I 预备篇

第一章　数据库系统引论

1.1　数据管理的发展

在众多的计算机应用中，有一类重要的计算机应用，称为数据密集型应用(data intensive applications)。它有下列三个特点：

(1) 涉及的数据量大，一般需存放在辅助存储器中，内存中只能暂存其中很小的一部分；

(2) 数据不随程序的结束而消失，而需长期保留在计算机系统中，这种数据称为持久数据(persistent data)；

(3) 数据为多个应用程序所共享，甚至在一个单位或更大范围内共享。

这是最大的计算机应用领域，管理信息系统、办公信息系统、银行信息系统、民航订票系统、情报检索系统等都属于这一类。管理这种大量的、持久的、共享的数据是这类计算机应用面临的共同问题。从五十年代末以来，数据管理一直是计算机科学技术领域中的一门重要的技术和研究课题。

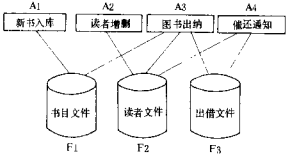


图 1-1　图书馆文件系统

早期的数据管理都采用文件系统(file system)。在文件系统中，数据按其内容、结构和用途组成若干命名的文件。文件一般为某一用户或用户组所有，但可供指定的其他用户共享。用户可以通过操作系统对文件进行打开、读、写和关闭等操作。图 1-1 是一个图书馆文件系统的例子，其中共有 F1、F2 和 F3 三个文件，A1、A2、A3 和 A4 是建立在这三个文件上的四个应用程序，图中的实线表示每个应用程序所访问的文件。例如，图书出纳应用程序若要处理一个借书申请，须访问 F1，查询有无此书；访问 F2，查询借阅者是否为本馆的读者；访问 F3，查询此书是否借出。这样的文件系统明显地有下列五个缺点。

(1) 编写应用程序很不方便。应用程序的设计者必须对所用的文件的逻辑及物理结构有清楚的了解。操作系统只提供打开、关闭、读、写等几个低级的文件操作命令，对文件的查询、修改等处理都须在应用程序内解决。应用程序还不可避免地在功能上有所重复，例如 A2、A3、A4 三个应用程序都要有查询 F2 的功能。因此，在文件系统上编写应用程序的生产率不高。如果有新的要求，例如要求提供某一专题书目，则必须请熟悉文件的程序员编写相应的应用程序，一般需要较长的时间，而不能由用户直接向系统查询。

(2) 文件的设计很难满足多种应用程序的不同要求，数据冗余往往是不可避免的。例

如，A2 要求 F2 包含读者的所有属性，而 A3 只要从 F2 查询借阅者是否为合法读者，A4 只要从 F2 查阅读者的地址；对文件的划分也有不同的要求，例如，A4 希望把读者的地址列入 F3 中，可以在处理时少打开一个文件，但这又与 F2 的内容重复，成为数据冗余（redundancy），而且 A3 又无此要求，为了兼顾各种应用程序的要求，在设计文件系统时，往往不得不增加数据冗余，有时甚至会出现大量的数据冗余。数据冗余不仅浪费存储空间，而且会带来数据的不一致（inconsistency）。例如，在 F2 和 F3 中都列入读者地址，如果读者搬家了，只修改 F2，不修改 F3，则会造成数据的不一致。在文件系统中，无维护数据一致性的监控机制，数据的一致性完全由用户负责维护。这在简单系统中还可勉强应付，在复杂的系统中，要保证数据的一致性，几乎是不可能的。

（3）文件结构的每一修改将导致应用程序的修改，应用程序的维护工作量很大。随着应用的环境和需求的变化，修改文件的结构是常有的事。例如在某些文件的记录中增加一些字段，扩充某些字段的长度，改变某些字段的计量单位及表示格式，把书目文件按学科分为几个文件等，都会引起应用程序的一连串的修改。众所周知，修改程序是很麻烦的事，首先要熟悉原有的程序，修改后还要对程序进行测试，以免因修改而引起不希望的副作用。这些问题之所以产生，主要是由于应用程序对文件的过分依赖；或者说，文件系统的数据独立性（data independence）不好。

（4）文件系统一般不支持对文件的并发访问（concurrent access）。在现代计算机系统中，为了有效地利用计算机的资源，一般允许多个应用程序并发地运行。文件当初就是作为某个程序的附属数据出现的。文件系统一般不支持多个应用程序对同一文件的并发访问，例如在图 1-1 中，如果在 A2 修改 F2 的同时，A3 或 A4 并发地查询 F2，则 A3 或 A4 读得的数据可能是正在修改中的数据，有可能不一致，甚至错误。为了避免这个问题的发生，可限制对文件的并发访问，或者为每个应用程序准备一个文件的复本。前一种办法降低了程序的并发程度；后一种办法增加了数据的冗余，从而会引起数据的不一致。

（5）由于数据缺少统一管理，在数据的结构、编码、表示格式、命名以及输出格式等方面不容易做到规范化、标准化；在数据的安全和保密方面，也难以采取有效的措施。

针对文件系统的上述缺点，人们逐步发展了以统一管理和共享数据为主要特征的数据库系统（database system）。在数据库系统中，数据不再仅仅服务于某个程序或用户，而是看成一个单位的共享资源，由一个叫做数据库管理系统（database management system，简称 DBMS）的软件统一管理。由于有 DBMS 的统一管理，应用程序不必直接介入诸如打开、关闭、读、写文件等低级操作，而由 DBMS 代办。用户也不必关心数据存储和其他实现的细节，可以在更高的抽象级别上观察和访问数据。文件结构的一些修改也可以由 DBMS 屏蔽，使用户看不到这些修改，从而减少应用程序的维护工作量，提高数据的独立性。由于数据的统一管理，人们可以从全单位着眼，合理组织数据，减少数据冗余；还可以更好地贯彻规范化和标准化，从而有利于数据的转移和更大范围内的共享。由于 DBMS 不是为某个应用程序服务，而是为整个单位服务，DBMS 做得复杂一些也是可以接受的。许多在文件系统中难以实现的功能在 DBMS 中都一一实现了。例如，适合不同类型用户的多种用户界面，保证并发访问时的数据一致性的并发控制（concurrent control），增进数据安全性（security）的访问控制（access control），在故障情况下保证数据一致性的恢复（recovery）功能，保证数据在语义上的一致性的完整性约束（integrity constraints）检查功能等。随着计算机应用的发展，

DBMS 的功能愈来愈强，规模愈来愈大，复杂性和开销也随之增加。目前，在一些功能非常明确且无数据共享问题的简单应用系统中，为了减少开销、提高性能，有时仍采用文件系统，不过在本节开头所列举的数据密集型应用系统中，基本上都使用数据库系统。

世界上第一个 DBMS 当推美国通用电气公司 Bachman 等人于 1964 年开发成功的 IDS (Integrated Data Store)。IDS 奠定了网状数据库(network database)的基础，并得到了广泛的发行和应用。IBM 公司于六十年代末推出了第一个商品化的层次(hierarchical)数据库管理系统 IMS(Information Management System)。层次和网状数据库描述数据的模型，是从过去应用程序处理数据时所用的数据结构概括而来的，尽管有一定的通用性，但其中保留了不少实现的细节，例如指针等。正如 Bachman 在接受 1973 年图灵奖的演说中所描写的那样，程序员好比领航员(navigator)，在错综复杂的数据库中航行。这说明在应用数据库时，一是需要熟悉数据库中的各种路径(相当于地图)，二是要有较高的应用数据库的技巧(相当于领航本领)。集中和共享这一基本目标在层次和网状数据库中已经达到，但用户观察和访问数据的抽象级别还不够高，数据独立性还不够好，数据库的使用也不够方便。层次和网状数据库是六十年代技术条件下的合理产物。它们为数据库技术奠定了基础，搭起了框架，打开了应用局面，至今还有这样的数据库在运行。但是，数据库技术最有意义的成就是关系数据库的发展。1970 年 Codd 提出了关系数据模型(relational data model)，以关系(relation)或称表(table)作为描述数据的基础。在其后的几年中，Codd 又发表了一系列文章，奠定了关系数据库的理论基础。

关系数据模型有严格的数学基础，抽象级别比较高，而且简单清晰，便于理解和使用。它一旦被提出，便立即受到数据库界的重视。但是，当时也有相当多的人认为关系数据模型仅仅是理想化的数据模型，用来实现 DBMS 是不现实的，尤其担心关系数据库的性能难以被用户接受。1974 年，数据库界开展了一场分别以 Codd 和 Bachman 为首的支持和反对关系数据库的大辩论。辩论促进了关系数据库的发展，不少工业和学术单位投入了关系 DBMS 原型的研制。因为只有研制出和已有 DBMS 的性能至少可以相比，而且实用的关系 DBMS，才能说明关系数据库的价值。在众多的关系 DBMS 原型中，功能最全面、技术上最有代表性的要算美国 IBM 公司的 System R 和加州大学 Berkeley 分校的 INGRES。这两个原型系统差不多都在 1977 年前后开始运行。它们不但证实了人们所期望的关系数据库的许多优点，例如高级的非过程语言接口、好的数据独立性等，而且也消除了人们对关系数据库性能的担心。这两个原型系统全面地提供了比较成熟的关系 DBMS 技术，为研制商品化的关系 DBMS 完成了技术上的准备。IBM 公司在 System R 的基础上先后推出了 SQL /DS 和 DB2 两个商品化的关系 DBMS。INGRES 也由 INGRES 公司商品化。到了八十年代，关系数据库成为发展的主流，几乎所有新推出的 DBMS 产品都是关系型的。数据库不但用在大型机和小型机上，在微机上也广泛地应用。除了上述的关系 DBMS 产品外，市场上又陆续出现了一系列产品，其中发行量比较大且在我国用得比较多的有 ORACLE、Sybase、Informix、Foxpro 等。随着微机的出现和计算机网络的广泛应用，分布式处理系统在八十年代得到很大的发展。与此相适应，分布式数据库系统也成为八十年代数据库研究的重点，并日趋成熟。目前，几乎所有分布式 DBMS 产品都是关系型的，而且几乎所有主要关系 DBMS 产品都扩充为分布式 DBMS。八十年代是关系数据库的全盛时代。随着计算机的广泛应用，新的应用又提出新的要求。人们开始发现关系数据库的许多限制和不足，这又推动了数据

库技术的新一轮的研究，道路无非有两条：一是改造和扩充关系数据库，以适应新的应用需求；二是改用新的数据模型，例如面向对象数据模型、基于逻辑的数据模型等，研制新型的数据库。目前，这两方面都取得了很大的进展，人们期望着一个“后关系数据库”(post-relational database)时代的到来。

数据管理和数据处理一样，都是计算机系统的最基本的支撑技术。尽管计算机科学技术经历了飞速的发展，但数据管理的这一地位没有变化。可以预期，数据管理将作为计算机科学技术的一个重要分支一直发展下去，社会愈信息化，对数据管理的要求也愈高。推动这门学科发展的动力是计算机应用，发展这门学科的基础是新的硬、软件技术。数据库是现阶段数据管理的主要形式。

1.2 数据库系统

数据库系统的简单结构如图 1-2 所示。图中的数据库是数据的汇集，它们以一定的组织形式存于存储介质上，一般是磁盘。数据库管理系统(DBMS)是管理数据库的软件，它实现数据库系统的各种功能。图中的应用是指以数据库为基础的各种应用程序，应用程序必须通过 DBMS 访问数据库。数据库既然是共享的，就需要有人进行数据库的规划、设计、协调、维护和管理等工作，负责这些工作的人员或集体称为数据库管理员(database administrator，简称 DBA)。应用程序、数据库管理系统、数据库和数据库管理员构成数据库系统。如果不引起混淆，数据库系统有时也简称数据库。

从图 1-2 可以看出，DBMS 是数据库系统的核心。DBMS 一般是通用软件，由厂家提供。不管是 DBMS 的研制人员，还是 DBMS 的使用人员，对 DBMS 的功能、结构和工作原理都应有所了解。DBMS 的功能因产品而异，下面的七个功能是现代 DBMS 一般应该具备的。

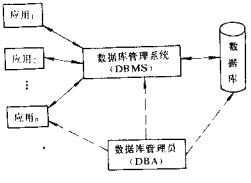


图 1-2 数据库系统

1. 提供高级的用户接口

数据是以一定的物理形式存储在数据库中的。如果让用户直接访问这种物理形式的数据，势必要求用户了解许多实现的细节，例如数据的物理地址、物理块的结构、指针以及用于控制的各种标志和附加信息等。而且，应用程序将依赖于数据的物理结构，有损于数据的独立性。DBMS 应该隐蔽掉这些实现细节，使用户看到的数据和他们平常所用的数据一样。这种抽象掉物理存储细节的数据称为数据的逻辑形式。通过 DBMS，数据的物理形式和逻辑形式应能互相映射。

对数据的访问，最终要落实到数据的物理形式，这涉及到访问路径和访问策略的选择。如果让用户介入这个过程，则数据库的使用将十分繁琐。现代 DBMS 都提供了非过程数据库语言(non-procedural database language)，用户只须提出要什么数据，而如何获得这些数据则不必关心，全由 DBMS 负责。

通过 DBMS，用户看到的数据是其逻辑形式，操纵数据库的语言是非过程数据库语言。

这两者构成了 DBMS 与用户的最基本的界面。为了适应不同用户的需要，DBMS 常提供各种接口。近年来又普遍地增加了图形接口，不少 DBMS 的接口软件的规模往往超过 DBMS 的核心软件。

2. 查询处理和优化

这里的查询(query)泛指用户对数据库所提的访问请求，不但包括数据检索，也包含修改数据和定义新的数据等请求。由于用户用非过程数据库语言查询数据库，查询处理过程的拟订和优化就由 DBMS 来完成。查询处理和优化是 DBMS 的基本任务，对 DBMS 的性能影响颇大。

3. 数据目录管理

在一般的程序设计语言中，数据的定义是程序的一部分。程序运行完毕，数据定义也就失效了。数据库中保留的是持久和共享的数据。数据的定义应该独立于应用程序，长期保留在数据库中，这就构成了数据目录。数据目录不但有数据的逻辑属性，而且还包含数据的存储结构的定义以及其它一切访问、管理数据所必需的信息。DBMS 在完成各种功能时都离不开数据目录，数据目录管理是 DBMS 的基本功能。

4. 并发控制

现代 DBMS 一般允许多个用户并发地访问数据库，这就不可避免地会发生冲突，例如对同一数据，一个用户要读，另一个用户要写，如果并发执行，则可能产生不希望的结果。为了解决冲突，DBMS 中需要有并发控制机制。

5. 恢复功能

数据库不但在正常运行时要保持一致性，在故障情况下，甚至数据库本身遭到破坏时，也要能恢复到一致状态。现代数据库一般都具有这样的恢复功能。

6. 完整性约束检查

数据都要遵守一定的约束，即使在程序设计语言中，也是如此。最简单的约束就是数据类型说明。通过数据类型检查，可以发现某些程序错误。由于数据库中的数据是持久和共享的，其正确性对一个单位十分重要，在保证数据的正确性方面要采取更多的措施。在 DBMS 中，不但要对数据进行语法检查，还要对数据进行语义检查。数据在语义上的约束称为完整性约束，例如，学生所选的课应该是开出的课程，每个学生的学号应该是唯一的，年龄不能是负数或大于 200 等，都是完整性约束。现代 DBMS 都程度不等地支持完整性约束检查功能。

7. 访问控制

数据库是共享资源，但不是任何用户都可以不受限制地访问任何数据。DBMS 应有控制用户访问权限的功能，这种功能叫访问控制。访问控制不但可以限制用户的访问范围，而且还可以限制用户可进行的操作。例如，对于学生成绩这一数据，访问控制可以做如下的限制：允许所有教务人员读，但只允许其中的成绩登录人员写。

DBMS 实现数据库系统的各种功能，也是数据库技术的凝聚点。选择 DBMS 是设计数据库系统的关键一步。但是，有了 DBMS 并不等于建立了数据库系统，还要设计和装入数据，开发以数据库为基础的应用程序，进行数据库系统的调整、维护和管理等。一般来说，在数据库领域有两方面的工作要做：一是实现 DBMS，这是系统软件研制者的任务；二是用 DBMS 构成数据库系统，这是数据库应用者的任务。本书在预备篇后，将分为两大部分：

是系统篇，介绍 DBMS；二是应用技术篇，介绍应用 DBMS 设计、建立、管理和维护数据库的技术。

1.3 数据、数据模型和数据模式

1.3.1 数据

为了了解世界、研究世界和交流情况，人们需要描述各种事物。用自然语言来描述虽然很直接，但事事用它来描述未免太繁琐，也不便于形式化。为此，人们常常只抽取那些感兴趣的事物的特征或属性，作为事物的描述。例如，一个大学生可用如下的记录来描述：(王彤，9093135，女，1975，江苏，计算机系，1993)。这样的记录，一般人可能不解其意，但是知道这个记录含义的人，可以从中得知王彤是位女大学生，学号为 9093135，1975 年出生，江苏人，1993 年考入计算机系。这种对事物描述的符号记录称为数据。数据有一定的格式，例如在上例中，姓名栏最多允许四个汉字，性别栏允许一个汉字，……，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。人们通过解释、推论、归纳、分析、综合等方法，从数据所获得的有意义的内容称为信息。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。

1.3.2 数据模型

数据模型是用来描述数据的一组概念和定义。一般来说，数据的描述包括两个方面。

(1) 数据的静态特性

它包括数据的基本结构、数据间的联系和数据中的约束。

(2) 数据的动态特性

它指定义在数据上的操作。

若以大家所熟悉的文件系统为例，它所用的数据模型包含文件(file)、记录(record)、和字段(field)等概念。对每个字段可以定义数据类型和长度，作为其约束，有打开、关闭、读、写等文件操作。这是一个简单的数据模型，没有描述数据间联系的手段。也有个别数据模型，例如 2.5 节将要介绍的 E-R 数据模型，只有描述数据静态特性的手段，还缺少操作的定义。

数据模型的好坏很难抽象地评价，这取决于它的用途。对人来说，总是希望数据模型能够尽可能自然地反映现实世界和接近人对现实世界的观察和理解，也就是数据模型要面向现实世界，面向用户。但是，数据模型又是实现 DBMS 的基础，它对系统的复杂性、性能影响颇大。从实现的角度来看，又希望数据模型接近数据在计算机中的物理表示，以便于实现、减少开销，也就是数据模型还不得不在一定程度上面向实现、面向计算机。这两方面的要求显然是矛盾的。数据库中解决这个矛盾的途径有点类似于程序设计语言。在程序设计语言中，有面向用户的高级程序设计语言，也有面向计算机的汇编语言，有时还有介于两者之间的中间语言，各有各的用途，面由编译系统完成高级到低级程序设计语言的转换。在数据库中，也是针对不同的使用对象和应用目的，采用多级数据模型，一般可分为下面三级。

1. 概念数据模型(conceptual data model)

DBMS 至少向用户提供一种数据模型，例如目前用得最多的关系模型。这类数据模型有诸多规定和限制，不便于非计算机专业人员的理解和应用。数据库是对一个单位(enterprise，泛指公司、机关、部门、工厂、车间、医院、学校等)的模拟，它的设计必须得到单位中广大工作人员的合作和参与。显然，一开始就用 DBMS 所提供的数据模型来设计数据库是不合适的。因为这既不便于非计算机专业人员的理解和参与，同时也会使数据库设计人员一开始就纠缠于实现的细节，不符合自顶向下、逐步求精的软件设计原则。概念数据模型是面向用户、面向现实世界的数据模型，是与 DBMS 无关的。它主要用来描述一个单位的概念化结构。采用概念数据模型，数据库设计人员可以在设计的开始阶段，把主要精力用于了解和描述现实世界上，而把涉及 DBMS 的一些技术性的问题推迟到设计阶段去考虑。可供选择的概念数据模型将在第二章介绍。

2. 逻辑数据模型(logical data model)

逻辑数据模型是用户从数据库所看到的数据模型。它与 DBMS 有关，DBMS 常以其所用的逻辑数据模型来分类。关系数据模型是目前最常用的逻辑数据模型。在关系数据模型以前，层次、网状模型曾经是主要的逻辑数据模型，现在仍有这种 DBMS 在运行。个别 DBMS 提供多种逻辑数据模型，例如既支持网状，又支持关系数据模型。用概念数据模型表示的数据必须转化为逻辑数据模型表示的数据，才能在 DBMS 中实现。逻辑数据模型既要面向用户，也要面向实现。目前流行的逻辑数据模型是现阶段技术条件下的折衷。随着计算机处理能力的提高和价格的下降，逻辑数据模型将更多地面向用户。

3. 物理数据模型(physical data model)

逻辑数据模型只反映数据的逻辑结构，例如文件、记录、字段等，而不反映数据的存储结构，例如物理块、指针、索引等。反映数据存储结构的数据模型称为物理数据模型。数据库的数据终须存储到介质上，每种逻辑数据模型在实现时，都有其对应的物理数据模型。物理数据模型不但与 DBMS 有关，而且还与操作系统和硬件有关。有关物理数据模型的内容将在第五章介绍。

概念数据模型只用于数据库的设计，逻辑数据模型和物理数据模型用于 DBMS 的实现。

1.3.3 数据模式

在程序设计语言中，一个数据有型(type)和值(value)之分。型是该数据所属数据类型的说明，而值是型的一个实例(instance 或 occurrence)，例如整数是型，而 93 是其一个值。用数据模型描述数据时，也有型、值之分。对某一类数据的结构、联系和约束的描述是型的描述，型的描述称为数据模式(data schema)。在同一数据模式下，可以有很多的值，即实例。例如，学生记录可以定义为图 1-3(*a*)的形式，这是数据模式。而图 1-3(*b*)是其一个实例。

数据模型和数据模式不应混淆，正像不应把程序设计语言和用程序设计语言所写的一段程序混为一谈一样。数据模型是描述数据的手段，而数据模式是用给定数据模型对具体数据的描述。

数据模式是相对稳定的，而实例是相对变动的。数据模式反映一个单位的各种事物的结构、属性、联系和约束，实质上是用数据模型对一个单位的模拟。而实例反映数据库的某

姓 名	学 号	性别	出生年份	籍 贯	系 别	入学时间
王 彤	9093135	女	1975	江苏	计算机	1992

(a)

(b)

图 1-3 数据模式及其实例

一时刻的状态，也就是这一单位在此时的状态。

在 DBMS 中，由于数据用多级数据模型来描述，相应地也有多级数据模式。美国国家标准协会(ANSI)的 ANSI/X3/SPARC 报告把数据模式分为三级(见图 1-4)。

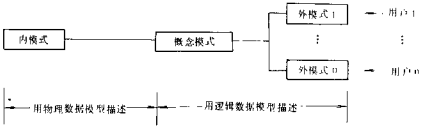


图 1-4 数据模式的分级

1. 概念模式(conceptual schema)

概念模式是用逻辑数据模型对一个单位的数据的描述。概念模式的设计是数据库设计的最基本任务。概念模式也称为逻辑模式(logical schema)。

2. 外模式(external schema)

外模式是用逻辑数据模型对用户所用到的那部分数据的描述。每个用户所感兴趣的数据不完全一样。另外，从保密的观点出发，也不宜让用户接触与己无关的数据。因此，每个用户的外模式不一定相同。外模式是概念模式的一部分或是从概念模式推导而来的。有了概念模式，设计外模式就比较省事了。

3. 内模式(internal schema)

内模式是用物理数据模型对数据的描述。概念模式与内模式之间可以互相映射，这个映射由 DBMS 来完成。内模式对一般用户是透明的，但是它的设计直接影响数据库的性能。因此，不但数据库的设计者和维护者应对内模式有充分的了解，数据库的用户最好对内模式也要有所了解，以便更好地使用数据库。

概念模式、外模式和内模式都存于数据目录中，是数据目录的最基本的内容。DBMS 通过数据目录管理和访问数据模式。

1.4 数据库应用

数据库应用就是选择合适的 DBMS，设计、建立、维护和管理数据库系统，为用户服务。与软件一样，数据库也有一个生存周期，它包含下列阶段。

1. 数据库系统的规划

它包括系统的应用范围和功能的确认、应用环境的分析、DBMS 及其支撑环境的选择

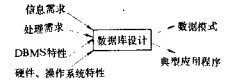


图 1-5 数据库设计的基本任务

和配置、人员的配备和培训，以及投资估算和效益分析等活动。

2. 数据库设计

数据库设计的基本任务如图 1-5 所示。

数据库设计实际上主要是数据的表示方法和存储结构的设计。在采用数据库技术以前，这些工作分散在应用程序中进行；采用数据库技术以后，可以通过数据库设计集中起来系统地进行。

3. 数据库的建立

首先根据数据库设计的结果，定义数据模式，规定访问权限，设置完整性约束。数据模式仅仅是个空的框架，要建立数据库还必须加载大量的数据。这是一项浩繁的工作，从数据的准备和正确性校验一直到数据的录入，需要大量的人力。这也是建立数据库的一项大的投资。对新建立的数据库还必须进行必要的测试和调整，以保证其符合设计的要求。此外，为了帮助用户熟悉新建立的数据库，还须编写用户指导书和进行人员培训。

4. 数据库的运行、管理和维护

数据库投入运行后，还必须监视其性能，听取用户的反映，必要时对数据库做相应的调整。数据库运行一段时间后，由于数据的增删，数据库的结构会变坏，废数据会增多，性能会下降，需要重组(reorganization)。为了在发生故障时能够恢复，需要定期地复制数据库的后备复本(backup)。此外，还要贯彻数据库的标准和执行各项规章制度，对用户提供咨询和服务。

5. 数据库的扩充和重构

一个单位的组成、结构和功能是会变化的，其对应的数据模式也须作相应的改变。因此，数据库经过一段时期运行后，扩充和重构(restructuring)是不可避免的。数据库的扩充和重构在某种意义上相当于数据库的一次重新设计，当然工作量要小得多。

习　题

1. 试比较文件系统和数据库系统，并指出其主要区别。
2. 何谓数据独立性？试说明其重要性。
3. 试区别下列名词：

(1) 数据模型与数据模式；

(2) 概念数据模型与概念数据模式；

(3) 数据库系统与数据库管理系统。

第二章　数据模型

2.1　层次数据模型

2.1.1　基本概念和结构

在现实世界中，有很多事物是按层次(hierarchy)组织起来的，例如，一个学校有若干系，一个系有若干班，一个班有若干学生。其他如动植物的分类、图书的编号、机关的组织，……，都是层次型的。层次数据模型的提出，首先是为了模拟这种按层次组织起来的事物。下面是层次数据模型的基本概念和结构。

1. 记录(record)和字段(field)

记录是用来描述某个事物或事物间关系的命名的数据单位，也是存储的数据单位。它包含若干字段。每个字段也是命名的，字段只能是简单的数据类型，例如整数、实数、字符串等。图 2-1 是一个名为系的记录，其中有四个字段：系名(字符串)、系号(字符串)、系主任名(字符串)、地点(字符串)。这是记录的型的定义，也就是记录的数据模式。图 2-2 是其一个实例。

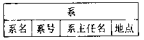


图 2-1　记录的型

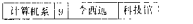


图 2-2　记录的一个实例

2. 双亲子女关系(parent-child relationship，简称 PCR)

这是层次数据模型中最基本的数据关系。它代表两个记录型之间一对多(记为 1：N)关系。例如一个系有多个班，就构成了如图 2-3 所示的双亲子女关系，在"1"方的记录型称为双亲记录，在"N"方的记录型称为子女记录。这是 PCR 的型，图 2-1 是其一个实例。

图 2-3　PCR 型

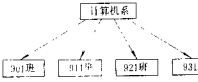


图 2-4　一个 PCR 实例

3. 层次数据模式(hierarchical data schema)

利用 PCR 可以构成层次数据模式，图 2-5 是一个层次数据模式的例子，每个方框代表一个记录型，每条弧线代表一个 PCR 型。层次数据模式应是一棵树。因此，在一个层次数据模式

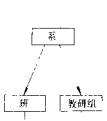


图 2-5　层次数据模式例子

中，除根以外，所有的记录型都应有唯一的双亲，但可以有多个子女。一个层次数据模式可有多个实例，这些实例组成一个森林。图 2-6 是图 2-5 数据模式的一个实例。在层次数据模式和它的实例中，一个双亲可能有多个子女。层次数据模型规定：子女按从左到右的次序排序。

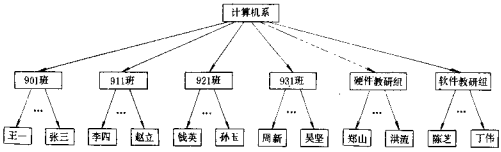


图 2-6　层次数据模式的一个实例

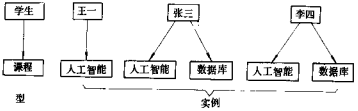
4. 虚拟记录(virtual record)

在现实世界中，很多数据不是层次型的。层次数据模型必须推广到模拟非层次型的数据，才有普遍意义。下面是几种非层次型的关系。

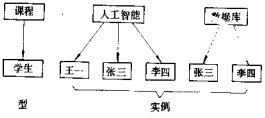
(1) 多对多的关系(记为 M：N 关系)

学生和课程是一个 M：N 关系，一个学生可以选多门课，一门课可有多个学生选修(见图 2-7)。图 2-7 的 M：N 关系可以用图 2-8(*a*)或图 2-8(*b*)的 PCR 表示，由于层次数据模型不允许一个记录有多个双亲，如果出现这种情况则必须为每个双亲复制一份记录。这种表示方法会引起大量的数据冗余。

图 2-7　M：N 关系



(*a*)　用 PCR 表示图 2-7 中的 M：N 关系



(*b*)　图 2-7 中的 M：N 关系的另一种表示法

图 2-8　用 PCR 表示 M：N 关系

(2) 一个记录型是两个以上的 PCR 子女

学生这个记录型可以是班记录型和运动队记录型的子女，如图 2-9 所示。图 2-9 的关系可以表示成两个 PCR 型，见图 2-10。为了遵守一个记录不得有多个双亲的原则，在两个 PCR 中都出现的学生记录必须复制，这同样会引起数据冗余问题。

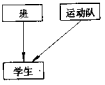


图 2-9 一个记录型有多个双亲

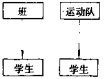


图 2-10 用 PCR 表示图 2-9 的关系

(3) 多元关系(n-ary relationship)

PCR 只代表两个记录型的 1∶N 关系。在现实世界中，往往有多个记录型参与的关系，例如供应商、零件和工程项目三个记录型构成一个三元的供应关系，如图 2-11 所示。为了用 PCR 表示这种三元关系，可增加一个名叫供应关系的记录型，它有一个字段，即供应量。图 2-12 是用 PCR 表示的三元关系，零件记录是根，可以一对多，没有冗余问题。但是，供应商和工程项目两种记录，凡是在供应关系中出现一次，都要有一个复本，将会导致较大的冗余。

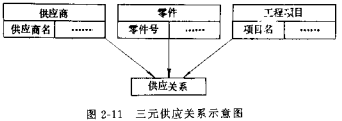


图 2-11 三元供应关系示意图

由上可知，用 PCR 表示上述三种关系都带来数据冗余。数据冗余不但浪费存储空间，而且会导致数据的不一致。为了避免数据冗余，如一个记录要在多处引用，则可在数据库中只存储一份这样的记录，其他引用的地方用其指针代替。这种用指针替代的记录称为虚拟记录。虚拟记录用下标 V 表示，指针用虚线箭头表示。图 2-8 的 M∶N 关系可以表示为图 2-13。因为在层次数据模型中，都是从根查起，为了既可以从课程查选课的学生，也可以从学生查其所选的课程，在图 2-13 中用了两个 PCR 型。请读者自行画出图 2-13 的实例图。图 2-14 是用虚拟记录表示的图 2-10 中的关系；图 2-15 是用虚拟记录表示的图 2-12 中的三元关系，由于采用了虚拟记录，消除了数据的冗余，但大量的指针会增加数据库的开销，数据模式也不够清晰、直观。

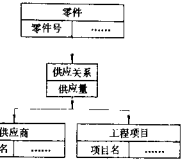


图 2-12 用 PCR 表示的图 2-11 的三元关系

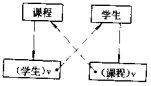


图 2-13　用虚拟记录表示 M：N 关系

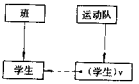


图 2-14　用虚拟记录表示的图 2-10 中的关系

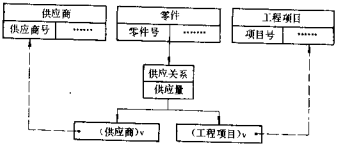


图 2-15　用虚拟记录表示的图 2-12 的三元关系

5. 层次数据的线性表示

由于存储器是线性的，层次数据必须变换成线性形式才能存储。层次数据模型规定用树的先序遍历(preorder traversal)的次序作为存储次序，即先遍历根，再从左至右遍历其子女。这样所生成的序列称为层次序列(hierarchical sequence)。图 2-6 可用层次序列表示为图 2-16。在层次序列中，不同类型的记录混在一起；为了区分起见，每个记录要加一个类型标志。这些类型标志仅仅出现在物理模型中，在逻辑模型中可隐去这样的标志。

计算机系	901班	王二	…	张三	911班	…	931班	…	吴坚	硬件教研组	郑山	…	洪流	软件教研组	…	丁伟

图 2-16　图 2-6 的层次序列

2.1.2　约束

层次数据模型的约束因 DBMS 而异，有些语义上的约束由应用程序来检查。下面是一些与层次数据模型直接有关的约束。

(1) 除了根记录以外，任何其他记录不能离开其双亲记录而孤立地存在。

这条约束有下列含义：在插入一个子女记录时，必须与一个双亲记录相联系，否则不能插入；在删除一个记录时，其子女记录也自动地被删除。

(2) 任何记录，不管其"虚实"，只允许有一个双亲记录，即层次数据模式及其实例总是树形。

(3) 虚拟记录的指针必须指向一个实际存在的记录。有虚拟记录指向的记录不得删除。

(4) 虚拟记录不得为根记录。

2.1.3 操作

在层次 DBMS 中，有一系列数据库操作的定义，例如数据模式的定义和修改，以及数据的查询和增、删、改等。下面以查找操作为例，说明层次数据模型的操作。在层次数据模型中，要查找一个记录，须从根记录开始，按给定条件沿一个层次路径(hierarchical path)，查找所需要的记录。下面介绍几个查找操作命令。

1. Get Unique（记为 GU）

按给定条件，沿层次路径，查找所需的记录。以图 2-6 为例，下面是作用在它上面的一个 GU 命令：

```
GU 系(系名='计算机系'),班(班名='911 班'),学生;
```

GU 后面是一个层次路径，系、班、学生是记录型，括号里是查询条件。学生后面未加条件，表示查找满足条件的第一个记录，即名叫'李四'的学生记录。在执行此 GU 命令后，李四这个记录就变成当前记录。

2. Get Next Within Parent(记为 GNP)

在当前记录的双亲下，按层次序列查找下一个记录。下面是个运用 GNP 命令的例子。

例 2-1

```
GU 系(系名='计算机系'),班(班名='911 班'),学生;
                                    /* 找到李四的记录 */
while not fail do GNP 学生;
                                    /* 找出当前记录李四的双亲 911 班的所有学生记录 */
```

执行上述语句的结果将是 911 班所有学生的记录。

3. Get Next(记为 GN)

按照层次序列，不受同一双亲的限制，查找当前记录的下一个记录。例如查找 911 班和 921 班的全部学生记录可用例 2-2 的语句。

例 2-2

```
GU 系(系名='计算机系'),班(班名='911 班'),学生;
while not fail do GNP 学生;          /* 找出 911 班所有学生记录 */
GN 学生;                             /* 找到 921 班的第一个学生记录，即我要 */
while not fail do GNP 学生;          /* 找出 921 班所有学生记录 */
```

从上可知，在使用层次数据模型的操作命令时，使用者必须熟悉数据的层次结构，正像领航员一样，在树形的结构中航行。如果有虚拟记录，还要按指针寻找所需的记录。就操作命令而言，每次操作只能取一个记录；若要取多个记录，必须借助于 while 之类的控制语句。

层次 DBMS 曾在 60 年代末至 70 年代初流行过。其中最有代表性的当推 IBM 公司的 IMS，至今仍有单位在继续使用。对于层次数据，层次 DBMS 的效率是很高的；但对于非层次数据，由于使用虚拟记录，大量的指针会使效率下降。层次 DBMS 提供给用户的数据模型和数据库语言比较低级，数据独立性也很差，使用层次数据库是不够方便的。随着关系 DBMS 的出现和发展，层次 DBMS 已逐步退出历史舞台，现在已没有必要专门学习它，但是层次数据模型是数据库发展早期的数据模型之一，关系数据模型以及其他一些数据模型是在与这些数据模型比较中发展起来的。为了了解数据模型的发展和加深对关系数据模型的理解，

对本节的层次数据模型和下一节的网状数据模型有所了解是必要的。

2.2 网状数据模型

2.2.1 基本概念和结构

1. 记录和数据项(data items)

与层次数据模型类似，在网状数据模型中，也是以记录为数据的存储单位。记录包含若干数据项，数据项相当于字段。但是，与层次数据模型不同，数据项不一定是简单的数据类型，也可以是多值的和复合的数据。简单的多值数据项称为向量(vector)，例如一个人有多个电话号码，则其电话号码数据项是多值的，可用有序集合表示。复合的多值数据项称为重复组(repeating group)，例如一个人有多个地址，而每个地址是个复合数据，它由省名、市名、街道名及号码、邮政编码等项组成。记录也有型、值之分。每个记录有一个唯一地标识它的数据库码(database key，简称 DBK)，DBK 可以看成记录的逻辑地址，可作记录的替身，或用来寻找记录。

2. 系(set)

在网状数据模型中，数据间的联系用系表示。系代表两个记录型之间的 1∶N 联系。系是命名的，也有型、值之分。例如，一个班有若干学生，可用名叫"班级-学生"的系表示它们之间的联系。系可用一条弧线表示(见图 2-17)，箭头指向"N"方。"1"方的记录称首记录(owner)，"N"方的记录称为属记录(member)。属记录值不一定要求属于同一记录类型，例如一个银行帐户可以作为首记录，而属记录是这个帐户的一笔笔的帐。这些帐可以是存款的、提款的、转帐的等等，它们分属于不同的记录型。如果一个系有多种类型的属记录，则称为多属系(multimember set)。多属系也可以分为多个系，但是用多属系表示，有时查找比较方便。多属系可用图 2-18 的形式表示。

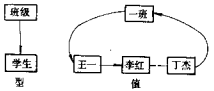


图 2-17 班级-学生系

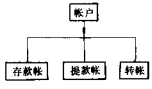


图 2-18 多属系

与层次数据模型不同，在网状数据模型中，一个记录型可以成为多个首记录型的属记录，也就是突破了数据模式必须为层次型的限制。因此，在模拟非层次数据时，网状数据模型不必采用增加记录复本或采用虚拟记录等别扭的办法。一个记录型可以作为几个系的首记录，也可以作为几个系的属记录，还可以既作为一个系的首记录，又可以作为另一个系的属记录。但一个记录型不能既作为一个系的首记录，又作为同一个系的属记录，即系不能直接用来表示一个记录型的自身的联系。但现实世界中存在这种记录型自身的联系，例如职工间的领导关系。网状数据模型为此引进了联系记录(linking 或 dummy record)的概念。联系记录用作自身联系的中间记录。图 2-19 表示职工间的领导关系，LINK 是联系记录型，EMP 是

职工记录型。在 EMP 与 LINK 之间定义了两个系 S_1 和 S_2。S_1 代表 1∶1 联系，即一个 EMP 记录可以用一个 LINK 记录做它的替身去联系其他 EMP 记录，以避免同一记录型的自身联系；S_2 是以 LINK 为首记录和以 EMP 为属记录的 1∶N 联系。通过 S_2 可以表示 LINK 所代表的 EMP 记录所领导的其他职工。图 2-19 (*b*)是其一个实例，用 E 和 L 分别代表 EMP 记录和 LINK 记录。从图中可以看出，E_4 领导 E_7，…，E_9；E_7 领导 E_8，E_5；E_9 领导 E_1，E_6。

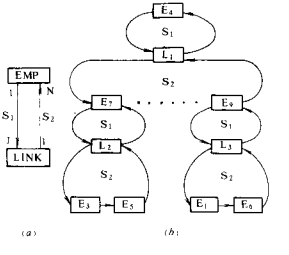


图 2-19　联系记录

一个记录值不能出现在同一系型的多个系值中，图 2-20 所示的情况是不允许的。图 2-20 表示学生(S)与课程(C)之间的联系，从语义上讲，一个学生可以选多门课，一门课可以被多个学生选，这是 M∶N 联系，在现实世界中是存在的，网状数据模型仍然借助于 LINK 记录表示 M∶N 联系。图 2-21 中每个 LINK 记录代表一个学生和一门课程的选课联系，S 与 LINK、以及 C 与 LINK 之间都是 1∶N 联系，可以定义 SL 和 CL 两个系。S、C 都是首记录，通过共同的属记录 LINK，表示 S、C 之间的 M∶N 联系。例如，S_1 与 S_2 都选 C_4，S_1 通过 L_1 与 C_4 联系，S_2 通过 L_7 与 C_4 联系，从而避免了同一记录出现在同型的不同系值中。LINK 记录也可以附有关于联系的数据项，例如图 2-21 中的 LINK 记录可以附有一个学生的某门课的成绩。LINK 记录还可以用于多元联系，图 2-22 表示图 2-11 的三元联系。LINK 实际上相当于供应关系，其中可附有供应量数据项。供应商、零件、工程项目分别与 LINK 之间定义三个系。通过公共的属记录 LINK 建立三者之间的三元联系。

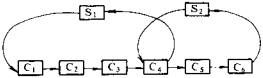


图 2-20　一个记录出现在同一系型的两个系值中

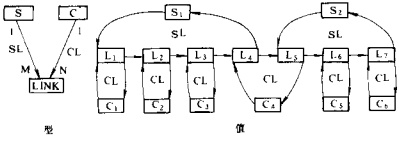


图 2-21　用联系记录表示 M∶N 联系

有一种特殊的系，叫做无首系或单值系(singular set)，这种系没有首记录。实际上这种系的首记录可看成是“系统”，可以省去，因此成为无首系。例如，一个单位(即“系统”)的所有部门可以组成一个无首系。由于“系统”只有一个，无首系也只有一个实例，因此无首系又