

# 第1章 Delphi7概述

本章主要内容:

- Delphi的历史
- Delphi7安装
- Delphi7集成开发环境
- Delphi7程序设计简介
- Delphi7的文件类型

21世纪职业教育规划教材

# Delphi程序设计基础

李文池 王佳祥 主 编  
张金莲 魏 虹 副主编  
王中生 主 审

ISBN 7-5084-3956-2

中国水利水电出版社

# 1.1 Delphi的历史

Delphi是第四代编程语言，是RAD（Rapid Application Development，快速应用程序开发）工具的代表。从核心上说，Delphi是一个Pascal编译器。

Delphi语言的发展历程如下：

1995年02月14日，Delphi1.0发布，号称 VB Killer。

1996年，Delphi2发布。

1997年，Borland公司推出了Delphi3；

1998年，推出的Delphi4 致力于使Delphi更易于使用；

1999年，Delphi5推出；

2001年6月，Borland公司推出了Delphi6；

2002年8月，Borland公司推出了Delphi7；

# 1.2 Delphi7的安装

## 1.2.1 安装Delphi7企业版的系统要求

- (1) Intel Pentium 166 MHz或配置更高的处理器;
- (2) 128Mb以上内存;
- (3) 完全安装企业版大约要占475Mb硬盘空间。
- (4) Microsoft Windows 98、2000、XP或更高版本的操作系统平台。
- (5) 此外，还要求系统配有CD-ROM 驱动器、VGA或性能更高的彩色显示器，鼠标等外设。

# 1.2 Delphi7的安装

## 1.2.2 Delph7企业版的安装



Delphi7安装界面

# 1.2 Delphi7的安装

## 【Delphi7】

就是Delphi7的集成开发环境。

## 【InterBase 6.5 Server】

这是Borland公司随Delphi一起发布的数据库服务器Local Server，也是一种大型SQL数据库，具有SQL数据库（如SQL Server、Oracle、DB2等）的大部分功能。

## 【InterBase 6.5 Desktop Edition】

提供了InterBase 6.5的一些管理工具，使程序员可以轻松构建和管理InterBase 6服务器。

## 【Remote Debugger Server】

远程调试服务器。

## 【ModelMaker 6.20】

提供了一种崭新的类和构件包的开发模式，在编写构件时只要利用这个工具将要设计的构件以框图的形式进行概念搭建，就可以自动生成所需的代码。

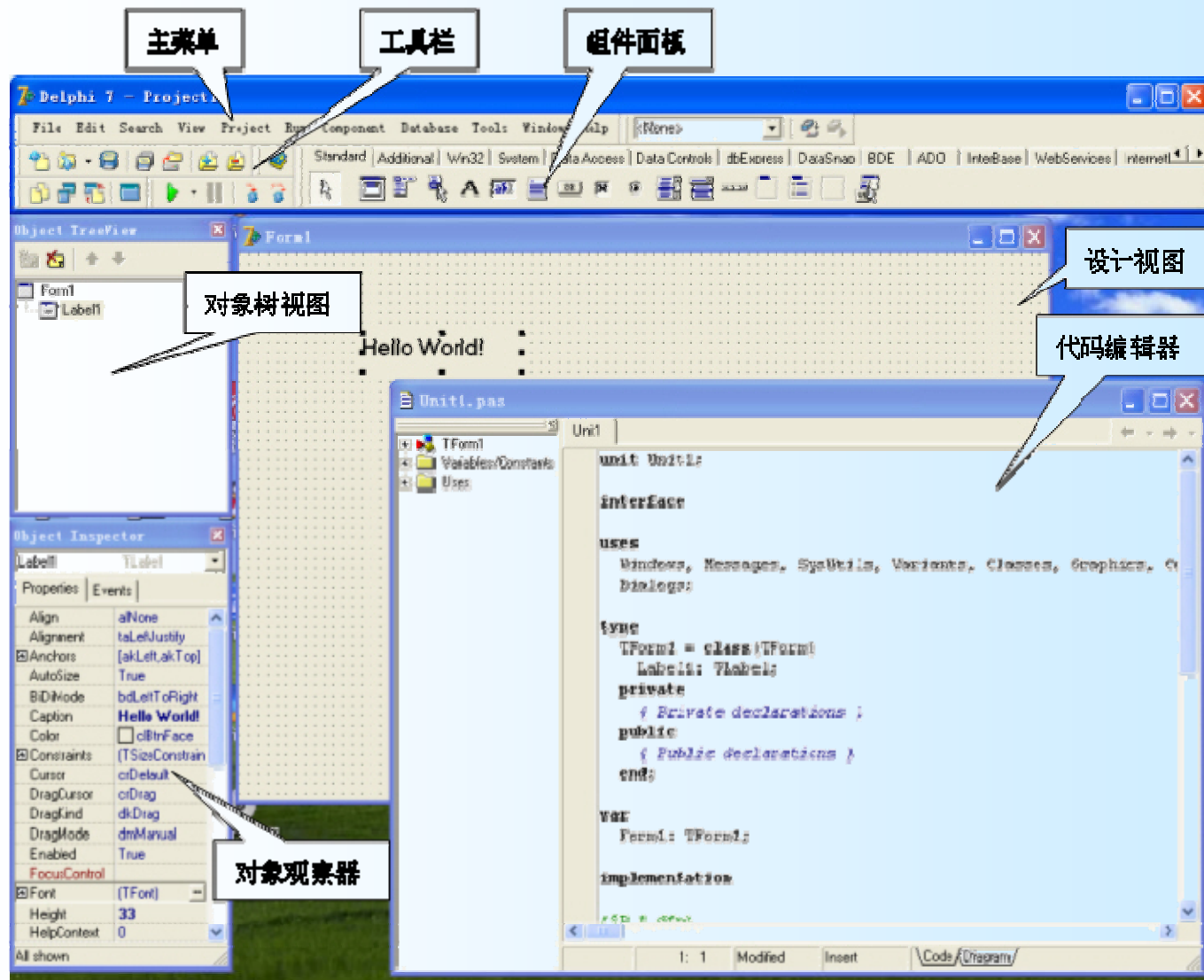
## 1.2 Delphi7的安装

---

### 【InstallShield Express】

InstallShield公司为Delphi 7度身定做的安装文件制作软件。

# 1.3 Delphi7集成开发环境



# 1.3 Delphi7集成开发环境

## 1.3.1 主窗口

### 1. 系统菜单

系统菜单是下拉式菜单，提供了**Delphi 7**集成开发环境中开发应用程序所需要的各种功能。

### 2. 工具栏

工具栏位于主窗口的左下端，由两排工具按钮组成，这些按钮是系统菜单命令的快捷方式，各种图标直观地表示了它能执行的动作。

（1）工具栏显示

（2）工具栏按钮的增删

### 3. 组件面板

组件面板包含了**Delphi**的可视化组件，例如，按钮、列表框、编辑框等。组件面板由若干组件页组成，利用它来选择需要的组件并将它放到窗体中去。

# 1.3 Delphi7集成开发环境

## 1.3.2 设计视图

设计视图是开展大部分设计工作的区域。首次启动Delphi7时，系统自动创建一个普通的应用程序项目，并会创建一个默认窗体Form1。

窗体相当于组件的容器，可以把组件放在窗体中，通过鼠标拖动操作来移动组件位置和改变尺寸，可随心所欲地安排它们，以此来开发应用程序的用户界面。

窗体上有网格（**Grids**），放置组件时网格可以用于定位，在程序运行时网格是不可见的。

## 1.3.3 代码编辑器（Code Editor）

在默认情况下，代码编辑器隐藏在设计视图之下，在代码编辑器和设计视图之间进行切换可以按F12键。

# 1.3 Delphi7集成开发环境

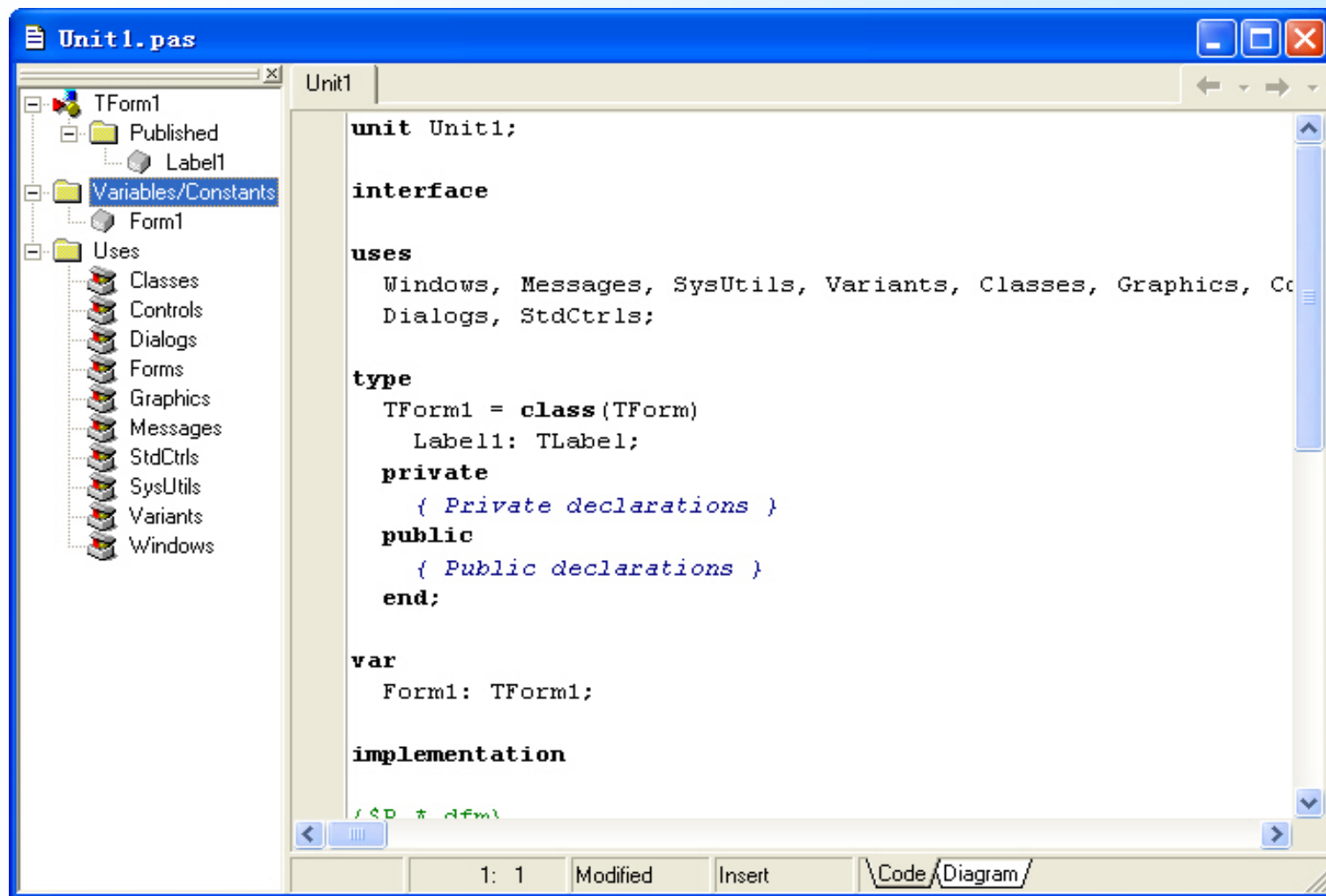


图1.3 Delphi7的代码编辑器

# 1.3 Delphi7集成开发环境

Delphi7提供了以下快捷的代码编辑功能:

## 1. 程序调试功能

如果在程序编译中发生错误或产生警告，会在代码编辑器下方“**Message**”窗口显示相关的错误、警告信息，点击某条信息，光标就会移动到代码中相应的行。

## 2. 帮助查询功能

当程序员对代码中的某个组件或关键字不清楚时，只需要将光标移到该单词上，然后按【**F1**】键，就会自动打开帮助，并显示相关内容。

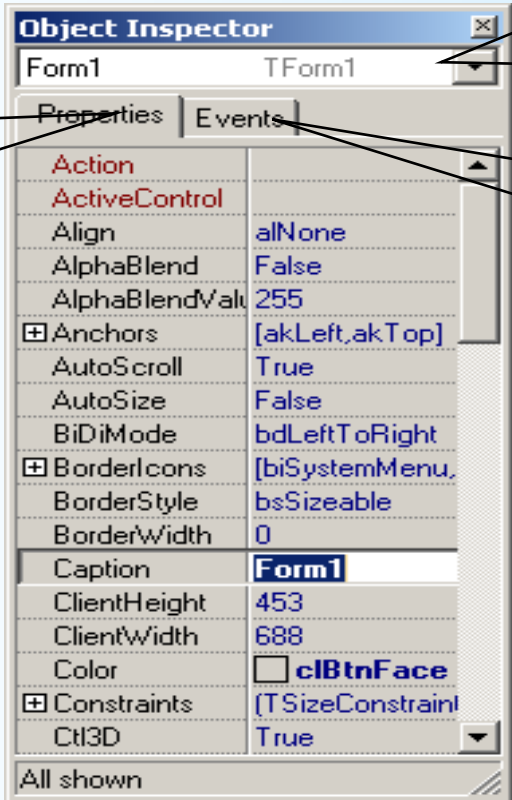
## 3. 代码分析

(1) Class Completion

(2) Code Insight

# 1.3 Delphi7集成开发环境

## 1.3.4 对象观察器（Object Inspector）



The screenshot shows the Object Inspector window in Delphi7. It has a title bar 'Object Inspector' and a close button. Below the title bar, there are two tabs: 'Form1' and 'TForm1'. The 'Properties' tab is selected, showing a list of properties for the selected object. The 'Events' tab is also visible. Callouts point to specific parts of the window:

- 对象选择器:** 用于选择要操作的组件。 (Object Selector: Used to select the component to be operated.)
- 属性选项卡:** 用来观察、设置窗体及其上的组件设计时的属性 (Properties tab: Used to observe and set the design-time properties of the form and its components.)
- 事件选项卡:** 列出组件能响应的各种事件 (Events tab: Lists the various events that the component can respond to.)

Property	Value
Action	
ActiveControl	
Align	alNone
AlphaBlend	False
AlphaBlendValue	255
⊕ Anchors	[akLeft,akTop]
AutoScroll	True
AutoSize	False
BiDiMode	bdLeftToRight
⊕ BorderIcons	[biSystemMenu,
BorderStyle	bsSizeable
BorderWidth	0
Caption	Form1
ClientHeight	453
ClientWidth	688
Color	<input type="checkbox"/> clBtnFace
⊕ Constraints	[TSizeConstrain
Ctl3D	True

All shown

# 1.3 Delphi7集成开发环境

## 1.3.5 对象树视图（Object TreeView）

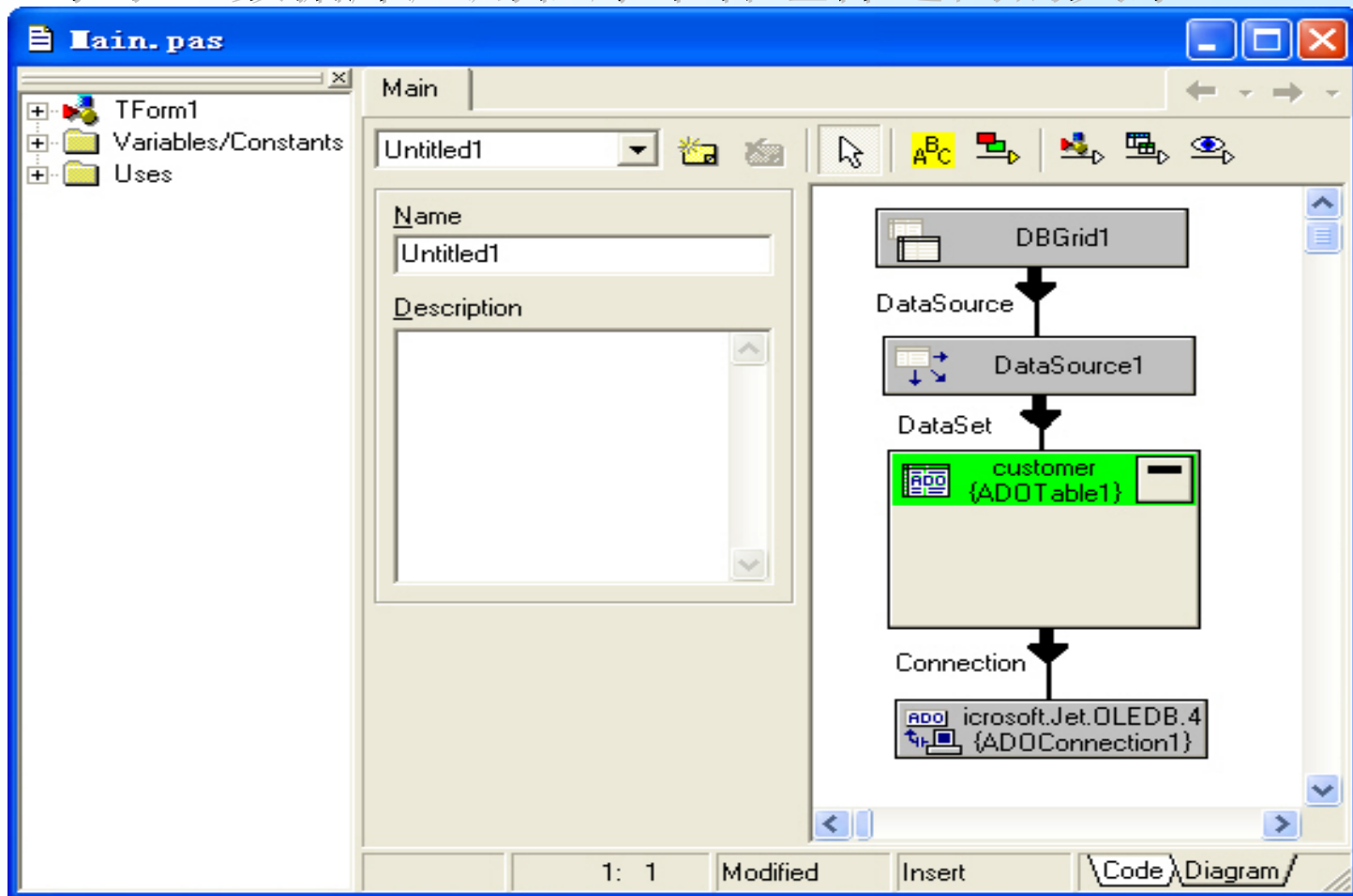
对象树视图除了可以显示窗体中所有的对象之外，还可以用树形结构表达组件之间的包含关系。当程序员在 **Object TreeView** 窗口中选择一个组件之后，这个组件会立刻出现在对象观查器中，程序员可以改变这个对象的属性值和添加事件处理过程。

当窗体中放置了大量的组件时，很难用鼠标直接选择对象，这时通过 **Object TreeView** 窗口可以很方便选择要找的对象，并且能看到和它相关的组件。

# 1.3 Delphi7集成开发环境

## 1.3.6 对象图表（Diagram）

对象图表是代码编辑器中的一个页面（Diagram）。下图显示了一数据库应用程序中各组件之间的关系。



# 1.4 Delphi7程序设计简介

## 【例1-1】

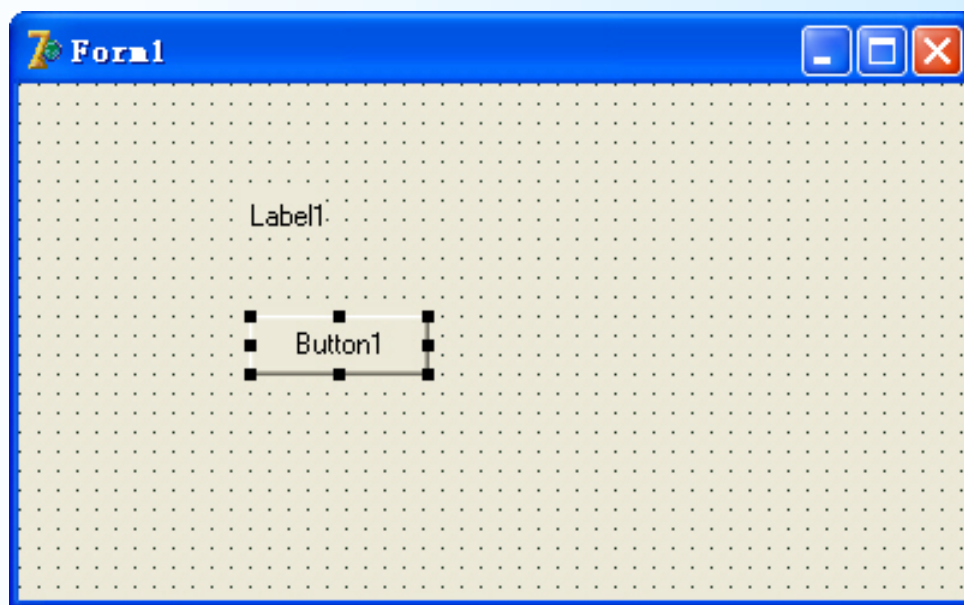
编写一个应用程序，在窗体中显示“Hello World!”，当点击其下的按钮时可以交替显示和隐藏“Hello World!”。运行时显示的界面如图1-6所示。



# 1.4 Delphi7程序设计简介

## 【实现步骤】

- (1) 启动Delphi 7
- (2) 新建应用程序
- (3) 定制窗体



# 1.4 Delphi7程序设计简介

表1-1 标签组件的属性设置

属 性	属 性 值	说 明
Alignment	taCenter	设置标签上显示的文字水平居中
AutoSize	False	使标签不根据Caption的长度自动改变宽度。
Caption	Hello,World!	在标签上显示的字符串
Font	字体： Arial， 大小： 36	设置标签显示的文字效果
Name	Label1	为标签对象指定名称，便于在程序中调用。

(4) 编写代码

(5) 运行程序

# 1.5 Delphi程序的基本结构

## 1.5.1 项目和项目文件

Delphi中，应用程序的所有相关文件都被组织在一个项目中，每个项目包含的文件很多，大部分是由Delphi自动创建并维护的。

每一个Delphi应用程序都有一个扩展名为dpr的项目文件，是由Delphi自动建立并维护的，一般情况下不必修改它的内容

缺省生成的项目源文件代码如下：

```
program Project1;  
uses  
    Forms,  
    Unit1 in 'Unit1.pas' {Form1};  
{$R *.res}
```

# 1.5 Delphi程序的基本结构

```
begin  
    Application.Initialize;  
    Application.CreateForm(TForm1, Form1);  
    Application.Run;  
end.
```

## 1.5.2 窗体文件

窗体在设计阶段可以用来放置各种组件，在运行阶段是与用户交互的界面。

窗体中的所有信息保存在两个主名相同扩展名不同的文件中，一个是扩展名为**dfm**的窗体文件，另一个是每个窗体对应的同名单元文件。

# 1.5 Delphi程序的基本结构

## 1.5.3 单元文件

单元文件保存了Delphi程序的基本模块，一般的单元文件都与一个窗体对应，包含了窗体及其组件的事件处理程序，在Delphi中编写的程序代码，绝大多数被保存在这种文件中，其扩展名为.pas。

缺省生成的窗体单元文件的源代码如下：

```
unit Unit1;    //单元文件的名字

interface    //接口部分的开始

uses          //引用的标准单元文件
    Windows, Messages, SysUtils,
    Variants, Classes,
    Graphics, Controls, Forms, Dialogs;
```

# 1.5 Delphi程序的基本结构

```
type          //类型声明
  TForm1 = class(TForm)
  private     //声明私有成员
    { Private declarations }
```

```
public        //声明公有成员
  { Public declarations }
end;          //结束类型声明
```

```
var           //声明变量或类的实例
  Form1: TForm1;
```

```
implementation //程序代码实现功能部分的开始
```

```
{ $R *.dfm }    //通过编译指令 $R 链接窗体文件
```

# 1.5 Delphi程序的基本结构

---

end.           //实现部分结束

# 1.5 Delphi程序的基本结构

## 1.5.4 Delphi7的文件类型

文件扩展名	文件类型说明	产生时间
BMP、ICO、CUR	位图、图标及光标图像文件	程序设计时
BGP	项目组文件，由多目标项目管理器产生	程序设计时
BPL	BORLAND PACKAGE LIBRARY（组件库文件）	编译连接后
CBA	压缩格式文件，做WEB发布时使用	设计时
CFG	项目配置文件。项目配置文件保存着项目的配置信息	设计时
DCP	DELPHI COMPONENT PACKAGE（Delphi组件包）	编译时
DCU	DELPHI COMPILED UNIT，编译原始文件后的中间产物	编译时
DFM	DELPHI FORM FILE（窗体文件）	程序设计时
~DFM	DFM的备份文件	程序设计时
DLL	DYNAMIC LINK LIBRARY（动态链接库文件）	编译连接时
DOF	DELPHI OPTION FILE，设计多语言项目时使用的语言翻译配置文件，多语言项目中每个窗体的每一种语言都有一个DNF文件	程序设计时
DPK	DELPHI PACKAGE，软件包项目的源代码文件	程序设计时
DPR	项目文件	程序设计时
~DPR	DPR的备份文件	程序设计时

# 1.5 Delphi程序的基本结构

文件扩展名	文件类型说明	产生时间
DSK	DESKTOP FILE, 保存现在DELPHI视窗的位置、正在编辑的文件以及其它桌面的设定文件	程序设计时
LIC	OCX文件相关的授权文件	编译连接时
OCX	OLE控件文件, 是一特殊的DLL文件可包含ACTIVEX控件或窗体	编译连接时
PAS	DELPHI源代码文件	程序设计时
~PAS	PAS的备份文件	程序设计时
RES、RC	项目的资源文件,包含项目的图标、光标及字体等信息	程序设计时
EXE	可执行文件	编译连接时
TLB	类型库文件	程序设计时

# 本章实训指导

---

1. 了解IDC集成开发环境的组成与功能，重点熟悉设计视图、代码编辑器和对象观察器；
2. 掌握应用程序的创建、打开、保存和编译运行操作；
3. 理解项目（project）的概念，了解Delphi中的文件类型；
4. 模仿本章【例1-1】创建一个应用程序，并查看项目文件夹下Delphi都建立了哪些文件，以及有些什么类型的文件。

## 第2章 Delphi语言基础

本章主要内容:

- 常量、变量、数据类型
- 程序语句
- 过程与函数
- 面向对象的编程

# 2.1 标识符和保留字

## 2.1.1 标识符

标识符用作常量、变量、数据类型、过程、函数、单元及程序等的名称。标识符由一个或多个**ASCII**码字符序列组成，定义标识的规则如下：

- (1) 标识符由字母、数字或下划线组成；
- (2) 标识符的第一个字符必须是字母或下划线；
- (3) 标识符的长度不应超过**255**个字符，超过**255**个字符只有前**255**个字符有效；
- (4) 不能将关键字（保留字）用作标识符；
- (5) 标识符不区分大、小写。

## 2.1.2 保留字

保留字又称为关键字，它在**Delphi**语言中有着特殊含义。保留字不能用作标识符，在实际编程中不应该把任何保留字用作标识符。

# 2.1 标识符和保留字

## Delphi中使用的关键字

and	array	As	asm	begin
Case	Class	Const	constructor	destructor
Dispinterface	Div	Do	downto	Else
End	except	exports	File	Finalization
Finally	for	function	Goto	If
Implementation	In	inherited	initialization	inline
Interface	is	label	library	Mod
Nil	not	object	Of	or
out	packed	procedure	program	property
raise	record	repeat	resourcestring	set
Shl	shr	string	then	threadvar
To	try	type	unit	until
uses	var	while	With	xor

## 2.2 常量与变量

### 2.2.1 常量

对于在程序运行期间保持不变的数据，**Delphi**允许通过声明常量来调用。声明常量不必指定数据类型，但需指定常量所代表的数据的值。

常量的声明格式如下：

**Const** 常量名=表达式;

例如：

**CONST**

Thousand = 1000;

Pi = 3.14159;

ErrMsg = '类型错误';

**Delphi** 根据常量的值来决定它的数据类型。

## 2.2 常量与变量

### 2.2.2 变量

变量用于在程序执行过程临时存放数据，其值可以被改变。变量分全局变量和局部变量。

变量说明的一般形式为：

**VAR**

变量名列表:类型名;

例如：

**VAR**

```
iCount: Integer;    //说明了一个整型变量
bCorrect: Boolean;  //说明了一个布尔型变量
cX, cY: Char;       //说明了两个字符型变量
```

## 2.3 数据类型

类型大致可以分为简单类型、字符串类型、结构类型、指针类型、过程类型和变体类型。简单类型又分为有序类型和实数类型。

### 2.3.1 有序类型

有序类型定义一个有次序的数值集合，除了它的第一个值以外，其它每个值都有一个唯一的前驱值；除了最后一个外，其它每个值都有一个唯一的后继值。并且，每个值都有一个序数决定它在这个类型中的位置

#### 1. 整型

整型是整数的一个子集。整型有通用整型和基本整型之分。

#### 2. 字符型

字符型的数据只能是单个字符，不能是一串字符。

### 3. 布尔型

布尔类型的标识符为**Boolean**。布尔型变量的取值仅有**False**和**True**两个值。

### 4. 枚举类型

枚举类型是一种自定义有序类型。在枚举类型中列出了所有该类型可能的取值，而不是指定现有类型的范围。

下例定义了一个枚举类型**TWeekDay**来表示一周中的七天，并说明了一个变量**WeekDay**为: **TWeekDay**类型。

**TYPE**

**TWeekDay = (Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday);**

**VAR**

**WeekDay: TWeekDay;**

## 2.3 数据类型

### 5. 子界类型

子界类型定义了某种类型的取值范围。语法如下：

**TYPE**

类型标识符 = 上界值 .. 下界值;

### 2.3.2 实数类型

实数类型定义了一类可以用浮点数表示的数字。

### 2.3.3 日期时间类型

**TDateTime** 不是编译器可直接识别的预定义类型，它在 **System** 单元定义：

**TYPE**

**TDateTime** = type Double;

### 2.3.4. 字符串类型

字符串表示一个字符序列。

## 2.3 数据类型

Delphi共有三种字符串类型：

- (1) ShortString
- (2) ANSIStrng
- (3) WideString

### 2.3.5 结构类型

结构类型的一个实例可包含多个值。

#### 1. 集合

集合类型表示一组值，该组值由集合所依据的有序类型定义。

例如：

**TYPE**

**TInts = 1..10;**

**TIntSet = SET OF TInts;**

## 2.3 数据类型

它声明一个叫做**TIntSet**的集合类型，它的值是从1 到10 之间所有可能的选择。

### 2. 数组

数组类型定义了一组指定类型的元素序列，在方括号中填入下标值就可访问数组中的元素。定义数组时，方括号也用来指定可能的下标值。

**VAR**

```
myArray: ARRAY [1..10] of Integer;
```

### 3. 记录类型

记录类型用于定义不同类型数据项的固定集合。记录中每个字段有它自己的类型。记录类型定义中列出了所有字段，每个字段对应一个字段名，通过字段名可以访问字段值。

## 2.3 数据类型

下面简单列举了记录类型的定义、类型变量的声明以及这类变量的使用：

```
TYPE
```

```
  TDate = record
```

```
    Year: Integer;
```

```
    Month: Byte;
```

```
    Day: Byte;
```

```
  end;
```

```
VAR
```

```
  BirthDay: TDate;
```

```
BEGIN
```

```
  BirthDay.Year := 1997;
```

```
  BirthDay.Month := 2;
```

```
  BirthDay.Day := 14;
```

```
...
```

```
END
```

## 2.3 数据类型

### 4. 过程类型

过程类型允许把过程和函数作为“值”看待，它可以赋给变量或传给其它过程和函数。比如，假设定义了一个叫做**Calc**的函数，它有两个整型参数并返回一个整数值：

```
FUNCTION Calc(X,Y: Integer): Integer;
```

说明变量**F**为过程类型后，可以把**Calc** 函数赋给变量**F**：

```
VAR    F: function(X,Y: Integer): Integer;
```

```
F := Calc;
```

在说明过程类型时，只取过程或函数头（**heading**）并把**procedure** 或**function** 后面的标志符去掉，剩下的就是过程类型的名称。变量**F**实际是指向函数的指针。

## 2.3 数据类型

### 5、指针类型

指针类型的变量存储的是内存地址，利用指针可以灵活地访问内存中的数据。

指针类型的声明格式如下：

**type** 指针类型标识符 = ^数据类型;

指针可以指向任何数据类型。

## 2.4 运算符与表达式

运算是对于数据进行计算的过程，记述各种不同运算的符号称为运算符。根据运算规则，用运算符将常量、变量、数值和函数组合起来就形成表达式，表达式运算的结果就是表达式的返回值。表达式可以传递给过程或函数的值参，但不能传递给过程或函数中的引用参数。

### 2.1.1 表达式

最简单的表达式是变量和常量，更复杂的表达式由简单表达式使用运算符、函数调用、集合构造器、索引和类型转换构成。

例如：

X	//变量
15	//整数常量
abs(X)	//函数调用
X * Y	//X和Y 的乘积

## 2.4 运算符与表达式

`X>=Y`      //条件表达式

`['a','b','c']` { 集合 }

`Char(48)` { 类型转换 }

### 2.4.2 Delphi语言中的运算符及其优先级

#### 1. 赋值运算符（:=）

赋值运算符“:=”是先计算赋值运算符右边表达式的值，再将结果赋给左边的变量。

#### 2. 算术运算符

`+`、`-`、`*`、`/`、`Div`、`Mod`

#### 3. 关系运算符

关系运算符是对两个类型相容且可以比较大小的数据进行比较，结果为布尔类型。

`=`等于、`<>`不等于、`<`小于、`>`大于、`<=`小于或等于、`>=`大于或等于、`In`属于。

## 2.4 运算符与表达式

### 4. 逻辑运算符

逻辑运算符对逻辑类型的操作数进行运算，结果为布尔型。

**not**逻辑非、**and**逻辑与、**or**逻辑或、**xor**逻辑异或。

### 5. 集合运算符

**+**集合的并集、**-**集合的差集、**\***集合的交集、**<=**、**>=**、**=**两个集合是否相等、**<>**两个集合是否不相等、**in**。

### 6. 运算符的优先级

各运算符的优先级

运算符	优先级
not	1（最高）
*, /, div, mod, and	2
+, -, or, xor	3
关系运算符	4（最低）

## 2.5 语句

语句用于控制程序的执行，即什么情况下执行什么样的操作。

### 2.5.1 注释与空白

#### 1. 注释

注释是添加在程序中用来说明代码功能的语句，它是非执行语句，对程序的执行部分不会产生任何影响，有助于提高程序的可读性。**elphi**中的注释有三种形式：

花括号“{}”注释：位于“{}”中的内容为注释。

双斜杠“//”注释：“//”之后到本行结束的内容为注释。

圆括号及星号对“(\*.....\*)”注释：(\*.....\*)中的内容为注释。

## 2.5 语句

### 2. 空白

(1)空行

(2)空格

(3)缩进

### 2.5.2 简单语句和复合语句

#### 1. 简单语句

赋值语句的格式如下：

变量名 := 表达式;

#### 2. 复合语句

begin

i:=1;

j:=i+1;

s:='abc';

end;

## 2.5 语句

### 3. with语句

**with**语句是一种简写方式，用来引用一个记录的字段，或一个对象的字段、属性和方法。

例如：

```
with label1 do begin    //label1是一个TLabel组件对象，
                        //常用作显示标签
    caption:='Red';      //将label1的caption属性设为'Red'。
    font.Color:=clRed;   //将label1的font.Color属性设为
                        //clRed。
end;
```

### 2.5.3 条件语句

条件语句通过条件检测，判断是否执行该条件语句中包含的语句。

## 2.5 语句

### 1. if语句

(1) if...then 语句

语法:

if 表达式 then 语句

例如:

if  $x \geq y$  then  $z := x$ ;

(2) if...then...else语句

语法:

if 表达式 then 语句1 else 语句2

例如:

**if  $x \geq y$  then**

**z:=x**

**else**

**z:=y;**

## 2.5 语句

if语句可以嵌套，当使用复合语句时，复合语句前后需要用begin和end括起来。

例如：

```
if x >= y then
  begin
    z := x;
    Count := Count + 1;
  end
else
  if Count = Last then
    Done := True
  else
    Exit;
```

## 2.5 语句

### 【例2-1】

编写一个程序，用于判断输入的年份是不是闰年。

注：是闰年的条件必须满足下列条件之一：

- (1) 能被4整除，但不能被100整除的年份；
- (2) 能被100整除，也能被400整除的年份。

### 【实现步骤】

- (1) 新建应用程序
- (2) 定制窗体



图2-1设计窗体界面

## 2.5 语句

表2-10 各组件属性设置

组件名	属 性	属 性 值
Label1	Caption	请输入年份:
Label2	Caption	显示结果
Edit1	Text	''
Button1	Caption	判断

(3) 编写代码

(4) 运行程序



图2-2 运行结果

## 2.5 语句

### 2. case语句

当选择的条件有多种可能性时，使用**case** 语句更为合适。**case** 语句包括用来选值的表达式、可能值序列或一个取值范围。这些值必须唯一，而且应属于有序类型。

其语法如下：

**case** 选择表达式 of

值1：语句1；

值2：语句2；

...

值n：语句n；

**End**

在**Case** 语句的末尾可以使用关键字**else**，当没有一个可选值与表达式匹配时，执行 **else** 后的语句。

## 2.5 语句

语句如下：

**case** 选择表达式 of

    值1：语句1；

    值2：语句2；

    ...

    值n：语句n；

**else**

    语句n+1；

**end**；

例如：

**case** MyChar of

    '+' : Text := 'Plus sign';

    '-' : Text := 'Minus sign';

    '\*, /': Text := 'Multiplication or division';

## 2.5 语句

```
'0'..'9': Text := 'Number';  
'a'..'z': Text := 'Lowercase character';  
'A'..'Z': Text := 'Uppercase character';  
else  
    Text := 'Unknown character';  
end;
```

### 2.5.4 循环控制语句

循环语句重复执行循环体（即语句或语句块），并根据设定的条件判断何时退出循环。**delphi**提供了三种循环语句：**while**语句、**repeat**语句、**for**语句。

#### 1. *while*语句

**while**的语法格式如下：

```
while 条件表达式 do  
    循环体；
```

## 2.5 语句

### 【例2-2】

编程计算 $1+2+3+\dots+99+100$ 的值；

### 【实现步骤】

- (1) 新建应用程序
- (2) 定制窗体

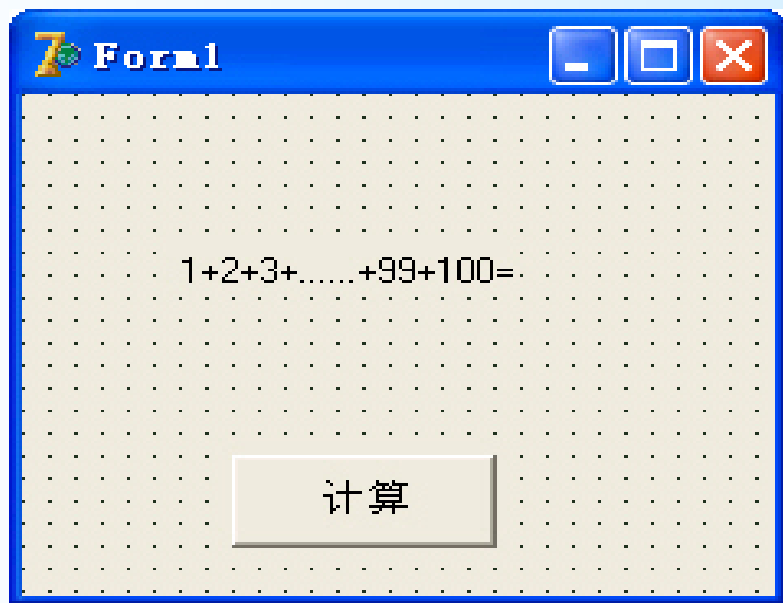


图2-3 设计窗体界面

## 2.5 语句

表2-11 各组件属性设置

组件名	属 性	属 性 值
Label1	Caption	1+2+3+...+99+100=
Button1	Caption	计算

(3) 编写代码

(4) 运行程序

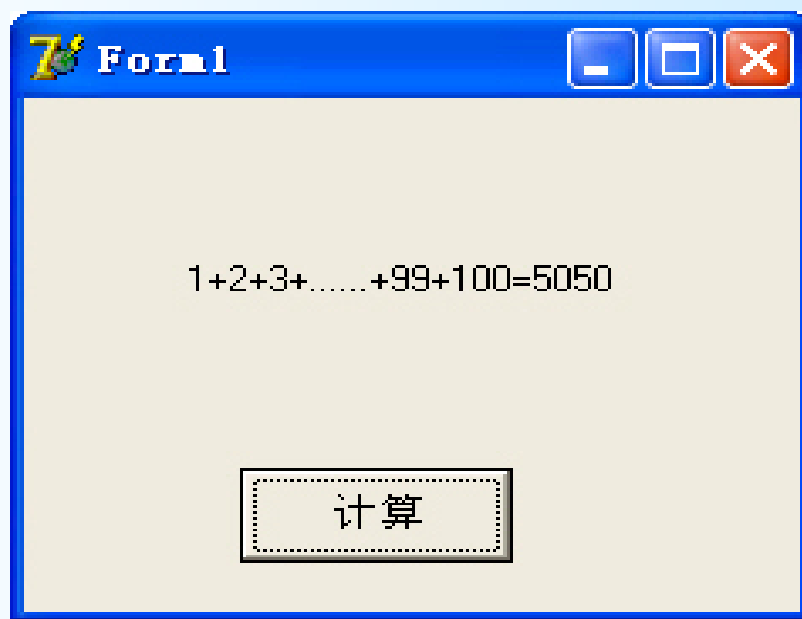


图2-4 运行结果

## 2.5 语句

### 2. repeat语句

**repeat**语句是先执行循环体，然后判断条件，根据条件决定是否继续循环，执行循环体的最少次数为1。

语法如下：

**repeat**

    循环体

**until** 条件表达式;

#### 【例2-3】

编写程序求两个正整数的最大公约数。

分析：我们使用“辗转相除法”求最大公约数，方法如下

:

以大数 $m$ 作被除数，小数 $n$ 作除数，相除得余数为 $r$ ；如果 $r$ 不为0 则将 $n$ 赋给 $m$ ， $r$ 赋给 $n$ ，相除得到新的 $r$ ，反复该过程直到 $r=0$ 。最后的 $m$ 就是最大公约数。

## 2.5 语句

### 【实现步骤】

- (1) 新建应用程序
- (2) 定制窗体

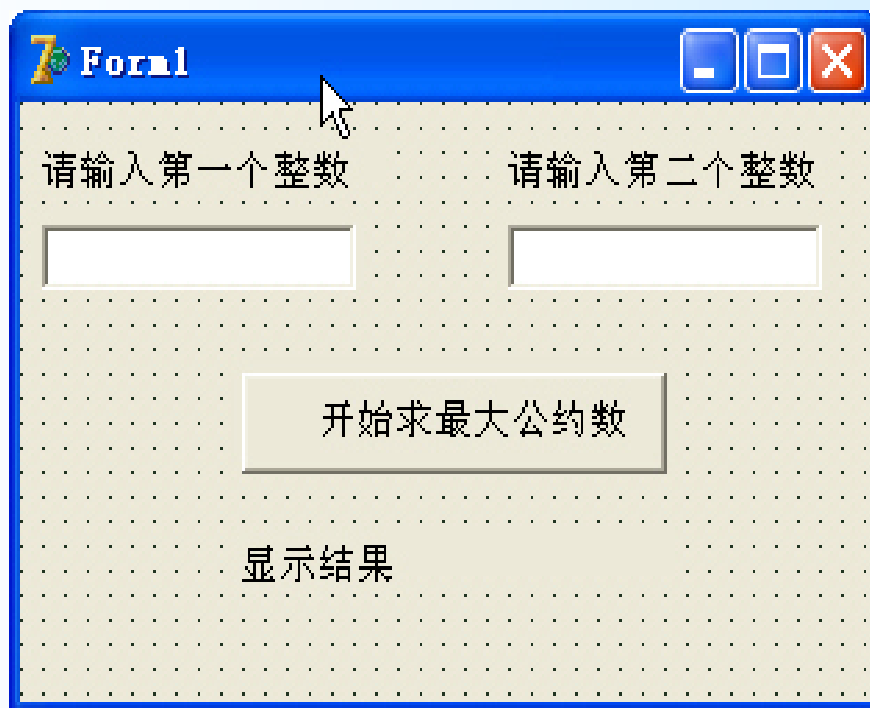


图2-5设计窗体界面

## 2.5 语句

表2-12 各组件属性设置

组件名	属 性	属 性 值
Label1	Caption	请输入第一个整数
Label2	Caption	请输入第二个整数
Label3	Caption	显示结果
Edit1	Text	"
Edit2	Text	"
Button1	Caption	开始求最大公约数

(3) 编写代码

(4) 运行程序

## 2.5 语句

The screenshot shows a standard Windows application window titled "Form1". Inside the window, there are two input fields for integers. The first field is labeled "请输入第一个整" (Please enter the first integer) and contains the value "124". The second field is labeled "请输入第二个整" (Please enter the second integer) and contains the value "432". Below these fields is a button labeled "开始求最大公约数" (Start calculating the greatest common divisor). At the bottom of the window, the result is displayed: "124与432的最大公约数是： 4" (The greatest common divisor of 124 and 432 is: 4).

图2-6运行结果

## 2.5 语句

### 3. for语句

在循环次数已知或可计算的场合，用**for**语句来实现循环比较简便。**for**语句分为递增式和递减式两种形式。

语法格式：

**for** 循环变量 = 初值 **to** 终值 **do**  
    循环体；

或

**for** 循环变量 = 初值 **downto** 终值 **do**  
    循环体；

- (1) 循环变量为循环计数器，只能为有序类型变量。
- (2) 初值和终值为循环变量的初始值和终止值。
- (3) **to**为计数递增，**downto**为计数递减。
- (4) 循环体可以是简单语句也可以是复合语句。

## 2.5 语句

(5) 对于递增循环，循环执行时先判断循环变量的值是否大于终值，若循环变量的值没有大于终值，则开始执行do后面的循环体，然后循环变量自动递增，并开始一个新的循环。

(6) 对于递减循环，循环执行时先判断循环变量的值是否小于终值，若循环变量的值没有小于终值，则开始执行do后面的循环体，然后循环变量自动递减，并开始一个新的循环。

### 【例2-4】

用for语句编程计算 $1+2+3+\dots+99+100$ 的值；

### 【实现步骤】

(1) 修改【例2-2】中的代码。

(2) 运行程序

## 2.6 过程与函数

过程和函数的主要区别在于过程定义使用保留字 **procedure**，执行后没有返回值，而函数定义使用保留字 **function**，执行后有返回值。

### 2.6.1 过程声明

过程声明的语法：

```
procedure 过程名(参数表);  
    局部声明部分  
begin  
    过程体语句块  
end;
```

## 2.6 过程与函数

### 2.6.2 函数声明

函数声明的语法:

```
function 函数名(参数表):返回值数据类型;  
    局部声明部分  
begin  
    函数体语句块  
end;
```

### 2.6.3 过程和函数调用

用户定义过程调用的形式与标准过程调用的形式是相同的, 用户定义函数调用的形式与标准函数调用的形式是相同的。过程调用的形式是将其作为一条单独的语句。函数也可以单独作为一条语句来调用, 也可以把函数作为表达式或表达式的一部分的形式进行调用。

## 2.6 过程与函数

### 【例2-5】

过程和函数的调用的例子。在程序中定义了一个名为showSum的过程，用于显示1到n的累加和；定义了一个名为intSum的函数，用于返回1到n的累加和。然后在程序中调用过程showSum和函数intSum。

### 【实现步骤】

- (1) 新建应用程序
- (2) 定制窗体

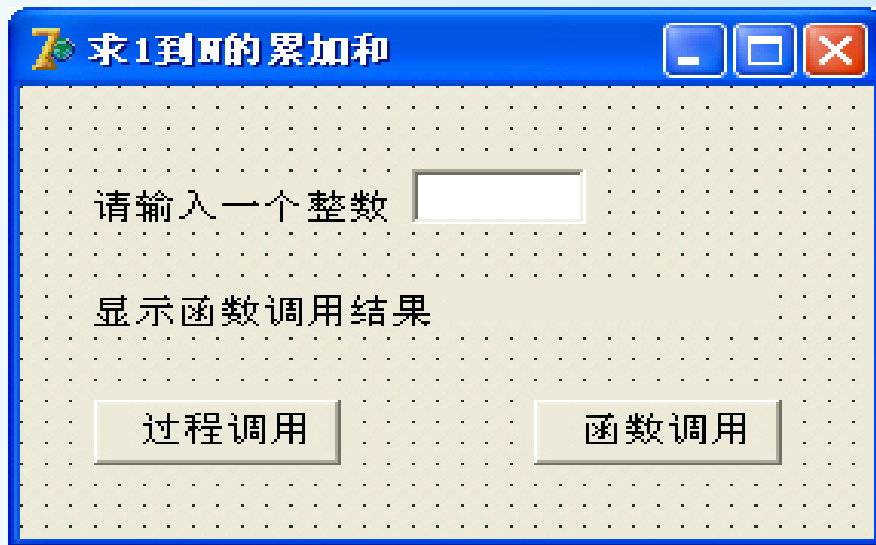


图2-7设计窗体界面

## 2.6 过程与函数

表2-13 各组件属性设置

组件名	属 性	属 性 值
Form1	Caption	求1到N的累加和
Label1	Caption	请输入第一个整数
Label2	Caption	显示函数调用结果
Edit1	Text	"
Button1	Caption	过程调用
Button1	Caption	函数调用

## 2.6 过程与函数

(3) 编写代码

(4) 运行程序

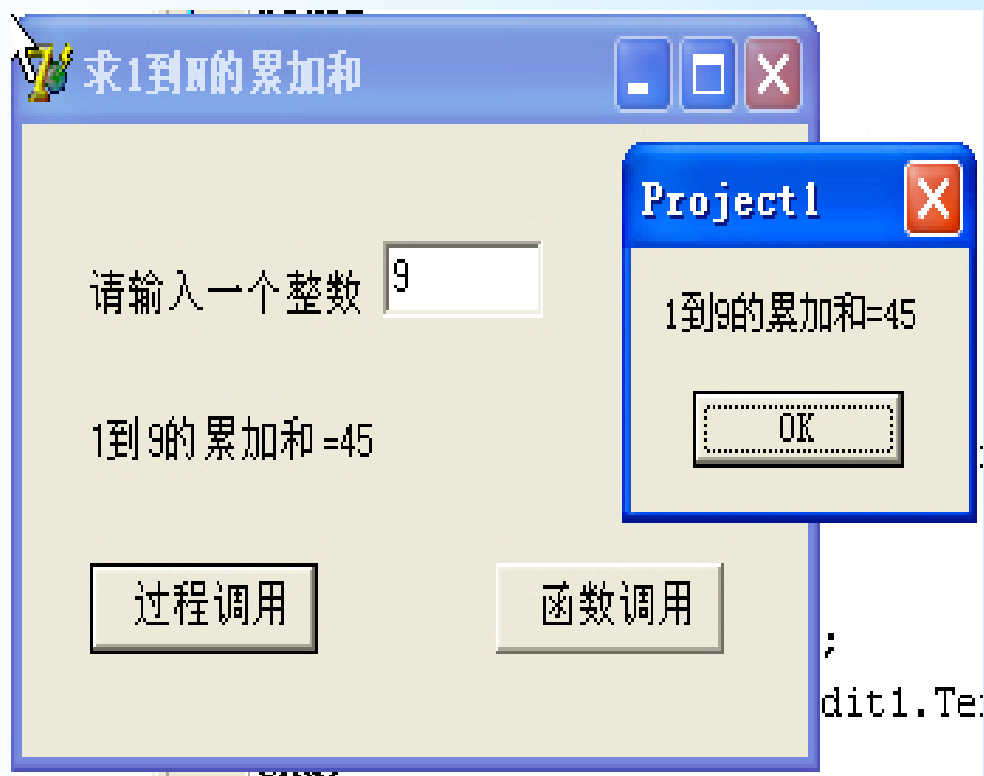


图2-8运行结果

## 2.7 面向对象的程序设计

### 2.7.1 类

类（**class**）描述了具有相似性质的一组对象，这组对象具有相同的数据结构，相同的操作，它定义了这组对象共同的属性和操作。类是一个抽象的概念，也称类类型，可以把类视为特殊数据类型。

#### 1. 类的定义：

声明类数据类型使用关键字**class**。语法如下：

**TYPE**

类名 = **Class**(父类)

成员列表

**END;**

## 2.7 面向对象的程序设计

类的成员可以是字段、方法和属性。

### (1) 字段

字段就是属于类的一个变量，它可以是任何类型，包括类类型（也就是说，字段可以存储对象的引用）。字段通常具有**private**属性。

给类定义字段非常简单，就像声明变量一样。字段声明必须出现在属性声明和方法声明之前。例如：

**FCapacity: Longint;**

### (2) 方法

方法是一个和类相关联的过程或函数。例如：

**procedure Clear;**

调用一个方法需指定它作用的对象（若是类方法，则指定类）。

## 2.7 面向对象的程序设计

例如：

```
stream.Clear;
```

### (3) 属性

属性用关键字**property**声明，它类似于字段，但又不同于字段，它常与读取和修改内部字段的方法相关联。

例如：

```
property Capacity: Longint read FCapacity write  
SetCapacity;
```

为属性指定了读取段的方法**FCapacity**和写字段的方法**SetCapacity**。**Longint**是属性的数据类型。

### 2. 类成员的访问权限

在类的声明中可以使用关键字**private**、**protected**、**public**、**published**、**automated**说明类成员的访问权限。访问权限决定了一个成员在哪些地方以及如何被访问。

## 2.7 面向对象的程序设计

### 3. 类的继承性

当声明一个类时，可以指定它的父类，例如：

```
type TSomeControl = class(TControl);
```

定义了一个叫做TSomeControl 的类，它继承自TControl。一个类自动从它的父类继承所有的成员，且可以声明新成员，也可以重新定义继承下来的成员，但不能删除祖先类定义的成员。

### 2.7.2 对象

#### 1. 对象的概念

对象（**Object**）是类的实例。通俗点说对象就是类类型的变量。对象拥有方法、属性和事件。

#### 2. 对象的构造

## 2.7 面向对象的程序设计

---

3. 对象的析构

4. 对象运算符

(1) **is**运算符

(2) **as**运算符

# 本章实训指导

1. 掌握常用数据类型;
2. 熟悉各种程序语句的用法;
3. 理解Delphi面向对象编程技术。
4. 超市为促销, 给出以下优惠措施
  - 购物100元以上, 9折优惠
  - 购物200元以上, 8折优惠
  - 购物300元以上, 7折优惠
  - 购物400元以上, 6折优惠
  - 购物500元以上, 5折优惠

编写相应程序完成输入购物款额, 计算出优惠价。

5. 如果一个数刚好出现在其平方的右边, 如5的平方为25, 6的平方为36, 这样的数我们称为同构数, 请编程找出1~9999的全部同构数。

# 第3章 窗体和组件

本章主要内容：

- 窗体
- 组件常用方法和属性
- 文本显示输入类组件的使用
- 按钮类组件的使用
- 列表类组件的使用

# 3.1 窗体和组件

## 3.1.1 窗体

认识窗体可以从了解窗体的属性、方法和事件入手。

窗体具有下面的一些重要属性：

(1)BorderIcons属性

(2)BorderStyle属性

(3)Caption属性

(4)Font属性

(5)FormStyle属性

从窗体类型的角度来看，Windows环境中的应用程序可以分为下面的三类：

- 多文档界面(MDI)应用程序
- 单文档界面(SDI)应用程序
- 对话框应用程序

## 3.1 窗体和组件

(6)Icon属性

(7)Name属性

(8)Position属性

(9)WindowState属性

### 3.1.2 组件

Delphi中有两类组件，可视化组件和非可视化组件。

### 3.1.3 组件的常用属性及事件

在Delphi中，每一个组件都具有特定的属性、事件和方法。组件的属性是组件特性的描述，包括组件的外观特性（如位置、尺寸、外形、字体、可视性）和非可视化的特性，组件常见的基本属性见表3-4。

# 3.1 窗体和组件

属 性	说 明
Height	高度
Width	宽度
Left	组件在容器内的水平坐标，相对于容器左边。
Top	组件在容器内的垂直坐标，相对于容器上边。
Align	组件上的对齐方式（居上、居下、居左、居右、居中）
Visible	设置组件是否可见，默认值为可见（值为true）
Caption	显示类组件的标题
Color	组件的背景颜色
Font	设置组件显示文本的字体
Ctl3D	是否以3D方式显示组件，默认值为true
ShowHint	是否显示组件的提示信息，默认值为true，与Hint连用
Hint	当鼠标指针移到组件上时，组件显示的提示信息
Enabled	是否允许用户操作组件，true表示允许，false表示不允许
Name	用于标识组件的名称，在程序中通过Name可以调用该组件
TabOrder	Tab次序

# 3.1 窗体和组件

组件的事件是对组件所做的某个动作或系统的某些行为（如按下鼠标、双击鼠标、窗体装入等）的反应。

事 件	说 明
OnClick	触发条件
OnDbClick	当鼠标双击时触发本事件
OnMouseDown	当鼠标左键按下时触发本事件
OnMouseMove	当鼠标移动时触发本事件
OnKeyDown	当按下任意键（包括组合键）时触发本事件
OnKeyPress	当按下任意键（单字符键）时触发本事件
OnKeyUp	当松开已按下键时触发本事件
OnEnter	当获得焦点时触发本事件
OnExit	当失去焦点时触发本事件
OnStartDrag	当开始拖动时触发本事件
OnDragDrop	当组件拖动操作结束时触发本事件

## 3.2 文本显示输入类组件

### 3.2.1 TLabel组件

#### 1. 概述

TLabel标签组件位于组件面板的Standard页上（如图3-1），可以显示一个只读的字符串。

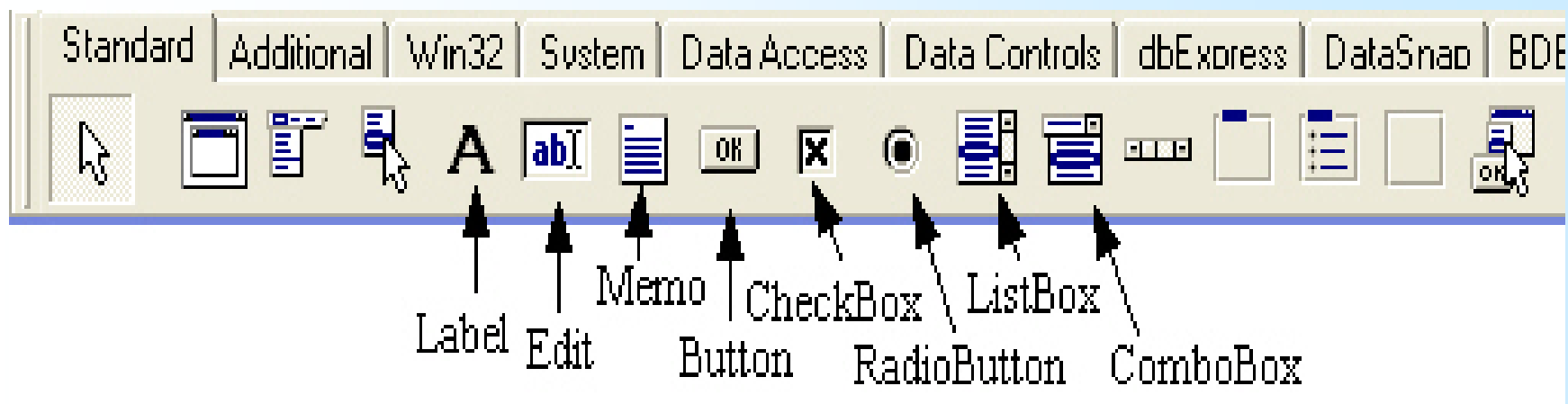


图3-1 Standard组件页

## 3.2 文本显示输入类组件

### 2. 主要属性和方法

- (1) Align属性
- (2) Alignment属性
- (3) AutoSize属性
- (4) Layout属性
- (5) TransParent属性
- (6) WordWrap属性

### 3.2.2 TEdit组件

#### 1. 概述

TEdit文本框组件位于组件面板的Standard选项卡上（如图3-1），可以显示、编辑单独的一行文本。

#### 2. 主要属性和方法

## 3.2 文本显示输入类组件

- (1) AutoSelect属性
- (2) AutoSize属性
- (3) CharCase属性
- (5) Hint属性
- (6) IME属性
- (7) PasswordChar属性
- (9) ClearSelect方法
- (10) CopyToClipboard方法
- (11) CutToClipboard方法
- (12) PasteFromClipboard方法
- (13) SelectAll方法

## 3.2 文本显示输入类组件

### 3.2.3 TMemo组件

#### 1. 概述

TMemo备注框组件位于组件面板的**Standard**选项卡上，可以显示、编辑多行文本。

#### 2. 主要属性

(1) Lines属性

(2) ScrollBars属性

(3) WantTabs属性

## 3.3 按钮类组件

### 3.3.1 TButton组件

#### 1. 概述

TButton命令按钮组件用于为用户提供选择执行的命令，通常称为命令按钮。

#### 2. 主要属性

- (1) Cancel属性
- (2) Caption属性
- (3) Default属性
- (4) Enabled属性
- (5) ModalResult属性
- (6) Name属性

## 3.3 按钮类组件

### 3.3.2 TBitBtn组件

#### 1. 概述

TBitBtn位图按钮组件的作用和TButton组件相同，唯一的区别是可以在位图按钮上同时显示文本信息和位图。

#### 2. 主要属性

- (1) Glyph属性
- (2) Kind属性
- (3) Layout属性
- (4) Margin属性
- (5) NumGlyphs属性
- (6) Spacing属性
- (7) Style属性

## 3.3 按钮类组件

### 3.3.3 TRadioButton组件

#### 1. 概述

TRadioButton单选按钮组件可以在多个条件中选择唯一的一个。通常在使用单选按钮时，总是将其进行分组。在同一组中，只能同时选中一个按钮，其余按钮自动取消选中。在实现单选按钮的分组时，可以有两种方法：

(1) 利用分组框组件（TGroupBox）和单选按钮组件（TRadioButton）实现。

(2) 利用单选按钮分组框（TRadioGroup）实现。

#### 2. 主要属性

(1) Checked属性

(2) Enable属性

(3) Visible属性

## 3.3 按钮类组件

### 3.3.4 TCheckBox组件

#### 1. 概述

TCheckBox复选框组件可以组成多项选择。用户可通过单击一个选项来选择或取消选择该选项，在一个选择组框中一次可做出多项选择。

#### 2. 主要属性

- (1) Alignment属性
- (2) AllowGrayed属性
- (3) State属性
- (4) Check属性

## 3.4 列表类组件

### 3.4.1 TListBox组件

#### 1. 概述

**TListBox**列表框组件可以显示一系列可滚动的项目列表，用户可以选中其中的一个或多个，但不能直接对这些项目进行修改。列表框中的项目列表是**Items**属性的值，可使用方法对列表框中的项目进行增加、删除和插入操作。

#### 2. 主要属性

- (1) **Columns**属性
- (2) **ExtendedSelect**属性
- (3) **IntegralHeight**属性
- (4) **ItemHeight**属性
- (5) **ItemIndex**属性
- (6) **Items**属性

## 3.4 列表类组件

(7) MultiSelect属性

(8) SelCount属性

(9) Selected属性

(10) Sorted属性

### 3.4.2 TComboBox组件

#### 1. 概述

TComboBox组合框组件由一个编辑框和一个下拉式列表框组成，可以从多个列表条目中选择一个。

#### 2. 主要属性

(1) DropDownCount属性

(2) Enabled属性

(3) Style属性

(4) Text属性

## 3.4 列表类组件

(5) ItemIndex属性

(6) SetText属性

### 【例3-1】

设计用户登录验证的窗体，用来接收用户输入的用户名和口令，单击“确定”按钮进行验证，根据输入是否正确显示相应的提示信息，单击“取消”按钮退出程序。

姓名：admin,密码：123456

### 【实现步骤】

- (1) 新建应用程序
- (2) 定制窗体

## 3.4 列表类组件



图3-2 窗体界面

表3-3 各组件属性设置

组件名	属 性	属 性 值
Label1	Caption	姓名:
Label2	Caption	密码:
edtName	Text	"
edtPass	Text	"
Button1	Caption	确定
Button2	Caption	取消

## 3.4 列表类组件

(3) 编写代码

(4) 运行程序

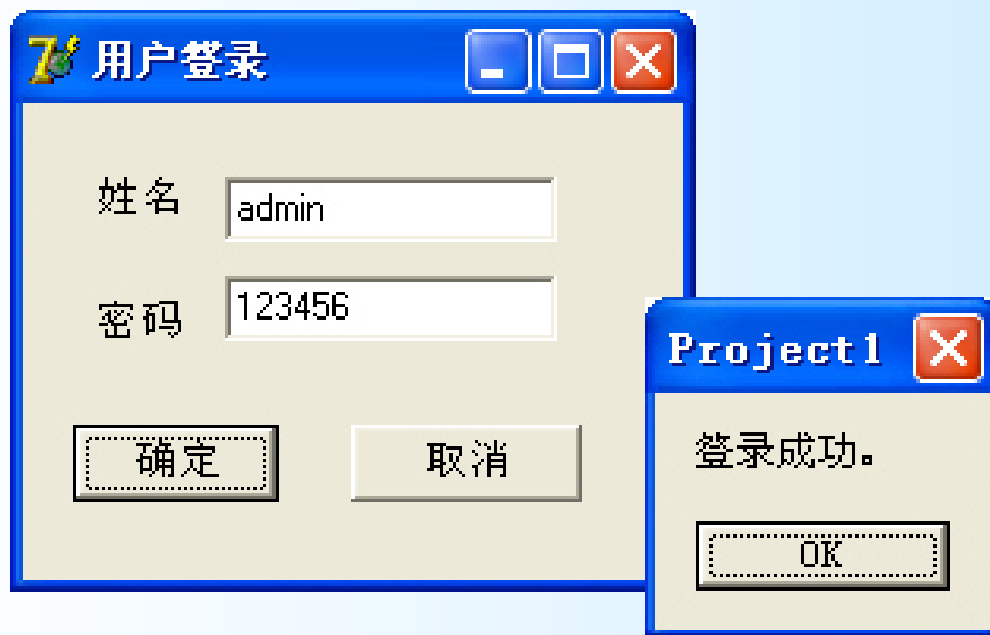


图3-3 程序进行结果

# 本章实训指导

1. 熟悉Delphi常用控件的方法和属性。
2. 掌握TLabel组件、TEdit组件和TMemo组件的用法。
3. 掌握TButton组件、TRadioButton组件、TCheckBox组件
4. 掌握TListBox组件、TComboBox组件的用法。
5. 利用常用组件设计一个具有加、减、乘、除功能的计算器，如图3-4所示。



图3-4 程序界面

# 第4章 菜单设计

本章主要内容：

- 如何创建主菜单
- 如何创建快捷菜单
- 如何使用菜单模板

# 4.1 创建主菜单

Windows应用程序的菜单通常分为两级：第一级是窗口标题下的菜单栏，称为主菜单；第二级是这些菜单所包含的下拉子菜单，称为菜单项。

按照菜单项的功能，可以将菜单项分为四种类型：

- (1) 命令菜单
- (2) 状态设置菜单
- (3) 对话框菜单项
- (4) 下拉菜单

## 4.1.1 TMainMenu组件

### 1. 概述

TMainMenu组件用来创建主菜单，它位于组件面板的Standard组件页上

## 4.1 创建主菜单

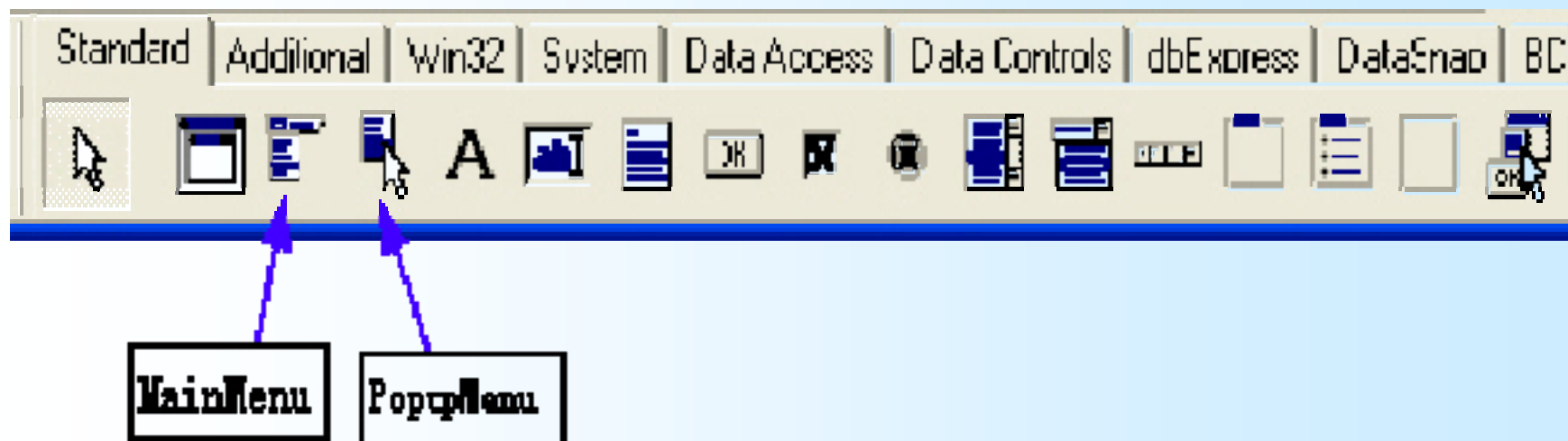


图4-1 TMainMenu和TPopupMenu组件

# 4.1 创建主菜单

## 2. 主要属性

(1) AutoHotkeys属性

(2) AutoMerge属性

(3) Images属性

(4) Items属性

## 4.1.2 主菜单的设计

通常会对菜单项件以下设置：

(1) 设置快捷键

(2) 定义热键

(3) 建立菜单分组

(4) 为菜单项增加图标显示

(5) 设计级联菜单

(6) 为菜单项指定动作

## 4.2 鼠标右键弹出式菜单

可以给窗体添加弹出式菜单，也可以给具体的某个控件添加弹出式菜单。另外，窗体中的某几个组件可以共用一个弹出式菜单，只要将这几个组件的**PopupMenu**属性设置成同一个弹出式菜单的名称就可以了。

### 4.2.1 TPopupMenu组件

#### 1. 概述

TPopu

#### 2. 主要属性

(1) Auto

(2) Han

(3) Help

(4) Item

(5) Pop

注意：当用户设计好一个菜单后，并不代表在程序执行时就可以使用，因为窗体（**Form**）并不知道还有快捷菜单，为此还要设置窗体的**PopupMenu**属性，单击**PopupMenu**属性右边的下三角按钮，选择窗体中所创建的快捷菜单的名字如**PopupMenu1**，这样在程序执行时，当右击窗体后，便会弹出相应的菜单。

## 4.2 鼠标右键弹出式菜单

### 4.2.2 鼠标右键弹出式菜单设计

选择组件面板的**Standard**选项卡上**TPopupMenu**组件，添加到窗体中，双击**TPopupMenu**组件弹出快捷菜单设计器。该窗口的菜单项设计与**MainMenuDesigner**完全相同。

如果要在某个窗体上实现单击鼠标右键弹出菜单，可以在该窗体上添加弹出式菜单，设计完毕后，还要在窗体的**PopupMenu**属性中选中该弹出式菜单。对于组件来说，也要设置**PopupMenu**属性。

## 4.3 使用模板设计菜单

Delphi中定义了一些菜单模板，在设计阶段可以方便地利用这些菜单模板向菜单中添加菜单项，而且添加的菜单项具有统一的风格。

### 4.3.1 使用菜单模板

在设计菜单时，要向菜单中添加菜单模板中定义的菜单项时，可以在菜单设计器中进行。双击窗体中的菜单组件或通过右键弹出的菜单命令**Menu Designer...**，打开菜单设计器。

### 4.3.2 定制菜单模板

在菜单设计器中先设计好一个或多个子菜单，单击鼠标右键，选中**SaveAs Template...**，将弹出**Save Template**对话框。在**Template Description**项中输入菜单模板的名称，可以使用中文。

## 4.3 使用模板设计菜单

### 4.3.3 删除菜单模板

在菜单设计器中单击鼠标右键，选中Delete Templates...命令，将弹出Delete Templates对话框，在其中选中将要删除的菜单模板，可以在按下Shift键或Ctrl键的同时利用鼠标或上、下箭头键选中多个菜单模板，按下OK按钮，则选中的菜单模板就被删除了。

#### 【例4-1】

设计一个具有如图4-2所示的菜单应用程序。

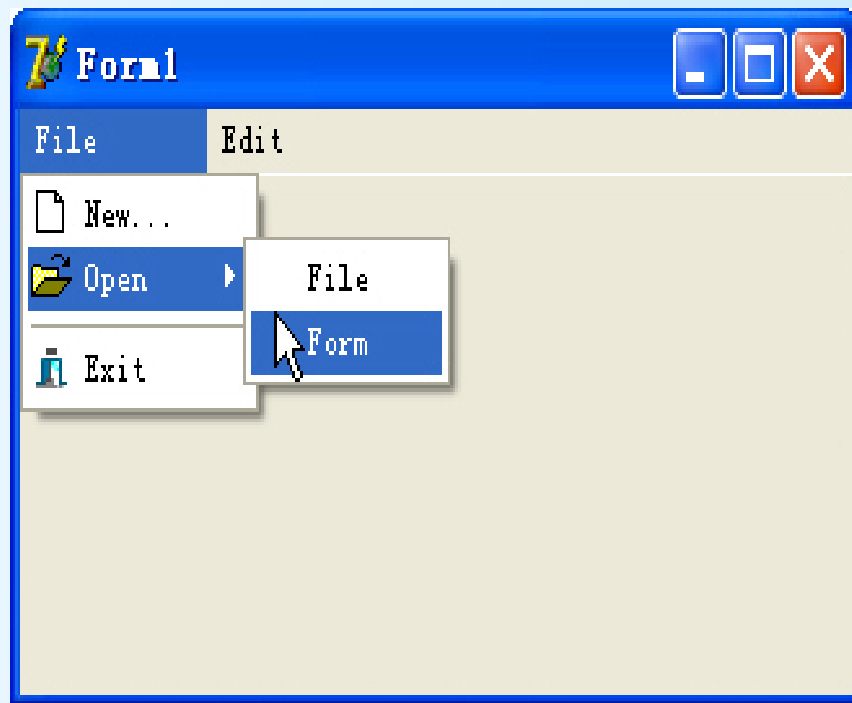


图4-2 菜单应用程序

## 4.3 使用模板设计菜单

### 【实现步骤】

- (1) 新建应用程序
- (2) 定制窗体

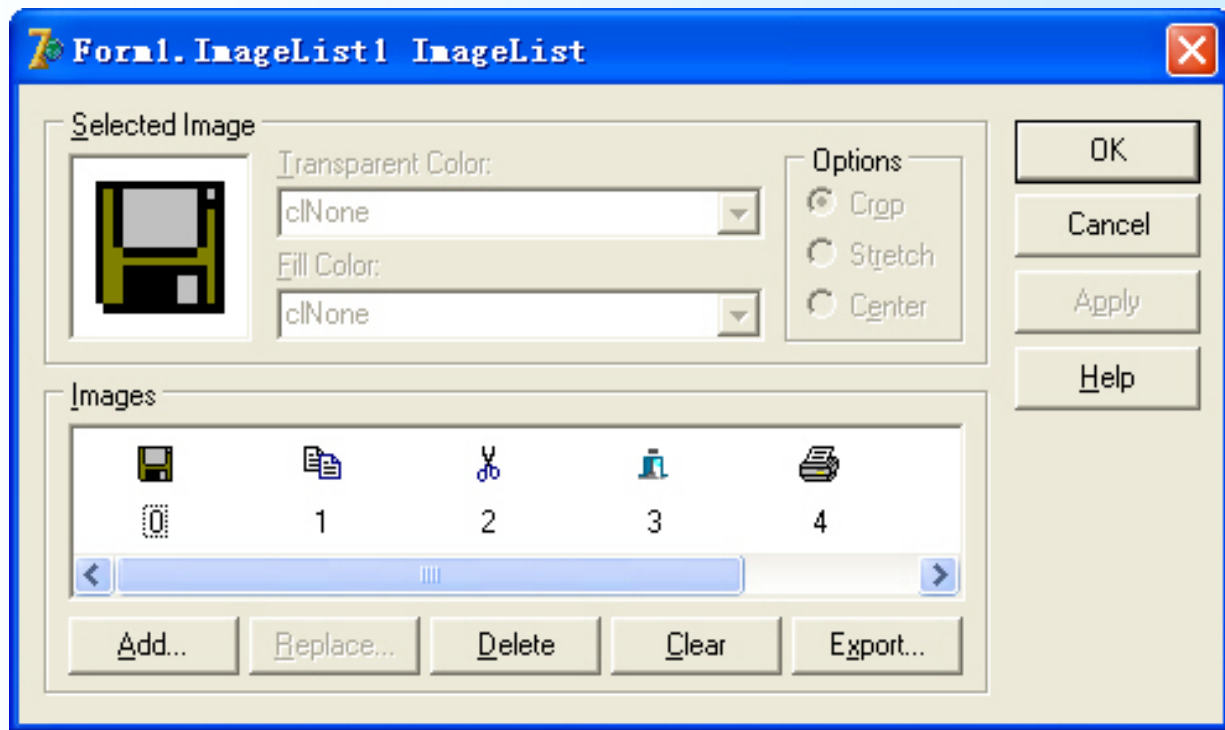


图4-3 ImageList编辑器

## 4.3 使用模板设计菜单



图4-4 图体界面

(3) 编写代码

(4) 运行程序

# 本章实训指导

---

1. 了解Windows中各种菜单类型;
2. 理解菜单的工作原理;
3. 掌握主菜单、鼠标右键弹出式菜单的设计。
4. 模仿Windows记事本的菜单界面编写一个程序。

# 第5章 数据库应用程序设计

本章主要内容：

- 数据库基础知识
- 数据库应用程序结构
- Delphi7中的ADO组件简介
- 如何为应用程序建立数据库连接

# 5.1 数据库基础知识

## 5.1.1 数据库的基本概念

### 1. 数据

数据（**Data**）是数据库中存储的基本对象。所谓数据，就是能被计算机识别与处理的符号。数据的种类很多，如数字、文字、表格、图形、图像、声音等都属于数据。

### 2. 数据库

数据库（**Database**，简称为**DB**）就是以一定的组织方式存储在计算机存储介质中的互相关联的数据的集合。它能以最佳方式、最少重复、最大独立性为多种应用提供共享服务。一个数据库常包含许多数据表、索引信息以及其他相关信息。

# 5.1 数据库基础知识

## 3. 数据库管理系统

数据库管理系（**Database Manngement System**，简称为**DBMS**）是支持人们建立、使用和修改数据库的软件系统。它是位于用户和操作系统之间层面的数据管理软件。它为用户或应用程序提供访问数据库的方法，包括数据库的建立、查询、更新及各种数据控制。

**DBMS**可以分为层次型、网状型、关系型和面向对象型等几种类型。

数据库在建立、使用和维护时由数据库管理系统统一管理，统一控制。数据库管理系统使用户方便地定义数据和操作数据，并能够保证数据的安全性、完整性、并发性及发生故障后的系统恢复。通常**DBMS**包括以下主要功能：

# 5.1 数据库基础知识

- (1) 数据定义功能
- (2) 数据操纵功能
- (3) 数据库的运行管理
- (4) 数据库的建立和维护功能
- (5) 数据库通信功能

## 4. 数据库系统

数据库系统（**DataBase System**, 简称为**DBS**）是指在计算机系统中引入数据库后的系统构成，一般由数据库、数据库管理系统及其开发工具、应用系统构成。如图5-1所示。

# 5.1 数据库基础知识

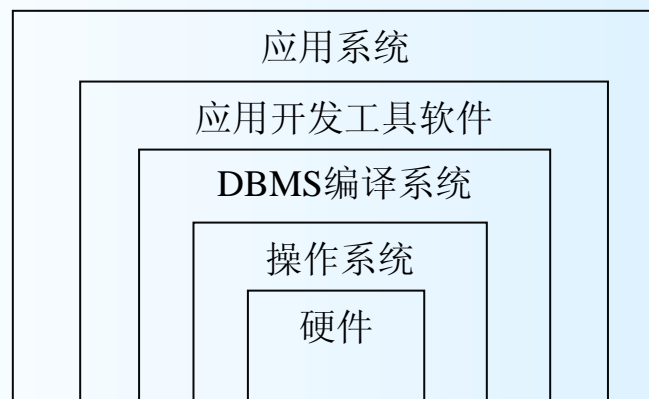


图5-1 DBS的层次结构

# 5.1 数据库基础知识

数据库系统从最终用户角度看，分为单用户结构、主从式结构、分布式结构和客户/服务器结构。

## (1) 单用户结构

整个数据库系统（包括应用程序、**DBMS**、数据）都装在一台微机上，由一个用户独占，不同机器之间不能共享数据。

## (2) 主从式结构

此结构指一个主机带多个终端的多用户结构，数据库系统（包括应用程序、**DBMS**、数据）都集中存放在主机上，所有处理任务都由主机完成，各个终端用户并发地存取数据库，共享数据资源。

## (3) 分布式结构

# 5.1 数据库基础知识

分布式结构的数据库系统是地理上（或物理上）分散而逻辑上集中的数据库系统。

## (4) 客户/服务器结构

随着工作站功能的增加和广泛使用，人们开始把**DBMS**功能和应用分开，网络中专门用于执行**DBMS**功能的计算机称为数据库服务器（简称服务器**Server**），其他安装**DBMS**的外围应用开发工具、且支持用户应用的计算机称为客户机（**Client**），这就是客户/服务器结构的数据库系统（**Client/Server**结构，简称**C/S**结构），它是目前普遍使用的数据库系统。

# 5.1 数据库基础知识

## 5. 关系数据库

关系数据库（**Relational Database**）是以关系模型作为数据的组织存储方式。关系数据库通常包含多张表，表由记录组成，记录由字段组成。

表（**Table**）：一个表就是一组相关的数据按行排列，象一张表格一样。

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

记录（**Record**）：在表中，每一行称为一条记录。

索引（**Index**）：索引是按照指定字段建立的顺序链表，能加快访问数据库的速度。

# 5.1 数据库基础知识

## 6. 数据库应用程序的设计

数据库应用程序的设计包括两个部分：

(1) 数据库设计

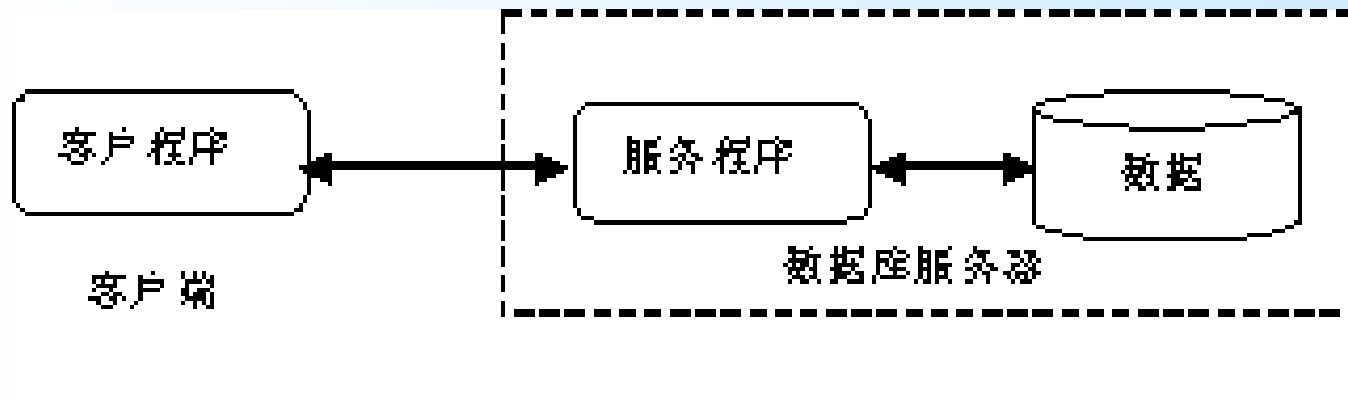
(2) 应用程序设计。

### 5.1.2 数据库产品简介

VFoxPro、Access、Paradox等属于单用户版数据库产品。这类数据库的数据被按照一定格式储存在磁盘里，使用时由应用程序通过相应的驱动程序甚至直接对数据文件进行读取。

MS SQL Server、Oracle Universal Server、Informix-Universal Server等属于大型数据库。这类数据库的数据集中存放服务器上，统一由运行在服务器上的数据库服务程序管理,用户使用客户端软件通过网络访问数据库服务程序

# 5.1 数据库基础知识



客户/服务器数据库系统结构

这类型数据库的特点是：适合于网络应用，可以同时被多个用户所访问，数据库管理系统可以赋予不同的用户以不同的安全访问权限，支持的数据量大，能完全地支持SQL语言。

# 5.1 数据库基础知识

## 5.1.3 常用SQL语句

### 1. SELECT语句

SELECT语句可以从数据库中按用户要求检索数据，并将查询结果以表格的形式返回。

SELECT语句的语法形式如下：

SELECT [ ALL | DISTINCT ] [ TOP  $n$  [ PERCENT ] ]

字段列表

[ INTO 新表 ]

FROM 源表

[ WHERE 搜索条件 ]

[ GROUP BY 分组字段 ]

[ HAVING 搜索条件 ]

[ ORDER BY 排序字段 [ ASC | DESC ] ]

# 5.1 数据库基础知识

## (1) 基本查询

```
SELECT au_fname, au_lname, phone  
FROM authors
```

该查询是把表中指定内容列出来。

## (2) 条件查询

```
SELECT au_fname, au_lname, phone AS Telephone  
FROM authors  
WHERE state = 'CA'
```

仅返回state为‘CA’的行。

## (3) 对查询结果排序:

```
SELECT au_fname, au_lname, phone AS Telephone  
FROM authors  
WHERE state = 'CA' and au_lname <> 'McBadden'  
ORDER BY au_lname ASC
```

# 5.1 数据库基础知识

## 2. INSERT语句

将新行添加到表或视图中。

INSERT语句的语法形式如下：

```
INSERT [INTO] 表名 [ ( 字段1 [, ... 字段n] ) ]  
VALUES ( { DEFAULT | NULL | 字段值1 } [, ... 字段值n] )
```

下面的例子中，通过Insert语句增加了一行记录到publishers表中。

```
INSERT publishers ( pub_id, pub_name, city, state )  
VALUES ('9998', 'San', 'wash', 'wa')
```

字段名与字段值必须对应，且数据类型一致。

# 5.1 数据库基础知识

## 3. DELETE语句

DELETE语句用来从表中删除数据。其语法为：

**DELETE [FROM] 表名 [ WHERE 搜索条件 ]**

从publishers中删除pub\_id为'9998'的记录记录的例子

。

**DELETE publishers WHERE pub\_id='9998'**

如果没有指定WHERE子句，表示删除所有记录。从employee表中删除所有记录的例子。

**DELETE from employee**

# 5.1 数据库基础知识

## 4. UPDATE语句

UPDATE语句用以更新表中一列或多列数据值。其语法如下：

UPDATE 表名

SET 字段名 = { 表达式 | DEFAULT | NULL } [, ..n ]

[ WHERE 搜索条件 ]

下面的语句把pub\_id为9998的记录的state字段修改为'NY'。

```
UPDATE publishers
```

```
SET state='NY'
```

```
WHERE pub_id='9998'
```

如果不加上WHERE子句，则UPDATE语句会修改表中的每一行数据。

# 5.1 数据库基础知识

## 5. CREATE TABLE语句

用CREATE TABLE语句创建表快捷明了。其语法如下：

**CREATE TABLE 表名**

**( 字段1 类型 ( 长度[, 小数位数] ) { [ DEFAULT 默认值] | [NULL] } [,...n] )**

下例显示了创建jobs表的定义。

**CREATE TABLE jobs**

**( job\_id smallint,  
job\_desc varchar(50) NOT NULL DEFAULT 'New  
Position',  
min\_lvl tinyint NOT NULL,  
max\_lvl tinyint NOT NULL )**

# 5.1 数据库基础知识

## 6. ALTER TABLE语句

ALTER TABLE语句可以添加或删除表的列。其语法如下：

```
ALTER TABLE 表名  
{ ADD [ 列名 类型 ( 长度[, 小数位数] ) ][,...n]  
  | DROP COLUMN 列名 [ ,...n ]  
}
```

下例添加一个允许空值的列，而且没有通过 DEFAULT 定义提供值。各行的新列中的值将为 NULL。

```
ALTER TABLE doc_exa ADD column_b  
VARCHAR(20) NULL
```

下例从一中删除字段column\_b。

```
ALTER TABLE doc_exa DROP COLUMN column_b
```

# 5.1 数据库基础知识

## 7. DROP TABLE语句

删除表定义及该表的所有数据、索引、触发器、约束和权限规范。

**DROP TABLE 表名**

表名: 是要删除表的名称。

例如: 删除employee表。

**DROP TABLE employee**

## 5.2 数据库应用程序结构

### 5.2.1 OLE DB

OLE DB可提供对存储在不同信息源的数据进行统一访问的能力，它是微软开发的一种底层数据库访问技术。

常用OLE DB 提供者有：

Microsoft Jet 4.0 OLE DB Provider

用于微软Access数据库。

Microsoft OLE DB Provider for ODBC Drivers

用于ODBC数据源。

Microsoft OLE DB Provider for Oracle

用于Oracle数据库。

Microsoft OLE DB Provider for SQL Server

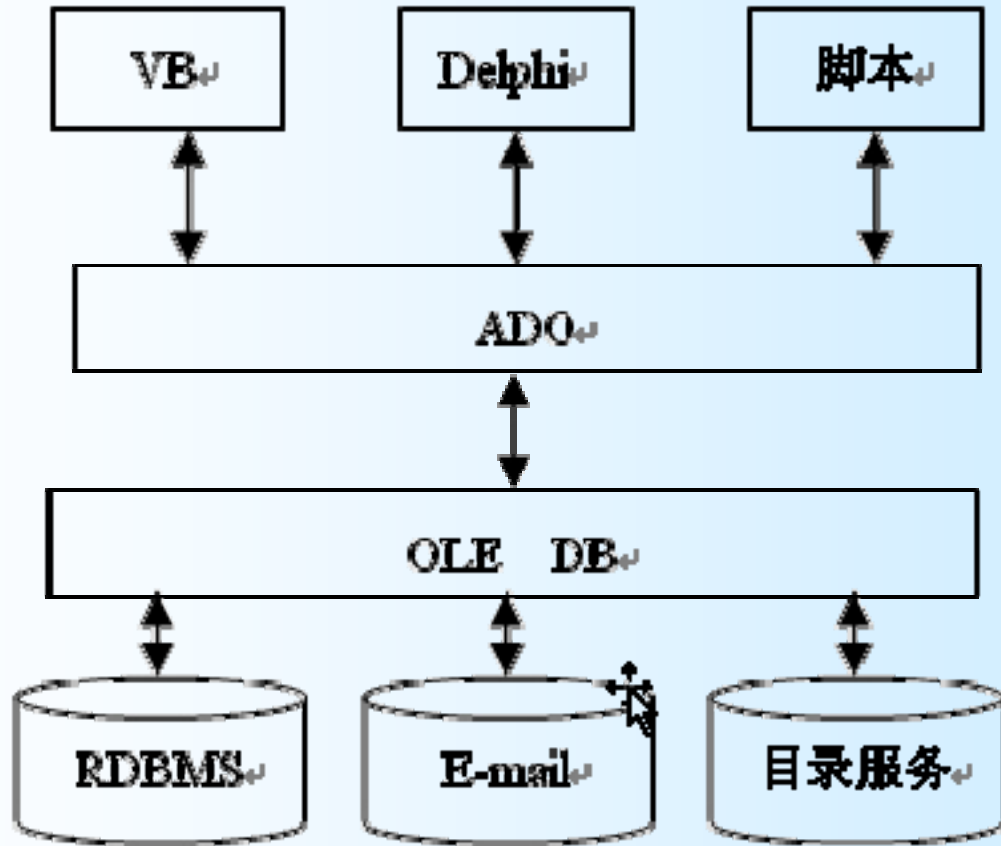
用于微软SQL Server数据库。

## 5.2 数据库应用程序结构

### 5.2.2 ADO (ActiveX Data Objects)

ADO是Microsoft ActiveX Data Objects的缩写,也是微软开发的适用于Windows 操作系统的数据库访问技术。它位于OLE DB的上层,封装了OLE DB的所有功能,为那些不能直接访问OLE DB的语言(如Visual Basic和脚本语言)提供编程接口。

## 5.2 数据库应用程序结构



OLE DB、ADO与编程语言和数据存储的关系

## 5.2 数据库应用程序结构

### 5.2.3 ODBC

ODBC（Open Database Connectivity）即开放的数据库连接技术。ODBC是微软和一些数据库厂商联合制定的，它通过应用程序接口API提供了一种跨平台的，用来访问关系数据库中数据的手段，是访问数据库的通用方法。

数据源是对数据库的有名联接，即DSN。

- 用户DSN
- 系统DSN
- 文件DSN

ODBC添加新数据源的步骤：

## 5.2 数据库应用程序结构

### 5.2.4 数据库应用程序结构

在Delphi7的数据库应用程序中，最基本的构成是3类组件：数据控制组件、数据集组件和数据源组件。

(1) 数据控制组（**Data Control**）件用于显示和修改数据库中的信息,为用户操作数据库提供一个可视化的界面,常用的组件有TDBEdit、TDBText、TDBGrid、TDBGrid。

(2) 数据集组件（**DataSet**）一方面通过ADO与实际的数据库相连接,另一方面通过TDataSource组件与数据库控制组件相连,常用的有TADODataSet、TADOTable、TADOQuery组件。

(3) 数据源组件（**DataSource**）负责将数据集组件和数据控制组件连接起来,。

## 5.2 数据库应用程序结构

使用ADO组件编写的应用程序通常具有以下结构：

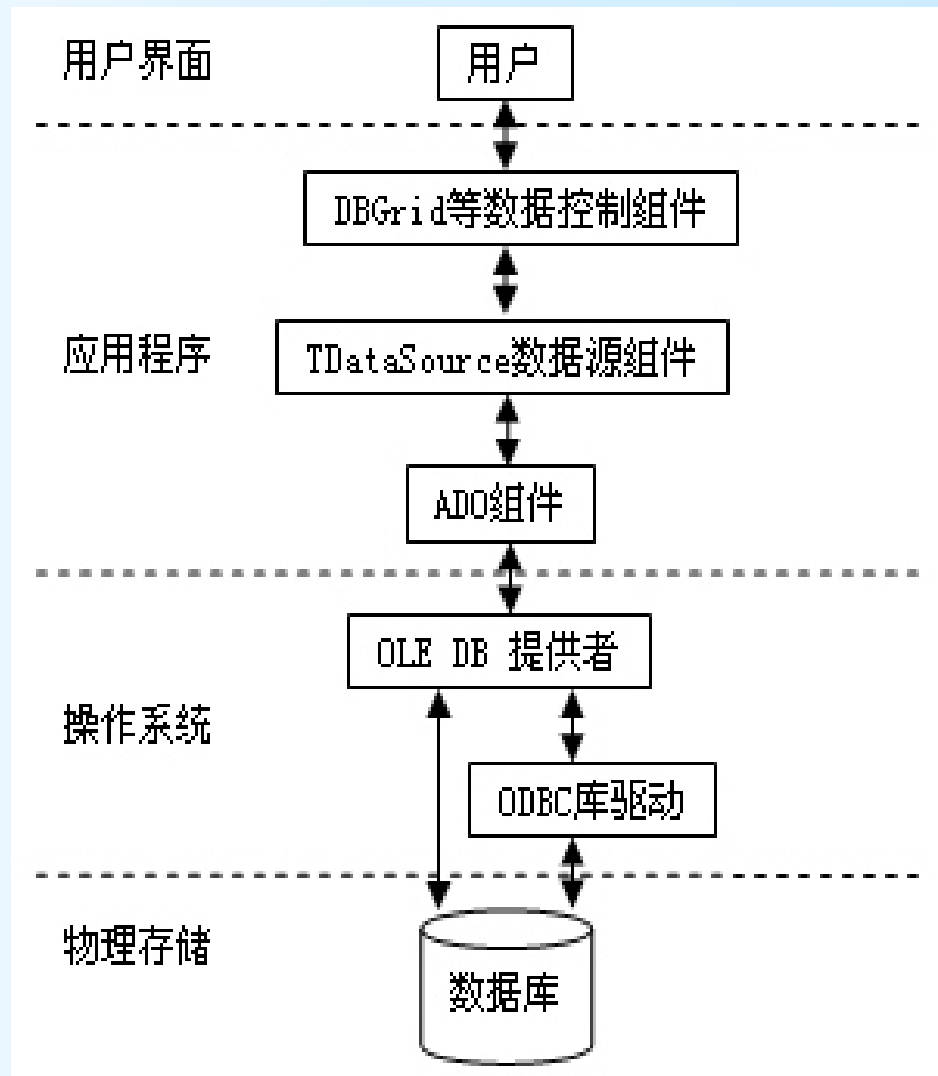


图5-8 基于ADO的数据库应用程序结构

## 5.3 Delphi7中的ADO组件

Delphi对ADO技术的支持，是通过将ADO常用对象封装进Delphi的ADO组件中，并结合Delphi本身的开放式数据库组件结构实现的。使用ADO组件，可以迅速实现数据库连接，建立数据库应用，使用ADO组件更利于我们升级和维护开发的系统。通过ADO连接的数据库应用程序分发时不需要进行额外配置，Windows系统中已默认安装了ADO的支持组件MDAC。

Delphi的ADO组件共有7个，位于ADO面板上，分别是：TADOConnection、TADODataSet、TADOTable、TADOQuery、TADOStoredProc、TADOCommand和RDSConnection。其中后面的6个组件可以直接连接到数据库，但更为常用的方式是通过TADOConnection组件连接到数据库。

## 5.3 Delphi7中的ADO组件

这些组件的作用如下：

**TADOConnection:** 该组件用于建立数据库的连接。

**TADODataSet:** 这是ADO提取及操作数据库数据的主要数据集，该组件可以从一个或多个基表中提取数据。

**TADOTable:** 主要用于操作和提取单个基表的数据。

**TADOQuery:** 该组件是通过SQL语句实现对数据库数据的提取及操作。

**TADOStoredProc:** 该数据集是专门用于运行数据库中的存储过程的。

**TADOCommand:** 该组件用于运行一些SQL命令。

**RDSConnection:** 一个进程或一台计算机传递到另一个进程或计算机的数据集合，用于远程数据访问。

## 5.3 Delphi7中的ADO组件

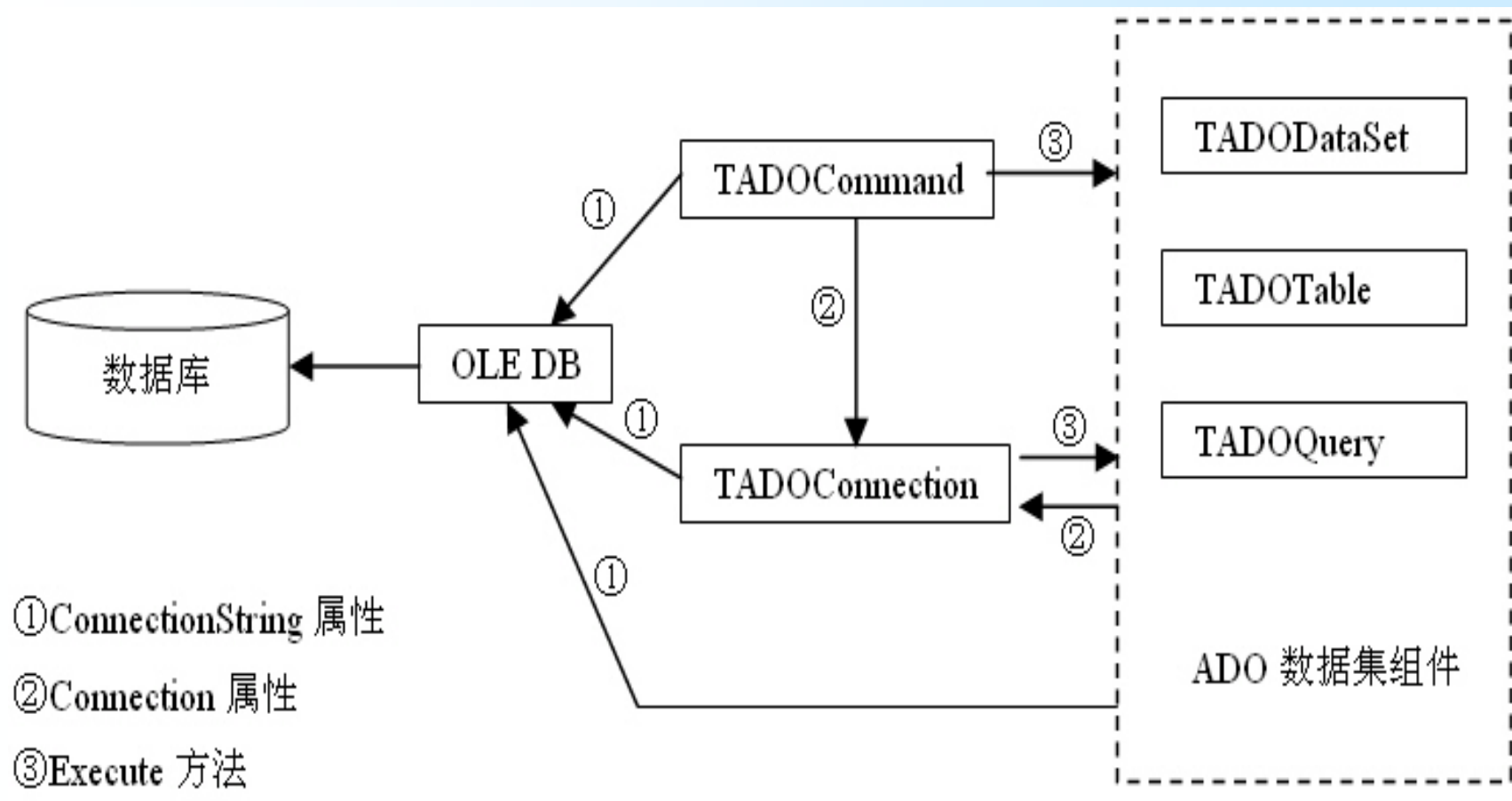


图5-9 ADO各组件之间的关系

## 5.4 连接数据库

### 5.4.1 连接本地数据库

将TADOConnection 组件连接到数据库是通过设置ConnectionString 属性实现的。

1. 使用OLD DB提供者连接ACCESS数据库
2. 使用ODBC数据源连接Paradox数据库

### 3.5.2 连接到数据库服务器

#### 【例5-1】

编写一个简单数据库应用程序，在窗体中显示dbdemos.mdb数据库中表orders的数据。dbdemos.mdb数据库是Delphi7自带的，通常位于“C:\Program Files\Common Files\Borland Shared\Data\”文件夹中。

窗体界面如图5-15所示。

## 5.4 连接数据库

The screenshot shows a Visual Basic form titled "Form1" with a blue title bar. Below the title bar is a toolbar containing three icons: a blue square with a white 'A', a blue square with a white 'D', and a blue square with a white 'O'. The main area of the form displays a data table with 8 columns: OrderNo, CustNo, SaleDate, ShipDate, EmpNo, ShipToContact, and ShipToAddr1. The table contains 11 rows of data. The form also features standard Windows window controls (minimize, maximize, close) in the top right corner and navigation buttons (back, forward) in the bottom left corner.

OrderNo	CustNo	SaleDate	ShipDate	EmpNo	ShipToContact	ShipToAddr1
1003	1351	1988-4-12	1988-5-3	114		
1004	2156	1988-4-17	1988-4-18	145	Maria Eventosh	PO Box 737
1005	1356	1988-4-20	1988-1-21	110		
1006	1380	1994-11-6	1988-11-7	46		
1007	1384	1988-5-1	1988-5-2	45		
1008	1510	1988-5-3	1988-5-4	12		
1009	1513	1988-5-11	1988-5-12	71		
1010	1551	1988-5-11	1988-5-12	46		
1011	1560	1988-5-18	1988-5-19	5		
1012	1563	1988-5-19	1988-5-20	118		
1013	1624	1988-5-25	1988-5-26	134		

图5-15 窗体界面

## 5.4 连接数据库

### 【实现步骤】

- (1) 启动Delphi 7
- (2) 新建应用程序
- (3) 定制窗体

表5-1 各组件属性设置

组件名	属 性	属 性 值	说 明
ADOConnection1	ConnectionString		连接串是自动生成的。略
ADOTable1	Connection	ADOConnection1	使用的连接对象是ADOConnection1。
	TableName	orders	指定打开的表名：orders
	Active	True	打开数据集。必须在TableName属性设置后才能设置该属性，否则会出错。
DataSource1	DataSet	ADOTable1	指定使用的数据集为ADOTable1。
DBGrid1	DataSource	DataSource1	指定为该对象显示的数据来自DataSource1。

## 5.4 连接数据库

---

### (4) 运行程序

# 本章实训指导

1. 使用MS SQL的查询分析器，练习常用SQL语句。
2. 理解数据库应用程序结构，掌握ADO组件连接数据库的方法。
3. 在MS SQL Server上建立一个数据库student，在student中添加一张表cjb，表中设置字段：班级、学号、姓名、课程名、成绩。然后用TADOConnection组件连接该数据库。

# 第6章 数据源组件和数据控制组件

本章主要内容：

- 数据源组件TDataSource
- 数据控制组件简介
- 各种数据控制组件的使用

# 6.1 数据源组件

TDataSource组件位于Data Access页，用于连接数据控制组件和数据集组件。

## 6.1.1 TDataSource组件的主要属性

Properties	Events
AutoEdit	True
DataSet	
Enabled	True
Name	DataSource1
Tag	0

TdataSource组件的属性页

# 6.1 数据源组件

## 1. DataSet属性

指定TDataSource组件所连接的数据集的名字。

可以在设计时指定该属的值，也可以在程序中指定。例如：

```
DataSource1.DataSet := ADODataSet1;
```

## 2. Enable属性

使用Enable属性可以暂时性地关闭数据源组件和与之相连的数据集组件的连接。

## 3. AutoEdit属性

它用于说明是否将与TDataSource组件相连的数据集置于编辑状态。

# 6.1 数据源组件

## 6.1.2 TDataSource组件的主要事件

### 1. OnDataChange事件

当与TDataSource相连的数据集中的记录指针的位置发生改变时，该事件就被触发，也就是说当程序调用数据集组件的Next、Previous、Insert、Append等方法导致记录指针的位置发生改变时，便会触发该事件。该事件一般用于保持应用中多个组件之间的同步。

例如：

```
procedure
TForm1.DataSource1DataChange(Sender:
TObject; Field: TField);
begin
    if (Field <> nil ) then
        showmessage( Field.DisplayName+' changed
to '+      Field.Asstring);
end;
```

# 6.1 数据源组件

## 2. OnUpdateData事件

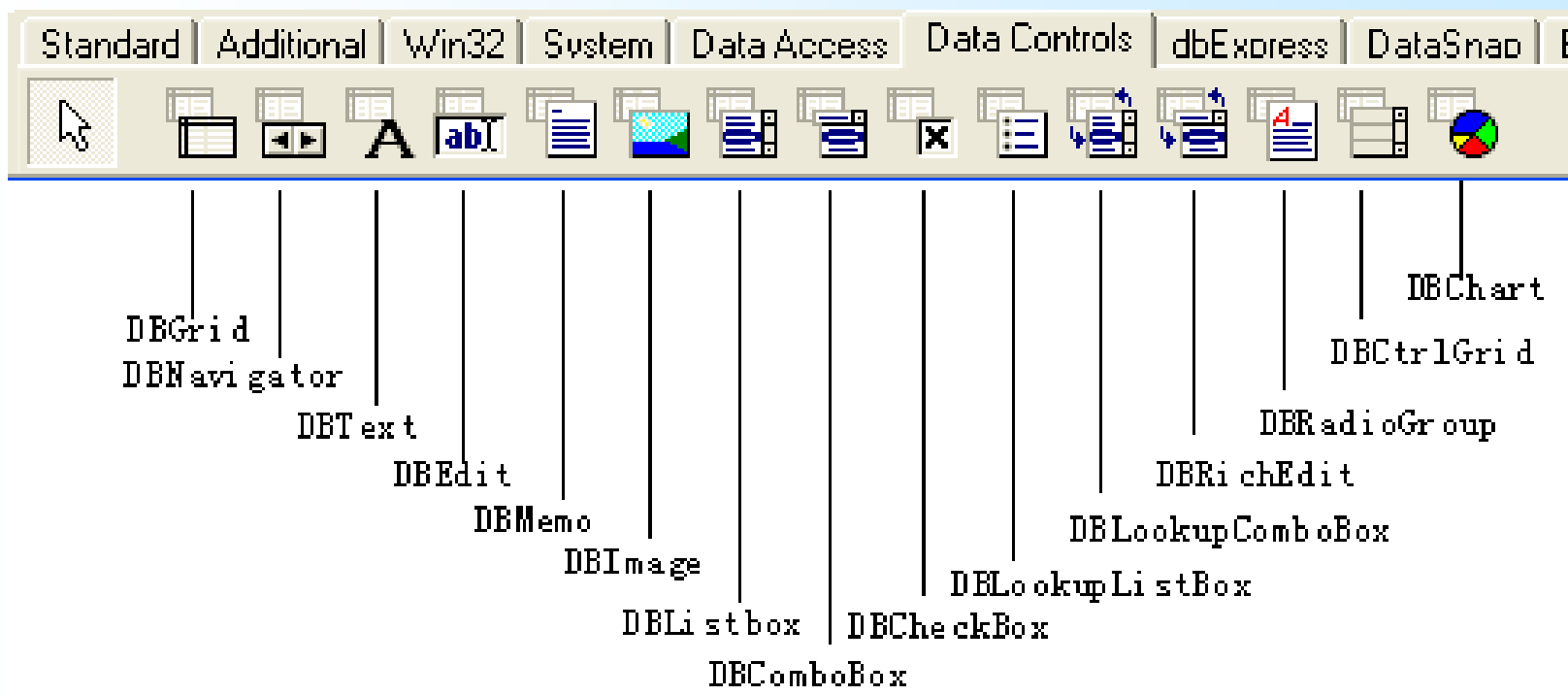
当数据集组件中当前记录将要被修改时，触发该事件。例如在程序调用**post**方法之后，但在修改后的数据真正被写回磁盘中的数据库文件之前触发该事件。

## 3. OnStateChange事件

当与**TDataSource**组件相连的数据集组件的状态发生改变时，便触发该事件。

## 6.2 数据控制组件简介

数据控制组件用于编辑和显示数据库中的数据，通常也称其为数据敏感组件，因为它们能够随着数据的变化而相应的变化。它们的主要功能是与数据源组件相配合，提供给用户一个对数据进行浏览、编辑等操作的界面。



## 6.2 数据控制组件简介

---

所有数据控制组件都有一个**DataSource**属性，该属性指定与其相关的数据源。

数据控制组件既能够把数据库中的数据显示到窗体中，又可以将经过修改的数据写回到数据库中。

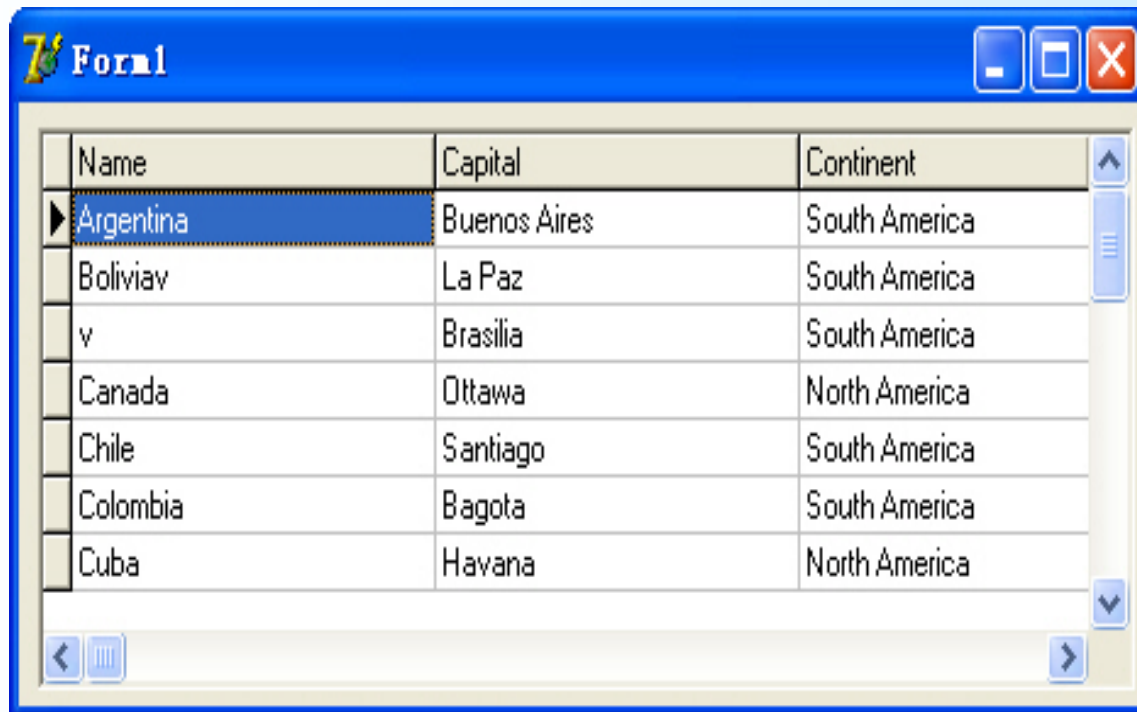
# 表6-1 数据控制组件及其功能

组件名	功 能
TDBGrid	用表格的形式显示数据集中的记录信息，表格中的各列可以在设计阶段使用字段编辑器创建也可以在运行过程中用程序设定
TDBNavigator	它提供了一组按钮用于数据集中的导航，编辑修改、插入、删除记录以及刷新数据的显示，TDBNavigator中包含的控制按钮在设计阶段可以进行选择
TDBText	用于显示数据集中当前记录的字段值
TDBEdit	用于显示和编辑数据集中当前记录指定的字段值
TDBMemo	用于显示数据集中的备注型字段，备注型字段中可以包含多行字符甚至可以是BLOB(大二进制对象)数据
TDBImage	用于显示数据集中的图像字段和BLOB数据
TDBListBox	当用户编辑修改表中当前记录的某个字段时，该组件是一个包含多个选择项的列表框，用户可以选择一个项做为字段的值
TDBComboBox	该组件是一个组合框，当用户编辑修改表中当前记录的一个指定字段时，可以直接在该组件中输入字段值也可以单击该组件从下拉式列表框中选择一个字段值
TDBCheck	当表中字段的值与该检查框的ValueChecked属性值相匹配时，该检查框被选中
TDBRadioGroup	使用该组件可以为用户提供一组选择项，但用户只能从中选择一个可选项
TDBLookupList	当用户要编辑修改数据集当前记录的指定字段时，使用该组件提供多个可选项，这多个可选项是从相关的其他表中读取的，且以列表框的形式提供给用户
TDBLookupCombox	该组件结合了TDBEdit组件和TDBComboBox组件的功能，用户可以直接向该组件中输入字段值，也可以从下拉式列表框中选择一个可选项，只是下拉式列表框中的可选项是从相关的其他的数据集中读取来的。

## 6.3 TDBGrid组件

TDBGrid组件是用来以表格形式显示和编辑数据集中记录信息的重要组件，在程序设计中经常用到，表格中的行对应于数据集中的记录，表格中的列对应于数据集中的字段。

TDBGrid组件通过数据源组件与数据集组件连接。下图是一个应用TDBGrid组件的窗体界面。



## 6.3 TDBGrid组件

### 6.3.1 TDBGrid组件的重要属性

#### 1. Columns属性

**Columns**属性用来读取和设置表格中列的特征，所有**Column**（列）对象都存储于**Columns**属性中。

一个**Column**对象代表**DBGrid**组件中的一列。

默认情况下，**DBGrid**组件会在表格中显示数据集中的所有字段，表格每一列的列名自动采用字段的名字。如果希望自己定义表格的列，在设计时可以使用**DBGrid**组件列编辑器手动设置**columns**集合。

设计时在**DBGrid**上右击，在弹出的上下文菜单中选择【**Columns editor**】项，激活列编辑器（如下图所示）。双击**DBGrid**也可以打开列编辑器,还有一种方式,就是在选中**TDBGrid**组件时单击【**Object Inspector**】的【**Columns**】属性。

## 6.3 TDBGrid组件



列编辑器及列Column对象与Columns对象的关系

# Column对象属性及说明

列属性	说 明
ButtonStyle	TColumnButtonStyle的对象，定义了显示在表格单元中的编辑按钮；其可能的选项是cbsAuto、cbsEllipsis和cbsNone
Color	字段单元的背景颜色
DropDownRows	当ButtonStyle特性为cbsAuto时，DropDownRows为显示的行数，该列有一个查找字段或者已经定义了的PickList
Expanded	Expanded如果为True，那么ObjectField被扩展，显示对象字段的附加列
FieldName	数据集的字段名，列的数据从该字段中获取值
Font	设置列中每一个单元的单元字体
ImeMode	列的输入方法编辑器，用于转换亚洲字符
ImeName	所使用的输入方法编辑器的名称
PickList	静态的选择列表（TStrings对象），该列表作为列字段的可能选项；例如一个Boolean字段可以在选择列表中使用True和False值
PopupMenu	指定TPopupMenu对象
ReadOnly	确定列中的数据是否可编辑
Title	一个嵌套的TColumnTitle对象，定义了本列的固定列单元的显示属性。包含子属性alignment(对齐)、Caption（标题）、Color（颜色）。
Visible	控制列的可视性
Width	控制列宽度

## 6.3 TDBGrid组件

### 2. DataSource属性

该属性设置为数据源组件名称，该数据源被指定为TDBGrid组件中显示数据的来源，它是TDBGrid组件中的最重要的属性。

### 3. Fields和FieldCount属性

Fields是一个集合属性，表示当前记录的所有字段。下例用一个循环显示表格当前行所有列的值。

```
procedure TForm1.Button1Click(Sender: TObject);  
var  
    I : Integer;  
begin  
    for I := 0 to DBGrid1.Columns.Count - 1 do  
        ShowMessage( DBGrid1.Fields[I].AsString );  
    end;
```

# 6.3 TDBGrid组件

## 4. SelectedField属性

通过SelectedField属性可以得到当前选定的单元格内的值。

## 5. Option属性

Option属性是一个集合，包含了描述TDBGrid组件的显示和操作属性。

Options属性的各个选项的用途见下表。

表6-3 Options属性的各个选项的用途

选 项	说 明
dgEditing	True:确保用户能够在表格中编辑、插入和删除数据集中的记录。是缺省值 False:在表格中不能编辑、插入和删除数据集中的记录
dgAlwaysShowEditor	True: 当用户选中记录中的一个字段时，自动地使该字段处于编辑状态 False:当一个字段被选中，它不能自动地变成编辑状态。是缺省值。
dgTitles	True:在表格的第一行中显示字段名或字段标题。是缺省此值。 False:在表格中不显示字段名或字段对应的标题

## 6.3 TDBGrid组件

dgIndicator	<p><b>True:</b>在表格的最左边用一个黑箭头标注当前记录指针所在的位置，在插入状态时，箭头变成星状，在编辑状时，箭头变成"I"头。是缺省值。</p> <p><b>False:</b>在表格中不标识当前记录指针的位置</p>
dgColumnResize	<p><b>True:</b>通过拖拉表格的垂直分隔线可以改变表格中各列的宽度，在具体操作时要拖拉各列中显示字段标题区域中的垂直分隔线。是缺省值。</p> <p><b>False:</b>表格中各列的宽度不能改变</p>
dgCloLines	<p><b>True:</b>在表格中显示各列之间的垂直分隔线。是缺省值。</p> <p><b>False:</b>在表格中不显示垂直分隔线</p>
dgRowLines	<p><b>True:</b>在表格中显示各行之间的水平分隔线。是缺省值。</p> <p><b>False:</b>在表格中不显示水平分隔线。</p>
DgTabs	<p><b>True:</b>可以在记录的各字段之间移动输入焦点(也即选择提示棒)，是缺省值。</p> <p><b>False:</b>不能在记录的名字段之间移动输入焦点，在表格中按Tab键时，直接跳出表格</p>
dgRowSelect	<p><b>True:</b> 选择提示棒覆盖整条记录中的全部字段</p> <p><b>False:</b>选择提示棒一次只覆盖记录中的一个字段。是缺省值。</p>
dgAlwaysShowSelection	<p><b>True:</b>在表格始终显示选择提示棒，即Selection 使其组件获得焦点时，也是如此。是缺省值。</p> <p><b>False:</b>只在当表格获得焦点时，才显示选择提示棒。</p>
dgConfirmDelete	<p><b>True:</b>当在表格中删除记录时，弹出确认信息。是缺省值。</p> <p><b>False:</b>在表格中删除记录时不弹出确认信息。</p>

## 6.3 TDBGrid组件

---

6. DragMode属性

7. DefalultDrawing属性

6.3.2 TDBGrid组件中的主要事件

见表6-3。

## 6.3 TDBGrid组件

表6-3 TDBGrid组件中的主要事件

事件	说 明
OnCellClick	当用户在单元格中单击鼠标左键时，触发该事件
OnColEnter	当用户进入表格各列时，触发该事件
OnColExit	当用户离开表格各列时，触发该事件
OnDblClick	当用户在表格中双击鼠标左键时，触发该事件
OnDragDrop	当用户在表格中用鼠标进行拖放操作时，触发该事件
OnDragOver	当用户在表格中用鼠标拖动表格时，触发该事件
OnDrawDataCell	用于定制绘制表格中各表格单元，当向表格中填充数据时触发该事件
OnEndDrag	当用户停止拖动表格时，触发该事件
OnEnter	当表格获得焦点时，触发该事件
OnExit	当表格失去焦点时，触发该事件
OnKeyDown	当用户在表格中按下任何键或组合键时，触发该事件
OnKeyPress	当用户在表格中按了任何一个数字键或字母键时，触发该事件
OnKeyUp	当用户在表格中释放任何被按下的键时，触发该事件

## 6.3 TDBGrid组件

### 6.3.3 TDBGrid组件应用实例

在 Delphi 语言的数据库编程中，TDBGrid 是显示数据的主要手段之一。但是 DBGrid 缺省的外观显得单调和缺乏创意。其实，我们完全可以在我们的程序中通过编程来达到美化TDBGrid 外观的目的。通过编程，我们可以改变TDBGrid 的表头、表格、表格线的前景色和背景色，以及相关的字体的大小和风格。

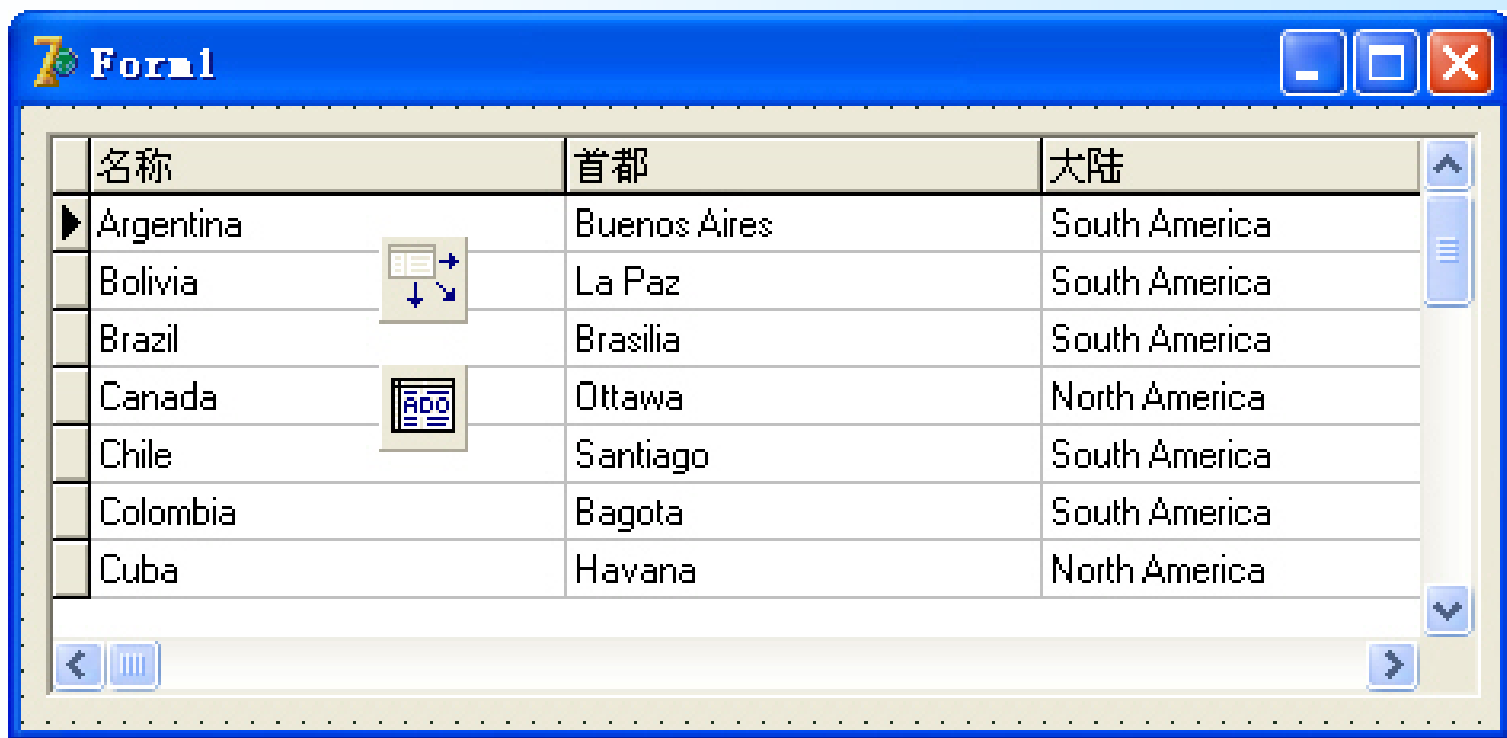
#### 【例6-1】

编写一个程序演示对 TDBGrid 各属性的设置，使 Delphi 显示的表格更加漂亮美观。使用数据库 dbdemos.mdb，在TDBGrid中显示country中的记录。

#### 【实现步骤】

- (1) 新建应用程序
- (2) 定制窗体

## 6.3 TDBGrid组件



程序设计界面

## 6.3 TDBGrid组件

(3) 编写代码

(4) 执行程序



The screenshot shows a Delphi application window titled 'Form1'. Inside the window is a TDBGrid component displaying a table with three columns: '名称' (Name), '首都' (Capital), and '大陆' (Continent). The table contains data for seven countries: Argentina, Bolivia, Brazil, Canada, Chile, Colombia, and Cuba. The 'Argentina' row is currently selected. The table has a standard grid appearance with alternating light green and light yellow rows. The window has a blue title bar with standard Windows controls (minimize, maximize, close) on the right.

名称	首都	大陆
Argentina	Buenos Aires	South America
Bolivia	La Paz	South America
Brazil	Brasilia	South America
Canada	Ottawa	North America
Chile	Santiago	South America
Colombia	Bagota	South America
Cuba	Havana	North America

图6-5 程序执行结果

## 6.3 TDBGrid组件

---

### 6.3.4 在DBGrid中的操作

在DBGrid中可以直接对表格中的数据作修改，当记录指针移动后，数据自动被保存到数据库。

## 6.4 TDBNavigator组件

TDBNavigator 组件主要用于在数据集中进行记录导航和为用户操纵数据集中的记录提供了一组简单明了的控制按钮。用户单击其中的按钮可以向前向后移动记录指针、插入记录、修改现存记录、提交对记录的修改、取消修改、删除记录，以及刷新记录的显示等。



TDBNavigator组件

# 6.4 TDBNavigator组件

## TDBNavigator组件按钮的功能

按钮名称	按钮功能
First	将当前记录指针移到数据集中第一条记录处
Prior	将记录指针移到当前记录的前一条记录处
Next	将记录指针移到当前记录的后一条记录处
Last	将当前记录指针移到数据集中最后一条记录处
Insert	调用数据集组件的Insert方法，在当前记录的前面插入一条新记录，并将数据集组件置为插入状态
Delete	删除当前记录，如果TDBNavigator组件的ConfirmDelete属性设置为true时，会弹出删除确认对话框
Edit	将数据集组件置为编辑状态，以便用户修改当前的记录
Post	提交对当前记录的修改
Cancel	取消对当前记录的修改，并将数据集组件置为浏览状态
Refresh	清除数据浏览组件的显示缓冲区，并用与其相连的数据集组件中的记录刷新显示缓冲区。

## 6.4 TDBNavigator组件

### 6.4.1 TDBNavigator组件的属性

1. DataSource属性
2. VisibleButtons属性
3. ShowHint属性
4. Hints属性
5. ConfirmDelete属性

### 6.4.2 TDBNavigator组件的事件

1. BeforeAction事件
2. OnClick事件

### 6.4.3 TDBNavigator组件的应用实例

#### 【例6-2】

为【例6-1】的应用程序的添加导航功能。

## 6.4 TDBNavigator组件

### 【实现步骤】

- (1) 打开项目
- (2) 添加组件并设置属性
- (3) 运行程序

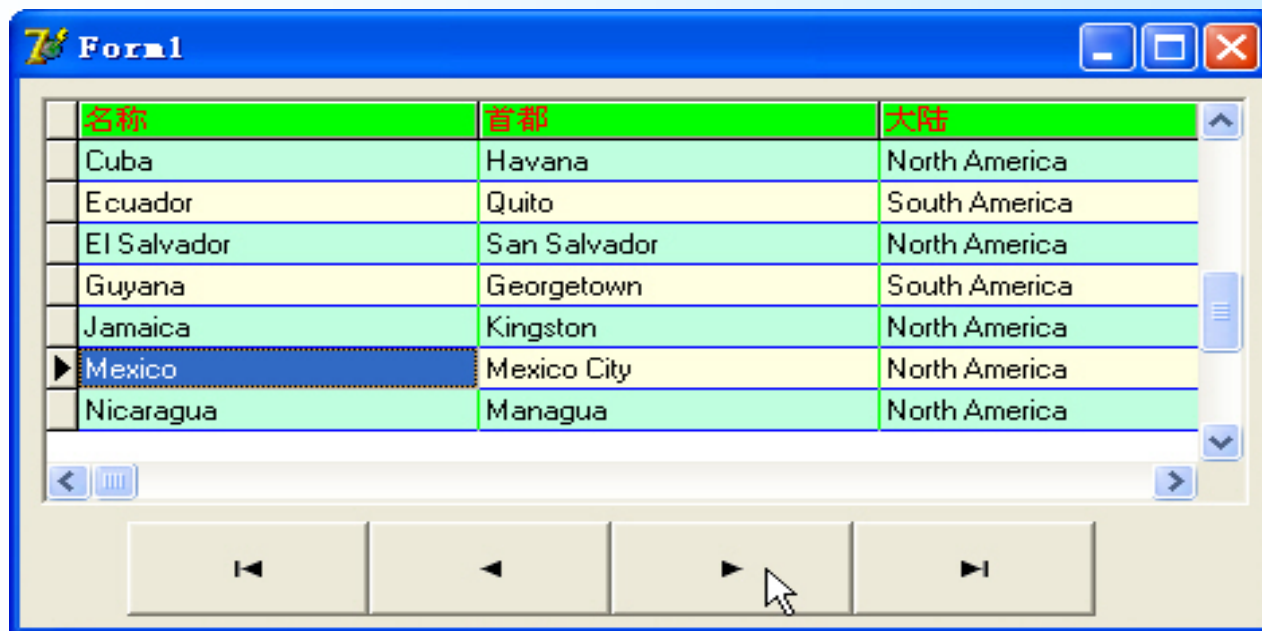


图6-7 程序执行结果

## 6.5 TDBText组件与TDBEdit组件

TDBText组件是一个只读的数据浏览组件，它和TLabel组件类似。TDBText组件用于显示数据库集中记录的指定字段的值。因为TDBText组件显示的是数据集中当前记录的指定的字段的值，所以它显示的内容也是动态的，它显示的内容随着记录指针的移动而变化。但该组件不能用于编辑数据库中的数据。

### 6.5.1 TDBText组件的主要属性

1. DataSource属性
2. DataField属性

DataField属性指定了TDBText组件应该显示当前记录中哪一个字段的值。

列如要让DBText1显示数据集中area字段的内容，语句如下：

```
DBText1.DataField:='area';
```

### 6.5.2 TDBEdit组件的主要属性

## 6.6 TDBMemo组件及其应用

TDBMemo组件主要用于显示和编辑数据集中的大二进制(BLOB)类型的字段值。TDBMemo组件能够显示多行文本，TDBMemo组件允许用户在其中输入和修改多行文本信息，它是Delphi中用来显示和编辑数据集中的大二进制类型的文本字段的唯一的数据控制组件。

### 6.6.1 TDBMemo组件的主要属性

1. ReadOnly属性
2. MaxLength属性
3. SScrollBar属性
4. WordWrap属性
5. Alignment属性
6. AutoDisplay属性

# 6.6 TDBMemo组件及其应用

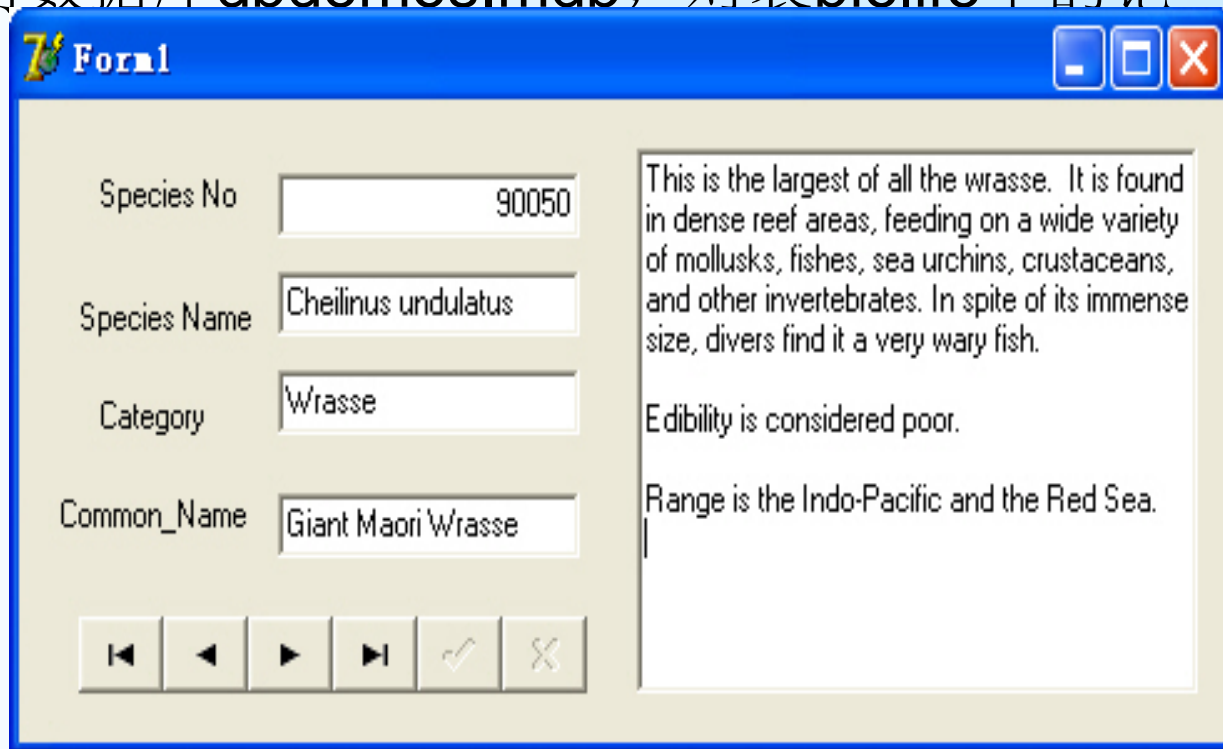
## 6.6.2 TDBMemo组件使用的简单实例

### 【例6-3】

使用TDBMemo组件建立一个应用程序，用于多行文本的编辑。使用数据库dbdemos.mdb，对表biolife中的记录进行操作。

### 【实现步骤】

- (1) 创建项目
- (2) 定制窗体
- (3) 运行程序



The screenshot shows a Windows-style application window titled "Form1". Inside the window, there is a form with four text input fields on the left and a large memo field on the right. The fields are labeled "Species No", "Species Name", "Category", and "Common\_Name". The values entered are "90050", "Cheilinus undulatus", "Wrasse", and "Giant Maori Wrasse" respectively. The memo field contains the following text: "This is the largest of all the wrasse. It is found in dense reef areas, feeding on a wide variety of mollusks, fishes, sea urchins, crustaceans, and other invertebrates. In spite of its immense size, divers find it a very wary fish. Edibility is considered poor. Range is the Indo-Pacific and the Red Sea." At the bottom of the form, there are six buttons: a back arrow, a left arrow, a right arrow, a forward arrow, a checkmark, and an 'X' button. The window has standard Windows controls (minimize, maximize, close) in the top right corner.

图4-8 程序执行结果

## 6.7 TDBImage组件

TDBImage组件与TDBMemo组件具有很多相似的属性，它是用来显示和编辑数据集中的BLOB类型的位图图像字段的。

默认情况下，在TDBImage组件中允许用户对位图图像进行编辑，如将图像复制到剪贴板上或从剪贴板上粘帖到TDBImage组件中，也可以在程序中调用CutToClipboard、CopyToClipboard和PasteFromClipboard方法来实现剪切、拷贝、粘帖功能，进行操作时要保证TDBImage的ReadOnly属性值为False。

TDBImage组件也具有一个AutoDisplay属性，该属性的控制和作用与TDBMemo 组件的AutoDisplay属性是一样的。

。

## 6.8 TDBComboBox组件

TDBComboBox组件中包含了TDBEdit组件的全部功能，它们具有相似性，不同的是在运行过程中TDBComboBox组件同时有一个下拉式列表框，在下拉式列表框中有一组可供选择的项供用户选择，这些可选项是在设计阶段由程序设计人员提供给TDBComboBox组件的Items属性的。TDBComboBox组件一定要对应数据集中的一个字段。

### 6.8.1 TDBComboBox组件的属性

1. Items属性
2. Style属性
3. DropDownCount属性
4. ItemHeight属性
5. Sorted属性

## 6.8 TDBComboBox组件

### 6.8.2 TDBComboBox组件的使用

建立一个使用TDBComboBox组件的应用程序的主要步骤如下：

- (1)向窗体中添加TDBComboBox组件。
- (2)设置TDBComboBox组件的DataSource属性，使它与数据集产生联系。
- (3)设置TDBComboBox组件的DataField属性，使它指向数据集中的特定字段。
- (4)设置TDBComboBox组件的Items属性，为列表框设计选项。

## 6.9 TDBListBox组件

TDBListBox组件的基本功能与TDBComboBox组件基本上是一样的，它们的不同之处在于TDBListBox组件没有下拉式列表框而是一个列表框，在列表框中显示一组供用户选择的可选项，在运行过程中，用户单击其中的可选项可以为TDBListBox 组件对应的字段赋一个字段值，但用户不能自己从键盘上输入一个列表框中不存在的字段值。

如果在应用程序中，TDBListBox对应数据集中一个具体的字段，那么当在数据集中移动记录指针时，当前记录中对应TDBListBox组件的字段的值在TDBListBox组件的列表框中将以高亮度显示，如果当前记录的该字段值不在列表框中，那么列表框中的可选项没有一项是高亮度地显示的。

# 6.10 TDBCheckBox组件

TDBCheckBox组件是一个可以进行是非选择的复选框。在TDBCheckBox组件中，不但能表示是或非，而且还能指定任意两种数值。

## 6.10.1 ValueChecked属性

该属性是字符串属性,当字段的内容与该属性值匹配时,该组件被选中.

## 6.10.2 ValueUnchecked属性

该属性也是字符串属性,当字段的内容与该属性值匹配时,该组件被清除.

## 6.10.3 Checked属性

**Checked**属性是一个**Boolean**类型变量，表示复选框是否被选择。

## 6.10.4 Caption属性

**Caption**属性是一个字符串，指定了复选框旁边的文本。

## 6.11 TDBRadioGroup组件

TDBRadioGroup组件提供一个由单选框组成的组。它也是和数据集中某个字段联系，并通过单选框的选择表示该字段的值。每一个TDBRadioGroup组件某个时刻只有一个单选框能被选中。

### 6.11.1 Columns属性

该属性用来设置排列单选按钮的列数,默认值为1.

### 6.11.2 Items属性

用来设置单选按钮旁显示的文字。它是一个TStrings类型的字符串数组，它的每一行对应一个选项的标签。

### 6.11.3 Values属性

与Items属性一样,也是一个字符串列表,设置方法也是与Items属性一样。表示相应单选项的值。

# 6.11 TDBRadioGroup组件

TDBRadioGroup组件提供一个由单选框组成的组。它也是和数据集中某个字段联系，并通过单选框的选择表示该字段的值。每一个TDBRadioGroup组件某个时刻只有一个单选框能被选中。

## 6.11.1 Columns属性

该属性用来设置排列单选按钮的列数,默认值为1.

## 6.11.2 Items属性

用来设置单选按钮旁显示的文字。它是一个TStrings类型的字符串数组，它的每一行对应一个选项的标签。

## 6.11.3 Values属性

与Items属性一样,也是一个字符串列表,设置方法也是与Items属性一样。表示相应单选项的值。

# 本章实训指导

---

1. 掌握TDataSource组件的用法。
2. 掌握常用数据控制组件的用法。
3. 使用所掌握数据控制组件编写一个简单的通讯录程序，要求实现记录的浏览、添加、删除和修改功能。

# 第7章 用TADOTable组件操纵数据

本章主要内容：

- TADOTable组件概述
- 记录的读取与修改
- 记录的添加与删除
- 数据查询
- 记录指针
- 数据过滤

# 7.1 TADOTable组件概述

作为数据集组件，TADOTable、TADODataSet、TADOQuery和TADOStoreProc组件都是继承自父类TCustomADODataSet，TCustomADODataSet是由TDataSet继承而来，如图7-1所示。所以它们在属性、事件及方法上有许多相同的地方。

TADOTable 组件只能通过ADO 访问数据库中单个数据表的数据

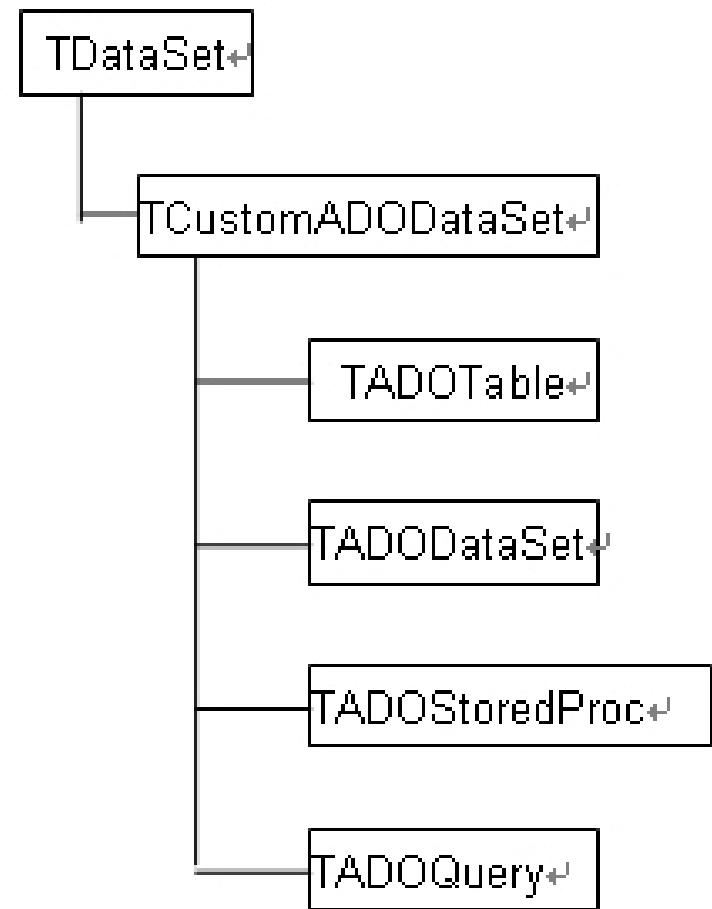


图 7-1 ADO 数据集组件的继承关系

# 7.1 TADOTable组件概述

## 7.1.1 TADOTable组件的属性

### 1. ADO数据集组件的共同属性

属 性	说 明
Active	Active属性指定数据集是否处于打开状态。设置Active属性为True, 则数据集被打开；设置Active属性为False, 则数据集被关闭。
State	State属性表明了当前数据集的状态
CacheSize	指定数据集的缓冲区大小。
CommandTimeout	CommandTimeout属性是一个整型数，指定执行一个命令的最大允许时间，默认值是30秒。
Connection	指定所使用的数据源连接组件的名称，即TADOConnection 组件的名称。
ConnectionString	即连接字符串，用于指定数据库的连接信息。
CursorLocation	指定数据库记录指针是采用客户端模式还是服务器端模式。
CursorType	指定在数据集中使用的记录指针类型。
AutoCalcFields	设为 True 则允许应用程序触发OnCalcFields事件。
BOF	为True时，表示当前指针指向第一条记录, 否则为False。
EOF	为True时，表示当前指针指向最后一条记录, 否则为False。
Bookmark	在数据集中设定标记，用于在一个数据集中获得或者设置当前记录。
Filter	设置过滤条件。

# 7.1 TADOTable组件概述

Filtered	相当于过滤的开关，当Filtered属性的值为True时，数据集从数据库中获取符合条件的记录；当Filtered属性的值为False时不执行过滤。
FilterOptions	确定过滤方式。
FieldCount	返回该数据集的字段数。
FieldDefList	返回数据集字段定义列表。
FieldDefs	表明数据集中字段的定义信息。如字段的数据类型、数据长度、名称等。
FieldList	数据集中字段组件的连续列表。
Fields	数据集中字段的集合，用于访问数据集中的字段。
FieldValues	可访问当前记录所有字段值列表。
Found	表示FindFirst、FindNext、FindLast或者FindPrior是否成功。
IndexName	指定当前使用的索引。
LockType	指定了数据集打开数据库时，对数据表中的记录的锁定类型。
MaxRecords	指定记录集中一次允许从数据库中返回的最大记录数。 默认值为0，表示不限制返回行数。
Modified	表示数据集是否被更改了。布尔型。
Name	数据集组件名称
RecNo	数据集的记录号。
RecordCount	与数据集相关的记录总数。
RecordSize	表示数据集中记录缓冲区的大小。

# 7.1 TADOTable组件概述

## 2. 对重要属性的说明

### (1)Active属性

**Active**属性确定数据集是否处于打开状态。设置**Active**属性为**True**,则数据集被打开,相当于调用**Open**方法,可以对数据库进行读或写操作;设置**Active**属性为**False**,则数据集被关闭,相当于调用**Close**方法。

设置**Active**为**True**,将触发**BeforeOpen**事件,然后设置**state**属性为**dsBrowse**,打开一个数据库游标,然后触发**AfterOpen**事件。

设置**Active**为**False**,将触发**BeforeClose**事件,然后设置**state**属性为**dsInactive**,最后触发**AfterClose**事件。

在更改影响到数据库的状态或数据显示组件的状态的数据集属性时,要提前将数据集的**Active**属性设置为**False**

# 7.1 TADOTable组件概述

## (2) State属性

State属性表明了当前数据集的状态，其取值的含义见表。

状态值	含 义
dsInactive	数据集处于关闭状态
dsSetKey	数据集处于查找键状态
dsBrowse	数据集处于浏览状态，用户可以浏览数据，但不能修改
dsEdit	数据集处于编程状态
dsInsert	数据集处于插入状态
dsCalcFields	数据集处于计算字段数据时的状态，这时不能修改数据
dsFilter	数据集处于过滤状态，可以显示数据，不能修改
dsNewValue	这是一个临时状态，当某个字段对应的TField组件的NewValue属性被修改时数据集处于这个状态。
dsOldValue	这是一个临时状态，当某个字段对应的TField组件的OldValue属性被修改时数据集处于这个状态。
dsCurValue	这是一个临时状态，当某个字段对应的TField组件的CurValue属性被修改时数据集处于这个状态。
dsBlockRead	数据集处于禁止读取状态
dsInternalCalc	数据集处于内部计算状态
dsOpening	数据集处于正被打开的状态

# 7.1 TADOTable组件概述

## (3) Connection属性和ConnectionString属性

指定所使用的数据源连接组件的名称，即TADOConnection 组件的名称。

## (4) CacheSize属性

指定数据集的缓冲区大小。数据集首先把数据从数据库中取出,然后保存在内存的一块区域中，这块内存区域就是所谓的缓冲区。如果设置CacheSize属性值为20则表示数据集将一次从数据库中提取20条记录，并将这 20 条记录保存在缓冲去中。缺省值为1，也是其最小值。

## (5) CursorLocation属性

指定数据库游标是采用客户端模式还是服务器端模式。  
clUseServer 使用服务器端的数据库游标，适用于数据量大的数据集； clUseClient 使用客户端的数据库游标的时候，数据将被下载到本地计算机上，并 在本地进行操作；

# 7.1 TADOTable组件概述

## (6) AutoCalcFields属性

该属性设为 **True** 则允许应用程序触发**OnCalcFields**事件。计算字段依赖于当前记录的一个或多个字段，通过已有的字段数据进行计算。

## (7) BOF属性和EOF属性

**BOF**属性和**EOF**属性用于判断当前记录指针的位置是否位于文件开始和结束处。它们都是只读的,为**Boolean**型。**BOF**属性为**True**时，表示当前指针指向第一条记录。**EOF**属性为**True**时，表示当前指针指向最后一条记录。

例如：

```
if ADOTable1.BOF and ADOTable1.EOF then  
    ShowMessage('数据集是空的。');
```

## (8) Fields属性

数据集中的字段集合，用于访问数据集中的字段。

# 7.1 TADOTable组件概述

例如：

```
Edit1.Text:=ADOTable1.Fields.Fields[6].AsString;
```

//读取字段值

```
ADOTable1.Edit; //将数据集设为编辑状态
```

```
ADOTable1.Fields.Fields[6].AsString:=Edit1.Text;
```

//设置字段值

```
ADOTable1.Post; //提交到数据库保存
```

# 7.1 TADOTable组件概述

## 3. TADOTable组件特有的属性

### (1) MasterSource属性

指定一个数据源组件用于主-细表关系。该数据源组件所连接的数据集组件将作为主表用于与本数据集组件建立主-细表关系。

### (2) ReadOnly属性

指定数据集中的数据是否只读。

### (3) TableDirect属性

指定这个表是通过表名访问还是通过SQL语句访问（仅限Select语句）。TableDirect属性为True表示可以通过SQL语句访问，为False，则只能通过表名访问。默认为False。

### (4) TableName属性

指定数据表名，数据集从中读取数据。

# 7.1 TADOTable组件概述

## 7.1.2 TADOTable组件的方法

### 1. ADO数据集类组件的共同方法

方 法	说 明
ActiveBuffer	返回一个PChar，包含激活记录的数据
Append	添加一个新的记录到数据集中
AppendRecord	添加一个新的记录到数据集中。以数组参数传递来的值填充字段
BookmarkValid	该方法传递一个Bookmark参数，如果此Bookmark在数据集中有效则返回True
Cancel	取消对数据集的修改，并设置数据集的状态为dsBrowse
CancelBatch	在批更新模式下，撤销一批正等待处理的更新
CancelUpdates	撤消一个准备执行的更新操作。用于一般更新模式，在Post执行之前调用
Clone	克隆另外一个数据集到当前调用组件
CheckBrowseMode	当前记录更改时，自动提交或取消数据更改。
ClearFields	清除激活记录的所有字段值
Close	关闭数据集
CompareBookmarks	比较两个书签，如果这两个书签引用同样的记录则返回0，如果第一个书签指定所引用的记录在数据集中的位置比第二个书签在数据集中的位置靠前则返回一个小于0的值，否则返回一个大于0的值
ControlsDisabled	Boolean特性，表示相应的控件是否失效

# 7.1 TADOTable组件概述

方 法	说 明
Create	构造函数
CreateBlobStream	从一个Field参数创建一个BlobStream
CursorPosChanged	使内部光标定位无效
Delete	删除当前的记录
DeleteRecords	删除记录集中的一行或多行记录。
Destroy	析构函数
DisableControls	在更新过程中使相应的控件无效
Edit	将记录的状态设置为dsEdit；记录在编辑模式下
EnableControls	使相应的控件有效
FieldByName	返回动态的TField，通过字段名搜索
FindField	如果找到指定的字段名则返回一个TField；否则返回nil
FindFirst	返回一个Boolean值，表示查找的成功或者失败；将光标定位在数据集中的第一个记录上
FindLast	返回一个Boolean值，表示查找的成功或者失败；将光标定位在数据集中的最后一个记录上
FindNext	返回一个Boolean值，表示查找的成功或者失败；将光标定位在数据集中当前记录的下一个记录上

# 7.1 TADOTable组件概述

方 法	说 明
FindPrior	返回一个Boolean值，表示查找的成功或者失败；将光标定位在数据集中当前记录的前一个记录上
First	将光标定位在第一条记录上
FreeBookmark	该方法传递一个用GetBookmark方法返回的书签，释放这个书签
GetBlobFieldData	返回BLOB字段值，根据FieldNo将值返回到一个字节数组：TBlobFieldData
GetBookmark	返回代表当前记录的书签
GetCurrentRecord	返回一个Boolean值，表示Buffer参数是否被当前记录缓冲区的值所填充
GetDetailDataSets	用每一个嵌套的数据集填充TList参数
GetDetailLinkFields	用字段组件（此组件构成了一个主细节关系）填充两个TList参数
GetFieldData	如果成功的话以字段数据填充一个缓冲区
GetFieldList	将所有由FieldNames参数指定的字段组件拷贝到TList参数中
GetFieldNames	返回数据集中所有字段名的一个列表，保存在TStrings参数中
GetIndexNames	获取当前数据集的索引名称的列表。
GotoBookmark	将光标定位到由Bookmark参数指定的记录中
Insert	将数据集设置为插入模式（State = dsInsert）
InsertRecord	插入一个记录，字段值由传递过来的变体数的常量数组填充

# 7.1 TADOTable组件概述

方 法	说 明
IsEmpty	一个Boolean值，表示数据集是否为空
IsLinkedTo	如果数据集已经连接到参数TDataSource，则返回True
IsSequenced	如果数据库表格由数据集表示则返回True，表示记录号码是否代表记录的顺序
Last	将光标定位到数据集中的最后一个记录
LoadFromFile	从一个文件中读以数据到ADO数据集中。
Locate	定位一条记录并把这条记录作为当前记录。如果找到记录则返回True
Lookup	由找到的记录中返回指定字段的值
MoveBy	将光标定位到由当前记录加上偏移量所代表的记录上
Next	将光标定位到下一个记录
Open	打开数据集
Post	将记录中的修改发送到数据库
Prior	将光标定位到前一个记录
Refresh	重新从数据库读取数据
Requery	刷新记录集，可以保持数据集的数据和数据库一致
Resync	从数据库中重新获取前一个、当前的和下一个记录

# 7.1 TADOTable组件概述

方 法	说 明
SaveToFile	与LoadFromFile方法对应，将一个ADO数据集中的数据保存到一个文件中
Seek	使用当前索引搜索记录，并移动数据集的指针。
SetFields	变体数组参数（这个数组中的值被传递到记录的字段中）设置字段值；状态必须设置为dsEdit或者dsInsert
Translate	拷贝源字符串到目标字符串，使字符串值在ANSI字符映射和BDE字符映射之间进行转换
UpdateBatch	在批更新模式下，将一批更新提交到数据库。
UpdateCursorPos	内部使用，以确保该光标被定位在激活的记录上
UpdateRecord	用于更新数据感知控件和数据集，以反映记录的更改
UpdateStatus	高速缓存的记录更新状态（usUnmodified, usModified, usInserted, usDeleted）

# 7.1 TADOTable组件概述

## 2. 重要方法的说明及使用

### (1) 打开（Open）和关闭（Close）数据集的方法

调用Open方法将Active属性设置为True。当Active属性为True时，可以从数据库中读取数据或向数据库中写数据。

调用Close方法设置Active属性值为False。当Active属性值为False时，数据集将被关闭，无法对数据库进行读/写操作。

例如：在实际程序代码中，语句：

`ADOTable1.Open;`

就等价于语句：

`ADOTable1.Active:=True;`

而语句：

`ADOTable1.Close;`

等价于语句：

`ADOTable1.Active:=False;`

# 7.1 TADOTable组件概述

## (2) DeleteRecords方法

删除记录集中的记录。

## (3) SaveToFile方法

把当前数据集中的数据按照指定的格式保存到指定的文件中，**FileName**为指定的文件名，**Format**为保存的文件格式。

## (4) GetIndexNames方法

查询表中的全部索引名，返回值将保存在参数**List**中，调用方法为：

```
ADODataset1.GetIndexNames(ListBox1.Items);
```

## (5) FieldByName方法

**FieldByName**方法是根据一个特定的字段名查询一个字段。**FieldName**是一个已存在的字段名。当只知道字段名时，可以调用**FieldByName**来获取关于该字段的信息。

# 7.1 TADOTable组件概述

## (6) 关于立即更新和批更新

ADO数据集可支持两类更新模式：立即更新和批更新。

使用立即更新，一旦调用**Update**方法，对数据的所有更改被立即写入数据库。

使用批更新，可以将多个记录的更改存入缓存，然后用**UpdateBatch**方法一次写入到数据库。

默认情况下为立即更新模式。要设置为批更新模式，必须设置下列属性的值：

**CursorType**属性为**ctKeySet**（默认）或**ctStatic**；  
**LockType**属性为**ltBatchOptimistic**。

# 7.1 TADOTable组件概述

## 7.1.3 TADOTable组件的事件

TADOTable组件与其他ADO数据集组件具有完全相同的事件，如下表所示。

事 件	说 明
AfterCancel	在Cancel方法之后触发该事件
AfterClose	在Close方法之后触发该事件
AfterDelete	在Delete方法之后触发该事件
AfterEdit	在数据集设置为编辑模式之后触发该事件
AfterInsert	当一个新记录插入到该数据集中时触发该事件
AfterOpen	打开数据集之后触发该事件
AfterPost	在Post方法之后触发该事件（比如说，当当前记录被修改了且激活的记录被更新的时候将触发该事件）
AfterRefresh	Refresh方法之后触发该事件
AfterScroll	当光标定位被改变的时候触发该事件
BeforeCancel	当调用Cancel方法的时候，在执行Cancel行为之前触发该事件
BeforeClose	当调用Close方法的时候，在执行Close行为之前触发该事件
BeforeDelete	当调用Delete方法的时候，在执行Delete行为之前触发该事件

# 7.1 TADOTable组件概述

事 件	说 明
BeforeEdit	当调用Edit方法的时候，在执行Edit行为之前触发该事件
BeforeInsert	当调用Insert方法的时候，在记录被插入到数据集之前触发该事件
BeforeOpen	当调用open方法的时候，在数据集被打开之前触发该事件
BeforePost	当调用Post方法的时候，在记录被发送到数据集之前触发该事件
BeforeScroll	当调用Scroll的时候，在光标位置改变之前触发该事件
OnCalcFields	当Calculated字段需要指定一个值的时候触发该事件
OnDeleteError	当执行Delete操作的时候，如果发生了一个EDatabaseEngine异常，将由CheckOperation触发该事件
OnEditError	在修改或添加记录时产生了意外会触发该事件
OnEndOfRecordset	在记录指针试图移动到末尾之后时发生
OnFetchComplete	当一次从数据库读取数据的操作完成时触发该事件
OnFetchProgress	在异步方式运行时，数据集组件读取数据时会周期性触发该事件
OnFieldChangeComplete	当某个字段被更新后触发该事件
OnFilterRecord	编写这个事件处理程序来测试过滤条件对于每个记录的过滤过程
OnMoveComplete	在一次记录指针移动完成后触发该事件
OnNewRecord	当一个新的记录添加到数据集中的时候触发该事件
OnPostError	在提交修改的记录时产生了意外会触发该事件

# 7.1 TADOTable组件概述

事 件	说 明
OnRecordChangeComplete	当一条记录被修改完成后触发该事件
OnRecordsetChangeComplete	当记录刷新后触发该事件
OnRecordsetCreate	在记录集被初始化完成后触发该事件
OnWillChangeField	在字段更新之前发生
OnWillChangeRecord	在记录被改变之前发生
OnWillChangeRecordset	在记录集被更新之前发生
OnWillMove	在移动记录指针之前发生

# 7.2 记录的读取与修改

## 7.2.1 Field对象

**Field**对象对应着数据集中的字段。**Field**对象可以在应用程序运行的过程中动态地产生，也可以利用字段编辑器创建成为永久字段。

**TField**类封装了所有字段操作的行为和数据类型，它的属性、方法和事件被用于：

- (1) 改变数据集中字段的值。
- (2) 将字段的值由一种类型转换为另一种类型。
- (3) 验证用户为字段键入的数据。
- (4) 定义字段中的数据如何显示和编辑。
- (5) 计算字段的值，当**OnCalcFields**产生时。

## 7.2.2 使用Fields属性

**TFields**有两个基本属生。一个是**Count**属性，指明**Fields**对象中的字段数；一个是**Fields**属性，包含**Fields**对象

## 7.2 记录的读取与修改

所管理的字段列表，通过指定索引号，可以访问单字段，索引号从0开始。

### 7.2.3 使用FieldValues属性

使用FieldValues属性，可以通过字段名读取字段的值。如果要显示name字段的内容，可使用语句：

```
showmessage(adotable1.FieldValues['name']);
```

FieldValues是数据集默认属性，因此，使用下面的语句效果是一样的。

```
showmessage(adotable1['name']);
```

### 7.2.4 使用FieldByName方法

FieldByName方法返回指定名称的字段对象，因此，可以使用FieldByName方法完成对字段的操作。

# 7.3 记录的添加与删除

## 7.3.1 记录的添加

可以使用数据集组件的Append、Insert、AppendRecord、InsertRecord方法添加记录，其中Append是在表末尾添加一条空白记录，Insert是在表的当前位置添加一条空白记录，AppendRecord用于在表末追加一条记录并对其赋值，InsertRecord是在表的当前位置添加一条记录并对其赋值。

1. 使用Append和Insert方法添加记录
2. 使用AppendRecord、InsertRecord方法添加记录

## 7.3.2 记录的删除

1. 使用Delete方法
2. 使用DeleteRecords方法
  - (1) 删除当前记录
  - (2) 删除所有记录

## 7.4 数据查询

ADO数据集组件提供的有关数据查询的方法有3个：  
Locate方法、lookup方法和Seek方法。

### 7.4.1 Locate方法

定位一条记录并把这条记录作为当前记录。

### 7.4.2 Lookup方法

搜索符合条件的记录，获取指定字段的值。找到记录后，记录指针不移动。

### 7.4.3 Seek方法

搜索记录并移动数据集的指针。

## 7.5 记录指针

每一个激活的数据集都有一个记录指针指向数据集中当前记录。在窗体中，TDBEdit、TDBText、TEBMemo等组件显示的数据就是当前记录的字段值。

通过移动记录指针，可以改变当前记录。

例如：

在数据集组件ADOTable1中移动记录指针到第一条记录：

ADOTable1.First;

移动记录指针到最后一条记录：

ADOTable1.Last;

向后移一条记录：

ADOTable.Next;

向前移一条记录：

ADOTable.Prior;

向前移2条记录：

## 7.5 记录指针

ADOTable1.MoveBy(-2);

判断数据集为空:

if ADOTable1.Bof and ADOTable1.eof then  
    ShowMessage('记录集为空。');

## 7.6 数据过滤

当我们在操作数据集时，往往需要对数据进行筛选。选用数据集组件**Filter**属性、**Filtered**属性、**FilterOptions**属性可以实现该功能，其行为类似于SQL语句中的WHERE子句。

### 7.6.1 利用TADOTable的Filter属性和Filtered属性

#### 1. 在设计时设置Filter属性

例如，设置**Filter**属性为：**Country='China'**，然后改变**Filtered**属性为**True**。则只能看到对应的**Country**字段内容为‘China’的记录。

**FilterOptions**属性确定过滤方式，可以包含2个选项，若包含**foCaseInsensitive**，表示忽略字母大小写；若包含**foNoPartialCompare**，表示可以用来进行局部匹配，把星号（“\*”）当作一个字符，否则星号（“\*”）被当作一个掩码字符。

## 7.6 数据过滤

### 2. 在程序运行期间进行动态过滤

要在程序运行时改变**Filter**属性，这包括两种情况：

(1)操作符右边为常量，例如：

```
with ADODataSet1 do  
begin  
    Filtered:= False;  
    Filter := 'State='+QuotedStr('CA')+  
'OR'+ 'State='+QuotedStr('HI');  
    Filtered := True;  
end;
```

QuotedStr()的作用是将字符串转换为带引号的字符串。

也可以使用双单引号。比如：

```
ADOTable1.Filter := 'State="CA" OR State="HI";
```

## 7.6 数据过滤

(2)操作符右边不为常量，可能是通过一个变量指定的值，或由一输入框给出的值。这时需要用到**Format**函数。

其代码形式为：

```
procedure TForm1.filterClick(Sender: TObject);
```

```
var
```

```
    StateValue:string;
```

```
begin
```

```
    StateValue:='CA';
```

```
    ADOtable1.filtered:=False;
```

```
    ADOtable1.Filter:=Format('State="%s",[StateValue]);
```

```
    //可改为： ADOtable1.Filter:=
```

```
    'State='+QuotedStr(StateValue);
```

```
    ADOtable1.filtered:=True;
```

```
end;
```

# 7.6 数据过滤

## 7.6.2 用OnFilterRecord事件筛选

当Filtered属性由False变为True时触发该事件。在OnFilterRecord事件处理程序中可以设置筛选条件。例如：

```
procedure TForm1.ADOTable1FilterRecord(DataSet:
TDataSet;
  var Accept: Boolean);
begin
  Accept:=DataSet['State']='CA';
end;
```

## 7.7 应用实例

### 【例7-1】

编写一应用程序，对dbdemos.mdb数据库中的表Customer进行以下操作：根据输入的CustNo修改Phone和Contact字段的内容。

要完成以上功能，可以分两步操作：

（1）用一个文本编辑框输入CustNo，根据输入的CustNo查询表customer中对应的记录。将Phone和Contact字段的值显示在另外两个文本编辑框中。

（2）在文本编辑框中修改Phone和Contact的值后，再将修改后的内容写入到数据库中。为便于观察结果，在窗体中可放置一个DBGrid组件。

### 【实现步骤】

（1）启动Delphi 7

（2）新建应用程序

## 7.7 应用实例

(3) 定制窗体          窗体界面如图7-2所示。

The screenshot shows a window titled "Form1" with a blue title bar. Inside, there is a table with the following data:

CustNo	Company	Phone	Contact
1221	Kauai Dive Shoppe	808-555-0269	Erica Norman
1231	Unisco	809-555-3915	George Weathers
1351	Sight Diver	357-6-876708	Phyllis Spooner
1354	Cayman Divers World Unlimited	011-5-697044	Joe Bailey
1356	Tom Sawyer Diving Centre	504-798-3022	Chris Thomas
1380	Blue Jack Aqua Center	401-609-7623	Ernest Barratt
1384	VIP Divers Club	809-453-5976	Russell Christopher
1510	Ocean Paradise	808-555-8231	Paul Gardner
1513	Fantastique Aquatica	057-1-773434	Susan Wong
1551	Marmot Divers Club	416-698-0399-78	Joyce Marsh-78

Below the table, there is a form area with a dotted background. It contains the following elements:

- Text label: "请输入 CustNo" (Please enter CustNo)
- Text input field for CustNo
- Button: "查询" (Query)
- Text label: "Phone:"
- Text input field for Phone
- Text label: "Contact:"
- Text input field for Contact
- Button: "修改" (Modify)
- Two small icons: a printer icon and a refresh/reset icon.

图7-2 窗体界面

# 7.7 应用实例

表7-8 各组件属性设置

组件名	属 性	属 性 值	说 明
ADOTable1	ConnectionString		使用连接串生成器建立数据库连接。
	TableName	'customer'	指定打开的表名: customer
	Active	True	
DataSource1	DataSet	ADOTable1	指定使用的数据集为ADOTable1。
DBGrid1	DataSource	DataSource1	
Label1	Caption	'请输入CustNo'	
Label2	Caption	'Phone '	
Label3	Caption	'Contact'	
Edit1	Text	"	
Edit2	Text	"	
Edit3	Text	"	
Button1	Caption	'查询'	
Button2	Caption	'修改'	

## 7.7 应用实例

### 【例7-2】

在浏览数据表Customer时，根据输入的Country值过滤记录。

要完成以上功能，可以用一个文本编辑框输入Country值，根据输入的Country来设置过滤条件。

### 【实现步骤】

(1) 启动Delphi 7

(2) 新建应用程序

(3) 定制窗体

窗体界面如图7-3所示。

## 7.7 应用实例

The screenshot shows a Windows application window titled "Form1". The window contains a data grid with the following columns: CustNo, Company, Country, State, City, Zip, and Add. The grid lists 15 customer records. Below the grid, there is a filter section with the label "设置过滤" (Set Filter) and a text input field labeled "请输入Country" (Please enter Country). To the right of the input field is a button labeled "执行" (Execute).

CustNo	Company	Country	State	City	Zip	Add
1221	Kauai Dive Shoppe	US	HI	Kapaa Kauai	94766-1234	4-97
1231	Unisco	Bahamas		Freeport		PO E
1351	Sight Diver	Cyprus		Kato Paphos		1 Ne
1354	Cayman Divers World Unlimited	British West Indies		Grand Cayman		PO E
1356	Tom Sawyer Diving Centre	US Virgin Islands	St. Croix	Christiansted	00820	632-
1380	Blue Jack Aqua Center	US	HI	Waipahu	99776	23-7
1384	VIP Divers Club	US Virgin Islands	St. Croix	Christiansted	02800	32 M
1510	Ocean Paradise	US	HI	Kailua-Kona	94756	PO E
1513	Fantastique Aquatica	Columbia		Bogota		Z32
1551	Marmot Divers Club	Canada	Ontario	Kitchener	G3N 2E1	872
1560	The Depth Charge	US	FL	Marathon	35003	1524
1563	Blue Sports	US	OR	Giribaldi	91187	203
1624	Makai SCUBA Club	US	HI	Kailua-Kona	94756	PO E

设置过滤

请输入Country

执行

图7-3 窗体界面

# 7.7 应用实例

表7-9 各组件属性设置

组件名	属 性	属 性 值	说 明
ADOTable1	ConnectionString		使用连接串生成器建立数据库连接。
	TableName	'customer'	指定打开的表名: customer
	Active	True	
DataSource1	DataSet	ADOTable1	指定使用的数据集为ADOTable1。
DBGrid1	DataSource	DataSource1	
GroupBox1	Caption	'设置过滤'	
Label1	Caption	'请输入Country '	
Edit1	Text	"	
Button1	Caption	'执行'	

## 7.7 应用实例

---

(4) 编写代码

(5) 运行程序

# 本章实训指导

1. 理解数据集组件，了解数据集组件共有的方法、属性和事件。
2. 掌握TADOTable组件操作数据的基本方法：读取与修改记录、添加与删除记录、查询记录、根据条件过滤记录。
3. 理解指针的概念，掌握指针的基本操作。
4. 使用所掌握的TADOTable组件的方法，为通讯录程序添加数据浏览和查询功能模块。

# 第8章 ADO组件的使用

本章主要内容：

- TADOConnection组件的使用
- TADOCommand组件的使用
- TADODataSet组件的使用
- TADOQuery组件的使用

# 8.1 TADOConnection组件概述

TADOConnection组件用于建立数据库的连接，该连接可被多个数据集所共享。

TADOConnection 组件提供的功能：

- 控制数据库的连接
- 控制服务器的注册
- 管理事务
- 为关联的数据集提供数据库连接
- 将SQL命令发送到数据库中
- 从数据库中提取数据

## 8.1.1 TADOConnection 的常用属性

### 1. Attributes属性

此属性用于设置连接的数据库的自动处理事务的行为，它是TXactAttributes类型的集合，包括两个集合元素。

# 8.1 TADOConnection组件概述

(1) **xaCommitRetaining** 表示提交一个事务后自动开始一个新的事务;

(2) **xaAbortRetaining** 表示回退一个事务的同时将开始一个新的事务。

## 2. CommandTimeout属性

命令超时属性，用于设置一个命令执行时所能等待的最大时间值。

例如，以下语句将命令执行时间设为120秒。

```
ADOConnection1.CommandTimeout:=120;
```

## 3. Connected属性

标识和数据库的连接是否处于激活状态。

## 4. ConnectionString属性

**ConnectionString**（连接字符串）用于指定数据库的连接信息。

# 8.1 TADOConnection组件概述

连接串的标准调用方式为：

ADOConnection1.ConnectionString :=

'Provider=ProviderRef;Remote Server= ServerRef ';

其中，连接串支持的常用参数见下表。

参数	说明
Provider	数据提供者名称，例如：MSDASQL.1
Password	登录数据库的口令
Persist Security Info	支持安全登录
User ID	登录数据库用户名
Data Source	数据源名称，数据源的设置需要额外的操作。

建议使用连接串生成器创建连接字符串。

# 8.1 TADOConnection组件概述

## 5. ConnectOptions属性

指定数据库连接是按照同步方式还是异步方式连接。

## 6. ConnectionTimeout属性

指定建立连接的最大允许时间。

## 7. CursorLocation属性

指定数据库游标是采用客户端模式还是服务器端模式。

## 8. DefaultDatabase属性

表明数据源成功连接后，数据库的路径，这是由数据源自动赋值的。

## 9. IsolationLevel属性

指定不同事务之间的相互独立的级别。

## 10. KeepConnection属性

指定如果在没有打开数据集的情况下是否仍然保持数据库的连接。

# 8.1 TADOConnection组件概述

## 11. LoginPrompt属性

指定在每次建立连接时是否弹出登录对话框提示用户登录，如果设为**False**则必须在 **ConnectionString**中指定登录数据库的用户名和密码。

## 12. Mode属性

指定连接对数据库的操作权限。

## 13. InTransactin属性

表明TADOConnection组件是否处于处理事务的状态中。

## 14. DataSets属性和DataSetCount属性

**DataSets**属性包含当前使用这个连接组件的数据集的列表。

**DataSetCount**属性表示该连接组件关联的数据集数目。

。

# 8.1 TADOConnection组件概述

## 8.1.2 TADOConnection 的主要方法

### 1. BeginTrans方法

开始启动一个新的事务，必须保证数据连接处于激活状态。

### 2. Cancel方法

撤消正在建立的到数据库的连接。

### 3. CommitTrans方法

向数据库提交一个事务，提交成功后，在事务中对数据库所作的修改则写入数据库中，同时一个事务也结束。

### 4. Execute方法

用来执行SQL命令。

### 5. GetProcedureNames方法

获取数据库服务器上的存储过程名称。

### 6. GetTableNames方法

获取数据库中的数据表名称。

# 8.1 TADOConnection组件概述

## 7. GetFieldNames方法

获取当前连接的数据库中某个指定的数据表的所有字段名。

## 8. Open方法

打开一个连接。

## 9. RollbackTrans方法

撤回一个没有全部执行的事务，事务撤回之后，事务中所作的任何修改都不会写入数据库。

## 10. Close方法

关闭一个连接。

## 8.1.3 ADOConnection 的主要事件

ADOConnection组件的主要事件如表8-4所示。

# 8.1 TADOConnection组件概述

表8-4 ADOConnection组件的主要事件

事件	说明
AfterConnect	发生在一个连接建立后
AfterDisconnect	发生在断开连接后
BeforeConnect	发生在连接建立前
BeforeDisconnect	发生在断开连接前
OnBeginTransComplete	发生在开始一个事务时
OnCommitTransComplete	发生在提交事务成功时
OnConnectComplete	发生在连接完成时
OnDisconnect	发生在连接断开时
OnExecuteComplete	发生在一个命令执行后
OnInfoMessage	发生在收到数据库的消息
OnLogin	发生在用户登录数据库的时候
OnRollbackTransComplete	发生在一个事务撤回之后
OnWillConnect	发生在发出一个连接数据请求的时候
OnWillExecute	发生在数据库收到一个 SQL 命令并将要执行的之前

## 8.2 TADOConnection组件的使用

### 8.2.1 控制服务器的注册

许多远程数据库服务器包含安全特性以阻止未经授权的访问。通常，这些服务器在允许数据库访问之前要求输入用户名和密码进行注册。

1. 在ConnectionString中加入用户名和密码。

例：

```
procedure TForm1.Button2Click(Sender: TObject);  
begin  
  adoconnection1.ConnectionString:='Provider=SQLOLE  
DB.1;Persist Security Info=False;Initial  
Catalog=pubs;Data Source=qq-1;User  
ID=sa;password=123456';  
  adoconnection1.Open();  
end;
```

## 8.2 TADOConnection组件的使用

2. 将用户名和密码作为Open方法的参数。

例：

```
ADOConnection1.Open('sa', '123456');
```

3. 使用OnWillConnect事件

```
procedure
```

```
TForm1.ADOConnection1WillConnect(Connection:  
TADOConnection;
```

```
var ConnectionString, UserID, Password: WideString;
```

```
var ConnectOptions: TConnectOption; var  
EventStatus: TEventStatus);
```

```
begin
```

```
UserID:='sa';
```

```
password:='123456';
```

```
end;
```

## 8.2 TADOConnection组件的使用

### 8.2.2 控制关联的数据集

可以使用DataSets属性与DataSetCount属性一起遍历当前所有激活的数据集。

#### 【例8-1】

编写一个程序，通过TADOConnection组件连接到数据库dbdemos.mdb，并调用GetTableNames方法获取数据库中所有表的名称，显示在一个列表框中。用一个TADOTable组件作为数据集，通过双击列表框中的表名来选取要打开的数据表，并用一个TDBGrid组件显示表中的记录。

#### 【实现步骤】

- (1) 新建应用程序
- (2) 定制窗体

## 8.2 TADOConnection组件的使用

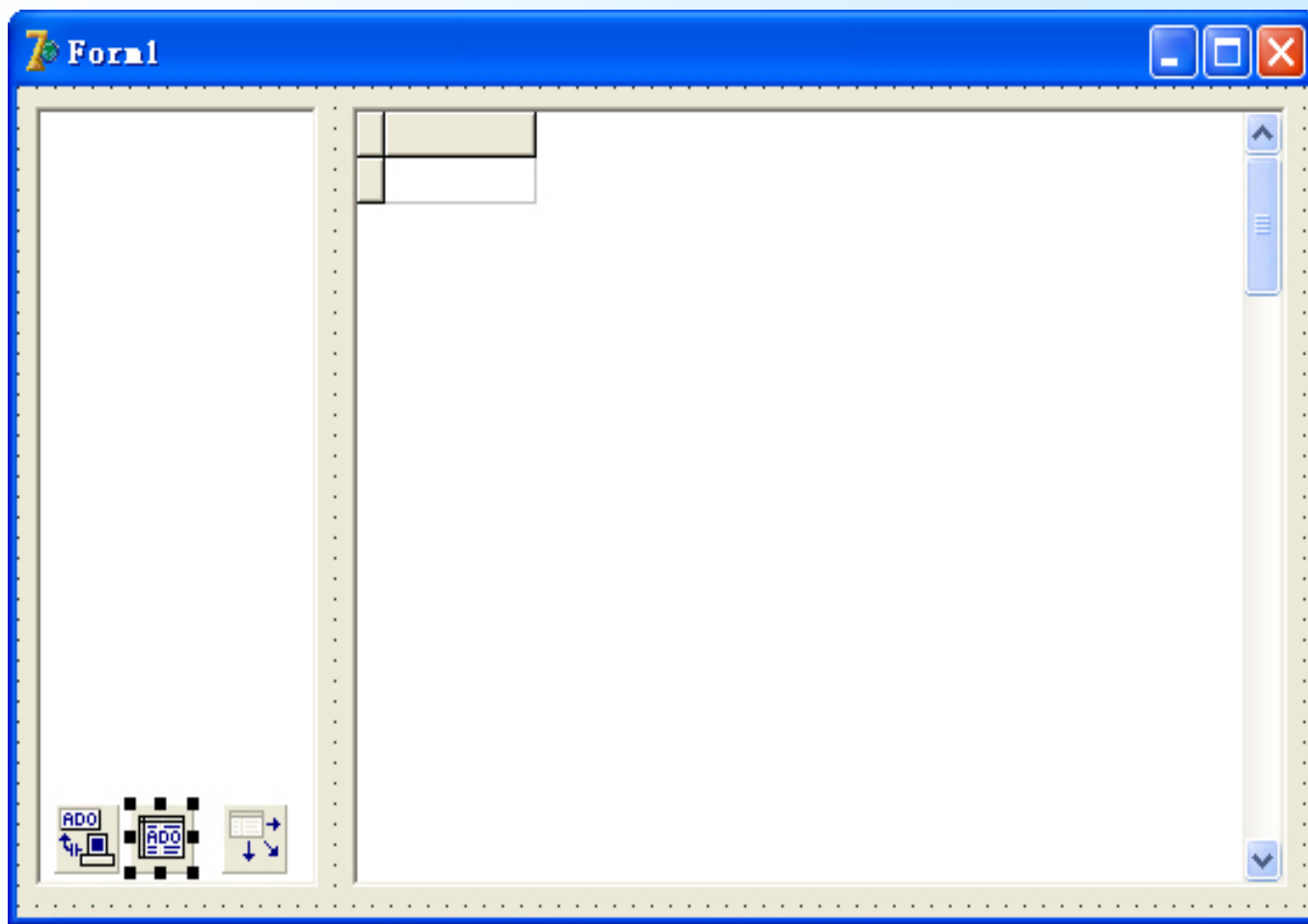


图8-1 设计窗体界面

## 8.2 TADOConnection组件的使用

表8-5 各组件属性设置

组件名	属 性	属 性 值	说 明
ADOConnect ion1	ConnectionStri ng		使用 <b>ConnectionString</b> 设置对话框设置该属性，连接到数据库dbdemos.mdb。
ADOTable1	Connection	ADOConnectio n1	指定使用的数据连接组件。
DataSource1	DataSet	ADOTable1	指定使用的数据集为ADOTable1。
DBGrid1	DataSource	DataSource1	为表格指定数据源。
ListBox1	Font.size	12	设定列表框中文字显示大小。
	Font.Name	Arial	设定列表框中文字显示字体。

## 8.2 TADOConnection组件的使用

- (3) 编写代码
- (4) 运行程序

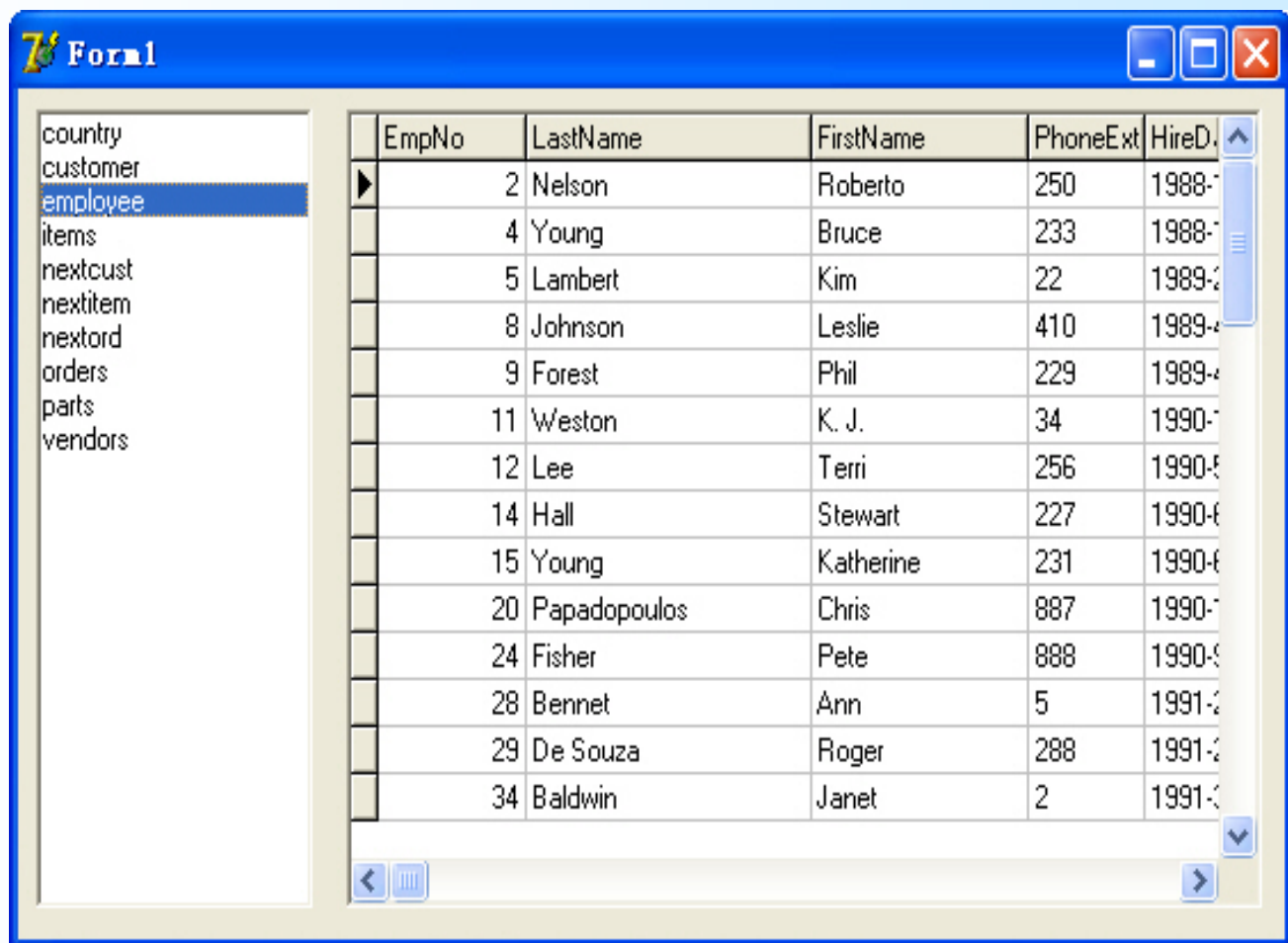
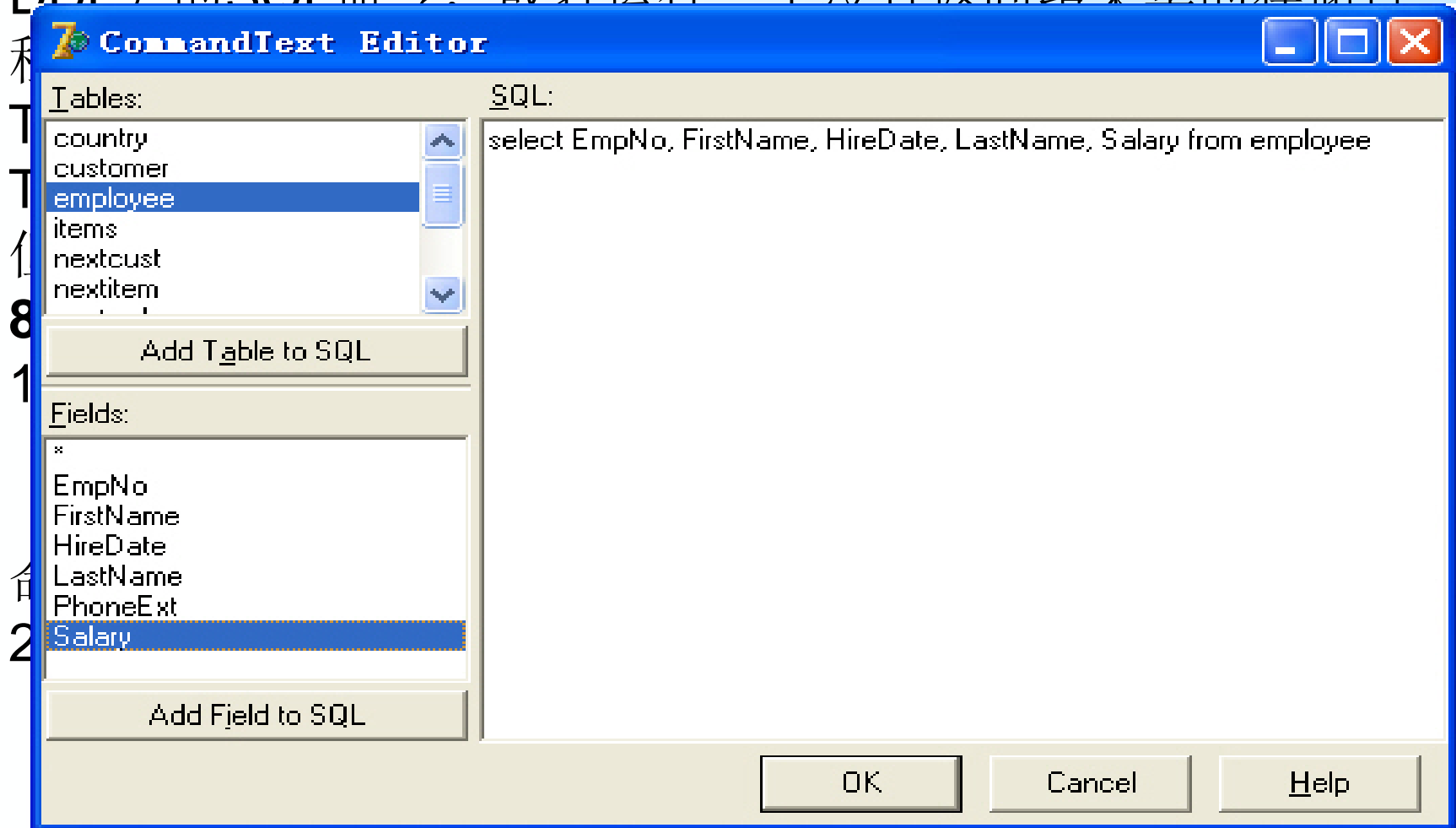


图8-2 程序运行界面

## 8.3 TADOCCommand组件概述

TADOCCommand组件主要用于运行一些数据定义语言（DDL）的SQL命令，或者运行一个没有返回结果集的存储过程。



## 8.3 TADOCommand组件概述

CommandType的选项及说明

选项	说明
cmdUnknown	CommandText中的命令类型是未知的
cmdText	CommandText中是一个SQL语句
cmdTable	CommandText中指定的是一个表的名称
cmdStoredProc	CommandText中指定的是一个存储过程的名称
cmdFile	CommandText中指定的是保存数据集的文件名
cmdTableDirect	CommandText中指定的是表的名称，并返回所有的列

### 3. Connection属性

指定所使用的数据连接组件的名称，即TADOConnection 组件的名称。

### 4. ConnectionString属性

ConnectionString（连接字符串）用于指定数据库的连接信息。

## 8.3 TADOCommand组件概述

### 5. CammandTimeout属性

CammandTimeout属性是一个整型数，指定执行一个命令的最大允许时间，默认值是30秒。在TADOCommand组件通过TADOConnection 组件与数据连接的情况下，如果同时两者指定了CammandTimeout属性，则优先考TADOCommand组件中设置的值。

### 6. ExecuteOption属性

指定调用Execute方法时的命令特征。

### 7. ParamCheck属性

指定在运行时如果SQL命令被改变，是否需要重建TADOCommand组件的参数列表。

### 8. Parameters属性

在 SQL 命令中或在存储过程中需要传递参数的时候才需要设置这个值，并且，在命令类型（CommandType）指定为cmdText或 cmdStoredProc时，参数才有效。

## 8.3 TADOCCommand组件概述

### 8.3.2 TADOCCommand 的主要方法

#### 1. Cancel方法

中止一个正在执行的命令。

#### 2. Execute方法

执行TADOCCommand 组件所包含的命令。

## 8.4 TADOCCommand组件的使用

TADOCCommand组件封装了ADO技术中的Command对象，通过它可以在数据库上执行SQL语句。在程序中使用TADOCCommand组件，需要完成以下步骤：

- （1）使用Connection属性连接到一个TADOCConnetion组件或使用ConnetionString属性直接连接数据库；
- （2）设置CommandType属性，指定CommandText属性中包括命令类型；
- （3）为CommandText属性设置命令文本，可以是SQL语句、表名、存储过程名；
- （4）调用Excute方法执行命令。

## 8.5 TADODataSet组件概述

TADODataSet可通过SQL命令从一个数据表或者是多个数据表获取数据集，也可以直接获取整个数据表的数据，它还可以执行存储过程，也可以从磁盘文件获取数据。

### 8.5.1 TADODataSet 的主要属性

#### 1. CommandText

CommandText属性指定数据集中所包含的命令，可以是SQL语句、一个表名或者一个存储过程名。

常用的调用形式为：

```
with ADODataSet1 do
```

```
begin
```

```
    Close;
```

```
    CommandType := cmdText;
```

```
    CommandText := 'SELECT * FROM Customer';
```

```
    Open;
```

```
end;
```

# 8.5 TADODataset组件概述

## 2. DataSource

设置数据源用来自另一个数据集的字段值去自动填充查询的参数。

### 8.5.2 TADODataset组件的重要方法

TADODataset的方法与前面介绍TADOTable的属性一样。

# 8.6 TADODataSet组件的使用

## 8.6.1 TADODataSet组件的数据集类型

TADODataSet是一个多用途的数据集，通过设置CommandType和CommandText属性，它可以作为表类型、查询类型、存储过程类型和文件类型的数据集使用。

### 1. 表类型数据集

作为表类型数据集时，TADODataSet从单一的数据表中获取所有行和列。

### 2. 查询类型数据集

TADODataSet也可以执行SQL查询命令以获取数据集，使用时需要先设置CommandType属性为cmdText，然后在CommandText属性中设置要执行的SQL命令。

### 3. 存储过程类型数据集

TADODataSet需要指定所执行的存储过程的名称。

## 8.6 TADODataset组件的使用

### 4. 文件类型数据集

使用这种类型数据集，需先设置CommandType属性为cmdFile，然后在CommandText属性中设置要打开的文件名。

#### 【例8-2】

在数据库dbdemos.mdb中，表customer存储的是顾客信息，表orders存储的是订单信息。编程实现能方便地通过选定顾客来浏览其订单信息。

将TADODataset组件的Parameters属性与DataSource属性配合使用，可以建立主/从关系的数据集。

#### 【实现步骤】

- (1) 建立应用程序
- (2) 定制窗体

## 8.6 TADODataSet组件的使用

Form1

主表

	CustNo	Company	Addr1	Addr2
▶	1221	Kauai Dive Shoppe	4-976 Sugarloaf Hwy	Suite 103
	1231	Unisco	PO Box Z-547	
	1351	Sight Diver	1 Neptune Lane	
	1354	Cayman Divers World Unlimited	PO Box 541	
	1356	Tom Sawyer Diving Centre	632-1 Third Frydenhoj	
	1380	Blue Jack Aqua Center	23-738 Paddington Lane	Suite 310
	1384	VIP Divers Club	32 Main St.	
	1510	Ocean Paradise	PO Box 8745	
	1513	Fantastique Aquatica	Z32 999 #12A-77 A.A.	

从表中显示的内容由主表中的CustNO字段控制

	OrderNo	CustNo	SaleDate	ShipDate	EmpNo	ShipToContact
▶	1023	1221	1988-7-1	1988-7-2	5	
	1076	1221	1994-12-16	1989-4-26	9	
	1123	1221	1993-8-24	1993-8-24	121	
	1169	1221	1994-7-6	1994-7-6	12	
	1176	1221	1994-7-26	1994-7-26	52	

图8-4 设计窗体界面

# 8.6 TADODataSet组件的使用

表8-7 各组件属性设置

组件名	属 性	属 性 值	说 明
ADOConnection1	ConnectionString		使用ConnectionString设置对话框设置该属性，连接到数据库dbdemos.mdb。
ADODataset1	Connection	ADOConnection1	指定使用的数据连接组件。
	CommandType	cmdTable	说明CommandText属性的值是表的名称
	CommandText	Customer	指定打开的数据表名
	Active	True	打开数据集。
DataSource1	DataSet	ADODataset1	指定使用的数据集。
DBGrid1	DataSource	DataSource1	为表格指定数据源，显示主表。
ADODataset2	Connection	ADOConnection1	指定使用的数据连接组件。
	CommandType	cmdText	说明CommandText属性值是SQL语句
	CommandText	select * from orders where CustNo=:CustNo	添加SQL查询语句

## 8.6 TADODataset组件的使用

组件名	属 性	属 性 值	说 明
ADODataset2	DataSource	DataSource1	指定与主表相连的数据源
	Parameters.item. Name	CustNo	与主表相关联的字段名
	Parameters.item. DataType	ftInteger	字段数据类型
	Active	True	打开数据集。
DataSource2	DataSet	ADODataset2	指定使用的数据集
DBGrid2	DataSource	DataSource2	为表格指定数据源。
TLabel1	Caption	主表	
TLabel2	Caption	从表中显示的 内容由主表中的 CustNo字段控制	

## 8.6 TADODataSet组件的使用

### (3) 运行程序

Form1

主表

	CustNo	Company	Addr1	Addr2
	1221	Kauai Dive Shoppe	4-976 Sugarloaf Hwy	Suite 103
	1231	Unisco	PO Box Z-547	
	1351	Sight Diver	1 Neptune Lane	
	1354	Cayman Divers World Unlimited	PO Box 541	
	1356	Tom Sawyer Diving Centre	632-1 Third Frydenhoj	
	1380	Blue Jack Aqua Center	23-738 Paddington Lane	Suite 310
▶	1384	VIP Divers Club	32 Main St.	
	1510	Ocean Paradise	PO Box 8745	
	1513	Fantastique Aquatica	Z32 999 #12A-77 A.A.	

关联字段

从表中显示的内容由主表中的CustNO字段控制

	OrderNo	CustNo	SaleDate	ShipDate	EmpNo	ShipToContact
▶	1007	1384	1988-5-1	1988-5-2	45	
	1027	1384	1988-7-7	1988-7-8	34	
	1033	1384	1988-8-1	1988-8-2	138	
	1100	1384	1989-6-20	1989-6-20	9	
	1107	1384	1992-10-24	1992-10-24	134	

图8-5 程序运行界面

## 8.7 TADOQuery组件概述

TADOQuery 组件借助于SQL语言的的强大功能访问多个数据表，可以实现数据浏览、修改和删除等操作，而且TADOQuery组件也可以实现参数查询。

通常情况下，使用TADOQuery组件是为了从数据集中查询一部分字段或记录，也可以使用INSERT、DELETE、UPDATE、ALTER TABLE等SQL命令实现数据库的更新、插入和删除记录的操作。

### 8.7.1 主要属性

#### 1. SQL属性

SQL属性是TStrings类型的变量，包含了TADOQuery 组件要执行的SQL命令。

请看下面的例子。

## 8.7 TADOQuery组件概述

```
with ADOQuery1 do
begin
    //重新写入SQL的时候 必须关闭原来的查询
    Close;
    SQL.Clear; //清除原来的SQL命令
    SQL.Add('SELECT EmpNo, LastName, FirstName,
    HireDate'); //设置新SQL命令
    SQL.Add('FROM Employee');
    Open; //执行有返回结果的SQL命令,获得想要的查
    询结果.
end;
```

# 8.7 TADOQuery组件概述

## 2. Parameters属性

Parameters属性中保存了SQL属性中的SQL命令中执行所需的参数。

## 3. RowsAffected属性

返回最近一次查询所影响的记录数。

## 8.7.2 主要方法

### 1. Open方法

Open方法用于执行SQL属性所指定的SQL命令，要返回记录集。

### 2. ExecSQL方法

ExecSQL方法用于执行SQL属性所指定的SQL命令,不需要返回记录集。如Insert、Update或删除等命令。

## 8.8 TADOQuery组件的使用

### 8.8.1 执行不需要返回结果的查询

通过调用ExecSQL方法，TADOQuery组件可以执行Insert、Update或Delete等SQL命令，这些命令执行后不会返回数据集。

#### 【例8-3】

编写应用程序用于修改数据库dbdemos.mdb中的表employee中的LastName字段。要求输入员工编号和姓，然后根据员工编号修改相应的记录。

#### 【实现步骤】

- (1) 创建应用程序
- (2) 定制窗体

## 8.8 TADOQuery组件的使用

The screenshot shows a Visual Basic form titled "Form1" with a blue title bar. The form contains a data grid displaying employee information. The grid has six columns: EmpNo, LastName, FirstName, PhoneExt, and HireDate. The data is as follows:

EmpNo	LastName	FirstName	PhoneExt	HireDate
2	Nelson	Roberto	250	1988-12-28
4	Young	Bruce	233	1988-12-28
5	Lambert	Kim	22	1989-2-6
8	Johnson	Leslie	410	1989-4-5
9	Forest	Phil	229	1989-4-17
11	Weston	K. J.	34	1990-1-17
12	Lee	Terri	256	1990-5-1
14	Hall	Stewart	227	1990-6-4
15	Young	Katherine	231	1990-6-14
20	Papadopoulos	Chris	887	1990-1-1

Below the grid, there are two input fields with labels "请输入员工编号:" and "请输入员工 LastName:". To the right of these fields is an "Update" button. The form also includes several icons for data operations, including a "ADO" icon with a question mark and a "ADO" icon with a plus sign.

图8-6 设计窗体界面

# 8.8 TADOQuery组件的使用

表8-8 各组件属性设置

组件名	属 性	属 性 值	说 明
ADOConnection1	ConnectionString		使用ConnectionString设置对话框设置该属性，连接到数据库dbdemos.mdb。
ADOTable1	Connection	ADOConnection1	指定使用的数据连接组件。
	TableName	employee	指定打开的数据表名
	Active	True	打开数据集
DataSource1	DataSet	ADOTable1	指定使用的数据集为ADOTable1。
DBGrid1	DataSource	DataSource1	为表格指定数据源。
ADOQuery1	Connection	ADOConnection1	指定使用的数据连接组件。
btnUpdate	Caption	UpDate	
Label1	Caption	请输入员工编号：	
Label2	Caption	请输入员工 LastName：	
edtNo	Text		为空
edtLastname	Text		为空

## 8.8 TADOQuery组件的使用

(3) 编写代码

(4) 运行程序



The screenshot shows a Java Swing window titled "Form1" with a blue title bar. Inside the window, there is a table with 6 columns: EmpNo, LastName, FirstName, PhoneExt, and HireDate. The table contains 10 rows of employee data. The row with EmpNo 109 (Hall, Kelly) is selected. Below the table, there are two input fields: "请输入员工编号:" with the value "109" and "请输入员工 LastName:" with the value "Hall". To the right of these fields is an "Update" button.

EmpNo	LastName	FirstName	PhoneExt	HireDate
83	Bishop	Dana	290	1992-6-1
85	MacDonald	Mary S.	477	1992-6-1
94	Williams	Randy	892	1992-8-8
105	Bender	Oliver H.	255	1992-10-8
107	Cook	Kevin	894	1993-2-1
109	Hall	Kelly	202	1993-2-4
110	Ichida	Yuki	22	1993-2-4
113	Page	Mary	845	1993-4-12
114	Parker	Bill	247	1993-6-1
118	Yamamoto	Takashi	23	1993-7-1

请输入员工编号: 109

请输入员工 LastName: Hall

Update

图8-7 程序运行界面

## 8.8 TADOQuery组件的使用

### 8.8.2 执行有返回结果的查询

通过调用Open方法或设置Active属性为True，TADOQuery组件可以执行Select命令和存储过程，并返回结果数据集。TADOQuery组件还可使用参数化查询，不需要修改SQL语句，给定不同的参数值，就可以获得不同的查询结果。

# 本章实训指导

- 1、掌握TADOConnection组件连接ACCESS数据库和MS SQL数据库服务器的方法。
- 2、掌握TADOCommand组件执行SQL命令方法。
- 3、掌握TADODataset组件的用法。
- 4、掌握TADOQuery组件的用法。
- 5、用TADODataset组件和TADOQuery组件重写通讯录程序，体会在功能实现上与TADOTable组件有何不同。

# 第9章 报表设计

本章主要内容:

- Rave Delphi组件
- Rave Reports组件之间的关系
- Rave报表应用程序结构
- Rave报表设计器的使用
- 如何组织数据库数据生成报表，并打印输出



# 9.1 Rave介绍

RAVE报表处理功能包括:

- 翻转文字注释
- 全图显示
- 版面调整
- 页面精确定位
- 打印配置
- 字体控制
- 打印预览
- 可重利用的报表内容
- 以PDF、HTML、RTF和text文件格式输出报表

# 9.1 Rave介绍

## 9.1.2 Rave报表设计器简介

Rave有自己的可视报表设计器，以自己的文件格式保存报表，扩展名为(.RAV)。

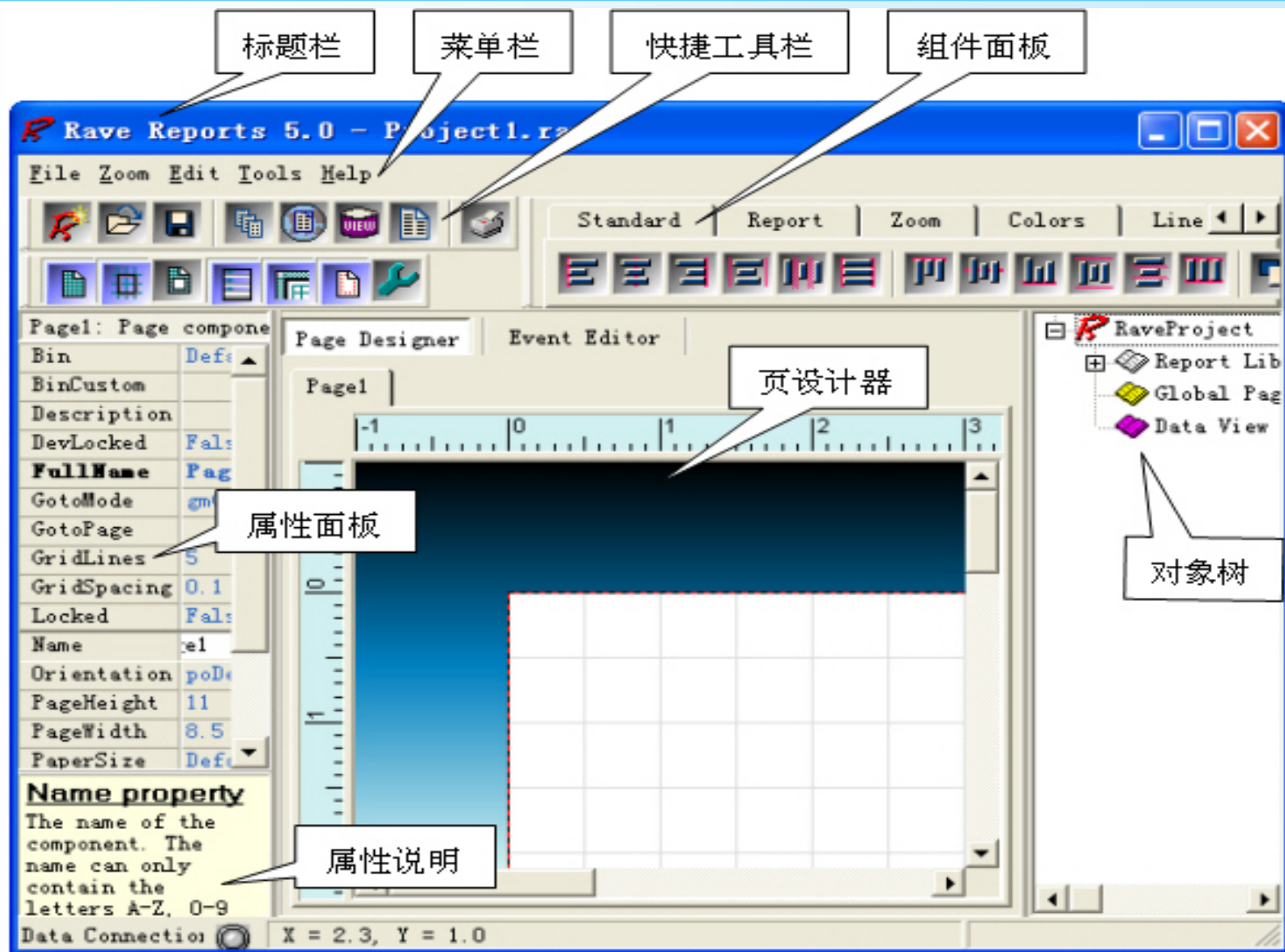
启动报表生成器的方式有两种：

（1）通过Delphi的工具菜单启动，选择【Tools】、【Rave Designer】。

（2）通过RvProject组件启动。必须先窗体中添加RvProject组件，然后双击该组件以启动Rave报表设计器。

Rave报表设计器的集成开发环境的界面包括标题栏、菜单栏、快捷工具栏、组件面板、属性面板和一些窗口。Rave5.0设计界面如下图所示。

# 9.1 Rave介绍



## 9.2 Rave Delphi组件

### 9.2.1 Rave Reports组件概述

在Delphi中，应用程序可以通过一组Rave Reports组件操纵报表，这些组件位于RAVE组件面板上，可以分为4类

:

- 引擎类组件（engine）
- 表现类组件（render）
- 数据连接类组件（data connection）
- Rave项目组件(Rave project)

各类组件之间的关系如图9-3所示。

## 9.2 Rave Delphi组件

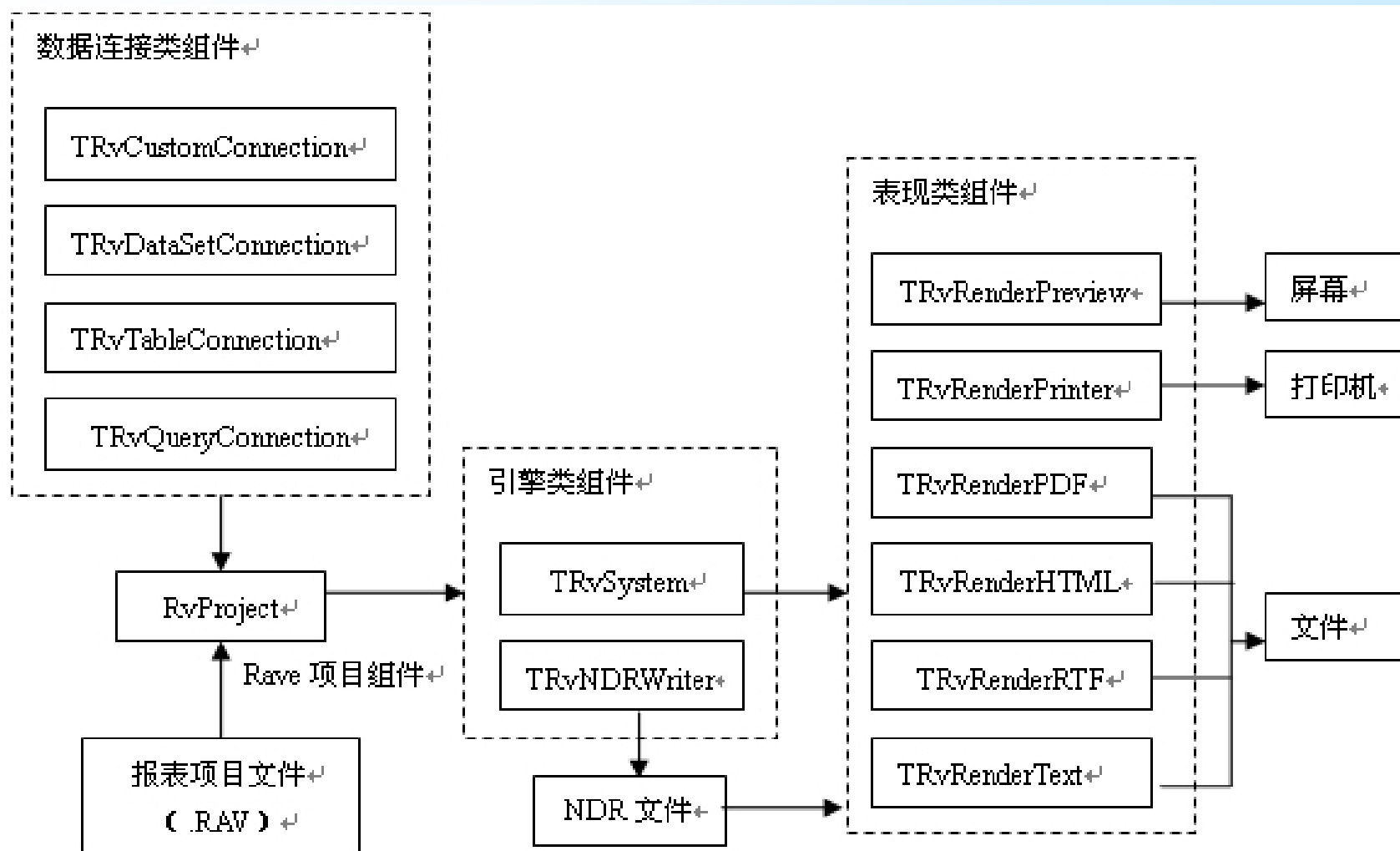


图 9-3 Rave Reports 组件之间的关系

# 9.2 Rave Delphi组件

## 9.2.2 TRvProject组件

### 1. TRvProject组件的属性

#### (1) DLLFile属性

如果LoadDesigner属性被设为True，这个属性用于设置报表设计器的动态链接库的文件名。

#### (2) Engine属性

当使用TrvProject组件打印报表时，指定使用的报表引擎，默认情况下是RvSystem。

#### (3) ProjectFile属性

指定报表项目文件。

#### (4) StoreRAV属性

如果要将报表项目文件内嵌到应用程序的EXE文件中，则本属性用于设置报表项目文件名。

#### (5) LoadDesigner属性

表示是否允许最终用户调用报表设计器。

## 9.2 Rave Delphi组件

### 2. TRvProject组件的方法

#### (1) SelectReport方法

SelectReport方法用于选择报表。

例：

```
procedure TFormMain.btnExecuteClick(Sender:
TObject);
begin
  RvProject.Open; //打开报表项目
  RvProject.SelectReport('Report1',False);
  //通过报表名称选择特定的报表
  RvProject.SetParam('Name','Leonel');
  //给特定参数设定值
  RvProject.Execute; //执行报表
  RvProject.Close; //关闭报表项目
end;
```

## 9.2 Rave Delphi组件

### (2) GetReportList方法

GetReportList方法同来获取报表列表。

例：

Var

TempReportList:TstringList;

Begin

TempReportList := TstringList.Create;

RvProject1.GetReportList(TempReportList,False);

End;

## 9.2 Rave Delphi组件

### (3) Execute方法

这个方法用于打印当前选定的**Rave**报表。

例：

**RvProject1.Execute ;**

就可以完成相应的打印工作，而这些具体的打印工作都通过**Rave**内部的操作进行处理。

### (4) ExecuteReport方法

这个方法用于打印指定名称的**Rave**报表。

### (5) Close方法

打印完成相应的报表后，需要调用**Close**方法关闭相应的报表项目文件。

### (6) Open方法

可以使用**Open**方法打开一个关闭的报表项目文件，以供打印和修改。

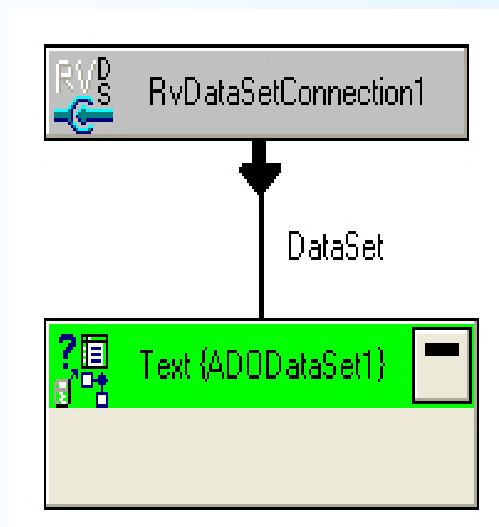
# 9.2 Rave Delphi组件

## 9.2.3 数据连接组件

Rave数据来源于应用程序，是通过数据连接组件与数据集组件相连来组织数据的。数据连接组件包括：

- TrvCustomConnection
- TrvDataSetConnection
- TRvQueryConnection
- TrvTableConnection

DataSet属性是它们共同的属性，用于指定连接的数据集。



## 9.2 Rave Delphi组件

### 9.2.4 引擎类组件

引擎类组件与TrvProject组件相互配合使用，在其中不仅可以设置相应的打印参数，而且可以设置最终报表的打印机配置、纸张大小及系统的设置。

#### 1. TRvSystem组件

TRvSystem可以发送报表到打印机或预览屏幕，并同时显示它们的设置及状态屏幕。

(1) TRvSystem组件的属性：

- DefaultDest属性
- SystemFiler属性
- SystemOptions属性
- SystemPreview属性
- SystemPrinter属性
- SystemSetups属性

## 9.2 Rave Delphi组件

- TitlePreview属性
- TitleSetup属性
- TitleStatus属性

### (2) TRvSystem组件的事件

OverridePreview, OverrideSetup 和 OverrideStatus 允许程序员替换Rave提供的缺省的屏幕。

### 2. TRvNDRWriter组件

TRvNDRWriter组件和TRvRenderPrinter及TrvRenderPreview配合使用，用特定的二进制格式存储报表直到它被打印或预览。使用TRvNDRWriter组件的报表处理过程如图9-5所示。

## 9.2 Rave Delphi组件

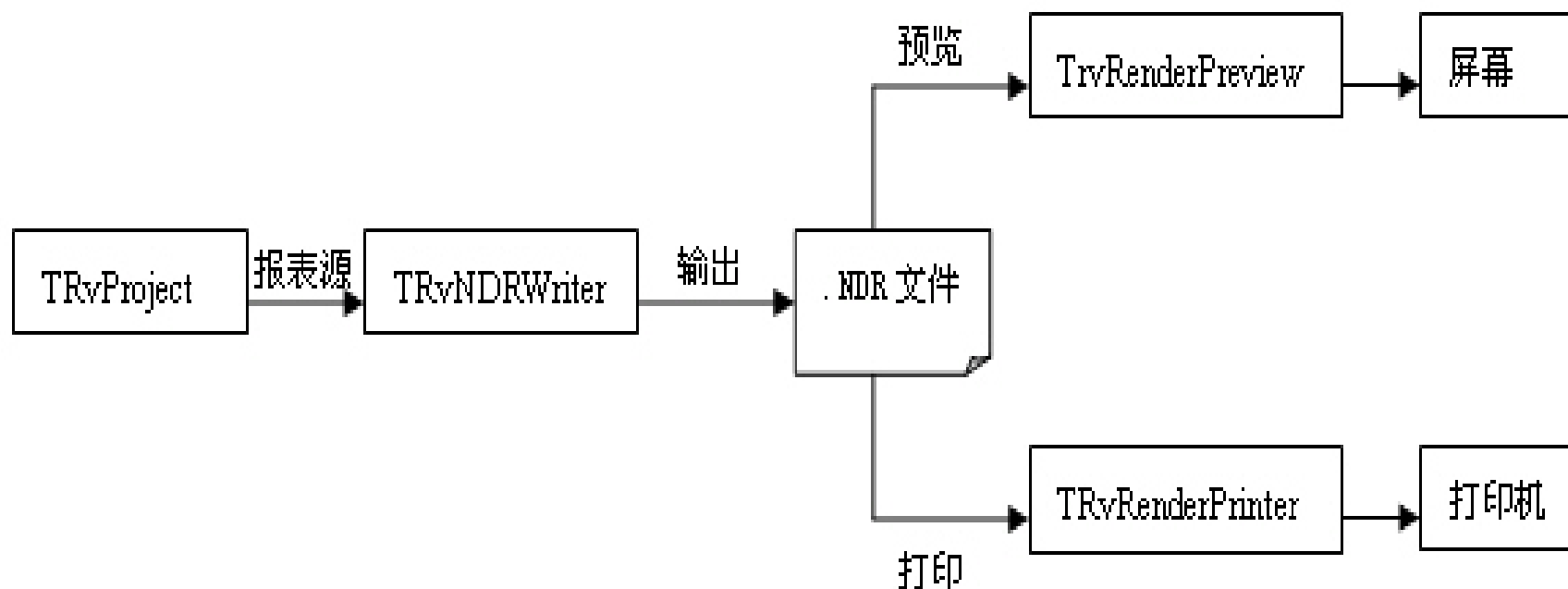


图 9-5 TRvNDRWriter 组件的报表处理过程

## 9.2 Rave Delphi组件

(1) TRvNDRWriter组件的属性:

- AccuracyMethod属性
- StreamMode属性
- FileName属性

(2) TRvNDRWriter组件的方法

- Start方法
- Finish方法
- Execute方法

### 9.2.5 表现类组件

#### 1. TrvRenderPreview组件

TRvRenderPreview组件接收TRvNDRWriter组件生成的文件并发送到屏幕打印。

## 9.2 Rave Delphi组件

### (1) TrvRenderPreview属性

- ScrollBox属性
- FileName和StreamMode属性
- GridHoriz和GridVert属性
- RulerType属性
- MarginMethod和MarginPercent属性
- ShadowDepth属性
- Monochrome属性
- NextPage和PrevPage属性
- ZoomInc属性

### (2) TRvRenderPreview的事件

- OnPageChange
- OnZoomChange

## 9.2 Rave Delphi组件

### 2. TrvRenderPrinter组件

TRvRenderPrinter组件接收由TRvNDRWriter组件产生的文件并发送到当前打印机。

### 3. TRvRenderPDF组件

TRvRenderPDF允许从报表中生成PDF文档。设置EmbedFonts属性为True可以让字体嵌入到PDF文档中。PDF中图像的质量可以用一个使用百分比的ImageQuality属性来设置。设置UseCompression为True可以生成压缩的PDF文档。

### 4. TRvRenderHTML组件

TRvRenderHTML转换一个NDR流或文件为HTML网页。支持文本、图像、线条和方框对象。输出结果为HTML4.0格式且匹配打印输出。

## 9.2 Rave Delphi组件

### 5. TRvRenderRTF组件

TRvRenderRTF转换一个NDR流或文件为RTF格式。RTF文档的输出结果匹配报表的打印格式。文档中的元素被独立的包含在“frames”中支持精确的页面定位。

### 6. TrvRenderText组件

TRvRenderText转换一个NDR流或文件为文本格式，在这个报表中只支持文本项，其他的对象如图形或线条将被忽略。

### 9.2.6 一个简单的报表应用程序

在一个简单的报表应用程序中通常使用Rave数据连接组件提供数据，使用RvProject组件执行报表，另外还要创建一个报表项目文件（.rav）。程序的基本结构如图9-6所示

:

## 9.2 Rave Delphi组件

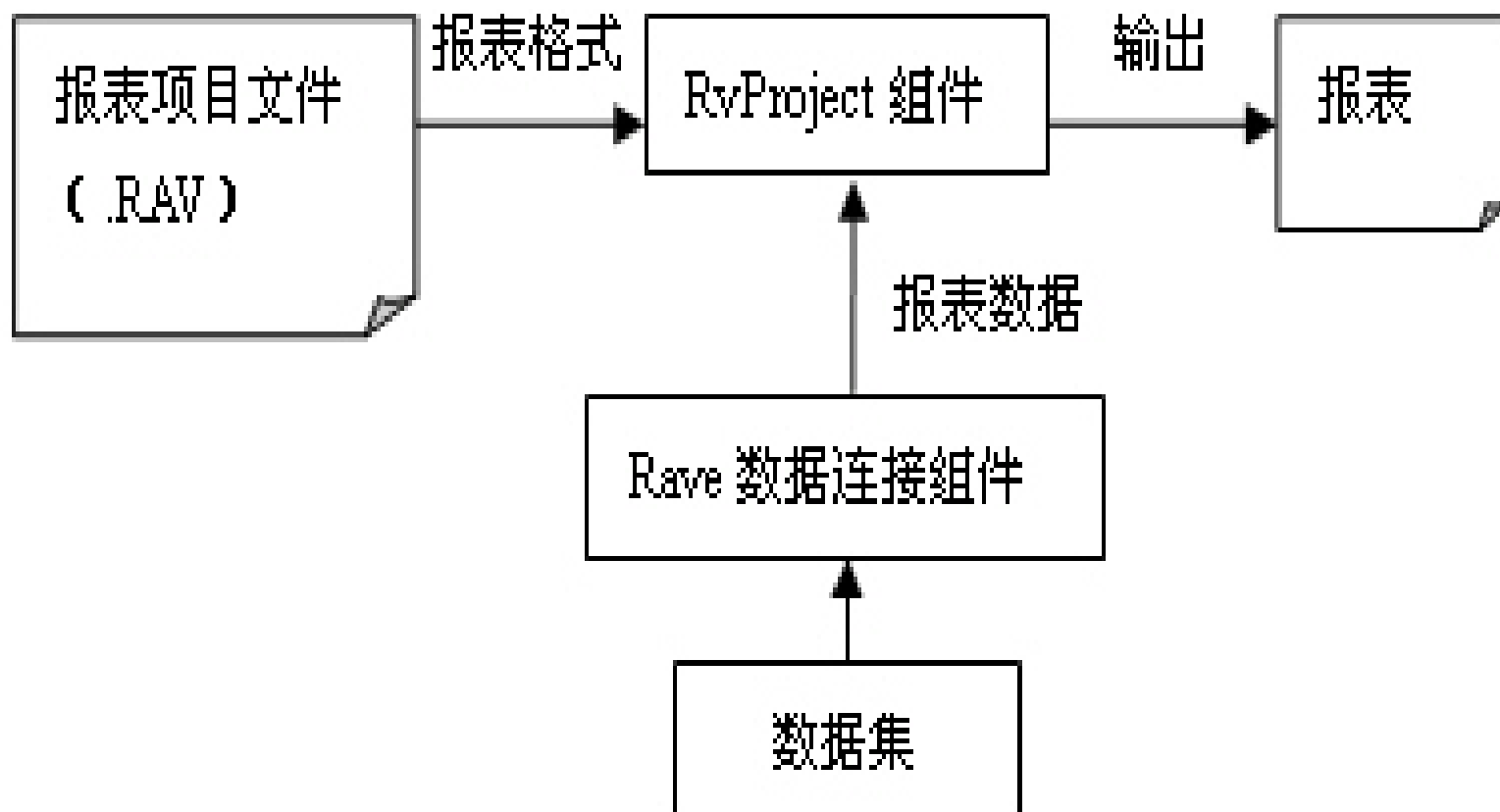


图9-6 数据库报表应用程序基本结构

## 9.2 Rave Delphi组件

### 【例9-1】

编写一个简单的报表应用程序，打印数据库dbdemos.mdb的表Employee中的信息。

### 【实现步骤】

- (1) 新建应用程序
- (2) 定制窗体



图9-7 设计窗体界面

## 9.2 Rave Delphi组件

表9-3 各组件属性设置

组件名	属 性	属 性 值	说 明
ADOConnection1	ConnectionString		使用ConnectionString设置对话框设置该属性，连接到数据库dbdemos.mdb。
ADOTable1	Connection	ADOConnection1	指定使用的数据连接组件。
	TableName	Employee	指定打开的数据表名
	Active	True	打开数据集。
RvProject1	projectfile	C:\Samples\9-1\myrave.rav	指定报表文件路径
RvDataConnection1	dataset	ADOTable1	与ADOTable1连接
Button1	Caption	执行报表	

### (3) 设计Rave报表

## 9.2 Rave Delphi组件

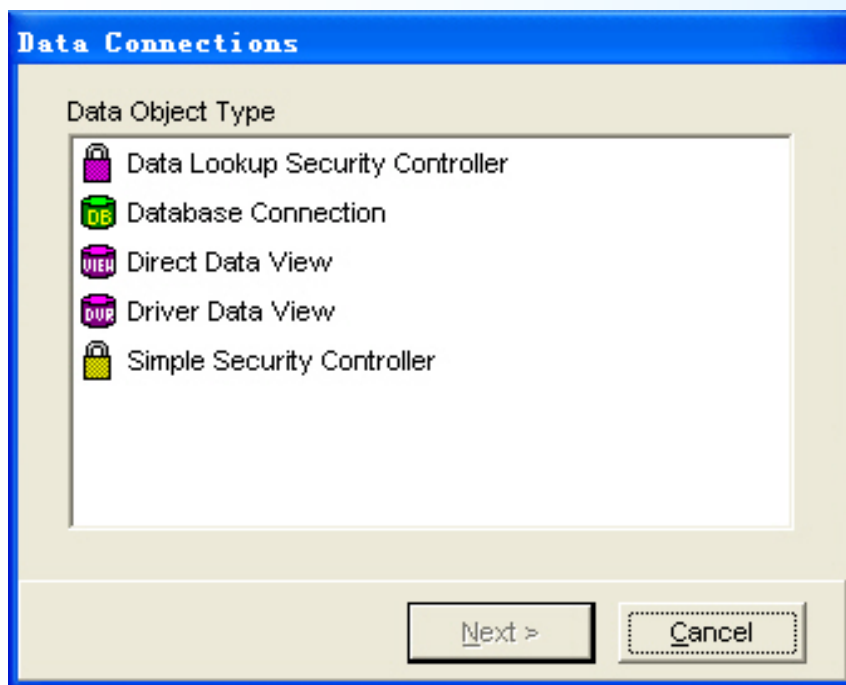


图9-7 数据对象类型选择对话框

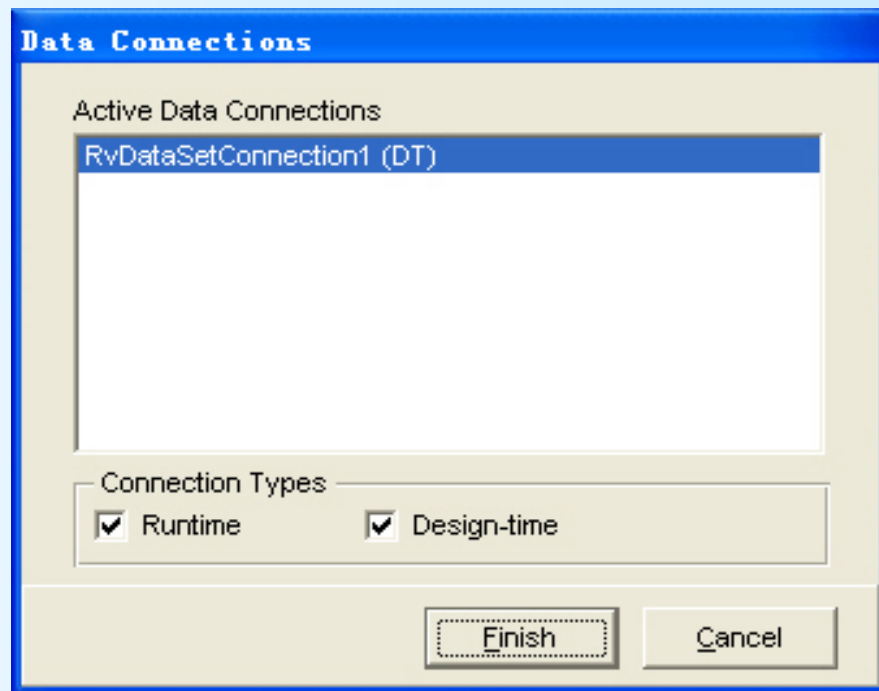


图9-8 数据连接选择对话框

## 9.2 Rave Delphi组件

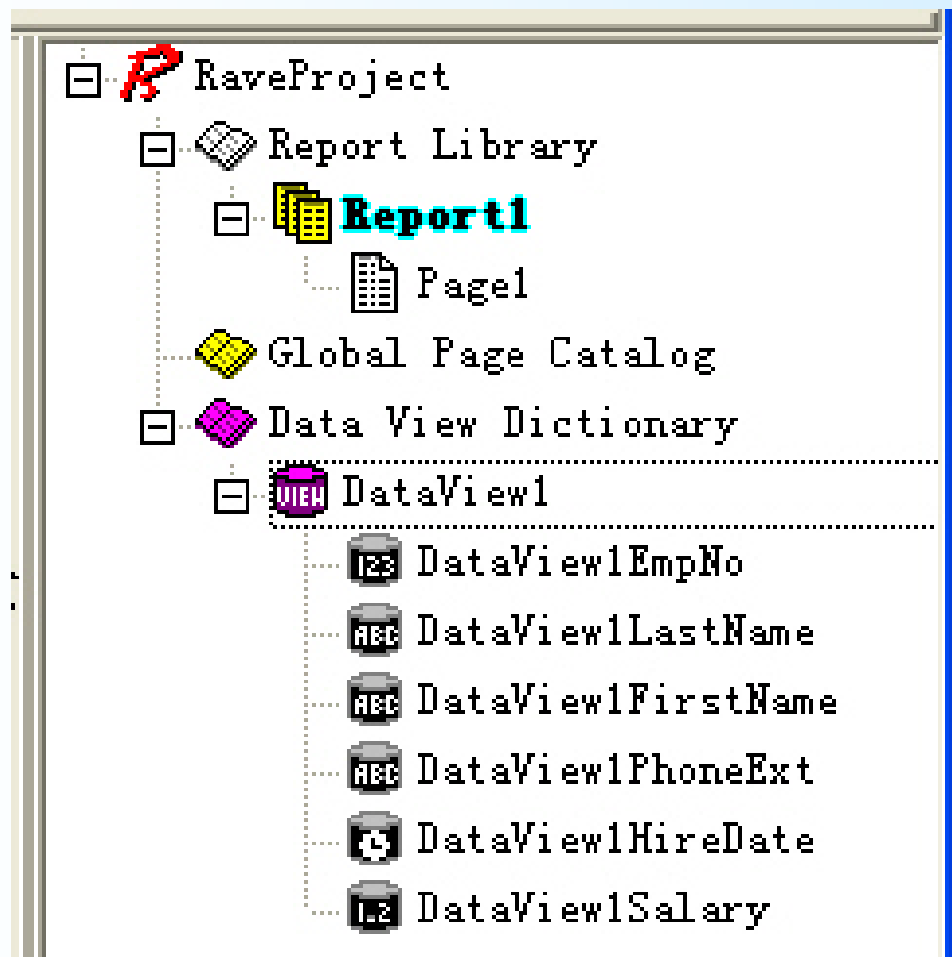


图9-9 数据视图

## 9.2 Rave Delphi组件

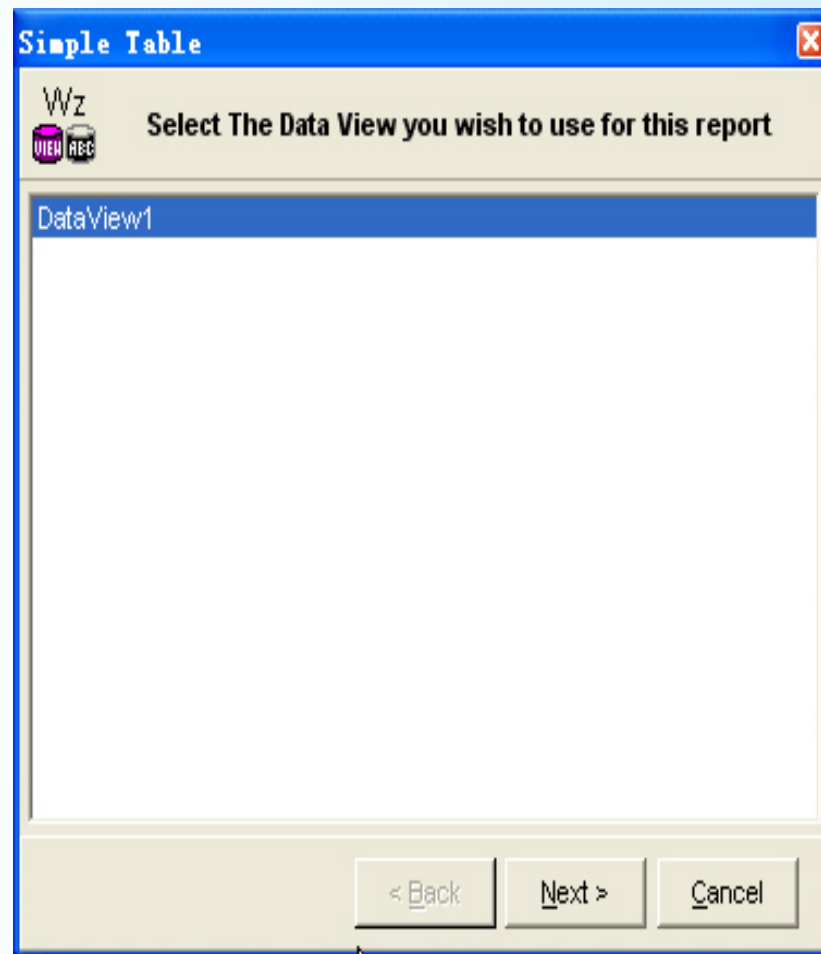


图9-10 数据视图选择对话框

## 9.2 Rave Delphi组件

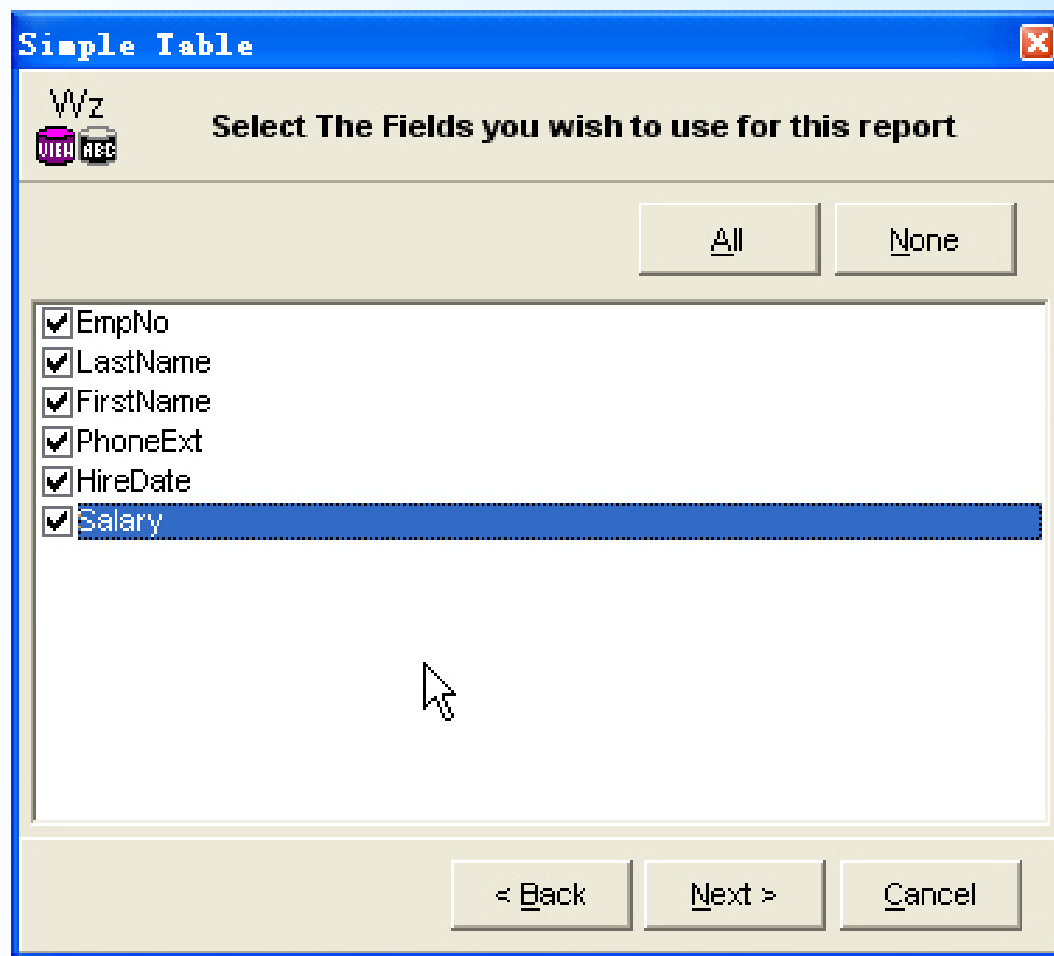


图9-11 选择报表中要使用的字段

## 9.2 Rave Delphi组件

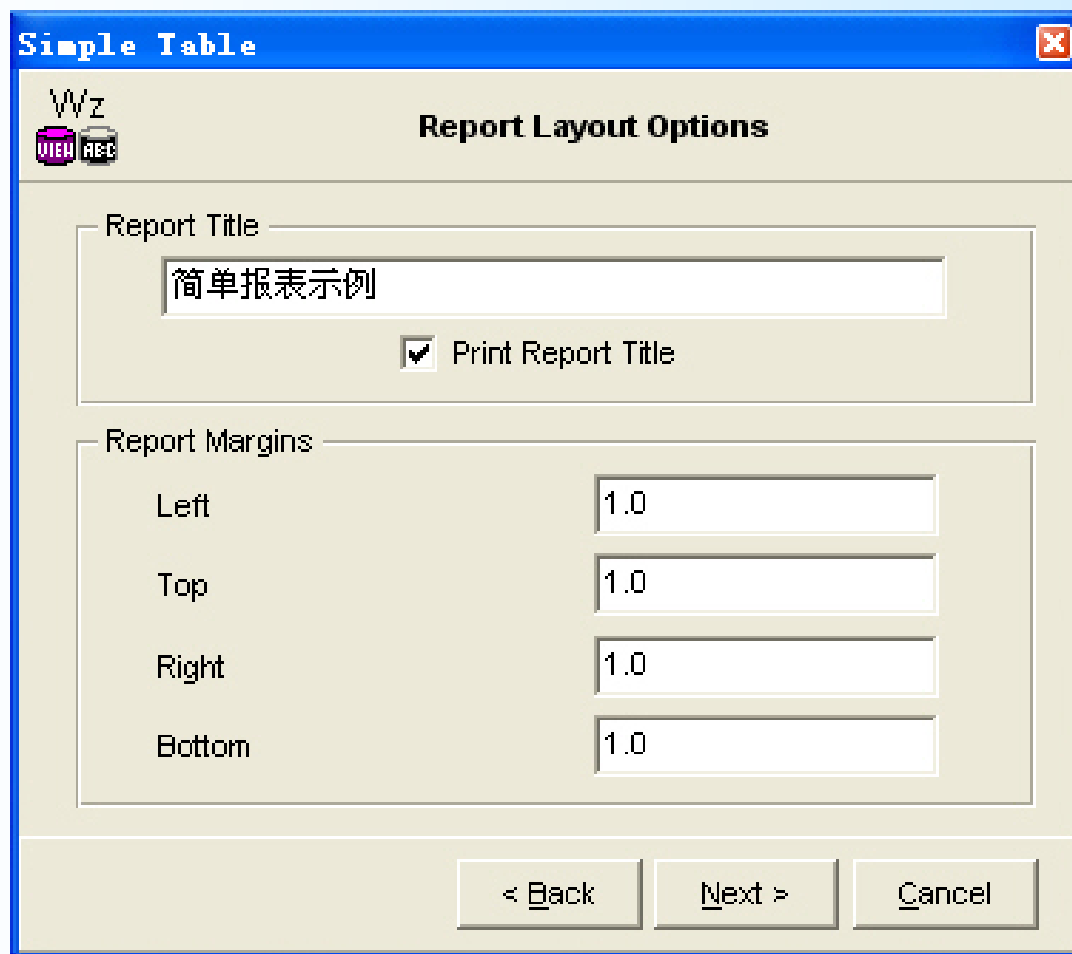


图9-12 报表布局对话框

## 9.2 Rave Delphi组件

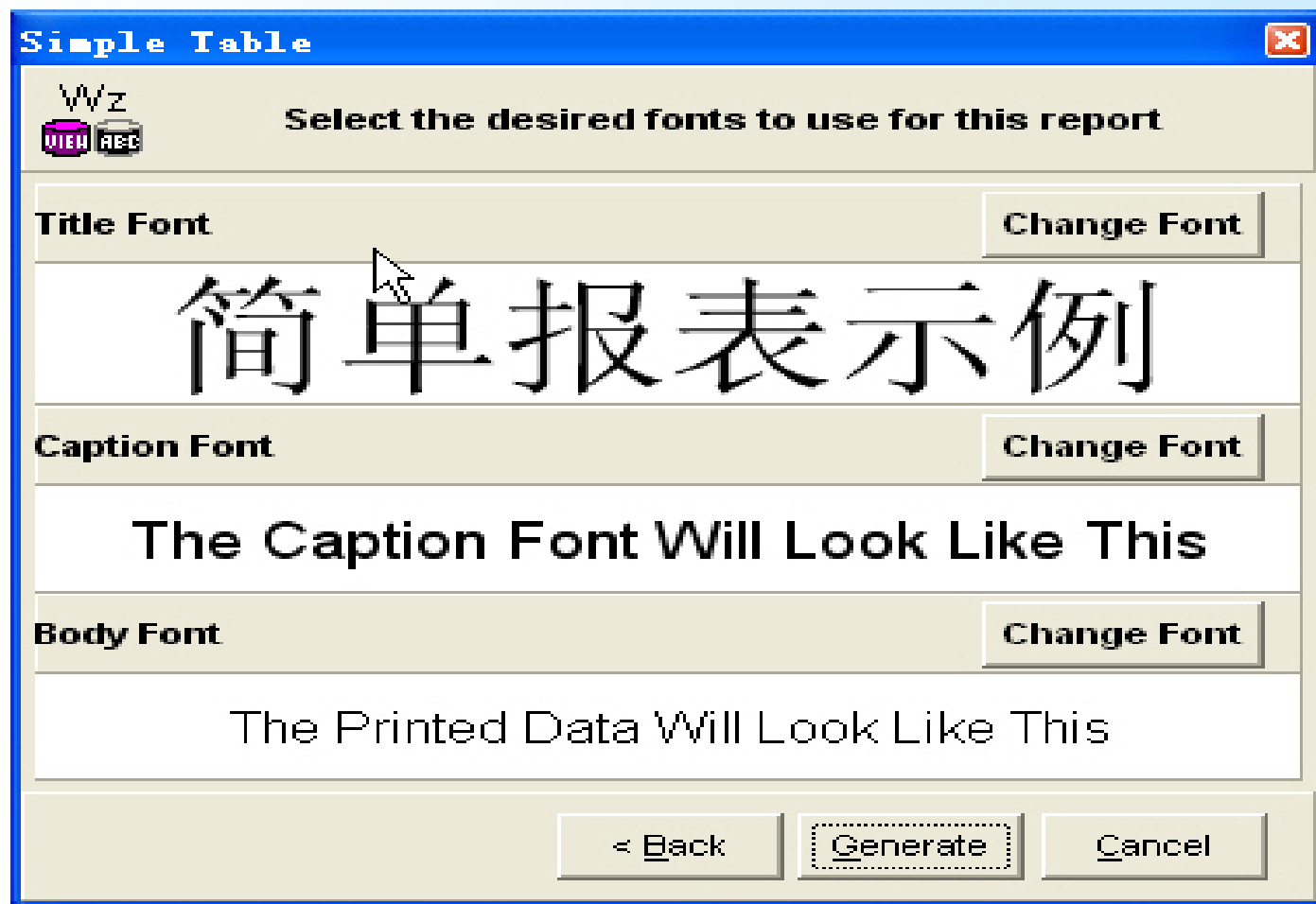


图9-13 设置报表报头、报表栏目

## 9.2 Rave Delphi组件

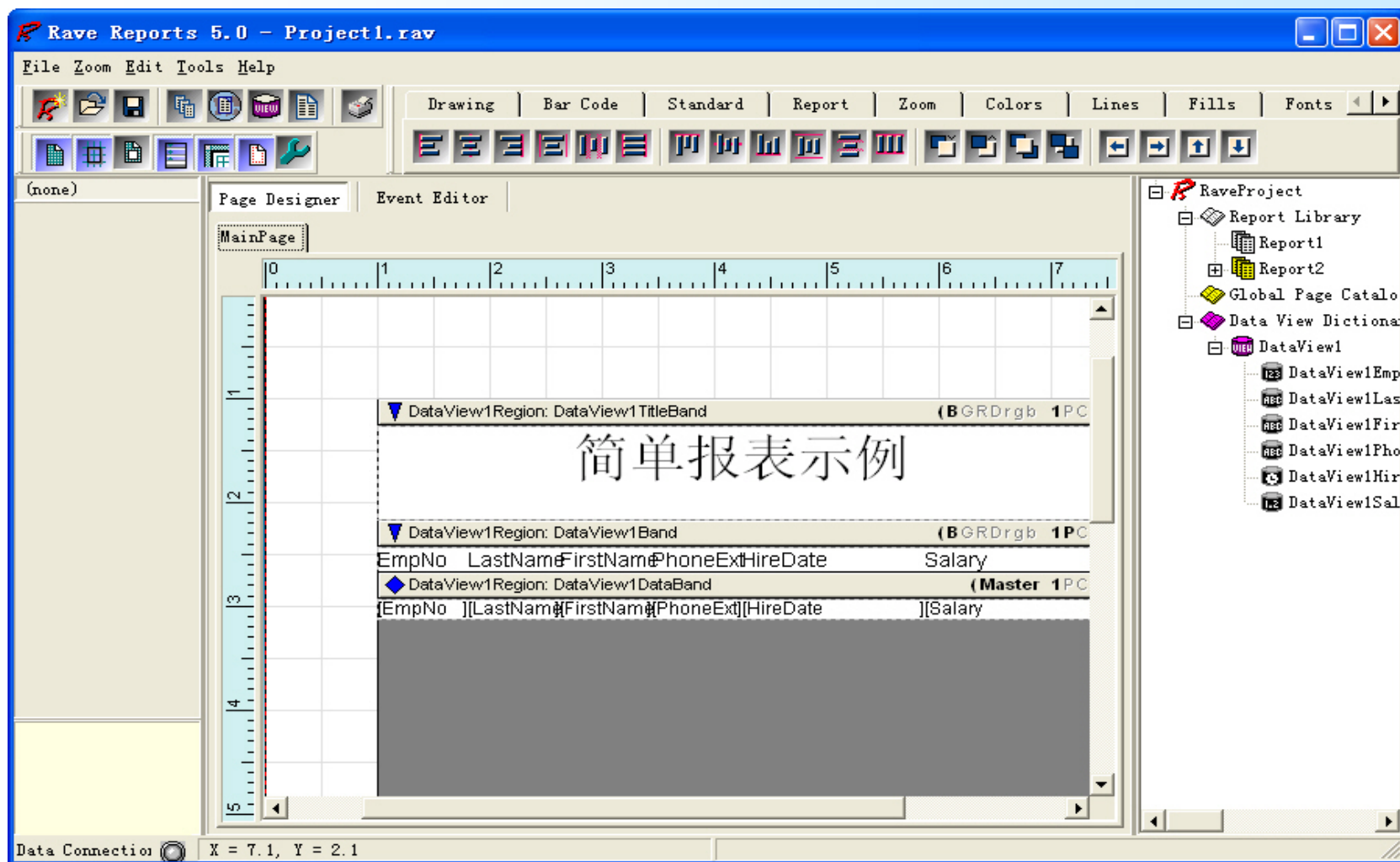


图9-14 向导创建的报表

## 9.2 Rave Delphi组件

(4) 代码编写

(5) 运行程序

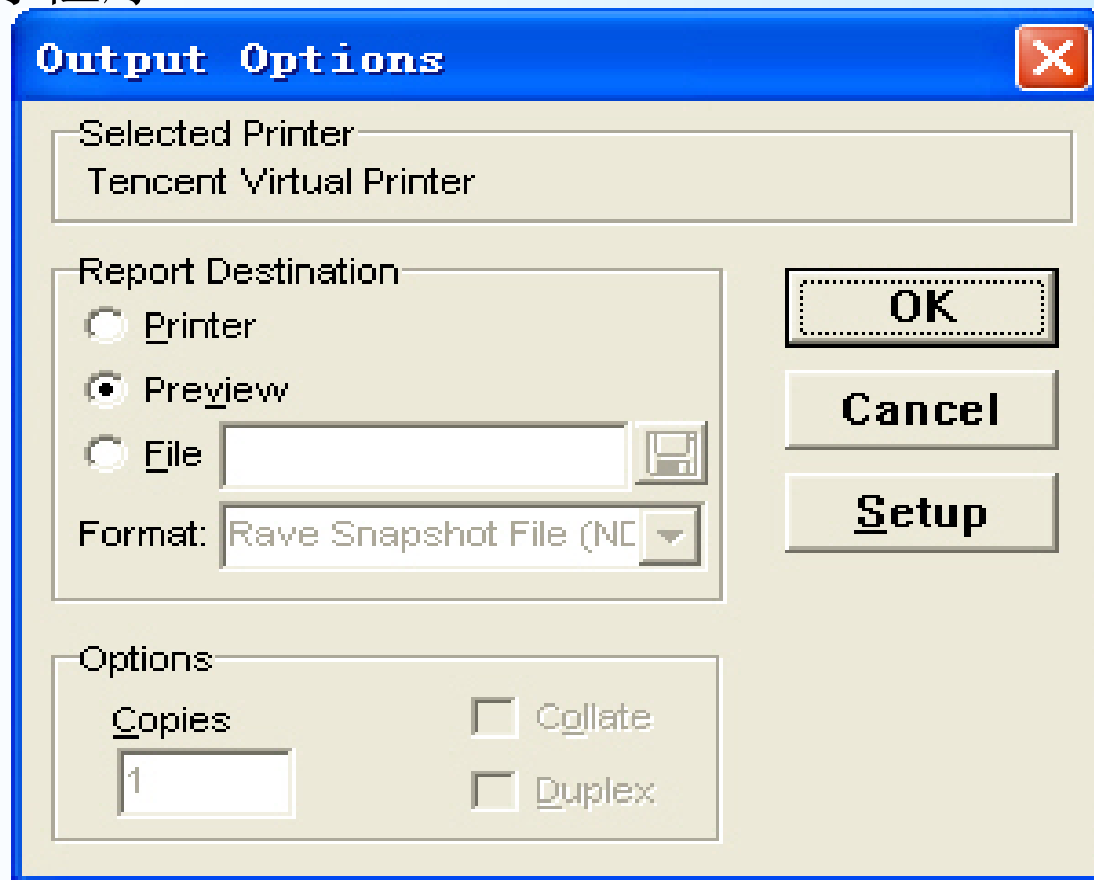
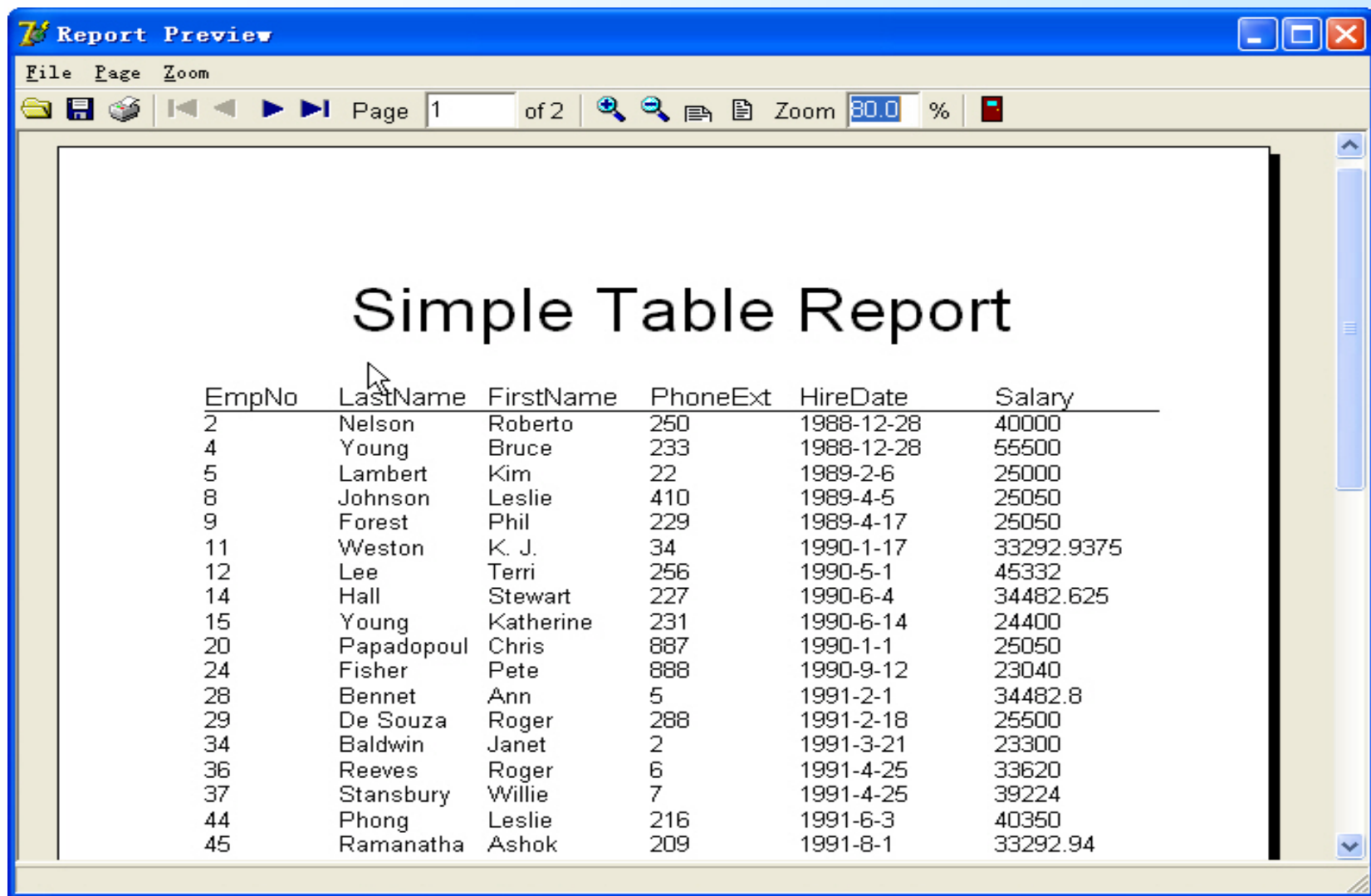


图9-15 打印输入选项对话框

## 9.2 Rave Delphi组件



The screenshot shows a 'Report Preview' window with a menu bar (File, Page, Zoom) and a toolbar. The report title is 'Simple Table Report'. The table contains 20 rows of employee data with columns: EmpNo, LastName, FirstName, PhoneExt, HireDate, and Salary. A mouse cursor is pointing at the 'LastName' header.

EmpNo	LastName	FirstName	PhoneExt	HireDate	Salary
2	Nelson	Roberto	250	1988-12-28	40000
4	Young	Bruce	233	1988-12-28	55500
5	Lambert	Kim	22	1989-2-6	25000
8	Johnson	Leslie	410	1989-4-5	25050
9	Forest	Phil	229	1989-4-17	25050
11	Weston	K. J.	34	1990-1-17	33292.9375
12	Lee	Terri	256	1990-5-1	45332
14	Hall	Stewart	227	1990-6-4	34482.625
15	Young	Katherine	231	1990-6-14	24400
20	Papadopoul	Chris	887	1990-1-1	25050
24	Fisher	Pete	888	1990-9-12	23040
28	Bennet	Ann	5	1991-2-1	34482.8
29	De Souza	Roger	288	1991-2-18	25500
34	Baldwin	Janet	2	1991-3-21	23300
36	Reeves	Roger	6	1991-4-25	33620
37	Stansbury	Willie	7	1991-4-25	39224
44	Phong	Leslie	216	1991-6-3	40350
45	Ramanatha	Ashok	209	1991-8-1	33292.94

图9-16 报表预览

## 9.3 Rave报表设计器

### 9.3.1 Rave报表组件

#### 1. Drawing组件面板

Drawing组件面板提供了一些用来设计线段、矩形、圆形等图形的组件，使用它我们可以绘制非灵活的各种图形，通过属性面板可以调节相应组件的属性，也可以使用一些组件面板中的组件来进行调节，比如调节它们的颜色可以使用Color组件面板，调节线宽、线型可以使用Lines组件面板。

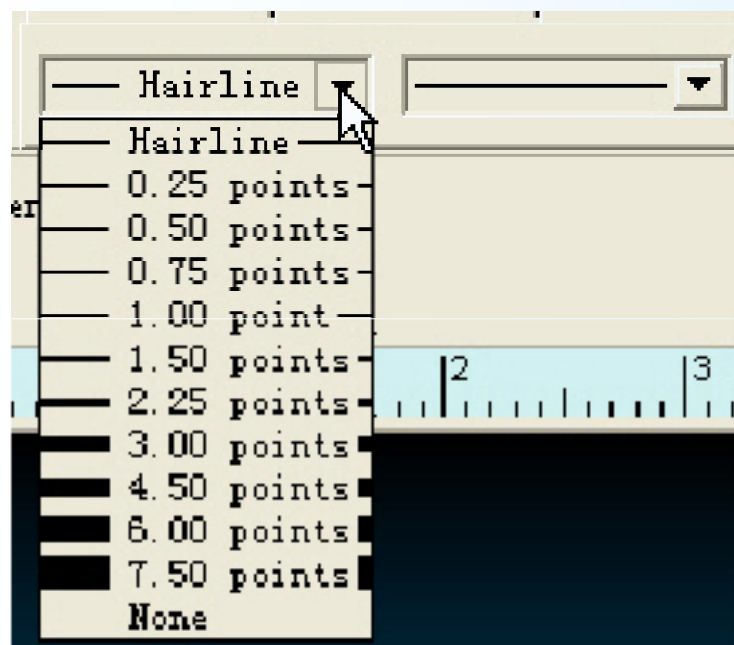


图 9-17 Drawing 组件面板

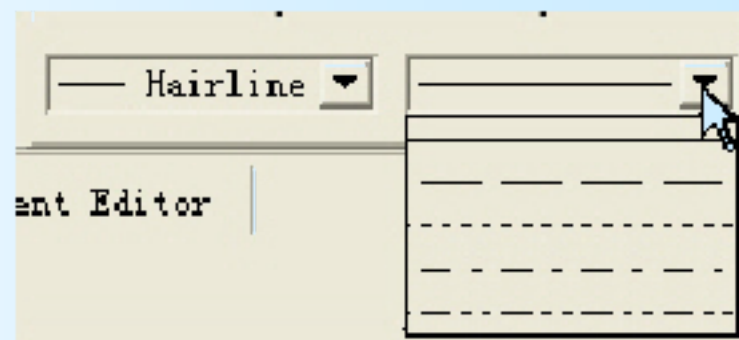


图 9-18 Colors 组件面板

## 9.3 Rave报表设计器



a) 选择线宽



b) 选择线型

图 9-19 Lines 组件面板

## 9.3 Rave报表设计器

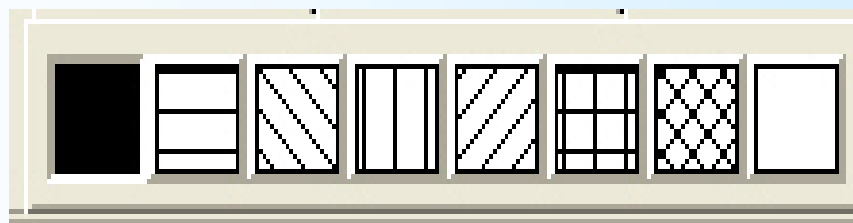


图9-20 Fill组件面板

### 2. Bar code组件

Bar code组件是可以进行一些条形码设计的组件。

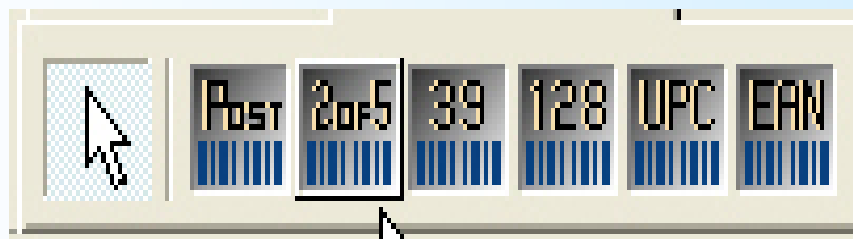


图9-21 Bar code组件面板

## 9.3 Rave报表设计器

### 3. Standard组件面板

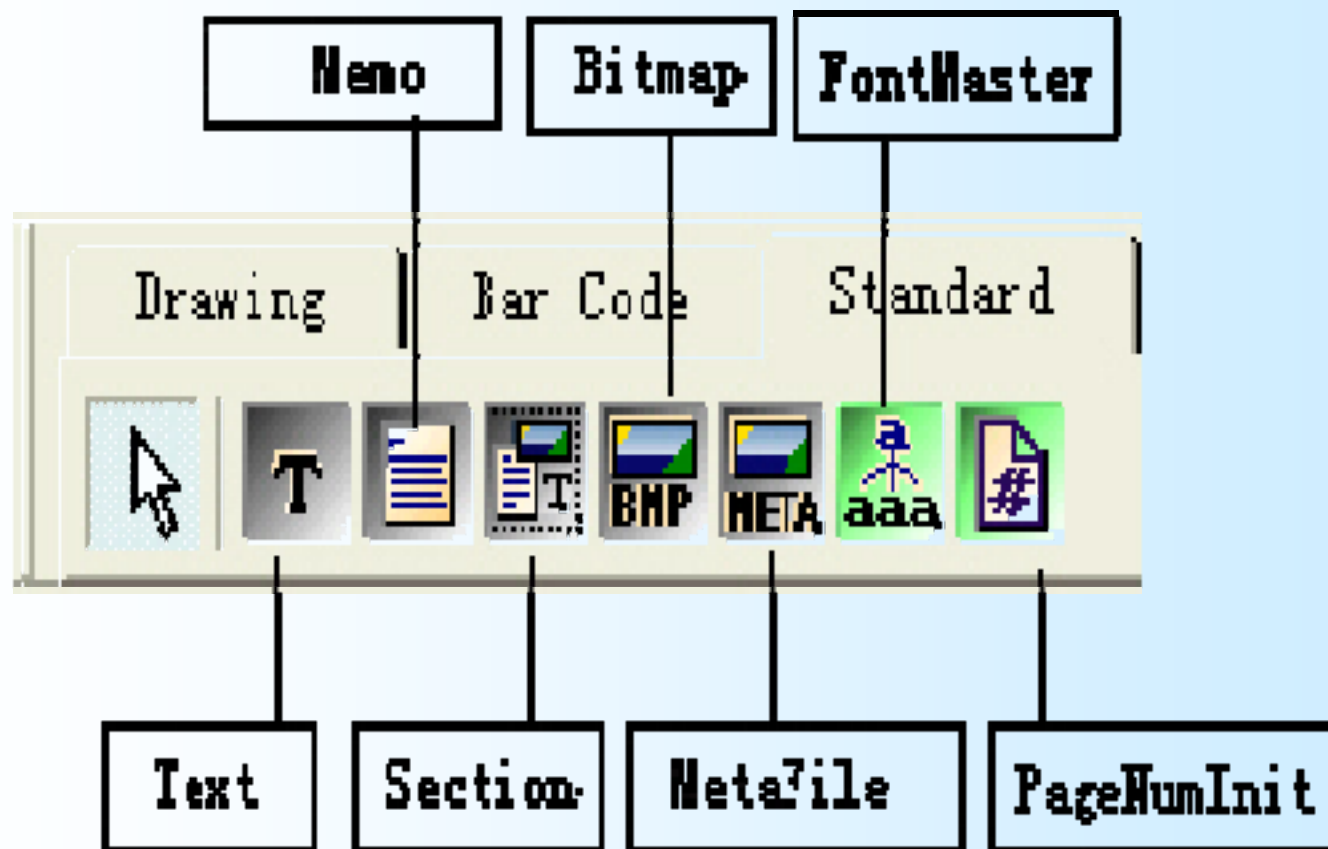


图9-22 Standard组件面板

## 9.3 Rave报表设计器

### (1) Text组件

专门用于在报表中输出相应的本文内容的一个组件。

### (2) Memo组件

与Delphi中的Memo组件非常的相似，可以用于输出多行文本。

### (3) Section组件

这个组件用于包含其他组件的，可以用它来进行分组。

### (4) Bitmap组件

用于在报表中显示相应的位图文件。

### (5) MetaFile组件

MetaFile组件与Bitmap组件几乎一致，它除了可以处理位图文件以外，还可以处理emf和wmf文件。

### (6) FontMaster组件

FontMaster组件是一个字体组件，在报表中使用它规定相应的字体组。

## 9.3 Rave报表设计器

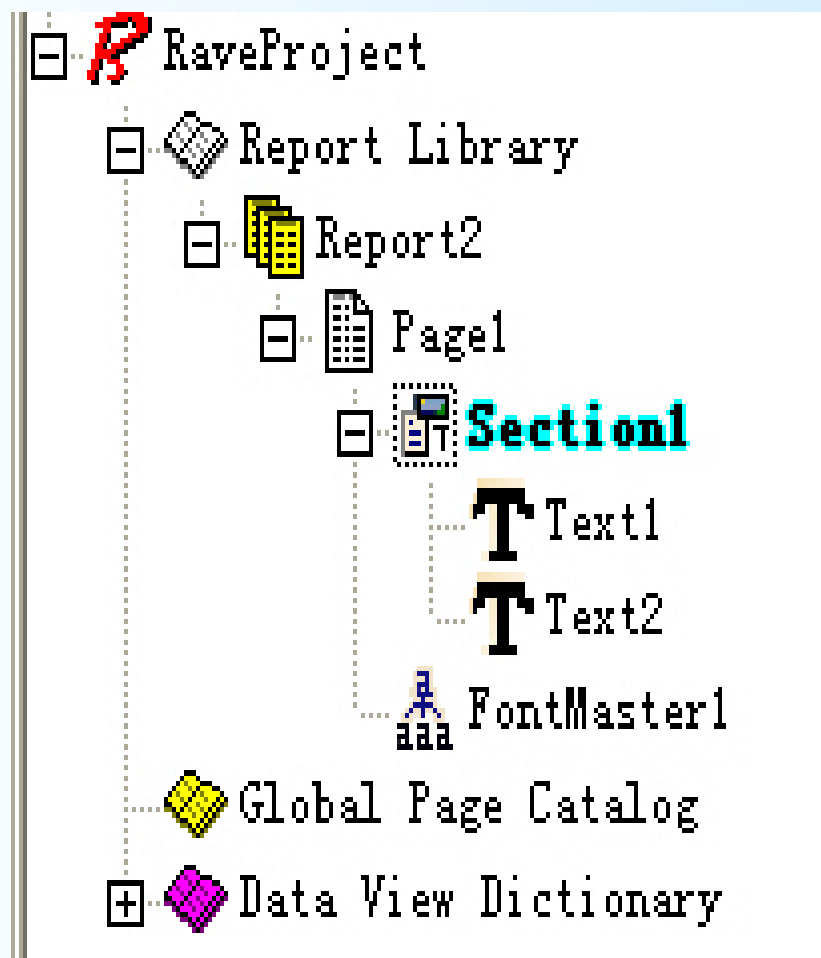
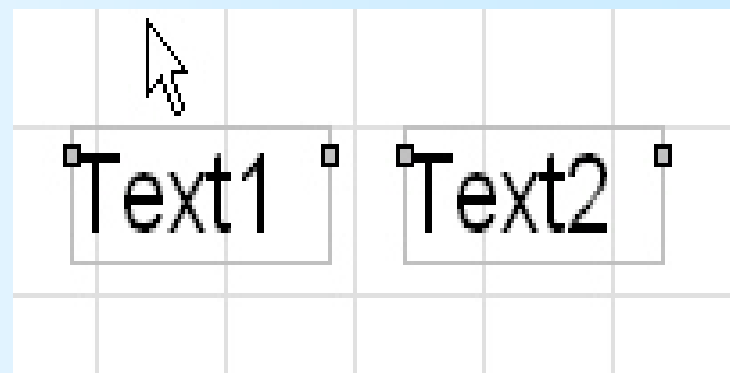


图9-23 用对象树选择FontMaster组件

## 9.3 Rave报表设计器



a) 设置FontMirror属性前



b) 设置FontMirror属性后

图9-24 通过FontMirror属性建立字体组

## 9.3 Rave报表设计器

### (7) PageNumInit组件

PageNumInit组件就是将相应的页码初始的号码进行处理，如图所示：

PageNumInit1: PageNumInit	
DevLocked	False
InitDataField	
InitDataView	
InitValue	1
Locked	False
Name	PageNumInit1
Tag	0

图9-25 用InitValue属性初始化页号

## 9.3 Rave报表设计器

### 4. Report组件面板

Report组件面板主要用于建立数据报表。

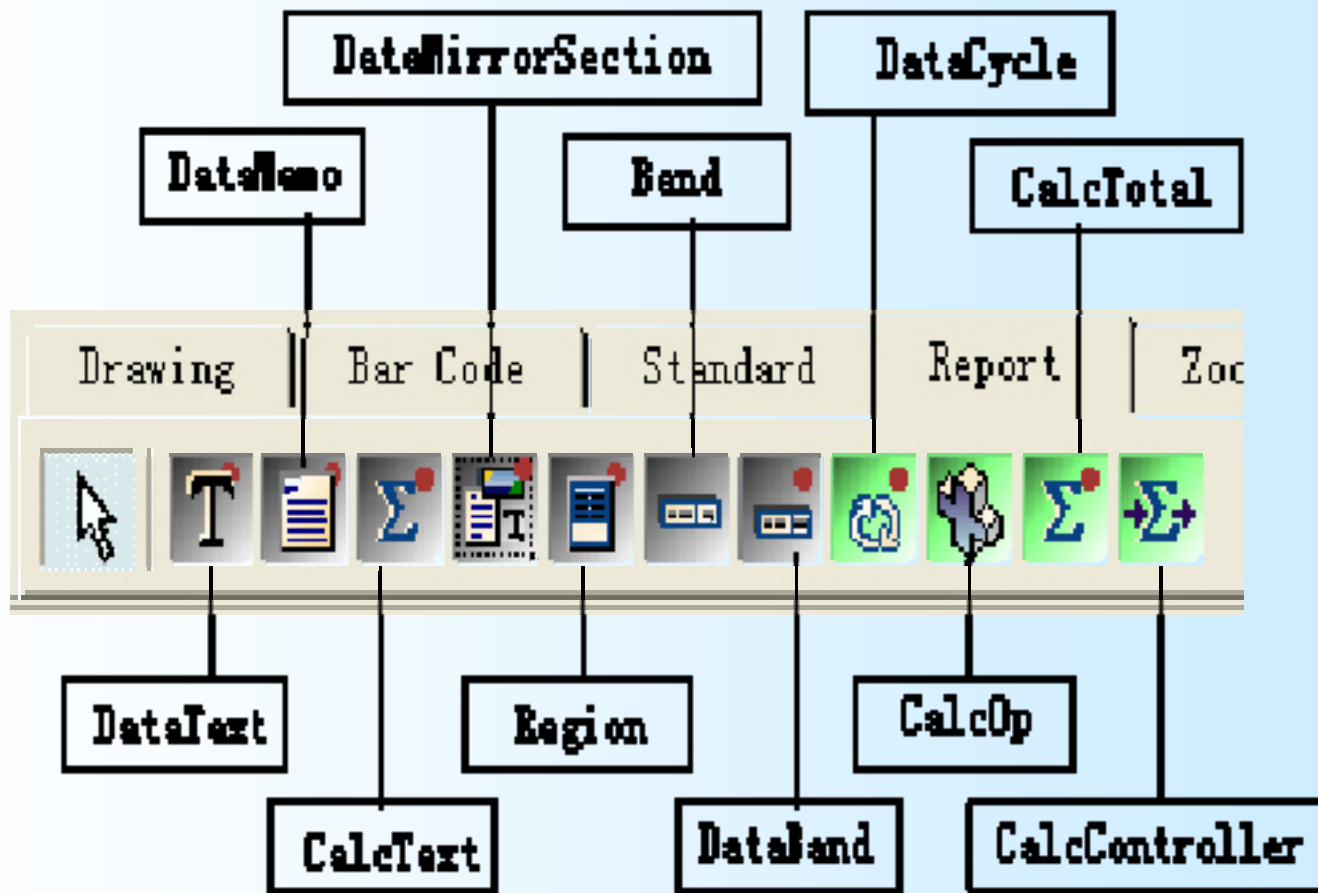


图9-25 Report组件面板

## 9.3 Rave报表设计器

### (1) DataText组件

DataText组件可以将数据库中的字段信息显示出来，当然与Text一样它只能显示单行信息。

### (2) DataMemo组件

DataMemo组件在Memo的基础上支持了数据库的字段显示，用于显示需要多行才能显示的字段内容。

### (3) CalcText组件

CalcText组件可以计算显示一个报表字段的最大值、最小值、总和、计数值、平均值。

### (4) DataMirrorSection组件

与Section组件的概念非常相似，它也是一个用于将一系列组件进行分组的组件。

### (5) Region组件

Region组件用于规定相应的报表打印的区域，报表中所有需要打印的内容都必须放在Region中。

## 9.3 Rave报表设计器

### (6) Band组件

Band组件是一个包容组件，可以包容一些报表组件，比如：Text、DataText组件等。

▼ DataView1Region: DataView1TitleBand (BGRDrgb 1PC)					
简单报表					
▼ DataView1Region: DataView1Band (BGRDrgb 1PC)					
EmpNo	LastName	FirstName	PhoneExt	HireDate	Salary
◆ DataView1Region: DataView1DataBand (Master 1PC)					
[EmpNo]	[LastName]	[FirstName]	[PhoneExt]	[HireDate]	[Salary]

图9-26 报表的带区

## 9.3 Rave报表设计器

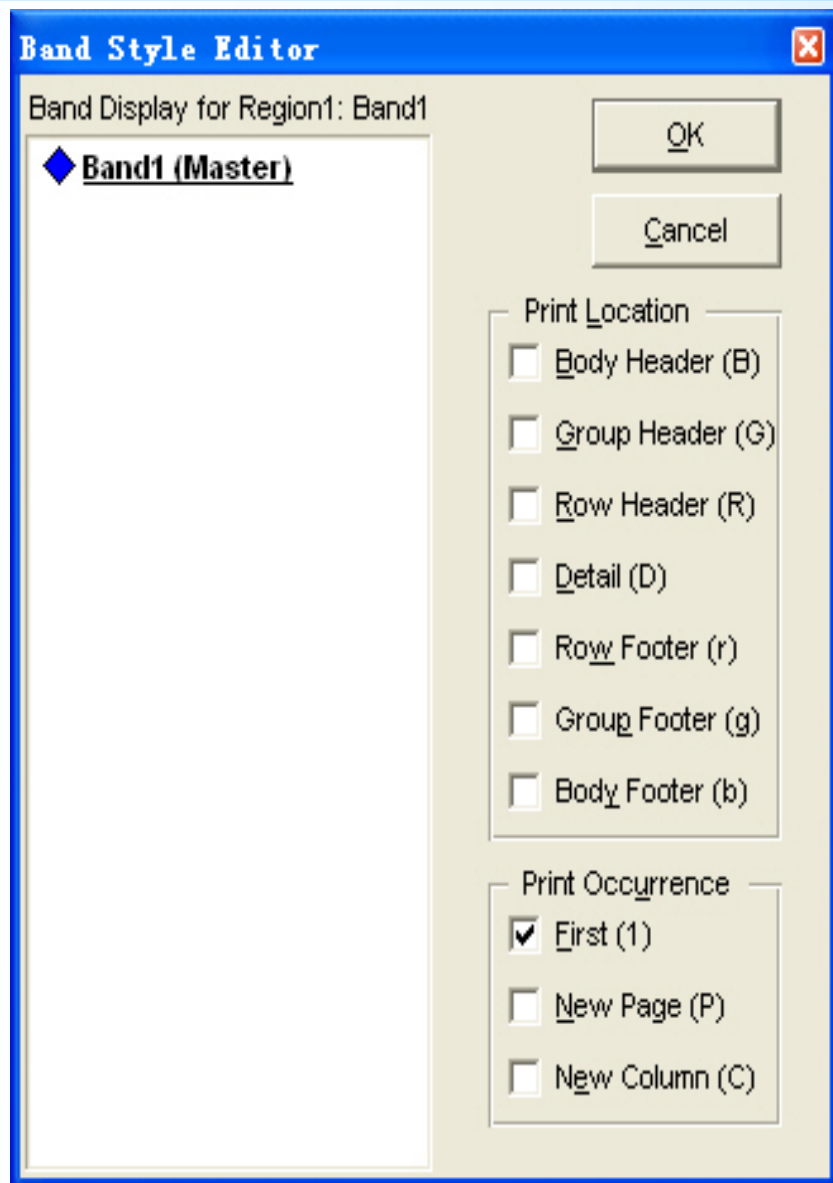


图9-27 BandStyle编辑器

## 9.3 Rave报表设计器

### (7) DataBand组件

**DataBand**组件与**Band**组件都是供打印的一个报表范围，而它们的不同在于**DataBand**可以直接作用于数据库，而且也可以在其中摆放相应的数据库报表组件，这样通过它就可以让相应的报表具有数据库数据的打印能力。

### (8) DataCycle组件

如果单从使用功能上来讲，**DataCycle**和**DataBand**组件之间是比较相似的。**DataBand**提供一个循环的条件来让数据库中的记录循环使用，但**DataBand**也有缺陷，如我们在平时的设计过程中，经常使用的简历打印（卡片式打印），**DataBand**就难于胜任。

## 9.3 Rave报表设计器

姓名↵	黄卫东↵	性别↵	男↵	年龄↵	29↵
职务↵	↵	职称↵	工程师↵	学历↵	专科↵
参加工作时间↵		1990 年↵	从事项目经理年限↵		4↵
工作技能↵		能够进行净化工程的商务谈判、图纸设计、预算编制、工程现场施工管理、洁净工程的调试和测试。↵			
工作意向↵		希望在一个充满团队精神的公司供职。↵			

图9-28 简历格式

## 9.3 Rave报表设计器

### 5. 报表缩放组件（Zoom）



图9-29 Zoom面板

### 6. Fonts面板

Fonts面板包含用于设置文本效果的快捷工具。



图9-30 Fonts面板

## 9.3 Rave报表设计器

### 7. Alignment面板

Alignment面板包含用于调整报表组件布局的快捷工具。

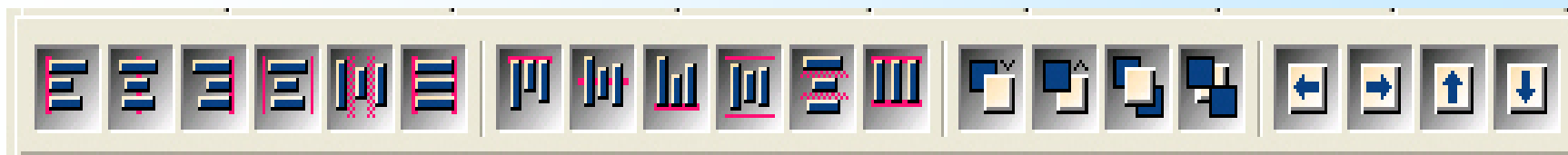


图9-31 Alignment面板

# 9.3 Rave报表设计器

## 9.3.2 自定义报表设计

### 【例9-2】

在【例9-1】的基础上建立一个自定义报表。

### 【实现步骤】

- (1) 建立新报表项目
- (2) 建立数据源
- (3) 新建打印区域 (Region)
- (4) 建立报表标题带区
- (5) 建立报表头带区
- (6) 建立报表数据带区
- (7) 放置数据显示对象

## 9.3 Rave报表设计器

◆ Region1: TitleBand (Master 1PC)					
员工信息报表					
▼ Region1: HeaderBand (BGRDr gb 1PC)					
EmpNo	LastName	FistName	PhoneExt	HireDate	Salary
◆ Region1: DataBand1 (Master 1PC)					
[EmpN]	[LastName]	[FirstName]	[PhoneExt ]	[HireDate ]	[Salary]

图9-32 设计好的报表页面

## 9.3 Rave报表设计器

### (8) 预览报表

员工信息报表

EmpNo	LastName	FistName	PhoneExt	HireDate	Salary
2	Nelson	Roberto	250	1988-12-28	40000
4	Young	Bruce	233	1988-12-28	55500
5	Lambert	Kim	22	1989-2-6	25000
8	Johnson	Leslie	410	1989-4-5	25050
9	Forest	Phil	229	1989-4-17	25050
11	Weston	K. J.	34	1990-1-17	33292.93
12	Lee	Terri	256	1990-5-1	45332
14	Hall	Stewart	227	1990-6-4	34482.62
15	Young	Katherine	231	1990-6-14	24400

图9-33 预览报表

## 9.4 Rave的高级特性

### 9.4.1 打印日期和页码

Rave中提供了许多Report Variable(报表变量)供程序员在设计报表时使用，包括：日期、时间、页码、打印机信息等。以【例9-2】中我们设计的自定义报表为例，下面我们为该报表加入打印日期和页码信息。

#### 1. 打印日期

在报表的底左侧部添加一个DataText组件用于显示报表打印日期，设置FontMirror属性为FontMaster1,然后点击属面板上DataField属性右侧的按钮，打开Data Text Editor对话框。从Report Variables列表中选择DataLong（长日期格式），再点击“Insert Report Var”按钮添加报表变量到DataText编辑框中，点击“OK”按钮退出对话框。注意，这里不需要设置DataView属性。

## 9.4 Rave的高级特性

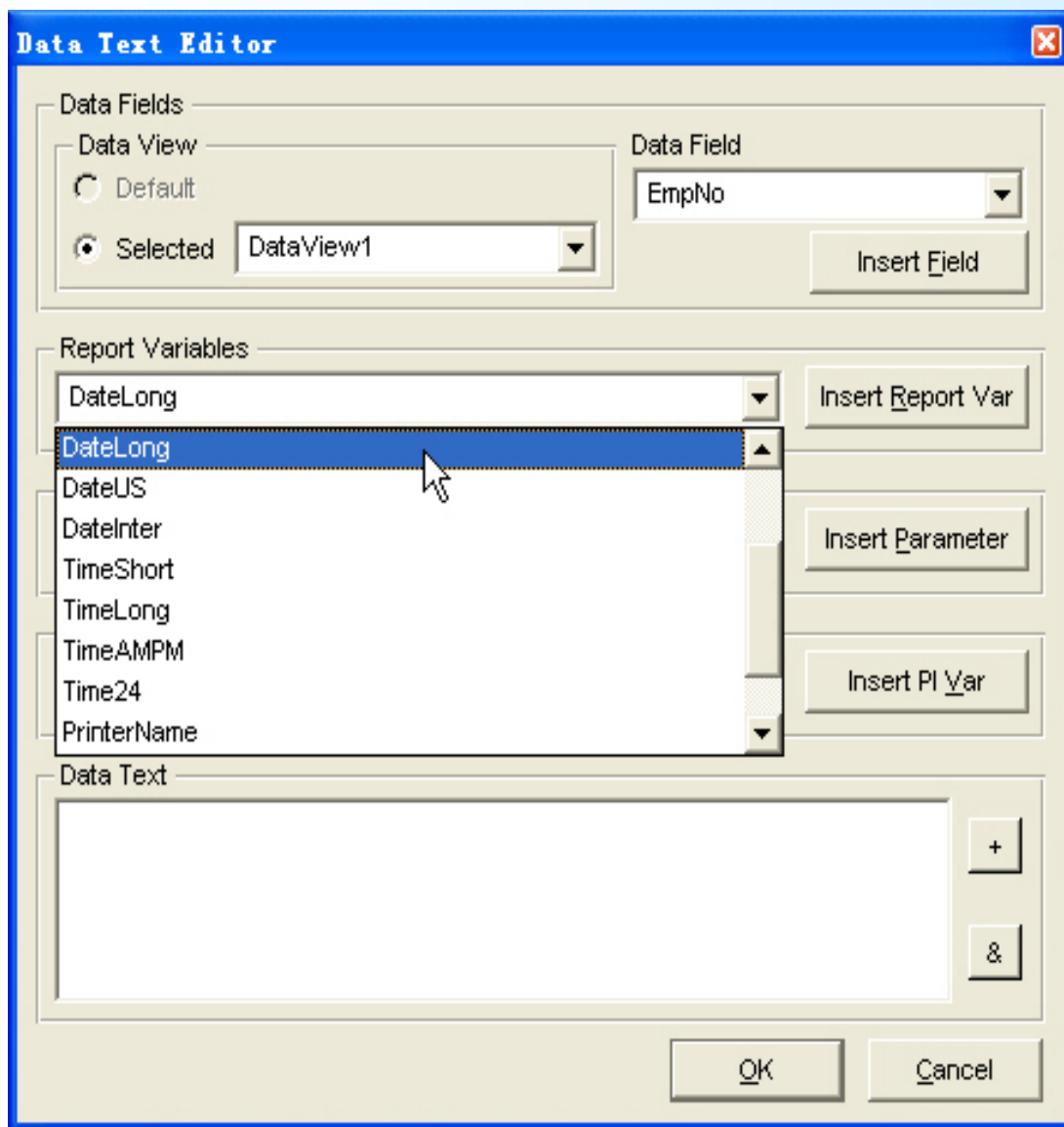


图9-34 选择日期格式

## 9.4 Rave的高级特性

### 2. 打印页码

在报表的底部右侧添加一个**DataText**组件用于显示报表打印日期，设置**FontMirror**属性为**FontMaster1**,然后点击属面板上**DataField**属性右侧的按钮，打开**Data Text Editor**对话框。从**Report Variables**列表中选择**CurrentPage**（当前页码），再点击“**Insert Report Var**”按钮添加报表变量到**DataText**编辑框中，点击“**OK**”按钮退出对话框。

重复以上操作，选择**TotalPages**（总页数）。

加入三个**Text**组件，然后作适当调整，使得打印结果如下图：

## 9.4 Rave的高级特性

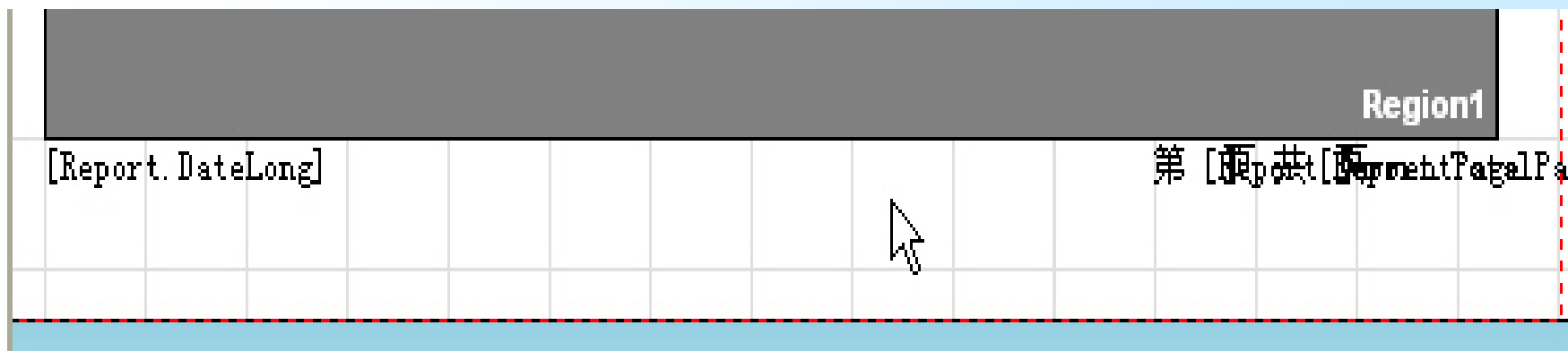


图9-35 设计时的页脚

65	O' Brien	Sue Anne	877	1992-3-23	31275
2005年8月1日				第 1 页, 共 2 页	

图9-36 预览时的页脚

## 9.4 Rave的高级特性

### 9.4.2 参数(Parameters) 的使用

参数可以存在于报表项目，报表，报表页面这三个地方，通过报表项目的对象树来选择报表项目和特定的报表对象，然后通过所选对象的parameters属性进行设定。参数可通过DataText部件来输出。

Region1: HeaderBand (BGDRgb 1PC)					
EmpNo	LastName	FistName	PhoneExt	HireDate	Salary
Region1: DataBand1 (Master 1PC)					
[EmpNo]	[LastName]	[FirstName]	[PhoneExt ]	[HireDate ]	[Salary]
Region1: Band1 (Master 1PC)					
工资总额: [Param.salarySum]					

图9-37 使用参数输出计算值

## 9.4 Rave的高级特性

145	Guckenhe	Mark	221	1994-5-2	32000
-----	----------	------	-----	----------	-------

工资总额: 1386202.292

2005年8月1日

第 2 页, 共 2 页

图9-37 使用参数输出的预览效果

## 9.4 Rave的高级特性

### 9.4.3 全局页面

如果有些报表部分在两个或更多报表中是相同的，就可以把这些相同部分放置到全局页面中以便所有报表共享。比如公司名称、日期和时间、当前页号和总页数等。

首先，加一个全局页面到报表项目中。在这个页面中，加一个**section**组件。**Sections**组件用于组件的逻辑分组。在**section**组件中加入我们想要共享的东西。

在特定报表的特定页，增加另一个**section**组件。设定该组件的**Mirror**属性为前面那个全局页面中的**Section**组件。

# 本章实训指导

1. 掌握**RAVE**报表设计器的使用；
2. 掌握常规报表的设计；
3. 了解**Delphi**中**Rave**组件，以及各组件之间的关系；
4. 掌握**Rave**报表应用程序的编写；
5. 以班级的期末成绩表为例，编写一个报表应用程序打印成绩，要求统计总分和平均分，并按总分排名次。

# 第10章 数据库高级应用技术

本章主要内容：

- 事务的概念
- Delphi中的事务控制
- Delphi中使用存储过程
- 主从结构的数据库应用程序

# 10.1 事务处理

## 10.1.1 事务的概念

当用Delphi创建数据库应用时，Delphi提供了用于所有数据库访问的事务控制。

一个事务（**transaction**）是指在一个数据库中的一张或多张表上执行的一组相关动作，在这一组动作被永久提交前，必须成功完成所有动作，如果该组动作中有一个动作操作失败，那么所有动作就会被取消。

## 10.1.2 事务的特性

事务具有不可分性、一致性、隔离性和持续性。

### 1. 不可分性

事务是一个工作单元，是在应用程序的 **BEGIN TRANSACTION** 和 **END TRANSACTION** 语句之间发生一系列操作。事务只执行一次，且是不可分的，即完成全部工作或者不做任何工作。

# 10.1 事务处理

## 2. 一致性

事务是一个完整的单元，因为它保持数据的一致性，将数据从一种数据一致状态转换到另一种数据一致状态。

## 3. 隔离性

事务是一个隔离单元，允许可并行执行的事务表现得像是在系统中运行的唯一事务。隔离性要求即使同时可能有其他事务正在运行，每个事务也像是操作数据存储区的唯一事务一样。事务应从不查看其他事务的中间阶段。

## 4. 持续性

事务也是一个恢复单元。如果事务成功，则即使在提交后计算机立即崩溃，系统仍将保证更新该事务。专用记录允许系统的重新启动过程完成未完成的操作，以使事务可持续。

# 10.1 事务处理

## 10.1.3 Delphi中事务控制的方式

### 1. Delphi中事务控制的方式

Delphi中事务控制的方式有两种：一种是隐式事务控制，一种是显式事务控制。

在默认情况下，Delphi通过ADO为应用程序提供隐式事务控制。当应用程序处于隐式事务控制时，Delphi为数据集中的每个记录的写操作进行隐式事务控制。它提交每一个独立的写操作，如Post和AppendRecord。

有两种显式控制事务方式：

一种是使用Delphi组件（如TADOConnection）所提供的方法和属性进行事务控制。

另一种是直接发送SQL命令到支持事务的数据库服务器进行事务控制。许多数据库都提供自己的事务处理方式。

# 10.1 事务处理

## 2. 使用TADOConnection控制事务

### (1) 事务处理过程

通常事务处理有以下这样一个过程：

```
BeginTrans; //开始一个新事务处理
```

```
try //捕捉异常
```

```
    . 数据库更新操作（如delete、insert、update等）
```

```
    .
```

```
    .
```

```
CommitTrans; //操作成功，提交事务
```

```
except //异常处理
```

```
RollBackTrans; //操作失败，回退事务
```

```
end;
```

# 10.1 事务处理

## (2) 开始一个事务

开始一个事务使用**BeginTrans**方法。当开始一个事务时，后来的所有读写数据库的操作都发生在那次事务的环境中，直到事务被明确地终止或提交了新事务。

使用例句：

```
Level := ADOConnection1.BeginTrans;
```

**BeginTrans**方法返回该事务的嵌套级别。

## (3) 提交一个事务

为了做永久性修改，事务完成后必须使用**CommitTrans**方法提交。

例如，下列语句将终止在上例中开始的事务：

```
ADOConnection1.CommitTrans
```

## (4) 回退一个事务

为了取消对数据库的修改，必须用**RollbackTrans**方法回退一个事务。

# 10.1 事务处理

例如：下列语句将回退一个事务：

**ADOConnection1.RollbackTrans;**

(5) 判断是否正在处理事务

判断是否正在处理事务，可以通过InTransaction属性。

例：

**IF ADOConnection1.InTransaction THEN**

**ADOConnection1.RollbackTrans;**

(6) 使用IsolationLevel属性

IsolationLevel属性描述TADOConnection组件事务的独立级别，事务的独立级别决定了事务与其它作用于相同表的事务是如何相互作用的。

# 10.1 事务处理

## 10.1.4 事务处理的实例

下面以一个简单的例子说明事务的处理过程。

### 【例10-1】

Delphi对TDBGrid组件使用的是默认的隐含事务控制，在表格修改一条记录后，当记录指针移动以后，数据就被写入到数据库中了。若表格很大、修改项目又很多时，如果中途想要放弃所作的修改，很难使表格恢复原样。这时就可以使用事务处理来解决这样的问题。编程实现对数据库dbdemos.mdb的表employee的事务操作。

### 【实现步骤】

1. 首先建立应用程序。
2. 定制窗体

# 10.1 事务处理



图10-1 程序设计界面

# 10.1 事务处理

表10-2 各组件属性设置

组件名	属 性	属 性 值	说 明
ADOConnection1	ConnectionString		使用ConnectionString设置对话框设置该属性，连接到数据库dbdemos.mdb。
ADOTable1	Connection	ADOConnection1	指定使用的数据连接组件。
	TableName	employee	指定打开的数据表名
	Active	True	打开数据集。
DataSource1	DataSet	ADOTable1	指定使用的数据集。
DBGrid1	DataSource	DataSource1	为表格指定数据源。
	ReadOnly	True	表格设置为只读
btnBegin	Caption	开始事务	
	Enable	True	允许使用按钮
btnCommit	Caption	提交事务	
	Enable	False	禁止使用按钮
btnRollBack	Caption	回退事务	
	Enable	False	禁止使用按钮
btnClose	Caption	关闭	
	Enable	True	允许使用按钮

# 10.1 事务处理

## 3. 编写代码

## 4. 程序执行分析

启动程序后，点击【开始事务】按钮，在表格中修改几条记录，然后点击【提交事务】按钮，关闭程序后重新进入，会发现修改内容已写入数据库中。

点击【开始事务】按钮，在表格中修改几条记录，然后点击【回退事务】按钮，关闭程序后重新进入，再次查看表格内容，会发现数据库没有被修改。

# 10.2 使用存储过程

## 10.2.1 存储过程概述

存储过程（**stored procedure**）是一组预先编译好的SQL代码。存储过程作为一个独立的数据库对象，可以作为一个单元被用户的应用程序调用，它在服务器上执行一系列重复性的与数据库相关的任务，并将结果传给客户应用程序。由于存储过程是已经编译好的代码，所以执行的时候不必再次进行编译，从而提高了程序的运行效率。

## 10.2.2 直接执行SQL命令来使用和管理存储过程

### 1. 使用和管理存储过程的应用程序

#### 【例10-2】

建立一个使用和管理存储过程的应用程序。使用MS SQL Server的示例数据库PUBS。

# 10.2 使用存储过程

## 【实现步骤】

- (1) 首先建立应用程序。
- (2) 定制窗体

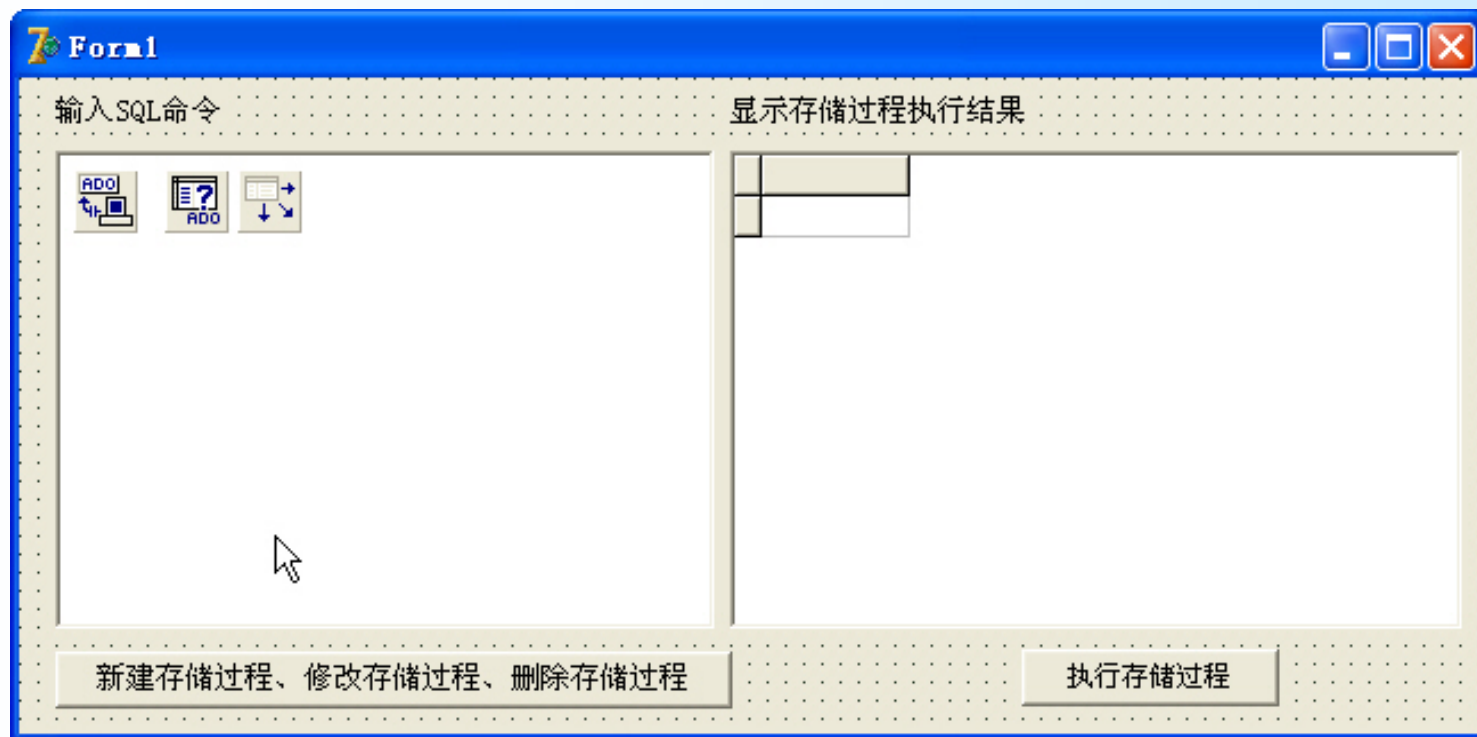


图10-2 程序设计界面

# 10.2 使用存储过程

表10-3 各组件属性设置

组件名	属 性	属 性 值	说 明
ADOConnection1	ConnectionString		使用ConnectionString设置对话框设置该属性，建立到数据库dbdemos.mdb的连接。
ADOQuery1	Connection	ADOConnection1	指定使用的数据连接组件。
DataSource1	DataSet	ADOQuery1	指定使用的数据集。
DBGrid1	DataSource	DataSource1	为表格指定数据源。
	ReadOnly	True	设置为只读
Button1	Caption	新建存储过程、 修改存储过程、 删除存储过程	
Button2	Caption	执行存储过程	
Memo1	Lines		为空
Label1	Caption	输入SQL命令	
Label2	Caption	显示存储过程执行结果	

## 10.2 使用存储过程

---

表10-3 各组件属性设置

# 10.2 使用存储过程

## 2. 使用和管理存储过程的SQL命令

### (1) 创建存储过程

创建存储过程的简单语法如下：

```
CREATE PROC[EDURE] procedure_name [;number]
  [{@parameter data_type} [VARYING] [=default]
  [OUTPUT] ][,...n]
  [WITH
    {RECOMPILE | ENCRYPTION |
    RECOMPILE,ENCRYPTION} ]
  [FOR REPLICATION]
  AS
  sql_statement [ ...n ]
```

## 10.2 使用存储过程

### (2) 执行存储过程

直接执行存储过程可以使用**EXECUTE**命令来执行。在Memo1的编辑框中输入：

**EXEC oakland\_authors**

或

**EXECUTE oakland\_authors**

点击按钮“执行存储过程”以执行存储过程oakland\_authors，结果将显示在DBGrid1表格中。

### (3) 修改存储过程

存储过程可以根据用户的要求或者基表定义的改变而改变。使用**ALTER PROCEDURE**语句可以更改先前通过执行**CREATE PROCEDURE**语句创建的过程，但不会更改权限，也不影响相关的存储过程或触发器。

## 10.2 使用存储过程

其语法形式如下：

```
ALTERPROC[EDURE]procedure_name[;number]
  [{@parameterdata_type}
  [VARYING][=default][OUTPUT]][,...n]
[ WITH {RECOMPILE | ENCRYPTION |
  ECOMPILE,ENCRYPTION} ]
[FOR REPLICATION]
AS
  sql_statement [ ...n ]
```

下面对存储过程oakland\_authors进行修改，使其能够显示出所有居住在加利福尼亚的作者，而不考虑其它地区居住的作者。

## 10.2 使用存储过程

修改存储过程的代码如下：

```
alter procedure oakland_authors  
with encryption  
as  
select au_fname, au_lname, address, city, zip  
from authors  
where state = 'ca'  
order by au_lname, au_fname
```

### （4）删除存储过程

删除存储过程可以使用**DROP**命令，**DROP**命令可以将一个或者多个存储过程或者存储过程组从当前数据库中删除，其语法形式如下：

```
drop procedure {procedure} [,...n]
```

## 10.2 使用存储过程

如将存储过程oakland\_authors从数据库中删除，则在Memo1的编辑框中输入以下代码：

```
drop procedure oakland_authors
```

### （5）带参数的存储过程

存储过程使用两种类型的参数，一种是输入参数，另一种是输出参数。不使用OUTPUT保留字的参数是输入参数，使用OUTPUT保留字的参数是输出参数。输入参数是单向的，而输出参数是双向的，可以向调用者返回值。

本例的存储过程中使用了输入参数。

```
create procedure author_infor
```

```
    @lastname varchar (40) , @firstname varchar (20)
```

```
as
```

```
select au_lname,au_fname,phone
```

```
from authors
```

```
where au_fname=@firstname and au_lname=@lastname
```

# 10.2 使用存储过程

## 10.2.3 用ADO组件操作存储过程

### 1. TADOStoredProc组件简介

TADOStoredProc组件主要用于操作远程服务器上的数据库中的存储过程。ADOStoredProc是一个数据集组件，ADO数据集组件所具有的公共属性、方法和事件，在ADOStoredProc都具有。

#### (1) TADOStoredProc的关键属性

- Active属性
- Connection属性
- ConnectionString属性
- Filter属性
- Filtered属性
- Parameters属性
- ProcedureName属性

## 10.2 使用存储过程

### (2) 关键方法

- Open属性
- Close属性
- Refresh属性
- NextRecordset属性

### 2. TADOStoredProc组件的使用

#### 【例10-3】

建立一个使用和管理存储过程的应用程序。使用MS SQL Server的示例数据库PUBS。

#### 【实现步骤】

- (1) 首先建立应用程序。
- (2) 定制窗体

## 10.2 使用存储过程

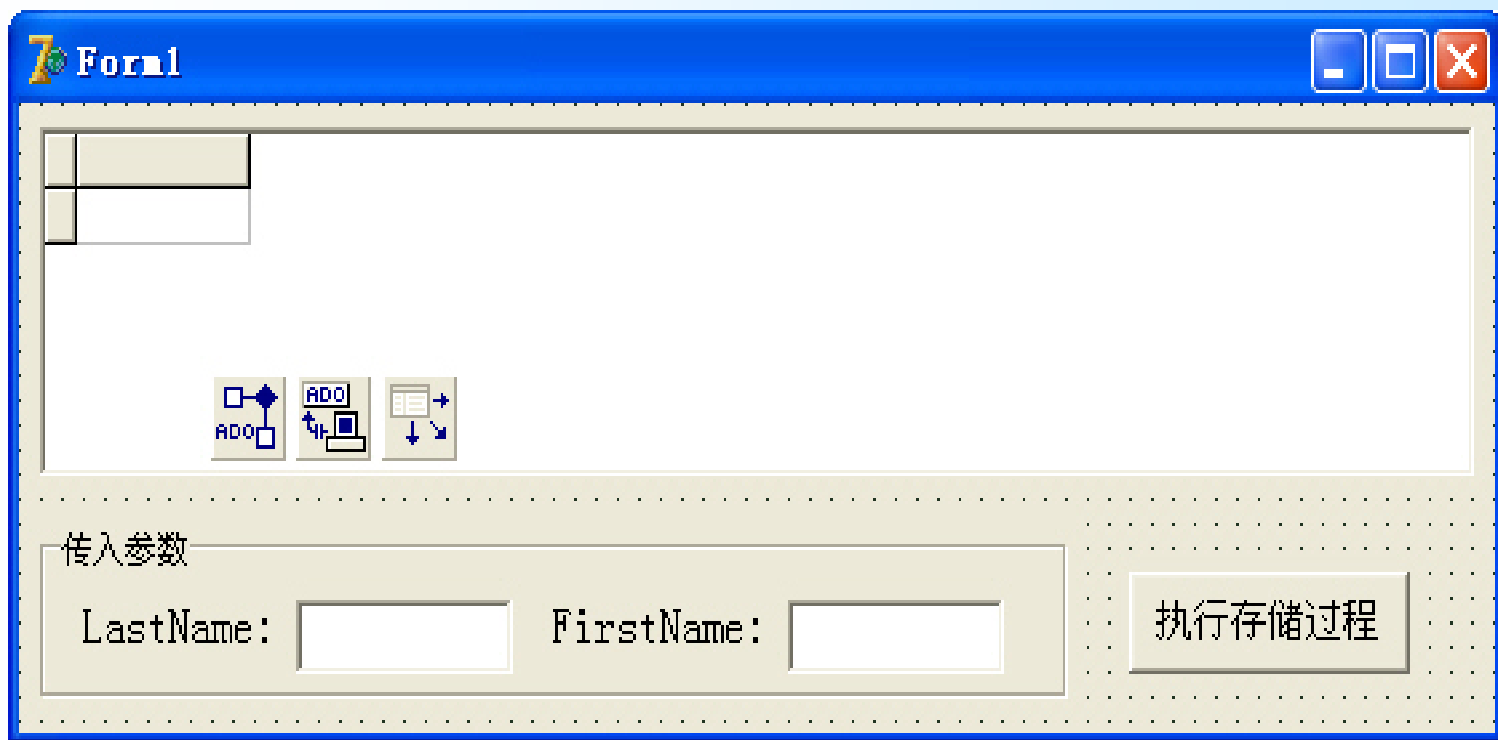


图10-5 程序设计界面

# 10.2 使用存储过程

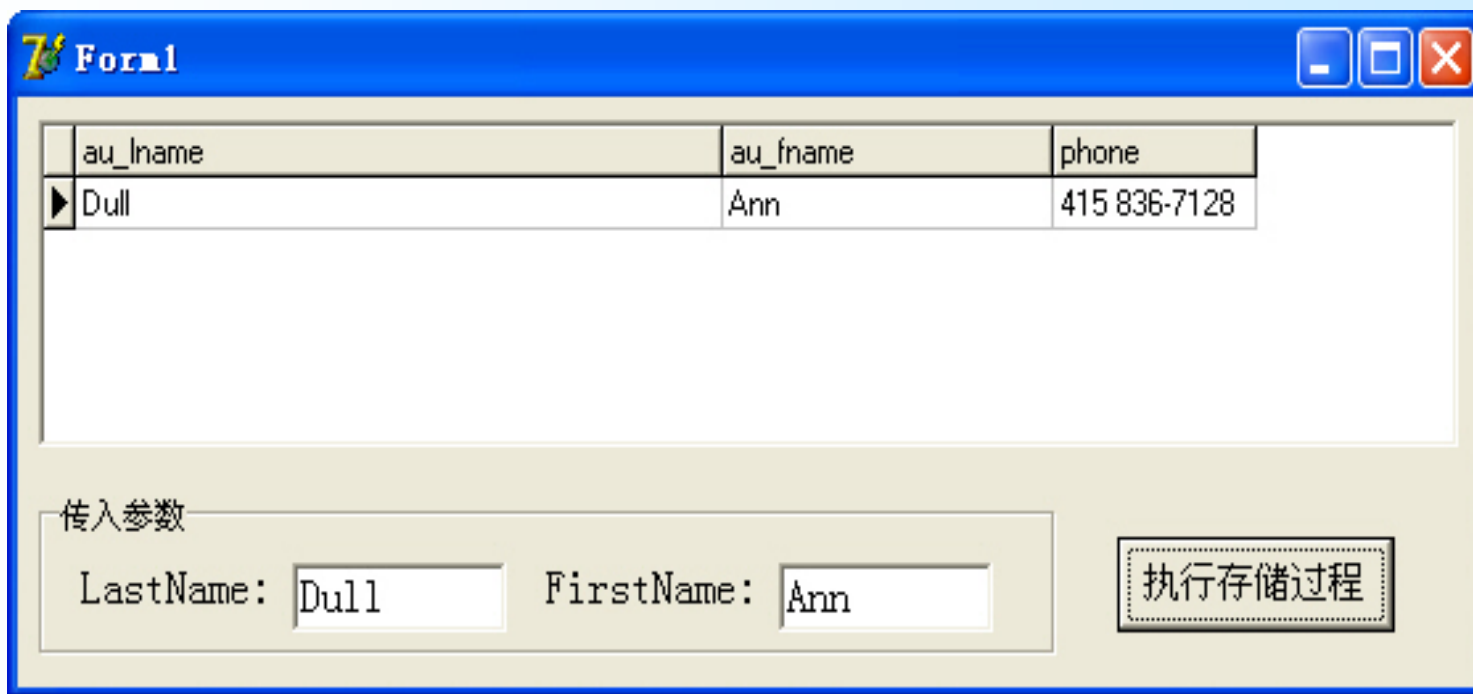
表10-4 各组件属性设置

组件名	属 性	属 性 值	说 明
ADOConnection1	ConnectionString		使用ConnectionString设置对话框设置该属性，连接到数据库Pubs。
ADOStoredProc1	Connection	ADOConnection1	指定使用的数据连接组件。
	ProcedureName	author_infor;1	指定使用的存储过程名
DataSource1	DataSet	ADOStoredProc1	指定使用的数据集。
DBGrid1	DataSource	DataSource1	为表格指定数据源。
	ReadOnly	True	设置为只读
Button1	Caption	执行存储过程	
Gropbox1	Caption	传入参数	
Label1	Caption	LastName	
Label2	Caption	FirstName	
Edit1	Text		为空
Edit2	Text		为空

## 10.2 使用存储过程

(3) 编写代码

(4) 运行程序



The screenshot shows a Windows application window titled "Form1". Inside the window, there is a table with three columns: "au\_lname", "au\_fname", and "phone". The first row of data shows "Dull", "Ann", and "415 836-7128". Below the table, there is a section labeled "传入参数" (Input Parameters). This section contains two text boxes: "LastName: Dull" and "FirstName: Ann". To the right of these text boxes is a button labeled "执行存储过程" (Execute Stored Procedure).

au_lname	au_fname	phone
Dull	Ann	415 836-7128

传入参数

LastName:  FirstName:

图10-6 执行结果

## 10.3 主从结构的数据库应用程序

有时我们需要将数据表格联系起来，这些表格具有一种一对多的关系。比如客户表和订单表，一个客户可能会有许多订单。当我们浏览客户订货信息时，希望在客户表选择一个客户后，能够显示该客户的所有订单。利用Delphi可以方便地设计出主从表结构的应用程序。

### 【例10-4】

使用Delphi自带的Access数据库dbdemos.mdb中的表customer和表Orders建立一个程序，实现主/从表。customer为主表，Orders为从表，两表之间通过字段CustNo链接。

### 【实现步骤】

- (1) 首先建立应用程序。
- (2) 创建数据模块

## 10.3 主从结构的数据库应用程序

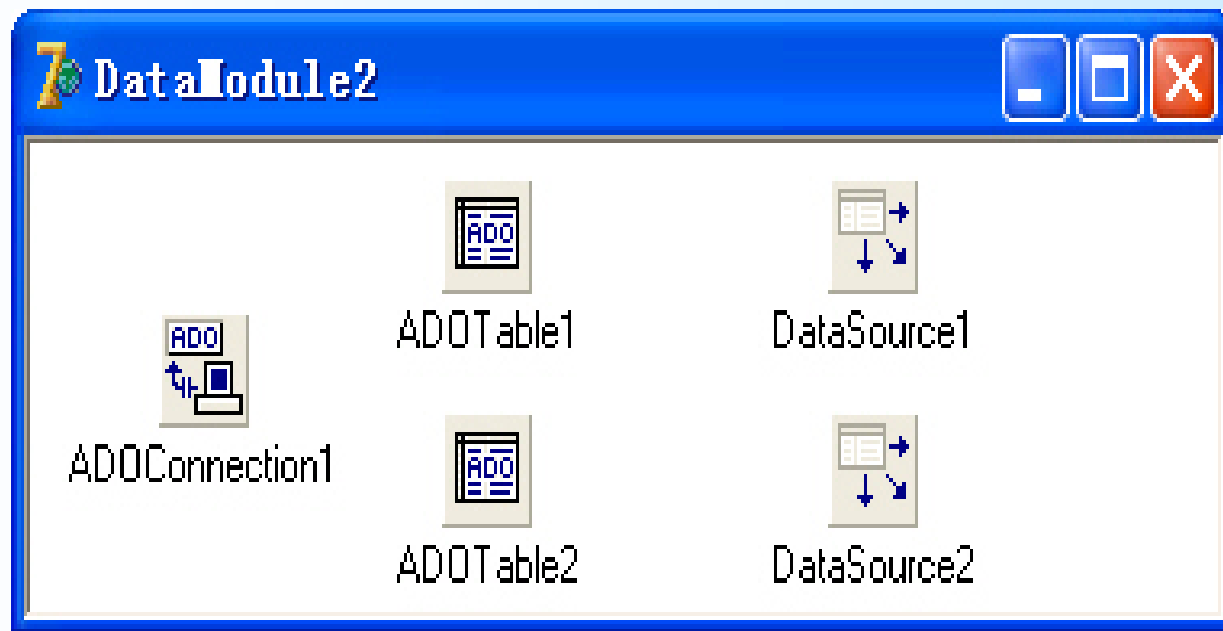


图10-7 数据模块设计界面

# 10.3 主从结构的数据库应用程序

表10-5 数据模块各组件属性设置

组件名	属 性	属 性 值	说 明
ADOConnection1	ConnectionString		使用ConnectionString设置对话框设置该属性，连接到数据库dbdemos.mdb。
ADOTable1	Connection	ADOConnection1	指定使用的数据连接组件。
	TableName	Customer	指定打开的数据表名
	Active	True	打开数据集。
DataSource1	DataSet	ADOTable1	指定使用的数据集。
ADOTable2	Connection	ADOConnection1	指定使用的数据连接组件。
	TableName	Orders	指定打开的数据表名
	MasterSource	DataSource1	指定主表数据源
	MasterFields	CustNo	设置主从表之间的关联字段
	Active	True	打开数据集。
DataSource2	DataSet	ADOTable2	

## 10.3 主从结构的数据库应用程序

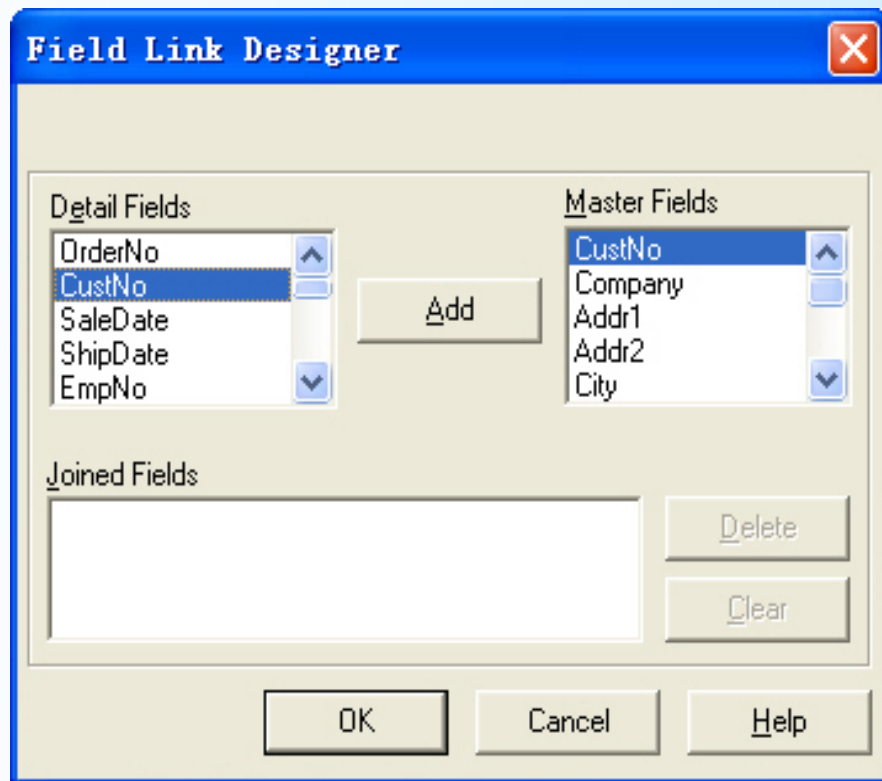


图10-8 链接字段设计器

## 10.3 主从结构的数据库应用程序

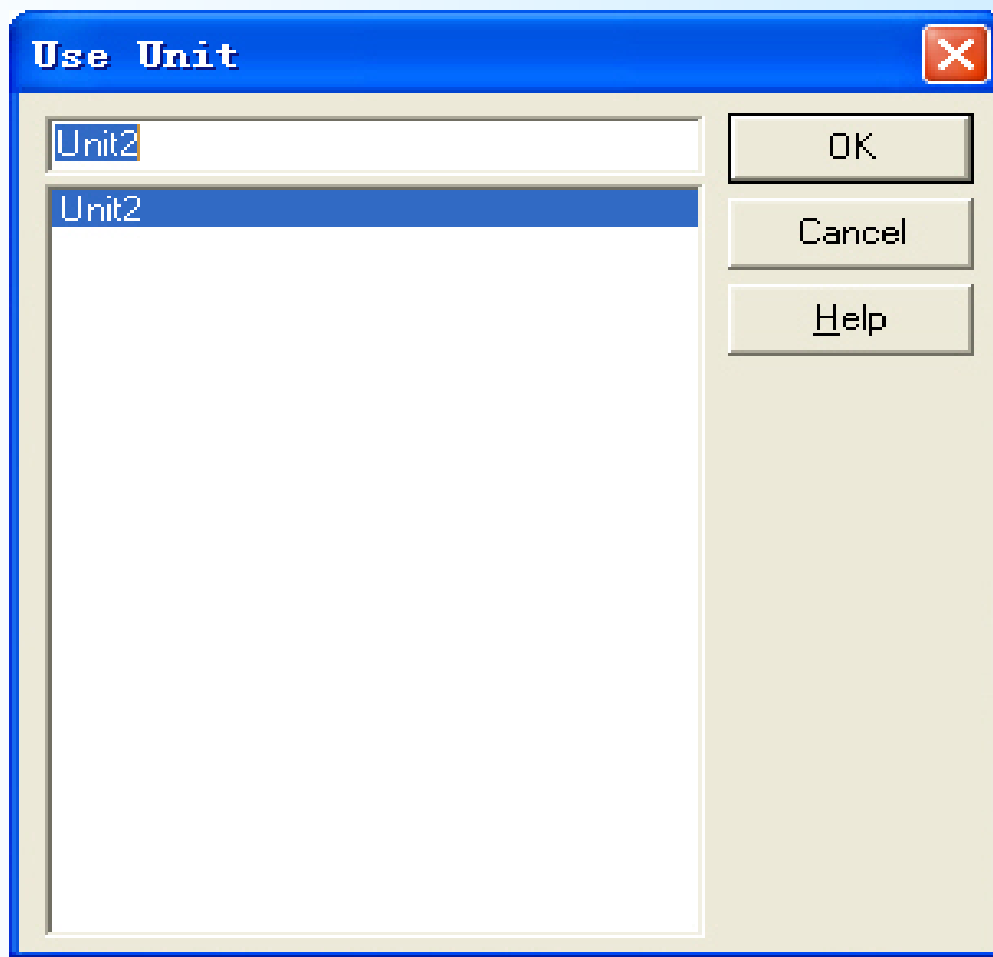


图10-9 引用单元对话框

# 10.3 主从结构的数据库应用程序

## (4) 定制窗体

The screenshot shows a Visual Basic form titled "Form1" with two data-bound tables. The first table displays customer information, and the second table displays order information. Both tables have scrollbars and navigation buttons.

CustNo	Company	Addr1	Addr2
1221	Kauai Dive Shoppe	4-976 Sugarloaf Hwy	Suite 103
1231	Unisco	PO Box Z-547	
1351	Sight Diver	1 Neptune Lane	
1354	Cayman Divers World Unlimited	PO Box 541	
1356	Tom Sawyer Diving Centre	632-1 Third Frydenhoj	

OrderNo	CustNo	SaleDate	ShipDate	EmpNo	ShipToContact
1023	1221	1988-7-1	1988-7-2	5	
1076	1221	1994-12-16	1989-4-26	9	
1123	1221	1993-8-24	1993-8-24	121	
1169	1221	1994-7-6	1994-7-6	12	
1176	1221	1994-7-26	1994-7-26	52	
1269	1221	1994-12-16	1994-12-16	28	

程序设计界面

## 10.3 主从结构的数据库应用程序

表10-6 窗体中各组件属性设

组件名	属 性	属 性 值	说 明
DBGrid1	DataSource	DataSource1	
	ReadOnly	True	表格为只读
DBGrid2	DataSourc	DataSource2	
	ReadOnly	True	表格为只读

## 10.3 主从结构的数据库应用程序

(5) 保存所有文件，然后运行程序

The screenshot shows a window titled 'Form1' with two data tables. The top table displays customer records, and the bottom table displays order records. The row for CustNo 1354 is selected in the top table, and its corresponding orders are listed in the bottom table.

CustNo	Company	Addr1	Addr2
1221	Kauai Dive Shoppe	4-976 Sugarloaf Hwy	Suite 103
1231	Unisco	PO Box Z-547	
1351	Sight Diver	1 Neptune Lane	
1354	Cayman Divers World Unlimited	PO Box 541	
1356	Tom Sawyer Diving Centre	632-1 Third Frydenhoj	

OrderNo	CustNo	SaleDate	ShipDate	EmpNo	ShipToContact
1104	1354	1992-7-18	1992-7-18	83	
1292	1354	1995-1-1	1995-1-1	136	

图10-10 主/从表关系的应用程序运行界面

# 本章实训指导

---

1. 掌握事务处理的应用程序的编写。
2. 掌握直接用SQL命令管理和执行存储过程。
3. 掌握使用TADOStoredProc组件执行存储过程。
4. 上机完成本章所有实例。

# 第11章 多层分布式应用的开发

本章主要内容：

- 多层分布式系统的概念
- 开发多层应用程序的有关组件
- 如何开发多层分布式数据库应用程序

# 11.1 多层数据库应用程序概述

## 11.1.1 数据库应用的体系结构

研究一下数据库应用的结构，可以发现它是由数据层、应用逻辑层、用户接口层构成的。数据层用于存取我们的数据，应用逻辑层用于规定用户如何使用数据，而用户接口层提供了用户访问数据的操作界面，而层间通过接口相连。这里只是在逻辑上将一个数据库应用分为三个层次。

实际开发数据应用系统时，可以在一个程序内实现所有三层功能，这就是单层数据库应用。前面所介绍的 **ACCESS** 数据库应用程序就是这种类型。

也可以将这三层分开，由不同的程序实现，并且还可以分别放在不同的机器上运行。这样就形成了双层、三层，甚至多层的数据库应用体系结构。

# 11.1 多层数据库应用程序概述

## 1. 单层结构

在单层数据库应用程序中，其体系结构如图11-1所示。

单层结构是一种早期的最简单的数据库系统，整个数据库应用系统安装在一台微机上，由一个用户独占，不同机器之间不能共享数据。

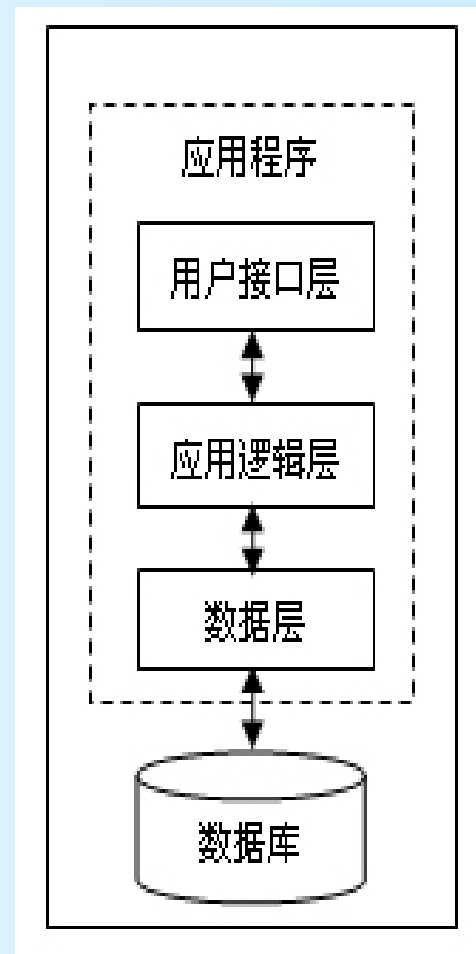


图11-1 单层结构的数据库应用系统

# 11.1 多层数据库应用程序概述

## 2. 双层结构

图11-2所示的是客户/服务器（Client/Server）体系结构，这是最典型的双层结构。

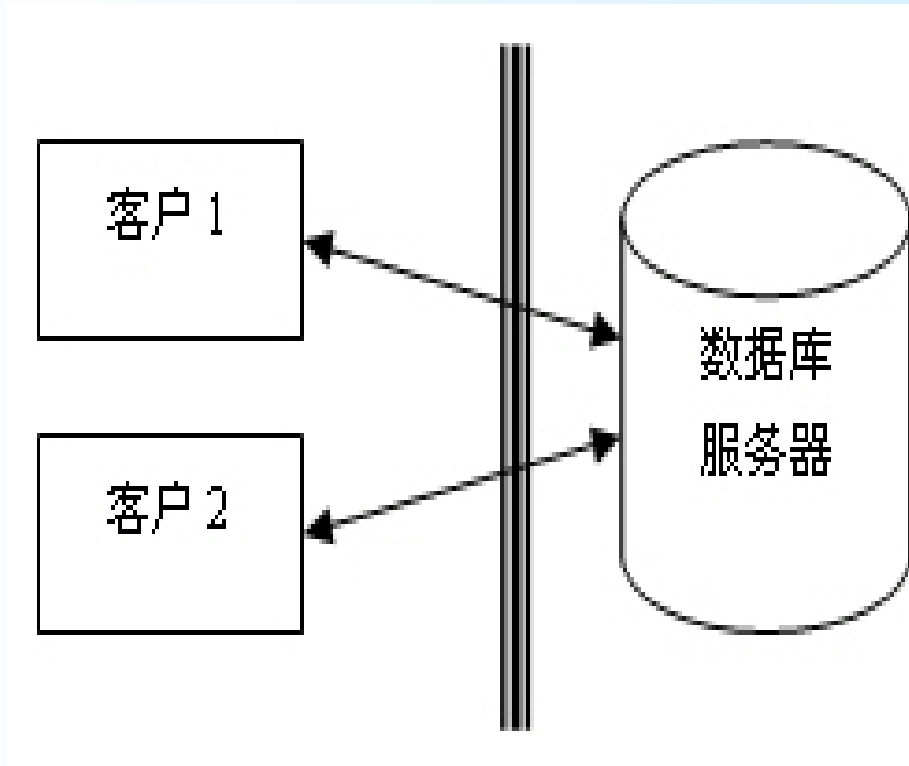


图11-2 客户/服务器体系结构

# 11.1 多层数据库应用程序概述

## 3. 三层结构与多层结构

三层结构如图11-3所示。在三层结构中，多了一个中间层，它把客户端和数据库服务器隔离开来，这样客户端对数据的所有操作都需要经过中间层的应用服务器，由应用服务器连接数据库服务器，应用服务器根据设定的应用逻辑对数据进行相应的处理，最后提交给后端数据库服务器。

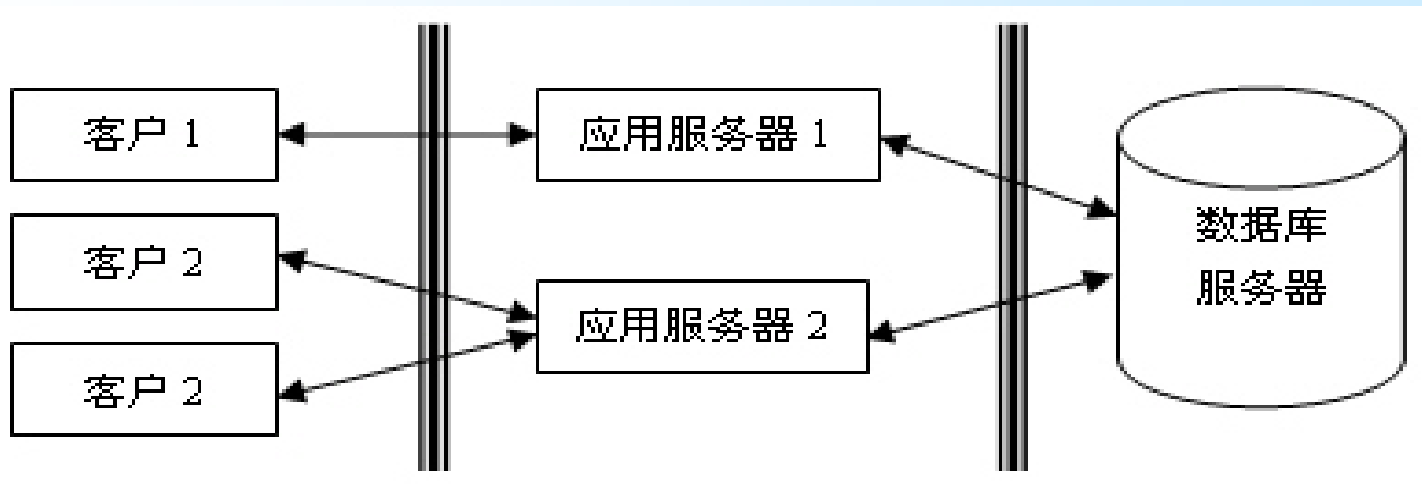


图11-3 三层体系结构

# 11.1 多层数据库应用程序概述

图11-4是一个基于B/W体系结构的多层数据库应用系统。

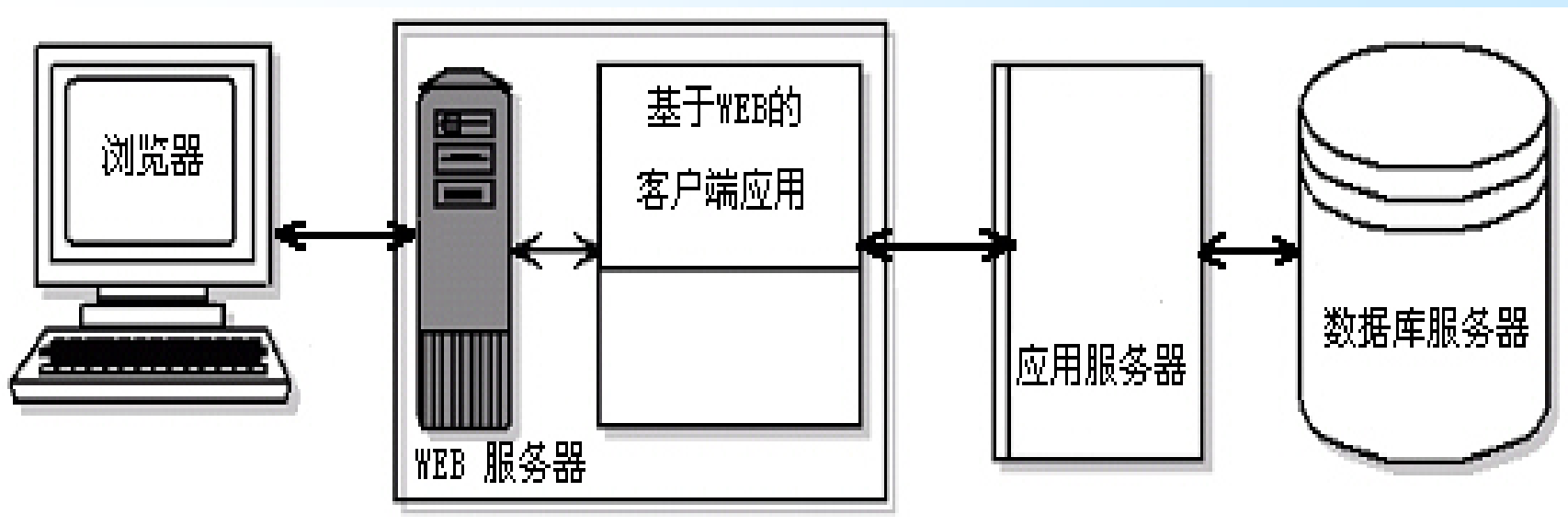


图11-4 基于B/W体系结构的多层数据库应用系统

# 11.1 多层数据库应用程序概述

## 11.1.2 Delphi的多层数据库应用体系结构

Delphi提出的MIDAS是把原来双层的数据库连接访问放到了应用服务器上，客户端只剩下了执行文件和MIDAS.DLL，前台和后台服务器通过DCOM机制互相沟通。其结构如图11-5所示。

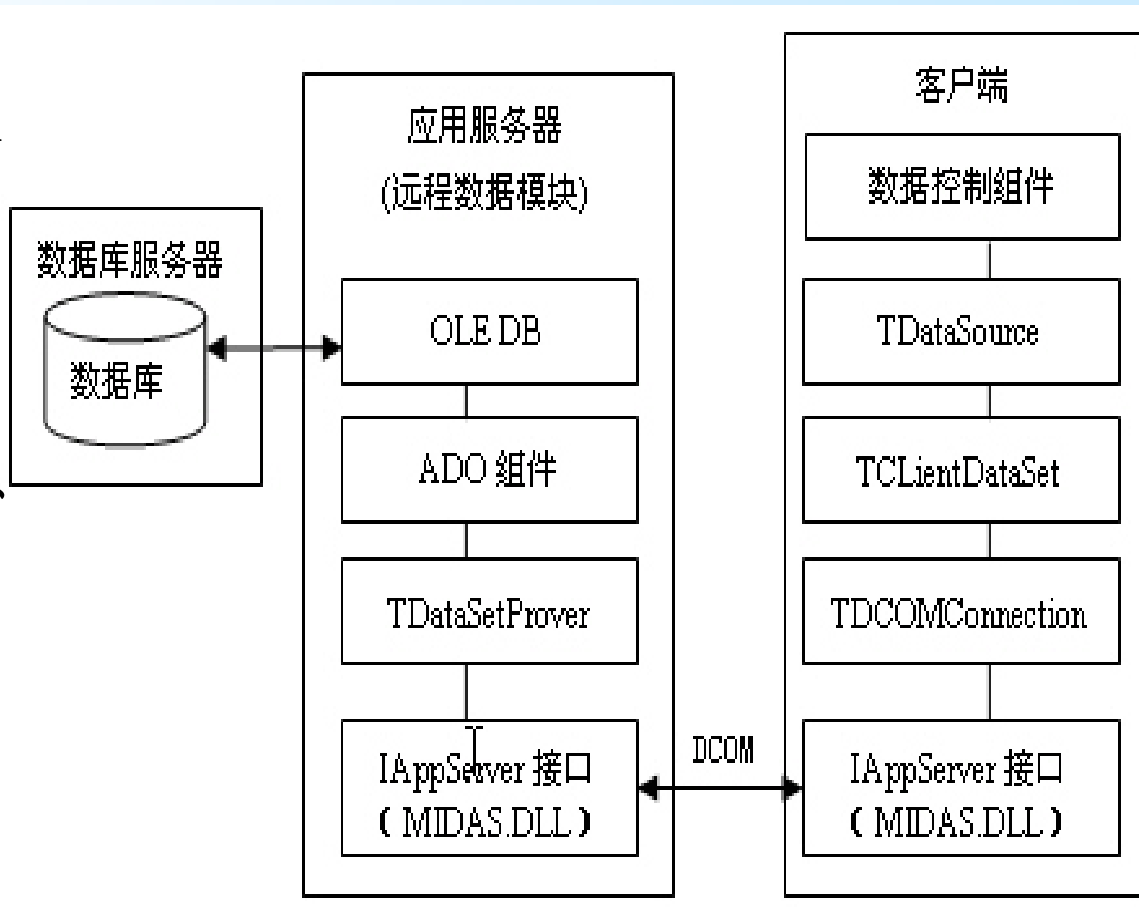


图11-5 MIDAS应用程序的结构

# 11.1 多层数据库应用程序概述

## 1. 应用服务器的结构

应用服务器的关键部件是**TRemoteDataModule**（远程数据模块），它提供了**IAppServer**接口。当客户程序与应用服务器建立了连接，就通过**IAppServer**接口来获得数据集提供者列表（list of providers）。

**TRemoteDataModule**组件是一个支持双重接口的自动化服务器，这种类型的远程数据模块适合于**DCOM**, **HTTP**, **Sockets**通讯方式。远程数据模块可以作为容器，但只能放置非可视化的组件。另外，远程数据模块上一般要放一个或几个**TDataSetProvider**或**TProvider**构件来提供**IProvider**接口。

**TDataSetProvider**组件存在于应用服务器端的远程数据模块中，用于连接数据模块中的数据集，并向客户端发送数据。

# 11.1 多层数据库应用程序概述

## 2. 客户端的结构

对于最终用户来说，多层体系结构中的客户程序与两层体系结构中的应用程序没有什么区别，在结构上，客户程序仍然通过标准的数据控件与用户交互。但与单机模式应用程序不同的是，多层体系结构中的客户程序是通过应用服务器提供的**IAppServer**接口获得数据的，也通过**IAppServer**接口申请更新数据。

在客户程序中，**MIDAS**连接组件**TDispatchConnection**扮演着极其重要的角色。不同的**MIDAS**连接组件使用不同的通讯协议将客户端连接到一个远程应用服务器，以获取一个**IAppServer**接口服务，然后获取数据集提供者列表。

通常使用**TDispatchConnection**派生的组件：

(1) **TDCOMConnection**组件使用**DCOM**将客户端连接到一个远程应用服务器。

# 11.1 多层数据库应用程序概述

(2) **TSocketConnection**组件使用**Sockets (TCP/IP)**将客户端连接到一个远程应用服务器。

(3) **TWebConnection**组件使用**HTTP**将客户端连接到一个远程应用服务器。

(4) **TSOapConnection**组件使用**SOAP**将客户端连接到一个远程应用服务器。

**TClientDataSet**组件是客户端的数据集组件，用于连接应用服务器中的**TDatasetProvider**，提供了客户端访问应用服务器商的**IAppServer**接口。

# 11.2 开发多层数据库应用的组件

## 11.2.1 TRemoteDataModule组件

TRemoteDataModule封装了多层数据库应用程序的对象和接口，它是由TDataModule类派生而来的。使用TRemoteDataModule向导建立DCOM应用服务器，会自动地添加TRemoteDataModule对象。在数据模块单元中会自动建立UpdateRegistry方法，用于注册该应用服务器。

### 1. TRemoteDataModule的基本属性

#### (1) Providers属性

表示远程数据模块中包含的已注册的Provider的集合。

#### (2) ProviderCount属性

表示远程数据模块中包含的已注册的Provider的数量。

#### (3) OldCreateOrder属性

指定何时发OnCreate事件和OnDestroy事件。

# 11.2 开发多层数据库应用的组件

## 2. TRemoteDataModule的基本方法

### (1) Lock方法

锁定数据模块，防止线程冲突。

### (2) RegisterProvider方法

用于在程序运行时调用底层的IAppServer接口获取一个可用的provider。

### (3) Unlock方法

解除对数据模块的锁定。

### (4) UnRegisterProvider方法

从AS\_GetProviderName方法返回的列表中删除一个Provider。

# 11.2 开发多层数据库应用的组件

## 9.2.2 TDataSetProvider组件

客户端的数据集是通过**Provider**（提供者）组件获取数据的。**Provider**接收来自客户端数据集的数据请求，然后索取所请求的数据，封装成可传输的数据包，并把数据传到客户端数据集。**Provider**接收来自客户端的数据更新，然后应用更新内容到数据库服务器、来源数据集，返回未能实施的更新到客户端数据集，作进一步调整。

### 1. TDataSetProvider的属性

# 11.2 开发多层数据库应用的组件

表11-1 TDataSetProvider的属性及说明

属 性	说 明
Constraints	是否传递数据库的约束给客户端，使得客户端能够在本地进行约束性检查，减少提交数据更新时产生的错误。缺省值为True。
DataSet	指定连接的数据集。
Params	集合属性，存储当前使用的参数列表。
ResolveToDataSet	指定数据更新是否直接提交给数据库服务器。缺省值为False。
Options	Options属性是一个集合，用于控制数据包的组成。选项见表
Resolver	获取自动建立的TCustomResolver组件，用于提交数据更新和调试更新错误。
UpdateMode	确定数据更新时如何定位一个记录。
Data	读取该属性时组件会自动调用GetRecord方法，获取所有的数据。
Exported	允许客户端通过IAppServer接口调用该组件。通常应该设为True。

# 11.2 开发多层数据库应用的组件

## 2. TDataSetProvider提交数据更新的方式

TDataSetProvider组件提供了两种方式来提交客户端的数据更新，向数据集提交数据更新和向数据库提交数据更新。

（1）向数据集提交数据更新

设置ResolveToDataSet属性为True。

（2）向数据库提交数据更新

设置ResolveToDataSet属性为False，表明

TDataSetProvider使用的更新组件为内部的TSQLResolver，它可直接提交客户的数据更新到数据集连接的数据库中，这样就绕开了使用的数据集。

### 11.2.3 TDCOMConnection组件

该组件使用DCOM建立客户端和应用服务器的连接，适用于建立企业内部网络（Intranet）。

## 11.2 开发多层数据库应用的组件

该组件的ComputerName属性用于设置服务器主机名称或者IP。该属性为空，表示服务器主机就为本本地机器。

### 11.2.4 TSocketConnection组件

要使用TCP/IP 方式来连接应用服务器，就需用到TSocketConnection组件。

TSocketConnection用Address属性或Host属性来定位应用服务器所在的计算机，前者用于指定IP地址，后者用于指定主机名，这两个属性是互斥的，只需要设置其中一个。此外，还要设置Port属性指定端口号，必须与应用服务器上运行的Scktsrver.exe或Scktsrvc.exe所使用的端口号一致，默认值是211。

# 11.2 开发多层数据库应用的组件

## 11.2.5 TClientDataSet组件

TClientDataSet组件用于获取远程数据模块的数据并建立本地的副本，提供了浏览和编辑数据的方法和属性，另外还提供了向数据库提交和调用数据的功能。该组件与TADOTable、TADODataSet等数据集组件一样，都是由TDataSet组件继承来的，因此具有许多数据集组件的公共属性、方法、事件；而最大的不同就是它是利用底层的IAppServer接口获取数据的。

在Delphi7中，TClientDataSet有三种途径获取数据：

- (1) 从文件中获取数据。
- (2) 从本地的另一个数据集中获取数据。
- (3) 通过IProvider接口从远程数据库服务器获取数据。

# 11.2 开发多层数据库应用的组件

1. TClientDataSet的属性
2. TClientDataSet的主要方法
3. TClientDataSet的主要事件
4. 提取数据

客户端通过TDispatchConnection组件与服务器建立了联系后，需要用TClientDataset组件连接到服务器中的provider上（从而连接到源数据集上）。首先，需要指定TClientDataset组件的RemoteServer属性为所连接的TDispatchConnection组件，使两者联系起来。完成之后，TClientDataset组件的列表框中就可以列出服务器的Provider列表，从中选择provider。最后，设置Active属性为True以打开TClientDataset从服务器上提取数据。

# 11.3 多层数据库应用程序开发

主要想解决这样几个问题：如何建立一个简单的分布式系统，如何使用**SQL**，如何传递附加信息和向客户提供服务器方法，如何建立主从表结构。

## 11.3.1 基于**DCOM**的客户/服务器程序开发

### 【例11-1】

编写一个服务器端程序。

服务器端程序实际上是个**COM** 项目，它本身连接数据源，再通过接口与客户端联系，这个**COM** 项目必须注册在服务器上。

连接的数据库是位于服务器**My-Server**上的数据库**dbdemos**。这个数据库是由**Delphi**自带的**dbdemos.mdb**导入到**MS SQL**服务器上的。

## 11.2 开发多层数据库应用的组件

### 【例9-2】

编写一个客户端程序。

在两层模式中，客户端(**Client**)程序是直接和服务器的数据源相连的。而多层模式，多个客户端连接的是一个应用程序服务器，应用程序服务器再连接到数据库服务器，因为收费是按客户数计算的，所以，数据库的使用费用比较低。

### 【例9-3】

在【例9-1】中的服务器端程序中加入进行客户计数的功能。

由于这个应用程序执行模式是 **Multiple Instance** 执行模式，所以当某个前台第一次连上线后，应用程序服务器会触发 **RemoteDataModule** 的 **OnCreate** 事件程序，而断线后又会执行 **OnDestroy** 事件的处理程序，因此就可以用这两个事件计算连上服务器的用户个数。至于查询个数的计算，则由 **TADODataSet** 的 **OnAfterOpen** 事件函数判断

## 11.2 开发多层数据库应用的组件

### 【例9-4】

在【例9-3】中的服务器端程序中为服务器接口添加属性和方法。

**TRemoteDataModule**提供的属性和方法是有限的，要增强应用服务器的服务可考虑为**IDcomServer**添加属性和方法，而客户端应用程序则可以通过**IAppServer**接口来调用服务器的属性和方法。

### 【例9-5】

通过**TClientDataSet**发送SQL命令。

前面在服务端执行由客户端输入SQL查询命令是通过添加新接口实现的，其实还有更简单的实现方法。通过设置**TClientDataSet**的**CommandText**属性，并在服务器端设置**DataSetProvider1**的**Options**属性为**poAllowCommandText**。

## 11.2 开发多层数据库应用的组件

**Params**属性用于向**Provider**提供查询参数值，可以在设计或是在运行期间设置，调用**Execute**或**Open**方法时一起传递给**provider**。如果运行带有参数的查询或是调用存储过程，在调用**Execute**时，**Params**属性会一起传递给**provider**。另外，若调用的是存储过程，执行后会返回输出参数。

# 11.2 开发多层数据库应用的组件

## 11.3.2 基于Socket的客户/服务器程序开发

### 1. TCP/IP协议和Socket（插座）

在客户端，用一个新的控件TSocketConnection代替原来的TDCOMConnection，其余的连接不变，这就大大简化了客户端的设计。但是，TSocketConnection使用的是TCP/IP协议进行通讯，因此，客户端总是连接到应用服务器的某个特定的端口（Server Port）。在服务器端需要另外配置和运行一个程序scktsrvr.exe(可以在<Delphi>\Bin目录下找到这个程序)，该服务程序用于监听服务端口，应用服务器使用该端口提供服务。

（1）选择服务端口

（2）启动Scktsrvr.exe程序

在服务器端，找到Delphi的目录，找到：Program Files\Borland\Delphi 6\Bin\Scktsrvr.exe。

## 11.2 开发多层数据库应用的组件

双击运行，就可以在Windows任务栏看到一个TCP/IP的Service启动的图标（下图圆圈中的图标就是）。



图11-21任务栏上显示的Scktsrvr.exe图标

在图标上单击右键，选Properties，就出现一个对话框：

# 11.2 开发多层数据库应用的组件

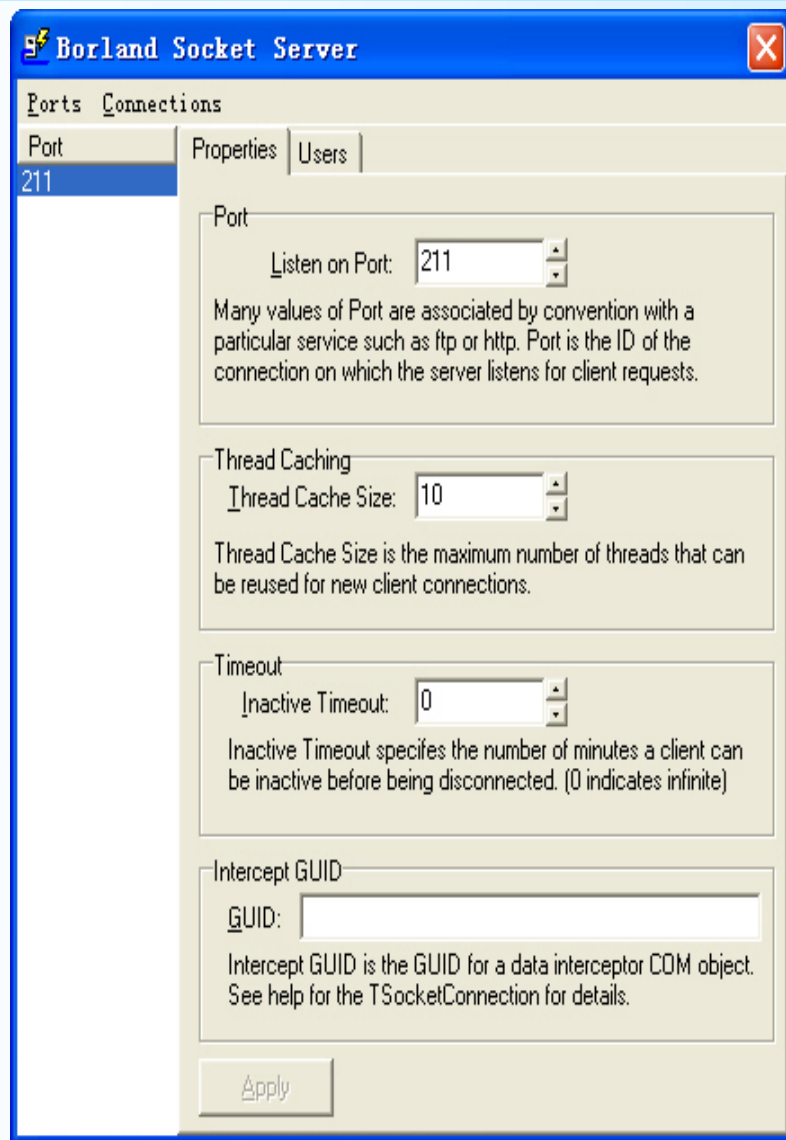


图11-22 Borland Socket Server设置界面

## 11.2 开发多层数据库应用的组件

把Socket Server变成Windows Server的方法，在DOS命令提示符下，进入C:\Program Files\Borland\Delphi7\Bin目录，然后执行Scktsrvr /install命令。安装成功后可以看到一个安装成功的信息显示。如图所示：

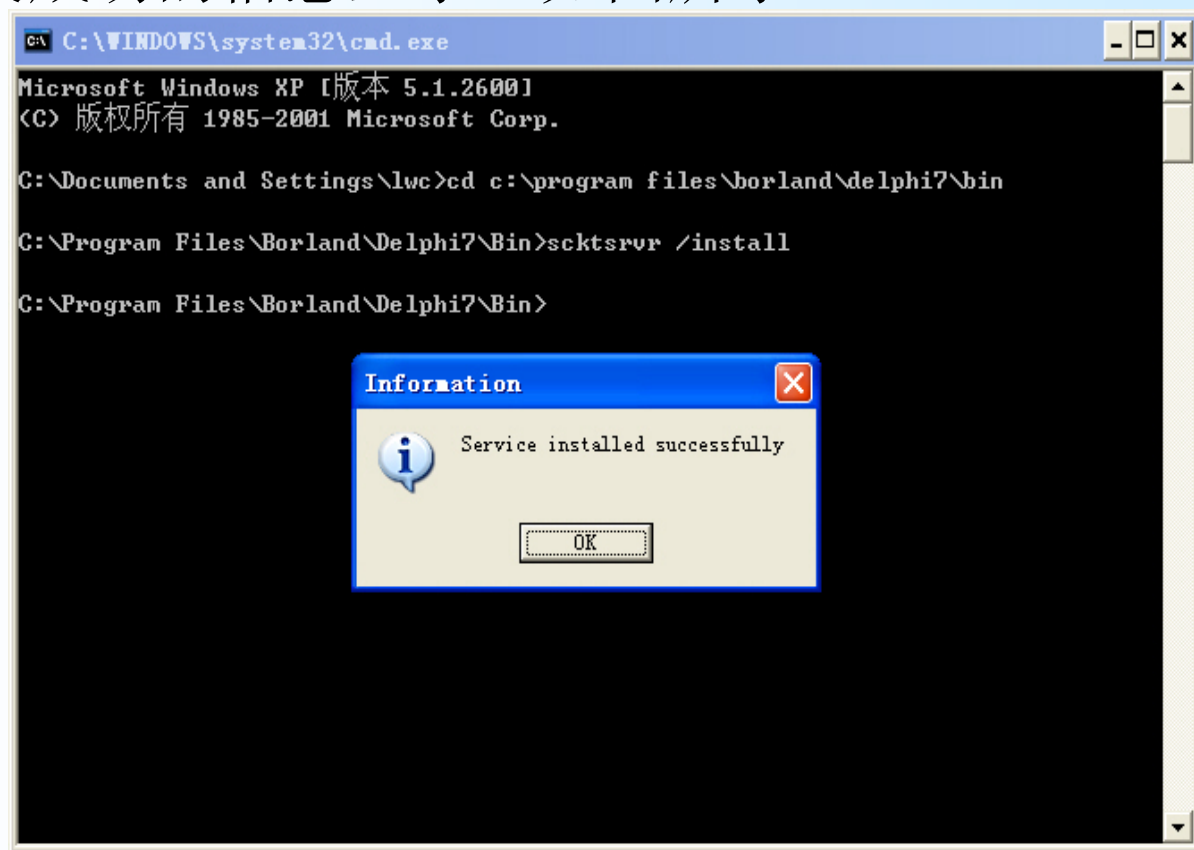


图11-23 从命令行安装Scktsrvr.exe

# 11.2 开发多层数据库应用的组件

这时打开【控制面板】的【管理工具】，启动【服务】工具程序。



图11-24 Windows服务管理器

# 11.2 开发多层数据库应用的组件

---

## (3) 客户端程序设计

# 本章实训指导

---

- 1、掌握创建多层应用程序的机制。
- 2、了解多层应用程序的组件。
- 3、掌握服务器程序的编写。
- 4、掌握客户程序的编写。
- 5、上机模仿本章实例，并理解程序设计思想。

# 第12章 图书馆管理信息系统设计

本章主要内容：

- 系统需求分析
- 系统总体设计
- 数据库的创建
- 系统实现代码编写

# 12.1 系统需求分析

图书馆管理信息系统是图书馆的支持系统，当前设计应满足以下需求：

- (1)图书馆将图书借给读者，读者和图书都必须在系统中注册。
- (2)系统通过设置帐号和密码控制用户访问，读者和管理员可修改自己的密码。
- (3)图书馆负责购买图书，对于流行的书一般要多买几本，如果图书破旧或过期则可以从图书系统中删除。
- (4)图书馆管理员负责与读者打交道，他们的工作要得到系统的支持。
- (5)借书者可以事先通过网络预约要借的图书，然后到管理员处办理正式借书手续。
- (6)图书借出有时间限制，一旦超期，管理员应及时通知借书者归还图书。

# 12.1 系统需求分析

---

- (7)图书馆可以方便地产生、更新和删除系统中与书目、借书者、借书记录和预约记录等信息。
- (8)系统能够运行在Windows系统下，用户可通过局域网操作，还应该有一个非常好的图形用户界面(GUI)
- (9)系统应该具有很好的可扩展性。

# 12.2 系统设计

## 10.2.1 系统功能模块划分

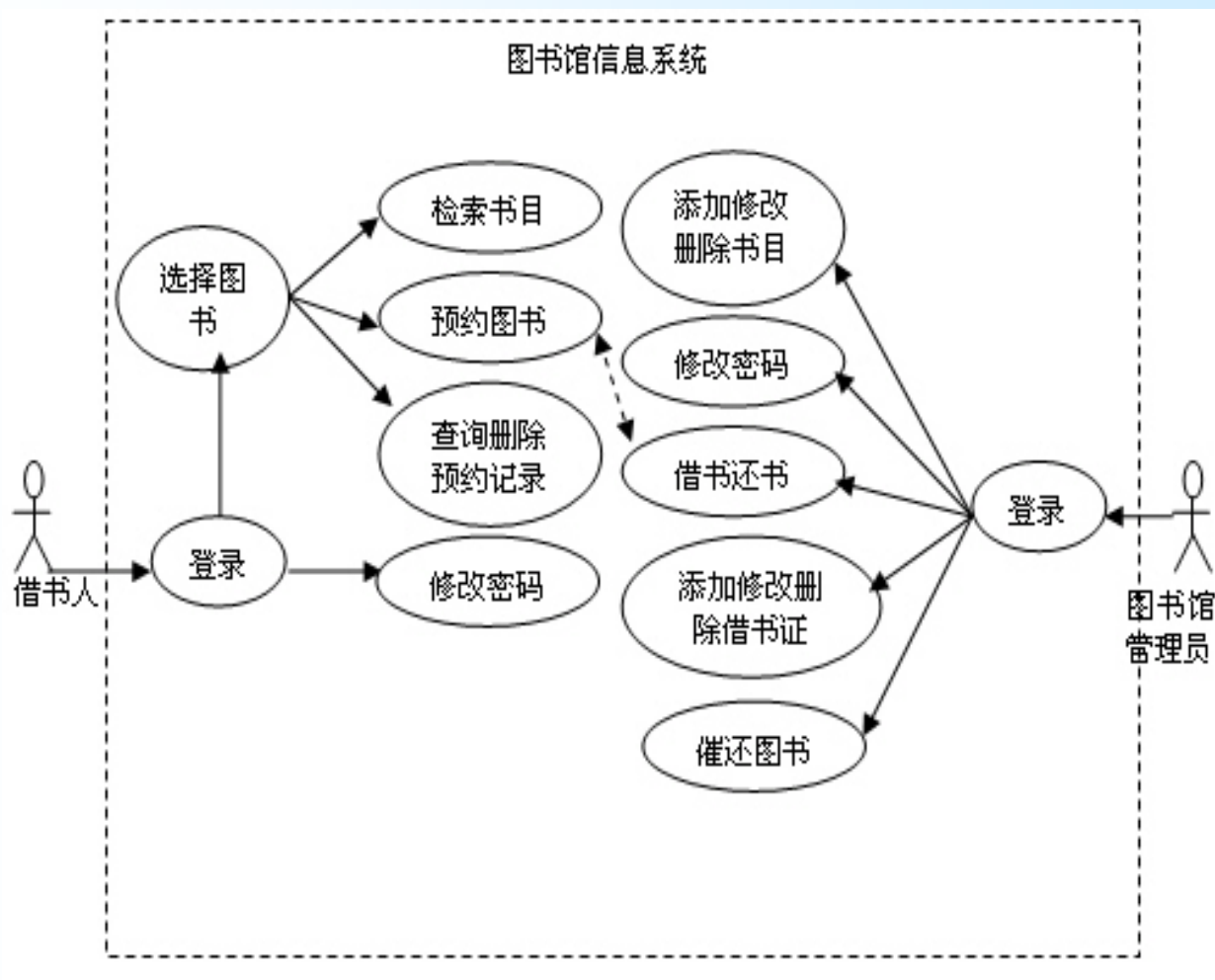


图12-1 图书馆信息系统功能模块划分图

# 12.2 系统设计

## 12.2.2 功能描述

### 1. 图书馆管理系统

- (1) 添加、修改、删除书目
- (2) 添加、修改、删除借书证
- (3) 修改管理员密码
- (4) 催还图书
- (5) 借书、还书
- (6) 登录

### 2. 图书馆服务系统

- (1) 检索书目
- (2) 预约图书
- (3) 查询、删除预约记录
- (4) 修改密码
- (5) 登录

# 12.2 系统设计

## 12.2.3 结构设计

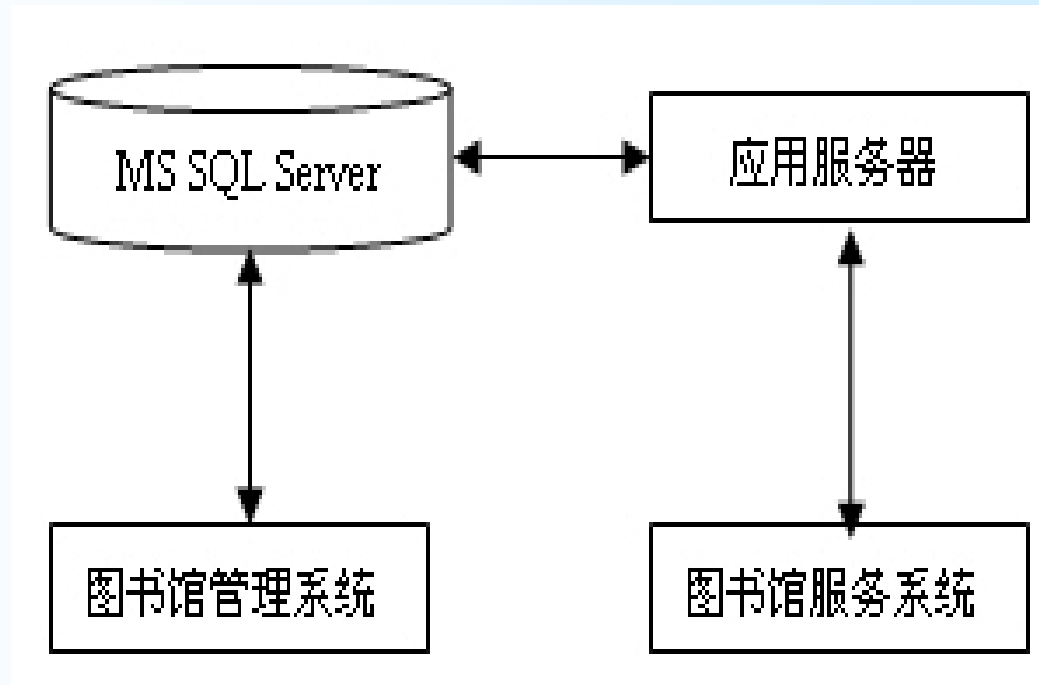


图12-2 图书馆信息系统结构图

# 12.2 系统设计

## 12.2.4 数据库设计

### 1. booksinf表

字段名称	字段类型	长度	说明
book_id	varchar	20	图书编号，设为主键
book_name	varchar	50	书名
book_type	varchar	20	类别
author	varchar	20	作者
publisher	varchar	50	出版社
pub_date	smalldatetime	8	出版日期
book_num	smallint	2	图书数量
borrow_num	smallint	2	借出数
booking_num	smallint	2	预订数
price	real	4	单价
locality	varchar	20	书架位置
mem	varchar	50	备注

# 12.2 系统设计

## 2. Admins表

字段名称	字段类型	长度	说明
name	varchar	10	管理员姓名，设为主键
password	varchar	16	管理员密码

## 3. Readers表

字段名称	字段类型	长度	说明
libcard_id	varchar	10	借书证号，设为主键
reader_name	varchar	10	读者姓名
password	varchar	16	读者密码
sex	varchar	2	性别
birth_date	smalldatetime	8	出生日期
dept	varchar	6	单位
phone	varchar	10	联系电话
address	varchar	50	住址
card_date	日期/时间	8	办证日期
status	varchar	4	借书证状态：启用、挂失、停用
mem	varchar	50	备注

# 12.2 系统设计

## 4. Records表

字段名称	字段类型	长度	说明
NO	int	4	记录编号，设为主键
libcard_id	varchar	10	借书证号
reader_name	varchar	10	读者姓名
book_id	varchar	20	图书编号
book_name	varchar	50	书名
borrow_date	smalldatetime	8	借书日期
return_date	smalldatetime	8	还书日期
conBorrow_date	smalldatetime	8	续借日期
status	varchar	1	状态：预约、借书、续借、还书

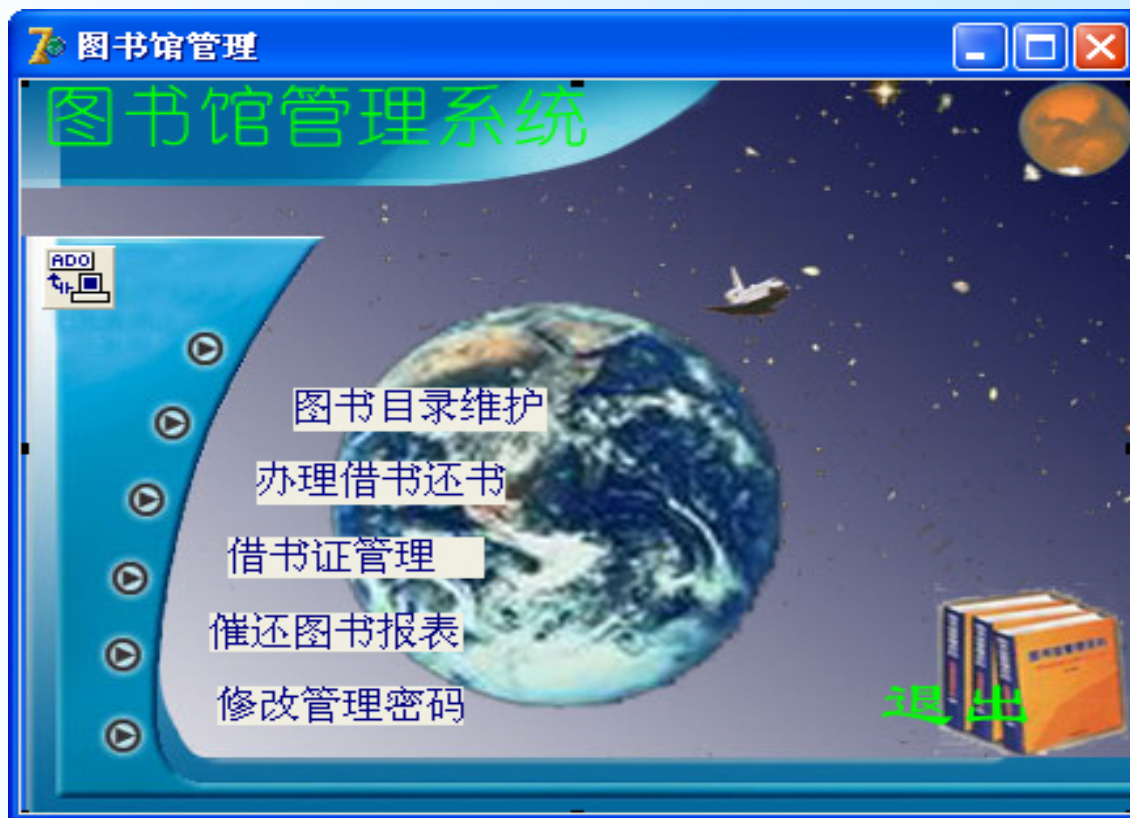
# 12.3 系统实现

## 12.3.1 创建系统目录

## 12.3.2 创建数据库

## 12.3.3 图书馆管理系统设计

### 1. 主界面



## 12.3 系统实现

### 2. 登录模块



A screenshot of a Windows-style dialog box titled "管理员登录" (Administrator Login). The window has a blue title bar with standard minimize, maximize, and close buttons. The main area is a light beige rectangle with a dotted border. Inside, there are two input fields: the first is labeled "姓名" (Name) and the second is labeled "密码" (Password). Below these fields are two buttons: "登录" (Login) and "退出" (Exit). A small icon with a question mark and a magnifying glass is located in the top-left corner of the main area.

管理员登录

姓名

密码

登录 退出

# 12.3 系统实现

## 3. 图书目录维护

图书目录维护

图书编号	书名	类别	作者	出版社	出版日期	馆藏数	借出数	预约数

所有的 = 查询 添加 删除 修改 退出

## 12.3 系统实现

### 4. 图书目录记录编辑界面



The screenshot shows a Windows-style application window titled "frm\_book\_record" with a standard title bar (minimize, maximize, close buttons). The main content area has a blue header with the title "图书目录卡" (Book Catalog Card) in white. Below the header is a light yellow form area containing ten text input fields arranged in two columns. The labels for these fields are in Chinese. At the bottom of the form are two buttons: "保存" (Save) and "取消" (Cancel).

Field Label	Field Name
图书编号	DBEdit1
出版日期	DBEdit6
书 名	DBEdit2
藏书数量	DBEdit7
图书类别	DBEdit3
单 价	DBEdit8
作 者	DBEdit4
存放位置	DBEdit9
出 版 社	DBEdit5
备 注	DBEdit10

Buttons: 保存 (Save), 取消 (Cancel)

## 12.3 系统实现

### 4. 借书证管理模块

读者信息维护

借书证号	读者姓名	密码	性别	出生日期	所在单位	联系电话	现住址	状态	办证日期	备注

ADO

所有的 =

查询 添加 删除 修改 退出

## 12.3 系统实现

The screenshot displays a Windows-style application window titled "frm\_reader\_record". The window has a blue title bar with standard minimize, maximize, and close buttons. The main content area has a light blue header with the text "借书证记录" (Borrowing Record). Below the header is a form with a light beige background. The form contains the following fields and controls:

Field Label	Control Type	Control ID
借书证号	Text Box	DBEdit1
读者姓名	Text Box	DBEdit2
密 码	Text Box	DBEdit3
性 别	ComboBox	DBComboBox1
出生日期	Text Box	DBEdit5
备 注	Text Box	DBEdit10
所在单位	Text Box	DBEdit6
联系电话	Text Box	DBEdit7
现在住址	Text Box	DBEdit8
办证日期	Text Box	DBEdit9
使用状态	ComboBox	DBComboBox2

At the bottom of the form, there are two buttons: "保存" (Save) and "取消" (Cancel).

图12-8 借书证记录编辑界面

## 12.3 系统实现

### 5. 办理借书还书模块

借书记录表 | 办理借还书手续

借书证  确定

记录号	借书证号	读者姓名	图书编号	书名	借书日期	借书日期

图12-9 借书还书管理界面

## 12.3 系统实现

借书还书管理

借书记录表 办理借还书手续

借书记录

借书证号 DBText1 读者姓名 DBText2 借阅状态 DBText16

图书编号 DBText11 图书名称 DBText14

借书日期 DBText12 续借日期 DBText15 还书日期 DBText13

图书信息

图书编号 DBText3 图书名称 DBText4

作者姓名 DBText5 出版社 DBText6

藏书数 DBText 借出数 DBText 预约数 DBText 存放位置 DBText10

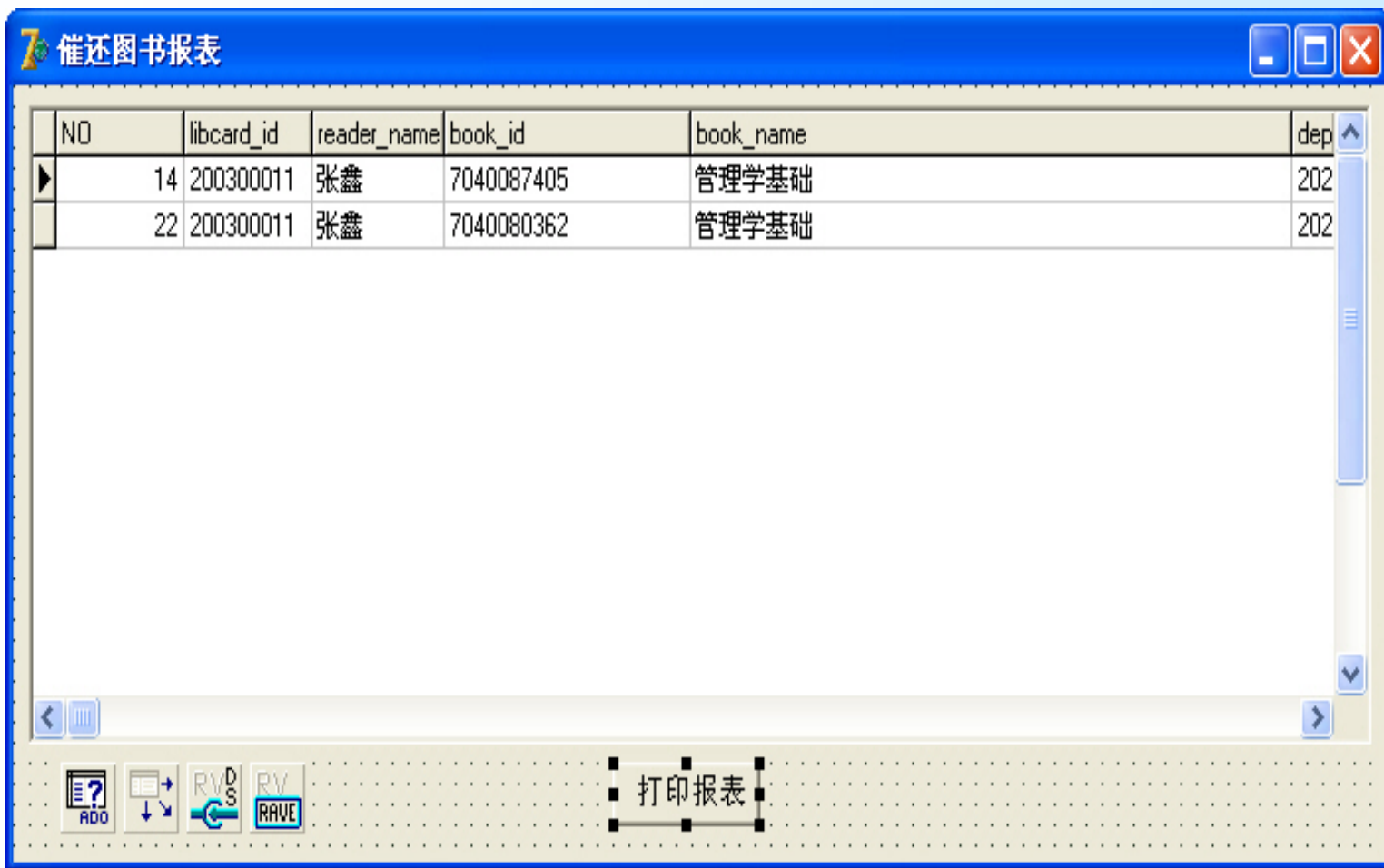
管理员操作

确认

图12-10 借书还书管理界面

## 12.3 系统实现

### 6. 催还图书报表和预约图书报表



催还图书报表

NO	libcard_id	reader_name	book_id	book_name	dep
14	200300011	张鑫	7040087405	管理学基础	202
22	200300011	张鑫	7040080362	管理学基础	202

打印报表

图12-11 预约图书报表界面

# 12.3 系统实现

## 7. 修改管理密码



图12-12 修改管理密码界面

# 12.3 系统实现

## 12.3.4 应用服务器设计

### 1. 首先建立应用程序

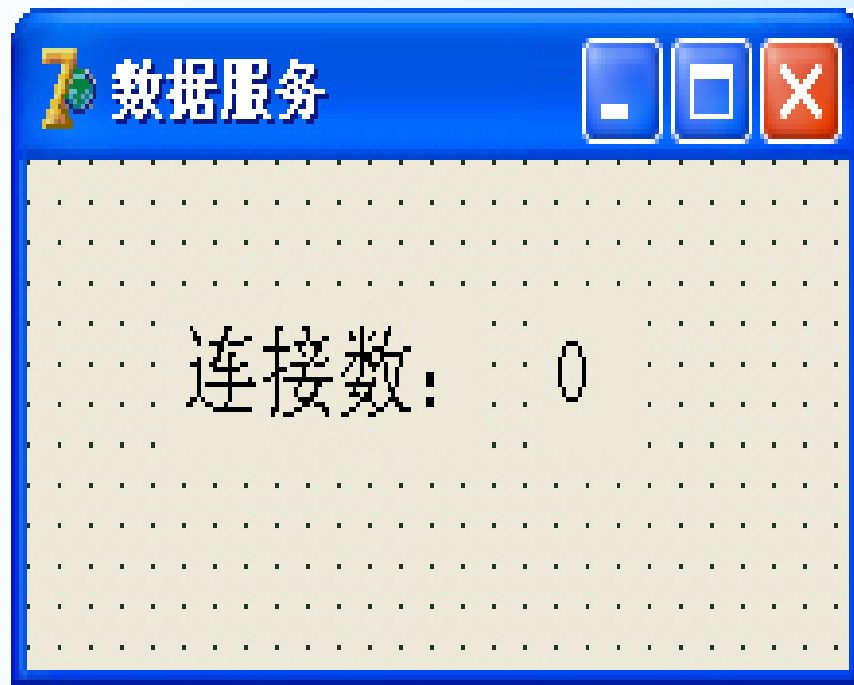


图12-13 服务器界面

# 12.3 系统实现

## 2. 建立数据模块

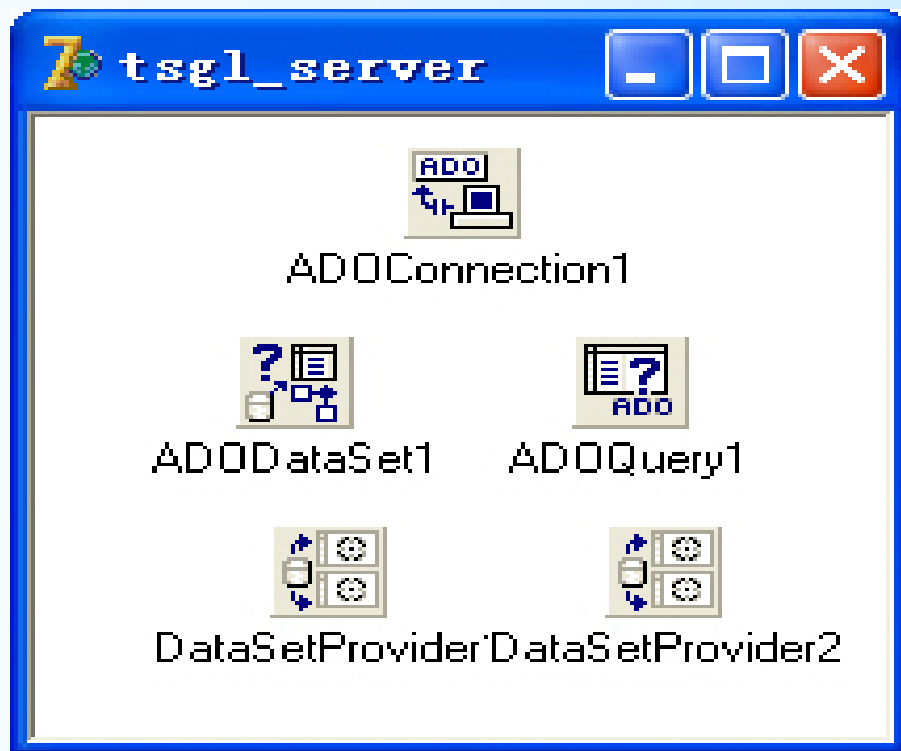


图12-14 数据模块

# 12.3 系统实现

## 3、数据模块代码设计

### 12.3.5 图书馆服务系统

#### 1. 服务系统主界面

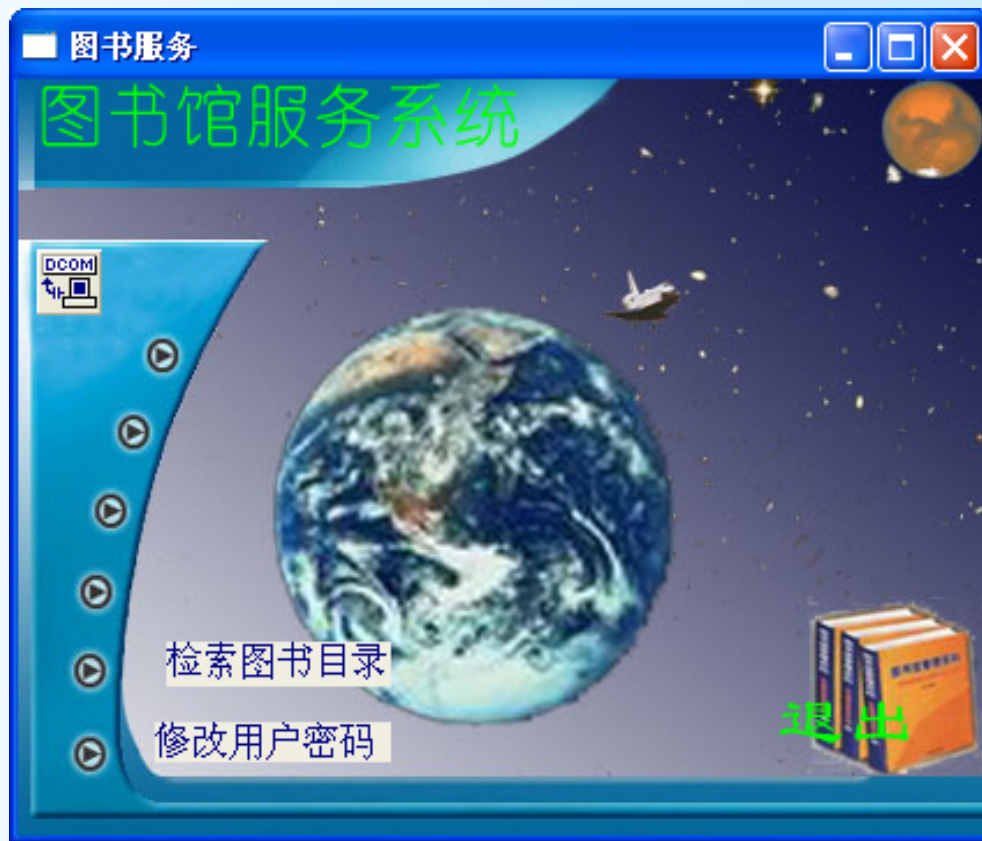


图12-15 服务系统主界面

# 12.3 系统实现

## 2. 登录界面

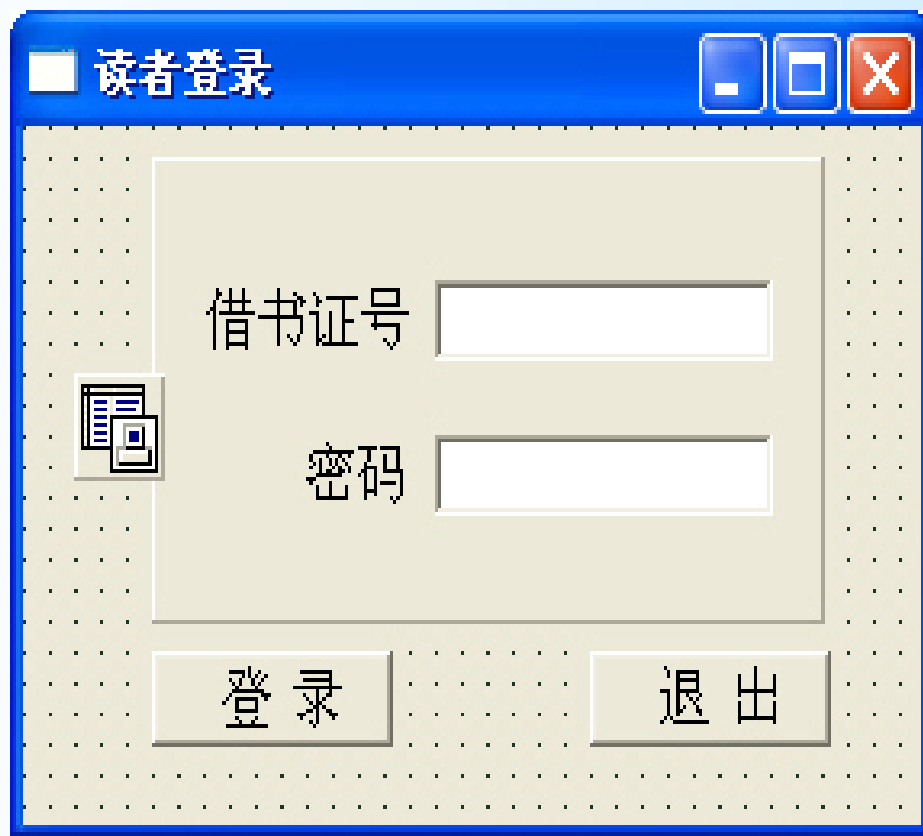


图12-16登录界面

# 12.3 系统实现

## 3. 图书选择界面

图书信息查询

图书编号	书名	类别	作者	出版社	出版日期
------	----	----	----	-----	------

所有的  
=  
  
查询

添加到预约

记录号	图书编号	书名	借书日期	还书日期	续借日期	借阅手续
-----	------	----	------	------	------	------

显示所有记录  
删除记录  
退出

图12-17 图书选择界面

# 12.3 系统实现

## 4. 修改密码模块



修改密码

原密码

新密码

再输一次

确定 取消

图12-18 修改密码界面

# 本章实训指导

---

- 1、掌握数据库应用程序开发的基本步骤。
- 2、上机调试本章程序，并模拟相关功能模块完成借书证管理模块、催还图书报表模块的设计。