POWERSOFT
ENTERPRISE
SERIES

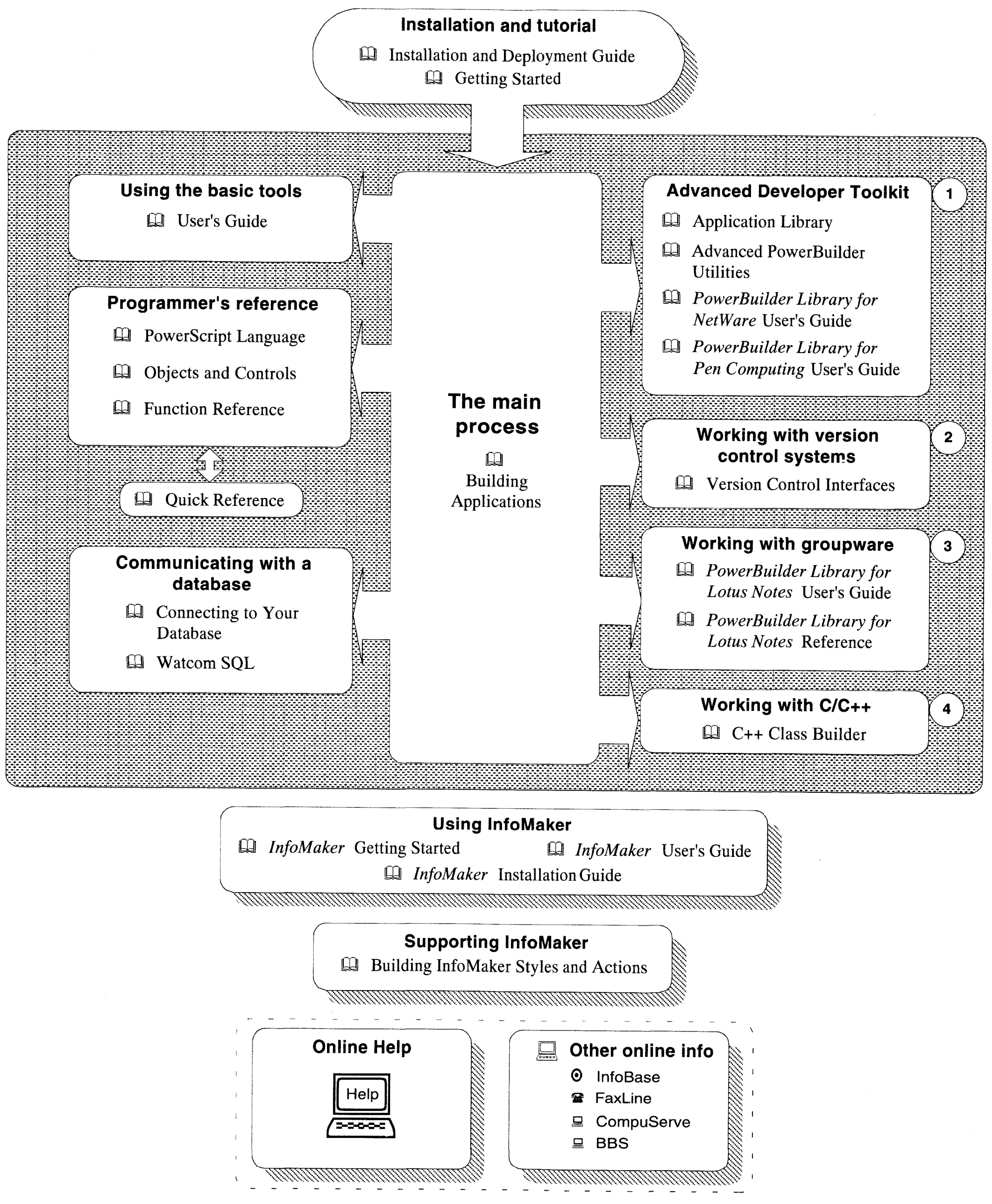
Getting Started

VERSION 4.0

PowerBuilder

December 1994

Roadmap to PowerBuilder Documentation


Installation and tutorial
Installation and Deployment Guide
Getting Started
Using the basic tools
User's Guide
Programmer's reference
PowerScript Language
Objects and Controls
Function Reference
Quick Reference
Communicating with a database
Connecting to Your Database
Watcom SQL
The main process
Building Applications
Advanced Developer Toolkit 1
Application Library
Advanced PowerBuilder Utilities
PowerBuilder Library for NetWare User's Guide
PowerBuilder Library for Pen Computing User's Guide
Working with version control systems 2
Version Control Interfaces
Working with groupware 3
PowerBuilder Library for Lotus Notes User's Guide
PowerBuilder Library for Lotus Notes Reference
Working with C/C++ 4
C++ Class Builder
Using InfoMaker
InfoMaker Getting Started
InfoMaker User's Guide
InfoMaker Installation Guide
Supporting InfoMaker
Building InfoMaker Styles and Actions
Online Help
Help
Other online info
InfoBase
FaxLine
CompuServe
BBS

1 Included with Enterprise and Team/ODBC; Desktop add-on **2** Included with Enterprise and Team/ODBC
3 Included with Enterprise; Team/ODBC and Desktop add-on **4** Included with Enterprise
No symbol = all PowerBuilder packages

Contents

About This Manual

Subject

This manual provides information you need to start using PowerBuilder:

- Part One describes how to start PowerBuilder, use online Help, and start the code examples
- Part Two is a tutorial in which you build your first PowerBuilder application

Audience

This manual is for use by anyone who will be building applications using PowerBuilder.

Online Help

When you have a question about using PowerBuilder, you can access its extensive online Help system. By accessing Help you can see:

- **Procedures** for accomplishing tasks in PowerBuilder
- **Reference information** about PowerBuilder topics or components
- **Context-sensitive information** about PowerBuilder functions or reserved words in scripts

For more information on getting Help in PowerBuilder, see Part One, "Welcome to PowerBuilder."

PowerBuilder documentation

Topic	Manual	Description
Installation and tutorial	*Installation and Deployment Guide*	Provides instructions for installing PowerBuilder; also includes setup information you need to deploy PowerBuilder applications on user computers
	Getting Started	Introduces you to PowerBuilder and provides a tutorial you can step through to learn the basics
The main process	*Building Applications*	Guides you through the process of using PowerBuilder to develop and maintain applications (highlights: project planning, basic and advanced techniques)
Using the basic tools	*User's Guide*	Tells how to use the painters to build objects in PowerBuilder
Programmer's reference	*PowerScript Language*	Gives an overview of PowerScript; also describes the language's data types and statements, as well as SQL statements that you can use in PowerScript
	Objects and Controls	Lists attributes, events, and related functions for PowerBuilder objects and controls
	Function Reference	Describes the PowerScript functions and the DataWindow painter functions; also describes filters, validation rules, display formats, text patterns, and DataWindow attributes
	Quick Reference	Lists PowerScript functions and DataWindow painter functions by category, showing syntax for each; also lists standard and enumerated data types

Topic	Manual	Description
Communicating with a database	*Connecting to Your Database*	Tells how to connect to a database from PowerBuilder (or InfoMaker); tells how to set up, define, and manage connections to ODBC data sources and Powersoft database interfaces
	Watcom SQL	Describes the Watcom SQL database and how to use it with PowerBuilder (or InfoMaker)
Advanced Developer Toolkit	*Application Library*	Describes the PowerBuilder Application Library, a collection of PowerBuilder objects that enable you to accelerate the process of building applications
	Advanced PowerBuilder Utilities	Describes the Advanced PowerBuilder Utilities, a series of tools that extend the PowerBuilder development environment
	PowerBuilder Library for NetWare *User's Guide*	Describes the windows, DataWindows, and functions used when developing applications with the PowerBuilder Library for NetWare
	PowerBuilder Library for Pen Computing *User's Guide*	Describes the windows, user objects, and functions used when developing pen-based applications with the PowerBuilder Library for Pen Computing
Working with version control systems	*Version Control Interfaces*	Tells how to use third-party version control systems to manage PowerBuilder objects

Topic	Manual	Description
Working with groupware	PowerBuilder Library for Lotus Notes *User's Guide*	Tells how to use the PowerBuilder Library for Lotus Notes to develop PowerBuilder applications that access Lotus Notes databases and provide mail services through the Vendor Independent Messaging (VIM) protocol
	PowerBuilder Library for Lotus Notes *Reference*	Describes the objects in the libraries provided with the PowerBuilder Library for Lotus Notes
Working with C/C++	*C++ Class Builder*	Tells how to use the C++ Class Builder to define C++ classes, compile them, store them in DLLs, and then execute their methods from PowerBuilder applications
Supporting InfoMaker	*Building InfoMaker Styles and Actions*	Tells how to use PowerBuilder to build InfoMaker form styles, which include layouts for displaying data and actions for processing data

PART ONE

WELCOME TO POWERBUILDER

PowerBuilder is a graphic PC-based client/server application development environment. Using PowerBuilder, you can develop object-based windows database applications without coding in C or C++.

This part provides the information you need to get up and running. It describes:

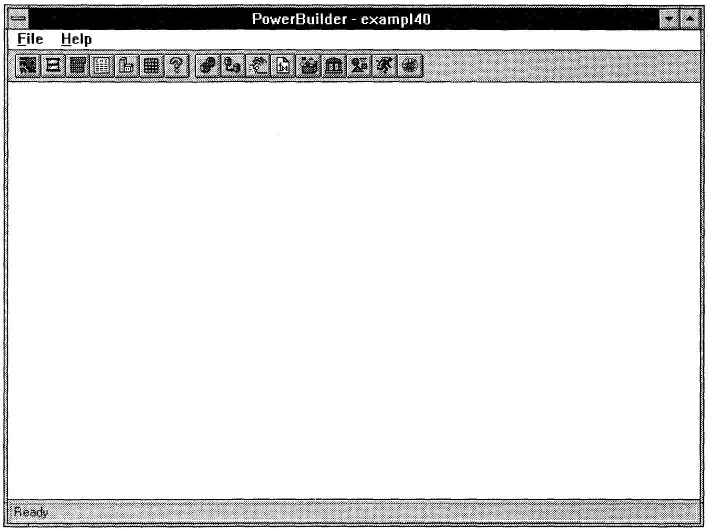
- Starting PowerBuilder
- Using online Help
- Using the code examples

Starting PowerBuilder

Before you start PowerBuilder, read the installation notes for this release to note any last-minute items.

❖ To start PowerBuilder:

♦ Double-click the PowerBuilder icon in the appropriate Windows Program Manager group. The PowerBuilder initial screen displays. If you installed the code examples, exampl40 (the code examples application name) displays in the title bar.



You are now ready to use PowerBuilder to build an application.

For databases other than Watcom SQL
If you want to connect to a database other than Watcom SQL, see the discussion on changing the database in *Connecting to Your Database.*

Starting PowerBuilder from the command line

You can start PowerBuilder from the command line or the Program Manager and optionally open one of the following painters (or the File editor):

Application painter	Menu painter
Database painter	Query painter
Data Pipeline painter	Report painter
DataWindow painter	Structure painter
Debug painter	User Object painter
Function painter	Window painter
Library painter	

To do so, use the following syntax:

{win} *directory*\pb040.exe { /P *paintername*}

The use of **win** is required when you start PowerBuilder from the command line.

Switch	Description
/P	Used with the painter name to identify the painter to access

Parameter	Description
directory	The fully qualified name of the directory containing PowerBuilder.
paintername	The name of the painter you want to open. The default is the window that displays when you begin a new PowerBuilder session. To open the File editor, set *paintername* to DO or DOSEditor. The painter name must uniquely identify the painter. You do not have to enter the entire name. For example, you can enter **w** to open the Window painter and **dataw** to open the DataWindow painter. If you enter the full name, omit any spaces in the name (enter **UserObject** and **DataPipeline**).

Examples

Enter this command at the command line to start PowerBuilder and open the Window painter:

```
win pb4\pb040.exe /P w
```

Enter this command in the Windows Program Manager Run dialog box to start PowerBuilder and open the DataWindow painter:

```
pb4\pb040.exe /P dataw
```

Using online Help

PowerBuilder has extensive online Help. It uses the Microsoft Windows Help facility.

Online Help supplements the information in the PowerBuilder manuals and provides step-by-step directions for using the painters and all the PowerBuilder features.

❖ To display context-sensitive Help for a dialog box:

- ♦ Click the Help button in a dialog box.

  PowerBuilder displays information about the dialog box. From there you can move to other topics.

❖ To display the main PowerBuilder online Help window:

- ♦ Click the Help button on the PowerBar.
  or
  Select Help➤Help Index from the menu bar.
  or
  Press F1.

  PowerBuilder displays the PowerBuilder Help contents window. From there you can move to other topics.

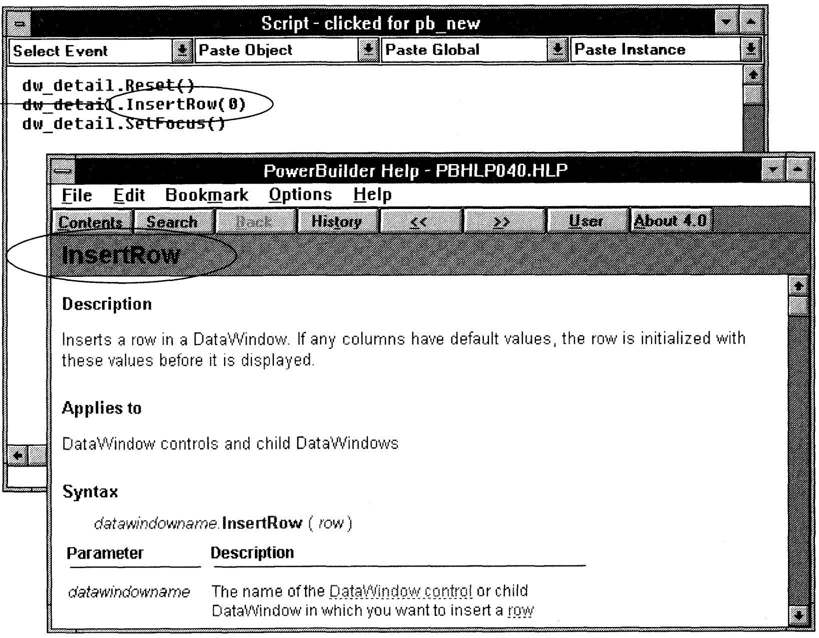
❖ To search for a topic within Help:

- ♦ Select Help➤Search for Help on from the menu bar.

  PowerBuilder displays the Search dialog box. You use this dialog box to type the item for which you want help and to select the appropriate help topic.

❖ To access online Help for a function or keyword in a script:

Select the function call or PowerScript keyword and press SHIFT+F1


Script - clicked for pb_new
Select Event
Paste Object
Paste Global
Paste Instance
PowerBuilder Help - PBHLP040.HLP
File Edit Bookmark Options Help
Contents Search Back History << >> User About 4.0

```
dw_detail.Reset()
dw_detail.InsertRow(0)
dw_detail.SetFocus()
```

InsertRow

Description

Inserts a row in a DataWindow. If any columns have default values, the row is initialized with these values before it is displayed.

Applies to

DataWindow controls and child DataWindows

Syntax

datawindowname.**InsertRow** (*row*)

Parameter	Description
datawindowname	The name of the DataWindow control or child DataWindow in which you want to insert a row

Using the code examples

The PowerBuilder 4.0 code examples demonstrate the features of PowerBuilder. The code examples use the Powersoft Demo DB database. If you did not install the Powersoft Demo DB database, you will not be able to run all the code examples.

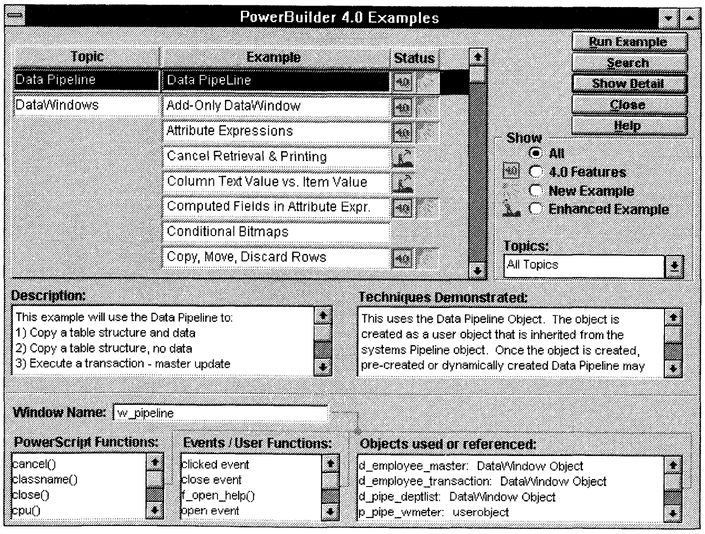
> **Use the sample applications too**
> In addition to the code examples discussed here, PowerBuilder also includes a number of *sample applications*. By using and reviewing these applications, you can learn many useful concepts and techniques.

❖ To start the code examples:

1 Make sure the EXAMPL40 application is the current application.

2 Click the Run button in the PowerBar.

The PowerBuilder 4.0 Code Examples window displays:



Topic	Example	Status
Data Pipeline	Data PipeLine	
DataWindows	Add-Only DataWindow	
	Attribute Expressions	
	Cancel Retrieval & Printing	
	Column Text Value vs. Item Value	
	Computed Fields in Attribute Expr.	
	Conditional Bitmaps	
	Copy, Move, Discard Rows	

Show: [✓ All] 4.0 Features New Example Enhanced Example

Topics: All Topics

Description:
This example will use the Data Pipeline to:
1) Copy a table structure and data
2) Copy a table structure, no data
3) Execute a transaction - master update

Techniques Demonstrated:
This uses the Data Pipeline Object. The object is created as a user object that is inherited from the systems Pipeline object. Once the object is created, pre-created or dynamically created Data Pipeline may

Window Name: w_pipeline

PowerScript Functions:
cancel()
classname()
close()
cpu()

Events / User Functions:
clicked event
close event
f_open_help()
open event

Objects used or referenced:
d_employee_master: DataWindow Object
d_employee_transaction: DataWindow Object
d_pipe_deptlist: DataWindow Object
p_pipe_wmeter: userobject

From this main window, you can do many things:

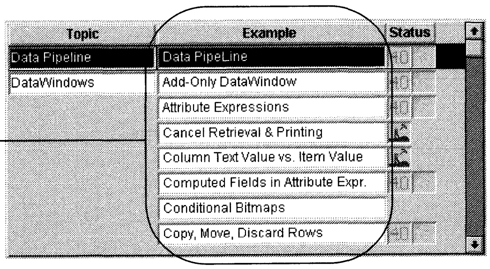
- Scroll through the list of code examples.
- Run a code example.

- Display online Help to get more information on using the code examples.
- Display a report showing detail information for the example.
- Restrict example display by status.
- Restrict example display by topic.
- Restrict example display by event, function, or object.

Examining the scripts and objects behind each example
To get the most out of the PowerBuilder code examples, you should examine each example's scripts, functions, and objects (such as windows and DataWindows). Part Two, "PowerBuilder Tutorial," will show you how to use PowerBuilder painters to display windows, DataWindows, and menus.

❖ To scroll through the list of code examples:

- Click on a line in the Example list.
  or
  Scroll using PAGE UP, PAGE DOWN, UP ARROW, and DOWN ARROW.



Topic	Example	Status
Data Pipeline	Data PipeLine	
DataWindows	Add-Only DataWindow	
	Attribute Expressions	
	Cancel Retrieval & Printing	
	Column Text Value vs. Item Value	
	Computed Fields in Attribute Expr.	
	Conditional Bitmaps	
	Copy, Move, Discard Rows	

The information in the bottom half of the window updates as you scroll. It includes:

- Example description
- Techniques demonstrated by the example
- Window name
- PowerScript functions used in the example
- Events and user-defined functions used in the example
- Objects used by the example

❖ To run a code example:

- ♦ Double-click an example in the list.
  or
  Click Run Example.

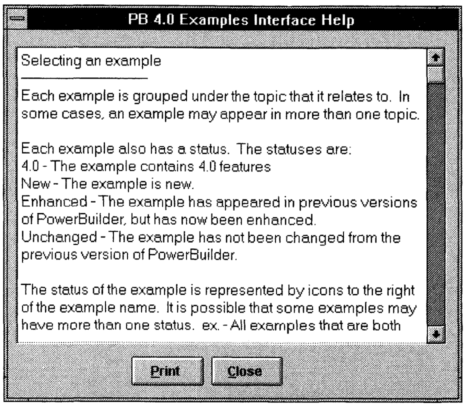
  The selected application executes.

  From within most examples, you can display additional information via online help.

❖ To display online help for the code examples:

- ♦ Click Help.

  The Help dialog box displays.



Selecting an example

Each example is grouped under the topic that it relates to. In some cases, an example may appear in more than one topic.

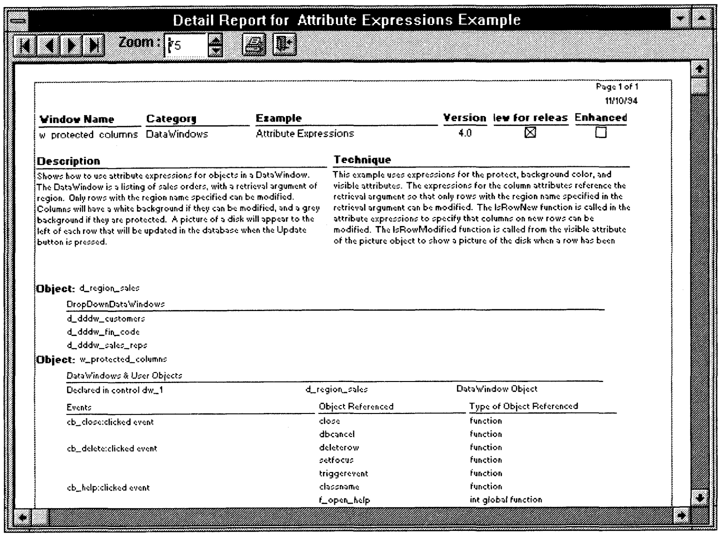
Each example also has a status. The statuses are:
4.0 - The example contains 4.0 features
New - The example is new.
Enhanced - The example has appeared in previous versions of PowerBuilder, but has now been enhanced.
Unchanged - The example has not been changed from the previous version of PowerBuilder.

The status of the example is represented by icons to the right of the example name. It is possible that some examples may have more than one status. ex. - All examples that are both

❖ To display a report showing detail information for a code example:

- ♦ From the main window, select an example and click Show Detail.

A detail report displays. You can zoom and print the report.



Page 1 of 1
11/10/94

Window Name	Category	Example	Version	lew for releas	Enhanced
w protected columns	DataWindows	Attribute Expressions	4.0	☒	☐

Description
Shows how to use attribute expressions for objects in a DataWindow. The DataWindow is a listing of sales orders, with a retrieval argument of region. Only rows with the region name specified can be modified. Columns will have a white background if they can be modified, and a grey background if they are protected. A picture of a disk will appear to the left of each row that will be updated in the database when the Update button is pressed.

Technique
This example uses expressions for the protect, background color, and visible attributes. The expressions for the column attributes reference the retrieval argument so that only rows with the region name specified in the retrieval argument can be modified. The IsRowNew function is called in the attribute expressions to specify that columns on new rows can be modified. The IsRowModified function is called from the visible attribute of the picture object to show a picture of the disk when a row has been

Object: d_region_sales

DropDownDataWindows
d_dddw_customers
d_dddw_fin_code
d_dddw_sales_reps

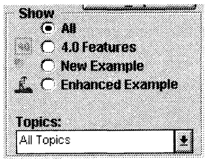
Object: w_protected_columns

DataWindows & User Objects

Declared in control dw_1	d_region_sales	DataWindow Object
Events	**Object Referenced**	**Type of Object Referenced**
cb_close:clicked event	close	function
	dbcancel	function
cb_delete:clicked event	deleterow	function
	setfocus	function
	triggerevent	function
cb_help:clicked event	classname	function
	f_open_help	int global function

❖ To restrict example display by status:

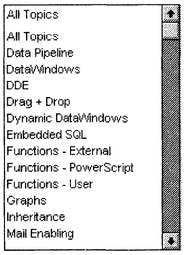
♦ Click the appropriate radio button in the Show box.



❖ To restrict example display by topic:

1 Click the down arrow in the Topics dropdown listbox.

A list of topics displays.



2 Click the topic you want to use to restrict display. You can also select multiple topics (using SHIFT+click for two or more topics in sequence and CTRL+click for two or more topics out of sequence).

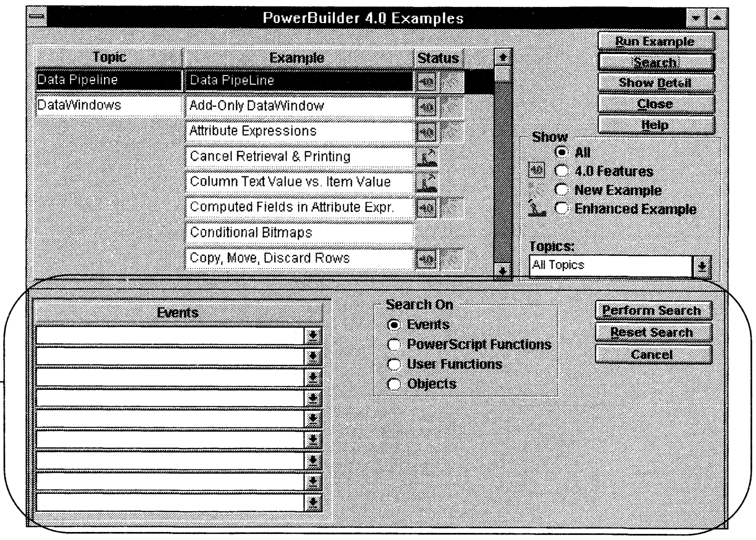
The main window displays examples for the selected topics only.

❖ To restrict example display by event, function, or object:

1 Click the Search button.

The bottom half of the window now contains a series of dropdown listboxes that you can use to show only those examples that demonstrate a particular event, PowerScript function, user-defined function, or object.

Use the bottom half of the window to restrict display by event, function, or object



PowerBuilder 4.0 Examples

Topic	Example
Data Pipeline	Data PipeLine
DataWindows	Add-Only DataWindow
	Attribute Expressions
	Cancel Retrieval & Printing
	Column Text Value vs. Item Value
	Computed Fields in Attribute Expr.
	Conditional Bitmaps
	Copy, Move, Discard Rows

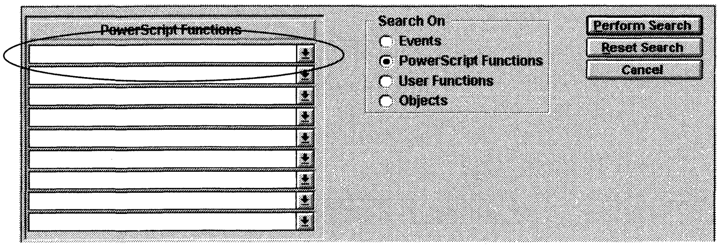
Show: [✓ All] 4.0 Features / New Example / Enhanced Example

Topics: All Topics

Search On: [✓ Events] PowerScript Functions / User Functions / Objects

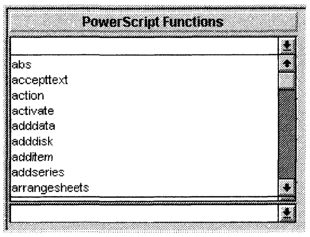
2 Click the appropriate radio button to indicate whether to search for events, PowerScript functions, user-defined functions, or objects.

This example searches for PowerScript functions. The title of the left-hand column changes to reflect your selection.

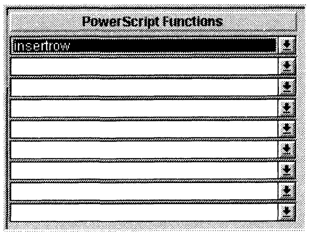


3 Click the down arrow in the PowerScript Functions dropdown listbox.

A list displays naming all PowerScript functions used in the code examples.



4 Click on the function you want.

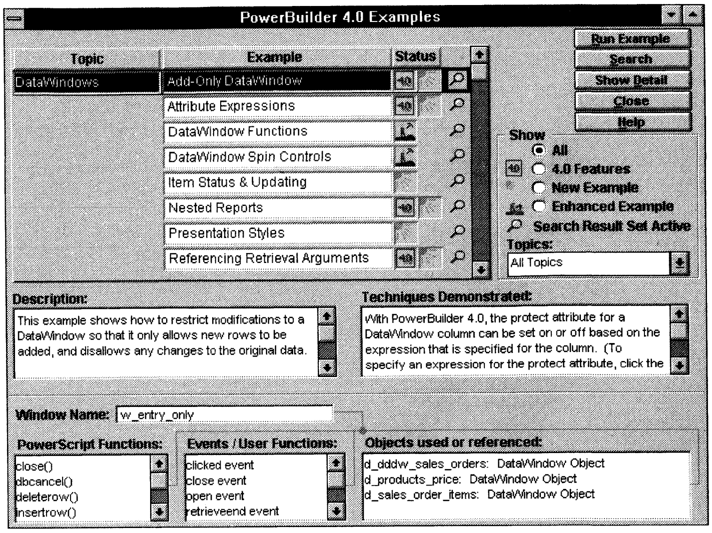


5 Optionally enter additional items to search on. Note the following:

- ♦ If you make multiple selections, examples using any of the selected items are displayed
- ♦ You can only select items of the type specified in the Search On box (events, functions, user-defined functions, or objects)

6 Click the Perform Search button.

The main window redisplays showing all examples that use the selected PowerScript function. The examples also display a magnifying glass icon to indicate that a search result set is active.



Topic	Example
DataWindows	Add-Only DataWindow
	Attribute Expressions
	DataWindow Functions
	DataWindow Spin Controls
	Item Status & Updating
	Nested Reports
	Presentation Styles
	Referencing Retrieval Arguments

Show
[✓ All]
4.0 Features
New Example
Enhanced Example
Search Result Set Active

Topics: All Topics

Description:
This example shows how to restrict modifications to a DataWindow so that it only allows new rows to be added, and disallows any changes to the original data.

Techniques Demonstrated:
With PowerBuilder 4.0, the protect attribute for a DataWindow column can be set on or off based on the expression that is specified for the column. (To specify an expression for the protect attribute, click the

Window Name: w_entry_only

PowerScript Functions:
close()
dbcancel()
deleterow()
insertrow()

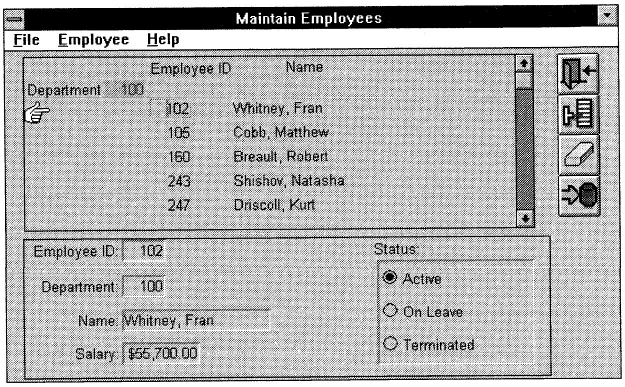
Events / User Functions:
clicked event
close event
open event
retrieveend event

Objects used or referenced:
d_dddw_sales_orders: DataWindow Object
d_products_price: DataWindow Object
d_sales_order_items: DataWindow Object

PART TWO

POWERBUILDER TUTORIAL

The PowerBuilder tutorial is a series of 13 lessons in which you build a complete Master/Detail database application. The window that displays when you run the finished application looks like this.



Department 100	Employee ID	Name
	102	Whitney, Fran
	105	Cobb, Matthew
	160	Breault, Robert
	243	Shishov, Natasha
	247	Driscoll, Kurt

Employee ID: 102
Department: 100
Name: Whitney, Fran
Salary: $55,700.00
Status: [✓ Active] On Leave Terminated

The top half of the window contains a list of employees with a pointer to a single employee; the bottom half of the window displays extra detail for the current employee.

What you will do

Lesson 1

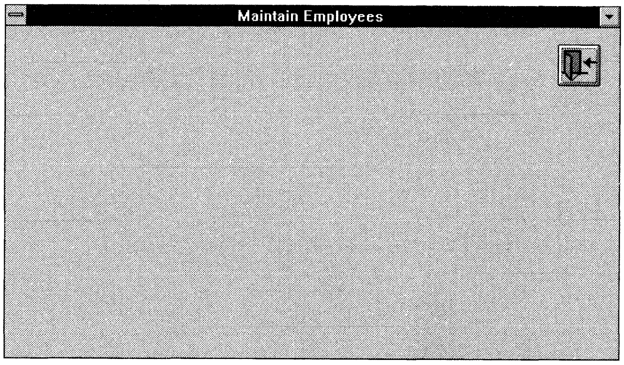
You will begin by using the PowerBuilder Application painter to create the application object and associate an icon with the application.

Lessons 2 and 3

Then you will use the Database painter to create a database and a new table, which contains employee data.

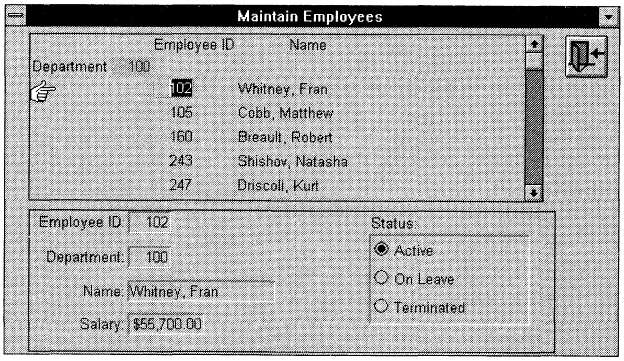
Lessons 4 and 5

Next you will create a window for the application with a single PictureButton.



Lessons 6 through 9

You will then create one DataWindow to list employees and another one to list detail information for a selected employee. You use these to maintain your table.



Department 100

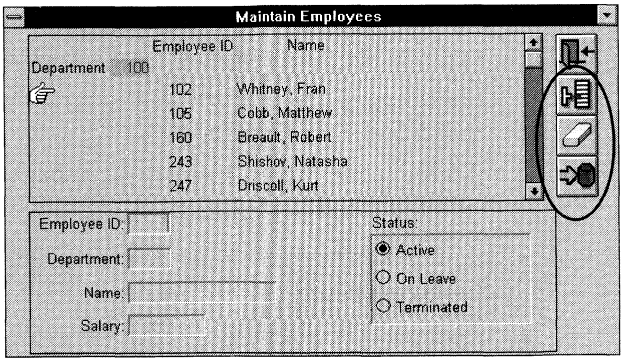
Employee ID	Name
102	Whitney, Fran
105	Cobb, Matthew
160	Breault, Robert
243	Shishov, Natasha
247	Driscoll, Kurt

Employee ID: 102
Department: 100
Name: Whitney, Fran
Salary: $55,700.00

Status
[✓ Active]
On Leave
Terminated

Lesson 10

Then you will add more PictureButtons to enhance the window and scripts to allow users to insert and delete data and update the database.

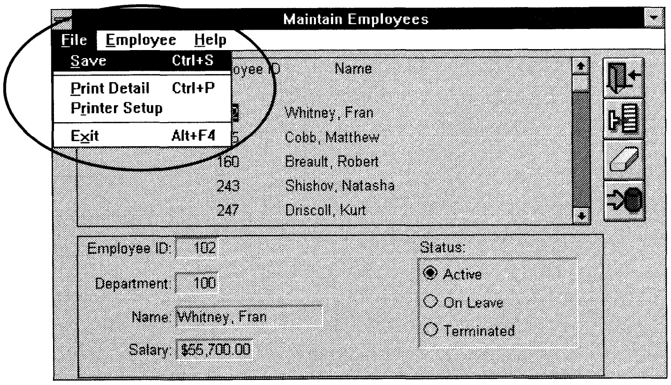


Employee ID	Name
Department 100	
102	Whitney, Fran
105	Cobb, Matthew
160	Breault, Robert
243	Shishov, Natasha
247	Driscoll, Kurt

Employee ID:
Department:
Name:
Salary:
Status: [✓ Active] On Leave Terminated

Lesson 11

Next you will build a menu.



Employee ID	Name
102	Whitney, Fran
105	Cobb, Matthew
160	Breault, Robert
243	Shishov, Natasha
247	Driscoll, Kurt

Employee ID: 102
Department: 100
Name: Whitney, Fran
Salary: $55,700.00
Status: [✓ Active] On Leave Terminated

Lesson 12

To see an example of inheritance, you will use this window to create another one.

Lesson 13

Finally, to complete the tutorial, you will create an EXE file that you or a user can run from Windows.

How long it will take

You can do the entire tutorial in one sitting in about four to five hours. Or you can stop after any lesson and continue at another time.

What you will learn

You will learn basic PowerBuilder techniques and concepts, including:

- How to use the Database painter to perform database definition, extended attribute definition, and data manipulation
- How to use the Application painter to define an application object and application-level scripts
- How to use the Window painter to create PictureButton controls, DataWindow controls, window-level scripts, and control-level scripts
- How to use the DataWindow painter to define selection, grouping, and display options
- How to use the PowerScript painter to define scripts for applications, windows, window controls, and menus
- How to use the Menu painter to define menus, menu items, accelerators, and shortcut keys

After you finish the tutorial, read the Appendix to review the tutor_pb application, its components, and the events in which you coded PowerScript statements.

This tutorial will not make you an expert in PowerBuilder. Only experience building real-world applications can do that. It will, however, give you hands-on experience and provide a foundation for continued growth.

After you finish this tutorial you should continue learning about PowerBuilder by reading the *Building Applications* manual.

Setting up

Before you start the tutorial, make sure that the files you need for the tutorial are on your hard disk, preferably in the same directory as PowerBuilder.

The tutorial lessons use the following files:

EMPTUT.SQL	SQL statements to provide data for a table
TUTOR_PB.HLP	Online help file
TUTORIAL.ICO	An icon
TUTDEL.BMP	A bitmap
TUTEXIT.BMP	A bitmap
TUTNEW.BMP	A bitmap
TUTSAVE.BMP	A bitmap

When you install PowerBuilder, these files are installed in the PowerBuilder directory. When you have finished with the tutorial you can delete them if you want.

LESSON 1

Creating the Application Object

The first step in building a PowerBuilder application is to create an application object. You are always developing within the scope of an application object. In this lesson you will:

- Create the tutorial application object
- Specify the library that will hold the application object
- Control PowerBuilder toolbars
- Assign an icon to the application

How long will this lesson take?
About 15 minutes.

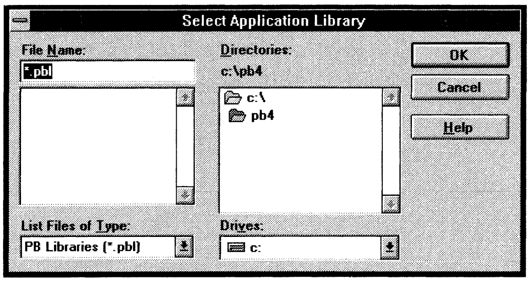
Create and save the application object

Where you are
<u>Lesson 1 Creating the Application Object</u>
➡ Create and save the application object
Control the toolbars
Specify an icon for the application

Now you will create an application object for an Employee Maintenance application. You will not put everything in it now. You will add more to the application object later in the tutorial.

1 **Start PowerBuilder, as described in Part One.**

- **If you have never worked with an application object and did not install the sample application** The Select Application Library dialog box displays.



PowerBuilder requires an application. If PowerBuilder cannot access an Application object you must create one now.

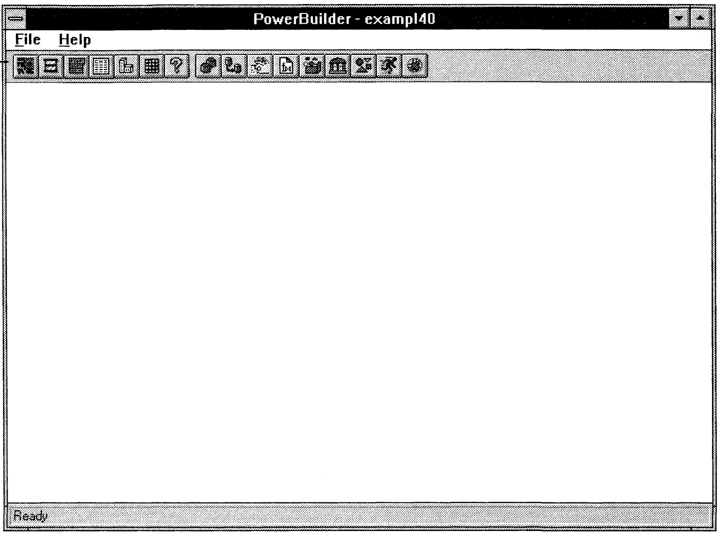
2 **Click Cancel.**

3 **Click New.**

The Select New Application dialog box displays. Proceed with step 4.

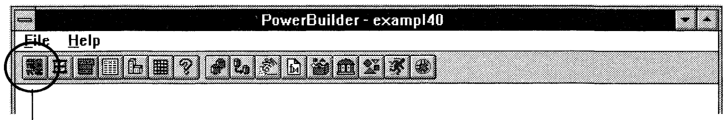
- **If you have previously worked with an application object** The initial window displays. It includes a toolbar called the PowerBar at the top of the window. This toolbar accesses all of the major PowerBuilder painters.

The PowerBar accesses painters



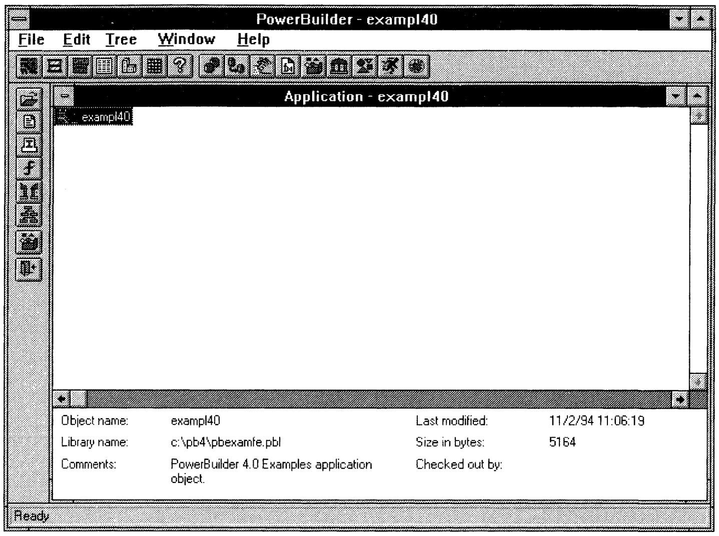
Your toolbar may not show text or may be positioned in some other area of the window. In a few minutes you will learn how to control the display of toolbars. For now, don't worry if your screen does not match the picture.

2 **Click the *Application painter* button in the PowerBar.**



Application painter

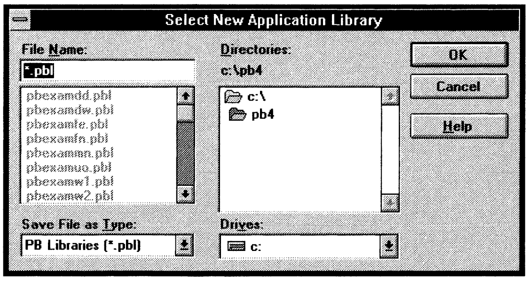
The current application object opens and displays in the workspace.



Object name: exampl40
Library name: c:\pb4\pbexamfe.pbl
Comments: PowerBuilder 4.0 Examples application object.
Last modified: 11/2/94 11:06:19
Size in bytes: 5164
Checked out by:

3 Select File➤New from the menu bar.

The Select New Application Library dialog box displays.

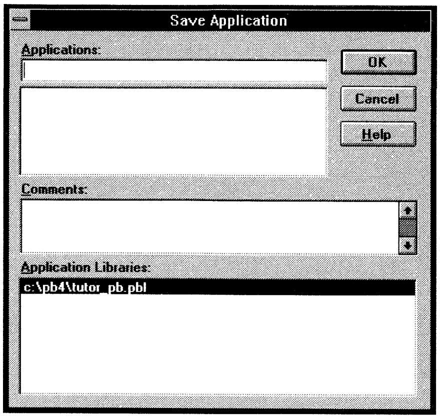


About PowerBuilder libraries
The .PBL extension stands for PowerBuilder library and is pronounced "pibble." All PowerBuilder objects (applications, windows, DataWindows, menus, and user-defined functions and objects) are stored in PowerBuilder libraries.

4 **Type** *tutor_pb.pbl* **in the File Name box.**

5 **Click** *OK.*

The Save Application dialog box displays.

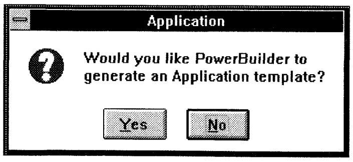


6 **Type** *tutor_pb* **in the Applications box.**
Press the TAB **key twice to move to the Comments box.**
Type *This is the PowerBuilder tutorial application.*

This associates a descriptive comment with the application object. The comment you add here displays in the Library painter and helps to identify objects. Comments are optional.

7 **Click** *OK.*

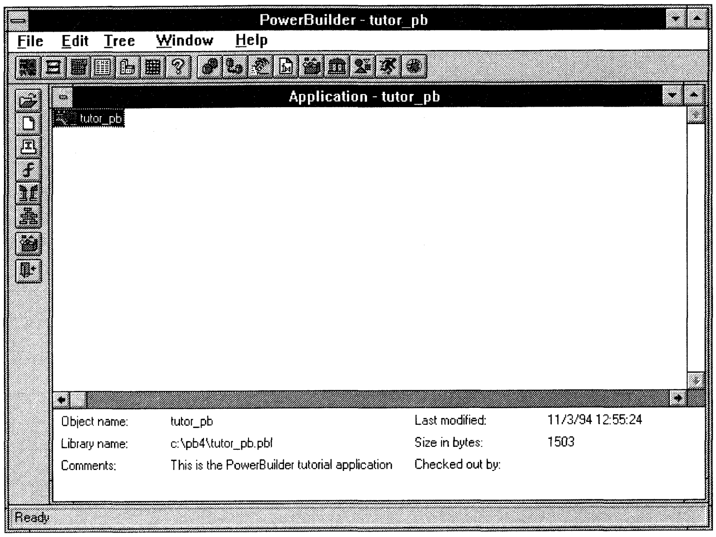
The Application dialog box displays.



Would you like PowerBuilder to generate an Application template?

8 **Click** *No.*

PowerBuilder creates a library named tutor_pb.pbl and creates and saves the application object named tutor_pb in the library. The Application painter workspace displays.



Object name:	tutor_pb	Last modified:	11/3/94 12:55:24
Library name:	c:\pb4\tutor_pb.pbl	Size in bytes:	1503
Comments:	This is the PowerBuilder tutorial application	Checked out by:	

At this point, the tutor_pb application is your current application.

Each time you start up, PowerBuilder opens the current application. Whenever you want, you can change the current application in the Application painter.

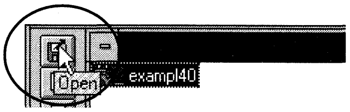
Control the toolbars

Where you are
Lesson 1 Creating the Application Object
Create and save the application object
➡ Control the toolbars
Specify an icon for the application

PowerBuilder has four toolbars: the PowerBar, the PainterBar, the StyleBar, and the ColorBar. You can control whether individual toolbars display and where they display. You can also choose whether to display text in the toolbars.

1 **Move the pointer to any one of the buttons on the PainterBar and pause. Do not click.**

PowerBuilder displays a **PowerTip** identifying the button's function.

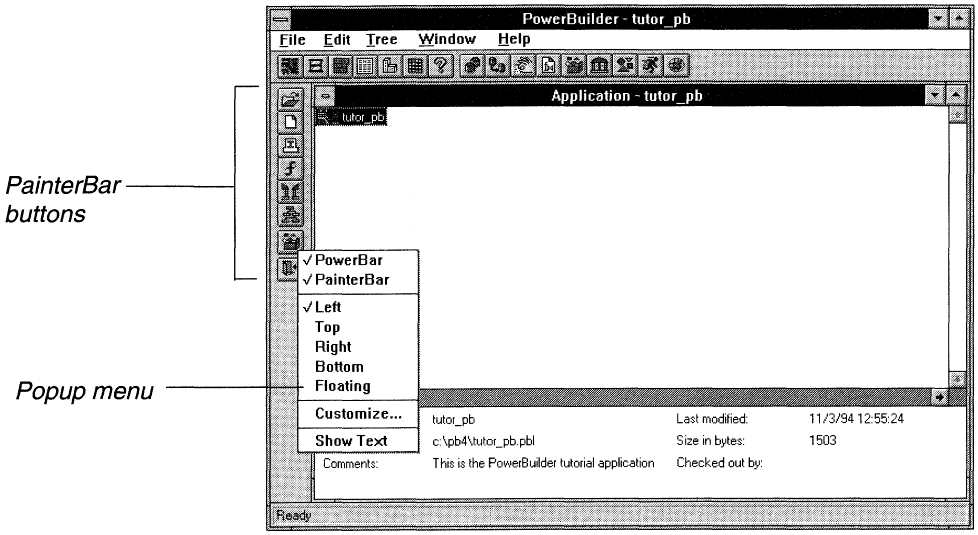


PowerTips are displayed whenever the pointer pauses over a toolbar button.

PowerTips don't display when toolbars show text
If your toolbar buttons show text, PowerBuilder suppresses the display of PowerTips. The following steps tell you how to control text display.

2 Move the pointer to any one of the buttons on the PainterBar and click the right mouse button.

The popup menu for the toolbars displays.



tutor_pb — Last modified: 11/3/94 12:55:24
c:\pb4\tutor_pb.pbl — Size in bytes: 1503
Comments: This is the PowerBuilder tutorial application — Checked out by:

About the popup menu
Throughout PowerBuilder, popup menus provide a fast way to do things. The right mouse button accesses the popup menu. The menu changes depending on the painter you are in and where you are in the workspace when you click the right mouse button.

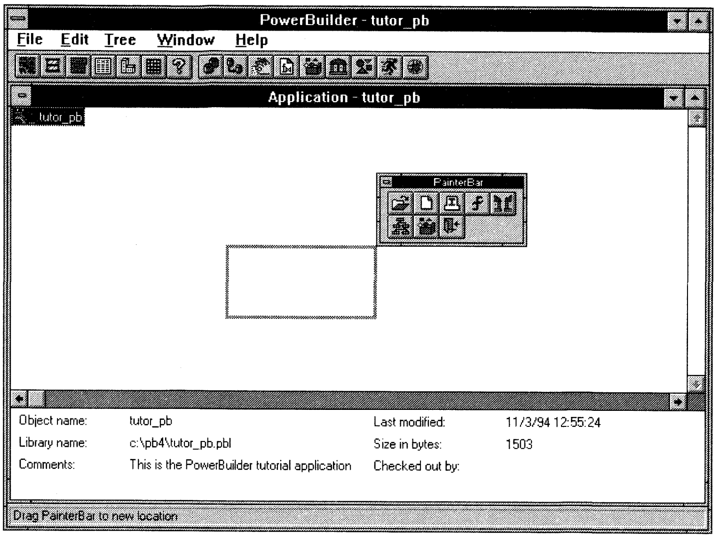
3 Click *Floating* in the popup menu.

The PainterBar changes to a floating toolbar.

4 Move the pointer to an empty area within the floating toolbar and drag the floating toolbar to a convenient position.

How to drag the toolbar
Press and hold the left mouse button. While pressing the button, move the mouse (an outline displays to show the current location of the toolbar).

When the toolbar is where you want it, release the mouse button.



Object name:	tutor_pb	Last modified:	11/3/94 12:55:24
Library name:	c:\pb4\tutor_pb.pbl	Size in bytes:	1503
Comments:	This is the PowerBuilder tutorial application	Checked out by:	

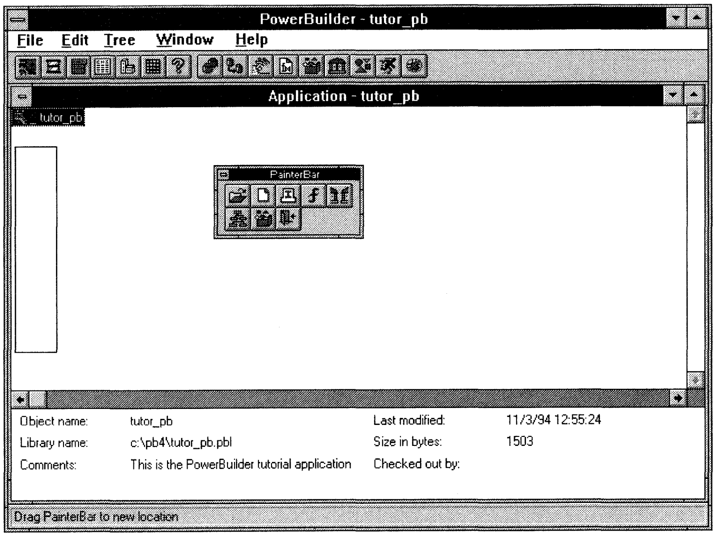
Drag PainterBar to new location

You can put toolbars in many different locations: left, top, right, bottom, and floating.

Some painters have so many buttons that you need to use the floating toolbar to display all buttons. Or you may be able to display all the buttons if you turn off the display of text.

You can also customize toolbars. You can remove buttons, add, or change the order of the buttons.

5 **Drag the floating toolbar toward the left side of the workspace. Release the mouse button when the toolbar becomes a vertical bar.**

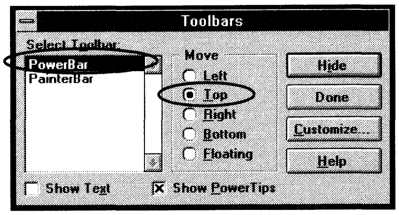


Object name: tutor_pb Last modified: 11/3/94 12:55:24
Library name: c:\pb4\tutor_pb.pbl Size in bytes: 1503
Comments: This is the PowerBuilder tutorial application Checked out by:

Drag PainterBar to new location

6 **Select Window➤Toolbars from the menu bar.**

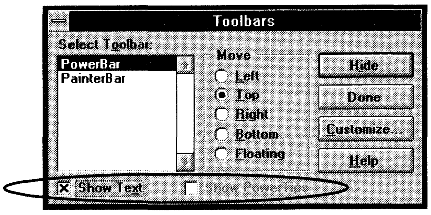
The Toolbars dialog box displays. The position of the currently selected toolbar shows as the selected radio button.



Move: Left, [✓ Top], Right, Bottom, Floating

☐ Show Text ☒ Show PowerTips

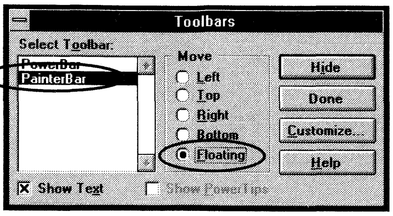
7 **Deselect the *Show PowerTips* checkbox. Select the *Show Text* checkbox.**



Move: Left, [✓ Top], Right, Bottom, Floating

☒ Show Text ☐ Show PowerTips

8 **Click** *PainterBar.*
Click *Floating.*



9 **Click** *Done.*

This makes the PainterBar floating.

Now you know several ways to move the toolbars.

For the rest of this tutorial, you should move the toolbars where you like to have them. Your screen may not match exactly the pictures in this book, depending on where and how you display toolbars. For example, the rest of the tutorial shows toolbar buttons with text displayed; you may decide to work with no text, displaying PowerTips as the need arises.

Specify an icon for the application

Where you are
Lesson 1 Creating the Application Object
Create and save the application object
Control the toolbars
➡ Specify an icon for the application

The icon you specify displays in the Windows workspace when you minimize your application while executing it.

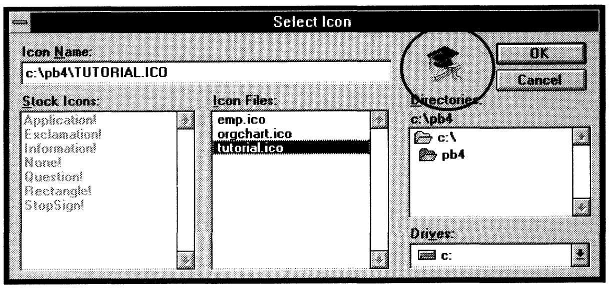
PowerBuilder includes the icon automatically when you create an application EXE file.

1 **Click the *Icon* button in the PainterBar.**

The Select Icon dialog box lists all available icons.

2 **Select *tutorial.ico*.**

The Tutorial icon displays in the dialog box.



Icon Name: c:\pb4\TUTORIAL.ICO

Stock Icons:
Application!
Exclamation!
Information!
None!
Question!
Rectangle!
StopSign!

Icon Files:
emp.ico
orgchart.ico
tutorial.ico

Directories:
c:\pb4
c:\
pb4

Drives: c:

3 **Click *OK*.**

You return to the Application painter workspace.

4 **Select File➤Save from the menu bar.**

This saves the updated application object.

5 **Click the *Return* button on the PainterBar.**

You return to the initial window.

LESSON 2

Creating the Table

Tables are the way relational databases organize information. In many organizations, database specialists maintain the database. If this is true in your organization, you may not be permitted to create and maintain tables within the database. However, to take full advantage of PowerBuilder, you need to know how to work with database tables.

PowerBuilder is installed with a single-user Watcom SQL database. You can create and work with tables in the Watcom SQL database.

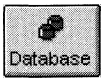
In this lesson you will:

- ♦ Create a new database for this tutorial
- ♦ Create a new table for employees

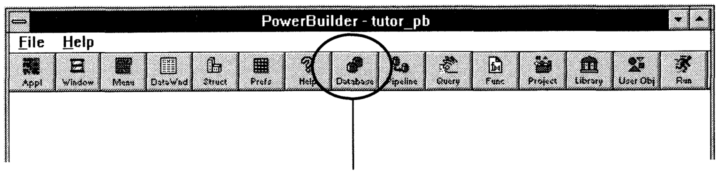
How long will this lesson take?
About 15 minutes.

Open the Database painter

Where you are
Lesson 2 Creating the Table
➡ Open the Database painter
Create a new database
Create the table



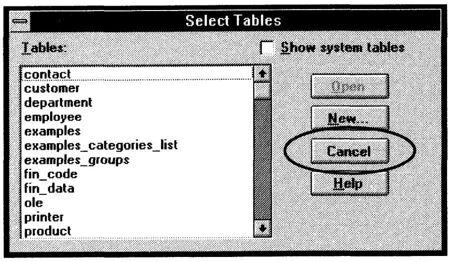
1 **Click the *Database painter* button in the PowerBar.**



Database painter

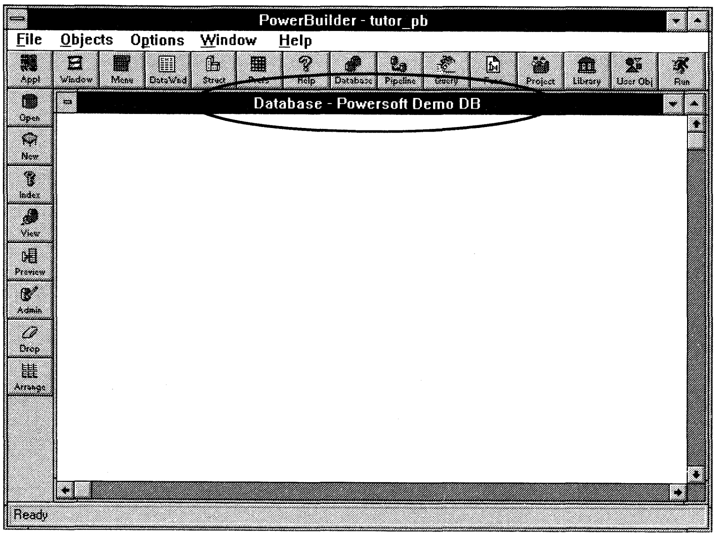
The Select Tables dialog box displays. It lists the tables in the database you are currently connected to and has a New button for creating new tables.

Because you are going to create a new database before you create your table, you will cancel out of this dialog box for now.



2 **Click *Cancel*.**

The Select Tables dialog box closes. You are in the Database painter. The workspace is blank. Notice that the workspace title bar identifies the database you are currently connected to.


PowerBuilder - tutor_pb
File Objects Options Window Help
Appl Window Menu DataWnd Struct Help Database Pipeline Query Project Library User Obj Run
Open New Index View Preview Admin Drop Arrange
Database - Powersoft Demo DB
Ready

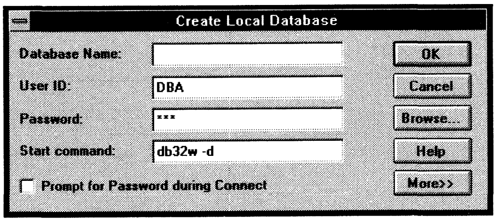
Create a new database

Where you are
Lesson 2 Creating the Table
Open the Database painter
➡ Create a new database
Create the table

For this tutorial, you will create a new database. When you are finished with the tutorial, you can easily delete the database.

1 **Select File➢Create Database from the menu bar.**

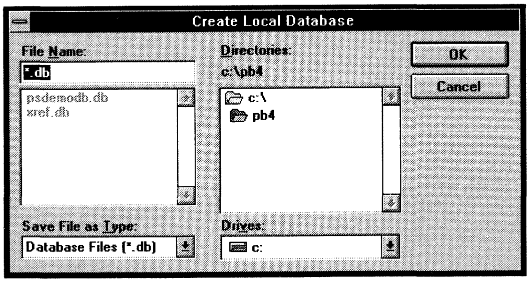
The Create Local Database dialog box displays.



Database Name:
User ID: DBA
Password: ***
Start command: db32w -d
☐ Prompt for Password during Connect

2 **Click** ***Browse.***

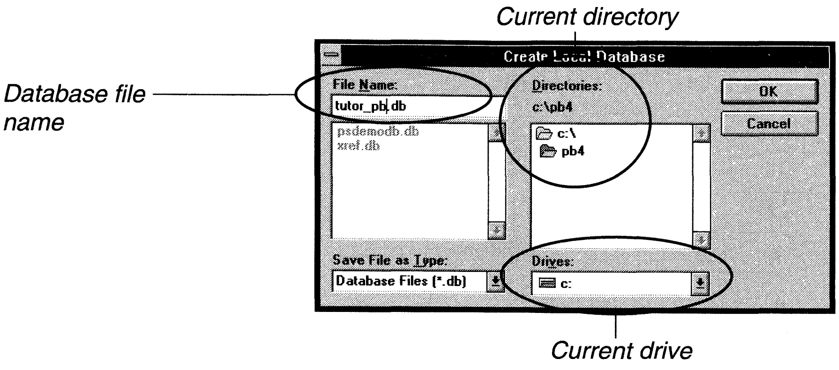
The Create Local Database dialog box displays.



File Name: *.db
psdemodb.db
xref.db
Directories: c:\pb4
c:\
pb4
Save File as Type: Database Files (*.db)
Drives: c:

3 Type *tutor_pb.db* in the File Name box.

If you want to change directory or drive for storing the database file, you can use the Directories or Drives boxes.

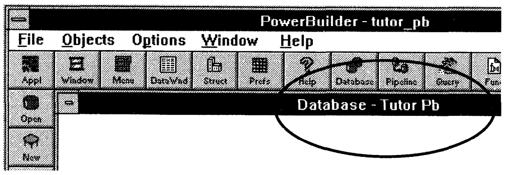


4 Click *OK*.

PowerBuilder redisplays the original Create Local Database dialog box.

5 Click *OK*.

PowerBuilder creates a local Watcom SQL database named Tutor_pb in the current directory and connects to the Tutor_pb database.



PowerBuilder also adds database information to the Windows ODBC.INI file and creates and adds a database profile to your PowerBuilder preferences file (PB.INI). This will allow you to connect to the database easily in the future.

Create the table

Where you are
Lesson 2 Creating the Table
Open the Database painter
Create a new database
➡ Create the table

To create the table, you use the Create Table dialog box. In this dialog box, you:

- Name the table and define its columns
- Define the primary key
- Request table creation

When you finish entering information, the Create Table dialog box will look like this.

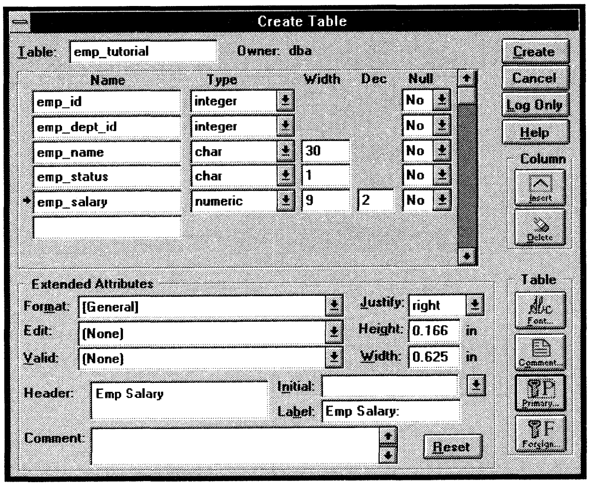

Create Table

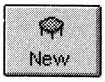
Table: emp_tutorial Owner: dba

Name	Type	Width	Dec	Null
emp_id	integer			No
emp_dept_id	integer			No
emp_name	char	30		No
emp_status	char	1		No
➡ emp_salary	numeric	9	2	No

Extended Attributes
Format: [General]
Edit: (None)
Valid: (None)
Justify: right
Height: 0.166 in
Width: 0.625 in
Header: Emp Salary
Initial:
Label: Emp Salary:
Comment:

When you click the Create button in this dialog box, PowerBuilder generates the SQL statements required to create the table. Then PowerBuilder submits the statements to the Watcom SQL database management system (DBMS), which creates the table.

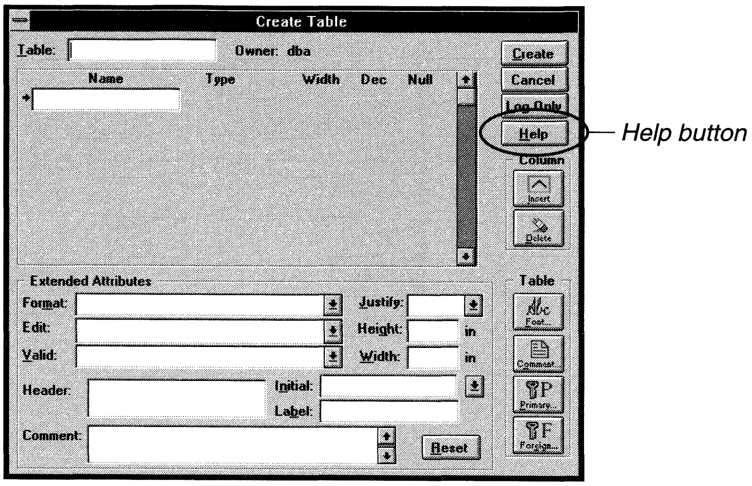
Display the Create Table dialog box



1 Click the *New* button on the PainterBar.

The Create Table dialog box displays. In the top part of this dialog box, you define the columns in the table. In the bottom part of the dialog box, you define extended attributes.

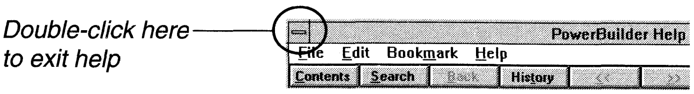
The dialog box also has a Help button, which displays Help for the dialog box.



2 Click the *Help* button.

Help for the Create Table dialog box displays. The online Help system is comprehensive. As you do the tutorial, use the Help button whenever you want to find out more about a dialog box.

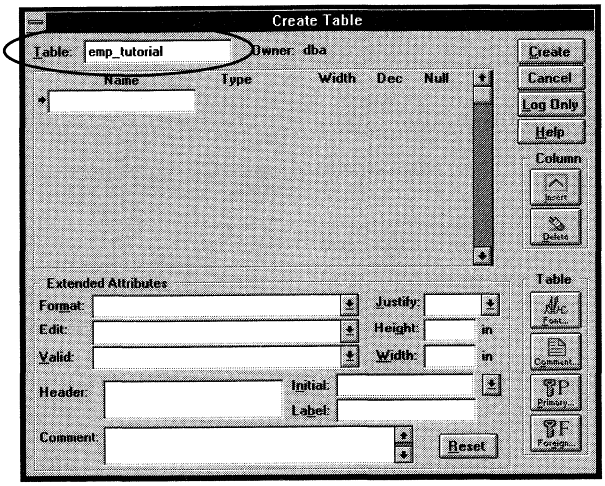
3 Double-click the Control-menu box to close the Help window.



The Help window closes and you return to the Create Table dialog box.

Name the table

1 **Click in the Table box to reposition the insertion point.**
Type *emp_tutorial*



2 **Press** TAB.

The insertion point moves to the Name box for the first column.

Define the columns in the table

1 **Type *emp_id* and press TAB.**

The insertion point moves to the Type box. The default data type (char) is highlighted. The data type for the emp_id column needs to be integer.

> **About default data types**
> Char is the default data type for the first column. The default data type for subsequent columns is the type of the previous column.

2 **Click the down arrow to the right of the word char and select *integer* from the menu of available data types (scroll the list if necessary).**

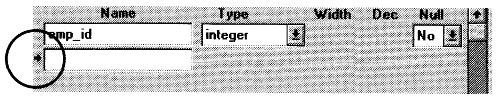
This changes the data type for emp_id to integer.

> **Another way to select a data type**
> You can also select a data type by typing the first letter of the data type in the Type column. For example, to select integer, type ***I***.

3 **Press TAB twice.**

This moves you past the setting of No for the column labeled Null. No means you do not want to allow the column you are defining to be empty (to have a Null value). Since you always want a value for emp_id, you will leave the value No.

The arrow pointer now points to the line for the next column.



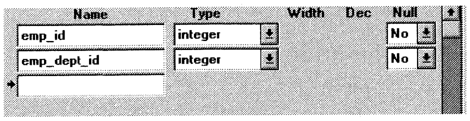
Name	Type	Width	Dec	Null
emp_id	integer			No

4 **Type *emp_dept_id***

The Type and Null values default to the values in the previous column (integer and No). These are correct, so you can leave them unchanged.

5 Press the DOWN ARROW key.

The arrow pointer moves down so you can define the next column.



Name	Type	Width	Dec	Null
emp_id	integer			No
emp_dept_id	integer			No

6 Finish entering information about the table columns.

The following table shows all the values needed to define the columns for the emp_tutorial table. You have already entered values for the first two columns (emp_id and emp_dept_id).

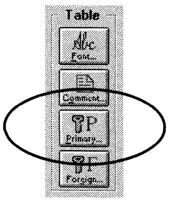
Name	Type	Width	Dec	Null
emp_id	integer			No
emp_dept_id	integer			No
emp_name	char	30		No
emp_status	char	1		No
emp_salary	numeric	9	2	No

Now you have provided the information required to define the emp_tutorial table. Next you will define a primary key for the table.

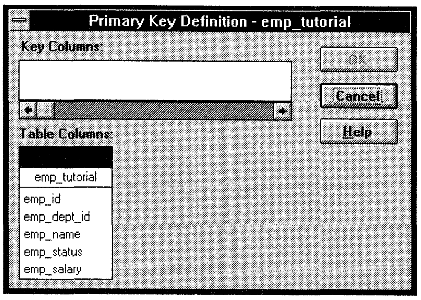
Define the primary key

To use a database table in a DataWindow, the table must have a primary key. The primary key uniquely identifies each row. In the emp_tutorial table, emp_id is the unique identifier of each row of data.

1 **Click the *Primary Key* button in the lower-right side of the Create Table window.**



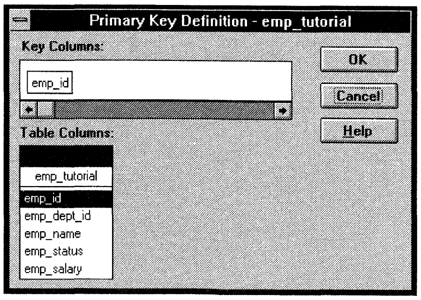
The Primary Key Definition dialog box displays.



You use this window to pick one or more columns as the primary key.

2 **Click *emp_id* in the list of columns.**

Emp_id displays in the Key Columns box.



3 Click ***OK.***

This defines emp_id as the primary key for your table. When you request table creation, PowerBuilder will include the SQL statement required to define a primary key.

Request table creation

1 **Make sure your Create Table dialog box looks like this. Click the *Create* button.**

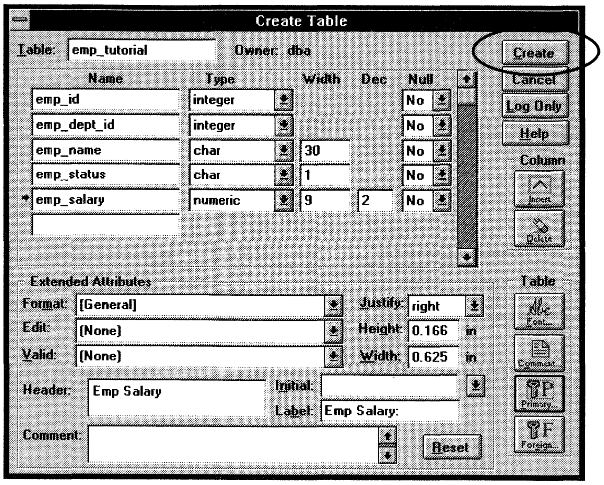


Table: emp_tutorial Owner: dba

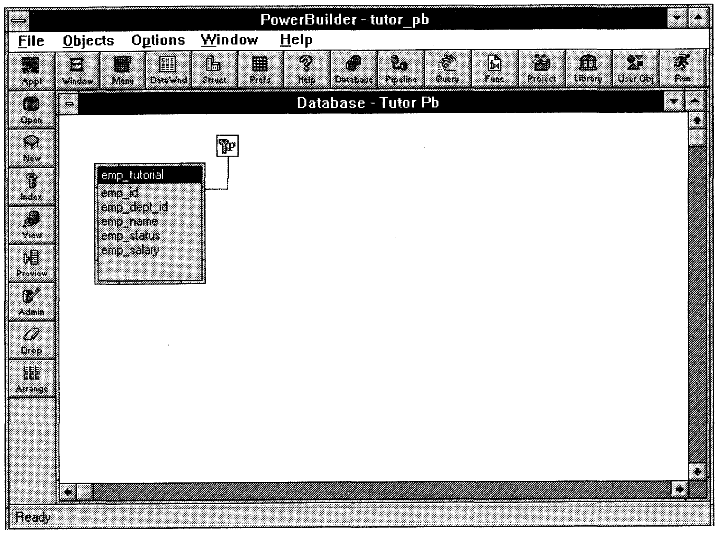
Name	Type	Width	Dec	Null
emp_id	integer			No
emp_dept_id	integer			No
emp_name	char	30		No
emp_status	char	1		No
emp_salary	numeric	9	2	No

Extended Attributes

Format:	[General]	Justify:	right	
Edit:	(None)	Height:	0.166	in
Valid:	(None)	Width:	0.625	in
Header:	Emp Salary	Initial:		
		Label:	Emp Salary:	
Comment:				

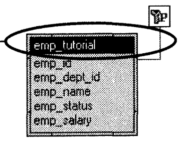
PowerBuilder generates the SQL statements required to create your new table and sends them to the Watcom SQL DBMS.

The DBMS creates the new table. The Database painter displays a representation of the new table in the workspace. The symbol coming from emp_id indicates that emp_id is the primary key for the table.

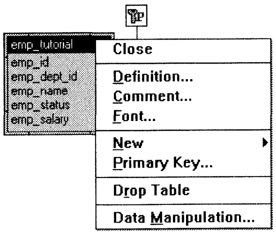

PowerBuilder - tutor_pb
File Objects Options Window Help
Appl Window Menu DataWnd Struct Prefs Help Database Pipeline Query Func Project Library User Obj Run
Open New Index View Preview Admin Drop Arrange
Database - Tutor Pb
emp_tutorial
emp_id
emp_dept_id
emp_name
emp_status
emp_salary
Ready

2 **Move the pointer to the table name (emp_tutorial) and click the right mouse button.**

Right-click the table name



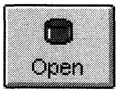
The popup menu for the table displays.



3 Click *Close*.

The Database painter closes the table.

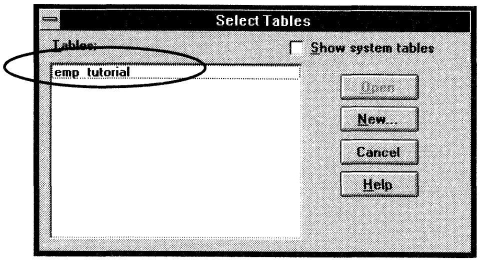
> **Removed from the workspace only**
> When you close a table, the Database painter just removes it from the workspace display. It does not delete it.



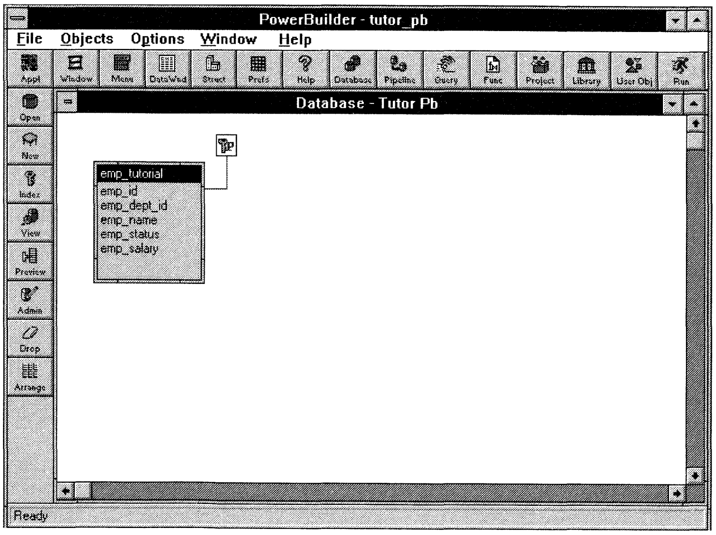
4 Click the *Open* button on the PainterBar.

The Select Tables dialog box displays.

5 Double-click *emp_tutorial*.



The Database painter reopens the table.


PowerBuilder - tutor_pb
File Objects Options Window Help
Appl Window Menu DataWnd Struct Prefs Help Database Pipeline Query Func Project Library User Obj Run
Open New Index View Preview Admin Drop Arrange
Database - Tutor Pb
emp_tutorial
emp_id
emp_dept_id
emp_name
emp_status
emp_salary
Ready

Now you know how to open and close tables in the Database painter. In the next lesson you will define extended attribute information and data for the columns in the emp_tutorial table.

Keep going
Because Lesson 3 builds upon what you've just learned about the Database painter, it's best to go directly to Lesson 3. If you must stop, be sure to review the techniques and concepts learned in Lesson 2 before beginning Lesson 3.

LESSON 3

Adding Extended Attribute Information and Data

Once you have created a table, you can provide information about its data. By specifying extended attribute information, you can determine how data is displayed and validated. Then you can add data to the table.

In this lesson you will:

- ♦ Add extended attribute information for the table
- ♦ Add data to the table using a SQL file
- ♦ Review the results of your work

How long will this lesson take?
About 30 minutes.

Add extended attribute information

Where you are
Lesson 3 Adding Extended Attribute Information and Data
➡ Add extended attribute information
Add data to the table
Examine the table
Add a new row

PowerBuilder stores extended attribute information in the database in a collection of system tables called the repository. PowerBuilder uses this information in a DataWindow to determine how data is displayed and how user-entered data is validated. Extended attributes include:

- ♦ Headers and labels for columns
- ♦ Initial values for columns
- ♦ Validation rules
- ♦ Display formats

When you create a DataWindow, PowerBuilder uses the extended attribute information as the default values for the DataWindow. In the DataWindow painter, you can override these values.

When you are working in a multiuser environment, defining this information in the Database painter offers two important advantages: time savings and consistency. Anytime anyone uses the data, they can access the repository information.

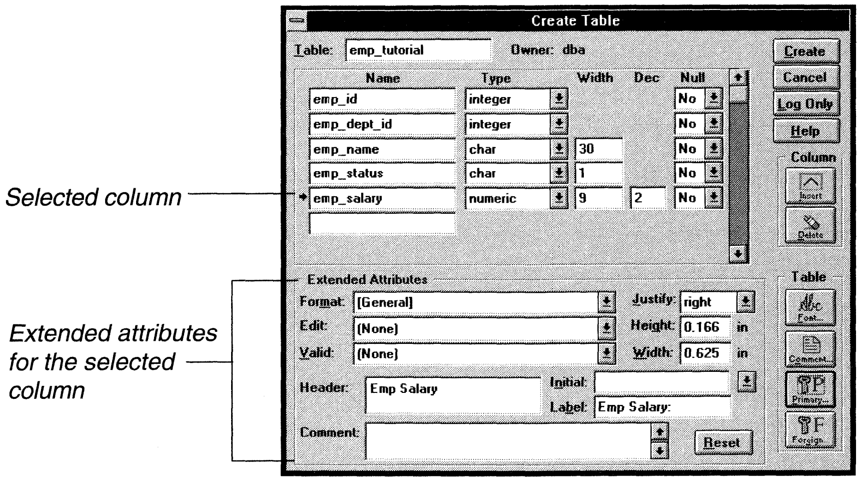


Table: emp_tutorial Owner: dba

Name	Type	Width	Dec	Null
emp_id	integer			No
emp_dept_id	integer			No
emp_name	char	30		No
emp_status	char	1		No
emp_salary	numeric	9	2	No

Extended Attributes
Format: [General]
Edit: (None)
Valid: (None)
Justify: right
Height: 0.166 in
Width: 0.625 in
Header: Emp Salary
Initial:
Label: Emp Salary:
Comment:

This table summarizes how you will extend the column definitions of the emp_tutorial table columns.

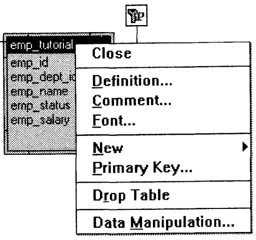
Column name	What you will do to extend its definition
emp_id	Modify header and label information
emp_dept_id	Modify header and label information
emp_name	Modify header and label information
emp_status	Modify header and label information Define an initial value
emp_salary	Modify header and label information Define a validation rule Define a display format

You specify extended attribute information in the bottom of the Create/Alter Table dialog box.

Define headers and labels for columns

1 **Move the pointer to the table name (emp_tutorial) and right-click.**

Right-click emp_tutorial



The popup menu for the table displays.

2 **Click** ***Definition.***

The Alter Table dialog box displays with the arrow pointer identifying the selected column emp_id.

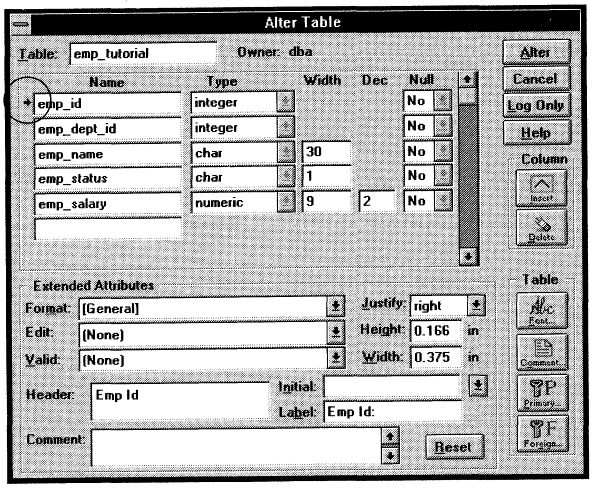


Table: emp_tutorial Owner: dba

Name	Type	Width	Dec	Null
emp_id	integer			No
emp_dept_id	integer			No
emp_name	char	30		No
emp_status	char	1		No
emp_salary	numeric	9	2	No

Extended Attributes

Format: [General] Justify: right

Edit: (None) Height: 0.166 in

Valid: (None) Width: 0.375 in

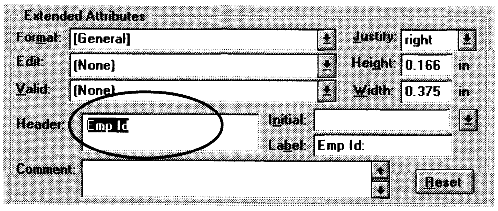
Header: Emp Id Initial:

Label: Emp Id:

Comment:

3 Highlight *Emp Id* in the Header box in the bottom of the dialog box.

This selects Emp Id, which is a default header created from the column name.



Extended Attributes

Format: [General] Justify: right

Edit: [None] Height: 0.166 in

Valid: [None] Width: 0.375 in

Header: Emp Id Initial:

Label: Emp Id:

Comment:

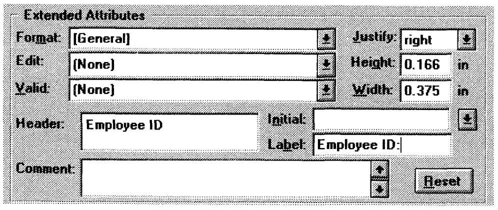
4 Type *Employee ID*

Two-line headers
You can create two-line headers. To do this, press CTRL+ENTER at the end of the first line and then type the second line.

5 Press TAB twice to move to the Label box.

Again, the default value is Emp Id and it is highlighted.

6 Type *Employee ID:*



Extended Attributes

Format: [General] Justify: right

Edit: [None] Height: 0.166 in

Valid: [None] Width: 0.375 in

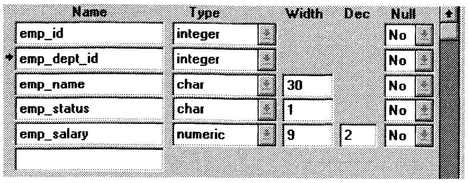
Header: Employee ID Initial:

Label: Employee ID:

Comment:

7 Move the pointer to *emp_dept_id* in the Name box and click.

The arrow pointer identifying the selected column moves to emp_dept_id.



Name	Type	Width	Dec	Null
emp_id	integer			No
➔ emp_dept_id	integer			No
emp_name	char	30		No
emp_status	char	1		No
emp_salary	numeric	9	2	No

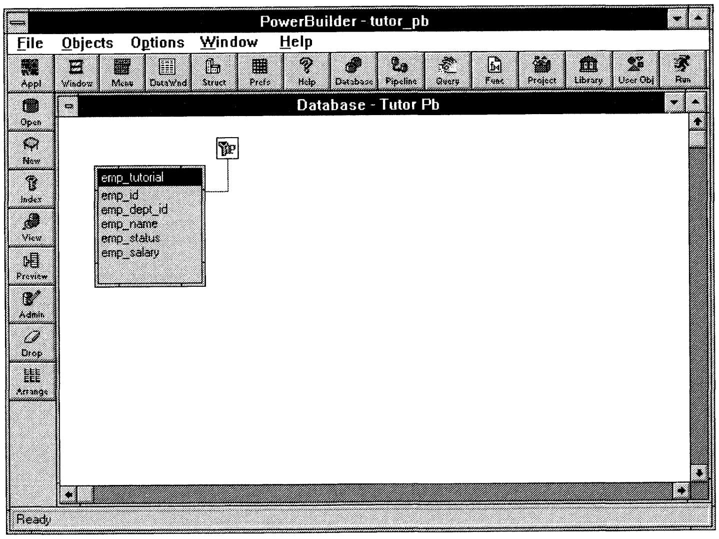
8 Finish entering header and label information for the emp_tutorial columns.

The following table shows all Header and Label values for the emp_tutorial table. You have already entered values for the first column (emp_id).

Name	Header	Label
emp_id	Employee ID	Employee ID:
emp_dept_id	Department	Department:
emp_name	Name	Name:
emp_status	Status	Status:
emp_salary	Salary	Salary:

9 Click the *Alter* button to process the changes.

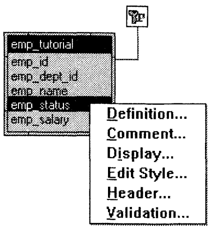
PowerBuilder changes the header and label information in the repository. You return to the Database painter workspace.


PowerBuilder - tutor_pb
File Objects Options Window Help
Appl Window Menu DataWnd Struct Prefs Help Database Pipeline Query Func Project Library User Obj Run
Open New Index View Preview Admin Drop Arrange
Database - Tutor Pb
emp_tutorial
emp_id
emp_dept_id
emp_name
emp_status
emp_salary
Ready

Define an initial value for emp_status

1 **Move the pointer to the emp_status column and right-click.**

Right-click emp_status



The popup menu for emp_status displays.

2 **Click** ***Definition.***

The Alter Table dialog box displays with the arrow pointer at the selected column emp_status.

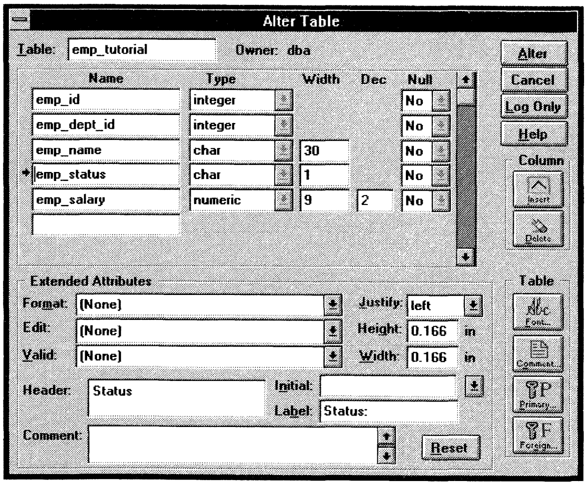


Table: emp_tutorial Owner: dba

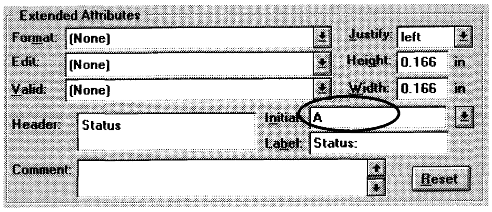
Name	Type	Width	Dec	Null
emp_id	integer			No
emp_dept_id	integer			No
emp_name	char	30		No
emp_status	char	1		No
emp_salary	numeric	9	2	No

Extended Attributes

Format: (None) Justify: left
Edit: (None) Height: 0.166 in
Valid: (None) Width: 0.166 in
Header: Status Initial:
Label: Status:
Comment:

3 Click the *Initial* box.

4 Type *A*



5 Click the *Alter* button to process the changes.

PowerBuilder changes the initial value information in the repository. You return to the Database painter workspace.

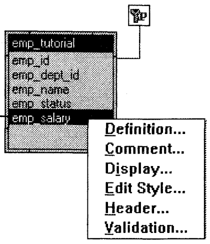
Whenever you create a DataWindow that includes the emp_status column, the default initial value for the column will be A (for Active).

In any DataWindow, you can change the initial value or even eliminate it. You specify extended attribute information to provide default values for a newly created DataWindow.

Define a validation rule for emp_salary

1 **Move the pointer to the *emp_salary* column and right-click.**

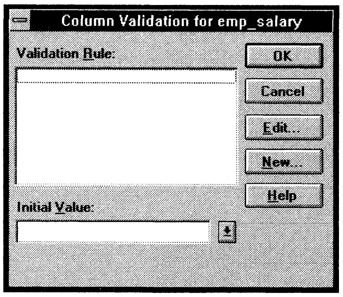
Right-click emp_salary



The popup menu for emp_salary displays.

2 **Click *Validation*.**

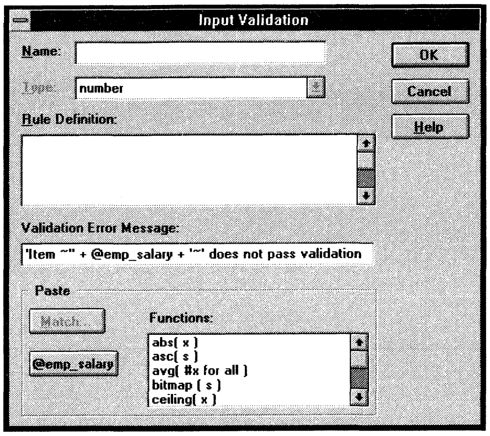
The Column Validation for emp_salary dialog box displays. Use this dialog box to assign a validation rule (and initial value) to a specific column.



You use the Edit and New buttons to edit and create validation rules. You will create a new one now.

3 Click the *New* button.

The Input Validation dialog box displays with the insertion point in the Name box. This dialog box is for creating and maintaining validation rules.



Type: number

Validation Error Message: 'Item ~"' + @emp_salary + '~" does not pass validation

```
abs( x )
asc( s )
avg( #x for all )
bitmap ( s )
ceiling( x )
```

4 Type *ValidSalary* in the Name box.

This assigns the name ValidSalary to the new validation rule.

5 Press TAB.

The insertion point advances to the Rule Definition box.

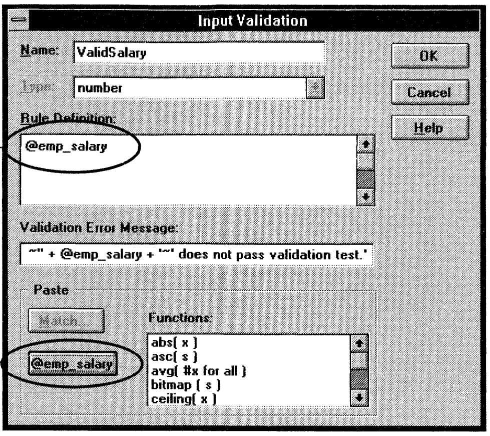
Specifying validation rules
You can specify rules by typing the entire rule on the keyboard or clicking buttons in the dialog box. This example uses both techniques.

6 **Click the *@emp_salary* button in the Paste box in the bottom-left corner of the dialog box.**

This pastes the @emp_salary placeholder into the Rule Definition box.

It is pasted here

Click here



Name: ValidSalary

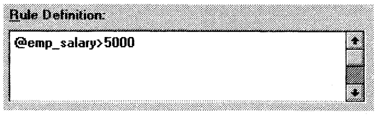
Type: number

Rule Definition: @emp_salary

Validation Error Message: "~"" + @emp_salary + '"~' does not pass validation test.'

7 **Click at the end of the pasted entry *@emp_salary* to position the insertion point.**
Type *>5000*

Your validation rule should look like this now:



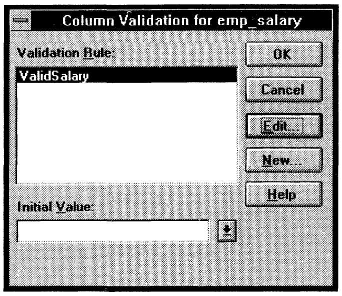
Rule Definition:

@emp_salary>5000

8 **Click *OK*.**

If the expression is not correct, an error message displays. Correct the error and click OK again.

PowerBuilder adds the validation rule to the repository and redisplays the Column Validation for emp_salary dialog box. The rule ValidSalary is highlighted. You want to associate this rule with the emp_salary column.

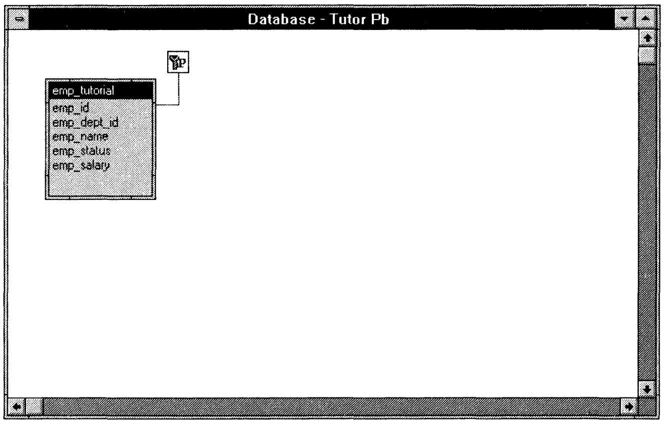


9 Click ***OK.***

This associates the ValidSalary rule with the emp_salary column.

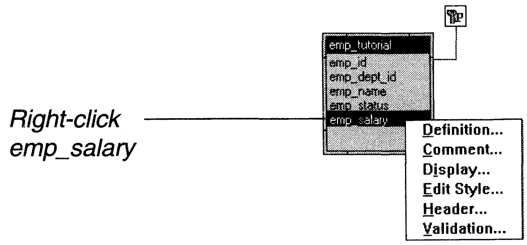
Now that you have the ValidSalary rule in the repository, you can use it for other columns. Once defined, validation rules display in the list of available rules.

You return to the Database painter workspace.



Define a display format for emp_salary

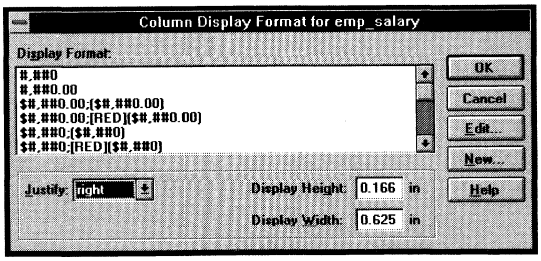
1 **Move the pointer to the *emp_salary* column and right-click.**



The popup menu for emp_salary displays.

2 **Click *Display*.**

The Column Display Format for emp_salary dialog box displays. Use this dialog box to assign a display format to a specific column.


Column Display Format for emp_salary

Display Format:

```
#,##0
#,##0.00
$#,##0.00;($#,##0.00)
$#,##0.00;[RED]($#,##0.00)
$#,##0;($#,##0)
$#,##0;[RED]($#,##0)
```

Justify: right

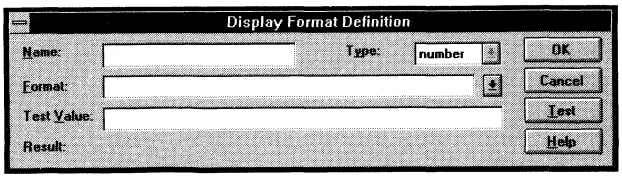
Display Height: 0.166 in

Display Width: 0.625 in

You use the Edit and New buttons to edit and create display formats.

3 **Click the *New* button.**

The Display Format Definition dialog box displays with the insertion point in the Name box. This dialog box is for creating and maintaining display formats.


Display Format Definition

Name:

Type: number

Format:

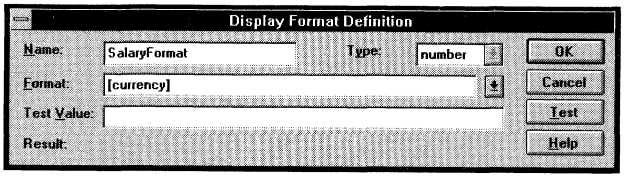
Test Value:

Result:

4 **Type *SalaryFormat* in the Name box.**

This assigns the name SalaryFormat to the new display format.

5 **Press TAB to move to the Format box and type *[currency]***



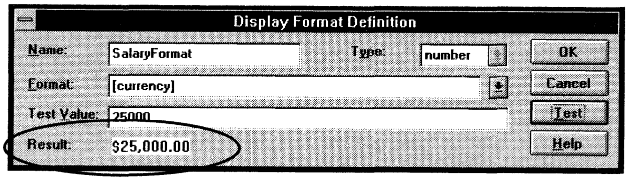
Name: SalaryFormat
Type: number
Format: [currency]
Test Value:
Result:

The value [currency] uses the Windows default currency format to display and print numbers.

6 **Click in the *Test Value* box and type *25000***

7 **Click the *Test* button.**

If the display format is correct and the [currency] format is set for United States currency, the value $25,000.00 will display in the Result display area.

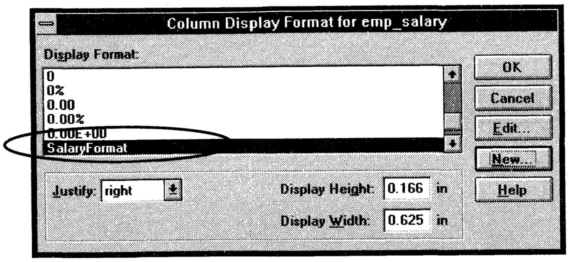


Name: SalaryFormat
Type: number
Format: [currency]
Test Value: 25000
Result: $25,000.00

For countries other than the United States
If you select a country other than the United States in the Windows Control Panel, the value displays in the correct format for that country.

8 Click *OK.*

This adds SalaryFormat to the repository and redisplays the Column Display Format for emp_salary dialog box. SalaryFormat is highlighted.


Column Display Format for emp_salary

Display Format:

```
0
0%
0.00
0.00%
0.00E+00
SalaryFormat
```

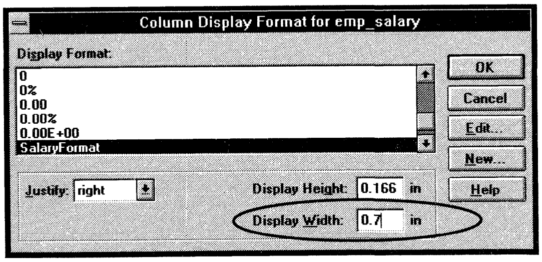
Justify: right

Display Height: 0.166 in

Display Width: 0.625 in

9 Type *0.7* in the Display Width box.

This makes the emp_salary column display a little wider so that the dollar sign isn't truncated.


Column Display Format for emp_salary

Display Format:

```
0
0%
0.00
0.00%
0.00E+00
SalaryFormat
```

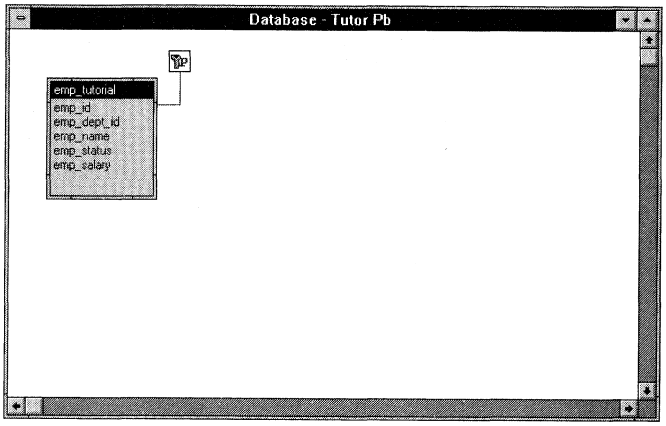
Justify: right

Display Height: 0.166 in

Display Width: 0.7 in

10 Click *OK*.

This assigns SalaryFormat to the emp_salary column and returns you to the Database painter workspace.



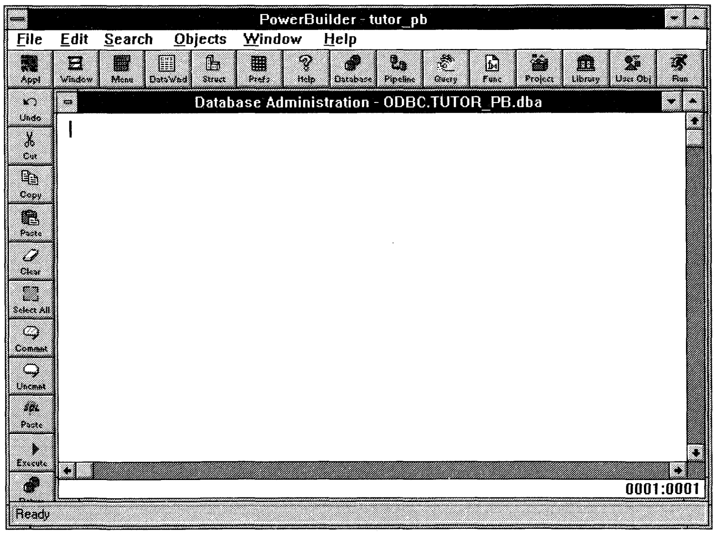
Add data to the table

Where you are
Lesson 3 Adding Extended Attribute Information and Data
Add extended attribute information
➡ Add data to the table
Examine the table
Add a new row

In this section you will add employee data to your new table.

1 Click the *Admin* button on the PainterBar.

The Database Administration window displays.

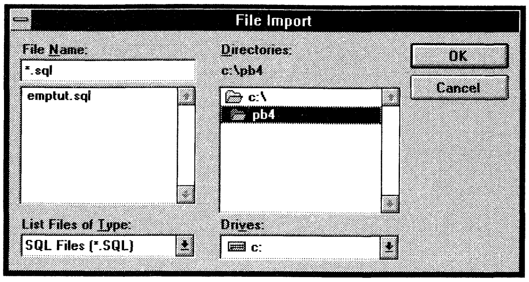

PowerBuilder - tutor_pb
File Edit Search Objects Window Help
Appl Window Menu DataWnd Struct Prefs Help Database Pipeline Query Func Project Library User Obj Run
Undo Cut Copy Paste Clear Select All Comment Uncmnt Paste Execute
Database Administration - ODBC.TUTOR_PB.dba
0001:0001
Ready

This window is a text editor in which you can enter, edit, and execute SQL statements.

To save time, this tutorial provides a file of SQL INSERT statements. You will import this file and execute the statements.

2 Select File➤Import from the menu bar.

The File Import dialog box displays.



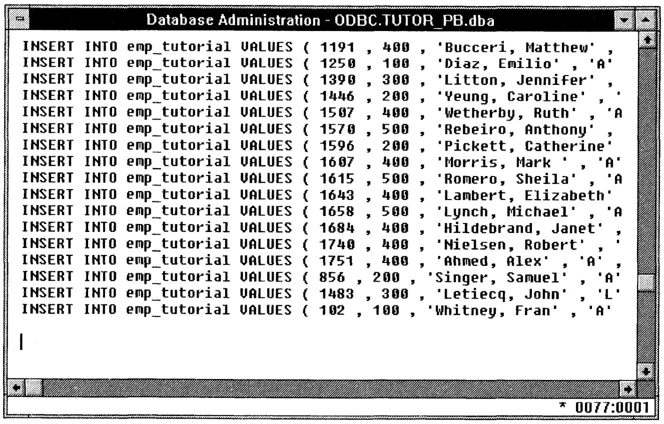
```
File Name:            Directories:
*.sql                 c:\pb4
emptut.sql            c:\
                      pb4
List Files of Type:   Drives:
SQL Files (*.SQL)     c:
```

This dialog box displays a list of SQL files in the current directory. (Your list may look different from this one.)

3 Double-click *emptut.sql* in the list of files.

> **If the emptut.sql file is not listed**
> Change to the directory in which you installed PowerBuilder.

PowerBuilder opens the file and copies its contents into the workspace. The file contains the SQL INSERT statements required to add rows to the emp_tutorial table.



```
INSERT INTO emp_tutorial VALUES ( 1191 , 400 , 'Bucceri, Matthew' ,
INSERT INTO emp_tutorial VALUES ( 1250 , 100 , 'Diaz, Emilio' , 'A'
INSERT INTO emp_tutorial VALUES ( 1390 , 300 , 'Litton, Jennifer' ,
INSERT INTO emp_tutorial VALUES ( 1446 , 200 , 'Yeung, Caroline' , '
INSERT INTO emp_tutorial VALUES ( 1507 , 400 , 'Wetherby, Ruth' , 'A
INSERT INTO emp_tutorial VALUES ( 1570 , 500 , 'Rebeiro, Anthony' ,
INSERT INTO emp_tutorial VALUES ( 1596 , 200 , 'Pickett, Catherine'
INSERT INTO emp_tutorial VALUES ( 1607 , 400 , 'Morris, Mark ' , 'A'
INSERT INTO emp_tutorial VALUES ( 1615 , 500 , 'Romero, Sheila' , 'A
INSERT INTO emp_tutorial VALUES ( 1643 , 400 , 'Lambert, Elizabeth'
INSERT INTO emp_tutorial VALUES ( 1658 , 500 , 'Lynch, Michael' , 'A
INSERT INTO emp_tutorial VALUES ( 1684 , 400 , 'Hildebrand, Janet' ,
INSERT INTO emp_tutorial VALUES ( 1740 , 400 , 'Nielsen, Robert' , '
INSERT INTO emp_tutorial VALUES ( 1751 , 400 , 'Ahmed, Alex' , 'A' ,
INSERT INTO emp_tutorial VALUES ( 856 , 200 , 'Singer, Samuel' , 'A'
INSERT INTO emp_tutorial VALUES ( 1483 , 300 , 'Letiecq, John' , 'L'
INSERT INTO emp_tutorial VALUES ( 102 , 100 , 'Whitney, Fran' , 'A'
```

Execute

4 Click the *Execute* button in the PainterBar.

This sends the SQL INSERT statements to the Watcom SQL DBMS to insert data into your new table.

This happens quickly
The Watcom SQL DBMS processes the SQL INSERT statements quickly. A message displays only if an error occurs. To confirm that the rows were added to the database (and to see an example of an error message), click the Execute button again. A duplicate key message will display because the row was added the first time you clicked Execute. The message box asks if you want to continue. Click the No button.

5 **Click the *Return* button in the PainterBar (not the Database Painter button in the PowerBar).**

The Save Changes dialog box displays, prompting you to save changes to the emptut.sql file.

6 **Click *No* in response to the Save Changes dialog box.**

If you do not see the Return button
Remember that you can change the position of the PainterBar (for how, see "Control the toolbars" in Lesson 1).

This closes the Database Administration window and returns you to the Database painter workspace.

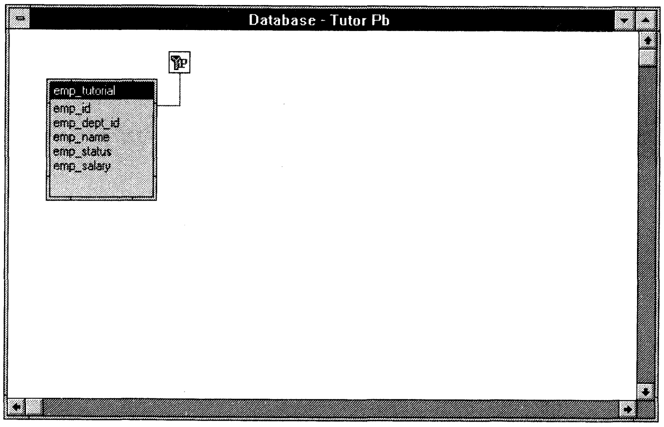
Examine the table

Where you are
Lesson 3 Adding Extended Attribute Information and Data
Add extended attribute information
Add data to the table
➡ Examine the table
Add a new row

You now have a complete table, with repository information and more than seventy employees. You can use the PowerBuilder Data Manipulation facility to display the data in your table and add, change, or delete the rows.

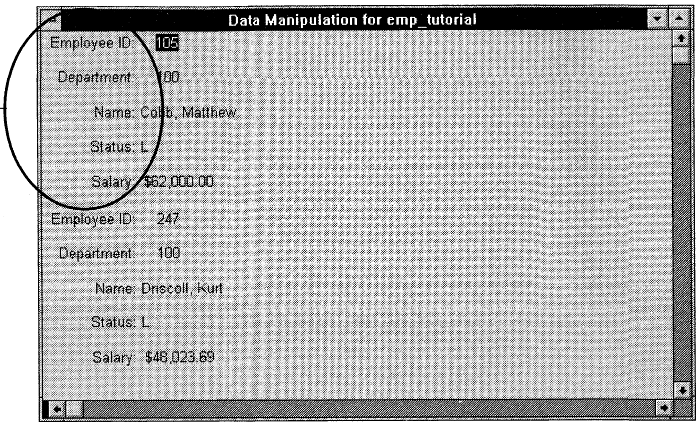
The Data Manipulation window includes a DataWindow that PowerBuilder creates using the columns in the selected table.

1 **Make sure you are still in the Database painter and that the emp_tutorial table is open.**



2 **Select Objects➤Data Manipulation from the menu bar and click** ***FreeForm.***

The Data Manipulation window opens. PowerBuilder retrieves all your employee data and displays as many employees as will fit in the window. Your window may only display one employee at a time.

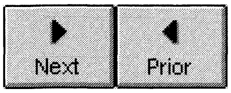

Labels come from the repository

Data Manipulation for emp_tutorial

```
Employee ID: 105
 Department: 100
      Name: Cobb, Matthew
     Status: L
     Salary: $62,000.00
Employee ID: 247
 Department: 100
      Name: Driscoll, Kurt
     Status: L
     Salary: $48,023.69
```

Notice that PowerBuilder uses the column labels from the extended attribute entries you made earlier. Notice that the salary format is currency as you specified in the extended attributes for the emp_salary column. This happens because the Data Manipulation window is really a DataWindow object.

Rows are not retrieved in any particular order
The employee that displays first is not necessarily the employee with the lowest ID. Unless you specify an order, relational databases do not retrieve rows in any particular order.



3 **Click the** ***Next*** **and** ***Prior*** **buttons in the PainterBar to display the data for other employees.**

You can also use the scroll bar on the right of the window to scroll through the data.

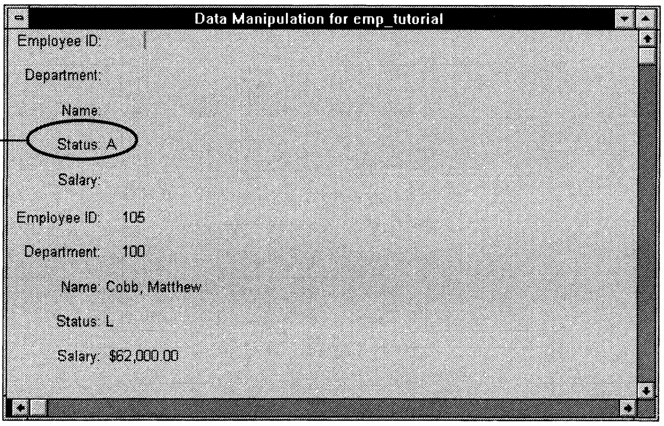
Add a new row

Where you are
<u>Lesson 3 Adding Extended Attribute Information and Data</u>
Add extended attribute information
Add data to the table
Examine the table
➡ Add a new row

Insrt Row

1 **Click the *Insert* button in the PainterBar.**

This inserts a new row. The status field displays **A**, as you specified in the extended attributes. To change the status, you enter a status in the Status column.



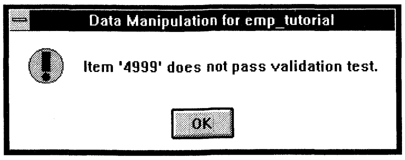
Employee ID:
Department:
Name:
Status: A
Salary:
Employee ID: 105
Department: 100
Name: Cobb, Matthew
Status: L
Salary: $62,000.00

Status has an initial value of A as defined in the repository

2 **Enter data in the *Employee ID*, *Department ID* (use *200*), and *Name* columns.**

3 **Enter a salary that is less than 5000 in the Salary column (don't enter a dollar sign or a comma) and press TAB.**

This value fails the validation test that you specified in extended attributes. PowerBuilder displays an error message.


Data Manipulation for emp_tutorial

Item '4999' does not pass validation test.

Handling errors in your program
You can use the ItemError DataWindow event in your program, to trap this error and display your own error message. Or you can use the DataWindow painter to override the validation rule's default message.

For information about trapping errors and overriding the default message, see *Building Applications* in the PowerBuilder documentation set.

4 **Click the *OK* button in the error message box and enter a higher value for salary.**

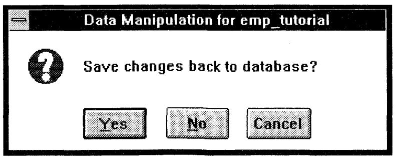
5 **Click the *Updt DB* button in the PainterBar (if you're using PowerTips, the text is Save Changes).**

This updates the database with the new row.

Changing and deleting data for an existing employee
You can also change and delete data for an existing employee. Be sure to click the Updt DB button to update the database with the changes.

6 **Select File➤Close from the menu bar to close the Data Manipulation window.**

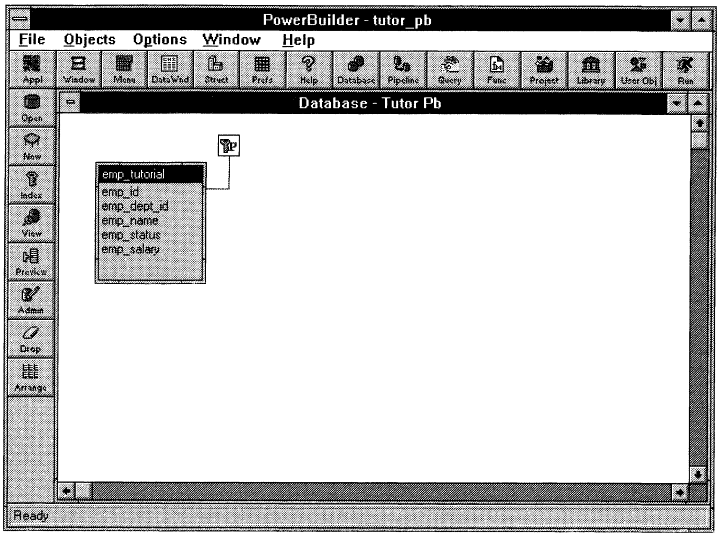
If you have made additional changes without clicking the Updt DB button, you are prompted to specify whether you want to update the database with your changes.


Data Manipulation for emp_tutorial

Save changes back to database?

7 Click *Yes* or *No*.

The Database painter workspace displays.


PowerBuilder - tutor_pb
File Objects Options Window Help
Appl Window Menu DataWnd Struct Prefs Help Database Pipeline Query Func Project Library User Obj Run
Open New Index View Preview Admin Drop Arrange
Database - Tutor Pb
emp_tutorial
emp_id
emp_dept_id
emp_name
emp_status
emp_salary
Ready

8 Select File➤Close from the menu bar to close the Database painter.

The initial window displays.

You have created a new database table, provided extended attribute information for its columns, and put data into it.

LESSON 4

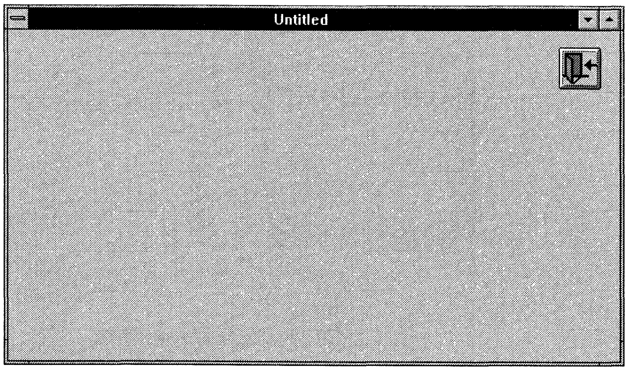
Building the Window

Windows are the main interface between the user and PowerBuilder applications. Windows can display information, request information from a user, and respond to mouse or keyboard actions.

In this lesson you will:

- Build a window for your application
- Add a PictureButton to close the window
- Build a script for the PictureButton
- Build a script for the application

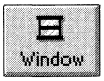
When you finish this lesson, your application window will have an Exit PictureButton that works. It will look like this during execution.



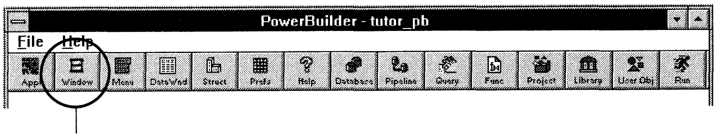
How long will this lesson take?
About 20 minutes.

Create a new empty window

Where you are
Lesson 4 Building the Window
➡ Create a new empty window
Add a PictureButton control
Display the attribute values
Add a script for the PictureButton control
Save the window
Add an application script
Run the application

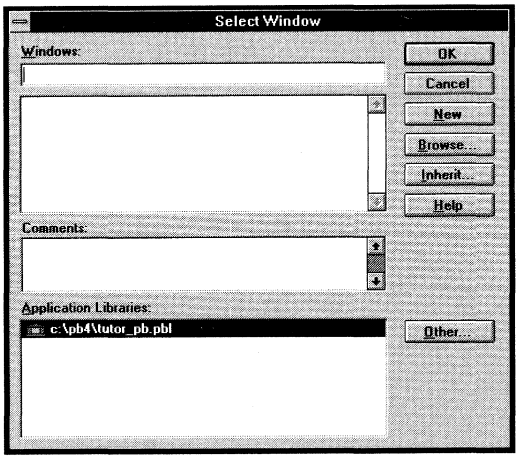


1 **Click the *Window painter* button in the PowerBar.**



Window painter

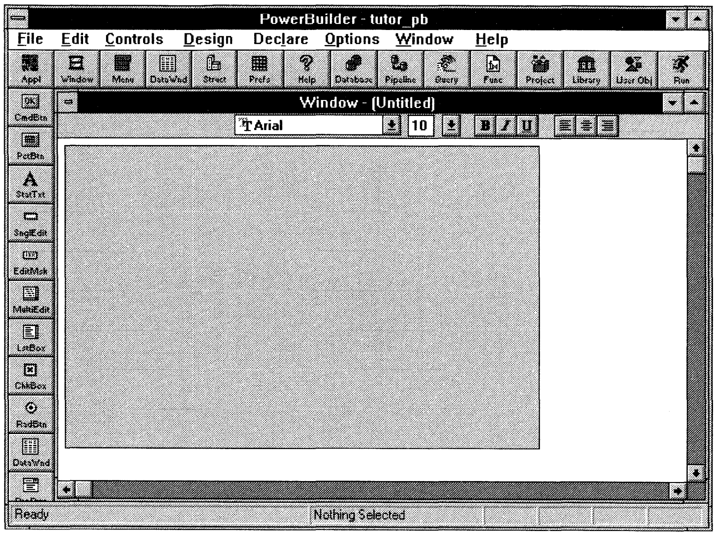
The Select Window dialog box opens.



From this dialog box you can open an existing window and modify it, create a new window, or inherit from an existing window to create a new descendent window.

2 Click the *New* button.

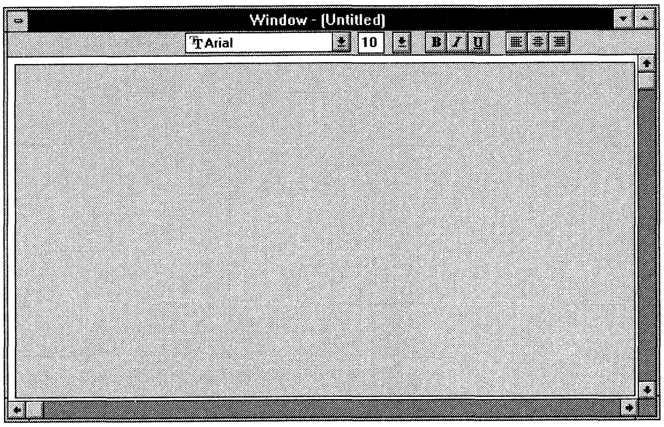
The Window painter workspace opens. The gray rectangle represents the window you are building.


PowerBuilder - tutor_pb
File Edit Controls Design Declare Options Window Help
Appl Window Menu DataWnd Struct Prefs Help Database Pipeline Query Func Project Library User Obj Run
CmdBtn PctBtn StatTxt SnglEdit EditMsk MultiEdit ListBox ChkBox RadBtn DataWnd
Window - (Untitled)
Arial 10
Ready
Nothing Selected

Your background color may be different
If your window displays with a white background instead of a gray background, the Default to 3D option has been disabled. When this option is enabled, the Window painter's default environment includes a gray background and 3D controls. To change this setting, select Options➤Default to 3D from the menu bar and re-open the Window painter.

3 Make the window rectangle larger.

To do this, move the pointer to the lower-right corner of the rectangle. When it turns into a double-headed arrow, drag the arrow down and to the right to enlarge the rectangle. Make it almost as large as the Window painter workspace.



Add a PictureButton control

Where you are
Lesson 4 Building the Window
Create a new empty window
➧ Add a PictureButton control
Display the attribute values
Add a script for the PictureButton control
Save the window
Add an application script
Run the application

The controls you can put in a window are pictured in the PainterBar and listed in the Controls menu.

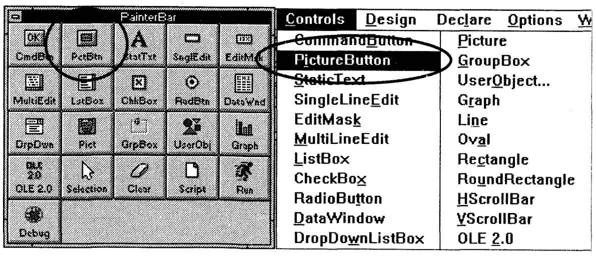
In this section you will put a PictureButton control in the window.

PictureButtons versus CommandButtons
This tutorial uses PictureButton controls, which display a button containing a bitmap that you specify, instead of CommandButtons, which simply display a button containing text. You could just as easily use CommandButtons.

To help you get acquainted with the PowerBuilder development environment, the bitmaps used for the tutorial's PictureButtons are similar to those used by PowerBuilder PainterBars. In your applications you should implement PictureButtons that have meaning to your application's users.

PctBtn

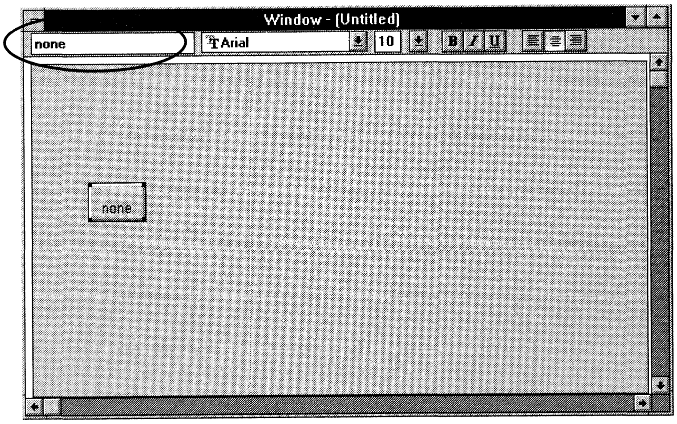
1 **Click the *PictureButton* button in the toolbar.**
or
Select Controls➤PictureButton from the menu bar.



Tip
If your display does not show all the buttons in the PainterBar, you may prefer to make the PainterBar floating (select Window➤Toolbars from the menu bar). Alternatively, you can turn off text and use PowerTips to remind you of each button's function.

2 **Click where you want the button in the window.**

A PictureButton displays at the selected location. Default text (**none**) displays on the button. The small black boxes in the corners indicate that this control is selected.

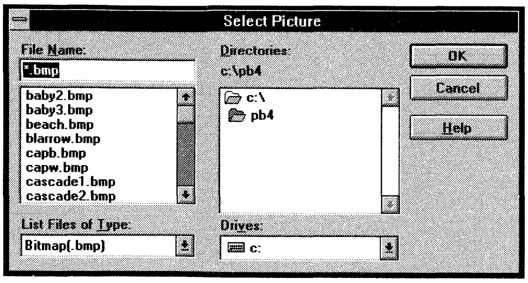


3 **Select *none* in the text box at the top of the Window painter and press** DELETE.

This eliminates the text from the PictureButton.

4 Double-click the button.

The Select Picture dialog box displays.


Select Picture

File Name: *.bmp

baby2.bmp
baby3.bmp
beach.bmp
blarrow.bmp
capb.bmp
capw.bmp
cascade1.bmp
cascade2.bmp

Directories: c:\pb4

c:\
pb4

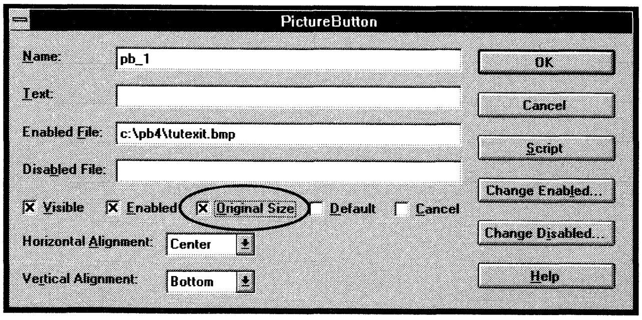
List Files of Type: Bitmap(.bmp)

Drives: c:

5 Scroll as needed and double-click the filename *tutexit.bmp*.

The Select Picture dialog box closes and the PictureButton dialog box displays.

6 Select the *Original Size* checkbox.


PictureButton

Name: pb_1

Text:

Enabled File: c:\pb4\tutexit.bmp

Disabled File:

[X] Visible [X] Enabled [X] Original Size [ ] Default [ ] Cancel

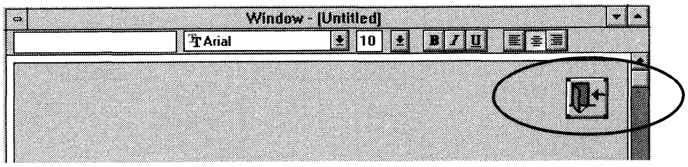
Horizontal Alignment: Center

Vertical Alignment: Bottom

7 Click *OK*.

The Window painter workspace displays.

**8 Adjust the location of the PictureButton.
Move it to the upper-right corner of the window.**



- ♦ Move the pointer near the center of the PictureButton.
- ♦ Press and hold the left mouse button.
- ♦ Drag the PictureButton to a new location.
- ♦ Release the mouse button.

How to delete controls
If you add a control to the window and later decide you do not want it, select the control and click the PainterBar's Clear button or press the DELETE key. This deletes the control and its scripts.

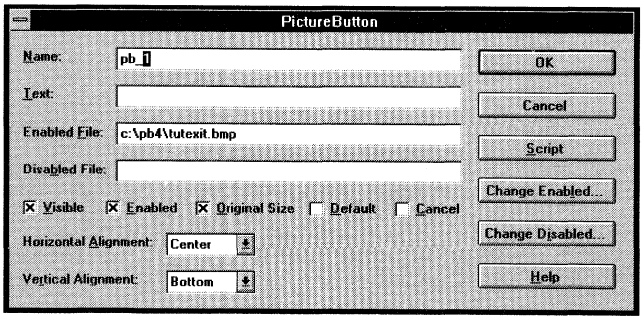
Display the attribute values

Where you are
Lesson 4 Building the Window
Create a new empty window
Add a PictureButton control
➡ Display the attribute values
Add a script for the PictureButton control
Save the window
Add an application script
Run the application

Every object and control has a style that determines how it looks. The style comes from the values assigned to its attributes. You can display some attributes of an object or control by double-clicking it.

◆ **Move the pointer to the PictureButton control that you just created and double-click.**

The PictureButton dialog box opens.



PictureButton

Name: pb_1
Text:
Enabled File: c:\pb4\tutexit.bmp
Disabled File:
[X] Visible [X] Enabled [X] Original Size [ ] Default [ ] Cancel
Horizontal Alignment: Center
Vertical Alignment: Bottom

This dialog box displays the values of some of the attributes of the PictureButton control.

Specify a name

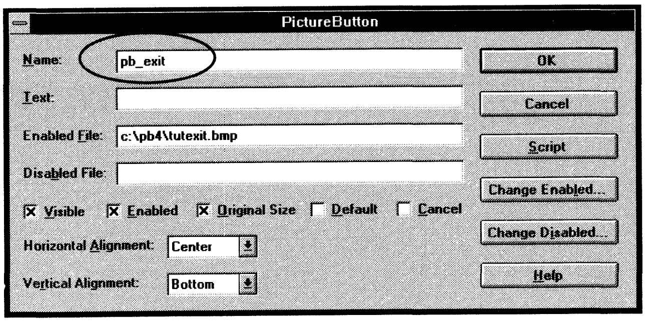
PowerBuilder assigns the PictureButton the default name pb_1. The 1 is highlighted, ready to change.

About default names for controls
PowerBuilder assigns each control a default name. The default name is the prefix specified in Preferences (for example, pb_) and the lowest number that will make the name unique.

It is not necessary to change the default name of a control. However, changing the number portion of the default name makes your application easier to understand.

1 **Type** *exit*

Exit replaces the highlighted 1 and the PictureButton control name becomes pb_exit.


PictureButton

Name: pb_exit
Text:
Enabled File: c:\pb4\tutexit.bmp
Disabled File:
[X] Visible [X] Enabled [X] Original Size [ ] Default [ ] Cancel
Horizontal Alignment: Center
Vertical Alignment: Bottom

2 **Click** *OK.*

You return to the Window painter workspace.

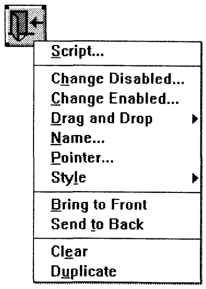
Add a script for the PictureButton control

Where you are
Lesson 4 Building the Window
Create a new empty window
Add a PictureButton control
Display the attribute values
➡ Add a script for the PictureButton control
Save the window
Add an application script
Run the application

Next you will add a one-line script to this button's Clicked event. When the user clicks on this PictureButton, PowerBuilder executes the Clicked event script.

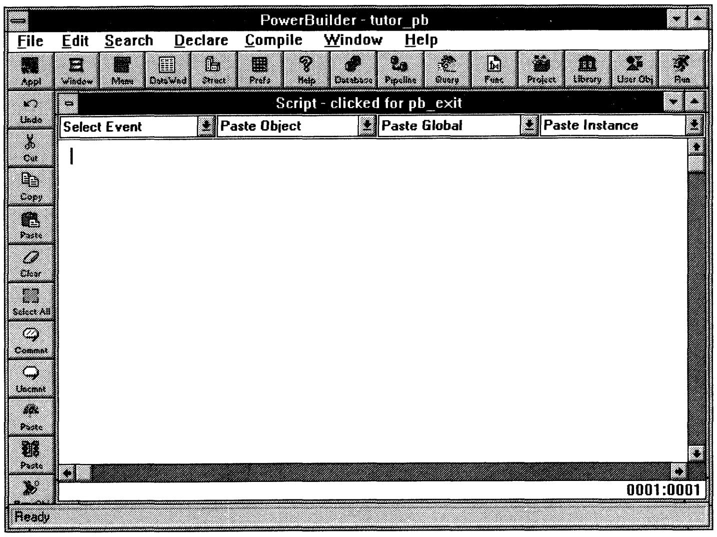
1 **Move the pointer to the button and right-click.**

The popup menu for the button displays.



2 Click *Script.*

The PowerScript painter displays.


PowerBuilder - tutor_pb
File Edit Search Declare Compile Window Help
Appl Window Menu DataWnd Struct Prefs Help Database Pipeline Query Func Project Library User Obj Run
Undo Cut Copy Paste Clear Select All Comment Uncmnt Paste Paste
Script - clicked for pb_exit
Select Event
Paste Object
Paste Global
Paste Instance
0001:0001
Ready

The PowerScript painter is a text editor. You can use the buttons in the PainterBar to:

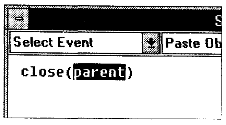
- **Undo** your last change
- **Cut** selected text and place it in the clipboard
- **Copy** selected text to the clipboard
- **Paste** text from the clipboard
- **Clear** selected text
- **Select All** text
- **Comment** out one or more selected lines
- **Uncomment** one or more selected lines
- **Paint SQL** statements
- **Paste** a statement into your script
- **Browse** object information
- **Return** to the Window painter

Build the script

1 **Type this script:**

```
close(parent)
```

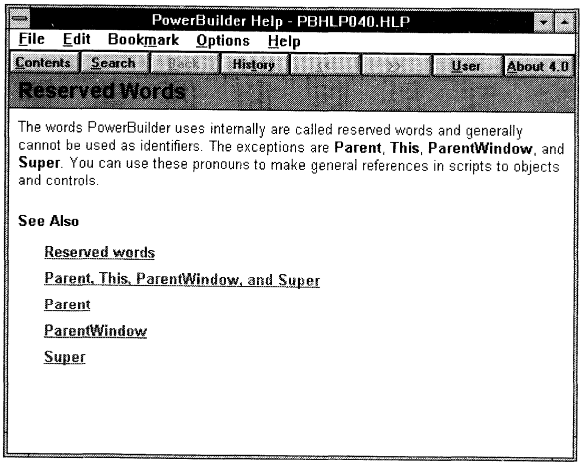
2 **Highlight parent.**



```
close(parent)
```

3 **Press SHIFT+F1.**

PowerBuilder displays help for the reserved word Parent.



PowerBuilder Help - PBHLP040.HLP
File Edit Bookmark Options Help
Contents Search Back History << >> User About 4.0

Reserved Words

The words PowerBuilder uses internally are called reserved words and generally cannot be used as identifiers. The exceptions are **Parent**, **This**, **ParentWindow**, and **Super**. You can use these pronouns to make general references in scripts to objects and controls.

See Also

Reserved words
Parent, This, ParentWindow, and Super
Parent
ParentWindow
Super

> **Accessing context-sensitive Help**
> Select a function or reserved word and press SHIFT+F1.

4 **Double-click the Control-menu box in the upper-left corner of the Help window to close it.**
or
Select File➤Exit from the menu bar.

Compile the script

♦ **Click the *Return* button.**
or
Select File➤Return from the menu bar.

Your script compiles and you return to the Window painter workspace.

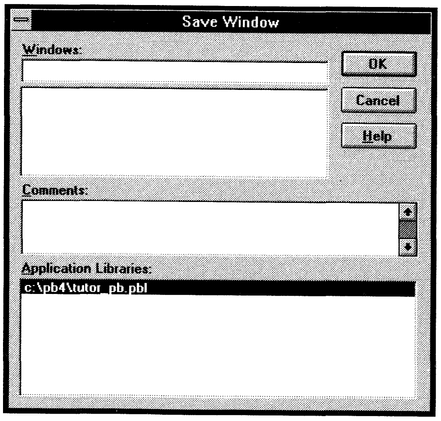
Handling errors in scripts
When there is an error in a script, an error window displays at the bottom of the PowerScript painter with the line number of the error and the error message. To move to the line with the error, click the error message. After you correct the error, click the Return button.

Save the window

Where you are
Lesson 4 Building the Window
Create a new empty window
Add a PictureButton control
Display the attribute values
Add a script for the PictureButton control
➡ Save the window
Add an application script
Run the application

1 **Select File➢Save from the menu bar.**

The Save Window dialog box displays so you can name and save your window.



Save versus Save As
You could also select Save As. For a new window, Save and Save As are equivalent.

2 **Type *w_employees* for the window name.**

3 **Enter comments in the Comments box.**

This documents your window.

4 **Click *OK*.**

PowerBuilder saves the new window in tutor_pb.pbl and returns you to the Window painter.

Since you will be using the Window painter later, you will leave it open.

About opening and closing painters
Up to this point in the tutorial you have closed each painter when you finished using it. Typically you will want to leave several painters open to keep them readily available as you work.

Add an application script

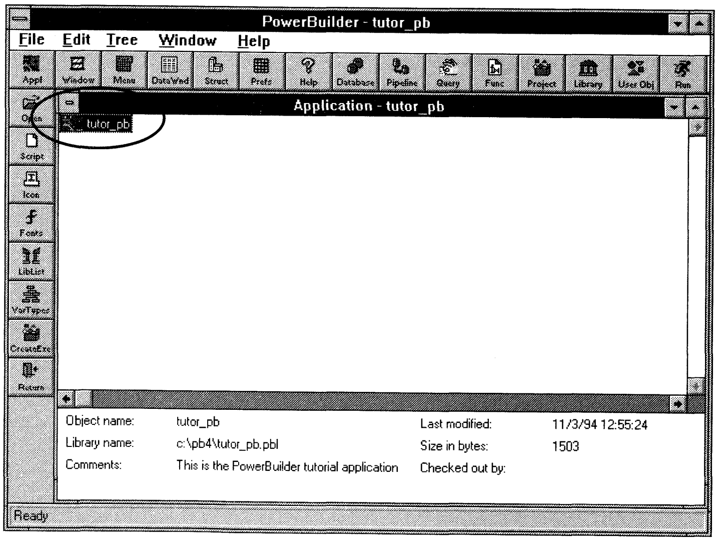
Where you are
<u>Lesson 4 Building the Window</u>
Create a new empty window
Add a PictureButton control
Display the attribute values
Add a script for the PictureButton control
Save the window
➡ Add an application script
Run the application

One of the most important functions of the PowerBuilder application object is opening the initial window of the application. All application objects have the appropriate instructions in the script for the Open event.

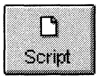
You will now build that script for your application.

1 **Click the *Application painter* button on the PowerBar.**

The Application painter displays.


PowerBuilder - tutor_pb
File Edit Tree Window Help
Appl Window Menu DataWnd Struct Prefs Help Database Pipeline Query Func Project Library User Obj Run
Open Script Icon Fonts LibList VarTypes CreateExe Return
Application - tutor_pb
tutor_pb
Ready

Object name: tutor_pb
Library name: c:\pb4\tutor_pb.pbl
Comments: This is the PowerBuilder tutorial application
Last modified: 11/3/94 12:55:24
Size in bytes: 1503
Checked out by:

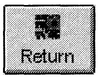


2 **Click the *Script* button in the PainterBar.**

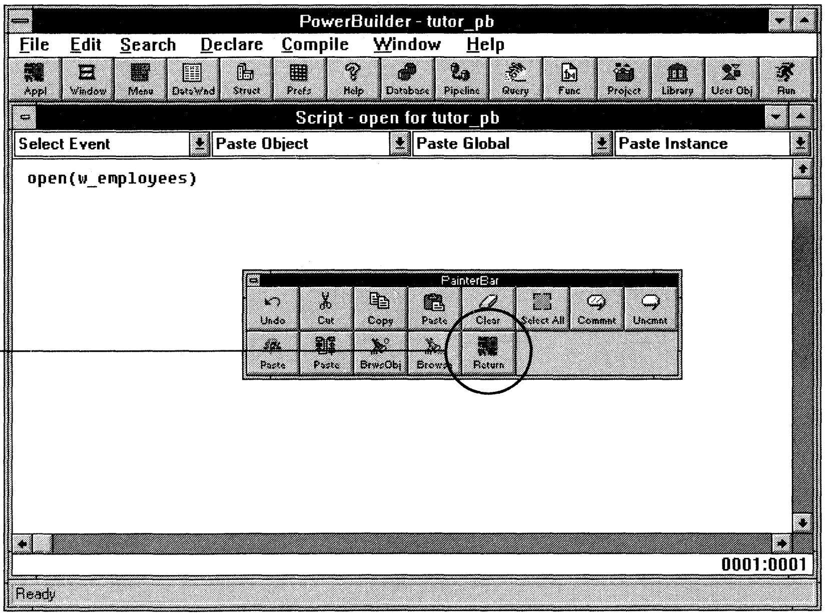
The PowerScript painter displays. This is the same PowerScript painter you were in before, except this time you reached it from the Application painter instead of the Window painter.

3 **Type the following one-line script:**

```
open(w_employees)
```



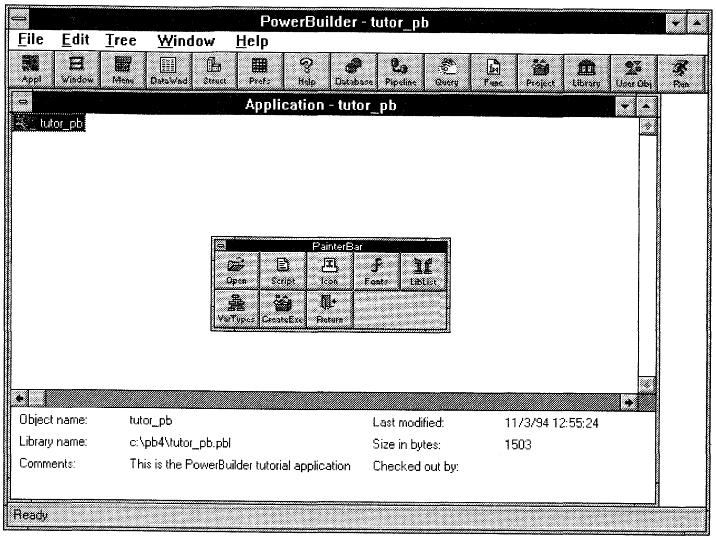
4 **Click the *Return* button in the PainterBar.**
or
Select File➤Return from the menu bar.


PowerBuilder - tutor_pb
File Edit Search Declare Compile Window Help
Appl Window Menu DataWnd Struct Prefs Help Database Pipeline Query Func Project Library User Obj Run
Script - open for tutor_pb
Select Event
Paste Object
Paste Global
Paste Instance
PainterBar
Undo Cut Copy Paste Clear Select All Commnt Uncmnt
Paste Paste BrwsObj Brows Return
0001:0001
Ready

```
open(w_employees)
```

The Return button on the PainterBar

PowerBuilder compiles your script and returns you to the Application painter.


PowerBuilder - tutor_pb
File Edit Tree Window Help
Appl Window Menu DataWnd Struct Prefs Help Database Pipeline Query Func Project Library User Obj Run
Application - tutor_pb
tutor_pb
PainterBar
Open Script Icon Fonts LibList
VarTypes CreateExe Return

Object name: tutor_pb
Library name: c:\pb4\tutor_pb.pbl
Comments: This is the PowerBuilder tutorial application
Last modified: 11/3/94 12:55:24
Size in bytes: 1503
Checked out by:

Ready

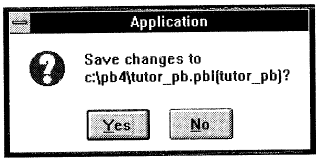
Run the application

Where you are
Lesson 4 Building the Window
Create a new empty window
Add a PictureButton control
Display the attribute values
Add a script for the PictureButton control
Save the window
Add an application script
➡ Run the application

You can now run your application to see how it works.

1 **Click the *Run* button on the PowerBar.**
or
Select File➤Run from the menu bar.

PowerBuilder prompts you to save the script you just wrote before running the application.

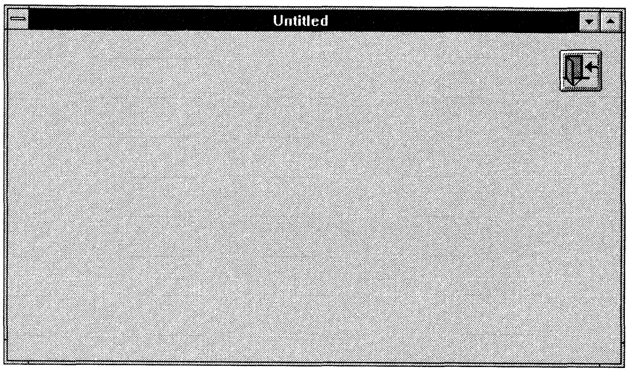


Save changes to
c:\pb4\tutor_pb.pbl(tutor_pb)?

2 Click *Yes*.

Your application runs. The script for the application Open event opens your window.

Your window may not fully display. In the next lesson you will set its opening position on the display screen.



3 Click the *Exit* PictureButton.
If necessary, drag the window so that you can access the button.

The script for the button's Clicked event executes. The window closes and you return to the Application painter.

4 Select File➢Close from the menu bar.

The Application painter closes and you return to the Window painter, which you left open.

Now that the basic window is working, you can add a title and other enhancements.

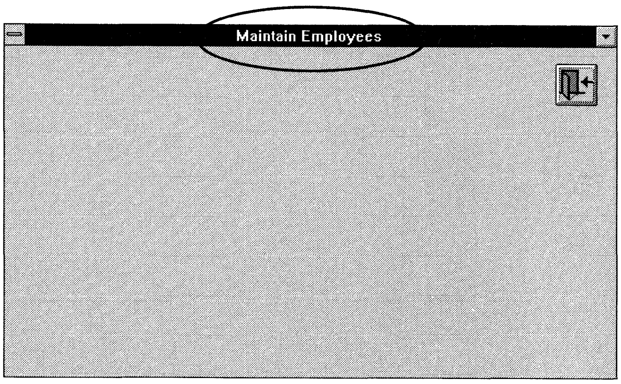
LESSON 5

Enhancing the Window

In this lesson you will:

- Define the window's style, which includes adding a title and changing some of the window's attributes
- Define the window's opening position on the screen

When you finish this lesson, your application window will look like this during execution.



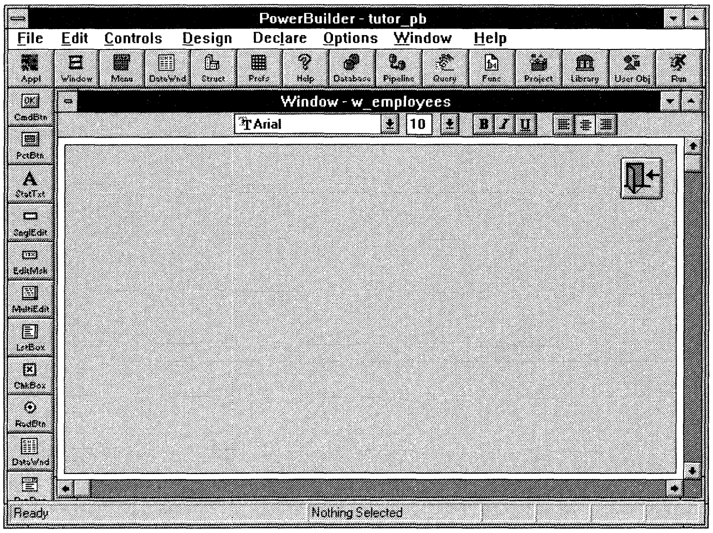
How long will this lesson take?
About 10 minutes.

Define the window's style

Where you are
Lesson 5 Enhancing the Window
➡ Define the window's style
Define the window's opening position
Run the application

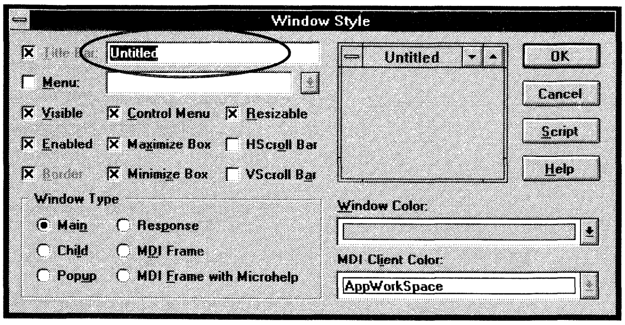
In the Window Style dialog box, you can set the window's title, its menu, and attributes such as whether it is resizable.

1 **Make sure your screen looks like this.**
If not, open the Window painter and select the *w_employees* window.


PowerBuilder - tutor_pb
File Edit Controls Design Declare Options Window Help
Appl Window Menu DataWnd Struct Prefs Help Database Pipeline Query Func Project Library User Obj Run
CmdBtn PctBtn StatTxt SnglEdit EditMsk MultiEdit ListBox ChkBox RadBtn DataWnd
Window - w_employees
Arial 10
Ready
Nothing Selected

2 **Select Design➢Window Style from the menu bar.**

The Window Style dialog box displays. The title bar text is highlighted.



3 **Type** *Maintain Employees*

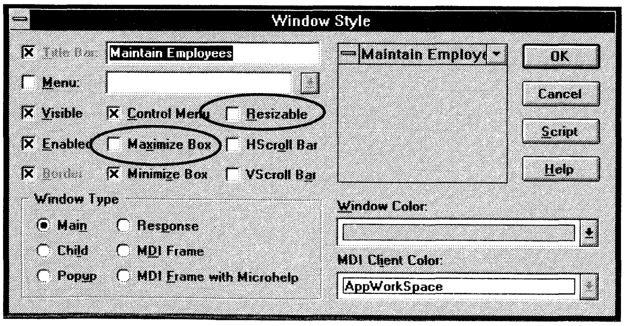
This replaces Untitled in the Title Bar box.

4 **Deselect the** *Maximize* **Box checkbox.**

Everything in the window will be visible at once so there will be no need to maximize the window.

5 **Deselect the** *Resizable* **checkbox.**

This will prevent users from resizing the window during execution.



6 **Click** *OK.*

This closes the Window Style dialog box and returns you to the Window painter workspace.

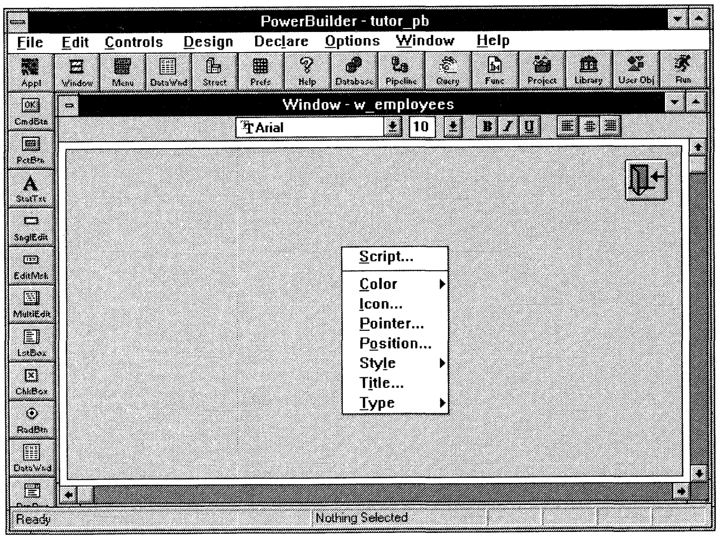
Define the window's opening position

Where you are
Lesson 5 Enhancing the Window
Define the window's style
➡ Define the window's opening position
Run the application

In the Window Position dialog box, you will set the opening position of the window. You can access the Window Position dialog box from Design on the main menu or from the popup menu for the window.

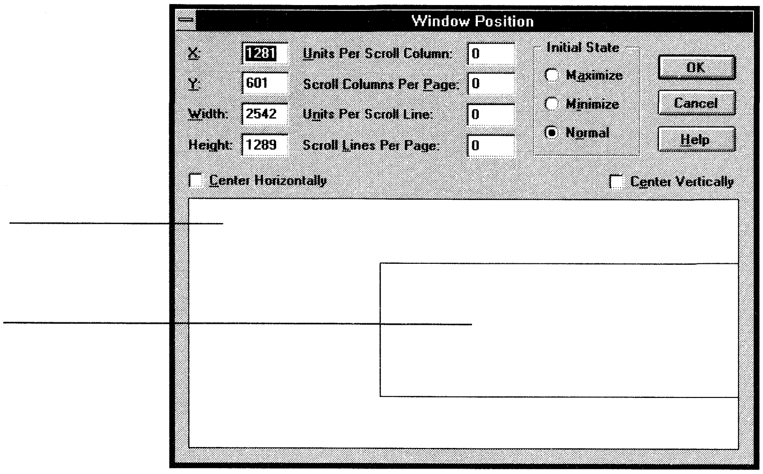
1 **Move the pointer to an unused spot in the window and right-click.**

The popup menu for the window displays.


PowerBuilder - tutor_pb
File Edit Controls Design Declare Options Window Help
Appl Window Menu DataWnd Struct Prefs Help Database Pipeline Query Func Project Library User Obj Run
CmdBtn PictBtn StatTxt SnglEdit EditMsk MultiEdit ListBox ChkBox RadBtn DataWnd
Window - w_employees
Arial 10
Script...
Color
Icon...
Pointer...
Position...
Style
Title...
Type
Ready
Nothing Selected

2 **Click** ***Position.***

The Window Position dialog box displays. The large rectangle represents the entire screen. The small rectangle represents the relative size and position of the window you are working on.



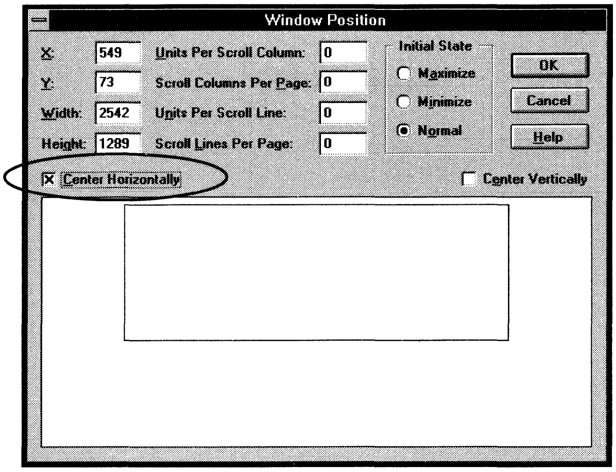
X: 1281
Y: 601
Width: 2542
Height: 1289
Units Per Scroll Column: 0
Scroll Columns Per Page: 0
Units Per Scroll Line: 0
Scroll Lines Per Page: 0
Initial State: [✓ Normal]
Center Horizontally: unchecked
Center Vertically: unchecked

3 **Click inside the small rectangle and drag it closer to the top of the workspace.**

This is where the window will display when it first opens during execution.

4 **Select the *Center Horizontally* checkbox.**

This centers your window horizontally on the screen during execution.



Window Position

X: 549 Units Per Scroll Column: 0

Y: 73 Scroll Columns Per Page: 0

Width: 2542 Units Per Scroll Line: 0

Height: 1289 Scroll Lines Per Page: 0

Initial State: ( ) Maximize ( ) Minimize (●) Normal

[X] Center Horizontally [ ] Center Vertically

5 **Click *OK*.**

The Window painter displays.

Run the application

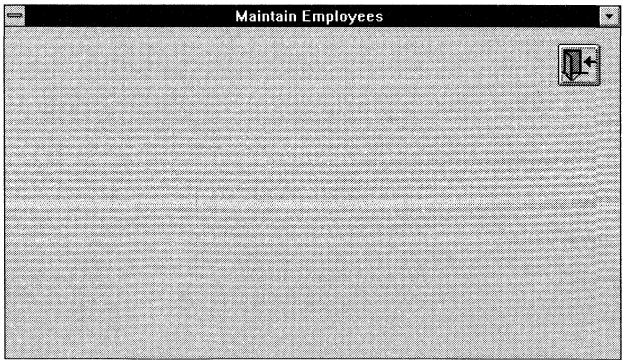
Where you are
Lesson 5 Enhancing the Window
Define the window's style
Define the window's opening position
➡ Run the application

1 **Click the *Run* button in the PowerBar.**
or
Select File➤Run from the menu bar.

PowerBuilder prompts you to save your changes.

2 **Click *Yes*.**

Your window displays. It is centered near the top of the screen, has a title, and cannot be maximized or resized.



3 **Click the *Exit* PictureButton.**

The Window painter displays. Again, you will leave the Window painter open.

Now you are ready to make this application work with the database.

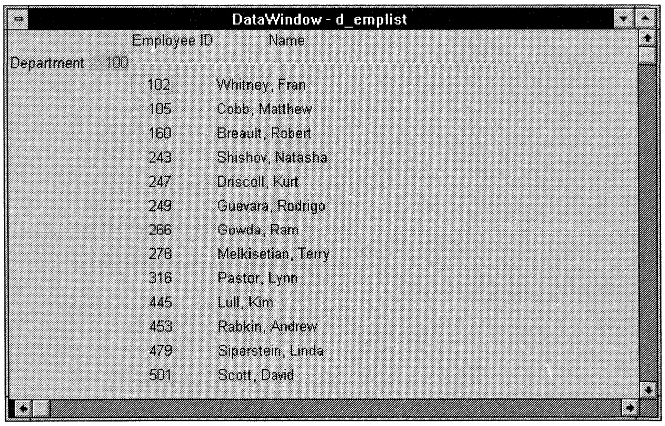
LESSON 6

Building the First DataWindow Object

DataWindow objects are one of the most powerful and useful features of PowerBuilder. They can connect to a database, retrieve rows, display the rows in various styles, and update the database.

In this lesson you will create a DataWindow object that retrieves and displays the employees in the emp_tutorial table.

When you finish this lesson and preview the DataWindow object you create, it will look like this. Note that the DataWindow object is *not* yet a part of your application window.



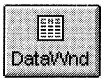
DataWindow - d_emplist

	Employee ID	Name
Department 100		
	102	Whitney, Fran
	105	Cobb, Matthew
	160	Breault, Robert
	243	Shishov, Natasha
	247	Driscoll, Kurt
	249	Guevara, Rodrigo
	266	Gowda, Ram
	278	Melkisetian, Terry
	316	Pastor, Lynn
	445	Lull, Kim
	453	Rabkin, Andrew
	479	Siperstein, Linda
	501	Scott, David

How long will this lesson take?
About 20 minutes.

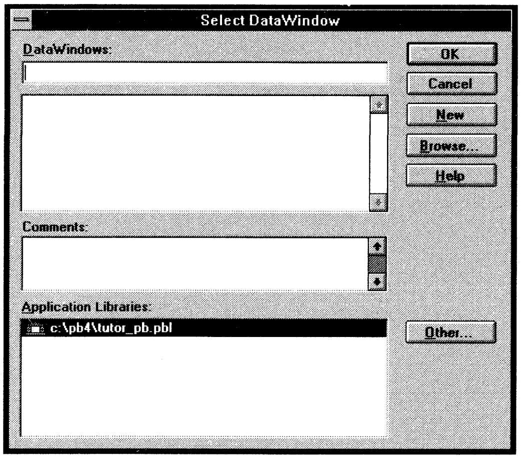
Create the new DataWindow object

Where you are
Lesson 6 Building the First DataWindow Object
➡ Create the new DataWindow object
Preview the DataWindow object
Save the DataWindow object
Create a group
Make cosmetic changes



1 **Click the *DataWindow painter* button in the PowerBar (not in the Window painter PainterBar).**

The DataWindow painter opens and the Select DataWindow dialog box displays.

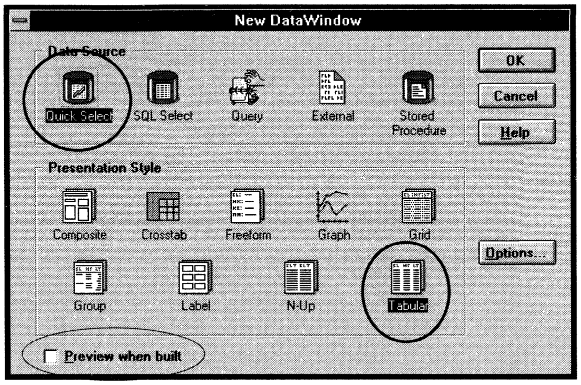


In the Select DataWindow dialog box, you can select an existing DataWindow object and modify it, or you can create a new one. You want to create a new DataWindow object.

2 Click *New*.

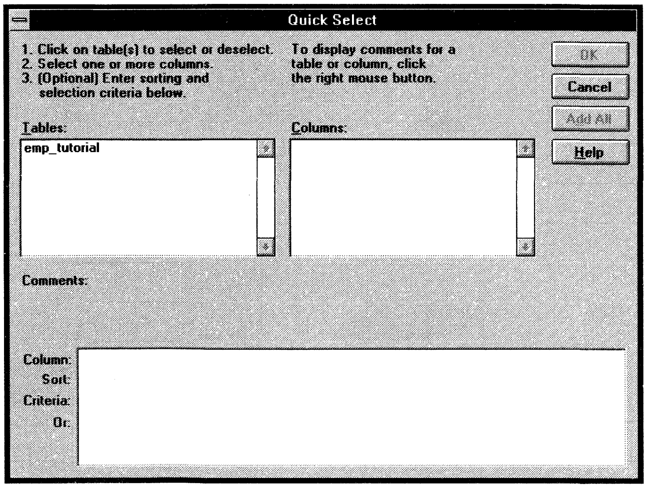
The New DataWindow dialog box displays.

3 Select *Quick Select* as the data source.
Select *Tabular* as the presentation style.
Make sure that the *Preview when built* checkbox is deselected.



4 Click *OK*.

The Quick Select dialog box displays because you specified Quick Select as the data source. The Tables box automatically lists all tables in the current database. The current database is the Tutor_pb database, which you created earlier.



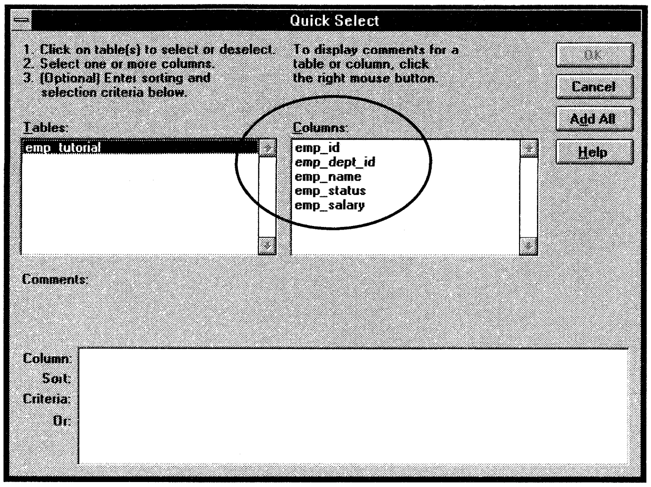
1. Click on table(s) to select or deselect.
2. Select one or more columns.
3. (Optional) Enter sorting and selection criteria below.

To display comments for a table or column, click the right mouse button.

emp_tutorial

5 Click ***emp_tutorial.***

This opens the table and lists its columns.



1. Click on table(s) to select or deselect.
2. Select one or more columns.
3. [Optional] Enter sorting and selection criteria below.

To display comments for a table or column, click the right mouse button.

Tables: emp_tutorial

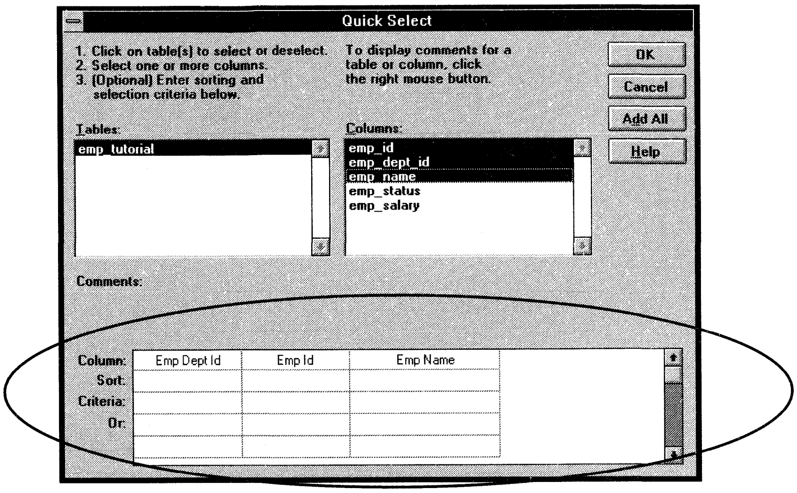
Columns: emp_id, emp_dept_id, emp_name, emp_status, emp_salary

For this DataWindow, you select three columns.

6 Click ***emp_dept_id*** **first.**
Then click ***emp_id.***
Then click ***emp_name.***

PowerBuilder displays the selected columns in a grid at the bottom of the Quick Select dialog box.

Tip
The column selection order determines the left to right display order in the DataWindow.


Quick Select
1. Click on table(s) to select or deselect.
2. Select one or more columns.
3. (Optional) Enter sorting and selection criteria below.
To display comments for a table or column, click the right mouse button.
OK
Cancel
Add All
Help
Tables:
emp_tutorial
Columns:
emp_id
emp_dept_id
emp_name
emp_status
emp_salary
Comments:
Column:
Sort:
Criteria:
Or:
Emp Dept Id
Emp Id
Emp Name

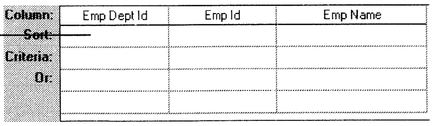
You can use this area to specify sort criteria (SQL ORDER BY clause) and selection criteria (SQL WHERE clause). In this case, you will only specify sort criteria.

You will sort on emp_dept_id and within emp_dept_id on emp_id.

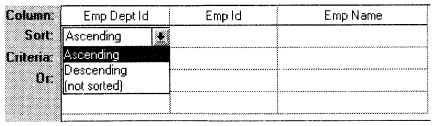
7 In the grid, click in the cell beside *Sort* and below *Emp Dept ID*.

A dropdown listbox displays.

Click in this cell

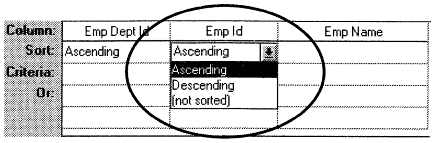


8 Select *Ascending* from the dropdown listbox.



This specifies sorting on the emp_dept_id column. Because this column is first, it is the highest level of sorting.

9 Click in the cell below the *Emp Id* heading. Select *Ascending* from the dropdown listbox.



Column:	Emp Dept Id	Emp Id	Emp Name
Sort:	Ascending	Ascending	
Criteria:			
Or:			

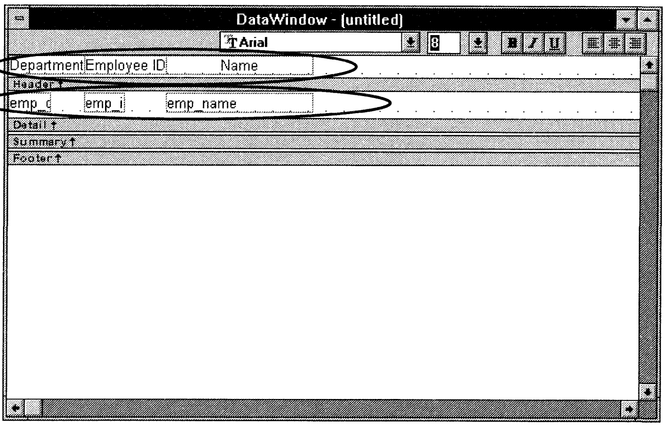
This specifies the next level of sorting.

10 Click *OK*.

PowerBuilder creates the new DataWindow object and displays it in the DataWindow painter workspace.

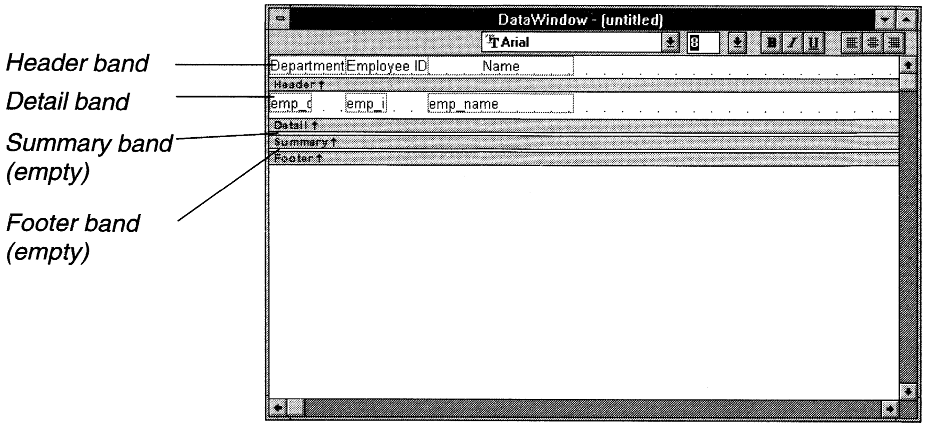
Column headers from the repository

The three columns you selected



In the header band, PowerBuilder displays the headings you specified in the extended attributes information when you created the table in the Database painter. They are just starting points; you can override them if you want.

The DataWindow painter workspace is divided into four areas called bands: header, detail, summary, and footer. You can modify these bands. For example, you can change their sizes; add text, lines, boxes, or ovals; and change colors and fonts. If a band is not open you can create as much space as needed by pointing at the bar (the gray area with an up arrow), pressing and holding the left mouse button, and dragging down until you see the desired amount of space.



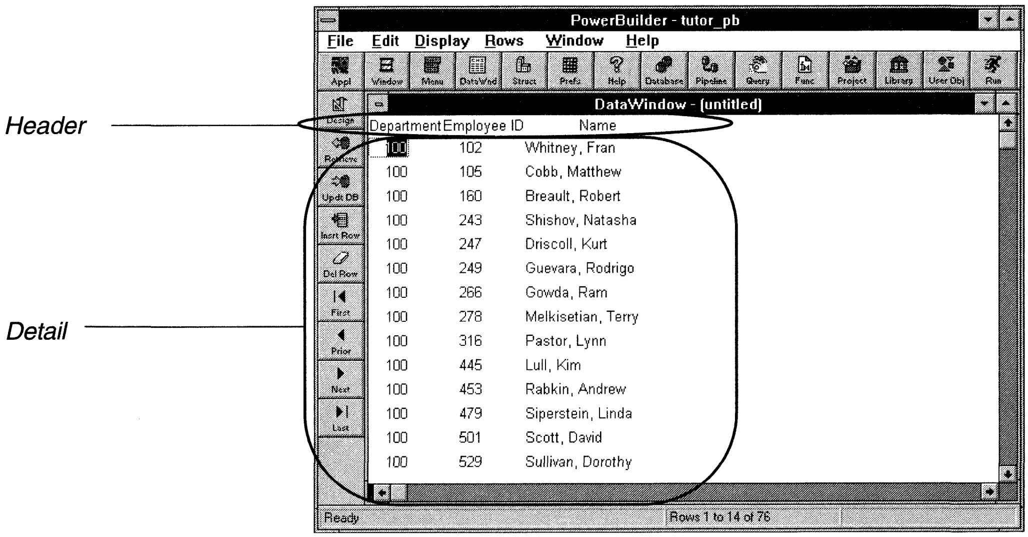
Preview the DataWindow object

Where you are
Lesson 6 Building the First DataWindow Object
Create the new DataWindow object
➡ Preview the DataWindow object
Save the DataWindow object
Create a group
Make cosmetic changes

You can preview your new DataWindow object immediately.

1 **Click the *Preview* button in the PainterBar.**

PowerBuilder displays the DataWindow as it will display during execution. To make it realistic, PowerBuilder retrieves and displays data in the DataWindow. Header information and the detail lines display. If you had specified footer information, it would also display.


Header
Detail
PowerBuilder - tutor_pb
File Edit Display Rows Window Help
Appl Window Menu DataWnd Struct Prefs Help Database Pipeline Query Func Project Library User Obj Run
DataWindow - (untitled)
Design Retrieve Updt DB Insrt Row Del Row First Prior Next Last
Ready
Rows 1 to 14 of 76

Department	Employee ID	Name
100	102	Whitney, Fran
100	105	Cobb, Matthew
100	160	Breault, Robert
100	243	Shishov, Natasha
100	247	Driscoll, Kurt
100	249	Guevara, Rodrigo
100	266	Gowda, Ram
100	278	Melkisetian, Terry
100	316	Pastor, Lynn
100	445	Lull, Kim
100	453	Rabkin, Andrew
100	479	Siperstein, Linda
100	501	Scott, David
100	529	Sullivan, Dorothy

All employees in the table display. Employees are sorted by department ID then employee ID number, as you specified.

2 **Click the *Design* button in the PainterBar.**

You return to the DataWindow painter workspace.

DataWindow buffers previewed rows
The DataWindow preview function keeps previewed rows in memory to speed up later previews. However, this may cause a problem if you change the DataWindow's format. In this case, click the Retrieve button on the PainterBar to refresh the display.

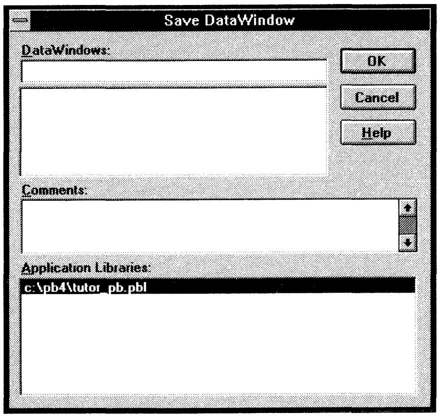
Save the DataWindow object

Where you are
Lesson 6 Building the First DataWindow Object
Create the new DataWindow object
Preview the DataWindow object
➡ Save the DataWindow object
Create a group
Make cosmetic changes

Now you will name the DataWindow object and save it. As you refine and improve the DataWindow object, you should save it periodically to keep from losing work. Whenever you leave the DataWindow painter, you are prompted to save your changes if you've made any.

1 **Select File➤Save from the menu bar.**

The Save DataWindow dialog box displays. The insertion point is in the DataWindows box.



2 **Type** *d_emplist*

This names the DataWindow object. The prefix **d** is standard for DataWindow objects.

3 **Enter comments in the Comments box.**

Comments identify the DataWindow object and display in the Library painter. Comments are optional.

4 Click *OK*.

PowerBuilder saves the DataWindow object and closes the Save DataWindow dialog box.

Create a group

Where you are
Lesson 6 Building the First DataWindow Object
Create the new DataWindow object
Preview the DataWindow object
Save the DataWindow object
➡ Create a group
Make cosmetic changes

In preview, the DataWindow displays employee data sorted by department ID, and within department ID by employee ID. However, it is not easy to see where one department ends and another begins.

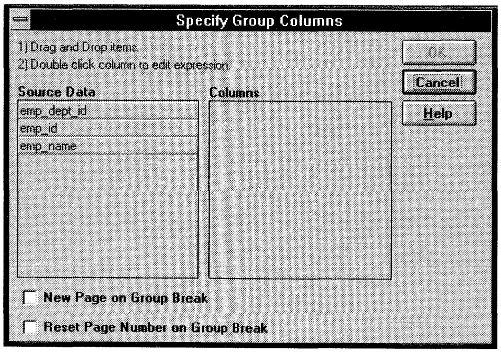
In this section, you will make the data easier to read. You will group the data by department ID, display the department ID at the top of each page of the DataWindow, and start each department on a new page.

To do this you will create a group and move the Department ID header and the emp_dept_id column to a new area in the DataWindow.

Create new group

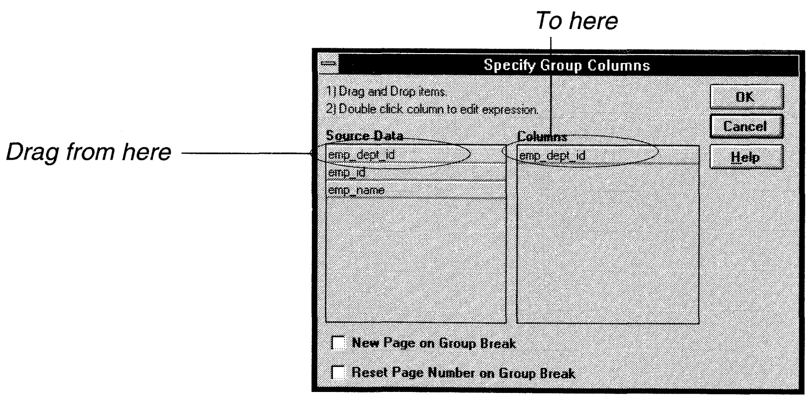
1 **Select Rows➤Create Group from the menu bar.**

The Specify Group Columns dialog box displays. You are going to group on the emp_dept_id column.



1) Drag and Drop items.
2) Double click column to edit expression.

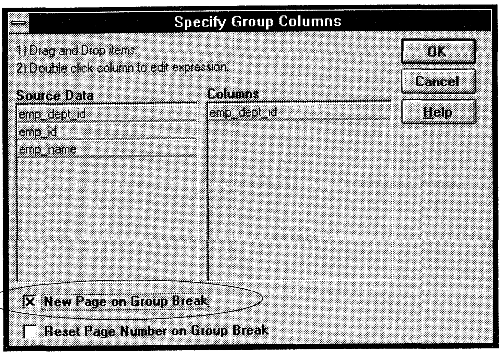
2 **Press and drag *emp_dept_id* from the *Source Data* box to the *Columns* box.**



1) Drag and Drop items.
2) Double click column to edit expression.

This groups information in the window by department ID.

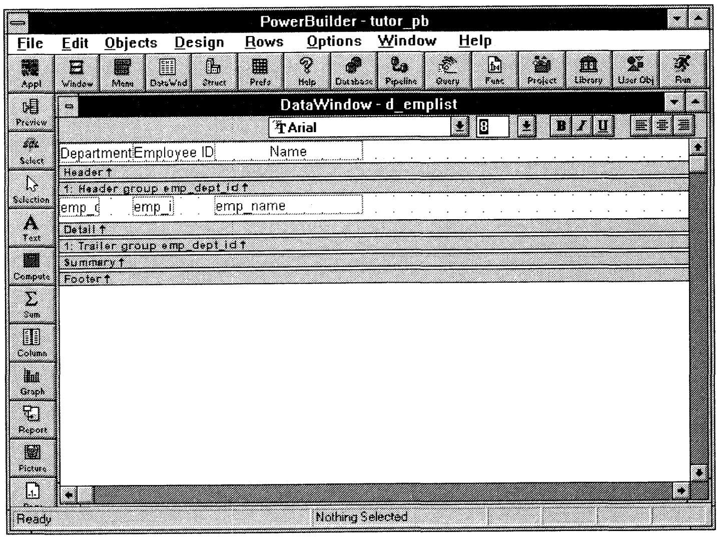
3 Select the *New Page on Group Break* checkbox.



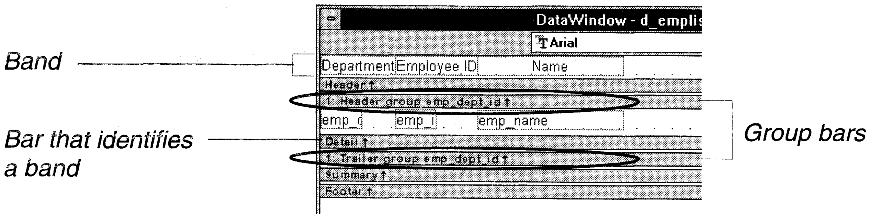
This generates a new page in the DataWindow each time the department ID changes.

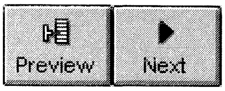
4 Click *OK*.

PowerBuilder saves the grouping and returns to the DataWindow workspace.


PowerBuilder - tutor_pb
File Edit Objects Design Rows Options Window Help
Appl Window Menu DataWnd Struct Prefs Help Database Pipeline Query Func Project Library User Obj Run
Preview Select Selection Text Compute Sum Column Graph Report Picture
DataWindow - d_emplist
Arial
Department Employee ID Name
Header ↑
1: Header group emp_dept_id ↑
emp_d emp_i emp_name
Detail ↑
1: Trailer group emp_dept_id ↑
Summary ↑
Footer ↑
Ready
Nothing Selected

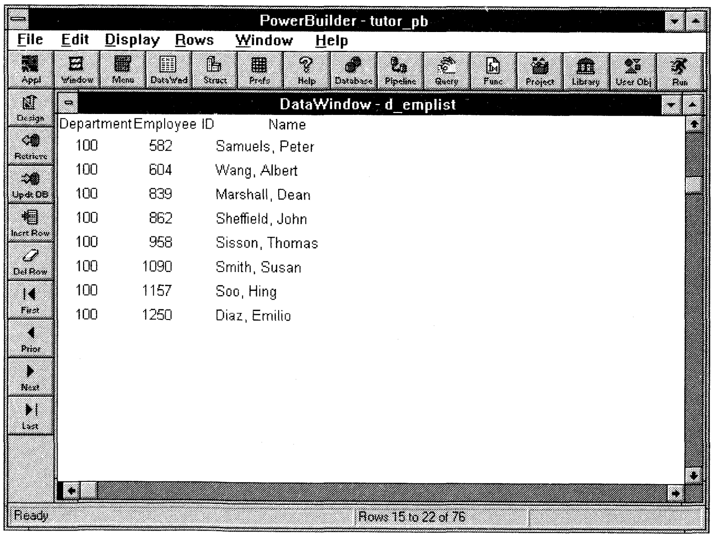
Notice that the DataWindow has new bands: a header and a trailer for the group. The bars representing the header and trailer contain the group number, the name of the column in the group, and an up arrow pointing to the area. At this point, the bands for the group header and trailer are narrow because they are empty.


Band
Bar that identifies a band
Group bars
DataWindow - d_empli
Arial
Department Employee ID Name
Header
1: Header group emp_dept_id
emp_d emp_i emp_name
Detail
1: Trailer group emp_dept_id
Summary
Footer

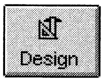


5 **Click the *Preview* button.**
Click the *Next* button in the PainterBar to see the page break between Department 100 and Department 200.

The DataWindow now starts a new page for each department. Since each page shows one department, it is unnecessary to repeat emp_dept_id on each line. Let's make the department ID display in the group header instead.


PowerBuilder - tutor_pb
File Edit Display Rows Window Help
Appl Window Menu DataWnd Struct Prefs Help Database Pipeline Query Func Project Library User Obj Run
Design Retrieve Updt DB Insrt Row Del Row First Prior Next Last
DataWindow - d_emplist
Ready
Rows 15 to 22 of 76

```
Department Employee ID    Name
  100         582     Samuels, Peter
  100         604     Wang, Albert
  100         839     Marshall, Dean
  100         862     Sheffield, John
  100         958     Sisson, Thomas
  100        1090     Smith, Susan
  100        1157     Soo, Hing
  100        1250     Diaz, Emilio
```



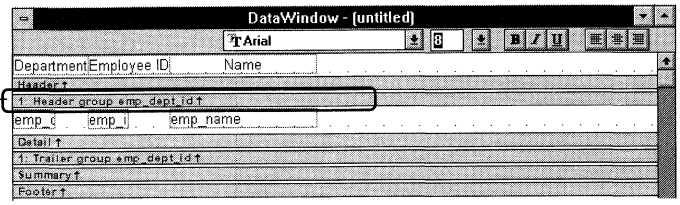
6 **Click the *Design* button in the PainterBar.**

You return to the DataWindow painter workspace.

Move department ID to the group header

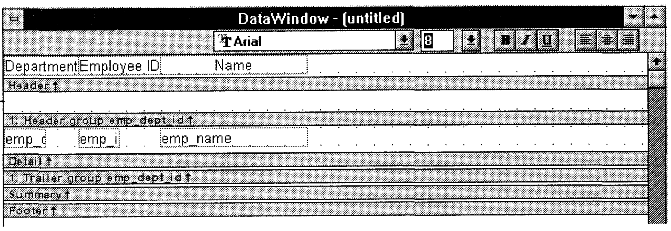
1 **Move the pointer to the bar titled *1:Header group emp_dept_id* until you see a double-pointed arrow.**
Drag the bar down to leave enough space for one line of information (see the second picture below).

Move the pointer to the group header bar and drag it down



This creates a space for heading information for the group.

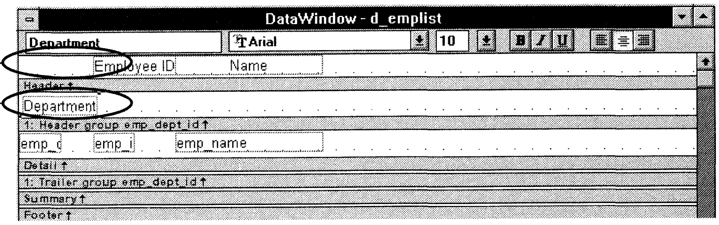
New space



2 **Move the pointer to the text for the Department ID heading, press the left mouse button, and drag the text into the group header band.**

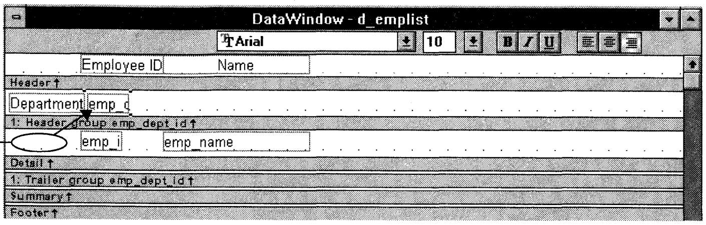
Drag from here

To here

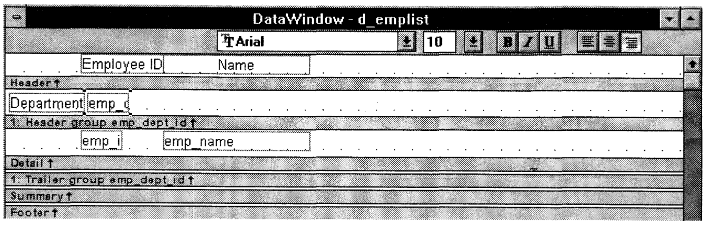


3 **Move the pointer to the *emp_dept_id* column in the detail area and drag it to the right of the Department heading in the group header band.**

Drag up to the group header band



The workspace looks like this after you put the Department heading and the emp_dept_id column in the group header.



4 **Click on the *Employee ID* heading.**
Hold down the CTRL key and click on the *Name* heading.

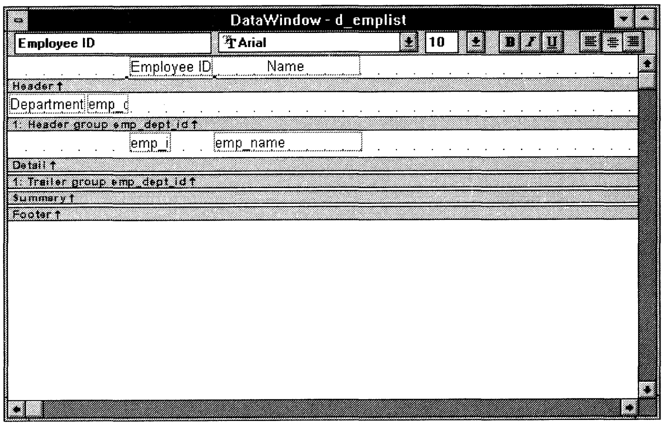
The two items are selected. This is one method of selecting multiple items.

You can also use lasso select to select multiple items
Another way to select multiple items is to press and drag across the items to be selected. This is called lasso select; for a more detailed explanation, see "Enhance the DataWindow Object" in Lesson 8, "Building the Second DataWindow Object."

5 **Press the left mouse button over either selected object and drag to the right about one-half inch.**
Release the left mouse button.

6 **Use the CTRL+click technique to select the emp_id and emp_name columns and drag them to the right about one-half inch.**

When you are finished, your workspace should look something like this.

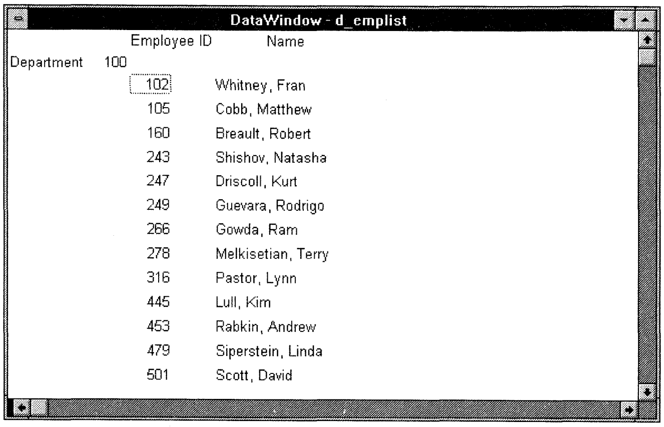


Preview again

1 **Click the *Preview* button.**

PowerBuilder displays the DataWindow as it will display during execution, and retrieves and displays data in it.

Again the employee data is sorted by department ID and then by employee ID. However, the department ID now displays at the top of the page.



```
                 Employee ID       Name
Department   100
                       102         Whitney, Fran
                       105         Cobb, Matthew
                       160         Breault, Robert
                       243         Shishov, Natasha
                       247         Driscoll, Kurt
                       249         Guevara, Rodrigo
                       266         Gowda, Ram
                       278         Melkisetian, Terry
                       316         Pastor, Lynn
                       445         Lull, Kim
                       453         Rabkin, Andrew
                       479         Siperstein, Linda
                       501         Scott, David
```

2 **Click the *Next* and *Prior* buttons to scroll through the data.**

Notice the group breaks.

3 **Click the *Design* button.**

You return to the DataWindow painter workspace.

Make cosmetic changes

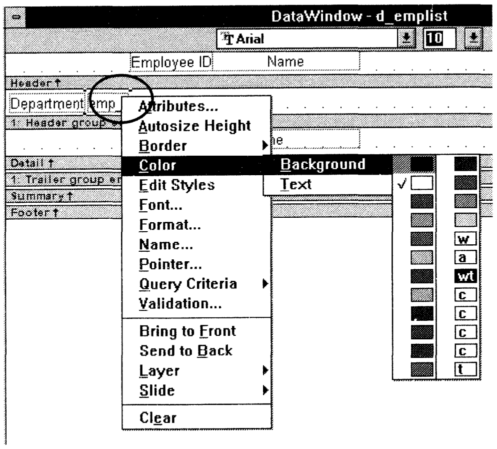
Where you are
Lesson 6 Building the First DataWindow Object
Create the new DataWindow object
Preview the DataWindow object
Save the DataWindow object
Create a group
➡ Make cosmetic changes

You can now make cosmetic changes to the DataWindow.

This example changes the background color and the colors used to display the department ID.

1 **Move the pointer to the *emp_dept_id* column in the group header band and right-click.**
Select *Color* from the popup menu.
Select *Background*.

The color menu displays.



2 **Select the *yellow* box in the color menu.**

The background of the column immediately turns yellow.

3 **Move the pointer back to the *emp_dept_id* column in the group header band and right-click.**
Select *Color* from the popup menu.
Select *Text*.

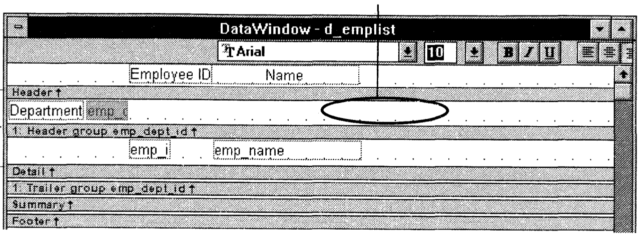
4 **Select the *dark blue* box in the color menu.**

The text of the column immediately turns dark blue.

5 **Move the pointer anywhere in the *Header band*, *Group Header band*, or *Detail band*, being careful not to point at any items or click in the bars.**

You might click in this area.

Don't click in the bars

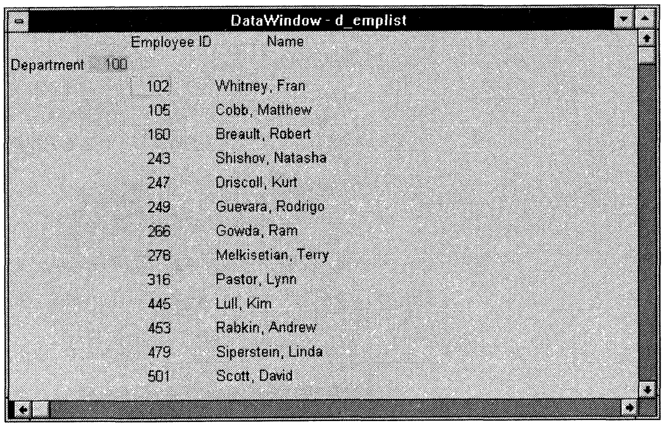


6 **Right-click.**
Select *Color* from the popup menu.

7 **Select the light gray box in the color menu.**

The DataWindow background immediately turns light gray.

8 Click the *Preview* button to see how it looks.



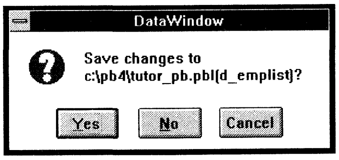
Employee ID	Name
Department 100	
102	Whitney, Fran
105	Cobb, Matthew
160	Breault, Robert
243	Shishov, Natasha
247	Driscoll, Kurt
249	Guevara, Rodrigo
266	Gowda, Ram
278	Melkisetian, Terry
316	Pastor, Lynn
445	Lull, Kim
453	Rabkin, Andrew
479	Siperstein, Linda
501	Scott, David

9 Click the *Design* button.

You return to the DataWindow painter workspace.

10 Select File➤Close from the menu bar.

PowerBuilder prompts you to save your changes.



Save changes to c:\pb4\tutor_pb.pbl(d_emplist)?

11 Click *Yes*.

PowerBuilder closes the DataWindow painter and returns to the Window painter.

At this point, you have a DataWindow object named d_emplist stored in the tutor_pb.pbl library. Next you will use this new object in your window.

LESSON 7

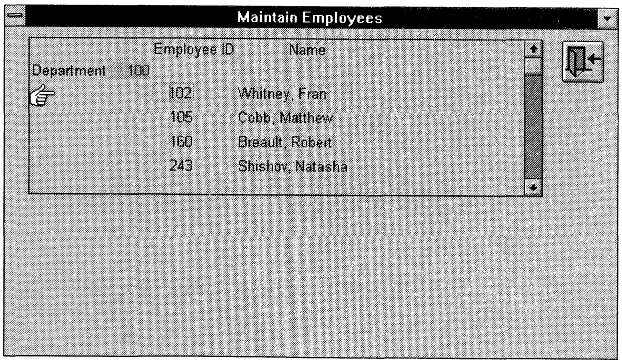
Adding the First DataWindow

After you create and save a DataWindow *object*, you can use it in a window. To do this, you place a DataWindow *control* in the window, associate the DataWindow object with it, and then write scripts for it.

In this lesson you will:

- ♦ Place a DataWindow control in the window you have built
- ♦ Associate the DataWindow object you built with the control
- ♦ Write a script for the window Open event to make the new DataWindow retrieve and display all the employees
- ♦ Update the script for the application Open event
- ♦ Add a script for the application Close event

When you finish this lesson, your application window will look like this during execution.



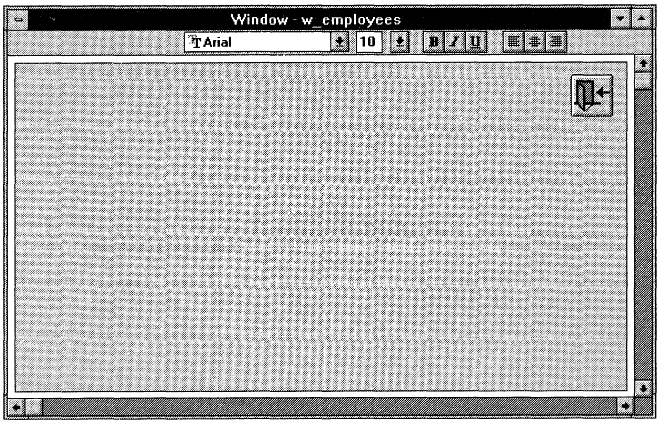
	Employee ID	Name
Department 100		
	102	Whitney, Fran
	105	Cobb, Matthew
	160	Breault, Robert
	243	Shishov, Natasha

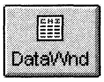
How long will this lesson take?
About 30 minutes.

Place the DataWindow control in the window

Where you are
Lesson 7 Adding the First DataWindow
➡ Place the DataWindow control in the window
Select the DataWindow object to display in the control
Specify attributes of the DataWindow control
Add a script for the Open event for the window
Assign values to SQLCA
Add a script for the Close event for the application
Run the application
Using script comments

1 **Make sure you are in the Window painter with the w_employees window displayed.**
If you are not, open the Window painter and select the *w_employees* window.





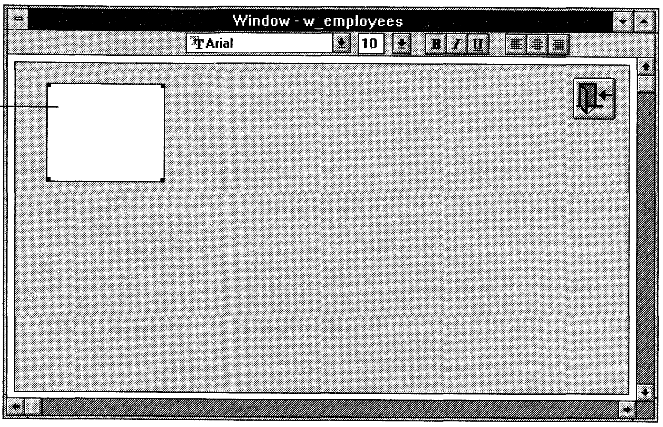
2 **Click the *DataWindow* button in the PainterBar (not in the PowerBar).**

PowerBuilder will place a DataWindow control at the next point you click in the workspace. A DataWindow control displays a DataWindow object in the window.

3 Click near the upper-left corner of your window.

A small rectangle displays. This is the DataWindow control.

DataWindow control



You can change the size and shape of the DataWindow control.

4 Make sure the control is selected (has black boxes in each corner).

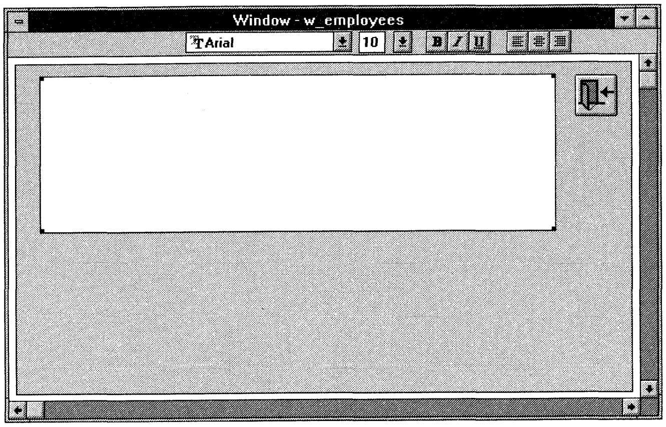
If it is not selected, click anywhere inside the DataWindow control to select it.

5 Move the pointer to the lower-right corner of the DataWindow control.

The pointer turns into a slanting, double-headed arrow.

6 **Hold down the mouse button and drag the control until it fills about half of your window's height and most of your window's width.**

Your screen should look like this now.

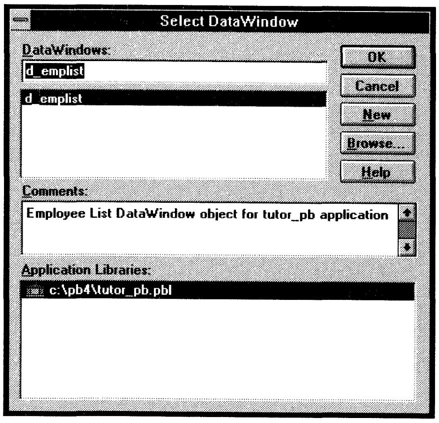


Select the DataWindow object to display in the control

Where you are
Lesson 7 Adding the First DataWindow
Place the DataWindow control in the window
➡ Select the DataWindow object to display in the control
Specify attributes of the DataWindow control
Add a script for the Open event for the window
Assign values to SQLCA
Add a script for the Close event for the application
Run the application
Using script comments

1 **Double-click inside the DataWindow control.**

The Select DataWindow dialog box displays.



d_emplist

d_emplist

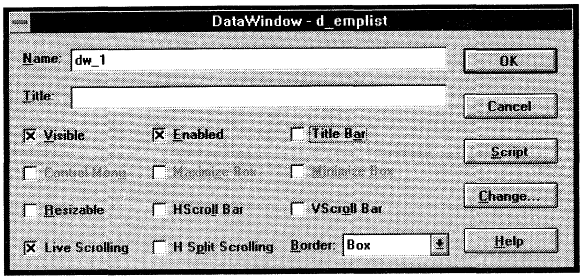
Employee List DataWindow object for tutor_pb application

c:\pb4\tutor_pb.pbl

The listbox at the top lists the DataWindow objects in the current library. At this point, there is only the DataWindow object you just built, d_emplist.

2 Click *OK*.

The DataWindow dialog box displays some of the attributes of the DataWindow control.



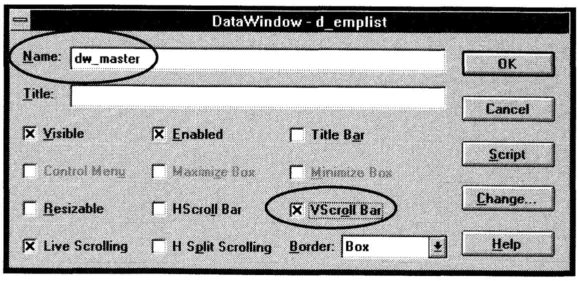
Specify attributes of the DataWindow control

As with most PowerBuilder objects and controls, you can change the visual and functional attributes of the DataWindow control. In this case, you just want to add a vertical scroll bar to make it easy to scroll up and down to see all the employees. Additionally, you will specify a meaningful name for the DataWindow control.

1 **Use the cursor to highlight the 1 of *dw_1* in the Name box. Type *master*.**

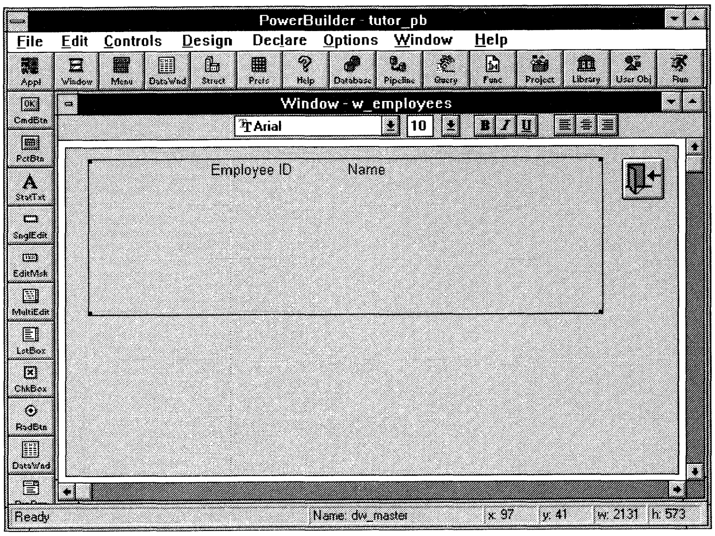
2 **Select the *VScroll Bar* checkbox.**

Do not change the default values of the other attributes.


DataWindow - d_emplist
Name: dw_master
Title:
[✓ Visible] [✓ Enabled] Title Bar
Control Menu Maximize Box Minimize Box
Resizable HScroll Bar [✓ VScroll Bar]
[✓ Live Scrolling] H Split Scrolling Border: Box

3 Click *OK.*

The Window painter displays with the DataWindow object named d_emplist in the dw_master DataWindow control.


PowerBuilder - tutor_pb
File Edit Controls Design Declare Options Window Help
Appl Window Menu DataWnd Struct Prefs Help Database Pipeline Query Func Project Library User Obj Run
CmdBtn PctBtn StatTxt SnglEdit EditMask MultiEdit LstBox ChkBox RadBtn DataWnd
Window - w_employees
Arial 10
Employee ID Name
Ready Name: dw_master x: 97 y: 41 w: 2131 h: 573

You see the headings. However, you do not see any group headings or data. The DataWindow will not execute its SELECT statement until you add some scripts and run the application. Also, you do not see the vertical scroll bar you requested; it displays only when you run the application.

About making the DataWindow object work
To make the DataWindow object work, you need to tell it at least two things:

- Where to obtain its data during execution
- What to do (for example, retrieve, or update)

Earlier, when you previewed the DataWindow object, you were in the PowerBuilder development environment. PowerBuilder obtained the information about database name, password, and other database variables from your PowerBuilder Preferences file (the PB.INI file).

But a user will not have the development environment and therefore will not have a PB.INI file. So you must provide the DataWindow object all the information it needs to access the database.

You can code all this information into your application. But that is often too inflexible. In practice, you will probably read the necessary information from a file or display a window for the user to enter the required information.

Or you can use a combination of both methods. For example, you can get the database name from a file and request the password in a window.

Passing database information to the DataWindow control
To pass required database information to a DataWindow control, you use a PowerBuilder transaction object. PowerBuilder provides a default transaction object. You can easily create more if you need them.

The default transaction object is named SQLCA, which stands for **SQL** **C**ommunications **A**rea. SQLCA is global (that is, it is available in any script) and it has several attributes (including database name and password).

A transaction object has attributes for sending information *to* the database and attributes for obtaining information (success or failure information for each database activity) *from* the database.

For more information about transaction objects, see *Building Applications*.

Add a script for the Open event for the window

Where you are
Lesson 7 Adding the First DataWindow
Place the DataWindow control in the window
Select the DataWindow object to display in the control
Specify attributes of the DataWindow control
➡ Add a script for the Open event for the window
Assign values to SQLCA
Add a script for the Close event for the application
Run the application
Using script comments

In this section you will write a script that tells the DataWindow object where to obtain its data and what database activities to perform. You will type some commands and use the Object browser to paste in other commands. Using the Object browser helps you to select the appropriate function and identify the necessary arguments.

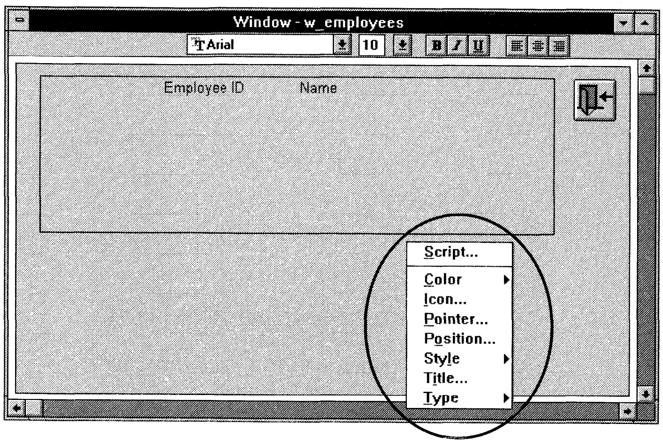
For more information on the Object browser, see the *User's Guide.*

The location for this script is in the Open event for the window (*not* the DataWindow control). That is because you want the DataWindow to retrieve all its data when the window opens.

By the time the window actually displays on the screen, all the employee names will be ready for display.

1 **Move the pointer to an unused space in your window and right-click.**

The popup menu for the window displays.



2 **Click *Script*.**

The PowerScript painter opens for the window.

3 **Make sure you are writing a script for the Open event for the window.**

The title bar of the PowerScript painter should read:

Script - open for w_employees

If the title is incorrect
If the object is not w_employees, select Edit➤Select Object from the menu bar and select w_employees.

If the event is not the Open event, click the Select Event listbox and select Open.

4 **Type the first line of the script to connect to the database.**
Press ENTER to position the insertion point at the beginning of a new line.
Add an IF statement to check SQLCA.SQLCODE (an attribute of the transaction object) for successful completion (press ENTER at the end of each line).

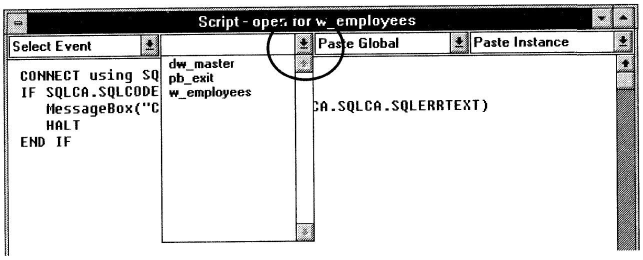
```
CONNECT using SQLCA;
IF SQLCA.SQLCODE <> 0 THEN
   MessageBox("Connect Error",SQLCA.SQLERRTEXT)
   HALT
END IF
```

PowerBuilder is not case sensitive, so use the case you prefer.

5 **Paste the next line of the script.**

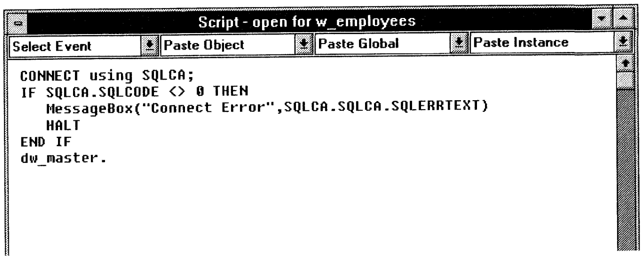
To paste, follow these directions:

1 **Press ENTER to position the insertion point at the beginning of the next line.**
Click the down arrow to the right of the Paste Object box.

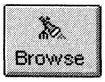


```
CONNECT using SQ
IF SQLCA.SQLCODE
   MessageBox("C
   HALT
END IF
```

2 **Click *dw_master*.**
Type a period.



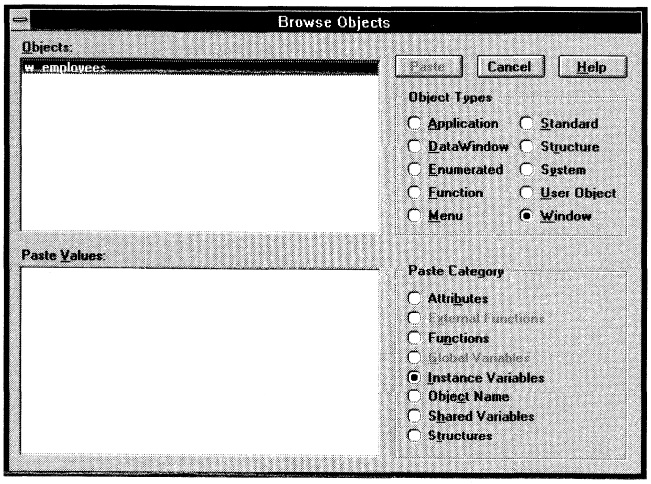
```
CONNECT using SQLCA;
IF SQLCA.SQLCODE <> 0 THEN
   MessageBox("Connect Error",SQLCA.SQLCA.SQLERRTEXT)
   HALT
END IF
dw_master.
```



3 **Click the *Browse* button.**
or
Select Edit➤Browse Objects from the menu bar.

The Browse Objects dialog box opens and displays the windows in the current application.

Make sure you are in Browse Objects
Make sure that you are in the Browse Objects (plural) dialog box and not the Browse Object (singular) dialog box, which displays attributes and functions for the current object only (in this case the w_employees window).

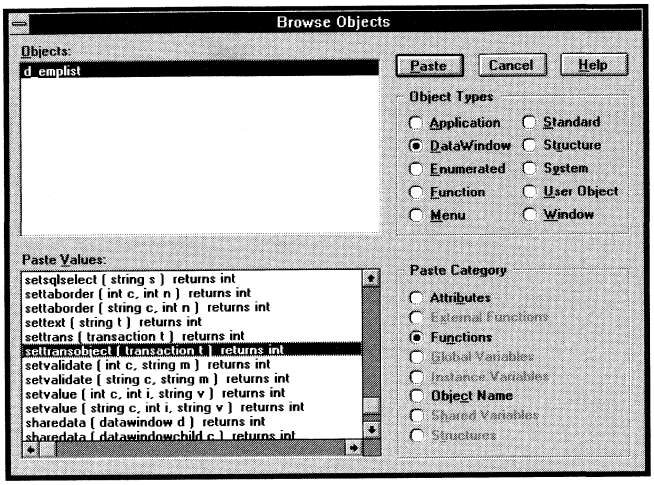


4 **Click the window name (w_employees) if it is not selected.**

5 **Click *DataWindow* in the Object Types box.**
Click *Functions* in the Paste Category box.

DataWindow functions display in the Paste Values box.

6 Scroll to the *SetTransObject* function and double-click it.



Objects:
d_emplist

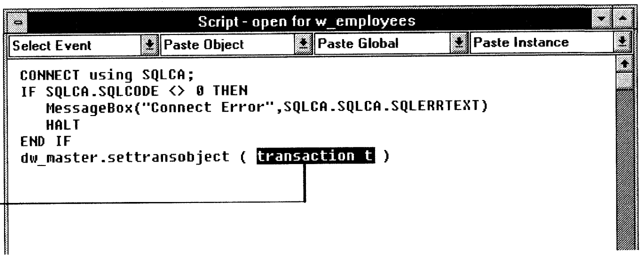
Object Types
[ ] Application [ ] Standard
[✓ DataWindow] [ ] Structure
[ ] Enumerated [ ] System
[ ] Function [ ] User Object
[ ] Menu [ ] Window

Paste Values:

```
setsqlselect ( string s ) returns int
settaborder ( int c, int n ) returns int
settaborder ( string c, int n ) returns int
settext ( string t ) returns int
settrans ( transaction t ) returns int
settransobject ( transaction t ) returns int
setvalidate ( int c, string m ) returns int
setvalidate ( string c, string m ) returns int
setvalue ( int c, int i, string v ) returns int
setvalue ( string c, int i, string v ) returns int
sharedata ( datawindow d ) returns int
sharedata ( datawindowchild c ) returns int
```

Paste Category
[ ] Attributes
[ ] External Functions
[✓ Functions]
[ ] Global Variables
[ ] Instance Variables
[ ] Object Name
[ ] Shared Variables
[ ] Structures

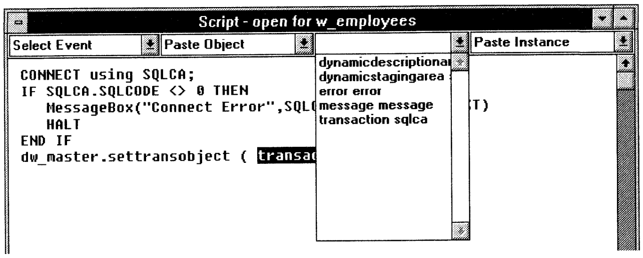
The SetTransObject function and its prototype parameter display in your script.



```
CONNECT using SQLCA;
IF SQLCA.SQLCODE <> 0 THEN
   MessageBox("Connect Error",SQLCA.SQLCA.SQLERRTEXT)
   HALT
END IF
dw_master.settransobject ( transaction t )
```

7 Click the down arrow to the right of *Paste Global.*

The list of global variables displays.

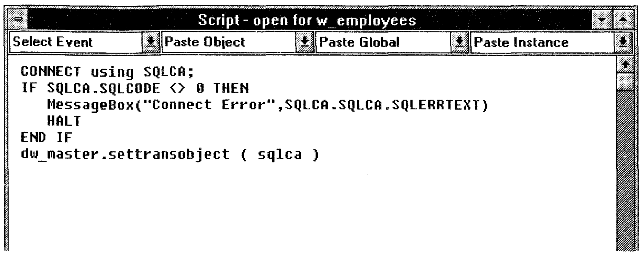


```
CONNECT using SQLCA;
IF SQLCA.SQLCODE <> 0 THEN
   MessageBox("Connect Error",SQLCA.SQLCA.SQLERRTEXT)
   HALT
END IF
dw_master.settransobject ( transac
```

```
dynamicdescriptiona
dynamicstagingarea
error error
message message
transaction sqlca
```

8 Click ***transaction sqlca.***

The variable sqlca displays as the argument in the SetTransObject function.

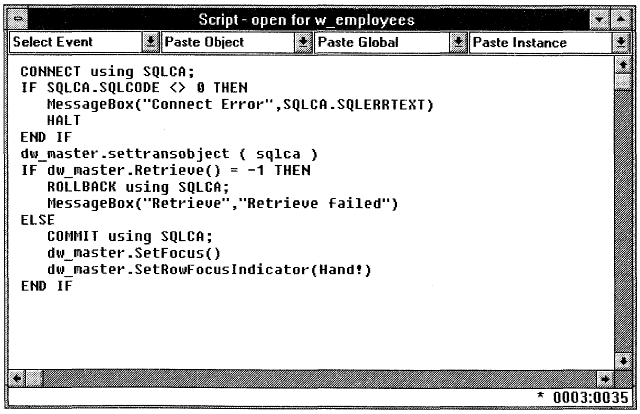


```
CONNECT using SQLCA;
IF SQLCA.SQLCODE <> 0 THEN
   MessageBox("Connect Error",SQLCA.SQLCA.SQLERRTEXT)
   HALT
END IF
dw_master.settransobject ( sqlca )
```

6 Type the next lines of the script:

```
IF dw_master.Retrieve () = -1 THEN
   ROLLBACK using SQLCA;
   MessageBox("Retrieve","Retrieve failed")
ELSE
   COMMIT using SQLCA;
   dw_master.SetFocus()
   dw_master.SetRowFocusIndicator(Hand!)
END IF
```

Your completed script should read like this:



```
CONNECT using SQLCA;
IF SQLCA.SQLCODE <> 0 THEN
   MessageBox("Connect Error",SQLCA.SQLERRTEXT)
   HALT
END IF
dw_master.settransobject ( sqlca )
IF dw_master.Retrieve() = -1 THEN
   ROLLBACK using SQLCA;
   MessageBox("Retrieve","Retrieve failed")
ELSE
   COMMIT using SQLCA;
   dw_master.SetFocus()
   dw_master.SetRowFocusIndicator(Hand!)
END IF
```

PowerScript summary

```
CONNECT using SQLCA;
```

This line uses the SQLCA transaction object to connect to the database. In just a few minutes you will write a script for the Open event *for the application* that puts values into SQLCA.

```
IF SQLCA.SQLCODE <> 0 THEN
   MessageBox("Connect Error",SQLCA.SQLERRTEXT)
   HALT
END IF
```

These lines check whether the CONNECT statement was successful. If the connect fails, it displays a message box and exits the application.

```
dw_master.settransobject ( sqlca )
```

This line tells the DataWindow object (in the control dw_master) to look in the transaction object named SQLCA for the database information such as database name and password.

```
IF dw_master.Retrieve () = -1 THEN
   ROLLBACK using SQLCA;
   MessageBox("Retrieve","Retrieve failed")
ELSE
   COMMIT using SQLCA;
   dw_master.SetFocus()
   dw_master.SetRowFocusIndicator(Hand!)
END IF
```

These lines execute the DataWindow's Retrieve function, which performs the SQL Select statement associated with the DataWindow object and places the retrieved rows in the dw_master DataWindow control. If the Retrieve succeeds, it performs a database commit, sets focus to the first row in the DataWindow control, and uses the enumerated variable Hand! to establish the hand pointer as the current row indicator. If the Retrieve fails, it performs a database rollback and displays a message.

For more information on enumerated data types, Commit, and Rollback, see the *PowerScript Language* manual. For more information on the MessageBox, SetFocus, and SetRowFocusIndicator functions, see the *Function Reference*.

Since this script executes as the window opens, the employees will show in the DataWindow control when the window first displays.

7 **Click the *Return* button in the PainterBar.**
or
Select File➤Return from the menu bar.

PowerBuilder compiles the script. If there are errors, an error window opens at the bottom of the screen. Correct your errors, then click the Return button again. You return to the Window painter.

8 **Select File➤Close from the menu bar.**

PowerBuilder displays the Save Changes message box.

9 **Click *Yes*.**

PowerBuilder closes the Window painter and returns you to the PowerBar.

Now it is time to assign values to SQLCA.

Assign values to SQLCA

Where you are
Lesson 7 Adding the First DataWindow
Place the DataWindow control in the window
Select the DataWindow object to display in the control
Specify attributes of the DataWindow control
Add a script for the Open event for the window
➡ Assign values to SQLCA
Add a script for the Close event for the application
Run the application
Using script comments

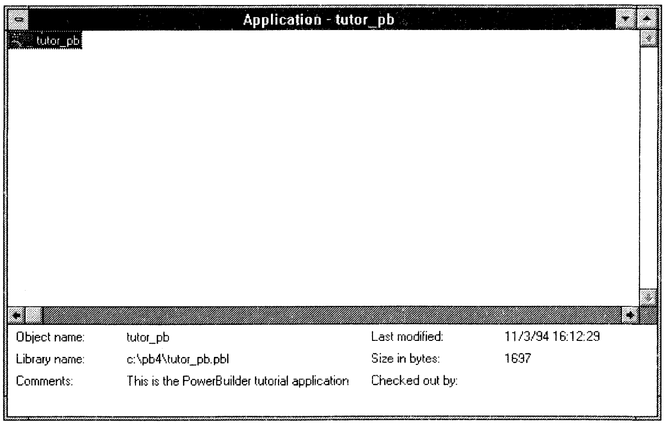
Since SQLCA is global, it may be used by many windows. So it is a good idea to get its values filled in at the very beginning of your application (that is, in the script for the application's Open event).

In this tutorial, the script will obtain the required information from the PB.INI file by using the ProfileString function.

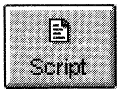
You can use an application-specific file instead of PB.INI
Even though the data is coming from PB.INI in this example, the ProfileString technique is the same as if you read from an application-specific file such as APPL.INI.

1 **Click the *Application painter* button in the PowerBar.**

The Application painter workspace displays. The tutor_pb application is the current application.

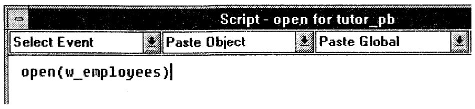


Object name:	tutor_pb	Last modified:	11/3/94 16:12:29
Library name:	c:\pb4\tutor_pb.pbl	Size in bytes:	1697
Comments:	This is the PowerBuilder tutorial application	Checked out by:	



2 **Click the *PowerScript* button in the PainterBar.**

The PowerScript painter displays the one-line script you entered earlier.



```
open(w_employees)
```

Since the script for the Open event for the window uses SQLCA, you need to assign values to SQLCA before opening the window.

The two lines you will enter use the ProfileString function to pass values for the DBMS and DbParm attributes of SQLCA. These are the two values the Watcom SQL database needs at connect time.

Each database has specific requirements
The values needed at connect time depend on the database you are connecting to.

For more information, see *Connecting to Your Database.*

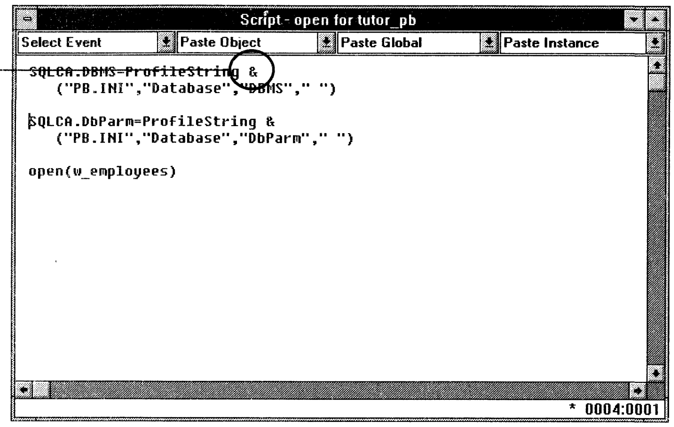
3 **Position the insertion point (I-beam) before *open(w_employees)* in the script.**
Press ENTER twice to make some room.

This adds two empty lines before the open statement.

4 **Type the new lines exactly as they appear below.**

```
SQLCA.DBMS=ProfileString &
   ("PB.INI","Database","DBMS"," ")
SQLCA.DbParm=ProfileString &
   ("PB.INI","Database","DbParm"," ")
```

Ampersand is the continuation character



```
SQLCA.DBMS=ProfileString &
   ("PB.INI","Database","DBMS"," ")

SQLCA.DbParm=ProfileString &
   ("PB.INI","Database","DbParm"," ")

open(w_employees)
```

The ProfileString functions look in the [Database] section of the PB.INI file to determine the DBMS and DbParm attributes.

PowerScript continuation character
Ampersand (&) is the PowerScript continuation character. If you want to use more than one line for a statement, you must use the continuation character.

5 **Click the *Return* button.**
or
Select File➤Return from the menu bar.

PowerBuilder compiles the script. If there are errors, an error window opens at the bottom of the screen. Correct your errors, then select Return again. You return to the Application painter.

If you have trouble compiling
You can comment out the lines in your script that do not compile.
See "Using script comments" on page 149.

Add a script for the Close event for the application

In this section you will write a script that tells the application to disconnect from the Database.

1 **Click the *PowerScript* button in the PainterBar.**

The PowerScript painter displays the Open event script you entered earlier.

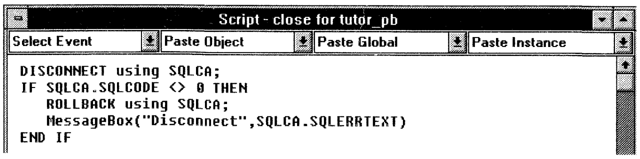
2 **Click the down arrow next to Select Event at the top of the PowerScript painter and select *Close*.**

The title bar of the PowerScript painter should read:

Script - close for tutor_pb

3 **Type the following lines exactly as they appear here.**

```
DISCONNECT using SQLCA;
IF SQLCA.SQLCODE <> 0 THEN
   ROLLBACK using SQLCA;
   MessageBox("Disconnect",SQLCA.SQLERRTEXT)
END IF
```



```
DISCONNECT using SQLCA;
IF SQLCA.SQLCODE <> 0 THEN
   ROLLBACK using SQLCA;
   MessageBox("Disconnect",SQLCA.SQLERRTEXT)
END IF
```

4 Click the *Return* button.
or
Select File➤Return from the menu bar.

PowerBuilder compiles the script. If there are errors, an error window opens at the bottom of the screen. Correct your errors, then select Return again. You return to the Application painter.

If your script continues to have errors, you can turn it into comments to leave the PowerScript painter. For how, see "Using script comments" on page 149.

PowerScript summary

```
DISCONNECT using SQLCA;
```
This line uses the SQLCA transaction object to disconnect from the database.

```
IF SQLCA.SQLCODE <> 0 THEN
   ROLLBACK using SQLCA;
   MessageBox("Disconnect",SQLCA.SQLERRTEXT)
END IF
```
These lines check whether the DISCONNECT statement was successful. If the disconnect fails, it performs a database rollback and displays a message box.

DISCONNECT includes COMMIT functionality
The Disconnect statement performs an implicit Commit for the specified transaction object, there is no need to code a Commit statement in this script.

Run the application

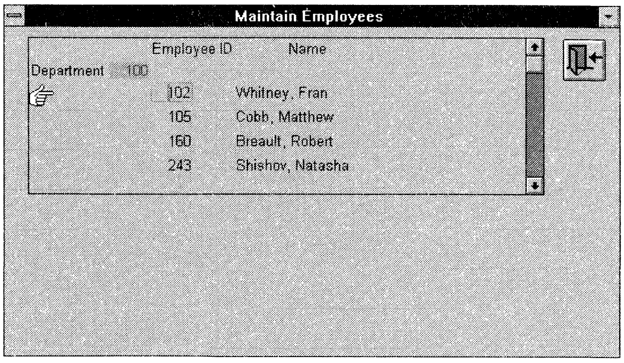
Where you are
Lesson 7 Adding the First DataWindow
Place the DataWindow control in the window
Select the DataWindow object to display in the control
Specify attributes of the DataWindow control
Add a script for the Open event for the window
Assign values to SQLCA
Add a script for the Close event for the application
➡ Run the application
Using script comments

1 **Click the *Run* button in the PowerBar.**
or
Select File➤Run from the menu bar.

PowerBuilder prompts you to save your changes.

2 **Click *Yes.***

Your window opens and displays a list of employees. You can scroll up and down to see them all. Your application has just connected to the database and retrieved information.



3 **Click the *Exit* PictureButton.**

You return to the Application painter.

4 **Select File➤Close from the menu bar.**

PowerBuilder closes the Application painter.

Using script comments

Where you are
Lesson 7 Adding the First DataWindow
Place the DataWindow control in the window
Select the DataWindow object to display in the control
Specify attributes of the DataWindow control
Add a script for the Open event for the window
Assign values to SQLCA
Add a script for the Close event for the application
Run the application
➡ Using script comments

PowerBuilder does not save a script until the script compiles successfully. So to save a script that does not yet compile, you can turn the script into comments. The compiler ignores the comments and saves the script. You can come back later, uncomment the code, and fix the problem.

A comment in PowerBuilder is indicated by either:

- Two forward slashes (//) at the start of a single-line comment.
- A slash and an asterisk (/*) at the start and an asterisk and a slash (*/) at the end of a single-line or multiline comment.

Commnt

To convert a line to a comment
Position the insertion point (I-beam) anywhere in the line and click the Comment button.

To convert a group of lines to comments
Drag through the lines selecting all or any part of them. Then click the Comment button.

Uncmnt

To "uncomment" a line
Position the insertion point anywhere in the line and click the Uncomment button.

To uncomment a group of lines
Drag through the commented lines selecting all or any part of them. Click the Uncomment button.

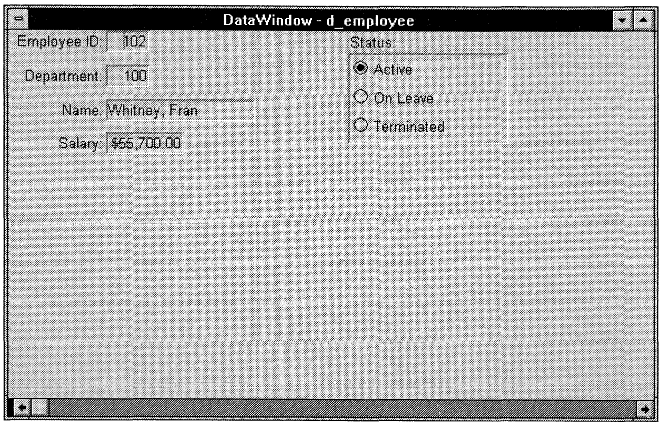
To select all lines for commenting or uncommenting
Click the Select All button or press **CTRL+A**.

LESSON 8

Building the Second DataWindow Object

In this lesson you will create a second DataWindow object to show the details for a selected employee.

When you finish this lesson and preview the DataWindow object, it will look like this.



Employee ID: 102
Department: 100
Name: Whitney, Fran
Salary: $55,700.00
Status:
● Active
○ On Leave
○ Terminated

How long will this lesson take?
About 20 minutes.

Create the new DataWindow object

Where you are
Lesson 8 Building the Second DataWindow Object
➡ Create the new DataWindow object
Preview the DataWindow object
Save the DataWindow object
Enhance the DataWindow object

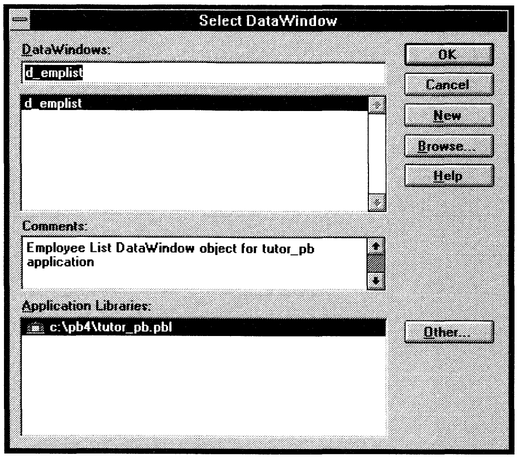
To create this DataWindow object, you will:

- ♦ Select the data source and style
- ♦ Select the table and columns
- ♦ Define a retrieval argument
- ♦ Specify a WHERE clause

Select the data source and style

1 **Click the *DataWindow painter* button in the PowerBar.**

The DataWindow painter opens and the Select DataWindow dialog box displays. It lists the d_emplist DataWindow, which is the one you created.



d_emplist

d_emplist

Employee List DataWindow object for tutor_pb application

c:\pb4\tutor_pb.pbl

2 **Click *New*.**

The New DataWindow dialog box displays.

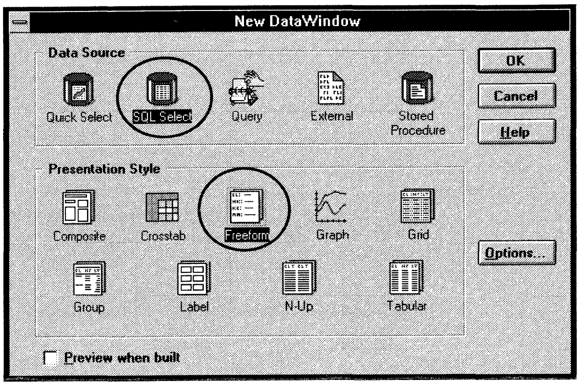
3 **Click *SQL Select* in the Data Source box.**

Choosing SQL Select will take you to the Select painter, where you can access SQL options.

In the first DataWindow, you chose Quick Select. This let you retrieve all the employees without actually going to the Select painter.

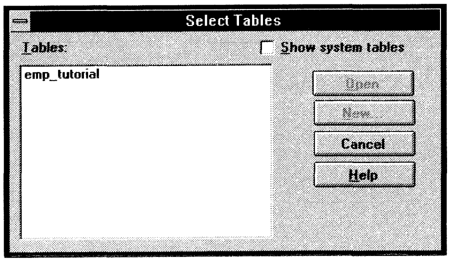
In this DataWindow, you want to retrieve data about the employee highlighted in the dw_master DataWindow control. You need to define a retrieval argument and WHERE criteria so you can pass an argument to the DataWindow during execution. In this case you will be passing the employee number. Retrieval arguments and WHERE criteria are defined in the Select painter.

4 Click *FreeForm* in the Presentation Style box.



5 Click *OK*.

Since the data source is SQL Select, you go to the Select painter and the Select Tables dialog box displays.

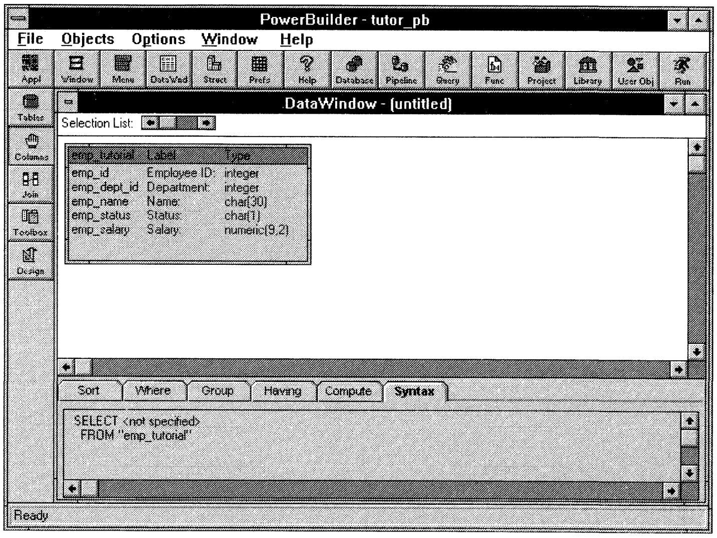


Select the table and columns

1 **Click *emp_tutorial* in the list of tables.**

2 **Click the *Open* button.**

The Select painter displays the table and its five columns.


PowerBuilder - tutor_pb
File Objects Options Window Help
Appl Window Menu DataWnd Struct Prefs Help Database Pipeline Query Func Project Library User Obj Run
Tables Columns Join Toolbox Design
DataWindow - (untitled)
Selection List:
Sort Where Group Having Compute Syntax
Ready

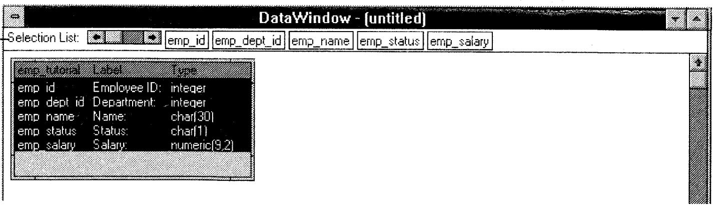
emp_tutorial	Label	Type
emp_id	Employee ID:	integer
emp_dept_id	Department:	integer
emp_name	Name:	char(30)
emp_status	Status:	char(1)
emp_salary	Salary:	numeric(9,2)

```
SELECT <not specified>
  FROM "emp_tutorial"
```

3 **Click each column name starting at the top with emp_id down to emp_salary.**

After you click a column name, the name is highlighted and displays in the Selection List area along the top of the workspace.

Selection list area


DataWindow - (untitled)
Selection List: emp_id emp_dept_id emp_name emp_status emp_salary

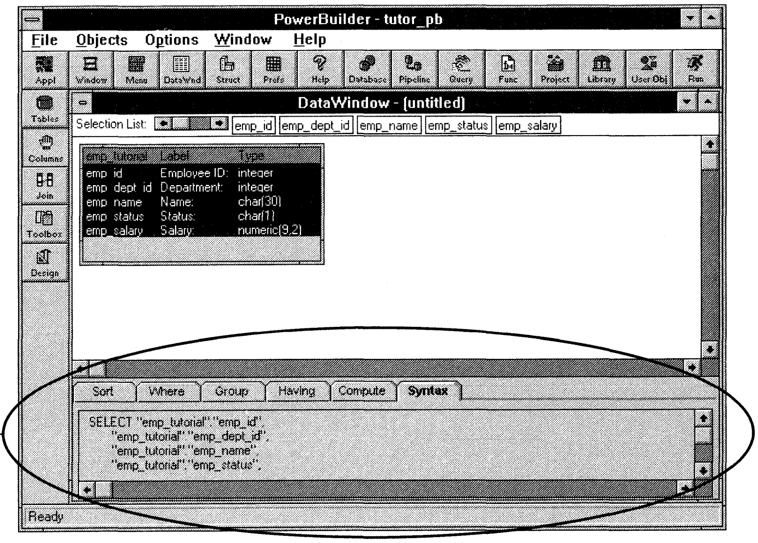
emp_tutorial	Label	Type
emp_id	Employee ID:	integer
emp_dept_id	Department:	integer
emp_name	Name:	char(30)
emp_status	Status:	char(1)
emp_salary	Salary:	numeric(9,2)

To quickly select all the columns
Point to the emp_tutorial heading area, click the right mouse button, and then click Select All. PowerBuilder highlights the columns in the table and displays them in the selection list.

Syntax

4 **Click the *Syntax* tab if it's not already on top.**

The generated SELECT statement displays in the tab area.


Tab area
PowerBuilder - tutor_pb
File Objects Options Window Help
Appl Window Menu DataWnd Struct Prefs Help Database Pipeline Query Func Project Library User Obj Run
Tables Columns Join Toolbox Design
DataWindow - (untitled)
Selection List: emp_id emp_dept_id emp_name emp_status emp_salary
Sort Where Group Having Compute Syntax
Ready

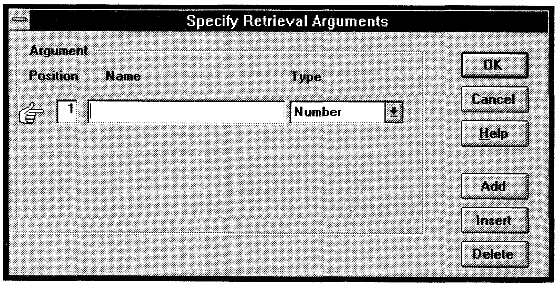
emp_tutorial	Label	Type
emp id	Employee ID:	integer
emp dept id	Department:	integer
emp name	Name:	char(30)
emp status	Status:	char(1)
emp salary	Salary:	numeric(9,2)

```
SELECT "emp_tutorial"."emp_id",
     "emp_tutorial"."emp_dept_id",
     "emp_tutorial"."emp_name",
     "emp_tutorial"."emp_status",
```

Define a retrieval argument

1 **Select Objects➢Retrieval Arguments from the menu bar.**

The Specify Retrieval Arguments dialog box displays.



2 **In the Name box, type** *employee_number*

The default data type is Number, which is what you want.

About retrieval argument names
You can choose any name you want for the retrieval argument; it is just a placeholder for the value you will pass during execution. Nonetheless, it is a good idea to make the name meaningful.

3 **Click** *OK.*

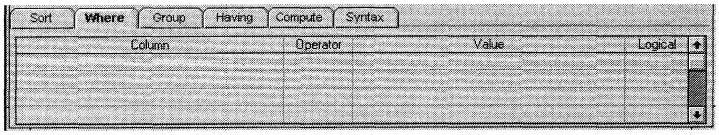
The retrieval argument is defined. Next you need to specify the WHERE clause that uses the retrieval argument employee_number.

Specify a WHERE clause

The WHERE clause uses the retrieval argument to retrieve a specific employee.

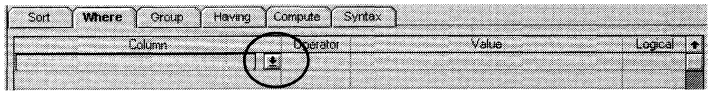
1 **Click the *Where* tab.**

The Where tab displays.



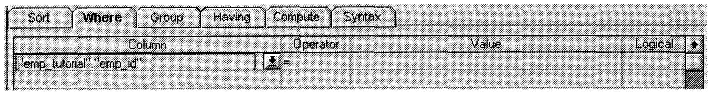
2 **Click in the box below Column.**

3 **Click the down arrow.**



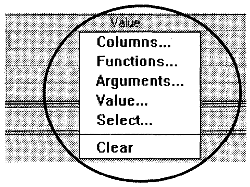
4 **Click *"emp_tutorial"."emp_id"*.**

It displays immediately and places an equal sign (=) in the Operator box. This is what you want, so do not change it.



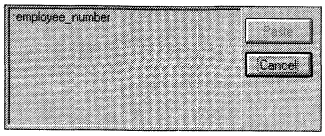
5 **Point in the box below *Value* and click the right mouse button.**

The popup menu displays.



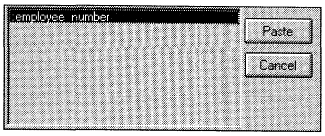
6 Click on *Arguments*.

The Arguments dialog box displays.



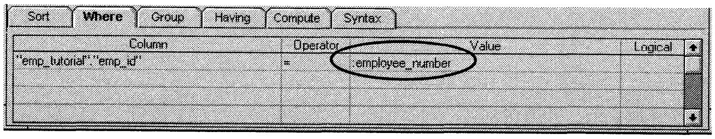
It contains the retrieval argument you specified earlier (employee_number) . The colon before employee_number indicates it is a PowerBuilder variable that can be used as an argument in a SQL statement.

7 Click *:employee_number*.



8 Click *Paste*.

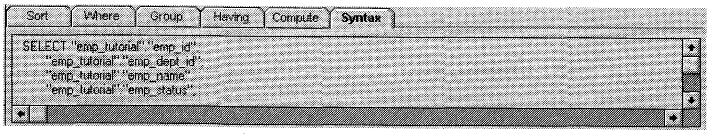
The Where tab displays with the retrieval argument.



Column	Operator	Value	Logical
"emp_tutorial"."emp_id"	=	:employee_number	

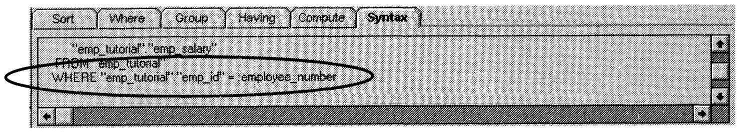
9 Click the *Syntax* tab.

The SELECT statement displays in the tab area.



```
SELECT "emp_tutorial"."emp_id",
     "emp_tutorial"."emp_dept_id",
     "emp_tutorial"."emp_name",
     "emp_tutorial"."emp_status",
```

10 Scroll down through the tab until you see the generated WHERE clause.

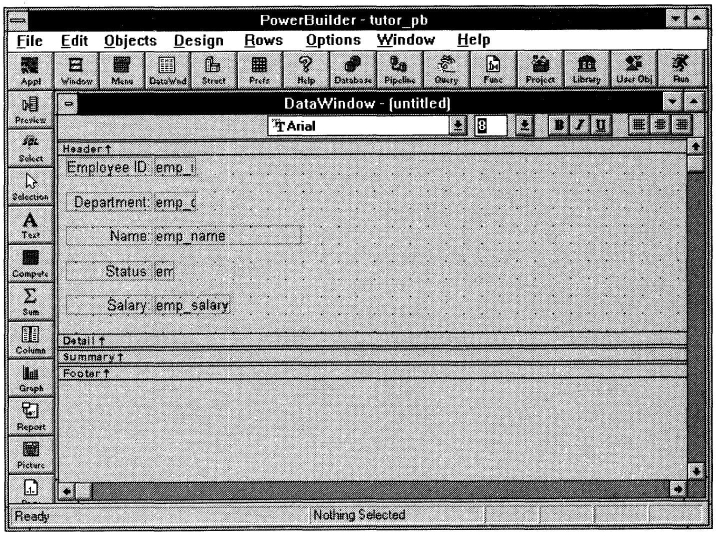


```
"emp_tutorial"."emp_salary"
FROM emp_tutorial
WHERE "emp_tutorial"."emp_id" = :employee_number
```

You have now created a complete SQL SELECT statement that says select all five columns from the emp_tutorial table where the emp_id column is equal to an argument that will be supplied during execution.

11 Click the *Design* button in the PainterBar.

PowerBuilder creates the new DataWindow object and displays it in the DataWindow painter workspace.



PowerBuilder uses the labels you specified in the extended attribute information in the Database painter. The labels are just the starting values; you can change them if you want.

Preview the DataWindow object

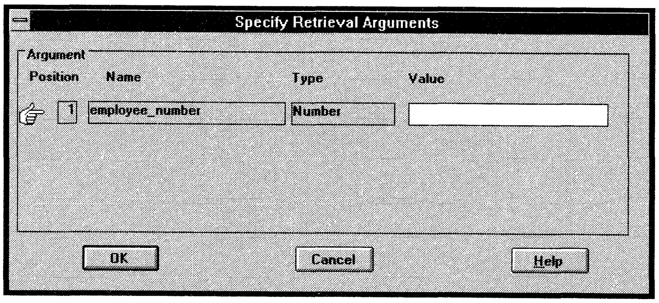
Where you are
Lesson 8 Building the Second DataWindow Object
Create the new DataWindow object
➡ Preview the DataWindow object
Save the DataWindow object
Enhance the DataWindow object

You can preview your new DataWindow object immediately.

1 **Click the *Preview* button in the PainterBar.**

Since this DataWindow object requires a retrieval argument, the Specify Retrieval Arguments dialog box displays.

PowerBuilder prompts for the argument. When you put this DataWindow into your application, you will write a script that will pass the required argument to the DataWindow automatically.


Specify Retrieval Arguments

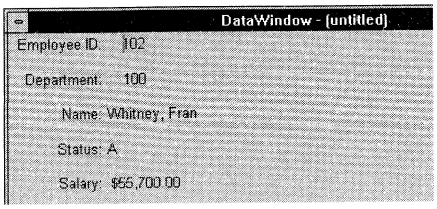
Argument

Position	Name	Type	Value
1	employee_number	Number	

PowerBuilder remembers your retrieval argument
The next time you preview the DataWindow, PowerBuilder will go directly to the DataWindow display, using the retrieval argument you specified the first time. PowerBuilder stores the argument until you leave the DataWindow painter. To access information for another employee, click the Retrieve button, which redisplays the Specify Retrieval Arguments dialog box.

2 **Type an employee number (for example, *102*, *105*, or *243*) and press** ENTER.

The DataWindow object connects to the database and retrieves the requested employee.



Employee ID: 102

Department: 100

Name: Whitney, Fran

Status: A

Salary: $55,700.00

3 **Click the *Design* button.**

You return to the DataWindow painter workspace.

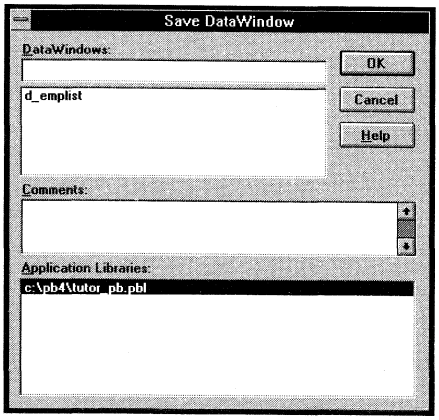
Save the DataWindow object

Where you are
Lesson 8 Building the Second DataWindow Object
Create the new DataWindow object
Preview the DataWindow object
➡ Save the DataWindow object
Enhance the DataWindow object

Now you will name the DataWindow object and save it. You could wait to save it until you are leaving the painter, but it's good practice to save your work frequently.

1 **Select File➤Save from the menu bar.**

The Save DataWindow dialog box displays.



2 **In the DataWindows box, type** *d_employee*
In the Comments box, type a comment.

3 **Click** *OK.*

You return to the DataWindow painter.

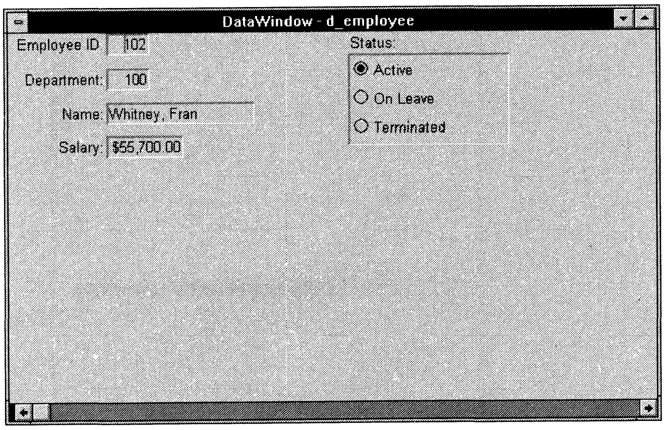
Enhance the DataWindow object

Where you are
<u>Lesson 8 Building the Second DataWindow Object</u>
Create the new DataWindow object
Preview the DataWindow object
Save the DataWindow object
➡ Enhance the DataWindow object

In this section you will modify the DataWindow object. You will:

- Rearrange the columns and text
- Define a RadioButton edit style for the Status column
- Define a 3D Lowered border style for all columns

When you are done, your DataWindow will look like this.

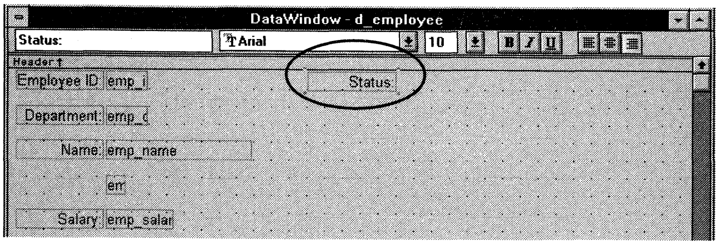


Employee ID: 102
Department: 100
Name: Whitney, Fran
Salary: $55,700.00
Status:
● Active
○ On Leave
○ Terminated

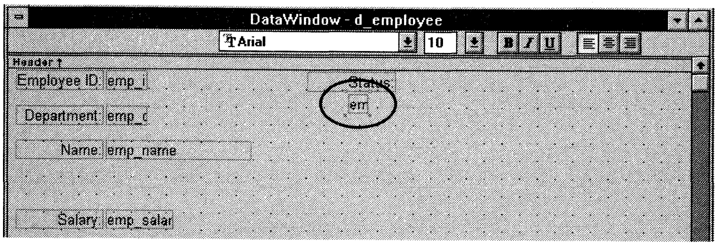
Columns in FreeForm DataWindows
Data fields on FreeForm style DataWindows are still called *columns*, even though they're shown in a nontabular display.

Rearrange the columns

1 **Click *Status:* and drag it to the top right of the DataWindow.**

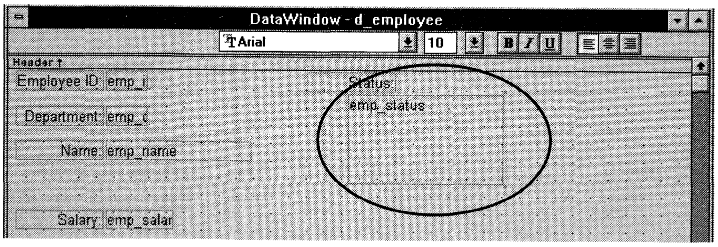


2 **Click the box that displays *em* and drag it under Status.**

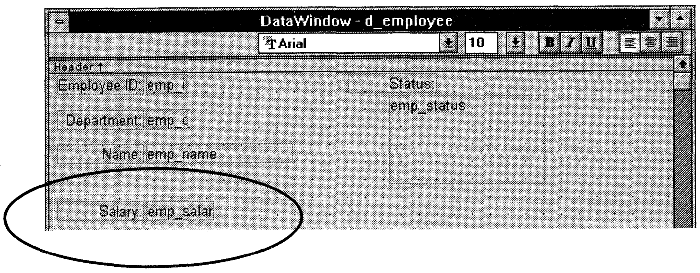


3 **Make this box bigger.**

Do this by moving the pointer over the lower-right corner until it turns into a slanting, two-headed arrow. Drag the lower-right corner of the box until it is about an inch and a half wide and an inch high. The complete name, emp_status, now displays.



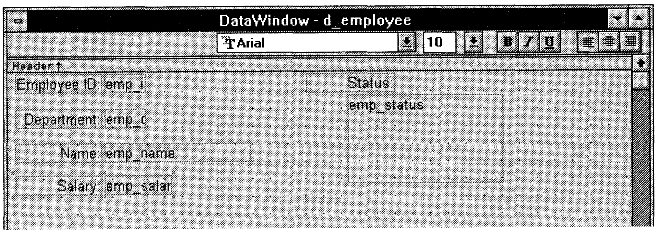
4 **Move the pointer above and to the left of the label *Salary:***
Press and hold the left mouse button.
Drag until the shaded box surrounds both the *Salary:* label and the *emp_salary* field.



5 **Release the left mouse button.**

The two items are selected. This is called lasso select. It's a handy technique for selecting multiple items.

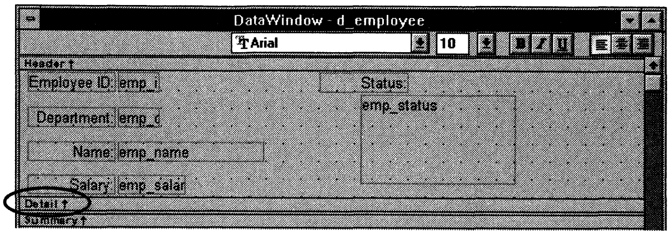
6 **Click either selected object and drag both objects up about half an inch so they are closer to the Name heading and emp_name column.**



Tip
You can also move selected items using the cursor keys.

7 Drag the horizontal bar named *Detail* until it is just under Salary.

This makes the entire DataWindow a little shorter.



Change the edit style

By default, DataWindow objects display the data they retrieve in edit boxes. But by using edit styles, you can display the data as radio buttons, checkboxes, dropdown listboxes, and dropdown DataWindows.

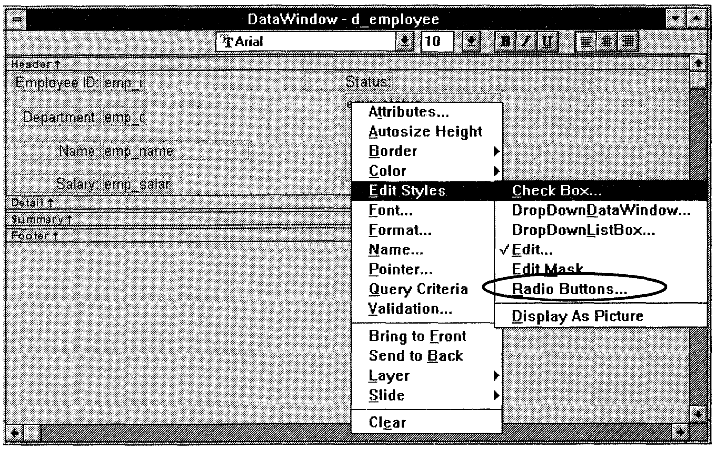
> **It's efficient to assign edit styles in the Database painter**
> In the Database painter, you can define and assign edit styles to columns. Then the DataWindow painter will use the edit styles automatically as the defaults whenever you generate a DataWindow that uses the columns. You can always override the defaults in the DataWindow painter.

In this DataWindow, you will change the edit style for emp_status from an edit box to radio buttons.

1 **Move the pointer to the *emp_status* box and right-click.**

The popup menu displays.

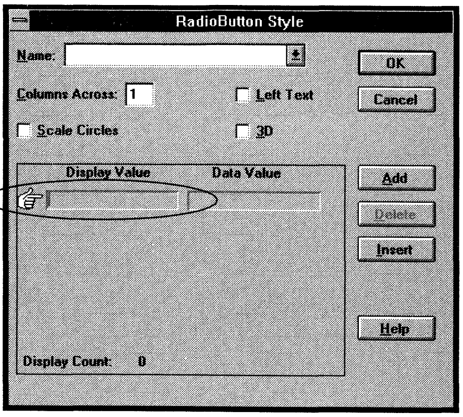
2 **Click *Edit Styles*.**
Then click *Radio Buttons*.



The RadioButton Style dialog box displays.

3 Click in the box under Display Value.

In this area you will create a code table. After typing each display value and data value press the DOWN ARROW key to display a line for the next value.

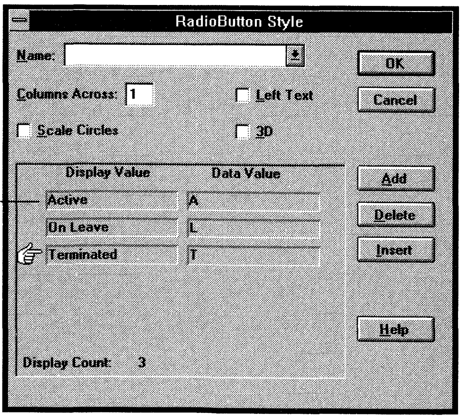


4 Type the following display and data values:

Display value	Data value
Active	A
On Leave	L
Terminated	T

The completed dialog box should match the following.

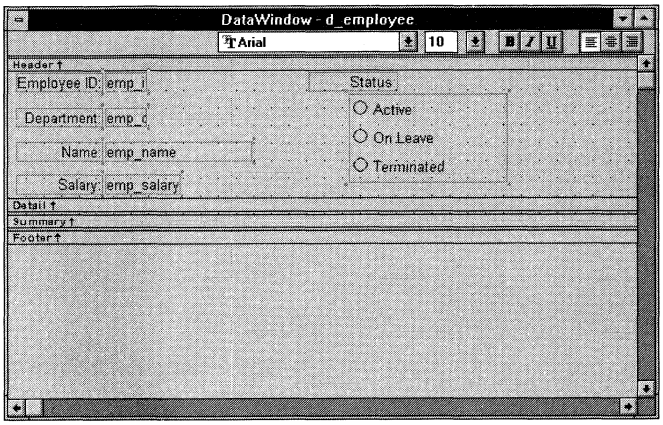
Press the DOWN ARROW *key between rows*



5 Click *OK*.

The display returns to the DataWindow painter workspace. The three display values you specified now display in the box under Status. You may need to change the size of the box to accommodate the radio buttons.

At this point, your DataWindow object should look approximately like this.



You just created a DataWindow object code table. Now, whenever the DataWindow object reads A from the database, it will mark the Active radio button as the current value.

The code table works in the other direction, too
For example, when a user selects the On Leave radio button, the next time the DataWindow object processes an update request, it will send L to the database.

Change the border style

By default, the columns displayed in a DataWindow have no border. By using border styles, you can enhance DataWindow display and emulate the look of many current applications.

In this DataWindow, you will change the border style for all columns from none to 3D Lowered.

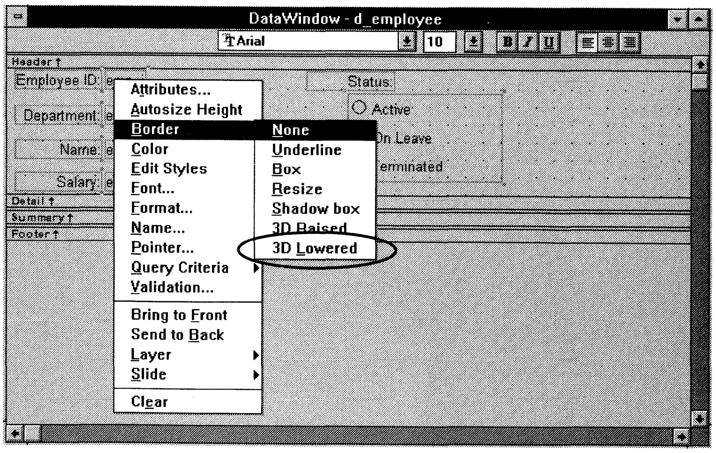
1 **Select Edit➤Select➤Select Columns from the menu bar.**

This selects all columns (leaving labels unselected).

> **You can also use CTRL+click**
> You can also use the CTRL+click technique to select multiple columns one by one.

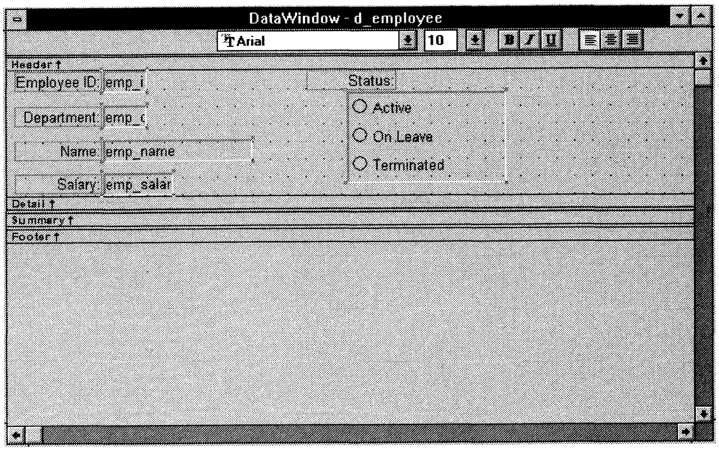
2 **Point to any of the selected columns and right-click.**

3 **Click *Border*.**



4 **Click** *3D Lowered.*

The DataWindow redisplays with the new border style.

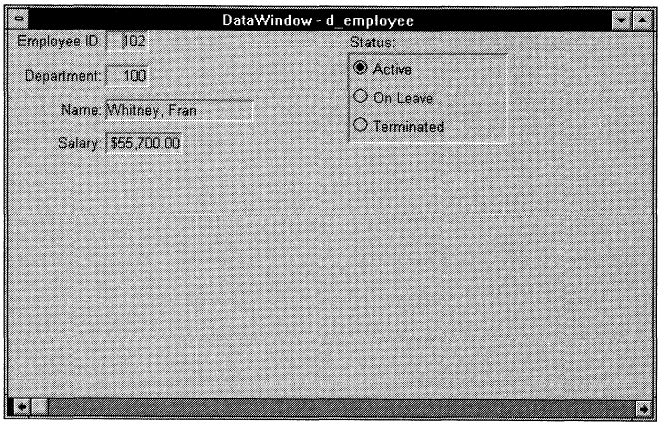


Preview again

1 **Click the *Preview* button in the PainterBar.**

The DataWindow connects to the database, uses the argument you specified the last time Preview was requested to execute the SELECT statement, and retrieves the specified employee.

> **You may need to specify a retrieval argument**
> If you have closed the DataWindow painter since the last Preview was requested, PowerBuilder will display the Specify Retrieval Arguments dialog box. If so, type an employee number (for example, 102, 105, or 243) and click OK.



DataWindow - d_employee

Employee ID: 102
Department: 100
Name: Whitney, Fran
Salary: $55,700.00

Status:
[✓ Active]
On Leave
Terminated

> **To change an employee's status**
> Click one of the radio buttons. Then click the Updt DB button in the PainterBar to update the database (if you're using PowerTips, the button text is Save Changes).

2 **Click the *Design* button.**

You return to the DataWindow painter workspace.

At this point, you could make many other changes to the DataWindow object. For example, you could change colors, add lines, remove columns, and change labels and display formats.

For this tutorial, the DataWindow object is ready to be used in the window you created earlier.

3 Select File➢Close from the menu bar.

PowerBuilder prompts you to save your changes.

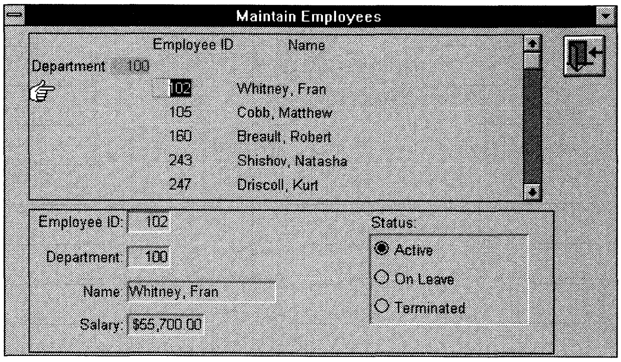
4 Click ***Yes.***

The DataWindow painter closes.

LESSON 9

Adding the Second DataWindow

In this lesson you will add the second DataWindow (control and object) to your window. When you finish this lesson, this is what your application window will look like during execution.



Maintain Employees

Department 100

Employee ID	Name
102	Whitney, Fran
105	Cobb, Matthew
160	Breault, Robert
243	Shishov, Natasha
247	Driscoll, Kurt

Employee ID: 102

Department: 100

Name: Whitney, Fran

Salary: $55,700.00

Status:
- [✓ Active]
- On Leave
- Terminated

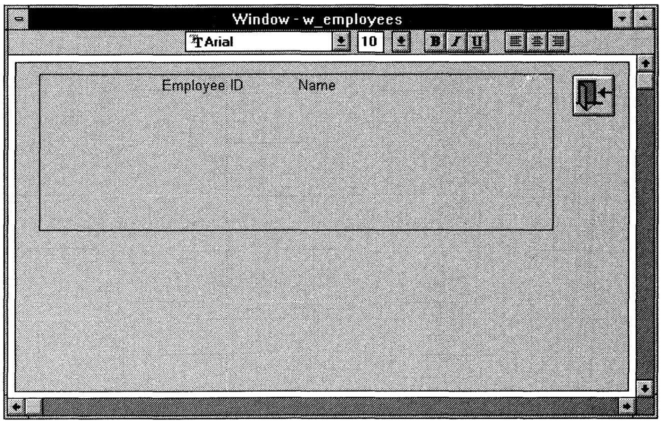
How long will this lesson take?
About 15 minutes.

Add the DataWindow control and object

Where you are
Lesson 9 Adding the Second DataWindow
➡Add the DataWindow control and object
Preview the window
Build a DataWindow script
Extend the script for the Open event for the window
Run the application

1 **Click the *Window painter* button in the PowerBar.**
Open the *w_employees* window.

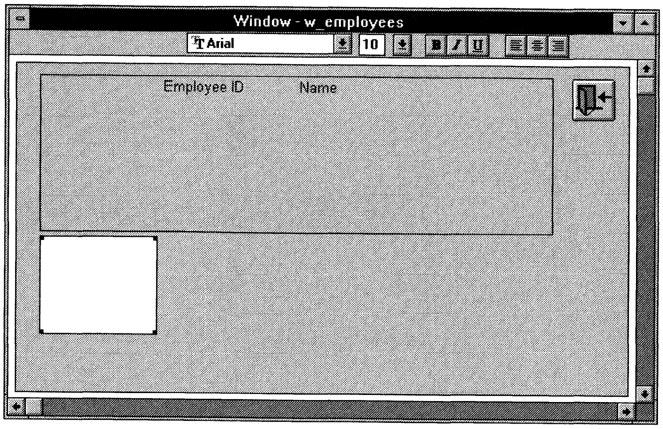
Your window displays with a DataWindow control and a PictureButton.



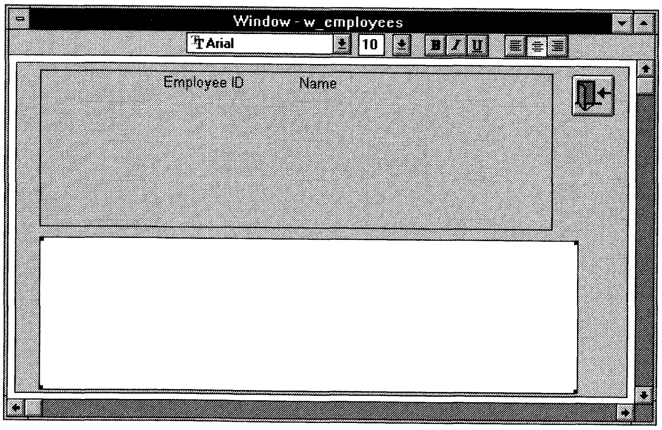
2 Click the *DataWindow* button in the PainterBar (not in the PowerBar).

3 Click below the first DataWindow near the left edge.

A rectangular DataWindow control displays.

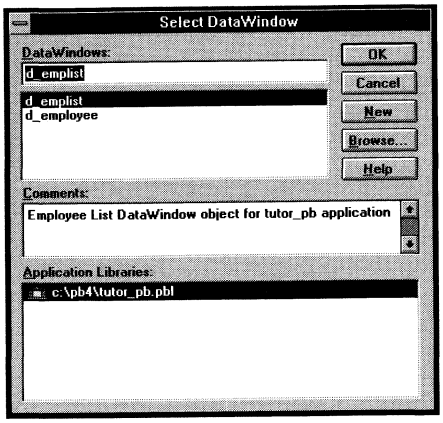


4 Change the shape of the control so that it looks like this.



5 Double-click inside the DataWindow control.

The Select DataWindow dialog box displays.

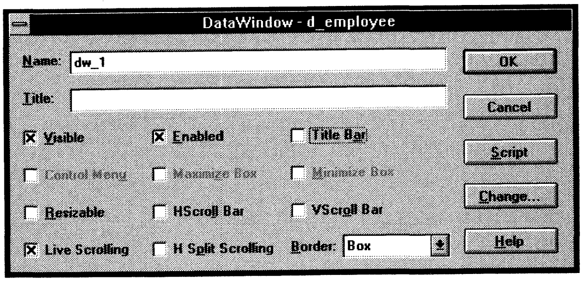


Employee List DataWindow object for tutor_pb application

Notice that both your DataWindow objects (d_emplist and d_employee) are stored in this library and listed.

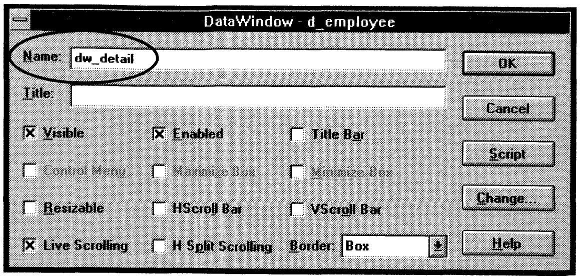
6 Double-click *d_employee* to select it.

The DataWindow dialog box displays some of the attributes of the DataWindow control.



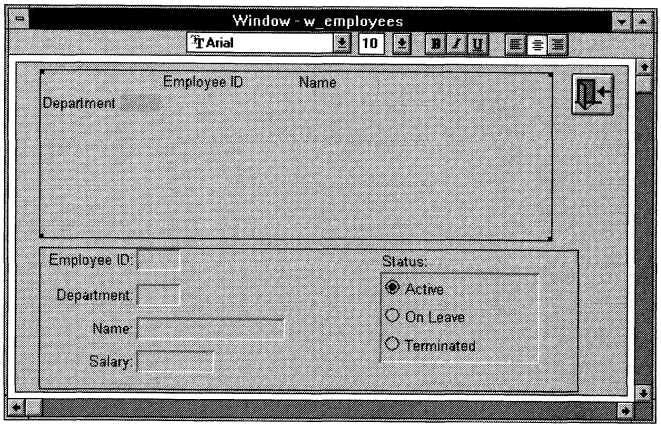
Notice that since your window already has a DataWindow control, PowerBuilder assigns this control the default name dw_1. You will change the name to dw_detail.

7 **Use the cursor to highlight the 1 of *dw_1* in the Name box. Type *detail*.**



8 **Click *OK*.**

The Window painter displays.



9 **Adjust the size and shape of the DataWindow control if necessary.**

If you do not see all the fields in the DataWindow control, make it bigger.

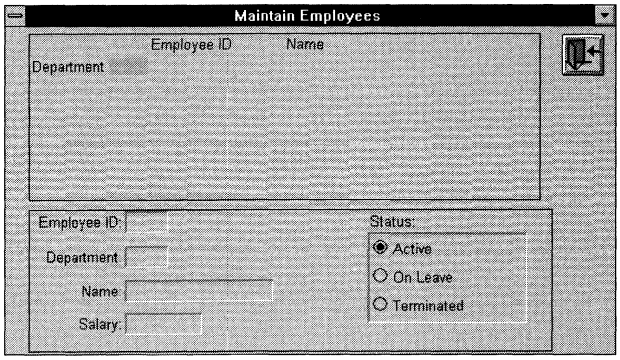
Preview the window

Where you are
Lesson 9 Adding the Second DataWindow
Add the DataWindow control and object
➡ Preview the window
Build a DataWindow script
Extend the script for the Open event for the window
Run the application

When you are in the Window painter workspace, you can preview your window to see how it looks.

1 **Select Design➤Preview from the menu bar.**

Your window displays and you can check its size, location, color, object placement, and other style features. However, this is strictly a visual preview; you cannot check how the scripts work.



2 **Select Design➤Preview from the menu bar again.**

The preview window closes and you return to the Window painter workspace.

Build a DataWindow script

Where you are
Lesson 9 Adding the Second DataWindow
Add the DataWindow control and object
Preview the window
➡ Build a DataWindow script
Extend the script for the Open event for the window
Run the application

In this section you will build a script to create a Master/Detail application. Here's how it will work when you are done:

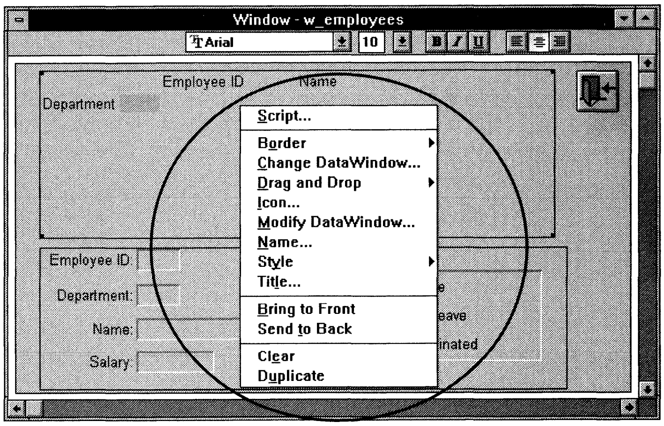
- When the window initially opens, a list of all employees displays in the top DataWindow control and detail information for the first employee displays in the bottom DataWindow control.
- When a user moves through the list of employees using the UP ARROW and DOWN ARROW keys or by clicking in a row, the details for the current employee display in the bottom DataWindow control.

To accomplish this, you will add a script for the RowFocusChanged event of the top DataWindow control (dw_master). That script will send a retrieval request and the ID number of the selected employee to the bottom DataWindow control (dw_detail).

RowFocusChanged also occurs upon DataWindow display
The RowFocusChanged event also occurs before the DataWindow is displayed. This allows your application to retrieve and display detail information for the first employee.

1 **Move the pointer to an unused area in the top DataWindow control and right-click.**

The popup menu for the DataWindow control displays.

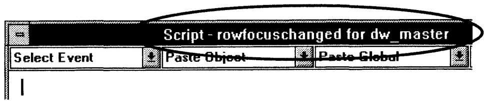


2 **Select *Script* from the popup menu.**

The PowerScript painter displays.

3 **Click the down arrow next to the Select Event listbox and select *rowfocuschanged*.**

The title should read: Script - rowfocuschanged for dw_master.



If the title is incorrect
If the object is not dw_master, select Edit➤Select Object from the menu bar and select dw_master.

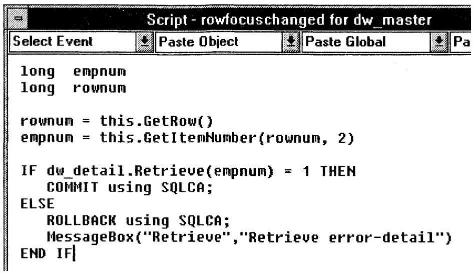
4 **Type the following script:**

```
long  empnum
long  rownum

rownum = this.GetRow()
empnum = this.GetItemNumber(rownum, 2)

IF dw_detail.Retrieve(empnum) = 1 THEN
  COMMIT using SQLCA;
```

```
ELSE
  ROLLBACK using SQLCA;
  MessageBox("Retrieve","Retrieve error-detail")
END IF
```



```
long  empnum
long  rownum

rownum = this.GetRow()
empnum = this.GetItemNumber(rownum, 2)

IF dw_detail.Retrieve(empnum) = 1 THEN
   COMMIT using SQLCA;
ELSE
   ROLLBACK using SQLCA;
   MessageBox("Retrieve","Retrieve error-detail")
END IF
```

5 **Click the *Return* button in the PainterBar.**
or
Select File➢Return from the menu bar.

PowerBuilder compiles the script. If there are errors, an error window opens at the bottom of the screen. Correct your errors, then click the Return icon again. You return to the Window painter.

PowerScript summary

`long empnum`
`long rownum`
The first two lines of this script declare local variables: empnum and rownum. (These variables, which are both the long data type, could have any names.)

`rownum = this.GetRow()`
The third line uses the GetRow function to determine the current row, as determined by the user either clicking in it or scrolling to it via arrow or tab keys. It stores the number of that row in the variable rownum. Also note the use of the This keyword, which refers to the current object (dw_master in this example).

```
empnum = this.GetItemNumber(rownum, 2)
```

The fourth line uses the GetItemNumber function to obtain the number in DataWindow dw_master that is in column 2 of the selected row. Since column 2 has employee numbers in it, this function gets the employee number for the current row. The employee number is stored in the variable empnum.

```
IF dw_detail.Retrieve(empnum) = 1 THEN
   COMMIT using SQLCA;
ELSE
   ROLLBACK using SQLCA;
   MessageBox("Retrieve","Retrieve error-detail")
END IF
```

This group of lines sends a retrieve request to DataWindow dw_detail along with the argument the DataWindow expects (an employee number). The IF statement that encloses the Retrieve function checks for successful completion. If the function succeeds, it performs a database commit; if the function fails, it performs a database rollback and displays a message box.

Production-level error checking
In a production application, you would also check SQLCA.SQLCODE after the COMMIT and ROLLBACK statements to ensure that they executed successfully.

Extend the script for the Open event for the window

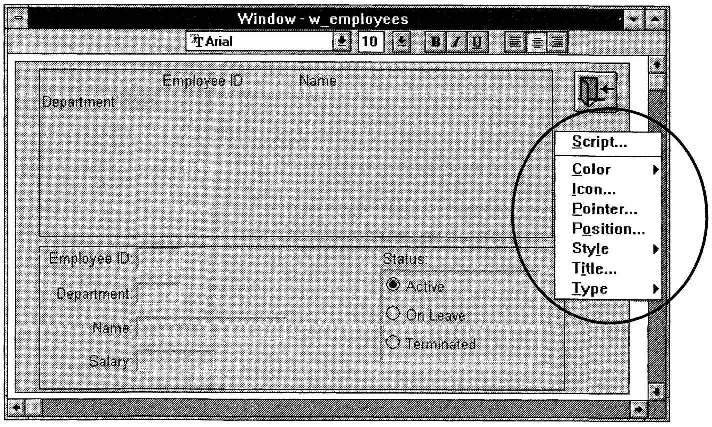
Where you are
Lesson 9 Adding the Second DataWindow
Add the DataWindow control and object
Preview the window
Build a DataWindow script
➡ Extend the script for the Open event for the window
Run the application

You also need to tell the second DataWindow where to look for the database variables (such as name and password). You can use the SQLCA transaction object again.

As with the first DataWindow, you use the script for the open event for the window to pass this information. You will extend the script you wrote earlier for the Open event for your window.

1 Move the pointer to a blank area of your window and right-click.

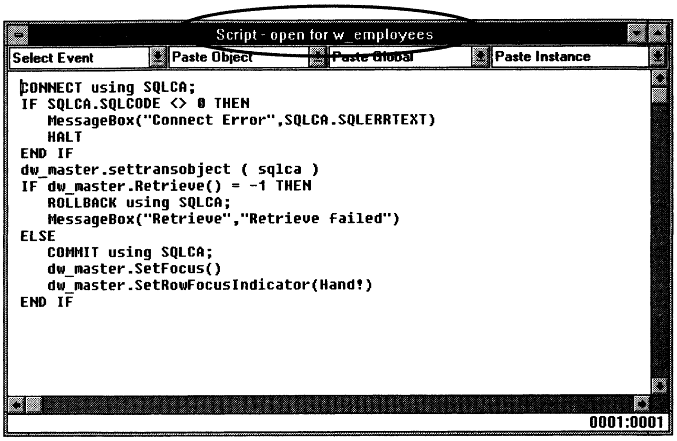
The popup menu for the window displays.



2 Select *Script* from the popup menu.

The PowerScript painter displays.

3 **Make sure the title reads:** *Script - open for w_employees*



```
CONNECT using SQLCA;
IF SQLCA.SQLCODE <> 0 THEN
   MessageBox("Connect Error",SQLCA.SQLERRTEXT)
   HALT
END IF
dw_master.settransobject ( sqlca )
IF dw_master.Retrieve() = -1 THEN
   ROLLBACK using SQLCA;
   MessageBox("Retrieve","Retrieve failed")
ELSE
   COMMIT using SQLCA;
   dw_master.SetFocus()
   dw_master.SetRowFocusIndicator(Hand!)
END IF
```

If the title is incorrect
If the object is not w_employees, select Edit➤Select Object from the menu bar and select w_employees.

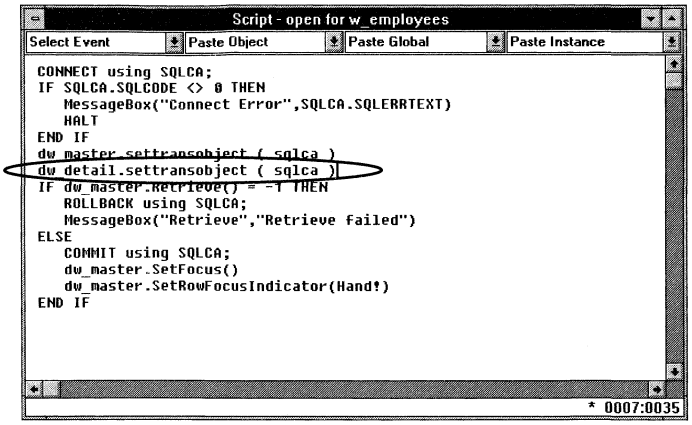
If the event is not the Open event, click the Select Event listbox and select Open.

You have already written a script for the Open event. Now you will add to it.

4 **Move the insertion point to the beginning of the line that reads** *IF dw_master.Retrieve () = -1 THEN.*
Press ENTER.
Press the UP ARROW.
Type this line:

```
dw_detail.settransobject ( sqlca )
```

This line tells the second DataWindow to look in the SQLCA transaction object for the values of the database variables.



```
CONNECT using SQLCA;
IF SQLCA.SQLCODE <> 0 THEN
   MessageBox("Connect Error",SQLCA.SQLERRTEXT)
   HALT
END IF
dw_master.settransobject ( sqlca )
dw_detail.settransobject ( sqlca )
IF dw_master.Retrieve() = -1 THEN
   ROLLBACK using SQLCA;
   MessageBox("Retrieve","Retrieve failed")
ELSE
   COMMIT using SQLCA;
   dw_master.SetFocus()
   dw_master.SetRowFocusIndicator(Hand!)
END IF
```

5 **Click the *Return* button.**
or
Select File➤Return from the menu bar.

PowerBuilder compiles the script and returns you to the Window painter.

Run the application

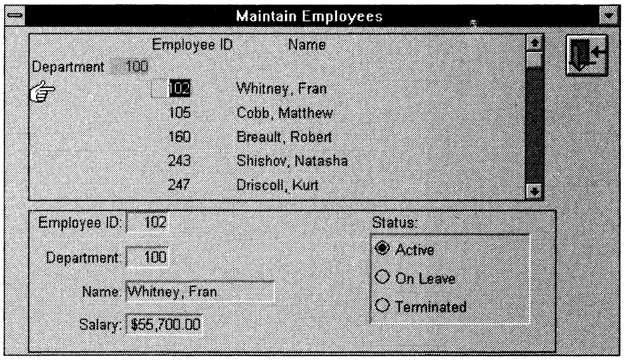
Where you are
Lesson 9 Adding the Second DataWindow
Add the DataWindow control and object
Preview the window
Build a DataWindow script
Extend the script for the Open event for the window
➡ Run the application

1 **Click the *Run* button in the PowerBar or the PainterBar.**
or
Select File➤Run from the menu bar.

PowerBuilder prompts you to save the current changes.

2 **Click *Yes*.**

Your window opens and displays the list of employees in the first department and detail information for the first employee in the list.



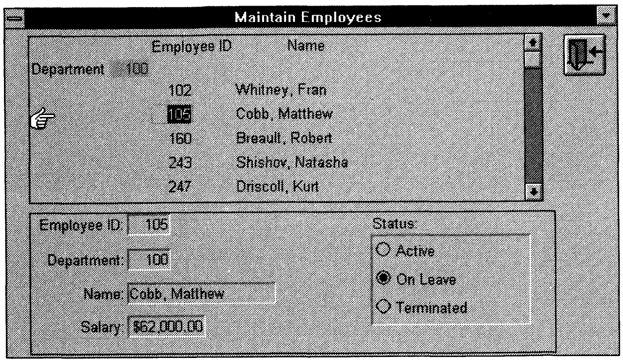
Employee ID	Name
Department 100	
102	Whitney, Fran
105	Cobb, Matthew
160	Breault, Robert
243	Shishov, Natasha
247	Driscoll, Kurt

Employee ID: 102
Department: 100
Name: Whitney, Fran
Salary: $55,700.00

Status:
[✓ Active]
On Leave
Terminated

3 Click another employee ID.

The second DataWindow retrieves the details for the employee you selected.



Employee ID	Name
Department 100	
102	Whitney, Fran
105	Cobb, Matthew
160	Breault, Robert
243	Shishov, Natasha
247	Driscoll, Kurt

Employee ID: 105
Department: 100
Name: Cobb, Matthew
Salary: $62,000.00
Status: Active [✓ On Leave] Terminated

4 Press PAGE DOWN **to display the next department (depending on the size of the dw_master DataWindow control, you may have to press Page Down several times).**

5 Click the *Exit* PictureButton.

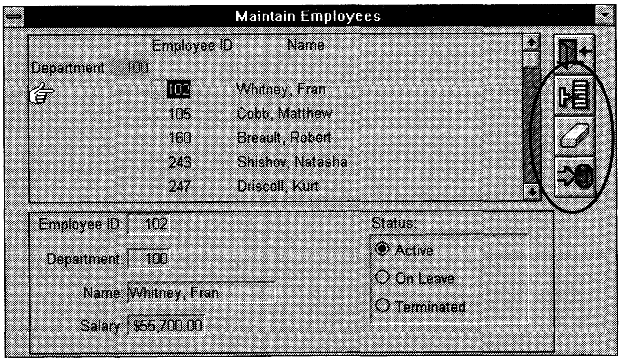
You return to the Window painter.

LESSON 10

Adding Database Maintenance Functionality

A DataWindow can also easily handle inserting, updating, and deleting rows in the database. In this lesson you will add PictureButtons for each of these database functions. Then you will write short scripts to make the buttons work.

When you finish this lesson and run your application, your application window will look like this.

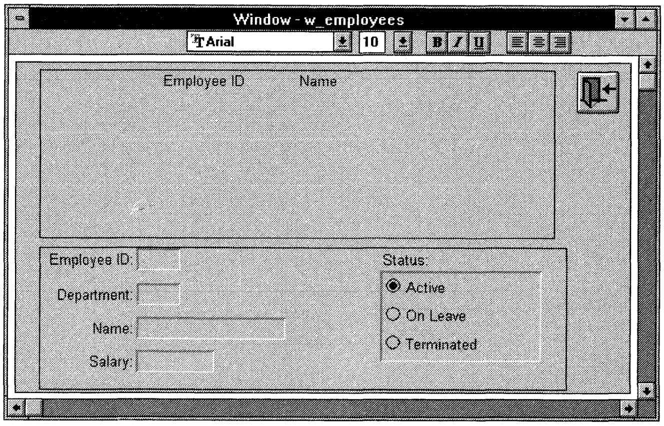


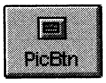
How long will this lesson take? About 20 minutes.

Add PictureButtons

Where you are

1 **At this point you should be in the Window painter with w_employees open.**
If you are not, open the Window painter and select *w_employees*.

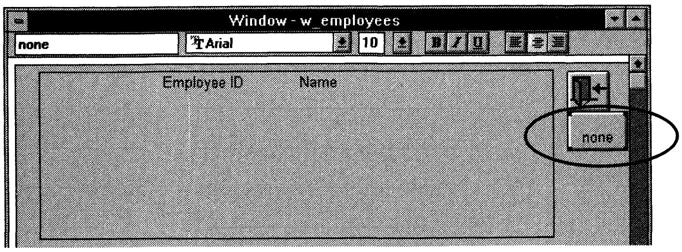




2 Click the *PictureButton* button in the PainterBar.

3 Click near the right side of the first DataWindow, under the *Exit* PictureButton.

A PictureButton displays where you clicked. The button is selected (has a black box at each corner).



4 Select *none* in the text box at the top of the Window painter and press DELETE.

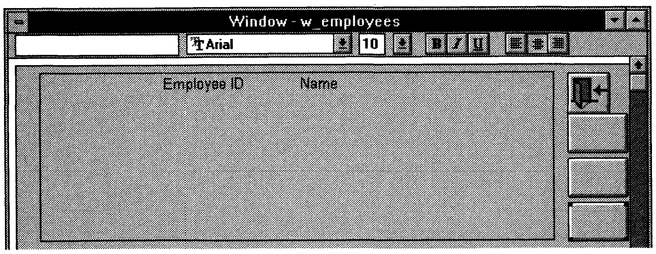
This eliminates the text from the PictureButton.

5 Press CTRL+T.

PowerBuilder creates a duplicate of the selected control.

6 Press CTRL+T again.

PowerBuilder creates another duplicate.



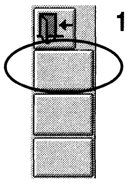
You now have three PictureButtons arranged vertically at the right of your window. Don't worry if the right-hand edge of the button is not visible. The button size will be adjusted automatically when you add the picture in the next step.

Make the New PictureButton work

Where you are
Lesson 10 Adding Database Maintenance Functionality
Add PictureButtons
➡ Make the New PictureButton work
Make the Delete PictureButton work
Make the Save PictureButton work
Align and space the PictureButtons
Test the application
Debug if necessary

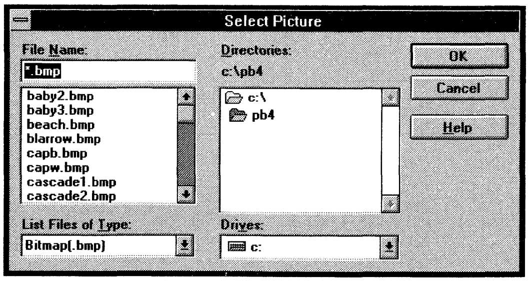
The New PictureButton will insert an empty row in the second DataWindow. You use this empty row to provide information for a new employee.

In a few minutes you will make the Save PictureButton work. The Save PictureButton is the button that actually adds the data in the new row to the database.



1 **Double-click the first of the three new PictureButtons.**

The Select Picture dialog box displays.



Select Picture

File Name: *.bmp

baby2.bmp
baby3.bmp
beach.bmp
blarrow.bmp
capb.bmp
capw.bmp
cascade1.bmp
cascade2.bmp

Directories: c:\pb4

c:\
pb4

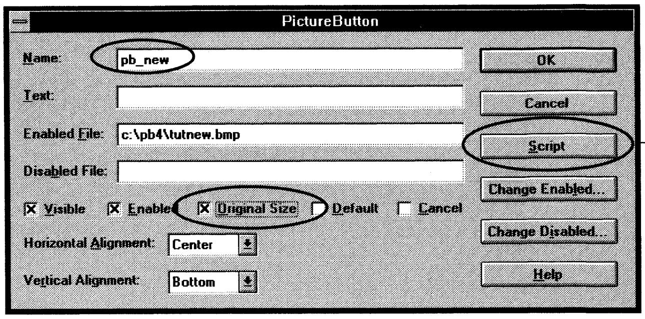
List Files of Type: Bitmap(.bmp)

Drives: c:

2 **Double-click the filename *tutnew.bmp*.**

The Select Picture dialog box closes and the PictureButton dialog box displays with the 1 of pb_1 highlighted.

3 **Type *new***
Select the *Original Size* checkbox.
Click the *Script* button.



Name: pb_new
Text:
Enabled File: c:\pb4\tutnew.bmp
Disabled File:
[X] Visible [X] Enabled [X] Original Size [ ] Default [ ] Cancel
Horizontal Alignment: Center
Vertical Alignment: Bottom

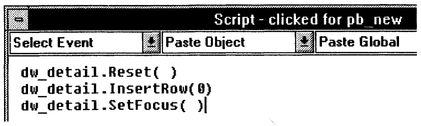
Script button

New replaces the highlighted 1 so the PictureButton control becomes pb_new. The PowerScript painter displays.

4 **Type the following script:**

```
dw_detail.Reset ()
dw_detail.InsertRow (0)
dw_detail.SetFocus ()
```

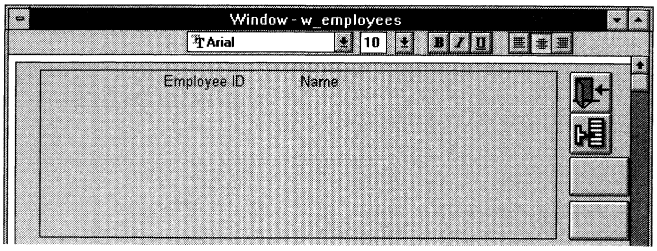
The first line clears (resets) the DataWindow. The second line adds a new row to the DataWindow. The third line positions the cursor in the dw_detail control.



```
dw_detail.Reset( )
dw_detail.InsertRow(0)
dw_detail.SetFocus( )
```

5 **Click the *Return* button in the PainterBar.**
or
Select File➤Return from the menu bar.

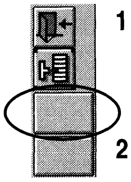
Your script compiles and you return to the Window painter. Your window now has a PictureButton for adding new employees. This PictureButton displays at its original size.



Make the Delete PictureButton work

Where you are

The Delete PictureButton will mark the row currently displayed in the dw_detail DataWindow for deletion.



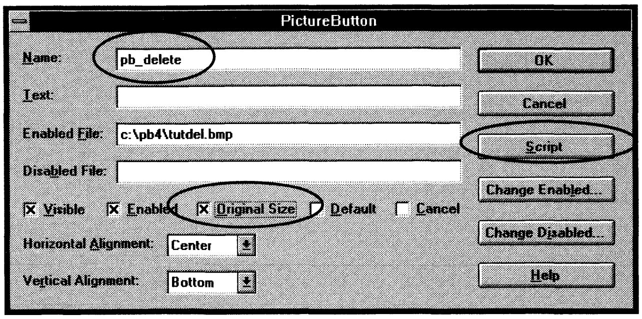
1 **Double-click the second of the three new PictureButtons.**

The Select Picture dialog box displays.

2 **Double-click the filename** ***tutdel.bmp.***

The Select Picture dialog box closes and the PictureButton dialog box displays with the 2 of pb_2 highlighted.

3 **Type** ***delete***
Select the ***Original Size*** **checkbox.**
Click the ***Script*** **button.**

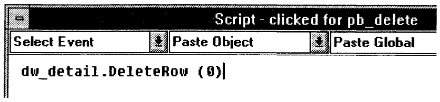


Name: pb_delete
Text:
Enabled File: c:\pb4\tutdel.bmp
Disabled File:
[X] Visible [X] Enabled [X] Original Size [ ] Default [ ] Cancel
Horizontal Alignment: Center
Vertical Alignment: Bottom

Delete replaces the highlighted 2 so the PictureButton control becomes pb_delete. The PowerScript painter displays.

4 Type the following script:

```
dw_detail.DeleteRow (0)
```

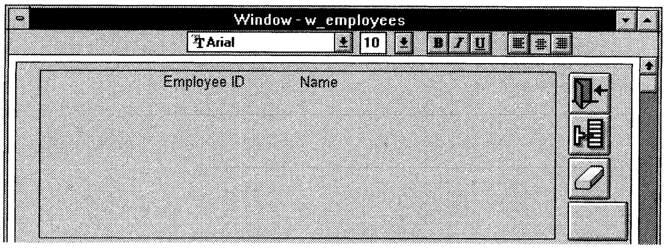


```
dw_detail.DeleteRow (0)
```

The argument 0 in the DeleteRow function specifies that the current row in the dw_detail control is to be deleted. This does not delete the row from the dw_master DataWindow control. You could add lines to the script that perform dw_master DeleteRow processing.

5 Click the *Return* button in the PainterBar.
or
Select File➢Return from the menu bar.

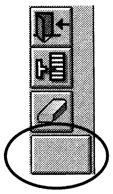
Your script compiles and you return to the Window painter. Your window now has a Delete PictureButton for deleting employees.



Make the Save PictureButton work

In the tutor_pb application, the changes you make in the DataWindow are not automatically applied to the database. The Save PictureButton will issue the Update function to apply your changes to the database:

- If you have just added a new employee using the New PictureButton, the Save PictureButton will add the new row to the database.
- If you have just deleted an employee using the Delete PictureButton, the Save PictureButton will delete the row from the database.
- If you have just changed employee information, the Save PictureButton will apply the changes to the database.



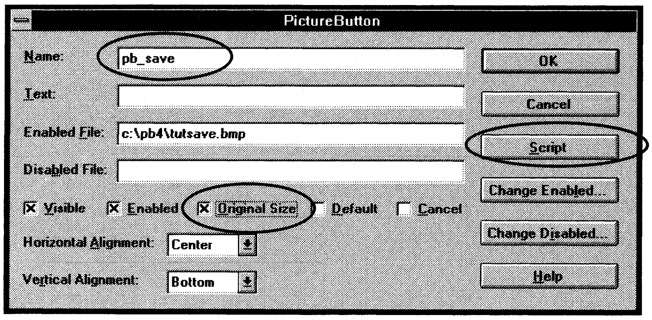
1 **Double-click the third PictureButton.**

The Select Picture dialog box displays again.

2 **Double-click the filename *tutsave.bmp*.**

The Select Picture dialog box closes and the PictureButton dialog box displays.

3 **Type** *save*
Select the *Original Size* **checkbox.**
Click the *Script* **button.**



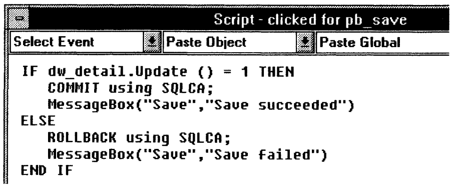
Name: pb_save
Text:
Enabled File: c:\pb4\tutsave.bmp
Disabled File:
[X] Visible [X] Enabled [X] Original Size [ ] Default [ ] Cancel
Horizontal Alignment: Center
Vertical Alignment: Bottom

Save replaces the highlighted 3 so the PictureButton control becomes pb_save. The PowerScript painter displays.

4 **Type the following script:**

```
IF dw_detail.Update () = 1 THEN
   COMMIT using SQLCA;
   MessageBox("Save","Save succeeded")
ELSE
   ROLLBACK using SQLCA;
   MessageBox("Save","Save failed")
END IF
```

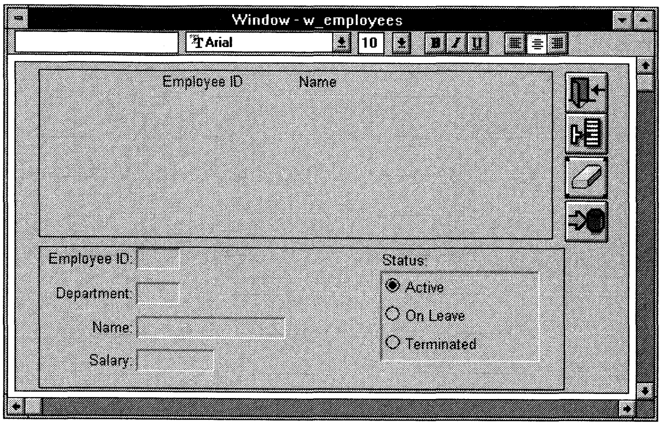
The first line updates the database with all changes to the dw_detail DataWindow. If the update succeeds, it performs a database commit and displays a success message. If the update fails, it performs a database rollback and displays a failure message.



```
IF dw_detail.Update () = 1 THEN
   COMMIT using SQLCA;
   MessageBox("Save","Save succeeded")
ELSE
   ROLLBACK using SQLCA;
   MessageBox("Save","Save failed")
END IF
```

5 **Click the *Return* button in the PainterBar.**
or
Select File➢Return from the menu bar.

Your script compiles and you return to the Window painter. Your window now has a Save PictureButton for applying changes to the database.



6 **Select File➢Save from the menu bar.**

This saves your changes before modifying PictureButton alignment and spacing.

Align and space the PictureButtons

Where you are
Lesson 10 Adding Database Maintenance Functionality
Add PictureButtons
Make the New PictureButton work
Make the Delete PictureButton work
Make the Save PictureButton work
➡ Align and space the PictureButtons
Test the application
Debug if necessary

Now you will align the PictureButtons and space them evenly.

1 **Use the CTRL+click method to select all four PictureButtons.**

Using the CTRL+click selection method gives you control over which button determines alignment, sizing, and spacing.

- **Alignment and sizing** PowerBuilder uses the *first* button you select to align and size the remaining buttons.
- **Spacing** PowerBuilder uses the *first two* buttons you select to adjust the spacing of the remaining buttons.

2 **Select Edit➤Align Controls from the menu bar.**

The align options display.

3 **Click the option (left, right, or middle) that represents the way you want to align the PictureButtons.**

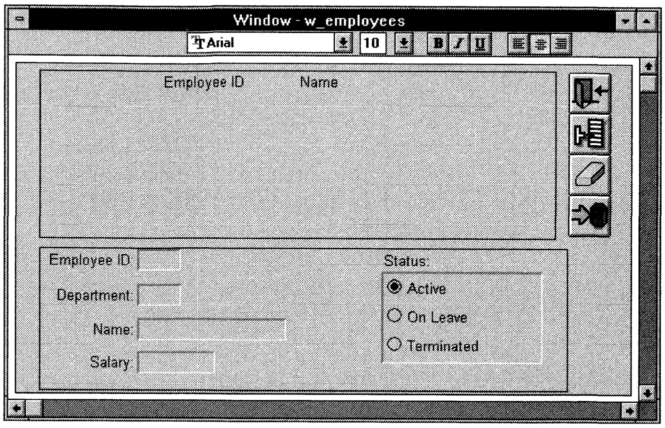
PowerBuilder adjusts the alignment.

4 **Select Edit➤Space Controls from the menu bar.**

5 **Click the option that adjusts space vertically.**

PowerBuilder adjusts the space between the PictureButtons.

Your window should look something like this now.



Test the application

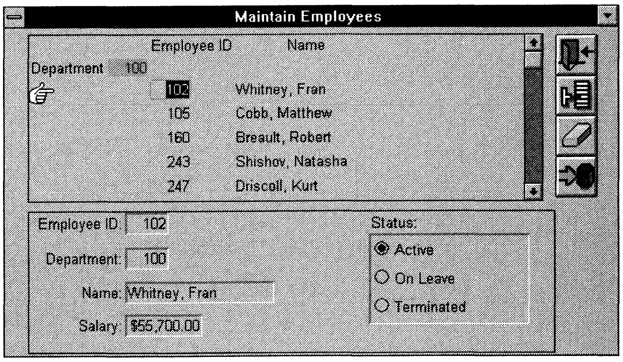
Where you are
Lesson 10 Adding Database Maintenance Functionality
Add PictureButtons
Make the New PictureButton work
Make the Delete PictureButton work
Make the Save PictureButton work
Align and space the PictureButtons
➡ Test the application
Debug if necessary

1 **Click the *Run* button in the PowerBar.**
or
Select File➤Run from the menu bar.

PowerBuilder prompts you to confirm that you want to save the current changes.

2 **Click *Yes*.**

Your application runs.


Maintain Employees

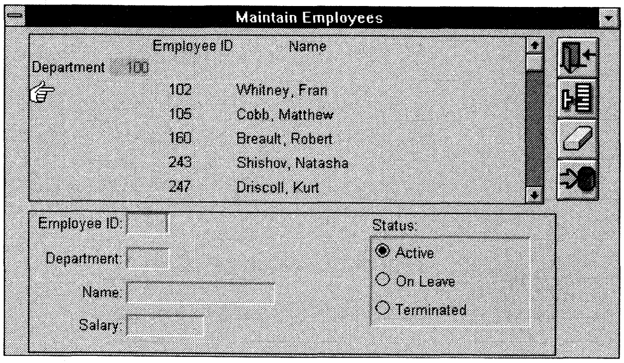
	Employee ID	Name
Department 100		
	102	Whitney, Fran
	105	Cobb, Matthew
	160	Breault, Robert
	243	Shishov, Natasha
	247	Driscoll, Kurt

Employee ID: 102
Department: 100
Name: Whitney, Fran
Salary: $55,700.00

Status:
[✓ Active]
On Leave
Terminated

3 **Click the *New* PictureButton.**

A new blank form displays in the lower DataWindow. Notice that the Status is Active as you requested for Initial Value in the database extended attribute information.


Maintain Employees

Department 100

Employee ID	Name
102	Whitney, Fran
105	Cobb, Matthew
160	Breault, Robert
243	Shishov, Natasha
247	Driscoll, Kurt

Employee ID:
Department:
Name:
Salary:

Status: [✓ Active] On Leave Terminated

4 **Click the *Employee ID* box.**

5 **Add an employee ID, department number, name, and salary for the new employee.**
Use the TAB key to move from box to box.

6 **Click the *Save* PictureButton.**

This sends the new employee data to the database and displays a confirmation message, as coded in the pb_save script.

In this tutorial, the new employee does not display in the top DataWindow. (You could code the script to include this feature.)

7 **Click an employee in the top DataWindow.**

That employee's data displays in the lower DataWindow.

8 **Change the employee's salary.**

9 **Click the *Save* PictureButton.**

This sends the revised employee data to the database and displays a confirmation message, as coded in the pb_save script.

You could program this update to occur automatically.

10 Select another employee in the top DataWindow.

That employee's data displays in the bottom DataWindow.

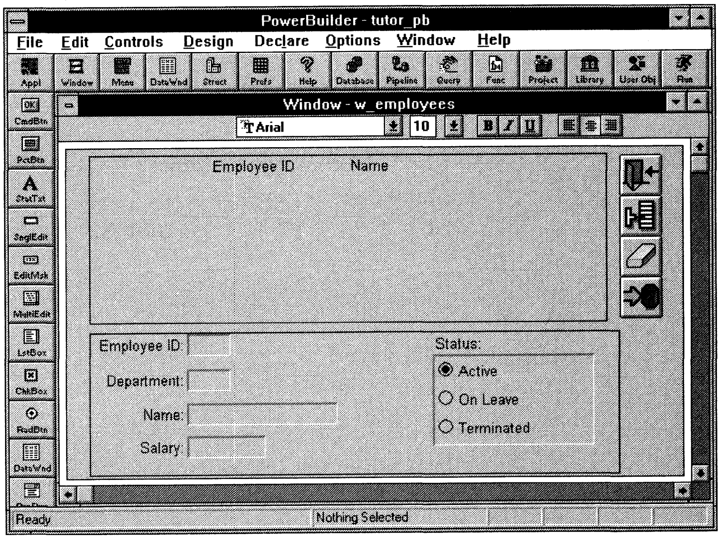
11 Click the *Delete* PictureButton.

The employee is deleted from the DataWindow immediately but will not be deleted from the database until you click the Save button.

Again, you could program this update to occur automatically.

12 Click the *Exit* PictureButton.

You return to the Window painter.


PowerBuilder - tutor_pb
File Edit Controls Design Declare Options Window Help
Appl Window Menu DataWnd Struct Prefs Help Database Pipeline Query Func Project Library User Obj Run
CmdBtn PictBtn StatTxt SnglEdit EditMsk MultiEdit ListBox ChkBox RadBtn DataWnd
Window - w_employees
Arial 10
Employee ID Name
Employee ID
Department
Name
Salary
Status:
Active
On Leave
Terminated
Ready
Nothing Selected

13 Select File➤Close from the menu bar.

You return to the Window painter.

You now have a complete application for viewing, adding, changing, and deleting employee data.

Debug if necessary

Where you are
Lesson 10 Adding Database Maintenance Functionality
Add PictureButtons
Make the New PictureButton work
Make the Delete PictureButton work
Make the Save PictureButton work
Align and the space PictureButtons
Test the application
➡ Debug if necessary

Your application should run without errors. However, if an error occurs, you can use Debug to find it.

For information about Debug, see the *User's Guide.*

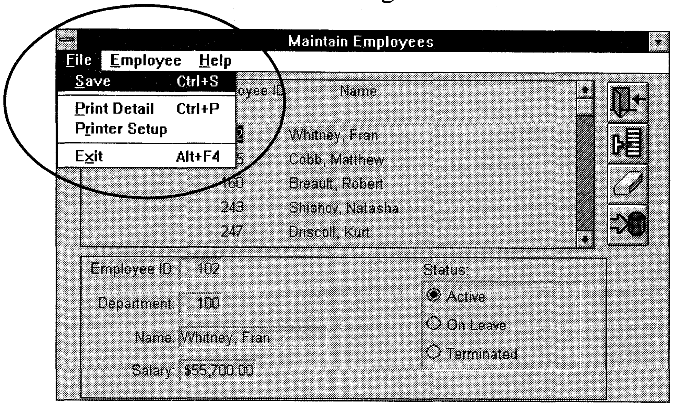
LESSON 11

Adding a Menu

In this lesson you will create a menu and add it to your window. Although you are creating the menu last in this tutorial, you could create it at any time.

Menus are separate objects that you create using the Menu painter. After you create a menu, you can attach it to as many windows as you want.

When you finish this lesson, your application window will have a menu bar. It will look like this during execution.



Employee ID	Name
	Whitney, Fran
	Cobb, Matthew
160	Breault, Robert
243	Shishov, Natasha
247	Driscoll, Kurt

Employee ID:	102
Department:	100
Name:	Whitney, Fran
Salary:	$55,700.00

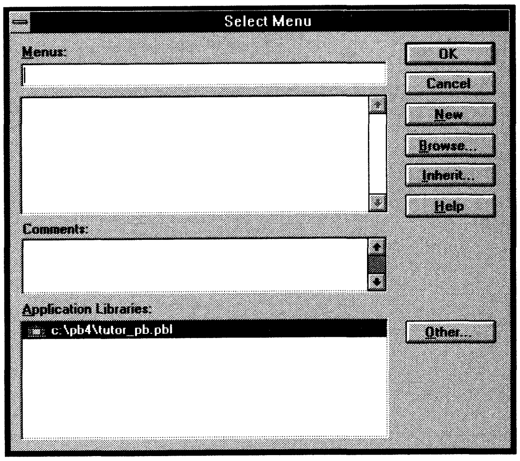
How long will this lesson take?
About 25 minutes.

Open the Menu painter

Where you are
Lesson 11 Adding a Menu
➡ Open the Menu painter
Add menu items
Add scripts
Save the menu
Add more menu items
Add more scripts
Add the menu to the window
Test the menu

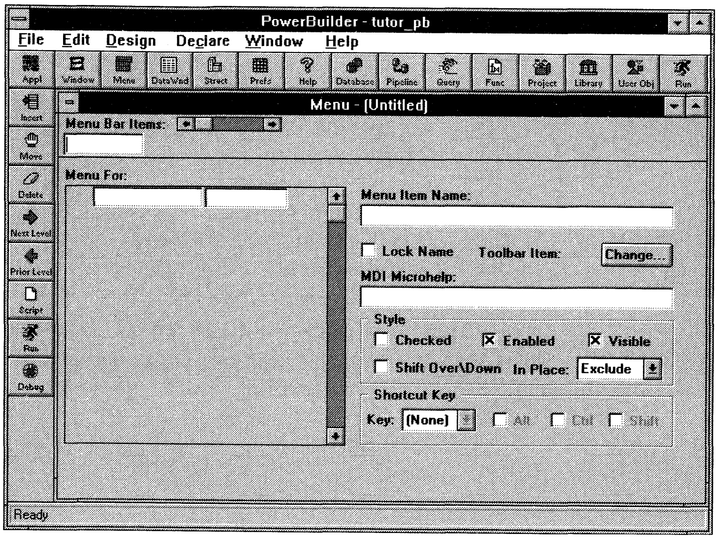
1 **Click the *Menu painter* button in the PowerBar.**

The Select Menu dialog box displays.



2 Click the *New* button to create a new menu.

The Menu painter workspace displays.



In the Menu painter, you build menus the way they will look when they're done. That is, you put the menu bar items across the top of the painter, and you build the menu for each menu bar item in a list underneath.

Add menu items

Where you are

Lesson 11 Adding a Menu

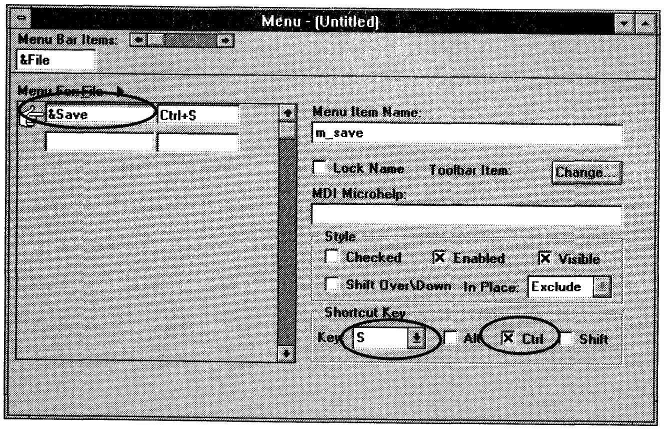
When you first open the Menu painter workspace, the insertion point is positioned at the first item in the menu bar.

1 **Type** *&File*
Press TAB.

The insertion point moves to the first menu item under Menu For.

The ampersand character defines accelerator keys
The & (ampersand) is the standard for defining accelerator keys, which provide mnemonic access to menus and window controls. In the tutor_pb application, for example, the user can access the File menu by pressing ALT+F.

2 **Type** *&Save*
Type *S* **in the Shortcut Key field.**
Select the *Ctrl* **checkbox.**
Press TAB.



&File
&Save Ctrl+S
Menu Item Name: m_save
Lock Name: [ ]
Checked [ ] Enabled [X] Visible [X]
Shift Over\Down [ ] In Place: Exclude
Key: S Alt [ ] Ctrl [X] Shift [ ]

This establishes a Save menu item under the File menu. S is the accelerator key, CTRL+S is the shortcut key, and m_save is the menu item name (you use this name to reference the menu item in scripts).

The pointer moves to the second menu item under File.

3 **Type – (hyphen)**
Press TAB.

The hyphen creates a separator line that spans the width of the menu and m_- is the menu item name.

4 **Type** *&Print Detail*
Type *P* **in the Shortcut Key field.**
Select the *Ctrl* **checkbox.**
Press TAB.

This establishes a Print Detail menu item. P is the accelerator key, CTRL+P is the shortcut key, and m_printdetail is the menu item name.

5 **Type** *P&rinter Setup...*
Press TAB.

This establishes a Printer Setup menu item. R is the accelerator key and m_printersetup is the menu item name.

Using ellipsis points
By convention, ellipsis points (...) following a menu item indicates that clicking the item will display a dialog box.

6 **Type another - (hyphen)**
Press TAB.

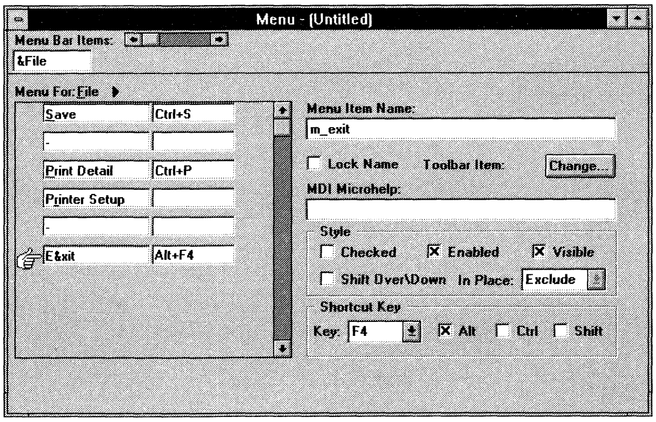
This establishes another separator line with a default menu item name of m_-, which is a duplicate of the automatically generated menu item name for the first hyphen. Duplicate menu item names are not allowed and the Invalid Menu Item Name dialog box displays, suggesting an alternative menu item name.

For more information on duplicate menu item names, see the *User's Guide.*

7 **Click *OK* to accept the suggested alternative name and close the dialog box.**

8 **Type *E&xit***
Select *F4* in the Shortcut Key dropdown listbox.
Select the *Alt* checkbox.

This establishes an Exit menu item. X is the accelerator key, ALT+F4 is the shortcut key, and m_exit is the menu item name.


Menu - (Untitled)

Menu Bar Items: &File

Menu For: File

Save	Ctrl+S
-	
Print Detail	Ctrl+P
Printer Setup	
-	
E&xit	Alt+F4

Menu Item Name: m_exit

☐ Lock Name Toolbar Item: Change...

MDI Microhelp:

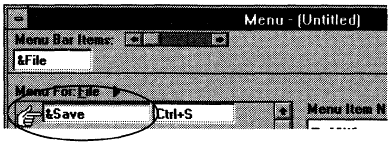
Style: ☐ Checked [X] Enabled [X] Visible
☐ Shift Over\Down In Place: Exclude

Shortcut Key: Key: F4 [X] Alt ☐ Ctrl ☐ Shift

Add scripts

Now you will add scripts for the Clicked events associated with each of the menu items you just defined. When the application executes and the user selects one of these menu items, PowerBuilder executes the associated script.

1 **Click in the *Save* box under the File menu that you added.**



2 **Click the *Script* button in the PainterBar.**

The PowerScript painter displays.

3 **Type the following script:**

```
w_employees.pb_save.TriggerEvent(Clicked!)
```

This triggers the Clicked event for the Save PictureButton in the w_employees window. That is, selecting File➤Save is the same as clicking the Save PictureButton. You could just as easily have entered script to update the dw_detail DataWindow, but then you would have to maintain script (including all the error checking) in two places (pb_save and the Save menu item).

4 **Click the *Return* button in the PainterBar.**
or
Select File➤Return from the menu bar.

Your script compiles and you return to the Menu painter.

5 **Click the *Print Detail* box.**
Click the *Script* button in the PainterBar.
Type the following script:

```
w_employees.dw_detail.Print ()
```

This uses the Print function to print the detail information for the current employee.

6 **Click the *Return* button in the PainterBar.**
or
Select File➤Return from the menu bar.

7 **Click the *Printer Setup* box.**
Click the *Script* button in the PainterBar.
Type the following script:

```
PrintSetup ()
```

This uses the PrintSetup function to display the standard Print Setup dialog box.

8 **Click the *Return* button in the PainterBar.**
or
Select File➤Return from the menu bar.

9 **Click the *Exit* box.**
Click the *Script* button in the PainterBar.
Type the following script:

```
Close(ParentWindow)
```

About the ParentWindow keyword
The reserved word ParentWindow is used only in menu scripts and means the window to which the menu is attached.

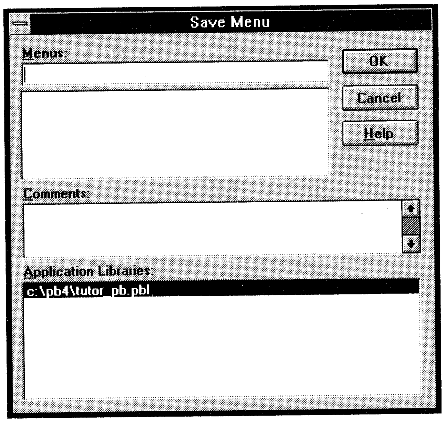
10 Click the *Return* button in the PainterBar.
or
Select File➢Return from the menu bar.

Save the menu

Where you are
Lesson 11 Adding a Menu
Open the Menu painter
Add menu items
Add scripts
➡ Save the menu
Add more menu items
Add more scripts
Add the menu to the window
Test the menu

1 **Select File➤Save from the menu bar.**

The Save Menu dialog box opens.



The insertion point is in the Menus box.

2 **Type *m_employees* as the name of the new menu.**
Type a comment in the Comments box.
Click *OK*.

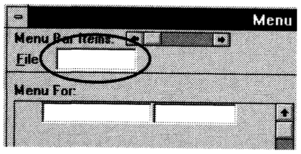
The name you just assigned to the menu displays in the title bar of the Menu painter workspace.

Add more menu items

Where you are
Lesson 11 Adding a Menu
Open the Menu painter
Add menu items
Add scripts
Save the menu
➡ Add more menu items
Add more scripts
Add the menu to the window
Test the menu

1 **Click just to the right of the File menu bar item that you added earlier in this lesson.**

A new empty box displays.



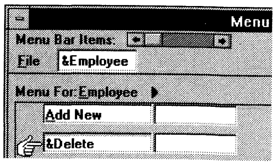
2 **Type** *&Employee*
Press TAB.

The insertion point moves to the first menu item under Employee.

3 **Type** *&Add New*
Press TAB.

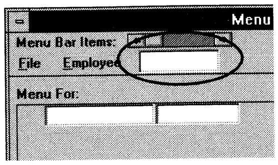
The insertion point moves to the second menu item under Employee.

4 **Type** *&Delete*



5 Click just to the right of the Employee menu item.

Another empty box displays.



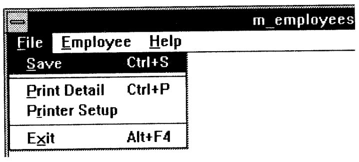
6 Type *&Help*
Press TAB.

The insertion point moves to the first menu item under Help.

7 Type *&Contents*
Select *F1* in the Shortcut Key dropdown listbox.

8 Select Design➢Preview from the menu bar.

PowerBuilder displays your menu at the top of an empty sample window. You can select items from this menu to confirm its design. For example, if you click on the File menu, you should see this:



9 Select Design➢Preview from the menu bar again.

The Menu painter workspace displays.

Add more scripts

Where you are
Lesson 11 Adding a Menu
Open the Menu painter
Add menu items
Add scripts
Save the menu
Add more menu items
➡ Add more scripts
Add the menu to the window
Test the menu

Now you will add script for menu items added in the previous step.

1 **Click the *Employee* box.**

2 **Click the *Add New* box.**

3 **Click the *Script* button in the PainterBar.**

The PowerScript painter displays.

4 **Type the following script:**

```
w_employees.pb_new.TriggerEvent(Clicked!)
```

This triggers the Clicked event for the New PictureButton in the w_employees window. That is, selecting Employee➤Add New is the same as clicking the New PictureButton.

5 **Click the *Return* button in the PainterBar.**
or
Select File➤Return from the menu bar.

Your script compiles and you return to the Menu painter.

6 **Click the *Delete* box.**
Click the *Script* button in the PainterBar.
Type the following script:

```
w_employees.pb_delete.TriggerEvent(Clicked!)
```

This triggers the Clicked event for the Delete PictureButton in the w_employees window. That is, selecting Employee➢Delete is the same as clicking the Delete PictureButton.

7 **Click the *Return* button in the PainterBar.**
or
Select File➢Return from the menu bar.

8 **Click the *Help* box.**

9 **Click the *Contents* box.**
Click the *Script* button in the PainterBar.
Type the following script:

```
ShowHelp("tutor_pb.hlp",Index!)
```

This opens the tutor_pb.hlp windows Help file.

Tutor_pb.hlp must be accessible
The tutor_pb.hlp file, which is delivered with PowerBuilder Version 4.0 and installed in the same directory as the other tutorial files, should be in a directory that is on your Path. If not, you must modify the ShowHelp function with the fully qualified name of the directory containing the tutor_pb.hlp file.

10 **Select File➢Close from the menu bar.**

PowerBuilder prompts you to save your changes.

11 **Click *Yes.***

The Menu painter closes.

You now have a complete menu that is ready to be associated with a window.

Add the menu to the window

Where you are
Lesson 11 Adding a Menu
Open the Menu painter
Add menu items
Add scripts
Save the menu
Add more menu items
Add more scripts
➡ Add the menu to the window
Test the menu

1 **Click the *Window painter* button in the PowerBar.**

The Window painter opens and the Select Window dialog box displays.

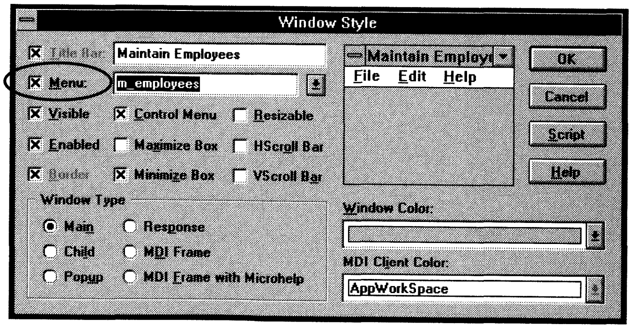
2 **Select *w_employees* and click *OK*.**

Your window displays in the workspace.

3 **Select Design➤Window Style from the menu bar.**

The Window Style dialog box displays.

4 **Select the *Menu* checkbox.**



Window Style

[✓] Title Bar: Maintain Employees
[✓] Menu: m_employees
[✓] Visible [✓] Control Menu [ ] Resizable
[✓] Enabled [ ] Maximize Box [ ] HScroll Bar
[✓] Border [✓] Minimize Box [ ] VScroll Bar

Window Type: ● Main ○ Response ○ Child ○ MDI Frame ○ Popup ○ MDI Frame with Microhelp

Maintain Employees — File Edit Help

Window Color:
MDI Client Color: AppWorkSpace

Since there is only one menu in this application, the menu you just created, m_employees, displays in the listbox.

If there were more menus, you could click the down arrow next to the menu name to see all the menus in this application.

When you checked the Menu checkbox, PowerBuilder displayed a menu in the sample window on the right. This menu is *not* the menu you just created; it is a generic menu to indicate you want a menu on this window.

5 **Click *OK*.**

The Window painter workspace displays.

Test the menu

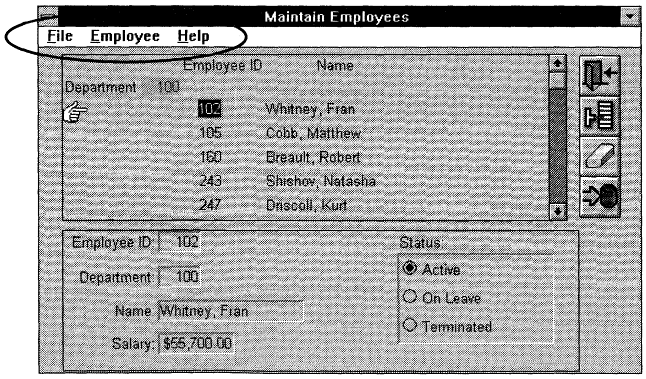
Where you are
Lesson 11 Adding a Menu
Open the Menu painter
Add menu items
Add scripts
Save the menu
Add more menu items
Add more scripts
Add the menu to the window
➡ Test the menu

1 **Click the *Run* button.**
or
Select File➢Run from the menu bar.

PowerBuilder prompts to confirm that you want to save the current changes and run your application.

2 **Click *Yes*.**

Your application runs. The window opens and displays a menu bar across the top.



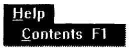
```
          Employee ID     Name
Department  100
               102        Whitney, Fran
               105        Cobb, Matthew
               160        Breault, Robert
               243        Shishov, Natasha
               247        Driscoll, Kurt

Employee ID: 102                       Status:
 Department: 100                       ● Active
       Name: Whitney, Fran             ○ On Leave
     Salary: $55,700.00                ○ Terminated
```

> **If your window doesn't look right**
> Adding a menu moves all the objects in the window down. As a result, the second DataWindow in your window may be partially off the bottom. If this happens, return to the Window painter and enlarge the window size.

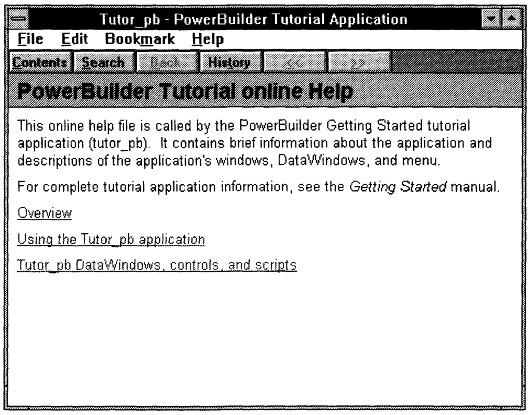
3 **Click the *Help* menu.**

The Contents menu item drops down.



4 **Click *Contents*.**

The Windows Help system opens displaying the tutor_pb.hlp Help file.


Tutor_pb - PowerBuilder Tutorial Application
File Edit Bookmark Help
Contents Search Back History << >>
PowerBuilder Tutorial online Help

This online help file is called by the PowerBuilder Getting Started tutorial application (tutor_pb). It contains brief information about the application and descriptions of the application's windows, DataWindows, and menu.

For complete tutorial application information, see the *Getting Started* manual.

Overview

Using the Tutor_pb application

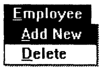
Tutor_pb DataWindows, controls, and scripts

5 **Press ALT+F4.**

The Maintain Employees window displays.

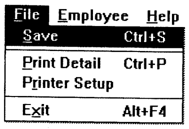
6 **Click the *Employee* menu.**

The Add New and Delete menu items drop down.



7 Click the *File* menu.

The Save, Print Detail, Printer Setup, and Exit menu items drop down.



8 Test other menu items.

Your application should run without errors. However, if an error occurs, you can use Debug to find it.

For information about Debug, see the *User's Guide.*

9 Select File➤Exit from the menu bar.

The script you wrote for this menu item [Close(ParentWindow)] runs.

Your application closes and you return to the Window painter.

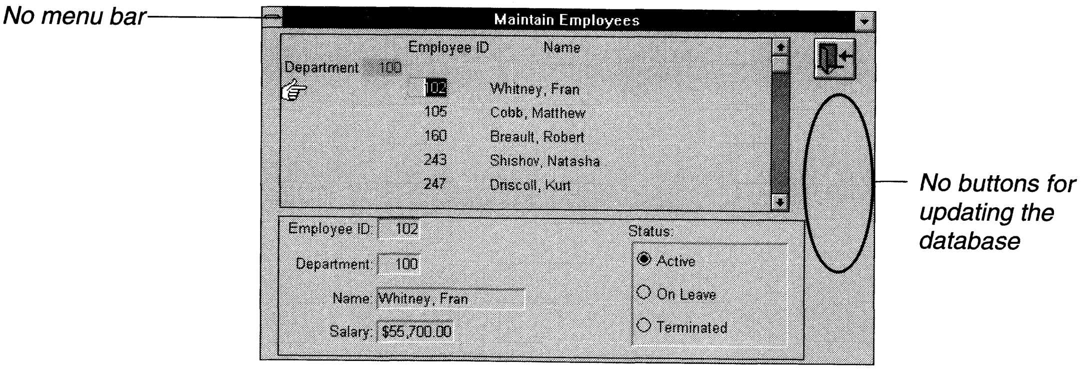
LESSON 12

Using Inheritance to Build a Window

In your applications, you will often have many windows that look similar and perform the same or similar activities. When this occurs, you will want the windows to have a standard look and you will not want to recode the scripts each time you create one of these windows. To do this, you can use inheritance.

In this lesson you will use inheritance to build a descendent window that looks like the window you just built (the ancestor) but limits the user's access. The descendent window will inherit its events, attributes, and scripts from the ancestor. You will modify it so it is a display-only window rather than an updatable window.

When you finish this lesson, the new application window will look like this.



Employee ID	Name
102	Whitney, Fran
105	Cobb, Matthew
160	Breault, Robert
243	Shishov, Natasha
247	Driscoll, Kurt

Department 100

Employee ID: 102
Department: 100
Name: Whitney, Fran
Salary: $55,700.00

Status:
● Active
○ On Leave
○ Terminated

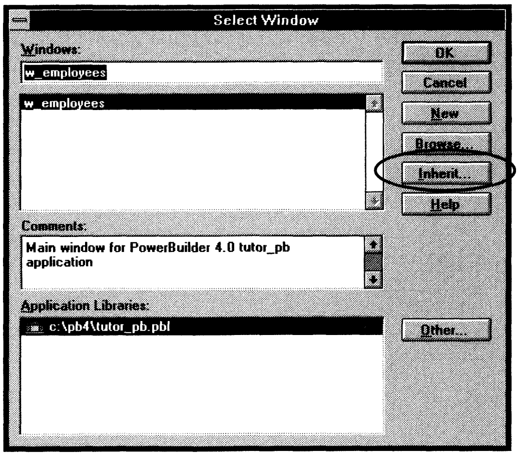
How long will this lesson take?
About 10 minutes.

Create the descendent window

Where you are
Lesson 12 Using Inheritance to Build a Window
➡ Create the descendent window
Modify the descendent window
Modify the application to display the window
Test the application

1 **You should be in the Window painter.**
Select File➤Open from the menu bar.

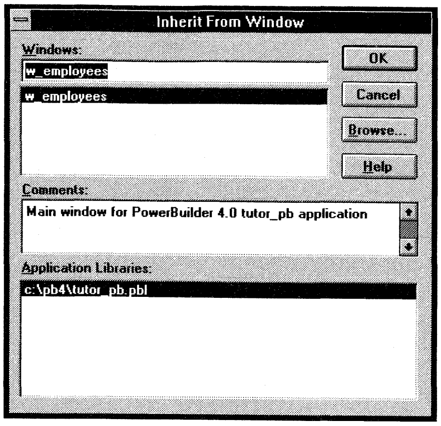
The Select Window dialog box displays. It includes an Inherit button.



Main window for PowerBuilder 4.0 tutor_pb application

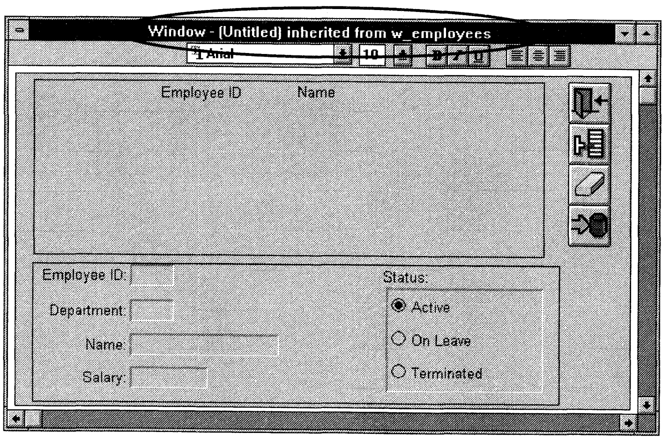
2 Click the *Inherit* button.

The Inherit From Window dialog box displays. You use this to build a window that inherits its appearance and scripts from an existing window.



3 Select *w_employees* and click *OK*.

The Window painter workspace opens and a new (Untitled) window inherited from the w_employees window displays.



Modify the descendent window

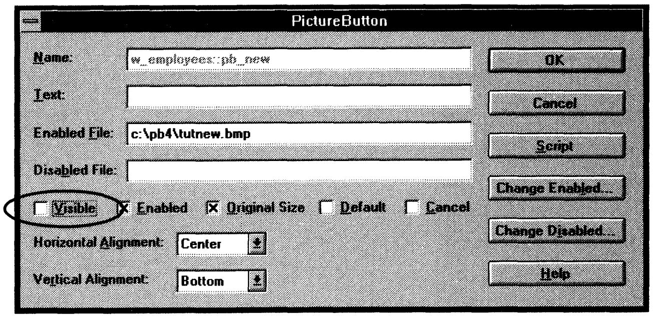
Where you are
Lesson 12 Using Inheritance to Build a Window
Create the descendent window
➡ Modify the descendent window
Modify the application to display the window
Test the application

You want to create a display-only window, so you do not want the user to see the bottom three PictureButtons. However, you cannot delete controls in a descendent window, so you will have to make them invisible. You will also remove the menu.

1 **Double-click the *New* PictureButton.**

The PictureButton dialog box displays.

2 **Deselect *Visible*.**



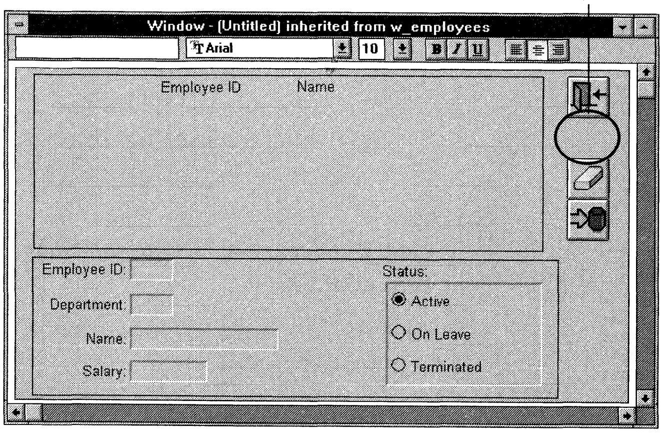
PictureButton

Name: w_employees::pb_new
Text:
Enabled File: c:\pb4\tutnew.bmp
Disabled File:
☐ Visible ☒ Enabled ☒ Original Size ☐ Default ☐ Cancel
Horizontal Alignment: Center
Vertical Alignment: Bottom

3 Click *OK*.

The PictureButton is no longer visible or active.

The button is no longer visible

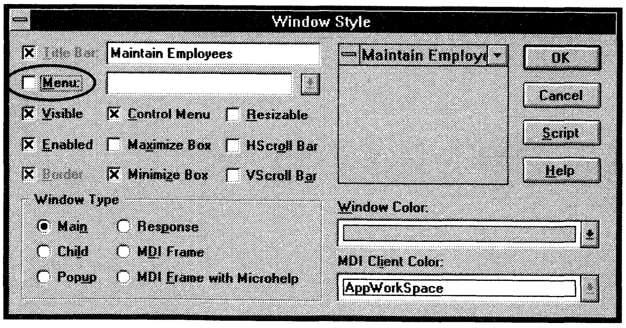


4 Repeat steps 1–3 for the *Delete* and *Save* PictureButtons.

5 Select Design➤Window Style from the menu bar.

The Window Style dialog box displays.

6 Deselect the *Menu* checkbox.



[X] Title Bar: Maintain Employees
[ ] Menu:
[X] Visible [X] Control Menu [ ] Resizable
[X] Enabled [ ] Maximize Box [ ] HScroll Bar
[X] Border [X] Minimize Box [ ] VScroll Bar
Window Type: (●) Main ( ) Response ( ) Child ( ) MDI Frame ( ) Popup ( ) MDI Frame with Microhelp
Maintain Employees
Window Color:
MDI Client Color: AppWorkSpace

7 **Click *OK.***

8 **Select File➤Close from the menu bar.**

A dialog box displays asking if you want to save changes.

9 **Click *Yes.***

The Save Window dialog box displays.

10 **Type *w_emp_display* in the Name box.**
Enter a comment in the Comments box.
Then click *OK.*

PowerBuilder saves your new window as a descendant of w_employees, closes the Window painter, and displays the PowerBar.

You can now reference your new window in scripts and create descendent objects that inherit their attributes, events, and scripts from this window.

Modify the application to display the window

Where you are
Lesson 12 Using Inheritance to Build a Window
Create the descendent window
Modify the descendent window
➡ Modify the application to display the window
Test the application

In a real application, you would probably check the user's ID and based on the user's level of access, display the appropriate window. But to keep it simple, in this tutorial you will just modify the script for the application Open event to display the descendent window.

1 **Open the Application painter by clicking the *Application* button in the PowerBar.**

The Application painter workspace displays.

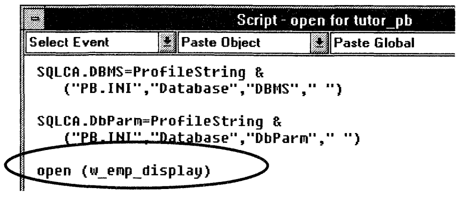
2 **Click the *Script* button in the PainterBar.**

The PowerScript painter workspace displays. The title should read: Script - open for tutor_pb.

If the event is incorrect
If the PowerScript painter displays an event other than Open, click on the down arrow next to the Select Event listbox and select Open.

3 **Select the window name *w_employees* in the last line of the script and type:**

```
w_emp_display
```



```
SQLCA.DBMS=ProfileString &
   ("PB.INI","Database","DBMS"," ")

SQLCA.DbParm=ProfileString &
   ("PB.INI","Database","DbParm"," ")

open (w_emp_display)
```

Now your script will display the new descendent window instead of the original window you created.

4 **Click the *Return* button.**
or
Select File➤Return from the menu bar.

Your script compiles and you return to the Application painter workspace.

5 **Select File➤Save from the menu bar.**

PowerBuilder saves your Application object with the revised Open script.

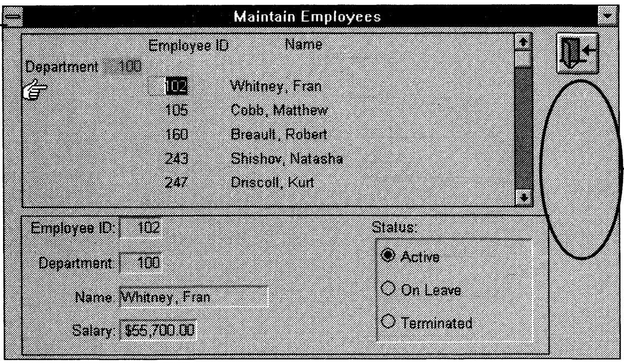
Test the application

Where you are
Lesson 12 Using Inheritance to Build a Window
Create the descendent window
Modify the descendent window
Modify the application to display the window
➡ Test the application

In this section you will test the finished application to be sure there are no errors before you build an executable file.

1 **To test the application, click the *Run* button in the PowerBar.**
or
Select File➤Run from the menu bar.

You can display data but you cannot insert rows, delete rows, or update the database.



Maintain Employees

Employee ID	Name
Department 100	
102	Whitney, Fran
105	Cobb, Matthew
160	Breault, Robert
243	Shishov, Natasha
247	Driscoll, Kurt

Employee ID: 102
Department: 100
Name: Whitney, Fran
Salary: $55,700.00
Status: [✓ Active] On Leave Terminated

There should not be any errors. But if there are, you can use Debug to find them.

Now before going on, you will restore the original window name in the script for the application Open event.

2 **Click the *Exit* PictureButton to close the application.**

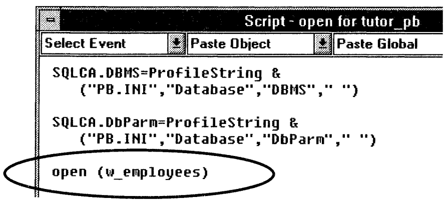
You return to the Application painter.

3 **Click the *Script* button in the PainterBar.**

The PowerScript painter workspace displays the current script for the application Open event.

4 **Select the window name *w_emp_display* in the last line of the script and type:**

```
w_employees
```



```
SQLCA.DBMS=ProfileString &
   ("PB.INI","Database","DBMS"," ")

SQLCA.DbParm=ProfileString &
   ("PB.INI","Database","DbParm"," ")

open (w_employees)
```

Now your script will open the original window you created.

5 **Click the *Return* button in the PainterBar.**
or
Select File➤Return from the menu bar.

Your script compiles and you return to the Application painter workspace.

6 **To make sure your application is working correctly before creating the EXE, click the *Run* button in the PowerBar.**
or
Select File➤Run from the menu bar.
Then click *Yes* to save your changes.

Your original application should run.

7 **Click the *Exit* PictureButton to close the application.**

You return to the Application painter.

LESSON 13

Creating the EXE File for the Application

When you finish an application, you can create an executable version of it for distribution to your users.

The users can run this executable version of your application from the Windows Program Manager like any other Windows application (such as Microsoft Excel or Word for Windows).

How long will this lesson take?
About 10 minutes.

Create the EXE file

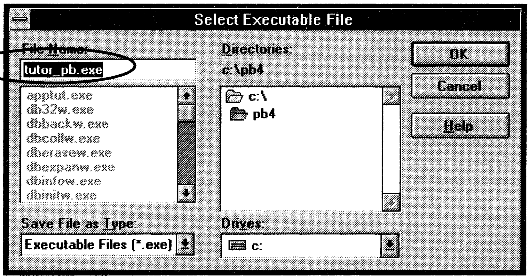
Where you are
Lesson 13 Creating the EXE File for the Application
➡ Create the EXE file
Test the EXE file

1 **Make sure you are in the Application painter and all other painters are closed.**

2 **Click the *Create Exe* button in the PainterBar (if you're using PowerTips, the text is Create Executable).**

If you haven't saved since modifying the Application Open event, a dialog box displays. If so, click Yes to save changes.

The Select Executable File dialog box displays. The default name for the executable file is the name of the application plus the EXE extension (in this case, tutor_pb.exe).



Select Executable File

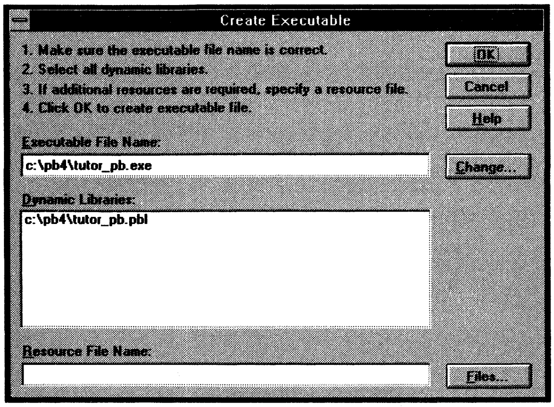
File Name: tutor_pb.exe
Directories: c:\pb4
Save File as Type: Executable Files (*.exe)
Drives: c:

About the EXE filename
If your application name has more than eight characters, PowerBuilder truncates it for the default EXE filename. You can choose any one- to eight-character name you want.

The default location is the directory in which you are working.

3 Click *OK*.

The Create Executable dialog box displays.



Create Executable

1. Make sure the executable file name is correct.
2. Select all dynamic libraries.
3. If additional resources are required, specify a resource file.
4. Click OK to create executable file.

Executable File Name: c:\pb4\tutor_pb.exe

Dynamic Libraries: c:\pb4\tutor_pb.pbl

Resource File Name:

About the Dynamic Libraries box
Do not select anything in the Dynamic Libraries box at the bottom of the screen.

For complete information about EXE creation, including the Project painter and dynamic libraries, see the *User's Guide*.

4 Click *OK*.

PowerBuilder creates an EXE file that contains definitions for the objects (such as windows, menus, and DataWindows) in your application, the bitmap files used in your PictureButtons, the application icon, and the compiled scripts. It does *not* contain any source code.

After creating the EXE file, PowerBuilder returns you to the Application painter workspace.

Test the EXE file

Where you are
Lesson 13 Creating the EXE File for the Application
Create the EXE file
➡ Test the EXE file

Now you are ready to test your new EXE file.

Running from another PC
To run your application from another PC, you need a copy of the Database Development and Deployment Kit.

1 **Click the down arrow in the upper-right corner of PowerBuilder to minimize PowerBuilder.**

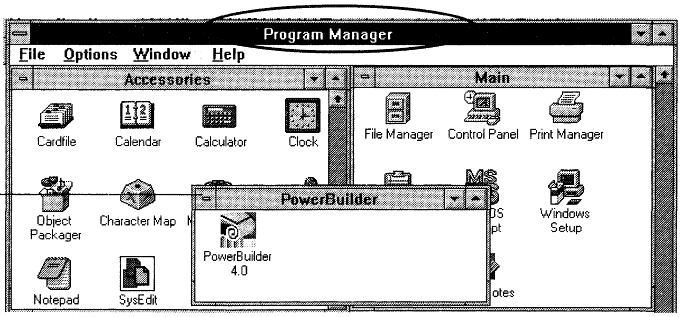
PowerBuilder collapses to an icon at the bottom of your screen.

2 **Open the Windows Program Manager.**

If necessary, double-click the Program Manager icon at the bottom of your screen.

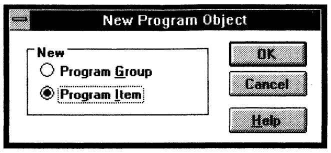
3 **Select the program group where you want the application icon.**

Program group



4 **Select File➤New from the Program Manager menu bar.**

The New Program Object dialog box displays. Program Item, which you want, is the default.

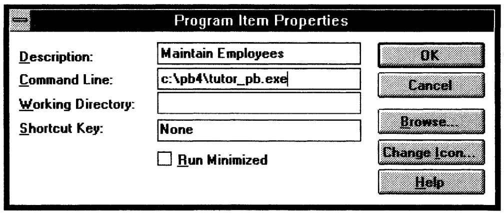


5 **Click** ***OK.***

The Program Item Properties dialog box displays.

6 **Type the description** *Maintain Employees*

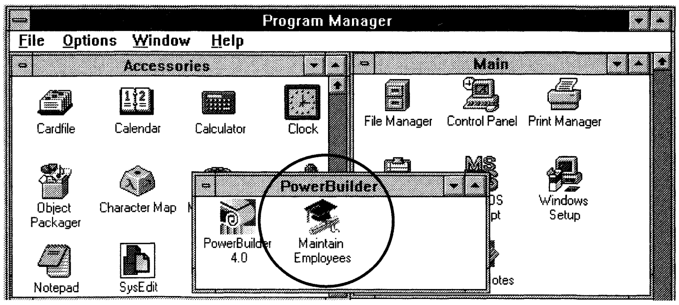
7 **Enter the fully qualified path to the EXE file in the Command Line box. For example:** *c:\pb4\tutor_pb.exe*



Description: Maintain Employees
Command Line: c:\pb4\tutor_pb.exe
Working Directory:
Shortcut Key: None
☐ Run Minimized

8 **Click** ***OK.***

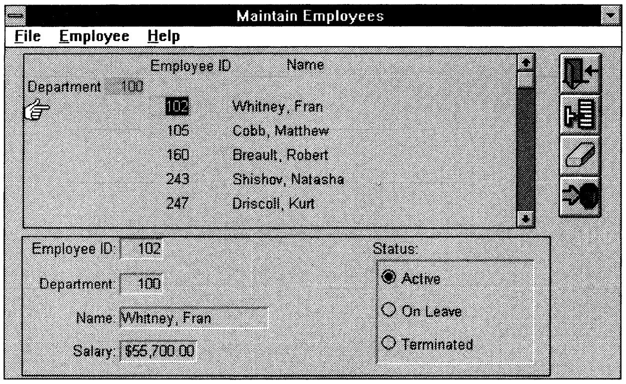
The Tutor_pb icon and description display in the selected program group in the Program Manager.



9 Double-click the icon for the *Maintain Employees* application.

Your application runs.

You can browse through employees, select them, and update information. You can also add new employees, change data, and make deletions.



	Employee ID	Name
Department 100		
	102	Whitney, Fran
	105	Cobb, Matthew
	160	Breault, Robert
	243	Shishov, Natasha
	247	Driscoll, Kurt

Employee ID: 102
Department: 100
Name: Whitney, Fran
Salary: $55,700.00
Status: [✓ Active] On Leave Terminated

10 Click the *Exit* PictureButton.

You return to Windows Program Manager or desktop.

11 Double-click the *PowerBuilder* icon at the bottom of your screen.

You return to PowerBuilder.

The EXE file is a compact, fully functional version of your application, ready to deliver to users.

What to do next
You should continue with the *Building Applications* manual.

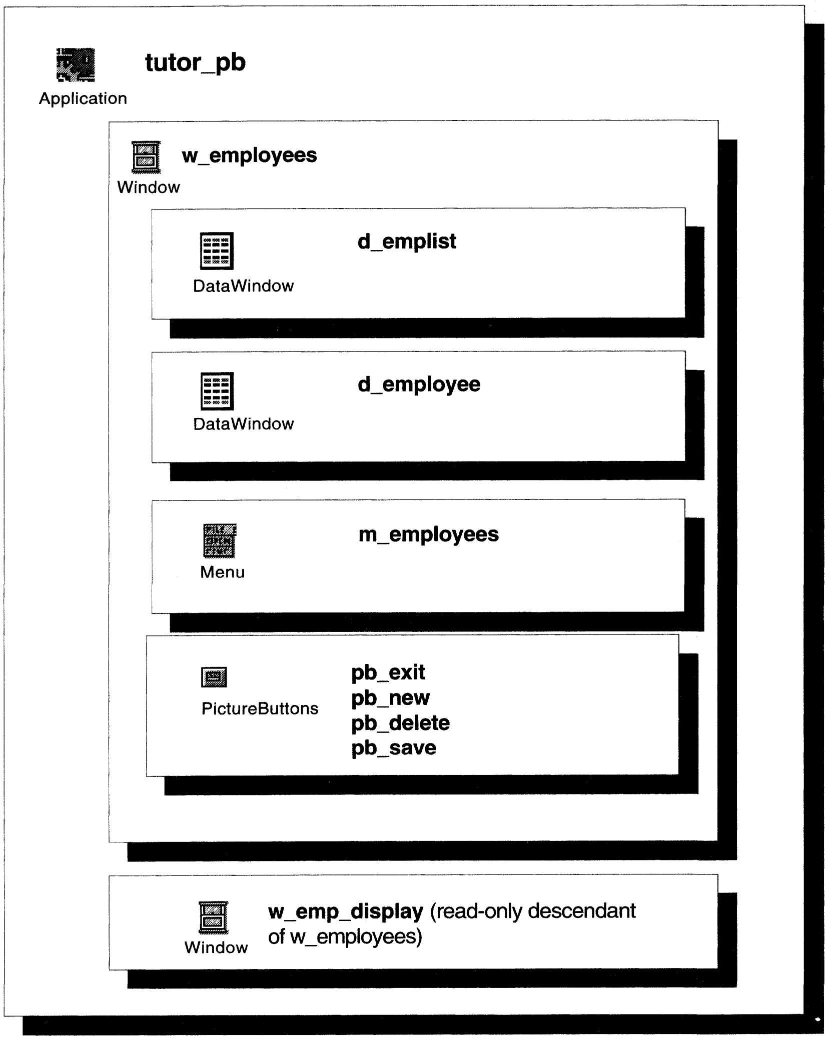
Additionally, look at the roadmap at the beginning of this manual. The roadmap is a guide to all the resources available as you learn to use PowerBuilder.

APPENDIX

Tutorial Application Review

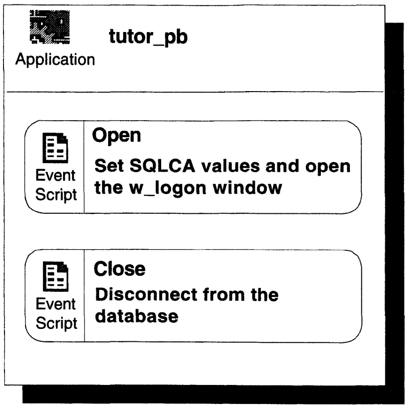
This appendix contains graphic representations of the tutor_pb application and its components. Use this information to review the objects, concepts, and organization of the tutor_pb application.

Application architecture

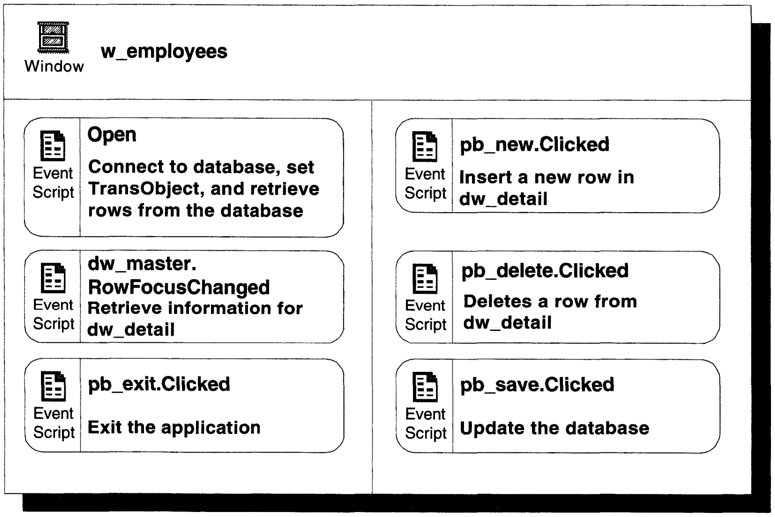


Code map

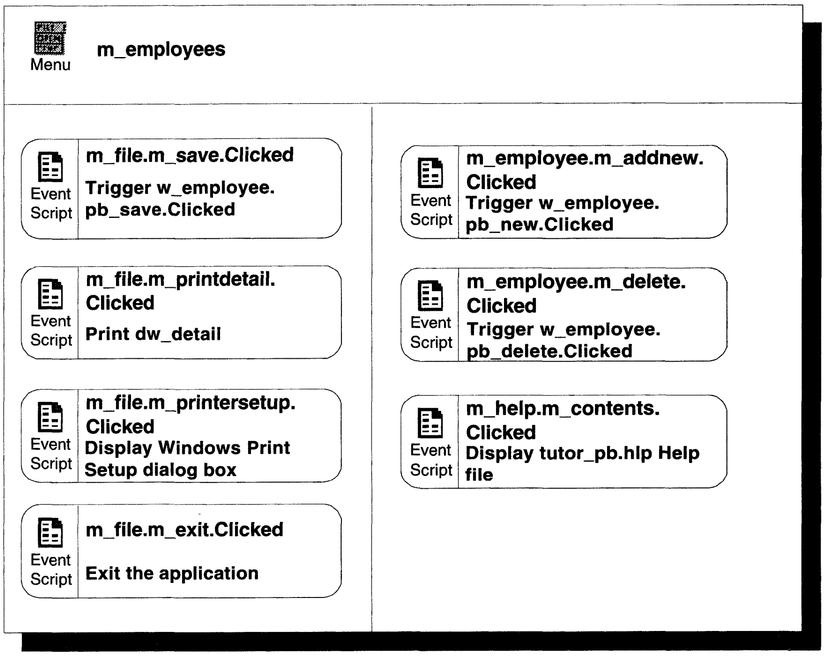
Application


Application
tutor_pb
Event Script
Open
Set SQLCA values and open the w_logon window
Event Script
Close
Disconnect from the database

Window



Menu


Menu
m_employees
Event Script
m_file.m_save.Clicked
Trigger w_employee. pb_save.Clicked
Event Script
m_file.m_printdetail. Clicked
Print dw_detail
Event Script
m_file.m_printersetup. Clicked
Display Windows Print Setup dialog box
Event Script
m_file.m_exit.Clicked
Exit the application
Event Script
m_employee.m_addnew. Clicked
Trigger w_employee. pb_new.Clicked
Event Script
m_employee.m_delete. Clicked
Trigger w_employee. pb_delete.Clicked
Event Script
m_help.m_contents. Clicked
Display tutor_pb.hlp Help file

Index