管理有哪些优点？
5. 在数据库管理方式中，应用程序是否需要关心数据的存储位置和存储结构？为什么？
6. 数据库系统由哪几部分组成，每一部分在数据库系统中的作用大致是什么？

第 2章

数据模型与数据库结构

本章将介绍数据库技术实现程序和数据相互独立的基本原理，即数据库的结构。在介绍数据库结构之前，先介绍数据模型的一些基本概念。本章的内容是理解数据库技术特色的基础。

2.1　数据和数据模型

现实世界的数据是散乱无章的，散乱的数据不利于人们对其进行有效的管理和处理，特别是海量数据。因此，必须把现实世界的数据按照一定的格式组织起来，以方便对其进行操作和使用，数据库技术也不例外，在用数据库技术管理数据时，数据被按照一定的格式组织起来，比如二维表结构或者是层次结构，以使数据能够被更高效地管理和处理。本节就对数据和数据模型进行简单介绍。

2.1.1　数据与信息

在介绍数据模型之前，我们先来了解数据与信息的关系。在第 1 章 1.2 节已经介绍了数据的概念，说明数据是数据库中存储的基本对象。为了了解世界、研究世界和交流信息，人们需要描述各种事物。用自然语言来描述虽然很直接，但过于繁琐，不便于形式化，而且也不利于用计算机来表达。为此，人们常常只抽取那些感兴趣的事物特征或属性来描述事物。例如，一名学生可以用信息（张三，201412101，男，河北，计 1401，软件工程）描述，这样的一行数据称为一条记录。单看这行数据我们不一定能准确知道其含义，但对其进行如下解释：张三的学号是 201412101，他是计 1401 班的男生，河北生源，软件工程专业，其内容就是确定的。我们将描述事物的符号记录称为数据，将从数据中获得的有意义的内容称为信息。数据有一定的格式，例如，姓名是长度不超过 4 个汉字的字符串（假设学生的姓名都不超过 4 个汉字），性别是一个汉字的字符。这些格式的规定是数据的语法，而数据的含义是数据的语义。因此，数据是信息存在的一种形式，只有通过解释或处理才能成为有用的信息。

一般来说，数据库中的数据具有静态特征和动态特征两个方面：

（1）静态特征。数据的静态特征包括数据的基本结构、数据间的联系以及对数据取值范围的约束。比如 1.2.1 节中给出的学生管理的例子。学生基本信息包含学号、姓名、性别、出生日期、联系电话、所在系、专业、班号，这些都是学生所具有的基本性质，是学生数据的基本结构。学生选课信息包括学号、课程号和考试成绩等，这些是学生选课的基本性质。但学生选课信息中的学号与学生基本信息中的学号是有一定关联的，即学生选课信息中的“学号”所能取的值应在学生基本信息中的“学号”取值范围之内，因为只有这样，学生选课信息中所描述的学生选课情况才是有意义的（我们不会记录不存在的学生的选课情况），这就是数据之间的联系。最后我们看数据取值范围的约束。我们知道人的性别一项的取值只能是“男”或“女”、课程的学分一般是大于 0 的整数值、学生的考试成绩一般在 0～100 分之间等，这些都是对某个列的数据取值范围进行的限制，目的是在数据库中存储正确的、有意义的数据。这就是对数据取值范围的约束。

（2）动态特征。数据的动态特征是指对数据可以进行的操作以及操作规则。对数据库数据的操作主要有查询数据和更改数据，更改数据一般又包括插入、删除和更新。

一般将对数据的静态特征和动态特征的描述称为**数据模型三要素**，即在描述数据时要包

括数据的基本结构、数据的约束条件（这两个属于静态特征）和定义在数据上的操作（属于数据的动态特征）三个方面。

2.1.2 数据模型

对于模型，特别是具体的模型，人们并不陌生。一张地图、一组建筑设计沙盘、一架飞机模型等都是具体的模型。人们可以从模型联想到现实生活中的事物。计算机中的模型是对事物、对象、过程等客观系统中感兴趣的内容的模拟和抽象表达，是理解系统的思维工具。数据模型（data model）也是一种模型，它是对现实世界数据特征的抽象。

数据库是企业或部门相关数据的集合，数据库不仅要反映数据本身的内容，而且要反映数据之间的联系。由于计算机不可能直接处理现实世界中的具体事物，因此，必须要把现实世界中的具体事物转换成计算机能够处理的对象。在数据库中用数据模型这个工具来抽象、表示和处理现实世界中的数据和信息。

数据库管理系统是基于某种数据模型对数据进行组织的，因此，了解数据模型的基本概念是学习数据库知识的基础。

在数据库领域中，数据模型用于表达现实世界中的对象，即将现实世界中杂乱的信息用一种规范的、易于处理的方式表达出来。而且这种数据模型即要面向现实世界（表达现实世界信息），同时又要面向机器世界（因为要在机器上实现出来），因此一般要求数据模型满足三个方面的要求：

第一， 能够真实地模拟现实世界。因为数据模型是抽象现实世界对象信息，经过整理、加工，成为一种规范的模型。但构建模型的目的是为了真实、形象地表达现实世界情况。

第二， 容易被人们理解。因为构建数据模型一般是数据库设计人员做的事情，而数据库设计人员往往并不是所构建的业务领域的专家，因此，数据库设计人员所构建的模型是否正确，是否与现实情况相符，需要由精通业务的用户来评判，而精通业务的人员往往又不是计算机领域的专家。因此要求所构建的数据模型要形象化，要容易被业务人员理解，以便于他们对模型进行评判。

第三， 能够方便地在计算机上实现。因为对现实世界业务进行设计的最终目的是能够在计算机上实现出来，用计算机来表达和处理现实世界的业务。因此所构建的模型必须能够方便地在计算机上实现，否则就没有任何意义。

用一种模型来同时很好地满足这三方面的要求在目前是比较困难的，因此在数据库领域中是针对不同的使用对象和应用目的，采用不同的数据模型来实现。

数据模型实际上是模型化数据和信息的工具。根据模型应用的不同目的，可以将模型分为两大类，它们分别属于两个不同的层次。

第一类是概念层数据模型，也称为概念模型或信息模型，它从数据的应用语义视角来抽取现实世界中有价值的数据并按用户的观点来对数据进行建模。这类模型主要用在数据库的设计阶段，它与具体的数据库管理系统无关，也与具体的实现方式无关。另一类是组织层数据模型，也称为组织模型（有时也直接简称为数据模型，本书后边凡是称数据模型的都指的是组织层数据模型），它从数据的组织方式来描述数据。所谓组织层就是指用什么样的逻辑结构来组织数据。数据库发展到现在主要采用了如下几种组织方式（组织模型）：层次模型（用树型结构组织数据）、网状模型（用图型结构组织数据）、关系模型（用简单二维表结构组织数据）以及对象-关系模型（用复杂的表格以及其他结构组织数据）。组织层数据模型主要是从计算机系统的观点对数据进行建模，它与所使用的数据库管理系统的种类有关，因为不同的数据库管理系统支持的数据模型可以不同。

为了把现实世界中的具体事物抽象、组织为某一具体 DBMS 支持的数据模型，人们通常首先将现实世界抽象为信息世界，然后再将信息世界转换为机器世界。即，首先把现

实世界中的客观对象抽象为某一种描述信息的模型，这种模型并不依赖于具体的计算机系统，而且也不与具体的 DBMS 有关，而是概念意义上的模型，也就是我们前边所说的概念层数据模型；然后再把概念层数据模型转换为具体的 DBMS 支持的数据模型，也就是组织层数据模型（比如关系数据库的二维表）。注意从现实世界到概念层数据模型使用的是“抽象”技术，从概念层数据模型到组织层数据模型使用的是“转换”技术，也就是说先有概念模型，然后再到组织模型。从概念模型到组织模型的转换是比较直接和简单的，我们将在第 9 章数据库设计中详细介绍转换方法。这个过程如图 2-1 所示。

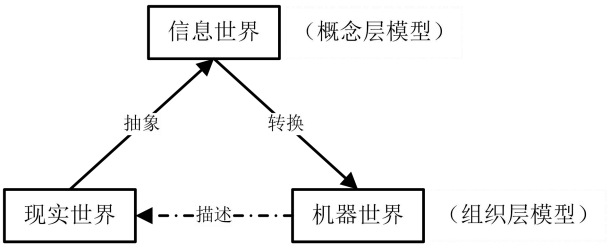


图 2-1　从现实世界到机器世界的过程

2.2　概念层数据模型

从图 2-1 可以看出，概念层数据模型实际上是现实世界到机器世界的一个中间层，机器世界实现的最终目的是为了反映和描述现实世界。本节介绍概念层数据模型的基本概念及基本构建方法。

2.2.1　基本概念

概念层数据模型是指抽象现实系统中有应用价值的元素及其关联关系，反映现实系统中有应用价值的信息结构，并且不依赖于数据的组织层数据模型。

概念层数据模型用于对信息世界进行建模，是现实世界到信息世界的第一层抽象，是数据库设计人员进行数据库设计的工具，也是数据库设计人员和业务领域的用户之间进行交流的工具，因此，该模型一方面应该具有较强的语义表达能力，能够方便、直接地表达应用中的各种语义知识；另一方面它还应该简单、清晰和易于被用户理解。因为概念模型设计的正确与否，即所设计的概念模型是否合理、是否正确地表达了现实世界的业务情况，是由业务人员来判定的。

概念层数据模型是面向用户、面向现实世界的数据模型，它与具体的 DBMS 无关。采用概念层数据模型，设计人员可以在数据库设计的开始把主要精力放在了解现实世界上，而把涉及 DBMS 的一些技术性问题推迟到后面去考虑。

常用的概念层数据模型有实体-联系（Entity-Relationship，E-R）模型、语义对象模型。本书只介绍实体-联系模型，这也是最常使用的一种概念模型。

2.2.2　实体-联系模型

如果直接将现实世界数据按某种具体的组织模型进行组织，必须同时考虑很多因素，设计工作也比较复杂，并且效果并不一定理想，因此需要一种方法能够对现实世界的信息结构进行描述。事实上这方面已经有了一些方法，我们要介绍的是 P. P. S. Chen 于 1976 年提出的实体-联系方法，即通常所说的 E-R 方法。这种方法由于简单、实用，因此得到了广泛的应用，也是目前描述信息结构最常用的方法。

实体-联系方法使用的工具称为E-R图，它所描述的现实世界的信息结构称为企业模式（Enterprise Schema），也把这种描述结果称为E-R模型。

实体-联系方法试图定义很多数据分类对象，然后数据库设计人员就可以将数据项归类到已知的类别中。我们将在第8章更详细地介绍E-R模型，在第9章介绍如何将E-R模型转换为关系数据模型。

在实体-联系模型中主要涉及三方面内容：实体、属性和联系。

（1）实体。实体是具有公共性质、并可相互区分的现实世界对象的集合，或者说是具有相同结构的对象的集合。实体是具体的，例如：职工、学生、教师、课程都是实体。

在E-R图中用矩形框表示具体的实体，把实体名写在框内，如图2-2(a)中的“经理”和“部门”实体。

实体中每个具体的记录值（一行数据），比如学生实体中的每个具体的学生就是学生实体中的一个实例，我们称之为实体的一个实例。（注意，有些书也将实体称为实体集或实体类型，而将每行具体的记录称为实体。）

（2）属性。每个实体都具有一定的特征或性质，这样我们才能根据实体的特征来区分一个个实例。属性就是描述实体或者联系的性质或特征的数据项，属于一个实体的所有实例都具有相同的性质，在E-R模型中，这些性质或特征就是属性。比如学生的学号、姓名、性别等都是学生实体具有的特征，这些特征就构成了学生实体的属性。实体应具有多少个属性是由用户对信息的需求决定的。例如，假设用户还需要学生的出生日期信息，则可以在学生实体中加一个“出生日期”属性。

在实体的属性中，将能够唯一标识实体的一个属性或最小的一组属性（称为属性集或属性组）称为实体的标识属性，这个属性或属性组也称为实体的码。例如，“学号”就是学生实体的码。

属性在E-R图中用圆角矩形表示，在圆角矩形框内写上属性的名字，并用连线将属性框与它所描述的实体联系起来，如图2-2(c)所示。

（3）联系。在现实世界中，事物内部以及事物之间是有联系的，这些联系在信息世界反映为实体内部的联系和实体之间的联系。实体内部的联系通常是指一个实体内部属性之间的联系，实体之间的联系通常是指不同实体属性之间的联系。比如在“职工”实体中，假设有职工号、职工姓名，所在部门和部门经理号等属性，其中“部门经理号”描述的是这个职工所在部门的经理的职工号。一般来说，部门经理也属于单位的职工，而且通常与职工采用的是一套职工编码方式，因此“部门经理号”与“职工号”之间有一种关联的关系，即“部门经理号”的取值在“职工号”取值范围内。这就是实体内部的联系。而“学生”和“系”之间就是实体之间的联系，“学生”是一个实体，假设该实体中有学号、姓名、性别、所在系等属性，“系”也是一个实体，假设该实体中包含系名、系联系电话、系办公地点等属性，则“学生”实体中的“所在系”与“系”实体中的“系名”之间存在一种关联关系，即“学生”实体中“所在系”属性的取值范围必须在“系”实体中“系名”属性的取值范围内。因此象“系”和“学生”这种关联到两个不同实体的联系就是实体之间的联系。通常情况下我们遇到的联系大多都是实体之间的联系。

联系是数据之间的关联关系，是客观存在的应用语义链。在E-R图中联系用菱形框表示，框内写上联系名，并用连线将联系框与它所关联的实体连接起来，如图2-2(a)中的“管理”联系。

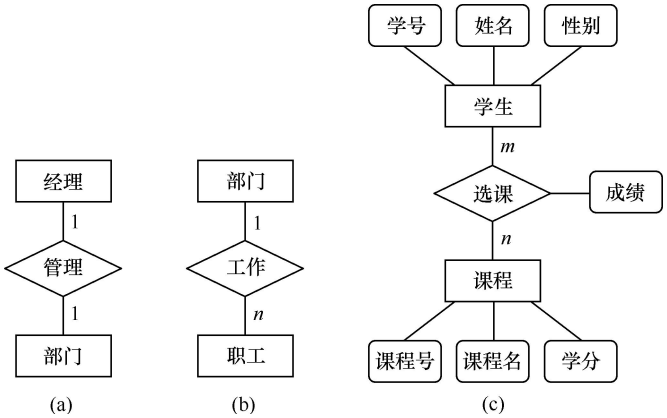


图 2-2　实体及其联系的示例

联系也可以有自己的属性，比如图 2-2(c)所示的“选课”联系中有“成绩”属性。

两个实体之间的联系通常有如下三类：

（1）一对一联系（1 : 1）。如果实体 A 中的每个实例在实体 B 中至多有一个（也可以没有）实例与之关联，反之亦然，则称实体 A 与实体 B 具有一对一联系，记作 1 : 1。

例如，部门和经理（假设一个部门只允许有一个经理，一个人只允许担任一个部门的经理）、系和正系主任（假设一个系只允许有一个正主任，一个人只允许担任一个系的主任）都是一对一的联系。一对一联系示例如图 2-2(a)所示。

（2）一对多联系（1 : n）。如果实体 A 中的每个实例在实体 B 中有 n 个实例（$n \geqslant 0$）与之关联，而实体 B 中的每个实例在实体 A 中最多只有一个实例与之关联，则称实体 A 与实体 B 是一对多联系，记作 1 : n。

例如，假设一个部门有若干职工，而一个职工只允许在一个部门工作，则部门和职工之间就是一对多联系。又比如，假设一个系有多名教师，而一个教师只允许在一个系工作，则系和教师之间也是一对多联系。一对多联系示例如图 2-2(b)所示。

（3）多对多联系（m : n）。如果实体 A 中的每个实例在实体 B 中有 n 个实例（$n \geqslant 0$）与之关联，而实体 B 中的每个实例，在实体 A 中也有 m 个实例（$m \geqslant 0$）与之关联，则称实体 A 与实体 B 是多对多联系，记为 m : n。

比如学生和课程，一个学生可以选修多门课程，一门课程也可以被多个学生选修，因此学生和课程之间是多对多的联系。多对多联系示例如图 2-2(c)所示。

实际上，一对一联系是一对多联系的特例，而一对多联系又是多对多联系的特例。

注意：实体之间联系的种类是与语义直接相关的，也就是由客观实际情况决定的。例如，部门和经理，如果客观情况是一个部门只有一个经理，一个人只担任一个部门的经理，则部门和经理之间是一对一联系。但如果客观情况是一个部门可以有多个经理，而一个人只担任一个部门的经理，则部门和经理之间就是一对多联系。如果客观情况是一个部门可以有多个经理，而且一个人也可以担任多个部门的经理，则部门和经理之间就是多对多联系。

E-R 图不仅能描述两个实体之间的联系，而且还能描述两个以上实体之间的联系。比如有顾客、商品、售货员三个实体，并且有语义：每个顾客可以从多个售货员那里购买商品，并且可以购买多种商品；每个售货员可以向多名顾客销售商品，并且可以销售多种商品；每种商品可由多个售货员销售，并且可以销售给多名顾客。描述顾客、商品和售货员之间的关联关系的 E-R 图如图 2-3 所示，这里将联系命名为“销售”。

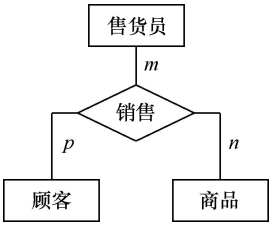


图 2-3　多个实体之间的联系示例

E-R 图广泛用于数据库设计的概念结构设计阶段。用 E-R 模型表示的数据库概念设计结果非常直观，易于用户理解，而且所设计的 E-R 图与具体的数据组织方式无关，并可以被直观地转换为关系数据库中的关系表。

2.3　组织层数据模型

组织层数据模型是从数据的组织形式的角度来描述信息，目前，在数据库技术的发展过程中用到的组织层数据模型主要有：层次模型（Hierarchical Model）、网状模型（Network Model）、关系模型（Relational Model）、面向对象模型（Object Oriented Model）和对象关系模型（Object Relational Model）。组织层数据模型是按组织数据的逻辑结构来命名的，比如层次模型采用树型结构。而且各数据库管理系统也是按其所采用的组织层数据模型来分类的，比如层次数据库管理系统就是按层次模型来组织数据，而网状数据库管理系统就是按网状模型来组织数据。

1970 年美国 IBM 公司研究员 E.F.Codd 首次提出了数据库系统的关系模型，开创了关系数据库和关系数据理论的研究，为关系数据库技术奠定了理论基础。关系模型从 20 世纪 70～80 年代开始到现在已经发展得非常成熟，本书的重点也是介绍关系模型。20 世纪 80 年代以来，计算机厂商推出的数据库管理系统几乎都支持关系模型，非关系系统的产品也大都加上了关系接口。

一般将层次模型和网状模型统称为非关系模型。非关系模型的数据库管理系统在 20 世纪 70 年代至 80 年代初非常流行，在数据库管理系统的产品中占主导地位，但现在已逐步被采用关系模型的数据库管理系统所取代。20 世纪 80 年代以来，面向对象的方法和技术在计算机各个领域，包括程序设计语言、软件工程、信息系统设计、计算机硬件设计等方面都产生了深远的影响，也促进了数据库中面向对象数据模型的研究和发展。

2.3.1 层次数据模型

层次数据模型（简称层次模型）是数据库管理系统中最早出现的数据模型。层次数据库管理系统采用层次模型作为数据的组织方式。层次数据库管理系统的典型代表是 IBM 公司的 IMS（Information Management System），这是 IBM 公司 1968 年推出的第一个大型的商用数据库管理系统。

层次数据模型用树形结构表示实体和实体之间的联系。现实世界中许多实体之间的联系本身就呈现出一种自然的层次关系，如行政机构、家族关系等。

构成层次模型的树由结点和连线组成，结点表示实体，结点中的项表示实体的属性，连线表示相连的两个实体间的联系，这种联系是一对多的。通常把表示“一”的实体放在上方，称为父结点；把表示“多”的实体放在下方，称为子结点。将不包含任何子结点的结点称为叶结点。如图 2-5 所示。

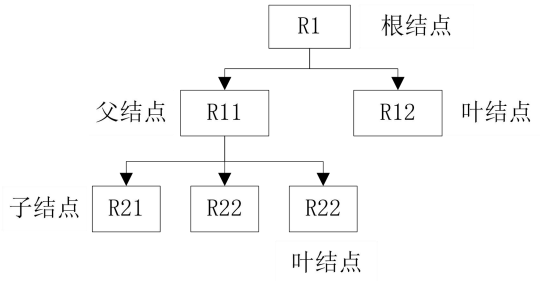


图 2-5 层次模型示意图

层次模型可以直接、方便的表示一对多的联系。但在层次模型中有以下两点限制：

（1）有且仅有一个结点无父结点，这个结点即为树的根；

（2）其他结点有且仅有一个父结点。

层次模型的一个基本特点是，任何一个给定的记录值只有从层次模型的根部开始按路径查看时，才能明确其含义，任何子结点都不能脱离父结点而存在。

图 2-6 所示为一个用层次结构组织的学院数据模型，该模型有 4 个结点，“学院”是根结点，由学院编号、学院名称和办公地点三项组成。“学院”结点下有两个子结点，分别为“教研室”和“学生”。“教研室”结点由“教研室名”、“室主任”和“室人数”三项组成，“学生”结点由“学号”、“姓名”、“性别”和“年龄”四项组成。“教研室”结点下又有一个子结点“教师”，因此，“教研室”是“教师”的父结点，“教师”是“教研室”的子结点。“教师”结点由“教师号”、“教师名”和“职称”项组成。

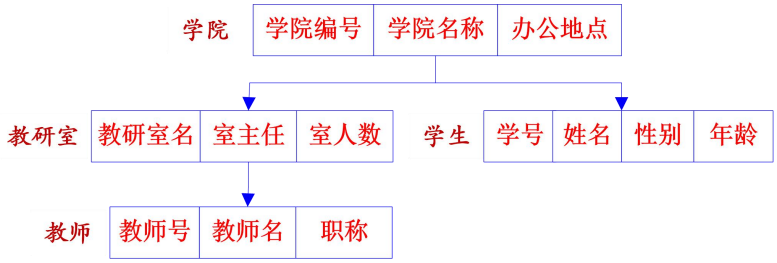


图 2-6 学院的层次数据模型

图 2-7 所示是图 2-6 数据模型对应的一些值。

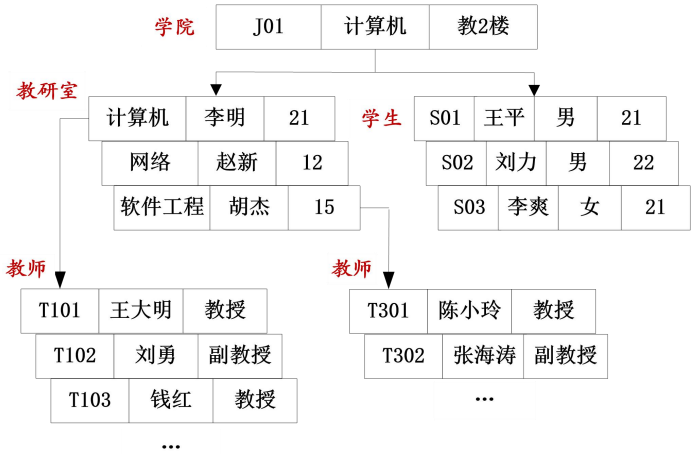


图 2-7 学院层次数据模型的一个值

层次数据模型只能表示一对多的联系，不能直接表示多对多联系。但如果把多对多联系转换为一对多联系，又会出现一个子结点有多个父结点的情况（如图 2-8 所示，学生和课程原本是一个多对多联系，在这里将其转换为两个一对多联系），这显然不符合层次数据模型的要求。一般常用的解决办法是把一个层次模型分解为两个层次模型，如图 2-9 所示。

层次数据库是由若干个层次模型构成的，或者说它是一个层次模型的集合。

2.3.2 网状数据模型

在现实世界中事物之间的联系更多的是非层次的，用层次数据模型表达现实世界中存在

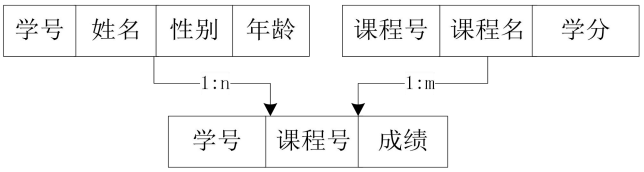


图 2-8 有两个父记录的结构

的联系有很多限制。如果去掉层次模型中的两点限制，即允许一个以上的结点无父结点，并且每个结点可以有多个父结点，便构成了网状模型。

用图形结构表示实体和实体之间的联系的数据模型就称为网状数据模型，简称网状模型。在网状模型中，同样使用父结点和子结点这样的术语，并且同样一般把父结点放置在子结点的上方。图 2-10 所示为几种不同形式的网状模型形式。

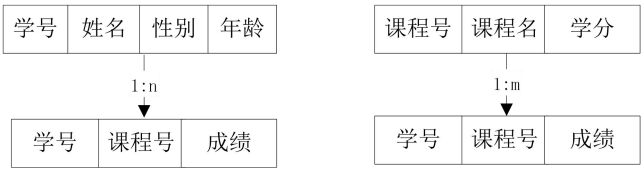


图 2-9 将图 2-8 分解成两个层次模型

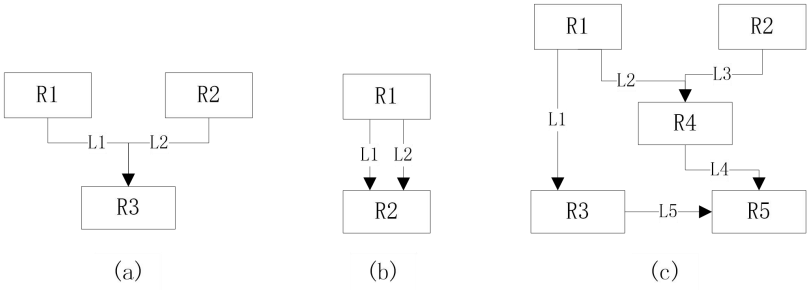


图 2-10 网状数据模型示例

从图 2-10 可以看出，网状模型父结点与子结点之间的联系可以不唯一，因此，就需要为每个联系命名。在图 2-10（a）中，结点 R3 有两个父结点 R1 和 R2，可将 R1 与 R3 之间的联系命名为 L1，将 R2 与 R3 之间的联系命名为 L2。图 2-10（b）和（c）与此类似。

由于网状数据模型没有层次数据模型的两点限制，因此可以直接表示多对多联系。但在网状模型中多对多联系实现起来太复杂，因此一些支持网状模型的数据库管理系统，对多对多联系还是进行了限制。例如，网状模型的典型代表 CODASYL（Conference On Data System

Language）就只支持一对多联系。

网状模型和层次模型在本质上是一样的，从逻辑上看，它们都是用连线表示实体之间的联系，用结点表示实体；从物理上看，层次模型和网状模型都是用指针来实现文件以及记录之间的联系，其差别仅在于网状模型中的连线或指针更复杂、更纵横交错，从而使数据结构更复杂。

网状数据模型的典型代表是 CODASYL 系统，它是 CODASYL 组织的标准建议的具体实现。层次模型是按层次组织数据，而 CODASYL 是按系（set）组织数据。所谓“系”可以理解为命名了的联系，它由一个父记录型和一个或若干个子记录型组成。图 2-11 所示为网状模型的一个示例，其中包含四个系，S-G 系由学生和选课记录构成，C-G 系由课程和选课记录构成，C-C 系由课程和授课记录构成，T-C 系由教师和授课记录构成。实际上，图 2-8 所示的具有两个父结点的结构也属于网状模型。

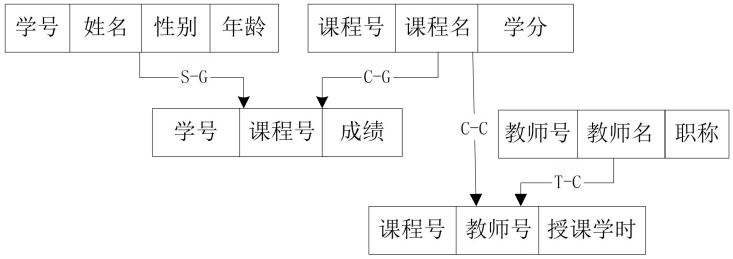


图 2-11 网状结构示意图

2.3.3 关系数据模型

关系数据模型是目前最重要的一种数据模型，关系数据库就是采用关系数据模型作为数据的组织方式。关系数据模型源于数学，它把数据看成是二维表中的元素，而这个二维表在关系数据库中就称为关系。关于关系的详细讨论将在第 3 章进行。

用关系（表格数据）表示实体和实体之间的联系的模型就称为关系数据模型。在关系数据模型中，实体本身以及实体和实体之间的联系都用关系来表示，实体之间的联系不再通过指针来实现。

表 2-1 和表 2-2 所示分别为“学生”和“选课”关系模型的数据结构，其中“学生”和“选课”间的联系是靠“学号”列实现的。

表 2-1 学生表

学　　号	姓　　名	年　　龄	性　　别	所　在　系
0811101	李勇	21	男	计算机系
0811102	刘晨	20	男	计算机系
0811103	王敏	20	女	计算机系
0821101	张立	20	男	信息管理系
0821102	吴宾	19	女	信息管理系

表 2-2 选课表

学　　号	课　程　号	成　　绩
0811101	C001	96
0811101	C002	80

0811101	C003	84
0811101	C005	62
0811102	C001	92
0811102	C002	90
0811102	C004	84
0821102	C001	76
0821102	C004	85
0821102	C005	73

在关系数据库中，记录值仅仅构成关系，关系之间的联系是靠语义相同的字段（称为连接字段）值表达的。理解关系和连接字段（即列）的思想在关系数据库中是非常重要的。例如，要查询“刘晨”的考试成绩，则首先要在“学生”关系中得到“刘晨”的学号值，然后根据这个学号值再在“选课”关系中找出该学生的所有考试记录值。

对于用户来说，关系的操作应该是很简单的，但关系数据库管理系统本身是很复杂的。关系操作之所以对用户很简单，是因为它把大量的工作交给了数据库管理系统来实现。尽管在层次数据库和网状数据库诞生之时，就有了关系模型数据库的设想，但研制和开发关系数据库管理系统却花费了比人们想象的要长得多的时间。关系数据库管理系统真正成为商品并投入使用要比层次数据库和网状数据库晚十几年。但关系数据库管理系统一经投入使用，便显示出了强大的活力和生命力，并逐步取代了层次数据库和网状数据库。现在耳熟能详的数据库管理系统，几乎都是关系数据库管理系统，比如 Microsoft SQL Server、Oracle、IBM DB2、Access 等都是关系型的数据库管理系统。

关系数据模型易于设计、实现、维护和使用，它与层次数据模型和网状数据模型的最根本区别是，关系数据模型采用非导航式的数据访问方式，数据结构的变化不会影响对数据的访问。

2.4　面向对象数据模型

面向对象数据模型是捕获在面向对象程序设计中所支持的对象语义的逻辑数据模型，它是持久的和共享的对象集合，具有模拟整个解决方案的能力。面向对象数据模型把实体表示为类，一个类描述了对象属性和实体行为。例如，一个“学生”类不仅仅有学生的属性，比如学号、姓名和性别等，还包含模仿学生行为（如选修课程）的方法。类-对象的实例对应于学生个体。在对象内部，类的属性用特殊值来区分每个学生（对象），但所有对象都属于类，共享类的行为模式。面向对象数据库通过逻辑包含（logical containment）来维护联系。

面向对象数据库基于把数据和与对象相关的代码封装成单一组件，外面不能看到其里面的内容。因此，面向对象数据模型强调对象（由数据和代码组成）而不是单独的数据。这主要是从面向对象程序设计语言继承过来的。在面向对象程序设计语言里，程序员可以定义包含它们自己的内部结构、特征和行为的新类型或对象类。这样，不能认为数据是独立存在的，而是与代码（成员函数的方法）相关，代码（code）定义了对象能做什么（它们的行为或有用的服务）。面向对象数据模型的结构是非常容易变化的。与传统的数据库（如层次、网状或关系）不同，对象模型没有单一固定的数据库结构。编程人员可以给类或对象类型定义任何有用的结构，例如，链接列表、集合、数组等等。此外，对象可以包含可变的复杂度，利

用多重类型和多重结构。

面向对象数据库管理系统（OODBMS）是数据库管理中最新的方法，它们始于工程和设计领域的应用，并且成为金融、通讯和万维网（WWW）应用欢迎的系统。它适用于多媒体应用以及复杂的很难在关系数据库管理系统中模拟和处理的关系。

2.5 数据库结构

考察数据库的结构可以有不同的层次或不同的角度。

- 从数据库管理角度看，数据库通常采用三级模式结构。这是数据库管理系统内部的系统结构。
- 从数据库最终用户角度看，数据库的结构分为集中式结构、文件服务器结构、客户/服务器结构等。这是数据库的外部结构。

本节我们讨论数据库的内部结构。它是为后续章节的内容建立一个框架结构，这个框架用于描述一般数据库管理系统的概念，但并不是所有的数据库管理系统都一定要使用这个框架，它在数据库管理系统中并不是唯一的，特别是一些“小”的数据库管理系统将难以支持这个结构的所有方面。这里介绍的数据库的结构基本上能很好地适应大多数数据库管理系统，而且，它基本上和 ANSI/SPARC DBMS 研究组提出的数据库管理系统的体系结构（称为 ANSI/SPARC 体系结构）是相同的。

2.5.1 模式的基本概念

数据模型（组织层数据模型）描述数据的组织形式，模式是用给定的数据模型对具体数据的描述。（就像用某一种编程语言编写具体应用程序一样）。

模式是数据库中全体数据的逻辑结构和特征的描述，它仅仅涉及“型”的描述，不涉及具体的值。关系模式是关系的“型”或元组的结构共性的描述，它实际上对应的是关系表的表头。

模式的一个具体值称为模式的一个实例，比如表 2-1 中的每一行数据就是其表头结构（模式）的一个具体实例。一个模式可以有多个实例。模式是相对稳定的（结构不会经常变动），而实例是相对变动的（具体的数据值可以经常变化）。数据模式描述一类事物的结构、属性、类型和约束，实质上是用数据模型对一类事物进行模拟，而实例是反映某类事物在某一时刻的当前状态。

虽然实际的数据库管理系统产品种类很多，支持的数据模型和数据库语言也不尽相同，数据的存储结构也各不相同，但它们在体系结构上通常都具有相同的特征，即采用三级模式结构并提供两级映像功能。

2.5.2 三级模式结构

数据库的三级模式结构是指数据库的外模式、模式和内模式，如图 2-12 所示。

广义地讲，

- 内模式：是最接近物理存储的，也就是数据的物理存储方式，包括数据存储位置、数据存储方式等。
- 外模式：是最接近用户的，也就是用户所看到的数据视图。
- 模式：是介于内模式和外模式之间的中间层，是数据的逻辑组织方式。

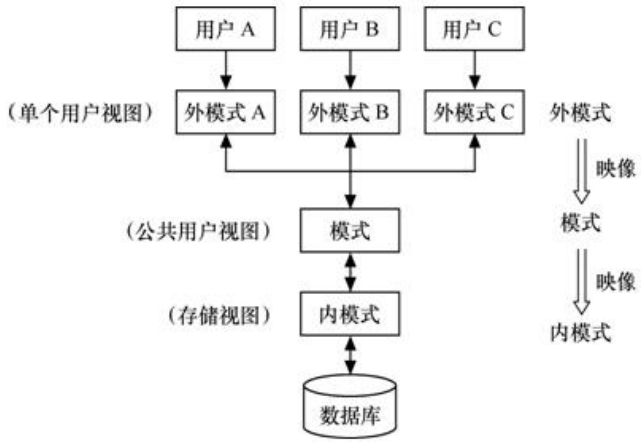


图 2-12　数据库的三级模式结构

外模式是面向每类用户的数据需求的视图，而模式描述的是一个部门或公司的全体数据。换句话说，外模式可以有许多，每一个都或多或少地抽象表示整个数据库的某一部分数据；而模式只有一个，它是对包含现实世界业务中的全体数据的抽象表示，注意这里的抽象指的是记录和字段这些更加面向用户的概念，而不是位和字节那些面向机器的概念。内模式也只有一个，它表示数据库的物理存储。

1．外模式

外模式也称为用户模式或子模式，它的内容来自模式。外模式是对现实系统中用户感兴趣的整体数据的局部描述，用于满足数据库不同用户对数据的需求。外模式是对数据库用户能够看见和使用的局部数据的逻辑结构和特征的描述，是数据库整体数据结构（即模式）的子集或局部重构。

外模式通常是模式的子集。一个数据库可以有多个外模式。由于它是各个用户的数据视图，如果不同的用户在应用需求、看待数据的方式、对数据保密要求等方面存在差异，则其外模式的描述就是不同的。即使对模式中同样的数据，在外模式中的结构、类型、长度等都可以不同。

例如：学生性别信息（学号，姓名，性别）视图就是表 2-1 所示关系的子集，它是宿舍分配部门所关心的信息，是学生基本信息的子集。又例如，学生成绩（学号，姓名，课程号，成绩）外模式是任课教师所关心的信息，这个外模式的数据就是表 2-1 的学生表（模式）和表 2-2 的选课表（模式）所含信息的组合（或称为重构）。

外模式同时也是保证数据库安全的一个措施。每个用户只能看到和访问其所对应的外模式中的数据，并屏蔽其不需要的数据，因此保证不会出现由于用户的误操作和有意破坏而造成数据损失。例如，假设有职工信息表，结构如下：

职工表（职工号，姓名，所在部门，基本工资，职务工资，奖励工资）

如果不希望一般职工看到每个职工的奖励工资，则可生成一个包含一般职工可以看的信息的外模式，结构如下：

职工信息（职工号，姓名，所在部门，基本工资，职务工资）

这样就可保证一般用户不会看到“奖励工资”项。

外模式也是关系的或接近关系的，外模式对应到关系数据库中是“外部视图”或简称为“视图”。视图的概念我们将在第 6 章介绍。

2．模式

模式也称为逻辑模式或概念模式，是对数据库中全体数据的逻辑结构和特征的描述，是所有用户的公共数据视图。模式表示数据库中的全部信息，其形式要比数据的物理存储方式抽象。它是数据库结构的中间层，既不涉及数据的物理存储细节和硬件环境，也与具体的应用程序、所使用的应用开发工具和环境无关。

模式由许多概念记录类型的值构成。例如，可以包含学生记录值的集合，课程记录值的集合，选课记录值的集合，等等。概念记录既不等同于外部记录，也不等同于存储记录，它是数据的一种逻辑表达。

模式实际上是数据库数据在逻辑级上的视图。一个数据库只有一种模式。数据库模式以某种数据模型为基础，综合地考虑了所有用户的需求，并将这些需求有机地结合成一个逻辑整体。定义数据库模式时不仅要定义数据的逻辑结构，比如数据记录由哪些数据项组成，数据项的名字、类型、取值范围等，而且还要定义数据之间的联系，定义与数据有关的安全性、完整性要求。

模式不涉及存储字段的表示，不涉及存储记录对列、索引、指针或其他存储的访问细节。如果模式以这种方式真正地实现了数据独立性，那么根据这些模式定义的外模式也会有很强的独立性。

关系数据库中的模式一定是关系的，关系数据库管理系统提供了模式定义语言（DDL）来定义数据库的模式。

3．内模式

内模式也称为存储模式。内模式是对整个数据库的底层表示，它描述了数据的存储结构，比如数据的组织与存储方式，是顺序存储、B 树存储还是散列存储、索引按什么方式组织、是否加密等。注意，内模式与物理层不一样，它不涉及物理记录的形式（即物理块或页，输入 / 输出单位），也不考虑具体设备的柱面或磁道大小。换句话说，内模式假定了一个无限大的线性地址空间，地址空间到物理存储的映射细节是与特定系统有关的，并不反映在体系结构中。

内模式不是关系的，它是数据的物理存储方式。其实，不管是什么系统，其内模式都是一样的，都是存储记录、指针、索引、散列表等。事实上，关系模型与内模式无关，它关心的是用户的数据视图。

2.5.3　模式映像与数据独立性

数据库的三级模式是对数据的三个抽象级别，它把数据的具体组织留给 DBMS，使用户能逻辑、抽象地处理数据，而不必关心数据在计算机中的具体表示方式与存储方式。为了能够在内部实现这三个抽象层的联系和转换，数据库管理系统在三个模式之间提供了以下两级映像（见图 2-12）：

- 外模式/模式映像。
- 模式/内模式映像。

正是这两级映像功能保证了数据库中的数据能够具有较高的逻辑独立性和物理独立性，使数据库应用程序不随数据库数据的逻辑或存储结构的变动而变动。

1．外模式/模式映像

模式描述的是数据的全局逻辑结构，外模式描述的是数据的局部逻辑结构。对应于同一个模式可以有多个外模式。对于每个外模式，数据库管理系统都有一个外模式到模式的映像，它定义了该外模式与模式之间的对应关系，即如何从外模式找到其对应的模式。这些映像定义通常包含在各自的外模式描述中。

当模式改变时（比如，增加新的关系、新的属性、改变属性的数据类型等），可由数据库管理员用外模式定义语句，调整外模式到模式的映像，从而保持外模式不变。由于应用程序一般是依据数据的外模式编写的，因此也不必修改应用程序，从而保证了程序与数据的逻辑独立性。

2．模式/内模式映像

模式/内模式映像定义了数据库的逻辑结构与物理存储之间的对应关系，该映像关系通常被保存在数据库的系统表（由数据库管理系统自动创建和维护，用于存放维护系统正常运行的表）中。当数据库的物理存储改变了，比如选择了另一个存储位置，只需要对模式/内模式映像做相应的调整，就可以保持模式不变，从而也不必改变应用程序。因此，保证了数据与程序的物理独立性。

在数据库的三级模式结构中，模式（即全局逻辑结构）是数据库的中心与关键，

它独立于数据库的其他层。设计数据库时也是首先设计数据库的逻辑模式。

数据库的内模式依赖于数据库的全局逻辑结构，但它独立于数据库的用户视图（也就是外模式），也独立于具体的存储设备。内模式将全局逻辑结构中所定义的数据结构及其联系按照一定的物理存储策略进行组织，以达到较好的时间与空间效率。

数据库的外模式面向具体的用户需求，它定义在模式之上，但独立于内模式和存储设备。当应用需求发生变化，相应的外模式不能满足用户的要求时，就需要对外模式做相应的修改以适应这些变化。因此设计外模式时应充分考虑到应用的扩充性。

原则上，应用程序都是在外模式描述的数据结构上编写的，而且它应该只依赖于数据库的外模式，并与数据库的模式和存储结构独立（但目前很多应用程序都是直接针对模式进行编写的）。不同的应用程序有时可以共用同一个外模式。数据库管理系统提供的两级映像功能保证了数据库外模式的稳定性，从而从底层保证了应用程序的稳定性，除非应用需求本身发生变化，否则应用程序一般不需要修改。

数据与程序之间的独立性，使得数据的定义和描述可以从应用程序中分离出来。另外，由于数据的存取由 DBMS 负责管理和实施，因此，用户不必考虑存取路径等细节，从而简化了应用程序的编制，减少了对应用程序的维护和修改工作。

习　题

一、选择题

1．数据库三级模式结构的划分，有利于

A．数据的独立性　　B．管理数据库文件

C．建立数据库　　D．操作系统管理数据库

2．在数据库的三级模式中，描述数据库中全体数据的逻辑结构和特征的是

A．内模式　　B.模式　　C．外模式　　D．用户模式

3．下列关于数据库中逻辑独立性的说法，正确的是

A. 当内模式发生变化时，模式可以不变
B. 当内模式发生变化时，应用程序可以不变
C. 当模式发生变化时，应用程序可以不变
D. 当模式发生变化时，内模式可以不变

4. 下列模式中，用于描述单个用户数据视图的是
A．内模式　　B．模式
C．外模式　　D．存储模式

5. 数据库中的数据模型三要素是指
A．数据结构、数据对象和数据共享
B．数据结构、数据操作和数据完整性约束
C．数据结构、数据操作和数据的安全控制
D．数据结构、数据操作和数据的可靠性

6. 下列关于 E-R 模型中联系的说法，错误的是
A．一个联系最多只能关联 2 个实体
B．联系可以可以是一对一的
C．一个联系可以关联 2 个或 2 个以上的实体
D．联系的种类是由客观世界业务决定的

7. 数据库中的三级模式以及模式间的映像提供了数据的独立性。下列关于两级映像的说法，正确的是
A．外模式到模式的映像是由应用程序实现的，模式到内模式的映像是由 DBMS 实现的
B．外模式到模式的映像是由 DBMS 实现的，模式到内模式的映像是由应用程序实现的
C．外模式到模式的映像以及模式到内模式的映像都是由 DBMS 实现的
D．外模式到模式的映像以及模式到内模式的映像都是由应用程序实现的

8. 下列关于概念层数据模型的说法，错误的是
A．概念层数据模型应该采用易于用户理解的表达方式
B．概念层数据模型应该比较易于转换成组织层数据模型
C．在进行概念层数据模型设计时，需要考虑具体的 DBMS 的特点
D．在进行概念层数据模型设计时，重点考虑的内容是用户的业务逻辑

二、简答题

1. 解释数据模型的概念，为什么要将数据模型分成概念层数据模型和组织层数据模型？
2. 组织层数据模型都有哪些？目前最常用的是哪个？
3. 实体之间的联系有几种？请为每一种联系举出一个例子。
4. 说明实体-联系模型中的实体、属性和联系的概念。
5. 指明下列实体间联系的种类：
（1）教研室和教师（假设一个教师只属于一个教研室，一个教研室可有多名教师）。
（2）商店和顾客。
（3）国家和首都（假设国家的首都不会变化，一个国家只有一个首都）
（4）飞机和乘客。
6. 数据库包含哪三级模式？试分别说明每一级模式的作用？
7. 数据库管理系统提供的两级映像的作用是什么？它带来了哪些功能？

第 3章

关系数据库

关系数据库是用数学的方法来处理数据库中的数据，它支持关系数据模型，现在绝大多数数据库管理系统都是关系型数据库管理系统。本章将介绍关系数据模型的基本概念和术语、关系的完整性约束以及关系操作，并介绍关系数据库的数学基础——关系代数。

3.1 关系数据模型

关系数据库使用关系数据模型组织数据，这种思想源于数学，最早提出类似方法的是 CODASYL 于 1962 年发表的“信息代数”一文， 1968 年 David Child 在计算机上实现了集合论数据结构。而真正系统、严格的提出关系数据模型的是 IBM 的研究员 E.F.Codd，他于 1970 年在美国计算机学会会刊(《Communication of the ACM》)上发表了题为“A Relational Model of Data for Shared Data Banks”的论文，开创了数据库系统的新纪元。以后，他连续发表了多篇论文，奠定了关系数据库的理论基础。

关系模型由关系模型的数据结构、关系模型的操作集合和关系模型的完整性约束三部分组成，这三部分也称为关系模型的三要素。下面我们首先介绍这三个方面的基本概念。

3.1.1 数据结构

关系数据模型源于数学，它用二维表来组织数据，而这个二维表在关系数据库中就称为关系。关系数据库就是表或者说是关系的集合。

关系系统要求让用户所感觉的数据就是一张张表。在关系系统中，表是逻辑结构而不是物理结构。实际上，系统在物理层可以使用任何有效的存储结构来存储数据，比如，有序文件、索引、哈希表、指针等。因此，表是对物理存储数据的一种抽象表示——对很多存储细节的抽象，如存储记录的位置、记录的顺序、数据值的表示以及记录的访问结构，如索引等，对用户来说都是不可见的。

3.1.2 数据操作

关系数据模型给出了关系操作的能力。关系数据模型中的操作包括：

- 传统的关系运算：并（Union）、交（Intersection）、差（Difference）、广义笛卡尔乘积（Extended Cartesian Product）；
- 专门的关系运算：选择(Select)、投影(Project)、连接(Join)、除(Divide)；
- 有关的数据操作：查询(Query)、插入(Insert)、删除(Delete)和更改(Update)。

关系模型的操作对象是集合（或表），而不是单个的数据行，也就是说，关系模型中操作的数据以及操作的结果（查询操作的结果）都是完整的集合（或表），这些集合可以是只包含一行数据的集合，也可以是不包含任何数据的空集合。而非在关系模型数据库中典型的操作是一次一行或一次一个记录。因此，集合处理能力是关系型数据库区别于其他类型数据库的一个重要特征。

在非关系模型中，各个数据记录之间是通过指针等方式连接的，当要定位到某条记录时，需要用户自己按指针的链接方向遍历查找，我们称这种查找方式为用户“导航”。而在关系数据模型中，由于是按集合进行操作，因此，用户只需要指定数据的定位条件，数据库管理系统就可以自动定位到该数据记录，而不需要用户来导航。这也是关系数据模型在数据操作

上与非关系模型的本质区别。

例如，若采用层次数据模型，对第 2 章图 2-7 所示的层次结构，若要查找“计算机学院软件工程教研室的张海涛老师的信息”，则首先需要从根节点的“学院”开始，根据“计算机”学院指向的“教研室”结点的指针，找到“教研室”层次，然后在“教研室”层次中逐个查找（这个查找过程也许是通过各结点间的指针实现的），直到找到“软件工程”结点，然后根据“软件工程”结点指向“教师”结点的指针，找到“教师”层次，最后再在“教师”层次中逐个查找教师名为“张海涛”的结点，此时该结点包含的信息即所要查找的信息。这个过程的示意图如图 3-1 所示，其中的虚线表示沿指针的逐层查找过程。

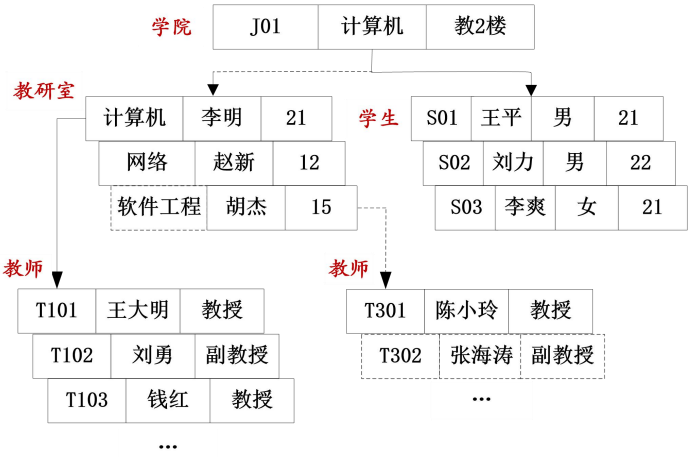


图 3-1 层次模型的查找过程示意图

而如果是在关系模型中查找信息，比如在表 3-1 所示的“学生”关系中查找“信息管理系学号为 0821101 的学生的详细信息”，用户只需提出这个要求即可，其余的工作就交给数据库管理系统来实现了。对用户来说，这显然比在层次模型中查找数据要简单得多。

数据库数据的操作主要包括四种：查询、插入、删除和更改数据。关系数据库中的信息只有一种表示方式，就是表中的行列位置有明确的值。这种表示是关系系统中唯一可行的方式（当然，这里指的是逻辑层）。特别地，关系数据库中没有连接一个表到另一个表的指针。在表 3-1 和表 3-2 所示关系中，表 3-1 所示的学生表的第一行数据与表 3-2 所示的学生选课表中的第 1 行有联系（当然也与第 2、3、4 行有联系），因为学生 0811101 选了课程。但在关系数据库中这种联系不是通过指针来实现的，而是通过学生表中“学号”列的值与学生选课表中“学号”列的值关联的（学号值相等）。但在非关系系统中，这些信息一般由指针来表示，这种指针对用户来说是可见的。因此，在非关系模型中，用户需要知道数据之间的指针链接关系。

表 3-1　　学生

学　　号	姓　　名	年　　龄	性　　别	所　在　系
0811101	李勇	21	男	计算机系
0811102	刘晨	20	男	计算机系
0811103	王敏	20	女	计算机系
0821101	张立	20	男	信息管理系
0821102	吴宾	19	女	信息管理系

表 3-2　　选课

学　　号	课　程　号	成　　绩
0811101	C001	96
0811101	C002	80
0811101	C003	84
0811101	C005	62
0811102	C001	92
0811102	C002	90
0811102	C004	84
0821102	C001	76
0821102	C004	85
0821102	C005	73

需要注意的是，当我们说关系数据库中没有指针时，并不是指在物理层没有指针，实际上，在关系数据库的物理层也使用指针，但所有这些物理层的存储细节对用户来说都是不可见的，用户所看到的物理层就是存放数据的数据库文件，他们能够看到的就是这些文件的文件名、存放位置等上层信息，而没有指针这样的底层信息。

关系操作是通过关系语言实现的，关系语言的特点是高度非过程化的。所谓非过程化是指：

- 用户不必关心数据的存取路径和存取过程，用户只需要提出数据请求，数据库管理系统就会自动完成用户请求的操作；
- 用户也没有必要编写程序代码来实现对数据的重复操作。

3.1.3　数据完整性约束

在数据库中数据的完整性是指保证数据正确性的特征。数据完整性是一种语义概念，它包括两个方面：

- 与现实世界中应用需求的数据的相容性和正确性；
- 数据库内数据之间的相容性和正确性。

例如，每个学生的学号必须是唯一的，性别只能是“男”和“女”，学生所选的课程必须是已经开设的课程等。因此，数据库是否具有数据完整性特征关系到数据库系统能否真实的反映现实世界情况，数据完整性是数据库的一个非常重要的内容。

数据完整性由一组完整性规则定义，而关系模型的完整性规则是对关系的某种约束条件。在关系数据模型中一般将数据完整性分为三类，即实体完整性、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（也称为引用完整性）是关系模型必须满足的完整性约束，是系统级的约束。用户定义的完整性主要是限制属性的取值在有意义的范围内，比如限制性别的取值范围为“男”和“女”。这个完整性约束也被称为域的完整性，它属于应用级的约束。数据库管理系统应该提供对这些数据完整性的支持。

3.2 关系模型的基本术语与形式化定义

在关系模型中，将现实世界中的实体、实体与实体之间的联系都用关系来表示，关系模型源于数学，它有自己严格的定义和一些固有的术语。

3.2.1 基本术语

关系模型采用单一的数据结构——关系来表示实体以及实体之间的联系，并且用直观的观点来看，关系就是二维表。

下面介绍关系模型中的有关术语。

1．关系（relation）

通俗的讲，关系就是二维表，二维表的名字就是关系的名字，表 3-1 所示的关系名是“学生”。

2．属性（attribute）

二维表中的每个列称为一个**属性**（或叫字段），每个属性有一个名字，称为属性名。二维表中对应某一列的值称为属性值；二维表中列的个数称为关系的元数。如果一个二维表有 n 个列，则称其为 n 元关系。表 3-1 所示的学生关系有学号、姓名、年龄、性别、所在系五个属性，是一个五元关系。

3．值域（domain）

二维表中属性的取值范围称为**值域**。例如在表 3-1 所示关系中，“年龄”列的取值为大于 0 的整数，“性别”列的取值为“男”和“女”两个值，这些都是列的值域。

4．元组（tuple）

二维表中的一行数据称为一个**元组**。表 3-1 所示学生关系中的元组有：

（0811101，李勇，21，男，计算机系）

（0811102，刘晨，20，男，计算机系）

（0811103，王敏，20，女，计算机系）

（0821101，张立，20，男，信息管理系）

（0821102，吴宾，19，女，信息管理系）

5．分量（component）：

元组中的每一个属性值称为元组的一个**分量**，n 元关系的每个元组有 n 个分量。例如，对于元组（0811101，李勇，21，男，计算机系），有 5 个分量，对应“学号”属性的分量是“0811101”、对应“姓名”属性的分量是“李勇”、对应“年龄”属性的分量是“21”、对应“性别”属性的分量是“男”，对应“所在系”属性的分量是“计算机系”。

6．关系模式（relation schema）

二维表的结构称为**关系模式**，或者说，关系模式就是二维表的表框架或表头结构。设有关系名为 R，属性分别为 A_1，A_2，…，A_n，则关系模式可以表示为：

R（A_1，A_2，…，A_n）

对每个 A_i（i＝1，…，n）还包括该属性到值域的映像，即属性的取值范围。例如，表 3-1 所示关系的关系模式为：

学生（学号，姓名，性别，年龄，所在系）

如果将关系模式理解为数据类型，则关系就是该数据类型的一个具体值。

7．关系数据库（relation database）

对应于一个关系模型的所有关系的集合称为关系数据库。

8．候选键

如果一个属性或属性集的值能够唯一标识一个关系的元组而又不包含多余的属性，则称该属性或属性集为**候选键**（candidate key）。比如，学生（学号，姓名，性别，年龄，所在系）的候选键是：学号。

候选键又称为候选关键字或候选码。在一个关系上可以有多个候选键。例如，假设为学生关系增加了“身份证号”列，则学生（学号，姓名，性别，年龄，所在系，身份证号）的

候选键就有两个：学号和身份证号。

9．主键

当一个关系中有多个候选键时，可以从中选择一个作为主键（Primary key）。每个关系只能有一个主键。

主键也称为主码或主关键字，是表中的属性或属性组，用于唯一地确定一个元组。主键可以由一个属性组成，也可以由多个属性共同组成。例如，表 3-1 所示的“学生”关系，学号是主键，因为学号的一个取值可以唯一地确定一个学生。而表 3-2 所示的“选课”关系的主键就由学号和课程号共同组成。因为一个学生可以修多门课程，而且一门课程也可以有多个学生选，因此，只有将学号和课程号组合起来才能共同确定一行记录。我们称由多个属性共同组成的主键为复合主键。当某个表是由多个属性共同作主键时，我们就用括号将这些属性括起来，表示共同作为主键。比如，表 3-2 所示的“选课”关系的主键是（学号，课程号）。

注意，我们不能根据关系在某个时刻所存储的内容来决定其主键，这样做是不可靠的，这样做只能是猜测。关系的主键与其实际的应用语义有关、与关系模式的设计者的意图有关的。例如，对于表 3-2 所示的“选课”关系，用（学号，课程号）作为主键在一个学生对一门课程只能有一次考试的前提下是成立的，如果实际情况是一个学生对一门课程可以有多次考试，则用（学号，课程号）作主键就不够了，因为一个学生对一门课程有多少次考试，则其（学号，课程号）的值就会重复多少遍。如果是这种情况，就必须为这个关系添加新的列，比如“考试次数”，并用（学号，课程号，考试次数）作为主键。

10．主属性（primary attribute）和非主属性（nonprimary attribute）

包含在任一候选键中的属性称为**主属性**。不包含在任一候选键中的属性称为**非主属性**。

关系中的术语很多可以与现实生活中的表格所使用的术语进行对应，如表 3-3 所示。

表 3-3　　术语对比

关系术语	一般的表格术语
关系名	表名
关系模式	表头（表所含列的描述）
关系	（一张）二维表
元组	记录或行
属性	列
分量	一条记录中某个列的值

3.2.2 形式化定义

在关系模型中，无论是实体还是实体之间的联系均由单一的结构类型表示——关系。关系模型是建立在集合论的基础上的，本节我们将从集合论的角度给出关系数据结构的形式化定义。

1．关系的形式化定义

为了给出关系的形式化定义，首先定义笛卡尔积：

设 D_1，D_2，…，D_n 为任意集合，定义笛卡尔积 D_1，D_2，…，D_n 为：

$$D_1 \times D_2 \times \cdots \times D_n = \{(d_1, d_2, \cdots, d_n) \mid d_i \in D_i, i=1, 2, \cdots, n\}$$

其中每一个元素（d_1，d_2，…，d_n）称为一个 n 元组（n-tuple），简称元组。元组中每一个 d_i 称为是一个分量。

比如设：

D_1＝{计算机系，信息管理系}

D_2＝{李勇，刘晨，吴宾}

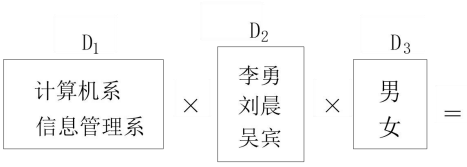
D_3＝{男，女}

则 $D_1 \times D_2 \times D_3$ 笛卡尔积为：

$D_1 \times D_2 \times D_3$＝{（计算机系，李勇，男），（计算机系，李勇，女），
（计算机系，刘晨，男），（计算机系，刘晨，女），
（计算机系，吴宾，男），（计算机系，吴宾，女），
（信息管理系，李勇，男），（信息管理系，李勇，女），
（信息管理系，刘晨，男），（信息管理系，刘晨，女），
（信息管理系，吴宾，男），（信息管理系，吴宾，女）}

其中（计算机系，李勇，男）、（计算机系，刘晨，男）等都是元组。“计算机系”、“李勇”、 “男’等都是分量。

笛卡尔积实际上就是一个二维表，上述笛卡尔积的运算如图 3-2 所示。



D_1	D_2	D_3
计算机系	李勇	男
计算机系	李勇	女
计算机系	刘晨	男
计算机系	刘晨	女
计算机系	吴宾	男
计算机系	吴宾	女
信息管理系	李勇	男
信息管理系	李勇	女
信息管理系	刘晨	男
信息管理系	刘晨	女
信息管理系	吴宾	男
信息管理系	吴宾	女

图 3-2 笛卡尔乘积示意图

图 3-2 中，笛卡尔积的任意一行数据就是一个元组，它的第一个分量来自 D_1，第二个分量来自 D_2，第三个分量来自 D_3。笛卡尔积就是所有这样的元组的集合。

根据笛卡尔积的定义可以给出关系的形式化定义：笛卡尔积 D_1，D_2，…，D_n 的任意一个子集称为 D_1，D_2，…，D_n 上的一个 n 元关系。

形式化的关系定义同样可以把关系看成二维表，给表中的每个列取一个名字，称为属性。n 元关系有 n 个属性，一个关系中的属性的名字必须是唯一的。属性 D_i 的取值范围（i＝1，2，…，n）称为该属性的**值域**（domain）。

比如，上述例子中，取子集：

R＝{（计算机系，李勇，男），（计算机系，刘晨，男），（信息管理系，吴宾，女）}

就构成了一个关系，其二维表的形式如表 3-4 所示，把第一个属性命名为“所在系”，第二个属性命名为“姓名”，第三个属性命名为“性别”。

表 3-4 一个关系

所在系	姓名	性别
计算机系	李勇	男
计算机系	刘晨	男
信息管理系	吴宾	女

从集合论的观点也可以将关系定义为：关系是一个有 K 个属性的元组的集合。

2．对关系的限定

关系可以看成是二维表，但并不是所有的二维表都是关系。关系数据库对关系有一些限定，归纳起来有如下几个方面：

（1）关系中的每个分量都必须是不可再分的最小属性。即每个属性都不能再被分解为更小的属性，这是关系数据库对关系的最基本的限定。例如，表 3-5 就不满足这个限定，因为在这个表中，“高级职称人数”不是最小的属性，它是由两个属性组成的一个复合属性。对于这种情况只需要将“高级职称人数”属性分解为“教授人数”和“副教授人数”两个属性即可，如表 3-6 所示，这时这个表就是一个关系。

表 3-5　　包含复合属性的表

系名	人数	高级职称人数	
		教授人数	副教授人数
计算机系	51	8	20
信息管理系	40	6	18
通讯工程系	43	8	22

不是最小属性

表 3-6　　不包含复合属性的表

系　名	人　数	教授人数	副教授人数
计算机系	51	8	20
信息管理系	40	6	16
通讯工程系	43	8	18

（2）表中列的数据类型是固定的，即列中的每个分量都是同类型的数据，来自相同的值域。

（3）不同列的数据可以取自相同的值域，每个列称为一个属性，每个属性有不同的属性名。

（4）关系表中列的顺序不重要，即列的次序可以任意交换，不影响其表达的语义。

（5）行的顺序也不重要，交换行数据的顺序不影响关系的内容。其实在关系数据库中并没有第 1 行、第 2 行等这样的概念，而且数据的存储顺序也与数据的输入顺序无关，数据的输入顺序不影响对数据库数据的操作过程，也不影响其操作效率。

（6）同一个关系中的元组不能重复，即在一个关系中任意两个元组的值不能完全相同。

3.3 完整性约束

数据完整性是指数据库中存储的数据是有意义的或正确的，也就是和现实世界相符。关系模型中的数据完整性规则是对关系的某种约束条件。它的数据完整性约束主要包括三大类：实体完整性、参照完整性和用户定义的完整性。

3.3.1 实体完整性

实体完整性是保证关系中的每个元组都是可识别的和唯一的。

实体完整性是指关系数据库中所有的表都必须有主键，而且表中不允许存在如下记录：

- 无主键值的记录。
- 主键值相同的记录。

若某记录没有主键值，则此记录在表中一定是无意义的。因为关系模型中的每一行记录都对应客观存在的一个实例或一个事实。比如，表 3-1 中的第一行数据描述的是“李勇”这个学生。如果将表 3-1 中的数据改为表 3-7 所示的数据，可以看到，第 1 行和第 4 行数据没有主键值，查看其他列的值发现这两行数据的其他各列的值都是一样的，于是会产生这样的疑问：到底是存在名字、年龄、性别完全相同的两个学生，还是重复存储了李勇学生的信息？这就是缺少主键值时造成的情况。如果为其添加主键值为表 3-8 所示的数据，则可以判定在计算机系有两个姓名、年龄、性别完全相同的学生。如果为其添加主键值为表 3-9 所示的数据，则可以判定在这个表中有重复存储的记录，而在数据库中存储重复的数据是没有意义的。

表 3-7　　缺少主键值的学生表

学　　号	姓　　名	年　　龄	性　　别	所　在　系
	李勇	21	男	计算机系
0811102	刘晨	20	男	计算机系
0811103	王敏	20	女	计算机系
	李勇	21	男	计算机系
0821101	张立	20	男	信息管理系
0821102	吴宾	19	女	信息管理系

表 3-8　　主键值均不同的学生表

学　　号	姓　　名	年　　龄	性　　别	所　在　系
0811101	李勇	21	男	计算机系
0811102	刘晨	20	男	计算机系
0811103	王敏	20	女	计算机系
0811104	李勇	21	男	计算机系
0821101	张立	20	男	信息管理系
0821102	吴宾	19	女	信息管理系

表 3-9　　主键值有重复的学生表

学　　号	姓　　名	年　　龄	性　　别	所　在　系
0811101	李勇	21	男	计算机系
0811102	刘晨	20	男	计算机系
0811103	王敏	20	女	计算机系
0811101	李勇	21	男	计算机系
0821101	张立	20	男	信息管理系

0821102	吴宾	19	女	信息管理系

当为表定义了主键时，数据库管理系统会自动保证数据的实体完整性，即保证不允许存在主键值为空的记录以及主键值重复的记录。

关系数据库中主属性不能取空值。在关系数据库中的空值是特殊的标量常数，它代表未定义的或者有意义但目前还处于未知状态的值。比如当向表 3-2 所示的“选课”关系中插入一行记录时，在学生还没有考试之前，其成绩是不确定的，因此，此列的值就为空。空值用“NULL”表示。

3.3.2 参照完整性

参照完整性也称为引用完整性。现实世界中的实体之间往往存在着某种联系，在关系模型中，实体以及实体之间的联系都是用关系来表示的，这样就自然存在着关系与关系之间的引用。参照完整性就是描述实体之间的联系的，这里的实体之间可以是不同的实体，也可以是同一个实体。

例 1．学生实体和班实体可以用下面的关系模式表示，其中主键用下划线标识：

学生（学号，姓名，性别，班号，年龄）

班（班号，所属专业，人数）

这两个关系模式之间存在着属性的引用关系，即“学生”关系中的“班号”引用或者是参照了“班”关系的主键“班号”。显然，“学生”关系中的“班号”的值必须是确实存在的班的班号的值，即在“班”关系中有该班号的记录。这种限制一个关系中某列的取值受另一个关系中某列的取值范围约束的特点就称为参照完整性。

例 2．学生、课程以及学生与课程之间的选课关系可以用如下三个关系模式表示，其中主键用下划线标识：

学生（学号，姓名，性别，专业，年龄）

课程（课程号，课程名，学分）

选课（学号，课程号，成绩）

这三个关系模式间也存在着属性的引用。“选课”关系中的“学号”引用了“学生”关系中的主键“学号”，即，“选课”关系中的“学号”的值必须是确实存在的学生的学号，也就是在“学生”关系中有这个学生的记录。同样，“选课”关系中的“课程号”引用了“课程”关系中的主键“课程号”，即“选课”中的“课程号”也必须是“课程”中存在的课程号。

与实体间的联系类似，不仅两个或两个以上的关系间可以存在引用关系，而且同一个关系的内部属性之间也可以存在引用关系。

例 3．有关系模式：职工（职工号，姓名，性别，直接领导职工号）

在这个关系模式中，“职工号”是主键，“直接领导职工号”属性表示该职工的直接领导的职工号，这个属性的取值就参照了该关系中“职工号”属性的取值，即“直接领导职工号”必须是确实存在的一个职工。

进一步定义外键：

定义：设 F 是关系 R 的一个或一组属性，如果 F 与关系 S 的主键相对应，则称 F 是关系 R 的**外键**（Foreign Key），并称关系 R 为参照关系（Referencing Relation），关系 S 为被参照关系（Referenced Relation）或目的关系（Target Relation）。关系 R 和关系 S 不一定是不同的关系。

显然，目标关系 S 的主键 K_s 和参照关系 R 的外键 F 必须定义在同一个域上。

在例 1 中，“学生”关系中的“班号”属性与“班”关系中的主键“班号”对应，因此，

“学生”关系中的“班号”是外键，引用了“班“关系中的”班号“（主键）。这里，“班”关系是被参照关系，“学生”关系是参照关系。

可以用图 3-3 所示的图形化的方法形象地表达参照和被参照关系。“班”和“学生”的参照与被参照关系的图形化表示如图 3-4（a）所示。

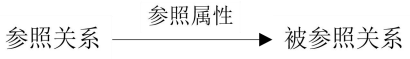


图 3-3 关系的参照表示图

在例 2 中，“选课”关系中的“学号”属性与“学生”关系中的主键“学号”对应，“课程号”属性与“课程”关系的主键“课程号”对应，因此，“选课”关系中的“学号”属性和“课程号”属性均是外键。这里“学生”关系和“课程”关系均为被参照关系，“选课”关系为参照关系，其参照关系图如图 3-4（b）所示。

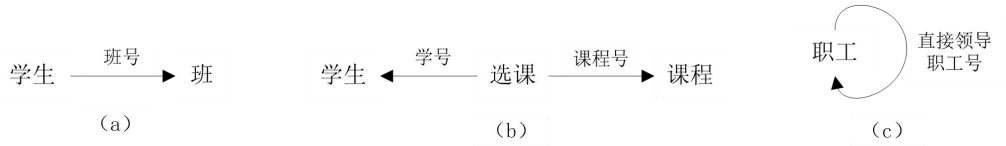


图 3-4 关系的参照图

在例 3 中，职工关系中的“直接领导职工号”属性与本身所在关系的主键“职工号”属性对应，因此，“直接领导职工号”是外键。这里，“职工”关系即是参照关系也是被参照关系，其参照关系图如图 3-4（c）所示。

需要说明的是，外键并不一定要与相对应的主键同名（如例 3）。但在实际应用中，为了便于识别，当外键与相应的主键属于不同的关系时，通常都给它们取相同的名字。

参照完整性规则就是定义外键与被参照的主键之间的引用规则。

对于外键，一般应符合如下要求：

- 或者值为空；
- 或者等于其所参照的关系中的某个元组的主键值。

例如，对于职工与其所在的部门可以用如下两个关系模式表示：

职工（职工号，职工名，部门号，工资级别）

部门（部门号，部门名）

其中，“职工”关系的“部门号”是外键，它参照了“部门”关系的“部门号”。如果某新来职工还没有被分配到具体的部门，则其“部门号”就为空值；如果职工已经被分配到了某个部门，则其部门号就有了确定的值（非空值）。

主键要求必须是非空且不重的，但外键无此要求。外键可以有重复值，这点我们从表 3-2 可以看出。

3.3.3 用户定义的完整性

用户定义的完整性也称为域完整性或语义完整性。任何关系数据库管理系统都应该支持实体完整性和参照完整性，除此之外，不同的数据库应用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应用领域定义的数据约束条件。它反映某一具体应用所涉及的数据必须满足应用语义的要求。

用户定义的完整性实际上就是指明关系中属性的取值范围，也就是属性的域，这样可以限制关系中属性的取值类型及取值范围，防止属性的值与应用语义矛盾。例如，学生的考试成绩的取值范围为 0~100，或取{优，良，中，及格，不及格}。

3.4 关系代数

关系模型源于数学，关系是由元组构成的集合，可以通过关系的运算来表达查询要求，而关系代数恰恰是关系操作语言的一种传统的表示方式，它是一种抽象的查询语言。

关系代数是一种纯理论语言，它定义了一些操作，运用这些操作可以从一个或多个关系中得到另一个关系，而不改变源关系。因此，关系代数的操作数和操作结果都是关系，而且一个操作的输出可以是另一个操作的输入。关系代数同算术运算一样，可以出现一个套一个的表达式。关系在关系代数下是封闭的，正如数字在算术操作下是封闭的一样。

关系代数是一种单次关系（或者说是集合）语言，即所有元组可能来自多个关系，但是用不带循环的一条语句处理。关系代数命令的语法形式有多种，本书采用的是一套通用的符号表示方法。

关系代数的运算对象是关系，运算结果也是关系。与一般的运算一样，运算对象、运算符和运算结果是关系代数的三大要素。

关系代数的运算可分为以下两大类：

- 传统的集合运算。这类运算完全把关系看成是元组的集合。传统的集合运算包括集合的广义笛卡尔积运算、并运算、交运算和差运算。
- 专门的关系运算。这类运算除了把关系看成是元组的集合外，还通过运算表达了查询的要求。专门的关系运算包括选择、投影、连接和除运算。

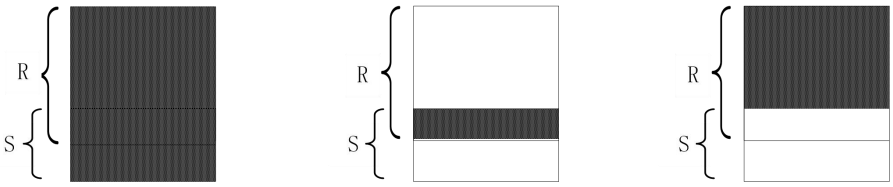
关系代数中的运算符可以分为四类：集合运算符、专门的关系运算符、比较运算符和逻辑运算符。表 3-10 列出了这些运算符，其中比较运算符和逻辑运算符是配合专门的关系运算符来构造表达式的。

表 3-10 关系运算符

运算符		含义
传统的集合运算	∪	并
	∩	交
	−	差
	×	广义笛卡尔积
专门的关系运算	σ	选择
	∏	投影
	⋈	连接
	÷	除
比较运算符	>	大于
	<	小于
	=	等于
	≠	不等于
	≤	小于等于
	≥	大于等于
逻辑运算符	¬	非
	∧	与
	∨	或

3.4.1 传统的集合运算

传统的集合运算是二目运算，设关系 R 和 S 均是 n 元关系，且相应的属性值取自同一个值域，则可以定义三种运算：并运算（∪）、交运算（∩）和差运算（－），但广义笛卡尔积并不要求参与运算的两个关系的对应属性取自相同的域。并、交、差运算的功能示意图如图 3-5 所示。



(a)R∪S　　(b)R∩S　　(c)R-S

图 3-5 并、交、差运算示意图

现在我们以图 3-6（a）和 3-6（b）所示的两个关系为例，说明这三种传统的集合运算。

顾客号	姓名	性别	年龄
S01	张宏	男	45
S02	李丽	女	34
S03	王敏	女	28

(a) 顾客表 A

顾客号	姓名	性别	年龄
S02	李丽	女	34
S04	钱景	男	50
S06	王平	女	24

(b) 顾客表 B

图 3-6 描述顾客信息的两个关系

1．并运算

设关系 R 与关系 S 均是 n 目关系，关系 R 与关系 S 的并记为：

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

其结果仍是 n 目关系，由属于 R 或属于 S 的元组组成。

图 3-7（a）显示了图 3-6（a）和图 3-6（b）两个关系的并运算结果。

顾客号	姓名	性别	年龄
S01	张宏	男	45
S02	李丽	女	34
S03	王敏	女	28
S04	钱景	男	50
S06	王平	女	24

(a) 顾客表 A∪顾客表 B

顾客号	姓名	性别	年龄
S02	李丽	女	34

(b) 顾客表 A∩顾客表 B

顾客号	姓名	性别	年龄
S01	张宏	男	45
S03	王敏	女	28

(c) 顾客表 A - 顾客表 B

图 3-7 集合的并、交、差运算示意图

2．交运算

设关系 R 与关系 S 均是 n 目关系，则关系 R 与关系 S 的交记为：

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

其结果仍是 n 目关系，由属于 R 并且也属于 S 的元组组成。

图 3-7（b）显示了图 3-6（a）和 3-6（b）两个关系的交运算结果。

3．差运算

设关系 R 与关系 S 均是 n 目关系，则关系 R 与关系 S 的差记为：

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

其结果仍是 n 目关系，由属于 R 并且不属于 S 的元组组成。

图 3-7（c）显示了图 3-6（a）和 3-6（b）两个关系的差运算结果。

4．广义笛卡尔积

广义笛卡尔积不要求参加运算的两个关系具有相同的目数。

两个分别为 m 目和 n 目的关系 R 和关系 S 的广义笛卡尔积是一个有（m＋n）个列的元组的集合。元组的前 m 个列是关系 R 的一个元组，后 n 个列是关系 S 的一个元组。若 R 有 K_1 个元组，S 有 K_2 个元组，则关系 R 和关系 S 的广义笛卡尔积有 $K_1 \times K_2$ 个元组，记做：

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

$t_r \hat{} t_s$ 表示由两个元组 t_r 和 t_s 前后有序连接而成的一个元组。

任取元组 t_r 和 t_s，当且仅当 t_r 属于 R 且 t_s 属于 S 时，t_r 和 t_s 的有序连接即为 R×S 的一个元组。

实际操作时，可从 R 的第一个元组开始，依次与 S 的每一个元组组合，然后，对 R 的下一个元组进行同样的操作，直至 R 的最后一个元组也进行同样的操作为止。即可得到 R×S 的全部元组。

图 3-8 所示为广义笛卡尔积操作的示意图。

A	B
a1	b1
a2	b2

×

C	D	E
c1	d1	e1
c2	d2	e2
c3	d3	e3

=

A	B	C	D	E
a1	b1	c1	d1	e1
a1	b1	c2	d2	e2
a1	b1	c3	d3	e3
a2	b2	c1	d1	e1
a2	b2	c2	d2	e2
a2	b2	c3	d3	e3

图 3-8 广义笛卡尔积操作示意图

3.4.2 专门的关系运算

专门的关系运算包括：选择、投影、连接、除等操作，其中选择和投影为一元操作，连接和除为二元操作。

下面我们以表 3-11 到表 3-13 所示的三个关系为例，介绍专门的关系运算。各关系包含的属性的含义如下：

Student：Sno（学号），Sname（姓名），Ssex（性别），Sage（年龄），Sdept（所在系）。

Course：Cno（课程号），Cname（课程名），Credit（学分），Semester（开课学期），Pcno（直接先修课）。

SC：Sno（学号），Cno（课程号），Grade（成绩）。

表 3-11　　Student

Sno	Sname	Ssex	Sage	Sdept
0811101	李勇	男	21	计算机系
0811102	刘晨	男	20	计算机系
0811103	王敏	女	20	计算机系
0811104	张小红	女	19	计算机系
0821101	张立	男	20	信息管理系
0821102	吴宾	女	19	信息管理系
0821103	张海	男	20	信息管理系

表 3-12 Course

Cno	Cname	Credit	Semester	Pcno
C001	高等数学	4	1	NULL
C002	大学英语	3	1	NULL
C003	大学英语	3	2	C002
C004	计算机文化学	2	2	NULL
C005	VB	2	3	C004
C006	数据库基础	4	5	C007
C007	数据结构	4	4	C005

表 3-13 SC

Sno	Cno	Grade
0811101	C001	96
0811101	C002	80
0811101	C003	84
0811101	C005	62
0811102	C001	92
0811102	C002	90
0811102	C004	84
0821102	C001	76
0821102	C004	85
0821102	C005	73
0821102	C007	NULL
0821103	C001	50
0821103	C004	80

1. 选择（Selection）

选择运算是从指定的关系中选出满足给定条件（用逻辑表达式表

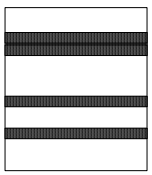


图 3-9 选择运算

达）的元组而组成一个新的关系。选择运算的功能如图 3-9 所示。

选择运算表示为：

$\sigma_F(R) = \{ t \mid t \in R \wedge F(t) = true \}$

其中 σ 是选择运算符，R 是关系名，t 是元组，F 是逻辑表达式，取逻辑“真”值或“假”值。

例 1．对表 3-11 所示的学生关系，从中选择计算机系学生信息的关系代数表达式为：

$\sigma_{Sdept='计算机系'}(Student)$

选择结果如表 3-14 所示。

表 3-14　　例 1 的选择结果

Sno	Sname	Ssex	Sage	Sdept
0811101	李勇	男	21	计算机系
0811102	刘晨	男	20	计算机系
0811103	王敏	女	20	计算机系
0811104	张小红	女	19	计算机系

2．投影（Projection）

投影运算是从关系 R 中选取若干属性，并用这些属性组成一个新的关系。其运算功能如图 3-10 所示。

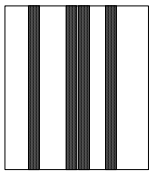


图 3-10 投影运算

投影运算表示为：

$\Pi_A(R) = \{ t.A \mid t \in R \}$

其中Π是投影运算符，R 是关系名，A 是被投影的属性或属性组。t.A 表示 t 这个元组中相应于属性（集）A 的分量，也可以表示为 t[A]。

投影运算一般由两个步骤完成：

（1） 选取出指定的属性，形成一个可能含有重复行的新关系；

（2） 删除重复行，形成结果关系。

例 2．对表 3-11 所示的学生关系，在 sname，sdept 两个列上进行投影运算，可以表示为：

$\Pi_{sname, sdept}(Student)$

投影结果如表 3-15 所示。

表 3-15　　例 2 的投影结果

Sname	Sdept
李勇	计算机系
刘晨	计算机系
王敏	计算机系
张小红	计算机系
张立	信息管理系
吴宾	信息管理系
张海	信息管理系

3．连接

连接运算用来连接相互之间有联系的两个关系，从而产生一个新的关系。这个过程由连接属性（字段）来实现。一般情况下连接属性是出现在不同关系中的语义相同的属性。连接是由笛卡尔乘积导出的，相当于把连接谓词看成选择公式。进行连接运算的两个关系通常是具有一对多联系的父子关系。

连接运算主要有如下几种形式：

- θ连接
- 等值连接（θ连接的特例）
- 自然连接
- 外部连接（简称外连接）
- 半连接。

θ连接运算一般表示为：

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

其中A和B分别是关系R和S上语义相同的属性或属性组，θ是比较运算符。连接运算从R和S的广义笛卡尔积R×S中选择（R关系）在A属性组上的值与（S关系）在B属性组上值满足比较运算符θ的元组。

连接运算中最重要也是最常用的连接有两个，一个是等值连接，一个是自然连接。

当θ为“＝”时的连接为等值连接，它是从关系R与关系S的广义笛卡尔积中选取A、B属性组值相等的那些元组，即：

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

自然连接是一种特殊的等值连接，它要求两个关系中进行比较的分量必须是相同的属性或属性组，并且在连接结果中去掉重复的属性列，使公共属性列只保留一个。即，若关系R和S具有相同的属性组B，则自然连接可记作：

$$R \bowtie S = \{t_r \hat{} t_s \mid t_r \in R \wedge t_s \in S \wedge t_r[A] = t_s[B]\}$$

一般的连接运算是从行的角度进行运算，但自然连接还需要去掉重复的列，所以是同时从行和列的角度进行运算。

自然连接与等值连接的差别为：

- 自然连接要求相等的分量必须有共同的属性名，等值连接则不要求；
- 自然连接要求把重复的属性名去掉，等值连接却不这样做。

表3-16　商品

商品号	商品名	进货价格
P01	34平面电视	2400
P02	34液晶电视	4800
P03	52液晶电视	9600

表3-17　销售

商品号	销售日期	销售价格
P01	2009-2-3	2200
P02	2009-2-3	5600
P01	2009-8-10	2800
P02	2009-2-8	5500
P01	2009-2-15	2150

例 3，设有如表 3-16 所示的“商品”关系和表 3-17 所示的“销售”关系，分别进行等值连接和自然连接运算：

等值连接：

自然连接：

等值连接的结果如表 3-18 所示，自然连接的结果如表 3-19 所示。

表 3-18 例 3 等值连接结果

商品号	商品名	进货价格	商品号	销售日期	销售价格
P01	34 平面电视	2400	P01	2009-2-3	2200
P01	34 平面电视	2400	P01	2009-8-10	2800
P01	34 平面电视	2400	P01	2009-2-15	2150
P02	34 液晶电视	4800	P02	2009-2-3	5600
P02	34 液晶电视	4800	P02	2009-2-8	5500

表 3-19 例 3 自然连接结果

商品号	商品名	进货价格	销售日期	销售价格
P01	34 平面电视	2400	2009-2-3	2200
P01	34 平面电视	2400	2009-8-10	2800
P01	34 平面电视	2400	2009-2-15	2150
P02	34 液晶电视	4800	2009-2-3	5600
P02	34 液晶电视	4800	2009-2-8	5500

从例 3 可以看到，当两个关系进行自然连接时，连接的结果由两个关系中公共属性值相等的元组构成。从连接的结果中我们看到，在“商品”关系中，如果某商品（这里是“P03”号商品）在“销售”关系中没有出现（即没有被销售过），则关于该商品的信息不会出现在连接结果中。也就是，在连接结果中会舍弃掉不满足连接条件（这里是两个关系中的“商品号”相等）元组。这种形式的连接称为内连接。

如果希望不满足连接条件的元组也出现在连接结果中，则可以通过**外连接**（OUTER JOIN）操作实现。外连接有三种形式：左外连接（LEFT OUTER JOIN）、右外连接（RIGHT OUTER JOIN）和全外连接（FULL OUTER JOIN）。

左外连接的连接形式为：$R\ *\bowtie\ S$

右外连接的连接形式为：$R\ \bowtie *\ S$

全外连接的连接形式为：$R\ *\bowtie *\ S$

左外连接的含义是把连接符号左边的关系（这里是 R）中不满足连接条件的元组也保留到连接后的结果中，并在连接结果中将该元组所对应的右边关系（这里是 S）的各个属性均置成空值（NULL）。

右外连接的含义是把连接符号右边的关系（这里是关系 S）中不满足连接条件的元组也保留到连接后的结果中，并在连接结果中将该元组对应的左边关系（这里是 R）的各个属性均置成空值（NULL）。

全外连接的含义是把连接符号两边的关系（R 和 S）中不满足连接条件的元组均保留到连接后的结果中，并在连接结果中将不满足连接条件的各元组的相关属性均置成空值（NULL）。

“商品”关系和“销售”关系的左外连接表达式为：

商品 $*\bowtie$ 销售

连接结果如表 3-20 所示。

表 3-20 商品和销售的左外连接结果

商品号	商品名	进货价格	销售日期	销售价格
P01	34 平面电视	2400	2009-2-3	2200
P01	34 平面电视	2400	2009-8-10	2800
P01	34 平面电视	2400	2009-2-15	2150
P02	34 液晶电视	4800	2009-2-3	5600
P02	34 液晶电视	4800	2009-2-8	5500
P03	52 液晶电视	9600	NULL	NULL

设有如表 3-21 和 3-22 所示的两个关系 R 和 S，则这两个关系的全外连接结果如表 3-23 所示。

A	B	C
a1	b1	c1
a2	b2	c1
a3	b1	c2
a4	b3	c1
a5	b2	c1

表 3-21 关系 R

E	B	D
e1	b1	d1
e2	b3	d1
e3	b1	d2
e4	b4	d1
e5	b3	d1

表 3-22 关系 S

表 3-23 关系 R 和 S 的全连接结果

A	B	C	E	D
a1	b1	c1	e1	d1
a1	b1	c1	e3	d2
a2	b2	c1	NULL	NULL
a3	b1	c2	e1	d1
a3	b1	c2	e3	d2
a4	b3	c1	e2	d1
a4	b3	c1	e5	d1
a5	b2	c1	NULL	NULL
NULL	b4	NULL	e4	d1

5. 除（Division）

（1）除法的简单描述

设关系 S 的属性是关系 R 的属性的一部分，则 R÷S 为这样一个关系：

- 此关系的属性是由属于 R 但不属于 S 的所有属性组成；
- R÷S 的任一元组都是 R 中某元组的一部分。但必须符合下列要求，即任取属于 R÷S 的一个元组 t，则 t 与 S 的任一元组连接后，都为 R 中原有的一个元组。

除法运算的示意图如图 3-11 所示。

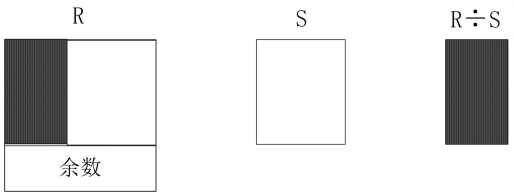


图 3-11 除法运算的示意图

（2）除法的一般形式

设有关系 R（X，Y）和 S（Y，Z），其中 X、Y、Z 为关系的属性组，则：

$$R（X，Y）÷ S（Y，Z）= R（X，Y）÷ \prod_{Y}（S）$$

（3）关系的除运算是关系运算中最复杂的一种，关系 R 与 S 的除运算的以上叙述解决了 R÷S 关系的属性组成及其元组应满足的条件要求，但怎样确定关系 R÷S 元组，仍然没有说清楚。为了说清楚这个问题，首先引入一个概念：

象集：给定一个关系 R（X，Y），X 和 Y 为属性组。定义，当 t［X］＝x 时，x 在 R 中的象集（Image Set）为：

$$Y_x=\{ t[Y] \mid t\in R \wedge t[X]=x \}$$

上式中：t［Y］和 t［X］分别表示 R 中的元组 t 在属性组 Y 和 X 上的分量的集合。

例如在表 3-11 所示的 Student 关系中，有一个元组值为：

（0821101，张立，男，20，信息管理系）

假设 X＝｛Sdept，Ssex｝，Y＝｛Sno，Sname，Sage｝，则上式中的 t[X]的一个值

x＝（信息管理系，男）

此时，Y_x为 t［X］＝x＝（信息管理系，男）时所有 t[Y]的值，即：

Yx=｛（0821101，张立，20），（0821103，张海，20）｝

也就是由信息管理系全体男生的学号、姓名、年龄所构成的集合。

又例如，对于表 3-13 所示的 SC 关系，如果设 X＝{Sno}，Y＝{Cno，Grade}，则当 X 取“0811101”时，Y 的象集为：

Y_x＝｛（C001，96），（C002，80），（C003，84），（C005，62）｝

当 X 取“0821103”时，Y 的象集为：

Yx＝｛（C001，50），（C004，80）｝

现在，我们再回过头来讨论除法的一般形式：

设有关系 R（X，Y）和 S（Y，Z），其中 X、Y、Z 为关系的属性组，则：

$$R÷S= \{t_r[X] \mid t_r\in R \wedge \prod_{Y}（S）\subseteq Y_x\}$$

图 3-8 给出了一个除运算的例子。

Sno	Cno
0611101	C001
0611101	C002
0611101	C003
0611101	C005
0611102	C001
0611102	C002
0611102	C004
0621102	C001
0621102	C004
0621102	C005
0621102	C007

÷

cno	cname
C001	高等数学
C005	VB

=

sno
0611101
0621102

图 3-12 除运算示例

图 3-12 所示的除结果的语义为至少选了“C001”和“C005”两门课程的学生的学号。

下面以表 3-11 至 3-13 所示 Student、Course 和 SC 关系为例，给出一些关系代数运算的例子。

例 5. 查询选了 C002 课程的学生的学号和成绩。

$\Pi_{Sno,\ Grade}(\sigma_{Cno\ =\ 'C002'}(SC))$

运算结果如图 3-13 所示。

Sno	Grade
061110	80

图 3-13 例 5 的结果

例 6. 查询信息管理系选了 C004 课程的学生的姓名和成绩。

由于学生姓名信息在 Student 关系中，而成绩信息在 SC 关系中，因此这个查询同时涉及到 Student 和 SC 两个关系。因此首先应对这两个关系进行自然连接，得到同一位学生的有关信息，然后再对连接的结果执行选择和投影操作。具体如下：

$\Pi_{Sname,\ Grade}(\sigma_{Cno\ =\ 'C004' \wedge Sdept='信息管理系'}(SC \bowtie Student))$

也可以写成：

$\Pi_{Sname,\ Grade}(\sigma_{Cno='C004'}(SC) \bowtie \sigma_{Sdept='信息管理系'}(Student))$

Sname	Grade
吴宾	85
张海	80

图 3-14 例 6 的结果

后一种实现形式是首先在 SC 关系中查询出选了“C004”课程的子集合，然后从 Student 关系中查询出“信息管理系”学生的子集合，然后再对这个子集合进行自然连接运算（Sno 相等），这种查询的执行效率会比第一种形式高。

运算结果如图 3-14 所示。

例 7. 查询选了第 2 学期开设的课程的学生姓名、所在系和所选的课程号。

这个查询的查询条件和查询列与二个关系有关：Student（包含姓名和所在系信息）以及 Course（包含课程号和开课学期信息）。但由于 Student 关系和 Course 关系之间没有可以进行连接的属性（要求必须语义相同），因此如果要让 Student 关系和 Course 关系进行连接，则必须要借助于 SC 关系，通过 SC 关系中的 Sno 与 Student 关系中的 Sno 进行自然连接，并通过 SC 关系中的 Cno 与 Course 关系中的 Cno 进行自然连接，可实现 Student 关系和 Course 关系之间的关联关系。

具体的关系代数表达式如下：

$\Pi_{Sname,\ Sdept,\ Cno}(\sigma_{Semester=2}(Course \bowtie SC \bowtie Student))$

也可以写成：

Sname	Sdept	Cno
李勇	计算机系	C003
刘晨	计算机系	C004
吴宾	信息管理系	C004

图 3-15 例 7 的结果

∏Sname, Sdept, Cno（σ $_{Semester=2}$（Course）⋈SC ⋈ Student）

运算结果如图 3-15 所示。

例 8. 查询选了“高等数学”且成绩大于等于 90 分的学生姓名、所在系和成绩。

这个查询涉及到 Student、SC 和 Course 三个关系，在 Course 关系中可以指定课程名（高等数学），从 Student 关系中可以得到姓名、所在系，从 SC 关系中可以得到成绩。

具体的关系代数表达式如下：

∏Sname, Sdept, Grade（σ Cname =‘高等数学’∧ Grade >= 90（Course ⋈ SC ⋈ Student））

也可以写成：

∏Sname, Sdept, Grade（σ Cname =‘高等数学’（Course）⋈ σ Grade >= 90（SC）⋈Student）

运算结果如图 3-16 所示。

例 9. 查询没选 VB 课程的学生的姓名和所在系。

实现这个查询的基本思路是从全体学生中去掉选了 VB 的学生，因此需要用到差运算。

具体的关系代数表达式如下：

∏Sname, Sdept(Student) - ∏Sname, Sdept（σ Cname =‘VB’(Course ⋈ SC ⋈ Student)）

也可以写成：

∏Sname, Sdept(Student) - ∏Sname, Sdept（σ Cname =‘VB’(Course)⋈ SC ⋈ Student）

运算结果如图 3-17 所示。

Sname	Sdept	Grade
李勇	计算机系	96
刘晨	计算机系	92

图 3-16 例 8 的结果

Sname	Sdept
张海	信息管理系

图 3-17 例 9 的结果

例 10. 查询选了全部课程的学生的姓名和所在系。

编写这个查询语句的关系代数表达式的思考过程如下：

(1) 学生选课情况可用$\Pi_{SNO,CNO}(SC)$表示；

(2) 全部课程可用$\Pi_{CNO}(Course)$表示；

(3) 查询选了全部课程的学生可用除法运算得到，即：

$$\Pi_{SNO,CNO}(SC) \div \Pi_{CNO}(Course)$$

这个关系代数表达式的操作结果为选了全部课程的学生的学号（Sno）的集合。

(4) 从得到的 Sno 集合再在 Student 关系中找到对应的学生姓名（Sname）和所在系（Sdept），这可以用自然连结和投影操作组合实现。最终的关系代数表达式为：

$$\Pi_{Sname,\ Sdept}(Student \bowtie (\Pi_{SNO,CNO}(SC) \div \Pi_{CNO}(Course)))$$

对所示数据来说该运算结果为空集合。

例 11. 查询计算机系选了第 1 学期开设的全部课程的学生的学号和姓名。

编写这个查询语句的关系代数表达式的思考过程与例 10 类似，只是将（2）改为：查询第 1 学期开设的全部课程，这可用Π_{CNO}(σ Semester=1(Course)表示。最终的关系代数表达式为：

∏Sno, Sname（σ Sdept =‘计算机系’（Student）⋈（$\Pi_{SNO,CNO}$(SC) ÷ ∏cno(σ Semester=1（Course））））

运算结果如图 3-18 所示。

表 3-24 对关系代数操作进行了总结。

Sno	Sname
0611101	李勇
0611102	刘晨

图 3-18 例 11 的结果

表 3-24 关系代数操作总结

操作	表示方法	功能
选择	σ_F（R）	产生一个新关系，其中只包含 R 中满足指定谓词的元组
投影	$\Pi_{a1,a2,\cdots,an}$（R）	产生一个新关系，该关系由指定的 R 中属性组成的一个 R 的垂直子集组成，并且去掉了重复的元组
连接	$R \underset{A\,\theta\,B}{\bowtie} S$	产生一个新关系，该关系包含了 R 和 S 的广义笛卡尔乘积中所有满足 θ 运算的元组
自然连接	$R \bowtie S$	产生一个新关系，由关系 R 和 S 在所有公共属性 x 上的相等连接得到，并且在结果中，每个公共属性只保留一个
（左）外连接	$R \ *\bowtie S$	产生一个新关系，将 R 在 S 中无法找到匹配的公共属性的 R 中的元组也保留在新关系中，并将对应 S 关系的各属性值均置为空
并	R∪S	产生一个新关系,它由 R 和 S 中所有不同的元组构成。R 和 S 必须是可进行并运算的
交	R∩S	产生一个新关系，它由既属于 R 又属于 S 的元组构成。R 和 S 必须是可进行交运算的
差	R-S	产生一个新关系，它由属于 R 但不属于 S 的元组构成。R 和 S 必须是可进行差运算的
广义笛卡尔积	R×S	产生一个新关系，它是关系 R 中的每个元组与关系 S 中的每个元组的并联的结果
除	R÷S	产生一个属性集合 C 上的关系，该关系的元组与 S 中的每个元组组合都能在 R 中找到匹配的元组，这里 C 是属于 R 但不属于 S 的属性集合

关系运算的优先级按从高到低的顺序为：投影、选择、乘积、连接和除（同级）、交、并和差（同级）。

习 题

一、选择题

1. 下列关于关系中主属性的描述，错误的是

   A. 主键所包含的属性一定是主属性

   B. 外键所引用的属性一定是主属性

   C. 候选码所包含的属性都是主属性

   D. 任何一个主属性都可以唯一地标识表中的一行数据

2. 设有关系模式：销售（顾客号，商品号，销售时间，销售数量），若一个顾客可在不同时

间对同一产品购买多次，同一个顾客在同一时间可购买多种商品，则此关系模式的主键是

A．顾客号　　B．产品号

C．（顾客号，商品号）　　D．（顾客号，商品号，销售时间）

3. 关系数据库用二维表来组织数据。下列关于关系表中记录的说法，正确的是

A．顺序很重要，不能交换　　B．顺序不重要

C．按输入数据的顺序排列　　D．一定是有序的

4. 下列不属于数据完整性约束的是

A．实体完整性　　B．参照完整性

C．域完整性　　D．数据操作完整性

5. 下列关于关系操作的说法，正确的是

A．关系操作是基于集合的操作

B．在进行关系操作时，用户需要知道数据的存储位置

C．在进行关系操作时，用户需要知道数据的存储结构

D．用户可以在关系上直接进行行定位操作

6. 下列关于关系的说法，错误的是

A．关系中的每个属性都是不可再分的基本属性

B．关系中不允许出现值完全相同的元组

C．关系中不需要考虑元组的先后顺序

D．关系中属性顺序的不同，关系所表达的语义也不同

7. 下列关于关系代数中选择运算的说法，正确的是

A．选择运算是从行的方向选择集合中的数据，选择运算后的行数有可能减少

B．选择运算是从行的方向选择集合中的数据，选择运算后的行数不变

C．选择运算是从列的方向选择集合中的若干列，选择运算后的列数有可能减少

D．选择运算是从列的方向选择集合中的若干列，选择运算后的列数不变

8. 下列用于表达关系代数中投影运算的运算符是

A．σ　　B．∏　　C．⋈　　D．+

9. 下列关于关系代数中差运算结果的说法，正确的是

A．差运算的结果包含了两个关系中的全部元组，因此有可能有重复的元组

B．差运算的结果包含了两个关系中的全部元组，但不会有重复的元组

C．差运算的结果只包含两个关系中相同的元组

D．“A-B”差运算的结果由属于 A 但不属于 B 的元组组成

10. 设有如下三个关系，学生（学号，姓名，性别），课程（课程号，课程名，学分）和选课（学号，课程号，成绩）。现要查询赵飞选的课程的课程名和学分，下列关系代数表达式正确的是

A．$\Pi_{课程名,学分}(\ \sigma_{姓名='赵飞'}(学生)\ \infty\ 课程\ \infty\ 选课)$

B．$\Pi_{课程名,学分}(\ \sigma_{姓名='赵飞'}(学生)\ \infty\ 选课\ \infty\ 课程)$

C．$\Pi_{课程名,学分}(\ \sigma_{姓名='赵飞'}(学生\ \infty\ 课程\ \infty\ 选课))$

D．$\Pi_{课程名,学分}(\ \sigma_{姓名='赵飞'}(课程\ \infty\ 学生\ \infty\ 选课\ ))$

二、简答题

1. 试述关系模型的三个组成部分。

2. 解释下列术语的含义：

（1） 主键

（2） 候选键

（3） 关系

（4） 关系模式

（5） 关系数据库

3. 关系数据库的三个完整性约束是什么？各是什么含义？

4. 利用表 3-11 至 3-13 所给的三个关系，写出实现如下查询的关系代数表达式。

（1）查询“信息管理系”学生的选课情况，列出学号、姓名、课程号和成绩。

（2）查询“VB”课程的考试情况，列出学生姓名、所在系和考试成绩。

（3）查询考试成绩高于 90 分的学生的姓名、课程名和成绩。

（4）查询至少选修了 0821103 学生所选的全部课程的学生姓名和所在系。

（5）查询至少选了“C001”和“C002”两门课程的学生姓名、所在系和所选的课程号。

第 4章

SQL 语言基础及数据定义功能

用户使用数据库时需要对数据库进行各种各样的操作，如查询数据，添加、删除和修改数据，定义、修改数据模式等。DBMS 必须为用户提供相应的命令或语言，这就构成了用户和数据库的接口。接口的好坏会直接影响用户对数据库的接受程度。

数据库所提供的语言一般局限于对数据库的操作，它不是完备的程序设计语言，也不能独立地用来编写应用程序。

SQL（Structured Query Language，结构化查询语言）是用户操作关系数据库的通用语言。虽然叫结构化查询语言，而且查询操作确实是数据库中的主要操作，但并不是说 SQL 只支持查询操作，它实际上包含数据定义、数据查询、数据操作和数据控制等与数据库有关的全部功能。

SQL 已经成为关系数据库的标准语言，所以现在所有的关系数据库管理系统都支持 SQL。本章将主要介绍 SQL 语言支持的数据类型以及定义基本表和索引的功能。

4.1 SQL 语言概述

SQL 语言是操作关系数据库的标准语言，本节介绍 SQL 语言的发展过程、特点以及主要功能。

4.1.1 SQL 语言的发展

最早的 SQL 原型是 IBM 的研究人员在 20 世纪 70 年代开发的，该原型被命名为 SEQUEL（Structured English QUEry Language）。现在许多人仍将在这个原型之后推出的 SQL 语言发音为“sequel”，但根据 ANSI SQL 委员会的规定，其正式发音应该是“ess cue ell”。随着 SQL 语言的颁布，各数据库厂商纷纷在其产品中引入并支持 SQL 语言，尽管绝大多数产品对 SQL 语言的支持大部分是相似的，但它们之间还是存在一定的差异，这些差异不利于初学者的学习。因此，我们在本章介绍 SQL 时主要介绍标准的 SQL 语言，我们将其称为基本 SQL。

从 20 世纪 80 年代以来，SQL 就一直是关系数据库管理系统（RDBMS）的标准语言。最早的 SQL 标准是 1986 年 10 月由美国 ANSI（American National Standards Institute）颁布的。随后，ISO（International Standards Organization）于 1987 年 6 月也正式采纳它为国际标准，并在此基础上进行了补充，到 1989 年 4 月，ISO 提出了具有完整性特征的 SQL，并称之为 SQL-89。SQL-89 标准的颁布，对数据库技术的发展和数据库的应用都起了很大的推动作用。尽管如此，SQL-89 仍有许多不足或不能满足应用需求的地方。为此，在 SQL-89 的基础上，经过 3 年多的研究和修改，ISO 和 ANSI 共同于 1992 年 8 月颁布了 SQL 的新标准，即 SQL-92（或称为 SQL2）。SQL-92 标准也不是非常完备的，1999 年又颁布了新的 SQL 标准，称为 SQL-99 或 SQL3。

不同数据库厂商的数据库管理系统提供的 SQL 语言略有差别，本书主要介绍 Microsoft SQL Server 使用的 SQL 语言（称为 Transact-SQL，简称 T-SQL）的功能，其他的数据库管理系统使用的 SQL 语言绝大部分是一样的。

4.1.2 SQL 语言特点

SQL 之所以能够被用户和业界所接受并成为国际标准，是因为它是一个综合的、功能强大且又比较简捷易学的语言。SQL 语言集数据定义、数据查询、数据操作和数据控制功能于

一身，其主要特点如下。

（1）一体化

SQL 语言风格统一，可以完成数据库活动中的全部工作，包括创建数据库、定义模式、更改和查询数据以及安全控制和维护数据库等。这为数据库应用系统的开发提供了良好的环境。用户在数据库系统投入使用之后，还可以根据需要随时修改模式结构，并且可以不影响数据库的运行，从而使系统具有良好的可扩展性。

（2）高度非过程化

在使用 SQL 语言访问数据库时，用户没有必要告诉计算机“如何”一步步地实现操作，而只需要用 SQL 语言描述要“做什么”，然后由数据库管理系统自动完成全部工作。

（3）简洁

虽然 SQL 语言功能很强，但它只有为数不多的几条命令，另外，SQL 的语法也比较简单，接近自然语言（英语），因此容易学习和掌握。

（4）可以多种方式使用

SQL 语言可以直接以命令方式交互使用，也可以嵌入到程序设计语言中使用。现在很多数据库应用开发工具（比如 C#、PowerBuilder、Delphi 等）都将 SQL 语言直接融入到自身的语言当中，使用起来非常方便。这些使用方式为用户提供了灵活的选择余地。而且不管是哪种使用方式，SQL 语言的语法都是一样的。

4.1.3 SQL 语言功能概述

SQL 语言按其功能可分为 4 大部分：数据定义、数据查询、数据更改和数据控制。表 4-1 列出了实现这 4 部分功能的动词。

表 4-1 SQL 语言的主要功能

SQL 功能	动　词
数据定义	CREATE、DROP、ALTER
数据查询	SELECT
数据更改	INSERT、UPDATE、DELETE
数据控制	GRANT、REVOKE、DENY

数据定义功能用于定义、删除和修改数据库中的对象，本章介绍的关系表、第 6 章介绍的视图、索引等都是数据库对象；数据查询功能用于实现查询数据的功能，数据查询是数据库中使用最多的操作；数据更改功能用于添加、删除和修改数据库数据。数据更改功能在有些书中也被称为数据操纵功能，也可以将数据查询和数据更改统称为数据操作。数据控制功能用于控制用户对数据的操作权限。

本章介绍数据定义功能中定义关系表的功能，同时介绍如何定义一些主要的完整性约束。第 5 章介绍实现数据查询和数据更改功能的语句。在介绍这些功能之前，我们先介绍 SQL 语言所支持的数据类型。

4.2 SQL 语言支持的数据类型

关系数据库的表结构由列组成，列指明了要存储的数据的含义，同时指明了要存储的数据的类型，因此，在定义表结构时，必然要指明每个列的数据类型。

每个数据库厂商提供的数据库管理系统所支持的数据类型并不完全相同，而且与标准的 SQL 也有差异，这里主要介绍 Microsoft SQL Server 支持的常用数据类型，同时也列出了

对应的标准 SQL 数据类型，便于读者对比。

4.2.1 数值型

1. 准确型

准确型数值是指在计算机中能够精确存储的数据，比如整型数、定点小数等都是准确型数据。表 4-2 列出了 SQL Server 支持的准确型数据类型，同时列出了对应的 ISO SQL 支持的准确型数据类型。

表 4-2 准确数值类型

SQL Server 数据类型	ISO SQL 数据类型	说　　明	存储空间
Bigint		存储从 -2^{63}（-9 223 372 036 854 775 808）到 $2^{63}-1$（9 223 372 036 854 775 807）范围的整数	8 字节
Int	Integer	存储从 -2^{31}（-2 147 483 648）到 $2^{31}-1$（2 147 483 647）范围的整数	4 字节
Smallint	Smallint	存储从 -2^{15}（-32 768）到 $2^{15}-1$（32 767）范围的整数	2 字节
Tinyint		存储从 0 到 255 之间的整数	1 字节
Bit	Bit	存储 1 或 0	1 字节
numeric（p,q）或 decimal（p,q）	decimal	定点精度和小数位数。使用最大精度时，有效值从 $-10^{38}+1$ 到 $10^{38}-1$。其中，p 为精度，指定可以存储的十进制数字的最大个数。q 为小数位数，指定小数点右边可以存储的十进制数字的最大个数，$0 <= q <= p$。q 的默认值为 0	最多 17 字节

2. 近似型

近似型用于存储浮点型数据，表示在其数据类型范围内的所有数据在计算机中不一定都能精确地表示。

表 4-3 列出了 SQL Server 支持的近似数据类型，同时列出了对应的 ISO SQL 支持的近似数据类型。

表 4-3 近似数据类型

SQL Server 数据类型	ISO SQL 数据类型	说　　明	存储空间
float	float	存储从 -1.79E + 308 到 1.79E + 308 范围的浮点型数	4 字节或 8 字节
real		存储从 -3.40E + 38 到 3.40E + 38 范围的浮点型数	4 字节

4.2.2 字符串类型

字符串型数据由汉字、英文字母、数字和各种符号组成。目前字符的编码方式有两种：

一种是普通字符编码，另一种是统一字符编码（unicode）。普通字符编码指的是不同国家或地区的编码长度不一样，比如：英文字母的编码是 1 个字节（8 位），中文汉字的编码是 2 个字节（16 位）。统一字符编码是对所有语言中的字符均采用双字节（16 位）编码。

表 4-4 列出了 SQL Server 支持的字符串型数据类型。

表 4-4 字符串类型

SQL Server 数据类型	说　　明	存储空间
char(n)	固定长度的字符串类型，*n* 表示字符串的最大长度，取值范围为 1～8000	n 字节
varchar(n)	可变长度的字符串类型，*n* 表示字符串的最大长度，取值范围为 1～8000	字符数，1 个汉字算 2 个字符
text	可存储 2^{31}-1 (2 147 483 647) 个字符的大文本类型。用于存储超过 8000 字节的列数据	字符数，1 个汉字算 2 个字符
varchar(max)	最多可存储 2^{31}-1 个字符的大文本类型。用于存储超过 8000 字节的列数据	字符数，1 个汉字算 2 个字符
nchar(n)	固定长度的 Unicode 字符串类型，*n* 表示字符串的最大长度，取值范围为 1～4000	2*n 字节
nvarchar(n)	可变长度的 Unicode 字符串类型，*n* 表示字符串的最大长度，取值范围为 1～4000	2*字符数，1 个汉字算 1 个字符
ntext	最多可存储 2^{30}-1 (1 073 741 823) 个字符的统一字符编码文本。用于存储超过 8000 字节的列数据	2*字符数，1 个汉字算 1 个字符
nvarchar(max)	最多可存储 2^{30}-1 个字符的统一字符编码文本。用于存储超过 8000 字节的列数据	2*字符数，1 个汉字算 1 个字符
binary(n)	固定长度的二进制字符数据，*n* 表示最大长度，取值范围为 1～8000	n 字节
varbinary(n)	可变长度的二进制字符数据，*n* 的取值范围为 1～8000	字符数，1 个汉字算 2 个字符
image	大容量的、可变长度的二进制字符数据，最多可存储 2^{31}-1 个十六进制数字。用于存储超过 8000 字节的列数据	所输入数据的实际长度
varbinary(max)	可变长度的二进制数据，最多为 2^{31}-1 个十六进制数字。用于存储超过 8000 字节的列数据	所输入数据的实际长度

说明：

（1） 在 SQL Server 的未来版本中将删除 ntext、text 和 image 数据类型，因此在新的开发工作中应避免使用这些数据类型，而是使用新的 nvarchar(max)、varchar(max)和 varbinary(max)数据类型。

（2） SQL 中的字符串常量要用单引号括起来，比如'计算机系'。

（3） 定长字符类型表示不管实际字符需要多少空间，系统分配固定地字节数。

如果空间未被占满时，则系统自动用空格填充。

（4） 可变长字符类型表示按实际字符需要的空间进行分配。

4.2.3 日期时间类型

SQL Server 2008 比之前的 SQL Server 版本增加了很多新的日期和时间数据类型，以便更好地与 ISO 兼容。表 4-5 列出了 SQL Server 2008 支持的常用日期时间数据类型。

表 4-5 常用日期时间类型

日期时间类型	说明	存储空间
date	定义一个日期，范围为 0001-01-01 到 9999-12-31。字符长度 10 位，默认格式为：YYYY-MM-DD。YYYY 表示 4 位年份数字，范围从 0001 到 9999；MM 表示 2 位月份数字，范围从 01 到 12；DD 表示 2 位日的数字，范围从 01 到 31（最高值取决于具体月份）	3 字节
time[(n)]	定义一天中的某个时间，该时间基于 24 小时制。默认格式为：hh:mm:ss[.nnnnnnn]，范围为 00:00:00.0000000 到 23:59:59.9999999。精确到 100 纳秒。 n 为秒的小数位数，取值范围是 0 到 7 的整数。默认秒的小数位数是 7(100ns)	3~5 字节
datetime	定义一个采用 24 小时制并带有秒的小数部分的日期和时间，范围为 1753-1-1 到 9999-12-31，时间范围是 00:00:00 到 23:59:59.997。默认格式为：YYYY-MM-DD hh:mm:ss.nnn，n 为数字，表示秒的小数部分（精确到 0.00333 秒）	8 字节
smalldatetime	定义一个采用 24 小时制并且秒始终为零(:00)的日期和时间，范围为 1900-1-1 到 2079-6-6。默认格式为：YYYY-MM-DD hh:mm:00。精确到分钟	4 字节

说明：对于新的开发工作，应使用 time、date、datetime2 和 datetimeoffset 数据类型，因为这些类型符合 ISO SQL 标准，而且这三种类型提供了更高精度的秒数。datetimeoffset 为全局部署的应用程序提供时区支持。

datetime 用 4 个字节存储从 1900 年 1 月 1 日之前或之后的天数(以 1990 年 1 月 1 日为分界点，1900 年 1 月 1 日之前的日期的天数小于 0，1900 年 1 月 1 日之前的日期的天数大于 0)，用另外 4 个字节存储从午夜（00:00:00）后代表每天时间的毫秒数。

Smalldatetime 与 datetime 类似，它用 2 个字节存储从 1900 年 1 月 1 日之后的日期的天数，用另外 2 个字节存储从午夜（00:00:00）后代表每天时间的分钟数。

注意在使用日期时间类型的数据时也要用单引号括起来，比如：’2014-12-6’、’2014-12-6 12:03:32’。

4.3 数据定义功能

我们在本书第 2 章介绍过，为方便用户访问和管理数据库，关系数据库管理系统将数据划分为三个层次，每个层次用一个模式来描述，这就是外模式、模式和内模式，并将此称为数据库的三级模式结构。外模式和模式在关系数据库中分别对应视图和表，内模式对应索引

等内容。因此，SQL 的数据定义功能包括表定义、视图定义、索引定义等。除此之外，SQL 标准是通过对象（例如表）对 SQL 所基于的概念进行描述的，这些对象大部分是架构对象，即对象都属于一定的架构，因此，数据定义功能还包括架构的定义。表 4-6 列出了 SQL 数据定义功能包括的主要内容。

表 4-6　　SQL 数据定义功能

对　　象	创　　建	修　　改	删　　除
架构	CREATE SCHEMA		DROP SCHEMA
表	CREATE TABLE	ALTER TABLE	DROP TABLE
视图	CREATE VIEW	ALTER VIEW	DROP VIEW
索引	CREATE INDEX	ALTER INDEX	DROP INDEX

本章介绍架构和表的定义，第 5 章介绍索引和视图的定义方法。

4.3.1　架构的定义与删除

架构（schema，也称为模式）是数据库下的一个逻辑命名空间，可以存放表、视图等数据库对象，它是一个数据库对象的容器。如果将数据库比喻为一个操作系统，那么架构就相当于操作系统中的文件夹，而架构中的对象就相当于这个文件夹中的文件。因此，通过将同名表放置在不同架构中，使得一个数据库中可以包含名字相同的关系表。

一个数据库可以包含一个或多个架构，由特定的授权用户名所拥有。在同一个数据库中，架构的名字必须是唯一的。属于一个架构的对象称为架构对象，即它们依赖于该架构。架构对象的类型包括：基本表、视图、触发器等。

一个架构可以由零个或多个架构对象组成，架构名字可以是显式的，也可以是由 DBMS 提供的默认名。对数据库中对象的引用可以通过架构名前缀来限定。不带任何架构限定的 CREATE 语句都指的是在当前架构中创建对象。

1. 定义架构

定义架构的 SQL 语句为 CREATE SCHEMA，其语法格式如下：

```
CREATE SCHEMA 〈架构名〉] AUTHORIZATION 〈用户名〉
CREATE SCHEMA {
      〈架构名〉
    | AUTHORIZATION 〈所有者名〉
    |〈架构名〉 AUTHORIZATION 〈所有者名〉
    }
  [ { 表定义语句 | 视图定义语句
      | 授权语句 | 收权语句 | 拒绝权限语句 }
  ]
```

如果没有指定 schema_name，则架构名隐含为所有者名（owner_name）。

执行创建架构语句的用户必须具有数据库管理员的权限，或者是获得了数据库管理员授予的 CREATE SCHEMA 的权限。

例 1. 为用户“ZHANG”定义一个架构，架构名为“S_C”。

```
CREATE SCHEMA S_C AUTHORIZATION ZHANG
```

例 2. 为用户“ZHANG”定义一个用隐含名字的架构。

```
CREATE  SCHEMA  AUTHORIZATION ZHANG
```

这个示例用用户名“ZHANG”定义了一个架构，该架构的隐含名字为“ZHANG”。

定义架构实际上就是定义了一个命名空间,在这个空间中可以进一步定义该架构的数据库对象，比如表、视图等。

在定义架构时还可以同时定义表、视图以及为用户授权等，即可以在 CREATE SCHEMA 语句中包含 CREATE TABLE、CREATE VIEW、GRANT 等语句。

例 3．在定义架构的同时定义表（具体定义表的 SQL 语句我们将在 4.3.2 介绍）。

```
CREATE SCHEMA TEST AUTHORIZATION ZHANG
   CREATE TABLE T1( C1 INT,
                    C2 CHAR(10),
                    C3 SMALLDATETIME,
                    C4 NUMERIC(4,1))
```

该语句为用户“ZHANG”创建了一个名为“TEST”的架构，并在其中定义了一个表 T1，说明表 T1 包含在 TEST 架构中。

2．删除架构

在 SQL 中，删除架构的语句是 DROP SCHEMA，其语法格式如下：

```
DROP SCHEMA <架构名> { <CASCADE> | <RESTRICT> }
```

其中：

- CASCADE 选项：删除架构的同时将该架构中所有的架构对象一起全部删除。
- RESTRICT 选项：如果被删除的架构中包含有架构对象，则拒绝删除此架构。

不同数据库管理系统的 DROP SCHEMA 语句的语法格式和执行略有不同。SQL Server 2008 的 DROP SCHEMA 语句没有可选项，其语法格式为：

```
DROP  SCHEMA <架构名>
```

而且在 SQL Server 2008 中只能删除不包含任何架构对象的架构，如果架构中含有架构对象，则拒绝删除架构。用户必须首先删除或移出架构所包含的全部对象，然后再删除架构。

4.3.2　基本表

表是数据库中最重要的对象，它用于存储用户的数据。在了解了数据类型的知识后，我们就可以开始创建表了。关系数据库的表是二维表，包含行和列，创建表就是定义表所包含的每个列，包括：列名、数据类型、约束等。列名是为列取的名字，一般为便于记忆，最好取有意义的名字，比如“学号”或“Sno”，而不要取无意义的名字，比如 a1；列的数据类型说明了列的可取值范畴；列的约束更进一步限制了列的取值范围，这些约束包括：列取值是否允许为空、主键约束、外键约束、列取值范围约束等。

本节介绍表（或称为基本表）的创建、删除以及对表结构的修改。

1．定义表及完整性约束

定义基本表使用 SQL 语言数据定义功能中的 CREATE TABLE 语句实现，其一般格式如下。

```
CREATE  TABLE  [<架构名>.]<表名> (
    { <列名>  <数据类型>  [ 列级完整性约束定义 [ ... n ] ] }
    [ 表级完整性约束定义 ] [ ,... n ]
 )
```

注意，默认时 SQL 语言不区分大小写。

参数说明如下。

- <表名>是所要定义的基本表的名字。
- <列名>是表中所包含的属性列的名字。
- 在定义表的同时还可以定义与表有关的完整性约束条件，这些完整性约束条件都会

存储在系统的数据字典中。如果完整性约束只涉及表中的一个列，则这些约束条件可以在“列级完整性约束定义”处定义，也可以在“表级完整性约束定义”处定义；但某些涉及表中多个属性列的约束，必须在“表级完整性约束定义”处定义。

上述语法中用到了一些特殊的符号，比如[ ]，这些符号是文法描述的常用符号，而不是 SQL 语句的部分。我们简单介绍一下这些符号的含义（在后边的语法介绍中也要用到这些符号），有些符号在上述这个语法中可能没有用到。

方括号（[ ]）中的内容表示是可选的（即可出现 0 次或 1 次），比如[列级完整性约束定义]代表可以有也可以没有“列级完整性约束定义”。花括号（{}）与省略号（…）一起，表示其中的内容可以出现 0 次或多次。竖杠（|）表示在多个选项中选择一个，比如 term1 | term2 | term3，表示在三个选项中任选一项。竖杠也能用在方括号中，表示可以选择由竖杠分隔的子句中的一个，但整个子句又是可选的（也就是可以没有子句出现）

在定义基本表时可以同时定义数据的完整性约束。定义完整性约束时可以在定义列的同时定义，也可以将完整性约束作为独立的项定义。在列定义同时定义的约束称之为**列级完整性约束定义**，作为表的独立的一项定义的完整性约束称之为**表级完整性约束**。在列级完整性约束定义处可以定义如下约束。

- NOT NULL：非空约束。限制列取值非空。
- PRIMARY KEY：主键约束。指定本列为主键。
- FOREIGN KEY：外键约束。定义本列为引用其他表的外键。
- UNIQUE：唯一值约束。限制列取值不能重复。
- DEFAULT：默认值约束。指定列的默认值。
- CHECK：列取值范围约束。限制列的取值范围。

在上述约束中，NOT NULL 和 DEFAULT 只能定义在“列级完整性约束定义”处，其他约束均可在“列级完整性约束定义”和“表级完整性约束定义”处定义。

（1） 主键约束

定义主键的语法格式为：

PRIMARY KEY [(<列名> [, … n])]

如果在列级完整性约束处定义单列的主键，则可省略方括号中的内容。

（2） 外键约束

外键大多数情况下都是单列的，它可以定义在列级完整性约束处，也可以定义在表级完整性约束处。定义外键的语法格式为：

[FOREIGN KEY (<列名>)] REFERENCES <外表名>(<外表列名>)

如果是在列级完整性约束处定义外键，则可以省略“FOREIGN KEY (<列名>)”部分。

（3） 唯一值约束

唯一值约束用于限制一个列的取值不重复，或者是多个列的组合取值不重复。这个约束用在事实上具有唯一性的属性列上，比如每个人的身份证号码、驾驶证号码等均不能有重复值。定义 UNIQUE 约束时注意如下事项：

- 有 UNIQUE 约束的列允许有一个空值；
- 在一个表中可以定义多个 UNIQUE 约束；
- 可以在一个列或多个列上定义 UNIQUE 约束。

在一个已有主键的表中使用 UNIQUE 约束定义非主键列取值不重是很有用的，比如学生的身份证号码，“身份证号”列不是主键，但它的取值也不能重复，这种情况就必须使用 UNIQUE 约束。

定义唯一值约束的语法格式为：

UNIQUE [(<列名> [, … n])]

如果在列级完整性约束处定义单列的唯一值约束，则可省略方括号中的内容。

（4） 默认值约束

默认值约束用 DEFAULT 约束来实现，它用于提供列的默认值，即当在表中插入数据时，如果没有为有 DEFAULT 约束的列提供值，则系统自动使用 DEFAULTDefault 约束定义的默认值。

一个默认值约束只能为一个列提供默认值，且默认值约束必须是列级约束。

默认值约束的定义有两种形式，一种是在定义表时指定默认值约束，一种是在修改表结构时添加默认值约束。

（1）在创建表时定义 DEFAULT 约束

DEFAULT 常量表达式

（2）为已创建好的表添加 Default 约束

DEFAULT 常量表达式 FOR 列名

（5） 列取值范围约束

限制列取值范围用 CHECK 约束实现，该约束用于限制列的取值在指定范围内，即约束列的取值符合应用语义，例如，人的性别只能是“男”或“女”，工资必须大于 2000（假设最低工资为 2000）。需要注意的是，CHECK 所限制的列必须在同一个表中。

定义 CHECK 约束的语法格式为：

CHECK(逻辑表达式)

注意，如果 CHECK 约束是定义多列之间的取值约束，则只能在“表级完整性约束定义”处定义。

例 1. 用 SQL 语句创建如下三张表：Jobs（工作）表、Employees（职工）表，其结构如表 4-7～表 4-8 所示。

表 4-7 Jobs 表

列 名	含 义	数 据 类 型	约 束
Jid	工作编号	char(6)	主键
Descp	工作描述	nvarchar(20)	非空
EduReq	学历要求	nchar(6)	默认值：本科
MinSalary	最低工资	int	
MaxSalary	最高工作	int	大于等于最低工资

表 4-8 Employees 表

列 名	含 义	数 据 类 型	约 束
Eid	职工号	char(10)	主键
Ename	姓名	nchar(6)	非空
Sex	性别	nchar(1)	取值范围:“男”或“女”
BrithDate	出生日期	date	
JobDate	参加工作日期	dateTime	默认为系统当前日期时间

Sid	身份证号	char(18)	取值不重
Jid	所干工作编号	char(6)	外键，引用工作表的工作编号
Tel	联系电话	char(11)	

这两张表的创建语句如下：

```
CREATE TABLE Jobs (
  Jid          char(6)     PRIMARY KEY,     --在列级定义主键
  Descp        nchar(20) NOT NULL,
  EduReq   nchar(6)  DEFAULT '本科',
  MinSalary  int      ,
  MaxSalary  int      ,
  CHECK( MaxSalary >= MinSalary )      --多列的 CHECK 约束必须定义在表级
)

CREATE TABLE Employees (
  Eid        char(10)      ,
  Ename      nvarchar(20) NOT NULL,
  Sex         nchar(1)        CHECK( Sex = '男' OR Sex = '女'),
  BirthDate date           ,
  JobDate   datetime       DEFAULT GetDate(),
  Sid         char(18)        UNIQUE,
  Jid         char(6)       ,
  Tel        char(11)      ,
  PRIMARY KEY(Eid)        ,              --在表级定义主键
  FOREIGN KEY(Jid) REFERENCES Jobs(Jid)
)
```

注:

① “--”为 SQL 语句的单行注释符。

② GetDate()是 SQL Server 系统提供的函数，其功能是返回系统的当前日期和时间。

2. 修改表结构

在定义基本表之后，如果需求有变化，需要更改表的结构，可以使用 ALTER TABLE 语句实现。ALTER TABLE 语句可以对表添加列、删除列、修改列的定义，也可以添加和删除约束。

不同数据库产品的 ALTER TABLE 语句的格式略有不同，我们这里给出 SQL Server 支持的 ALTER TABLE 语句的简化语法格式，对于其他的数据库管理系统，可以参考它们的语言参考手册。

```
ALTER TABLE [<架构名>.]<表名>
{
    ALTER  COLUMN <列名> <新数据类型>      -- 修改列定义
  | ADD  <列名> <数据类型> [约束]              -- 添加新列
  | DROP  COLUMN <列名>                    -- 删除列
  | ADD  [constraint <约束名>] 约束定义    -- 添加约束
  | DROP  <约束名>                             -- 删除约束
}
```

例 1.为 Employees 表添加工资列，此列的列名为 Salary，数据类型为 int，允许空。

```
ALTER TABLE Employees
  ADD Salary  INT
```

例 2. 将 Jobs 表的 Descp 列的数据类型改为 NCHAR(40)。

```
ALTER TABLE Jobs
   ALTER COLUMN Descp NCHAR(40)
```

例 3. 删除 Employees 表的 Tel 列。

```
ALTER TABLE Employees
   DROP COLUMN Tel
```

例 4. 删除 Jobs 表中 MinSalary 列添加约束：大于等于 1600。

```
ALTER TABLE Jobs
   ADD CHECK( MinSalary >= 1600 )
```

3. 删除表

可以使用 DROP TABLE 语句删除表，其语法语句格式为：

```
DROP  TABLE  〈表名〉 {,〈表名〉}
```

例 5. 删除 Employees 表。

```
DROP TABLE Employees
```

注意删除表时必须先删外键所在表，然后再删除被参照的主键所在表。创建表时必须先建立被参照的主键所在表，后建立外键所在表。

习　题

一、选择题

1.　下列关于 SQL 语言特点的叙述，错误的是

A. 使用 SQL 语言访问数据库，用户只需提出做什么，而无需描述如何实现

B. SQL 语言比较复杂，因此在使用上比较难

C. SQL 语言可以在数据库管理系统提供的应用程序中执行，也可以在命令行方式下执行

D. 使用 SQL 语言可以完成任何数据库操作

2.　下列所述功能中，不属于 SQL 语言功能的是

A. 数据库和表的定义功能　　B. 数据查询功能

C. 数据增、删、改功能　　D. 提供方便的用户操作界面功能

3.　设某职工表中有用于存放年龄（整数）的列，下列类型中最合适年龄列的是

A. int　　B. smallint

C. tinyint　　D. bit

4.　设某列的类型是 char(10)，存放“数据库”，占用空间的字节数是

A. 10　　B. 20

C. 3　　D. 6

5.　设某列的类型是 nchar(10)，存放“数据库”，占用空间的字节数是

A. 10　　B. 20

C. 3　　D. 6

6.　设某列的类型是 varchar(10)，存放“数据库”，占用空间的字节数是

A．10　　B．20
C．3　　D．6

7．　设某列的类型是 nvarchar(10)，存放“abc”，占用空间的字节数是

A．10　　B．20
C．3　　D．6

8．　下列约束中用于限制列的取值范围的约束是

A．PRIMARY KEY　　B．CHECK
C．DEFAULT　　D．　UNIQUE

9．　下列约束中用于限制列取值不重的约束是

A．PRIMARY KEY　　B．CHECK
C．DEFAULT　　D．　UNIQUE

10．　下列关于 DEFAULT 约束的说法，正确的是

A．一个 DEFAULT 约束可用于一个表的多个列上
B．DEFAULT 约束只能作为表级完整性约束
C．DEFAULT 约束只能作为列级完整性约束
D．DEFAULT 约束既可作为表级完整性约束也可作为列级完整性约束

二、简答题

1．　SQL 语言的特点是什么？具有哪些功能？
2．　tinyint 类型定义的数据的取值范围是多少？
3．　SmallDatatime 类型精确到哪个时间单位？
4．　定点小数类型 numeric(p,q)中的 p 和 q 的含义分别是什么？
5．　char(n)和 nchar(n)中 n 的取值范围分别是多少？
6．　架构的作用是什么？
7．　写出定义如下架构的 SQL 语句。
（1）为用户“张三”定义一个架构，架构名为“图书”。
（2）为用户“Teacher”定义一个架构，架构名同用户名。