第 3 章　数据库基础知识

数据库技术是计算机软件领域的一个重要分支，它是因计算机信息系统与应用系统的需求而发展起来的。程序员应了解数据库的基本内容，理解数据库系统的总体框架，了解数据库系统在计算机系统中的地位以及数据库系统的功能。

3.1　基本概念

3.1.1　数据库系统

数据是描述事物的符号记录，它具有多种表现形式，可以是文字、图形、图像、声音和语言等。信息是对现实世界事物的存在方式或状态的反映。信息具有可感知、可存储、可加工、可传递和可再生等自然属性，信息已是社会各行各业不可缺少的资源，这也是信息的社会属性。数据是经过组织的位集合，而信息是具有特定释义和意义的数据。

数据库系统（DataBase System，DBS）由数据库、硬件、软件和人员 4 大部分组成。

（1）数据库。数据库（DataBase，DB）是指长期储存在计算机内、有组织、可共享的数据集合。数据库中的数据按一定的数学模型组织、描述和存储，具有较小的冗余度，较高的数据独立性和易扩展性，并可被各类用户共享。

（2）硬件。硬件是指计算机系统中的各种物理设备，包括存储数据所需的外部设备。硬件的配置应满足整个数据库系统的需要。

（3）软件。软件包括操作系统、数据库管理系统（Database Management System，DBMS）及应用程序。DBMS 是数据库系统中的核心软件，需要在操作系统的支持下工作，解决如何科学地组织和储存数据、高效地获取和维护数据。

（4）人员。主要包括系统分析员和数据库设计人员、应用程序员、最终用户和数据库管理员 4 类人员。

系统分析员负责应用系统的需求分析和规范说明，他们和用户及数据库管理员一起确定系统的硬件配置，并参与数据库系统的概要设计；数据库设计人员负责数据库中数据的确定、数据库各级模式的设计。

应用程序员负责编写使用数据库的应用程序，这些应用程序可对数据进行检索、建立、删除或修改。

最终用户应用系统提供的接口或利用查询语言访问数据库。

数据库管理员（DataBase Administrator，DBA）负责数据库的总体信息控制。其主要职责包括：决定数据库中的信息内容和结构；决定数据库的存储结构和存取策略；定义数据库的安全性要求和完整性约束条件；监控数据库的使用和运行；数据库的性能改进、数据库的重组和重构，以提高系统的性能。

3.1.2 数据库管理技术的发展

数据处理是对各种数据进行收集、存储、加工和传播的一系列活动。数据管理是数据处理的中心问题，是对数据进行分类、组织、编码、存储检索和维护。数据管理技术发展经历了 3 个阶段：人工管理阶段、文件系统阶段和数据库系统阶段。

1．人工管理阶段

早期的数据处理都是通过手工进行的，当时的计算机上没有专门管理数据的软件，也没有诸如磁盘之类的设备来存储数据，那时应用程序和数据之间的关系如图 3-1 所示。这种数据处理具有以下几个特点。

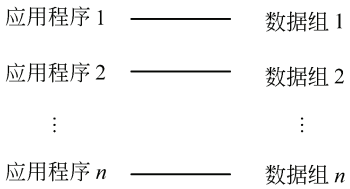


图 3-1　应用程序和数据的关系

（1）数据量较少。数据和程序一一对应，即一组数据对应一个程序，数据面向应用，独立性很差。由于不同应用程序所处理的数据之间可能会有一定的关系，因此会有大量的重复数据。

（2）数据不保存。因为在该阶段计算机主要用于科学计算，数据一般不需要长期保存，需要时输入即可。

（3）没有软件系统对数据进行管理。程序员不仅要规定数据的逻辑结构，而且在程序中还要使用其物理结构，包括存储结构的存取方法、输入/输出方式等。也就是说，数据对程序不具有独立性，一旦数据在存储器上改变物理地址，就需要改变相应的用户程序。

手工处理数据有两个特点：一是应用程序对数据的依赖性太强；二是数据组和数据组之间可能有许多重复的数据，造成数据冗余。

2．文件系统阶段

20 世纪 50 年代中期以后，计算机的硬件和软件技术飞速发展，除了科学计算任务外，计算机逐渐用于非数值数据的处理。由于大容量的磁盘等辅助存储设备的出现，使得专门管理辅助存储设备上的数据的文件系统应运而生。文件系统是操作系统中的一个子系统，它按一定的规则将数据组织成为一个文件，应用程序通过文件系统对文件中的数据进行存取和加工。文件系统对数据的管理，实际上是通过应用程序和数据之间的一种接口实现的，如图 3-2 所示。

文件系统的最大特点是解决了应用程序和数据之间的一个公共接口问题，使得应用程序采用统一的存取方法来操作数据。在文件系统阶段中，数据管理的特点如下。

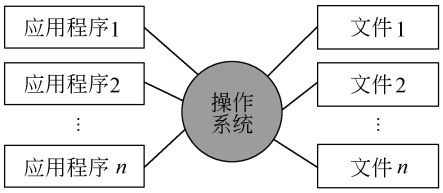


图 3-2　应用程序与文件的关系

（1）数据可以长期保留，数据的逻辑结构和物理结构有了区别，程序可以按名访问，不必关心数据的物理位置，由文件系统提供存取方法。

（2）数据不属于某个特定的应用，即应用程序和数据之间不再是直接的对应关系，可以重复使用。但是，文件系统只是简单地存取数据，相互之间并没有有机的联系，即数据存取依赖于应用程序的使用方法，不同的应用程序仍然很难共享同一数据文件。

（3）文件组织形式的多样化，有索引文件、链接文件和 Hash 文件等。但文件之间没有联系，相互独立，数据间的联系要通过程序去构造。

文件系统具有数据冗余度大、数据不一致和数据联系弱等缺点。

（1）数据冗余度大。文件与应用程序密切相关，相同的数据集合在不同的应用程序中使用时，经常需要重复定义、重复存储。例如，学生学籍管理系统中的学生情况，学生成绩管理系统的学生选课，教师教学管理的任课情况，所用到的数据很多都是重复的。这样，相同的数据不能被共享，必然导致数据的冗余。

（2）数据不一致性。由于相同数据重复存储，单独管理，给数据的修改和维护带来难度，容易造成数据的不一致。例如，人事处修改了某个职工的信息，但生产科该职工相应的信息却没有修改，造成同一个职工的信息在不同的部门结果不一样。

（3）数据联系弱。文件系统中数据组织成记录，记录由字段组成，记录内部有了一定的结构。但是，文件之间是孤立的，从整体上看没有反映现实世界事物之间的内在联系，因此很难对数据进行合理地组织以适应不同应用的需要。

3．数据库系统阶段

数据库系统是由计算机软件、硬件资源组成的系统，它实现了大量关联数据有组织地、动态地存储，方便多用户访问。它与文件系统的重要区别是数据的充分共享、交叉访问、与应用程序高度独立。

数据库系统阶段，数据管理的特点如下。

（1）采用复杂的数据模型表示数据结构。数据模型不仅描述数据本身的特点，还描述数据之间的联系。数据不再面向某个应用，而是面向整个应用系统。数据冗余明显减少，实现了数据共享。

（2）有较高的数据独立性。数据库也是以文件方式存储数据的，但它是数据的一种更高级

的组织形式。在应用程序和数据库之间由 DBMS 负责数据的存取，DBMS 对数据的处理方式和文件系统不同，它把所有应用程序中使用的数据以及数据间的联系汇集在一起，以便于应用程序查询和使用。这一阶段程序和数据的关系如图 3-3 所示。

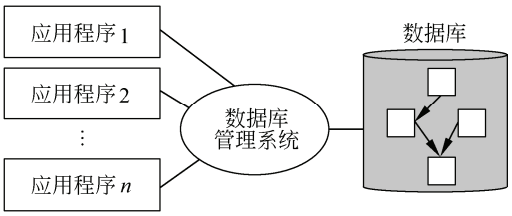


图 3-3　应用程序与数据库的关系

在数据库系统中，数据库对数据的存储按照统一结构进行，不同的应用程序都可以直接操作这些数据（即对应用程序的高度独立性）。数据库系统对数据的完整性、唯一性和安全性都提供一套有效的管理手段（即数据的充分共享性）。数据库系统还提供管理和控制数据的各种简单操作命令，使用户编写程序时容易掌握（即操作方便性）。

3.1.3　大数据

1．大数据产生背景

大数据（Big Data）产生的背景主要包括如下 4 个方面。

（1）数据来源和承载方式的变革。由于物联网、云计算、移动互联网等新技术的发展，用户在线的每一次点击，每一次评论，每一个视频点播，就是大数据的典型来源；而遍布地球各个角落的手机、PC、平板电脑及传感器成为数据来源和承载方式。可见，只有大连接与大交互，才有大数据。

（2）全球数据量出现爆炸式增长。由于视频监控、智能终端、网络商店等的快速普及，使得全球数据量出现爆炸式增长，未来数年数据量会呈现指数增长。根据麦肯锡全球研究院（MGI）估计，全球企业 2010 年在硬盘上存储了超过 7EB（1EB 等于 10 亿 GB）的新数据，而消费者在 PC 和笔记本等设备上存储了超过 6EB 新数据。据 IDC（Internet Data Center）预测，至 2020 年全球以电子式形存储的数据量将达 32ZB。

（3）大数据已经成为一种自然资源。许多研究者认为：大数据是“未来的新石油”，已成为一种新的经济资产类别。一个国家拥有数据的规模、活性及解释运用的能力，将成为综合国力的重要组成部分。

（4）大数据日益重要，不被利用就是成本。大数据作为一种数据资产当仁不让地成为现代商业社会的核心竞争力，不被利用就是企业的成本。因为，数据资产可以帮助和指导企业对全业务流程进行有效运营和优化，帮助企业做出最明智的决策。

2．大数据的特征

大数据（Big Data）是指“无法用现有的软件工具提取、存储、搜索、共享、分析和处理的海量的、复杂的数据集合”。业界通常用“4V”来概括大数据的特征。

大量化（Volume）指数据体量巨大。随着 IT 技术的迅猛发展，数据量级已从 TB 发展至 PB 乃至 ZB，可称海量、巨量乃至超量。当前，典型个人计算机硬盘的容量为 TB 量级，而一些大企业的数据量已经接近 EB 量级。

多样化（Variety）指数据类型繁多。相对于以往便于存储的以文本为主的结构化数据，非结构化数据越来越多，包括网络日志、音频、视频、图片、地理位置信息等，这些多类型的数据对数据的处理能力提出了更高要求。

价值密度低（Value）指大量的不相关信息导致价值密度的高低与数据总量的大小成反比。以视频为例，一部 1 小时的视频，在连续不间断的监控中，有用数据可能仅有一两秒。因此，如何通过强大的机器算法更迅速地完成数据的价值“提纯”，如何对未来趋势与模式的可预测分析、深度复杂分析（机器学习、人工智能 VS 传统商务智能咨询、报告等），成为目前大数据背景下亟待解决的难题。

快速化（Velocity）指处理速度快。大数据时代对其时效性要求很高，这是大数据区分于传统数据挖掘的最显著特征。因为，大数据环境下，数据流通常为高速实时数据流，而且需要快速、持续的实时处理；处理工具亦在快速演进，软件工程及人工智能等均可能介入。

3．理解大数据

大数据不仅仅是指海量的信息，更强调的是人类对信息的筛选、处理，保留有价值的信息，即让大数据更有意义，挖掘其潜在的“大价值”这才是对大数据的正确理解。为此有许多问题需要研究与解决。

（1）提高并发数据存取的性能要求及数据存储的横向扩展问题。目前，多从架构和并行等方面考虑解决。

（2）实现大数据资源化、知识化、普适化的问题。解决这些问题的关键是对非结构化数据的内容理解。

（3）非结构化海量信息的智能化处理问题。主要解决自然语言理解、多媒体内容理解、机器学习等问题。

大数据时代主要面临三大挑战：软件和数据处理能力、资源和共享管理以及数据处理的可信力。软件和数据处理能力是指应用大数据技术，提升服务能力和运作效率，以及个性化的服务，比如医疗、卫生、教育等部门。资源和共享管理是指应用大数据技术，提高应急处置能力和安全防范能力。数据处理的可信力是指需要投资建立大数据的处理分析平台，实现综合治理、

业务开拓等目标。

4．大数据产生的安全风险

2012 年瑞士达沃斯论坛上发布的《大数据大影响》报告称：“数据已成为一种新的经济资产类别，就像货币或黄金一样”。因此，大数据也带来了更多安全风险。

（1）大数据成为网络攻击的显著目标。在互联网环境下，大数据是更容易被“发现”的大目标。这些数据会吸引更多的潜在攻击者，如数据的大量汇集，使得黑客成功攻击一次就能获得更多数据，无形中降低了黑客的攻击成本，增加了“收益率”。

（2）大数据加大了隐私泄露风险。大量数据的汇集不可避免地加大了用户隐私泄露的风险。因为数据集中存储增加了泄露风险；另外，一些敏感数据的所有权和使用权并没有明确界定，很多基于大数据的分析都未考虑到其中涉及的个体隐私问题。

（3）大数据威胁现有的存储和安防措施。大数据存储带来新的安全问题。数据大集中的后果是复杂多样的数据存储在一起，很可能会出现将某些生产数据放在经营数据存储位置的情况，致使企业安全管理不合规。大数据的大小也影响到安全控制措施能否正确运行。安全防护手段的更新升级速度无法跟上数据量非线性增长的步伐，就会暴露大数据安全防护的漏洞。

（4）大数据技术成为黑客的攻击手段。在企业用数据挖掘和数据分析等大数据技术获取商业价值的同时，黑客也在利用这些大数据技术向企业发起攻击。黑客会最大限度地收集更多有用信息，比如社交网络、邮件、微博、电子商务、电话和家庭住址等信息，大数据分析使黑客的攻击更加精准。

（5）大数据成为高级可持续攻击的载体。传统的检测是基于单个时间点进行的基于威胁特征的实时匹配检测，而高级可持续攻击（APT）是一个实施过程，无法被实时检测。此外，大数据的价值低密度性，使得安全分析工具很难聚焦在价值点上，黑客可以将攻击隐藏在大数据中，给安全服务提供商的分析制造很大困难。黑客设置的任何一个会误导安全厂商目标信息提取和检索的攻击，都会导致安全监测偏离应有方向。

（6）大数据技术为信息安全提供新支撑。当然，大数据也为信息安全的发展提供了新机遇。大数据正在为安全分析提供新的可能性，对于海量数据的分析有助于信息安全服务提供商更好地刻画网络异常行为，从而找出数据中的风险点。对实时安全和商务数据结合在一起的数据进行预防性分析，可识别钓鱼攻击，防止诈骗和阻止黑客入侵。网络攻击行为总会留下蛛丝马迹，这些痕迹都以数据的形式隐藏在大数据中，利用大数据技术整合计算和处理资源有助于更有针对性地应对信息安全威胁，有助于找到攻击的源头。

3.2 数据模型

3.2.1 基本概念

模型是对现实世界特征的模拟和抽象，数据模型是对现实世界数据特征的抽象。人们常见的航模飞机、地图、建筑设计沙盘等都是具体的模型。最常用的数据模型分为概念数据模型和基本数据模型。

（1）概念数据模型。也称信息模型，是按用户的观点对数据和信息建模，是现实世界到信息世界的第一层抽象。它强调语义表达功能，易于用户理解，是用户和数据库设计人员交流的语言，主要用于数据库设计。这类模型中最著名的是实体联系模型（E-R 模型）。

（2）基本数据模型。按计算机系统的观点对数据建模，是现实世界数据特征的抽象，用于 DBMS 的实现。基本的数据模型有层次模型、网状模型、关系模型和面向对象模型（Object Oriented Model）。

从事物的客观特性到计算机中的具体表示涉及三个数据领域：现实世界、信息世界和机器世界。

（1）现实世界。现实世界的数据就是客观存在的各种报表、图表和查询要求等原始数据。计算机只能处理数据，所以首先要解决的问题是按用户的观点对数据和信息建模，即抽取数据库技术所研究的数据，分门别类，综合出系统所需要的数据。

（2）信息世界。是现实世界在人们头脑中的反映，人们用符号、文字记录下来。在信息世界中，数据库常用的术语是实体、实体集、属性和码。

（3）机器世界。是按计算机系统的观点对数据建模。机器世界中数据描述的术语有字段、记录、文件和记录码。

信息世界与机器世界相关术语的对应关系如下。

- 属性与字段。属性是描述实体某方面的特性，字段标记实体属性的命名单位。例如，用“书号、书名、作者名、出版社、日期”5 个属性描述书的特性，对应有 5 个字段。
- 实体与记录。实体表示客观存在并能相互区别的事物（如一个学生、一本书）；记录是字段的有序集合，一般情况下，一条记录描述一个实体。例如，“10121，DATABASE SYSTEM CONCEPTS,China Machine Press, 2014-2”描述的是一个实体，对应一条记录。
- 码与记录码。码也称为键，是能唯一区分实体的属性或属性集；记录码是唯一标识文件中每条记录的字段或字段集。
- 实体集与文件。实体集是具有共同特性的实体的集合，文件是同一类记录的汇集。例

如，所有学生构成了学生实体集，而所有学生记录组成了学生文件。

- 实体型与记录型。实体型是属性的集合，如表示学生学习情况的属性集合为实体型（Sno, Sname, Sage, Grade, SD, Cno, …）。记录型是记录的结构定义。

3.2.2 数据模型的三要素

数据模型的三要素是指数据结构、数据操作和数据的约束条件。

（1）数据结构。数据结构是所研究的对象类型的集合，是对系统静态特性的描述。

（2）数据操作。数据操作是指对数据库中各种对象（型）的实例（值）允许执行的操作的集合，包括操作及操作规则。如操作有检索、插入、删除、修改，操作规则有优先级别等。数据操作是对系统动态特性的描述。

（3）数据的约束条件。是一组完整性规则的集合。也就是说，对于具体的应用数据必须遵循特定的语义约束条件，以保证数据的正确、有效和相容。例如，某单位人事管理中，要求在职的男职工的年龄必须满足“18≤年龄≤60”，工程师的基本工资不能低于 1500 元，每个职工可担任一个工种，这些要求可以通过建立数据的约束条件来实现。

3.2.3 E-R 模型

在数据库设计中，常用实体联系模型（E-R 模型）来描述现实世界到信息世界的问题，它是软件设计中的一个重要工具。E-R 模型易于用户理解，是用户和数据库设计人员交流的语言。

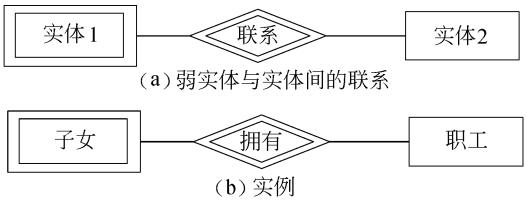
1．E-R 图

概念模型中常用的方法为 E-R 方法。该方法直接从现实世界中抽象出实体和实体间的联系，然后用直观的 E-R 图来表示数据模型。E-R 图中的主要构件如表 3-1 所示。

表 3-1 E-R 图中的主要构件

构　件	说　明
矩形	表示实体集
双边矩形	表示弱实体集
菱形	表示联系集
双边菱形	表示弱实体集对应的标识性联系
椭圆	表示属性
线段	将属性与相关的实体集连接，或将实体集与联系集相连
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

在 E-R 图中，弱实体集以双边框的矩形表示，而对应的标识性联系以双边框的菱形表示。例如，职工实体和子女关系实体之间的联系如图 3-4 所示。



（a）弱实体与实体间的联系

（b）实例

图 3-4　弱实体与实体间的联系

在 E-R 图中，实体集中作为主码的一部分属性以下划线标明。另外，在实体集与联系的线段上标识联系的类型。

在不引起误解的情况下，本章中实体集有时简称实体，联系集有时简称联系。

特别需要指出的是，E-R 模型强调的是语义，与现实世界的问题密切相关。例如，不同学校的教学管理方法可能不同，因此具有不同的语义，从而得到不同的 E-R 模型。E-R 模型的主要概念有实体、联系和属性。

2．实体

实体是现实世界中可以区别于其他对象的“事件”或“物体”。例如，企业中的每个员工都是一个实体。每个实体有一组特性（属性）来表示，其中的某一部分属性可以唯一标识实体，如职工实体中的职工号。实体集是具有相同属性的实体集合。例如，学校所有教师具有相同的属性，因此教师的集合可以定义为一个实体集；学生具有相同的属性，因此学生的集合可以定义为另一个实体集。

3．联系

实体的联系分为实体内部的联系和实体与实体之间的联系。实体内部的联系反映数据在同一记录内部各字段间的联系。这里主要讨论实体集之间的联系。

1）两个不同实体之间的联系

两个不同实体集之间存在一对一、一对多和多对多的联系类型。

- 一对一：指实体集 E1 中的一个实体最多只与实体集 E2 中的一个实体相联系，记为 1∶1。
- 一对多：表示实体集 E1 中的一个实体可与实体集 E2 中的多个实体相联系，记为 1∶*n*。
- 多对多：表示实体集 E1 中的多个实体可与实体集 E2 中的多个实体相联系，记为 *m*∶*n*。

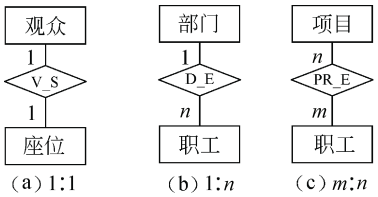
例如，图 3-4 表示两个不同实体集之间的联系。其中：

（1）电影院里一个座位只能坐一个观众，因此观众与座位之间是一个 1∶1 的联系，联系名为 V_S，用 E-R 图表示如图 3-5（a）所示。

（2）部门 DEPT 和职工 EMP 实体集，若一个职工只能属于一个部门，那么这两个实体集之

间应是一个 1∶*n* 的联系，联系名为 D_E，用 E-R 图表示如图 3-5（b）所示。

（3）工程项目 PROJ 和职工 EMP 实体集，若一个职工可以参加多个项目，一个项目可以由多个职工参加，那么这两个实体集之间应是一个 *m*∶*n* 的联系，联系名为 PR_E，用 E-R 图表示如图 3-5（c）所示。



（a）1∶1　（b）1∶*n*　（c）*m*∶*n*

图 3-5　两个不同实体集之间的联系

2）两个以上不同实体集之间的联系

两个以上不同实体集之间存在 1∶1∶1、1∶1∶*n*、1∶*m*∶*n* 和 *r*∶*m*∶*n* 的联系。例如，图 3-5 表示了三个不同实体集之间的联系。其中：

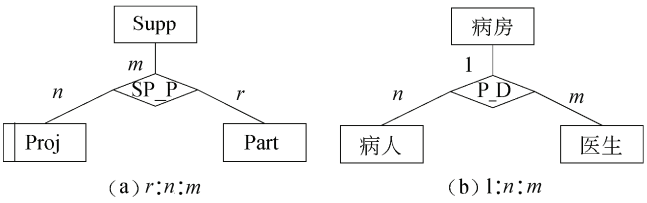
（1）图 3-6（a）表示供应商 Supp、项目 Proj 和零件 Part 之间的多对多（*r*∶*n*∶*m*）联系，联系名为 SP_P。表示供应商为多个项目供应多种零件，每个项目可用多个供应商供应的零件，每种零件可由不同的供应商供应的语义。

（2）图 3-6（b）表示病房、病人和医生之间的一对多（1∶*n*∶*m*）联系，联系名为 P_D。表示一个特护病房有多个病人和多个医生，一个医生只负责一个病房，一个病人只占用一个病房的语义。

注意，三个实体集之间的多对多联系和三个实体集两两之间的多对多联系的语义是不同的。例如，供应商和项目实体集之间的“合同”联系，表示供应商为哪几个工程签了合同。供应商与零件实体集之间的“库存”联系，表示供应商库存零件的数量。项目与零件两个实体集之间的“组成”联系，表示一个项目需要使用哪几种零件。

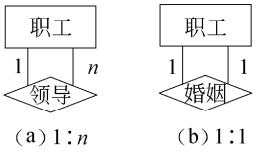
3）同一实体集内的二元联系

同一实体集内的各实体之间也存在 1∶1、1∶*n* 和 *m*∶*n* 的联系，如图 3-7 所示。从图 3-7 可知，职工实体集中的“领导”联系（即领导与被领导联系）是 1∶*n* 的。但是，职工实体集中的“婚姻”联系是 1∶1 的。



（a）*r*∶*n*∶*m*　（b）1∶*n*∶*m*

图 3-6　三个不同实体集之间的联系



（a）1∶*n*　（b）1∶1

图 3-7　同一实体集之间的 1∶*n* 和 1∶1 联系

4．属性

属性是实体某方面的特性。例如，职工实体集具有职工号、姓名、年龄、参加工作时间和通信地址等属性。每个属性都有其取值范围，如职工号为 0001～9999 的 4 位整型数，姓名为 10 位的字符串，年龄的取值范围为 18～60 等。在同一实体集中，每个实体的属性及值域是相同的，但可能取不同的值。E-R 模型中的属性有如下分类。

（1）简单属性和复合属性。简单属性是原子的、不可再分的，复合属性可以细分为更小的部分（即划分为别的属性）。有时用户希望访问整个属性，有时希望访问属性的某个成分，那么在模式设计时可采用复合属性。例如，职工实体集的通信地址可以进一步分为邮编、省、市、街道。若不特别声明，通常指的是简单属性。

（2）单值属性和多值属性。单值属性是指属性对于一个特定的实体都只有单独的一个值。例如，对于一个特定的职工，在系统中只对应一个职工号、职工姓名，这样的属性叫做单值属性。但是，在某些特定情况下，一个属性可能对应一组值。例如，职工可能有 0 个、1 个或多个亲属，那么职工亲属的姓名可能有多个，这样的属性称为多值属性。

（3）NULL 属性。当实体在某个属性上没有值或属性值未知时，使用 NULL 值，表示无意义或不知道。

（4）派生属性。派生属性可以从其他属性得到。例如，职工实体集中有“参加工作时间”和“工作年限”属性，那么“工作年限”的值可以由当前时间和参加工作时间得到。这里，“工作年限”就是一个派生属性。

5．实例分析

【例 3-1】 某学校有若干个系，每个系有若干名教师和学生；每个教师可以主讲若干门课程，并参加多个项目；每个学生可以同时选修多门课程。请设计该学校教学管理的 E-R 模型，要求给出每个实体、联系的属性。

解：根据需求分析结果，该校教学管理系统有 5 个实体：系、教师、学生、项目和课程。

（1）设计各实体属性如下：

系（系号，系名，主任名）

教师（教师号，教师名，职称）

学生（学号，姓名，年龄，性别）

项目（项目号，名称，负责人）

课程（课程号，课程名，学分）

（2）各实体之间的联系有：教师担任课程的 1∶*n*“任课”联系；教师参加项目的 *n*∶*m*“参加”联系；学生选修课程的 *n*∶*m*“选修”联系；系、学生及教师之间的 1∶*n*∶*m*“领导”联系。

其中，“参加”联系有一个排名属性，“选修”联系有一个成绩属性。

通过上述分析，设计该校的教学管理系统的 E-R 模型如图 3-8 所示。

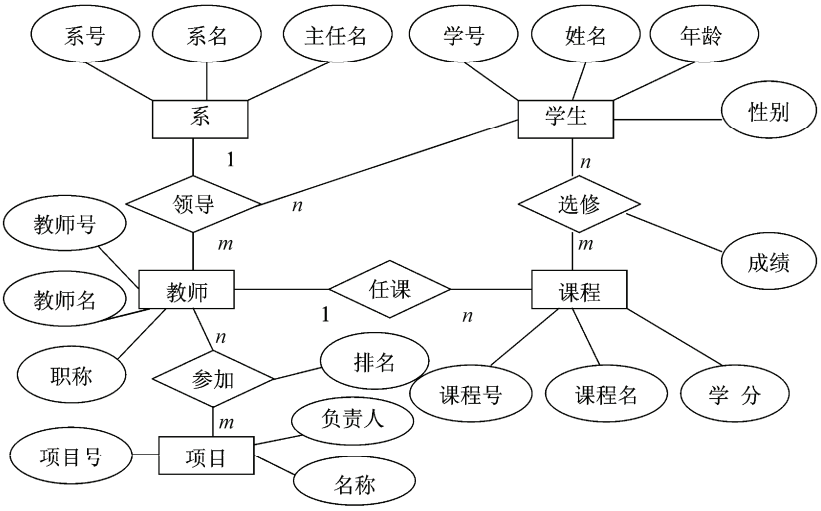


图 3-8　学校教学管理系统的 E-R 模型

3.2.4　基本的数据模型

1．层次模型（Hierarchical Model）

层次模型采用树型结构表示数据与数据间的联系。在层次模型中，每个结点表示一个记录类型（实体），记录之间的联系用结点之间的连线表示，并且根结点以外的其他结点有且仅有一个双亲结点。上一层和下一层类型的联系是 1∶*n* 联系（包括 1∶1 联系）。

【例 3-2】 某商场的部门、员工和商品三个实体的层次模型如图 3-9（a）所示。其中，每个部门有若干个员工，每个部门负责销售的商品有若干种，即该模型还表示部门到员工之间的一对多（1∶*n*）联系，部门到商品之间的一对多（1∶*n*）联系。

图 3-9（a）给出的只是商场层次模型的“型”，而不是“值”。在数据库中，“型”就是数据库模式，而“值”就是数据库实例。模式是数据库的逻辑设计，而数据库实例是给定时刻数据库中数据的一个快照。图 3-9（b）表示商场销售部的一个实例。该实例表示在某一时刻销售部是由李军负责，下属 4 个员工，负责销售的商品有 5 种。

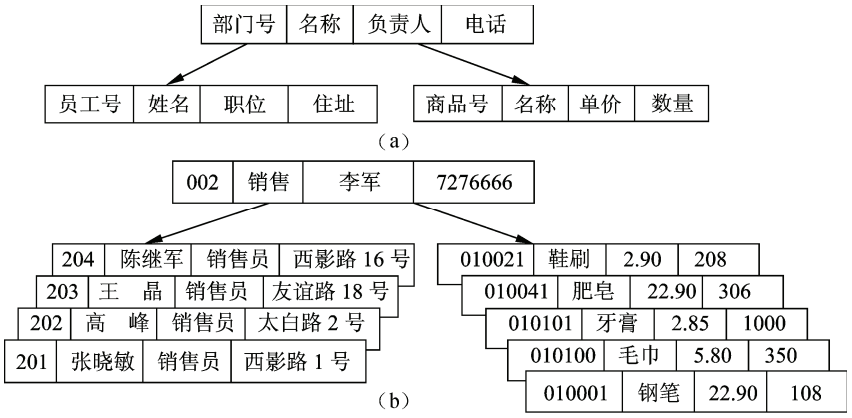


图 3-9　层次模型

层次模型的特点是记录之间的联系通过指针实现，比较简单，查询效率高。

层次模型的缺点是只能表示 1∶*n* 的联系，尽管有许多辅助手段实现 *m*∶*n* 的联系，但较复杂且不易掌握；由于层次顺序严格且复杂，对插入和删除操作的限制比较多，导致应用程序编制比较复杂。

IBM 公司在 1968 年推出的 IMS 系统（信息管理系统）是典型的层次模型系统，20 世纪 70 年代在商业领域得到了广泛的应用。

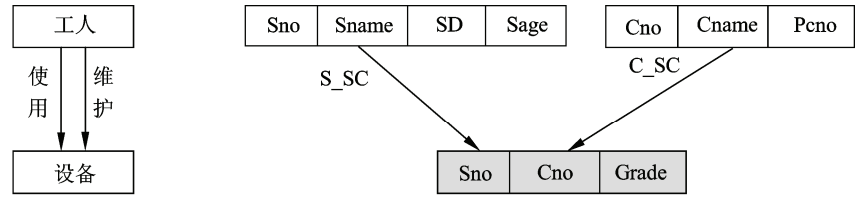
2．网状模型（Network Model）

采用网络结构表示实体类型及实体间联系的数据模型称为网状模型。在网状模型中，允许一个以上的结点无双亲，每个结点可以有多于一个的双亲。网状模型是一个比层次模型更普遍的数据结构，层次模型是网状模型的一个特例。网状模型可以直接地描述现实世界。

网状模型中的每个结点表示一个记录类型（实体），每个记录类型可以包含若干个字段（实体的属性），结点间的连线表示记录类型之间一对多的联系。与层次模型的主要区别如下：网状模型中子女结点与双亲结点的联系不唯一，因此需要为每个联系命名；网状模型允许复合链，即两个结点之间有两种以上的联系，如图 3-10（a）所示的工人与设备之间的联系；网状模型不能表示记录之间的多对多联系，需要引入联结记录来表示多对多的联系。

【例 3-3】 学生、课程以及他们之间的多对多联系不能直接用网状模型表示。因为一个学生可以选若干门课，而一门课可以被多个学生选。为此，引入选课联结记录，如图 3-10（b）所示。这样，学生与选课之间的S-SC是一对多的联系，课程与选课之间的C-SC也是一对多的

联系。其中，Sno、Sname、SD、Sage、Cno、Cname、Pcno 和 Grade 分别表示学号、姓名、系、年龄、课程号、课程名、先修课程号和成绩。



（a）工人与设备　　（b）学生与课程

图 3-10　网状模型举例

网状模型的主要优点是能更为直接地描述现实世界，具有良好的性能，存取效率高。

网状模型的主要缺点是结构复杂。例如，当应用环境不断扩大时，数据库结构就变得很复杂，不利于最终用户掌握，编制应用程序难度也比较大。

3．关系模型（Relational Model）

关系模型是目前最常用的数据模型之一。关系数据库系统采用关系模型作为数据的组织方式，在关系模型中用表格结构表达实体集，以及实体集之间的联系，其最大特点是描述的一致性。关系模型是由若干个关系模式组成的集合。关系模式可记为 R（A_1，A_2，A_3，…，A_n），其中，R 表示关系名，A_1、A_2、A_3、…、A_n 表示属性名。

一个关系模式相当于一个记录型，对应于程序设计语言中类型定义的概念。关系是一个实例，也是一张表，对应于程序设计语言中变量的概念。变量的值随程序运行可能发生变化，类似地，当关系被更新时，关系实例的内容也随时间发生了变化。

在关系模型中用主码导航数据，表格简单、直观易懂，用户只需要简单的查询语句就可以对数据库进行操作，即用户只需指出“干什么”或“找什么”，而不必详细说明“怎么干”或“怎么找”，无须涉及存储结构和访问技术等细节。

【例 3-4】 教学管理数据库的 4 个关系模式如下：

S (Sno,Sname,SD,Sage,Sex)　　；学生 S 关系模式，属性为学号、姓名、系、年龄和性别

T (Tno,Tname,Age,Sex)　　；教师 T 关系模式，属性为教师号、姓名、年龄和性别

C (Cno,Cname,Pcno)　　；课程 C 关系模式，属性为课程号、课程名和先修课程号

SC (Sno,Cno,Grade)　　；学生选课 SC 关系模式，属性为学号、课程号和成绩

关系模式中带下划线的属性是主码属性。如图 3-11 所示是教学模型的一个具体实例。

S 学生关系

Sno	Sname	SD	Age	Sex
01001	贾皓昕	IS	20	男
01002	姚勇	IS	20	男
03001	李晓红	CS	19	女

T 教师关系

Tno	Tname	Age	Sex
001	方铭	34	女
002	章雨敬	58	男
003	王平	48	女

SC 选课

Sno	Cno	Grade
01001	C001	90
01001	C002	91
01002	C001	95
01002	C003	89
03001	C001	91

C 课程关系

Cno	Cname	Pcno
C001	MS	
C002	IC	
C003	C++	C002
C004	OS	C002
C005	DBMS	C004

图 3-11　关系模型的实例

3.3　DBMS 的功能和特征

3.3.1　DBMS 的功能

DBMS 主要实现共享数据有效地组织、管理和存取，因此 DBMS 应具有如下几个方面的功能。

1．数据定义

DBMS 提供数据定义语言（Data Definition Language，DDL），用户可以对数据库的结构进行描述，包括外模式、模式和内模式的定义；数据库的完整性定义；安全保密定义，如口令、级别和存取权限等。这些定义存储在数据字典中，是 DBMS 运行的基本依据。

2．数据库操作

DBMS 向用户提供数据操纵语言（Data Manipulation Language，DML），实现对数据库中数据的基本操作，如检索、插入、修改和删除。DML 分为两类：宿主型和自含型。所谓宿主型，是指将 DML 语句嵌入某种主语言（如 C、COBOL 等）中使用；自含型是指可以单独使用 DML 语句，供用户交互使用。

3．数据库运行管理

数据库在运行期间多用户环境下的并发控制、安全性检查和存取控制、完整性检查和执行、

运行日志的组织管理、事务管理和自动恢复等是 DBMS 的重要组成部分。这些功能可以保证数据库系统的正常运行。

4．数据组织、存储和管理

DBMS 分类组织、存储和管理各种数据，包括数据字典、用户数据和存取路径等。要确定以何种文件结构和存取方式在存储级上组织这些数据，以提高存取效率。实现数据间的联系、数据组织和存储的基本目标是提高存储空间的利用率。

5．数据库的建立和维护

数据库的建立和维护包括数据库的初始建立、数据的转换、数据库的转储和恢复、数据库的重组和重构、性能监测和分析等。

6．其他功能

如 DBMS 与网络中其他软件系统的通信功能，一个 DBMS 与另一个 DBMS 或文件系统的数据转换功能等。

3.3.2 DBMS 的特征与分类

1．DBMS 的特征

通过 DBMS 管理数据具有如下特点。

（1）数据结构化且统一管理。数据库中的数据由 DBMS 统一管理。由于数据库系统采用复杂的数据模型表示数据结构，数据模型不仅描述数据本身的特点，还描述数据之间的联系。数据不再面向某个应用，而是面向整个应用系统。数据易维护、易扩展，数据冗余明显减少，真正实现了数据的共享。

（2）有较高的数据独立性。数据的独立性是指数据与程序独立，将数据的定义从程序中分离出去，由 DBMS 负责数据的存储，应用程序关心的只是数据的逻辑结构，无须了解数据在磁盘上的数据库中的存储形式，从而简化了应用程序，大大减少了应用程序编制的工作量。数据的独立性包括数据的物理独立性和数据的逻辑独立性。

（3）数据控制功能。DBMS 提供了数据控制功能，以适应共享数据的环境。数据控制功能包括对数据库中数据的安全性、完整性、并发和恢复的控制。

① 数据库的安全性保护。数据库的安全性（security）是指保护数据库以防止不合法的使用所造成的数据泄露、更改或破坏。这样，用户只能按规定对数据进行处理，例如，划分了不

同的权限，有的用户只能有读数据的权限，有的用户有修改数据的权限，用户只能在规定的权限范围内操纵数据库。

② 数据的完整性。数据库的完整性是指数据库的正确性和相容性，是防止合法用户使用数据库时向数据库加入不符合语义的数据。保证数据库中数据是正确的，避免非法的更新。

③ 并发控制。在多用户共享的系统中，许多用户可能同时对同一数据进行操作。DBMS的并发控制子系统负责协调并发事务的执行，保证数据库的完整性不受破坏，避免用户得到不正确的数据。

④ 故障恢复。数据库中的4类故障是事务内部故障、系统故障、介质故障及计算机病毒。故障恢复主要是指恢复数据库本身，即在故障引起数据库当前状态不一致后，将数据库恢复到某个正确状态或一致状态。恢复的原理非常简单，就是要建立冗余（redundancy）数据。换句话说，确定数据库是否可恢复的方法就是其包含的每一条信息是否都可以利用冗余地存储在别处的信息重构。冗余是物理级的，通常认为逻辑级是没有冗余的。

2．DBMS分类

DBMS通常可分为如下3类。

（1）关系数据库系统（Relation DataBase System，RDBS）。RDBS是支持关系模型的数据库系统。在关系模型中，实体以及实体间的联系都是用关系来表示。在一个给定的现实世界领域中，相应于所有实体及实体之间联系的关系的集合构成一个关系数据库，也有型和值之分。关系数据库的型也称为关系数据库模式，是对关系数据库的描述，是关系模式的集合；关系数据库的值也称为关系数据库，是关系的集合。关系数据库模式与关系数据库通常统称为关系数据库。在微型计算机方式下常见的FoxPro和Access等DBMS，严格地讲不能算是真正的关系型数据库，对许多关系类型的概念并不支持，但它却因为简单实用、价格低廉，目前拥有很大的用户市场。

（2）面向对象的数据库系统（Object-Oriented DataBase System，OODBS）。OODBS是支持以对象形式对数据建模的数据库管理系统，包括对对象的类、类属性的继承，对子类的支持。面向对象数据库系统主要有两个特点：面向对象数据模型能完整描述现实世界的数据结构，能表达数据间嵌套、递归的联系；具有面向对象技术的封装性和继承性，提高了软件的可重用性。

（3）对象关系数据库系统（Object-Oriented Relation Database System，ORDBS）。ORDBS是在传统的关系数据模型基础上，提供元组、数组、集合一类更为丰富的数据类型以及处理新的数据类型操作的能力，这样形成的数据模型被称为“对象关系数据模型”。基于对象关系数

据模型的 DBS 称为对象关系数据库系统。

3.4 数据库模式

数据库系统是数据密集型应用的核心，其体系结构受数据库运行所在的计算机系统的影响很大。从数据库管理系统的角度看，数据库系统体系结构一般采用三级模式结构。

3.4.1 模式

实际上，数据库的产品很多，它们支持不同的数据模型，使用不同的数据库语言，建立在不同的操作系统上。数据的存储结构也各不相同，但体系结构基本上都具有相同的特征，采用“三级模式和两级映像”。如图 3-12 所示。

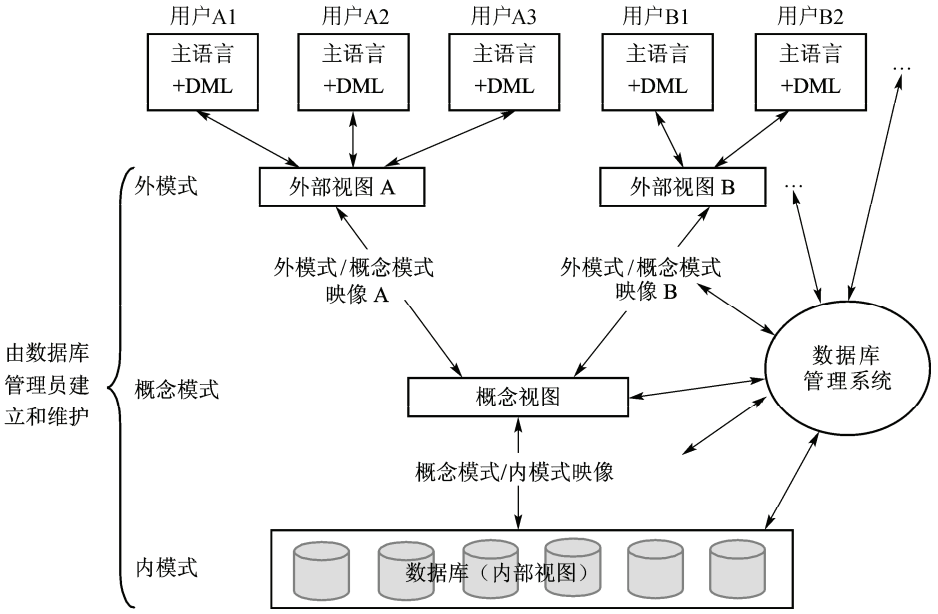


图 3-12 数据库系统体系结构

数据库系统采用三级模式结构，这是数据库管理系统内部的系统结构。

数据库系统设计员可在视图层、逻辑层和物理层对数据抽象，通过外模式、概念模式和内模式来描述不同层次上的数据特性。

1．概念模式

概念模式也称模式，是数据库中全部数据的逻辑结构和特征的描述，它由若干个概念记录类型组成，只涉及行的描述，不涉及具体的值。概念模式的一个具体值称为模式的一个实例，同一个模式可以有很多实例。

概念模式反映的是数据库的结构及其联系，所以是相对稳定的；而实例反映的是数据库某一时刻的状态，所以是相对变动的。

需要说明的是，概念模式不仅要描述概念记录类型，还要描述记录间的联系、操作、数据的完整性和安全性等要求。但是，概念模式不涉及存储结构、访问技术等细节。只有这样，概念模式才算做到了“物理数据独立性”。

描述概念模式的数据定义语言称为“模式 DDL（Schema Data Definition Language）”。

2．外模式

外模式也称用户模式或子模式，是用户与数据库系统的接口，是用户用到的那部分数据的描述。它由若干个外部记录类型组成。用户使用数据操纵语言对数据库进行操作，实际上是对外模式的外部记录进行操作。

描述外模式的数据定义语言称为“外模式 DDL”。有了外模式后，程序员不必关心概念模式，只与外模式发生联系，按外模式的结构存储和操纵数据。

3．内模式

内模式也称存储模式，是数据物理结构和存储方式的描述，是数据在数据库内部的表示方式。需要定义所有的内部记录类型、索引和文件的组织方式，以及数据控制方面的细节。

例如，记录的存储方式是顺序存储、B 树结构存储还是 Hash 方法存储；索引按照什么方式组织；数据是否压缩存储，是否加密；数据的存储记录结构有何规定。

需要说明的是，内部记录并不涉及物理记录，也不涉及设备的约束。比内模式更接近于物理存储和访问的那些软件机制是操作系统的一部分（即文件系统）。例如，从磁盘上读、写数据。

描述内模式的数据定义语言称为“内模式 DDL”。

总之，数据按外模式的描述提供给用户；按内模式的描述存储在磁盘上；而概念模式提供了连接这两级模式的相对稳定的中间层，并使得两级中任意一级的改变都不受另一级的牵制。

3.4.2 三级模式两级映像

数据库系统在三级模式之间提供了两级映像：模式/内模式的映像、外模式/模式的映像。这两级映射保证了数据库中的数据具有较高的物理独立性和逻辑独立性。

- 模式/内模式的映像：实现概念模式到内模式之间的相互转换。
- 外模式/模式的映像：实现外模式到概念模式之间的相互转换。

数据的独立性是指数据与程序独立，将数据的定义从程序中分离出去，由 DBMS 负责数据的存储，从而简化应用程序，大大减少应用程序编制的工作量。数据的独立性是由 DBMS 的二级映像功能来保证的。数据的独立性包括数据的物理独立性和数据的逻辑独立性。

数据的物理独立性是指当数据库的内模式发生改变时，数据的逻辑结构不变。由于应用程序处理的只是数据的逻辑结构，这样物理独立性可以保证，当数据的物理结构改变了，应用程序不用改变。但是，为了保证应用程序能够正确执行，需要修改概念模式/内模式之间的映像。

数据的逻辑独立性是指用户的应用程序与数据库的逻辑结构是相互独立的。数据的逻辑结构发生变化后，用户程序也可以不修改。但是，为了保证应用程序能够正确执行，需要修改外模式/概念模式之间的映像。

3.5 关系数据库与关系运算

3.5.1 关系数据库的基本概念

1．属性和域

在现实世界中，一个事物常常取若干特征来描述，这些特征称为属性（attribute）。例如，用学号、姓名、性别、系别、年龄和籍贯等属性来描述学生。每个属性的取值范围对应一个值的集合，称为该属性的域（domain）。例如，学号的域是 6 位整数；姓名的域是 20 个字符；性别的域为男，女。

一般在关系数据模型中，限制所有的域都是原子数据（atomic data）。例如，整数、字符串是原子数据，而集合、记录、数组是非原子数据。关系数据模型的这种限制称为第一范式（First Normal Form，1NF）条件。

2．笛卡儿积与关系

【定义 3.1】 设 $D_1,D_2,\cdots,D_{\mathrm{i}},\cdots,D_{\mathrm{n}}$ 为任意集合，定义 $D_1,D_2,\cdots,D_{\mathrm{i}},\cdots,D_{\mathrm{n}}$ 的笛卡儿积为

$$D_1\times D_2\times\cdots\times D_i\times\cdots\times D_n=\{(d_1,d_2,\cdots,d_i,\cdots,d_n)\mid d_i\in D_i,i=1,2,3,\cdots,n\}$$

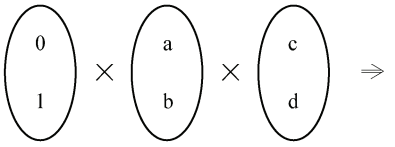
其中，每一个元素$(d_1,d_2,\cdots,d_i,\cdots,d_n)$叫做一个 n 元组（n-tuple，属性的个数），元组的每一个值 d_i 叫做元组的一个分量，若 $D_i\left(i=1,2,3,\cdots,n\right)$ 为有限集，其基数（元组的个数）为 $m_i\left(i=1,2,3,\cdots,n\right)$，则 $D_1\times D_2\times\cdots\times D_i\times\cdots\times D_n$ 的基数 M 为 $M=\prod_{i=1}^{n}m_i$，笛卡儿积可以用二维表来表示。

【例 3-5】 若 $D_1=\{0,1\}$、$D_2=\{\mathrm{a},\mathrm{b}\}$、$D_3=\{\mathrm{c},\mathrm{d}\}$，求 $D_1\times D_2\times D_3$。

解：根据定义，笛卡儿积中的每一个元素应该是一个三元组，每个分量来自不同的域，因此结果为

$$D_1\times D_2\times D_3=\{(0,\mathrm{a},\mathrm{c}),(0,\mathrm{a},\mathrm{d}),(0,\mathrm{b},\mathrm{c}),(0,\mathrm{b},\mathrm{d}),(1,\mathrm{a},\mathrm{c}),(1,\mathrm{a},\mathrm{d}),(1,\mathrm{b},\mathrm{c}),(1,\mathrm{b},\mathrm{d})\}$$

用二维表表示如图 3-13 所示。



D_1	D_2	D_3
0	a	c
0	a	d
0	b	c
0	b	d
1	a	c
1	a	d
1	b	c
1	b	d

图 3-13　$D_1\times D_2\times D_3$ 笛卡儿积的二维表表示

【定义 3.2】 $D_1\times D_2\times D_3\times\cdots\times D_n$ 的子集叫做在域 $D_1,D_2,D_3,\cdots,D_n$ 上的关系，记为 R（$D_1,D_2,D_3,\cdots,D_n$），称关系 R 为 n 元关系。

从定义 3.2 可知，一个关系也可以用二维表来表示。关系中属性的个数称为“元数”，元组的个数称为“基数”。

关系模型中的术语说明如图 3-14 所示。其中，学生关系模式可表示为：学生(Sno,Sname,SD,Sex)。该学生关系的主码为 Sno，属性分别为 Sno、Sname、SD 和 Sex，属性 Sex 的域为男、女。该学生关系的元数为 4，基数为 6。

3．主要术语

关系的主要术语（基本概念）解释如下。

- 目或度（Degree）：属性个数 n 是关系的目或度。

- 候选码（Candidate Key）：若关系中某一属性（或属性组）的值能唯一地标识一个元组，则称该属性（属性组）为候选码。
- 主码（Primary Key）：若一个关系有多个候选码，则选定其中一个为主码。
- 主属性（Key attribute）：包含在任何候选码中的属性称为主属性。
- 非码属性（Non-Key attribute）：不包含在任何候选码中的属性称为非码属性。

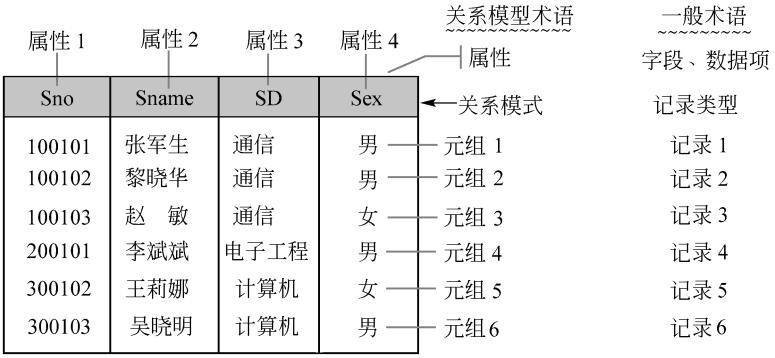


图 3-14　学生关系中相关术语的对应说明

- 外码（Foreign key）：如果关系模式 *R* 中的属性（属性组）不是该关系的码，但它是其他关系的码，那么该属性（属性组）对关系模式 *R* 而言是外码。例如，客户与贷款之间的借贷联系 c-l（c-id,loan-no），属性 c-id 是客户关系中的码，所以 c-id 是外码；属性 loan-no 是贷款关系中的码，所以 loan-no 也是外码。
- 全码（All-key）：关系模型的所有属性组是这个关系模式的候选码，称为全码。

例如，关系模式 *R*（*T*，*C*，*S*），属性 *T* 表示教师，属性 *C* 表示课程，属性 *S* 表示学生。假设一个教师可以讲授多门课程，某门课程可以由多个教师讲授，学生可以听不同教师讲授的不同课程，那么，要想区分关系中的每一个元组，这个关系模式 *R* 的码应为全属性 *T*、*C* 和 *S*，即 All-key。

4．关系的性质

一个基本关系具有以下 5 条性质。

（1）分量必须取原子值，每一个分量必须是不可再分的数据项。

（2）列是同质的，每一列中的分量必须是同一类型的数据，来自同一个域。

（3）属性不能重名，每列为一个属性，不同的列可来自同一个域。例如，课程关系 C（Cno, Cname, Pcno），其中 Cno 为课程号，Pcno 为先修课程号，它们都来自同一个域。

（4）行列的顺序无关。因为关系是一个集合，所以不考虑元组间的顺序。

（5）任何两个元组不能完全相同，这是由主码约束来保证的。但有些数据库若用户没有定义完整性约束条件，允许有两行以上相同的元组。

5．关系的三种类型

（1）基本关系（通常又称为基本表或基表）。是实际存在的表，它是实际存储数据的逻辑表示。

（2）查询表。查询结果对应的表。

（3）视图表。是由基本表或其他视图表导出的表。由于它本身不独立存储在数据库中，数据库中只存放它的定义，所以常称为虚表。

3.5.2 关系数据库模式

在数据库中要区分型和值。关系数据库中的型也称为关系数据库模式，是关系数据库结构的描述，包括若干域的定义以及在这些域上定义的若干关系模式。关系数据库的值是这些关系模式在某一时刻对应的关系的集合，通常称为关系数据库。

【定义 3.3】 关系的描述称为关系模式（Relation Schema）。可以形式化地表示为

$$R(U，D，\text{dom}，F)$$

其中，R 表示关系名；U 是组成该关系的属性名集合；D 是属性的域；dom 是属性向域的映像集合；F 为属性间数据的依赖关系集合。

通常将关系模式简记为

$$R(U)\text{或}R\left(A_1,A_2,A_3,\cdots,A_n\right)$$

其中，R 为关系名，$A_1,A_2,A_3,\cdots,A_n$ 为属性名或域名，属性常常由其类型和长度说明。通常在关系模式主属性上加下划线表示该属性为主码属性。

【例 3-6】 学生关系 S 有学号 Sno、学生姓名 Same、系名 SD、年龄 SA 属性；课程关系 C 有课程号 Cno、课程名 Cname、先修课程号 PCno 属性；学生选课关系 SC 有学号 Sno、课程号 Cno、成绩 Grade 属性。关系模式可定义如下。

（1）学生关系模式 S（Sno，Sname，SD，SA）。

（2）课程关系模式 C（Cno，Cname，PCno） Dom（PCno）= Cno。先修课程号 PCno 来自 Cno 域。在关系模式中，各列属性必须取不同的名字，因此不能将 PCno 直接改为 Cno。

（3）学生选课关系模式 SC（Sno，Cno，Grade）。SC 关系中的 Sno、Cno 分别为外码，因为它们分别是 S、C 关系中的主码。

3.5.3 完整性约束

完整性规则保证授权用户对数据库进行修改不会破坏数据的一致性。关系模型的完整性规则是对关系的某种约束条件，分为实体完整性、参照完整性（也称引用完整性）和用户定义完整性 3 类。

（1）实体完整性（Entity Integrity）。规定基本关系 R 的主属性 A 不能取空值。

（2）参照完整性（Referential Integrity）。存在于两个关系之间，也称引用完整性，用于描述关系模型中实体和实体间的联系。

例如，员工和部门关系模式表示如下，其中员工号和部门号是主码。

员工（员工号，姓名，性别，参加工作时间，部门号）

部门（部门号，名称，电话，负责人）

这两个关系存在着属性的引用，即员工关系中的“部门号”必须是部门关系中某部门的编号。也就是说，员工关系中的“部门号”属性的取值要参照部门关系的“部门号”属性的取值。

参照完整性规定：对于关系 R 和 S，若 F 是关系 R 的外码，它与关系 S 的主码 K_s 相对应（基本关系 R 和 S 不一定是不同的关系），则 R 中每个元组在 F 上的取值必须满足或者取空值（F 的每个属性值均为空值）；或者等于 S 中某个元组的主码值。

（3）用户定义的完整性（User Defined Integrity）。就是针对某一具体的关系数据库的约束条件，反映某一具体应用所涉及的数据必须满足的语义要求，由应用环境决定。例如，规定职工的年龄必须大于等于 18、小于等于 60。

3.5.4 关系代数运算

关系操作的特点是操作对象和操作结果都是集合。关系代数运算符有 4 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，如表 3-2 所示。

表 3-2 关系代数运算符

运算符		含义	运算符		含义
集合运算符	∪ — ∩ ×	并 差 交 笛卡儿积	比较运算符	> ≥ < ≤ = ≠	大于 大于等于 小于 小于等于 等于 不等于
专门的关系运	σ π ⋈	选择 投影 连接	逻辑运算符	¬ ∧ ∨	非 与 或

算符	÷	除			

传统的集合运算是从关系的水平方向进行的，包括并、交、差及广义笛卡儿积。专门的关系运算既可以从关系的水平方向进行运算，也可以从关系的垂直方向进行运算，包括选择、投影、连接以及除法。

并、差、笛卡儿积、投影和选择是5种基本的运算，其他运算可以由基本运算导出。

1．并（union）

关系 R 与 S 具有相同的关系模式，即 R 与 S 的结构相同。关系 R 与 S 的并由属于 R 或属于 S 的元组构成，记作 $R \cup S$，其形式定义为 $R \cup S = \{t \mid t \in R \vee t \in S\}$，其中 t 为元组变量。

2．差（difference）

关系 R 与 S 具有相同的关系模式，关系 R 与 S 的差由属于 R 但不属于 S 的元组构成，记作 $R–S$，其形式定义为 $R - S = \{t \mid t \in R \wedge t \notin S\}$。

3．广义笛卡儿积（extended cartesian product）

两个元数分别为 n 目和 m 目的关系 R 和 S 的广义笛卡儿积是一个（$n+m$）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，记作 $R \times S$。若 R 和 S 中有相同的属性名，可在属性名前加关系名作为限定，以示区别。若 R 有 K_1 个元组，S 有 K_2 个元组，则 R 和 S 的广义笛卡儿积有 $K_1 \times K_2$ 个元组。

4．投影（projection）

投影运算从关系的垂直方向进行运算，在关系 R 中选择出若干属性列 A 组成新的关系，记作 $\pi_A(R)$，其形式定义为 $\pi_A(R) = \{t[A] \mid t \in R\}$。

5．选择（selection）

选择运算从关系的水平方向进行运算，是从关系 R 中选择满足给定条件的若干个元组，记作 $\sigma_F(R)$，其形式定义为 $\sigma_F(R) = \{t \mid t \in R \wedge F(t) = \text{True}\}$。其中，$F$ 中的运算对象是属性名（或列的序号），运算符为比较运算符（$<, \leqslant, >, \geqslant, =, \neq$）和逻辑运算符（$\wedge, \vee, \neg$）。例如，$\sigma_{1 \geqslant 6}(R)$ 表示选取 R 关系中第1个属性值大于等于第6个属性值的元组；$\sigma_{1>'6'}(R)$ 表示选取 R 关系中第1个属性值大于'6'的元组。

【例3-7】 设有关系 R、S，如图3-15所示，求 $R \cup S$、$R - S$、$R \times S$、$\pi_{A,C}(R)$、$\sigma_{A>B}(R)$、$\sigma_{3<4}(R \times S)$。

A	B	C
a	b	c
b	a	d
c	d	e
d	f	g

（a）关系 R

A	B	C
b	a	d
d	f	g
f	h	k

（b）关系 S

图 3-15　关系 R、S

解：$R \cup S$、$R-S$、$R \times S$、$\pi_{A,C}(R)$、$\sigma_{A>B}(R)$ 和 $\sigma_{3<4}(R \times S)$ 的结果如图 3-16 所示。

其中，$R \times S$ 后生成的关系属性名有重复，按照关系“属性不能重名”的性质，通常采用“关系名.属性名”的格式。对于 $\sigma_{3<4}(R \times S)$ 的含义，是 $R \times S$ 后“选取第 3 个属性值小于第 4 个属性值”的元组。由于 $R \times S$ 的第 3 个属性为 R.C，第 4 个属性是 S.A，因此 $\sigma_{3<4}(R \times S)$ 的含义也是 $R \times S$ 后“选取 R.C 值小于 S.A 值”的元组。

$R \cup S$

A	B	C
a	b	c
b	a	d
c	d	e
d	f	g
f	h	k

$R-S$

A	B	C
a	b	c
c	d	e

$\pi_{A,C}(R)$

A	C
a	c
b	d
c	e
d	g

$\sigma_{A>B}(R)$

A	B	C
b	a	d

$R \times S$

R.A	R.B	R.C	S.A	S.B	S.C
a	b	c	b	a	d
a	b	c	d	f	g
a	b	c	f	h	k
b	a	d	b	a	d
b	a	d	d	f	g
b	a	d	f	h	k
c	d	e	b	a	d
c	d	e	d	f	g
c	d	e	f	h	k
d	f	g	b	a	d
d	f	g	d	f	g
d	f	g	f	h	k

$\sigma_{3<4}(R \times S)$

R.A	R.B	R.C	S.A	S.B	S.C
a	b	c	d	f	g
a	b	c	f	h	k
b	a	d	f	h	k
c	d	e	f	h	k

图 3-16　运算结果

6．交（intersection）

关系 R 与 S 具有相同的关系模式，关系 R 与 S 的交由属于 R 同时又属于 S 的元组构成，关系 R 与 S 的交记作 $R \cap S$，其形式定义为 $R \cap S = \{t \mid t \in R \wedge t \in S\}$。显然，$R \cap S = R - (R - S)$，或者 $R \cap S = S - (S - R)$。

7．连接（join）

连接运算是从两个关系 R 和 S 的笛卡儿积中选取满足条件的元组。因此，可以认为笛卡儿积是无条件连接，其他的连接操作认为是有条件连接。连接运算可分为 θ 连接、等值连接及自然连接。

（1）θ 连接。从 R 与 S 的笛卡儿积中选取属性间满足一定条件的元组。记作 $R \underset{X\theta Y}{\rhd\lhd} S$，$'X\theta Y'$ 为连接的条件，θ 是比较运算符，X 和 Y 分别为 R 和 S 上度数相等且可比的属性组。

（2）等值连接。当 θ 为“=”时，称之为等值连接，记为 $R \underset{X=Y}{\rhd\lhd} S$。

（3）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组，并且在结果集中将重复属性列去掉，记作 $R \rhd\lhd S$。

【例 3-8】 设有关系 R、S，如图 3-17 所示，求 $R \rhd\lhd S$。

解：本题 R 与 S 关系中相同的属性组为 A 和 C，因此，结果集中的属性列应为 A、B、C、D，如图 3-18 所示。

A	B	C
a	b	c
b	a	d
c	d	e
d	f	g

（a）关系 R

A	C	D
a	c	d
d	f	g
b	d	g

（b）关系 S

图 3-17　关系 R、S

A	B	C	D
a	b	c	d
b	a	d	g

图 3-18　$R \rhd\lhd S$

【例 3-9】 对于图 3-15 所示的关系 R、S，求 $R \underset{R.A<S.B}{\rhd\lhd} S$。

解：本题连接的条件为 R.A<S.B，意为将 R 关系中属性 A 的值小于 S 关系中属性 B 的值的元组取出来作为结果集的元组。结果集为 $R \times S$ 后选出满足条件的元组，并且结果集的属性为 R.A、R.B、R.C、S.A、S.B、S.C，如图 3-19 所示。

R.A	*R*.B	*R*.C	*S*.A	*S*.B	*S*.C
a	b	c	d	f	g
a	b	c	f	h	k
b	a	d	d	f	g
b	a	d	f	h	k
c	d	e	d	f	g
c	d	e	f	h	k
d	f	g	d	f	g
d	f	g	f	h	k

图 3-19 $R \underset{R.A<S.B}{\triangleright\triangleleft} S$

3.6 关系数据库 SQL 语言简介

SQL（Structured Query Language）是一种通用的、功能强大的标准查询语言，已在众多商用 DBMS（如 DB2、Oracle、Ingres、Sysbase 和 SQL Server 等）实现。SQL 不仅包含数据查询功能，还包括插入、删除、更新和数据定义功能。SQL 用户可以是应用程序，也可以是终端用户。

3.6.1 SQL 概述

目前，SQL 主要有三个标准：ANSI（美国国家标准机构）SQL；对 ANSI SQL 进行修改后在 1992 年采用的标准称之为 SQL-92（或 SQL2）；SQL-99 标准（或 SQL3 标准）。SQL-99 从 SQL-92 扩充而来，增加了对象关系特征和许多新的功能。

1．SQL 的特点

（1）综合统一。SQL 集数据定义、数据操纵和数据控制功能于一体，语言风格统一，可独立完成数据库生命周期的所有活动。

（2）高度非过程化。非关系数据模型的数据操纵语言是面向过程的，若要完成某项请求时，必须指定存储路径；而 SQL 语言是高度非过程化语言，进行数据操作时，只要指出“做什么”，无须指出“怎么做”，存储路径对用户来说是透明的，提高了数据的独立性。

（3）面向集合的操作方式。非关系数据模型采用的是面向记录的操作方式，操作对象是一条记录。而 SQL 语言采用面向集合的操作方式，其操作对象、查找结果可以是元组的集合。

（4）两种使用方式。用户可以在终端键盘上输入 SQL 命令，对数据库进行操作，称为自含式语言；或者将 SQL 语言嵌入到高级语言程序中使用（称为嵌入式语言）。

SQL 语言功能极强，只用了 9 个动词表示其核心功能，如下所示。

- 数据查询：SELECT。
- 数据定义：CREATE、DROP、ALTER。
- 数据操纵：INSERT、UPDATE、DELETE。
- 数据控制：GRANT、REVOKE。

2．SQL 支持三级模式结构

SQL 支持关系数据库的三级模式结构，其中，视图对应外模式、基本表对应模式、存储文件对应内模式，如图 3-20 所示。

3．SQL 的基本组成

SQL 由以下几个部分组成：数据定义语言、交互式数据操纵语言、事务控制、嵌入式 SQL 和动态 SQL、完整性控制和权限管理。下面主要介绍基本的 DDL、DML、嵌入式 SQL。

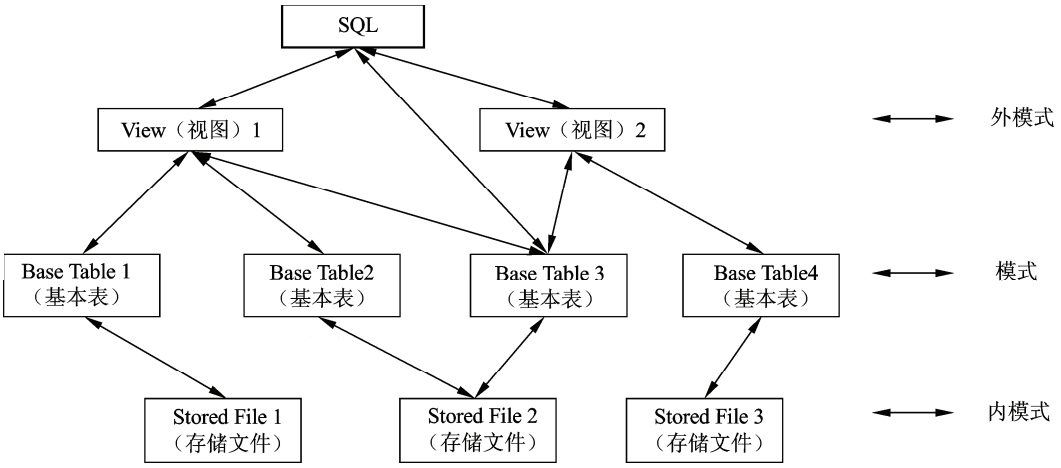


图 3-20　关系数据库的三级模式结构

3.6.2　SQL 数据定义

SQL DDL 提供定义关系模式和视图、删除关系和视图、修改关系模式的命令。

1．创建表（CREATE TABLE）

语句格式：CREATE TABLE <表名>(<列名><数据类型>[列级完整性约束条件]

[,<列名><数据类型>[列级完整性约束条件]]…
[,<表级完整性约束条件>]);

列级完整性约束条件有 NULL（空）、UNIQUE（取值唯一），NOT NULL UNIQUE 表示取值唯一且不能取空值。

【例 3-10】 建立一个供应商、零件数据库。其中，“供应商”表 S（Sno，Sname，Status，City）分别表示供应商代码、供应商名、供应商状态和供应商所在城市。“零件”表 P（Pno，Pname，Color，Weight，City）分别表示零件号、零件名、颜色、重量和产地。数据库应满足如下要求。

（1）供应商代码不能为空，且值是唯一的。供应商的名也是唯一的。

（2）零件号不能为空，且值是唯一的。零件名不能为空。

（3）一个供应商可以供应多个零件，而一个零件可以由多个供应商供应。

解：根据题意，应分别建立供应商和零件关系模式。供应商和零件之间是一个多对多的联系。在关系数据库中，多对多联系必须生成一个关系模式，而该模式的码是该联系两端实体的码加上联系的属性构成的，若该联系名为 SP，则关系模式为 SP（Sno，Pno，Qty），其中，Qty 表示零件的数量。

用 CREATE 建立一个供应商、零件表数据库如下：

```
CREATE TABLE S ( Sno    CHAR(5) NOT NULL UNIQUE,
                 Sname    CHAR(30) UNIQUE,
                 Status    CHAR(8),
                 City    CHAR(20),
                 PRIMARY KEY(Sno));
CREATE TABLE P (Pno    CHAR(6),
                 Pname    CHAR(30) NOT NULL,
                 Color    CHAR(8),
                 Weight NUMERIC(6,2),
                 City    CHAR(20),
                 PRIMARY KEY(Pno));
CREATE TABLE SP (Sno    CHAR(5),
                 Pno    CHAR(6),
                 Qty    NUMERIC (9),
                 PRIMARY KEY(Sno,Pno),
                 FOREIGN KEY(Sno) REFERENCES S(Sno),
                 FOREIGN KEY(Pno) REFERENCES P(Pno));
```

从上述定义可以看出，Sno CHAR(5) NOT NULL UNIQUE 语句定义了 Sno 的列级完整性约束条件，取值唯一且不能取空值。

需要说明的是：

（1）PRIMARY KEY(Sno)已经定义了 Sno 为主码，所以 Sno　CHAR(5) NOT NULL UNIQUE 语句中的 NOT NULL UNIQUE 可以省略。

（2）FOREIGN KEY(Sno) REFERENCES S(Sno)定义了在 SP 关系中 Sno 为外码，其取值必须来自 S 关系的 Sno 域。同理，在 SP 关系中 Pno 也为外码。

2．修改表和删除表

1）修改表（ALTER TABLE）

语句格式：

```
ALTER TABLE <表名>[ADD<新列名><数据类型>[完整性约束条件]]
                  [DROP<完整性约束名>]
                  [MODIFY <列名><数据类型>];
```

例如，向“供应商”表 S 增加 Zap“邮政编码”可用如下语句：

```
ALTER TABLE S ADD Zap CHAR(6);
```

注意，不论基本表中原来是否已有数据，新增加的列一律为空。

又如，将 Status 字段改为整型可用如下信息：

```
ALTER TABLE S MODIFY Status INT;
```

2）删除表（DROP TABLE）

语句格式：

```
DROP TABLE <表名>;
```

例如，执行 DROP TABLE Student，此后关系 Student 不再是数据库模式的一部分，关系中的元组也无法访问。

3．定义和删除索引

数据库中的索引与书籍中的索引类似。在书中利用索引可以快速查找所需信息，无须阅读整本书。在数据库中，利用索引可以快速查找所需数据。数据库中的索引是某个表中一列或者若干列值的集合，以及相应的指向表中物理标识这些值的数据页的逻辑指针清单。

索引分为聚集索引和非聚集索引。聚集索引是指索引表中索引项的顺序与表中记录的物理顺序一致的索引。

1）建立索引

语句格式：

```
CREATE [UNIQUE][CLUSTER]INDEX <索引名>
                    ON <表名>（<列名>[<次序>][, <列名>[<次序>]]…）
```

参数说明如下。

- 次序： 可选 ASC（升序）或 DSC（降序），默认值为 ASC。
- UNIQUE：表明此索引的每一个索引值只对应唯一的数据记录。
- CLUSTER：表明要建立的索引是聚集索引，意为索引项的顺序是与表中记录的物理顺序一致的索引组织。

【例 3-11】 假设供应销售数据库中有供应商 S、零件 P、工程项目 J、供销情况 SPJ 关系，对供应商关系 S 依照 Sno 按升序建立索引；对零件 P 依 Pno 按升序建立索引；对工程项目 J 依 Jno 按升序建立索引；对供销情况 SPJ 依 Sno 按升序、Pno 按降序、Jno 按升序建立索引。

解：根据题意建立的索引如下。

```
CREATE UNIQUE INDEX S-SNO ON S(Sno);
CREATE UNIQUE INDEX P-PNO ON P(Pno);
CREATE UNIQUE INDEX J-JNO ON J(Jno);
CREATE UNIQUE INDEX SPJ-NO ON SPJ(Sno ASC,Pno DESC,JNO ASC)。
```

2）删除索引

语句格式：

```
DROP INDEX <索引名>;
```

例如，执行 DROP INDEX StudentIndex; 此后索引 StudentIndex 不再是数据库模式的一部分。

4．定义、删除、更新视图

视图是从一个或者多个表或视图中导出的表，其结构和数据是建立在对表的查询基础上的。与真实的表一样，视图也包括几个被定义的数据列和多个数据行，但从本质上讲，这些数据列和数据行来源于其所引用的表。因此，视图不是真实存在的基本表，而是一个虚拟表。

1）视图的创建

语句格式：

```
CREATE VIEW 视图名 [(列名)[,<列名>]]
        AS <子查询>
        [WITH CHECK OPTION];
```

注意，视图的创建中必须遵循如下规定。

① 子查询可以是任意复杂的 SELECT 语句，但通常不允许含有 ORDER BY 子句和 DISTINCT 短语。

② WITH CHECK OPTION 表示对 UPDATE、INSERT、DELETE 操作时保证更新、插入或删除的行满足视图定义中的谓词条件（即子查询中的条件表达式）。

③ 组成视图的属性列名或者全部省略或者全部指定。如果省略属性列名，则隐含该视图由 SELECT 子查询目标列的诸属性组成。

【例 3-12】 设由学号、姓名、年龄、性别、所在系构成的学生关系模式 Student 为（Sno，Sname，Sage，Sex，SD），建立“计算机系”（CS 表示计算机系）学生的视图，并要求进行修改、插入操作时保证该视图只有计算机系的学生。

```
CREATE VIEW CS-STUDENT
    AS SELECT Sno,Sname,Sage,Sex
    FROM Student
    WHERE SD='CS'
    WITH CHECK OPTION;
```

由于 CS-STUDENT 视图使用了 WITH CHECK OPTION 子句，因此，对该视图进行修改、插入操作时 DBMS 会自动加上 SD='CS'的条件，保证该视图中只显示计算机系的学生。

2）视图的撤销

语句格式：

```
DROP VIEW <视图名>;
```

3.6.3 SQL 数据查询

SQL 的数据操纵功能包括 SELECT（查询）、INSERT（插入）、DELETE（删除）和 UPDATE（修改）这 4 条语句。SQL 语言对数据库的操作十分灵活方便，原因在于 SELECT 语句中成分丰富多样的元组，有许多可选形式，尤其是目标列和条件表达式。

1．SELECT 基本结构

数据库查询是数据库的核心操作，SELECT 语句是用于查询的 SQL 语句。

语句格式：

```
SELECT [ALL|DISTINCT]<目标列表达式>[,<目标列表达式>]…
                FROM <表名或视图名>[,<表名或视图名>]
                [WHERE <条件表达式>]
                [GROUP BY <列名 1>[HAVING<条件表达式>]]
                [ORDER BY <列名 2>[ASC|DESC]…];
```

SQL 查询中的子句顺序为 SELECT、FROM、WHERE、GROUP BY、HAVING 和 ORDER BY。其中，SELECT、FROM 是必须的，HAVING 子句只能与 GROUP BY 搭配起来使用。

- SELECT 子句对应的是关系代数中的投影运算，用来列出查询结果中的属性。其输出可以是列名、表达式、集函数（AVG、COUNT、MAX、MIN、SUM），DISTINCT 选项可以保证查询的结果集中不存在重复元组。
- FROM 子句对应的是关系代数中的笛卡儿积，它列出的是表达式求值过程中需扫描的关系，即在 FROM 子句中出现多个基本表或视图时，系统首先执行笛卡儿积操作。
- WHERE 子句对应的是关系代数中的选择谓词，其条件表达式中可以使用的运算符如表 3-3 所示。

典型的 SQL 查询形式如下，对应的关系代数表达式为 $\pi_{A_1,A_2,\cdots,A_n}\left(\sigma_p\left(r_1\times r_2\times\cdots\times r_m\right)\right)$。

```
SELECT   A1,A2,…,An
  FROM   r1,r2,…,rm
  WHERE  p;
```

表 3-3　WHERE 子句的条件表达式中可以使用的运算符

运　算　符		含　义	运　算　符		含　义
集合成员运算符	IN	在集合中	算术运算符	>	大于
	NOT IN	不在集合中		⩾	大于等于
字符串匹配运算符	LIKE	与_和%进行单个或多个字符匹配		<	小于
				⩽	小于等于
空值比较运算符	IS NULL	为空		=	等于
	IS NOT NULL	不能为空		≠	不等于
			逻辑运算符	AND	与
				OR	或
				NOT	非

2．简单查询

最简单的查询是找出关系中满足特定条件的元组。简单查询只需要使用 3 个保留字 SELECT、FROM 和 WHERE。

【例 3-13】 设有学生、课程和学生选课关系，其基本表分别用 S、C 和 SC 表示。

S (Sno,Sname,SD,Sage,Sex)　属性表示学号、姓名、系、年龄和性别
C (Cno,Cname,Pcno)　属性表示课程号、课程名和先修课程号
SC (Sno,Cno,Grade)　属性表示学号、课程号和成绩

（1）查询学生基本表中计算机系 CS 学生的学号、姓名及年龄。

```
SELECT  Sno，Sname，Sage
  FROM  S
  WHERE  SD='CS';
```

一般情况下，为了便于理解查询语句的结构，在写 SQL 语句时要将保留字如 FROM 或 WHERE 作为每一行的开头。如果一个查询或子查询非常短，也可以直接将它们写在一行上。

（2）查询数学系 MS 全体学生的详细信息。

```
SELECT  *  FROM  S  WHERE  SD='MS';
```

（3）查询学生的出生年份（当前年份为 2009）。

```
SELECT  Sno, 2009-Sage  FROM  S;
```

3．连接查询

若查询涉及两个以上的表，则称为连接查询。

【例 3-14】 以例 3-13 中 S、C 和 SC 为基本表，用连接查询和嵌套查询实现。

（1）检索选修了课程号为 C1 的学生，列出学号和学生姓名。

```
SELECT  Sno, Sname
  FROM  S, SC
  WHERE S.Sno=SC.Sno AND SC.Cno='C1';
```

（2）检索选修课程名为 MS 的学生号和学生姓名。

```
SELECT  Sno, Sname
  FROM  S, SC, C
  WHERE  S. Sno=SC.Sno AND SC.Cno=C.Cno AND C.Cname='MS';
```

（3）检索至少选修了课程号为 C1 和 C3 的学生学号。

```
SELECT  Sno
  FROM  SC SCX,  SC SCY
  WHERE  SCX.Sno=SCY. Sno AND SCX.Cno=' C1' AND SCY.Cno='C3';
```

4．子查询与聚集函数

1）子查询

子查询也称为嵌套查询。嵌套查询是指一个 SELECT-FROM-WHERE 查询块可以嵌入另一个查询块之中。在 SQL 中允许多重嵌套。

【例 3-15】 用嵌套查询检索选修课程名为 MS 的学生学号和姓名。

```
SELECT   Sno, Sname
  FROM   S
  WHERE   Sno IN (SELECT   Sno
                  FROM SC
                  WHERE   Cno IN (SELECT   Cno
                                  FROM C
                                  WHERE   Cname=' MS'));
```

其中，SELECT　Cno FROM C WHERE　Cname=' MS'从课程基本表中查出名为 MS 的课程编号；然后根据该课程编号，在学生选课关系基本表中找出所有选修了该课程的学生的学号；最后在学生关系基本表中查出这些学生的姓名。

2）聚集函数

聚集函数以一个值的集合为输入，返回单个值的函数。SQL 提供了 5 个预定义的集函数：平均值 AVG、最小值 MIN、最大值 MAX、求和 SUM 以及计数 COUNT。如表 3-4 所示。

表 3-4　集函数的功能

集 函 数 名	功　　能
COUNT([DISTINCT|ALL]*)	统计元组个数
COUNT([DISTINCT|ALL]<列名>)	统计一列中值的个数
SUM([DISTINCT|ALL]<列名>)	计算一列（该列应为数值型）中值的总和
AVG([DISTINCT|ALL]<列名>)	计算一列（该列应为数值型）值的平均值
MAX([DISTINCT|ALL]<列名>)	求一列值的最大值
MIN([DISTINCT|ALL]<列名>)	求一列值的最小值

使用 ANY 和 ALL 谓词必须同时使用比较运算符，其含义及等价的转换关系如表 3-5 所示。

表 3-5　ANY、ALL 谓词含义及等价的转换关系

谓　词	语　　义	等价转换关系
>ANY	大于子查询结果中的某个值	>MIN

>ALL	大于子查询结果中的所有值	>MAX
<ANY	小于子查询结果中的某个值	<MAX
<ALL	小于子查询结果中的所有值	<MIN
>=ANY	大于等于子查询结果中的某个值	>=MIN
>=ALL	大于等于子查询结果中的所有值	>=MAX
<=ANY	小于等于子查询结果中的某个值	<=MAX
<=ALL	小于等于子查询结果中的所有值	<=MIN
<>ANY	不等于子查询结果中的某个值	--
<>ALL	不等于子查询结果中的任何一个值	NOT IN
=ANY	等于子查询结果中的某个值	IN
=ALL	等于子查询结果中的所有值	--

用集函数实现子查询通常比直接用 ALL 或 ANY 查询效率要高。

【例 3-16】 查询选修了课程号为 C1 的学生的最高分和最低分以及高低分之间的差距。

```
SELECT MAX(Grade),MIN(Grade),MAX(Grade)-MIN(Grade)
  FROM SC
  WHERE Cno='C1';
```

【例 3-17】 查询比计算机系 CS 所有学生年龄都要小的其他系的学生姓名及年龄。

方法 1：（用 ALL 谓词）

```
SELECT   Sname, Sage
FROM   S
WHERE   Sage < ALL (SELECT   Sage
                    FROM S
                    WHERE   SD='CS')
          AND SD<>'CS';
```

其中，用 SELECT Sage FROM S WHERE SD='CS'先查出计算机系学生的所有年龄值。

方法 2：（用 MIN 集函数）从表 3-4 可见，<ALL 可用<MIN 代换，结果如下。

```
SELECT   Sname, Sage
  FROM   S
  WHERE   Sage< (SELECT   MIN (Sage)
                    FROM S
                    WHERE   SD='CS'   )
           AND SD<>'CS';
```

其中，用 SELECT MIN (Sage) FROM S WHERE SD='CS'先查出计算机系年龄最小的学生的年龄，只要其他系的学生年龄比这个年龄小，那么就应在结果集。

【例 3-18】 查询其他系中比计算机系某一学生年龄小的学生姓名及年龄。

方法 1：（用 ANY 谓词）

```
SELECT   Sname, Sage
  FROM   S
  WHERE   Sage< ANY (SELECT   Sage
                     FROM S
                     WHERE   SD='CS')
            AND SD<>'CS';
```

方法 2：（用 MAX 集函数）从表 3-4 可见，<ANY 可用<MAX 代换，结果如下。

```
SELECT   Sname, Sage
  FROM   S
  WHERE   Sage < (SELECT   MAX (Sage)
                  FROM S
                  WHERE   SD='CS'   )
          AND SD<>'CS';
```

方法 2 实际上是找出计算机系年龄最大的学生的年龄，只要其他系的学生年龄比这个年龄小，那么就应在结果集中。

5．分组查询

1）GROUP BY 子句

在 WHERE 子句后面加上 GROUP BY 子句可以对元组进行分组，保留字 GROUP BY 后面跟着一个分组属性列表。SELECT 子句中使用的聚集操作符仅用在每个分组上。

【例 3-19】 对于学生数据库中的 SC 关系，查询每个学生的平均成绩。

```
SELECT   Sno, AVG(Grade)
      FROM   SC
      GROUP BY Sno;
```

该语句是将 SC 关系的元组重新组织，并进行分组使得学号为 3001 的元组被组织在一起，3002 的元组被组织在一起，依次类推，然后分别求出每个学生的平均成绩。

2）HAVING 子句

假如元组在分组前按照某种方式加上限制，使得不需要的分组为空，在 GROUP BY 子句后面跟一个 HAVING 子句即可。

【例 3-20】 对于供应商数据库中的 S、P、J、SPJ 关系，查询至少有三家供应商（包含三家）供应零件的工程项目，列出工程号及所用零件的平均数量，并按工程号降序排列。

```
SELECT   Jno，AVG(Qty)
    FROM   SPJ
    GROUP BY Jno
    HAVING COUNT(DISTINCT(Sno)) > 2
    ORDER BY Jno DESC;
```

根据题意“某工程至少用了三家供应商（包含三家）供应的零件”，应该按照工程号分组（GROUP BY Jno），同时用“HAVING COUNT(DISTINCT(Sno)) > 2”限定供应商的数目不得少于三个。

需要注意的是，一个工程项目可能用了同一个供应商的多种零件，因此，在统计供应商数目的时候需要加上 DISTINCT，以避免重复统计导致错误的结果。例如，按工程号 Jno='J1'分组的结果如表 3-6 所示（Sno 为供应商代码，Pno 为零件号，Jno 为工程号），统计供应商的数目时，“COUNT(DISTINCT(Sno))”的统计结果为 5，“COUNT(Sno)”的统计结果为 7。

表 3-6　按工程号 Jno='J1'分组

Sno	Pno	Jno	Qty
S1	P1	J1	200
S2	P3	J1	400
S2	P3	J1	200
S2	P5	J1	100
S3	P1	J1	200
S4	P6	J1	300
S5	P3	J1	200

当元组含有空值时，需要注意：

（1）空值在任何聚集操作中被忽视。它对求和、求平均值和计数都没有影响。它也不能是某列的最大值或最小值。例如，COUNT(*)是某个关系中所有元组的个数，但 COUNT(A)却是 A 属性非空的元组个数。

（2）NULL 值又可以在分组属性中看作是一个一般的值。例如，SELECT A ,AVG(B) FROM R 中，当 A 的属性值为空时，就会统计 A IS NULL 的所有元组中 B 的平均值。

6．使用别名

SQL 用 AS 子句为关系和属性指定不同的名称或别名，以增加可读性，其格式为

```
Old-name AS new-name
```

AS 子句既可出现在 SELECT 子句中，也可出现在 FROM 子句中。

【例 3-21】 查询计算机系学生的 Sname 和 Sage，但 Sname 用姓名表示，Sage 用年龄表示。其语句如下：

```
SELECT   Sname AS  姓名, Sage AS  年龄
  FROM   S
  WHERE SD='CS'
```

SQL 中的元组变量必须和特定的关系相联系。在 FROM 子句中，元组变量通过 AS 子句来定义。

【例 3-22】 查询选修了“C1”课程的学生姓名 Sname 和成绩 Grade。其语句如下：

```
SELECT   Sname, Grade
  FROM   S AS x, SC AS y
  WHERE   x.Sno=y.Sno AND y.Cno= 'C1'
```

元组变量在比较同一关系的两个元组时非常有用。

7．字符串操作

操作符 LIKE 用于对字符串进行模式匹配。使用两个特殊的字符来描述模式：“%”匹配任意字符串；“_”匹配任意一个字符。模式是大小写敏感的。例如：

Marry%匹配任何以 Marry 开头的字符串；%idge%匹配任何包含 idge 的字符串，如 Marryidge、Rock Ridge、Mianus Bridge 和 Ridgeway。

“_ _”匹配只含两个字符的字符串；“_ _%”匹配至少包含两个字符的字符串。

【例 3-23】 学生关系模式为 S（Sno,Sname,Sex,SD,Sage,Addr），其中，Sno 为学号，Sname 为姓名，Sex 为性别，SD 为所在系，Sage 为年龄，Addr 为家庭住址。

① 查询家庭住址包含“科技路”的学生姓名。

② 检索单姓并且名字为“晓军”的学生姓名、年龄和所在系。

解：① 查询家庭住址包含“科技路”的学生姓名。

```
SELECT    Sname
    FROM    S
    WHERE    Addr LIKE '%科技路%';
```

② 查询单姓并且名字为“晓军”的学生姓名、年龄和所在系。

```
SELECT    Sname,Sage,SD
      FROM    S
      WHERE    Sname LIKE '__晓军';
```

为了使模式中包含特殊模式字符（即“%”和“_”），在 SQL 中允许使用 ESCAPE 关键词来定义转义符。转义字符紧靠着特殊字符，并放在它前面，表示该特殊字符被当成普通字符。

例如，匹配所有以 ab%cd 开头的字符串。

```
LIKE 'ab\%cd%' ESCAPE '\'
```

其中，由 ESCAPE '\'定义了一个转义符号“\”，说明“\%”中的%不是通配符。

例如，匹配所有以 ab_cd 开头的字符串。

```
LIKE 'ab!_cd%' ESCAPE '!'
```

其中，由 ESCAPE '!'定义了一个转义符号“!”，说明“!_”中的“_”不是通配符。

8．视图的查询

【例 3-24】 建立“计算机系”（CS 表示计算机系）学生的视图如下，并要求进行修改、插入操作时保证该视图只有计算机系的学生。

```
CREATE VIEW CS-STUDENT
    AS SELECT Sno,Sname,Sage,Sex
    FROM S
    WHERE SD='CS'
    WITH CHECK OPTION;
```

此时要查询计算机系年龄小于 20 岁的学生学号及年龄的 SQL 语句如下：

```
SELECT    Sno, Sage    FROM    CS-STUDENT    WHERE Sage<20;
```

该语句执行时，通常先转换成等价的对基本表的查询，然后执行查询语句。即当查询视图表时，系统先从数据字典中取出该视图的定义，然后将定义中的查询语句和对该视图的查询语句结合起来，形成一个修正的查询语句。对上例修正之后的查询语句为：

```
SELECT    Sno, Sage    FROM S    WHERE SD='CS' AND Sage<20;
```

3.6.4 SQL 数据更新

1．插入语句

可以指定一个元组或者用查询语句选出一批元组，向基本表中插入数据。插入语句的基本格式如下：

```
INSERT INTO <基本表名>[（字段名[,字段名]…）]
            VALUES(常量[,常量]…);
INSERT INTO <基本表名>（字段名[,字段名]…）
            SELECT  查询语句;
```

【例 3-25】 将学号为 3002、课程号为 C4、成绩为 98 的元组插入 SC 关系中。

```
INSERT    INTO    SC
      VALUES('3002', 'C4',98);
```

在某些情况下，可以通过定义的视图向基本表中插入数据。

【例 3-26】 首先创建一个基于表 employees 的新视图 v_employees，然后使用该视图向表 employees 中添加一条新的数据记录。

```
CREATE    VIEW    v_employees(number, name, age, sex, salary, dept)
   AS
   SELECT    number, name, age, sex, salary, dept
      FROM    employees
      WHERE    dept = '客户服务部';


INSERT INTO v_employees
VALUES(001,'李力',22,'m',2000, '技术开发部');
```

2．删除语句

```
DELETE FROM <基本表名>
[WHERE  条件表达式];
```

【例 3-27】 删除表 employees 中姓名为张然的记录。

```
DELETE    FROM employees
      WHERE name='张然';
```

3．修改语句

```
UPDATE <基本表名>
SET <列名>=<值表达式>[,<列名>=<值表达式>…]
[WHERE <条件表达式>]
```

【例 3-28】 将教师的工资增加 5%。

```
UPDATE    teachers
      SET Salary = Salary*1.05;
```

【例 3-29】 将教师的工资小于 1000 的增加 5%的工资。

```
UPDATE    teachers
      SET Salary = Salary*1.05
      WHERE Salary < 1000;
```

也可以使用视图更新基本表中的数据记录。应该注意的是，更新的只是数据库中的基本表。

【例 3-30】 创建一个基于表 employees 的视图 v_employees，然后通过该视图修改表 employees 中的记录，语句如下：

```
CREATE    VIEW    v_employees
   AS
   SELECT    *    FROM    employees
      UPDATE    v_employees
      SET name='张然'
      WHERE    name='张三'
```

3.6.5 SQL 的访问控制

访问控制是指对数据访问的控制，分为授权语句和收回权限语句。

1．授权语句的格式

```
GRANT <权限>[,<权限>]…
      [ON<对象类型><对象名>]
      TO <用户>[,<用户>]…
      [WITH GRANT OPTION];
```

注意，不同类型的操作对象有不同的操作权限，常见的操作权限如表 3-7 所示。

表 3-7　常见的操作权限

对象	对象类型	操 作 权 限
属性列	TABLE	SELECT, INSERT, UPDATE, DELETE, ALL PRIVILEGES（4 种权限的总和）
视图	TABLE	SELECT, INSERT, UPDATE, DELETE, ALL PRIVILEGES（4 种权限的总和）
基本表	TABLE	SELECT, INSERT, UPDATE, DELETE, ALTER, INDEX, ALL PRIVILEGES（6 种权限的总和）
数据库	DATABASE	CREATETAB 建立表的权限，可由 DBA 授予普通用户

说明：

（1）接收权限的用户可以是单个或多个具体的用户，也可以是 PUBLIC，即全体用户。

（2）若指定了 WITH GRANT OPTION 子句，则获得了某种权限的用户还可以将此权限赋给其他用户。

【例 3-31】 把数据库 SPJ 中供应商 S、零件 P、项目 J 表的各种权限赋予指定用户。

（1）将对供应商 S、零件 P、项目 J 的所有操作权限赋给用户 User1 及 User2，其授权语句如下：

```
GRANT ALL PRIVILEGES ON TABLE S, P, J    TO    User1, User2;
```

（2）将对供应商 S 的插入权限赋给用户 User1，并允许将此权限赋给其他用户，其授权语句如下：

```
GRANT INSERT ON TABLE S TO    User1 WITH GRANT OPTION;
```

（3）DBA 把数据库 SPJ 中建立表的权限赋给用户 User1，其授权语句如下：

```
GRANT CREATETAB ON DATABASE SPJ TO    User1;
```

2．收回权限语句格式

```
REVOKE <权限>[,<权限>]…
    [ON<对象类型><对象名>]
    FROM <用户>[,<用户>]…;
```

【例 3-32】 收回用户对数据库 SPJ 中供应商 S、零件 P、项目 J 表的操作权限。

（1）将用户 User1 及 User2 对供应商 S、零件 P、项目 J 的所有操作权限收回。

```
REVOKE ALL PRIVILEGES ON TABLE S, P, J FROM    User1, User2;
```

（2）将所有用户对供应商 S 的所有查询权限收回。

```
REVOKE SELECT ON TABLE S FROM PUBLIC;
```

（3）将 User1 用户对供应商 S 的供应商编号 Sno 的修改权限收回。

```
REVOKE UPDATE(Sno) ON TABLE S FROM User1;
```

3.6.6 嵌入式 SQL

SQL 提供了将 SQL 语句嵌入到某种高级语言中的使用方式，需要处理的关键问题是如何识别嵌入的 SQL 语句，如何处理主语言和 SQL 间的通信问题。

DBMS 可采用两种处理方法：一种是预编译，另一种是修改和扩充主语言使之能处理 SQL 语句。目前采用较多的是预编译的方法，即由 DBMS 的预处理程序先扫描源程序，识别出 SQL 语句，然后把它们转换为主语言调用语句，以使主语言编译程序能识别它，最后由主语言的编译程序将整个源程序编译成目标代码。

为了区分主语言与 SQL 语言，需要在所有的 SQL 语句之前加前缀“EXEC SQL”，而 SQL 的结束标志随主语言的不同而不同。

例如，在 PL/1 和 C 中以分号“;”结束，格式为 EXEC SQL <SQL 语句>;。

在 COBOL 语言中以“END-EXEC”结束，格式为 EXEC SQL <SQL 语句> END-EXEC。

例如，SQL 语句“DROP TABLE Student;”嵌入到 C 程序时，应写作：

```
EXEC SQL DROP TABLE    Student;
```

嵌入式 SQL 与主语言之间的通信采用如下 3 种方式。

（1）SQL 通信区（SQL Communication Area，SQLCA）。向主语言传递 SQL 语句执行的状态信息，使主语言能够根据此信息控制程序流程。

（2）主变量，也称为共享变量。通过主变量由主语言向 SQL 语句提供参数，主变量由主语言的程序定义，并用 SQL 的 DECLARE 语句说明。引用时，为了与 SQL 属性名相区别，需在主变量前加“:”。

【例 3-33】 根据共享变量 given_sno 值查询学生关系 S 中学生的姓名、年龄和性别。

```
EXEC SQL SELECT Sname,Sage,Sex
      INTO :Mname,:Mage,:Msex
      FROM S
WHERE sno=:given_sno;
```

（3）游标。SQL 语言是面向集合的，一条 SQL 语句可产生或处理多条记录。而主语言是面向记录的，一组主变量一次只能放一条记录，所以，引入游标，通过移动游标指针来决定获

取哪一条记录。

3.7 数据库设计

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，使之能有效地存储数据，满足各种用户的需求（信息要求和处理要求）。

数据库设计过程参照软件系统生命周期的划分方式，把数据库应用系统的生命周期分为数据库规划、需求描述与分析、数据库设计与应用程序设计、实现、测试、运行维护6个阶段。

（1）数据库规划。数据库规划是创建数据库应用系统的起点，是数据库应用系统的任务陈述和任务目标。任务陈述定义了数据库应用系统的主要目标，而每个任务目标定义了系统必须支持的特定任务。数据库规划过程还必然包括对工作量的估计、使用的资源和需要的经费等。同时，还应当定义系统的范围和边界以及它与信息系统的其他部分的接口。

（2）需求描述与分析。需求描述与分析是以用户的角度，从系统中的数据和业务规则入手，收集和整理用户的信息，以特定的方式加以描述，是下一步工作的基础。

（3）数据库设计与应用程序设计。数据库设计是对用户数据的组织和存储设计；应用程序设计是在数据库设计基础上对数据操作及业务实现的设计，包括事务设计和用户界面设计。

（4）数据库系统实现。数据库系统实现是依照设计，使用 DBMS 支持的数据定义语言实现数据库的建立，用高级语言（Basic、Delphi、C、C++和 Power Builder 等）编写应用程序。

（5）测试阶段。测试阶段是在数据系统投入使用之前，通过精心制定的测试计划和测试数据来测试系统的性能是否满足设计要求，发现问题。

（6）运行维护。数据库应用系统经过测试、试运行后即可正式投入运行。运行维护是系统投入使用后，必须不断地对其进行评价、调整与修改，直至系统消亡。

在任一设计阶段，一旦发现不能满足用户数据需求时，均需返回到前面的适当阶段，进行必要的修正。经过如此的迭代求精过程，直到能满足用户需求为止。在进行数据库结构设计时，应考虑满足数据库中数据处理的要求，将数据和功能两方面的需求分析、设计和实现在各个阶段同时进行，相互参照和补充。

在数据库设计中，对每一个阶段设计成果都应该通过评审。评审的目的是确认某一阶段的任务是否全部完成，从而避免出现重大的错误或疏漏，保证设计质量。评审后还需要根据评审意见修改所提交的设计成果，有时甚至要回溯到前面的某一阶段，进行部分重新设计乃至全部重新设计，然后再进行评审，直至达到系统的预期目标为止。

1．数据库设计的基本步骤

在确定了数据库设计的策略以后，就需要相应的设计方法和步骤。多年来，人们提出了多种数据库设计方法，多种设计准则和规范。

1978年10月召开的新奥尔良（New Orleans）会议提出的关于数据库设计的步骤，简称新奥尔良法，是目前得到公认的，较为完整和权威的数据库设计方法，它把数据库设计分为如下4个主要阶段。

（1）用户需求分析。是指数据库设计人员采用一定的辅助工具对应用对象的功能、性能和限制等要求所进行的科学分析。需求分析阶段生成的结果主要包括数据和处理两个方面。

① 数据。数据字典、全系统中的数据项、数据流和数据存储的描述。

② 处理。数据流图和判定表、数据字典中处理过程的描述。

（2）概念结构设计。概念结构设计是对信息分析和定义，如视图模型化、视图分析和汇总。对应用对象精确地抽象、概括而形成的独立于计算机系统的企业信息模型。描述概念模型常用的工具是E-R图。E-R图的设计要对需求分析阶段所得到的数据进行分类、聚集和概括，确定实体、属性和联系。概念结构的具体工作步骤包括选择局部应用，逐一设计分 E-R 图，进行E-R图合并形成基本的E-R图。

（3）逻辑结构设计。逻辑结构设计的目的是把概念设计阶段的概念模型（如基本E-R图）转换成与选用的具体机器上的DBMS所支持的逻辑模型，即将抽象的概念模型转化为与选用的DBMS产品所支持的数据模型（如关系模型）相符合的逻辑模型，它是物理设计的基础。包括模式初始设计、子模式设计、应用程序设计、模式评价以及模式求精。

逻辑设计可分为如下3步。

① 将概念模型（E-R图）转换为一般的关系、网状、层次模型。

② 将关系、网状、层次模型向特定的DBMS支持下的数据模型转换。

③ 对数据模型进行优化。

（4）物理结构设计。物理结构设计是指逻辑模型在计算机中的具体实现方案。数据库在物理设备上的存储结构与存取方法称为数据库的物理结构，对于一个给定的逻辑数据模式选取一个最适合应用环境的物理结构的过程，称为数据库的物理设计。通常对于关系数据库物理设计的主要内容包括为关系模式选择存取方法、设计关系、索引等数据库文件的物理结构。

当各阶段发现不能满足用户需求时，均需返回到前面适当的阶段，进行必要的修正。经过如此不断地迭代和求精，直到各种性能均能满足用户的需求为止。

2．数据库的实施与维护

数据库设计结束后进入数据库的实施与维护阶段，在该阶段中主要有如下工作。

（1）数据库实现阶段的工作。建立实际数据库结构；试运行；装入数据。

（2）其他有关的设计工作。数据库的重新组织设计；故障恢复方案设计；安全性考虑；事务控制。

（3）运行与维护阶段的工作。数据库的日常维护（安全性、完整性控制，数据库的转储和恢复）；性能的监督、分析与改进；扩充新功能；修改错误。