新版软件步步高

PowerBuilder 6.0

高级教程

捷新工作室 编著

- 最流行的数据库客户端开发环境
- 优秀的第四代语言开发环境
- 数据库开发人员必备之利器
- 内容深入、实例丰富
- 操作性强、易于掌握
- 重点介绍实用技术要点

國防工業出版社

新版软件步步高(提高本)

PowerBuilder 6.0 高级教程

捷新工作室　编著

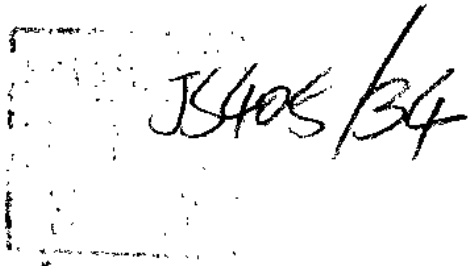
国防工业出版社
·北京·

图书在版编目(CIP)数据

PowerBuilder 6.0 高级教程/捷新工作室编著 .—北京：国防工业出版社,1999.7
(新版软件步步高·提高本)
ISBN 7-118-02090-0

Ⅰ.P… Ⅱ.捷… Ⅲ.数据库管理系统-软件工具,PowerBuilder 6.0-基本知识 Ⅳ.TP311.56

中国版本图书馆 CIP 数据核字(1999)第 18543 号



国防工业出版社出版发行
(北京市海淀区紫竹院南路 23 号)
(邮政编码 100044)
北京怀柔新华印刷厂印刷
新华书店经售
*
开本 787×1092 1/16 印张 14½ 328 千字
1999 年 7 月第 1 版 1999 年 7 月北京第 1 次印刷
印数:1—4000 册 定价:20.00 元

总　序

在信息时代，知识成为推动社会生产力发展的一个最重要的因素，知识经济的轮廓在一些发达国家已经出现。以知识为基础的知识经济，其重要特点是信息产业的迅猛发展和产业的信息化，其内核是创新。我国是发展中国家，应该更加重视运用世界最新技术成果，有重点有选择地引进先进技术，增强自主创新能力，逐步实现技术发展的跨越。

在充分估量未来科学技术，特别是以计算机技术为先导的高技术发展对综合国力、社会经济结构和人民生活的巨大影响的基础上，为实现科教兴国战略多于实事，国防工业出版社组织了由数十位在计算机应用开发第一线工作的年富力强的博士、硕士组成的捷新工作室，编写出版《新版软件步步高(基础本)》和《新版软件步步高(提高本)》丛书。前者重在普及，后者追求提高，两者结合起来，力图满足多层面读者的需要。

《新版软件步步高(基础本)》的出版目的是普及新版软件的基本知识、基本操作技术，主要面向初学者，包括初次涉足该领域的机关、公司、企事业单位技术人员、大专院校师生及各类培训班学员，也可作为广大再就业职工现想的培训和学习教材。另外，对新技术感兴趣的读者也可将本丛书作为了解世界最新技术的窗口。

《新版软件步步高(提高本)》的出版目的是为有一定基础的读者找到提高专业技术水平和最新软件开发、操作技术的最佳途径，主要面向中高级读者，包括对该软件有一定基础知识，又希望提高自己专业技术水平的机关、公司、企事业单位技术人员、大专院校师生及各类高级培训班学员。

丛书的共同特点是突出一个“新”字，强调一个“精”字，力争一个“快”字。“新”是指软件的版本新；“精”是指精选的国内外流行最广、叫得最响的新版软件；“快”是指在保证质量的前提下，实现周期短，面市快。

丛书的内容覆盖最新高级语言开发环境(Visual J++ 6.0 Visual Basic 6.0, Visual FoxPro 6.0, Delphi 4.0)，微机操作系统(中文 Windows 98，中文 Windows NT 5.0)，图形图像处理软件(Photoshop 5.0，中文 CorelDRAW 8.0)，Internet 浏览器(Internet Explorer 5.0)，Web 页面设计环境(FrontPage 98)，Internet 开发环境(Visual InterDev 6.0)，大型数据库客户端开发工具(PowerBuilder 6.0)等。它们都是 1998 年以来推出的最新版软件。同时，我们还将把握新技术的脉搏，适时充实新的内容。

我们相信，丛书的出版必将为广大读者开辟跟踪、掌握、运用、进而创造新技术的最佳途径。

由于时间仓促，书中疏漏之处，敬请广大读者指正。

前　言

PowerBuilder是图形界面的分布式数据库前端开发工具，利用它可以快速开发出面向对象的数据库应用程序。相对于以前的版本，PowerBuilder 6.0增加了平台支持的数量，增强了建立因特网和内部网上分布式应用的能力，扩展了语种支持。显著的变化是，PowerBuilder 6.0已经开始从传统的客户/服务器开发环境转变成完全支持World Wide Web应用的开发环境。

PowerBuilder 6.0新增和增强的特性包括多个方面。首先，PowerBuilder 6.0增强了整个开发环境的性能和易用性，改进了部分对象和画笔的功能，在界面上继续向Windows 95的界面标准靠拢，并大量采用了Windows 95标准界面。

其次，数据窗口是PowerBuilder中功能十分强大的对象之一。对自由风格和列表风格的数据窗口来说，用户可以在保持原有风格的同时将所需的内容自动转换成HTML页面，也可以在数据窗口中放上按钮对象，通过单击按钮完成系统预定义的操作或编程人员自定义的操作，利用组框还可以把数据窗口中的数据有机地组织在一起。

再次，PowerSoft将以前单独出售的Internet开发工具箱集成到PowerBuilder的开发环境中。新的Java代理生成器使Java客户机能够直接访问应用逻辑和PowerBuilder 6.0对象。

除了功能上的增强和改进外，PowerBuilder 6.0提供了一个全新的调试器，它具备条件断点、查看内存对象、浏览指定源代码、查看变量、单步执行、断点执行等一系列实用功能。PowerBuilder 6.0还提供了一个应用程序性能跟踪与分析工具，通过该工具，可以找到应用程序存在的性能瓶颈。

本书是一本详细介绍PowerBuilder 6.0数据库开发技术的中、高级参考书，内容包括数据窗口控件、用户对象和用户事件、在数据窗口中设计报表、掌握数据查询技术、使用PowerSoft Repository、数据仓库的使用、连接多个数据库、使用数据管道、调试和发行应用程序等。PowerBuilder 6.0的强大功能只有通过亲自使用才能真正体会到，希望读者能从本书获得掌握这一强大工具的捷径。

本书内容精练，实例丰富，适合有较好PowerBuilder开发基础的读者、各大专院校师生、各类中高级数据库开发技术培训班学员参考使用。

内 容 简 介

PowerBuilder 6.0 是 Sybase 公司的子公司 PowerSoft 推出的新一代数据库应用开发工具，本书是一本详细介绍 PowerBuilder 6.0 数据库开发技术的中高级参考书，内容包括数据窗口控件、用户对象和用户事件、在数据窗口中设计报表、掌握数据查询技术、使用 PowerSoft Repository、数据仓库的使用、连接多个数据库、使用数据管道、调试和发行应用程序等。

本书内容精练，实例丰富，适合有较好 PowerBuilder 开发基础的中高级读者、各大专院校师生、各类中高级数据库开发技术培训班学员参考使用。

目 录

第一章　数据窗口控件

数据窗口控件是应用程序在窗口中展示数据窗口对象的唯一途径，数据窗口控件与数据窗口对象的结合构成了应用程序访问和操作数据库数据的主要手段。两者的结合在提供强大功能的同时，不可避免地也带来了更大的复杂性。数据窗口一方面功能强大、灵活方便，另一方面又复杂多变、难以驾驭。

与其它 PowerBuilder 对象一样，数据窗口控件也有一组属性、事件和函数，不过它们的数量相当庞大，掌握起来有一定的难度。数据窗口对象常用属性可以通过数据窗口控件的对象函数来访问，当然，通过使用属性表达式，应用程序几乎可以操纵数据窗口的所有属性。

数据窗口在处理数据时很有特色，它在客户机的本地内存中开辟了四个缓冲区：主缓冲区、删除缓冲区、过滤缓冲区、原始缓冲区，从数据库中检索到数据后，数据窗口根据不同情况把数据放置到不同的缓冲区。四个缓冲区各司其职，协作完成数据的增删改操作，最后把结果提交给数据库管理系统。

除了在 DataWindow 画笔中可以定义数据窗口对象外，PowerBuilder 还提供了根据 SQL SELECT 语句和指定的属性动态创建数据窗口的能力，这样，就能够构造更加灵活的应用程序，以适应不断变化的用户需求。

数据窗口的属性、事件和函数是灵活运用数据窗口的基础，也是数据窗口的奥妙所在。事实上，只要有足够的代码（不考虑效率的话），数据窗口能够完成任何功能。

另外，在本章中我们还将简单介绍功能与数据窗口控件相似、但没有可视特征的 PowerBuilder 对象——数据仓库（DataStore）的用法，有关数据仓库的较详细的内容将在第六章介绍。

1.1　使用数据窗口控件

在《PowerBuilder 6.0 基础教程》第八章中，我们介绍了使用 DataWindow 画笔定义数据窗口对象的方法，运用这些方法能够设计出与众不同、表达形式丰富的数据窗口对象，但设计好的数据窗口对象如何在应用程序中使用呢？通过数据窗口控件发挥数据窗口对象的功能，是运用数据窗口对象的一条有效途径（另一条途径是在数据仓库中使用数据窗口对象）。数据窗口控件与其它控件（比如单选钮、复选框、单行/多行编辑框、各种列表框等）一样，是附属于窗口的一个对象，它像桥梁一样把窗口和数据窗口对象联系起来。通过数据窗口控件，程序能够显示、修改、控制数据窗口对象，并且响应用户的操作。数据窗口控件功能十分强大，具有众多的事件、属性和函数，本节中我们介绍数据窗口控件的一般使用方法，另外说明应用程序如何连接数据库。

1.1.1 运用数据窗口的方法

无论用 PowerBuilder 开发哪种数据库应用程序，在应用程序中运用数据窗口的一般方法都是相似的，基本步骤如下。

(1) 使用 DataWindow 画笔创建数据窗口对象。

(2) 在窗口上放置数据窗口控件。

(3) 通过属性设置或编码将数据窗口控件与数据窗口对象联系起来。

(4) 设置数据窗口控件的属性，以控制它的外观和行为。

(5) 如果应用程序尚未与数据库建立连接，那么在某个事件处理程序中使用 CONNECT 语句与数据库建立连接。

(6) 将数据窗口控件与事务对象联系起来(使用数据窗口控件的对象函数SetTrans Object ( )或 SetTrans ( ))。

(7) 使用数据窗口控件的对象函数 Retrieve ( )把数据库中的数据装入数据窗口中。

(8) 针对具体应用，编写某些数据窗口控件事件的事件处理程序，响应用户的操作。

(9) 需要时使用数据窗口控件的对象函数 Update ( )保存用户对数据的修改。

1.1.2 放置数据窗口控件

通过数据窗口控件运用数据窗口对象是数据窗口的最常见用法。数据窗口控件可以放置到窗口上，也可以放置到定制可视用户对象中，它们的处理方法都是一致的。

在窗口上放置数据窗口控件的步骤如下。

(1) 单击画笔栏上的 Window 画笔，打开要放置数据窗口控件的窗口。

(2) 单击 Window 画笔工具栏图标 上的箭头，打开控件图标列表，从中选择 DataWindow 控件图标，其位置如图 1.1 所示。

(3) 单击窗口上想放置数据窗口控件的地方，此处即出现一个空白的方框，该方框就是数据窗口控件。

图 1.1 DataWindow(数据窗口)控件图标

(4) 根据需要移动数据窗口控件的位置和改变它的大小。

在定制可视用户对象中放置数据窗口控件的方法与在窗口中放置的方法十分类似，可参照操作。

数据窗口对象是用 DataWindow 画笔定义的对象，它以多种风格表现、操作数据库中的数据，并且以独立对象的形式保存在 PowerBuilder 应用库中。

数据窗口控件是粘贴到窗口上的一个对象，它在 Window 画笔中定义，并且不能作为独立对象保存到应用库中。

1.1.3 连接数据窗口控件与对象

窗口上放置了数据窗口控件后，还需要将数据窗口对象与它结合起来才能发挥数据窗口控件的作用，其结合方法如下。

(1) 双击放置在窗口上的数据窗口控件，或右击数据窗口控件并从弹出菜单中选择 Properties 菜单项。

(2) 系统显示 DataWindow 属性对话框。

(3) 在 General 标签页中，DataWindow Object Name 编辑框用于指定与数据窗口控件相关联的数据窗口对象，在该编辑框中可以直接键入数据窗口对象的名字，不过更常用的方法是单击 Browse...按钮，系统显示如图 1.2 所示的 Select DataWindow 对话框。

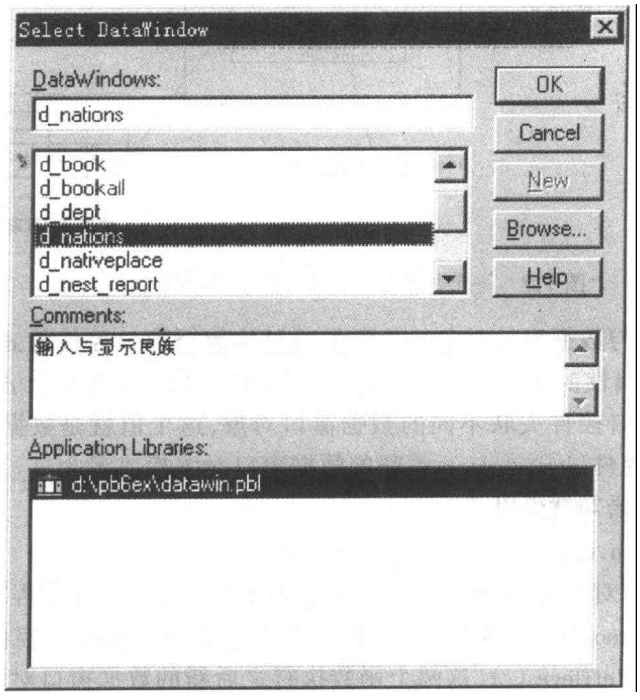


图 1.2 Select DataWindow 对话框

(4) 在该对话框底部的 Application Libraries 列表框中单击包含要选择数据窗口对象的应用库。

(5) 通过单击在中部列表框中选择数据窗口对象，此时该对象的名称出现在顶部的编辑框中、相应数据窗口对象的注释显示在 Comments 中(当然，直接在编辑框中键入数

据窗口对象名也可以)。

(6) 单击 OK 按钮返回属性对话框。

(7) 在属性对话框中单击 OK 按钮,就把数据窗口对象与数据窗口控件结合了起来。

关闭数据窗口控件的属性对话框后,数据窗口对象就出现在数据窗口控件中了。从上面的操作可以看出,数据窗口控件与图片框有许多相似之处,它的大小控制了数据窗口对象在窗口上占据空间的大小。

1.1.4 动态链接数据窗口对象

除了在数据窗口控件的属性对话框中直接设置数据窗口控件所关联数据窗口对象外,还能够在程序中动态地链接新的数据窗口对象,这样,一个数据窗口控件就能够在不同的时刻动态显示不同的数据窗口对象。图 1.3 显示了一个数据窗口控件在不同时刻对应多个数据窗口对象的情形。

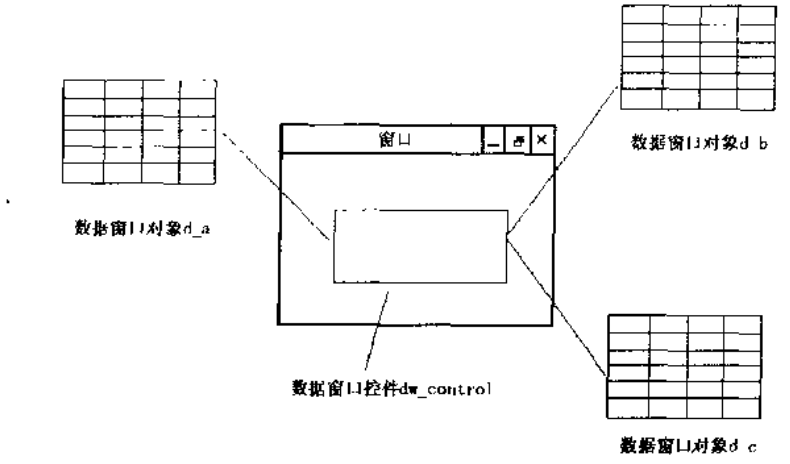


图 1.3 一个数据窗口控件关联多个数据窗口对象

数据窗口控件与数据窗口对象的关联是通过给数据窗口控件的 dataobject 属性赋值实现的,dataobject 属性的数据类型为字符串(String)。程序中通过给 dataobject 属性赋不同的值而让数据窗口控件关联不同的数据窗口对象,这个值就是数据窗口对象的名称。例如,将数据窗口控件 dw _ person 关联的数据窗口对象换成名为 d _ person _ next,只要在程序中写上下述语句就可以了:

```
dw _ person.dataobject = "d _ person _ next"
```

当然,数据窗口对象 d _ person _ next 要已经定义并保存在应用程序库中。

每当修改了 dataobject 属性后,还需要依次重新执行数据窗口控件的对象函数 SetTransObject ( )和 Retrieve ( ),这两个函数执行之后新的数据窗口对象才能在数据窗口控件中显示出来。

1.1.5 为数据窗口控件分配事务对象

用 DataWindow 画笔定义数据窗口对象时,如果选择了从数据库中提取数据的数据源,那么在数据窗口控件能够检索数据之前,还必须为数据窗口控件分配事务对象。当然,该事务对象还必须已经与数据库建立连接。

1. 与数据库建立连接

应用程序与数据库的连接通过事务对象来完成，在建立连接前需要首先给事务对象的相关属性赋值，然后用嵌入式 SQL 语句 CONNECT 建立连接。不同的数据库管理系统使用的事务对象属性也不尽相同。

如果应用程序只访问一个数据库，那么使用 PowerBuilder 的缺省事务对象 SQLCA 就可以了。SQLCA 是个全局对象，在应用程序的任何地方都可以访问。下面是使用 SQLCA 与 ODBC 数据源建立连接的简单示例。

```
//设置事务对象属性
SQLCA.DBMS = "ODBC"
SQLCA.DBParm = "ConnectString = 'DSN = Powersoft Demo DB V6; UID = dba;
    PWD = sql'"
//与数据库连接
CONNECT USING SQLCA;
//检查连接是否成功
If SQLCA.SQLCode < 0 Then
MessageBox("连接失败",SQLCA.SQLErrText,Exclamation!)
End If
```

上面的示例中直接把连接参数写在程序中，这种方式在应用程序需要访问其它数据库时要修改代码，更合理的方式是利用 PowerBuilder 的初始化文件 PB.INI，下面是更通用的代码：

```
environment env                    // 保存环境信息
string      startupfile            // 保存初始化文件名

/* 获得环境信息 */
IF ( GetEnvironment(env) < > 1 ) THEN
    MessageBox( "系统出错", "得不到环境信息~n终止应用 ..." )
    HALT  //终止应用程序的执行
END IF

/* 根据当前使用的操作系统选择初始化文件 */
CHOOSE CASE env.OSType
    CASE Windows!, WindowsNT!
        startupfile = "pb.ini"
    CASE Sol2!, AIX!, OSF1!, HPUX!
        startupfile = ".pb.ini"
    CASE Macintosh!
        startupfile = "PowerBuilder Preferences"
    CASE ELSE
```

```
        MessageBox( "系统出错", "未知的操作系统。\n 终止应用..." )
        HALT
    END CHOOSE
/* 根据当前 PB.INI 的设置值设置 SQLCA 属性 */
SQLCA.DBMS       = ProfileString (startupfile, "database", "dbms", "")
SQLCA.database   = ProfileString (startupfile, "database", "database",
                   "")
SQLCA.userid     = ProfileString (startupfile, "database", "userid", "")
SQLCA.dbpass     = ProfileString (startupfile, "database", "dbpass", "")
SQLCA.logid      = ProfileString (startupfile, "database", "logid", "")
SQLCA.logpass    = ProfileString (startupfile, "database", "LogPassWord",
                   "")
SQLCA.servername = ProfileString (startupfile, "database", "servername", "")
SQLCA.dbparm     = ProfileString (startupfile, "database", "dbparm", "")

//与数据库连接
CONNECT USING SQLCA;
//检查连接是否成功
If SQLCA.SQLCode < 0 Then
  MessageBox("连接失败",SQLCA.SQLErrText,Exclamation!)
End If
```

需要时,应用程序也可以创建新的事务对象,以适应同时连接到多个数据库管理系统的要求。创建新的事务对象的方法是:先说明一个类型为 Transaction 的事务对象变量,然后用 CREATE 语句创建事务对象实例。下面的语句创建了一个名字为 DBTrans 的事务对象实例:

```
transaction DBTrans
DBTrans = CREATE transaction
```

创建了对象实例后,像前面介绍的 SQLCA 那样给事务对象属性赋值,然后用 CONNECT 语句建立与数据库的连接。

用 CREATE 语句创建的事务对象不再使用时,应该用 DESTROY 语句删除该对象,以节省系统资源。

某个数据库连接不再使用时,应该及时地使用 DISCONNECT 语句断开与数据库的连接。数据库连接是数据库服务器的宝贵资源。

2. 为数据窗口控件分配事务对象

在使用数据窗口控件检索数据前,必须通知数据窗口使用哪个事务对象来操作数据库(实际上也就是告诉数据窗口从哪个数据库中检索数据)。要完成这个任务,可以使用数据窗口控件的对象函数 SetTransObject( )。其语法格式为:

```
dwcontrol.SetTransObject ( transaction )
```

其中，dwcontrol 是数据窗口控件的名称，transaction 是事务对象名。SetTransObject ( )函数执行成功时返回 1，失败时返回 - 1。通过该函数的返回值，应用程序能够知道 SetTransObject( )函数的执行情况。

例如：

dw_1.SetTransObject (SQLCA)

把数据窗口控件 dw_1 与连接数据库的事务对象 SQLCA 联系在一起。

一般在数据窗口控件所在窗口的 Open 事件中执行 SetTransObject( )函数。

数据窗口控件还有一个对象函数用于数据窗口控件与事务对象建立联系，这个函数就是 SetTrans ( )，其语法格式与 SetTransObject ( )函数相同。两者的区别在于：SetTrans ( )函数不需要应用程序使用 CONNECT 语句建立事务对象与数据库的连接，也不需要用 DISCONNECT 语句断开与数据库的连接，PowerBuilder 会自动完成这些任务。每当操作数据库时，PowerBuilder 都会完成连接、操作、断开数据库这一系列步骤。因此，当数据窗口检索数据时，数据窗口做一次 CONNECT、检索、DISCONNECT 操作；当数据窗口更新数据库时，它又完成一次 CONNECT、更新、DISCONNECT 操作。当然，事务对象的属性还是要按照连接具体数据库的要求来设置。熟悉数据库管理系统的人都知道，建立连接、断开连接的操作会大大降低应用程序的执行效率。对绝大多数数据库管理系统来说，CONNECT、DISCONNECT 操作是件极为耗时的工作(启动 PowerBuilder 后，第一次使用 DataWindow 画笔时，Select DataWindow 对话框的出现速度明显慢于第二次使用 DataWindow 画笔就是个明证)。但 SetTrans ( )函数也并非一无是处，如果数据库管理系统支持的用户数较少，而你希望有更多的用户使用数据库时，SetTrans ( )函数增加了各用户成功连接的机会。SetTransObject ( )函数在使用之前要求首先建立事务对象与数据库的连接，然后一直保持这一连接，直到代码执行 DISCONNECT 语句后才断开与数据库的连接。因此，它在检索和更新数据时所花的时间只是检索和更新所需的时间，效率上明显高于 SetTrans ( )函数。因此，应该尽可能使用 SetTransObject ( )函数建立数据窗口控件与事务对象的联系。

在调用 SetTransObject( )函数或 SetTrans( )函数前，数据窗口控件必须已经与某个数据窗口建立起联系，否则函数执行失败。

1.1.6 装入数据

数据窗口控件与事务对象建立联系之后，已经在两者之间架起了一座桥梁，但是，数据还在数据库中，现在需要调用数据窗口控件的对象函数 Retrieve( )把数据装入数据窗口中。

1. Retrieve( )函数

Retrieve( )函数的语法格式为：

dwcontrol.Retrieve ({argument, argument . . .})

其中：

dwcontrol 是数据窗口控件名。

argument 用于检索参数，参数个数及类型要与数据窗口对象数据源中使用的 SQL

SELECT 语句中的检索参数相同。

该函数执行成功时返回一个长整数指示显示的数据行数(即主缓冲区中的数据行数),失败时返回 -1。如果参数中有空值(NULL),则该函数返回空值。

例如,语句

```
lrc = dw_1.Retrieve ( )
```

执行无参数检索,返回值放到 long 型变量 lrc 中。

再如,假定数据窗口控件 dw_emp 包含的数据窗口对象使用下述 SQL SELECT 语句:

```
SELECT Name, sal, rgn From Employee
    WHERE sal > :Salary and rgn = :Region
```

其中 :Salary 和:Region 是 DataWindow 画笔中定义的参数,对应的检索语句可以为:

```
lrc = dw_emp.Retrieve (2000,"北京")
```

当数据窗口对象的数据源中定义了检索参数,而 Retrieve ( )函数根本未提供参数时,程序运行后系统会自动显示类似于图 1.4 所示的 Specify Retrieval Arguments 对话框,在这个对话框中用户可以输入检索参数;如果 Retrieve ( )函数提供了参数,但参数个数少于所需个数时,系统将显示出错信息,没有任何数据检索到数据窗口中;如果 Retrieve ( )函数提供了参数,但参数个数多于所需个数时,系统将忽略多出的参数。

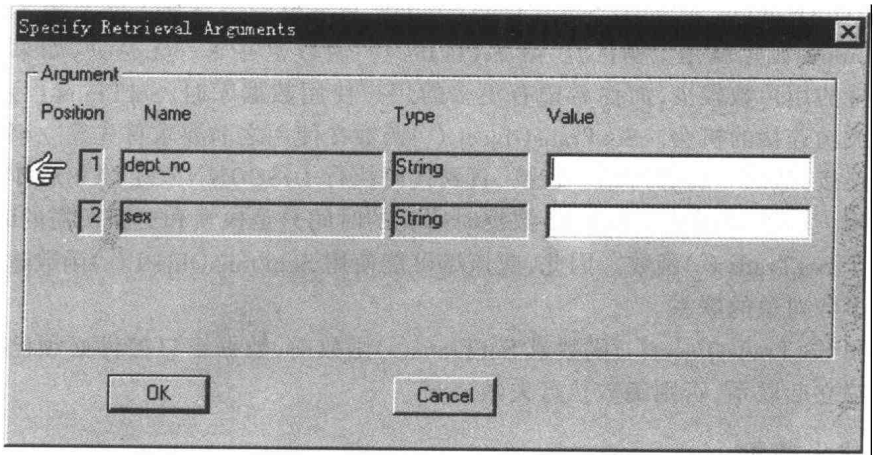


图 1.4 Specify Retrieval Arguments 对话框

2. 检索数据时发生的事件

执行 Retrieve ( )函数时触发下列的数据窗口控件事件。

(1) DBError:检索失败时触发。

(2) RetrieveRow:每检索一行数据触发一次,利用该事件可向用户提供当前已检索数据条数的信息,并可在检索过程中终止检索,但对该事件编程后,将会显著增加检索时间,严重影响程序的运行效率。该事件的返回值为:

1) 0——继续检索(缺省值);

2) 1——停止检索。

(3) RetrieveStart:检索开始时触发,此时数据库尚未检索数据,应用程序可以控制是