Visual C++ .NET
数据访问开发技术

李博轩 等编著

Visual C++ .NET 开发技术丛书

内容简介

本书通过大量实例深入浅出地介绍了 Visual C++.NET 数据访问开发技术。全书共 9 章，主要内容包括：数据访问的基础知识，MFC ODBC，ODBC API，OLE DB 消费者/提供者，ADO 和 RDO，以及 MFC DAO 等编程技术。对于每种编程技术，本书都给出了具有代表性的应用实例，使读者能够通过实例的学习，迅速掌握数据库的编程技术。

本书适合需要利用 Visual C++.NET 进行程序设计的专业或非专业开发人员参考使用，这对透彻理解 Visual C++.NET 数据访问技术有很大的帮助。如果在学习的同时能够结合实际的开发或使用，效果将更好。

图书在版编目(CIP)数据

Visual C++.NET 数据访问开发技术/李博轩等编著.—北京：国防工业出版社，2002.10
(Visual C++.NET 开发技术丛书)
ISBN 7-118-02893-2

Ⅰ.V… Ⅱ.李… Ⅲ.C 语言－程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2002)第 047892 号

国防工业出版社出版发行
(北京市海淀区紫竹院南路 23 号)
(邮政编码 100044)
新艺印刷厂印刷
新华书店经售
*
开本 787×1092 1/16 印张 26¾ 612 千字
2002 年 10 月第 1 版 2002 年 10 月北京第 1 次印刷
印数：1—4000 册 定价：39.00 元

前言

所谓数据访问应用程序，就是能够从数据源中获取并操作数据的程序。它是目前最流行的计算机程序之一，其应用范围几乎覆盖了所有领域。从简单的数据输入、数据浏览和数据批处理程序，到复杂的客户/服务器应用程序，都属于数据库应用程序的范畴。不仅如此，各种类型的应用程序都有可能使用数据库，而不仅仅是进行磁盘文件的输出/输入。例如，一个用于阅读e-mail的客户程序看起来并不像传统的数据库应用程序，但是它的确可能使用数据库来存储地址、信件以及其他信息。

大多数应用程序都需要某些形式的数据访问。Visual C++ .NET 提供了以下几种优秀的数据访问技术：ADO，OLE DB，ODBC 和 DAO。如果要修改现存应用程序，为了便于维护，可以继续使用当前的数据访问技术，例如 DAO。不过如果希望应用程序的生命期更长，则应考虑为受控应用程序（Managed Application）使用 ADO.NET，而为原生应用程序（Native Application）使用 ADO。实际上，新的数据访问技术通常能缩短开发时间、简化代码和提供更好的性能。

目前用于数据库前端工具的开发环境有很多，例如：Delphi、Visual FoxPro、Visual Basic、PowerBuilder 以及 SQL Server 等。随着 Visual C++数据库开发功能的不断增强，它作为一种方便易用的前端开发工具在实际开发中被广泛使用。自 Visual C++ 4.0 版本起，就对数据库开发提供了很好的支持。而在 Visual C++ .NET 中，对数据库访问的技术更加成熟，功能更加强大。使用 Visual C++已经可以开发出功能强大、访问速度快、应用广泛、占用资源较少的数据库应用程序。

对数据库应用的开发，Visual C++ .NET 提供了全方位支持。作为一种开发环境，它对数据库的支持已经可以与 Visual Basic、Access、Delphi、FoxPro、Paradox 等相媲美，在对某些特性的支持上还超过了这些技术。Visual C++ .NET 中提供了多种多样的访问技术，例如 ODBC、MFC ODBC、DAO、OLE DB 和 ADO 等。这些技术各有特点、共同组成了强大的开发 Visual C++数据库应用的集成开发环境。

本书易于理解，对 Visual C++ .NET 数据访问技术的所有方面几乎都有介绍。本书面向中、高级读者。对于那些以前使用过 Visual C++的早期版本或其他结构化语言编过程序的读者来说，本书会更容易掌握。如果从头开始学习，那么很快就能熟练掌握 Visual C++ .NET 数据访问技术，并能充分利用其强有力的功能编写精彩、完美的 Windows 应用程序。

本书除封面署名作者外，武装、于佳音、朱石、周一兵、薛文涛、林茵茵、黄剑波、李义、王芳、沈鹏、刘树声、季洪飞、司马小凡、王江辉、廖晓筠、沈冰、汤春明、许颖、赵立峰、李国梁、胡建明、龚雪梅、黄君玲、吴强、郭士明等同志也为本书的出版付出了不同程度的劳动，在此一并表示感谢。

由于时间所限，书中错误和疏漏之处在所难免，敬请指正。

目录

第 1 章　数据访问技术概述 1
1.1　数据库基础知识 1
1.1.1　数据库技术的发展 1
1.1.2　数据库系统的特点 3
1.1.3　数据模型 5
1.1.4　系统结构 6
1.1.5　数据库管理系统 7
1.2　常用数据访问技术 8
1.2.1　MFC ODBC 8
1.2.2　ODBC API 9
1.2.3　OLE DB 9
1.2.4　ADO.NET 10
1.2.5　ADO 11
1.2.6　MFC DAO 13
1.3　Visual C++ .NET 的数据访问开发优势 14
本章小结 15
第 2 章　MFC ODBC 编程基础 16
2.1　MFC ODBC 概述 16
2.2　CDatabase 类 17
2.2.1　CDatabase 对象的种类 17
2.2.2　CDatabase 类的构建函数 18
2.2.3　数据库属性函数 21
2.2.4　数据库操作函数 23
2.2.5　数据库重载函数 25
2.2.6　CDatabase 类的数据成员 26
2.3　CRecordset 类 26
2.3.1　CRecordset 对象的种类 26
2.3.2　CRecordset 类的构建函数 27
2.3.3　记录集属性函数 30
2.3.4　记录集更新函数 33
2.3.5　记录集滚动函数 36
2.3.6　其他记录集操作函数 38
2.3.7　记录集重载函数 44
2.3.8　CRecordset 类的数据成员 47
2.4　事务 49
2.5　RFX 机制 50
2.5.1　RFX 与记录集对象 50
2.5.2　记录字段交换过程 51

2.5.3 RFX 函数 53
2.5.4 Visual C++向导所生成的 RFX 代码 53
2.5.5 Bulk RFX 机制 57
2.5.6 CFieldExchange 类 58
本章小结 60
第 3 章 掌握 MFC ODBC 编程技术 61
3.1 MFC ODBC 数据库开发概述 61
3.2 管理数据源 62
3.2.1 配置数据源 62
3.2.2 操作数据源 64
3.2.3 推广连接字符串 65
3.3 常规记录集管理编程 66
3.3.1 记录集类的结构 66
3.3.2 打开和关闭记录集 67
3.3.3 滚动记录集 68
3.3.4 定位记录集 70
3.3.5 过滤记录集 71
3.3.6 排序记录集 72
3.3.7 参数化记录集 72
3.3.8 锁定记录集 74
3.4 高级记录集管理编程 75
3.4.1 多表联合查询 75
3.4.2 使用预定义查询 78
3.4.3 动态绑定数据表列 80
3.4.4 实现多行存取 86
3.4.5 管理数据库中的大型数据项 88
3.4.6 得到 SUMs 以及其他合计结果 89
3.4.7 使用事务 89
3.4.8 联合使用 ODBC API 和 SQL 90
3.5 MFC ODBC 数据库应用程序的结构 91
3.5.1 创建支持数据库的应用程序 91
3.5.2 数据库应用程序的文档/视图结构 93
3.5.3 使用 CRecordView 类 94
3.6 学生信息管理系统开发实例剖析 97
3.6.1 创建应用程序框架 97
3.6.2 由 Application Wizard 生成的数据库实现代码 100
3.6.3 定制应用程序界面 110
3.6.4 显示数据库记录信息 111
3.6.5 实现多表动态参数化查询 112

3.6.6 操作数据库记录 117
3.6.7 实现多视图查询 121
本章小结 132
第 4 章 使用 ODBC API 访问数据源 133
4.1 ODBC 概 述 133
4.1.1 ODBC 的优越性 134
4.1.2 ODBC 的结构组成 135
4.2 ODBC 一致性 140
4.2.1 ODBC API 一致性 140
4.2.2 ODBC SQL 一致性 142
4.3 ODBC API 编程的基本概念 143
4.3.1 句柄 143
4.3.2 缓冲区 146
4.3.3 光标 148
4.3.4 ODBC 数据类型 152
4.3.5 ODBC 诊断 155
4.4 使用 ODBC API 的一般流程 156
4.5 初始化工作环境 157
4.5.1 分配环境句柄 157
4.5.2 设置环境属性 157
4.5.3 分配连接句柄 158
4.5.4 设置连接属性 158
4.6 数据源连接 159
4.6.1 数据源连接概述 159
4.6.2 使用 SQLConnect 建立连接 161
4.6.3 使用 SQLDriverConnect 建立连接 163
4.6.4 使用 SQLBrowseConnect 建立连接 166
4.7 操作数据源 169
4.7.1 选择 SQL 语法 169
4.7.2 构造 SQL 语句 170
4.7.3 分配语句句柄 172
4.7.4 设置语句属性 173
4.7.5 执行语句 173
4.7.6 参数化 SQL 语句 177
4.7.7 释放语句句柄 191
4.8 检索结果集 191
4.8.1 检索结果集元数据 192
4.8.2 绑定结果集列 193
4.8.3 检索单行记录 196

4.8.4 检索大型数据 197
4.8.5 检索多行记录 198
4.8.6 使用可滚动光标检索结果集 203
4.8.7 多结果集 205
4.9 更新结果集 205
4.9.1 定位更新和删除 206
4.9.2 使用 SQLSetPos 更新数据 209
4.9.3 使用 SQLBulkOperations 函数更新数据 211
4.9.4 使用 SQLSetPos 与 SQLBulkOperations 函数处理大型数据 212
4.9.5 事务处理 213
4.10 解除与数据源的连接 213
4.11 联合使用 ODBC API 和 MFC ODBC 214
4.11.1 创建应用程序框架 214
4.11.2 定制应用程序界面 214
4.11.3 定制 API 函数的结果集 216
4.11.4 查询并显示数据源信息 220
本章小结 223
第 5 章 OLE DB 编程基础 224
5.1 COM 简介 224
5.2 OLE DB 概述 226
5.2.1 统一数据访问 227
5.2.2 COM 的益处 227
5.2.3 OLE DB 消费者和提供者 228
5.3 OLE DB 对象模型 228
5.3.1 数据源 228
5.3.2 会话 229
5.3.3 命令 229
5.3.4 行集 230
5.3.5 存取器 231
5.3.6 事务 231
5.3.7 枚举器 231
5.3.8 错误 232
5.3.9 通告 232
5.3.10 OLE DB 模板、标志和其他实现 232
5.3.11 设计 OLE DB 结构 232
本章小结 233
第 6 章 设计 OLE DB 消费者 234
6.1 OLE DB 消费者的基本概念 234
6.1.1 OLE DB 消费者如何访问数据 234

6.1.2 数据源和会话 235
6.1.3 存取器和行集 235
6.1.4 命令和表 237
6.1.5 用户记录 237
6.1.6 纲要行集 239
6.2 使用 OLE DB 消费者向导 239
6.2.1 消费者向导的使用步骤 239
6.2.2 剖析由向导生成的代码 244
6.3 使用 OLE DB 消费者模板 252
6.3.1 使用数据库标志 252
6.3.2 确定字段状态 255
6.3.3 遍历行集 258
6.3.4 使用参数化查询 259
6.3.5 获取数据 260
6.3.6 更新行集 261
6.3.7 使用存储过程 265
6.3.8 使用存取器 267
6.3.9 访问 XML 数据 275
6.3.10 使用纲要行集获取元数据 277
6.3.11 事务支持 278
6.3.12 使用 OLE DB 记录视图 279
6.3.13 使用现存的 ADO 记录集 280
6.3.14 同步更新行引用 280
6.3.15 使用书签 280
6.3.16 获取 BLOB 数据 281
6.3.17 获取通告 283
6.4 OLE DB 消费者设计实例 283
6.4.1 使用消费者获取数据 283
6.4.2 为消费者添加书签支持 285
6.4.3 为消费者添加 XML 支持 287
6.4.4 手动添加消费者支持 289
本章小结 290
第 7 章 设计 OLE DB 提供者 291
7.1 OLE DB 提供者的基本编程概念 291
7.1.1 OLE DB 规范的支持级别 291
7.1.2 OLE DB 提供者模板结构 292
7.2 使用 OLE DB 提供者向导 297
7.2.1 提供者向导的使用步骤 297
7.2.2 剖析由向导生成的代码 300

7.3 使用 OLE DB 提供者模板 306
7.3.1 为提供者添加接口 306
7.3.2 在提供者中引用属性 306
7.3.3 在提供者中设置属性 307
7.3.4 在提供者中动态绑定列 308
7.3.5 在提供者中支持自由线程 309
7.3.6 测试和调试 310
7.3.7 转换提供者不支持的数据 311
7.3.8 支持通告 311
7.3.9 支持纲要行集 312
7.3.10 提供书签支持 317
7.3.11 OLE DB 一致性检查 320
7.4 OLE DB 提供者设计实例 322
7.4.1 创建简单的只读提供者 323
7.4.2 增强简单的只读提供者 325
7.4.3 创建可更新的提供者 333
本章小结 340
第 8 章 ADO 和 RDO 341
8.1 使用 ActiveX 控件 341
8.1.1 插入 ActiveX 控件 341
8.1.2 ActiveX 控件的封装类 343
8.1.3 使用 OLE/COM 对象查看器 343
8.1.4 在设计时设置 ActiveX 控件属性 344
8.1.5 设置 ActiveX 控件的事件处理函数 345
8.1.6 修改 ActiveX 控件的运行时行为 345
8.2 数据绑定 346
8.2.1 ADO 和 RDO 数据访问 346
8.2.2 支持数据绑定的 ActiveX 控件 346
8.2.3 ADO 数据绑定 349
8.2.4 RDO 数据绑定 350
8.2.5 捕捉错误 351
8.2.6 数据绑定的限制 352
8.3 配置数据库连接 353
8.3.1 ODBC 连接 353
8.3.2 Oracle 连接 355
本章小结 355
第 9 章 维护 MFC DAO 数据访问 356
9.1 MFC DAO 概述 356
9.1.1 DAO 对象 356

9.1.2 MFC 与 DAO 357
9.1.3 MFC DAO 类不能处理的 DAO 对象 358
9.1.4 DAO 能够操作的数据库 359
9.1.5 选择 MFC DAO 和 MFC ODBC 359
9.2 MFC DAO 数据库类 360
9.2.1 CDaoDatabase 类 361
9.2.2 CDaoRecordSet 类 362
9.2.3 CDaoTableDef 类 364
9.2.4 CDaoQueryDef 类 365
9.3 DFX 机制 366
9.3.1 DFX 与记录集对象 367
9.3.2 记录字段交换过程 367
9.3.3 DFX 函数 370
9.3.4 CDaoFieldExchange 类 371
9.4 使用工作区对象 372
9.4.1 操作隐式 MFC DAO 对象 373
9.4.2 显式打开默认工作区 374
9.4.3 管理事务 374
9.4.4 隔离事务 376
9.4.5 检索和设置数据库引擎属性 376
9.5 使用数据库对象 376
9.5.1 新建数据库 377
9.5.2 打开数据库 378
9.5.3 关闭数据库 379
9.6 使用记录集对象 380
9.6.1 记录集类的结构 380
9.6.2 创建记录集 382
9.6.3 滚动记录集定位 384
9.6.4 书签定位与绝对定位 385
9.6.5 条件定位 386
9.6.6 使用记录缓存 388
9.6.7 动态绑定 389
9.6.8 添加、删除和编辑记录集 390
9.6.9 排序记录集 392
9.6.10 过滤记录集 392
9.7 使用查询定义对象 393
9.7.1 新建查询定义 394
9.7.2 保存查询定义 396
9.7.3 打开查询定义 397

9.7.4 使用临时查询定义 398
9.7.5 执行查询定义 398
9.7.6 使用直通查询 399
9.7.7 参数化查询定义 399
9.8 使用表定义对象 400
9.8.1 新建数据库表 401
9.8.2 打开表定义 408
9.8.3 表型记录集 409
9.9 使用外部数据源 409
9.9.1 附加表 410
9.9.2 直接打开外部数据源 411
9.9.3 优化性能 412
本章小结 413

第 1 章　数据访问技术概述

大多数应用程序都需要某些形式的数据访问。如果要创建新应用程序，可以选择以下几种优秀的数据访问技术：ADO.NET、ADO、OLE DB、ODBC 或 DAO。如果要修改现有的应用程序，则可能为了便于维护，而继续使用应用程序当前使用的数据访问技术，例如 DAO。不过如果希望应用程序的生命期更长，则应考虑为受控应用程序（managed application）使用 ADO.NET，而为原生应用程序（native application）使用 ADO。实际上，新的数据访问技术通常能缩短开发时间、简化代码和提供更好的性能。本章将向读者介绍有关数据库开发的基础知识。

本章要点：

- 数据库基本知识
- 常用数据访问技术
- Visual C++ .NET 的开发优势

1.1　数据库基础知识

数据库是现代计算机系统的一个重要组成部分，是人们有效地进行数据存储、共享和处理的工具。现代计算机已不再仅仅应用于科学计算上，而是广泛应用于各种管理工作中，从某种意义上说，管理的过程就是信息的流动和处理过程。在管理活动中要涉及大量的信息存储、共享、流动和处理，因此，要使管理工作现代化，首先就要有一种工具来管理大量的信息，这种客观要求导致数据库这门技术的产生并得以迅速发展。现代的管理信息系统几乎都是以数据库作为其核心的，可以相信，它在现代社会生活中将会发挥越来越重要的作用。

1.1.1　数据库技术的发展

数据库技术至今已经有近 40 年历史，其发展大致经历了以下几个时期。

1. 摇篮时期

这个时期主要是指 20 世纪 60 年代。那时文件系统已经十分成熟，出现了多种文件组织方法和文件系统，外存储器已有了磁盘、磁鼓等直接存取、存储设备，而计算机也已经从用于科技计算大量地转到用于管理工作中。数据处理的客观需要使得人们开始寻找新的方法和工具，数据库的概念在这一时期开始形成，比较有影响的工作包括：

• 1963 年，C. W. Bachman 设计开发的 IDS（Integrated Data Store）系统开始投入运行，它可以使多个 COBOL 程序共享数据库。

• 1968 年，网状数据库系统 TOTAL 等开始出现。

• 1969 年，McGee 等人开发的层次式数据库系统，亦即 IBM 公司的 IMS 系统发布。

2. 发展时期

这个时期主要是指 20 世纪 70 年代。在这一时期，以 CODASYL 方式建立的网状数据库运行于各种计算机上，数据库的应用越来越广泛，成为信息系统开发不可缺少的工具。同时，随着商业及管理应用的广泛开展，以关系模型为中心的关系数据库基础理论研究不断充实，为关系数据库的形成奠定了基础，已开始出现较为完备的关系数据库系统。这一时期较为重要的工作有：

• CODASYL（Conference On Data System Languane）的一系列工作。CODASYL 是美国数据系统语言协会的简称。CODASYL 关于数据库的一系列工作和报告澄清了许多概念，建立了若干权威性的观点，例如 DBTG 报告给出了关于数据库的语言规范，它提出了许多基本概念，对网状系统的研制和发展起了重大的影响，很多网状系统都是采用 DBTG 模型的。CODASYL 的工作极大地推动了数据库技术的发展。

• 1970 年，IBM 公司的 San Jose 研究所的 E.F.Code 发表了“大型共享数据库数据的关系模型”的著名论文，开创了数据库的关系方法和关系规范化理论的研究，他本人因此荣获 1981 年度 ACM 图灵奖。关系方法由于其理论上的完美和结构上的简单，它的出现对数据库技术的发展起着至关重要的影响。

• 实验性关系数据库开始建立。20 世纪 70 年代中期，IBM 公司的 San Jose 研究所在 IBM 370 系列上研制了 SYSTEM R 关系型数据库管理系统，加州大学伯克利分校在 Vax 系列机上实现了 INGRES 关系数据库管理系统，这些试验系统在关系数据库管理系统的实现技术和系统性能方面做了大量的工作。

• 1978 年，美国标准化组织发表了关于数据库系统结构的最终报告，即 ANSI/X3/SPARC 建议，它规定了数据库系统的总体结构以及特征等。

•1979 年，美国 ORACLE 公司推出了第一个商品化的关系数据库系统，即 ORACLE V2.0 版。

除了上述工作外，这一时期还广泛开展了关于分布式数据库技术的研究。

3. 成熟时期

这个时期主要是指 20 世纪 80 年代至今。这一时期，大量商品化的关系数据库系统问世与广泛推广，既有适用于大型机的系统，也有适用于小型、微型机的系统。数据库的应用深入到人类生活的各个领域，从国家自然资源管理、国防建设、银行业务、交通运输、情报检索、人口统计、航空旅游服务到各种企业管理，办公自动化等。关系数据库技术已经十分成熟，数据库技术的研究开始转向新的应用领域提出的新的需求，例如为了支持工程数据中设计周期长、数据结构复杂、需要反映数据的时间属性等要求，出现了采用面向对象方法的工程数据库；又如，为了能使数据库技术用于处理知识，数据库技术与人工智能相结合涌现了专家数据库系统、演绎数据库系统、知识库系统等智能化数据库系统，即新一代的数据库系统。

这一时期分布式数据库技术也逐步走向实用，例如 1986 年相继推出了 INGRES/

STAR 和 SQL*STAR。后者是 ORACLE 公司推出的开发型分布式关系数据库系统，亦即 ORACLE RDBMS V5.1 版。尽管这些分布式数据库产品尚不能支持一些复杂的分布功能，如分布式并发控制等。但完全能满足大量存在的分布式查询处理的要求。

1.1.2　数据库系统的特点

什么是数据库系统？这个问题很难用一句话来概括。为了使读者一开始就对这个问题有一个初步的认识，我们简单地把数据库系统看做是“管理大量的、持久的、可靠的、共享的数据的工具”。从这个简单的定义可以看出数据库系统是一种管理数据的工具。它的管理对象有以下特征：“大量”，这表明数据量很大，不能放在通常的内存中，需要大容量的外层作为支持；“持久”，表明数据必须长久地被保留，亦即数据不是简单地为某一特定的应用准备的，不是一当应用完成数据就随之消失（如科学计算中常见的现象）；“可靠”，是指一旦系统发生软硬件故障，可恢复数据库；“共享”，是指若干个用户应当能够按照一定有序的方式存取数据库中的数据，避免同步存取可能会造成的错误。

熟悉文件系统的读者也许会问，文件系统在某种程度上也具有以上特征，那么文件系统和数据库有区别吗？和文件系统相比，数据库系统有哪些显著的特点呢？

首先，数据库中的数据是高度结构化的。在文件系统中，从整体上来说，数据是无结构的，即文件的记录型之间没有联系，它只关心记录内部数据项之间的联系。一个典型的文件结构是由一组按照树状形式组织的数据项的集合，如图 1-1 所示。

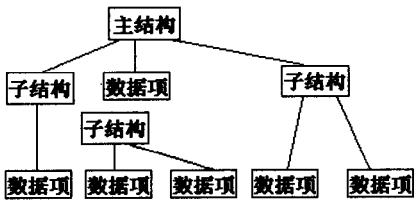

图 1-1　典型的文件组织形式

数据库系统不仅要考虑数据项之间的联系，更要考虑记录型之间的联系，因为在同一个组织系统中，存在于不同文件中的数据之间仍然有着这样或那样的联系。例如，在一个企业中通常都会有人事系统、工资系统、库存系统、销售系统等，如果采用文件系统，它们的数据都必须存放在相互分离的文件中，但事实上这些系统之间显然存在着客观联系，使用文件系统就会人为地将其割裂开来，使它们的联系只有通过应用程序才能得以体现。而在数据库中则不仅要存储数据，而且还要描述这些子系统之间的联系，这种联系在数据库中是通过所谓的存取路径来实现的。例如在关于学生选课情况的管理中，需要知道学生的基本情况，课程的基本情况，亦即选课情况，它们尽管也可以分别放在不同的文件系统中，但由于这 3 种信息实体实际上是有着密切联系的，如图 1-2 所示。

显然，如果要查找诸如“安锦的学习成绩”、“获满分的学生姓名及课程名称”等信息，就必须沿着图中的两条存取路径，从一个记录型到另一个记录型。可以说通过存取路径来表示自然的数据联系，是数据库系统与文件的根本区别。

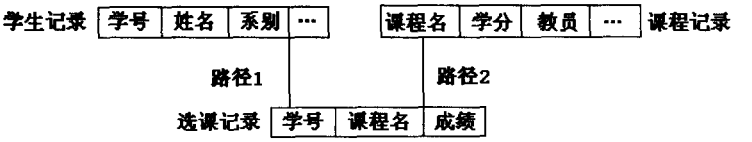

图 1-2　选课管理系统中的信息联系

其次，数据库中的数据是面向系统的，而不是面向某个应用的，因此数据库中数据的共享程度比文件系统更高。实现数据的共享是数据库的重要特征。文件作为数据的组织方式，基本上还是面向具体应用的，例如 COBOL 语言的文件说明是在程序内部进行的，程序与数据的关系如图 1-3 所示。

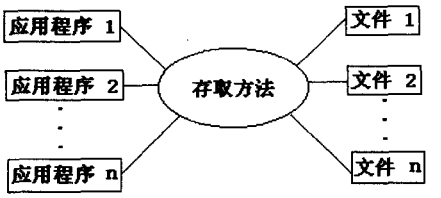

图 1-3　文件系统中的应用程序与数据的关系

在数据库中，数据不是为某个具体应用而准备的，而是从整个组织系统出发，考察整个系统中的各种信息需求，统一地进行数据的组织、定义和存储。数据的定义是与应用程序分开的，因此数据库可以为全系统中的各种应用所使用，达到数据共享的目的，因此说数据库是面向系统的。数据库系统中程序与数据的关系如图 1-4 所示。

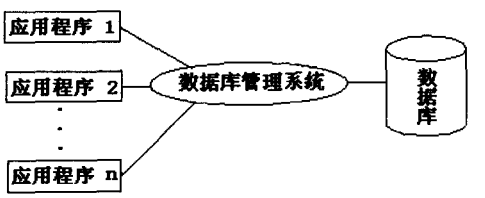

图 1-4　数据库系统中程序与数据的关系

数据面向系统的另一个直接好处是可以使信息结构比较稳定，而且易于扩充，这使得一个组织中的数据库的建设可以比较独立地进行。

第三，数据库系统比文件系统具有更高的独立性。在使用数据库时，应用程序对存储结构有较高的独立性。这种独立性是由系统在存储结构和逻辑结构之间提供的映像来获得的，这样，当存储结构或者物理结构改变时，只要相应地改变系统的逻辑结构和物理结构之间的映像就可以使逻辑结构保持不变，从而建立在逻辑结构上的应用程序也保持不变，这称为物理独立性。另一方面，一个数据库系统所拥有的数据比某个特定的应用所需要的数据要多得多，因此对每个应用还要提供局部的逻辑结构，这种局部逻辑结构只是总体逻辑结构的子集，局部逻辑结构和全局逻辑结构之间使用映射进行联系。这样就可以做到当总体逻辑结构变化时，局部逻辑结构保持不变，而程序员根据局部逻辑结构编写的程序也可以不变，这就是所谓的逻辑独立性。提高数据独立性是数据库所追求的一个主要目标。

第四，具有较好的保护数据安全性和维护数据一致性的措施。由于数据库的数据是面向系统的，一个组织系统中的信息系统是系统的神经中枢，因此数据库中的信息通常是非常重要的，并不是任何用户都可以查看使用的。因此数据库系统都需要有一定的授权机制以防止对数据的不合法使用，只有那些被授权可以存取数据库的人或程序才能执行对数据库的存取。

另外，在数据库中强调共享，在一个多用户系统中会出现许多用户同时使用数据库的情况，这种并发的存取动作如果不加以控制必将导致严重的后果，这就如同十字路口的交通，如果没有交通警察的控制指挥和交通规则的约束，必然会引起撞车等严重事故。我们将数据库的这种功能称为数据操纵的一致性。另一方面，数据库中的数据是对客观世界中的某些实体性质的反映，因此它的存在应该遵守一定的原则。例如，年龄这个数据是反映客观世界中人的年龄这个属性的，因此数据库中年龄值就不应该出现负值或极大的正值，例如 250 岁。比如在会计做账中，收支应该平衡，这在会计数据库中就要求在“任何”时候，收支之和平衡相抵。数据库中数据的这种正确性要求，我们称为数据的语义一致性。

最后，在数据库中对记录的存取不一定以记录为单位。我们知道，文件系统对数据的存取都是以记录为单位的，如果记录很长，而我们只是需要其中很少的几个字段，那么以记录为单位的存取就显得很浪费，数据库则可克服这一点，仅仅将需要的信息取出，例如只取出记录中的某个字段值。

上面我们从数据库所管理的数据对象的特征的角度和与文件系统相比较的角度描述了数据库系统的特点，综合起来，可以说数据库是管理系统中大量、持久、可靠、共享的数据的工具，这些数据具有最小的冗余度和较高的数据与程序的独立性，而且数据库应该能保持数据安全性、维护数据一致性。

1.1.3　数据模型

在上一节中已经向读者介绍了，数据库中的数据是高度结构化的，即数据库不仅要考虑记录内数据项之间的联系，还要考虑记录之间的联系。所谓数据模型主要就是指描述这种联系的数据结构的形式。

在数据库发展的历史上，最有影响的数据模型包括以下几种：

- 层次数据模型：这种模型是用树形结构来描述客观世界实体及其联系。
- 网状数据模型：这种模型是用网状结构来描述客观世界实体及其联系。
- 关系数据模型：这种模型是用二维表结构来描述客观世界实体及其联系。
- 实体联系数据模型：这种模型是用实体联系图来描述客观世界实体及其联系。

由于 Visual C++所提供的方法主要是针对关系数据模型的（OLE DB 除外），因此这里将主要介绍关系数据模型。

在关系数据模型中，信息被组织成一系列二维表的结构，每一张二维表被称为一个关系（relation）或表（table）。每一个表中的信息只用来描述客体世界中的一件事情，例如雇员信息，如表 1-1 所示。

表（Table）也称为关系，由表名、列名以及若干行组成。例如在上表中，表名为