Visual C++ 数据库编程实战

韩存兵 编 著

科学出版社
www.sciencep.com

责任编辑：但明天
封面设计：王　翼

★本书特点★

全面剖析利用 Visual C++ 进行 Access、ODBC、SQL Server、MySQL、Oracle 等数据库应用系统的开发过程、技巧与方法，实例典型，方法实用

★本书适用于★

- 数据库应用系统开发人员
- 高校计算机专业师生
- 业余爱好者
- 专业程序员

需要本书或技术支持的读者，请与北京中关村083信箱（邮编100080）发行部联系，电话：010-62528991，62524940，62521921，62521724，82610344，82675588（总机），传真：010-62520573，E-mail：yanmc@bhp.com.cn

ISBN 7-03-012144-9

9 787030 121448 >

ISBN 7-03-012144-9
定价：45.00 元

数据库编程实战

韩存兵 编著

科学出版社
www.sciencep.com

内 容 简 介

这是一本讨论利用 Visual C++进行数据库应用系统开发的技术指导书。数据库应用是应用软件的根本，本书从数据库系统模型，全面讨论使用 Visual C++提供的多种数据库访问技术——ODBC API，MFC ODBC，ADO，DAO 以及 OLE DB 等进行 Access，ODBC，SQL Server，Oracle，MySQL 等数据库系统应用程序开发的基本技能与方法。

全书共分为 6 章，内容涉及数据库系统体系结构，Visual C++的 Access 访问技术、ODBC 访问技术、SQL Server 访问技术、MySQL 访问技术以及 Oracle 访问技术等知识。内容全面，讲解透彻，实例典型，方法实用，具有很强的技术指导性。

本书主要面向数据库系统应用开发人员，也可作为业余爱好者、专业程序员的参考书。

需要本书或技术支持的读者，请与北京中关村 083 信箱（邮编 100080）发行部联系，电话：010-62528991，62524940，62521921，62521724，82610344，82675588（总机） 传真：010-62520573 E-mail：yanmc@bhp.com.cn。

图书在版编目（CIP）数据

Visual C++ 数据库编程实战 / 韩存兵编著. —北京：科学出版社，2003.9

ISBN 7-03-012144-9

Ⅰ. V… Ⅱ. 韩… Ⅲ. C++语言—程序设计 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2003）第 077737 号

责任编辑：但明天 /责任校对：赵文博

责任印刷：媛明 /封面设计：王 翼

科学出版社 出版

北京东黄城根北街 16 号

邮政编码：100717

http://www.sciencep.com

北京市媛明印刷厂印刷

科学出版社发行 各地新华书店经销

*

2003 年 9 月第一版 开本：787×1092 1/16

2003 年 9 月第一次印刷 印张：30 1/2

印数：1—5 000 字数：691 000

定价：45.00 元

前 言

VC 是 Microsoft 公司开发的基于 C/C++的集成开发工具，是 Visual Studio 系列中功能最强大、代码效率最高的开发工具。用户可以使用 VC 以二种方式编写基于 Win32 的应用程序：一种是基于 Windows API 的 C 编程方式，另一种方式是基于 MFC 的 C++编程方式。C 编程方式是传统的、久经考验的编程方式，代码的效率较高，但开发难度和开发工作量较大。C++编程方式的代码运行效率相对较低，但开发难度小、开发工作量小、源代码效率高。

本书共分 6 章，主要介绍使用 Visual C++进行各类数据库访问以及编程的各个方面，基本涵盖了 VC 的主要方面，也体现了面向对象编程的思想。

第 1 章　数据库应用基础　介绍了有关数据库的结构模型、SQL 查询语言，以及利用 Visual C++进行数据库访问的接口技术。

第 2 章　Visual C++和 Access 访问技术　介绍如何创建 Access 数据库、MFC 与 DAO、用 DAO 访问数据库、DAO 和 ODBC，以及利用 Visual C++的 DAO 技术进行多线程编程处理等内容。

第 3 章　Visual C++和 SQL Server 访问技术　首先对 SQL Server 2000 的数据对象创建过程进行介绍，然后讨论 ADO 的数据库访问技术，最后介绍了在 Visual C++下如何利用 ADO 编程访问、操作 SQL Server 2000 数据库系统的方法。

第 4 章　Visual C++和 ODBC 访问技术　介绍了 MFC 的 ODBC 类、ODBC API 以及如何用 ODBC API 和 MFC ODBC 操作数据库。

第 5 章　Visual C++和 Oracle　首先对 Oracle 9i 大型数据库系统作介绍，然后讨论了 OCI 连接 Oracle 技术，最后实例探讨 Visual C++利用 OCI 操作 Oracle、封装 OCI 类等技术与方法。

第 6 章　Visual C++和 MySQL 访问技术　包括 MySQL 数据库创建、MyODBC 数据源的建立，以及 Visual C++如何利用 MySQL C API 进行 MySQL 数据库访问与操作的细节。

本书内容讲解循序渐进、结构严谨，对各个主题的讲解都与具体实例相结合。读者可以很快地掌握 VC 语言特点和编程技巧，并能快速、灵活地应用这些知识点。

本书由韩存兵组织编写，其他参加本书部分编写、录排、校对工作的人员还有：田野、龚昊远、田蕴哲、龚志翔、李红玲、白红利等，在此一并表示感谢。

由于编者经验有限，加之时间仓促，书中难免会有疏漏和不足之处，恳请专家和读者不吝赐教，批评指正。

作 者

2003 年 8 月

目　录

第 1 章　数据库应用基础

数据库技术是 20 世纪 60 年代发展起来的计算机技术。1968 年 9 月，美国 IBM 公司最早推出了信息管理系统（Information Management System）。此后的三十多年中，数据库技术的发展日新月异，对计算机信息处理，社会生产，乃至人们的社会生活都产生了巨大的推动作用。人们对数据库的使用也变得越来越广泛，而越来越多的人也迫切需要掌握数据库访问技术。本章主要介绍用 Visual C++访问数据库技术的基础知识。

主要内容：

- 数据库简介
- 关系数据库的基本概念，包括关系的定义、关系数据库的主要元素、关系数据库的规范化理论和 E-R 方法、关系数据库管理系统的十二条准则，以及流行的关系数据库管理系统等
- 结构化查询语言（SQL）简介
- Visual C++访问数据库的主要技术

1.1　数据库简介

1.1.1　数据库历史

数据库，就是数据的集合。但是，数据库中的数据并非相互独立地简单归结到一起，而是根据数据之间固有的关系分门别类地存储起来的。也就是说，数据库是存储在计算机系统内的有一定结构的数据的集合。数据是由数据库管理系统进行管理的。

为了访问数据库中的信息，已经开发出许多方法。回顾历史，有两个产品为组织信息提供了两种截然不同的数据模型：1968 年 IBM 发布的 IMS 和 20 世纪 70 年代 Cullinet Software 的 IDMS。IMS 提出了不同类型的记录通过层次结构相互联系的层次数据模型。例如，一个银行数据库系统可以把公司实体记录和诸如总部地址、电话号码这样的信息放在层次结构的顶部；接下来是银行的各个业务部门。在每个部门分支下，是该部门的出纳员和其他职员的记录。当要查询某个出纳员时，程序就会沿着各个分层导航。另一方面，在 CODASYL 委员会数据库任务组 1971 年发表的报告的基础上诞生的 IDMS，被称为网状模型。网状模型是层次模型的一个推广，某一级的一个记录集合在上一级中可能对应两个不同的包含层次。

当然，IMS 和 IDMS 还有许多特性。简单地说，层次模型把数据组织成一棵根在上、叶在下的有向树。网状模型把数据组织成无环有向图，使得网状模型更容易表达现实世界中的数据结构。这些产品的主要缺点是对数据的查询很难执行，一般需要熟悉复杂的数据导航结构的专业程序员编写相应程序。今天，仍然有相当多的公司在使用这两种数据库。IMS 仍然是 IBM 重要的利润来源。但是，这些使用中的 IMS 和 IDMS 已经是“遗产系统”了，而且，很难把这些系统转化成现代的数据模型。虽然某些公司用原有的系统已经足够，

但任何想安装新系统的公司都会选择一个支持更新的数据模型的数据库管理系统。

近20年来，数据库系统产品使用最广泛的数据模型是关系模型。关系模型使用灵活，即使用户不是程序员，也可以快捷轻易地写出查询语句。利用关系模型的数据库管理系统称为关系数据库管理系统（RDBMS）。最近，一种更新的数据模型——对象-关系模型在许多产品中正逐渐取代关系模型。利用对象-关系模型的数据库管理系统称为对象-关系数据库管理系统（ORDBMS）。因为对象-关系模型实际上是关系模型的扩展，ORDBMS也支持RDBMS中的数据。因此有些作者将这种产品作为RDBMS/ORDBMS类型，如果不愿区分它们，可笼统地称之为数据库管理系统。

1.1.2 数据库的主要作用

数据库能够为人类带来如下好处：

（1）数据共享。共享不仅指现有的应用程序可以共享数据库的数据，而且新的应用程序也能对这些数据进行操作。换句话说，不向数据库中添加任何新数据也可能满足新应用程序的数据要求。

（2）减少冗余。在非数据库系统中，每个应用程序都有自己的专用文件。这种情况经常导致在存储数据上有相当大的冗余，结果浪费存储空间。例如，一个有关人事的应用程序和一个有关教育的应用程序可能同时拥有包含职员的部门信息的文件。但是，这两个文件可以集成起来消除冗余，只要数据管理员意识到两个应用程序的数据要求；也就是说企业应有必要的全局控制。

（3）避免不一致（某种程度上）。这是前一点必然的结果。假定一种实际情况——雇员E3在部门D8工作——数据库中有两个不同的条目。还假定DBMS也没有意识到冗余的存在（也就是对冗余失控）。则必然会有两个记录不一致的情况：当其中一个更新时，另一个不变。这种情况称为数据库不一致。显然，处于不一致状态的数据库可能提供给用户错误的或矛盾的信息。

当然，如果指定事实是由一条记录表示（也就是如果排除了冗余），那么这样的不一致就不会发生。另一种选择是，冗余没有排除但是受到控制（被DBMS得知），那么数据库管理系统就可以保证对用户来说数据库总是一致的，DBMS确保两个记录中的任何一条改变会自动地应用到另一条，这一过程即为传播更新。

（4）提供事务支持。事务是一个逻辑工作单元，它包括一些数据库操作（特别是，一些更新操作）。常见的例子如从帐户A到帐户B转移一定的现金数。显然，这里要求两个更新操作：一个是从帐户A提出现金；另一个是把现金数存入帐户B。如果用户已经说明两个更新是同一事务的一部分，那么系统要确保两个操作要么都做，要么都不做；即使在系统执行过程中出现故障（比如因为电源断）也应如此。

这种情况说明的事务的原子性不是事务支持的惟一优点，但是与其他一些优点不同，它甚至可以应用到单用户的情况。

（5）保持完整性。完整性的问题是确保数据库中的数据是正确的。同样事实的两条记录的不一致，就是缺少完整性的例子。当然，只要在存储的数据中有冗余，就会引起这样的问题。即使没有冗余，数据库也可能包含错误的信息。例如，可能显示雇员一周工作了400小时而不是40小时，或者属于一个不存在的部门。数据库的集中控制可以有效地避免

此类问题。做法是通过支持数据管理员定义一些完整性约束（也称为商业规则），由 DBA 加以实施，完整性约束在任何操作执行时都得到有效的检验。

值得指出的是，数据完整性在数据库中要比在各自独立的文件系统中重要得多，因为数据库中的数据是共享的。要是没有正确的控制，有可能一个用户错误地更新数据库而生成的错误数据，会殃及其他无辜的用户。目前，数据库厂商对数据库的完整性约束的支持还相当不够（尽管最近这一方面的情况有所改善）。这一事实很不幸，因为数据库完整性既基本又非常重要。

（6）增强安全性。数据库管理员可以确保访问数据库的惟一方式是通过正确的通道，因此可以定义安全性约束或规则。当试图访问敏感数据时，要检查这些安全性约束或规则。对于数据库的每条信息的不同类型的访问（修改、插入或删除等）可建立不同的约束。

没有这样的约束，数据的安全性可能比传统的文件系统更处于危险之中，也就是说，某种意义上数据库系统的集中性要求相称的、好的安全系统。

（7）平衡相互冲突的请求。数据库管理员了解企业的全局的需要，在他的指示下能建立系统的结构以提供对企业最佳的全局服务。例如，所选择的数据的物理表示应尽可能使重要的应用以最快的方式访问数据（可能会以降低其他某些应用的访问速度为代价）。

（8）加强标准化。数据库管理员对数据库集中控制，可以确保所有表示数据的可用标准都可以观察到。可用标准可包括下面的任意一种或全部：部门标准、安装标准、社团标准、工业标准、国家标准和国际标准。标准化的数据表示可以很有效地支持数据交换或者两个系统间的数据移动（随着分布式系统的出现，这一点越来越重要。同时，数据命名和文档标准也有效地支持了数据共享和易理解性。

以上列出的大多数优点都是比较显而易见的。但是，有一点则不然，那就是对数据的独立性的支持（严格地说，数据独立性是数据库系统的客观目标，而不仅仅是一个必要的优点）。数据独立性的概念十分重要。

1.1.3 三种数据模型

1. 层次性数据模型

（1）基本结构

层次模型用树形结构来表示各类实体以及实体间的联系。每个结点表示一个记录类型，结点之间的连线表示记录类型间的联系，这种联系只能是父子联系。每个记录类型可包含若干个字段，这里，记录类型描述的是实体，字段描述实体的属性。

任何一个给定的记录值只有按其路径查看时，才能显出它的全部意义，没有一个子女记录值能够脱离双亲记录值而独立存在。

其限制如下：

- 只有一个结点没有双亲结点，称之为根结点。
- 根以外的其它结点有且只有一个双亲结点。

这就使得层次数据库系统只能处理一对多的实体关系。

（2）多对多联系在层次模型中的表示

用层次模型表示多对多联系，必须首先将其分解成一对多联系。分解方法有两种：冗

余结点法和虚拟结点法。

（3）基本操作

层次数据模型的操纵主要有查询、插入、删除和更新。进行插入、删除、更新操作时要满足层次模型的完整性约束条件。

- 进行插入操作时，如果没有相应的双亲结点值就不能插入子女结点值。
- 进行删除操作时，如果删除双亲结点值，则相应的子女结点值也被同时删除。
- 进行更新操作时，应更新所有相应记录，以保证数据的一致性。

（4）存储结构

- 邻接法。按照层次树前序遍历的顺序把所有记录值依次邻接存放，即通过物理空间的位置相邻来实现层次顺序。
- 链接法。用指引元来反映数据之间的层次联系。

（5）优缺点

优点如下：

- 数据模型比较简单，操作简单。
- 对于实体间联系是固定的，且预先定义好的应用系统，性能较高。
- 提供良好的完整性支持。

缺点如下：

- 不适合于表示非层次性的联系。
- 对插入和删除操作的限制比较多。
- 查询子女结点必须通过双亲结点。
- 由于结构严密，层次命令趋于程序化。

2. 网状数据模型

（1）基本结构

网状数据模型是一种比层次模型更具普遍性的结构，它去掉了层次模型的两个限制，允许多个结点没有双亲结点，允许结点有多个双亲结点，此外它还允许两个结点之间有多种联系（称之为复合联系）。

（2）操作

网状数据模型的操纵主要包括查询、插入、删除和更新数据。

- 插入操作允许插入尚未确定双亲结点值的子女结点值。
- 删除操作允许只删除双亲结点值。
- 更新操作时只需更新指定记录即可。
- 查询操作可以有多种方法，可根据具体情况选用。

（3）存储结构

网状数据模型的存储结构依具体系统不同而不同，常用的方法是链接法，包括单向链接、双向链接、环状链接、向首链拉等。此外还有其它实现方法，如指引元阵列法、二进制阵列法、索引法等。

（4）优缺点

优点如下：

- 能够更为直接地描述现实世界。
- 具有良好的性能，存取效率较高。

缺点如下：

- 其 DDL 语言极其复杂。
- 据独立性较差。由于实体间的联系本质上通过存取路径指示的，因此应用程序在访问数据时要指定存取路径。

3. 关系数据模型

（1）基本结构

在用户看来，一个关系模型的逻辑结构是一张二维表，它由行和列组成。在关系模型中，实体以及实体间的联系都是用关系来表示。

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

（2）操作

关系数据模型的操纵主要包括查询、插入、删除和更新数据。这些操作必须满足关系的完整性约束条件。

关系模型中的数据操作是集合操作，操作对象和操作结果都是关系，即若干元组的集合。

关系模型把存取路径向用户隐蔽起来，用户只要指出“干什么”，不必详细说明“怎么干”，从而大大地提高了数据的独立性，提高了用户生产率。

（3）存储结构

关系数据模型中，实体及实体间的联系都用表来表示。在数据库的物理组织中，表以文件形式存储，每一个表通常对应一种文件结构。

（4）优缺点

优点有：

- 关系模型是建立在严格的数学概念的基础上的。
- 无论实体还是实体之间的联系都用关系来表示。对数据的检索结果也是关系（即表），因此概念单一，其数据结构简单、清晰。
- 关系模型的存取路径对用户透明，从而具有更高的数据独立性，更好的安全保密性，也简化了程序员和数据库开发者的工作。

缺点如下：

- 由于存取路径对用户透明，查询效率往往不如非关系数据模型。因此为了提高性能，必须对用户的查询请求进行优化，增加了开发数据库管理系统的负担。

1.1.4　数据库体系结构

ANSI/SPARC 体系结构分为 3 层：内模式、概念模式和外模式（见图 1-1）。

- 内模式（存储模式）是最接近物理存储的，即数据的物理存储方式。
- 外模式（用户模式）是最接近用户的，即用户所看到的数据视图。
- 概念模式（公共逻辑模式，有时称为逻辑模式）是介于前两者之间的间接层次。

请注意，外模式是单个用户的数据视图，而概念模式是一个部门或企业的数据视图。换句话说，“外部视图”（即外模式）会有许许多多，每一个都或多或少地抽象表示整个数据库的某一部分，而“概念视图”（概念模式）只有一个，它包含对现实世界数据库的抽象表示（大多数用户只对整个数据库的某一部分感兴趣）。同样，“内部视图”（即内模式）也只有一个，表示数据库的物理存储。

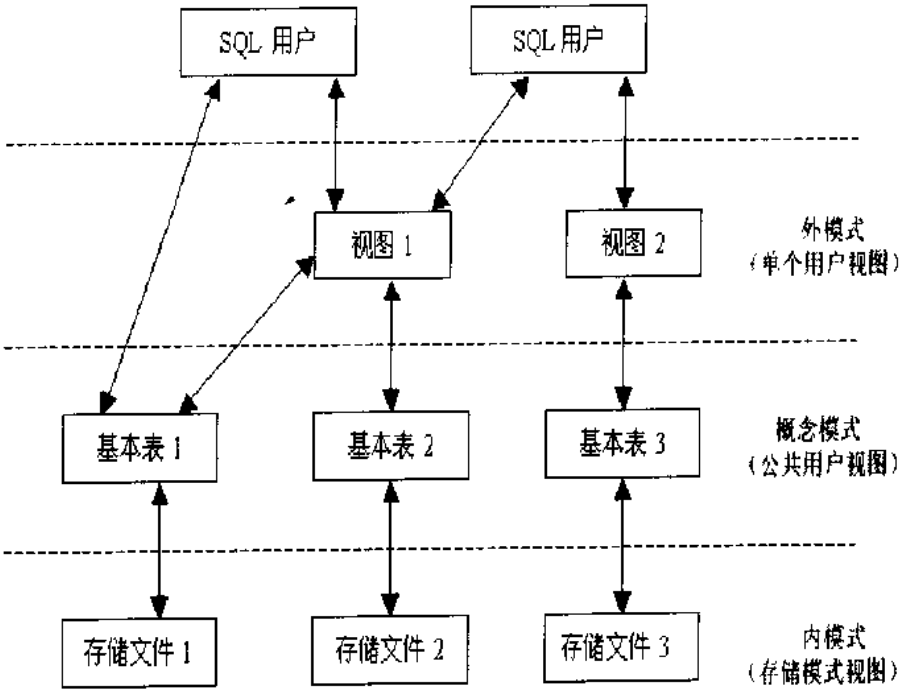

图 1-1 数据库体系结构

1. 外模式

外模式就是单个用户的数据视图。用户可以是应用程序员或某最终用户（这里 DBA 是一个特例，与其他用户不同，DBA 还要了解概念模式和内模式）。

每个用户都有自己使用的语言。对于应用程序员，可能会使用常规的编程语言（如 PL/I，C++，Java）或其他专用语言。这些专用语言通常被称作“第四代”语言（4GL），这是由于（a）机器语言、汇编语言和像 PL/I 这样的语言被作为第一、二、三代语言；（b）专用语言是对“第三代”语言（3GL）的发展，就像第三代语言对汇编语言的发展和汇编语言对机器语言的发展一样。

对于最终用户，其使用的语言或者是一种查询语言或是某特定目的的语言。这些语言可能是表格驱动的或菜单驱动的，可处理用户的请求并由在线应用程序来支持。

重要的是所有这些语言都包含数据子语言，即数据库对象和操作的整个语言的一个子集。数据子语言嵌入在相应的主语言中。主语言负责提供各种非数据库的功能，如局部变量、计算操作、逻辑分支，等等。一个指定系统可以支持多种主语言和多种数据库子语言；目前大多数系统支持的特定的数据子语言是 SQL 语言。多数系统允许 SQL 既可以作为独

立的交互查询语言，又可嵌入到诸如 PL/I 或 Java 等主语言中。

理论上，任何特定的数据子语言至少包含两个子语言：数据定义语言（DDL）和数据操纵语言（DML）。数据定义语言支持对数据库对象的定义或说明，数据操纵语言支持对这些对象的操作和处理。

每个外部视图都是通过外模式定义的，外模式包括外部视图中的各种外部记录类型的基本定义。外模式是使用用户数据子语言的 DDL 部分来写的（因此有时 DDL 也当作外部数据定义子语言）。例如，雇员外部记录类型就可以定义为 6 个字符的雇员号字段加 5 位十进制数的工资字段等等。此外，在外模式和其下面的概念模式之间要定义映像。

2. 概念模式

概念视图表示数据库的全部信息内容，其形式要比数据的物理存储方式抽象些。通常，它与任何特定用户观察数据的方式都不同。广义上讲，概念视图更接近于实际数据，而不像某一用户所看到的数据，这些数据受到特定语言或可能使用的硬件限制。

概念视图由许多概念记录类型的值构成。例如，它可能包括部门记录值的集合，雇员记录值的集合，供应商记录值的集合，零件记录值的集合等等。概念记录既不和外部记录相同，也不和存储记录相同。

概念视图是由概念模式定义的，概念模式包括各种概念记录型的定义。概念模式是用另一种数据定义语言来写的，即概念数据定义语言。如果可以实现物理记录的独立性，那么概念视图根本不涉及物理表示和访问的技术，它们只定义信息的内容。这样，在概念模式中不能涉及存储字段表示、存储记录队列、索引、哈希算法、指针或其他存储和访问的细节。如果概念视图以这种方式真正地实现数据独立性，那么根据这些概念模式定义的外模式也会有很强的独立性。

概念视图是整个数据库内容的视图，概念模式是该视图的定义。如果把概念模式只理解为类似 COBOL 程序中简单的记录定义一样的一组定义，那是不准确的。在概念模式中的定义应包括许多额外的特征，诸如安全性和完整性约束。到目前为止，有些权威人士认为概念模式的根本目的是描述整个企业的情况，而不只是数据本身，而且还包括数据的使用情况：数据在企业中的流动情况，在每一部门的用处，以及对它实行的审计和其他控制，等等。必须强调的是，目前的系统实际上还不能支持这种程度的概念模式。目前大多数系统支持的“概念模式”实际上只不过是把单个的外模式合并起来，再加上了一些安全性约束和完整性约束。但是将来的系统很可能在支持概念模式上会更加复杂。

3. 内模式

体系结构的第三层是内模式。内部视图是整个数据库的低层表示；它由许多内部记录型中每一类型的许多值组成。“内部记录”是 ANSI/SPARC 对存储记录的称谓。内部视图与物理层仍然不同，因为它并不涉及物理记录的形式（即物理块或页），也不考虑具体设备的柱面或磁道大小。换句话说，内部视图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，它未反映在体系结构中。块或页是输入/输出的单位，在一次输入/输出操作中，二级存储和主存之间传输的数据量。典型的页面大小是 1K、2K 或 4K 字节（1K = 1024byte）。

内部视图由内模式来描述，内模式不仅定义各种存储记录，而且也说明存在什么索引，存储记录怎么表示，存储记录是在什么物理队列中，等等。

内模式是用另一种数据定义语言——内部数据定义语言来写的。本书使用更直观的词“存储结构”或“存储的数据库”代替“内部视图”，用“存储结构定义”代替“内模式”。

最后指出一点，在某些特殊的情况下，应用程序（尤其是实用程序）可能会直接操作内模式而不是外模式。当然，这种做法肯定是不妥的，这会带来一定的安全性和完整性隐患（即安全性约束和完整性约束会被绕过），并且应用程序会失去数据独立性。但有时为了保证功能或性能上的要求，又不得不这样做。这就像使用高级语言系统的用户，偶尔会使用汇编语言以满足特定的功能或性能上的要求一样。

4. 映像

除了三级模式本身，数据库体系结构还定义映像关系，即概念模式/内模式间的映像和外模式/概念模式间的映像。一般地讲，概念模式/内模式的映象定义了概念视图和存储的数据库的对应关系，说明了概念记录和字段在内部层次怎样表示。如果数据库的存储结构改变了，即改变了存储结构的定义，那么概念模式/内模式的映象必须进行相应的改变，以便概念模式能够保持不变（当然，对这些变动的管理是数据库管理员的责任）。换句话说，为了保持数据的物理独立性，内模式变化所带来的影响必须与概念模式隔离开来。

外模式/概念模式间的映象定义了特定的外部视图和概念视图之间的对应关系。一般地讲，这两层之间存在的差异情况与概念视图和存储模式之间存在的差异情况是类似的。例如，字段可能有不同的数据类型，字段和记录名可以改变，几个概念字段能合成一个单一的字段，等等。可能同时存在多个外部视图，多个用户可共享一个特定的外部视图，不同的外部视图可能有交叉。

很显然，就像概念模式/内模式的映象是物理独立性的关键，外模式/概念模式的映像是逻辑独立性的关键。如果用户和用户的应用程序能相对于数据库物理结构的改变而保持不变，系统就提供了物理独立性。同样地，如果用户和用户的应用程序对于数据库逻辑结构的改变（即在概念或公共逻辑层的改变）能保持不变，系统就提供了逻辑独立性。

此外，多数系统允许定义以其他形式表示的某些外部视图（通过外模式/外模式间的映象），而不总是要求一个明确的到概念层的映射定义。这是一个很有用的特征，特别是当几个外部视图相互非常类似时。关系系统一般支持并提供这种能力。

1.1.5 关系数据库

基于关系数据模型（简称关系系统）的 DBMS 在数据库市场上已经占据了主导地位。而且，过去 30 年中大量主要的数据库研究是基于此模型的。不可否认，1969 ~ 1970 年间关系模型的建立，在整个数据库领域的历史中无疑是最重要的事件。由于这些原因，加上关系模型有坚实的逻辑和数学基础，使之因此成为数据库原理的主要教学内容，本书主要针对关系系统。

那么究竟什么是关系系统呢？很显然本书在此还不能给一个完满的解答，但是可以先给出一个大致的定义，之后再给出更详尽的答案。简言之，关系系统是指数据以表（而且只有表）的形式呈现给用户，且提供给用户的操作（如检索）以表为操作对象，即操作是

从旧表中生成新的表。例如，“选择”操作，是提取某指定的表的行子集，而“投影”操作是提取一个表的某些列的子集，表的行子集和列子集本身都可以看作一个表。

这样的系统之所以称为“关系的”，是因为表的数学用语为关系。可以这样区分关系系统和非关系系统，如前所述，关系系统的用户把数据看作表，而且只能是表。非关系系统的用户则把数据看作其他的数据结构，代替或者扩展关系系统中的表结构。这些结构需要相应的操作。例如，像 IBM 的 IMS 这样的层次系统，展现给用户的数据是树结构（层次）的集合的形式，对这些结构的操作符包括对遍历指针（表示整个层次路径的指针），这是与关系系统相区分的重要特征，关系系统没有这样的指针。

第一代关系产品出现于 20 世纪 70 年代末 80 年代初。在写本书之际，绝大多数数据库系统都是关系系统，它们可以在各种硬件和软件的平台上运行。最主要的产品包括：IBM 公司的 DB2，Computer Associates International 公司的 Ingres II，Informix Software 公司的 Informix Dynamic Server，微软公司的 Microsoft SQL Server，Oracle 公司的 Oracle 9i，以及 Sybase 公司的 Sybase Adaptive Server。本书中以后涉及到这些产品时，分别只提及缩写名字 DB2，Ingres，Informix，SQL Server，Oracle，Sybase 等。

如前所述，关系系统基于正规的关系基础或理论，即关系数据模型。直观地说，关系系统是这样的系统：

（1）结构化方面，数据库中的数据对用户来说是表，并且只是表。

（2）完整性方面，数据库中的这些表满足一定的完整性约束。

（3）操纵性方面，用户可以使用用于表操作的操作符。例如，为了检索数据，需要使用从一个表导出另一个表的操作符。其中，选择、投影和连接这三种尤为重要。

1. 关系的定义

可以用集合代数中的笛卡尔乘积方法来定义关系：

笛卡尔乘积：给定一组集合 D_1，D_2，$D_3 \cdots D_n$，这 n 个集合的笛卡尔乘积 $D_1 \times D_2 \times D_3 \cdots \times D_n$ 是所有可能的有序元组$<d_1, d_2, d_3 \cdots, d_n>$的集合，其中 $d_1 \in D_1$，$d_2 \in D_2 \cdots$，$d_n \in D_n$。

一个元组是组成该元组的所有分量的有序集合。

举例来说，假设有两个集合：员工={张三，李四，王五，赵六}，职位={管理，开发，质保，营销}，那么，员工、职位的笛卡尔乘积就是：

员工×职位 = {（张三，管理），（张三，开发），（张三，质保），（张三，营销），
（李四，管理），（李四，开发），（李四，质保），（李四，营销），
（王五，管理），（王五，开发），（王五，质保），（王五，营销）
（赵六，管理），（赵六，开发），（赵六，质保），（赵六，营销）}

上例中，员工的基数是 4，职位的基数也是 4，那么，笛卡尔乘积“员工×职位”的基数是 4×4=16。

关系的定义：对于给定的一组集合 D_1，D_2，$D_3 \cdots D_n$，如果 R 是该组集合的笛卡尔乘积 $D_1 \times D_2 \times D_3 \cdots \times D_n$ 的一个子集，那么称 R 是集合 D_1，D_2，$D_3 \cdots D_n$ 上的一个关系，或者称为表。关系都有关系名。

关系就是笛卡尔乘积的子集，因此，关系实际上就是一个二维表。二维表的表名就是关系名。二维表的每一列叫做属性（Attribute）。n 个关系就有 n 个属性。一个关系中的每

一个属性都有属性名，各个属性的属性名都不相同。一个属性的取值范围叫做该属性的域（Domain）。不同属性可以有相同的域。二维表的每一行值就是一个元组。由于二维表的每一列都有自己的属性名，所以，列与列之间的次序可以相互交换。

关系数据库是以表的集合的形式存储和展示数据的。通过建立表与表之间的关系，就定义了这种数据库类型的逻辑结构。

关系数据库模式的优点在于：

- 将数据组织成表的集合，使数据库设计易于理解。
- 为数据定义、检索与更新提供了相对完整的语言。
- 提供了数据完整性原则，用来定义具有一致性的数据库状态，从而提高了数据的可靠性。

2. 关系数据库的元素

关系数据库是以表的集合的形式来展现数据的。例如，Microsoft Access 的数据库范例中包含某个公司的业务信息。该数据库中有一个表列出了一系列的供应商信息，请参见图 1-2。

供应商：表

	供应商ID	公司名称	联系人姓名	联系人头衔	地址	城市	地区	邮政编码
▶	1	佳佳乐	陈小姐	采购经理	西直门大街 110 号	北京	华北	100023
	2	康富食品	黄小姐	订购主管	幸福大街 290 号	北京	华北	170117
	3	妙生	胡先生	销售代表	南京路 23 号	上海	华东	248104
	4	为全	王先生	市场经理	永定路 342 号	北京	华北	100045
	5	日正	李先生	出口主管	体育场东街 34 号	北京	华北	133007
	6	德昌	刘先生	市场代表	学院北路 67 号	北京	华北	100545
	7	正一	方先生	市场经理	高邮路 115 号	上海	华东	203058
	8	康堡	刘先生	销售代表	西城区灵镜胡同 310 号	北京	华北	100872
	9	菊花	谢小姐	销售代理	青年路 90 号	沈阳	东北	534567
	10	金美	王先生	市场经理	玉泉路 12 号	北京	华北	105442
	11	小当	徐先生	销售经理	新华路 78 号	天津	华北	307853
	12	义美	李先生	国际市场经理	石景山路 51 号	北京	华北	160439
	13	东海	林小姐	外国市场协调员	北辰路 112 号	北京	华北	127478
	14	福满多	林小姐	销售代表	前进路 234 号	福州	华南	848100
	15	德级	镇小姐	市场经理	东直门大街 500 号	北京	华北	101320

记录：1　共有记录数：29

图 1-2　Access 数据库中的供应商表

表是相关信息的逻辑归类。表由行与列组成。行（也称记录）包含了表中某一个条目的信息。例如，在上面的供应商表中，一行就代表某一家供应商的信息。

列（也称字段）组成行，一行中包含多个列，每一列都表示该行的某一项信息。例如，上面的供应商表中，有供应商 ID、公司名称、联系人姓名、联系人头衔、公司地址等列。

行跟列不同，每一行没有明确的命名。为了惟一地表示表中的各行，必须设立表的主关键字。主关键字可以是某一列，也可以是几列的组合，在这个表中，它必须具有对各行的惟一值。例如，上例中的供应商 ID 列就是主关键字。

此外，还可以对表定义外关键字，用以指定各个表之间的关系。外关键字指向相关表中的主关键字字段。

3. DBMS 的 12 条准则

1985 年，关系模型奠基人 E.F.Codd 提出了判断数据库管理系统是否为关系型的 12 条