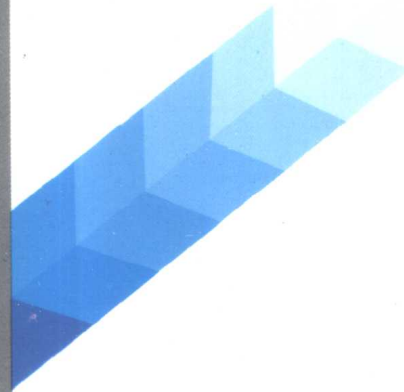
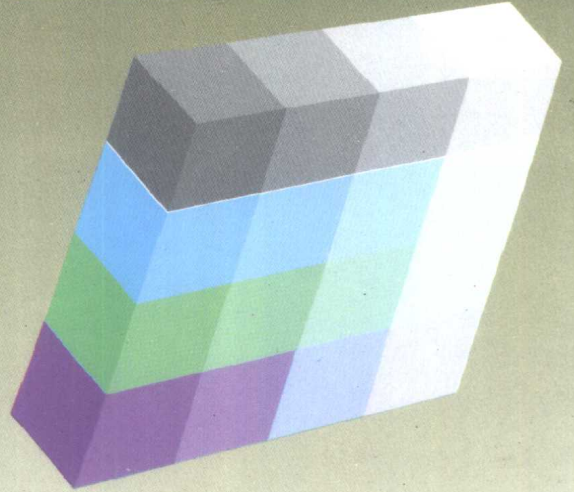
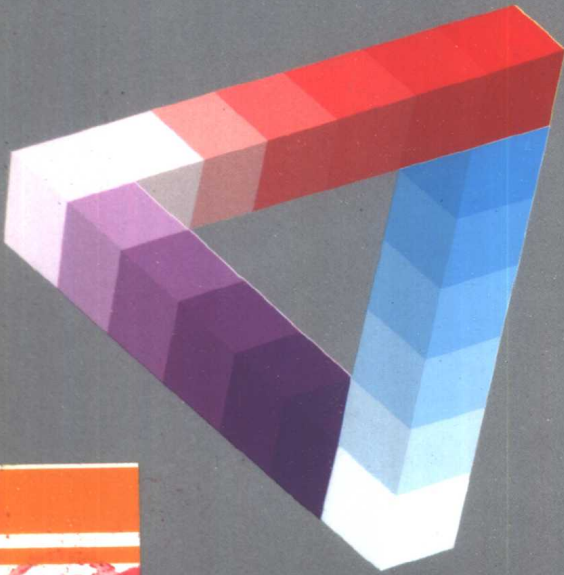
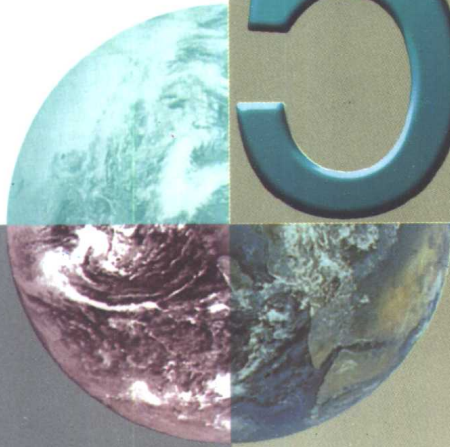


姜洪 张希 等编著

Delphi

5

数据库技术



国防工业出版社

Delphi 5

姜洪 张希 编著



Delphi
5
数据库技术

国防工业出版社

·北京·

内 容 简 介

本书全面介绍了 Delphi 5 的数据库编程技术,包括 Delphi 的数据库系统、数据访问组件、数据控制组件、数据模块、字段对象、SQL 编程、报表、交叉数据表、Client/Server、MIDAS、Web 编程等内容。

本书内容全面又不失简洁,范例丰富。既可作为广大用户学习 Delphi 5 数据库编程的入门指导书,也可作为程序员编程时的参考手册。

图书在版编目(CIP)数据

Delphi 5数据库技术/姜洪等编著. —北京:国防工业出版社,2001.4

ISBN 7-118-02461-9

I . D . . . II . 姜 . . . III . DELPHI 语言 - 程序设计
IV . TP312

中国版本图书馆 CIP 数据核字(2000)第 87469 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)

北京奥隆印刷厂印刷

新华书店经营

*

开本 787×1092 1/16 印张 23¼ 532 千字

2001 年 4 月第 1 版 2001 年 4 月北京第 1 次印刷

印数:1—4000 册 定价:32.00 元

(本书如有印装错误,我社负责调换)

前 言

有这样一种说明:真正的程序员用 C,聪明的程序员用 Delphi。笔者编的第一个程序就用的是 Turbo Pascal 5 (Delphi 前 Borland 开发的同种语言的编程平台);后来又经历了从 TC20 到使用 BC31 又到 VC5,在因项目开发需要而转向 Delphi 后才真正对这句话有所感触。Delphi 与 VC 相比,虽速度上没有优势,但语法约束性强,出错容易查找,可视化程度更高。Delphi 同 VB、PB 等语言同为真正的可视化语言,基于窗口和对象的技术;与 Windows 平台紧密结合;对数据库及网络开发有强大的支持,编译快速高效;很多建模工具,如 SA、Rose 均对其有支持;能够比较平滑地从分析和设计过渡到编码实现,从而在众多开发工具中独树一帜。

在所有可视化开发工具中,Delphi 无疑是出类拔萃的。然而,对大多数开发者而言,选择 Delphi 的真正原因是因为其强大的数据库处理功能。可以说,将传统的应用开发和数据库应用开发完美地结合在一起,是 Delphi 的真正优越之处。因此,不掌握 Delphi 的数据库开发技术,就谈不上掌握了 Delphi。同时应该看到,在实际应用中,大部分都和数据库有关。

Delphi 的数据库功能强大,适应于多种数据库结构,从客户机/服务器模式到多层数据库结构模式;它有高效率的数据库管理系统和新一代更先进的数据库引擎;具有最新的数据分析手段和大量的企业组件。Delphi 通过内置的数据库引擎(BDE)提供了一个高效、可伸缩的数据库开发平台,可快速开发基于各类 DBMS 的应用程序。

本书详细地介绍了 Delphi 数据库开发的各种技术,同时结合了大量实例。通过本书的学习,读者可以掌握 Delphi 开发数据库程序的基本方法和技巧、数据库设计的技术及基于 C-S、Web 的数据库开发技术。

在读者阅读本书之前要求了解 Object Pascal,并有一定编程基础,对 Windows 可视化编程及 Delphi 有一定的了解。

参与本书编写工作的还有章务淳、杨为、张中卫、刘关、袁武、孟仁信、郑可维、黄化舒、吴廖、徐宇、肖峰、赵宇飞、胡晓峰、王友平、周剑波等。

由于作者水平有限,时间仓促,书中难免有疏漏和错误,恳请广大读者提出宝贵意见。

目 录

第 1 章 Delphi 数据库系统	1
1.1 数据库系统概述	1
1.1.1 数据库	1
1.1.2 数据库管理系统	3
1.1.3 关系数据库模型	4
1.1.4 数据库应用程序	4
1.2 Delphi 数据库体系结构	6
1.2.1 数据访问组件	7
1.2.2 数据控制组件	7
1.3 Delphi 5 在数据库方面的新技术	8
1.3.1 ADO 数据集组件	9
1.3.2 数据模块设计体	9
1.3.3 IBX 组件	11
1.3.4 MIDAS 方面的增强	12
1.4 数据库应用程序的开发步骤	13
1.4.1 概述	13
1.4.2 数据库应用程序的开发步骤	13
第 2 章 数据访问组件	15
2.1 数据访问组件简介	15
2.2 数据集组件 TDataSet 及其使用	16
2.2.1 数据集组件的当前状态	17
2.2.2 数据集的打开和关闭	18
2.2.3 数据记录指针的移动	19
2.2.4 数据更新	20
2.2.5 书签管理	23
2.2.6 控制连接	24
2.2.7 记录缓冲区管理	25
2.2.8 过滤记录	27
2.2.9 数据集组件的响应事件	28
2.3 TTable 组件及其使用	30
2.3.1 TTable 组件的重要属性	31
2.3.2 TTable 组件的重要方法和事件	36

2.3.3	用 TTable 组件动态创建数据库	38
2.4	TDataSource 组件及其使用	39
2.4.1	TDataSource 组件的重要属性	39
2.4.2	TDataSource 组件的重要方法和事件	40
2.5	TBatchMove 组件及其使用	41
2.5.1	TBatchMove 组件的重要属性	42
2.5.2	TBatchMove 组件的重要方法和事件	43
2.6	TNestedTable 组件简介	43
2.7	TSession 组件及其使用	44
2.7.1	TSession 组件的重要属性	44
2.7.2	TSession 组件的重要方法和事件	46
2.7.3	TSession 组件应用实例	50
2.8	TUpdateSQL 组件及其使用	52
2.8.1	TUpdateSQL 组件的重要属性	53
2.8.2	TUpdateSQL 组件的重要方法和事件	54
第 3 章	数据控制组件	55
3.1	数据控制组件简介	55
3.2	TDBText 组件及其使用	57
3.2.1	TDBText 组件的重要属性	58
3.2.2	TDBText 组件的应用实例	58
3.3	TDBEdit 组件及其使用	59
3.4	TDBGrid 组件及其使用	60
3.4.1	TDBGrid 组件的重要属性	60
3.4.2	TDBGrid 组件的重要方法和事件	63
3.4.3	TDBGrid 组件的应用实例	64
3.5	TDBNavigator 组件及其使用	65
3.5.1	TDBNavigator 组件的重要属性	66
3.5.2	TDBNavigator 组件的重要方法和事件	67
3.6	TDBMemo 组件及其使用	68
3.6.1	TDBMemo 组件的重要属性	68
3.6.2	TDBMemo 组件的重要方法和事件	69
3.7	TDBImage 组件及其使用	69
3.7.1	TDBImage 组件的重要属性	69
3.7.2	TDBImage 组件的重要方法和事件	69
3.7.3	TDBImage 组件和 TDBMemo 组件的应用实例	69
3.8	数据控制组件中的列表框和组合框	70
3.8.1	TDBListBox 组件及其使用	71
3.8.2	TDBComboBox 组件及其使用	72
3.8.3	TDBLookupListBox 组件及其使用	73

3.8.4	TDBLookupComboBox 组件及其使用	74
3.9	TDBCheckBox 组件及其使用	75
3.10	TDBRadioGroup 组件及其使用	76
3.11	TDBRichEdit 组件及其使用	77
3.12	TDBCtrlGrid 组件及其使用	77
3.12.1	TDBCtrlGrid 组件的重要属性	77
3.12.2	TDBCtrlGrid 组件的重要方法和事件	78
3.12.3	TDBCtrlGrid 组件的应用实例	79
3.13	TDBChart 组件及其使用	80
第 4 章	数据库应用开发基础	86
4.1	创建第一个 Delphi 数据库应用程序	86
4.1.1	用 Database Desktop 创建数据库	86
4.1.2	使用 Database Form Wizard 创建数据库窗体	89
4.1.3	了解由 Database Form Wizard 创建的数据库窗体	90
4.1.4	运行应用程序	94
4.1.5	创建数据库应用程序的步骤	94
4.2	创建主要—明细型数据库应用程序	95
4.2.1	一对多关系的主要—明细型数据库应用	95
4.2.2	一对多—多关系的主要—明细型数据库应用	98
4.3	使用数据模块	99
4.3.1	为什么使用数据模块	99
4.3.2	如何使用数据模块	100
4.3.3	使用数据模块的一个实例	104
4.4	数据库中的数据处理	104
4.4.1	查询数据库表中的数据	105
4.4.2	修改数据库表中的记录	109
4.4.3	输入数据的有效性验证	115
4.5	字段对象的使用	117
4.5.1	字段对象的类型	118
4.5.2	创建永久性的字段对象	118
4.5.3	使用计算字段	120
4.5.4	字段对象的属性设置	122
4.5.5	字段对象的重要方法和事件	125
4.5.6	字段对象的访问	126
4.5.7	字段对象的读取和赋值	127
4.5.8	设定字段对象的显示格式	127
第 5 章	SQL 编程	129
5.1	SQL 语言简介	129
5.1.1	SQL 语言的历史	129

5.1.2 SQL 语言的组成	130
5.1.3 SQL 语言的特点	130
5.2 SQL 的数据操作	131
5.2.1 SQL 的数据类型	131
5.2.2 从数据库中检索数据	132
5.2.3 数据更新	136
5.3 TQuery 组件及其使用	137
5.4 编写第一个简单的 SQL 程序	138
5.4.1 SQL 命令文本的编写	141
5.4.2 SQL 程序的执行	143
5.4.3 通过 TQuery 组件如何获得活动的数据	144
5.4.4 Prepare 方法的使用	145
5.4.5 异构查询	145
5.5 动态 SQL 的编程	146
5.6 SQL 编程实例	149
第 6 章 报表的制作和使用	152
6.1 QuickReport 组件	152
6.1.1 TQuickRep 组件及其使用	154
6.1.2 TQRSubDetail 组件及其使用	159
6.1.3 TQRBand 组件及其使用	160
6.1.4 TQRChildBand 组件及其使用	162
6.1.5 TQRGroup 组件及其使用	163
6.1.6 TQRLabel 组件及其使用	163
6.1.7 TQRDBText 组件及其使用	164
6.1.8 TQRExpr 组件及其使用	164
6.1.9 TQRSysData 组件及其使用	166
6.1.10 TQRMemo 组件及其使用	167
6.1.11 TQRRichText 组件及其使用	167
6.1.12 TQRDBRichText 组件及其使用	167
6.1.13 TQRShape 组件及其使用	167
6.1.14 TQRImage 组件及其使用	168
6.1.15 TQRDBImage 组件及其使用	168
6.1.16 TQRCompositeReport 组件及其使用	168
6.1.17 TQRPreview 组件及其使用	169
6.1.18 TQRChart 组件及其使用	169
6.2 快速生成报表	169
6.2.1 使用 Quick Report Wizard	169
6.2.2 使用 Report Templates	173
第 7 章 Decision Cube 组件	186

7.1	Decision Cube 组件简介	186
7.2	TDecisionQuery 组件及其使用	187
7.2.1	指定交叉数据表中的维	188
7.2.2	编写 SQL 语句	188
7.3	TDecisionCube 组件及其使用	189
7.3.1	TDecisionCube 组件的重要属性	189
7.3.2	Decision Cube Editor 编辑框	191
7.3.3	TDecisionCube 组件的主要方法和事件	193
7.4	TDecisionGrid 组件及其使用	195
7.4.1	TDecisionGrid 组件的重要属性	195
7.4.2	TDecisionGrid 组件的重要方法和事件	197
7.5	TDecisionGraph 组件及其使用	199
7.6	TDecisionPivot 组件及其使用	199
7.6.1	TDecisionPivot 组件的重要属性	201
7.6.2	TDecisionPivot 组件的重要方法和事件	201
7.7	TDecisionSource 组件及其使用	201
7.7.1	TDecisionSource 组件的重要属性	202
7.7.2	TDecisionSource 组件的重要方法和事件	203
7.8	Decision Cube 组件的应用实例	206
第 8 章	客户/服务器应用程序开发	210
8.1	客户/服务器应用开发原理	210
8.1.1	客户/服务器体系结构	210
8.1.2	各种数据库服务器产品介绍	212
8.1.3	Delphi 的客户/服务器解决方案——IDAPI	215
8.2	Delphi 客户/服务器应用开发环境的构造	223
8.2.1	建立和管理数据库别名 (Alias)	224
8.2.2	配置 SQL Link 驱动程序缺省设置的方法	225
8.2.3	连接 SQL 服务器	228
8.2.4	Delphi 与各种数据库的链接	231
8.3	Delphi 客户/服务器数据库开发技术	240
8.3.1	使用 TDatabase 组件连接数据库服务器	240
8.3.2	处理 Client/Server 事务控制	248
8.3.3	使用存储过程	251
8.3.4	从开发平台到服务器的向上适化	255
8.4	客户/服务器应用实例分析	257
8.4.1	数据库环境介绍	258
8.4.2	应用程序分析	260
第 9 章	创建多层数据库应用程序	273
9.1	多层数据库技术概述	273

9.1.1	多层数据库应用程序简介	273
9.1.2	采用多层数据库结构的特点	274
9.2	Delphi 中的多层数据库技术	275
9.2.1	Delphi 多层数据库技术基础	275
9.2.2	基于 MIDAS 技术的多层数据库应用程序	276
9.3	创建应用程序服务器	278
9.4	创建客户端应用程序	281
9.5	深入 MIDAS 技术	284
9.5.1	设置远程数据模块	284
9.5.2	连接到应用程序服务器	289
9.5.3	客户端程序的结构	291
9.5.4	管理与服务器的连接	293
9.5.5	在客户端纠错	295
9.5.6	存取数据原理	295
9.5.7	更新数据	297
9.5.8	刷新数据	303
9.5.9	传递数据	304
9.6	注册、安装应用程序服务器	305
9.6.1	注册应用程序服务器为自动操作服务器	305
9.6.2	将 MTS 对象安装到 MTS 数据包中	306
9.6.3	注册应用程序服务器的接口	307
9.7	自定义应用程序服务器	307
9.7.1	扩展应用程序服务器的接口	307
9.7.2	使用自定义的数据表	308
9.8	多层应用程序的事务处理管理	308
9.9	分布式主要-明细型关系	309
9.10	公文包模式	310
9.11	连接缓冲池	310
9.12	安全	311
9.13	把客户端应用程序作为 ActiveX 组件发布	311
9.13.1	把客户端应用程序作为 ActiveX 组件发布	311
9.13.2	为客户端应用程序创建 ActiveForm	312
9.14	多层数据库应用程序的应用实例	313
第 10 章	Web 服务器应用程序开发	319
10.1	几种 Web 服务器应用开发技术介绍	319
10.1.1	CGI	319
10.1.2	Web Server API	320
10.1.3	ASP	320
10.1.4	JDBC	321

10.2 Delphi 对 Web 服务器应用程序的支持	321
10.2.1 WWW 工作原理	321
10.2.2 Web 服务器的工作原理	322
10.2.3 Delphi 对 Web 服务器应用程序支持的几种方式	322
10.2.4 Web 服务器应用程序的结构	323
10.2.5 各组件如何协调工作.....	324
10.2.6 如何迅速创建应用.....	325
10.2.7 利用数据库组件轻松响应用户请求.....	327
10.3 用 Delphi 创建 Web 数据库应用程序	328
10.3.1 TWebModule 和 TWebDispatcher 组件及其使用	328
10.3.2 TWebRequest 和 TWebResponse 对象及其使用	334
10.3.3 TPageProducer 组件及其使用	335
10.3.4 TDataSetTableProducer 和 TQueryTableProducer 组件及其使用	338
10.3.5 使用 Cookie	343
10.3.6 重定向到另一个 Web 站点	346
10.3.7 从 HTML 表单中获取信息	346
10.3.8 数据流.....	348
10.4 Web 服务器应用程序实例	352

第 1 章 Delphi 数据库系统

如果程序员需要访问和处理大量数据,就要用到数据库。数据库能比较简单和容易地管理这些数据。如果程序员不用数据库,就不得不编写复杂的程序,实际上,有时甚至无法完成任务,更说不上高效地处理数据了。

计算机应用从科学计算进入数据处理是一个划时代的转折。计算机已从少数科学家手中的珍品变为广大科技人员和管理人员工作中的得力助手和有用工具。

在进行具体的数据库编程前,我们应该先了解数据库的有关概念、功能以及数据库是怎样组织起来的。

1.1 数据库系统概述

数据处理是指对各种形式的数据进行采集、存储、加工和传播的一系列活动的总和。其目的是从大量原始数据中抽取、推导出对人们有价值的信息,以作为行动和采取对策的依据;也为了借助计算机科学地保存和管理复杂的数据,以便能有效和充分地利用这些宝贵的信息资源。

数据处理的中心问题是数据管理。数据管理是指对数据的分类、组织、编码、存储、检索和维护。数据管理随着计算机硬件业和软件业的不断发展而不断发展。30 多年来经历了以下三个阶段:

- (1) 人工管理阶段;
- (2) 文件管理阶段;
- (3) 数据库系统阶段。

数据库系统阶段是目前为止最后一个阶段,也是最有效地进行管理的一个阶段。数据库提供了一种存储数据的手段,数据库工具则提供了相应的操纵数据库和浏览数据的机制。

数据库系统提供了一种把相关的信息聚集在一起的方法,同时还提供了在某个集中的地方存储和维护这些信息的方法。数据库系统包括如下三个部分(图 1-1):

- (1) 数据库;
- (2) 数据库管理系统(DBMS);
- (3) 数据库应用程序。

数据库是按一定的结构,组织在一起的相关数据的集合。DBMS 是用来组织和管理数据信息的。数据库应用程序使我们能够获取、显示和更新由 DBMS 存储的数据。

1.1.1 数据库

很多应用程序都要用到数据库。在使用数据库之前,为了数据库能符合应用程序的

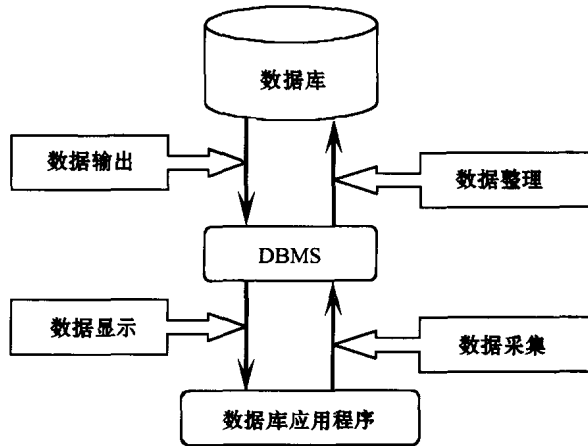


图 1-1 数据库系统的基本结构

功能,我们要考虑使用什么样的数据库。

在本书中,我们将讨论三种类型的数据库布局:独立数据库、用户共享数据库和客户机/服务器数据库。

下面介绍几种不同的数据库类型。

1. 独立数据库

独立数据库,就是这个类型的数据库和其他数据库没有关系,并发时间也不可能发生,所以说此种数据库是独立的。独立数据库是最容易处理的数据库,它可以不用考虑很多问题。独立数据库包括两个方面的内容:存储在局部文件系统的数据库和访问驻留在同一台计算机上的数据库工具。

独立数据库虽然简单,但它是一切数据库的基础,是非常有用的一种数据库类型。如果用户的应用程序用的是独立数据库,那么每个用户都可以保存一个独立数据库。对这个应用程序的某一个用户来说,他不需要访问其他人的数据。

Delphi 支持很多功能强大的数据库系统。可以利用随 Delphi 带来的 Database Desktop 来创建和操纵 Paradox 和 dBase 数据库。这些数据库可以作为独立数据库,也可以作为文件共享数据库。

2. 文件共享数据库

文件共享数据库和独立数据库几乎一模一样,只是文件共享数据库可能被多个客户通过网络进行访问。也就是说,文件共享数据库提供了更大层次上的可访问度,而且网络操作响应了网络时代的需求。

文件共享数据库还有一个特点就是没有预定网络的类型。数据库工具包不管现在在 Novell 网络上运行,还是 Microsoft NT,或是 Banyan,或是在其他类型的网络上运行,都能正常地工作,仅仅是因为它把数据库看作文件。

当应用程序需要对数据库完成大量的计算和并发访问时,文件共享数据库就不适用于这个应用程序了,这时候就要用到客户机/服务器数据库。

3. 客户机/服务器数据库

数据库访问的高级解决办法是使用客户机/服务器数据库。在这种情况下,指定一台

专用的计算机作为服务器,各个客户对数据的访问都在服务器内完成。

在客户机/服务器数据库中,客户请求服务器完成指定的任务,由客户机来决定是等待服务器的处理结果,还是处理其他更迫切的事务。与此同时,对服务器进行优化处理,服务器就可以更快的速度来完成处理请求。

尽管客户机/服务器数据库的体系结构有很多好的性能,但也存在着明显的缺点。

与文件共享解决办法相比较,客户机/服务器解决办法往往更加昂贵。

客户服务器软件还需要一种协议,例如 TCP/IP,要通过协议才进行交谈。尽管这种办法比较灵活,但需要附加的配置和管理功能。

4. 多级数据库

在网络上有一种新的激动人心的数据处理方法,这种方法称为多级数据库处理系统。这种系统涉及到“小型”客户,应用服务器和后台服务器的组合。其结果是形成了一种允许失败和数据重复的系统。

1.1.2 数据库管理系统

数据库管理系统(DBMS)是用于描述、管理和维护数据库的程序系统,是数据库系统的核心组成部分。它建立在操作系统的基础上,对数据库进行统一的管理和控制。其主要功能有:

- (1) 描述数据库 描述数据库的逻辑结构、存储结构、语义信息等;
- (2) 管理数据库 控制整个数据库系统的运行,控制用户的并发性访问,检验数据的安全、保密与完整性;
- (3) 存取数据 执行数据检索、插入、删除、修改等工作;
- (4) 维护数据库 控制数据库初始数据的装载、记录工作日志、监视数据库性能、修改更新数据库、重新组织数据库、恢复出现故障的数据库;
- (5) 其他功能 与网络中其他软件系统的数据通信以及组织数据的传输,一个 DBMS 与另一个 DBMS 或文件系统的数据库转换功能等。

数据库管理系统 DBMS 主要有以下四种类型:

- 文件管理系统;
- 层次数据库系统;
- 网状数据库系统;
- 关系数据库系统。

DBMS 中存储了大量的数据信息,其目的是为用户提供数据信息服务,而数据库应用程序正是与 DBMS 进行通信,并访问 DBMS 中的数据,它是 DBMS 实现其对外提供数据信息服务这一目的的唯一途径。简单地说,数据库应用程序是一个允许用户插入、修改、删除并报告数据库中的数据计算机程序。

在一般情况下,DBMS 和数据库应用程序都驻留在同一台计算机上并在同一台计算机上运行,很多情况下两者甚至结合在一个程序中,以前使用的大多数机器都是这样的。但随着 DBMS 技术的发展,目前的数据库系统迅速地向客户/服务器方向发展。数据库应用程序在一个或多个客户工作站(客户机)上运行,并且通过网络与运行在其他计算机上(服务器)的一个或多个 DBMS 进行通信。

1.1.3 关系数据库模型

四种不同类型的 DBMS 的结构各有其特点,但在功能实现上基本一致。因为目前关系数据库应用最为广泛,所以重点对关系数据库中的几个概念进行介绍。

1. 关系数据库(Relational Database)

关系数据库是应用数学的方法来处理数据库数据。一个关系数据库是由若干个表组成的。在 Delphi 中,数据库概念对应到物理文件上是不同的。对于 dBase、Foxpro、Paradox 这三种数据库,数据库对应于一个子目录,而其他类型如 MS Access、Btrieve 则是指某个文件。这是因为前者的表为一个单独的文件,而后者的表是聚集在一个数据库文件中的。

2. 表(Table)

一个表就是一组相关的数据按行排列,像一张表格一样。比如一个公司里每个员工的待遇情况,存在一个表中,每一行对应一个员工,在这一行中,包括员工的员工号、姓名、以及基本工资和奖金等。

3. 字段(Field)

在表中,每一列称为一个字段,每个字段都有相应的描述信息,如数据类型、数据宽度等。

4. 记录(Record)

在表中,每一行称为一个记录。

5. 索引(Index)

为了加快访问数据库的速度,许多数据库都加了索引。

6. 关系模型

关系模型有三部分组成:数据结构,关系操作,关系的完整性。下面简单地介绍一下这三个部分的具体含义。

在关系模型中,关系是唯一的一种数据结构,无论是实体还是实体之间都由单一的数据结构类型即关系来表示。

关系模型给出了关系操作的特点,但不不对 DBMS 的语言提出具体的语法要求。

关系语言是高度非过程化的语言。用户不必请求建立特定的存取路径,存取路径的选择是由 DBMS 的优化机制完成的。此外,用户也不必求助于循环、递归来完成数据操作。

关系模型有三类完整性约束:实体完整性、参照完整性和用户自定义完整性。关于这三个完整性的具体含义,读者可以看一些有关数据库的资料。这已经超出了本书的范围,就不再多说了。

1.1.4 数据库应用程序

DBMS 中存储了大量的数据信息,其目的是为用户提供数据信息服务。而数据库应用程序能够与 DBMS 进行通信,并访问 DBMS 中的数据,它是 DBMS 实现其对外提供数据信息服务的唯一途径。简单地说,数据库应用程序是一个允许用户插入、修改、删除并报告数据库中的数据的计算机程序。数据库应用程序在传统上是由程序员用一种或多种

通用的程序设计语言编写的,但是近年来出现了多种面向用户的数据库应用程序开发工具,这些工具可以简化使用 DBMS 的过程,并且不需要专门编程。Delphi 就是一种强有力的数据库应用程序开发工具。

用来生成数据库应用程序的语言主要分为五大类型:过程化语言、结构化查询语言(SQL)、面向对象语言(OOL)、“宏”语言和范例查询语言(QBE)。

1. 过程化语言

标准的计算机程序设计语言如 Pascal、C 和 Basic 都是过程化语言,这些语言可以通过某种应用程序接口(API)来创建数据库应用程序,这种 API 由一组标准的函数(或调用)组成,这些函数和调用扩展了语言的功能,使之能访问数据库中的数据。当程序设计人员用过程化语言创建数据库应用时,必须把应用的代码编写成一系列的过程,每个过程执行一部分的工作。

上述过程化语言一般用来创建非数据库应用程序,它们通常称为“第三代程序设计语言”(3GL),还有一些过程化语言是某种特定的 DBMS 专用的,这些语言一般称为“第四代程序设计语言”(4GL),即数据库专用语言。常用的数据库专用的过程化语言有 dBase 语言、Paradox 数据库的 PAL 语言等。

2. 结构化查询语言

结构化查询语言 SQL (Structure Query Language) 是基于关系模型的数据库查询语言,它是一种非过程化的程序语言,也就是说,没有必要写出将如何做某事,只需写出做到什么就可以了。写出的语句可看作是一个问题,称为“查询”(Query),针对这个查询,得到所需的查询结果。下面是一个例子:

```
Select StuID , Name from Student Where StuID = 1019
```

这个查询的意思是从数据库表 Student 中将学生证号(StuID)等于 1019 的学生选出来,并列出的学生证号(StuID)和姓名(Name)。

把 SQL 描述为子语言更合适一些,因为它没有任何屏幕处理或用户输入/输出的能力,它主要目的是为了提供访问数据库的标准方法,而不管数据库应用的其余部分使用什么语言编写,它既是为数据库的交互式查询而设计的(因此被称为动态 SQL),同时也可在过程化语言编写的数据库应用程序中使用(因此被称为嵌入式 SQL)。

3. 面向对象语言

在用于开发数据库应用程序的语言中,还可以使用目前最常见的面向对象程序设计语言,如 C++、Object Pascal 等, OOP 代表了一种完全不同的程序设计方法,在这种程序设计方法中,活动定义为“对象”上发生的操作,而不是作为一系列过程来定义的。在数据库应用程序中使用 OOP 语言的情况正在不断增加。

4. “宏”语言

“宏”语言不是一种完全的程序设计语言,它实际上是一个用户手工输入的表,这个表被输入到应用程序中,以便自动执行一定的任务。对于某个特定应用的高级语言,宏语言通常可以在低档 DBMS 软件中或数据库服务器的前端中找到。

5. 范例查询语言

范例查询语言 QBE (Query By Example) 严格地讲不是一种语言,它是面向用户提供的的一个或多个空表界面,这些空表对应与数据库中的表。用户可以通过键盘选择需要查

询的列,并在适当的列中填入条件从而定义查询的检索条件,然后 DBMS 就把 QBE 转换成相应的动作,以完成用户要求的查询任务。

1.2 Delphi 数据库体系结构

直到目前为止,计算机软件的开发分为两个不同的体系,其中一个体系是使用传统的程序设计语言(如 Pascal、Basic 和 C 等)开发数值控制、数值运算等软件,围绕它们的重点是算术、数据结构以及近年产生的面向对象技术;另一个体系则是通用的数据库工具,我们根本不需要编写任何 Object Pascal 代码便可以创建一个简单的数据库应用。

Delphi 是 Borland 公司于 1994 年底发布的用于开发数据库应用程序的工具,它是面向对象的,它是目前开发客户/服务器数据库应用程序的强有力的工具。Delphi 在 Window3.1 以上版本的系统环境下运行,目前具有两个版本:Delphi 的标准版本和客户/服务器版本。标准版本包含一个 BDE(Borland Database Engine)的局部拷贝,它允许用户创建能访问 dBase、Paradox 和 Local InterBase 服务器的数据库应用,它还支持具有 ODBC 接口的数据库。Delphi 的客户/服务器版本包括 Borland SQL Link,它能直接访问 ORACLE、SyBase 和 Microsoft SQL Server,Informix 以及 InterBase 数据库服务器。

Delphi 是可以访问多种数据库管理系统的数据库,凭借窗体(Forms)和报表(Reports),BDE 可以访问诸如 Paradox、dBase、本地 InterBase 服务器的数据库,也可以访问远程数据库服务器上的数据库(如:ORACLE、SyBase、Informix 等客户/服务器数据库中的数据库),或任何经 ODBC(Open Database Connectivity)可访问的数据库管理系统中的数据库。

Delphi 使用可视化的组件创建数据库应用,跟创建其他的非数据库应用程序一样,数据库组件都必须具备一定的属性,程序设计人员可以在涉及过程中设置组件的多样属性,也可以在程序运行过程中通过程序来设置组件的各种属性。

在 Delphi 中访问和操纵数据库最容易的途径是使用所提供的数据库可视组件。数据库可视组件存储在可视部件库的两个页面上,也就是控制面板上的数据访问组件页(Data Access)和数据控制组件页(Data Control)。

Delphi 组件面板中的这两个页面是专门为开发数据库应用程序创建的。

数据访问组件页上的组件主要用于说明有关的数据库的信息,如应用程序要访问(连接)的数据库,要访问数据库中的具体的数据库表,以及要访问表中的哪些字段等,在实际的开发应用中常用的组件有 TDataSource、TTable、TQuery 等。

数据控制组件页上的组件主要用于显示浏览数据库中的数据信息,为用户提供了一个可视化的界面,常用的组件有:TDGrid、TDBEdit、TDBCheck 等,可以让用户对数据库中的信息进行有效地浏览、编辑、插入和删除等操作。

TTable、TQuery、TStoredProc 组件负责与实际的数据库表联系,并从中获取数据信息,因而它们又常常被称为数据集组件,在程序设计过程中是可见的,但在程序运行时是不可见的,它们通过 BDE 为应用程序提供与数据库的连接,数据控制组件通过 TDataSource 组件与数据集相连,为用户提供一个可视化的界面,并在其中显示数据库中的数据信息。