Powersoft.
Open Tools from Sybase, Inc.

AppModeler™

User's Guide

PowerDesigner®

Version 6

May 1997

Contents

About This Book

Subject

This book describes how to use PowerDesigner AppModeler functions and features. It provides information on how to create and manage a Physical Data Model (PDM) and the objects it contains. The book has separate chapters on reverse engineering from a database, using triggers and procedures, and creating and modifying a database.

A special section tells you how to generate PowerBuilder, Visual Basic, Power++, Delphi and Web applications using predefined templates that ship with AppModeler. The book also describes how to manipulate graphics and generate customized reports from the PDM.

Audience

This book is for anyone who is using AppModeler as a design tool for database construction and modification, or for the generation of fourth generation language (4GL) applications. Although it does not assume you have knowledge about any particular topic, having some familiarity with relational databases and SQL is helpful. Previous experience with PowerBuilder, Visual Basic, Power++, Delphi, or NetImpact Dynamo is useful if you are developing applications with an AppModeler generator.

Where to find information

Whether you install AppModeler for PowerBuilder, Visual Basic, Power++, Delphi, or the Web, you always have access to the core AppModeler database design functions. To help you do your work more easily, this book is divided into parts that focus on particular goals.

If you want to	Use these parts of the book
Learn about the environment	Part One, AppModeler Functions and Features
Build a Physical Data Model	Part Two, Physical Data Model
Generate a PowerBuilder, Visual Basic, Power++, Delphi, or Web application	Part Three, Application Generation
Change the graphic display and create model reports	Part Four, Model Presentation
Consult reference material	Part Five, Appendixes

PART ONE

AppModeler Functions and Features

This part describes technical characteristics of AppModeler as well as the hardware and software it requires.

CHAPTER 1

Features and Environment

About this chapter

This chapter presents the major features of this product, as well as its setup requirements.

Contents

Functional overview

This product is a powerful design tool for client/server applications. It gives you all the advantages of a physical data design and client/server interface implementation.

With this product, you can:

- Build a Physical Data Model (PDM) for a target database management system (DBMS), taking into account the specifics of the chosen DBMS
- Customize the PDM to suit physical and performance considerations
- Generate a database creation script for the target DBMS
- Generate referential integrity triggers if they are supported by the target database
- Customize and print model reports
- Reverse engineer existing databases and applications
- Define extended attributes for 4GL application generation

Desktop functions

Desktop editions of this product include nearly all standard features. However, because the desktop edition is designed for use by a single user tool it does not offer teamwork functions as follows:

- No submodel support
- No model extraction or consolidation

❖ **To check if you have a desktop version:**

1 Open the Dictionary menu.

If this menu does not have a Submodel item, then your version does not support submodels.

2 Open the File menu.

If this menu does not have a Consolidate item or an Extract item, then your version does not support extraction or consolidation.

Database support

Different versions of this product support different database management systems (DBMS) for database generation and for reverse engineering.

❖ **To display the list of databases supported for generation:**

1 Select Database➢Change Target Database.

Database Name dropdown listbox contains a list of DBMS available for database generation.

2 Click Cancel.

A message box tells you that the target database has not been changed.

❖ **To display the list of databases supported for reverse engineering:**

1 Select File➢Reverse Engineering.

Database Name dropdown listbox contains a list of DBMS supported for reverse engineering.

2 Click Cancel.

Client/server tools interface

Some versions of this product provide support for 4GL interfaces.

❖ **To display the list of supported 4GL interfaces:**

♦ Open the Client menu.

This menu lists all possible 4GL generations. If you do not have a Client menu, then your version does not provide direct support for 4GL interfaces.

Setup

This product uses a standard Windows setup program.

Minimum system requirements

This product has the following minimum system requirements:

- Microsoft Windows 3.1
  Personal computer with 80486 processor
  8 MB of random access memory (RAM)
  or
  Microsoft Windows 95
  Personal computer with 80486 processor
  12 MB of RAM
  or
  Microsoft Windows NT version 3.51
  Personal computer with 80486 processor
  12 MB of RAM
- VGA or higher-resolution graphics adapter and compatible color monitor
- CD-ROM drive
- 12 MB of hard disk space

Running setup

❖ **To run the setup program:**

3 Insert the PowerDesigner product CD-ROM in the drive.

4 Double-click the setup icon from the File Manager or the Explorer.

5 Follow the instructions on the screen.

> **Insufficient disk space**
> When there is insufficient disk space, the setup procedure is immediately aborted.

For complete setup instructions, see *PowerDesigner Installation Guide*.

Installed directories and files

Program directories

By default, the AppModeler Setup program creates the following directories:

Directory name	Long name*	Contents
C:\PWRS\PD6	Program Files\Powersoft\ PowerDesigner 6	Program files
C:\PWRS\PD6\DEF	...\Definition Files	DEF files for database definition
C:\PWRS\PD6\EXA	...\Extended Attributes	EXA files for extended attribute definitions
C:\PWRS\PD6\EXAMPLES	...\Examples	Examples
C:\PWRS\PD6\QUERIES	...\Queries	Query files
C:\PWRS\PD6\REPORTPL	...\Report Templates	Report templates
C:\PWRS\PD6\APPGEN	...\Application Generators	Templates for application generators
C:\PWRS\PD6\TOOLS	...\Tools	Tool files

* The three dots in a long name indicate the path: Program Files\Powersoft\PowerDesigner 6.

Short directory names in this book
You can install this product with long or short directory names. This book makes reference preferentially to short names. For example, the directory REPORTPL refers to the directory with the long name Report Templates.

Application Generation templates

The APPGEN (or Application Generators) directory contains separate directories for template sets for each AppModeler generator.

Directory	Long name	Generator
PB4TPL	PowerBuilder 4 Templates	AppModeler for PowerBuilder
PB5TPL	PowerBuilder 5 Templates	AppModeler for PowerBuilder
VB3TPL	Visual Basic 3.0 Templates	AppModeler for Visual Basic
VB4TPL	Visual Basic 4.0 Templates	AppModeler for Visual Basic
VB4TPL16	Visual Basic 4.0 16-bit Templates	AppModeler for Visual Basic
OP15TPL	Optima++ 1.5 Templates	AppModeler for Power++
DL20TPL	Delphi 2.0 Templates	AppModeler for Delphi
NITPL	NetImpact Dynamo Templates	AppModeler for the Web

Executable files

The PowerDesigner 6 (PD6) directory contains the following executable files:

Filename	Executable program
PDPB6.EXE	AppModeler for PowerBuilder
PDVB6.EXE	AppModeler for Visual Basic
PDVB4AD.EXE	AppModeler for Visual Basic Add-In
PDPP6.EXE	AppModeler for Power++
PDDL6.EXE	AppModeler for Delphi
PDDL2AD.DCU	AppModeler for Delphi Add-In
PDWEB6.EXE	AppModeler for the Web

Initialization parameters

If you install a 16-bit version of this product, its initialization parameters are stored in the PD6.INI file in the Windows directory.

If you install a 32-bit version of the this product, its initialization parameters are stored in the Registry in the following key:

```
HKEY_CURRENT_USER\Software\Powersoft\PowerDesigner 6
```

Using property sheets

Property sheets present the characteristics (properties) of objects in the PowerDesigner dictionary.

The title bar of the property sheet identifies the displayed object—for example, Model Properties or Table Properties.

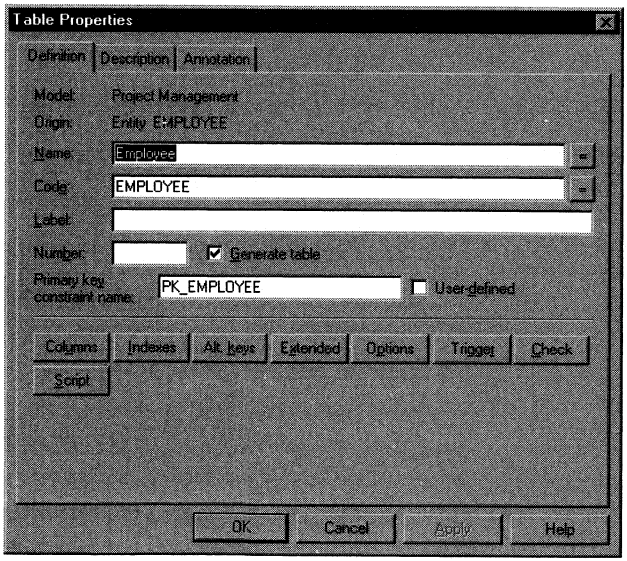
Most property sheets group properties onto three tabbed pages: Definition, Description, and Annotation. You access different pages by clicking their tabs.

Displaying a property sheet

❖ **To display a property sheet:**

- Double-click an object symbol in the model.
  or
  Click the property tool and click an object symbol in the model.

  The property sheet appears.



Validating or canceling changes to properties

❖ **To validate changes to properties:**

- Click the Apply button.
  or
  Click OK to close the property sheet.

❖ **To cancel changes to properties:**

- Click Cancel.

Using lists

Lists present the properties of all objects of the same type. For example, you can display the list of tables or columns.

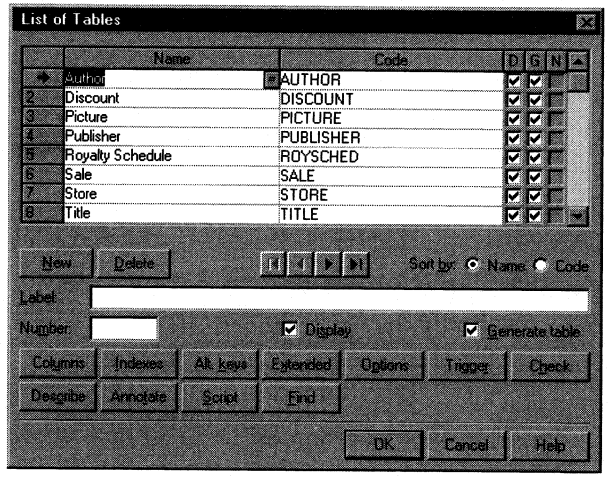
The title bar of the list identifies the object—for example, List of Tables.

Displaying a list

❖ **To display a list of objects:**

- Select Dictionary➤*List of* submenu.

  A list dialog box appears.



Modifying properties on a list

You can modify the properties of any object in a list. In addition, you can apply the same modifications to more than one object simultaneously.

❖ **To modify the properties of one object on a list:**

1 Select Dictionary➤*List of* submenu.

  A list dialog box appears.

2 Select an item in the list.

An arrow appears at the beginning of the line.

3 Type changes directly in the list columns and in the boxes that appear below the list.

4 Click OK.

❖ **To modify the properties of more than one object at a time:**

1 Select Dictionary➤*List of* submenu.

A list dialog box appears.

2 Click the number before the name of an item in the list.

The entire line is selected and an arrow appears at the beginning of the line.

3 Press CTRL while you click the numbers of other items in the list.

All clicked lines are selected.

4 Type changes in the boxes that appear below the list.

These changes apply to all selected objects.

5 Click OK.

Validating or canceling changes to a list

❖ **To validate changes to a list:**

♦ Click OK.

❖ **To cancel changes to a list:**

♦ Click Cancel.

Changing cursor position on a list

The following buttons change the cursor position on a list:

Button	Moves cursor to
	Top of list
	Previous item in list
	Next item in list
	End of list

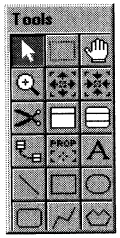
Arranging items on a list

In lists that do not have Sort By options, you can use the following buttons to modify the order of items:

Button	Moves selected items to
	Top of list
	Up one page
	Up one line
	Down one line
	Down one page
	Bottom of list

Using the tool palette

The buttons in the tool palette enact all major functions needed to build and modify a PDM, as follows:



Identifying tools in the tool palette

The following tables indicate the names and actions of all tools in the tool palette.

Selection and zoom tools

Tool	Name	Action
	Pointer	Select symbol
	Lasso	Select symbols in an area
	Grabber	Select and move all symbols
	Zoom	Select a zoom area
	Zoom in	Zoom in (increase view scale)
	Zoom out	Zoom out (decrease view scale)
	Scissors	Delete symbol

Object tools

Tool	Name	Action
	Table	Insert table symbol
	View	Insert view symbol
	Reference	Insert reference symbol
	Property	Display object property sheet

Graphic tools

Tool	Name	Action
	Text	Insert text
	Line	Draw a line
	Rectangle	Draw a rectangle
	Oval	Draw an oval
	Rounded rectangle	Draw a rounded rectangle
	Polyline	Draw a jagged line
	Polygon	Draw a polygon

Selecting and releasing tools in the tool palette

❖ **To select a tool in the tool palette:**

- Click the tool.

❖ **To release a tool in the tool palette:**

- Right click anywhere outside the tool palette.

  The Pointer tool is selected by default.

CHAPTER 2

Managing Models

About this chapter

This chapter describes model management functions.

Contents

Defining models

Opening an existing model

❖ **To open an existing model:**

1 Select File➢Open.

or

Click the Open tool.

A standard Windows file selection dialog box appears.

2 Select a file with the type PDM Model (*.PDM).

3 Click OK.

The model windows displays the selected model.

Creating a model

Creating a model requires that you do the following:

- Open a new file
- Identify its project
- Give the model a name and a code

After you create a model, you can enrich its definition by entering properties and associating objects.

❖ **To create a model:**

1 Select File➢New.

or

Click the New tool.

PowerDesigner opens a new window for the model.

2 Select Dictionary➢Model Properties.

The model property sheet appears.

3 If you are starting a new project with this model, type a new project name.

Later you apply this name to other models that you want to include in the same project.

4 If you are attaching this model to an existing project, type the project name.

This name identifies PDM and submodels that work together in a project.

5 Type the corresponding project code.

6 Type a model name and code.
or
Type a name and click the button at the end of the Code box.
or
Type a code and click the button at the end of the Name box.

7 Click OK.

Modifying model properties

The Model property sheet displays the definition of the current model. From this property sheet you can modify the model definition.

❖ **To modify the model properties:**

1 Select Dictionary➢Model Properties.

or
If you have inserted a title box, click the Property tool, and then the title box.

The model property sheet appears.

2 Type changes to model properties.

3 Click OK.

Saving and closing a model

❖ **To save the current model:**

♦ Select File➢Save.

or
Click the Save tool.

The first time you save a model, the Save As dialog box asks you to indicate a name and a path for the new model.

All subsequent times that you save a PDM, the previous version of the model is copied to a file with the PDB extension.

❖ **To save the current model under another name or directory:**

♦ Select File➤Save As.

This displays a standard Windows Save As dialog box.

❖ **To close the current model:**

♦ Select File➤Close.

PowerDesigner closes the work window for the current model. If you have made any changes since the last save of the current model, PowerDesigner prompts you to save any changes.

Deleting models

When you delete a model, you delete all its submodels. When you delete a submodel, the global model remains unchanged.

❖ **To delete a model:**

1 Select File➤Utilities➤Delete.

A file selection dialog box appears.

2 Select a model file (PDM file).

3 Click OK.

A confirmation box appears.

4 Click OK.

Sending a model via a messaging application

PowerDesigner uses the messaging application programming interface (MAPI) to send model files by electronic mail.

Via this interface, you use your internal messaging system to send model files directly to other team members.

This feature is not available in the Windows 3.11 environment.

❖ **To send a model using a messaging system:**

1 Select File➤Send.

A file selection dialog box appears.

2 Select a file.

3 Click OK.

The file transfers to your internal messaging system, which may ask you for additional information, such as a destination.

Using the dictionary

You can store your models in a dictionary using PowerDesigner Metaworks.

For complete setup instructions, see *PowerDesigner Installation Guide*.

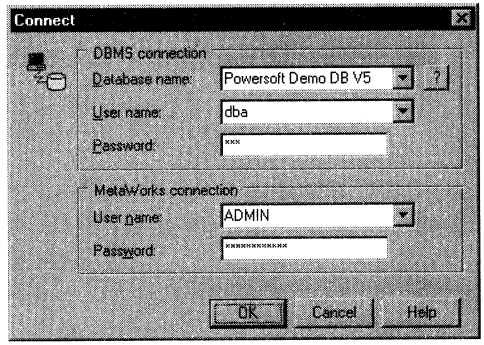
Extracting a model

When a dictionary stores a model, you open it by extracting it from the dictionary.

❖ To extract a model:

1 Select File➤Extract.

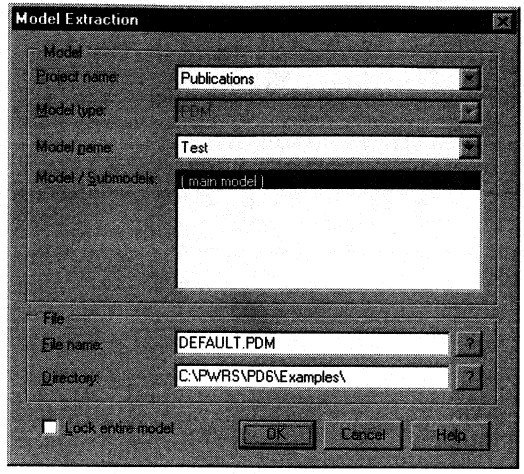
The Metaworks window opens and a Connect dialog box appears.



2 Type connection parameters.

3 Click OK.

An Extraction dialog box appears.



4 Select a project from the Project Name list.
Select PDM from the Model Type list.
Select a model name from the Model Name list.
Type a destination file name for the model.
Type a destination directory for the model.

5 Click OK.

The model window displays the selected model.

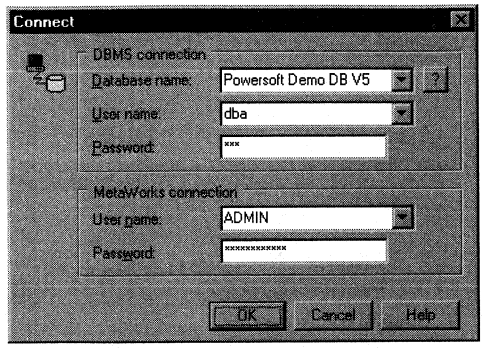
Consolidating a model

To save a model in a dictionary, you consolidate it.

❖ To consolidate a model:

1 Select File➤Consolidate.

The Metaworks window opens and a Connect dialog box appears.



2 Type connection parameters.

3 Click OK.

A Consolidation dialog box appears.

4 Verify model information.

If necessary, type changes to model information.

5 Click OK.

Setting model options

Model options apply to an entire model. You save them with the model.

The model options are:

- General options
- Name format
- Code format
- Text editor

For information on general options, see the chapter "Physical Data Model Basics."

Name and code formats

Options

In name and code formats, you can specify valid characters to accept and invalid characters to refuse.

Option	Description
Maximum length	Maximum number of characters
Uppercase Lowercase Mixed case	Indicates authorized letter cases
Valid characters	List of authorized characters
All	Accept all characters
Invalid characters	List of unauthorized characters
No Accent	Replaces accents with unaccented characters

Character selection

You can include valid and invalid characters as part of name and code formats.

The following syntax applies to valid and invalid characters.

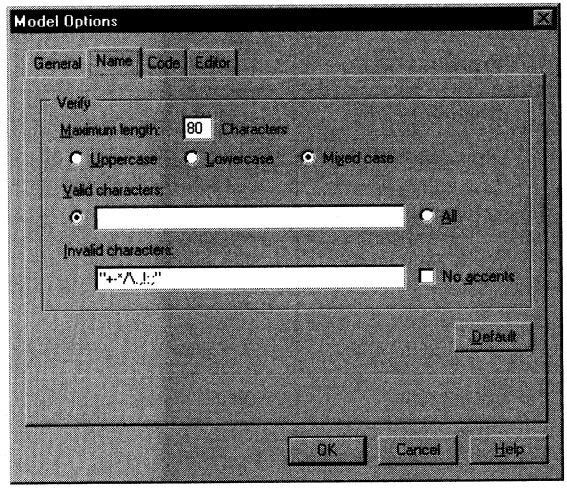
Character set	Syntax	Example
Interval of characters	Characters in single quotation marks separated by a dash	`'a'-'z'`
Single character	Character in quotation marks	`"a"`
Character string	String in quotation marks	`"xyz"`

❖ To define name and code formats:

1 Select File➤Model Options.

The Model Options dialog box appears.

2 Click the Name tab.

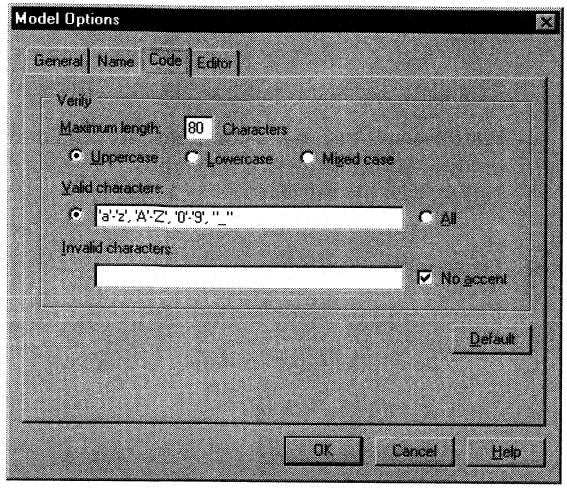


3 Type a maximum length.

4 Select uppercase, lowercase, or mixed case.

5 Type valid characters.
or
Select All.
Type invalid characters.
Select no accent if you want to replace accented characters.

6 Click the Code tab.



7 Repeat step 3-5 to set code format.

8 Click OK.

Using default name and code formats

By default, name format respects the ANSI table standard used by Windows and does not accept these characters: +-*/\.,!:;.

By default, code format accepts these characters: 'a'-'z', 'A'-'Z', '0'-'9', "_".

❖ To use default name and code formats:

1 Select File➢Model Options.

The Model Options dialog box appears.

2 Click the Name tab.

3 Click the Default button.

Default values appear.

4 Click the Code tab.

5 Click the Default button.

6 Click OK.

Selecting a text editor

You can choose a text editor for the current model.

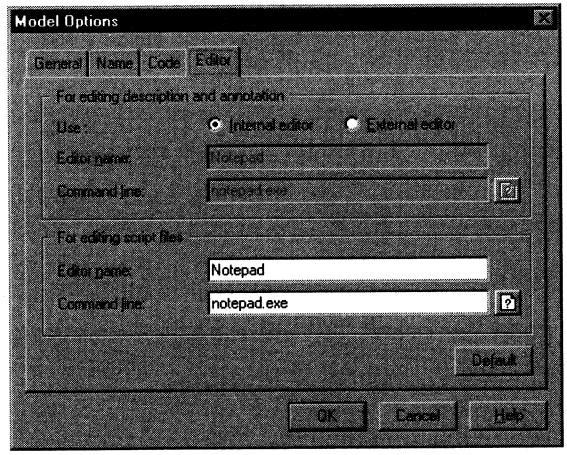
For descriptions and annotations, you can choose either the PowerDesigner internal editor or an external editor. For script files, you must choose an external editor. By default, Notepad is the external editor.

❖ To select a text editor:

1 Select File➤Model Options.

The Model Options dialog box appears.

2 Click the Editor tab.



3 Click Internal Editor for description and annotation.
or
Click External Editor for description and annotation.
Type the name of the editor in the Editor Name box.
Type the name of the editor program in the Command Line box.

4 Type the name of the editor for script files in the Editor Name box.

5 Type the name of the editor program in the Command Line box.

6 Click OK.

Microsoft Write or Word for Windows as external editors
If you choose Microsoft Write or Word for Windows as your external editor, you access them in text-only mode.

CHAPTER 3

Managing Submodels

About this chapter

This chapter discusses the uses of submodels.

Contents

Submodel support

Desktop versions of this product do not support submodels.

Defining submodels

Submodels help you manage large models. You can divide any model into several submodels.

A model that is the source of submodels is called a **global model**.

A submodel contains a selection of objects from the global model.

All of the same graphical operations are available in submodels and global models.

> **No submodels in desktop versions**
> If you are using a desktop version of this product, you cannot define submodels.

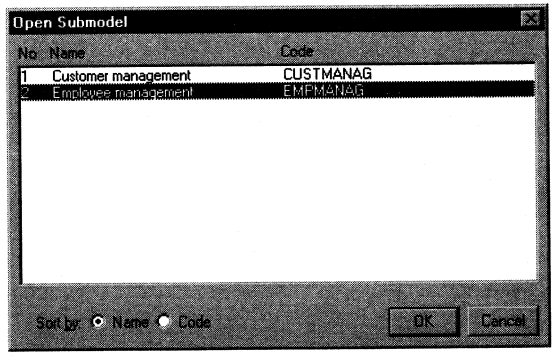
Opening a submodel

❖ **To open a submodel:**

1 Open a global model.

2 Select Dictionary➤Submodel➤Open.

A list of submodels appears.



3 Select a submodel.

4 Click OK.

Creating a submodel

You can create a submodel from either a global model or another submodel. In both cases, the new submodel links directly to the global model. A submodel does not depend on another submodel.

Objects that you select at the time of submodel creation are automatically placed in the new submodel window.

❖ To create a submodel:

1 Select objects in the current model that you want to include in the submodel.

> **Select connected symbols**
> You can select a group of symbols by selecting a central symbol then selecting Edit➤Select Connected Symbols.

2 Select Dictionary➤Submodel➤New.

A new window displays the submodel.

3 Select Dictionary➤Model Properties.

The model property sheet for the submodel appears.

4 Type a model name and code.
or
Type a name and click the button at the end of the Code box.
or
Type a code and click the button at the end of the Name box.

5 Click OK.

6 Select File➤Save.

You save the new submodel, as well as the global model.

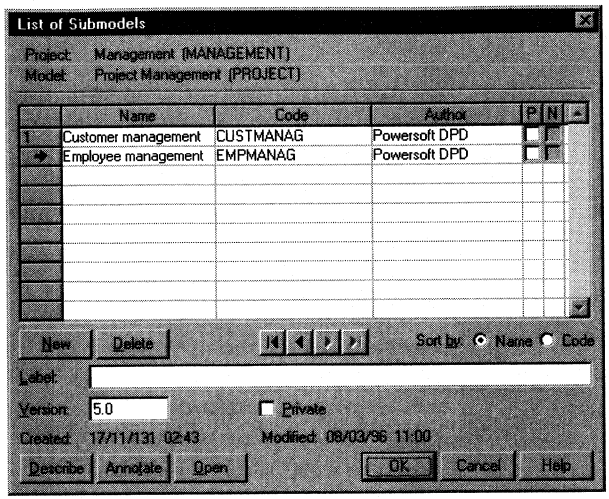
Listing submodels

You can display a list of the submodels attached to the current global model.

❖ To display the list of submodels:

♦ Select Dictionary➤Submodel➤List of Submodels.

The list of submodels appears.



Saving submodels

When you save a global model or any of its submodels, saved changes affect all related models.

❖ **To save submodels:**

- Select File➤Save from the global model or any of its submodels.

Deleting submodels

When you delete a submodel, its corresponding objects remain present in the global model.

❖ **To delete a submodel:**

1. Open a global model.
2. Select Dictionary➤Submodel➤List of Submodels.
3. Click the submodel on the list that you want to delete.

   An arrow appears at the beginning of the line.
4. Click the Delete button.
5. Click OK.

Sharing objects among related models

The global model automatically takes into account all changes you make in the submodel, from object definitions to domains and data items.

There is only one object in the dictionary for each object in the global model, regardless of how many times it appears in submodels.

You can freely copy and move symbols among related models, as follows:

- From a global model to a submodel
- From a submodel to a global model
- From a submodel to another submodel

Adding objects to a submodel

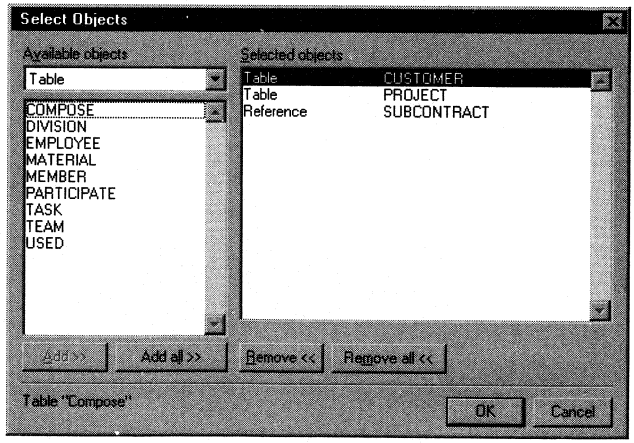
From a list of objects in the global model, you can select objects to appear in the current submodel.

❖ To add objects to a submodel:

1 Open a submodel.

2 Select Dictionary➤Submodel➤Add/Remove Objects.

The Select Objects dialog box displays a list of available and selected objects.



3 Select an object type from the dropdown listbox.

The list of available objects shows all objects of that type in the global model.

4 Select one or more objects.

5 Click Add.

The selected objects appear in the list of selected objects.

6 Click OK.

Removing objects from a submodel

From a list of objects in the submodel, you can select objects to remove from the current submodel.

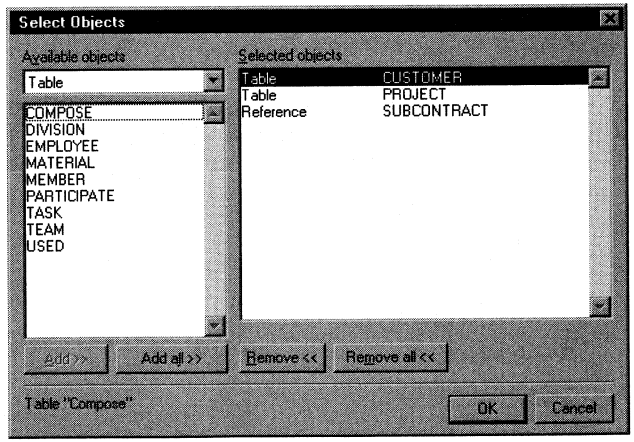
When you remove an object from a submodel, it remains in the global model. This action does not delete the corresponding object.

❖ To remove objects from a submodel:

1 Open a submodel.

2 Select Dictionary➤Submodel➤Add/Remove Objects.

The Select Objects dialog box displays a list of available and selected objects.



3 Select one or more objects from the list of selected objects.

4 Click Remove.

The selected objects disappear from the list of selected objects. They remain in the list of available objects for the global model.

5 Click OK.

Updating graphics in the global model

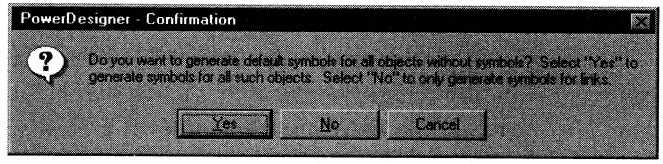
When you create an object in a submodel, the object is automatically created in the global model. However, the symbol of the new object is not added to the global model. You can update the display of the global model to reflect changes in its submodels.

If you create an object with graphical synonyms in a submodel, only one symbol appears in the updated global model.

❖ To update graphics in the global model:

1 Select Dictionary➤Submodel➤Update Graphics.

A confirmation box appears.



2 Click Yes to update all symbols.
or
Click No to update link symbols only.

Copying symbols to related models

You can copy object symbols and their synonyms among related models (a global model and its submodels).

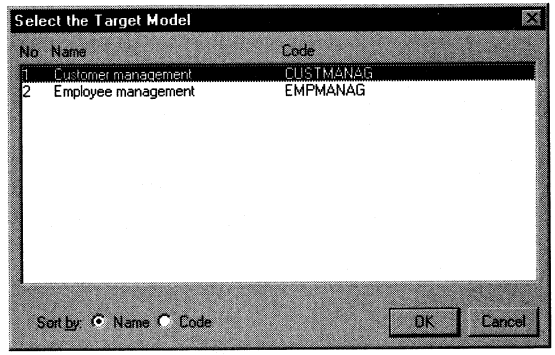
When you copy a symbol from one model to another, the symbol retains its display properties.

❖ To copy symbols to a related model:

1 Select one or more symbols in the model.

2 Select Dictionary➤Submodel➤Copy Symbols.

A dialog box asks you to select a target model from the list of related models.



3 Select a model in the list.

4 Click OK.

Moving symbols to related models

You can move object symbols and their synonyms among related models (a global model and its submodels).

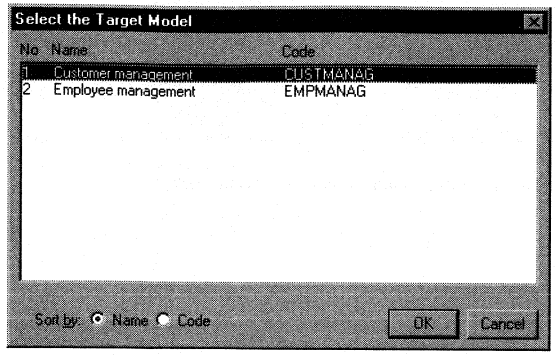
When you move a symbol from one model to another, the symbol retains its display properties.

❖ To move symbols to a related model:

1 Select one or more symbols in the model.

2 Select Dictionary➤Submodel➤Move Symbols.

A dialog box asks you to select a target model from the list of related models.



3 Select a model in the list.

4 Click OK.

Selecting symbols from a submodel

In the workspace of a global model, you can select all symbols that appear in a related submodel.

❖ **To select symbols from a submodel:**

1 Select Dictionary➤Submodel➤Select Symbols.

The list of submodels appears.

2 Click a submodel in the list.

3 Click OK

Symbols from the selected submodel appear selected in the global model.

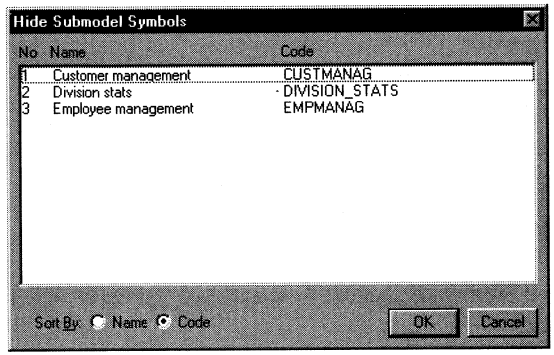
Hiding symbols from a submodel

In the workspace of a global model, you can hide all symbols in the global model that appear in a related submodel.

❖ **To hide symbols from a submodel:**

1 Select Dictionary➤Submodel➤Hide Symbols.

The list of submodels appears.



2 Click a submodel.

3 Click OK

Symbols from the selected submodel disappear from the global model.

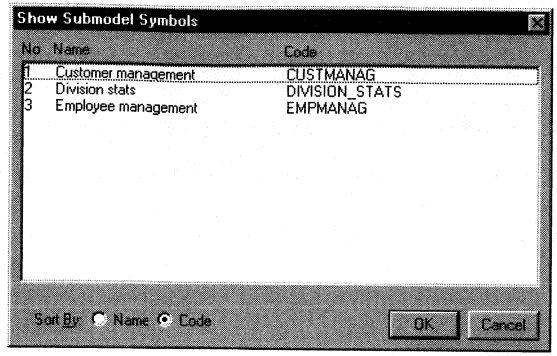
Showing hidden symbols from a submodel

In the workspace of a global model, you can show hidden symbols from a related submodel.

❖ **To show hidden symbols from a submodel:**

1 Select Dictionary➤Submodel➤Show Symbols.

The list of submodels appears.



2 Click a submodel.

3 Click OK.

Symbols from the selected submodel appear in the global model.

Update graphics for new symbols
The symbols of objects that you create in a submodel do not appear when you select Dictionary➤Submodel➤Show Symbols. They appear when you select Dicitonary➤Submodel➤Update Graphics.

CHAPTER 4

Managing Objects

About this chapter

This chapter explains the different ways to identify, delete, and copy objects in models.

Contents

Identifying objects

The first time you access the definition of any object in a model, the Name and Code boxes contain default values, and the Label box is empty. By putting significant information in these boxes, you make it possible to identify objects and their role in the overall project.

Naming objects

Names are like titles. During the design process, they clearly identify the project, its models, and objects.

By default, project, model, and object names have 80 characters maximum and can include uppercase and lowercase letters.

You define the names of models and objects on property sheets and lists.

Giving codes to objects

Codes are references for projects, models, and objects. These codes appear in scripts that result from the design process and feed the actual physical structure of a database.

By default, all codes have 80 characters maximum and include upper-case letters only.

You define the codes of models and objects on property sheets and lists.

Labeling objects

A label describes a model or an object. It provides more information than the name. For example "Organizing projects, employees, budgets" describes the model Project Management.

Labels are optional.

You define the labels of models and objects on property sheets and lists.

Attaching a description to an object

Descriptions provide detailed information about a model or the objects it contains.

In general, the description includes important information that does not fit into the boxes of the definition. For example, a description of the Employee table might read: *This table has one occurrence for each employee in our worldwide operations. This base should grow by 20 percent in 1996.*

The description is a free-text field that can be several lines long.

Depending on the object, you can add a description from a property sheet or a list of objects.

❖ To attach a description from a property sheet:

1 Double-click an object in the model to display the property sheet for an object.
or
Double-click the title box to display the property sheet for the model.

2 Click the Description tab.

3 Type a description.

> **Inserting a tab in a description**
> When the description page of a property sheet is open, you insert a tab in a description by pressing CTRL+TAB and you go to the next page by pressing CTRL+SHIFT+TAB.

4 Click OK.

❖ To attach a description from a list:

1 Select a list item from the Dictionary menu.

2 Click an item on the list.

An arrow appears at the beginning of the line.

3 Click the Describe button.

A text window appears.

4 Type a description.

5 Click OK.

Attaching an annotation to an object

The annotation contains your own notes regarding the implementation of a model or the objects it contains.

The annotation is a free-text field that can be several lines long. For example, an annotation of the Employee table might read: *Verify list of suggestions with Director of Human Resources.*

❖ To attach an annotation from a property sheet:

1. Double-click an object in the model to display the property sheet for an object.
   or
   Double-click the title box to display the property sheet for the model.
2. Click the Annotation tab.
3. Type an annotation.

   Inserting a tab in an annotation
   When the annotation page of a property sheet is open, you insert a tab in an annotation by pressing CTRL+TAB and you go to the next page by pressing CTRL+SHIFT+TAB.
4. Click OK.

❖ To attach an annotation from a list:

1. Select a list item from the Dictionary menu.
2. Click an item on the list.

   An arrow appears at the beginning of the line.
3. Click the Annotate button.

   A text window appears.
4. Type an annotation.
5. Click OK.

Deleting objects

When you delete object symbols, you need to indicate whether to simply delete the symbol from the model display or to also delete the associated object from the dictionary.

If a symbol is purely graphic You delete the symbol without confirmation.

If a symbol is associated with an object in the dictionary You choose to delete either the symbol and the associated object or the symbol only.

Confirming deletion

The following table describes the confirmation choices possible for object deletion.

Confirmation choice	Result in display	Result in dictionary
Delete symbol and associated object	Symbols disappear	Objects removed
Detach symbol	Symbols disappear	Objects remain

If you delete a symbol and its associated object, they disappear from all related models and from the dictionary.

If you detach a symbol from a model, it remains visible in other related models.

Deletion option

The model option Confirm Delete indicates whether or not to confirm object deletion in a global model as follows:

Deletion option	Model type	Result
Confirm selected	Global model and submodel	Choice to delete selected symbols and their associated objects in dictionary *or* detach selected symbols only
Confirm cleared	Global model	Delete selected symbols and their associated objects in dictionary
Confirm cleared	Submodel	Detach selected symbols from submodel, associated global model remains unchanged

❖ **To confirm deletions:**

1 Select File➤Model Options.

The Model Options dialog box opens to the General page.

2 Select the Confirm checkbox.

3 Click OK.

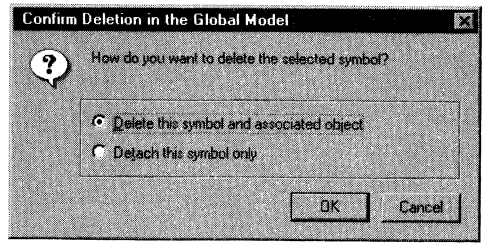
Deleting symbols

❖ **To delete symbols from a model:**

1 Click the Scissors tool and click the symbol to delete.
or
Select one or more symbols and double-click the Scissors tool.

If the symbol is purely graphic It disappears from the model.

If the symbol has an associated object in the dictionary A Confirm dialog box appears.



2 Click a confirmation choice.

3 Click OK.

Canceling last action
You can always select Edit➤Undo to cancel last action.

Deleting an object in a list

When you delete an object from a list, the type of deletion depends on whether you are displaying the list from a global model or from a submodel.

Model	Deletion from list
Global model	Delete symbol and associated object in the dictionary
Submodel	Detach symbols from submodel

❖ To delete an object from a list:

1 Select a list item from the Dictionary menu.

2 Click an item on the list.

An arrow appears at the beginning of the line.

3 Click the Delete button.

4 Click OK.

Deleting related objects

The following table describes the effects on related objects when you delete objects from the data dictionary.

Deleted object	Effect on related objects
Domain	Columns retain the domain's data type, but you can modify them individually
Table	Delete columns

Copying objects

There a two ways to copy an object:

- **Duplication** Creates a new object in the data dictionary
- **Synonym** Creates an additional symbol for an existing object

Duplicating an object

Duplicating an object creates a new object in the dictionary. The new object is almost an exact copy of the original.

The name must be unique for an object. A number follows the name in the object that results from duplication. For example, if you duplicate a table named Employee, the new table is named Employee2.

> **Same code for different references**
> The model options Unique Code determines the code assigned to references that result from duplication. You access model options by selecting File➢Model Options.

❖ **To duplicate an object in the data dictionary:**

1 Select an object.

2 Select Edit➢Duplicate.
or
Press CTRL+U.

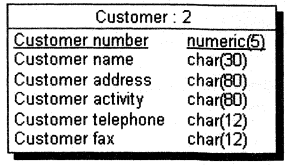
Creating a synonym for an object

A **synonym** is an additional symbol for a single object in a dictionary.

By reproducing the same object at different places in the model, synonyms can improve the readability of models by making links shorter.

In the model display, a synonym displays the name of the object followed by a colon and a number.

This table is a synonym for the table Customer:



❖ To create a synonym for an object:

1 Select an object.

2 Select Edit➢Create a Synonym.

CHAPTER 5

Using Business Rules

About this chapter

This chapter describes how business rules help you model information.

Contents

What is a business rule?

A business rule is actually a rule that your business follows. A business rule could be a government-imposed law, a customer requirement, or an internal guideline.

Starts as an observation

Business rules often start as simple observations, for example "customers call toll-free numbers to place orders." During the design process, they develop into more detailed expressions--for example, what information a customer supplies when placing an order or how much a customer can spend based on a credit limit.

Guides modeling

Business rules guide and document the creation of a model. For example, the rule "an employee belongs to only one division" can help you graphically build the link between an employee and a division.

Complements graphics

Furthermore, business rules complement model graphics with information that is not easily represented graphically. For example, some rules specify physical concerns in the form of formulas and validations rules. These technical expressions do not have a graphical representation.

Types of business rules

In PowerDesigner, there are different types of business rules:

Rule type	Describes	Example
Definition	Characteristics or properties of an object in the information system	A customer is a person identified by a name and an address
Fact	Certainty or existence in the information system	A client may place one or more orders
Formula	Calculation employed in the information system	The total order is the sum of all the order line costs
Validation	Constraint on a value in the information system	The sum of the order totals for a given client must not be greater than that client's allowance

Creating a business rule

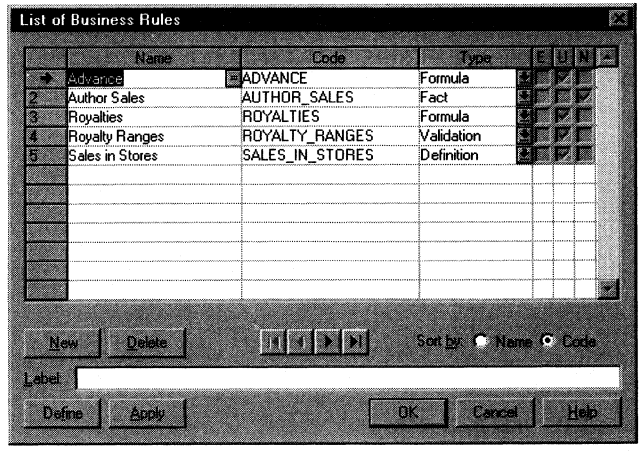
Before you create business rules, formulate your rules by asking yourself the following questions:

- What business problems do I want to address?
- Are there any procedures that my system must respect?
- Do any specifications dictate the scope of my project?
- Do any constraints limit my options?
- How do I describe each of these procedures, specifications, and constraints?
- How do I classify these descriptions: as definitions, facts, formulas, or validation rules?

❖ To create a business rule:

1 Select Dictionary➢List of Business Rules.

The list of business rules appears.



2 Click a blank line in the list.
or
Click the New button.

An arrow appears at the beginning of the line.

3 Type a name and a code.
or
Type a name and click the button in the code column.
or
Type a code and click the button in the name column.

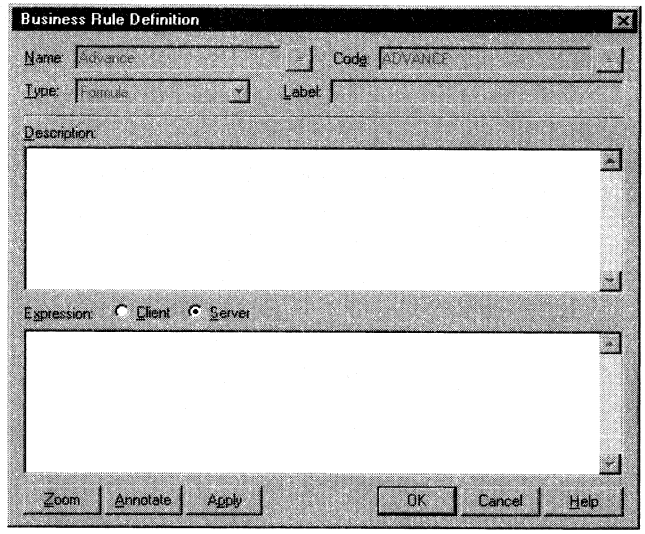
4 Click the Type column.

This column becomes a dropdown listbox.

5 Select a type from the dropdown listbox.

6 Click the Define button.

The Business Rule Definition dialog box appears.



7 Type a description of the business rule in the Description textbox.

8 Click OK in each of the dialog boxes.

Business rule properties

The business rule definition includes the following properties:

Property	Description	Maximum length	Unique in model?
Name	Name for the rule	80	✓
Code	Reference name for the rule	80	✓
Label	Descriptive label for the rule	254	—
Type	Indicates whether the rule is a definition, a fact, a formula, or a validation	—	—
Expression	Presence of associated expression	—	—
Used	Indicates whether the rule applies to an object	—	—
Notes	Presence of associated notes	—	—

Applying business rules to objects

From the list of business rules, you can apply a business rule to existing objects. You can also apply a business rule to objects from their property sheets or lists.

Applying a business rule from the list

❖ **To apply a business rule to objects:**

1 Select Dictionary➤List of Business Rules.

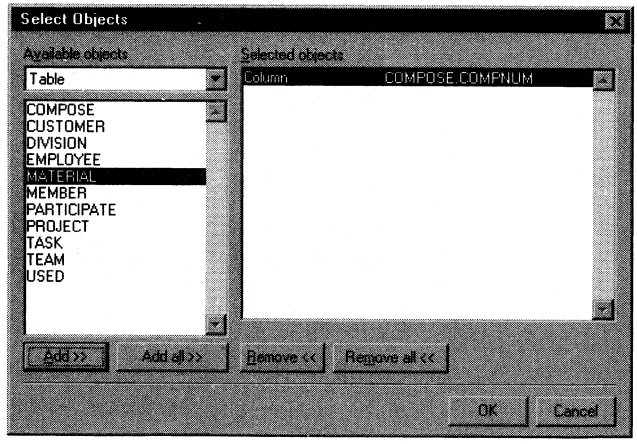
The list of business rules appears.

2 Click the rule that you want to apply.

An arrow appears at the beginning of the line.

3 Click the Attach button.

The Select Objects dialog box displays available and selected objects.



4 Select an object type from the dropdown listbox of available objects.

The list of objects of that type appears.

5 Select one or more objects from the list.

6 Click the Add button.

The selected objects appear in the Selected Objects list.

7 Click OK.

Applying a business rule to the current object

Starting from a property sheet, you can choose a rule to apply to the current object.

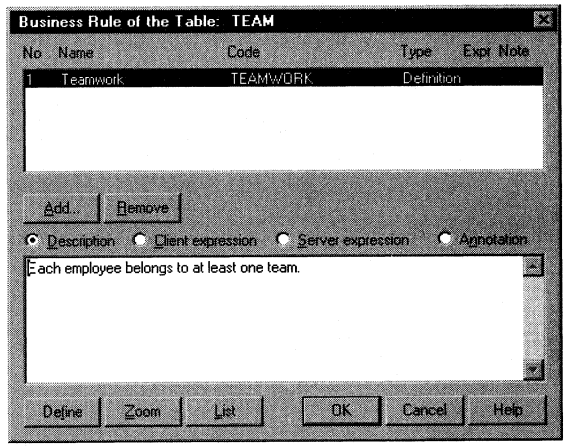
❖ To apply a business rule from a property sheet:

1 Double-click an object in your model.

The object property sheet appears.

2 Click the Rules button.
or
If the property sheet does not show a Rules button, click the Check button and then the Rules button.

A dialog box displays the business rules attached to the object.



3 Click the Add button.

A list of business rules appears.

4 Select one or more business rules.

5 Click OK in each of the dialog boxes.

You return to the model.

Using business rule expressions

A business rule starts out as a description. As you develop your model and analyze your business problem, you can complete a rule by adding a technical expression. The syntax of expressions depends on the target database or fourth-generation language (4GL) that you use

Each rule can include two types of expression:

Expression	Relates to	Syntax defined by
Client	Application	4GL
Server	Database	DBMS

You can physically generate expressions as follows:

- Generate validation rules that are attached to tables, domains, or columns in scripts as check parameters
- Insert rule expressions in triggers and stored procedures

Attaching an expression to a business rule

❖ **To attach an expression to a business rule:**

1 Select Dictionary➤List of Business Rules.

The list of available rules appears.

2 Click a rule.

An arrow appears at the beginning of the line.

3 Click the Define button.

The Business Rule Definition dialog box appears.

4 Click the Server radio button to define a server expression.
or
Click the Client radio button to define a client expression.

5 Type an expression in the Expression textbox.

6 Click OK in each of the dialog boxes.

Editing a business rule expression

You can use editing features to insert script language in a business rule expression.

A business rule expression can include the following categories of statements:

Category	Description
Function	Executes DBMS-specific function
Operator	Logical operator

It can also include the code for any of the following objects:

Tables
Columns
Joins

If you attach a business rule to a domain or a column, the business rule expression can include the following variables:

Variable	Value
%Domain%	Code of the domain to which the business rule is attached
%Column%	Code of the column to which the business rule is attached

❖ **To edit a business rule expression:**

1 Select Dictionary➤List of Business Rules.

The list of available rules appears.

2 Select a rule.

An arrow appears at the beginning of the line.

3 Click the Define button.

The Business Rule Definition dialog box appears.

4 Click the Server radio button to edit a server expression.
or
Click the Client radio button to edit a client expression.

5 Click inside the Expression textbox.

6 Click the Zoom button.

The SQL Query Editor window appears.

7 Click the position in the definition textbox where you want to insert the script language.

8 Select a category from the category list.

9 Select an item from the list for that category.

10 Click the Add button.

The selected item is inserted at the cursor position.

11 Click OK in each of the dialog boxes.

Generating a validation rule expression

You can generate validation rules that are attached to tables, domains, or columns in scripts as check parameters.

At generation, validation rule expressions can instantiate one of the following variables:

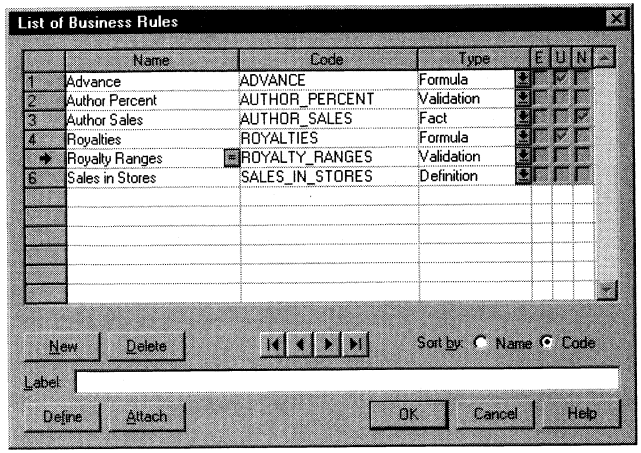
Variable	Description
%column%	Code of the column to which the business rule applies
%domain%	Code of the domain to which the business rule applies
%table%	Code of the table to which the business rule applies

❖ **To generate a validation rule expression:**

1 Select Dictionary➤List of Business Rules.

The list of business rules appears.

2 Select a validation rule in the list.



3 Click the Attach button.

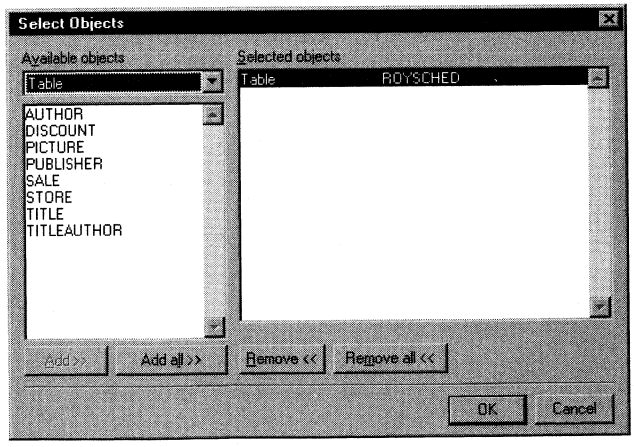
The Business Rule Definition dialog box displays available and selected objects.

4 Select Table from the Available Objects dropdown listbox.
or
Select Domain from the Available Objects dropdown listbox.
or
Select Column from the Available Objects dropdown listbox.

5 Select an object in the list.

6 Click Add.

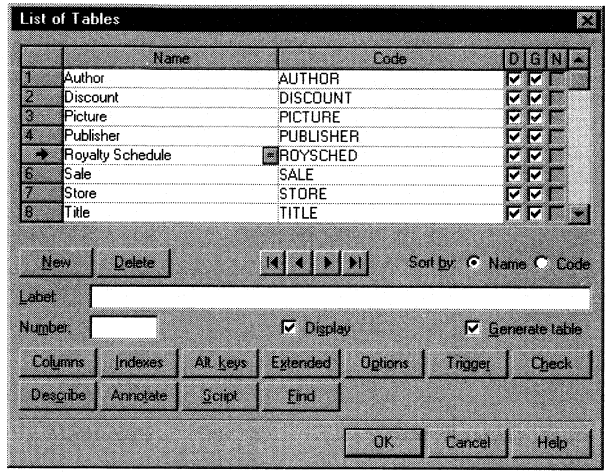
The table, domain, or column appears in the list of Selected Objects.



7 Click OK.

8 Select Dictionary➤List of Tables.
or
Select Dictionary➤List of Domains.
or
Select Dictionary➤List of Columns.

The list appears.



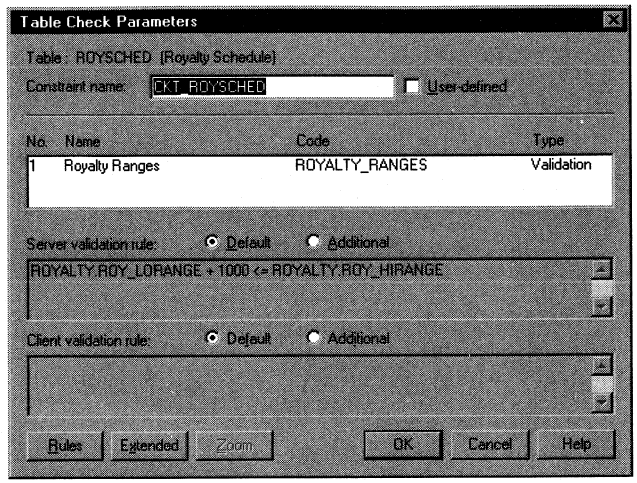
9 Select the object to which you applied the business rule.

10 Click the Check button.

The Check Parameters dialog box appears.

11 Click the Validation Rules tab.

The validation rule and its expressions appear on the Validation Rules page.



Validation rule expressions
You cannot modify the expression attached to a validation rule from the Check Parameters dialog box. You define validation rule expressions from the list of business rules, by clicking the Define button.

12 Click OK in each of the dialog boxes.

When you generate the database script, the validation rule expression appears in the resulting script for database generation, for example:

```
create table ROYSCHED
(
    TITLE_ISBN        char(10)                not null,
    ROY_LORANGE       T_QUANTITY                      ,
    ROY_HIRANGE       T_QUANTITY                      ,
    ROY_AMOUNT        T_AMOUNT                        ,
    primary key (TITLE_ISBN),
    check (ROYALTY.ROY_LORANGE + 1000 <=
ROYALTY.ROY_HIRANGE)
);
```

Inserting a rule expression in a trigger or stored procedure

You can a insert business rule expression in a trigger or stored procedure.

You have the following insertion options:

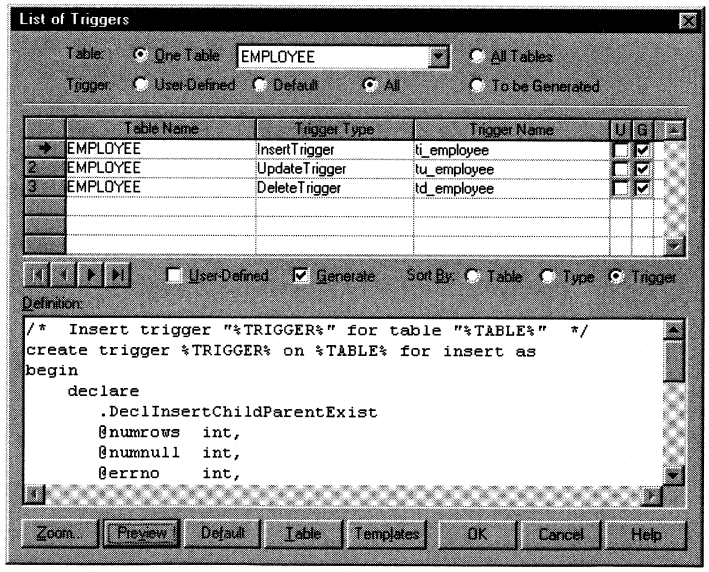
Insertion option	What it inserts
Add Client Expression Macro	Call to the client expression in the form: **.ClientExpression(***rule_code***)**
Add Server Expression Macro	Call to the server expression in the form: **.ServerExpression(***rule_code***)**
Add Client Expression Definition	Syntax of client expression
Add Server Expression Definition	Syntax of server expression

Inserting a rule expression in a trigger

❖ To insert a rule expression in a trigger:

1 Select Dictionary➤Triggers and Procedures➤List of Triggers.

The list of triggers appears.



2 Click the One Table radio button.
Select a table from the dropdown listbox.
Click the All radio button.

The list shows all the triggers attached to the selected table.

3 Select a trigger in the list.

4 Click the Zoom button.

The trigger definition dialog box displays the definition of the selected trigger.

5 Select Business Rules from the Category list.

The list of business rules that have server or client expressions appears in the Rule list.

6 Select a rule in the Rules list.

The description of the rule appears in the Description box.

7 Click the position in the definition textbox where you want to insert the business rule expression.

8 Click the Add button.

A context menu appears.

9 Select an insertion option from the context menu.

The expression is inserted in the Definition textbox.

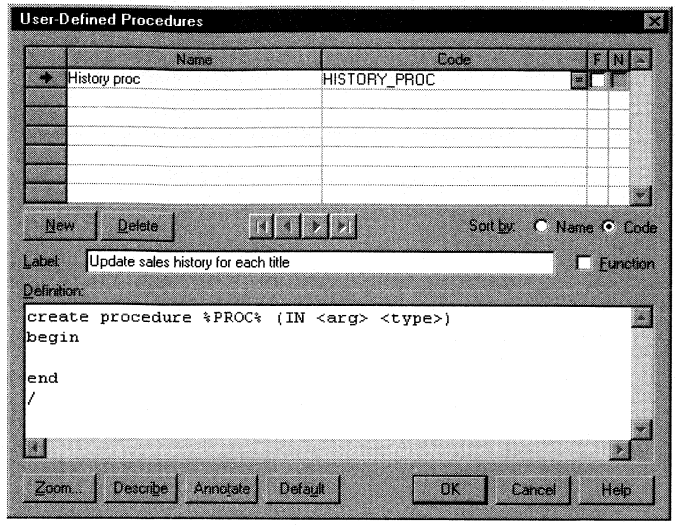
10 Click OK in each of the dialog boxes.

Inserting a rule expression in a stored procedure

❖ **To insert a rule expression in a stored procedure:**

1 Select Dictionary➤Triggers and Procedures➤List of Procedures.

The list of procedures appears.



2 Select a procedure in the list.

3 Click the Zoom button.

The procedure definition dialog box appears.

4 Select Business Rules from the Category list.

The list of business rules that have server or client expressions appears in the Rule list.

5 Select a rule in the Rules list.

The description of the rule appears in the Description box.

6 Click the position in the definition textbox where you want to insert the business rule expression.

7 Click the Add button.

A context menu appears.

8 Select an insertion option from the context menu.

The expression is inserted in the Definition textbox.

9 Click OK in each of the dialog boxes.

PART TWO

Physical Data Model

This part presents the Physical Data Model (PDM).

CHAPTER 6

Physical Data Model Basics

About this chapter

This chapter presents the Physical Data Model (PDM) and explains the role of physical modeling in the design process.

Contents

What is a PDM?

The PDM specifies the physical implementation of the database.

With the PDM, you consider the details of actual physical implementation. It takes into account both software or data storage structures. You can modify the PDM to suit your performance or physical constraints.

PDM roles

The PDM fills the following roles:

- Represent the physical organization of data in a graphic format
- Generate database creation and modification scripts
- Define referential integrity triggers and constraints
- Generate extended attributes
- Reverse engineer existing databases

PDM creation

There are several ways to create a PDM:

- Create a PDM from scratch
- Reverse engineer a database into a PDM

Objects in a PDM

A PDM represents the interaction of the following objects:

Object	Description
Table	Collection of rows (records) that have associated columns (fields)
Column	Data structure that contains an individual data item within a row (record), model equivalent of a database field
Primary key	Column or columns whose values uniquely identify a row in a table
Foreign key	Column or columns whose values depend on and migrate from a primary key in another table
Index	Data structure that is based on a key and that speeds access to data and controls unique values
Reference	Link between the primary key and the foreign key of different tables
View	Data structure that results from a SQL query and that is built from data in one or more tables

Defining a PDM

Defining PDM options

You can set the following PDM options:

Option	Description
Enforce	When selected, does not allow columns to diverge from domains for certain uses (see table below)
Default data type	Data type to apply to columns and domains if no data type is selected
Unique code	When selected, requires that references have unique codes
Auto-migrate FK	When selected, migrates foreign keys automatically
Confirm delete	When selected, offers choice between deleting symbols only or deleting symbols and their associated objects in the dictionary

You can enforce domains for the following uses:

Use	Columns in the domain cannot have divergent
Data type	Data type, length, and precision
Check	Check parameters
Rules	Business rules
Mandatory	Mandatory or optional property
Extended	Extended attributes

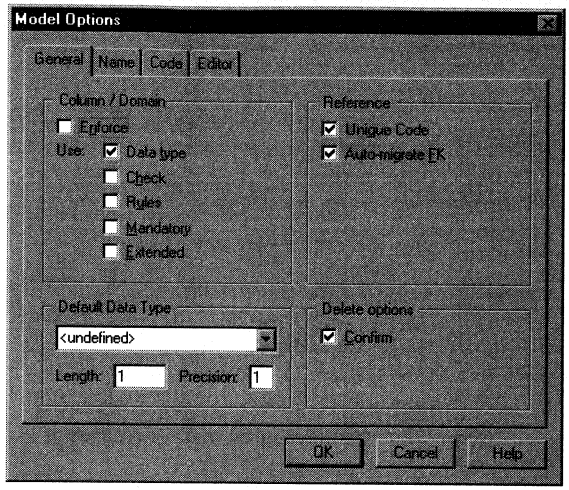
If you modify PDM options, these options apply to the current PDM and every PDM you create subsequently.

For information on other model options, see the chapter "Managing Models."

❖ To define PDM options:

1 Select File➤Model Options.

The Model Options dialog box opens to the General page.



2 Select PDM options.

3 Click OK.

PDM properties

The PDM definition includes the following properties:

Property	Description	Maximum length	In title box?
Project name	Name for the project. This name identifies PDM and submodels that work together in a project	80	✓
Project code	Reference name for the project	80	—
Name	Name for the model. This name makes the model identifiable	80	✓
Code	Reference name for the model. On the physical level, this code is generated in database scripts	80	—
Label	Descriptive label for the model	254	—
Author	Name or initials of the author of the model	80	✓
Language	Language of the model. For a limited number of languages, this property also indicates the language of the title box and of default report templates	—	✓
Version	Specifies the version of the model	8	✓
Created	Date and time of model creation	—	—
Modified	Date and time of last model modification	—	✓
File	Full pathname of the PDM file	—	—

Modifying model properties

The Model property sheet displays the definition of the current model. From this property sheet you can modify the model definition.

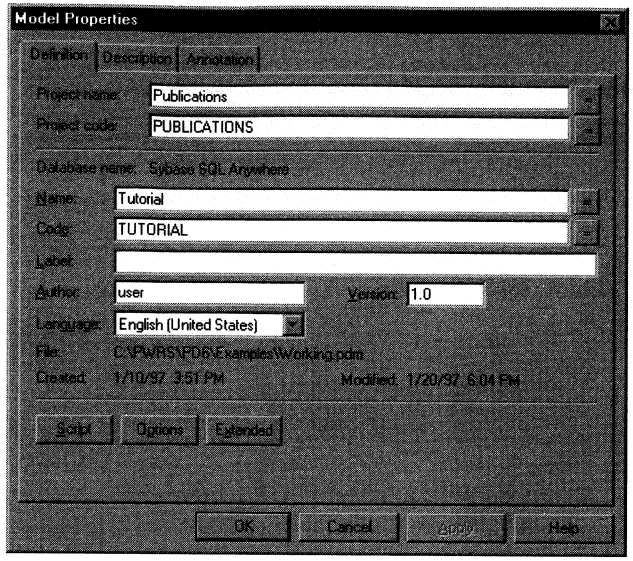
❖ **To modify the model properties:**

1 Select Dictionary➤Model Definition.

or

If you have inserted a title box, click the Property tool and then the title box.

The model property sheet appears.



2 Type changes to model properties.

3 Click OK.

CHAPTER 7

Building Physical Data Models

About this chapter

This chapter describes how to build a Physical Data Model (PDM). It explains the role of each object in a PDM. It also shows how to modify PDM objects.

Contents

Defining tables

A table represents a collection of data arranged in columns and rows. It is the model equivalent of a database table.

Creating a table

There are two approaches to creating a table.

- Insert a table symbol in the model
- Add a new table to the list of tables

Either way, you must assign a name and a code to the table.

❖ To create a table by inserting a symbol in the model:

1 Click the Table tool.

2 Click anywhere in the model window.

The following symbol appears at the click position:

At creation, a table is named Tab_*n*, where *n* is a number assigned in the order of the creation of objects.

3 Click the Pointer tool.

4 Double-click the new table symbol in the model.

The table property sheet appears.

5 Type a table name and a table code.
or
Type a name and click the  button in the Code column.
or
Type a code and click the  button in the Name column.

6 Click OK.

❖ To create a table by inserting it in the list:

1 Select Dictionary➢List of Tables.

The list of available tables appears.

2 Click a blank line in the list.
or
Click the New button.

An arrow appears at the beginning of the line.

3 Type a table name and a table code.
or
Type a name and click the  button in the Code column.
or
Type a code and click the  button in the Name column.

4 Click OK.

A symbol for this table is inserted in the current model.

Table properties

The table definition includes the following properties:

Property	Description	Maximum length	Unique in model?
Model	Name for the model	—	—
Origin	Name of the corresponding entity in the CDM, if the PDM is the result of generation from a CDM	—	—
Name	Name for the table. This name improves the readability of the model	80	✓
Code	Reference name for the table. This code is generated in database scripts	80	✓
Label	Descriptive label for the table	254	—
Number	Number of rows, either estimated number or actual number at the time of reverse engineering. This number is used to compute database size	—	—
Primary key constraint name	Name of the primary key constraint	30	✓
User-defined	Indicates a user-defined constraint name	—	—
Generate table	Indicates to generate the table in the database	—	—

The list of tables includes two additional properties in the form of checkboxes:

Property	Description
Display	Indicates whether or not this table is displayed in the model
Notes	Presence of associated notes

Modifying table properties

There are two approaches to modifying table properties:

- Modify a table property sheet
- Modify an entry in the list of tables

Accessing a table property sheet

The Table property sheet displays table definitions which you can modify.

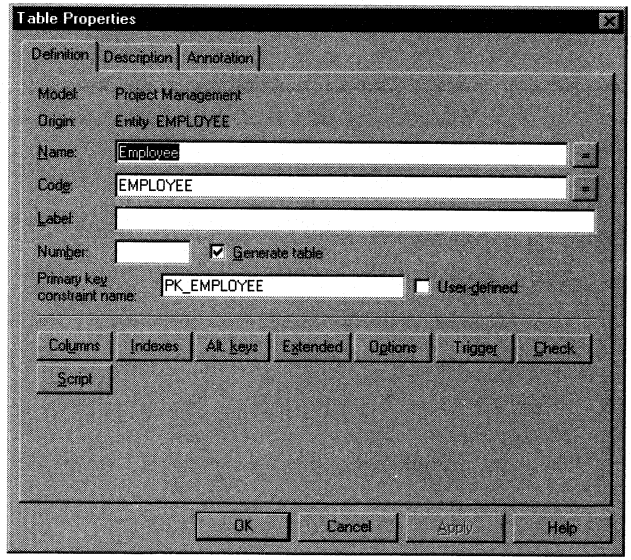
❖ To modify table properties from property sheet:

1 Double-click the table in the model with the Pointer tool.

or

Click the Property tool and click a table in the model.

The table property sheet appears.



2 Type changes to table properties.

3 Click OK.

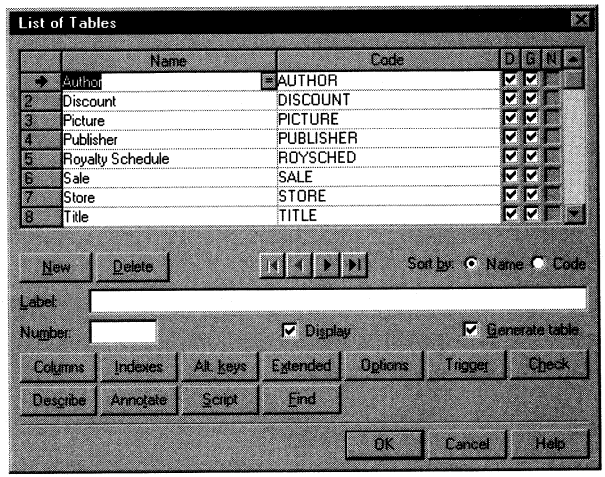
Accessing the list of tables

The list of tables includes all tables attached to the current model. You can modify certain table properties from the list.

❖ To modify table properties from the list:

1 Select Dictionary➤List of Tables.

The list of tables appears.



2 Click the table that you want to modify.

An arrow appears at the beginning of the line.

3 Modify the name or the code directly in the list.

4 Modify the label and the number in the boxes that appear below the list.

5 Select or clear the checkboxes Display and Generate Table.

6 Click OK.
or
Click a different table in the list.

List of tables for submodels
If you are working on a submodel, the list of tables only displays tables defined in that submodel.

If you create a table in a submodel, the table is also created in the global model.

Arranging items in a list attached to a table

Each table has it own list of columns, list of indexes, and list of alternate keys. These lists only include columns, indexes, and alternate keys for a particular table.

In a list attached to a table, you can arrange items in any order. This order becomes the order of generation in a database.

The following buttons change the position of selected items in the list.

Button	Moves selected items to
	Top of list
	Up one page
	Up one line
	Down one line
	Down one page
	Bottom of list

❖ **To arrange items in a list attached to a table:**

1 Double-click a table symbol.

The table property sheet appears.

2 Click the Columns button to display the list of columns attached to the table.
or
Click the Indexes button to display the list of indexes attached to the table.
or
Click the Alt. Keys button to display the list of alternate keys attached to the table.

3 Select one or more items in the list.

4 Use the arrow buttons to move the items in the list.

5 Click OK.

Naming a table constraint

A table constraint is a named check that enforces data requirements of check parameters.

Whenever you place a data restriction on a table, a constraint is created automatically. You have the option of specifying a name for the constraint. If you choose not to specify a name for the constraint, the database creates a default constraint name for you automatically.

This name helps you to identify and customize a table constraint in scripts for database creation and modification.

❖ **To name a table constraint:**

1 Double-click a table in the model.

The table property sheet appears.

2 Click the Check button.

The Check Parameters dialog box appears.

3 Type changes to the constraint name in the Constraint Name box.

The User-Defined checkbox is selected automatically.

> **Undo changes to a constraint name**
> You can always return to the default constraint name by clearing the User-Defined checkbox.

4 Click OK in each of the dialog boxes.

Displaying text in table symbols

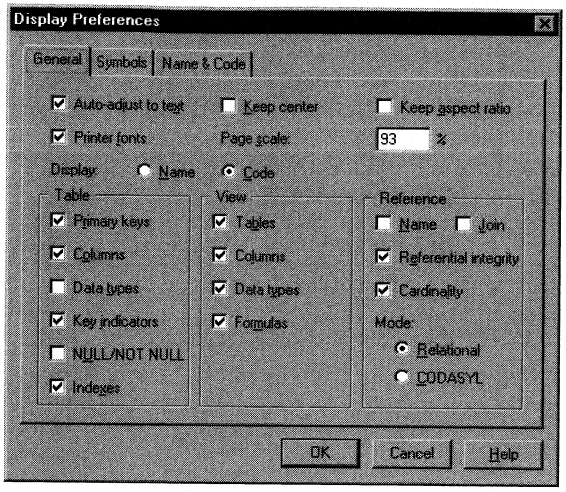
You can display the following text in a table symbol:

Text display preference	When selected, what it displays
Primary keys	Primary key columns, underlined
Columns	Columns that are not primary key columns
Data types	Data type for each column
Key indicators	<pk> and <fk> indicators next to primary key columns and foreign key columns, respectively
NULL/NOT NULL	Column indicator: null, not null, identity, or with default (DBMS-dependent)
Indexes	Indexes

❖ To display text in a table symbol:

1 Select File➤Display Preferences.

The Display Preferences dialog box opens to the General page.



2 Select or clear checkboxes in the Table groupbox.

3 Click OK.

Defining domains

Domains help you identify the types of information in your project. They define the set of values for which a column is valid. Applying domains to columns makes it easier to standardize data characteristics for columns in different tables.

In a PDM, you can associate the following information with a domain:

- Data type, length, and precision
- Check parameters
- Business rules
- Mandatory property
- Extended attributes

Creating a domain

❖ **To create a domain:**

1. Select Dictionary➢List of Domains.

   The list of available domains appears.

2. Click a blank line in the list.
   or
   Click the New button.

   An arrow appears at the beginning of the line.

3. Type a name and a code.
   or
   Type a name and click the  button in the Code column.
   or
   Type a code and click the  button in the Name column.

4. Select a data type as described below.

5. Click OK
   or
   Click another domain line.

Domain properties

Each domain definition includes the following properties:

Property	Description	Maximum length	Unique in model?
Name	Name for the domain	80	✓
Code	Reference name for the domain	80	✓
Used	Indicates if the domain is used by a column	—	—
Data type	Form of the data corresponding to the domain, such as numeric, alphanumeric, Boolean, or others	—	—
Length	Maximum number of characters	—	—
Precision	Number of places after the decimal point, for data values that can take a decimal point	—	—

Indicating data type, length, and precision

The data types that you can select in a PDM depend on your target database.

Length and precision

The properties length and precision do not apply to all data types. Furthermore, depending on data type, length may indicate a maximum or a fixed number of characters.

In the list of available data types, a variable indicates where you have to type a length or precision, as follows:

Variable	Replace with
%n	Length
%s	Length with precision
%p	Decimal precision

For example, if you are using Sybase SQL Anywhere and you choose the data type char(%n), you can choose a length of ten by typing char(10).

Undefined data type

All target databases allow you to select the <undefined> data type. The <undefined> data type indicates which domains remain without data types. If an <undefined> data type is present when you generate your database, it is replaced by the default data type for your database.

Selecting a data type for a domain

❖ **To select a data type for a domain:**

1 Select Dictionary➤List of Domains.

The list of domains appears.

2 Click the domain on the list that you want to define.

An arrow appears at the beginning of the line.

3 Click the button in the Data Type column.

4 Select a data type from the dropdown list.

> **Undefined data type**
> If you do not want to select a data type immediately, you can choose the <undefined> data type. When you generate the database, this data type is replaced by the default data type for your database, as defined in the DEF file.

5 Type the maximum number of characters for the column in the Length box.

6 If the data type can include values that take a decimal point, type the number of places after the decimal point in the Precision box.

7 Click OK.

Displaying columns that use a domain

If you attach a column to a domain, the domain supplies the data type and related data characteristics in the column definition.

Each domain must indicate a data type. It may also indicate length, decimal precision, check parameters, business rules, and extended attributes.

You can attach many columns to the same domain. In the list of domains, domains that have attached columns show the checkbox in the U column selected.

❖ **To display the list of columns that use a domain:**

1 Select Dictionary➤List of Domains.

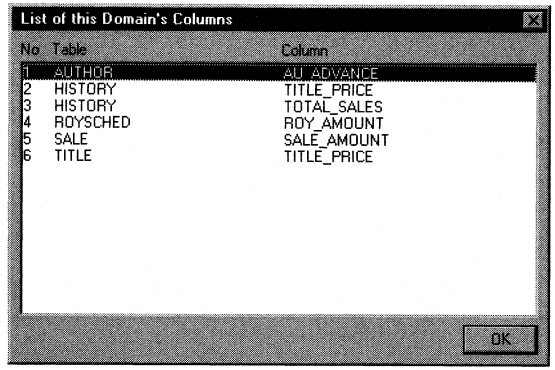
The list of domains appears.

2 Click a domain with a selected checkbox in the U column.

An arrow appears at the beginning of the line.

3 Click the Used By button.

A dialog box displays the list of columns that use the current domain.



4 Click OK.

You return to the list of domains.

Enforcing domains in a PDM

You can choose to enforce domains.

Your choice of whether or not to enforce domains has the following results:

Domain enforcement	Result
Not enforced	Columns that are divergent from the domain definition can remain attached to the domain
Enforced	Columns that are divergent from the domain (for certain uses) must be detached from the domain

You can enforce domains for the following uses:

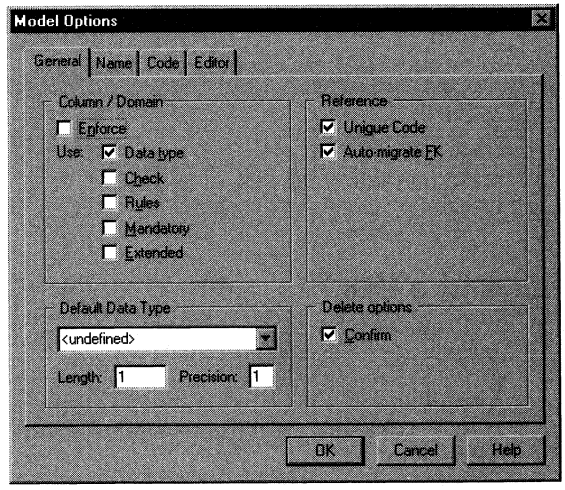
Use	Columns in the domain cannot have divergent
Data type	Data type, length, and precision
Check	Check parameters
Rules	Business rules
Mandatory	Mandatory or optional property
Extended	Extended attributes

If you modify domain enforcement, this option applies to the current PDM and every PDM you create subsequently.

❖ To enforce domains:

1 Select File➤Model Options.

The Model Options dialog box opens to the General page.



2 Select the Enforce checkbox.

3 Select checkboxes for uses that you want to enforce.

4 Clear checkboxes for uses that you do not want to enforce.

5 Click OK.

Defining columns

A column contains an individual data item within a row. It is the model equivalent of a database column.

Creating a column

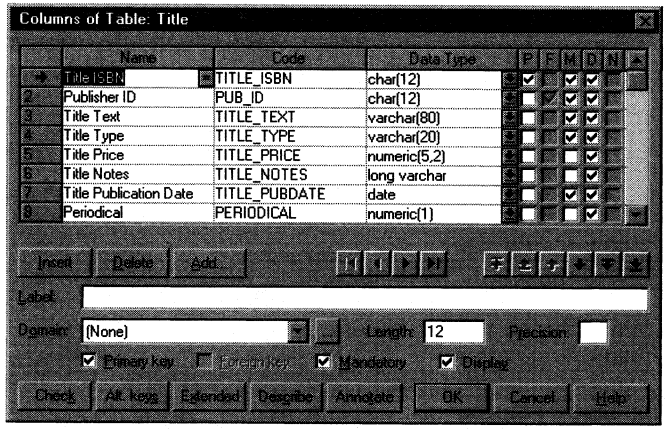
You always attach a column to a table.

❖ To create a column:

1 Double-click a table in the model with the Pointer tool.
or
Click the Property tool and click a table in the model.

2 Click the Columns button.

A dialog box lists columns in the table.



3 Click a blank line in the list.
or
Click the Insert button.

An arrow appears at the beginning of the line.

4 Type a name and a code.
or
Type a name and click the button in the Code column.
or
Type a code and click the button in the Name column.

5 Click OK.

Column properties

Each column definition includes the following properties:

Property	Description	Maximum length
Name	Name for the column. This name improves the readability of the model	80
Code	Reference name for the column	80
Data type	Form of the data corresponding to the column, such as numeric, alphanumeric, boolean, or others	—
Domain	Name of the associated domain	—
Primary key	When selected, designates a column whose values uniquely identify a row in the table	—
Foreign key	When selected, designates a column that depends on and migrates from a primary key column in another table	—
Mandatory	When selected, indicates a column that must be assigned a value	—
Display	When selected, indicates a column that is displayed in the model	—
Identity	(For Sybase System 10, Sybase System 11, and MS SQL Server 6 only) When selected, indicates that the column is an identity column	—
With default	(For Ingres, DB2, SQLBase, Xdb, Datacom, and ASK only) When selected, indicates that the column has a default value	—
Column N	When selected, indicates the presence of associated notes	—

Attaching a column to a domain

If you attach a column to a domain, the domain supplies the data type and related data characteristics. It may also indicate check parameters, business rules, and extended attributes.

❖ **To attach a column to a domain:**

1 Double-click a table in the model with the Pointer tool.

or

Click the Property tool and click a table in the model.

2 Click the Columns button.

A dialog box lists columns associated with the table.

3 Click a column in the list.

An arrow appears at the beginning of the line.

4 Select a domain from the Domain dropdown listbox.

5 Click OK.

In the Type column, the domain's data type replaces the data type previously defined for the column.

Selecting a data type for a column

There are two ways to select a data type:

- **Attach the column to a domain** The domain dictates a data type, a length, and a level of precision, as well as optional check parameters
- **Do not attach the column to a domain** You select a data type along with a length, a level of precision, and optional check parameters

> **About check parameters**
> Check parameters indicate data ranges and validation rules. You can set check parameters for domains, tables, and columns.

❖ **To select a data type for a column:**

1 Double-click a table in the model with the Pointer tool.

or

Click the Property tool and click a table in the model.

2 Click the Columns button.

A dialog box lists columns associated with the table.

3 Click the Type column in the line of the column that you want to modify.

An arrow appears at the beginning of the line.

4 Click the button in the Type column.

5 Select a data type from the dropdown list.

> **Undefined data type**
> If you do not want to select a data type immediately, you can choose the <undefined> data type. When you generate the database, this data type is replaced by the default data type for your database, as defined in the DEF file.

6 Type the maximum number of characters for the column in the Length box.

7 If the data type can include values that take a decimal point, type the number of places after the decimal point in the Precision box.

8 Click OK.

The change of data types appears in the list of columns.

Duplicating a column

You can duplicate a column from one table and add it to another table.

❖ **To duplicate a column:**

1 Double-click a table in the model with the Pointer tool and click the Columns button.

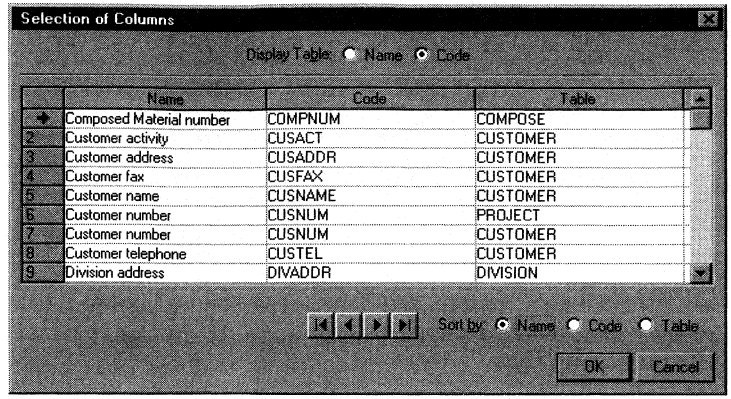
or

Click the Property tool and press CTRL while you click a table in the model.

A Columns of Table dialog box lists columns in the table.

2 Click the Add button.

The Selection of Columns dialog box lists all columns attached to other tables in the PDM.



3 Select one or more columns in the list.

4 Click OK.

The duplicated columns appear in the list of columns for the current table.

5 Click OK.

Making a column mandatory or optional

If a column is mandatory, every table row must have a value for that column. If a column is not mandatory, table rows do not have to have a value for that column.

By default, the only column that is mandatory is the primary key.

❖ To define a column as mandatory or optional:

1 Select Dictionary➤List of Columns.

The list of column appears.

2 Click a column in the list.

An arrow appears at the beginning of the line.

3 Select the Mandatory checkbox to make the column mandatory.
or
Clear the Mandatory checkbox to make the column optional.

You cannot clear the Mandatory checkbox for a primary key column because a primary key column must always be mandatory.

4 Click OK.

Naming a column constraint

A column constraint is a named check that enforces data requirements of check parameters.

Whenever you place a data restriction on a column, it generates a constraint automatically. You have the option of specifying a name for the constraint. If you do not specify a name for the constraint, PowerDesigner creates a default constraint name automatically.

This name helps you to identify and customize a column constraint in scripts for database creation and modification.

❖ To name a column constraint:

1 Select Dictionary➤List of Columns.

The list of column appears.

2 Click a column in the list.

An arrow appears at the beginning of the line.

3 Click the Check button.

The Check Parameters dialog box appears.

4 Type changes to the constraint name in the Constraint Name box.

The User-Defined checkbox is selected automatically.

> **Undo changes to a constraint name**
> You can always return to the default constraint name by clearing the User-Defined checkbox.

5 Click OK in each of the dialog boxes.

Configuring the display of the list of columns

❖ To configure the display of the list of columns:

1 Select Dictionary➤List of Columns.

2 Select a Display Table radio button:

Radio button	Displays
Name	Table names in the Table column
Code	Table codes in the Table column

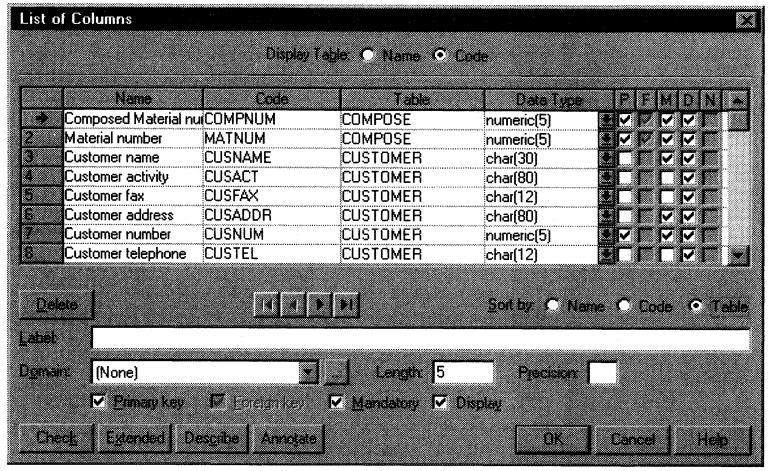
3 Select a Sort By radio button:

Radio button	Sorts alphabetically by
Name	Name of the column
Code	Code of the column
Table	Name or code of the table in the Table column

4 Click OK.

Example

The following list of columns is sorted by table code.



Defining references

References link the primary key in a parent table to the foreign key in a child table.

Reference properties

The definition of a reference link includes the following properties:

Property	Description	Maximum length
Name	Name for the reference. This name improves the readability of the model	80
Code	Reference name for the reference link	80
Label	Descriptive label for the reference	254
Primary key columns	Codes of primary key columns in the parent table	80
Foreign key columns	Codes of foreign key columns in the child table	80
Generate reference	Indicates to generate the reference in the database	—

Defining a code option for references

You can define the following code option for references:

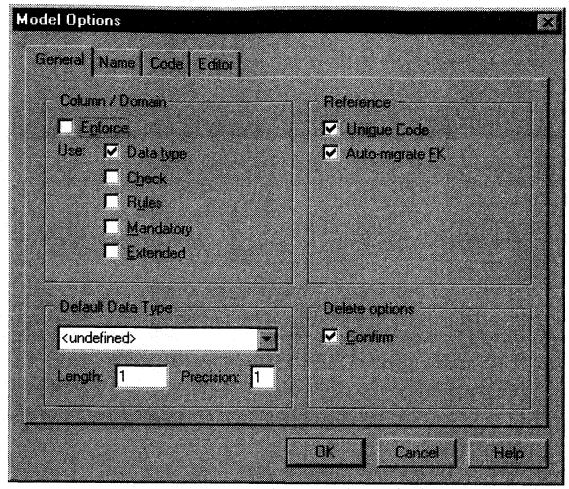
Option	Result when selected	Result when cleared
Unique code	Each reference has a unique code	Different references can have the same code

If you do not select Unique Code, two references can have the same code. In this case, you differentiate the references by the tables they link.

❖ **To define a code option for references:**

1 Select File➤Model Options.

The Model Options dialog box opens to the General page.



2 Select the Unique Code checkbox in the Reference groupbox to require a unique code for each reference.
or
Clear the Unique Code checkbox in the Reference groupbox to allow more than one reference to have the same code.

3 Click OK.

Creating a reference

When you create a reference between two tables, the primary key in the parent table becomes a foreign key in the child table.

If a column in the child table has the same name as the primary key in the parent table, the reference automatically selects this column.

❖ To create a reference:

1 Click the Reference tool in the tool palette.

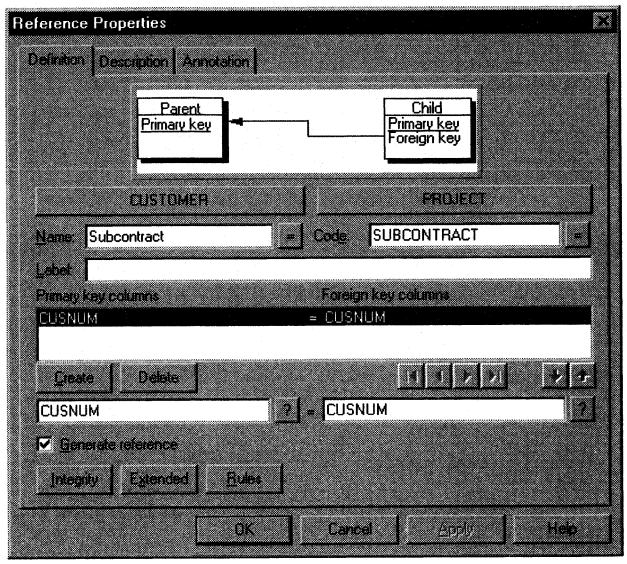
2 Drag and drop the reference link from the child table to the parent table containing the primary key to migrate.

The link appears between the two tables.

> **Dragging a reference to a different table**
> You can change the table at either end of a reference by clicking the reference and holding down CTRL as you drag one of its attach points to a different table.

3 Double-click the new link in the model.

The reference property sheet appears.



4 Type a name and a code for the reference.
or
Type a name and click the button after the Code column.
or
Type a code and click the button after the Name column.

5 Click the button under the Primary key column.

The list of keys in the parent table appears.

6 Double-click the primary key.

7 Click the button under the Foreign key column

The list of keys in the child table appears.

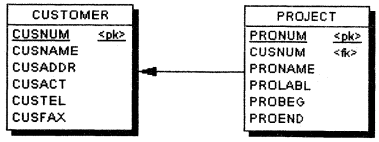
> **Automatic identification of foreign key**
> If there is a column in the child table with the same name as the primary key in the parent table, the column in the child table becomes a foreign key.

8 Click the key you want to designate as a foreign key.
Click OK.

9 Select the Generate Reference checkbox to generate the reference.
or
Clear the Generate Reference checkbox if you do not want to generate the reference.

10 Click OK.

In this model, a reference links the Customer table to the Project table by the column CUSNUM.



Sorting the list of references

The list of references offers the following sort options:

Option	Description
Name	Sorts alphabetically by the name of the reference
Code	Sorts alphabetically by the code of the reference
Parent	Sorts alphabetically by the name of the parent table
Child	Sorts alphabetically by the name of the child table

❖ To display and sort the list of references:

1 Select Dictionary➢List of References.

2 Select the radio button Name, Code, Parent, or Child.

The list sorts references according to your choice.

3 Click OK.

Rebuilding references

The procedure for rebuilding references does the following:

- Deletes existing references
- Generates new references based on column names and structures

❖ **To rebuild references:**

1 Select Dictionary➤Rebuild References.

A confirmation box asks you to confirm your choice.

2 Click Yes.

Using referential integrity

Referential integrity refers to rules governing data consistency, specifically the interaction between primary keys and foreign keys in different tables.

Referential integrity dictates what happens when you update or delete a value in a referenced column in the parent table and when you delete a row containing a referenced column from the parent table.

You can define referential integrity for:

- Particular references using a reference property sheet
- All references using a trigger

Referential integrity as a generation option
For certain target databases you can define referential integrity as a generation option. However many databases do not accept referential integrity as a generation option (in a trigger or a declaration). In these cases, when you generate a database generation script, it does not include the definition of referential integrity.

Referential integrity properties

The definition of referential integrity includes the following properties:

Property	Description	Maximum length
Name	Name of the reference	80
Code	Reference name of the reference link	80
Constraint name	Name of the referential integrity constraint	30
User-defined	Indicates a user-defined constraint name	—
Minimum	Minimum number of instances of a child table for each instance of a parent table	—
Maximum	Maximum number of instances of a child table for each instance of a parent table	—
Update constraint	How updating the primary key value in the parent table affects the foreign key value in the child table	—
Delete constraint	How deleting a row in the parent table affects the child table	—
Mandatory parent	Each foreign key value in the child table must have a corresponding primary key value in the parent table	—
Change parent allowed	A foreign key value in the child table can change to select another primary key value in the parent table	—

Cardinality

Minimum and maximum cardinality can take any of the following values:

Cardinality	Minimum	Maximum
0	A parent does not have a minimum number of children. Child is optional	—
1	At least one child must exist for each parent	Only one child can exist for each parent
any integer	Minimum number is the number indicated	Maximum number is the number indicated
n	—	Any number of children can exist for a each parent

Update and delete constraints

Update constraint and delete constraint take any of the following values:

Value	Change to parent table	Result in child table
None	Update or delete parent value	None
Restrict	Cannot update or delete parent value if one or more matching child values exist	None
Cascade	Update or delete parent value	Update or delete matching child values
Set null	Update or delete parent value	Set matching child values to NULL
Set default	Update or delete parent value	Set matching child values to default value

Defining referential integrity

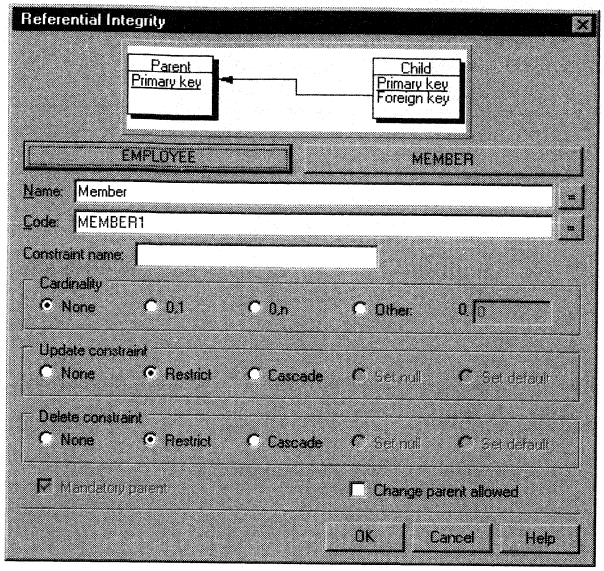
❖ **To define referential integrity:**

1 Double-click a reference in the model.

The reference property sheet appears.

2 Click the Integrity button.

The Referential Integrity dialog box appears.



3 (Optional) Type a constraint name.

If you do not specify a constraint name, PowerDesigner creates a default constraint name automatically.

4 Select values for Cardinality, Update Constraint and Delete Constraint.

5 Select Mandatory Parent if you want each row in the child table to have a corresponding row in the parent table.

6 Select Change Parent if you want to be able to update the foreign key value in the child table.

7 Click OK.

Displaying text with reference symbols

You can display the following text with reference symbols:

Label	Displays
Name	Name or code of the reference
Join	Statement of keys that are shared by the two tables
Referential integrity	Update and delete referential integrity constraints
Cardinality	Maximum and minimum number of child tables per parent table

Referential integrity label

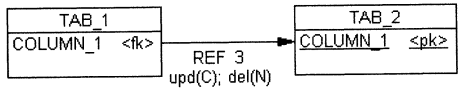
The referential integrity label indicates update and delete integrity, as follows:

Label	Referential integrity
upd()	Update
del()	Delete

A letter between the parentheses indicates the type of constraint, as follows:

Label	Referential integrity
( )	None
(R)	Restrict
(C)	Cascade
(N)	Set null
(D)	Set default

For example, the following referential integrity label indicates cascade on update and set null on delete.



Cardinality label

The cardinality label indicates the minimum and maximum number of children as follows:

[*minimum*,*maximum*]

For example, the cardinality label [0,n] indicates that any number of children is acceptable.

❖ **To display text with reference symbols:**

1 Select File➤Display Preferences.

The Display Preferences dialog box opens to the General page.

2 Select or clear the checkboxes in the Reference groupbox.

3 Click OK.

Moving text on a reference symbol
When a reference symbol displays text, the text position is based on the position of handles. You can add a handle on the reference symbol by pressing CTRL while you click the symbol.

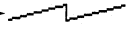
Modifying a reference graphically

From the PDM graphic, you can modify a reference as follows:

- Bend a reference symbol
- Straighten a reference symbol
- Drag a reference to a different table
- Select reference display mode

Bending a reference symbol

❖ **To bend a reference symbol:**

1 Select Format➤Line Style➤  .

2 Press CTRL while you click a point on the reference symbol where you want to insert an angle.

This point becomes a handle.

3 Release CTRL.

4 Drag and drop the handle.

Straightening a reference symbol

❖ **To straighten a reference symbol:**

1 Click a reference symbol that has angles.

Handles appear on the reference symbol.

2 Press CTRL while you click a handle.

The handle and its angle disappear.

Dragging a reference to a different table

❖ **To drag a reference to a different table:**

1 Click a reference symbol.

2 Press CTRL while you drag one of its attach points to a different table.

Selecting a reference display mode

❖ **To select a reference display mode:**

1 Select File➢Display Preferences.

The Display Preferences dialog box opens to the General page.

2 Select one of the following radio buttons:

Mode	Reference points
Relational	From child to parent
CODASYL	From parent to child

3 Click OK.

Defining keys

A key is made up of one or more columns such that each of its values corresponds to one and only one row in the table. Each key can generate a unique index or a unique constraint in a target database.

The PDM supports three types of keys:

Key	Description
Primary	Column or combination of columns whose values uniquely identify a row in a table
Foreign	Column or combination of columns whose values are required to match a primary key in some other table
Alternate	Column or combination of columns, not the primary key columns, whose values uniquely identify a row in a table

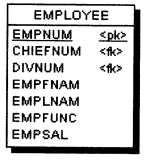
Designating a primary key

A primary key is made up of one or more columns unique to the table, such that each value of the primary key corresponds to one and only one row in the table.

Every table must have a primary key, composed of one or more of its columns.

In the PDM, columns that are primary keys by default appear underlined.

In the table pictured here, Employee number is the primary key for the table Employee. This means that each employee must have one unique employee number.



You define a column as the primary key of a table from the list of columns.

❖ **To designate a primary key:**

1 Double-click a table in the model with the Pointer tool.
or
Click the Property tool and click a table in the model.

2 Click the Columns button.

A dialog box lists columns associated with the table.

3 Select one or more columns in the list.

4 Select the checkbox in the P column.
or
Select the Primary key checkbox in the bottom half of the dialog box.

5 Click OK.

Designating a foreign key

A foreign key is a primary key that migrates from another table.

Every reference links the primary key in one table to the foreign key of another.

❖ **To designate a foreign key column:**

1 Click the Reference link tool in the tool palette.

2 Drag and drop a reference link from the child table to the parent table which contains the primary key that you want to migrate.

The link appears between the two tables.

3 Double-click the new link in the model.

The reference property sheet appears.

4 Type a name and a code for the Reference link.
or
Type a name and click the button at the end of the Code box.
or
Type a code and click the  button at the end of the Name box.

> **Automatic identification of foreign key**
> If there is a column in the child table with the same name as the primary key in the parent table, the column in the child table becomes a foreign key automatically. In this case, you can skip steps 5-8.

5 Click the  button under the Primary key column.

The list of primary key columns in the parent table appears. This list does not include keys already joined by the reference.

6 Select the column that you want to designate as a primary key.

7 Click OK.

8 Click the button under the Foreign key column.

The list of columns in the child table appears. This list does not include keys already joined by the reference.

9 Select the column that you want to designate as a foreign key.

10 Click OK in each of the dialog boxes.

Auto-migrating foreign keys

When you draw a reference between two tables, you can automatically migrate primary key columns in the parent table to foreign key columns in the child table.

The following table shows results of different actions when you choose to auto-migrate foreign keys:

Action	Result
Modify reference attach point	Migrate primary key in parent table to foreign key in child table Delete unused foreign key columns Modify reference join
Delete primary key	Delete corresponding foreign key and reference join
Modify primary key	Modify corresponding foreign key

❖ **To auto-migrate foreign keys:**

1 Select File➢Model Options.

2 Click the General tab.

3 Select the Auto-migrate FK checkbox and click OK.

Designating an alternate key

An alternate key is made up of one or more columns such that each of its values corresponds to one and only one row in the table. Each alternate key can generate a unique index or a unique constraint in a target database.

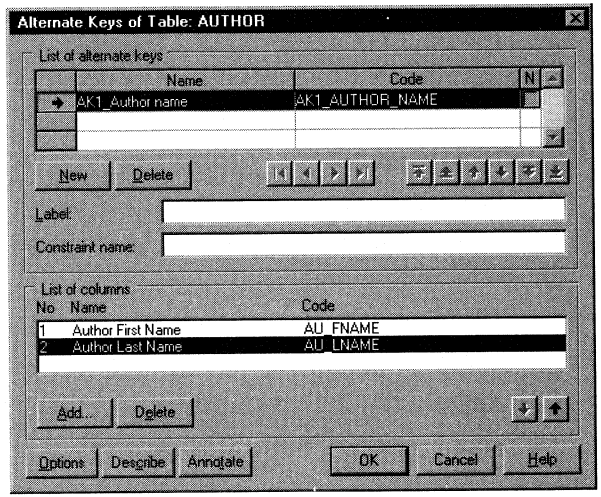
An alternate key can also be a foreign key. An alternate key cannot be a primary key.

❖ To designate an alternate key:

1 Double-click a table in the model with the Pointer tool.

2 Click the Alt. Keys button.

A dialog box lists alternate keys defined for the current table.



3 Type a name and a code for the alternate key.
or
Type a name and click the button in the Code column.
or
Type a code and click the button in the Name column.

> **Alternate key naming convention**
> The naming convention for an alternate key is AK followed by the number of the key column code; for example AK1_CUSNAME.

4 Click the Add button.

The list of columns that are not primary key columns appears.

5 Select one or more columns that you want to designate as an alternate key.

6 Click OK.

7 (Optional) Type a constraint name.

If you do not specify a constraint name, PowerDesigner creates a default constraint name automatically.

8 Click OK in each of the dialog boxes.

Naming key constraints

Naming key constraints helps you to identify and customize key constraints in scripts for database creation and modification. The constraint name gives you greater flexibility for modifying keys in subsequent database generations.

If you do not specify a constraint name, PowerDesigner creates a default constraint name automatically.

Naming a primary key constraint

A primary key constraint is a named check that enforces the uniqueness and the presence of values in a primary key column.

In the name of a primary key constraint, you can use the following variable:

Variable	Description
%TABLE%	Code of the table

❖ To name a primary key constraint:

1 Double-click a table in the model.

The table property sheet appears.

2 Type changes to the name in the Primary key Constraint Name box.

The User-Defined checkbox is selected automatically. You can always undo changes to the name by clearing this checkbox.

3 Click OK.

Naming a foreign key constraint

In the name of a foreign key constraint, you can use the following variables:

Variable	Description
%REFR%	Code of the reference
%NO%	Number of the reference, for use when more than one reference links the same parent table and child table
%PARENT%	Code of the parent table
%CHILD%	Code of the child table

❖ **To name a foreign key constraint:**

1 Double-click a reference in the model.

The reference property sheet appears.

2 Click the Integrity button.

The Referential Integrity dialog box appears.

3 Type changes to the name in the Constraint Name box.

The User-Defined checkbox is selected automatically. You can always undo changes to the name by clearing this checkbox.

4 Click OK in each of the dialog boxes.

Naming an alternate key constraint

In the name of an alternate key constraint, you can use the following variables:

Variable	Description
%AKEY%	Code of the alternate key
%TABLE%	Code of the table
%NO%	Number of the alternate key, when one table has more than one alternate key

❖ **To name an alternate key constraint:**

1 Double-click a table in the model.

The table property sheet appears.

2 Click the Alt Keys button.

The Alternate Keys dialog box appears.

3 Select an alternate key in the list.

4 Type changes to the name in the Constraint Name box.

The User-Defined checkbox is selected automatically. You can always undo changes to the name by clearing this checkbox.

5 Click OK in each of the dialog boxes.

Defining indexes

An index is a data structure associated with a table that is logically ordered by the values of a key. It improves database performance and access speed.

Index properties

Each index definition includes the following properties:

Property	Description	Maximum length
Name	Name of the index	80
Code	Reference name of the index	80
Label	Descriptive label for the index	254
Index type	Indicate whether an index is primary, foreign, unique or clustered	—
Table	Indicate table to index	—
Column	Indicate column to index	—
Sort order	Indicate to sort column in ascending or descending order	—
Type	(for RedBrick, Unify, and Oracle only) Proprietary index type	—

The following index types exist:

Key type	Description
Primary key	Uniquely identifies a row in the table
Foreign key	Depends on and migrates from a primary key in another table
Unique	Index in which no two rows can have the same index value. All primary key indexes must be unique
Cluster	Index in which the physical order and the logical (indexed) order is the same

One cluster index per table
A table cannot have more than one cluster index.

Creating an index

❖ To create an index:

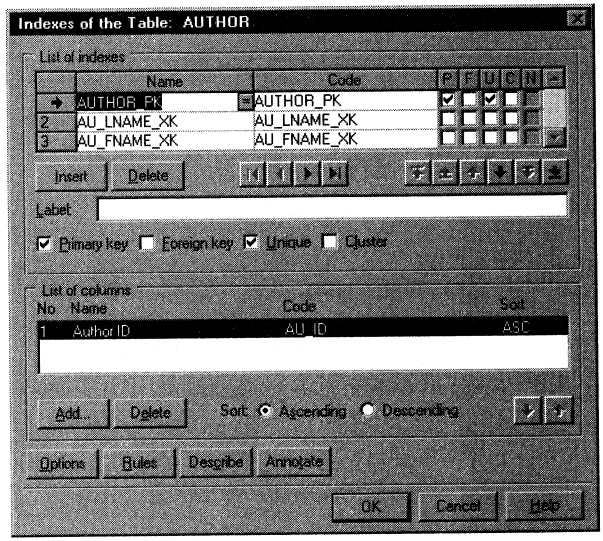
1 Double-click the table in the model with the Pointer tool.

or

Click the Property tool and click a table in the model.

2 Click the Indexes button.

A dialog box lists indexes associated with the table.



3 Click a blank line in the list.

or

Click the Insert button.

An arrow appears at the beginning of the line.

4 Type a name and a code for the index.

or

Type a name and click the  button in the Code column.

or

Type a code and click the  button in the Name column.

> **Index naming conventions**
> The naming convention for a primary key index is PK followed by the table code; for example PK_EMPLOYEE. The naming convention for a foreign key index is FK followed by the number of key and the table code; for example FK2_PROJECT.

5 Select or clear the following checkboxes: Primary Key, Foreign Key, Unique, and Cluster.

If you select the Primary Key checkbox, the Unique checkbox is selected automatically.

6 Click the Add button.

A dialog box lists the columns in the current table.

7 Select one or more columns that you want to index.

If you want to create a primary key index, select the primary key columns. If you want to create a foreign key index, select the foreign key columns.

8 Click OK.

You return to the list of indexes. The selected column appears in the list of columns.

9 Select Ascending or Descending radio buttons to indicate the sort order for each column.

10 Click OK in each of the dialog boxes.

Removing a column from an index

❖ **To remove a column from an index:**

1 Double-click the table in the model with the Pointer tool.
or
Click the Property tool and click a table in the model.

2 Click the Indexes button.

A dialog box lists indexes associated with the table.

3 Click an index in the list.

An arrow appears at the beginning of the line.

4 Select a column in the list of columns.

5 Click the Delete button.

6 Click OK.

Rebuilding indexes

The procedure for rebuilding indexes does the following:

- Clears existing primary key indexes and foreign key indexes
- Generates primary key indexes
- Generates all foreign key indexes regardless of threshold values and estimated number of occurrences

❖ **To rebuild indexes:**

1 Select Dictionary➢Rebuild Indexes.

A confirmation box asks you to confirm your choice.

2 Click Yes.

Deleting an index

There are two ways to delete an index:

- Delete an index from the list of indexes
- Delete an index from a table

Deleting an index from the list of indexes

❖ **To delete an index from the list of indexes:**

1 Select Dictionary➢List of Indexes.

The list of indexes appears.

2 Click the index to delete from in the list.

An arrow appears at the beginning of the line.

3 Click the Delete button.

4 Click OK.

Deleting an index from a table

❖ To delete an index from a table:

1 Double-click the table in the model with the Pointer tool.

or

Click the Property tool and click a table in the model.

2 Click the Indexes button.

A dialog box lists indexes associated with the table.

3 Click the index that you want to delete from the list.

An arrow appears at the beginning of the line.

4 Click the Delete button.

5 Click OK.

Defining views

A view is an alternative way of looking at the data in one or more tables. It is made up of a subset of columns from one or more tables.

You define a SQL query for each view.

Creating a view

You have two ways to create a view:

- Create a view for tables selected in the PDM
- Create an empty view then select tables from a list of tables

❖ To create a view for selected tables:

1 Select one or more table symbols in the PDM.

2 Select Dictionary➤Views➤New.

A view symbol appears in the model. At creation, a view is named View_*n*, where *n* is a number assigned in the order of the creation of objects.

3 Double-click the new view symbol in the model.

The view property sheet appears.

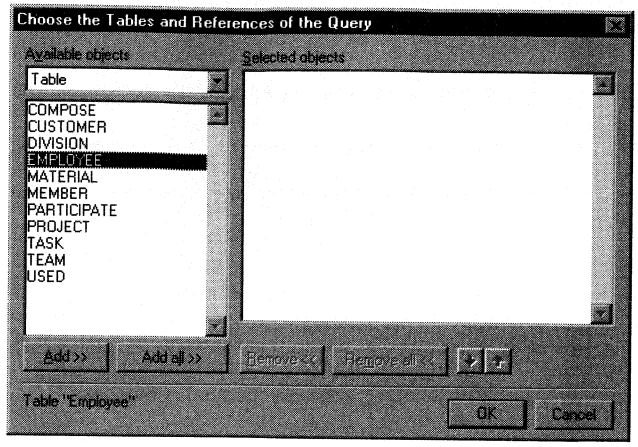
4 Type a name and a code for the view.
or
Type a name and click the  button at the end of the Code box.
or
Type a code and click the  button at the end of the Name box.

5 Click OK.

❖ To create an empty view then select tables:

1 Select Dictionary➤Views➤New.

A dialog box displays the list of available tables.



2 Select tables for the view and click Add.

If you select tables that share a reference link, that reference is automatically selected.

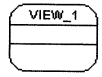
3 Select References from the dropdown listbox.

The list of references appears.

4 Select references for the view and click Add.

5 Click OK.

A view symbol appears in the model.



At creation, a view is named View_*n*, where *n* is a number assigned in the order of the creation of objects.

6 Double-click the new view symbol in the model.

The view property sheet appears.

7 Type a name and a code for the view.
or
Type a name and click the button at the end of the Code box.
or
Type a code and click the button at the end of the Name box.

8 Click OK.

View properties

The view definition includes the following properties:

Property	Description	Maximum length	Unique in model?
Model	Name for the model	—	—
Name	Name for the view. This name improves the readability of the model	80	✓
Code	Reference name for the view. This code is generated in database scripts	80	✓
Label	Descriptive label for the view	254	—
Usage	Group of properties that affect database script generation	—	—

In addition, you can select any of the following usage properties:

Property	Description
Query only	Defines view for consultation only. View cannot update tables
Updatable	Defines view for consultation and update. View can update tables
Generate View	Includes view generation as part of database generation script
With check option	Implements controls on view insertions
User-defined SQL	Implements a user-defined SQL query instead of a standard query

Modifying view properties

There are two approaches to modifying view properties:

- Modify a view property sheet
- Modify an entry in the list of views

Accessing a view property sheet

The View property sheet displays view definitions that you can modify.

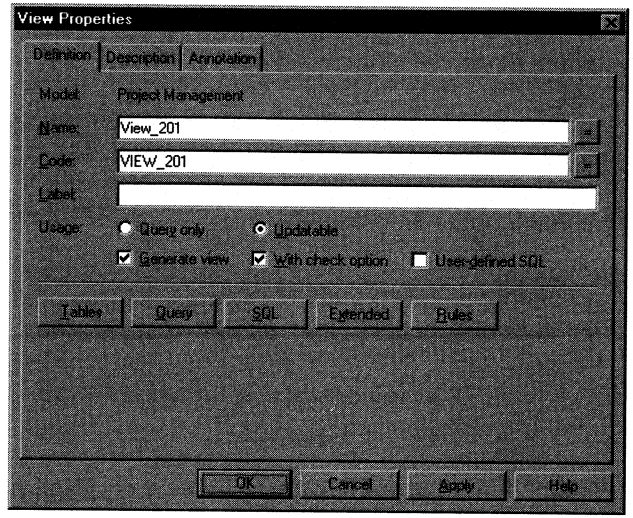
❖ To modify view properties from property sheet:

1 Double-click the view in the model with the Pointer tool.

or

Click the Property tool and click a view in the model.

The view property sheet appears.



2 Modify the name, the code, the label, and the usage in the corresponding boxes.

3 Click OK, the Description tab, or the Annotation tab.

Accessing the list of views

The list of views includes all views attached to the current model. You can modify certain view properties from the list.

❖ To modify view properties from the list:

1 Select Dictionary➤Views➤List of Views.

The list of views appears.

2 Click the view to modify.

An arrow appears at the beginning of the line.

3 Modify the name or the code directly in the list.

4 Modify the label and the usage in the boxes that appear below the list.

5 Select the checkbox in the D column to display the view symbol.
or
Clear the checkbox in the D column to remove the view symbol.

6 Click OK.
or
Click a different view in the list.

Selecting tables and references for a view

Based on your selection of tables, columns, and references, PowerDesigner generates a query in SQL syntax.

The resulting query uses FROM statements for tables and WHERE statements for reference column joins.

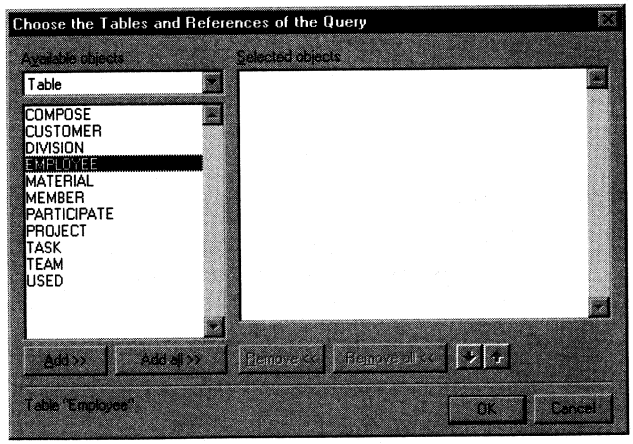
❖ To select tables and references for a view:

1 Double-click a view in the model.

The view property sheet appears.

2 Click the Tables button.

A dialog box displays the list of available tables.



3 Select tables from the list of available objects and click Add to add them to the view.
or
Select tables from the list of selected objects and click Remove to remove them from the view.

If you select tables that share a reference link, that reference is automatically selected. If you remove tables that share a reference link, that reference is automatically removed.

4 Click OK.

5 Select References from the dropdown listbox.

The list of references appears.

6 Select references from the list of available objects and click Add to add them to the view.
or
Select references from the list of selected objects and click Remove to remove them from the view.

7 Click OK in each of the dialog boxes.

Selecting columns for a view

Selecting columns for a view makes them available for use in the query definition.

The resulting query includes columns in SELECT statements.

❖ To define columns for use in a SQL query:

1 Double-click a view in the model.

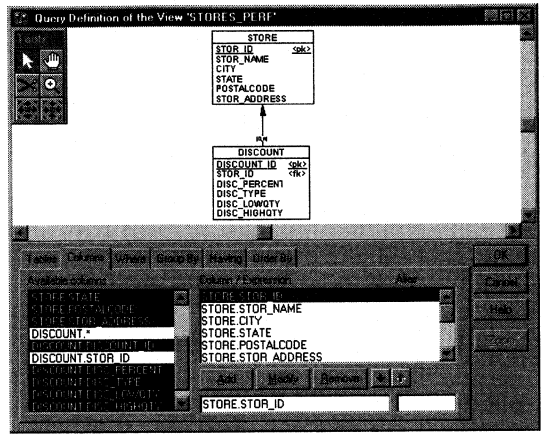
The view property sheet appears.

2 Click the Query button.

The query window opens.

3 Click the Columns tab.

The Columns page opens.



4 Select columns from the list of available columns and click Add to add them to the view.
or
Select columns from the list of selected columns and click Remove to remove them from the view.

5 If you want to modify the name or the alias of a column, select a column from the list of selected columns and click Modify.

6 Type the name and the alias in the boxes.

7 Click OK in each of the dialog boxes.

Assigning an alias to a table or a column

You can assign an alias to a table or a column. You use this alias in the query script instead of typing out the full code.

❖ **To assign an alias to a table or a column:**

1 Double-click a view in the model.

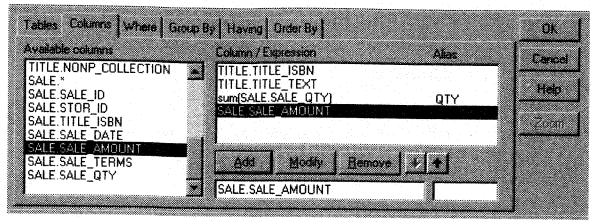
The view property sheet appears.

2 Click the Query button.

The query window opens.

3 Click the Tables tab.
or
Click the Columns tab.

The Tables page or the Columns page opens.



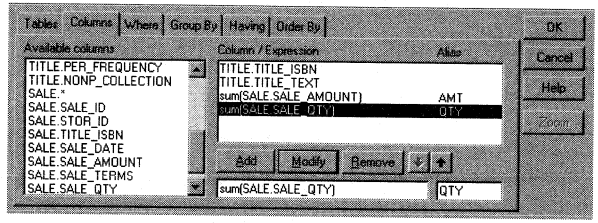
4 Select a table or a column from the list of selected tables or columns.

The code appears in a box below the Add button.

5 Type an alias in the box that follows the Code box.

6 Click the Modify button.

The alias appears in the list of selected tables or columns.



7 Click OK in each of the dialog boxes.

Inserting clauses in a query

You can add the following types of clauses to a query:

SQL clause	Description
WHERE	Specify search conditions
GROUP BY	Divide output of a table into groups
HAVING	Specify grouping conditions
ORDER BY	Sort query results by one or more columns

❖ To insert clauses in a query:

1 Double-click a view in the model.

The view property sheet appears.

2 Click the Query button.

The query window opens.

3 Click the Where, Group By, Having, or Order By tab.

4 Select columns and operators.

5 Click OK.

Displaying query syntax

You can display the query that PowerDesigner will generate based on the view definition.

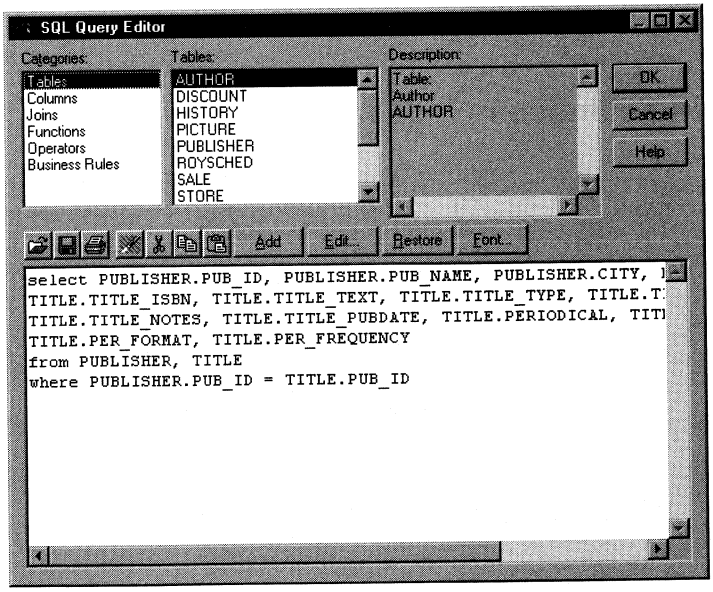
❖ To display query syntax:

1 Double-click a view in the model.

The view property sheet appears.

2 Click the SQL button.

The SQL Query Editor displays the syntax of the query that PowerDesigner will generate based on your view definition.



3 Click OK in each of the dialog boxes.

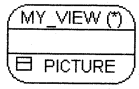
Editing a query

You can edit the SQL query for a view. The edited query is not automatically updated when you modify objects in the model. If you include objects in the view that are not in the model, the objects do not appear in the view symbol.

You choose whether or not to generate the query you edit.

Query type	Generation
Not user-defined	Generates query based on view definition, does not generate the edited query (for documentary purposes only)
User-defined	Generates edited query

When a view has a user-defined query, an asterisk appears in the view symbol next to the name or code.



❖ **To edit a query:**

1 Double-click a view in the model.

The view property sheet appears.

2 (Optional) Select the User-Defined checkbox if you want to generate the query you edit instead of the default query based on the view definition.

The Query button is grayed.

3 Click the SQL button.

The SQL editor window opens.

4 Edit the SQL query text.

5 Click OK in each of the dialog boxes.

Verifying SQL query syntax

You check the validity of the SQL queries using an ODBC driver which test the query with the target database.

❖ **To verify SQL query syntax:**

1 Double-click a view in the model.

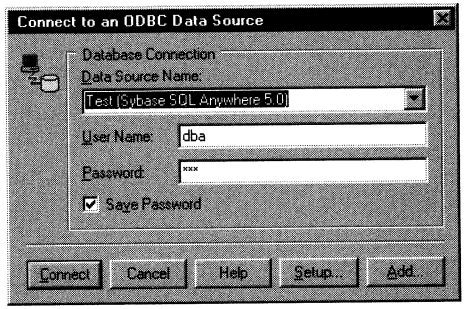
The view property sheet appears.

2 Click the Query button.

The query window opens.

3 Click the verify query tool.

The Connect To An ODBC dialog box appears.



4 Select a data source from the dropdown listbox.
Type a user name and password.
Click Connect.

If the query is correct You see the message The SQL syntax of the query is correct.

If there are errors in the query The target database sends back error messages via the ODBC driver. These messages are database-dependent.

Displaying text in view symbols

You can display the following text in a view symbol:

Text display preference	When selected, what it displays
Tables	Tables in the view
Columns	Columns in the view
Data types	Data type for each column
Formulas	Asterisk (*) for multiple column selection

❖ **To display text in a view symbol:**

1 Select File➤Display Preferences.

The Display Preferences dialog box opens to the General page.

2 Select or clear checkboxes in the View groupbox.

3 Click OK.

Defining extended attributes

Extended attributes complete the definition of objects in the PDM. You can add extended attributes for documentation purposes or for eventual use in external applications.

For example, you can use extended attributes to add labels in different languages to PDM objects, or you can use extended attributes to describe how to display information in an external application.

You can create default extended attributes for all objects of the same type, for example for all tables. All objects of the given type inherit extended attributes. You can then modify the extended attributes for a specific object.

Extended attribute properties

You can define one or more extended attributes for each type of object in a PDM. Extended attribute definitions include the following properties:

Property	Description	Maximum length
Name	Name of the extended attribute	80
Type	Data type	—
Label	Descriptive label	254
Modified	Indicates the existence of a modified default value for a particular object	—
Value	Default value	254

Standard types of extended attributes

PowerDesigner includes the following standard types of extended attributes.

Type	Format	Example
Boolean	Two opposing values	TRUE
Color	RGB-coded color	255 0 0
Date	Day, month, year	03/31/96
File	Filename	C:\PWRS\PD6\EXA\VB.EXA
Float	Floating decimal number	18.6
Font	FontName,FontSize, FontStyle	Arial,12,N
FontName	Font name	Arial
FontSize	Font size	10
FontStyle	Font style: N for normal B for bold I for italic U for underline S for strikethrough	BI
Hex	Hexadecimal number	0xffff
Integer	Integer	10
Path	Directory name	C:\PWRS\PD6
String	Character string	"Error: Access denied"
Time	Hour, minute, second	10:00:00

Identification of standard and nonstandard types
The names of standard types appear in parentheses. You can create a type by typing its name without parentheses directly in the Type box.

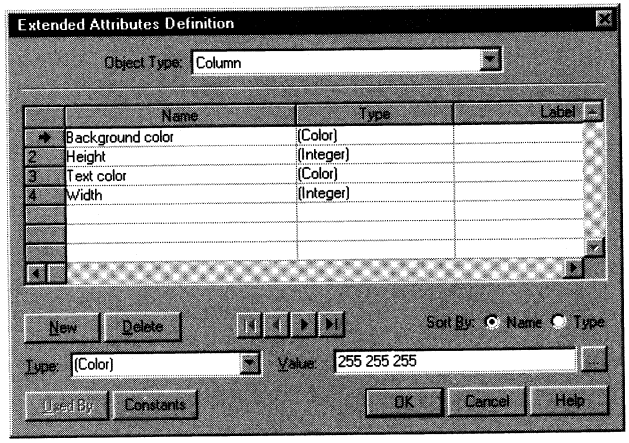
Creating extended attributes

You must provide a name and data type for each extended attribute that you create. Labels and default values are optional.

❖ **To create extended attributes:**

1 Select Dictionary➤Extended Attributes➤List of Attributes.

The list of extended attributes appears.



2 Click a blank line in the list.

An arrow appears at the beginning of the line.

3 Type a name in the Name column.

4 Select a type from the Type dropdown listbox.

5 Type a default value in the Value box.
or
Click the ... button to display a list of available default values.
Select a value.
Click OK.

6 Click OK.

Defining a constant

You can define a constant and use the constant as the value of an extended attribute.

Constants have the following properties:

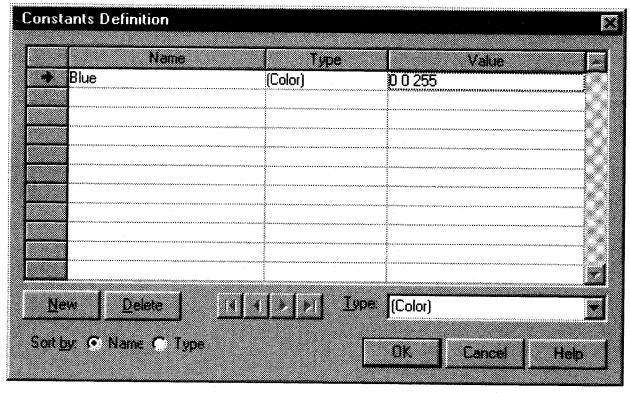
Property	Description
Name	Name of the constant
Type	Standard type of extended attribute
Value	Value of the constant

For example, you can define a constant with the name Blue. The value of this constant is the RGB encoding for blue. You can then select this constant as a value for any extended attribute with the type Color.

❖ To define a constant:

1 Select Dictionary➤Extended Attributes➤List of Constants.

The list of constants appears.



2 Click a blank line in the list.

An arrow appears at the beginning of the line.

3 Type a name in the Name column.

4 Select a type from the Type dropdown listbox.

5 Type the value for the constant in the Value column.
or
Click the Value column, click the ... button that appears in the column, select a value, and click OK.

6 Click OK.

Using a constant as a default value

You can use a constant as the default value for an extended attribute.

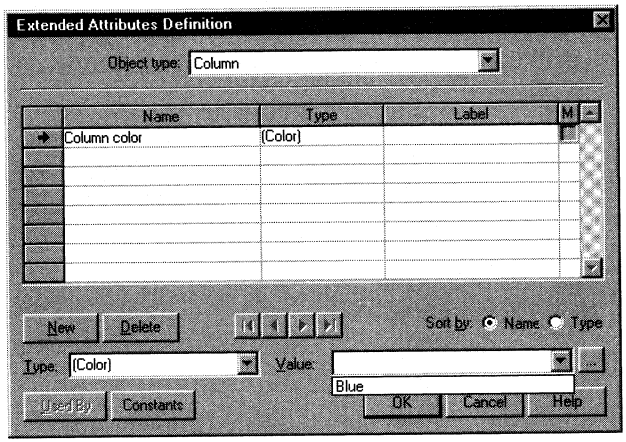
❖ To use a constant as the default value of an extended attribute:

1 Select Dictionary➤Extended Attributes➤List of Attributes.

The list of extended attributes appears.

2 Click an attribute in the list.

An arrow appears at the beginning of the line. If a constant exists for the type of extended attribute, the Value box becomes a dropdown listbox.



3 Select a constant from the Value dropdown listbox

4 Click OK.

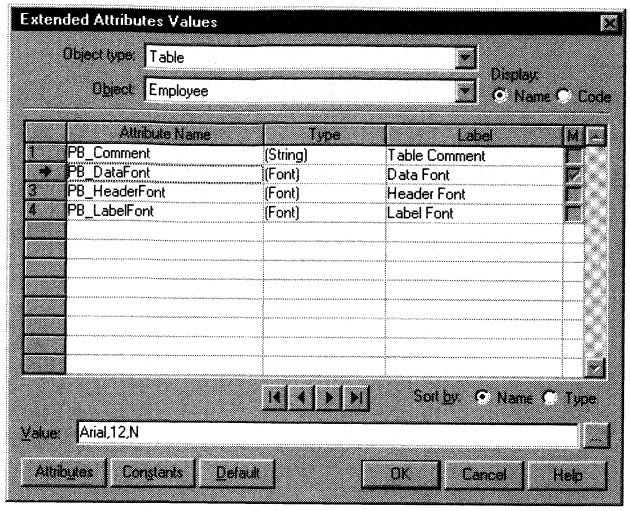
Modifying the value of an extended attribute

Default values for extended attributes apply to all objects of the same type, for example all tables or all columns. However, you can modify the value of an extended attribute for a particular object.

❖ To modify the value of an extended attribute for an object:

1 Select Dictionary➤Extended Attributes➤List of Attribute Values.

The list of attribute values appears.



2 Select an object type from the Object Type dropdown listbox.
Select an object name from the Object dropdown listbox.

The list of attribute values displays values for the selected object.

3 Click an attribute in the list.

An arrow appears at the beginning of the line.

4 Type a new value in the Value box.
or
Click the button to display a list of available values for the attribute type.
Select a value.
Click OK.

5 Click OK.

Exporting extended attributes

You can export extended attributes to an EXA file. In another PDM, you can then import the extended attributes defined in this EXA file.

❖ To export extended attributes:

1 Open the PDM from which you want to reuse the extended attributes.

2 Select Dictionary➤Extended Attributes➤Export Attributes.

A Save As dialog box appears.

3 Type an EXA filename.

4 Click OK.

Importing extended attributes

You can import extended attributes defined in an EXA file.

❖ **To import extended attributes:**

1 Select Dictionary➤Extended Attributes➤Import Attributes.

An Open dialog box appears.

2 Select an extended attributes file (EXA file).

3 Click OK.

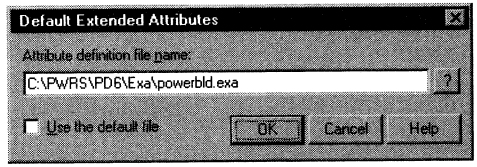
Defining default extended attributes

Default extended attributes apply to every new PDM. You define default extended attributes by selecting an EXA file.

❖ **To define default extended attributes:**

1 Select Dictionary➤Extended Attributes➤Default Attributes.

The Default Extended Attributes dialog box appears.



2 Click the ? button.

A list of EXA files appears.

3 Select an EXA file.

4 (Optional) Select the Use The Default File checkbox if you want to import this EXA into the current model.

5 Click OK.

Defining check parameters

Check parameters indicate data ranges and validation rules. You can attach check parameters to domains and columns.

There are two types of check parameters:

Parameter type	Description
Standard parameters	Common data controls (minimum, maximum, and accepted values)
Validation rules	Customized rules for data validation

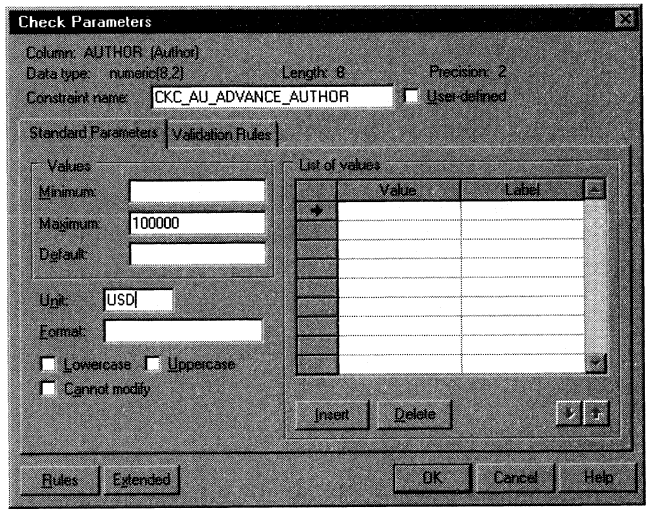
Standard parameter properties

Standard parameters indicate common data controls for a domain or column.

The following table lists standard parameters:

Parameter	Description
Minimum	Lowest acceptable numeric value
Maximum	Highest acceptable numeric value
Default	Value assigned in absence of an expressly entered value
Unit	Standard measure
Format	Data format (for example, 9999.99)
Lowercase	Forces all alphabetical characters to lowercase
Uppercase	Forces all alphabetical characters to uppercase
Cannot Modify	Protects from changes, results in a nonmodifiable column in the physical data table
List of Values	Authorized values
Label	String that identifies an authorized value in the list

The following check parameters do not authorize advances over $100,000.



Setting standard parameters

❖ To set standard parameters:

1. Select Dictionary➢List of Domains.
   or
   Select Dictionary➢List of Columns.

   The list of domains or columns appears.
2. Select a domain or column.

   An arrow appears at the beginning of the line.
3. Click the Check button.

   The Check Parameters property sheet appears.
4. Click the Standard Parameters tab.
5. Type your choice of Standard Parameters.
6. Click OK.

Managing quotation marks around values

For a domain or a column, standard parameters can indicate minimum, maximum, and default values as well as a lists of values.

In general, if the data type of domain or column is a string data type, quotation marks surround its values in the generated script. The generation of single or double quotation marks depends on the target DBMS.

However, quotation marks are not generated in the following cases:

- You define a data type that is not recognized as a string data type by PowerDesigner
- Value is surrounded by tilde characters
- Value is a keyword defined in the DEF file (for example, NULL)

In addition, if the value is already surrounded by quotation marks additional quotation marks are not generated.

The following table shows the way a value for a string data type is generated in a script:

Value for string data type	Result in script
Active	'Active'
'Active'	'Active'
"Active"	'"Active"'
~Active~	Active
NULL	NULL

Using a validation rule in check parameters

Expressions in validation rules can generate check parameters when they are attached to domains or columns.

At generation, validation rule expressions can instantiate the following variables:

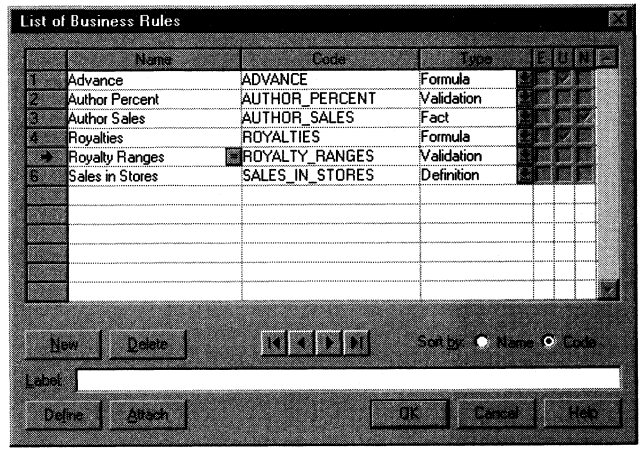
Variable	Description
%column%	Code of the column to which the business rule applies
%domain%	Code of the domain to which the business rule applies

❖ To use a validation rule in check parameters:

1 Select Dictionary➤List of Business Rules.

The list of business rules appears.

2 Select a validation rule in the list.



3 Click Apply.

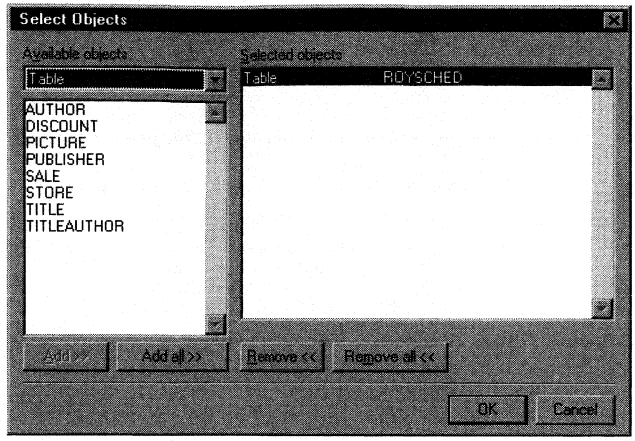
The Business Rule Definition dialog box displays available and selected objects.

4 Select Domain from the Available Objects dropdown listbox.
or
Select Column from the Available Objects dropdown listbox.

5 Select a domain or column in the list.

6 Click Add.

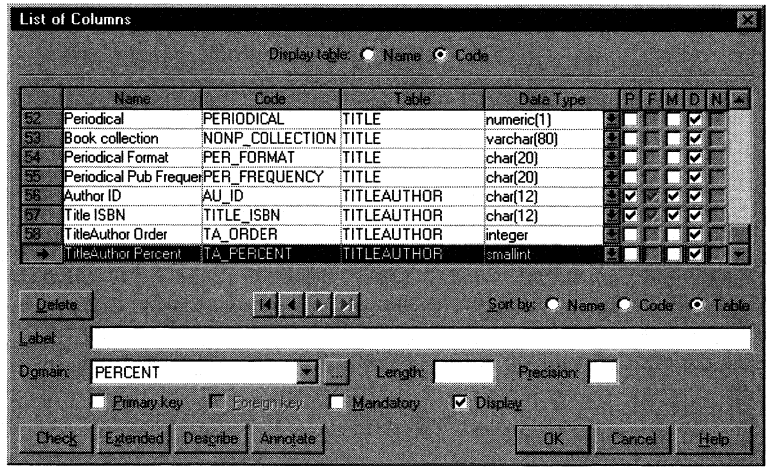
The domain or column appears in the list of Selected Objects.



7 Click OK.

8 Select Dictionary➤List of Domains.
or
Select Dictionary➤List of Columns.

The list appears.



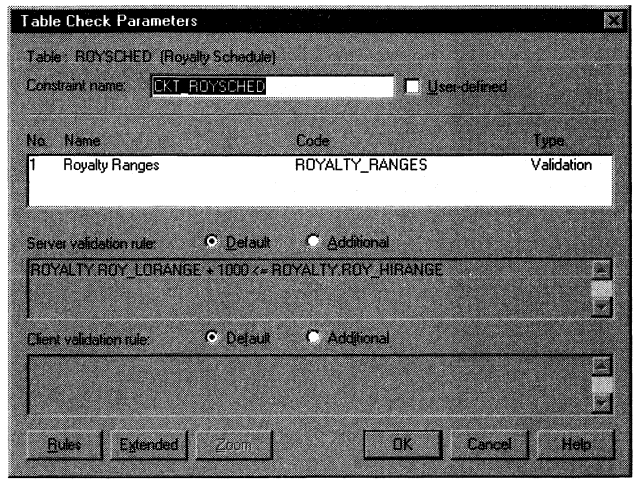
9 Select the domain or the column to which you applied the business rule.

10 Click the Check button.

The Check Parameters dialog box appears.

11 Click the Validation Rules tab.

The validation rule and its expressions appear on the Validation Rules page.



Validation rule expressions
You cannot modify the expression attached to a validation rule from the Check Parameters dialog box. You define validation rule expressions from the list of business rules, by clicking the Define button.

12 Click OK in each of the dialog boxes.

CHAPTER 8

Managing Physical Data Models

About this chapter

This chapter describes checking the validity of a Physical Data Model (PDM) and merging two PDM.

Contents

Checking a PDM

You can check the validity of a PDM at any time.

The following general rules apply to PDM:

- Each table must have at least one column
- Each table must have a primary key
- Each index must have at least one column
- Each reference must have at least one column pair

The procedure that generates a database starts by checking the validity of the PDM. If an error is found, the database (or database generation script) is not generated.

PDM check options

When you check a PDM, two types of messages can result:

Message	Description
Error	Major problem that impedes database generation
Warning	Minor problem or recommendation
Database constraint	Problem linked to incompatibility with target database

You choose what types of messages to display for the following PDM objects:

- Table
- Index
- Column
- Reference
- All objects

Old model specific checks
S-Designor detected certain errors which you cannot recreate with PowerDesigner. If you are using a model created with S-Designor, you can detect these errors by selecting the Old Model Specific Checks checkbox.

You must choose one of the following correction options:

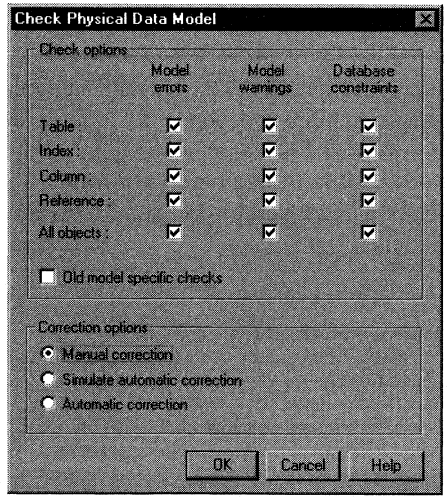
Option	Description
Manual correction	Displays error and warning messages
Simulate automatic correction	Displays error and warning messages Displays description of automatic correction
Automatic correction	Displays error and warning messages Corrects certain errors automatically

Checking a global PDM

❖ **To check a global PDM:**

1 Select Dictionary➤Check Model.

The Check Physical Data Model dialog box appears.

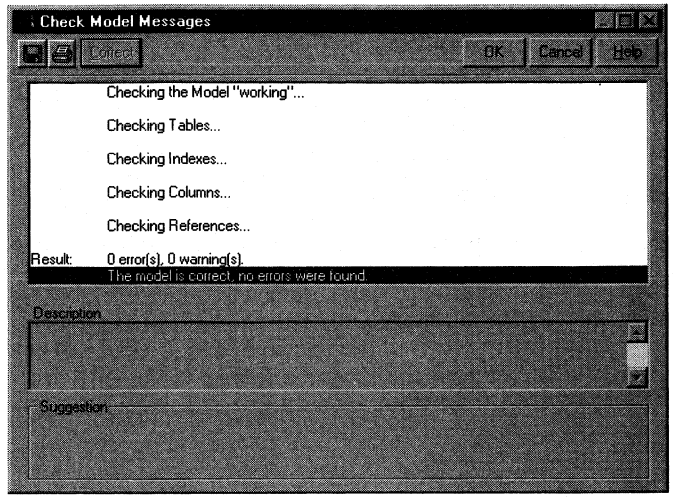


2 Select check options.

3 Click OK.

The Check Model Messages dialog box displays errors and warnings, based on your choice of check options.

If there are no errors in the model, the last message indicates that the model is correct. If any errors are found, the last message indicates that the model is incorrect.



4 Click OK.

Checking a PDM submodel

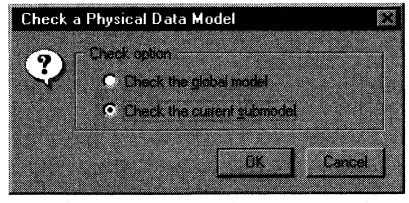
You can restrict checks to the current submodel. This allows you to verify your own submodels independently of those belonging to other designers.

> **No submodels in desktop versions**
> If you are using a desktop version of this product, you cannot define submodels.

❖ To check a submodel of a PDM:

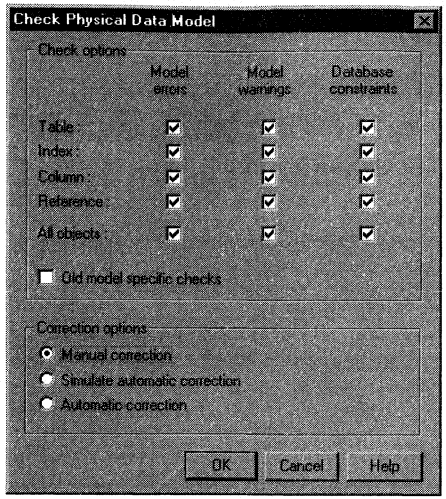
1 Select Dictionary➤Check Model.

A confirmation box appears.



2 Click Check the Current Submodel to check the validity of the submodel.

The Check Physical Data Model dialog box appears.

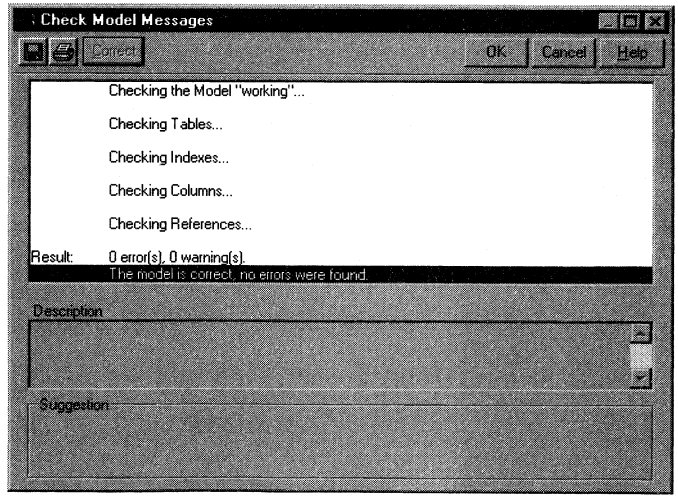


3 Select check options.

4 Click OK.

The Check Model Messages dialog box displays errors and warnings, based on your choice of check options.

If there are no errors in the model, the last message indicates that the model is correct. If any errors are found, the last message indicates that the model is incorrect.



5 Click OK.

Making corrections based on PDM check results

You can use the check procedure to locate problems in the PDM. The check results include suggestions for correcting problems.

You must choose one of the following correction options:

Option	Description
Manual correction	Displays error and warning messages
Simulate automatic correction	Displays error and warning messages Displays description of automatic correction
Automatic correction	Displays error and warning messages Corrects certain errors automatically

Automatic correction fixes the following types of errors and warnings:

- Inappropriate foreign key columns
- Non-unique names
- Code too long for target database (if you select Database Constraints)
- Inappropriate primary keys, foreign keys, and unique indexes

You must always correct other errors manually.

Making manual corrections to a PDM

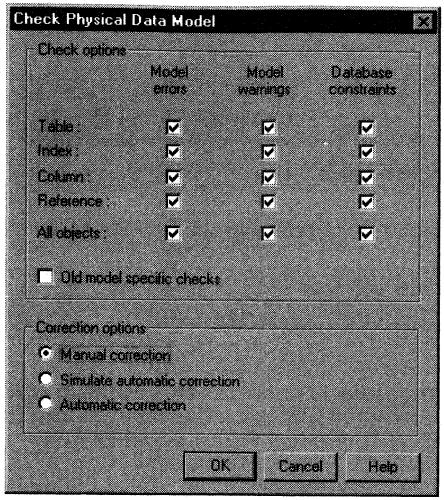
❖ **To make manual corrections to a PDM:**

1 Select Dictionary➢Check Model.

The Check Physical Data Model dialog box appears.

2 Select check options.

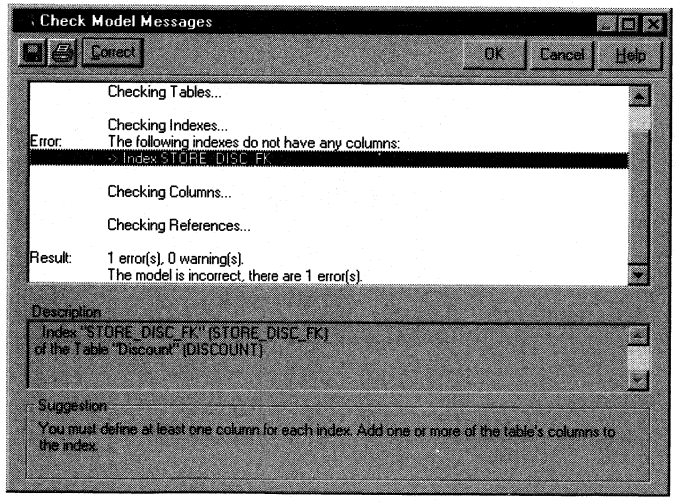
3 Select the Manual Correction radio button.



4 Click OK.

The Check Model Messages dialog box displays errors and warnings, based on your choice of check options.

5 Select a line starting with an arrow directly below an error or a warning.



The Description textbox indicates the object that is the source of the error or warning. The Suggestion textbox explains the reason for the error or warning and suggests a way to correct it.

6 Click the Correct button.

A dialog box in which you can correct the problem appears.

7 Make the suggested correction.

8 Click OK.

You return to the Check Model Messages dialog box.

9 Repeat steps 4 to 7 for another error or warning.
or
Click OK to return to the PDM workspace.

Simulating automatic corrections to a PDM

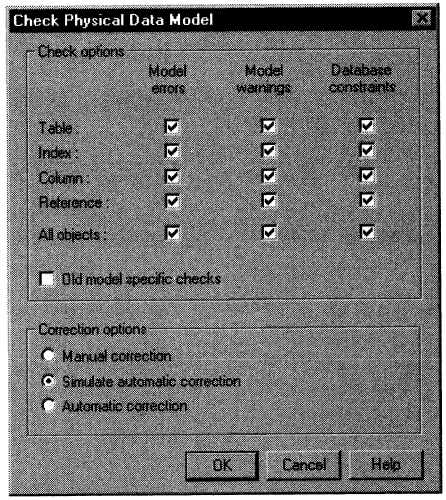
❖ To simulate automatic corrections to a PDM:

1 Select Dictionary➢Check Model.

The Check Physical Data Model dialog box appears.

2 Select check options.

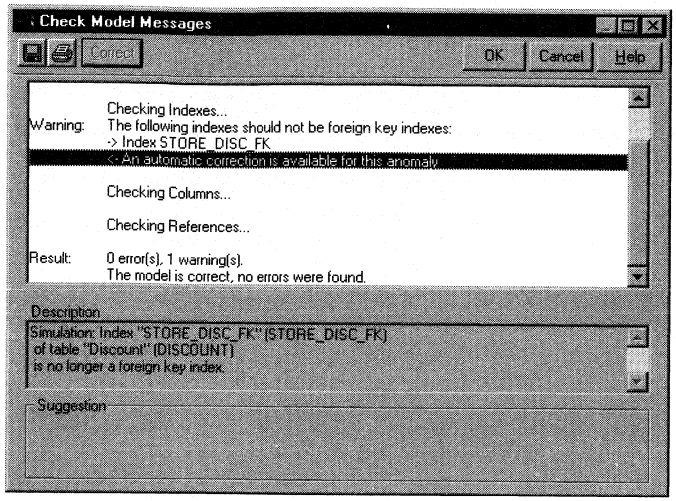
3 Select the Simulate Automatic Correction radio button.



4 Click OK.

The Check Model Messages dialog box displays errors and warnings, based on your choice of check options.

5 Select a line stating "An automatic correction is available for this anomaly."



The Description textbox describes automatic correction for this error or warning.

6 Click OK to return to the PDM workspace.

Making automatic corrections to the PDM

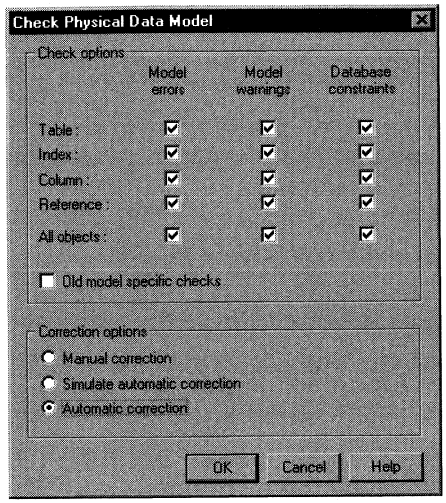
❖ To make automatic corrections to a PDM:

1 Select Dictionary➢Check Model.

The Check Physical Data Model dialog box appears.

2 Select check options.

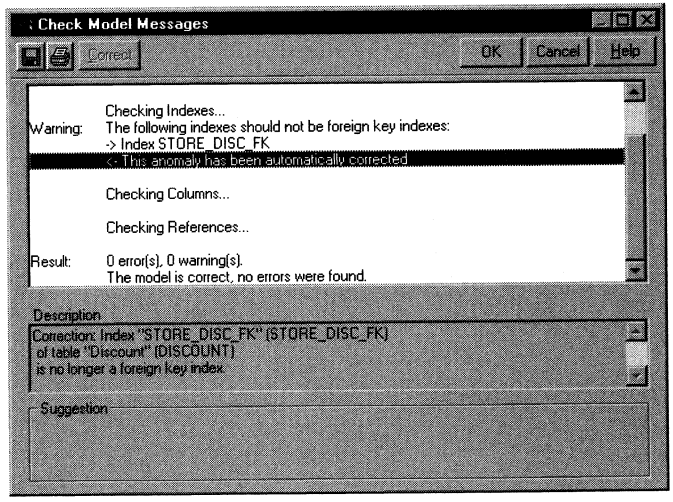
3 Select the Automatic Correction radio button.



4 Click OK.

The Check Model Messages dialog box displays errors and warnings, based on your choice of check options.

5 Select a line stating "This anomaly has been automatically corrected."



The Description textbox describes automatic correction for this error or warning.

6 Click OK to return to the PDM workspace.

Displaying errors and warnings

If you close the message window by clicking OK, all error and warning messages are stored in memory.

❖ **To display errors and warnings found by the last check:**

- Select Dictionary➤Display Messages.

Merging two PDM

You can merge two PDM. The merge makes it possible to form a single model that combines design efforts performed independently by several team members.

When the merge process finds two objects have the same code, you indicate whether or not the definition of the object in the second model should replace the definition in the current model.

❖ To merge two PDM:

1 Open a PDM.

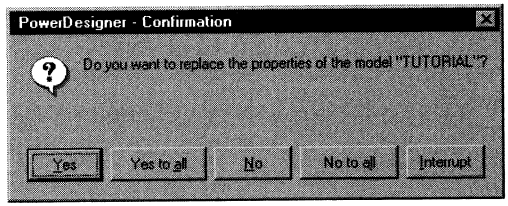
2 Select File➤Utilities➤Merge.

A file selection dialog box displays.

3 Select a PDM.

4 Click OK.

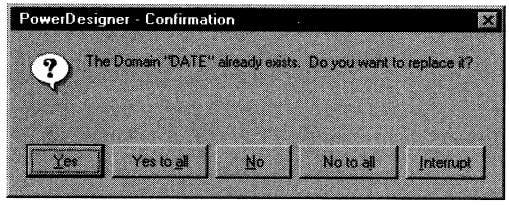
A confirmation message box appears:



5 Click one of the following buttons:

Button	Action
Yes	Replace model properties in current PDM with model properties from second PDM
Yes to All	Replace model properties in current PDM with model properties in second PDM *and* replace all objects in current PDM with objects from the second PDM, if objects have the same code
No	Do not replace model properties in current PDM with model properties from second PDM
No to All	Do not replace model properties in current PDM with model properties in second PDM *and* do not replace any objects in current PDM with objects from the second PDM, if objects have the same code
Interrupt	Discontinue merge

If you selected Yes or No, whenever the merge process encounters two objects with the same code, it displays a confirmation message box.



6 If this PowerDesigner Confirmation box appears, click one of the following buttons:

Button	Action
Yes	Replace object in current PDM with object from second PDM
Yes to All	Replace all objects in current PDM with objects from the second PDM, if objects have the same code
No	Do not replace object in current PDM with object from second PDM
No to All	Do not replace any objects in the current PDM with objects from the second PDM
Interrupt	Discontinue merge

Merging unique codes with a non-unique codes
If you selected the model option Unique Code for the current PDM and did not select this model option in the second PDM, the merge process assigns unique codes automatically. An information box displays a message telling you that a reference will be assigned a new code.

7 Select File➤Save As to save the merged model.

Importing an ERwin model into a PDM

You can import a model built with ERwin into a PDM.

The following objects are imported directly:

Business rule
Domain
Table
Column
Reference
Index
Key
Referential integrity constraint
Tablespace
PowerBuilder extended attribute
Target database
Description
Annotation
Text

The import process translates ERwin objects into PDM objects as follows:

Object in an ERwin model	Imported object in a PDM
Stored display and subject area	Submodel
Relationship and subtype relationship	Reference
Valid value	Check parameter
Segment	Storage

You cannot import ERwin triggers, ERwin reports, or ER1 files. If you are using ERwin/SQL, generate a database from your ER1 file and then reverse engineer that database into a PDM.

❖ **To import an ERwin model:**

1 Select File➤Import.

A standard file selection dialog box displays.

2 Select a file with the ERX extension.

3 Click OK.

A message box lists all imported objects.

4 Click OK.

CHAPTER 9

Reverse Engineering

About this chapter

This chapter describes AppModeler reverse engineering functions.

Contents

Generating a PDM from an existing database

You can generate a PDM from an existing database.

For each database version, an Open Database Connectivity (ODBC) driver and a DEF file serve as intermediaries between the database and PowerDesigner.

Reverse engineering options

Reverse engineering options are DBMS-dependent. Unavailable options appear grayed and you cannot select them.

You can select the following reverse engineering options:

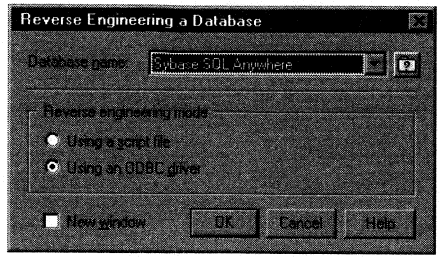
Option	Description
Qualifier	Database-dependent table qualifier (prefix to table name)
Owner	Retrieve tables belonging to a specific user
Reverse as table	Select database structures to reverse engineer as tables in the PDM: Table View System table Synonym
Table	For the tables that result from reverse engineering, select related objects to generate in the PDM: Primary keys Foreign keys Alternate keys Options Checks
Other objects	Select database structures (other than tables) to reverse engineer in the PDM: Indexes Procedures Views Triggers

Generating a PDM from a database

❖ To generate a PDM from a database:

1 Select File➤Reverse Engineering.

The Reverse Engineering dialog box appears.



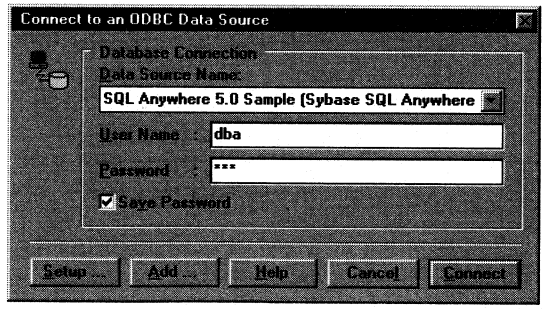
2 Select a database from the dropdown listbox.

3 Click the radio button labeled Using An ODBC Driver.

4 Select New Window if you want to create a new PDM.
or
Clear New Window if you want to merge new objects into the open PDM.

5 Click OK.

A dialog box asks you to identify a data source and connection parameters.

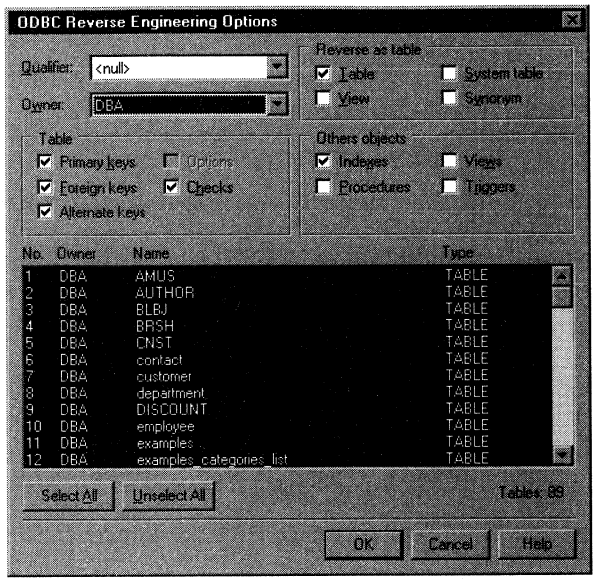


6 Select a data source from the dropdown listbox.

7 Type your user name and password.

8 Click Connect and, if prompted by your data source, enter additional connection parameters.

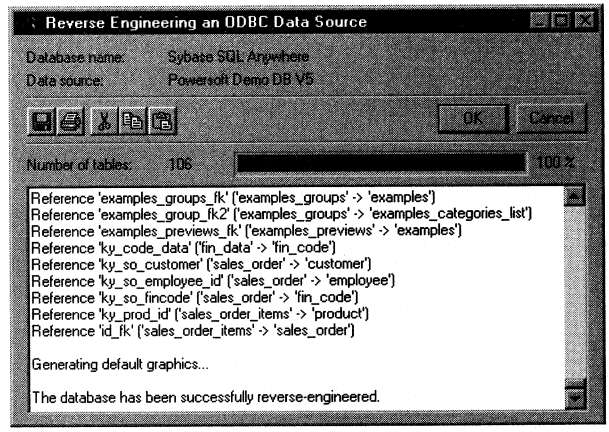
The Reverse Engineering Options dialog box appears.



9 Select reverse engineering options.

10 Click OK.

A message window reports progress of the generation.



11 Click OK.

The new PDM appears in the PDM window.

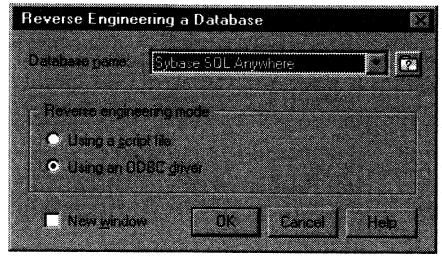
Adding a data source

You can add a data source for reverse engineering.

❖ To add a data source:

1 Select File➤Reverse Engineering.

The Reverse Engineering dialog box appears.



2 Select a database from the dropdown listbox.

3 Click the radio button labeled Using An ODBC Driver.

4 Click OK.

A dialog box asks you to identify a data source and connection parameters.

5 Click the Add button.

A list of data sources appears.

6 Click the Add button.

A list of ODBC drivers appears.

7 Double-click an ODBC driver.

You return to the Reverse Engineering dialog box.

8 Click Setup to configure the data source.
or
Click Connect to connect to the data source and to continue reverse engineering.
or
Click Cancel to return to the PDM.

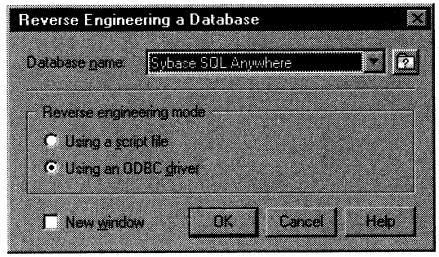
Configuring a data source

You can configure the data source for reverse engineering.

❖ To configure a data source:

1 Select File➤Reverse Engineering.

The Reverse Engineering dialog box appears.



2 Select a database from the dropdown listbox.

3 Click the radio button labeled Using An ODBC Driver.

4 Click OK.

A dialog box asks you to identify a data source and connection parameters.

5 Click the Setup button.

A setup window appears. The information requested in the window depends on the ODBC driver being used.

6 Enter data source parameters and click OK.

7 Click Connect to connect to the data source and to continue reverse engineering.
or
Click Cancel to return to the PDM.

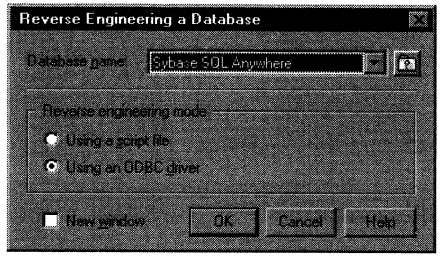
Generating a PDM from a database creation script

You can generate a PDM directly from a database creation script.

❖ To generate a PDM from a script:

1 Select File➤Reverse Engineering.

The Reverse Engineering dialog box appears.



2 Select a database from the dropdown listbox.

3 Click the radio button labeled Using a Script File.

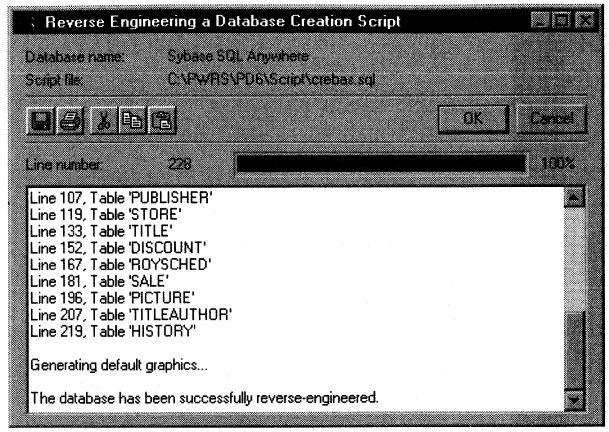
4 Select New Window if you want to create a new PDM.
or
Clear New Window if you want to merge new objects into the open PDM.

5 Click OK.

A standard file open dialog box appears.

6 Select a script file and click OK.

A message box shows generation progress.



7 Click OK.

The new PDM appears in the PDM window.

Multiple script files
If you have several script files, combine all the script files into a single master file before performing reverse engineering.

CHAPTER 10

Triggers and Procedures

About this chapter

This chapter presents triggers and procedures used to generate database constraints.

Contents

What is a trigger?

A trigger is a special form of stored procedure that goes into effect when you insert, delete, or update a specified table or column.

You can use triggers to enforce referential integrity. Referential integrity triggers exist for certain databases, such as Sybase and Oracle.

This chapter uses the following terminology:

Term	Description
Trigger template	Generic model for generating trigger scripts
Template item	Reusable block of script that implements referential integrity
Trigger	Trigger template attached to a table and possibly customized for a table
Trigger script	Executable SQL script containing specific information for a table

You can customize trigger templates, template items, and triggers. Changes in a trigger template can apply to the current model or to all models. Changes in triggers apply to a specific table in the current model.

Using trigger templates

PowerDesigner generates the triggers based on trigger templates.

Each trigger template indicates referential integrity constraints for one of the three trigger types: insert, update, and delete. Each target DBMS has one template per trigger type.

For some target databases, a trigger can include a call to a related procedure. Whenever this is possible, you can define templates for these procedures as well.

Identifying trigger template types

The types of trigger templates that are available depends on the target database management system (DBMS).

The following templates types exist, but are not available for all DBMS.

Insert templates

Template type	Generates trigger or procedure that executes ...
InsertTrigger	With insert
BeforeInsertTrigger	Before insert
AfterInsertTrigger	After insert
InsertProc	When called by InsertTrigger
BeforeInsertProc	When called by BeforeInsertTrigger
AfterInsertProc	When called by AfterInsertTrigger

Update templates

Template type	Generates trigger or procedure that executes ...
UpdateTrigger	With update
BeforeUpdateTrigger	Before update
AfterUpdateTrigger	After update
UpdateProc	When called by UpdateTrigger
BeforeUpdateProc	When called by BeforeUpdateTrigger
AfterUpdateProc	When called by AfterUpdateTrigger

Delete templates

Template type	Generates trigger or procedure that executes ...
DeleteTrigger	With delete
BeforeDeleteTrigger	Before delete
AfterDeleteTrigger	After delete
DeleteProc	When called by DeleteTrigger
BeforeDeleteProc	When called by BeforeDeleteTrigger
AfterDeleteProc	When called by AfterDeleteTrigger

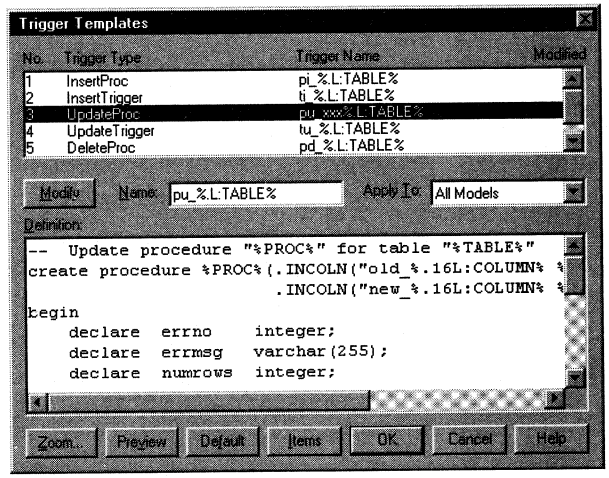
Modifying a trigger template

You can apply a modified template to the current model or to all existing and future models for the target DBMS.

❖ **To modify a template:**

1 Select Dictionary➤Triggers and Procedures➤Trigger Templates.

The list of templates appears.



2 Click a template in the list.

The template definition appears in the Definition text box.

3 Select Current Model from the Apply To dropdown listbox, to modify the template in the current model only.
or
Select All Models from the Apply To dropdown listbox, to modify the template in all models.

> **CDF file stores modified trigger templates**
> The CDF file stores modified trigger templates that you apply to all models. This file is created the first time you select All Models and modify a trigger template. The CDF file takes the same name as the DEF file and goes in the DEF directory. For example, if your PDM uses SQL Anywhere 5, your CDF file has the name SQLANY5.CDF.

4 Type modifications to the template in the Definition text box.

5 Click the Modify button.

A star appears next to the template on the list. It indicates that you have modified the template.

6 Click OK.

Trigger naming conventions

The trigger template indicates naming conventions for trigger scripts that it generates. The naming convention consists of a prefix indicating the trigger type followed by the table code.

The default naming conventions include a variable (%L:TABLE). The name of the resulting trigger script replaces this variable with a lower-case table code. For example, a resulting trigger script may have the name ti_employee.

Insert triggers

Template type	Trigger name
InsertTrigger	ti_%L:TABLE%
BeforeInsertTrigger	tib_%L:TABLE%
AfterInsertTrigger	tia_%L:TABLE%
InsertProc	pi_%L:TABLE%
BeforeInsertProc	pib_%L:TABLE%
AfterInsertProc	pia_%L:TABLE%

Update triggers

Template type	Trigger name
UpdateTrigger	tu_%L:TABLE%
BeforeUpdateTrigger	tub_%L:TABLE%
AfterUpdateTrigger	tua_%L:TABLE%
UpdateProc	pu_%L:TABLE%
BeforeUpdateProc	pub_%L:TABLE%
AfterUpdateProc	pua_%L:TABLE%

Delete triggers

Template type	Trigger name
DeleteTrigger	td_%L:TABLE%
BeforeDeleteTrigger	tdb_%L:TABLE%
AfterDeleteTrigger	tda_%L:TABLE%
DeleteProc	pd_%L:TABLE%
BeforeDeleteProc	pdb_%L:TABLE%
AfterDeleteProc	pda_%L:TABLE%

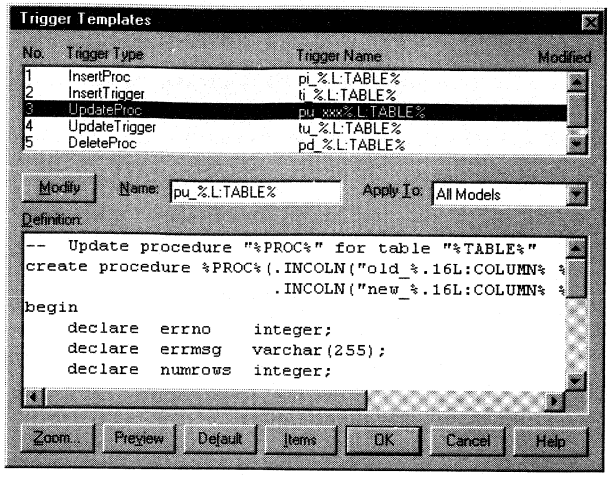
Changing trigger naming conventions

You can change the trigger naming convention in the template.

❖ **To change trigger naming conventions in a template:**

1 Select Dictionary➤Triggers and Procedures➤Trigger Templates.

The list of trigger templates appears.



2 Click a trigger template.

3 Select Current Model from the Apply To dropdown listbox, to modify the template in the current model only.
or
Select All Models from the Apply To dropdown listbox, to modify the template in all models.

4 Type a new name in the Name box.

5 Click Modify.

6 Click OK.

Using procedures in trigger templates

For some target databases, a trigger template can call a related procedure in the form of another template. In this case, the procedure template offers an added level of flexibility to the trigger.

Example with procedure

Allbase SQL accepts procedure calls. The template InsertTrigger calls the procedure, as follows:

```
-- Insert trigger "%TRIGGER%" for table "%TABLE%"
create rule %TRIGGER% after insert on %TABLE%
referencing new as new_ins
   execute procedure
%PROC%(.FKCOLN("new_ins.%COLUMN%", "", ",", ");")
```

The template InsertProc defines the procedure, as follows:

```
-- Insert procedure "%PROC%" for table "%TABLE%"
create procedure %PROC%(.FKCOLN("new_%.16L:COLUMN%
%COLTYPE%", "", ",", ") as")
begin
    declare errno    integer;
    declare errmsg   varchar(255);
    declare numrows  integer;
    .DeclInsertChildParentExist

    .InsertChildParentExist

end;
```

Example without procedure

Oracle 7 does not accept procedure calls. The template InsertTrigger works independently, as follows:

```
-- Insert trigger "%TRIGGER%" for table "%TABLE%"
create trigger %TRIGGER% before insert
on %TABLE% for each row
declare
    integrity_error  exception;
    errno            integer;
    errmsg           char(200);
    dummy            integer;
    found            boolean;
    .DeclInsertChildParentExist

begin
    .InsertChildParentExist

-- Errors handling
exception
    when integrity_error then
       raise_application_error(errno, errmsg);
end;
```

Editing a trigger template

You can use editing features to insert script language in a trigger template.

A trigger template can include the following categories of statements:

Category	Description
Variable	Provides table-specific information in resulting trigger scripts
Template item	Name that refers to a script which implements integrity constraints
Macro	Executes standard function
Function	Executes DBMS-specific function
Operator	Logical operator

❖ To edit a trigger template:

1 Select Dictionary➤Triggers and Procedures➤Trigger Templates.

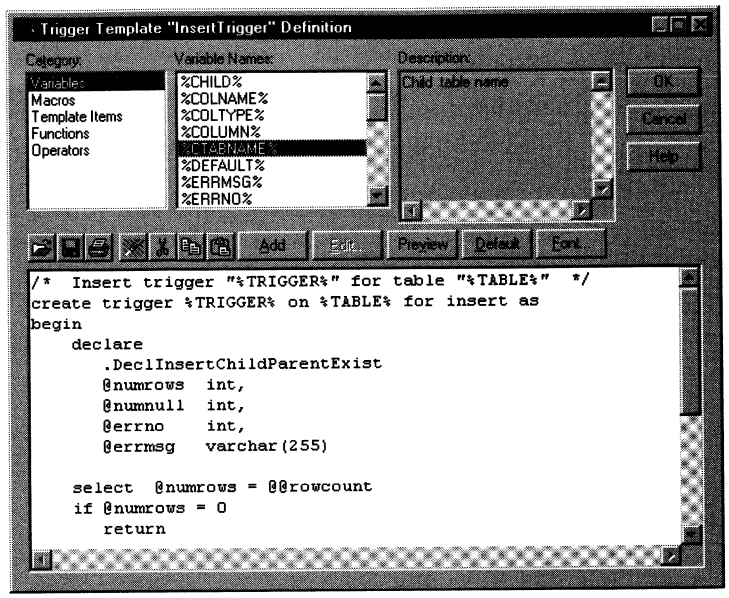
The list of templates appears.

2 Click a template in the list.

3 Select Current Model from the Apply To dropdown listbox, to modify the template in the current model only.
or
Select All Models from the Apply To dropdown listbox, to modify the template in all models.

4 Click the Zoom button.

The template definition dialog box displays the definition of the selected trigger.



5 Click the position in the template definition textbox where you want to insert the script language.

6 Select a category from the category list.

For example, select Variable to display the list of variables.

7 Select an item from the list for that category.

For example, select %CTABLENAME%.

8 Click the Add button.

The selected item is inserted at the cursor position.

9 Click OK.

You return to the list of templates.

10 Click the Modify button.

A star appears next to the template on the list. It indicates that you have modified the template.

11 Click OK.

Using template items

Templates items are reusable blocks of script that implement referential integrity constraints. The availability of template items depends on trigger type and target DBMS.

Trigger templates do not include the details of referential integrity constraints. In a template, the name of a template item takes the place of the script of a referential integrity constraint.

For example, an Oracle 7 trigger template contains the following template item:

```
.InsertChildParentExist
```

This item corresponds to the following definition:

```
.FOREACH_PARENT()
--  Parent "%PARENT%" must exist when inserting a
child in "%CHILD%"
.DEFINE "CURSOR" "cpk%REFNO%_%.25L:TABLE%"
if .JOIN(":new.%FK% is not null", "", " and", " then")
   open  %CURSOR%(.JOIN(":new.%FK%", "", ",", ");")
   fetch %CURSOR% into dummy;
   found := %CURSOR%%FOUND;
   close %CURSOR%;
   if not found then
      .ERROR(-20002, "Parent does not exist in
"%PARENT%". Cannot create child in
"%CHILD%".")
   end if;
end if;
.ENDFOR
```

You can modify and edit the definition of a template item.

Identifying template items

The template items that are available depends on the target database management system (DBMS).

The following template items exist, but are not available for all DBMS.

Insert constraints

The template items below implement referential integrity in insert trigger templates.

Template item	Integrity constraint	Description
Mandatory parent	DeclInsertChildParentExist InsertChildParentExist	Parent must exist when inserting a child

Update constraints

The template items below implement referential integrity in update trigger templates.

Template item	Integrity constraint	Description
DeclUpdateChildParentExist UpdateChildParentExist	Mandatory parent	Parent must exist when updating a child
DeclUpdateChildChangeParent UpdateChildChangeParent	Change parent not allowed	Cannot modify parent code in child
DeclUpdateParentRestrict UpdateParentRestrict	Restrict on update	Cannot modify parent if child exists
DeclUpdateParentCascade UpdateParentCascade	Cascade on update	Modify parent code in all children
DeclUpdateChangeColumn UpdateChangeColumn	Non-modifiable column	Cannot modify column
DeclUpdateParentSetNull UpdateParentSetNull	Set null on update	Set parent code to null in all children
DeclUpdateParentSetDefault UpdateParentSetDefault	Set default on update	Set parent code to default in all children

Delete constraints

The template items below implement referential integrity in delete trigger templates.

Template item	Integrity constraint	Description
DeclDeleteParentRestrict DeleteParentRestrict	Restrict on delete	Cannot delete parent if child exists
DeclDeleteParentCascade DeleteParentCascade	Cascade on delete	Delete parent code in all children
DeclDeleteParentSetNull DeleteParentSetNull	Set null on delete	Delete in parent sets child to null
DeclDeleteParentSetDefault DeleteParentSetDefault	Set default on delete	Delete in parent sets child to default

Constraint messages

You can insert the following template items in any trigger template. They generate error messages that indicate the violation of an integrity constraint.

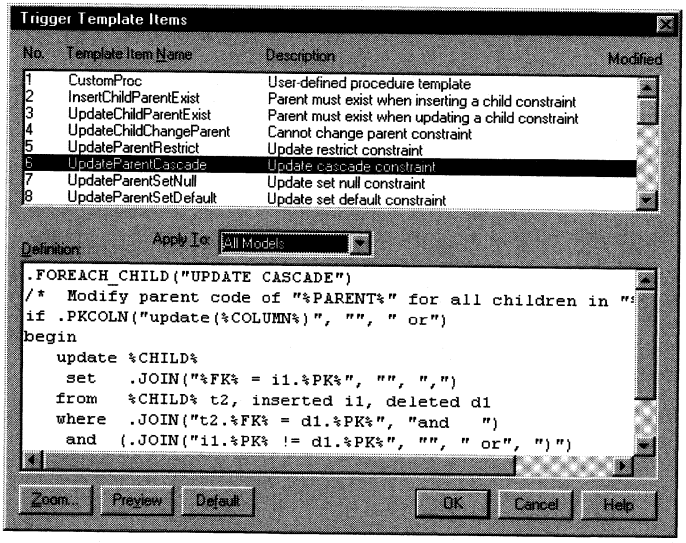
Template item	Description
UseErrorMsgText	Error handling without a message table
UseErrorMsgTable	Error handling with a message table

Modifying a template item

❖ **To modify a template item:**

1 Dictionary➤Triggers and Procedures➤Trigger Template Items.

The list of template items appears.



2 Click a template item.

Its definition appears in the lower panel.

3 Select Current Model from the Apply To dropdown listbox, to modify the item in the current model only.
or
Select All Models from the Apply To dropdown listbox, to modify the item in all models.

4 Type changes to the definition.

5 Click OK.

Editing a template item

You can use editing features to insert script language in a template item.

A template item can include the following categories of statements:

Category	Description
Variable	Provides table-specific information in resulting trigger scripts
Macro	Executes standard function
Function	Executes DBMS-specific function
Operator	Logical operator

❖ To edit a template item:

1 Select Dictionary➢Triggers and Procedures➢Trigger Template Items.

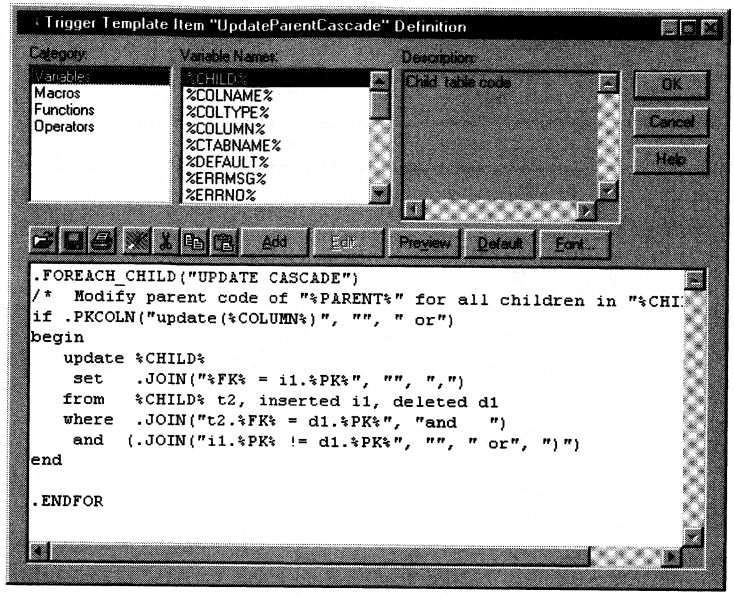
The list of template items appears.

2 Click a template item in the list.

3 Select Current Model from the Apply To dropdown listbox, to modify the item in the current model only.
or
Select All Models from the Apply To dropdown listbox, to modify the item in all models.

4 Click the Zoom button.

The template item definition dialog box displays the definition of the selected item.



5 Click the position in the definition textbox where you want to insert the script language.

6 Select a category from the category list.

For example, select Variable to display the list of variables.

7 Select an item from the list for that category.

For example, select %CHILD%.

8 Click the Add button.

The selected item is inserted at the cursor position.

9 Click OK.

You return to the list of template items.

10 Click OK.

Using triggers

A trigger is a trigger template attached to a table in a model. If you modify a trigger, it applies only to a specific table. In the resulting scripts, specific table values replace template variables.

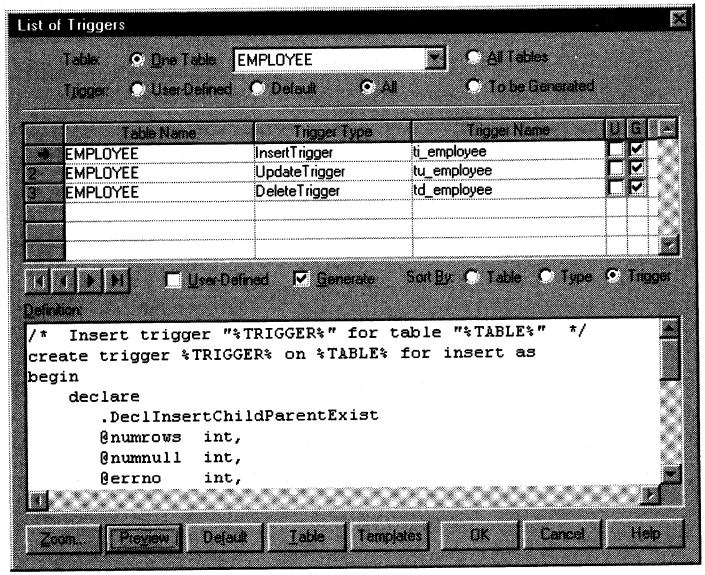
Modifying a trigger

Based on trigger templates, PowerDesigner generates default triggers for each table. If you modify a default trigger, it becomes a user-defined trigger.

❖ To modify a trigger:

1 Select Dictionary➤Triggers and Procedures➤List of Triggers.

The list of triggers appears.



2 Click the One Table radio button.

3 Select a table from the dropdown listbox.

4 Click the All radio button.

The list shows all the triggers attached to the selected table.

5 Click a trigger in the list.

The trigger definition appears in the Definition text box.

6 Type modifications to the trigger.

The User-Defined checkbox is selected automatically.

7 Click OK.

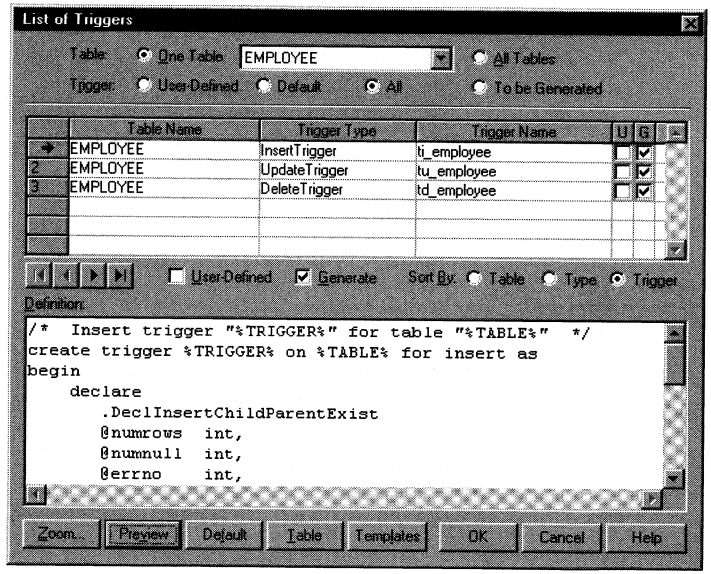
Previewing a trigger

You preview a trigger in order to display the generated contents of a trigger with instantiated variables.

❖ To preview a trigger:

1 Select Dictionary➤Triggers and Procedures➤List of Triggers.

The list of triggers appears.



2 Click the One Table radio button.

3 Select a table from the dropdown listbox.

4 Click the All radio button.

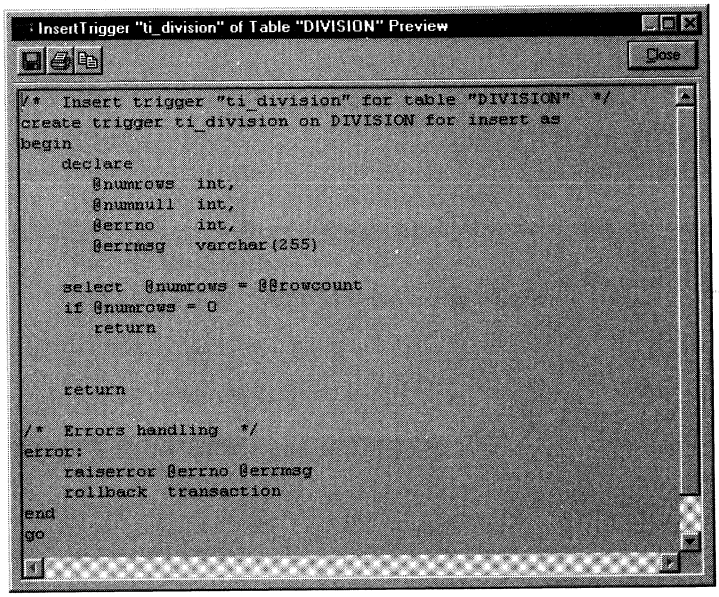
The list shows all the triggers attached to the selected table.

5 Click a trigger in the list.

The trigger definition appears in the Definition text box.

6 Click the Preview button.

The Trigger Preview window displays the trigger with instantiated variables.



7 Click OK in each of the dialog boxes.

Editing a trigger

You can use editing features to insert script language in a trigger.

A trigger can include the following categories of statements:

Category	Description
Variable	Provides table-specific information in resulting trigger scripts
Macro	Executes standard function
Function	Executes DBMS-specific function
Operator	Logical operator

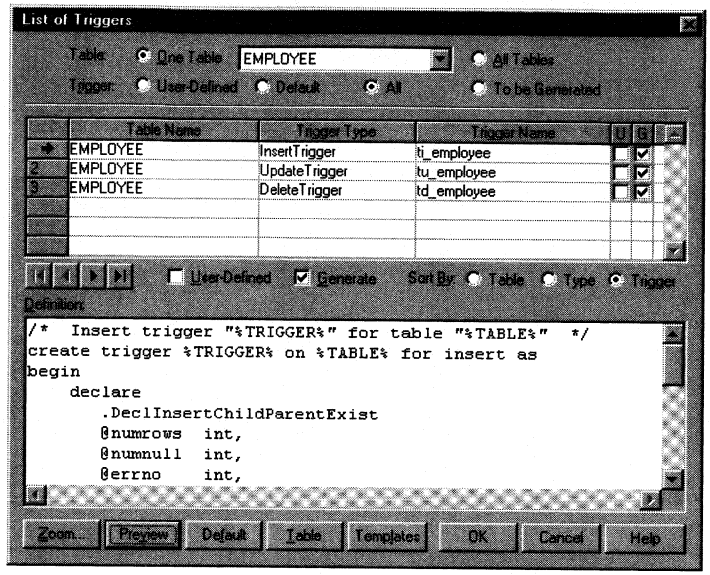
It can also include the code for any of the following objects:

Tables
Columns
Joins
Business rules

❖ To edit a trigger:

1 Select Dictionary➤Triggers and Procedures➤List of Triggers.

The list of triggers appears.



2 Click the One Table radio button.

3 Select a table from the dropdown listbox.

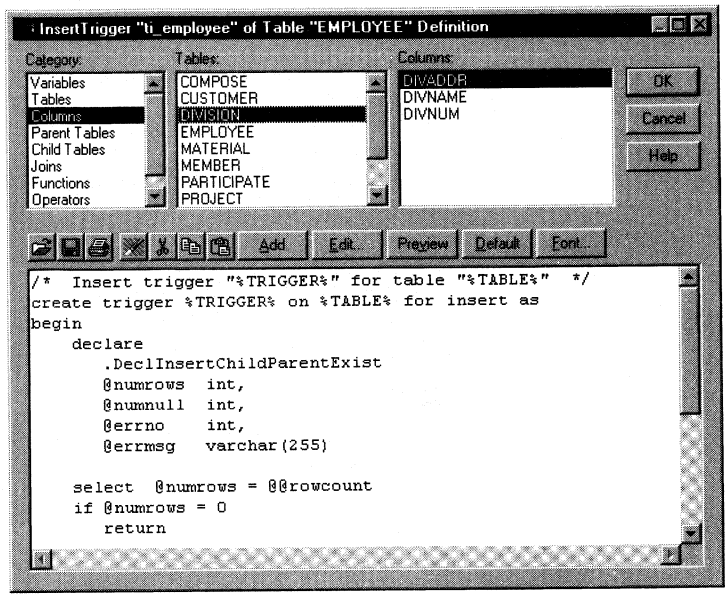
4 Click the All radio button.

The list shows all the triggers attached to the selected table.

5 Click a trigger in the list.

6 Click the Zoom button.

The trigger definition dialog box displays the definition of the selected trigger.



7 Click the position in the definition textbox where you want to insert the script language.

8 Select a category from the category list.

For example, select Variable to display the list of variables.

9 Select an item from the list for that category.

For example, select %CHILD%.

10 Click the Add button.

The selected item is inserted at the cursor position.

11 Click OK.

You return to the list of triggers. The User-Defined checkbox is selected.

12 Click OK.

Defining stored procedures and functions

You can define stored procedures and functions for any DBMS that supports them.

Functions return results. Procedures do not return results.

Stored procedures and functions do not implement integrity constraints.

Defining templates for stored procedures and functions

You define templates for stored procedures and functions as template items.

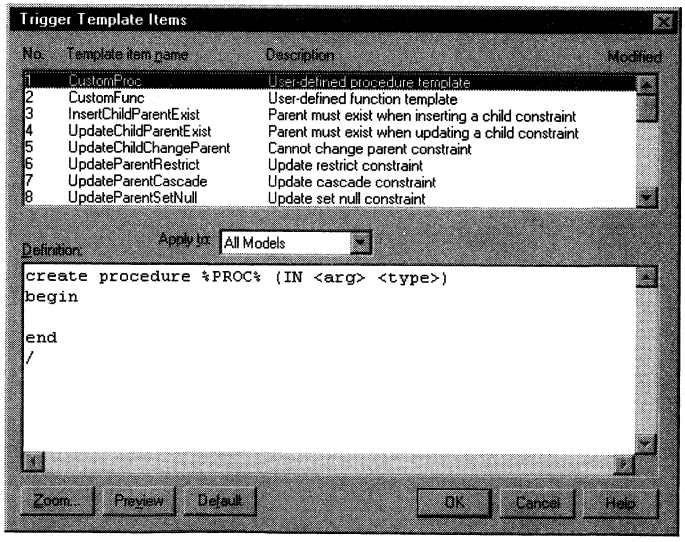
These templates provide the structure for creating stored procedures and functions.

❖ To define a template for stored procedures or functions:

1 Select Dictionary➤Triggers and Procedures➤Template Items.

2 Click CustomProc to edit the stored procedure template.
or
Click CustomFunc to edit the function template.

The Definition textbox displays the template.



3 Select Current Model from the Apply To dropdown listbox, to modify the template in the current model only.
or
Select All Models from the Apply To dropdown listbox, to modify the template in all models.

4 Type changes to the template.

5 Click OK.

Creating stored procedures and functions

You create stored procedures and functions using the structure of templates.

❖ **To create a stored procedure or a function:**

1 Select Dictionary➤Triggers and Procedures➤List of Procedures.

The User-Defined Procedures dialog box displays the list of stored procedures and functions.

2 Click a blank line on the list.

An arrow appears at the beginning of the line.

3 Type a name and a code.
or
Type a name and click the button in the code column.
or
Type a code and click the button in the name column.

4 If you are creating a function, select the F checkbox.

The procedure or the function template appears in the definition window.

5 Type the procedure or function in the Definition textbox.

6 Click OK.

Editing a stored procedure or a function

You can use editing features to insert script language in a procedure or function.

A procedure or a function can include the following categories of statements:

Category	Description
Function	Executes DBMS-specific function
Macro	Executes standard function
Operator	Logical operator

It can also include the code for any of the following objects:

Tables
Columns
Joins
Business rules

❖ To edit a stored procedure or a function:

1 Select Dictionary➤Triggers and Procedures➤List of Procedures.

The User-Defined Procedures dialog box displays the list of stored procedures and functions.

2 Click a procedure or function in the list.

An arrow appears at the beginning of the line.

3 Click the Zoom button.

The Procedure Definition dialog box displays the definition of the selected item.

4 Click the position in the definition textbox where you want to insert the script language.

5 Select a category from the category list.

For example, select Variable to display the list of variables.

6 Select an item from the list for that category.

For example, select %CHILD%.

7 Click the Add button.

The selected item is inserted at the cursor position.

8 Click OK in each of the dialog boxes.

Using variables in triggers

You can use variables in trigger templates, template items, and triggers. During trigger script generation, these variables are replaced by values for a specific model or table.

List of variables in triggers

Table variables

Table names and codes instantiate the following variables:

Variable	Value
%TABLE%	Table code
%TABNAME%	Table name
%CHILD%	Child table code
%CTABNAME%	Child table name
%PARENT%	Parent table code
%PTABNAME%	Parent table name

Column variables

Column names and codes instantiate the following variables:

Variable	Value
%COLUMN%	Column code
%COLNAME%	Column name
%COLTYPE%	Column data type
%FK%	Foreign key column code
%FKNAME%	Foreign key column name
%PK%	Primary key column code
%PKNAME%	Primary key column name

Reference variables

Reference names and codes instantiate the following variables:

Variable	Value
%REFR%	Reference code
%REFNAME%	Reference name
%REFNO%	Reference number

Trigger and procedure variables

Information about triggers and procedures instantiates the following variables:

Variable	Value
%TRIGGER%	Trigger name
%TRGTYPE%	Trigger type
%PROC%	Procedure name

Error message variables

Information about error messages instantiates the following variables:

Variable	Value
%ERRMSG%	Error message
%ERRNO%	Error number
%MSGNO%	Name of error number column in the message table
%MSGTAB%	Name of error message table
%MSGTXT%	Name of error message column in the message table

Formatting variables

Variables have a syntax that can force a format on their values, as follows:

- Force values to lower-case or upper-case characters
- Truncate the length of values

When a variable does not specify formatting, its values have the same format as in the PDM.

Format code	Format of variable value in script
.L	Lower-case characters
.T	Removes blank spaces
.U	Upper-case characters
.n	Maximum length where n is the number of characters
*.n*J	Justifies to fixed length where n is the number of characters

You embed formatting options in variable syntax as follows:

%.format:variable%

For example:

```
%.L:TABLE%
```

The table below shows formatted variables and their results in a script for the table EMPLOYEE.

Template statement with variable	Resulting script statement
create trigger %TABLE%	create trigger EMPLOYEE
create trigger %.L:TABLE%	create trigger employee
create trigger %.U:TABLE%	create trigger EMPLOYEE
create trigger %.4:TABLE%	create trigger EMPL
create trigger %.4L:TABLE%	create trigger empl

Using macros

You can use predefined macros in trigger templates, template items, triggers, and procedures. Macros perform specific functions.

ALLCOL

Description

Repeats a statement for each column in a table

Syntax

.ALLCOL("*statement*","*prefix*","*suffix*","*last_suffix*")

Argument	Description
statement	Statement to repeat for each column
prefix	Prefix for each new line
suffix	Suffix for each new line
last suffix	Suffix for the last line

Example

In a trigger for the table AUTHOR, the following macro:

```
.ALLCOL("%COLUMN% %COLTYPE%","","","","")
```

generates the following trigger script:

```
AU_ID char(12),
AU_LNAME varchar(40),
AU_FNAME varchar(40),
AU_BIOGRAPH long varchar,
AU_ADVANCE numeric(8,2),
AU_ADDRESS varchar(80),
CITY varchar(20),
STATE char(2),
POSTALCODE char(5),
AU_PHONE char(12);
```

DEFINE

Description

Defines a variable and initializes its value.

Syntax

.DEFINE "*variable*" "*value*"

Argument	Description
variable	Variable name (without % signs)
value	Variable value (may include another variable surrounded by % signs)

Example

In a trigger for the table AUTHOR, the following macro:

```
.DEFINE "TRIGGER" "T_%TABLE%"
message 'Error: Trigger(%TRIGGER%) of table %TABLE%'
```

generates the following trigger script:

```
message 'Error: Trigger(T_AUTHOR) of table AUTHOR';
```

DEFINEIF

Description

Defines a variable and initializes its value if the test value is not null

Syntax

.DEFINEIF "*test_value*" "*variable*" "*value*"

Argument	Description
test_value	Value to test
variable	Variable name (without % signs)
value	Variable value (may include another variable surrounded by % signs)

Example

For example, to define a variable for a default data type:

```
%DEFAULT%
.DEFINEIF "%DEFAULT%" "_DEFLT"" "%DEFAULT%"
Add %COLUMN% %DATATYPE% %_DEFLT%
```

ERROR

Description

Handles errors

Syntax

.ERROR (*errno*, "*errmsg*")

Argument	Description
errno	Error number
errmsg	Error message

Example

```
.ERROR(-20001, "Parent does not exist, cannot insert
child")
```

FKCOLN

Description

Repeats a statement for each foreign key column in a table

Syntax

.FKCOLN("*statement*","*prefix*","*suffix*","*last_suffix*")

Argument	Description
statement	Statement to repeat for each column
prefix	Prefix for each new line
suffix	Suffix for each new line
last suffix	Suffix for the last line

Example

In a trigger for the table TITLEAUTHOR, the following macro:

```
message .FKCOLN("'%COLUMN% is a foreign key
column'","","","","";")
```

generates the following trigger script:

```
message 'AU_ID is a foreign key column,
TITLE_ISBN is a foreign key column;'
```

Column variable only
For columns, the macro FKCOLN only accepts the variable %COLUMN%.

FOREACH_CHILD

Description

Repeats a statement for each parent-to-child reference in the current table fulfilling a condition

Syntax

.FOREACH_CHILD ("*condition*")
"*statement*"
.ENDFOR

Argument	Description
condition	Reference condition (see below)
statement	Statement to repeat

Condition	Selects
UPDATE RESTRICT	Restrict on update
UPDATE CASCADE	Cascade on update
UPDATE SETNULL	Set null on update
UPDATE SETDEFAULT	Set default on update
DELETE RESTRICT	Restrict on delete
DELETE CASCADE	Cascade on delete
DELETE SETNULL	Set null on delete
DELETE SETDEFAULT	Set default on delete

Example

In a trigger for the table TITLE, the following macro:

```
.FOREACH_CHILD("DELETE RESTRICT")
--  Cannot delete parent "%PARENT%" if children still
exist in "%CHILD%"
.ENDFOR
```

generates the following trigger script:

```
--  Cannot delete parent "TITLE" if children still
exist in "ROYSCHED"
--  Cannot delete parent "TITLE" if children still
exist in "SALE"
--  Cannot delete parent "TITLE" if children still
exist in "TITLEAUTHOR"
```

FOREACH_COLUMN

Description

Repeats a statement for each column in the current table fulfilling a condition

Syntax

.FOREACH_COLUMN ("*condition*")
"*statement*"
.ENDFOR

Argument	Description
condition	Column condition (see below)
statement	Statement to repeat

Condition	Selects
empty	All columns
PKCOLN	Primary key columns
FKCOLN	Foreign key columns
NMFCOL	Non-modifiable columns (columns that have Cannot Modify selected as a check parameter)
INCOLN	Triggering columns (primary key columns, foreign key columns; and non-modifiable columns)

Example

In a trigger for the table TITLE, the following macro:

```
.FOREACH_COLUMN("NMFCOL")
-- "%COLUMN%" cannot be modified
.ENDFOR
```

generates the following trigger script:

```
-- "TITLE_ISBN" cannot be modified
-- "PUB_ID" cannot be modified
```

FOREACH_PARENT

Description

Repeats a statement for each child-to-parent reference in the current table fulfilling a condition

Syntax

.FOREACH_PARENT ("*condition*")
"*statement*"
.ENDFOR

Argument	Description
condition	Reference condition (see below)
statement	Statement to repeat

Condition	Selects references defined with ...
empty	All references
FKNULL	Non-mandatory foreign keys
FKNOTNULL	Mandatory foreign keys
FKCANTCHG	Non-modifiable foreign keys

Example

In a trigger for the table SALE, the following macro:

```
.FOREACH_PARENT("FKCANTCHG")
--  Cannot modify parent code of "%PARENT%" in child
"%CHILD%"
.ENDFOR
```

generates the following trigger script:

```
--  Cannot modify parent code of "STORE" in child
"SALE"
--  Cannot modify parent code of "TITLE" in child
"SALE"
```

INCOLN

Description

Repeats a statement for each primary key column, foreign key column, or non-modifiable column in a table

Syntax

.INCOLN("*statement*","*prefix*","*suffix*","*last_suffix*")

Argument	Description
statement	Statement to repeat for each column
prefix	Prefix for each new line
suffix	Suffix for each new line
last suffix	Suffix for the last line

Example

In a trigger for the table TITLE, the following macro:

```
.INCOLN("%COLUMN% %COLTYPE%","","","",";")
```

generates the following trigger script:

```
TITLE_ISBN char(12),
PUB_ID char(12);
```

JOIN

Description

Repeats a statement for column couple in a join

Syntax

.JOIN("*statement*","*prefix*","*suffix*","*last_suffix*")

Argument	Description
statement	Statement to repeat for each column
prefix	Prefix for each new line
suffix	Suffix for each new line
last suffix	Suffix for the last line

Example

In a trigger for the table TITLE, the following macro:

```
where .JOIN("%PK%=%FK%", " and", "", ";")
message 'Reference %REFR% links table %PARENT% to
%CHILD%'
```

generates the following trigger script:

```
message 'Reference TITLE_PUB links table PUBLISHER to
TITLE
```

> **Primary key and foreign keys variables only**
> For columns, the macro JOIN only accepts the variables %PK% and %FK%.

NMFCOL

Description

Repeats a statement for each non-modifiable column in a table. Non-modifiable columns have Cannot Modify selected as a check parameter.

Syntax

.NMFCOL("*statement*","*prefix*","*suffix*","*last_suffix*")

Argument	Description
statement	Statement to repeat for each column
prefix	Prefix for each new line
suffix	Suffix for each new line
last suffix	Suffix for the last line

Example

In a trigger for the table TITLE, the following macro:

```
.NMFCOL("%COLUMN% %COLTYPE%","","","","")
```

generates the following trigger script:

```
TITLE_ISBN char(12),
PUB_ID char(12);
```

PKCOLN

Description

Repeats a statement for each primary key column in a table

Syntax

.PKCOLN("*statement*","*prefix*","*suffix*","*last_suffix*")

Argument	Description
statement	Statement to repeat for each column
prefix	Prefix for each new line
suffix	Suffix for each new line
last suffix	Suffix for the last line

Example

In a trigger for the table TITLEAUTHOR, the following macro:

```
message .PKCOLN("'%COLUMN% is a primary key
column'","","","","")
```

generates the following trigger script:

```
message 'AU_ID is a primary key column',
        'TITLE_ISBN is a primary key column';
```

Column variable only
For columns, the macro PKCOLN only accepts the variable %COLUMN%.

Generating triggers and procedures

You can create or modify database triggers and procedures directly via an ODBC driver or indirectly using a script.

Trigger generation parameters

Generation parameters indicate the following:

- Triggers and procedures to generate
- Commands to generate in the trigger script

The availability of generation parameters depends on the target database. Unavailable parameters display in gray and are not selectable.

The tables below indicate the generation that results from the selection of generation parameters.

Selection parameters

Parameter	Resulting generation command
Triggers: All	Apply generation parameters to all triggers, both default and user-defined
Triggers: By default	Apply generation parameters to default triggers only
Triggers: User-defined	Apply generation parameters to user-defined triggers only
Tables: All tables	Apply generation parameters to triggers and procedures for all tables
Tables: List	Apply generation parameters to triggers and procedures for selected tables only
Procedures: All	Apply generation parameters to all procedures
Procedures: List	Apply generation parameters to selected procedures only

Creation parameters

Parameter	Resulting generation command
Create procedure	Create procedure
Create trigger	Create trigger
For insert	Generate insert triggers
For update	Generate update triggers
For delete	Generate delete triggers

Deletion parameters

Parameter	Resulting generation command
Drop procedure	Delete procedure
Drop trigger	Delete trigger

Database parameters

Parameter	Resulting generation command
Open database	Open existing database
Close database	Close existing database

Error message parameters

Parameter	Resulting generation command
Standard	Generate standard error messages
User-defined	Generate user-defined error messages

Script options and referential integrity options for triggers

Script options indicate the format of generation scripts for triggers and procedures.

Referential integrity options indicate whether to generate referential integrity as a trigger or as a declarative statement.

The availability of these options depends on the target database. Unavailable options display in gray and you cannot select them.

Script options

Option	Result of selection
Uppercase	Script contains all uppercase characters
Lowercase	Script contains all lowercase characters
No accent	Non-accented characters replace accented characters in script
Codes	Script refers to objects by their codes
Names	Script refers to objects by their names
Windows (ANSI)	Script uses ANSI character set
DOS (OEM)	Script uses DOS character set
Add titles	Each section of the script includes commentary in the form of titles (for example, Database Name: TUTORIAL)
Check model	Check the PDM before generating the database or script, and stop generation if an error is found

Referential integrity options

There are two ways to generate referential integrity in a database:

Option	Result of selection
Trigger	Generates a trigger
Declarative	Generates a declarative statement in a script

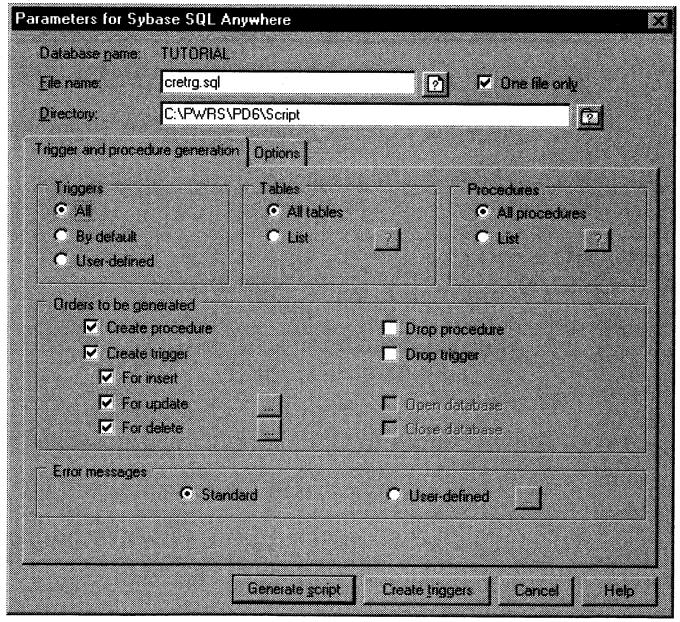
Generating a trigger creation script

PowerDesigner generates trigger and procedure scripts that you can run in your DBMS environment.

❖ To generate a trigger and procedure script:

1 Select Database➤Generate Triggers and Procedures.

The Generation Parameters dialog box appears.



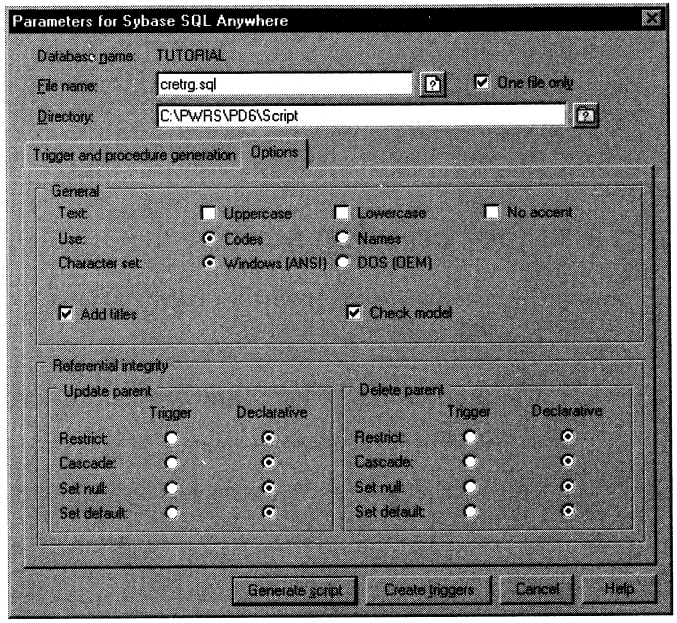
2 Type a filename in the File Name box.

3 Type a directory name in the Directory box.

4 Select generation parameters.

5 Click the Options tab.

The Options page appears.



6 Select script options and referential integrity options.

7 Click the Generate Script button.

Creating triggers directly in a database

PowerDesigner can generate triggers directly. To do so, you must connect to a data source via an ODBC driver.

❖ To create triggers directly in a database:

1 Select Database➤Generate Triggers and Procedures.

The Generation Parameters dialog box appears.

2 Type a filename in the File Name box.

3 Type a directory name in the Directory box.

4 Select generation parameters.

5 Click the Options tab.

The Options page appears.

6 Select script options and referential integrity options.

7 Click the Create Triggers button.

A dialog box asks you to identify a data source and connection parameters.

8 Select a data source from the dropdown listbox.

9 Type your user name and password.

10 Click Connect and, if prompted by your data source, enter additional connection parameters.

A message window shows the progress of the generation process.

CHAPTER 11

Client Interface

About this chapter

This chapter describes standard 4GL extended attributes and how to transfer these attributes and database scripts between a 4GL and a Physical Data Model (PDM).

Contents

For more information

For a complete list of standard extended attributes, see the appendix, "Client Reference." For a description of PowerBuilder catalog attributes, see the chapter "Using PowerBuilder Catalog Attributes."

Lists of extended attributes for application generation are included in the appendix "Generation Variables and Attributes."

Using 4GL extended attributes

You can import 4GL extended attributes to a PDM from an EXA file. You can also define extended attributes for all new PDMs by setting a default EXA file.

For information on importing, exporting, and setting extended attribute defaults, see the chapter "Building a Physical Data Model."

4GL extended attributes files

PowerDesigner supplies sets of extended attributes for certain 4GL in predefined EXA files

4GL	EXA file
Axiant	COGNOS.EXA
NS-Access	NSACCESS.EXA
PowerBuilder 4	POWERBLD.EXA
PowerBuilder 5	POWERPFC.EXA
PowerHouse	COGNOS.EXA
Progress	PROGRESS.EXA
Uniface	UNIFACE.EXA

Using a variable as a default value

You can define the default value for an extended attribute as a variable.

PowerDesigner provides standard variables that are compatible with all 4GL. Depending on the database that you use, you may or may not have access to additional variables.

During generation, the interface replaces these variables with default values for a particular table or column.

❖ **To use a variable as the default value of an extended attribute:**

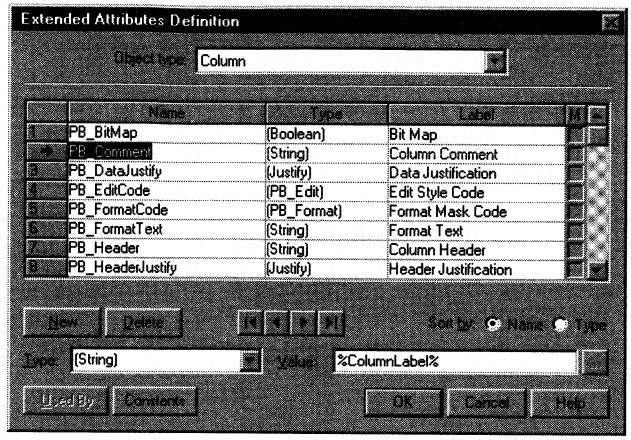
1 Select Dictionary➤Extended Attributes➤List of Attributes.

The list of extended attributes appears.

2 Click an attribute in the list.

An arrow appears at the beginning of the line.

3 Type a variable name in the Value textbox.



4 Click OK.

Using standard variables

PowerDesigner includes the following standard variables. You can use these variables in extended attribute values with any 4GL.

Variable	Value
%TableName%	Table name
%TableCode%	Table code
%TableLabel%	Table label
%ColumnName%	Column name
%ColumnCode%	Column code
%ColumnLabel%	Column label

Using 4GL variables

You can define a default value for an extended attribute as a variable. During generation, the interface replaces the variable with the default value for a particular table or column.

You can only use the variables listed below for the designated 4GL.

Progress variables

For Progress, you can use the following variables:

Variable	Value
%ColumnMaximum%	Maximum value entered in the check parameters of a column
%ColumnMinimum%	Minimum value entered in the check parameters of a column

Axiant and PowerHouse variables

For Axiant and PowerHouse, you can use the following variables:

Variable	Value
%+%	& followed by a carriage return and four spaces
%Format%	%+% followed by "FORMAT" followed by column format
%Heading%	%+% followed by "HEADING" followed by column label
%Help%	%+% followed by "LABEL" followed by column name
%Initial%	%+% followed by "INITIAL" followed by default value of column
%MinMax%	%+% followed by "VALUES min TO max" followed by minimum and maximum values of column
%Picture%	%+% followed by "PICTURE" followed by column format
%Shift%	%+% followed by "DOWNSHIFT" followed by lowercase value
%DomainCode%	Domain code
%Size%	Column size
%Type%	Character, date, or numeric

Uniface variables

For Uniface, you can use the following variable:

Variable	Value
%Format%	Check parameter format of a column

Transferring information to 4GL tools

You can transfer the design information from a PDM to 4GL tools. These tools can implement data structures and extended attributes.

Generating a Progress database definition

You use a database definition (DF) file to transfer information to Progress. This file makes it possible to implement PDM design choices in a Progress application.

This file translates PDM objects into Progress objects, as follows:

PDM object	Progress object
Table and reference	File
Column	Field
Index	Index
Column index	Field index

For Progress 7, you generate separate trigger definition scripts (T files).

Preparation

Before you can generate a Progress definition file, you must prepare the PDM as follows:

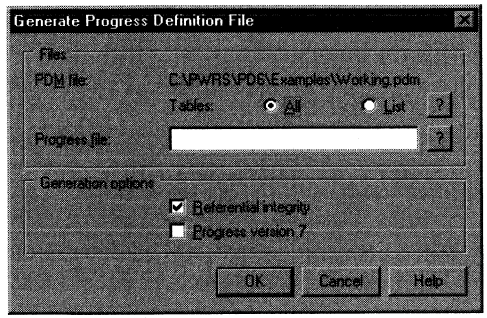
- Change the target database to Progress
- Import Progress extended attributes (PROGRESS.EXA)

❖ **To generate a Progress database definition:**

1 Select Client➤Other Interfaces➤Progress.

> **AppModeler for the Web**
> The Web menu replaces the Client menu in AppModeler for the Web.

The Generate Progress Definition Files dialog box appears.



2 Select the All radio button to generate all tables.
or
Select List or the ? button to display the list of tables.
Select the tables you want to generate.
Click the Add button.
Click OK.

3 Type the name of the destination DF file.

4 Select Referential Integrity, if you want to generate referential integrity.

5 Select Progress Version 7 if you are using this version.

6 Click OK.

Implementation

From Progress, you upload the resulting DF file (and T files if you are using Progress 7).

Generating a Uniface Conceptual Schema

You use a Case Interface Format (CIF) file to transfer information to Uniface. This file makes it possible to implement PDM design choices in a Uniface Conceptual Schema. This schema is an entity-relationship data model.

The CIF file translates PDM objects into Uniface objects as follows:

PDM object	Conceptual schema object
Domain	Domain
Table	Entity
Index	Key
Column	Field
Reference	Relationship

Preparation

Before you can generate a Uniface Conceptual Schema, you must prepare the PDM as follows:

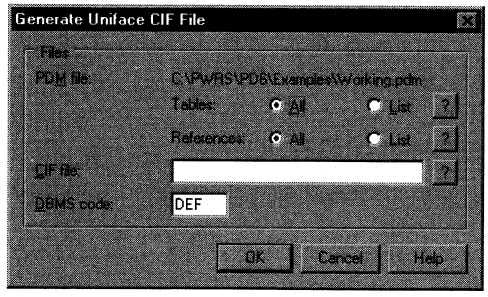
- Select a target database supported by Uniface
- Import Uniface extended attributes (UNIFACE.EXA)

❖ To generate a Uniface conceptual schema:

1 Select Client➤Other Interfaces➤Uniface.

> **AppModeler for the Web**
> The Web menu replaces the Client menu in AppModeler for the Web.

The Generate CIF File for Uniface dialog box appears.



2 Select the All radio button to generate all tables.
or
Select List or the ? button to display the list of tables.
Select the tables you want to generate.
Click the Add button.
Click OK.

3 Select the All radio button to generate all references.
or
Select List or the button to display the list of references.
Select the references you want to generate.
Click the Add button.
Click OK.

4 Type the name of the destination CIF file.

5 Type the three-letter Uniface mnemonic for the target database in the DBMS Code field.

6 Click OK.

Implementation

From Uniface, you upload the resulting CIF file.

Generating Axiant and PowerHouse definitions

You use a PowerHouse Definition Language (PDL) file to transfer information to Axiant and PowerHouse. This file makes it possible to implement PDM design choices in an Axiant or PowerHouse dictionary.

The PDL file translates PDM objects into Axiant and PowerHouse objects as follows:

PDM object	Axiant or PowerHouse object
Domain	USAGE
Column	ELEMENT

Preparation

Before you can generate information for Axiant or PowerHouse, you must prepare the PDM as follows:

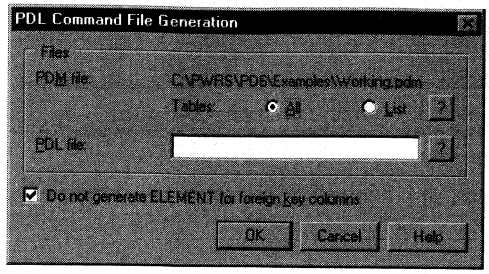
- Select a target database supported by Axiant or PowerHouse
- Import Cognos extended attributes (COGNOS.EXA)

❖ **To generate Axiant or PowerHouse definitions:**

1 Select Client➤Other Interfaces➤Axiant.
or
Select Client➤Other Interfaces➤PowerHouse.

> **AppModeler for the Web**
> The Web menu replaces the Client menu in AppModeler for the Web.

The PDL Command File Generation dialog box appears.



2 Select the All radio button to generate all tables.
or
Select List or the ? button to display the list of tables.
Select the tables you want to generate.
Click the Add button.
Click OK.

3 Type the name of the destination PDL file.

4 Select the checkbox Do Not Generate Element for Foreign Key Columns, if you want to generate primary key columns only.
or
Clear this checkbox, if you want to generate foreign keys that have unique names.

> **Renaming foreign keys before generation**
> In Axiant and PowerHouse dictionaries, element names must be unique. The generation process only takes into account the first occurrence of a key name. If you want to generate elements for foreign keys, they cannot have the same names as primary keys.

5 Click OK.

Implementation

For PowerHouse, run the **use** command to include the resulting PDL file in a PowerHouse dictionary.

From Axiant, you upload the resulting PDL file into the dictionary.

Generating an NS-Access import file

You use an NS-DK Import file to transfer database design information to NS-Access. This file contains an internal representation of the database for NS-DK and an external representation for the target DBMS. This file makes it possible to implement PDM design choices in an NS-Dict/2 dictionary.

The import file translates PDM objects into NS-Access objects as follows:

PDM object	NS-Access object
Domain	Typedef
Table	Table
Column	Column
Index and index column	Key

Preparation

Before you can generate an NS-Access import file you must prepare the PDM as follows:

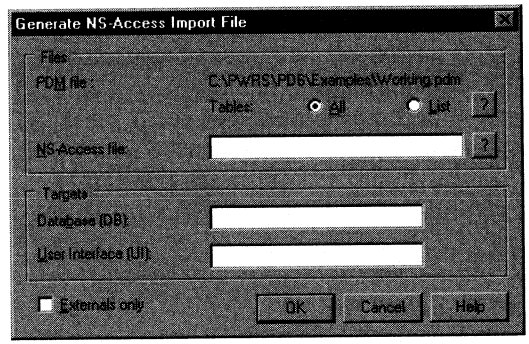
- Select a target database supported by NS-Access
- Import NS-Access extended attributes (NSACCESS.EXA)

❖ To generate an NS-Access import file:

1 Select Client➤Other Interfaces➤NS-DK/1 (NS-Access).

> **AppModeler for the Web**
> The Web menu replaces the Client menu in AppModeler for the Web.

The Generate NS-Access Import File dialog box appears.



2 Select the All radio button to generate all tables.
or
Select List or the ? button to display the list of tables.
Select the tables you want to generate.
Click the Add button.
Click OK.

3 Type the name of the NS-Access import file.

4 Type the logical name of the target database as defined in NS-Access in the Database field.

5 Type the logical name of the user interface as defined in NS-Access in the User Interface field.

6 Select Externals only, if you want to generate external representation of the database only.

> **Generating multiple database descriptions**
> For NS-Access, the internal representation of the database does not change when you change target databases. If you generate NS-Access extended attributes from the same PDM for different target databases you only have to generate the internal representation once. For all subsequent target databases you can generate the external representation only.

7 Click OK.

Implementation

From NS-Access, you import the resulting NS-Access Import File.

Generating NS-Design segment definitions

You transfer database information in the form of segment definitions to NS-Design. Resource segment (SEG) files contain these definitions. You generate one SEG file per table.

Resource segment files translate PDM objects into NS-Design objects as follows:

PDM object	NS-Design object
Table	Segment
Index	Index

Preparation

Before you can generate NS-Design resource segment files, you must prepare the PDM as follows:

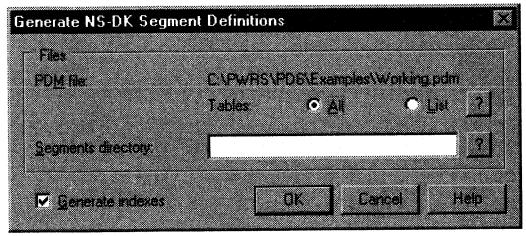
- Select a target database supported by NS-Design

❖ **To generate NS-Design resource segment files:**

1 Select Client➤Other Interfaces➤NS-DK/1 (NS-Design).

AppModeler for the Web
The Web menu replaces the Client menu in AppModeler for the Web.

The Generate Segment Definitions for NS-DK dialog box appears.



2 Select the All radio button to generate all tables.
or
Select List or the ? button to display the list of tables.
Select the tables you want to generate.
Click the Add button.
Click OK.

3 Type the name of the directory in which to generate the SEG files.

4 Select the Generate Indexes checkbox, if you want to generate indexes.

5 Click OK.

Implementation

From NS-Design, you import the resulting SEG files.

Transferring scripts between Omnis and a PDM

PowerDesigner ships with a special interface for transferring SQL scripts between Omnis 7 and a PDM. The PDOMNIS.LBR file provides this interface.

> **Copy the PDOMNIS.LBR file**
> The PowerDesigner Setup program installs PDOMNIS.LBR in the PD6\TOOLS directory. You must copy this file to the EXTERNAL directory in the Omnis path.

Generating a SQL script for Omnis

You can generate SQL scripts from a PDM and import them into an Omnis 7 database using the PDOMNIS.LBR interface.

The interface only works in a Windows environment. After you generate an Omnis dictionary from a SQL script, you can open the dictionary in other environments supported by Omnis.

❖ To generate a SQL script file for Omnis:

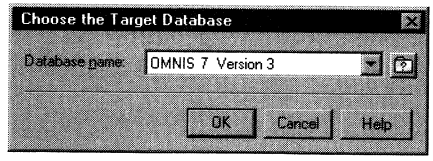
1 Open your PDM.

2 Select Database➤Change Target Database.

A confirmation box asks you if you want to continue.

3 Click Yes.

A database selection dialog box appears.



4 Select Omnis 7 Version 3 from the Database Name dropdown listbox. Click OK.

A message box tells you that the database has been changed.

5 Select Database➤Generate Database.

The Omnis 7 generation parameters dialog box appears.

For information on database script generation options, see the chapter on "Database Creation and Modification."

6 Click the Generate Script button.

A message window shows the progress of the script generation. After generation is complete, it tells you how to generate the script in Omnis.

7 Open Omnis.

8 Select File➤New Library from the Omnis menu bar.
or
Select File➤Open Library from the Omnis menu bar.

A library selection dialog box opens. You can hold down the ALT key while you select a library to avoid startup procedures.

9 Type a library name and click OK.
or
Select a library and click OK.

The PowerDesigner menu is activated in the Omnis menu bar.

10 Select PowerDesigner➤Generate Dictionary.

A file selection dialog box opens.

11 Select the script file you generated from your PDM and click OK.

A message box tells you if the generation is successful.

Reverse engineering a SQL script from Omnis

You can generate scripts from an Omnis 7 database and reverse engineer them into a PDM.

❖ To reverse engineer a SQL script file from Omnis:

1 Open Omnis.

2 Select File➤Open Library from the Omnis menu bar.

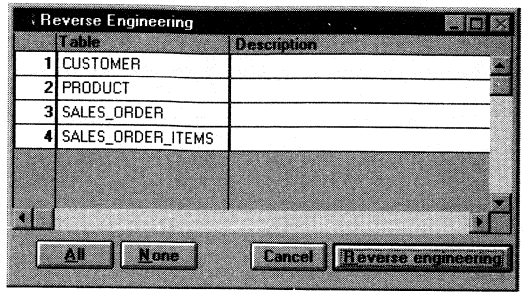
A library selection dialog box opens. You can hold down the ALT key while you select a library to avoid startup procedures.

3 Select a library that you want to reverse engineer.
Click OK.

The PowerDesigner menu is activated in the Omnis menu bar.

4 Select PowerDesigner➤Reverse Engineering from the Omnis menu bar.

The list of tables appears in a Reverse Engineering dialog box.



5 Select the tables you want to reverse engineer.
or
Select the All button.

6 Select the Reverse Engineering button.

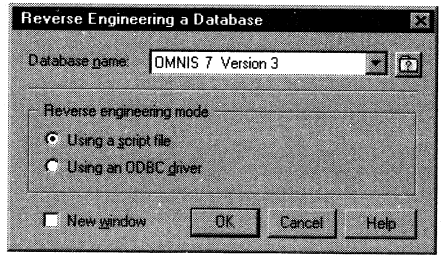
A library selection dialog box opens.

7 Type a filename for the script file to reverse engineer and click OK.

After the script is generated, a message box asks if you want to view the script.

8 Open the PDM to which you want to add the Omnis database tables.
Select File➤Reverse Engineering.

A reverse engineering dialog box opens.



9 Select Omnis 7 Version 3 from the Database Name dropdown listbox.

10 Select the Using a Script File radio button.
Click OK.

A file selection dialog box opens.

11 Select the script file you want to reverse engineer.
Click OK.

A message box tells you if the reverse engineering is succesful.

CHAPTER 12

Database Creation and Modification

About this chapter

This chapter explains how to generate and modify databases using scripts and via ODBC drivers.

Contents

Using the ODBC interface

The Open Database Connectivity (ODBC) interface allows PowerDesigner to access data in different database management systems (DBMS) using Structured Query Language (SQL) as a standard for accessing data.

An ODBC driver processes function calls and SQL requests coming from PowerDesigner and sends them to a data source. It also returns request results to PowerDesigner.

The data source provides all the target database information: its operating system, DBMS, and network platform. You can define more than one data source for the same driver.

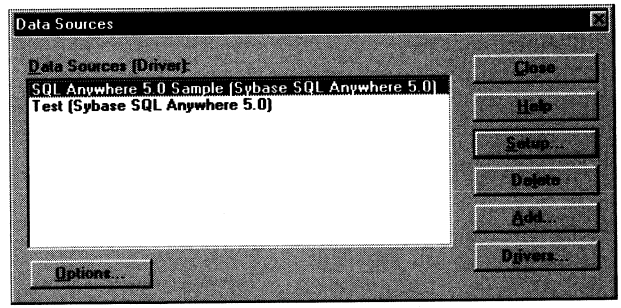
Defining a data source

You define data sources using the program ODBC Administrator. PowerDesigner gives you direct access to this program.

❖ To define a data source:

1 Select Database➤Configure Database.

ODBC Administrator dialog box asks you to identify a data source.



2 Click the Add button.

A list of available ODBC drivers appears.

3 Click a driver.

4 Click OK.

A dialog box displays driver-dependent options.

5 Type or select information required by the DBMS.

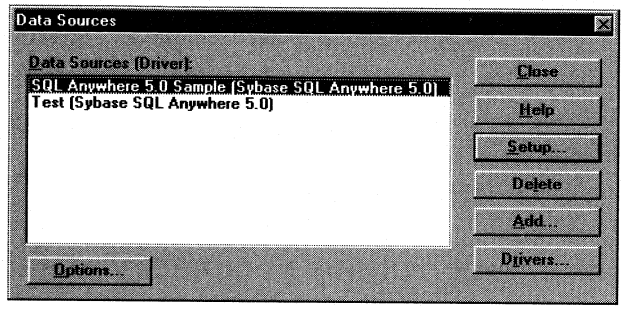
6 Click OK.

Configuring a data source

❖ **To configure a data source:**

1 Select Database➤Configure Database.

A dialog box asks you to identify a data source.



2 Select a data source from the dropdown listbox.

3 Click the Setup button.

A dialog box displays driver-dependent options.

4 Type or select information required by the DBMS.

5 Click OK.

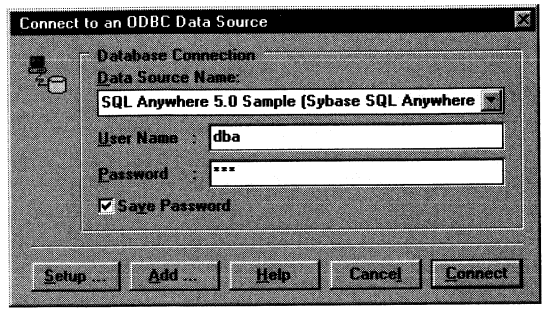
Connecting to a data source

By connecting to a data source, you can send direct requests to a DBMS.

❖ **To connect to a data source:**

1 Select Database➤Connect.

A dialog box asks you to identify a data source and connection parameters.



2 Select a data source from the dropdown listbox.

3 Type your user name and password.

4 Click Connect and, if prompted by your data source, enter additional connection parameters.

Displaying information about a connected database

The extent of information available about a database depends on the DBMS and the ODBC driver in use.

❖ **To display information about a connected database:**

1 Connect to a data source.

2 Select Database➤Database Information.

Disconnecting from a data source

❖ **To disconnect from a data source:**

♦ Select Database➤Disconnect.

Accessing a database

Changing target database

When you create a PDM, you select a target database. If you change the target database, the PDM is altered to become compatible with the new database.

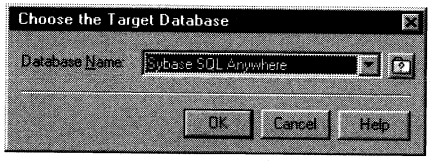
❖ To change the target database for a PDM:

1 Select Database➤Change Target Database.

A confirmation box asks you to confirm your choice.

2 Click Yes.

The database selection dialog box appears.



3 Select a target database from the dropdown listbox.

4 Click OK.

The name of the selected database appears in the status bar.

Displaying data from a database

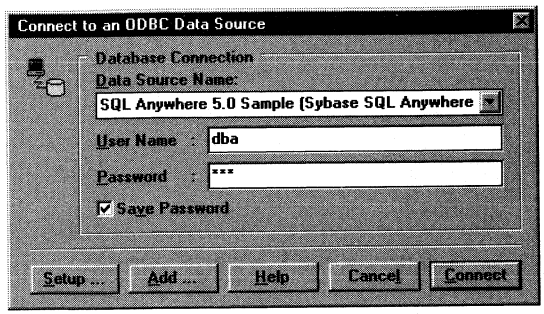
If the database corresponding to the PDM already exists, you can display the data that corresponds to a table, view, or reference in the PDM.

❖ To display data from a connected database:

1 Right-click a table, view, or reference.

2 Select View Data from the context menu.

A dialog box asks you to identify a data source and connection parameters.



3 Select a data source from the dropdown listbox.

4 Type your user name and password.

5 Click Connect and, if prompted by your data source, enter additional connection parameters.

A Query Results windows list all the database records corresponding to the selected table, view, or reference.

6 Click the Close button.

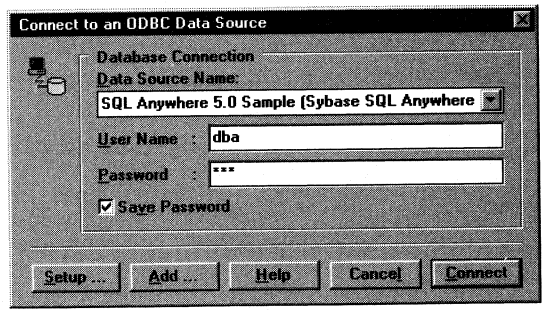
Executing SQL queries

You can send SQL queries to a connected data source and display the results.

❖ To execute SQL queries:

1 Select Database➤Execute SQL.

A dialog box asks you to identify a data source and connection parameters.

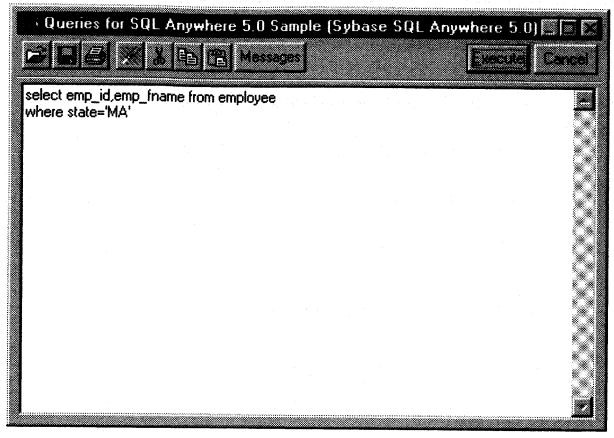


2 Select a data source from the dropdown listbox.

3 Type your user name and password.

4 Click Connect and, if prompted by your data source, enter additional connection parameters.

A Queries window appears.

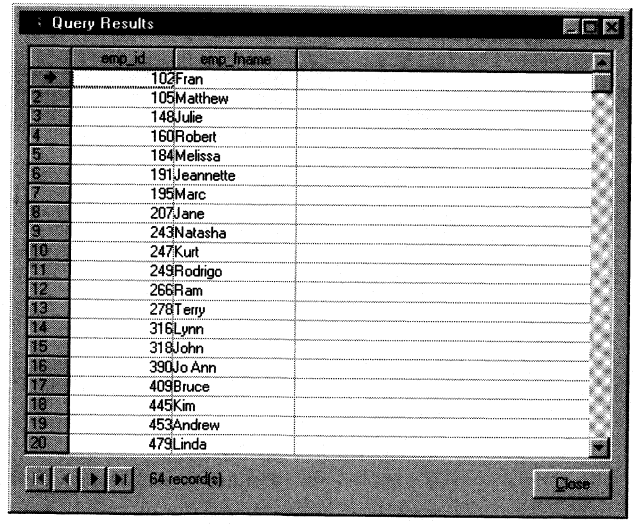


5 Type one or more SQL queries in the window.

The syntax of some SQL queries may be DBMS-dependent.

6 Click the Execute button.

A list displays the results of your queries.



7 Click Close.

You return to the queries window.

8 Click Cancel.

Computing database size

You can compute database size based on:

- Estimated number of records in tables
- Size of columns in tables
- Number of indexes

❖ To compute database size:

1 Select Dictionary➤List of Tables.

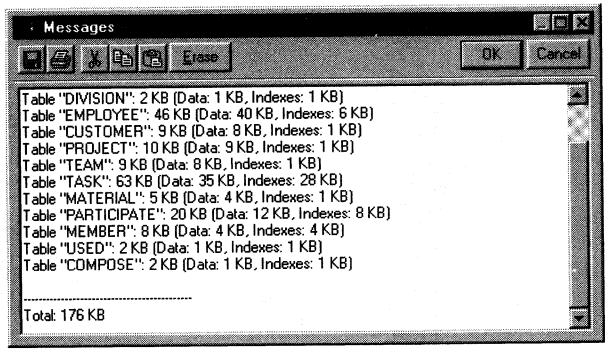
2 Click each table in the list.

3 Wherever the Number box is empty, type the estimated number of records in the Number box.

4 Click OK.

5 Select Database➤Compute Database Size.

A message window displays the size of each table, as well as the total database size.



Warning message
If you do not indicate the estimated number of records for all tables, you receive the following warning message: Warning: The number of records in table *table_name* is not defined.

6 Click OK.

Configuring tablespace and storage

Tablespace and storage indicate the physical location of database objects (tables and indexes). The definition of tablespace and storage is DBMS-specific. Different DBMS use different terms for tablespace and storage.

The terms tablespace and storage both refer to named partition that stores tables and indexes. Tablespace refers to a partition in a database. Storage refers to a partition on a storage device.

For some DBMS, a tablespace can use a declared storage in its definition.

Sample tablespace and storage commands

The following table lists commands that implement PowerDesigner tablespace and storage options in a selection of DBMS:

DBMS	Tablespace	Storage
Allbase	CREATE DBEFILESET	CREATE DBEFILE
DB2	CREATE TABLESPACE	CREATE STOGROUP
Oracle 5	CREATE SPACE	Not applicable
Oracle 6 or 7	CREATE TABLESPACE	Storage structure
RDB 4 to 6	CREATE STORAGE MAP	CREATE STORAGE AREA
SQLBase	Not applicable	CREATE STOGROUP
Sybase System 10 System 11 SQL Server	Not applicable	SP-ADDSEGMENT
SQL Anywhere Watcom 3 or 4	CREATE DBSPACE	Not applicable

Tablespace or storage not applicable
When tablespace or storage options are not applicable for a DBMS, you do not have access to the corresponding Database menu item.

Defining tablespace and storage

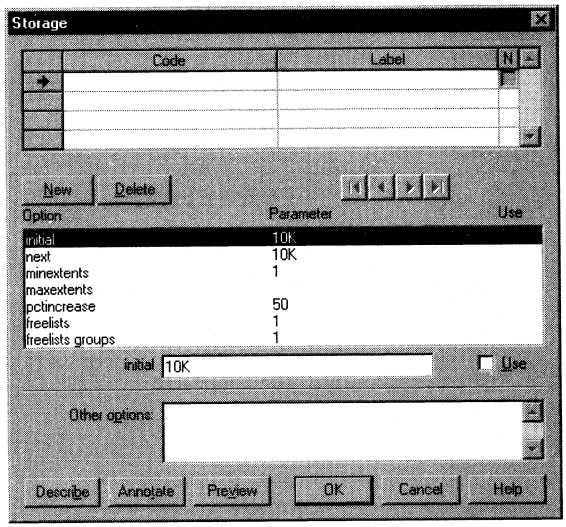
The lists of tablespace and storage options offers pre-defined parameters for each DBMS where applicable. The lists show default values and value lists for certain parameters which correspond to the recommended values for the DBMS.

For information on tablespace and storage options for a particular DBMS, see its reference manual.

❖ To define tablespace and storage:

1 Select Database➤Tablespace.
or
Select Database➤Storage.

The list of codes for tablespace or storage appears.



Menu item not available
When tablespace or storage options are not applicable for a DBMS, you do not have access to the corresponding Database menu item.

2 Click a blank line in the list.

An arrow appears at the beginning of the line.

3 Type a code in the Code column.

4 Select an option from the option list.

Below the list, the name of the option appears next to a text box which contains its default parameters.

5 Type parameter values in the text box.

6 If you want to include more options in the tablespace or storage definition, select other options and type their parameter values.

7 Click the Modify button.

8 Select the Use checkbox.

9 Click OK.

Previewing tablespace and storage commands

You can preview tablespace and storage commands before you generate them.

❖ To preview tablespace and storage commands:

1 Select Database➤Tablespace.
or
Select Database➤Storage.

The list of tablespace or storage options appears.

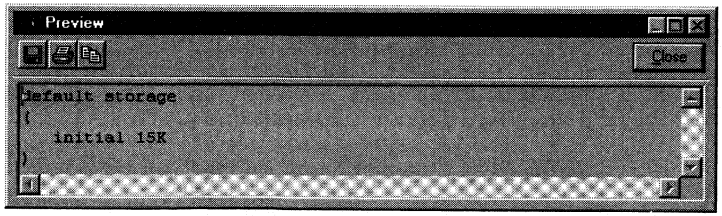
2 Click an option code in the list.

An arrow appears at the beginning of the line.

3 Select the Use checkbox.

4 Click the Preview button.

The preview window shows the corresponding command for the target DBMS.



5 Click Close.

6 Click OK.

Customizing scripts

You can customize scripts as follows:

- Insert scripts at the beginning and end of database creation script
- Insert scripts before and after a table creation command

Inserting begin and end scripts for database creation

In a database creation script, you can insert the following scripts:

Script	Where it is inserted
Begin script	Before the command that creates the database
End script	After the last command in the database creation script

You can use the following variables in these scripts:

Variable	Description
%DATABASE%	Name of the current PDM
%DATE%	Date of script generation
%DBMSNAME%	Name of the DBMS for the target database
%NAMESCRIPT%	Filename of script file
%PATHSCRIPT%	Filename and path of script file
%STARTCMD%	Command that runs the script
%USER%	Author of the current model

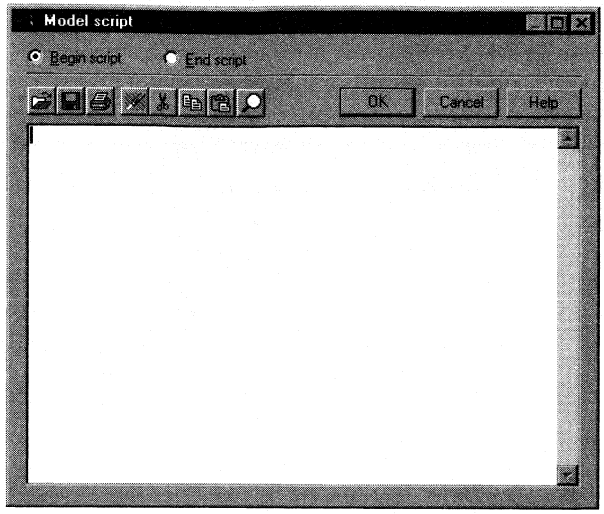
❖ To insert begin and end scripts for database creation:

1 Select Dictionary➢Model Definition.

The model property sheet appears.

2 Click the Script button.

The Model Script dialog box appears.



3 Type the DBMS-specific commands to insert in the beginning of the script.

4 Click the End script radio button.

5 Type the DBMS-specific commands to insert at the end of the script.

6 Click OK in each of the dialog boxes.

Inserting begin and end scripts for table creation

For each table, you have the option to insert the following scripts:

Script	Where it is inserted
Begin script	Immediately before the table creation command (after the table title)
End script	Immediately after the table creation command

These scripts can appear in database creation scripts and database modification scripts.

You can use the following variables in these scripts:

Variable	Description
%DATABASE%	Name of the current PDM
%DATE%	Date of script generation
%DBMSNAME%	Name of the DBMS for the target database
%NAMESCRIPT%	Filename of script file
%PATHSCRIPT%	Filename and path of script file
%STARTCMD%	Command that runs the script
%TABLE%	Name or code of current table (based on display preferences)
%TCODE%	Code of the current table
%TLABL%	Label of the current table
%TNAME%	Name of the current table
%USER%	Author of the current model

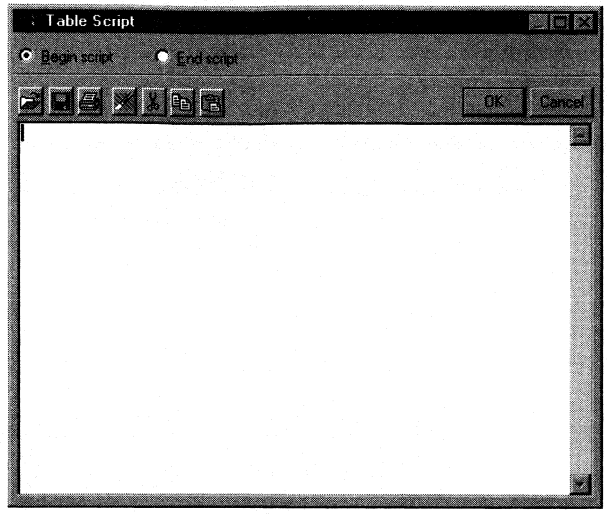
❖ To insert begin and end scripts for table creation:

1 Double-click a table symbol.

The table property sheet appears.

2 Click the Script button.

The Table Script dialog box appears. The Begin Script radio button is selected.



3 Type the script that you want to insert before the table creation command.

4 Click the End Script radio button.

5 Type the script that you want to insert after the table creation command.

> **Switching between scripts**
> By clicking the Begin Script and End Script radio buttons, you can move back and forth between scripts before validating them.

6 Click OK in each of the dialog boxes.

Editing a customized script

Using the script editor, you can insert the following categories of script language into a customized script:

Category	What it inserts in the script
Variable	Variable which is instanciated when the script is generated
Table	Table code
Column	Column code
Join	Parent table code, child table code, or join statement
Function	DBMS-specific function
Operator	Logical operator

Editing a begin or end script for database creation

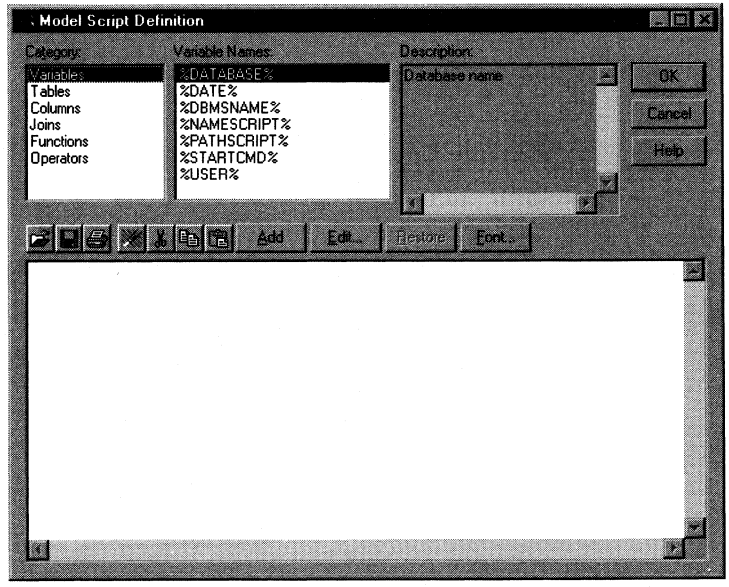
❖ To edit a begin or end script for database creation:

1 Select Dictionary➤Model Properties.

2 Click the Script button.

3 Click the button.

The Model Script Definition dialog box appears.



4 Click the position in the definition textbox where you want to insert the script language.

5 Select a category from the category list.

For example, select Variable to display the list of variables.

6 Select an item from the list for that category.

For example, select %DATABASE%.

7 Click the Add button.

The selected item is inserted at the cursor position.

8 Click OK in each of the dialog boxes.

Editing a begin or end script for table creation

❖ To edit a begin or end script for table creation:

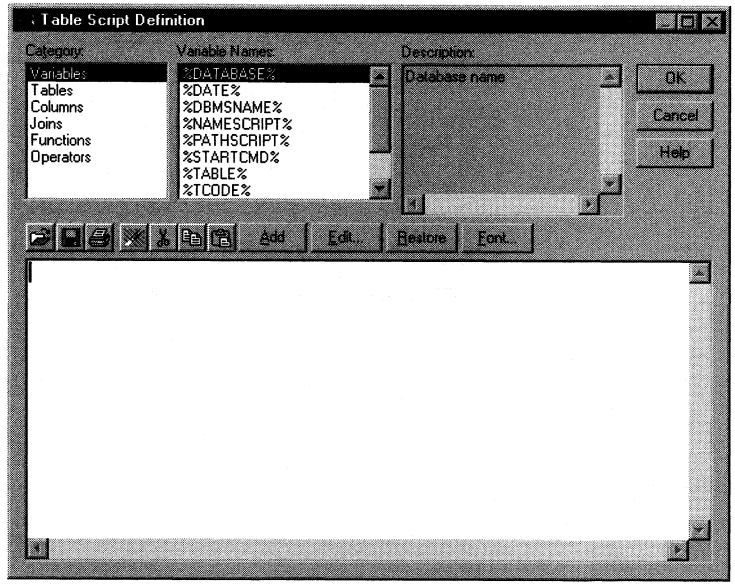
1 Double-click a table symbol.

The table property sheet appears.

2 Click the Script button.

3 Click the button.

The Table Script Definition dialog box appears.



4 Click the position in the definition textbox where you want to insert the script language.

5 Select a category from the category list.

For example, select Variable to display the list of variables.

6 Select an item from the list for that category.

For example, select %DATABASE%.

7 Click the Add button.

The selected item is inserted at the cursor position.

8 Click OK in each of the dialog boxes.

Formatting variables in customized scripts

Variables have a syntax that can force a format on their values, as follows:

- Force values to lower-case or upper-case characters
- Truncate the length of values

When a variable does not specify formatting, its values have the same format as in the PDM.

Format code	Format of variable value in script
.L	Lower-case characters
.T	Removes blank spaces
.U	Upper-case characters
.n	Maximum length where n is the number of characters
*.n*J	Justifies to fixed length where n is the number of characters

You embed formatting options in variable syntax as follows:

%.*format*:*variable*%

For example:

```
%.L:TABLE%
```

Generating a database

You can create or modify a database in different ways:

- Directly execute a script on a connected data source via an ODBC driver
- Generate a script to be executed on a DBMS at a later time

In both cases, the database generation commands are saved in a script file. You must always provide the following information about the script file:

Parameter	Description
File name	Destination filename for the script file
Directory	Destination directory for the script file
One file only	When selected creates one script file, instead of a separate script file for each table

Creation parameters for tables, indexes, views, and columns

Creation parameters indicate what to generate from the database structure defined by the PDM.

The availability of these parameters depends on the target database. Unavailable parameters display in gray and you cannot select them.

Table creation parameters

Parameter	Result of selection
All tables	Apply creation parameters to all tables
List	Apply creation parameters to selected tables only
Create table	Create table
Primary key	Generate primary key for table
Foreign key	Generate foreign key for table
Declarative integrity	Generate declarative referential integrity for table
Alternate key	Generate alternate key for table
Check	Generate check parameters and validation rules for table
Physical options	Generate physical options for table
Begin script	Insert customized script before table creation
End script	Insert customized script after table creation
Comment	Generate comment containing table label or name
Drop table	If table exists, drop table before creating new table

Index creation parameters

Parameter	Result of selection
Create index	Create index
Primary key	Generate primary key index
Foreign key	Generate foreign key index
Alternate key	Generate alternate key index
Other indexes	Generate indexes for all key columns with a defined index
Physical options	Generate physical options for index
Drop index	If index exists, drop index before creating new index

View creation parameters

Parameter	Result of selection
All views	Apply creation parameters to all views
List	Apply creation parameters to selected views only
Create view	Create view
Comment	Generate comment containing view label or name
Drop view	If view exists, drop view before creating new view

Column creation parameters

Parameter	Result of selection
User-defined type	Generate user-defined data types
Default value	Assign default value to column at creation
Check	Generate check parameters and validation rules for column
Comment	Generate comment indicating column label or name

Database creation parameters

Creation parameters indicate what to generate from the database structure defined by the PDM.

The availability of these parameters depends on the target database. Unavailable parameters display in gray and you cannot select them.

Database parameters

Parameter	Result of selection
Create database	Create database
Physical options	Generate physical options for database
Begin script	Insert customized script before database creation
End script	Insert customized script after database creation
Open database	Open database
Close database	Close database
Drop database	If database exists, drop database before creating new database

Tablespace and storage parameters

Parameter	Result of selection
Create tablespace	Create tablespace
Drop tablespace	If tablespace exists, drop tablespace database before creating new tablespace
Create storage	Create storage
Drop storage	If storage exists, drop storage before creating new storage

User-defined data type parameters

Parameter	Result of selection
Create data type	Create user-defined data type
Check	Generate check parameters and validation rules for user-define data type
Drop data type	If data type exists, drop data type before creating new data type

Database modification parameters

Modification parameters indicate what to generate based on modification in the database structure defined by the PDM.

The availability of these parameters depends on the target database. Unavailable parameters display in gray and you cannot select them.

Table modification parameters

Parameter	Result of selection
All tables	Apply modification parameters to all tables
List	Apply modification parameters to selected tables only
Modify table	Modify table
Primary key	Generate primary key for table
Foreign key	Generate foreign key for table
Declarative integrity	Generate declarative referential integrity for table
Alternate key	Generate alternate key for table
Table check	Generate check parameters and validation rules for table
Column check	Generate check parameters and validation rules for column
Default value	Assign default value to column at creation
User-defined type	Generate user-defined data types
Physical options	Generate physical options for table
Drop temporary tables	If table exists, drop table before creating new table
Add begin script	Insert customized script before table alter
Add end script	Insert customized script after table alter

Index modification parameters

Parameter	Result of selection
Modify index	Modify index
Primary key	Generate primary key index
Foreign key	Generate foreign key index
Alternate key	Generate alternate key index
Other indexes	Generate indexes for all key columns with a defined index
Physical options	Generate physical options for index

View modification parameters

Parameter	Result of selection
Modify view	Modify view

Trigger modification parameters

Parameter	Result of selection
Modify trigger	Modify trigger
For insert	Generate triggers for insert
For update	Generate triggers for update
For delete	Generate triggers for deletion

Script options and referential integrity options

Script options indicate the format of database generation scripts.

Referential integrity options indicate how to generate referential integrity, as a trigger or a declarative statement.

The availability of these options depends on the target database. Unavailable options display in gray and you cannot select them.

Script options

Option	Result of selection
Uppercase	Script contains all uppercase characters
Lowercase	Script contains all lowercase characters
No accent	Non-accented characters replace non-accented characters in script
Codes	Script refers to objects by their codes
Names	Script refers to objects by their names
Windows (ANSI)	Script uses ANSI character set
DOS (OEM)	Script uses DOS character set
Add titles	Each section of the script includes commentary in the form of titles (for example, Database Name: TUTORIAL)
ODBC syntax	Generates script in ODBC syntax instead of DBMS-specific syntax
Check model	Check the PDM before generating the database or script, and stop generation if an error is found

ODBC syntax
If you select the option ODBC Syntax, you generate triggers using the ODBC.DEF file for databases that have SQL support, or DBMS-specific DEF files plus DBC files for databases that do not have SQL support (for example, Access95). This option activates an ODBC syntax that low-level ODBC drivers can interpret more easily.

Referential integrity options

There are two ways to generate referential integrity in a database:

Option	Result of selection
Trigger	Generates a trigger
Declarative	Generates a declarative statement in a script

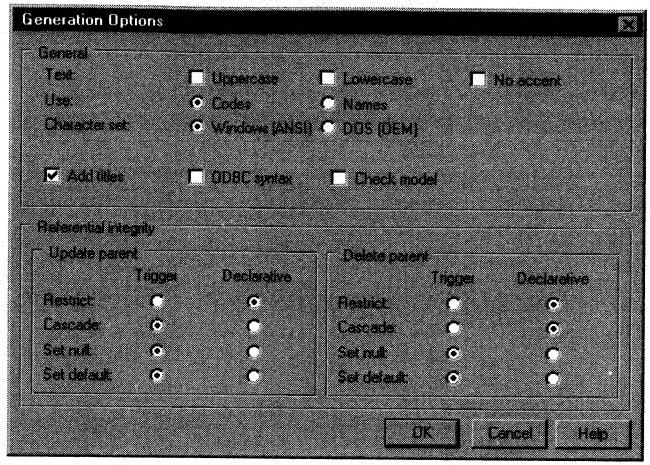
Defining script options and referential integrity options

You can define script options and referential integrity options. When you create or modify a database, these options are selected by default.

❖ To define script options and referential integrity options:

1 Select Database➤Generation Options.

The Generation Options dialog box appears.



2 Select script options and referential integrity options.

3 Click OK.

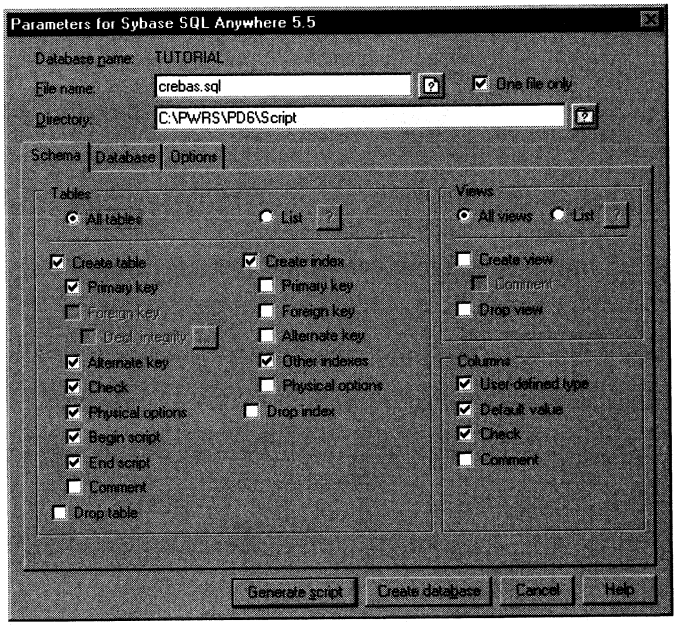
Generating a database creation script

PowerDesigner can generate a database creation script that you can run in your DBMS environment.

❖ To generate a database creation script:

1 Select Database➤Generate Database.

The Generation Parameters dialog box opens to the Schema page.



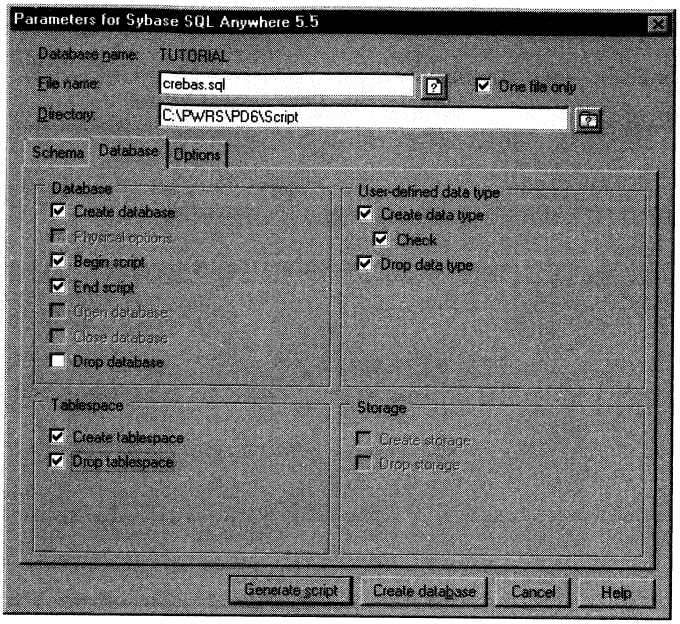
2 Type a destination filename for the script file in the File Name box.

3 Type a destination directory for the script file in the Directory box.

4 Select creation parameters for tables, indexes, views, and columns.

5 Click the Database tab.

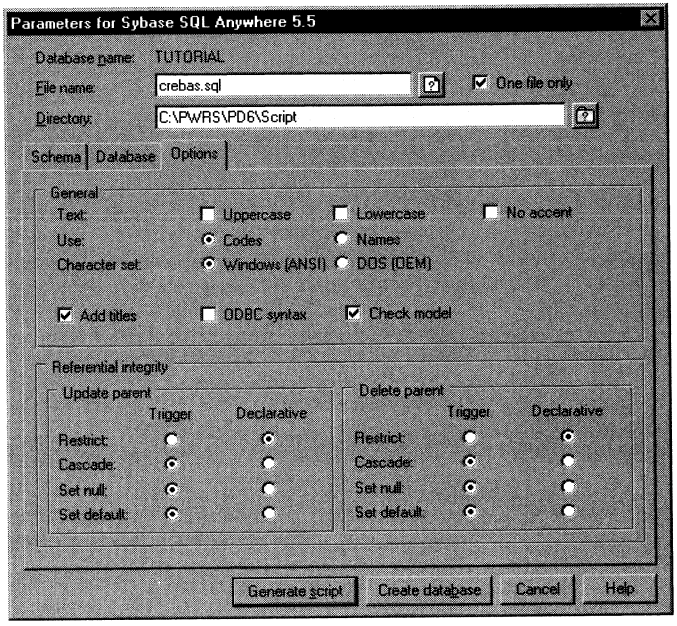
The Database page appears.



6 Select database creation parameters.

7 Click the Options tab.

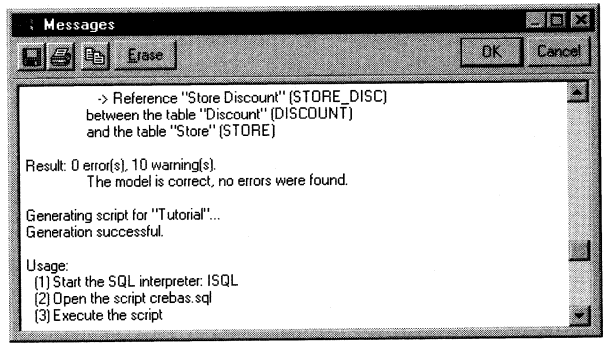
The Options page appears.



8 Select script options and referential integrity options.

9 Click the Generate script button.

A message window shows the progress of the generation process, and indicates the syntax for running the script.



A confirmation box asks you if you want to view the generated script.

10 Click No.

11 Click OK.

Creating a database directly

PowerDesigner can generate database structure directly. To do so, you must connect to a data source via an ODBC driver.

❖ To create a database directly:

1 Select Database➤Generate Database.

The Generation Parameters dialog box opens to the Schema page.

2 Type a destination filename for the script file in the File Name box.

3 Type a destination directory for the script file in the Directory box.

4 Select creation parameters for tables, indexes, views, and columns.

5 Click the Database tab.

The Database page appears.

6 Select database creation parameters.

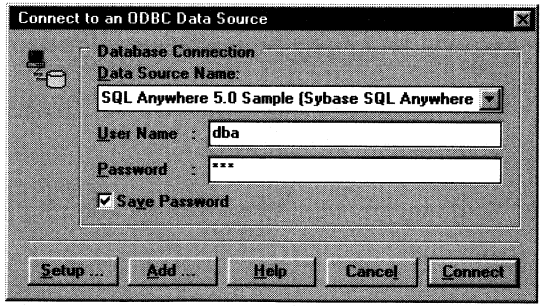
7 Click the Options tab.

The Options page appears.

8 Select script options and referential integrity options.

9 Click the Create Database button.

A dialog box asks you to identify a data source and connection parameters.

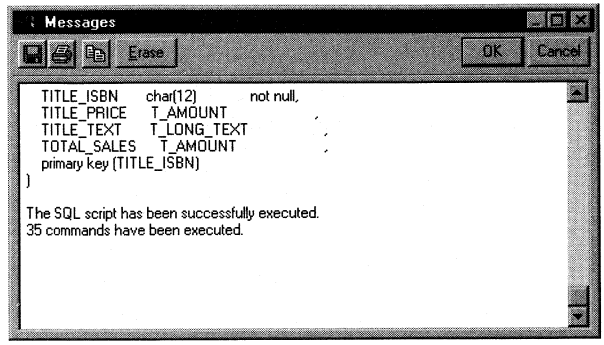


10 Select a data source from the dropdown listbox.

11 Type your user name and password.

12 Click Connect and, if prompted by your data source, enter additional connection parameters.

A message window shows the progress of the generation process.



13 Click OK.

Archiving a PDM

You can only modify a database if you have already archived its PDM.

❖ **To archive a PDM:**

1 Select Database➤Archive Model.

2 Enter a filename with the APM extension.

3 Click OK.

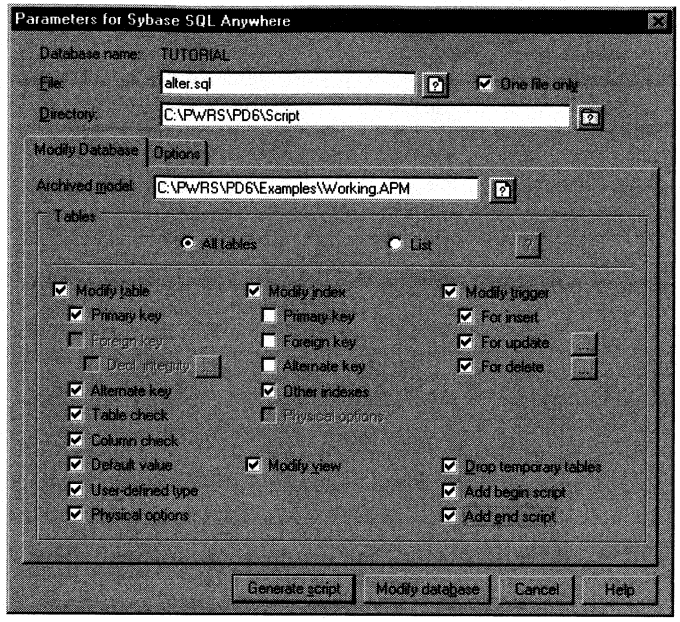
Generating a database modification script

PowerDesigner can generate a database modification script that you can run in your DBMS environment.

❖ **To generate a database modification script:**

1 Select Database➤Modify Database.

The Modification Parameters dialog box appears.



2 Type a destination filename for the script file in the File box.

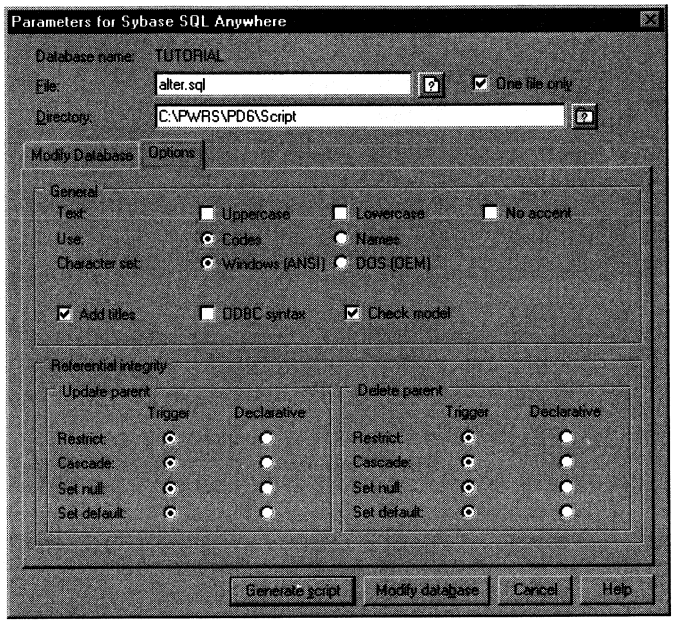
3 Type a destination directory for the script file in the Directory box.

4 Type an archive (APM) filename in the Archived Model box.

5 Select modification parameters.

6 Click the Options tab.

The Options page appears.



7 Select script options and referential integrity options.

8 Click the Generate Script button.

A message window shows the progress of the modification process, and indicates the syntax for running the script. A confirmation box asks you if you want to view the generated script.

9 Click No.

10 Click OK.

Modifying a database directly

PowerDesigner can modify database structure directly. To do so, you must connect to a data source via an ODBC driver.

❖ To modify a database directly:

1 Select Database➤Modify Database.

The Modification Parameters dialog box appears.

2 Type a destination filename for the script file in the File Name box.

3 Type a destination directory for the script file in the Directory box.

4 Type an archive (APM) filename in the Archived Model box.

5 Select modification parameters.

6 Click the Options tab.

The Options page appears.

7 Select script options and referential integrity options.

8 Click the Modify Database button.

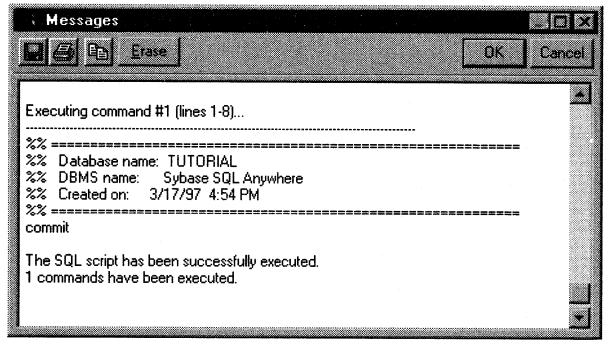
A dialog box asks you to identify a data source and connection parameters.

9 Select a data source from the dropdown listbox.

10 Type your user name and password.

11 Click Connect and, if prompted by your data source, enter additional connection parameters.

A message window shows the progress of the alter process.



12 Click OK.

PART THREE

Application Generators

This part describes the AppModeler application generators for PowerBuilder, Visual Basic, Power++, Delphi, and the Web.

CHAPTER 13

PowerBuilder Generator

About this chapter

This chapter provides an overview of the AppModeler PowerBuilder Generator (PBGen) and how it works.

Contents

Before you begin

In this chapter, the term **application** refers to the application that you generate using PBGen.

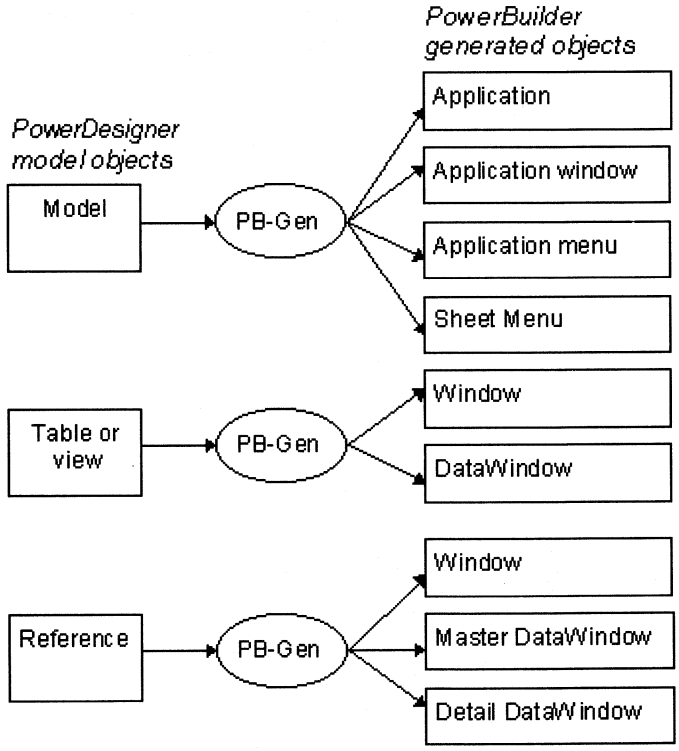
For information on the generation and reverse engineering of attributes from the catalog tables of the PowerBuilder repository, see the next chapter, "Using PowerBuilder Catalog Attributes."

Generator basics

PBGen facilitates the generation of PowerBuilder applications, based on objects and attributes defined in a PDM.

Object transformations

PBGen uses data from a PDM to instantiate predefined templates and generate applications, windows, user objects, and menus. The following schema shows the object transformations performed by PBGen:



MDI and SDI window types

AppModeler ships with application templates that generate Multiple Document Interface (MDI) applications.

In an MDI application, there are two types of windows: a frame or application window and sheet windows. PBGen instantiates the frame window with information from the model. It instantiates the sheet windows with information from tables, views, and references.

Single Document Interface (SDI) applications contain only one type of window: the main window. AppModeler does not ship with templates for SDI applications.

PBGen installed files

The AppModeler Setup program installs the following files for PBGen:

Filename	Default directory*	Description
POWERBLD.EXA	C:\PWRS\PD6\EXA	Extended attributes import file for PowerBuilder 4.0
POWERPFC.EXA	C:\PWRS\PD6\EXA	Extended attributes import file for PowerBuilder 5.0 and PFC libraries
PB4TEMPL.PBL	C:\PWRS\PD6\APPGEN \PB4TPL	Template library for PowerBuilder 4.0
PFC500TP.PBL	C:\PWRS\PD6\APPGEN \PB5TPL	Template library for PowerBuilder 5.0 and PFC libraries

* This table lists short directory names. During installation, you can select long filenames or alternative directory paths.

For more about the installation of AppModeler for PowerBuilder, see the *PowerDesigner Installation Guide*.

Naming conventions

While developing an application with PBGen, you have to specify names for objects, such as windows and menus. Follow the same naming conventions throughout your project.

Name prefixes

By default, PBGen uses the following object naming conventions:

- Use the first character to specify a prefix that identifies the type of object
- Follow the initial character with an underscore (_) and a character string that uniquely describes the object

The following table presents PBGen default naming conventions for PowerBuilder objects:

Type of object	Prefix	Example
Application	a_	a_demo
Window	w_	w_customer
Menu	m_	m_demo_sheet

Valid characters

A name must start with a letter. It can contain letters, numbers, and other valid PowerBuilder characters: underscore (_), dollar ($), and pound (#) signs. Do not use dashes. Do not use percentage signs, except in template names.

PowerBuilder does not differentiate uppercase and lowercase letters in names. All PowerBuilder names have a maximum length of 40 characters.

Building an application

Before you can generate an application using PBGen, you need to do the following:

- Define model extended attributes
- Select PowerBuilder libraries
- Select application templates
- Indicate locations and the database profile

> **PowerBuilder ORCA library**
> You must install the ORCA library file that ships with PowerBuilder and list this path in your environment path settings. If you use PowerBuilder 4.0, you must install PBORC040.DLL. If you use PowerBuilder 5.0, you must install PBORC050.DLL.

Defining model extended attributes

Model extended attributes associate a PDM with the following application and menu properties:

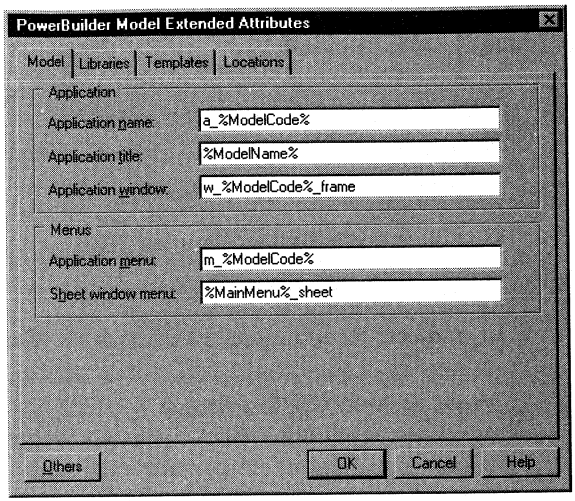
Property	Description
Application name	Name of the application
Application title	Title that appears in the title bar of the application
Application window	Name of frame (MDI) or main (SDI) window
Application window menu	Name of the menu for the frame or main window
Sheet window menu	Name of the menu for the sheet windows

The default values for these properties contain variables that PBGen instantiates with your model code or model name.

❖ To define model extended attributes:

1 Select Client➤PowerBuilder Model Attributes.

The PowerBuilder Model Extended Attributes dialog box appears.



2 Type changes to application and menu properties, as needed.

3 Click OK.

Selecting PowerBuilder libraries

You attach source and target PowerBuilder libraries to the application you are building. PowerBuilder libraries are stored in PBL files.

Selecting PowerBuilder source libraries

Source libraries include the template library you use for your application and all ancestor libraries of the template library. Ancestor libraries contain objects or functions that the template library inherits.

Library	Provides
Templates library	Application design information
Library list	Additional and required design information that the templates library inherits

You can use the template libraries that ship with PBGen. If you use the PB4TEMPL.PBL template library, you must first install the PowerBuilder Application Library. If you use the PFC500TP.PBL library, you must first install the PFC Library.

If you are using one of the template libraries that ship with AppModeler, you need to select additional libraries as follows:

Templates library	Inherits from	Library List
PB4TEMPL.PBL	PowerBuilder 4.0 Application Library	SYS.PBL UTLWIN.PBL UTLFUNC.PBL
PFC500TP.PBL	PowerBuilder 5.0 PFC Library	PFCAPSRV.PBL PFCDWSRV.PBL PFCMAIN.PBL PFCWNSRV.PBL PFEAPSRV.PBL PFEDWSRV.PBL PFEMAIN.PBL PFEWNSRV.PBL

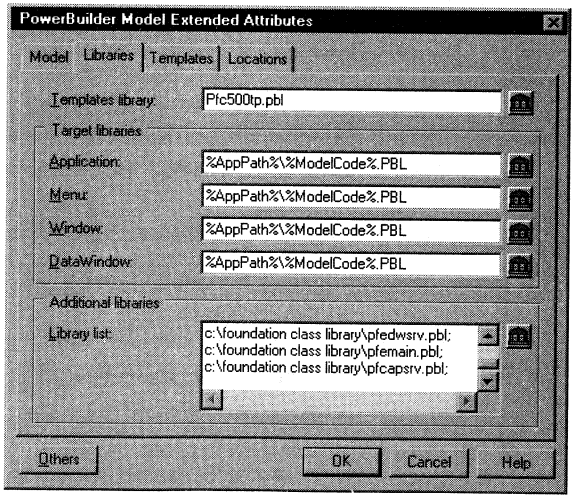
❖ To select PowerBuilder source libraries:

1 Select Client➤PowerBuilder Model Attributes.

The Model Extended Attributes dialog box opens to the Model page.

2 Click the Libraries tab.

The Libraries page displays a selection of libraries.



3 Click the button next to the Templates Library box.
Select a template library and click OK.
or
Type the path and filename of the template library you want to use for your application.

4 Click the button next to the Library List listbox.
Select all ancestor libraries for your template library and click OK.

Verify that all the libraries you need are included in the Library List listbox.

5 Click OK.

Selecting PowerBuilder target libraries

Target libraries are PBL files that receive and store the application information you define with PBGen. If the target libraries do not already exist, PBGen creates the corresponding PBL files the first time you generate your application.

You select a target library for the following PowerBuilder objects:

- Application
- Menu
- Window
- DataWindow

Library path

You can use the same PBL file for all target libraries. If you do not indicate a path, PBGen generates objects to target libraries in the PWRS\PD6 path. If you use the %AppPath% variable, PBGen generates objects to target libraries in the application directory that you set on the Locations page of the PowerBuilder Model Extended Attributes dialog box.

If you modify a path or filename in one of the target library boxes of the PowerBuilder Model Extended Attributes dialog box, PBGen automatically adds the modified information to any blank target library boxes. It adds this information after you exit the dialog box, or after you place the focus in another zone of the dialog box.

❖ **To select PowerBuilder target libraries:**

1 Select Client➤PowerBuilder Model Attributes.

The Model Extended Attributes dialog box opens to the Model page.

2 Click the Libraries tab.

The Libraries page displays a selection of libraries.

3 Click a target library button to display a list of PBL files. Select a PBL file and click OK.
or
Type the path and PBL filename for each target library.

4 Click OK.

Viewing the name of a target library

Certain PBGen dialog boxes contain fields that represent your target library names in the form of a variable:

Variable	Represents value
%WindowLibrary%	Target library for window definitions
%DatawindowLibrary%	Target library for DataWindow definitions

All of these fields are followed by a button that gives you access to the instantiated name of the library that PBGen will use as an application target.

❖ **To view the name of a target library:**

1 Click the library button next to the library textbox containing the variable name.

A standard Windows file selection dialog box appears. The instantiated name of the target library is selected by default.

2 Click Cancel.

Selecting application templates

A template is text that indicates the structure and appearance of a PowerBuilder object. Separate templates provide design information for applications, windows, menus, and user objects.

You select the following templates at the application level:

Template	What it defines
Application template	Global application structure
Application window	Main or frame window
Application menu	Menu bar and items in the frame (MDI) or main (SDI) window
Sheet window menu	Menu bar and items when sheet windows are selected

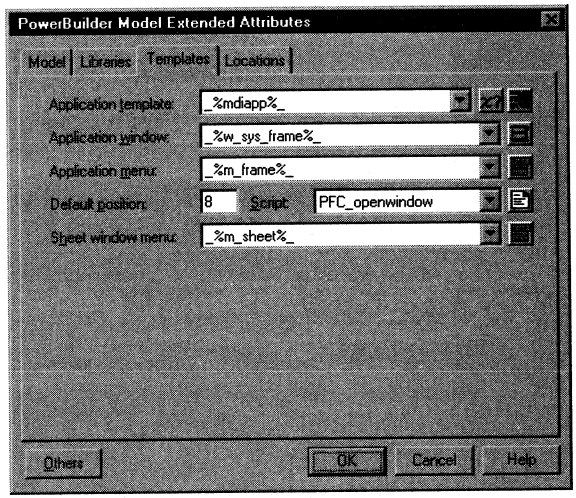
❖ To select application templates:

1 Select Client➤PowerBuilder Model Attributes.

The Model Extended Attributes dialog box opens to the Model page.

2 Click the Templates tab.

The Templates page opens.



3 Click the button to display a list of application templates. Select a template and click OK.

or

Select an application template from the dropdown listbox.

The application template in the PBGen template libraries is named _%mdiapp%_.

Empty list of templates
The templates library determines the list of available templates. You must select a templates library before selecting individual templates, otherwise the list of available templates is empty.

For more information on selecting a templates library, see "Selecting PowerBuilder source libraries" on page 268.

4 Click the button to display a list of window templates.
Select a template and click OK.
or
Select an application window template from the dropdown listbox.

The application window template in the PBGen template libraries is named _%w_sys_frame%_. This is a frame window template for MDI applications.

5 Click the button next to the Application Menu dropdown listbox.
Select a template and click OK.
or
Select an application menu template from the dropdown listbox.

6 Click the button next to the Sheet Window Menu dropdown listbox.
Select a template and click OK.
or
Select a sheet menu template from the dropdown listbox.

7 Click OK.

Menu properties on Templates page
You can select certain menu position and script properties from the Templates page.

For information on these properties, see "Defining menus" on page 278 and "Adding a window menu script" on page 279.

Assigning a value to a user-defined variable

From PowerBuilder, you can declare instance, shared, or global variables in your application template. To generate values for these user-defined variables with PBGen, you must name them using the _% prefix and the %_ suffix. Variable names must not exceed 30 characters.

❖ To assign a value to a user-defined variable from PBGen:

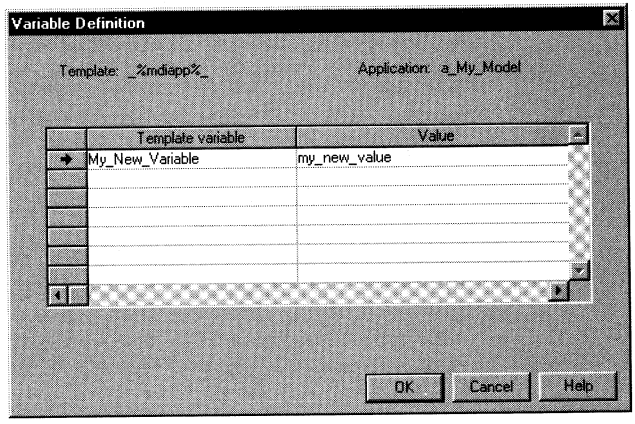
1 Select Client➤PowerBuilder Model Attributes.

The Model Extended Attributes dialog box appears.

2 Click the Templates tab.

3 Click the button next to the Application Template textbox.

The Variable Definition dialog box displays the list of user-defined variables in the application template. Prefixes and suffixes surrounding the variable names are not displayed.



4 Click a user-defined variable in the list.

An arrow appears at the beginning of the line.

5 Type a value in the Value column.

6 Click OK.

You return to the Model Extended Attributes dialog box.

7 Click OK.

Indicating locations and the database profile

To generate an application, you must indicate the following locations:

Location	Description
PowerBuilder path	Name of the directory in which PowerBuilder is installed
Database profile	Name of the database profile that identifies a data source and connection parameters
Application directory	Name of the directory where you generate an INI file and a LOG file
INI file	Name of the INI file for the resulting application
Profile section	Section in the INI file in which to copy the database profile

In order for PBGen to save the PowerBuilder objects that you generate, it must connect to a database. The database profile identifies a data source and connection parameters so that you can connect automatically.

Database profile information

The PB.INI file defines the database profiles you can select from the Database Profile dropdown listbox.

PBGen copies the selected database profile to the INI file that you name in the INI File box. PBGen can create a new INI file to store this information. By default, PBGen generates an INI file that uses the model code in its filename. It generates the INI file to the path you enter in the Application Directory box.

You can type a relative path in the Application Directory box. PBGen will then look for the application directory in the PowerDesigner 6 path. If the directory name you type does not exist in this path, PBGen will create it.

PowerBuilder 4.0

If you generate for PowerBuilder 4.0 using PB4TEMPL.PBL as your template source library, the INI file section to which you copy database profile information must be named Database1. This is the profile section name default of the POWERBLD.EXA file.

PowerBuilder 5.0

If you generate for PowerBuilder 5.0 using the PFC500TP.PBL template source library, the name of the database profile section of the new INI file should include the name of the profile you select from the PB.INI file. The POWERPFC.EXA default for this section name is Profile %db_profile%.

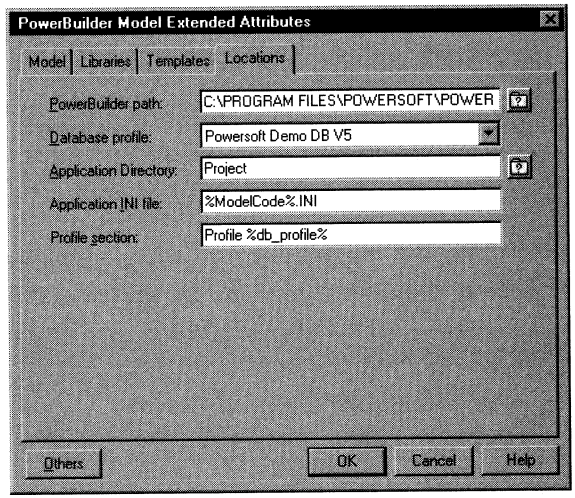
❖ To indicate locations and the database profile:

1 Select Client➤PowerBuilder Model Attributes.

The Model Extended Attributes dialog box appears open to the Model page.

2 Click the Locations tab.

The Locations page displays default file locations.



3 Click the button next to the PowerBuilder Path textbox.
Select a directory and click OK
or
Type the full pathname of your PowerBuilder directory.

4 Select a profile from the Database Profile dropdown listbox.

5 Click the button next to the Application Directory.
Select a directory and click OK
or
Type the name of the directory where you want to generate an application INI file and a generation log file.

6 Type changes to the Application INI file, if necessary.

PBGen asks for confirmation before it generates a file with the same name as an existing file in your application directory.

7 Type changes to the Profile Section box, if necessary.

The profile section should be Database1 if you are using templates that inherit from the PowerBuilder Application Libraries. It should be Profile %db_profile% if you are using templates that inherit from the PowerBuilder PFC Libraries.

8 Click OK.

Defining menus

A PowerBuilder menu template defines a menu design.

Menu properties

PBGen menu properties apply to all menus in an application, as follows:

Property	Description
Name (Application window menu)	Name for the definition of the menu bar and its items in the frame window or the main window
Template (Application window menu)	Template that defines the menu bar and its items in the frame window or the main window
Name (Sheet window menu)	Name for the definition of the menu bar and its items in the frame window when a sheet window is selected
Template (Sheet window menu)	Template that defines the menu bar and its items in the frame window when a sheet window is selected
Library	Path and filename of menu target library
Script	PowerScript code that executes when you select a generated menu item from the application menu bar
Position	Numbered position of the window access menu in the menu bar. By default, this menu contains an item to access each window in the application
Mode	Menu generation mode: Generate or Update

Generation mode

You can modify menu properties, except for the generation mode, from the PowerBuilder Model Extended Attributes dialog box. You can only change the generation mode from the Application Generation tree view.

From the tree view, you can choose the following menu generation modes:

Mode	Description
Generate	Generates a new menu if it does not already exist in the application
Update	Updates a menu if it already exists in the application

For information on defining menu properties from the PowerBuilder Model Extended Attributes dialog box, see the section "Building an application" on page 267. For information on modifying menu properties from the tree view, see "Modifying menu properties" on page 304.

Menu position

PBGen can generate menu items that link to application windows. By default, these menu items appear in a single menu identified by its position in the menu bar.

For example, in the template library PB4TEMPL.PBL, the template _%m_frame%_ defines five menu positions: File, Functions, Actions, Window, and Help menus. If you assign the window access menu to position 1, the File menu contains items that open application windows.

> **Menu position not visible**
> Certain menu positions may only be visible when a sheet window is selected. For example, in the PB4TEMPL.PBL menu templates, the Actions menu and the Window menu (positions 3 and 4) are only visible when a sheet window is selected.

You can define the position of the window access menu from the Model Extended Attributes dialog box or from the Menu Properties dialog box.

For information on defining menu properties from the PowerBuilder Model Extended Attributes dialog box, see the section "Building an application" on page 267. For information on modifying menu properties from the tree view, see "Modifying menu properties" on page 304.

Adding a window menu script

A script defines the action that occurs when you select a window menu item in your generated application.

The AppModeler Setup program installs two openwindow scripts. If you are using the PB4TEMPL.PBL as your template library, you must use the openwindow script that POWERBLD.EXA sets as the default:

```
ParentWindow.TriggerEvent("openwindow", 0, "%WindowName%")
```

If you are using PFC500TP.PBL as your template, you must use the PFC_openwindow script that POWERPFC.EXA sets as the default:

```
Message.StringParm = "%WindowName%" ; of_SendMessage("openwindow")
```

These scripts call an openwindow user event that is defined in the menu templates.

You can define a script that performs other actions from the window menu item event in your generated application. PBGen can save your script in the registry or PD6.INI file.

❖ To add a window menu script:

1 Select Client➤PowerBuilder Model Attributes.

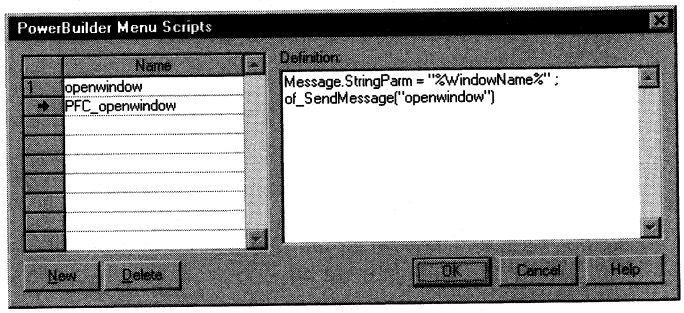
The Model Extended Attributes dialog box opens to the Model page.

2 Click the Templates tab.

The Templates page opens.

3 Click the button next to the script field.

The Menu Scripts dialog box appears.



4 Click a blank line in the Name column.

An arrow appears at the beginning of the line.

5 Type a script name in the Name column.

6 Use PowerScript syntax to type a script in the Description textbox.

7 Click OK.

You return to the Templates page.

8 Select the new script from the Script dropdown listbox.

9 Click OK.

Defining windows

You define windows by associating window attributes with tables, views, or references. Each resulting window displays information coming from one table, one view, or one reference.

Defining window attributes

Window attributes indicate how to generate a window based on a table, view, or reference. These attributes apply to sheet windows in an MDI application and main windows in an SDI application.

Attribute	Description
Generate	When selected, selects the window for generation
Name	Window name
Template	Window template
Target PB library	PBL file where you generate window definition
Title	Text to display in the window title bar
Menu option	Indicates whether or not a window corresponds to a menu item, and indicates the text of the menu item

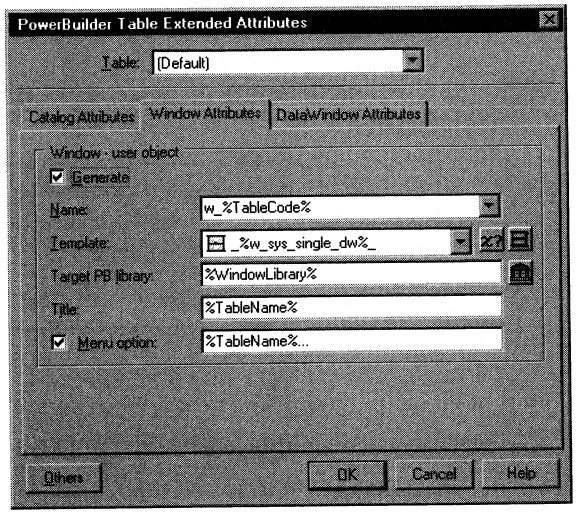
You can define window attributes that apply by default to all tables, all views, or all references. You can also define window attributes for a specific table, view, or reference.

❖ To define window attributes:

1. Select Client➤PowerBuilder Table Attributes.
   or
   Select Client➤PowerBuilder View Attributes.
   or
   Select Client➤PowerBuilder Reference Attributes.

   A PowerBuilder Extended Attributes dialog box appears.
2. Click the Window Attributes tab.
3. Select an object from the Table, View, or Reference dropdown listbox.
   or
   Select Default as the object selection.

If you select Default, the new default values apply to all tables, all views, or all references for which you have not previously modified the same attributes. This includes all new tables, new views, or new references that you add to the PDM.



4 Type changes to the window name, if needed.

5 Click the button to display a list of window or templates.
Select a template and click OK
or
Select a window template from the Template dropdown listbox.

6 Click the button to display a list of PBL files.
Select a PBL file and click OK.
or
Type changes to the Target PB Library box, if needed.

7 Type changes to the window title, if needed.

8 Select the Menu option checkbox if you want a menu item to access this window.

The checkbox is selected by default.

9 Type changes to the menu option, if needed.

10 Click OK.

Selecting a user object as a window template

In PowerBuilder, you can define a user object in a template library. You can then select this user object as a window template in PBGen.

The template libraries PB4TEMPL.PBL and PFC500TP.PBL do not contain user objects.

PBGEN naming convention for user objects
PBGen only recognizes user objects with names that respect the syntax:

_%object_name%_

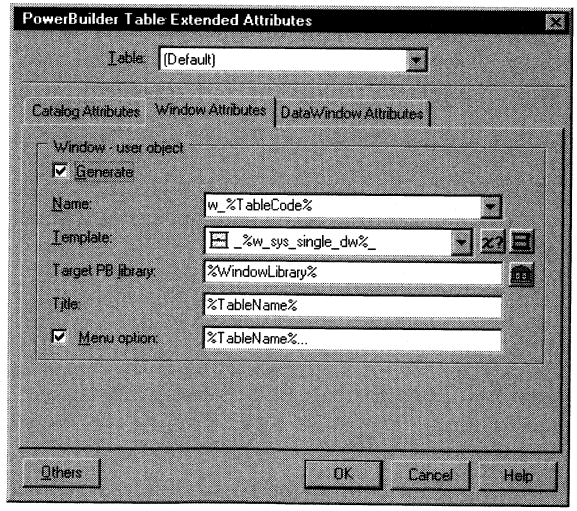
❖ To select a user object as a window template:

1 Select Client➤PowerBuilder Table Attributes.
or
Select Client➤PowerBuilder View Attributes.
or
Select Client➤PowerBuilder Reference Attributes.

A PowerBuilder Extended Attributes dialog box appears.

2 Click the Window Attributes tab.

The Window Attributes page opens.



3 Click the button to display a list of window templates.
Select a user object and click OK.
or
Select a user object from the Template dropdown listbox.

4 Click OK.

Assigning a value to a user-defined variable in a window template

From PowerBuilder, you can declare instance, shared, or global variables in your window templates. To generate values for these user-defined variables with PBGen, you must name them using the _% prefix and the %_ suffix. Variable names must not exceed 30 characters.

❖ **To assign a value to a user-defined variable in a window template:**

1 Select Client➤PowerBuilder Table Attributes.
or
Select Client➤PowerBuilder View Attributes.
or
Select Client➤PowerBuilder Reference Attributes.

A PowerBuilder Extended Attributes dialog box appears.

2 Click the Window Attributes tab.

3 Select Default from the object selection dropdown listbox.
or
Select a table, view, or reference from the *Object* dropdown listbox.

4 Click the button next to the Template textbox.

The Variable Definition dialog box displays the list of user-defined variables in the window template without the surrounding prefixes and suffixes.

5 Click a user-defined variable in the list.

An arrow appears at the beginning of the line.

6 Type a value in the Value column.

7 Click OK.

You return to the PowerBuilder Extended Attributes dialog box.

8 Click OK.

Defining DataWindow objects

A DataWindow displays information from one table or one view. You define a DataWindow by attaching extended attributes to a table or a view. In most cases, PBGen uses the same table or view to define a window and its DataWindow.

A standalone DataWindow is not attached to a particular window. For information on defining a standalone DataWindow, see "Defining a standalone DataWindow" on page 288.

A Master/Detail DataWindow displays information linked by a reference in the PDM. For information on defining Master/Detail DataWindow objects, see "Defining master DataWindow and detail DataWindow attributes" on page 289.

Defining DataWindow attributes

DataWindow attributes indicate how PBGen will generate a DataWindow based on a table or a view.

Attribute	Description
Generate	When selected, selects the DataWindow for generation
Name	Name of the DataWindow
Updatable	When selected, allows modification of information that the DataWindow displays
Style	Display format for the DataWindow
Target PB library	PBL file for DataWindow definitions

Updating a DataWindow with a view data source
If the DataWindow displays information from a view that is not updatable, then you cannot update the information in the DataWindow, even if you select Updatable as a DataWindow attribute.

You can define DataWindow attributes that apply by default to all tables or all views. You can also define DataWindow attributes that apply to a specific table or view.

❖ To define DataWindow attributes:

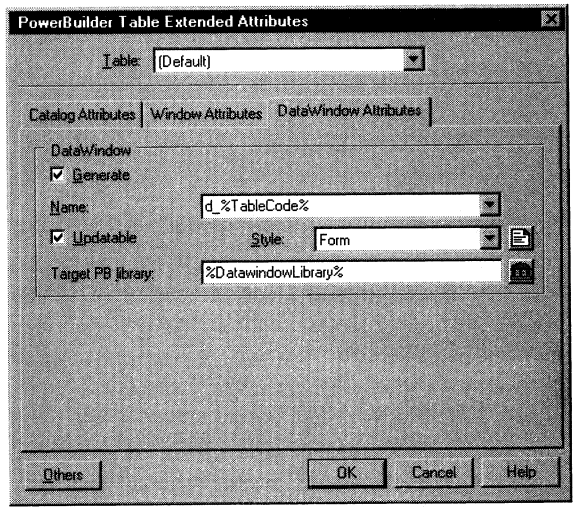
1 Select Client➤PowerBuilder Table Attributes.
or
Select Client➤PowerBuilder View Attributes.

A PowerBuilder Extended Attributes dialog box appears.

2 Click the DataWindow Attributes tab.

3 Select a table or view from the Table or View dropdown listbox.
or
Select Default as the object selection.

If you select Default, the new default values apply to all tables, or all views, for which you have not previously modified the same attributes. This includes all new tables or new views that you add to the PDM.



4 Type changes to the DataWindow name, if needed.

5 Select the Updatable checkbox to allow modification of data displayed by the DataWindow.
or
Clear the Updatable checkbox to protect data displayed by the DataWindow.

6 Select a DataWindow style from the dropdown listbox.

7 Click the button to display a list of PBL files.
Select a PBL file and click OK.
or
Type changes in the Target PB Library box, if needed.

8 Click OK.

Applying a new style to a DataWindow

A DataWindow style indicates how a DataWindow presents data. The AppModeler Setup program installs standard form, grid, label, and tabular DataWindow styles, as well as 3D versions of the form, grid, and tabular styles.

Form3D style syntax

Columns in the 3D DataWindow styles that ship with AppModeler have a lowered 3D border. The syntax for the Form3D style in PBGen is:

```
Style(Type=Form) DataWindow(Color=12632256)
Text(Background.Color=12632256)
Column(Background.Color=12632256 Border=5)
```

You can create DataWindow style syntax in the DWSyntax utility that ships with PowerBuilder. You can copy this syntax to the style definition dialog box in PBGen to create new styles for DataWindows that you generate.

When you create a DataWindow style for a table or a view, it is added to the registry or PD6.INI file. It is automatically available for all other tables or views. In addition, it is available for master and detail DataWindow objects.

For information on applying styles to Master/Detail DataWindow objects, see "Applying a new style to a master DataWindow or detail DataWindow" on page 293.

❖ **To apply a new DataWindow style:**

1 Select Client➤PowerBuilder Table Attributes.
or
Select Client➤PowerBuilder View Attributes.

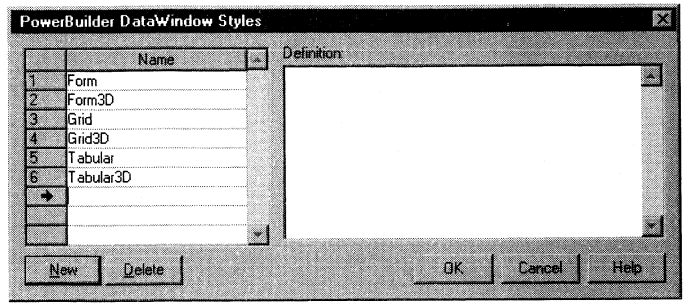
A PowerBuilder Extended Attributes dialog box appears.

2 Click the DataWindow Attributes tab.

The DataWindow Attributes page opens.

3 Click the button next to the Style listbox.

This opens the PowerBuilder DataWindow Styles dialog box.



4 Click the New button.

An arrow appears at the beginning of a blank line.

5 Type a name in the Name column.

6 Type a style definition in the Definition textbox.

7 Click OK.

8 Select the new style name from the Style dropdown listbox.

9 Click OK.

Defining a standalone DataWindow

A standalone DataWindow is not attached to a window template, so you can reuse a standalone DataWindow in several different windows. For example, you can define a DropDownDataWindow for customer names, and reuse it in Customer and Sales Order windows.

You use PowerBuilder to attach standalone DataWindows to windows.

❖ To define a standalone DataWindow attributes:

1 Right click a table or view in the PDM.

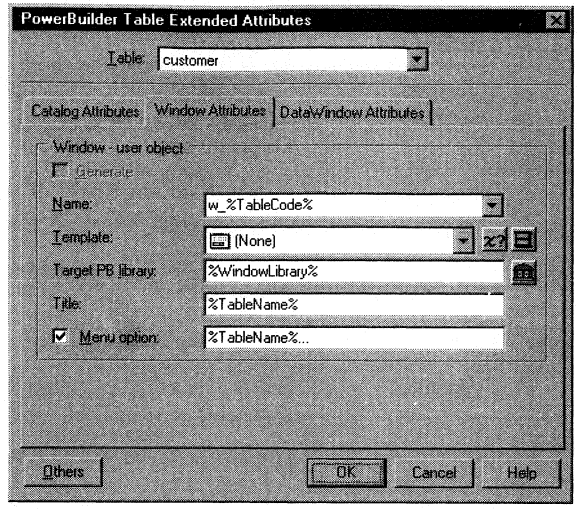
Its context menu appears.

2 Select Extended Attributes.

The PowerBuilder Extended Attributes dialog box appears.

3 Click the Window Attributes tab.

The Windows Attributes page opens.



4 Select None from the Template dropdown listbox.

5 Click OK.

Defining master DataWindow and detail DataWindow attributes

You use a master DataWindow and a detail DataWindow to display information linked by a reference in the PDM.

Master attributes indicate how to generate the master DataWindow based on a reference. Detail attributes indicate how to generate the detail DataWindow based on a reference.

Attribute	Description
Datasource (master)	Source of information to display in the master DataWindow: either the parent table or a view containing the primary key of the parent table
Datasource (detail)	Source of information to display in the detail DataWindow: either the child table or a view containing the foreign key of the child table
Generate	When selected, selects the DataWindow for generation
Name	Name of the DataWindow
Updatable	When selected, allows modification of information that the DataWindow displays
Style	Display format for the DataWindow
Target PB library	PBL file for DataWindow definitions

Updating a DataWindow with a view as its data source
If the master DataWindow or detail DataWindow displays information from a view that is not updatable, you cannot update the information in the DataWindow, even if you select Updatable as a DataWindow attribute.

Defining master attributes

You can define attributes that apply by default to all master DataWindow objects generated from references. If you modify default master attributes, the new default values apply to all references for which you have not previously modified the same attributes. This includes all new references.

You can also define master attributes that apply to a DataWindow generated from a specific reference.

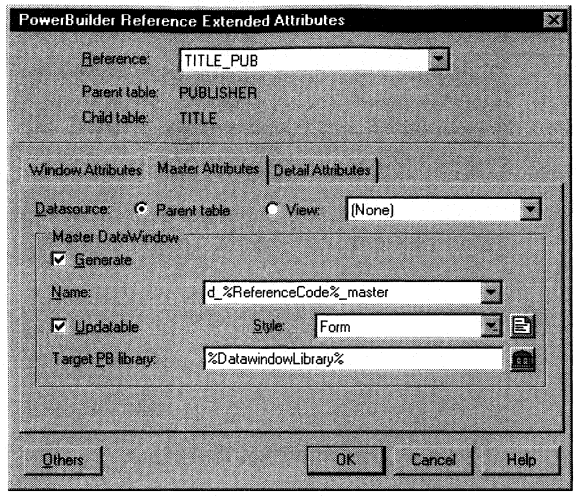
❖ **To define master attributes:**

1 Select Client➤PowerBuilder Reference Attributes.

A PowerBuilder Extended Attributes dialog box appears.

2 Click the Master Attributes tab.

The Master Attributes page displays master attributes.



3 Select Default from the Reference dropdown listbox to modify default master attributes.
or
Select a reference from the Reference dropdown listbox to modify master attributes for a specific reference.

4 Select the Parent Table radio button.
or
Select a view from the View data source dropdown listbox.

This listbox only shows views that contain the primary key of the parent table. You cannot select a view if you selected Default in the Reference dropdown listbox.

5 Type changes to the master DataWindow name, if needed.

6 Select the Updatable checkbox to allow modification of data displayed by the master DataWindow.
or
Clear the Updatable checkbox to protect data displayed by the master DataWindow.

7 Select a DataWindow style from the dropdown listbox.

8 Click the button to display a list of PBL files.
Select a PBL file and click OK.
or
Type changes to the Target PB Library box, if needed.

9 Click OK.

Defining detail attributes

You can define detail attributes that apply by default to all detail DataWindow objects generated from references. If you modify default detail attributes, the new default values apply to all references for which you have not previously modified the same attributes. This includes all new references.

You can also define detail attributes that apply to a DataWindow generated from a specific reference.

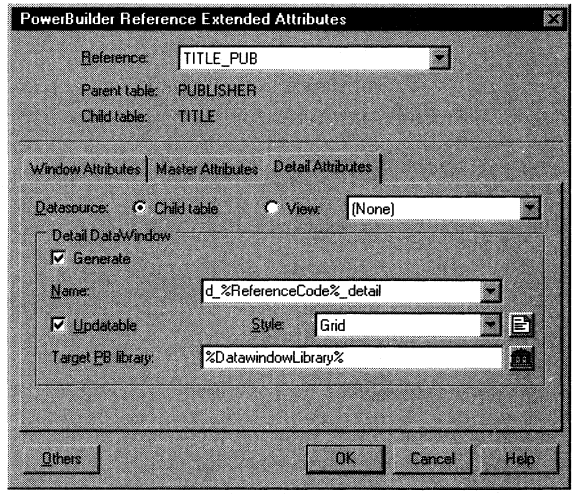
❖ To define detail attributes:

1 Select Client➤PowerBuilder Reference Attributes.

A PowerBuilder Extended Attributes dialog box appears.

2 Click the Detail Attributes tab.

The Detail Attributes page displays default attributes.



3 Select Default from the Reference dropdown listbox to modify default detail attributes.
or
Select a reference from the Reference dropdown listbox to modify detail attributes for a specific reference.

4 Select the Child Table radio button.
or
Select a view from the dropdown listbox.

This listbox only shows views that contain the foreign key of the child table. You cannot select a view if you selected Default in the Reference dropdown listbox.

5 Type changes to the detail DataWindow name, if needed.

6 Select the Updatable checkbox to allow modification of data displayed by the detail DataWindow.
or
Clear the Updatable checkbox to protect data displayed by the detail DataWindow.

7 Select a DataWindow style from the dropdown listbox.

8 Click the button to display a list of PBL files.
Select a PBL file and click OK.
or
Type changes to the Target PB Library box, if needed.

9 Click OK.

Applying a new style to a master DataWindow or detail DataWindow

A DataWindow style indicates how a master DataWindow or detail DataWindow presents data. The EXA default styles for master and detail DataWindows are:

Generated object	Default style	What it does
Master DataWindow	Form	Arranges data in labeled columns
Detail DataWindow	Grid	Arranges data in a table

When you create a DataWindow style for a reference, it is automatically available for all other references, tables, and views.

❖ **To apply a new style to a master DataWindow or detail DataWindow:**

1 Select Client➤PowerBuilder Reference Attributes.

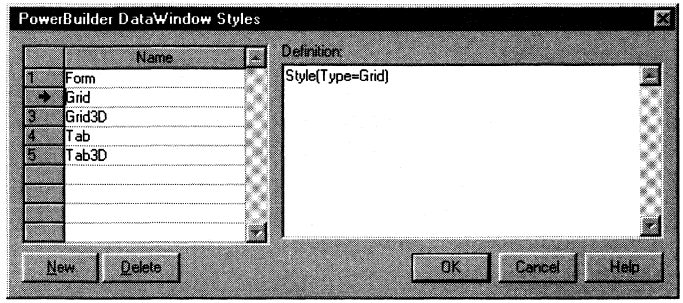
The PowerBuilder Reference Extended Attributes dialog box appears.

2 Click the Master Attributes tab.
or
Click the Detail Attributes tab.

The DataWindow Attributes page opens.

3 Click the button next to the Style listbox.

This opens the PowerBuilder DataWindow Styles dialog box.



4 Click the New button.

An arrow appears at the beginning of a blank line.

5 Type a name in the Name column.

6 Type a style definition in the Definition textbox.

You can use the DWSyntax utility that ships with PowerBuilder to help create DataWindow style definitions.

7 Click OK.

8 Select the new DataWindow style from the Style dropdown listbox.

9 Click OK.

Selecting objects to generate

The Application Generation window is the launch point for PowerBuilder object generation. It displays a tree view containing the following objects in a hierarchical list:

- Application
- Application window
- Application window menu and sheet window menu
- Windows or user objects for selected tables, views, and references
  or
  Windows or user objects corresponding to filter definitions that you set in the previous PBGen session
- DataWindows for selected tables, views, and references
- Columns of each DataWindow

From the objects listed in the tree view, you select the objects that you want to generate in your PBGen session.

> **Your first PBGen session**
> If you do not select any PDM objects and you did not use a selection filter in a previous PBGen session, items are listed in the tree view for every PDM object.

Expanding and collapsing nodes

The items in the tree view are organized hierarchically under nodes. A **node** is an item that contains dependent items at a lower level.

Node types

The main node represents the application you will generate. Dependent items of the main application node can include menus, windows, user objects, and standalone DataWindow objects. Window nodes contain their dependent DataWindow objects. DataWindow nodes contain their dependent DataWindow columns.

Node status

You can expand a node to show its dependent items, or collapse a node to hide its dependent items.

A plus sign to the left of a node indicates that the node is collapsed and contains hidden items. To select dependent items of this node, you must first expand the node.

A minus sign to the left of a node indicates that the node is expanded. You can select individual dependent items under this node.

❖ **To expand or collapse a node:**

- Click the plus or minus sign to the left of the node.

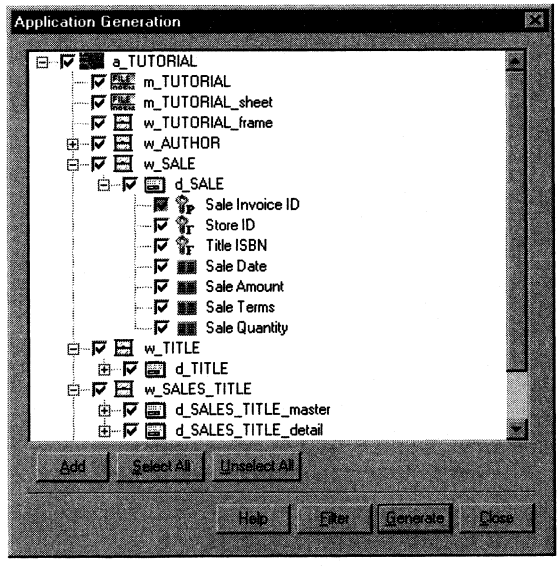
Generating an application from selected objects

To create a list of objects to generate, you can select objects directly from the PDM, or you can reuse objects you selected in the last PBGen session.

❖ **To generate a project from selected objects:**

1. Use the mouse or selection tools to select objects in the PDM.
   or
   Verify that no objects are selected in the PDM graph if you want to reuse objects selected during the previous PBGen session.
2. Select Client➢Generate PowerBuilder Application.

   The Application Generation window appears.



3. Click any plus signs to the left of tree view items.

   You expand the tree view so that all items are visible.
4. Verify that all objects you want to generate are checked.

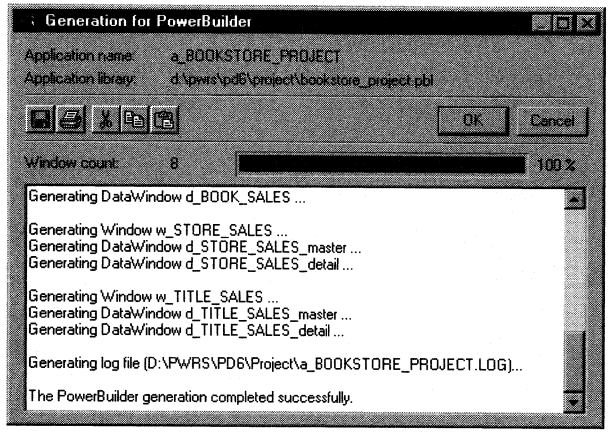
5 Click the Generate button.

PBGen asks for confirmation before it generates an application or object to a PBL with the same name as an existing file in the application directory.

Reviewing the generation log

Generation progress

You can follow the progress of the generation from the log window that appears after the generation begins.



Detecting errors

PBGen indicates if there are any errors in the generation or if the generation is successful. In case of a generation error, another window displays a compilation error message and the script that causes the error. You can save the script and the error message to a text file.

It is possible for additional errors to appear when you run the application. PBGen cannot detect errors due to incorrect or incomplete template variable instantiation. If you are using a user-defined template, be sure to assign compatible values to user-defined variables. The default values in templates that ship with AppModeler should guarantee correct behavior at runtime.

For information on assigning values to user-defined variables, see the chapter "Building an Application Structure."

Log filename

The application log file, *MODELCODE*.LOG, is created automatically in your application directory. It contains model and generation information. You can read the log file with a text editor. After you close the Generation log window, you can still display the generation log from PBGen.

❖ **To review the generation log from PBGen:**

- Select Dictionary➤Display Messages.

  A Messages window displays the log of the previous generation.

Modifying the generation selection

There are several ways to modify a preliminary object selection from the Application Generation window. To modify the object selection, you can:

- Set the generation flag for objects in the tree view
- Add objects to the tree view
- Apply a filter

Setting the object generation flag

A checkmark to the left of an item in the tree view indicates that its generation flag is selected.

Selecting columns in the tree view

Although you can select a checkbox of a column without selecting the checkbox of the DataWindow containing the column, you must generate the DataWindow in order to generate the column.

By clicking the Select All button beneath the tree view, you select the generation flags of all items in the tree view, except for columns. By clicking Unselect All, you clear the generation flags of all items in the tree view, except for columns. You can only clear the checkboxes of columns one by one.

You cannot clear the checkbox of a primary key column and you cannot clear the checkbox of a foreign key column in a DataWindow based on a non-reflexive reference. The tree view does not display checkboxes for columns in a view, or for columns in a reference that use a view as their data source. When you generate their DataWindow, these columns are always generated.

Items sharing a data source

Two references can share the same data source (parent table or child table). Changes to the column generation flags for the master DataWindow of one of the references are automatically reflected in the columns of the master DataWindow of the other reference. If detail DataWindows share the same child table data source, their column generation flags are also linked, except where the column is a foreign key of the master data source for one of the references, but not for the other.

❖ To set the generation flags for objects in the tree view:

1 Click the plus signs in front of tree view items.

You expand the tree view.

2 Select checkboxes to the left of the objects you want to generate.

3 Clear checkboxes to the left of objects you do not want to generate.

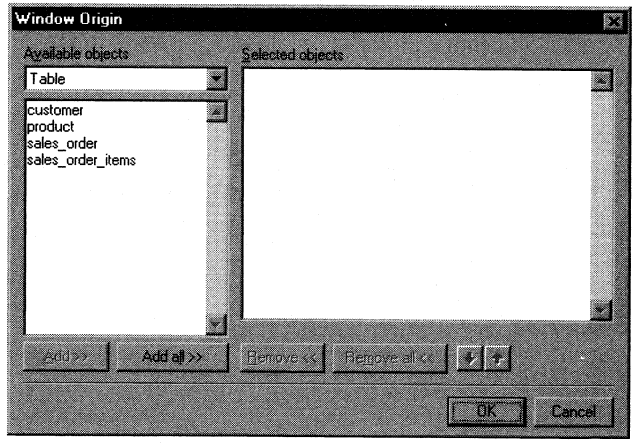
Adding objects to the tree view

You can choose objects to add as items in the tree view. If you add objects to the tree view without using a filter, and if you close the Application Generation window without generating a PowerBuilder application, PBGen does not save your modified selections.

❖ To add objects to the tree view:

1 Click the Add button below the tree view.

The Window Origin dialog box appears.



2 Select a PDM object type from the Available Objects dropdown listbox.

The Available Objects listbox displays all objects of the type selected that are not in the tree view.

3 Select objects in the Available Objects listbox and click Add.
or
Double-click an object you want to add to the tree view.

The objects you selected move to the Selected Objects listbox. If you added an object by mistake, you can select it in the Selected Objects listbox and click Remove.

4 Repeat steps 2 and 3 for other PDM object types.

5 Click OK.

You return to the Application Generation window. The objects you added to the Selected Objects listbox appear as items in the tree view.

Applying a filter to modify object selection

Filters help handle large amounts of information. A large PDM can include hundreds of objects. You can use a filter to restrict the items included in the tree view in the Application Generation window.

After you apply a filter, only the matching items appear in the tree view. The selections you make with a filter are saved by PBGen, whether or not you generate a PowerBuilder application in the current session.

Types of filters

There are four types of filters:

Filter	Action
All objects	Selects every object in the PDM
Objects of submodel	Selects objects from a named submodel
Modified objects	Compares objects in the current PDM to objects in an archived PDM and selects objects that have changed
User-defined list	Lets you select a list of objects from the PDM

You can select or clear object type checkboxes with any of the filter types.

User-defined filter

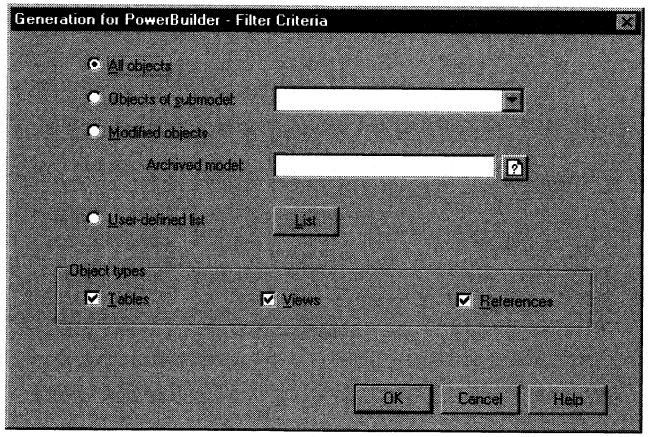
If you use the user-defined filter, you can add any object in the PDM to the tree view. You can use the buttons in the Form Origin dialog box to make customized filter selections:

Button	Function
Add	Transfers a selected object from the Available Objects listbox to the Selected Objects listbox and adds it to the tree view
Add All	Transfers all objects of the selected object type
Remove	Transfers a selected object from the Selected Objects listbox to the Available Objects listbox and removes it from the tree view
Remove All	Transfers all objects of the selected object type
	Moves the selected object up in the list order of Selected Objects and in the tree view
	Moves the selected object down in the list order of Selected Objects and in the tree view

❖ To select objects using user-defined filter criteria:

1 Click the Filter button in the Application Generation window.

The Filter Criteria dialog box appears.



2 Click the  button beside the User-Defined List option.

The Window Origin dialog box appears.

3 Select a PDM object type from the Available Objects dropdown listbox.

The Available Objects listbox displays all objects of the type selected that are not in the tree view. The Selected Objects listbox displays all objects that are in the tree view.

4 Use the buttons of the Window Origin dialog box to add, remove, or reposition objects in the listboxes and in the tree view generation list.

5 Click OK.

You return to the Filter Criteria dialog box.

6 Verify that checkboxes are selected for the object types that you want to appear in the tree view.

7 Click OK.

You return to the Application Generation window. The tree view displays the new selections you made with the user-defined filter.

Fine-tuning before and after generation

Before you generate or regenerate your application, you can fine tune and test it from the tree view of the Application Generation window as follows:

- Modify object properties
- Select columns to generate
- Open the generated application

Modifying object properties

You can modify properties for all objects in the tree view.

Modifying application properties

You can display or modify the following application properties from the tree view:

Property	Description
Name	Name of the application
Title	Title that serves as the application header
Template	Name of the application template
Library	Path and filename of the application target library
Database profile	Name that identifies a data source and connection parameters

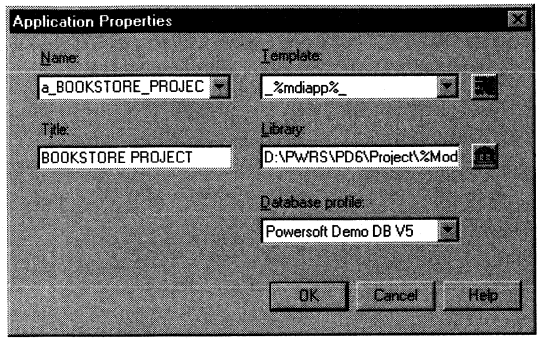
❖ **To modify application properties from the tree view:**

1 Right-click the application item in the tree view.

The item context menu appears.

2 Select Properties from the context menu.

The Application Properties dialog box appears.



3 Type changes to the application properties as needed.

4 Click OK.

Modifying menu properties

You can display or modify the following menu properties from the tree view:

Property	Description
Name (Application window menu)	Name for the definition of the menu bar and its items in the frame window or the main window
Template (Application window menu)	Template that defines the menu bar and its items in the frame window or the main window
Name (Sheet window menu)	Name for the definition of the menu bar and its items in the frame window when a sheet window is selected
Template (Sheet window menu)	Template that defines the menu bar and its items in the frame window when a sheet window is selected
Library	Path and filename of menu target library
Position	Numbered position of the window access menu in the menu bar. By default, this menu contains an item to access each window in the application
Mode	Menu generation mode: Generate or Update

For more information on menu properties, see "Defining menus" on page 278.

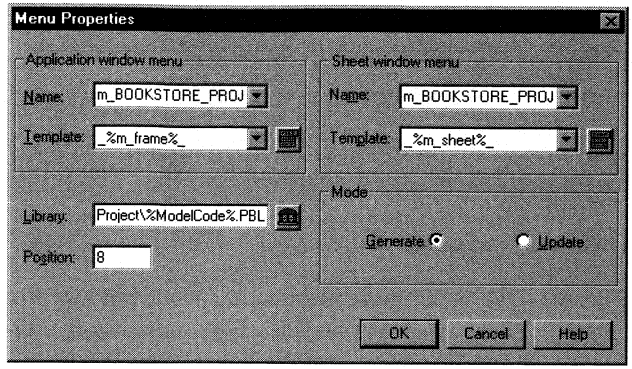
❖ To modify menu properties from the tree view:

1 Right-click a menu item in the tree view.

The item context menu appears.

2 Select Properties from the context menu.

The Menu Properties dialog box appears.



3 Type changes to the menu properties as needed.

4 Click OK.

Modifying application window properties

You can display or modify application window properties from the tree view:

Property	Description
Name	Name of the application window
Template	Name of the application window template

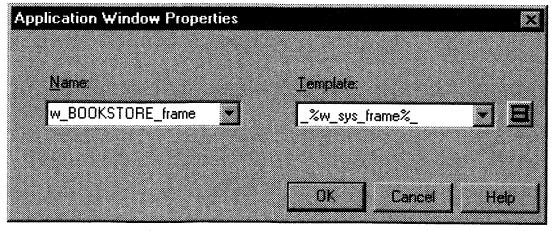
❖ To modify application window properties from the tree view:

1 Right-click the application window item in the tree view.

The item context menu appears.

2 Select Properties from the context menu.

The Application Window Properties dialog box appears.



3 Type changes to the application properties as needed.

4 Click OK.

Modifying window or DataWindow properties

You can access PowerBuilder Extended Attributes dialog boxes from the context menus of window or DataWindow items in the tree view. The dialog box that you open depends on the window origin of the tree view item.

You can then modify window properties and DataWindow properties for the selected tree view item.

For more information about window properties, see "Defining windows" on page 281. For more information about DataWindow properties, see "Defining DataWindow objects" on page 285.

❖ To modify window or DataWindow properties from the tree view:

1 Right-click a window or DataWindow item in the tree view.

The item context menu appears.

2 Select Properties from the context menu.

The PowerBuilder Extended Attributes dialog box opens to the Window Attributes or DataWindow Attributes page.

3 Type changes to the window or DataWindow properties as needed.

4 Click OK.

Modifying column properties

You can modify the catalog attributes of a column from the tree view.

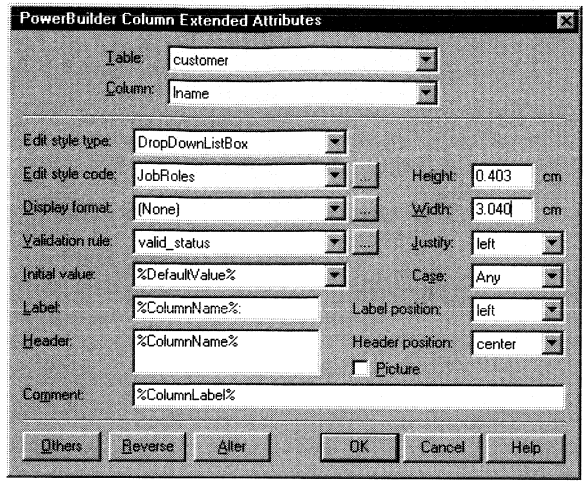
For information on PowerBuilder catalog attributes, see the chapter "Using PowerBuilder Catalog Attributes."

1 Right-click a , , or column in the tree view.

The item context menu appears.

2 Select Column Properties from the context menu.

The PowerBuilder Column Extended Attributes dialog box appears.



3 Type changes to the column extended attributes as needed.

4 Click OK.

Selecting columns to generate

PBGen lists the columns to be generated when you expand all the items displayed in the tree view.

From the tree view, you can select which columns of a table you want to generate in a DataWindow. You can also select which columns you want to generate in a master or detail DataWindow of a reference if it uses a table for its data source. However, if you do not generate a DataWindow, you cannot generate columns of that DataWindow.

For more information on selecting individual columns for generation, see "Setting the object generation flag" on page 298.

Opening the generated application

You can open the PowerBuilder application painter from the Application Generation window. From the PowerBuilder Application painter, you can test the last generated version of the application.

To open the painter, your application must already exist in the PowerBuilder data source. If you have not already generated the application, PowerBuilder displays an error message.

❖ **To open the generated application from PBGen:**

1 Select Client➤Generate PowerBuilder Application.

2 Right-click the  application item.

A context menu appears.

3 Select Run Application from the context menu.

The PowerBuilder Application painter opens the application you generated.

Using PBGen templates

PBGen ships with two complete and independent template libraries. The templates in PB4TEMPL.PBL work in conjunction with the PowerBuilder 4.0 Application Library. The templates in PFC500TP.PBL use the PowerBuilder 5.0 Foundation Class Library.

PBGen template libraries contain the following template objects:

Template name	Description
%mdiapp%	Application template
_%w_sys_frame%_	MDI frame window template
_%w_sys_multi_dw%_	Template for a window with a multiple row DataWindow. For use with tables or views
_%w_sys_single_dw%_	Template for a window with a single row DataWindow. For use with tables or views
_%w_sys_shared_dw%_	Template for a window with a Master/Detail DataWindow. For use with reflexive references*
_%w_sys_mast_detail_dw%_	Template for a window with a Master/Detail DataWindow. For use with references
%dw_main%_	Template for a DataWindow
_%m_frame%_	Frame menu template
_%m_sheet%_	Sheet menu template that inherits from _%m_frame%_

* PB4TEMPL.PBL only. If you use PFC500TP.PBL, select _%w_sys_mast_detail_dw%_ for a reflexive reference.

Using menu templates

The PBGen template libraries contain separate templates for the frame window menu and the sheet window menus.

PB4TEMPL.PBL menu templates

The menu templates defined in the PB4TEMPL.PBL library have five menu positions. In applications generated from these templates, File, Function, and Help menus are always visible. Actions and Window menus are not always visible.

Menu position	Menu name	Visible in frame menu?	Visible in sheet menu?
1	File	✓	✓
2	Function	✓	✓
3	Actions	—	✓ in windows based on _%w_sys_shared_dw%_ and _%w_sys_mast_detail_dw%_
4	Window	—	✓
5	Help	✓	✓

In windows that contain a master DataWindow and detail DataWindow, the Action menu allows you to insert and delete master and detail items.

PFC500TP.PBL menu templates

The menu templates defined in the PFC500TP.PBL library have eight menu positions. In applications generated from these templates, File, Tools, Function, and Help menus are always visible. The Edit and Window menus are visible only in the sheet menu bar. The View and Insert menus are never visible.

The position of menus is different for menus in frame and sheet windows.

Menu position	Menu name	Position in frame menu bar	Position in sheet menu bar
1	File	1	1
2	Edit	—	2
3	View	—	—
4	Insert	—	—
5	Tools	2	3
6	Window	—	5
7	Help	4	6
8	Function	3	4

The View and Insert menus are not visible in applications you generate with PFC500TP.PBL templates. If you assign items to these positions, you can make them visible in your application by editing the menus in PowerBuilder.

The menu position that you assign in PBGen is not necessarily the same as the menu position in your generated application. For example, if you use PFC500TP.PBL templates and you want your window menu items to appear in the Functions menu, you would assign them menu position 8. They would appear in position 3 in the frame menu and in position 4 in the sheet menus.

Use with frame window template
If you use the PB4TEMPL.PBL or the PFC500TP.PBL libraries, you must always use the _%m_frame%_ and _%m_sheet%_ menu templates in conjunction with the _%w_sys_frame%_ frame window template.

Using a select window template

Dynamic select window

For windows with a single row DataWindow, you must query the database to view or modify successive records. PBGen can generate a dynamic select window to query the database.

In both PB4TEMPL.PBL and PFC500TP.PBL libraries, the template _%w_sys_single_dw%_ generates a sheet window from which you can open a select window by clicking the Query button.

Alternative select window

Using the PB4TEMPL.PBL library, you can generate a select window from a PDM view. To do this, you attach the template _%w_sys_single_dw%_ to a table and the template _%w_select%_ to a view of that table. In the sheet window generated from the table, the Query button opens the window generated from the view.

Using the PFC500TP.PBL library
There is no _%w_select%_ template in the PFC500TP.PBL library. The _%w_sys_single_dw%_ template in this library generates a dynamic select window.

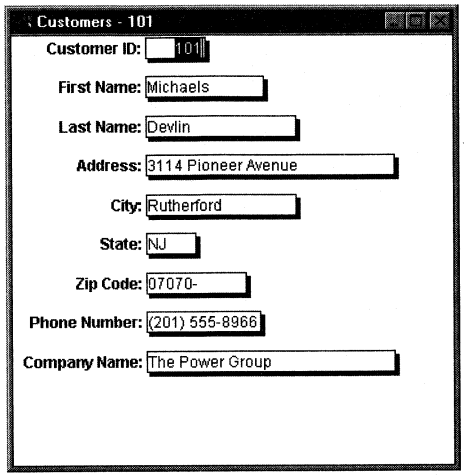
Rules for naming select windows

A select window based on a view must have the same name, except for the suffix, as the sheet window that opens it. The select window has the suffix *_select*. The corresponding sheet window can have the suffix *_sheet* or no suffix at all.

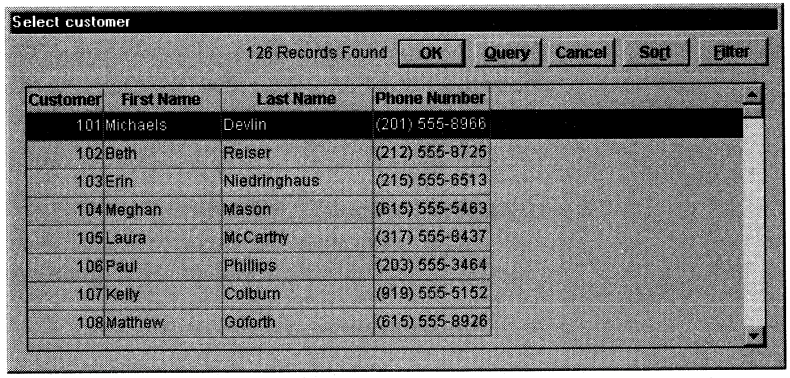
For example, if the window generated from a table is *w_author*, the select window generated from the view must be *w_author_select*.

Sample windows

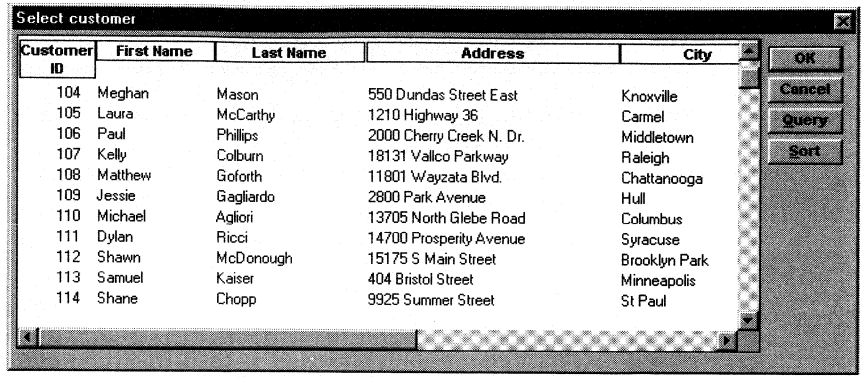
For example, the window below applies the template _%w_sys_single_dw%_ in the PB4TEMPL.PBL library.



This sheet window can call a select window like the one below, defined from the template _%w_select%_.



If you do not define a select window, PowerBuilder 4.0 dynamically generates the following window for queries.

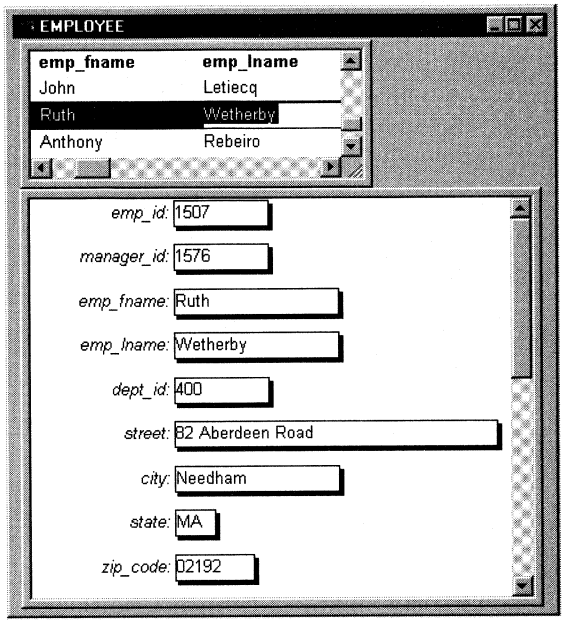


Using master/detail templates

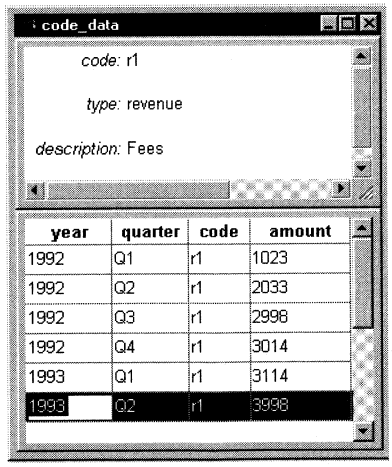
You can use the _%w_sys_mast_dtl_dw%_ template in PB4TEMPL.PBL and PFC500TP.PBL to generate master and detail DataWindow objects from a reference. However, if you are using PB4TEMPL.PBL, you must use the _%w_sys_shared_dw%_ template to generate objects from a reflexive reference.

Sample windows with master/detail objects

The table Employee contains a reflexive reference. When attached to this table, the template _%w_sys_shared_dw%_ generates the following window:



This window applies the template _%w_sys_mast_dtl_dw%_ to a reference linking two tables having the external key *code*.



Creating PBGen templates

PBGen allows you to generate PowerBuilder objects based on your own templates. PBGen recognizes templates with names that include the _% prefix and the _% suffix. After you rename your templates using this convention, you can access them from PBGen dialog boxes and generate applications based on these templates.

From PowerBuilder, you can also declare variables in your PBL library templates. Using PBGen you can assign values to these variables for an application or for a window.

Building a template library

The first step in creating a template library from an existing class library or framework is to develop a sample working application using the specific class library or framework. All template objects you design must work within the class library or framework.

PBGen uses a limited number of types of objects from a template library. Using PBGen, you can can generate one of each of the following types of objects:

- Application object
- MDI or main window and its associated menu object
- Common menu object for generated windows
- Window or user object, and DataWindow object for each table or view
- Window or user object, and master and detail DataWindow objects for each reference

You can use mulple generation passes to generate additional objects from a PDM to a single PBL.

Using template variables

Application templates, window templates, and DataWindow templates use variables in object definitions and scripts. PBGen creates an object by using the structure in a template and instantiating its variables.

For example, the PowerBuilder library PB4TEMPL.PBL contains an application template named _%mdiapp%_. In this template, the script for the open event uses variables, as follows:

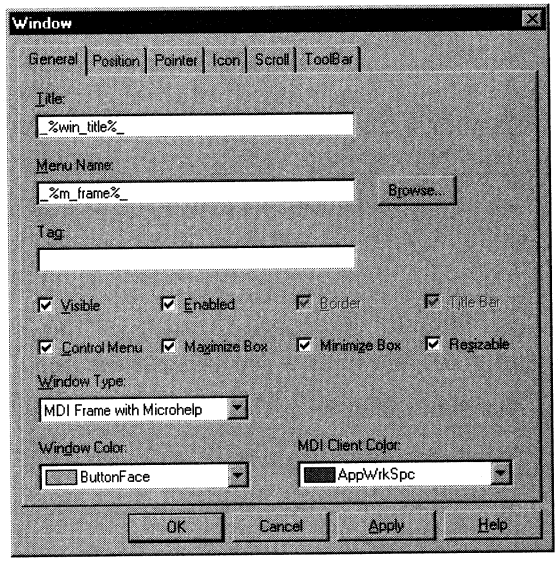
```
// _%app_title%_
gs_db_profile_section = "Profile _%db_profile%_"
f_app_open("_%db_ini_file%_", _%win_main%_, false)
```

In the application generated by PBGen, the resulting script assigns values to variables, as follows:

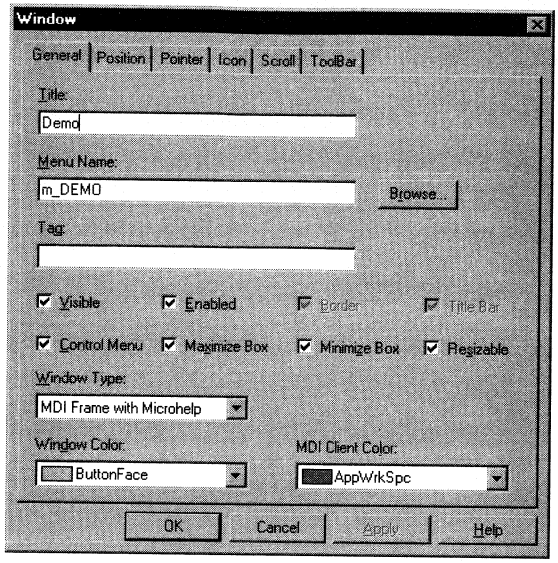
```
// Pubs
gs_db_profile_section = "Profile Pubs"
f_app_open("C:\PB4\PB.INI", w_PUBS, false)
```

Template variables viewed from PowerBuilder

The Window property sheet below shows the template _%w_sys_frame%_ as viewed in PowerBuilder 5.0.



After generation, the system variables _%win_title%_ and _%m_frame%_ are replaced by their corresponding values: Demo and m_demo in the example.



PBGen template variables

PBGen template variables have reserved names. If they exist in a template script, PBGen instantiates their values. These variables do not appear in the list of user-defined variables.

To define application objects and scripts, PBGen assigns values to the template variables.

Application variables

Variable	Description
_%app_title%_	Application title
_%db_profile%_	Database profile (connection parameters)
_%db_ini_file%_	Database profile INI filename
_%profile_section%_	Database profile section in INI file
_%win_main%_	Application window name
_%app_name%_	Application name
_%win_title%_	Window title (window variable)

Window and DataWindow variables

Variable	Description
_%win_title%_	Window title (window variable)
_%dw_main%_	Main DataWindow name
_%dw_master%_	Master DataWindow name
_%dw_detail%_	Detail DataWindow name

Variables that generate lists

Variable	Description
"%dw_pkey%"	List of key columns in the main DataWindow (single or master)
"%dw_fkey%"	List of foreign key columns for detail synchronization
"%d_retrieve%"	Detail retrieval arguments
"%dw_keytype% "	List of key types in a Master/Detail DataWindow (0=string, 1=number, 2=date, 3=time, 4=dateTime)

Debugging template objects

If you create your own templates from working applications and name them using the PBGen conventions described above, it is possible that you will need to correct generation errors. It is usually easier to debug the generated application and apply the changes back to the template objects than it is to keep changing the template objects and regenerating.

Sometimes ORCA errors occur during generation. If this happens, an error dialog box is displayed. By clicking the Save button in this error dialog box, you can save a text file containing the object syntax that caused the errors. You can then modify the syntax and manually import the text into PowerBuilder using the library painter.

CHAPTER 14

Using PowerBuilder Catalog Attributes

About this chapter

This chapter describes the transfer of catalog attributes between AppModeler and the PowerBuilder repository. It also describes how to generate a PDM view as a PowerBuilder query.

Contents

Before you begin

For information on importing and editing other client extended attributes, see the chapter "Client Interface." For information on PowerBuilder catalog attribute default values, see the appendix "Generation Variables and Attributes."

Using the PowerBuilder repository

Catalog attributes are extended attributes from system tables in the PowerBuilder repository. The repository contains information about how to display data from a database in screens and reports.

If the PowerBuilder repository defines values for catalog attributes, you can extract them from the repository and use them in the current PDM. This is a type of reverse engineering.

After you define values for catalog attributes, you can attach them to domains, tables, and columns. You can also generate them in the PowerBuilder repository.

Reverse engineering catalog attributes

You can reverse engineer catalog attributes from the PowerBuilder repository via an ODBC driver. This enables you to take advantage of catalog attributes already defined in the PowerBuilder dictionary.

After you reverse engineer catalog attributes, you can display definitions of these catalog attributes and attach them to domains and columns.

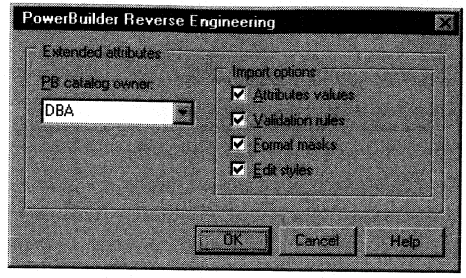
❖ To reverse engineer catalog attributes:

1 Select Client➤Reverse PowerBuilder Attributes.
or
Click the Reverse button in any of the following dialog boxes:

PowerBuilder Edit Styles
PowerBuilder Display Formats
PowerBuilder Validation Rules
PowerBuilder Table Extended Attributes
PowerBuilder Column Extended Attributes

> **Connecting to a data source**
> If you are not already connected to a data source, the ODBC Connection dialog box appears. You must connect to the data source via an ODBC driver.

The PowerBuilder Reverse Engineering dialog box appears.



2 Select a PowerBuilder catalog owner from the dropdown listbox.

The dropdown listbox only includes catalog owners for the data source you selected.

3 Select the All radio button.
or
Select the List radio button.

If you select List, the List of Tables dialog box appears. You can then select the tables from which you want to reverse engineer catalog attributes and click OK to return to the PowerBuilder Reverse Engineering dialog box.

4 Select checkboxes next to the types of catalog attributes you want to reverse engineer.

5 Clear checkboxes next to the types of catalog attributes you do not want to reverse engineer.

6 Click OK.

Generating catalog attributes

You can generate catalog attributes via an ODBC driver.

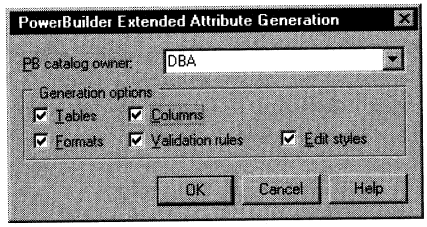
❖ To generate catalog attributes:

1 Select Client➤Generate PowerBuilder Attributes.
or
Click the Alter button in any of the following dialog boxes:

PowerBuilder Edit Styles
PowerBuilder Display Formats
PowerBuilder Validation Rules
PowerBuilder Table Extended Attributes
PowerBuilder Column Extended Attributes

> **Connecting to a data source**
> If you are not already connected to a data source, the ODBC Connection dialog box appears. You must connect to the data source via an ODBC driver.

The PowerBuilder Extended Attribute Generation dialog box appears.



2 Select a PowerBuilder catalog owner from the dropdown listbox.

The dropdown listbox only includes catalog owners for the data source you selected.

3 Select the All radio button.
or
Select the List radio button.

If you select List, the List of Tables dialog box appears. You can then select the tables from which you want to generate catalog attributes and click OK to return to the PowerBuilder Extended Attribute Generation dialog box.

4 Select the checkboxes next to the catalog attributes that you want to generate.

5 Clear the checkboxes next to catalog attributes that you do not want to generate.

6 Click OK.

Updating catalog attributes

When you open a model created with a version of S-Designor earlier than 4.2.1, display formats and validation rules are automatically added to the table lists in the PowerBuilder Display Formats and PowerBuilder Validation Rules dialog boxes.

Display formats

The display formats defined in the PowerBuilder Extended Attributes Definition list of an older version PDM are assigned names and formats based on values entered for certain list items. The following table shows the correspondence between the old values for these items and the new entries in the PowerBuilder Display Formats dialog box:

Former value for	PowerBuilder Display Formats column entry
PB_FormatCode	Name
PB_FormatText	Format

Validation rules

The validation rules defined in the PowerBuilder Extended Attributes Definition list of an older version PDM are assigned names, definitions, and validation message errors based on values entered for certain list items. The table below shows the correspondence between the old values for these items and the new entries in the PowerBuilder Validation Rules dialog box:

Former value for	PowerBuilder Validation Rules column entry
PB_ValidCode	Name
PB_ValidText	Definition
PB_ValidMsg	Validation Error Message

Generating PowerBuilder queries

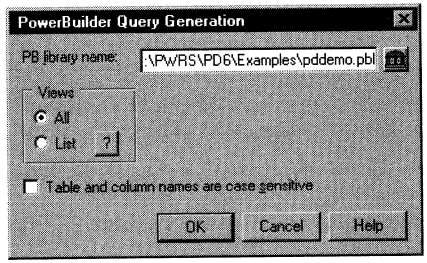
You can generate PowerBuilder query objects from PDM views, but not from user-defined views. You can select the views you want to generate from the PowerBuilder Query Generation dialog box.

AppModeler can create a PBL file to store the query objects, if you type a new filename in the PB Library Name box. Before you can generate a PowerBuilder query, you must verify that the path to the ORCA DLL files that you install with PowerBuilder are listed in your environment path settings.

❖ To generate PowerBuilder queries:

1 Select Client➤Generate PowerBuilder Queries.

The PowerBuilder Query Generation dialog box appears:



2 Specify a target PowerBuilder library.

3 Select the view definitions that you want to store as PowerBuilder queries.

The choices are All or List. If you select List, the List of Views dialog box appears. You can select view definitions from this dialog box that you want to generate as PowerBuilder queries and click OK to return to the PowerBuilder Query Generation dialog box.

4 Select the Table and Column Names Are Case-Sensitive checkbox.
or
Clear the checkbox if you want PowerBuilder to ignore letter case in table and column names.

5 Click OK.

A message box tells you that the queries have been successfully generated.

Modifying PowerBuilder catalog attributes

You can display and edit values for catalog attributes, or create new catalog attributes, by selecting one of the following items from the Client menu:

- PowerBuilder Edit Styles
- PowerBuilder Display Formats
- PowerBuilder Validation Rules

Each of these items corresponds to a type of catalog attribute.

Editing edit styles

Edit styles specify how column data is presented in DataWindow objects generated with PowerBuilder.

The following table shows the correspondence between the edit styles dialog boxes in the AppModeler and PowerBuilder environments:

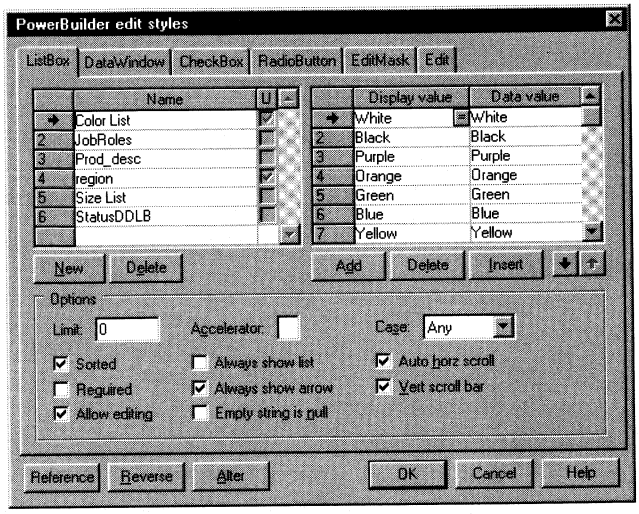
AppModeler	PowerBuilder edit style
ListBox	DropDownListBox
DataWindow	DropDownDataWindow
CheckBox	CheckBox
RadioButton	Radio
Edit Mask	Edit Mask
Edit	Edit

Each edit style type has different properties or options. You can reverse engineer edit style values from a PowerBuilder data source.

For information on reverse engineering edit styles, see "Reverse engineering catalog attributes" on page 320.

❖ To edit PowerBuilder edit styles:

1 Select Client➤PowerBuilder Edit Styles.



2 Click a tab for the edit style type to access.

3 Click the New button.
Type values for edit style properties and options.
or
Type changes to edit style values and options.

4 Click OK.

Creating a DataWindow edit style

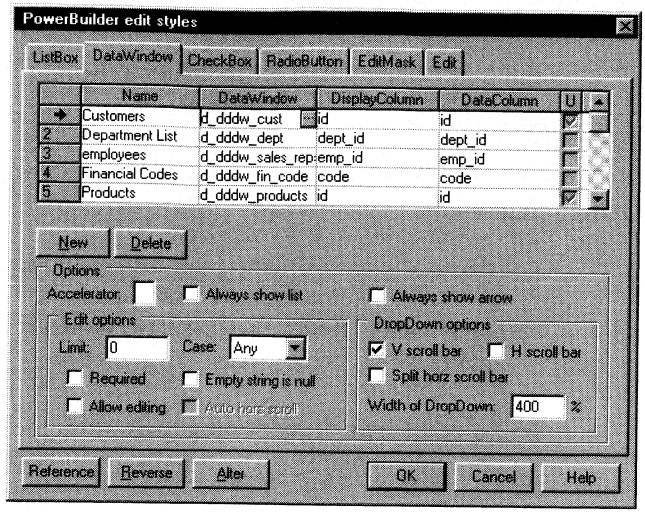
If you create a DataWindow edit style, you must attach it to a DataWindow that is defined in a PowerBuilder library (PBL file).

❖ To create a DataWindow edit style:

1 Select Client➤PowerBuilder Edit Styles.

2 Click the DataWindow tab.

The PowerBuilder Edit Styles dialog box opens to the DataWindow page.



3 Click the New button.

An arrow appears at the beginning of a blank line.

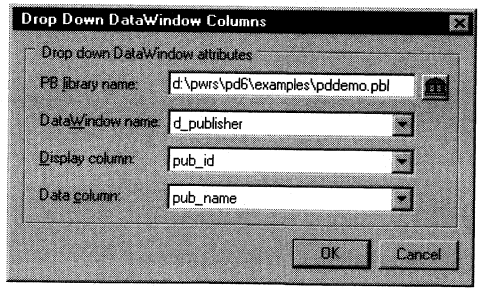
4 Type a name for the edit style in the Name column.

5 Click the DataWindow column on this line.

A ... button appears in the column.

6 Click the ... button.

The Drop Down DataWindow Columns dialog box appears.



7 Select the library (PBL file) in which the DataWindow is defined.

8 Select the DataWindow name from the DataWindow dropdown listbox.

9 Select the display column and data column values from their respective listboxes.

10 Click OK.

The values you selected appear in the corresponding columns of the DataWindow edit style list.

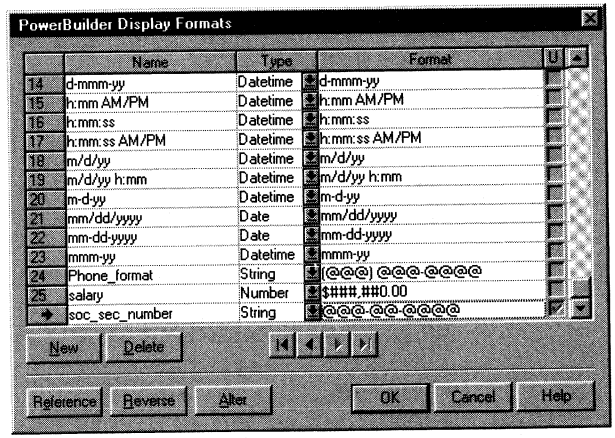
Editing display formats

Display formats customize the display of column data in a DataWindow object. The options in the PowerBuilder Display Formats dialog box correspond to those in the Display Formats dialog boxes in the PowerBuilder environment.

❖ To edit PowerBuilder display formats:

1 Select Client➤PowerBuilder Display Formats.

The PowerBuilder Display Formats dialog box appears.



2 Click the New button.
Type values for the Name and Format of the new display format.
or
Type changes to the Name and Format properties of the display format.

3 Select a data type from the Type dropdown listbox.

4 Click OK.

Editing validation rules

Validation rules check data before updating a database table associated with a DataWindow. The options in the PowerBuilder Validation Rules dialog box correspond to those in the Validation Rules dialog boxes in the PowerBuilder environment.

Using variables

You can use variables or functions in a validation rule definition or error message. You can add variables by clicking the @column button or the @domain button. The @column or @domain variables are replaced by the name of the column or domain to which they are attached.

Using functions

You can add functions to the rule definition by clicking a function from the Functions listbox. If the cursor is in the Validation Error Message box before you click a function in the list or a variable button, the function or variable is added to the validation error message instead of the rule definition.

PowerBuilder variables

You can use the following variables for PowerBuilder generation:

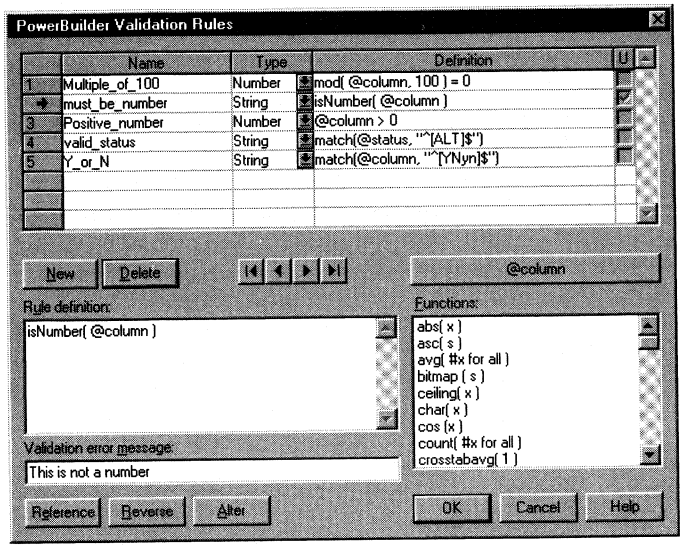
Variable	Value
%Default%	Check parameter default value of a column
%DomainCode%	Domain code
%Format%	Check parameter format of a column
%MinMax%	Validation rule using Min/Max values from check parameters of a column
%Rule%	Validation rule. You cannot combine this variable with any other variable or character

For information about other system variables that you can use in generating PowerBuilder catalog attributes, see the appendix "Generation Variables and Attributes."

❖ To edit PowerBuilder validation rules:

1 Select Client➤PowerBuilder Validation Rules.

The PowerBuilder Display Formats dialog box appears.



2 Click the New button.
Type a validation rule name in the Name box.
or
Type changes to the validation rule name.

3 Type a rule definition in the Rule Definition listbox.

4 (Optional) Type an error message in the Validation Error Message box.

5 Select a data type from the Type dropdown listbox.

6 Click OK.

Attaching PowerBuilder catalog attributes

You can attach values for catalog attributes to specific tables, domains, and columns. You can also attach catalog attributes to all tables, domains, or columns by default.

Attaching catalog attributes to a table

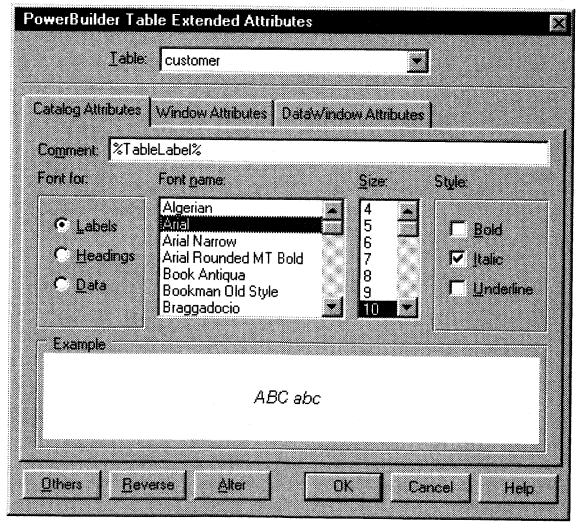
You can attach the following catalog attributes to a specific table or to all tables by default:

Catalog attribute	Description
Font, size, and style	Indicates how to display text coming from the table in headers, labels, and table data
Comment	Placeholder for the table label (by default, %TableLabel%)

❖ To change or display the catalog attributes of a PDM table:

1 Select Client➤PowerBuilder Table Attributes.

The PowerBuilder Table Extended Attributes dialog box appears.



2 Select a table name from the Table dropdown listbox.
or
Select Default in the Table dropdown listbox.

If you select Default, the new default value applies to all tables for which you have not previously modified the same catalog attributes. This includes all new tables.

3 Click the Labels radio button.

4 Select a font name, size, and style for labels.

5 Click the Heading radio button.

6 Select a font name, size, and style for headings.

7 Click the Data radio button.

8 Select a font name, size, and style for data.

9 Click OK.

Displaying other extended attributes
The Others button opens a dialog box that lists extended attributes for other Client tools and customized attributes for PowerBuilder.

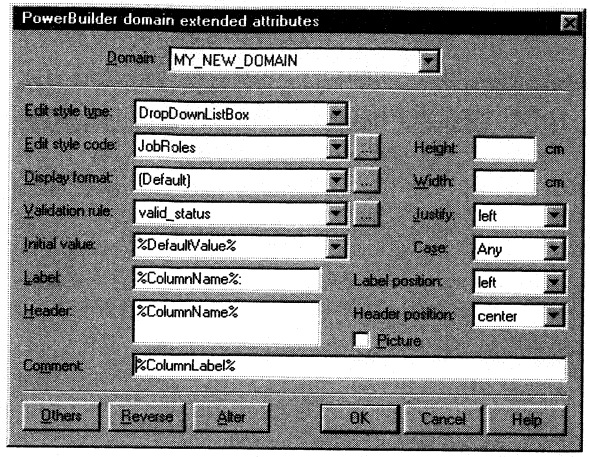
Attaching an edit style to a domain

You can attach an edit style to a domain and apply this edit style to all columns in the domain.

❖ **To attach an edit style to a domain:**

1 Select Client➤PowerBuilder Domain Attributes.

The PowerBuilder Domain Extended Attributes dialog box appears.



2 Select a domain from the Domain dropdown listbox.
or
Select Default in the Domain dropdown listbox.

If you select Default, the new default value applies to all domains for which you have not previously modified the edit style. This includes all new domains.

3 Select a type from the Edit Style Type dropdown listbox.
or
Select All to choose all edit style types.

Edit style type determines the edit styles that appear in the Edit Style Code dropdown listbox. If you select All, the Edit Style Code box displays all edit styles.

4 Select an Edit Style Code from the dropdown listbox.
or
Select None if you do not want to attach an edit style to the domain.

Edit style definitions
You can access edit style definitions by clicking the ... button next to the Edit Style Code dropdown listbox.

5 Click OK.

If you modified the edit style for a named domain, a message box asks if you want to modify catalog attributes for columns currently attached to the domain.

- Click Yes to modify catalog attributes for all columns attached to the selected domain.
  or
  Click No to limit your modifications to columns that you subsequently attach to the selected domain.

Attaching a display format to a domain

You can attach a display format to a domain and apply this display format to all columns in the domain.

❖ **To attach a display format to a domain:**

1 Select Client➢PowerBuilder Domain Attributes.

The PowerBuilder Domain Extended Attributes dialog box appears.

2 Select a domain from the Domain dropdown listbox.
or
Select Default in the Domain dropdown listbox.

If you select Default, the new default value applies to all domains for which you have not previously modified the display format. This includes all new domains.

3 Select a display format from the dropdown listbox.
or
Select None if you do not want to attach a display format to the domain.

Display format definitions
You can access display format definitions by clicking the  button next to the Display Format dropdown listbox.

4 Click OK.

If you modified the display format for a named domain, a message box asks if you want to modify catalog attributes for columns currently attached to the domain.

- Click Yes to modify catalog attributes for all columns attached to the selected domain.
  or
  Click No to limit your modifications to columns that you subsequently attach to the selected domain.

Creating a default display format for a domain

The default display format for a domain is the format defined in the Check Parameter dialog box for the domain.

If you create or modify the default format from this dialog box, PBGen automatically transfers the format value to the corresponding display format.

❖ To create a default display format for a domain:

1 Select Client➤PowerBuilder Domain Attributes.

The PowerBuilder Domain Extended Attributes dialog box appears.

2 Select a domain from the Domain dropdown listbox.

3 Select Default in the Display Format dropdown listbox.

4 Click OK.

You return to the PDM.

5 Select Dictionary➤List of Domains.

The List of Domains dialog box appears.

6 Select the same domain that you selected in step 2.

An arrow appears at the beginning of the line.

7 Click the Check button.

The Check Parameters dialog box opens to the Standard Parameters page.

8 Type a value in the Format textbox.

9 Click OK twice.

You return to the PDM.

10 Select Client➤PowerBuilder Domain Attributes.

11 Select the same domain that you selected in step 2.

The Default display for the Display Format textbox changes to df_*DomainCode*.

12 Click the button next to the Display Format dropdown listbox.

PowerBuilder Display Formats dialog box displays the following values for the new display format:

Display format	Default value assigned
Name	df_*DomainCode*
Type	Same data type as the domain
Value	Value entered in the Format box in the domain Check Parameters dialog box

Attaching a validation rule to a domain

You can attach a validation rule to a domain and apply this rule to all columns in the domain.

❖ **To attach a validation rule to a domain:**

1 Select Client➤PowerBuilder Domain Attributes.

The PowerBuilder Domain Extended Attributes dialog box appears.

2 Select a domain from the Domain dropdown listbox.
or
Select Default in the Domain dropdown listbox.

If you select Default, the new default value applies to all domains for which you have not previously modified the validation rule. This includes all new domains.

3 Select a validation rule from the dropdown listbox.
or
Select None if you do not want to attach a validation rule to the domain.

> **Validation rule definitions**
> You can access validation rule definitions by clicking the button next to the Validation Rule dropdown listbox.

4 Click OK.

If you modified the validation rule for a named domain, a message box asks if you want to modify catalog attributes for columns currently attached to the domain.

- Click Yes to modify catalog attributes for all columns attached to the selected domain.
  or
  Click No to limit your modifications to columns that you subsequently attach to the selected domain.

Creating a default validation rule for a domain

The default validation rule for a domain is the format defined as the Client Validation Rule in the Check Parameters dialog box for the domain.

If you create or modify the expression from the Check Parameters dialog box, PBGen automatically transfers the rule to the corresponding catalog attribute.

❖ **To create a default validation rule for a domain:**

1 Select Client➤PowerBuilder Domain Attributes.

The PowerBuilder Domain Extended Attributes dialog box appears.

2 Select a domain from the Domain dropdown listbox.

3 Select Default in the Validation Rule dropdown listbox.

4 Click OK.

5 Select Dictionary➤List of Domains.

The List of Domains dialog box appears.

6 Select the same domain that you selected in step 2.

An arrow appears at the beginning of the line.

7 Click the Check button.

The Check Parameters dialog box appears.

8 Click the Validation Rules tab.

9 Click the Additional radio button above the Client Validation Rule textbox.

10 Type the text of the Client Validation Rule.

11 Click OK twice.

12 Select Client➤PowerBuilder Domain Attributes.

13 Select the same domain that you selected in step 2.

The Default display for the Display Format textbox changes to dr_*DomainCode*.

14 Click the button next to the Display Format textbox.

PowerBuilder Display Formats dialog box displays the following values for the new display format:

Display format	Default value assigned
Name	dr_*DomainCode*
Type	Same data type as the domain
Value	Client validation rule from the Check Parameters dialog box

Attaching other catalog attributes to a domain

In addition to an edit style, a display format, and a validation rule, you can assign values to the following catalog attributes:

Catalog attribute	Description
Initial value	Initial data value for a column attached to the domain
Label and header	Text that identifies columns in DataWindow objects
Comment	Text that describes the purpose of a column
Height and width	Display size for data in a DataWindow object
Justify	Alignment for data in a DataWindow object
Case	Specifies how character data is displayed
Label and header position	Alignment of label and header text
Picture checkbox	When selected, indicates a picture in the DataWindow

❖ To attach other catalog attributes to a domain:

1 Select Client➤PowerBuilder Domain Attributes.

The PowerBuilder Domain Extended Attributes dialog box appears.

2 Select a domain from the Domain dropdown listbox.
or
Select Default in the Domain dropdown listbox.

If you select Default, the new default values apply to all domains for which you have not previously modified the same attributes. This includes all new domains.

3 Type values for catalog attributes.

4 Click OK.

If you modified catalog attributes for a named domain, a message box asks if you want to modify the catalog attributes for columns currently attached to the domain.

- Click Yes to attach catalog attributes to all columns attached to the selected domain.
  or
  Click No to limit your modifications to columns that you subsequently attach to the selected domain.

Attaching catalog attributes to a column

You can attach values for catalog attributes to a specific column or to all columns by default.

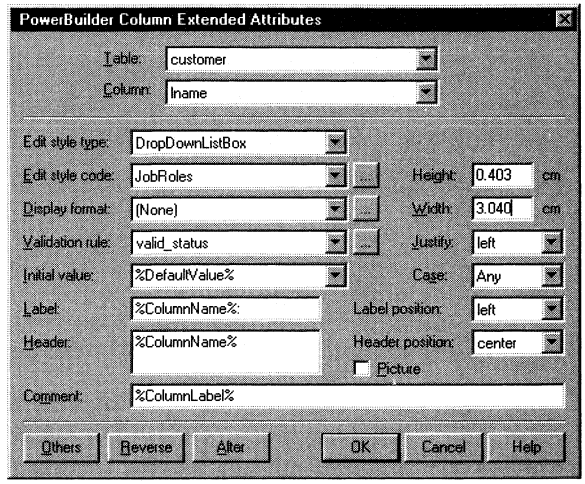
In addition to an edit style, a display format, and a validation rule, you can assign values to the following catalog attributes:

Catalog attribute	Description
Initial value	Initial data value for a column
Label and header	Text that identifies columns in DataWindow objects
Comment	Text that describes the purpose of a column
Height and width	Display size for data in a DataWindow object
Justify	Alignment for data in a DataWindow object
Case	Specifies how character data is displayed
Label and header position	Alignment of label and header text
Picture checkbox	When selected, indicates a picture in the DataWindow

❖ To attach an edit style to a column:

1 Select Client➤PowerBuilder Column Attributes.

The PowerBuilder Column Extended Attributes dialog box appears.



2 Select a column from the Column dropdown listbox.
or
Select Default in the Column dropdown listbox.

If you select Default, the new default value applies to all columns for which you have not previously modified the edit style. This includes all new columns.

3 Select an Edit Style Type from the dropdown listbox.
or
Select All to choose all edit style types.

If you select an edit style type, the Edit Style Code box displays that type of edit style. If you select All, the Edit Style Code box displays all edit styles.

4 Select an Edit Style Code from the dropdown listbox.
or
Select None if you do not want to attach an edit style to the column.

5 Select a Display Format from the dropdown listbox.
or
Select None if you do not want to attach a display format to the domain.

6 Select a Validation Rule from the dropdown listbox.
or
Select None if you do not want to attach a validation rule to the domain.

Catalog attribute definitions
You can access the definitions of catalog attributes by clicking the button next to the corresponding dropdown listbox.

7 Type values for other catalog attributes.

8 Click OK.

Creating a default display format for a column

The default display format for a column is the format defined in the Check Parameters dialog box for that column.

If you create or modify the default format from the Check Parameters dialog box, PBGen automatically transfers the format value to the corresponding display format.

❖ **To create a default display format for a column:**

1 Select Client➤PowerBuilder Column Attributes.

The PowerBuilder Column Extended Attributes dialog box appears.

2 Select a column from the Column dropdown listbox.

3 Select Default in the Display Format dropdown listbox.

4 Click OK.

5 Select Dictionary➤List of Columns.

The List of Columns dialog box appears.

6 Select the column for which you selected the default display format.

An arrow appears at the beginning of the line.

7 Click the Check button.

The Check Parameters dialog box opens to the Standard Parameters page.

8 Type a value in the Format textbox.

9 Click OK twice.

10 Select Client➤PowerBuilder Column Attributes.

11 Select the column for which you assigned the default value.

The Default display for the Display Format box changes to cf_*ColumnCode*.

12 Click the button next to the Display Format box.

PowerBuilder Display Formats dialog box displays the following values for the new display format:

Display format	Default value assigned
Name	cf_*ColumnCode*
Type	Same data type as the column
Value	Value entered in the Format box in the domain Check Parameters dialog box

Creating a default validation rule for a column

The default validation rule for a column is the format defined as the Client Validation Rule in the Check Parameters dialog box for the column.

If you create or modify the expression from the Check Parameters dialog box, PBGen automatically transfers the rule to the corresponding catalog attribute.

❖ To create a default validation rule for a column:

1 Select Client➤PowerBuilder Column Attributes.

The PowerBuilder Column Extended Attributes dialog box appears.

2 Select a column from the Column dropdown listbox.

3 Select Default in the Validation Rule dropdown listbox.

4 Click OK.

5 Select Dictionary➤List of Columns.

The List of Columns dialog box appears.

6 Select the column for which you selected the default validation rule.

An arrow appears at the beginning of the line.

7 Click the Check button.

The Check Parameters dialog box appears.

8 Click the Validation Rules tab.

9 Click the Additional radio button above the Client Validation Rule textbox.

10 Type the text of the Client Validation Rule.

11 Click OK.

12 Click OK.

13 Select Client➤PowerBuilder Column Attributes.

14 Select the domain for which you assigned the default value.

The Default display for the Display Format box changes to cr_*ColumnCode*.

15 Click the button next to the Display Format box.

PowerBuilder Display Formats dialog box displays the following values for the new display format:

Display format	Default value assigned
Name	cr_*ColumnCode*
Type	Same data type as the column
Value	Client validation rule from the Check Parameters dialog box

Displaying the list of objects that use a catalog attribute

In the dialog boxes for PowerBuilder edit styles, display formats, and validation rules, the U column indicates if an attribute is attached to a PDM column or domain.

❖ To display the list of objects that use a catalog attribute:

1 Select Client➤PowerBuilder Edit Styles and click a tab control for an edit style.
or
Select Client➤PowerBuilder Display Formats.
or
Select Client➤PowerBuilder Validation Rules.

2 Click a catalog attribute that has a checkmark in the U column.

An arrow appears at the beginning of the line and the Reference button becomes selectable.

3 Click the Reference button.

A list dialog box displays the columns and domains that use the catalog attribute.

4 Click OK twice.

CHAPTER 15

Visual Basic Generator

About this chapter

This chapter provides an overview of the AppModeler Visual Basic Generator (VBGen) and how it works.

Contents

Before you begin

VBGen works with Visual Basic 3.0 or 4.0. It works with the Professional or Enterprise Editions of Visual Basic. If you want to use VBGen to generate projects for the Standard Edition of Visual Basic, you must create your own set of templates or modify the templates that ship with AppModeler.

Generator basics

VBGen is the AppModeler generator for Visual Basic. It generates Visual Basic projects, forms, and controls using predefined templates and information from objects and attributes that you define in a PDM.

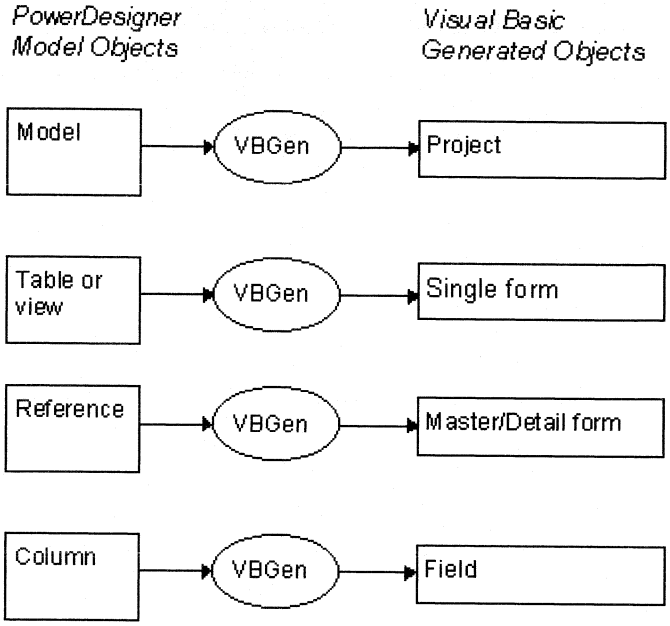
You start AppModeler for Visual Basic by double-clicking PDVB6.EXE or by clicking its icon in the Program Manager or taskbar (Windows 95).

What VBGen does

VBGen uses data from a PDM to instantiate predefined templates and generate Visual Basic applications. VBGen generates an entire Visual Basic application, including:

File type	What it is	Description
VBP *or* MAK	Project file for Visual Basic 4.0 *or* Project file for Visual Basic 3.0	Refers to other application files (FRM, BAS, and OCX or VBX)
FRM	Form files, including separate files for main forms and each data form	Contains menus, controls, code for event handlers, and form procedures and functions

The following schema shows the object transformations performed by VBGen:



Installed files and directories

VBGen includes the following disk files that you install with the Setup program:

Filename	Directory	Description
PDVB6.EXE	PD6*	Executable file
PD6.INI#	WINDOWS	Initialization information
VB.EXA *or* VB3.EXA	PD6\EXA*	Default attribute import file

* Only short path names are shown in the table. Long names are PowerDesigner 6 (PD6) and Extended Attributes (EXA).

Initialization information for the 32-bit version is stored in the registry key: HKEY_CURRENT_USER\Software\Powersoft\PowerDesigner 6\AppModeler for Visual Basic

VBGen templates

VBGen ships with several templates and resource files. The Setup program installs them in the PD6\VB4TPL or VB4TPL16 or VB3TPL directory.

If you use a Windows registry, references to template files are included in subkeys in the registry path HKEY_CURRENT_USER\Software\Powersoft\PowerDesigner 6\AppModeler for Visual Basic. Otherwise, they are included in the template section of the PD6.INI file.

Project and form templates

Project and form templates that ship with VBGen are:

Filename	INI section or registry subkey	Template type
PROJECT.VPT	Project Templates	Project
MDIMAIN.VMT	Main Form Templates	Application form
SINGLE.VFT F1SINGLE.VFT	Table Form Templates View Form Templates	Single data source form
MSTRDETL.VFT	Reference Form Templates	Master/detail data source form

Associated Visual Basic files

The project template includes a PROJECT.VBP or PROJECT.MAK project file. The project file refers to the PDTOOLS.BAS module that VBGen copies to your project directory.

For Visual Basic 4.0, each VBGen form template includes an FRM and FRX file. For example, SINGLE.VFT ships with SINGLE.FRM and SINGLE.FRX files. FRX files save binary data for certain custom controls.

The Setup program installs these files in your VBGen template directory.

Filenames	Description
PROJECT.VBP *or* PROJECT.MAK	Visual Basic project file
MDIMAIN.FRM	Visual Basic main form file
SINGLE.FRM	Visual Basic form file with single data source
F1SINGLE.FRM*	Visual Basic form file with single data source to be used with Formula One grid
MSTRDETL.FRM	Visual Basic form file with master/detail data source
PDTOOLS.BAS	Module containing functions for data controls. It also adds a toolbar and a status bar (VB4 only) to your projects. VBGen copies it to your project directory at the same time you generate your project

* Not available for Visual Basic 3.0

Miscellaneous project and form files

Some Visual Basic files you install with AppModeler are not required for VBGen to function properly. They refer to each of the form files and icons used in the templates that ship with VBGen.

Filenames	Description
FORMTMPL.VBP *or* FORMTMPL.MAK	Visual Basic project file that contains references to the FRM files associated with each VBGen form template
ALLICONS.FRM*	Contains all the icons used in the templates that ship with VBGen

* Not available with the 16-bit version of AppModeler

Field templates

Field style templates that ship with VBGen include templates for computed fields and various controls, as follows:

Filename	INI section or registry subkey	Template type
IMAGE.VCT	Image Field Templates	Bitmap
COMPUTED.VCT	Computed Field Templates	Computed fields
CHECKBOX.VCT CHKBOX2S.VCT CHKBOX3S.VCT	CheckBox Field Templates	Checkbox
DBCOMBO1.VCT* CHKLIST.VCT	DropDown Field Templates	Combination box
RADIOH.VCT RADIOHV.VCT RADIOV.VCT RADIOVH.VCT	RadioButton Field Templates	Radio buttons
LABELBOX.VCT MASKEDIT.VCT MLTXTBOX.VCT RTFBOX.VCT# SPINNUM.VCT TEXTBOX.VCT	Edit Field Templates	Text area

* Not available for Visual Basic 3.0

Not available with the 16-bit version of AppModeler

Grid templates

VBGen ships with four grid templates for Visual Basic 4.0. Two of these templates, DBGRID.VGT and F1GRID.VGT, are not available for Visual Basic 3.0.

References to grid templates are added to the Grid Templates subkey of the registry or the Grid Templates section of PD6.INI.

To use grid templates you must install the following OCX or VBX controls:

Filename	OCX or VBX control required	Visual Basic control name	OCX or VBX source
DBGRID.VGT	Database Grid	DBGrid	Ships with Visual Basic 4.0
TDBGRID.VGT	TrueGrid	TDBGrid	Apex Software Corp.
F1GRID.VGT	FormulaOne	F1Book	Visual Components
FPGRID.VGT	Spread	vaSpread	FarPoint Technologies

For the grid templates that require binary data files, the following files ship with VBGen:

Filename	Form type	Binary data files
DBGRID.VGT	Single Master Detail	DBGRID.FRX MDBGRID.FRX DDBGRID.FRX
TDBGRID.VGT	Single Master Detail	TDBGRID.FRX MTDBGRID.FRX DTDBGRID.FRX
FPGRID.VGT	Single Master Detail	FPGRID.FRX MFPGRID.FRX DFPGRID.FRX

Ensuring that your project refers to an existing database

The projects you develop with VBGen use information from existing databases. You must ensure that the database exists and contains the fields and data types defined in your PDM before your project can work properly.

❖ **To ensure that your project refers to an existing database:**

- Generate a database directly from the PDM.
  or
  Reverse engineer an existing database to create the PDM.

Importing extended attributes

The VB.EXA and VB3.EXA files contain default attribute values for object generation from VBGen. You can import these attributes and values into any PDM built with PowerDesigner or S-Designor. The VB.EXA attribute values are loaded automatically into new PDMs.

❖ **To import extended attributes:**

1 Select Dictionary➤Extended Attributes➤Import Attributes.

2 Select the VB.EXA file if you are generating for Visual Basic 4.0.
or
Select the VB3.EXA file if you are generating for Visual Basic 3.0.

> **For Visual Basic 3.0**
> If you are working regularly with Visual Basic 3.0, rename the current VB.EXA as VB4.EXA. You can then copy or rename VB3.EXA as VB.EXA, and the default attribute values for Visual Basic 3.0 will be loaded automatically for new PDMs.

3 Click OK.

A message box tells you if the extended attribute import is successful.

Changing the filename format

The 32-bit version of AppModeler for Visual Basic generates long filenames by default and the 16-bit version generates short filenames. You can force the generation of short filenames in the 32-bit version by changing the UseShortNames string value in the registry.

❖ **To change the filename format of the 32-bit version of AppModeler:**

1 Open the registry to HKEY_CURRENT_USER\Software\Powersoft\ PowerDesigner 6\ AppModeler for Visual Basic\VB4 32-bit.

2 Double-click the string name Use Short Names.

3 Type **Yes** for the new string data value.

> **If you generate projects for 16-bit compatibility**
> You must also use directory (path) names that have eight characters or less and do not include blank characters.

Building a project

A project file manages an Visual Basic application. Project files contain addresses and filenames for Visual Basic forms and for other project resources.

A project file generated by VBGen includes information about project templates and project properties.

Selecting a project template and project properties

VBGen generates a project file by instantiating templates with values from the PDM. You attach project and main form templates to the PDM and define certain project properties:

Property	Description
Template set*	VB3, VB4 16-bit, VB4 32-bit, or customized target
Project template	Name of template you use to build your project
Project name	Filename for the project
Project title	Title for the project
Project directory	Directory where you generate project files
Application form template	Name of template you use to build your main form
Application form name	Filename for the main form of your project
Menu items position	Menu bar position for data form menu items

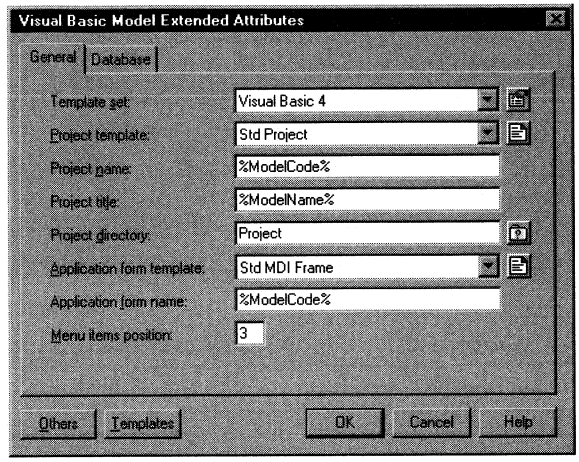
* The Template Set dropdown listbox includes any set of templates that you name as a string value in the AppModeler for Visual Basic/Template Sets registry key or in the PD6.INI file.

VBGen can create a new project directory if you type the directory name in the Project Directory box. If you do not type a complete path, VBGen looks for the directory in the PWRS\PD6 path, or in its long name equivalent.

❖ To select a project template and project properties:

1 Select Client➤Visual Basic Model Attributes.

The Visual Basic Model Extended Attributes dialog box opens to the General properties page.



2 Select a template set from the Template Set dropdown listbox.

Only template sets listed in your registry or PD6.INI file are available in the dropdown listbox.

3 Select a project template from the Project Template dropdown listbox. Select a main form template from the Application Form Template dropdown listbox.

The Setup program installs the Std Project and Std MDI Main Form templates as the default selections.

4 (Optional) Type changes to Project Name and Application Form Name.

You do not need to add VBP, MAK, or FRM extensions to the project name or application form name. VBGen adds them automatically during generation.

5 Use the button to select a project directory.
or
Type a directory name in the Project Directory dropdown listbox.

The files you generate with VBGen will be saved in this directory.

6 Make changes to any of the other General page properties that you want to implement in your application.

7 Click OK.

Defining a database connection

To generate an application that displays information from a database, you must specify connection information. You can specify this information using one of the following:

- A local database Jet engine recognized by Visual Basic
- An ODBC data source

Defining a Jet engine connection

You can use a Jet engine to connect to any Database Management System (DBMS) supported by Visual Basic.

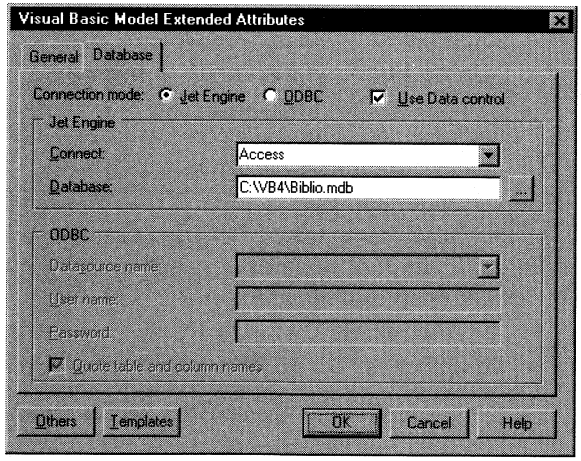
❖ To define a Jet engine database connection for your project:

1 Select Client➤Visual Basic Model Attributes.

The Visual Basic Model Extended Attributes dialog box opens to the General page.

2 Click the Database tab.

The Database properties page opens.



3 Select the Jet Engine radio button.

The ODBC zone is grayed after you select this button. The Use Data Control checkbox is selected.

4 Select a DBMS from the Connect dropdown listbox.

5 Use the button to select the database for your application.
or
Type the database name in the Database textbox.

By default, VBGen limits the search to databases that have a file type corresponding to the DBMS you selected in step 4.

6 Click OK.

Defining an ODBC data source

If you plan to display information in the projects that you generate through an ODBC connection, you must define an ODBC data source.

Using nonstandard data access control

If you want to use the Visual Basic remote data control (RDO) or another nonstandard control, you must specify the control in your form templates.

For example, VBGen ships with the F1SINGLE.VFT form template that defines a nonstandard Formula One grid contol. Forms generated with this template may not work properly if you do not clear the Use Data Control checkbox.

SQL queries

If the table and column codes in your PDM contain special characters (blank spaces, slashes, and so on), VBGen ensures that quotation marks are generated around these names in the SQL queries of your Visual Basic project.

> **Using blank spaces in column codes**
> The use of blank spaces in column codes is not recommended. If you use column codes containing blank spaces in views, queries generated from these columns may not execute properly.

If you select the Quote Table and Column Names checkbox, VBGen generates quotation marks around every table and column name used in the SQL queries of your project.

❖ **To define an ODBC data source connection for your project:**

1 Select Client➤Visual Basic Model Attributes.

2 Click the Database tab.

The Database properties page opens.

3 Select the ODBC radio button.

The Jet engine zone is grayed after you select this button.

4 Clear the Use Data Control checkbox if you do not want to use the standard Visual Basic data access control (DAO).

5 Select the data source you want your application to access.

The Datasource dropdown listbox only includes data sources that you have already defined with an ODBC administrator.

6 Type the name and password needed to access the data source data.

7 Select the Quote Table and Column Names checkbox.
or
Clear the Quote Table and Column Names checkbox.

8 Click OK.

Defining forms and fields

Forms are the primary interface elements of Visual Basic projects. Each form must have its own FRM file. VBGen generates FRM files from individual tables, views, and references.

The term **data form** is used to distinguish child forms from the main form (frame window) of a project. Data forms contain data that is presented in fields that you define with VBGen field templates.

Fields are form elements that correspond to database columns. In VBGen, field templates are used to define a Visual Basic control or set of controls that can display data from database rows.

Controls are form or field elements that you can use in applications to get user input or to display output. Textboxes, listboxes, and command buttons are examples of standard Visual Basic controls.

Selecting form templates and form properties

For each table, view, and reference in the PDM, you can define the following general properties:

Property	Description
Generate form	Select to generate form
Form template	Name of template you use to build your form
Form name	Form name for the Visual Basic project window
Form title	Title for the form
Form filename	Filename for form (VBGen adds an FRM extension)
Menu option checkbox	Select to generate a menu item to open form
Menu option textbox	Name of the menu item to open the form

What is a menu option?

The Menu Option checkbox indicates whether or not a menu item will be generated for a form. The menu option name is the name of the menu item that opens the form in the project you generate. If you want VBGen to generate the menu option name as a menu item, you must select the Menu Option checkbox.

Menu hot key
You define a menu hot key by typing an ampersand (&) before a letter of the menu option name. Pressing the hot key opens the data form from the generated frame and sheet menus.

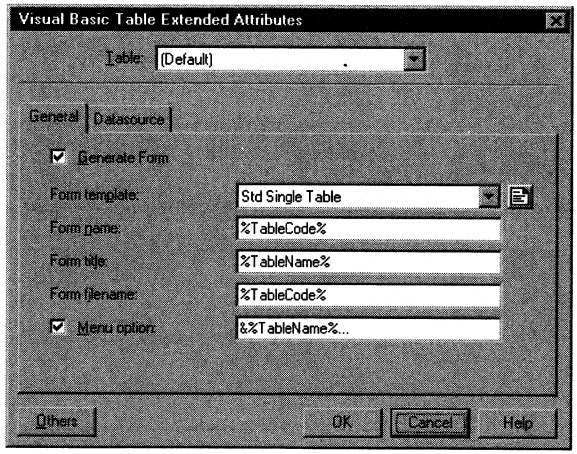
Defining general properties

You can use default general properties for a PDM object type or you can define form properties for each table, view, or reference.

❖ To define general properties:

1 Select Client➤Visual Basic Table Attributes.
or
Select Client➤Visual Basic View Attributes.
or
Select Client➤Visual Basic Reference Attributes.

The Visual Basic *Object* Extended Attributes dialog box appears.



2 Select a table, view, or reference from the *Object* dropdown listbox

If the dropdown listbox displays Default, any changes to object properties that you make in this dialog box apply to all objects of the same type for which these properties have not been previously modified. This includes all new objects that you add to the PDM.

3 Select a form template from the Form Template dropdown listbox.

4 Make changes to other properties, as needed.

5 Click OK.

Defining data source properties for a table or view

Datasource properties for a table or view include the read-only property and the presentation style of the form you generate. The read-only property specifies whether or not users will be able to update the data source from a form that you generate.

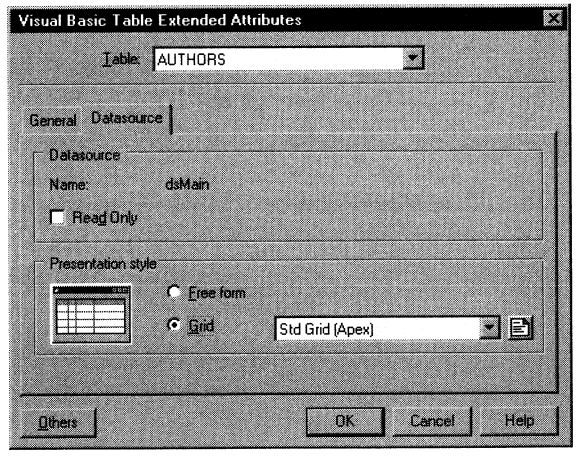
❖ To define data source properties for a table or view:

1 Select Client➤Visual Basic Table Attributes.
or
Select Client➤Visual Basic View Attributes.

2 Select a table or view from the dropdown listbox at the top of the Visual Basic Extended Attributes dialog box.

3 Click the Datasource tab.

The Extended Attributes dialog box opens to the Datasource page.



4 Select or clear the Read Only checkbox.

Select the Read Only checkbox if you don't want users of the application to be able to update the data source.

5 Select a form presentation style.

6 If you select Grid, select the Grid template you want to use from the dropdown listbox.

7 Click OK.

Defining data source properties for a reference

A reference has a master data source and a detail data source. The data source can be a table or view. Depending on your selection, the objects you generate from a reference can use information from table or view columns.

❖ To define data source properties for a form based on a reference:

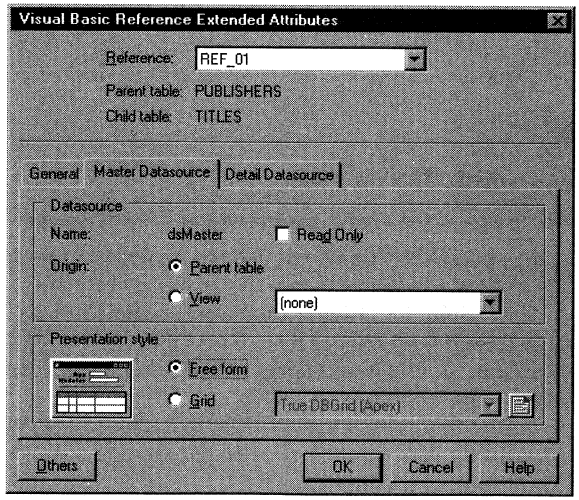
1 Select Client➤Visual Basic Reference Attributes.

The Visual Basic Reference Extended Attributes dialog box appears.

2 Select a reference from the Reference dropdown listbox.

3 Click the Master Datasource tab.

The Master Datasource page opens.



4 Click the Parent Table radio button.
or
Select a view from the View dropdown listbox.

Only views that contain the parent table primary keys are available from the dropdown listbox.

5 Select or clear the Read Only checkbox.

Select the Read Only checkbox if you don't want users of the application to be able to update the master data source.

6 Select a master form presentation style.

7 If you select Grid, select a Grid template from the dropdown listbox.

8 Click the Detail Datasource tab.

The Detail Datasource page opens.

9 Click the Child Table radio button.
or
Select a view from the View dropdown listbox.

Only views that contain a foreign key of the selected reference are available from the dropdown listbox.

10 Select or clear the Read Only checkbox.

Select the Read Only checkbox if you don't want users of the application to be able to update the detail data source.

11 Select a detail form presentation style.

12 If you select Grid, select the Grid template you want to use from the dropdown listbox

13 Click OK.

Assigning computed fields templates

If you use calculated expressions in SQL statements for a view, you must use a computed fields template to generate a form based on the view.

Using aliases

To improve readability of SQL statements, you can use aliases. However, do not type **as** between an expression and its alias. VBGen adds this to the SQL statement when it generates a form based on the view.

Example

For example, the following SQL statement uses a calculated expression (the group function *count*) with the alias BOOKS:

```
SELECT P.NAME, count(*) BOOKS
FROM PUBLISHERS P, TITLES T
WHERE P.PUBID = T.PUBID
GROUP BY P.NAME
```

VBGen converts this to the following Visual Basic code:

```
sQuery = " Select   P.NAME, count(*) As BOOKS"
sQuery = sQuery & " From     PUBLISHERS P, TITLES T"
sQuery = sQuery & " Where    P.PUBID = T.PUBID"
sQuery = sQuery & " Group By P.NAME"
```

Default template

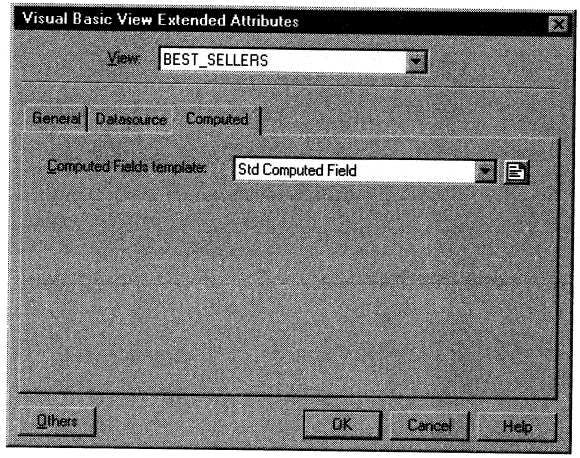
VBGen ships with a computed field template that enables you to display calculated expressions. By default, VBGen assigns this template to all calculated expressions.

❖ **To assign a new computed field template:**

1 Select Client➤Visual Basic View Attributes.

2 Click the Computed tab.

The Visual Basic View Extended Attributes dialog box opens to the Computed page.



3 Select a template from the Computed Fields Template dropdown listbox.

You can only assign one computed fields template to a form.

4 Click OK.

Defining field attributes

To use information from a database in a generated application, you can attach field styles and templates to columns or domains.

The default field style is Edit. TEXTBOX.VCT is the default field template for the Edit field style. The default field template for a field style is defined in the Field Styles subkey of the registry or in the Field Styles subsection of the PD6.INI file.

For information on adding field styles or changing default field templates, see "Adding a template or a field style" on page 388.

Attaching a field style and field template to a column

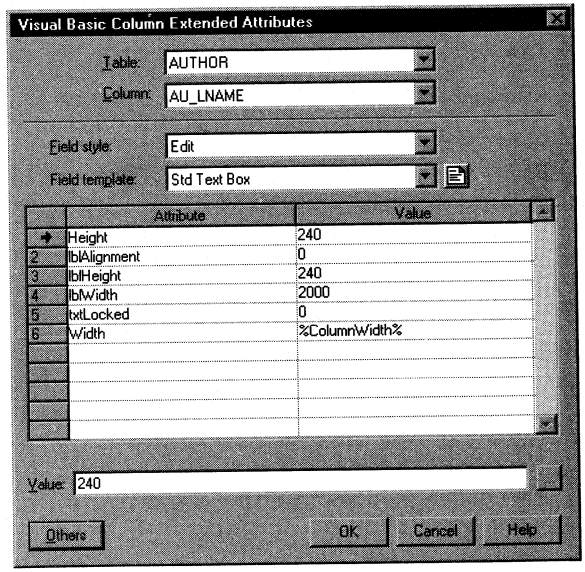
Field styles describe the types of Visual Basic controls you generate in your application. You can use any standard or personalized Visual Basic control that you define in a field template.

Each field template has a list of attributes. You can modify the values of these attributes from VBGen.

❖ To attach a field style and template to a column:

1 Select Client➤Visual Basic Column Attributes.

The Visual Basic Column Extended Attributes dialog box appears.



2 Select a table from the Table dropdown listbox.

3 Select a column from the Column dropdown listbox.

4 Select a style from the Field Style dropdown listbox.

5 Select a template from the Field Template dropdown listbox.

6 Double-click an item in the Attribute column.

The attribute is selected and the cursor moves to the Value box.

7 Type a value for the field style attribute in the Value box.

8 Double-click another attribute and change its value if needed.

> **Using the keyboard**
> You can use the F6 key to switch between the list of attributes and the Value textbox.

9 Click OK.

Attaching a field style and template to a domain

You can attach field styles and templates to domains. These apply automatically to any columns you attach to the domain. You can also update extended attributes for columns already attached to the domain.

❖ To attach a field style and template to a domain:

1 Select Client➤Visual Basic Domain Attributes.

The Visual Basic Domain Extended Attributes dialog box appears.

2 Select a domain from the Domain dropdown listbox.

3 Select a field style from the Field Style dropdown listbox.

4 Select a template from the Field Template dropdown listbox.

5 Double-click an item from the Attribute column.

The attribute is selected and the cursor moves to the Value box.

6 Type a value for the field style attribute in the Value box.

7 Select another attribute and change its value if needed.

8 Click OK.

A message box asks if you want to update the columns attached to the domain.

9 Click Yes.
or
Click No.

Selecting objects to generate

You can generate a Visual Basic application from objects you select in the PDM

Tree view items

The objects that you want to generate must be listed in a tree view in the Application Generation window. This window contains an object list in a tree view with the following items:

- Project object
- Main form object
- Form objects for selected tables, views, and references
  or
  Form objects corresponding to filter definitions that you set in the previous VBGen session
- Columns of each form

> **Your first VBGen session**
> If you do not select any PDM objects and you did not use a selection filter in a previous VBGen session, form objects are listed in the Application Generation window for every table, view, and reference.

Checking items to be generated

Each item in the tree view must be checked to generate the object it represents to your project. You can use the Select All button to check all form and project level items in the tree view, or you can use the Unselect All button to clear these checkmarks.

Generating a menu

If you use the MDIMAIN.VMT template to generate a main form, you also generate a main form menu bar containing five menu items: File, Edit, Topics, Window, and Help.

If you generate your project using the menu option, and you do not change the menu position default (menu position = 3), the data forms you generate are accessible from the Topics menu.

The Edit menu allows a user of your project to cut, copy, and paste information in the data forms. The Window menu provides access to any open data forms and allows you to arrange them in a tile or cascade display.

For more information on data form menu options, see "Selecting form templates and form properties" on page 358.

Generating a project from selected objects

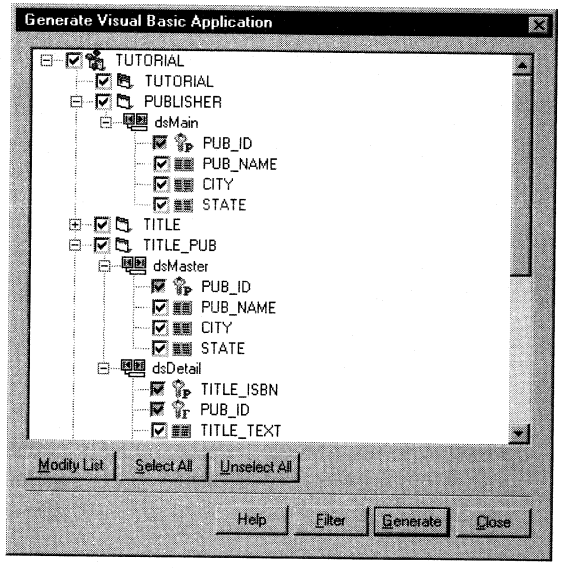
To create a list of objects to generate, you can select objects directly from the PDM or you can reuse objects you selected in the last VBGen session.

❖ To generate a project from selected objects:

1 Select the objects you want to generate.
or
Verify that no objects are selected in the PDM graph to reuse objects selected during the previous VBGen session.

2 Select Client➤Generate Visual Basic Application.

The Application Generation window appears.



3 Click any plus signs to the left of tree view items.

All tree view items are now visible.

4 Verify that all objects you want to generate are checked.

5 Click the Generate button.

Reviewing the generation progress

You can follow the progress of the generation from the log window that appears after the generation begins. VBGen indicates if there are any errors in the generation or if the generation is successful. Before VBGen attempts to generate a file with the same name as an existing file in the project directory, a message box asks you for confirmation.

The project log file, *MODELCODE*.LOG, is created automatically in your project directory. It contains model and generation information. You can read the log file with a text editor.

Modifying the generation selection

There are several ways to modify a preliminary object selection from the Application Generation window. To modify the object selection, you can:

- Set the generation flag for objects in the tree view
- Add objects to the tree view or remove objects from the tree view
- Apply a filter

Setting the object generation flag

A checkmark to the left of an item in the tree view indicates that its generation flag is selected.

Project and form items

By clicking the Select All button beneath the tree view, you can set the generation flags for all project and form items. By clicking Unselect All, you clear the generation flags of all project and form items in the tree view.

Column items

You must expand the tree view to list the column items in the tree view. The Select All and Unselect All buttons do not affect the column items. You can only clear the checkboxes of column items one by one.

You cannot clear the generation flag of a primary key column. The tree view does not display checkboxes for columns in a view or for columns in a reference that use a view as their data source. These columns are always generated when you generate their parent form.

Items sharing data source

Two references can share the same data source (parent table or child table). Changes to column generation flags of the master or detail form of one of the references are automatically reflected in the columns of master or detail forms based on the other reference. However, changing the column generation flag for a table does not affect the generation flag for the column of a master or detail form that uses the table as its data source.

❖ **To set the generation flags for objects in the tree view:**

1 Click the plus signs in front of tree view items.

You expand the tree view.

2 Select the checkboxes in front of the objects you want to generate.

3 Clear the checkboxes of objects that you do not want to generate.

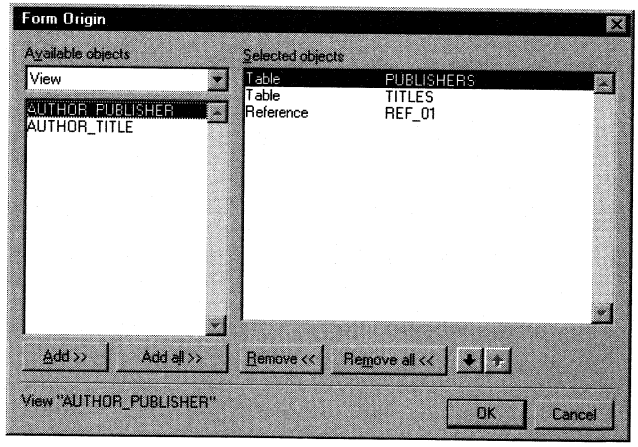
Adding or removing objects from the tree view

If you modify the list of objects in the tree view without using a filter, the modified selections you make are only valid for the current VBGen session. If you close the Application Generation window without generating a Visual Basic project, VBGen does not save your modified selections.

❖ **To add or remove objects from the tree view:**

1 Click the Modify List button below the tree view.

The Form Origin dialog box appears.



2 Select a PDM object type from the Available Objects dropdown listbox.

The Available Objects listbox displays all objects of the type selected that are not in the tree view.

The Selected Objects listbox displays all objects that are in the tree view.

3 Select objects in the Available Objects listbox and click Add.
or
Select objects in the Selected Objects listbox and click Remove.

Selected objects move from one listbox to the other.

> **Using the Add All or Remove All buttons**
> You can click the Add All or Remove All buttons to move all objects of a given type from one listbox to the other.

4 Repeat steps 2 and 3 for other PDM object types.

5 Click OK.

You return to the Application Generation window. The objects you added to the Selected Objects listbox appear in the tree view. Objects you removed from this listbox disappear from the tree view.

Applying a filter to modify object selection

Filters help handle large amounts of information. A large PDM can include hundreds of objects. You can use a filter to augment or restrict the items included in the tree view in the Application Generation window.

After you apply a filter, only the matching items appear in the tree view. The selections you make with a filter are saved by VBGen, whether or not you generate your selections during the current session.

Types of filters

There are four types of filters:

Filter	Action
All objects	Selects every object in the PDM
Objects of submodel	Selects objects from a named submodel
Modified objects	Compares objects in the current PDM to objects in an archived PDM and selects objects that have changed
User-defined list	Lets you select a list of objects from the PDM

You can select or clear object type checkboxes with any of the filter types.

User-defined filter

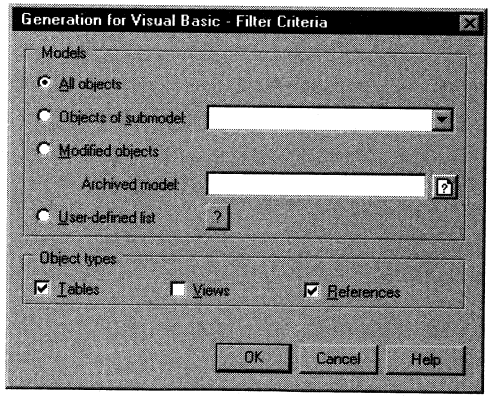
If you use the user-defined filter, you can add any object in the PDM to the tree view. You can use the buttons in the Form Origin dialog box to make customized filter selections:

Button	Function
Add	Transfers a selected object from the Available Objects listbox to the Selected Objects listbox and adds it to the tree view
Add All	Transfers all objects of the selected object type
Remove	Transfers a selected object from the Selected Objects listbox to the Available Objects listbox and removes it from the tree view
Remove All	Transfers all objects of the selected object type
	Moves the selected object up in the list order of Selected Objects and in the tree view
	Moves the selected object down in the list order of Selected Objects and in the tree view

❖ To select objects using user-defined filter criteria:

1 Click the Filter button in the Application Generation window.

The Filter Criteria dialog box appears.



2 Click the  button beside the User-Defined List option.

The Form Origin dialog box appears.

3 Select a PDM object type from the Available Objects dropdown listbox.

The Available Objects listbox displays all objects of the type selected that are not in the tree view.

The Selected Objects listbox displays all objects that are in the tree view.

4 Use the buttons of the Form Origin dialog box to add, remove, or reposition objects in the listboxes and in the tree view generation list.

5 Click OK.

You return to the Filter Criteria dialog box.

6 Verify that checkboxes are selected for the object types that you want to appear in the tree view.

7 Click OK.

You return to the Application Generation window. The tree view displays the new selections you made with the user-defined filter.

Fine-tuning before and after generation

Before you generate or regenerate your application, you can fine tune and test it from the tree view of the Application Generation window as follows:

- Modify model properties
- Modify extended attributes
- Select columns to generate
- Include or remove menu options
- Open the generated project

Modifying model properties

You can display or modify model properties and extended attributes from the context menus of the project or main form tree view items.

❖ **To modify model properties from the tree view:**

1. Right-click the project item in the tree view.
   or
   Right-click the main form item in the tree view.

   The item context menu appears.
2. Select Model Properties.

   The Model Properties dialog box appears.
3. Make changes to the model properties as needed.
4. Click the Extended button.

   The Visual Basic Model Extended Attributes dialog box appears.
5. Make changes to the model extended attributes as needed.
6. Click OK in each of the dialog boxes.

Modifying extended attributes

You can display and modify extended attributes from the context menus of data form items in the tree view. Column extended attributes are also accessible from the tree view.

❖ **To modify extended attributes of a table, view, or reference:**

1 Right-click a data form item in the tree view.

The data form item context menu appears. The context menu selection depends on the data source origin of the data form.

2 Select Table Attributes.
or
Select View Attributes.
or
Select Reference Attributes.

The Visual Basic Extended Attributes dialog box appears.

3 Make changes to the object extended attributes as needed.

4 Click OK.

❖ **To modify extended attributes of a column:**

1 Right-click a , , or column in a table, view, or reference.
or
Right-click a column of a view representing a calculated field.

2 Select Column Attributes.

3 Make changes to the column extended attributes as needed.

4 Click OK twice.

Selecting columns to generate

From the tree view, you can select which columns of a table you want to generate in a data form. You can also select which columns you want to generate in a master or detail data form of a reference if it uses a table for its data source.

You must expand a form and its data source item to list the columns of that form. You expand a tree view item by clicking a plus sign to the left of it.

For more information on selecting individual columns for generation, see "Setting the object generation flag" on page 368.

Including or removing menu options

VBGen generates an application menu based on a main form template. You can select menu options to add to the application menu in object Extended Attributes dialog boxes. These menu options open data forms that you generate with VBGen.

From the Application Generation window, you can choose to generate a menu without menu options. If you generate without menu options, the menu options selections in the object Extended Attributes dialog boxes do not change. These menu options are saved in the PDM and you can use them the next time you generate your application menu.

❖ To include or remove menu options from the tree view:

1 Right-click the main form item in the tree view.

The main form item context menu appears.

2 Select Include Menu Options.

Selecting this item alternatively places or removes a checkmark before the item name. A checkmark indicates that menu options to open data forms will be generated with the main menu.

Opening the project from VBGen

If you generated a project with VBGen, you can test it from the tree view.

❖ To open the project from VBGen:

1 Right-click the project item in the tree view.

The project context menu appears.

2 Select Run Project.

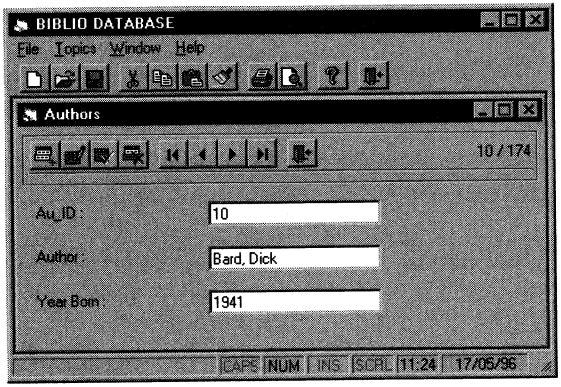
You open the project in Visual Basic, or a message box tells you that Visual Basic is already open.

> **Open Visual Basic file**
> If the open file of Visual Basic is an earlier version of your project file, close the file without saving it, or save it under a different filename. You can then open the project you generated using the File➢Open Project menu in Visual Basic.

Example forms using predefined templates

SINGLE.VFT + TEXTBOX.VCT

The following example shows a form generated using the SINGLE.VFT form template and several instances of the TEXTBOX.VCT field template.

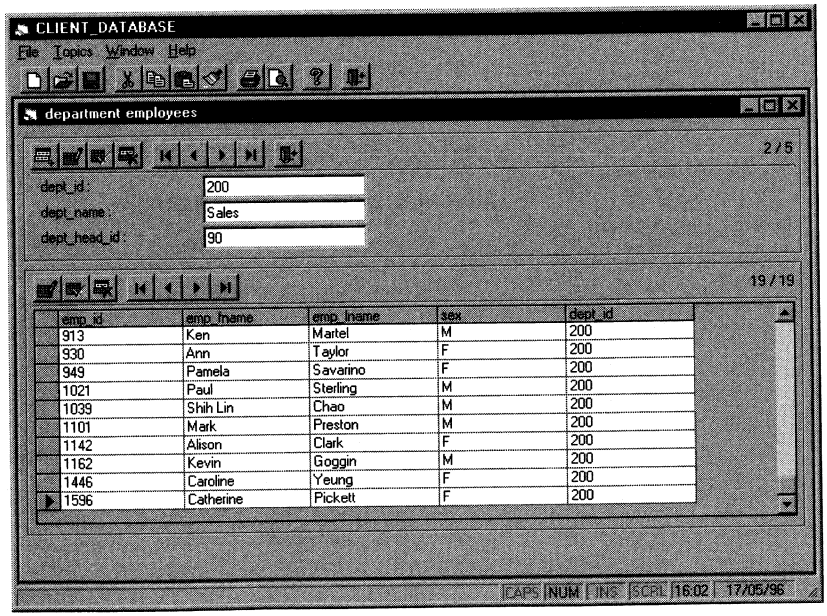


MSTRDETL.VFT + GRID.VGT

The following form was generated using the MSTRDETL.VFT form template.

The master form uses a free form presentation style. The columns of the master form were generated using the TEXTBOX.VCT template.

The detail form uses a grid presentation style. It was generated using the DBGRID.VGT template.



Using templates

You use project and form templates, along with information from PDM objects and extended attributes, to generate Visual Basic project and form files. Field templates define the controls that you can generate in data forms.

For information on previewing templates, see the chapter "Visual Basic Add-In."

What project and form templates define

Project (VPT)

The project template has a VPT extension. Using information from the PDM, it generates a VBP file or an MAK file.

The project template can provide references to resource files, certain OCX or VBX controls, and project window size. It uses the variables %ProjectName% and %ProjectTitle% to refer to values for the project name and title that you enter in the Project name and Project title textboxes of the Visual Basic Model Extended Attributes dialog box.

Main form (VMT)

The main form template has a VMT extension. It generates an FRM file and an FRX file if needed. It refers to a predefined FRM that can contain menu items, image lists, database connection information and a modality attribute (MDI child property in Visual Basic).

Data forms (VFT)

VFT templates provide offset values for the first controls added to generated FRM files. They also refer to predefined FRM files that you can modify directly in Visual Basic. Form templates generate FRM files from PDM tables, views, and references.

VBGen ships with the SINGLE.VFT and F1SINGLE.VFT form templates that you can use to generate forms from PDM tables and views. Use the MSTRDETL.VFT template to generate forms from references.

The form templates that ship with VBGen add database record numbers to the upper right-hand corner of generated forms. These numbers indicate the number of the current record and the total number of records in the database.

Modifying form templates for an SDI application

You can modify the SINGLE.FRM and MSTRDETL.FRM template files to generate SDI applications. By changing the MDIChild attribute, you can generate SDI-type forms instead of MDI-type forms.

Before you modify template files
Make a copy of all template files before you attempt to modify them.

❖ **To modify form templates for SDI applications:**

1 Open SINGLE.FRM in Visual Basic.

2 Change the MDIChild property to False.

3 Save your changes.

4 Repeat steps 1–3 for MSTRDETL.FRM.

You can now select your modified templates in VBGen to generate SDI-type forms.

Removing image lists from data forms

The form templates that ship with VBGen include image lists with each data form they generate. This facilitates development, but could be inconvenient if you want to limit the disk space of generated applications and executables.

Visibility at run time

To reduce the size of the forms that you generate, you can create data forms without image lists by modifying the form templates. A function call to the PDTOOLS.BAS module ensures that the images will still be present at run time. However, they will not be visible at design time if you remove them from the data form templates.

You can remove image lists from SINGLE.FRM and MSTRDETL.FRM directly in Visual Basic, or by editing these template forms using a text editor. If you need to reduce the size of your applications, it is still preferable that you remove the image lists from the generated forms rather than the template forms.

Before you modify template files
Make a copy of all template files before you attempt to modify them.

What field templates define

Field templates define the controls that you generate in Visual Basic data forms. Field templates usually define one bound control, and one label control, for identification of the data that the bound control displays. You attach field templates to columns or domains.

Recommended field styles

Certain field styles are appropriate to columns of a specific data type, and vice versa. Recommended correspondences are:

Data type*	Recommended field style	Template
Boolean	Checkbox	CHECKBOX.VCT *or* CHKBOX2S.VCT
Counter	Label box	LABELBOX.VCT
Integer	Spin control textbox	SPINNUM.VCT
OLE	Bitmap	IMAGE.VCT
Text (limited)	Textbox	TEXTBOX.VCT
Text (unlimited)	Multiline *or* Rich text box	MLTXTBOX.VCT *or* RTFBOX.VCT

* The data type name depends on the DBMS that you use.

Using the dropdown combination box template

The dropdown list field template DBCOMBO1.VCT substitutes displayed information from one column with information from another column in a different table. The tables must be linked by a common key column.

The values of the replacement column are listed in the generated dropdown combination box control. You must assign the column code of the replacement column to the cboListField attribute to use this template.

VBGen instantiates the names of the common key and the source table from default variables that the template defines for the cboBoundColumn and cboRowSource attributes.

Listbox index number

If you use more than one dropdown combination listbox in a form, you must change the cboFieldNo attribute. This attribute assigns an index number to the combination listbox control. You can use a maximum of sixteen dropdown combination listboxes to a form. You can assign the values 1-9 or A-F to cboFieldNo attributes.

Zoom button

The cboZoomForm attribute defines an optional zoom button positioned to the right of the dropdown listbox control. The default value for this attribute is the name of the data form generated from the table containing the replacement column. To open this form using the zoom button, the form must be listed in the project that you generate.

If you assign a null value to the cboZoomForm attribute, the zoom button is not generated.

❖ **To use the dropdown combination listbox template:**

1 Select Client➢Visual Basic Column Attributes.

2 Select a table from the Table dropdown listbox.

3 Select a column from the Column dropdown listbox.

4 Select DropDown from the Field Style dropdown listbox.

5 Select DB Dropdown List from the Field Template dropdown listbox.

6 Double-click the cboListField attribute.

7 Type the name of the column you want to use to replace information from the current column.

8 Click OK.

Using radio button templates

VBGen ships with field templates that define radio buttons enclosed in a frame. These templates differ only in the arrangement of the radio buttons within the frame.

Template	Arrangement
RADIOH.VCT	Radio buttons arranged left to right in a single row
RADIOHV.VCT	Radio buttons arranged left to right in *n* columns
RADIOV.VCT	Radio buttons arranged top to bottom in a single column
RADIOVH.VCT	Radio buttons arranged top to bottom in *n* rows

ListColumn attribute

To change the number of radio button columns generated with the RADIOHV.VCT template, you can change the ListColumn attribute in the Visual Basic Column Extended Attributes dialog box. If you set the ListColumn value to 1, the generated field is the same as if you used the RADIOV.VCT template. The template default value is 3.

ListRow attribute

To change the number of radio button rows generated with the RADIOVH.VCT template, you can change the ListRow attribute in the Visual Basic Column Extended Attributes dialog box.

You define the radio button labels in the list of values of the Check Parameters dialog box for a column or a domain.

❖ To assign radio button labels:

1 Select Dictionary➤List of Columns.

2 Select the column for which you want to generate radio buttons.

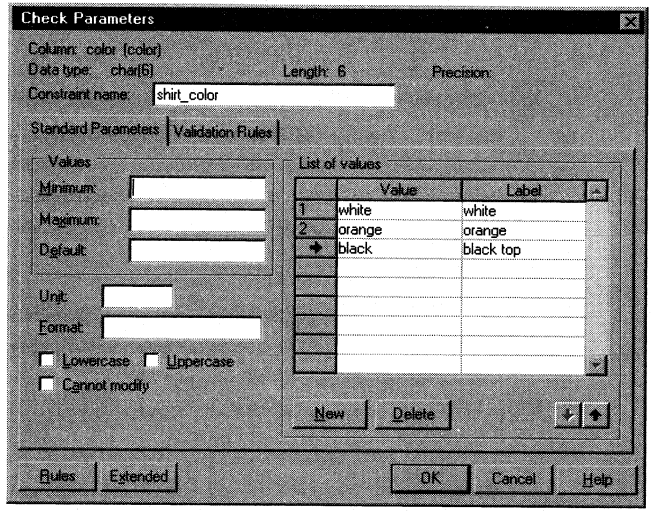
3 Click the Check button.

The Check Parameters dialog box appears.

4 Type the database values for this column in the Values column of the List of Values.

5 Type the labels you want to display in the Labels column.

These become the radio button labels in the forms that you generate with radio button templates.



6 Click the Extended button.

7 Select RadioButtons from the Field Style dropdown listbox.

8 Select a radio button template from the Field Template dropdown listbox.

9 Click OK in each dialog box.

Using the standard dropdown template

VBGen uses the List of Values in the Check Parameters dialog box to instantiate controls it generates using the CHKLIST.VCT template. You must type values and labels in this list to use this template.

The labels you type in the Labels column become list entries in the dropdown listbox that you generate with the CHKLIST.VCT template.

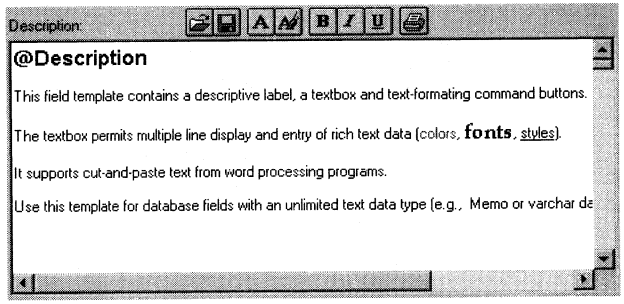
Using the spin button template

VBGen generates the spin button control using the Maximum, Minimum and Default Values that you define in the Check Parameters dialog box for a column. You must type values for these check parameters to use the SPINNUM.VCT template.

Using the rich textbox template

You can only use the RTFBOX.VCT template with the 32-bit version of AppModeler. It generates a separate toolbar above a textbox. The open folder tool in this toolbar enables you to insert text files in the generated form. From the toolbar, you can also print, save to a file, and modify the formatting (fonts, colors, and styles) of the textbox display.

The Description field shown below was generated the RTFBOX.VCT template:



Using a grid presentation style template

If you attach a grid template to a form, field templates attached to PDM columns are not used by VBGen. Column field styles are not generated when the free form presentation style is not selected.

Default grid template

Setup installs DBGRID.VGT as the default grid template. This template is not available for Visual Basic 3.0. If you import the VB3.EXA file, the default grid template becomes TDBGRID.VGT. You still need to install the TrueGrid VCX to use this template with Visual Basic 3.0.

Except for the DBGRID.VGT grid presentation style, all other grid templates that ship with VBGen require installation of a nonstandard grid OCX or VCX.

Formula One grid
The Formula One grid works only with ODBC. It is not available for Visual Basic 3.0. You must only use this grid with a form template that defines a Formula One control, for example, F1SINGLE.VFT. You cannot update a database from a Formula One grid.

Viewing template information

You can view template descriptions and other template information directly from VBGen. The type of information available depends on the type of template you select for viewing.

From VBGen you can access information for the following template types:

Type	Accessibility in VBGen from	Viewable sections
Project	Model Extended Attributes	Description Modules
Main form	Model Extended Attributes	Description MenuScript
Data form	Table Extended Attributes View Extended Attributes Reference Extended Attributes	Description Screen
Computed Fields	View Extended Attributes	Description Controls
Field	Column Extended Attributes Domain Extended Attributes	Description Controls Attributes
Grid	Table Extended Attributes View Extended Attributes Reference Extended Attributes	Description Controls

Modifying template information

You can modify the template information using a text editor.

Before you modify template files
Make a copy of all template files before you attempt to modify them.

If you change information in a template section that is viewable in VBGen, the modifications you make appear in the Show Template dialog box.

For information on using the AppModeler for Visual Basic Add-In to view and modify template sections, see the chapter "AppModeler for Visual Basic Add-In".

❖ To view template information:

1 Click the  show template button in a VBGen dialog box.

2 Select a template from the Template Name dropdown listbox.

3 Select a template section to view from the Show Section dropdown listbox.

The section you select is displayed in a scrollable window.

4 Click OK.

Customizing templates and template sets

You can create your own templates from existing projects and then add template path information directly to the registry or the PD6.INI file from VBGen.

From VBGen, you can also create a customized template set by adding the set name to the Template Sets registry subkey or the Template Sets section of the PD6.INI file.

> **Using predefined templates**
> If your project uses the VBGen templates that ship with AppModeler, you do not need to add customized templates or template sets.

Creating project and form templates

You can create your own project templates and form templates to use with VBGen. You can save the template files that you create to a new template set directory.

With the AppModeler for Visual Basic Add-In, you can preview form templates that you create. You can also use the Add-In to create or modify templates.

For information on using the AppModeler for Visual Basic Add-In, see the chapter "Visual Basic Add-In."

❖ To create project and form templates:

1. Create and save a working Visual Basic project.
2. Using a text editor, replace names and identifiers in your project and form files with AppModeler system variables.

   You use variables for names and identifiers that you want VBGen to instantiate from the PDM.

   For information on AppModeler system variables, see the appendix "Generation Variables and Attributes."
3. Copy and rename the PROJECT.VPT project template, keeping the VPT extension.
4. Make changes to the new project template as needed.
5. Rename your Visual Basic VBP project file using the filename prefix of your VPT project template.

6 Copy and rename the MDIMAIN.VMT main form template, keeping the VMT extension.

7 Make changes to the new main form template as needed.

8 Rename your main form FRM and FRX files with the filename prefix of your VMT template.

9 Copy and rename the SINGLE.VFT file, the F1SINGLE.VFT file, or the MSTRDETL.VFT file, keeping the VFT extension.

10 Make changes to the new form template as needed.

11 Rename your form FRM and FRX files with the filename prefix of your VFT template.

Creating field templates

You create field templates by placing controls in an open Visual Basic form. You can isolate the code for the controls and save it to a new file that you name with a VCT extension. The controls must be included in a @Controls section of the VCT field template. You can replace the names in a field template with system or user-defined variables.

Using variables in field templates

VBGen can instantiate an attribute that is defined by a variable in the @Attributes section of a field template. If you make reference to this attribute in other sections of the template, the attribute must be surrounded by percent signs.

%FieldCode% variable

If you create your own field templates, you can name the template controls using the %FieldCode% variable. VBGen can instantiate the %FieldCode% variable of a control identifier with an index number that it concatenates to the column code. This ensures that each control is unique, even if the column code is used more than once in the same form.

For example, a label control description from the TEXTBOX.VCT field template uses the %FieldCode% variable:

```
BEGIN VB.Label lbl_%FieldCode%
   Caption     =  "%ColumnCode%:"
   Left        =  100
   Top         =  16
   Width       =  %lblWidth%
   Height      =  %lblHeight%
   Alignment   =  %lblAlignment%
END
```

The label control name is lbl_*%FieldCode%*. VBGen replaces %FieldCode% by the column code, and an index number, if necessary, to differentiate two labels for the same column (for example, in master/detail forms where the column names are the same for fields in the master and detail sections).

%ColumnWidth% variable

The TwipsPerChar attribute in VBGen data form templates defines this constant as 150 Visual Basic units (twips) per character.

The TEXTBOX.VCT field template uses the %ColumnWidth% variable as the default value for the textbox control it defines:

```
lblWidth 2000 ' Description label width
lblHeight 240 ' Description label height
lblAlignment 0 ' Alignment of text in label box
Width %ColumnWidth% '(computed) Width of textbox
Height 240 'Height of textbox
Left 2100 'Textbox position (relative to label)
```

Using switches in variable names

Undefined variables

If you type a **?** after the first percent sign in a variable name in a field template, VBGen only generates the line of code containing the variable if the variable is not null. The variable can be a system variable or a user-defined variable.

For example, the template description of a textbox control shown below uses the system variable %?ColumnLength%. If the PDM does not have a value for the length of a column you generate with this template, VBGen does not generate the MaxLength line that contains this variable.

```
Begin VB.TextBox txt_%FieldCode%
   DataField     = "%DataField%"
   DataSource    = "%DataSource%"
   Left          = > 30
   Top           = 10
   Width         = %Width%
   Height        = %Height%
   Text          = "%.Q:ColumnName%"
   MaxLength     = %?ColumnLength%
End
```

Formatting variables

You can use the special formatting switch **.Q:** in a variable to protect quotation marks in objects you generate with VBGen. As with other formatting switches, you insert it after the first percent sign of a variable name.

At generation, VBGen adds a duplicate quotation mark to the text generated from the line containing the formatting switch. Visual Basic can then recognize quotation marks in the text as literal characters.

For information on other formatting switches that you can use with AppModeler variables, see the chapter "Database Creation and Modification."

Adding a template or a field style

Adding a template

You can add project, form, field, or grid templates to the registry or PD6.INI file directly from VBGen. You must include a template name and filename for each template that you add.

You must select a template type for each template that you add, and you must type a template name and filename. For the 32-bit version of VBGen, the new template name is included as a string value in the registry under the template type subkey you select. For the 16-bit version of VBGen, the new template name is included under the template type subsection of the PD6.INI file.

Adding a field style

For each field style that you add, you must include a field style name and a default filename. After you validate the new field style name, the new field style appears in the list of template types for your current template set. You can then add other field templates for this template type, in addition to the template that you selected as the field style default.

For long lists of field styles, a scroll bar in the Type dropdown listbox permits you to view all available VBGen template types.

Your field style name appears as a string value in the registry under the Field Styles subkey or in the Field Styles section of the PD6.INI file. The default template filename appears as a data value for the new field style name.

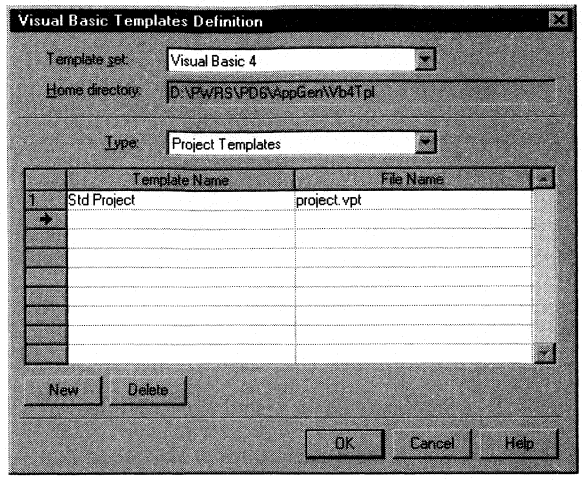
❖ **To add a template or a field style:**

1 Select Client➤Visual Basic Model Attributes.

The Visual Basic Model Extended Attributes dialog box appears.

2 Click the Templates button.

The Visual Basic Templates Definition dialog box appears.



3 Select the template set to which you want to add a template or field style.

Only template sets listed in your registry or PD6.INI file are included in the Template Set dropdown listbox.

4 Select a template type from the Type dropdown listbox.
or
Select Field Styles from the Type dropdown listbox.

5 Click the New button.

An arrow appears at the beginning of the first blank line.

6 Type a name in the Template Name column.
or
Type a name in the Field Style Name column.

7 Click the File Name column for your new template.
or
Click the Default File Name column for your new field style.

A ... button appears in the column.

8 Click the ... button to browse available drives.
Select a filename and click OK.
or
Type a filename in the column.

You can type the complete path and filename, or you can use the relative path from the template set home directory.

9 Click OK.

Adding a template set

The AppModeler Setup program installs template sets for the Visual Basic 3.0 version and the 16-bit and 32-bit versions of Visual Basic 4.0. You can create your own template sets to access custom-designed VBGen templates.

You can select a home directory for each template set and use relative addresses for your individual template files. When you add templates to a template set, you can always use a full address for the location of a template file that is not in the template set home directory.

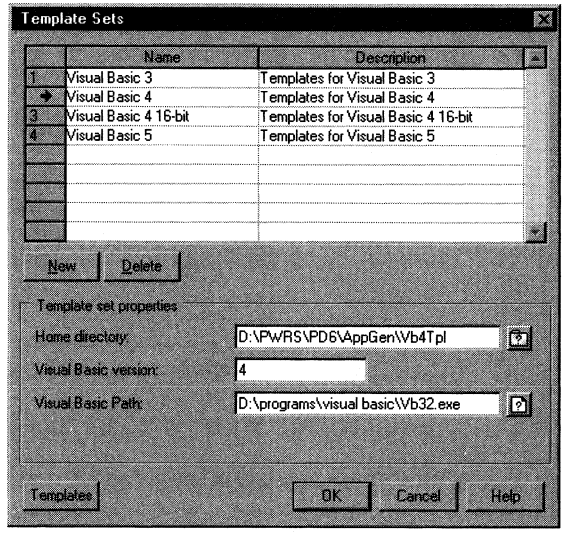
❖ To add a template set:

1 Select Client➤Visual Basic Model Attributes.

The Visual Basic Model Extended Attributes dialog box appears.

2 Click the button next to the Template Set box.

The Template Sets dialog box appears.



3 Click the New button.

The arrow moves to a new row in the list of template sets.

4 Type a name for your new template set.

5 (Optional) Type a description for the template set.

6 Click the button next to the Home Directory box.
Select the Home directory for your template set and click OK.
or
Type the directory for your template set in the Home Directory box.

7 Type the Visual Basic version you want to use with your template set.

You can type 4.0 for the 16-bit and 32-bit versions of Visual Basic 4.0.

8 Click the button next to the Visual Basic Path box.
Select your Visual Basic Path and click OK.
or
Type the path and filename in the Visual Basic Path box.

9 Click OK.

Your template set is added as a new string value to the Template Sets registry subkey or INI file section, and as a new key or new section under the AppModeler for Visual Basic.

CHAPTER 16

Visual Basic Add-In

About this chapter

This chapter describes how to use the AppModeler for Visual Basic Add-In to create or modify VBGen templates.

Contents

Before you begin

AppModeler ships with two versions of the AppModeler for Visual Basic Add-In. One version of the Add-In is for use with the 32-bit version of Visual Basic 4.0. The other version is for use with Visual Basic 5.0.

The AppModeler Setup program copies both versions of the Add-In to your hard drive.

For more information on VBGen templates, see the chapter "Visual Basic Generator." For more information on VBGen variables and extended attributes for Visual Basic, see the appendix "Generation Variables and Attributes."

Using the AppModeler for Visual Basic Add-In

The AppModeler for Visual Basic Add-In appears in the Visual Basic Add-Ins menu after you select it from the Visual Basic Add-In Manager. You can display the Add-In as a tool bar or as a form containing a tree view.

Including the Add-In in the Visual Basic Add-Ins menu

If you previously installed a 32-bit version of Visual Basic, the AppModeler Setup program registers a version of the AppModeler for Visual Basic Add-In with the Visual Basic Add-In Manager. Otherwise, you must register the Add-In by running one of the following executable files:

Visual Basic version	Add-In executable file
4.0 32-bit	PDVB4AD.EXE
5.0	PDVB5AD.EXE

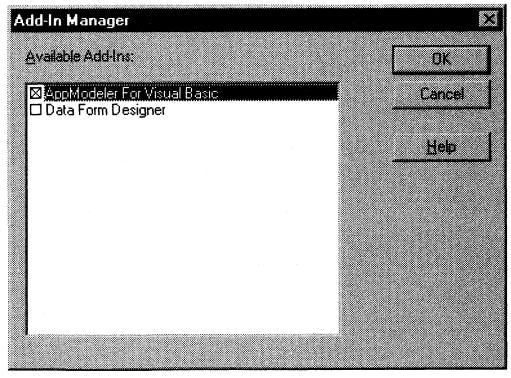
After the Add-In is registered with the Add-In Manager, you must add the AppModeler for Visual Basic Add-In to the Visual Basic menu. You only need to do this the first time you start the Add-In.

❖ To include the Add-In in the Visual Basic Add-Ins menu:

1 Open Visual Basic.

2 Select Add-Ins➤Add-In Manager.

The Add-In Manager dialog box appears.



3 Select the checkbox beside AppModeler for Visual Basic in the Available Items listbox.

4 Click OK.

AppModeler for Visual Basic is now an item in the Add-Ins menu.

Previewing VBGen templates

The AppModeler for Visual Basic Add-In can appear as a toolbar or as a form containing a tree view. The Add-In tree view shows the registry items listing VBGen templates.

Selecting a template set

AppModeler ships with template sets for different versions of Visual Basic. From the Add-In, you can modify templates for any AppModeler for Visual Basic template set, but you can only preview templates for a 32-bit version of Visual Basic.

You can use the Add-In to preview form and field templates in Visual Basic. You cannot preview project templates. If the template contains system variables, you can assign values to these variables before you preview them.

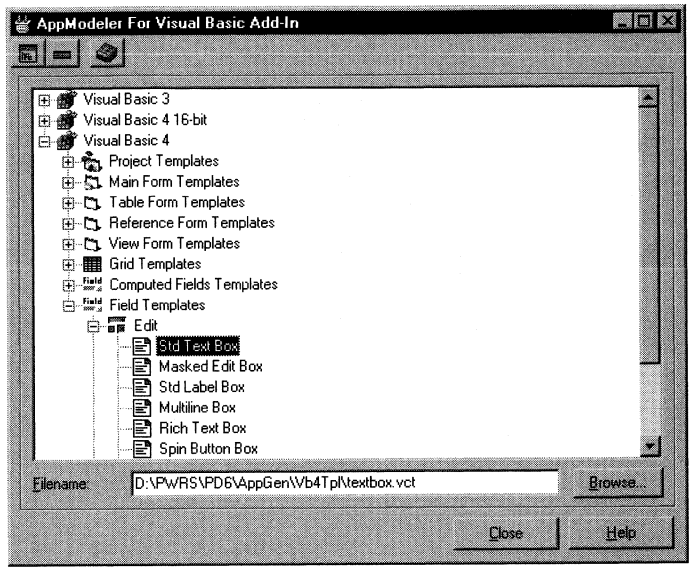
For information on assigning values to system variables from the Add-In, see "Assigning a value to a system variable" on page 409.

❖ **To preview VBGen templates from the Add-In:**

1 Select Add-Ins➤AppModeler for Visual Basic from the Visual Basic menu bar.

2 If the Add-In displays as a tool bar, click the  button.

The AppModeler for Visual Basic Add-In displays as a form containing a tree view.

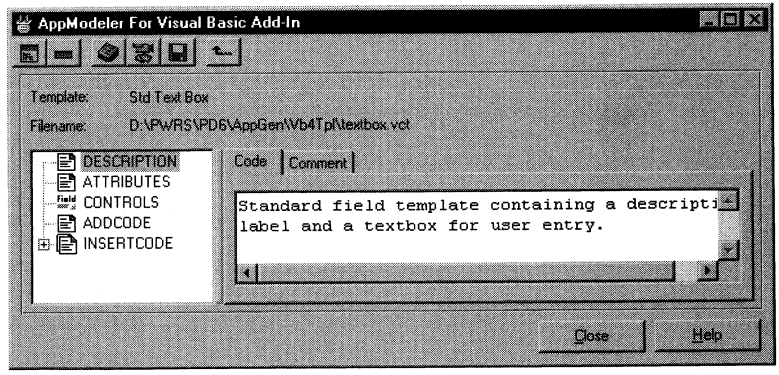


3 Double-click a template set in the tree view.
Right-click the ⊞ signs under the template set.

You expand the tree view to show the templates in the template set you selected.

4 Double-click a icon for the template you want to display.

The template definition window appears. The name and the filename of the template appear in boxes at the top of the window.



The template sections pane at the left of the window lists all the sections of the template. The tab control pane at the right shows the contents of any section you select in the template sections pane.

The template sections pane and the tab control pane are separated by a movable splitter bar.

5 Select a template section.

The template sections available in the listbox depend on the template you selected. Code and comments in the template section you selected appear in the tab control pane.

6 (Optional) Make changes to the template code or comments.

You can use the Add-In to check the syntax of changes you make to the template code.

For information on checking code syntax, see "Verifying the syntax of template code" on page 407.

7 Click the  preview button.

Visual Basic creates a temporary form showing the template that you selected. The Add-In loses the focus and folds up into a toolbar.

8 Click the  AppModeler for Visual Basic Add-In icon in the taskbar.

The focus returns to the Add-In. It appears as a toolbar.

9 Click the  preview button a second time.

You close the temporary form in Visual Basic, and the focus returns to the Add-In. The Add-In displays as a full length form.

10 Click the  button.

If you made any changes to the template, a message box asks if you want to save the changes. You can click Yes to save the changes or click No to cancel the changes. You return to the Add-In tree view.

Using the Add-In to modify the registry

You can use the Add-In to add or remove AppModeler for Visual Basic templates, template sets, and field styles. The registry reflects the changes you make from the Add-In.

Adding or removing a template set

AppModeler installs template sets corresponding to different Visual Basic versions. Template sets are the top-level items in the Add-In tree view.

If you use the Add-In to add a template set, the new template set key in the registry contains empty subkeys for project templates, form templates, field templates, computed field templates, and grid templates.

❖ **To add or remove a template set:**

1 Select Add-Ins➤AppModeler for Visual Basic from the Visual Basic menu bar.

The AppModeler for Visual Basic Add-In appears.

2 Click the  maximize button, if the Add-In appears as a tool bar.

3 Right-click a  template set item.

These are the top-level items in the tree view. You open a context menu when you right-click an item in the tree view.

4 Select New Target from the context menu to add a template set.
Type a name for the template set.
or
Select Remove from the context menu to remove the template set.

If you selected Remove, a confirmation box appears. Click Yes to remove the template set and all template references contained in the set.

Adding or removing a field style

The Field Styles registry key lists the types of field templates available for generation of Visual Basic controls. From the Add-In tree view, you can add and remove field styles from the registry.

The Add-In creates a default field template for each field style you add. Unless you type a filename in the Path box for the default field template, the Add-In assigns the name NEWFIELD.VCT for the default template file. The Add-In also creates headings and sections in the new field template.

Using the default path
If you type a filename, but not a path, you save your template in the directory indicated on the General Page of the Options dialog box. You click the  button in the toolbar to access this dialog box.

❖ **To add a field style:**

1 If the Add-In tree view is contracted, double-click the icon to the left of a template set in the tree view.

You expand the tree view.

2 Right-click the Field Templates item in the tree view.

The Field Templates item is one level below a template set item. You open the Field Templates context menu.

3 Select New Field Style from the context menu.

A new field style item is added beneath the Field Templates item in the tree view. It is selected.

4 Type a name for your new field style.

Your new field style is entered in the registry.

5 Click the ⊞ sign that appears to the left of your new field style.
or
Double-click your new field style.

The tree view expands to show a default template.

6 Type a path and filename of a VCT template in the Path box.
or
Click the Browse button to open a file selection dialog box.
Select a VCT file for the default template.

❖ **To remove a field style:**

1 If the Add-In tree view is contracted, double-click the icon to the left of a template set in the tree view.

You expand the tree view.

2 Click the ⊞ sign to the left of the Field Template item.

You display the field styles for your template set.

3 Right-click the field style you want to remove.

4 Select Remove Field Style.

A confirmation box appears.

5 Click Yes.

You remove the field style and all its templates from the registry.

Adding, removing, or renaming a template

You can add a project, form, grid, or computed fields template from second-level items in the Add-In tree view. Second-level items are items one step below the template set item.

You add field templates from third-level tree view items. Template extensions for the different template types are indicated in the table that follows:

Template type	Extension
Project Template	VPT
Main Form Template	VMT
Table Form Template View Form Template Reference Form Template	VFT
Grid Template	VGT
Field Template Computed Field Template	VCT

The Add-In creates a file for your template if you type a new filename in the Path box under the Add-In tree view.

> **Using the default path**
> If you type a filename, but not a path, you save your template in the directory indicated on the General Page of the Options dialog box. You click the  button in the toolbar to access this dialog box.

❖ To add a template:

1. If the Add-In tree view is contracted, double-click a  icon to the left of a template set item in the tree view.
2. Right-click a second level item to which you want to add a project, form, grid, or computed fields template.
   or
   Click the  sign to the left of the Field Template item.
   Right-click a  field style item.

   A context menu appears.
3. Select New Template from the context menu.

   The new template item appears in the tree view under the tree view item that you right-clicked. It is selected.
4. Type a name for the new template.

   The new template name is entered in the registry.

5 Type a path and filename in the Path box.
or
Click the Browse button to open a file selection dialog box.
Select a file and click OK.

❖ **To remove or rename a template:**

1 If the Add-In tree view is contracted, double-click a template set item.
Click the ⊞ sign to the left of second-level and third-level items.

2 Right-click the icon to the left of the template you want to remove or rename.

The template context menu appears.

3 Select Remove from the context menu to remove the template.
or
Select Rename from the context menu and type a new template name.

Editing template sections

The main purpose of the AppModeler for Visual Basic Add-In is to help edit templates for use with VBGen. You can type changes to a template on the Code page of the Add-In, or you can select controls in a Visual Basic form and add the controls and their attributes to the template. You can also add InsertCode and InsertColumnCode sections to certain templates.

The Add-In can display errors in syntax for the code you add to a template, in a color of your choice. You can view the list of AppModeler system variables and assign default values to these variables from the Add-In.

Creating controls and attributes in a field template

You can use the Add-In to add any controls that you select in a Visual Basic form. You add the controls to the Controls section of a VBGen field template. You can rename the controls using AppModeler system variables.

Control names

If you use ¦ characters to enclose part of a control name in Visual Basic, the Add-In converts the enclosed part of the control name to a variable surrounded by percent signs. Otherwise, the Add-In attaches the %FieldCode% variable to the name of the control in Visual Basic, before you add it to a VBGen template.

VBGen can instantiate the %FieldCode% variable with an index number that it concatenates to the column code. This ensures that each control is unique, even if the column code is used more than once in the same form.

Control properties

You can also select attributes to add to the Attributes section of your template. The attributes page of the List of Attributes dialog box lists properties of the Visual Basic controls that you selected. Only one attribute value is listed for a property that is shared by more than one control.

Custom control libraries

If you add a custom control, you must add the corresponding custom control library to the Include section of your field template. The Add-In adds custom control libraries automatically if you select them from a list of available libraries.

> **If you are editing an existing template file**
> Controls and attributes that you add to an existing template replace any controls and attributes previously listed in the Controls and Attributes sections of the template. No changes are made to the Description and InsertCode sections of a template when you add controls and attributes.

The table below shows the items added to your template after you click OK from the List of Attributes dialog box.

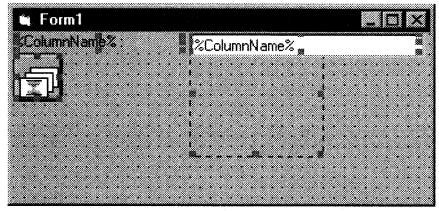
Template section	Item added from List of Attributes dialog box
AddCode	Code attached to the controls listed on the Controls page
Attributes	Attributes that you select on the Attributes page
Controls	All controls listed on the Controls page
Include	Custom control libraries you select on the Custom Control Libraries page

❖ To create controls and attributes in a field template:

1 Select the controls in a Visual Basic form that you want to create in a template.

Handles appear around the controls in the form.

For example, the form below has four selected controls: a label, a textbox, a listbox, and an image list. The textbox is not visible at runtime.



2 From Visual Basic, select File➤Save Project.

You can skip this step if you are not adding a custom control to your template.

3 From the Add-In tree view, double-click a new field template to which you want to add the controls.

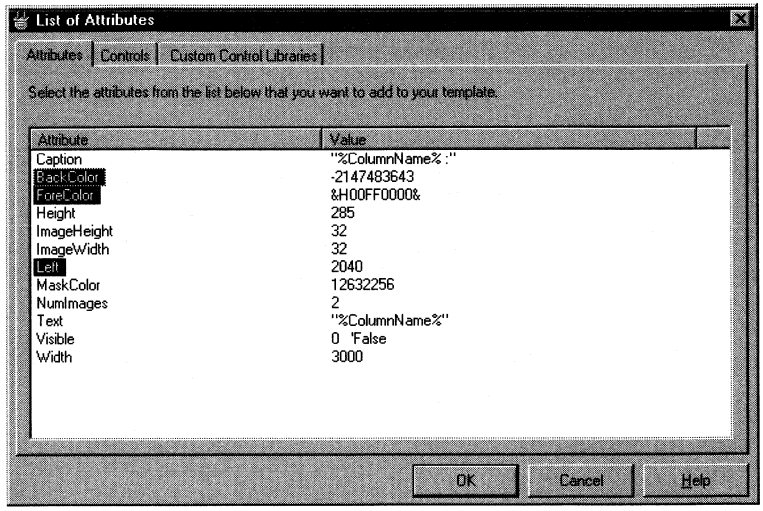
The Add-In displays the sections of the template you selected.

4 Right-click a section in the template sections pane.

A context menu appears.

5 Select Create From Visual Basic from the context menu.

The List of Attributes dialog box appears.

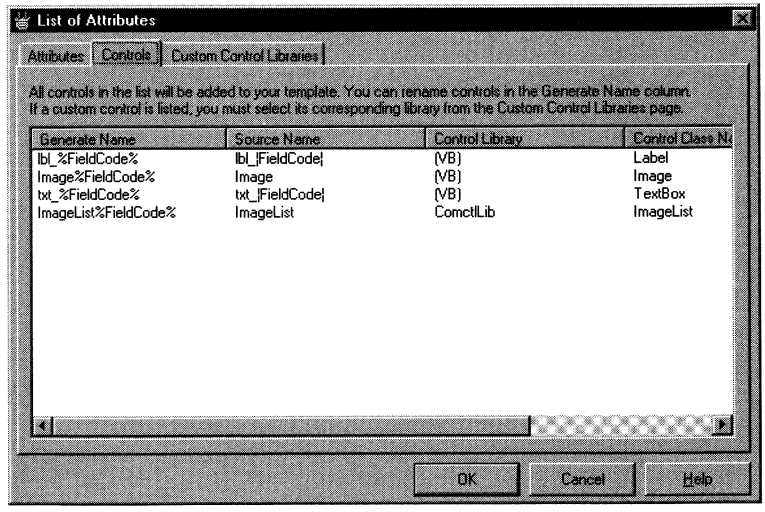


6 Select the attributes from the list that you want to add to your template.

Only attributes you select will be added to the template. Attributes previously listed in the template will be erased.

7 Click the Control tab.

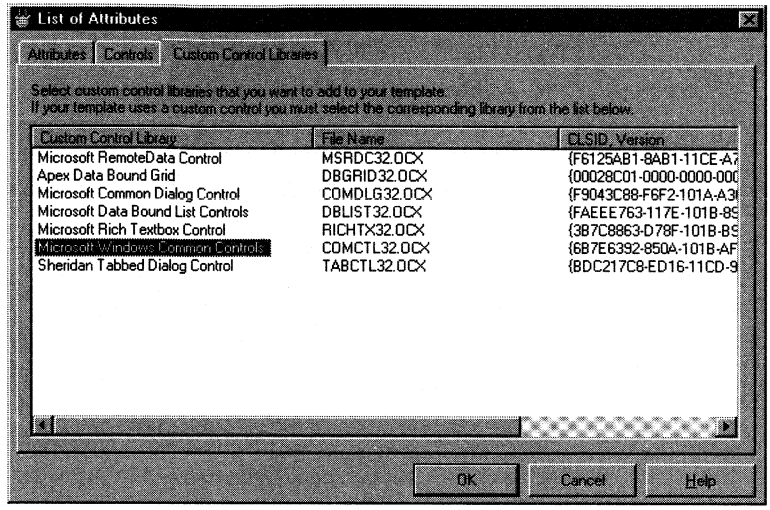
The Control page opens.



8 (Optional) Type a new name for each control in the Generate Name column.

9 Click the Custom Control Libraries tab.

The Custom Control Libraries page opens.



10 Select custom control libraries to add to your template.

You must select all libraries that correspond to custom controls listed on the Controls page of this dialog box.

11 Click OK.

A message tells you that the controls have been added to your template.

12 Click OK.

You return to the template sections pane of the Add-In.

13 Click the  button.

You save the changes you made to the template.

14 Click the  button.

You return to the Add-In tree view.

Adding InsertCode and InsertColumnCode template sections

InsertCode

You can add an InsertCode section to all field templates, computed field templates, and grid templates. If an InsertCode section already exists, you can create additional InsertCode subsections.

An InsertCode subsection specifies code in a field template that VBGen adds to the form template. The code is added to an InsertCode subsection having the same name as the InsertCode subsection of the field template, just before generation of a form.

InsertColumnCode

For grid templates, you can also add an InsertColumnCode section and subsections. VBGen adds code specified in an InsertColumnCode section to the form template, as many times as there are columns in the grid attached to the template. The code is added to the form InsertCode subsection that has the same name as the InsertColumnCode subsection of the grid template.

❖ To add an InsertCode template section:

1. Double-click a field template, computed field template, or grid template from the Add-In tree view.

   The template definition window appears.

2. Right-click an item in the template sections pane.

   A context menu appears.

3. Select Add InsertCode from the context menu.

   If an InsertCode section did not already exist in the template, an InsertCode section and subsection are added.

   If an InsertCode section already existed in the template, an InsertCode subsection is added.

4. Type a name for the new InsertCode subsection.
5. Add code to the Code page for the new InsertCode subsection.
6. (Optional) Add a comment to the Comment page for the InsertCode subsection.
7. Click the  button.

   You save your changes.

❖ To add an InsertColumnCode template section:

1. Double-click a grid template from Add-In tree view.

   The template definition window appears.

2. Right-click anywhere in the template sections pane.

   A context menu appears.

3 Select Add InsertColumnCode from the context menu.

If an InsertColumnCode section did not already exist in the template, an InsertColumnCode section and subsection are added. If an InsertColumnCode section already existed in the template, an InsertColumnCode subsection is added.

4 Type a name for the new InsertColumnCode subsection.

5 Add code to the Code page for the new InsertColumnCode subsection.

6 (Optional) Add a comment to the Comment page for the InsertColumnCode subsection.

7 Click the  button.

Verifying the syntax of template code

The AppModeler for Visual Basic Add-In can help you to verify the syntax of any changes you make to template code. The Add-In can display attributes, system variables, normal text, and code syntax errors in different colors upon demand.

❖ **To verify the syntax of template code:**

1 Double-click a template from the tree view.

2 Select a template section to verify.

3 If the Comment tab has the focus, click the Code tab.

4 Right-click anywhere in the Code page.

5 Select Coloration from the context menu.

The Add-In colors the information in the Code pane. If you have not changed the default colors, syntax errors display in red.

Selecting a color for code and text

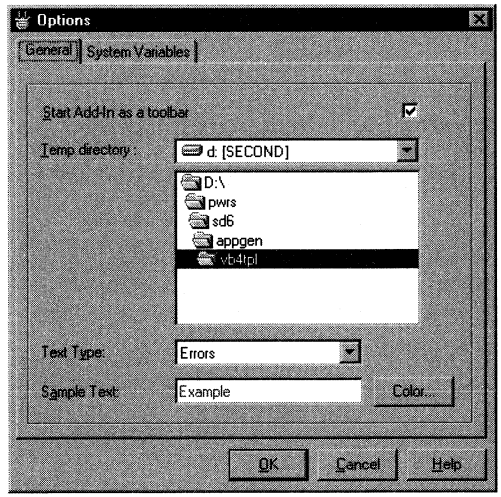
You can change the colors that the Add-In uses to display template text and code. The default colors are listed in the table below:

Syntax element	Default color
Attributes	Green
System variables	Blue
Errors	Red
Normal text	Black

❖ To select a color for code and text:

1 Click the  button from the Add-In tool bar.

The Options dialog box appears.



2 Select a syntax element from the Text Type dropdown listbox.

3 Click the Color button.

A standard color palette appears.

4 Select the color you want for the element you selected.

5 Click OK.

The text in the Sample Text box changes to the color you selected.

6 Click OK.

The Options dialog box closes. Your changes are saved.

Assigning a value to a system variable

You can assign a value to an AppModeler system variable from the AppModeler for Visual Basic Add-In. The values you assign are only for use in previewing your templates from the Add-In. They are not transferred to VBGen.

For information on previewing your template from the Add-In, see "Previewing VBGen templates" on page 395.

❖ To assign a value to a system variable:

1 Click the  button from the Add-In tool bar.

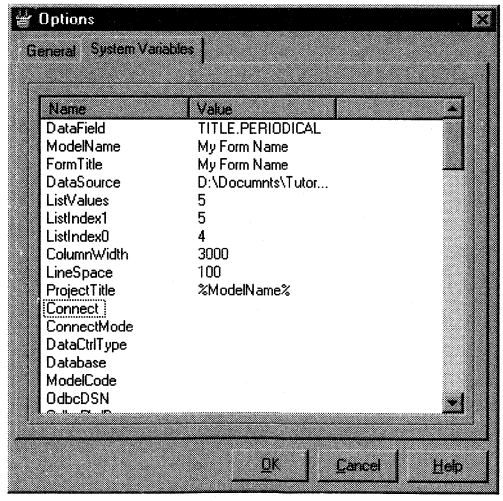
The Options dialog box appears.

2 Click the System Variables tab.

The System Variables page opens.

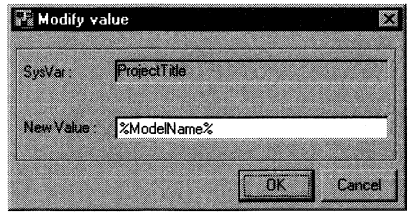
3 (Optional) Click the Value column.

You arrange the system variables by the Value column instead of the Name column. If you click the Value column a second time, you arrange the variables by the Value column, but in descending order rather than ascending order.



4 Double-click a system variable in the list.

The Modify Value dialog box appears.



5 Type a value in the New Value box.

6 Click OK in each of the dialog boxes.

Assigning user-defined variables to a field template

If you substitute your own variables for fixed values in the Controls section of a template, you must also list these variables in the Attributes section.

Variable syntax

Variables in the Controls section are enclosed by percent signs (%). In the Attributes section, you do not use the percent signs. However, if you assign a system variable as a value for a user-defined variable in the Attributes section, you must use percent signs around the system variable.

In VBGen, field template attributes appear in the list of attributes of the Visual Basic Column Extended Attributes dialog box. You can view comments that you add to a template from the Show Template dialog box.

❖ **To assign user-defined variables to a field template:**

1 Double-click a field template from the Add-In tree view.

The Add-In form displays the sections of the template you selected.

2 Select the Controls section.

3 If the Comment page is open, click the Code tab.

4 Select the value of the control property to which you want to assign a variable.

In the example below, the value *picture* is selected in the code for the Caption property of the btn_%FieldCode% control.

```
Begin VB.CommandButton  btn_%FieldCode%
   Caption          =   "picture"
   Height           =   495
   Left             =   %Left%
   Top              =   1200
   Width            =   615
End
```

5 Type the variable name surrounded by percent signs.

You can use a variable name that indicates the control property. For example, you can use %btnCaption% to identify the Caption property for a button control.

```
Begin VB.CommandButton  btn_%FieldCode%
   Caption          =   "%btn_Caption%"
   Height           =   495
   Left             =   %Left%
   Top              =   1200
   Width            =   615
End
```

Because you have not yet defined the variable as an atttibute, if you select Coloration from the Code page context menu, the variable name receives the color defined for a syntax error.

6 Select the Attributes section of your template.

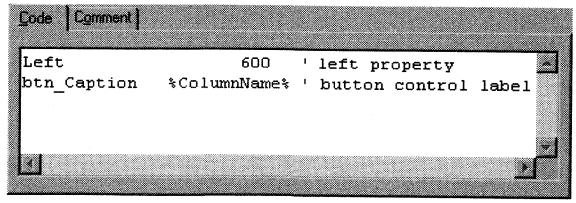
7 Type the variable name on a new line without percent signs.

The variable becomes a template attribute. If there are other attributes in this section, you can arrange the attributes in the order that you want them to appear in the list of attributes of the Visual Basic Column Extended Attributes dialog box in VBGen.

8 (Optional) Type several blank spaces and a default value after the attribute that you added.

If you use a system variable for a default value, you surround the value by percent signs.

9 (Optional) Type a single quote mark followed by a comment that explains the attribute function.



10 Click the  button.

Your changes are saved.

11 Click the  button.

You return to the tree view.

How to use template sections and symbols

The AppModeler for Visual Basic Add-In displays all the sections of VBGen templates. Each section of a template is separated from the preceding section by ## signs. Each section starts with the @ character.

Other special signs can affect the way AppModeler generates a project.

VBGen template sections

The table below lists the different possible sections of a VBGen template and gives a brief description.

Template section	Description
Include	Lists complementary tools that must be loaded for the Visual Basic project to function correctly*
Description	Explains the template function and behavior
Attributes	Lists the user-defined variables that VBGen can instantiate during generation of a Visual Basic project
Controls	Defines the appearance of controls in a form generated by VBGen
AddCode	Adds code defined in a field template to a form in which the field template is used. The code is added after any existing code
InsertCode	Adds code at precise locations, usually after a reference in a form template
InsertColumnCode	Repeats the InsertCode sequence for each column of a table, view, or reference
Master	Defines the appearance of a grid control in a master form
Detail	Defines the appearance of a grid control in a detail form
MasterField	Provides the master field qualifier for synchronizing master and detail forms
Screen	Defines offsets for the first fields added to a form
MenuScript	Defines the main form and sheet form menus
Modules	Lists modules to be copied to the generated project

* Uses the keyword Object, followed by an equals sign, and then by the complementary tools required. Multiple tools or OCX objects are separated by semicolons.

Using symbols in template code

Certain symbols have a special meaning in VBGen templates.

Symbol	VBGen meaning
>	Assigns a relative reference for the placement of a control
@	Defines the beginning of a template section
.Q:	Adds duplicate quotation marks to the text generated, enabling Visual Basic to interpret quotation marks in the text as literal characters*
%	Surrounds a system or user-defined variable
?	Indicates that a line of code will not be generated if the instantiated value of a system variable is null*
##	Separates template sections

* This sign must precede the variable name, but remains inside the percent signs that surround the variable. For example, %.Q:ColumnName% or %?ListValues%.

CHAPTER 17

Power++ Generator

About this chapter

This chapter provides an overview of the AppModeler Power++ Generator (P++Gen) and explains how it works.

Contents

Before you begin

Optima++, Power++, and AppModeler for Power++ are only available in 32-bit versions. AppModeler ships with templates for Optima++ 1.5.

Generator basics

P++Gen is the AppModeler generator for Power++. It generates Power++ projects, forms, DataWindow objects and controls from objects and attributes that you define in a PDM. AppModeler ships with predefined templates that you can use with P++Gen for Power++ object generation.

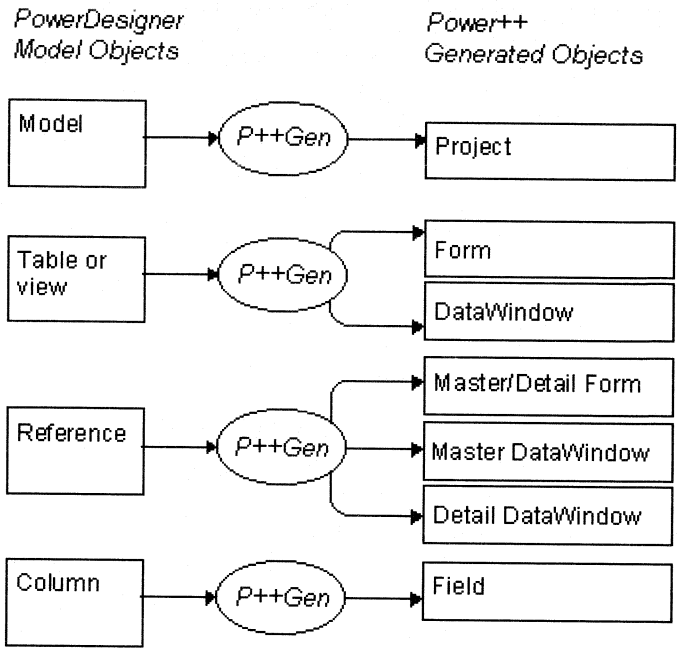
You start AppModeler for Power++ by double-clicking PDPP6.EXE or by clicking its icon in the Program Manager or taskbar.

What P++Gen does

P++Gen uses information from a PDM to instantiate predefined templates and generate Power++ applications. P++Gen generates an entire Power++ application, including:

Power++ file type	Description
WXP	Main project file
WXT	Project target file
WXC	Class file
WXR	Resource file
WXU	User preference file
WXF	Form file

The following schema shows the object transformations performed by P++Gen:



Installed files and directories

AppModeler for Power++ includes the following disk files that you install with the Setup program:

Filename	Directory*	Description
PDPP6.EXE	PD6	Executable file
POWERPP.EXA	PD6\EXA	Default extended attribute import file

*Only short path names are shown in the table. Long names are PowerDesigner 6 (PD6) and Extended Attributes (EXA).

P++Gen templates

AppModeler for Power++ ships with several predefined templates and resource files. The Setup program installs them in the OP*nn*TPL or PP*nn*TPL directory, where *nn* is the version number (for example, OP15TPL for Optima++ version 1.5). Template directories are located in the PowerDesigner application generator path.

References to template files are included in subkeys of the registry path HKEY_CURRENT_USER\ Software\Powersoft\ PowerDesigner 6 \AppModeler for Power++\Power++ *version*. The Setup progam creates subkeys for all versions of Power++ supported by AppModeler.

Project and form templates

Project and form templates that ship with P++Gen are:

Filename	Registry subkey	Template type
PROJECT.WPT	Project Templates	Project
MDIMAIN.WMT	Main Form Templates	Application form
SINGLEDW.WFT	Table Form Templates View Form Templates	DataWindow form for single data source
MDETLDW.WFT	Reference Form Templates	DataWindow form for master/detail data source
SINGLE.WFT	Table Form Templates View Form Templates	Bound control form for single data source
MSTRDETL.WFT	Reference Form Templates	Bound control form for master /detail data source

Associated project and form files

The Setup program installs other project- and form-related template files in the P++Gen template directory. P++Gen copies or generates and modifies these files in your project directory.

Filename	How P++Gen uses it	Result file
APPLIC.WXC	Copies project file that opens the main form	APPLIC.WXC
WRES.WXR	Copies with modifications	WRES.WXR
PROJECT.WXP PROJECT.WXT	Generates a project file	*NAME*.WXP* *NAME*.WXT*
PD_SBAR.WXC PD_TOOLS.WXC	Copies for data control functions and project status bar	PD_SBAR.WXC PD_TOOLS.WXC
MDIMAIN.WXF	Generates a main form	*NAME*.WXF*
SINGLEDW.WXF MSTDETDW.WXF	Generates data forms containing DataWindow objects	*FORMNAME*.WXF#
SINGLEDW.001 MDETLDW.001 MDETLDW.002	Copies and renames binary files for DataWindows, if needed	*FORMNAME*.001# *FORMNAME*.001# *FORMNAME*.002#
SQRY.WXF MQRYDQRY.WXF	Copies intact to provide frames for data forms containing bound controls	SQRY.WXF MQRYDQRY.WXF
SQRYDLG.WXF DQRYDLG.WXF MQRYDLG.WXF	Generates as many times as required to provide dialog boxes that are inserted in frame files at run time. The dialog boxes contain bound controls	*FORMNAME*.WXF# D_*FORMNAME*.WXF# M_*FORMNAME*.WXF#
ABOUTBOX.WXF	Generates About box that you can open from the project Help menu	ABOUTBOX.WXF

* You enter this name on the General page of the Power++ Model Extended Attributes dialog box. The Project Name is used for the project files and the Application Form Name for the main form file.

You enter this name in the Form Filename box on the General page of the Power++ *Object* Extended Attributes dialog box. The D_ and M_ prefixes for master/detail dialog boxes are specified in the MSTRDETL.WFT template.

The Setup program also installs bitmaps and icons in the P++Gen template directory. These are copied to your project directory. The bitmaps are used as navigation tools in generated forms.

Presentation style templates

Presentation style affects how data is presented in generated forms. P++Gen ships with DataWindow and grid presentation style templates.

Filename	Description
DWFFRM3D.WDT	DataWindow template
DWGRID3D.WDT	DataWindow grid template
LISTVIEW.WGT	Standard grid control
GRID.WGT	Advanced grid control

The DataWindow Templates subkey of the registry lists available DataWindow templates and the Grid Templates subkey lists available grid templates.

Field templates

Field templates affect the way data is presented in forms that use bound controls. P++Gen ships with the following field templates:

Filename	Registry subkey	Template type
COMPUTED.WBT	Computed Field Templates	Computed fields
CHECKBOX.WBT	CheckBox Field Templates	Checkbox
DBCOMBO.WBT CHKLIST.WBT	DropDown Field Templates	Combination box
IMAGE.WBT	Image Field Templates	Bitmap
RADIOH.WBT RADIOHV.WBT RADIOV.WBT RADIOVH.WBT	RadioButton Field Templates	Radio buttons
LABELBOX.WBT MASKEDIT.WBT MLTXTBOX.WBT SPIN.WBT TEXTBOX.WBT	Edit Field Templates	Text area

Ensuring that your project refers to an existing database

The projects you develop with P++Gen use information from existing databases. You must ensure that the database exists and contains the fields and data types defined in your PDM before your project can work properly.

❖ **To ensure that your project refers to an existing database:**

- Generate a database directly from the PDM.
  or
  Reverse engineer an existing database to create the PDM.

Importing extended attributes for Power++

The POWERPP.EXA file contains default attribute values for object generation from P++Gen. The POWERPP.EXA attribute values are loaded automatically into a new PDM. You can import these attributes and values into any PDM built with PowerDesigner or S-Designor.

❖ **To import extended attributes for Power++:**

1 Select Dictionary➤Extended Attributes➤Import Attributes.

2 Select the POWERPP.EXA file.

3 Click OK.

A message box tells you that the extended attribute import is successful.

4 Click OK.

Building a Power++ project

A project file manages a Power++ application. Project files contain addresses and filenames for Power++ forms and for other project resources.

A project file generated by P++Gen includes information about project templates and project properties.

Selecting a project template and project properties

P++Gen generates a project file by instantiating a project template with values from the PDM. You attach project and main form templates to the PDM and define certain properties:

Property	Description
Template set*	Optima++ 1.5 or customized target
Project template	Name of template you use to build your project
Project name	Filename you generate for the project
Project title	Title for the project
Project directory	Directory in which you generate project files
Application form template	Name of template you use to build your main form
Application form name	Filename you generate for the main form
Menu items position	Menu bar position for data form menu items

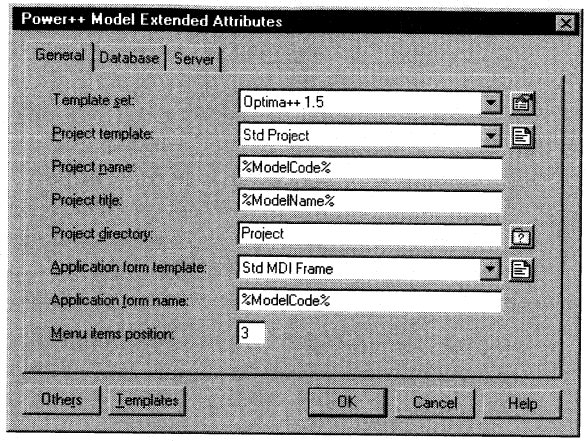
* In addition to the Setup default choices, the Template Set dropdown listbox lists any template set that you name as a string value in the Template Sets registry key.

P++Gen can create a new project directory if you type the directory name in the Project Directory box. If you do not type a complete path, P++Gen looks for the directory in the PWRS\PD6 path, or in its long name equivalent.

❖ To select a project template and project properties:

1 Select Client➤Power++ Model Attributes.

The Power++ Model Extended Attributes dialog box appears.



2 Select a target from the Template Set dropdown listbox.

Only template sets listed in your registry are available in the dropdown listbox.

3 Select a project template from the Project Template dropdown listbox. Select a main form template from the Application Form Template dropdown listbox.

The Setup program installs the Std Project and Std MDI Frame templates as the default selections.

4 (Optional) Type changes to Project Name and Application Form Name.

You do not need to add WXP or WXF extensions to the project name or application form name. P++Gen adds them automatically during generation.

5 Click the button to select a target directory.
or
Type a directory name in the Project Directory box.

The files you generate with P++Gen will be saved in this directory.

6 Type changes to any of the other General page properties that you want to implement in your application.

7 Click OK.

Defining database properties

Connection information

You must enter database connection parameters to generate an application displaying information from a database. For your database connection, you can select an ODBC driver or a native driver supported by Power++.

SQL queries

If the table and column codes in your PDM contain special characters (blank spaces, slashes, and so on), P++Gen ensures that quotation marks are generated around these names in the SQL queries of your Power++ project.

> **Using blank spaces in column codes**
> The use of blank spaces in column codes is not recommended. If you use column codes containing blank spaces in views, queries generated from these columns may not execute properly.

If you select the Quote Table and Column Names checkbox, P++Gen generates quotation marks around every table and column name used in the SQL queries of your project. You can also specify whether or not to include a table owner name in SQL queries.

Defining database properties for an ODBC connection

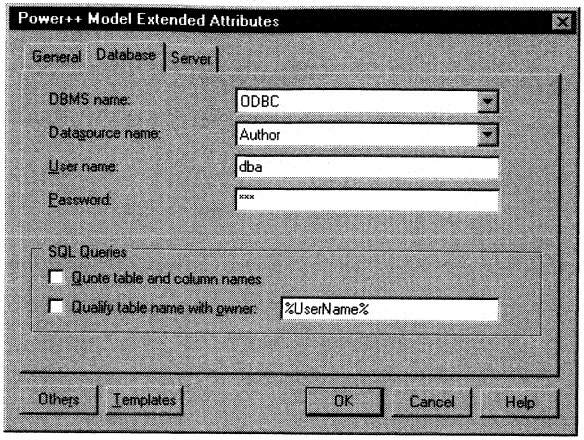
If you select an ODBC driver, the Power++ project you generate uses the data source name, user name, and password that you specify on the Database page of the Power++ Model Extended Attributes dialog box.

❖ **To define database properties for an ODBC connection:**

1 Select Client➢Power++ Model Attributes.

2 Click the Database tab.

The Database properties page opens.



3 Select ODBC from the DBMS Name box.

4 Select the data source you want your application to access.

The Datasource dropdown listbox only includes data sources that you have already defined with an ODBC administrator.

5 Type the name and password needed to access the data source data.

6 (Optional) Select the Quote Table and Column Names checkbox.

Table and column names will be placed in quotation marks for all SQL queries in the Power++ project that you generate.

7 (Optional) Select the Qualify Table Name with Owner checkbox.

The owner name will be added before table names for all SQL queries in the Power++ project that you generate.

8 Click OK.

Defining database properties for a native driver

You can only use a native driver with the Enterprise versions of Optima++ or Power++. Native drivers work only with DataWindow presentation styles.

If you select a native driver for your DBMS, the Power++ project you generate uses the server name, login ID, and login password that you specify on the Server page of the Power++ Model Extended Attributes dialog box.

Some native drivers also require a database name, a database user name and a database password. You can enter this information on the Datasource page of the Power++ Model Extended Attributes dialog box.

For information on native driver parameters and server name syntax, see the *DataWindow Builder Connection Reference*.

❖ To define database properties for a native driver:

1 Select Client➤Power++ Model Attributes.

2 Click the Database tab.

The Database properties page opens.

3 Select a native driver from the DBMS Name box.

4 If your native driver requires a database name, type a database name in the Datasource Name box.

5 If your native driver requires a database user ID and password, type a user and password in the corresponding boxes.

6 (Optional) Select the Quote Table and Column Names checkbox.

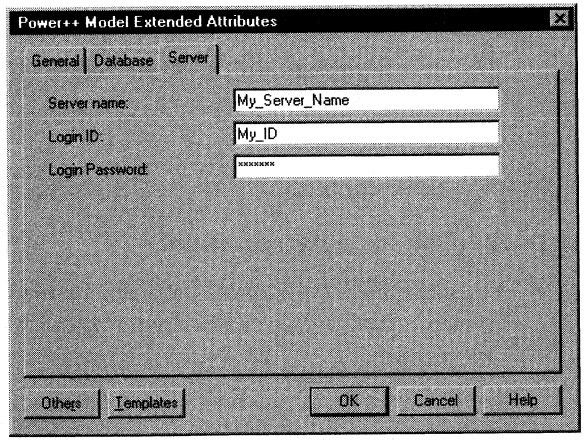
Table and column names will be placed in quotation marks for all SQL queries in the Power++ project that you generate.

7 (Optional) Select the Qualify Table Name with Owner checkbox.

The owner name will be added before table names for all SQL queries in the Power++ project that you generate.

8 Click the Server tab.

The Server properties page opens.



9 Type your server name, login ID, and login password.

10 Click OK.

Defining Power++ forms and fields

Forms are the primary interface elements of Power++ projects. Each form must have its own WXF file that correspond to individual tables, views, and references.

The term **data form** is used to distinguish child forms from the main form (frame window) of a project. Data forms contain data that is presented in fields that you define with P++Gen field templates.

Fields are form elements that correspond to database columns. In P++Gen, field templates define an Power++ control or set of controls that can display data from the database.

Controls are form or field elements that you can use in applications to get user input or to display output. Textboxes, listboxes, and command buttons are examples of standard Power++ controls.

Selecting form templates and form properties

For each table, view, and reference in the PDM, you can define the following general properties:

Property	Description
Generate form	Select to generate a form
Form template	Name of template you use to build a form
Form name	Name of form in the Power++ project window
Form title	Title for a form
Form filename	Filename for a form (P++Gen adds a WXF extension)
Menu option checkbox	Select to generate a menu item to open a form
Menu option textbox	Name of the menu item that will open a form

What is a menu option?

The Menu Option checkbox indicates whether or not a menu item will be generated for a form. The menu option name is the name of the menu item that opens the form in the project you generate. If you want P++Gen to generate the menu option name as a menu item, you must select the Menu Option checkbox.

Menu hot key
You define a menu hot key by typing an ampersand (&) before a letter of the menu option name. Pressing the hot key opens the data form from the generated frame and sheet menus.

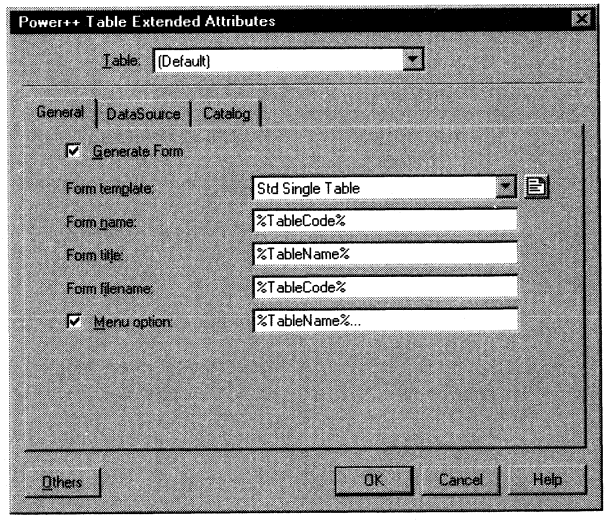
Defining general properties

You can use default general properties for a PDM object type or define form properties for each table, view, and reference.

❖ To define general properties:

1 Select Client➤Power++ Table Attributes.
or
Select Client➤Power++ View Attributes.
or
Select Client➤Power++ Reference Attributes.

The Power++ *Object* Extended Attributes dialog box appears.



2 Select a table from the Table dropdown listbox.
or
Select a view from the View dropdown listbox.
or
Select a reference from the Reference dropdown listbox.

If the dropdown listbox displays Default, any changes to object properties that you make in this dialog box apply to all objects of the same type for which these properties have not been previously modified. This includes all new objects that you add to the PDM.

3 Select a form template from the Form Template dropdown listbox.

4 Make changes to other properties, as needed.

5 Click OK.

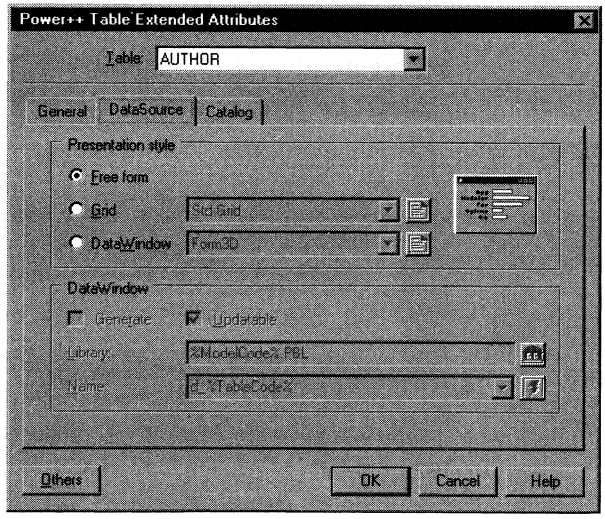
Defining data source properties for a table or view

❖ To define data source properties for a table or view:

1 Select Client➤Power++ Table Attributes.
or
Select Client➤Power++ View Attributes.

2 Click the Datasource tab.

The Datasource page opens.



3 Select a Presentation Style radio button.

A default presentation style is displayed. It must be compatible with the form template that you select on the General page of this dialog box.

For more information on selecting a data presentation style, see the section "Defining data presentation style" on page 431.

4 Click OK.

Defining data source properties for a reference

A reference has a master data source and a detail data source. The data source can be a table or view. Depending on your selection, the objects you generate from a reference can use information from table or view columns.

❖ To define the data source properties for a reference:

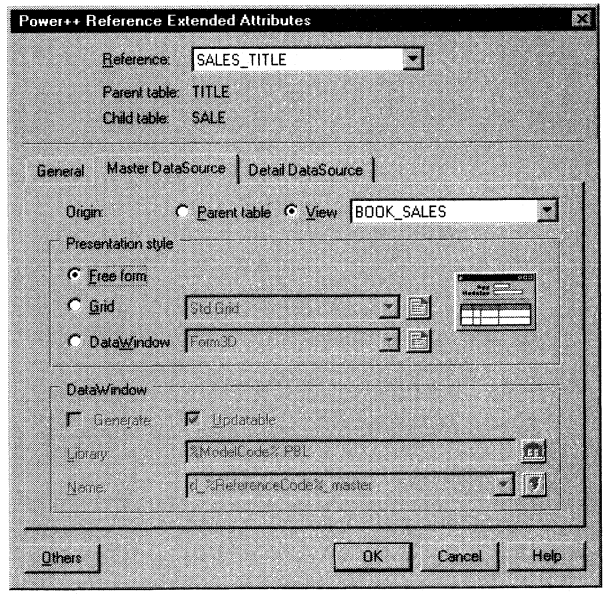
1 Select Client➤Power++ Reference Attributes.

The Power++ Reference Extended Attributes dialog box opens to the General page.

2 Select a reference from the dropdown listbox at the top of the Power++ Reference Extended Attributes dialog box.

3 Click the Master Datasource tab.

The Master Datasource page opens.



4 Click the Parent Table radio button.
or
Select a view from the View dropdown listbox.

Only views that contain the parent table primary keys are available from the dropdown listbox. Selecting a view from the dropdown listbox automatically selects the View radio button.

5 (Optional) Select a data presentation style for the master form.

A default presentation style is displayed. It must be compatible with the form template selectable on the General page of this dialog box.

For more information on selecting a data presentation style, see the section "Defining data presentation style" on page 431.

6 Click the Detail Datasource tab.

The Detail Datasource page opens.

7 Click the Child Table radio button.
or
Select a view from the View dropdown listbox.

Only views that contain a foreign key of the selected reference are available from the dropdown listbox.

8 (Optional) Select a data presentation style for the detail form.

A default presentation style is displayed. It must be compatible with the form template selectable on the General page of this dialog box.

9 Click OK.

Defining data presentation style

You can assign or modify the data presentation style for generated forms. P++Gen uses the following presentation styles:

Style	How it displays data	How it affects data presentation
Free Form	In bound controls	Different bound controls may be used for different fields in the same form
Grid	In a grid or list view format	All fields in the form appear in a grid
DataWindow	In a DataWindow	Data styles for all fields of a form are predefined in an existing DataWindow or are assigned to a new DataWindow using DataWindow Edit Styles

Using a free form presentation style

The free form presentation style displays data using controls defined by field templates. You can assign a different field template to each column of a table or view.

If you use the free form presentation style, P++Gen generates column labels as tool tips for the Power++ control that displays column data. You can add a column label in the List of Columns dialog box.

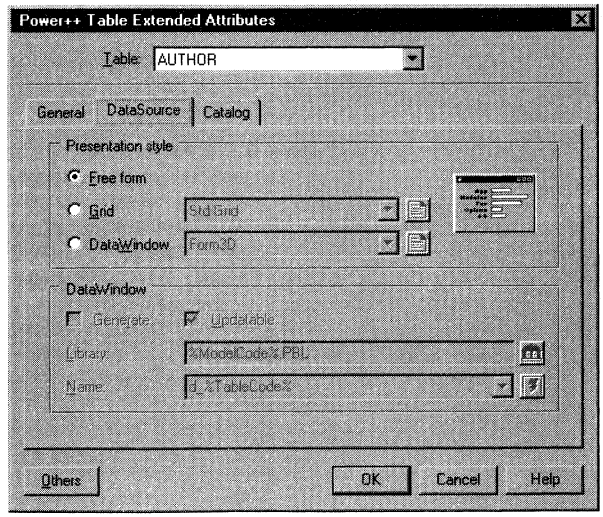
❖ To use a free form presentation style:

1 Select Client➤Power++ Table Attributes.
or
Select Client➤Power++ View Attributes.
or
Select Client➤Power++ Reference Attributes.

2 Select a table, view, or reference from the dropdown listbox at the top of the Power++ Extended Attributes dialog box.

3 Click the Datasource tab.
or
Click the Master Datasource tab.
or
Click the Detail Datasource tab.

The Extended Attributes dialog box opens to the Datasource page that you select.



4 If you clicked the Master Datasource or Detail Datasource tab, select a data source origin.

For more information on selecting a data source origin, see the section "Defining data source properties for a reference" on page 430.

5 Select the Free Form radio button.

6 Click OK.

Using a grid presentation style

The grid presentation style displays data in grid controls that you attach to generated forms. A grid template defines the grid control that P++Gen generates. The grid template you use must be compatible with the form template you select on the General page of the Power++ Extended Attributes dialog box.

For example, the LISTVIEW.WGT and GRID.VGT grid templates that ship with P++Gen are compatible with the SINGLE.WFT and MSTRDTL.WFT form templates, but you cannot use them with the SINGLEDW.WFT or MDETLDW.WFT form templates.

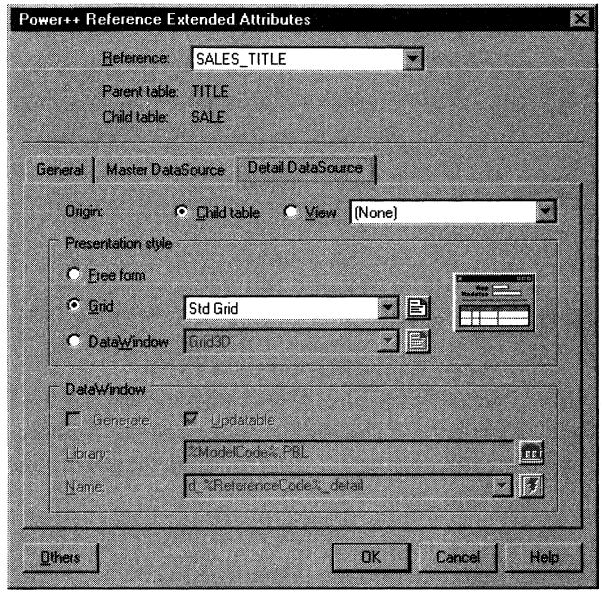
❖ To use a grid presentation style:

1 Select Client➤Power++ Table Attributes.
or
Select Client➤Power++ View Attributes.
or
Select Client➤Power++ Reference Attributes.

2 Select a table, view, or reference from the dropdown listbox at the top of the Power++ Extended Attributes dialog box.

3 Click the Datasource tab.
or
Click the Master Datasource tab.
or
Click the Detail Datasource tab.

The Extended Attributes dialog box opens to the Datasource page that you select.



4 If you clicked the Master Datasource or Detail Datasource tab, select a data source origin.

For more information on selecting a data source origin, see the section "Defining data source properties for a reference" on page 430.

5 Select the Grid radio button.

6 Select a grid template from the Grid dropdown listbox.

7 Click OK.

Using a DataWindow presentation style

You can only use the DataWindow presentation style with the Enterprise versions of Optima++ or Power++

The DataWindow presentation style displays data in a DataWindow that P++Gen attaches to generated forms. If you use a DataWindow in a form, the field templates attached to columns are not generated for that form.

You can use an existing DataWindow to present form data or you can generate a new DataWindow. If you use an existing DataWindow, the controls defined in the DataWindow must correspond to columns of the table, view, or reference that you are generating as a form.

File locations for DataWindow generation
The AppModeler Setup program installs the PDSYS.PBL file and the PBORC050.DLL file in your AppModeler path. You need these files to use DataWindow objects with P++Gen.

Template compatibility

The DataWindow template you use must be compatible with the form template you select on the General page of the Power++ Extended Attributes dialog box.

For example, the Form 3D DataWindow template (DWFFRM3D.WDT) that ships with P++Gen is compatible with the DataWindow Single Table or DataWindow Single View template (SINGLEDW.WFT), but not with the Std Single Table template (SINGLE.WFT).

If you select the DataWindow Master/Detail template (MDETLDW.WFT), you must use a DataWindow in both the master form and the detail form.

DataWindow properties

If you select a DataWindow presentation style, you can define the following P++Gen DataWindow properties:

Property	Description
DataWindow	Name of the DataWindow template
Generate	Select to generate a new DataWindow or replace an existing DataWindow
Updatable	Select to generate a DataWindow from which the data source can be updated. Only meaningful if you also select the Generate checkbox
Library	Name of PBL file that contains the DataWindow you want to attach to your form or that will contain the DataWindow that you generate with P++Gen
Name	Name of the DataWindow

❖ **To use a DataWindow presentation style:**

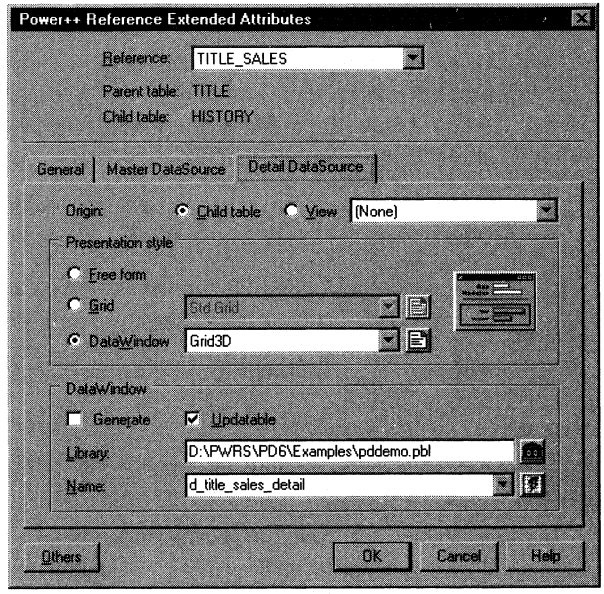
1 Select Client➤Power++ Table Attributes.
or
Select Client➤Power++ View Attributes.
or
Select Client➤Power++ Reference Attributes.

2 Select a table, view, or reference from the dropdown listbox at the top of the Power++ Extended Attributes dialog box.

3 Select a form template that supports a DataWindow presentation style.

4 Click the Datasource tab.
or
Click the Master Datasource tab.
or
Click the Detail Datasource tab.

The Extended Attributes dialog box opens to the Datasource page that you select.



5 If you clicked the Master Datasource or Detail Datasource tab, select a data source origin.

For more information on selecting a data source origin, see the section "Defining data source properties for a reference" on page 430.

6 Select the DataWindow radio button.

7 Select a DataWindow template from the dropdown listbox.

8 Click the button, select a PBL library, and click OK.
or
Type the name of a PBL library in the Library box.

Whether you are using an existing DataWindow or generating a new DataWindow, you must select a PBL library.

9 Type the DataWindow name.
or
Select a DataWindow name from the dropdown listbox.

Only DataWindow names from the library that you selected appear in the dropdown listbox.

Whether you are using an existing DataWindow or generating a new DataWindow, you must select a DataWindow name. In either case, P++Gen attaches the DataWindow you name to the form that you generate.

10 (Optional) Click the  button to edit the named DataWindow. Edit the DataWindow and select File➤Exit from the DataWindow Builder menu to return to P++Gen.

11 Select the Generate checkbox.
or
Clear the Generate checkbox.

Select the checkbox to generate a new DataWindow. Clear the checkbox to use an existing DataWindow.

12 Select the Updatable checkbox.
or
Clear the Updatable checkbox.

13 Click OK.

Selecting a computed fields template

If you use calculated expressions in a view, you must use a computed fields template to generate a form based on that view. In P++Gen you can assign different computed fields template to different views, but not to different columns in the same view.

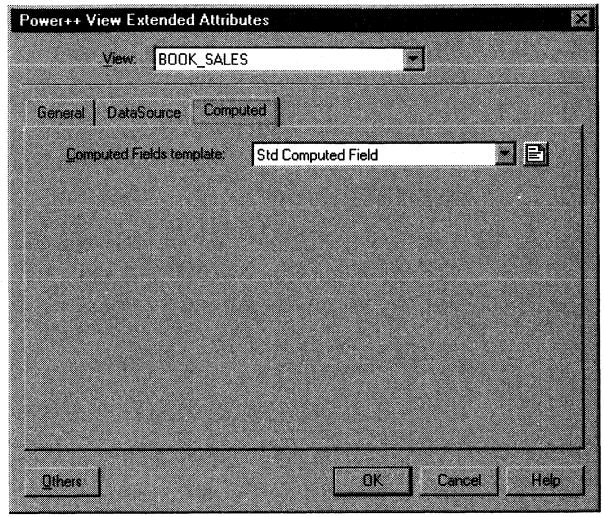
P++Gen ships with a single computed fields template that enables you to display all calculated expressions. P++Gen automatically assigns this template to all columns using these expressions.

❖ To select a computed fields template:

1 Select Client➤Power++ View Attributes.

2 Click the Computed tab.

The Computed page of the Power++ View Extended Attributes dialog box opens.



3 Select a template from the Computed Fields Template dropdown listbox.

Unless you add a template path to the Computed Fields registry key, the Std Computed Field template (COMPUTED.WBT) is the only selection available from the dropdown listbox.

For information on adding a template path to your registry, see "Creating project and form templates" on page 470.

4 Click OK.

Defining field attributes

To use information from a database in forms generated with bound controls, you need to attach field styles and templates to columns or domains.

The default field style is Edit. TEXTBOX.WBT is the default field template for the Edit field style. The default field template for a field style is modifiable in the Field Styles subkey of the registry.

For information on adding field styles or changing default field templates, see "Adding a template or a field style" on page 473.

Attaching a field style and field template to a column

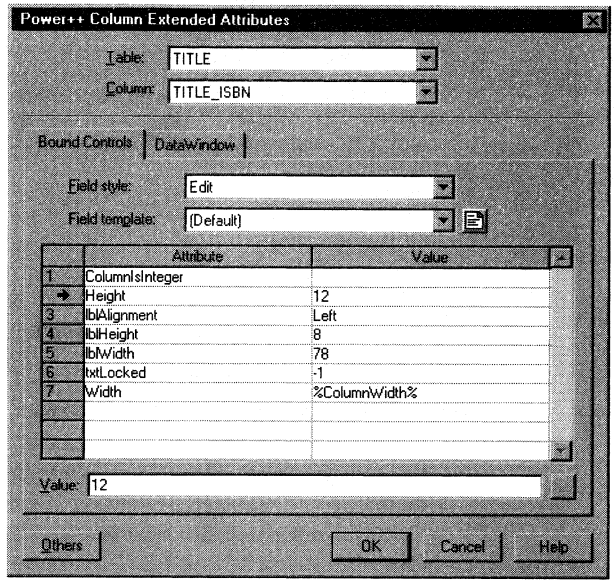
A field style indicates the type of Power++ controls to generate for a field in a data form that uses the free form presentation style. A field style can use any standard or personalized Power++ control that you define in a field template.

Each field template has a list of attributes. You can modify the values of these attributes from P++Gen.

❖ To attach a field style and field template to a column:

1 Select Client➤Power++ Column Attributes.

The Power++ Column Extended Attributes dialog box appears.



2 Select a table from the Table dropdown listbox.

3 Select a column from the Column dropdown listbox.

4 Select a style from the Field Style dropdown listbox.

5 Select a template from the Field Template dropdown listbox.

6 Double-click an item in the Attribute column.

The attribute is selected and the cursor moves to the Value box.

7 Type a value for the field style attribute in the Value box.

8 Double-click another attribute and change its value if needed.

9 Click OK.

Attaching a field style and field template to a domain

You can attach a field style and field template to a domain. These apply automatically to any columns you attach to the domain. You can also update the values of field template attributes for columns attached to the domain.

❖ **To attach a field style and field template to a domain:**

1 Select Client➢Power++ Domain Attributes.

The Power++ Domain Extended Attributes dialog box appears.

2 Select a domain from the Domain dropdown listbox.

3 Select a field style from the Field Style dropdown listbox.

4 Select a template from the Field Template dropdown listbox.

5 Double-click an item from the Attribute column.

The attribute is selected and the cursor moves to the Value box.

6 Type a value for the field style attribute in the Value box.

7 Select another attribute and change its value if needed.

8 Click OK.

A message box asks if you want to update the columns attached to the domain.

9 Click Yes.
or
Click No.

Defining and attaching catalog attributes

P++Gen can use attributes that are defined in the catalog tables of the PowerBuilder repository.

Using the AppModeler client interface, you can import catalog attributes for a table or a DataWindow. You can generate catalog attributes separately from an Power++ project.

For more information on catalog attributes and their AppModeler default values, see the appendix "Generation Variables and Attributes."

Importing catalog attributes

You can import catalog attributes from a data source and attach them to tables, columns, and domains.

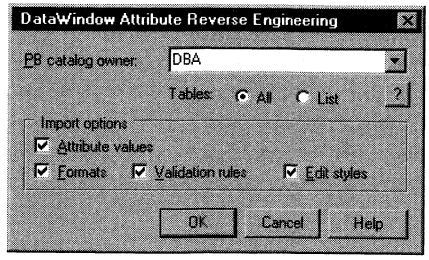
❖ To import catalog attributes:

1 Select Client➤Reverse DataWindow Attributes.
or
Click the Reverse button in any of the following dialog boxes:

DataWindow Edit Styles
DataWindow Display Formats
DataWindow Validation Rules
Power++ Table Extended Attributes Catalog page
Power++ Column Extended Attributes DataWindow page

> **Connecting to a data source**
> If you are not already connected to a data source, the ODBC Connection dialog box appears. You must connect to the data source via an ODBC driver.

The DataWindow Attribute Reverse Engineering dialog box appears.



2 Select a PowerBuilder catalog owner from the dropdown listbox.

The dropdown listbox only includes catalog owners for the data source you selected.

3 Select the All radio button.
or
Select the List radio button.

If you select List, the List of Tables dialog box appears. You can then select the tables from which you want to reverse engineer catalog attributes, and click OK to return to the DataWindow Attribute Reverse Engineering dialog box.

4 Select the checkboxes next to the catalog attributes that you want to reverse engineer.

5 Clear the checkboxes next to the catalog attributes that you do not want to reverse engineer.

6 Click OK.

Generating catalog attributes

You can generate catalog attributes in a data source independently of the projects that you generate with P++Gen.

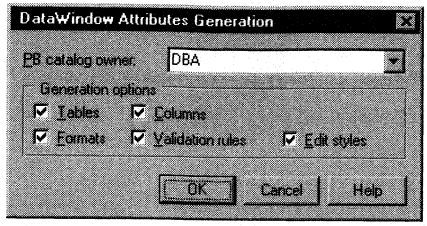
❖ **To generate catalog attributes:**

1 Select Client➢Generate DataWindow Attributes.
or
Click the Alter button in any of the following dialog boxes:

DataWindow Edit Styles
DataWindow Display Formats
DataWindow Validation Rules
Power++ Table Extended Attributes Catalog page
Power++ Column Extended Attributes DataWindow page

> **Connecting to a data source**
> If you are not already connected to a data source, the ODBC Connection dialog box appears. You must connect to the data source via an ODBC driver.

The DataWindow Attribute Generation dialog box appears.



2 Select a PowerBuilder catalog owner from the dropdown listbox.

The dropdown listbox only includes catalog owners for the data source you selected.

3 Select the All radio button.
or
Select the List radio button.

If you select List, the List of Tables dialog box appears. You can then select the tables from which you want to generate catalog attributes, and click OK to return to the DataWindow Attribute Generation dialog box.

4 Select the checkboxes next to the catalog attributes you want to generate.

5 Clear the checkboxes next to catalog attributes you do not want to generate.

6 Click OK.

Defining catalog attributes for a table

Catalog attributes for a table include the fonts and display styles for labels, headings, and data. You define or modify catalog attributes for a table in the Power++ Table Extended attributes dialog box.

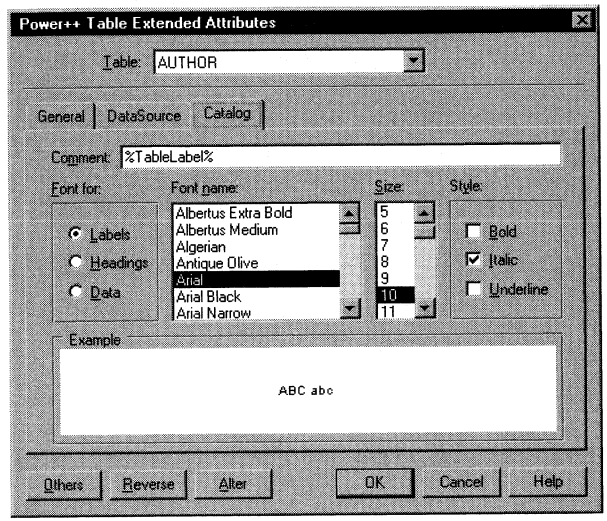
❖ To define catalog attributes for a table:

1 Select Client➤Power++ Table Extended Attributes.

2 Select a table from the Table dropdown listbox.

3 Click the Catalog tab.

The Catalog page opens.



4 (Optional) Type a comment in the Comment box.

You can use the comment box to save additional information about a table.

5 Select a Font For radio button.
Select a Font Name.
Select a Size.
Select or clear a Style checkbox.

6 Click OK.

Defining catalog attributes for a DataWindow

Catalog attributes for a DataWindow include edit styles, display formats, and validation rules. You can attach these attributes to a column or a domain that you generate in a DataWindow.

Attaching an edit style to a column or a domain

Edit styles specify how column data is presented in DataWindow objects. The edit style types for DataWindow objects in AppModeler correspond to those of the Edit Styles dialog boxes in the PowerBuilder environment.

For more information on DataWindow edit styles, see the *PowerBuilder User's Guide* or the *DataWindow Builder User's Guide*.

You can select an edit style type before you select an edit style to attach to a column or domain. The available edit style types are:

DropDownListBox
DropDownDataWindow
Check Box
RadioButton
Edit Mask
Edit

By selecting an edit style type, you limit the edit style choices that appear in the Edit Style Code dropdown listbox for attachment to a column or domain.

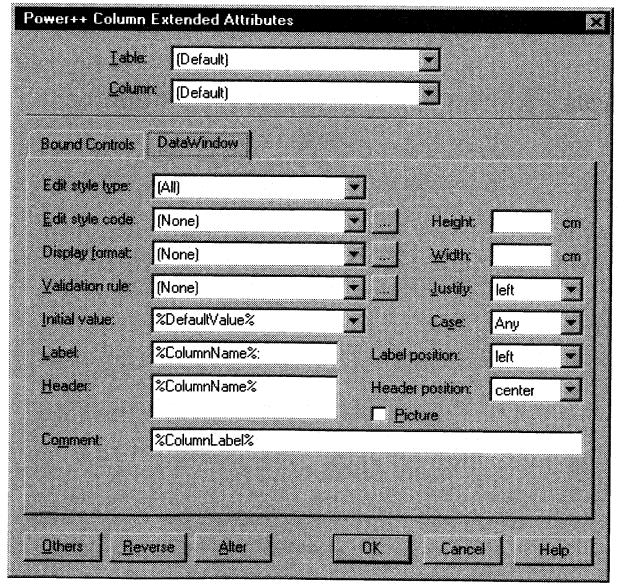
If you select the DropDownDataWindow edit style type, the DataWindow you generate with P++Gen must be generated to the PBL library that contains the DataWindow you select for the Edit Style Code.

❖ To attach an edit style to a column or a domain:

1 Select Client➤Power++ Column Attributes.
or
Select Client➤Power++ Domain Attributes.

The Power++ Column Extended Attributes dialog box or the Power++ Domain Extended Attributes dialog box appears.

2 Click the DataWindow tab.



3 Select a table and column from the Table and Column dropdown listboxes.
or
Select a domain from the Domain dropdown listbox.

If you select Default, the new default value applies to all columns or all domains for which you have not previously modified the edit style. This includes all new columns or domains.

4 Select a type from the Edit Style Type dropdown listbox.

5 Select an Edit Style Code from the dropdown listbox.

> **Edit style definitions**
> You can access edit style definitions by clicking the button next to the Edit Style Code dropdown listbox.

6 Click OK.

If you modified the edit style for a named domain, a message box asks if you want to modify catalog attributes for columns currently attached to the domain.

- Click Yes to modify catalog attributes for all columns attached to the selected domain.
  or
  Click No to limit your modifications to columns that you subsequently attach to the selected domain.

Attaching a display format to a column or a domain

Display formats customize the display of column data in a DataWindow. The options in the DataWindow Display Formats dialog box correspond to those in the Display Formats dialog boxes in the PowerBuilder environment.

For more information on DataWindow display formats, see the *PowerBuilder User's Guide* or the *DataWindow Builder User's Guide*.

❖ To attach a display format to a column or a domain:

1 Select Client➤Power++ Column Attributes
or
Select Client➤Power++ Domain Attributes.

The Power++ Column Extended Attributes dialog box or the Power++ Domain Extended Attributes dialog box appears.

2 Click the DataWindow tab.

3 Select a table and column from the Table and Column dropdown listboxes.
or
Select a domain from the Domain dropdown listbox.

If you select Default, the new default value applies to all columns or all domains for which you have not previously modified the display format. This includes all new columns or domains.

4 Select a display format from the dropdown listbox.

> **Display format definitions**
> You can access display format definitions by clicking the button next to the Display Format dropdown listbox.

5 Click OK.

If you modified the display format for a named domain, a message box asks if you want to modify catalog attributes for columns currently attached to the domain.

- Click Yes to modify catalog attributes for all columns attached to the selected domain.
  or
  Click No to limit your modifications to columns that you subsequently attach to the selected domain.

Attaching a validation rule to a column or a domain

Validation rules check data before updating a database table associated with a DataWindow. The options in the DataWindow Validation Rules dialog box correspond to those in the Validation Rules dialog boxes in the PowerBuilder environment.

For more information on DataWindow validation rules, see the *PowerBuilder User's Guide* or the *DataWindow Builder User's Guide*.

❖ To attach a validation rule to a column or a domain:

1 Select Client➤Power++ Column Attributes.
or
Select Client➤Power++ Domain Attributes.

The Power++ Column Extended Attributes dialog box or the Power++ Domain Extended Attributes dialog box appears.

2 Click the DataWindow tab.

3 Select a table and column from the Table and Column dropdown listboxes.
or
Select a domain from the Domain dropdown listbox.

If you select Default, the new default value applies to all columns or domains for which you have not previously modified the validation rule. This includes all new columns or domains.

4 Select a validation rule from the dropdown listbox.

> **Validation rule definitions**
> You can access validation rule definitions by clicking the button next to the Validation Rule dropdown listbox.

5 Click OK.

If you modified the validation rule for a named domain, a message box asks if you want to modify catalog attributes for columns currently attached to the domain.

- Click Yes to modify catalog attributes for all columns attached to the selected domain.
  or
  Click No to limit your modifications to columns that you subsequently attach to the selected domain.

Attaching other catalog attributes

In addition to an edit style, a display format, and a validation rule, you can assign values to the following catalog attributes for a column or a domain:

Catalog attribute	Description
Initial value	Initial data value for a column
Label and header	Text that identifies columns in a DataWindow
Comment	Text that describes the purpose of a column
Height and width	Display size for data in a DataWindow field
Justify	Alignment for data in a DataWindow field
Case	Specifies how character data is displayed
Label and header position	Alignment of label and header text
Picture checkbox	Select to indicate a picture in the DataWindow

❖ **To attach other catalog attributes to a column or a domain:**

1 Select Client➤Power++ Column Attributes.
or
Select Client➤Power++ Domain Attributes.

The Power++ Column Extended Attributes dialog box appears, or the Power++ Domain Extended Attributes dialog box appears.

2 Click the DataWindow tab.

3 Select a table and column from the Table and Column dropdown listboxes.
or
Select a domain from the Domain dropdown listbox.

If you select Default, the new default values apply to all columns or all domains for which you have not previously modified the same attributes. This includes all new columns or domains.

4 Type values for catalog attributes.

5 Click OK.

If you modified the catalog attributes for a named domain, a message box asks if you want to modify catalog attributes for columns currently attached to the domain.

- Click Yes to modify catalog attributes for all columns attached to the selected domain.
  or
  Click No to limit your modifications to columns that you subsequently attach to the selected domain.

Generating a project

You can generate a Power++ application from objects you select in the PDM.

Tree view items

The objects that you want to generate must be listed in a tree view in the Application Generation window. This window contains an object list in a tree view with the following items:

- Project object
- Main form object
- Forms for tables, views, and references you select from the PDM
  or
  Forms corresponding to filters you set in the last P++Gen session
- DataWindow objects, if a DataWindow presentation style is selected
- Columns of each form or DataWindow object

Your first P++Gen session
If you do not select any objects in the PDM and you did not use a selection filter in a previous P++Gen session, form objects are listed in the Application Generation window for every table, view, and reference.

Checking items to be generated

Each item in the tree view that you want to generate must have a checkmark in front of it. You can use the Select All button to check all form and project level items in the tree view, or you can use the Unselect All button to clear these checkmarks.

The checkbox to the left of a DataWindow object is not affected by the Select All or Unselect All buttons. If this checkbox is selected, P++Gen creates or regenerates the DataWindow to a PBL, as well as to the project form. If this checkbox is not selected and the form is generated, P++Gen can insert an existing DataWindow in the form.

Generating a menu

If you use the MDIMAIN.WMT template to generate a main form, you also generate a main form menu bar containing three menu items: File, Topics, and Help. Two additional menu items, Edit and Window, are not visible until you open a data form.

If you generate your project using the menu option, and you do not change the menu position default (menu position = 3), the data forms you generate are accessible from the Topics menu.

The Edit menu allows a user of your project to cut, copy, and paste information in the data forms, or to undo a previous action. The Window menu provides access to any open data forms and allows you to arrange them in a tile or cascade display.

For more information on data form menu options, see "Selecting form templates and form properties" on page 427.

Generating a project from selected objects

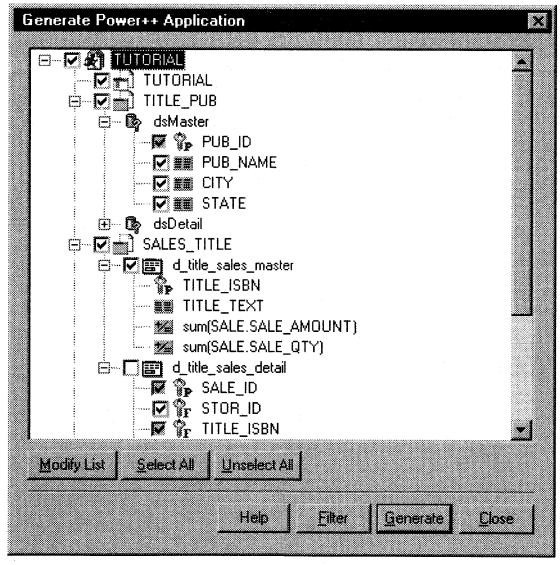
To create a list of objects to generate, you can select objects directly from the PDM or you can reuse objects you selected in the last P++Gen session.

❖ To generate a project from selected objects:

1 Use selection tools to select objects in the PDM.
or
Verify that no objects are selected in the PDM to reuse objects selected during the previous P++Gen session.

2 Select Client➢Generate Power++ Application.

The Application Generation window appears.



3 Click any plus signs to the left of tree view items.

All tree view items are now visible.

4 Verify that all objects you want to generate are checked.

5 Click the Generate button.

Reviewing the generation log

You can follow the progress of the generation from the log window that appears after the generation begins. P++Gen indicates if there are any errors in the generation or if the generation is successful. Before P++Gen attempts to generate a file with the same name as an existing file in the project directory, a message box asks you for confirmation.

The project log file, *MODELCODE*.LOG, is created automatically in your project directory. It contains model and generation information. You can read the log file with a text editor.

Cleaning your project directory
After the generation is complete, you can open your project in Power++. If you generate your project more than once, select Run➤Clean from the Power++ menu before you build or run your project. Cleaning erases files that are left from an earlier Power++ project.

Modifying the generation selection

There are several ways to modify a preliminary object selection from the Application Generation window. To modify the object selection, you can:

- Set the generation flag for objects in the tree view
- Add objects to or remove objects from the tree view
- Apply a filter

Setting the object generation flag

A checkmark to the left of an item in the tree view indicates that its generation flag is selected.

Project and form items

By clicking the Select All button beneath the tree view, you can set the generation flags for all project and form items. By clicking Unselect All, you can clear the generation flags of all project and form items in the tree view.

DataWindow and column items

You must expand the tree view to list the DataWindow and column items in the tree view. The Select All and Unselect All buttons do not affect the DataWindow and column items in the tree view. You can only clear the checkboxes of DataWindow and column items one by one.

The checkbox to the left of a DataWindow item indicates whether or not it will be generated to a PBL file. You can include an existing DataWindow in a form by clearing the checkbox to the left of the DataWindow item and selecting the checkbox of the form object that contains it.

You cannot clear the generation flag of a primary key column. The tree view does not display checkboxes for columns in a view, or for columns in a reference that use a view as their data source. These columns are always generated when you generate their parent form.

Items sharing a data source

Two references can share the same data source (parent table or child table). Changes to column generation flags of the master or detail form of one of the references are automatically reflected in the columns of forms based on the other reference. However, changing the column generation flag for a table does not affect the generation flag for the column of a master or detail form that uses the table as its data source.

❖ **To set the generation flags for objects in the tree view:**

1 Click the plus signs in front of tree view items.

You expand the tree view.

2 Select checkboxes to the left of the objects you want to generate.

3 Clear checkboxes to the left of objects you do not want to generate.

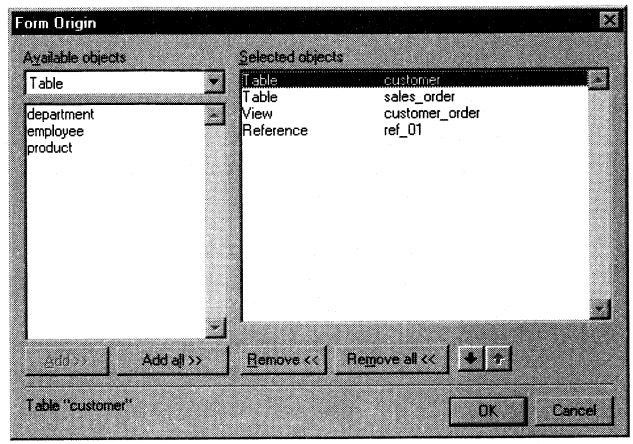
Adding or removing objects from the tree view

If you modify the list of objects in the tree view without using a filter, the modified selections you make are only valid for the current P++Gen session. If you close the Application Generation window without generating an Power++ project, the tree view does not save your modified selections.

❖ **To add or remove objects from the tree view:**

1 Click the Modify List button below the tree view.

The Form Origin dialog box appears.



2 Select a PDM object type from the Available Objects dropdown listbox.

The Available Objects listbox displays all objects of the type selected that are not in the tree view. The Selected Objects listbox displays all objects that are in the tree view.

3 Select objects in the Available Objects listbox and click Add.
or
Select objects in the Selected Objects listbox and click Remove.

Selected objects move from one listbox to the other.

> **Using the Add All or Remove All buttons**
> You can click the Add All or Remove All buttons to move all objects of a given type from one listbox to the other.

4 Repeat steps 2 and 3 for other PDM object types.

5 Click OK.

You return to the Application Generation window. The objects you added to the Selected Objects listbox appear in the tree view. Objects you removed from this listbox disappear from the tree view.

Applying a filter to modify object selection

Filters help handle large amounts of information. A large PDM can include hundreds of objects. You can use a filter to augment or restrict the items included in the tree view in the Application Generation window.

After you apply a filter, only the matching items appear in the tree view. The selections you make with a filter are saved by P++Gen, whether or not you generate your selections during the current session.

Types of filters

There are four types of filters:

Filter	Action
All objects	Selects every object in the PDM
Objects of submodel	Selects objects from a named submodel
Modified objects	Compares objects in the current PDM to objects in an archived PDM and selects objects that have changed
User-defined list	Lets you select a list of objects from the PDM

You can select or clear object type checkboxes with any of the filter types.

User-defined filter

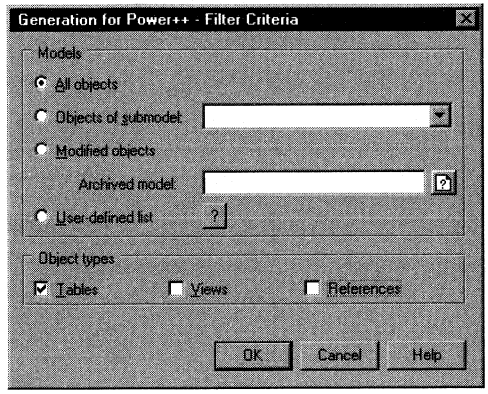
If you use the user-defined filter, you can add any object in the PDM to the tree view. You can use the buttons in the Form Origin dialog box to make customized filter selections:

Button	Function
Add	Transfers a selected object from the Available Objects listbox to the Selected Objects listbox and adds it to the tree view
Add All	Transfers all objects of the selected object type
Remove	Transfers a selected object from the Selected Objects listbox to the Available Objects listbox and removes it from the tree view
Remove All	Transfers all objects of the selected object type
	Moves the selected object up in the list order of Selected Objects and in the tree view
	Moves the selected object down in the list order of Selected Objects and in the tree view

❖ **To select objects using user-defined filter criteria:**

1 Click the Filter button in the Application Generation window.

The Filter Criteria dialog box appears.



2 Click the ? button beside the User-Defined List option.

The Form Origin dialog box appears.

3 Select a PDM object type from the Available Objects dropdown listbox.

The Available Objects listbox displays all objects of the type selected that are not in the tree view. The Selected Objects listbox displays all objects that are in the tree view.

4 Use the buttons of the Form Origin dialog box to add, remove, or reposition objects in the listboxes and in the tree view generation list.

5 Click OK.

You return to the Filter Criteria dialog box.

6 Verify that checkboxes are selected for the object types that you want to appear in the tree view.

7 Click OK.

You return to the Application Generation window. The tree view displays the new selections you made with the user-defined filter.

Fine-tuning before and after generation

Before you generate or regenerate your application, you can fine tune and test it from the tree view of the Application Generation window as follows:

- Modify model properties
- Modify extended attributes
- Select columns to generate
- Include or remove menu options
- Open the generated project

Modifying model properties

You can display or modify model properties and extended attributes from the context menus of the project or main form tree view items.

❖ **To modify model properties from the tree view:**

1 Right-click the project item in the tree view.
or
Right-click the main form item in the tree view.

The item context menu appears.

2 Select Model Properties.

The Model Properties dialog box appears.

3 Make changes to the model properties as needed.

4 Click the Extended button.

The Power++ Model Extended Attributes dialog box appears.

5 Make changes to the extended attributes as needed.

6 Click OK in each of the dialog boxes.

Modifying extended attributes

You can display and modify extended attributes from the context menus of data form items in the tree view. Column extended attributes are also accessible from the tree view.

❖ **To modify extended attributes of a table, view, or reference:**

1 Right-click a data form item in the tree view.

The data form item context menu appears. The context menu selection depends on the data source origin of the data form.

2 Select Table Attributes.
or
Select View Attributes.
or
Select Reference Attributes.

The Power++ Extended Attributes dialog box appears.

3 Make changes to the object extended attributes as needed.

4 Click OK.

❖ **To modify extended attributes of a column:**

1 Right-click a , , or column of a table, view, or reference.
or
Right-click a view column representing a calculated field.

The column context menu appears.

2 Select Column Attributes.

3 Make changes to the column extended attributes as needed.

4 Click OK.

Selecting columns to generate in a data form

From the tree view, you can select which columns of a table you want to generate in a data form. You can also select which columns you want to generate in a master or detail data form of a reference, if it uses a table for its data source.

You must expand a form and its data source item to list the columns of that form. You expand a tree view item by clicking a plus sign to the left of it.

For more information on selecting individual columns for generation, see "Setting the object generation flag" on page 452.

Including or removing menu options

P++Gen generates an application menu based on a main form template. You can select menu options to add to the application menu in object Extended Attributes dialog boxes. These menu options open data forms that you generate with P++Gen.

From the Application Generation window, you can choose to generate a menu without menu options. If you generate without menu options, the menu options selections in the object Extended Attributes dialog boxes do not change. These menu options are saved in the PDM and you can use them the next time you generate your application menu.

❖ To include or remove menu options from the tree view:

1 Right-click the main form item in the tree view.

The main form item context menu appears.

2 Select Include Menu Options.

Selecting this item alternatively places or removes a checkmark before the item name. A checkmark selects the menu options for generation with the main menu.

Opening the project from P++Gen

If you generated a project with P++Gen, you can test it from the tree view.

❖ To open the project from P++Gen:

1 Right-click the project item in the tree view.

2 Select Run Project from the context menu.

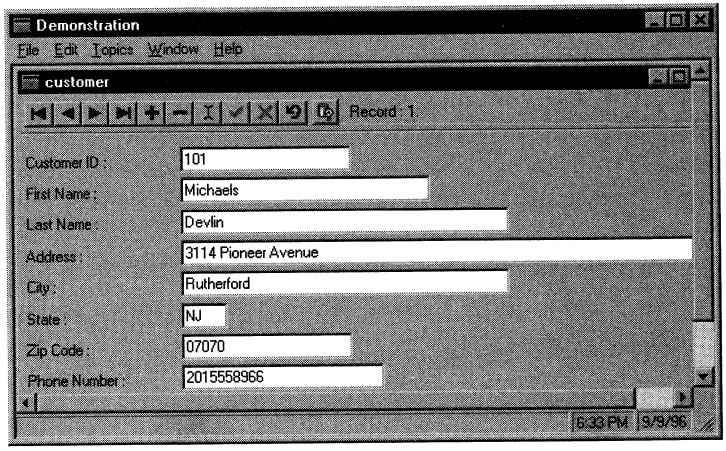
You open the project in Power++ or Optima++.
or
A message box tells you that Power++ or Optima++ is already open.

> **Open Power++ or Optima++ file**
> If the open file in Power++ or Optima++ is an earlier version of your project file, close the file without saving it, or save it under a different filename. To avoid resource conflicts, select Run➢Clean in Power++ or Optima++ before you build or run the new project file.

Examples of forms based on predefined templates

SINGLE.WFT + TEXTBOX.WBT

The following example shows a form generated using the SINGLE.WFT form template and several instances of the TEXTBOX.WBT field template.

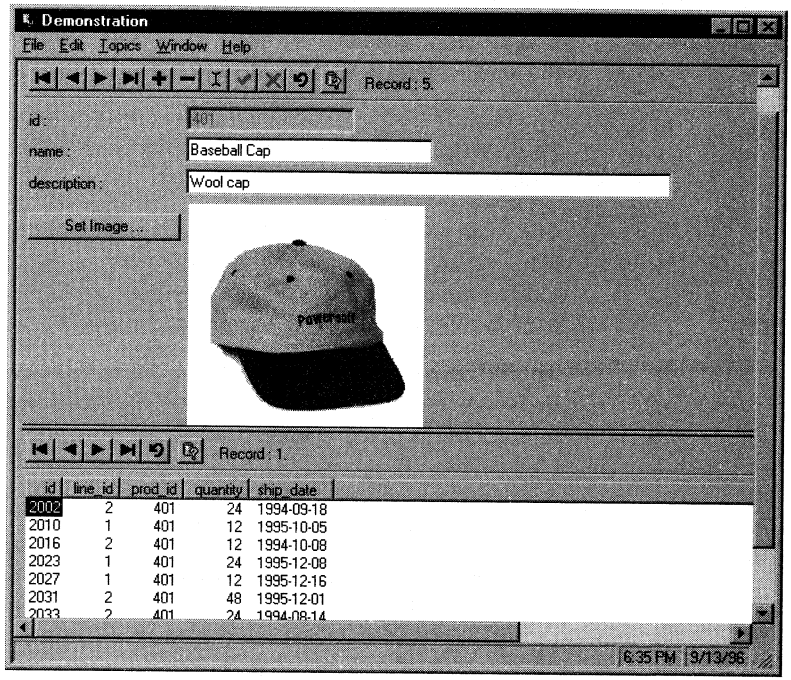


MSTRDETL.WFT
+ LISTVIEW.WGT

The following example shows a master/detail form generated with the MSTRDETL.WFT form template.

The master form has a free form presentation style. Columns of the master form were generated using the LABELBOX.WBT, TEXTBOX.WBT, and IMAGE.WBT field templates.

The detail form uses a grid presentation style. It was generated using the LISTVIEW.WGT template.



Using templates

You use project and form templates, along with information from PDM objects and extended attributes, to generate Power++ project and form files. Field templates define the controls that you can generate in data forms.

What project and form templates define

Project (WPT)

The project template has a WPT extension. Using information from the PDM, it generates a WXP file and a WXT file.

The project template can contain a list of files to copy during generation of a Power++ project. If the project template lists a file, P++Gen copies it to projects generated with the template.

Main form (WMT)

The main form template has a WMT extension. It generates a WXF file that can contain menu items, database connection information, and a status bar. You can modify the menu from P++Gen to add menu items that open data forms.

The MDIMAIN.WMT template that ships with AppModeler is based on the Power++ WMDI Form Class. If you generate a main form with this template, all data forms that you open from the main form must be based on the WMDI Child Class. To generate an SDI project, you must use a main form template based on the W Form Class.

Data form (WFT)

Data form templates have WFT extensions. They can provide offset values for the first controls added to generated WXF files. They refer to predefined WXF files that you can modify directly in Power++.

The WXF files that ship with P++Gen are all based on the Power++ WMDI Child Class. They include a standard Power++ navigation control and a nonstandard reset query button.

What field templates define

Field templates define the controls that you generate in Power++ data forms that use the free form presentation style. Field templates usually define one bound control and a label control for identification of the data that the bound control displays. You attach field templates to columns or domains.

Recommended field styles

Certain field styles are appropriate to columns of a specific data type, and vice versa. Recommended correspondences are:

Data type*	Recommended field style	Template
Boolean	Checkbox	CHECKBOX.WBT
Counter	Label box	LABELBOX.WBT
Integer	Spin control textbox	SPINNUM.WBT
OLE	Image	IMAGE.WBT
Text (limited)	Textbox	TEXTBOX.WBT
Text (unlimited)	Multiline	MLTXTBOX.WBT

* The data type name depends on the DBMS that you use.

Using the dropdown combination box template

The dropdown list field template DBCOMBO.WBT substitutes displayed information from one column with information from another column in a different table. The tables must be linked by a common key column.

The values of the replacement column are listed in the generated dropdown combination box control. You can assign the column code of the replacement column to the DataLookupColumn attribute or you can use an expression referring to several columns.

P++Gen instantiates the names of the common key and the source table from default variables that the template defines for the DataLookupKeyColumn and DataLookupRowSource attributes.

Using an expression

If you assign an expression to the DataLookupColumn attribute, it must correspond to SQL syntax supported by the DBMS of your data source. For example, if you use SQL Anywhere, the column codes must be separated by || concatenation marks. If you use Access as your DBMS, you must use the + sign as the concatenation operator.

For example, to list the color, name, and size columns in a combination box, you could use the following expression as a value for DataLookupColumn:

```
(color || ' ' || name || ' (' || prod_size || ')')
```

The result of this expression is shown below in a combination box field with zoom button that was generated by P++Gen:

Zoom button

The DataLookupZoomForm attribute defines an optional zoom button positioned to the right of the dropdown listbox control. The default value for this attribute is the name of the data form generated from the table containing the replacement column. To open this form using the zoom button, the form must be listed in the project that you generate.

The form that you open with the zoom button is set to the record from which you make the zoom. You cannot navigate in the zoomed form unless you click the reset query button beside the standard Power++ navigation control.

If you assign a null value to the DataLookupZoomForm attribute, the zoom button is not generated.

❖ **To use the dropdown combination listbox template:**

1. Select Client➢Power++ Column Attributes.
2. Select a table from the Table dropdown listbox.
3. Select a column from the Column dropdown listbox.
4. Select DropDown from the Field Style dropdown listbox.
5. Select DB Dropdown List from the Field Template dropdown listbox.
6. Double-click the DataLookupColumn attribute.
7. Type the name of the column you want to use to replace information from the current column.
8. Click OK.

Using radio button templates

P++Gen ships with radio button templates that define radio buttons enclosed in a frame. These templates differ only in the arrangement of the radio buttons within the frame.

Template	Arrangement
RADIOH.WBT	Radio buttons arranged left to right in a single row
RADIOHV.WBT	Radio buttons arranged left to right in *n* columns
RADIOV.WBT	Radio buttons arranged top to bottom in a single column
RADIOVH.WBT	Radio buttons arranged top to bottom in *n* rows

ListColumn attribute

To change the number of radio button columns generated with the RADIOHV.WBT template, you can change the ListColumn attribute in the Power++ Column Extended Attributes dialog box. If you set the ListColumn value to 1, the generated field is the same as if you used the RADIOV.WBT template. The template default value is 3.

ListRow attribute

To change the number of radio button rows generated with the RADIOVH.WBT template, you can change the ListRow attribute in the Power++ Column Extended Attributes dialog box.

You define the radio button labels in the List of Values in the Check Parameters dialog box for a column or a domain.

❖ To use a radio button template:

1 Select Dictionary➤List of Columns.

2 Select the column for which you want to generate radio buttons.

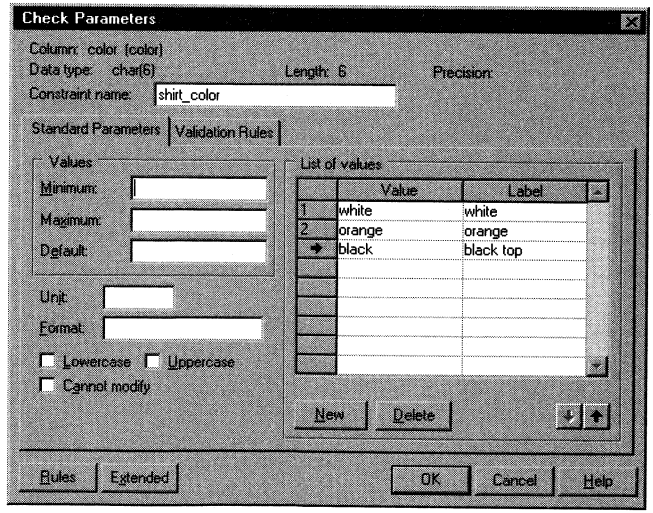
3 Click the Check button.

The Check Parameters dialog box appears.

4 Type the database values for this column in the Values column of the List of Values.

5 Type the labels you want to display in the Labels column.

These become the radio button labels in the forms that you generate with radio button templates.



6 Click the Extended button.

7 Select RadioButtons from the Field Style dropdown listbox.

8 Select a radio button template from the Field Template dropdown listbox.

9 Click OK in each dialog box.

Using the standard dropdown template

P++Gen uses the List of Values in the Check Parameters dialog box to instantiate controls it generates using the CHKLIST.WBT template. You must type values and labels in this list to use this template.

The labels you type in the Labels column become list entries in the dropdown listbox that you generate with the CHKLIST.WBT template.

Using the spin button template

The spin button template uses default variables for the attributes HiRange, LowRange, and Value.

P++Gen instantiates these variables with the Maximum, Minimum and Default Values that you define in the Check Parameters dialog box for a column. You must type values for these check parameters to use the SPINNUM.WBT template. Otherwise, you can replace the default variables with values in the Power++ Column Extended Attributes dialog box.

Viewing template information

You can view template descriptions and other template information directly from P++Gen. The type of information available depends on the type of template you select for viewing.

From P++Gen you can access information for the following template types:

Template	Access in P++Gen	Viewable sections
Project	Model Extended Attributes	Description Modules
Main form	Model Extended Attributes	Description MenuScript
Data form	Table Extended Attributes View Extended Attributes Reference Extended Attributes	Description Attributes
Field	Column Extended Attributes Domain Extended Attributes	Description Controls Attributes
Grid	Table Extended Attributes View Extended Attributes Reference Extended Attributes	Description Controls
DataWindow	Table Extended Attributes View Extended Attributes Reference Extended Attributes	Description Style Controls

Modifying template descriptions

You can modify the template descriptions if you open the template file with a text editor.

> **Before you modify template files**
> Make a copy of all template files before you attempt to modify them.

If you change information in a template section that is viewable in P++Gen, the modifications you make appear in the Show Template dialog box.

❖ To view a template description:

1 Click the button in an Power++ Extended Attributes dialog box.

2 Select a template from the Template Name dropdown listbox.

3 Select a template section to view from the Show Section dropdown listbox.

The section you select is displayed in a scrollable window.

4 Click OK.

Using DataWindow templates

P++Gen uses DataWindow attributes (edit styles, formats, and validation rules) to define the appearance of columns in a DataWindow that it generates. Field templates attached to columns are not used for DataWindow generation.

For information on DataWindow attributes, see "Defining catalog attributes for a DataWindow" on page 444.

If you use the DataWindow template DWFFRM3D.WDT that ships with AppModeler, you must use the SINGLEDW.WFT or the MDETLDW.WFT template to generate a form that can contain the DataWindow.

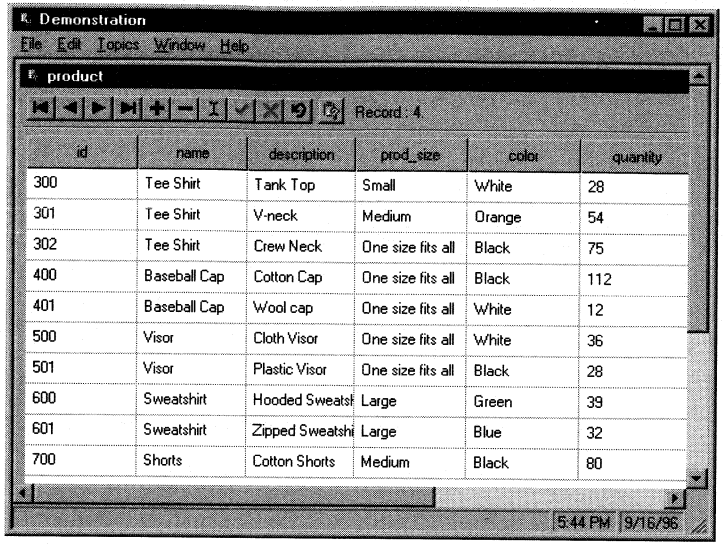
Using ORCA
You must include the path for the Powersoft Shared PBORC050.DLL file in the AUTOEXEC.BAT path to use DataWindow objects with P++Gen.

Using grid templates

If you attach a grid template to a form, P++Gen generates column extended attributes in addition to a grid control. However, the grid templates that ship with AppModeler do not use these attributes or the field templates attached to the columns. All form data generated with P++Gen grid templates is presented in the same grid format.

Setup installs GRID.WGT as the default grid template. The LISTVIEW.WGT grid is read-only and cannot be used to update a data source.

An example of a standard grid generated in a Power++ data form is shown below:



Customizing templates and template sets

You can create your own templates from existing projects and then add template path information directly to the registry from P++Gen.

From P++Gen, you can also create a customized template set and add the set name to the Template Sets registry subkey.

Creating project and form templates

You can create your own project templates and form templates to use with P++Gen.You should save the template files that you create to a new template set directory.

❖ **To create project and form templates:**

1 Create and save a Power++ project.

2 Using a text editor, replace names and identifiers in your project and form files with AppModeler system variables.

You use variables for names and identifiers that you want P++Gen to instantiate from the PDM.

For information on AppModeler system variables, see the appendix "Generation Variables and Attributes."

3 Copy and rename the PROJECT.WPT project template, keeping the WPT extension.

4 Make changes to the new project template as needed.

5 Save your WRES.WXR Power++ project resource file to your template set directory.

6 Rename your Power++ WXP and WXT project files using the filename prefix of your WPT project template.

7 Copy and rename the MDIMAIN.WMT main form template, keeping the WMT extension.

8 Make changes to the new main form template as needed.

9 Rename your main form WXF file with the filename prefix of your WMT template.

10 Copy and rename the SINGLE.WFT file or the MSTRDETL.WFT file, keeping the WFT extension.

11 Make changes to the new form template as needed.

12 Rename your form WFM files with the filename prefix of your WFT template.

Creating field templates

You create field templates by placing controls in an open Power++ form. You can isolate the code for the controls and save it to a new file that you name with a WBT extension. The controls must be included in a @Controls section of the WBT field template. You can replace the names in a field template with system or user-defined variables.

Using variables in field templates

P++Gen can instantiate an attribute that is defined by a variable in the @Attributes section of a field template. If you refer to this attribute in other sections of the template, the attribute must be surrounded by percent signs.

%FieldCode% variable

If you create your own field templates, you can name the template controls using the %FieldCode% variable. P++Gen can instantiate the %FieldCode% variable of a control identifier with an index number that it concatenates to the column code. This ensures that each control is unique, even if it is used more than once in the same form.

For example, a label control description from the TEXTBOX.WBT field template uses the %FieldCode% variable:

```
@begin Object "WLabel"
 WSS%?lblAlignment% 1;
 WSSNotify 1;
 WSChild 1;
 WSVisible 1;
 Text "%ColumnName% :%clrWindowBackground";
 ResID -1;
 DesignName lbl_%FieldCode%;
 DesignRect 3,6,%lblWidth%,%lblHeight%;
@end;
```

The label control name is lbl_*%FieldCode%*. P++Gen replaces %FieldCode% by the column code, and an index number, if necessary, to differentiate two labels for the same column (for example, in master/detail forms where the column names are the same for fields in the master and detail sections).

%ColumnWidth% variable

The %ColumnWidth% variable defines the width of a textbox control based on the length of the text declared for the column multiplied by a constant. The UnitsPerChar attribute in P++Gen data form templates defines this constant as eight Power++ units per character.

The TEXTBOX.WBT field template uses the %ColumnWidth% variable as the default value for the textbox control it defines:

```
lblWidth          78 ' Description label width
lblHeight          8 ' Description label height
lblAlignment    Left ' Alignment of text in label box

txtLocked         -1 ' Lock the ForeignKey
ColumnIsInteger      ' Indicate if column is integer

Width %ColumnWidth% ' (computed) width of textbox
Height            12 ' Height of textbox
```

Using switches in variable names

Undefined variables

If you type a **?** after the first percent sign in a variable name in a field template, P++Gen only generates the line containing the variable if the variable is not null. The variable can be a system variable or a user-defined variable.

For example, the template description of a textbox control shown below uses the system variable %?ColumnLength%. If the PDM does not have a value for the length of a column you generate with this template, P++Gen does not generate the TextLimit line that contains this variable.

```
@begin Object "WTextBox"
 WSChild 1;
 WSExClientEdge 1;
 WSVisible 1;
 DataColumns "%DataField%";
 DataSource "%FormName%::%DataSource%";
 TextLimit "%?ColumnLength%";
 ToolTipText "%.Q:ColumnLabel%";
 ResID -1;
 DesignName textb_%FieldCode%;
 DesignRect 84,3,%Width%,%Height%;
@end;
```

Formatting variables

You can use the special formatting switch **.Q:** in a variable to protect quotation marks in objects you generate with P++Gen. As with other formatting switches, you insert it after the first percent sign of a variable name.

At the moment of instantiation, P++Gen adds a backslash before quotation marks in text generated from the line containing the formatting switch. Power++ can then recognize the quotation mark as a literal character.

For information on other formatting switches that you can use with AppModeler variables, see the chapter "Database Creation and Modification."

Adding a template or a field style

Adding a template

You can add project, form, field, or grid templates to the registry directly from P++Gen. You must include a template name and filename for each template that you add.

You must select a template type for each template that you add, and you must type a template name and filename. The new template name is included as a string value in the registry under the template type subkey you select.

Adding a field style

For each field style that you add, you must include a field style name and a default template filename. After you validate the new field style name, the new field style appears in the list of template types for your current template set. You can then add other field templates for this template type, in addition to the template that you selected as the field style default.

For long lists of field styles, a scroll bar in the Type dropdown listbox permits you to view all available P++Gen template types.

Your field style name appears as a string value in the registry under the Field Styles subkey. The default template filename appears as a data value for the new field style name.

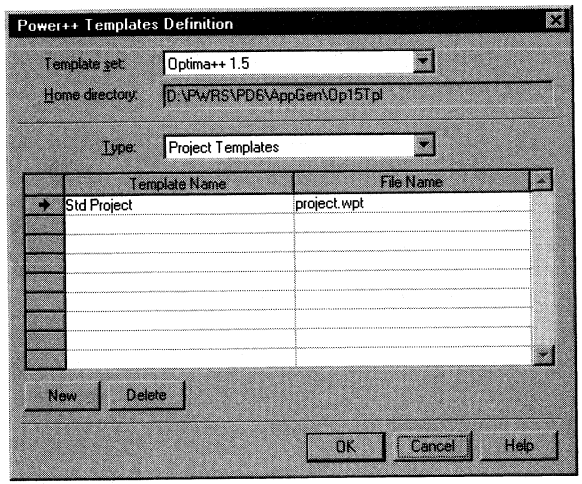
❖ **To add a template or a field style:**

1 Select Client➤Power++ Model Attributes.

The Power++ Model Extended Attributes dialog box appears.

2 Click the Templates button.

The Power++ Templates Definition dialog box appears.



3 Select the template set to which you want to add a new template or field style.

Only template sets listed in your registry are included in the Template Set dropdown listbox.

4 Select a template type from the Type dropdown listbox.
or
Select Field Styles from the Type dropdown listbox.

5 Click the New button.

An arrow appears at the beginning of the first blank line.

6 Type a name in the Template Name column.
or
Type a name in the Field Style Name column.

7 Click the File Name column for your new template.
or
Click the Default File Name column for your new field style.

A button appears in the column.

8 Click the button to browse available drives.
Select a filename and click OK.
or
Type a filename in the column.

You can type the complete path and filename, or you can use the relative path from the template set home directory.

9 Click OK.

Adding template sets

The AppModeler Setup program installs a template set for Power++ 1.5. You can create your own template sets to access custom-designed P++Gen templates.

You can select a home directory for each template set and use relative addresses for your individual template files. When you add templates to a template set, you can always use a full address for the location of a template file that is not in the template set home directory.

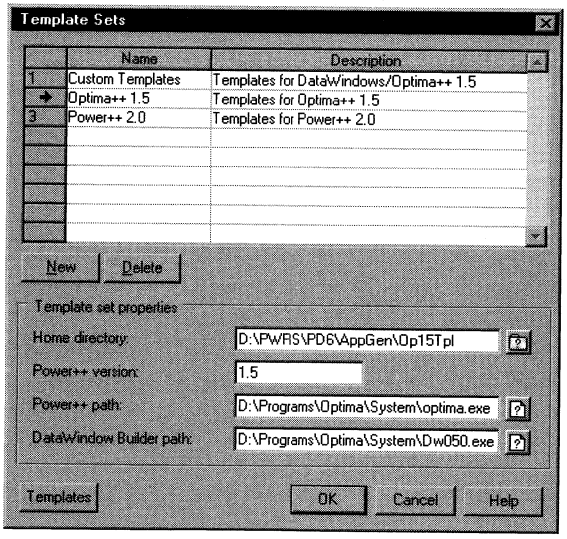
❖ To add a template set:

1 Select Client➤Power++ Model Attributes.

The Power++ Model Extended Attributes dialog box appears.

2 Click the button next to the Template Set box.

The Template Sets dialog box appears.



3 Click the New button.

The arrow moves to a new row in the list of template sets.

4 Type a name for your new template set.

5 (Optional) Type a description for the template set.

6 Click the button next to the Home Directory box.
Select the Home directory for your template set and click OK.
or
Type the directory for your template set in the Home Directory box.

7 Type the Power++ version you want to use with your template set.

8 Click the button next to the Power++ program box.
Select your Power++ program and click OK.
or
Type the path and filename for your Power++ program.

9 Click the button next to the DataWindow Builder Path box.
Select the path for your DataWindow Builder program and click OK.
or
Type the path and filename for your DataWindow Builder program.

10 Click OK.

Your template set is added as a new string value to the Template Sets registry subkey, and as a new key under the AppModeler for Power++ subkey.

CHAPTER 18

Delphi Generator

About this chapter

This chapter provides an overview of the AppModeler Delphi Generator (DelphiGen) and explains how it works.

Contents

Before you begin

DelphiGen works with the Enterprise, Professional, and Desktop versions of Delphi 2.0.

Generator Basics

DelphiGen is the AppModeler generator for Delphi. It generates Delphi projects, forms, and controls from objects and attributes that you define in a PDM. AppModeler ships with predefined templates that you can use with DelphiGen for Delphi object generation.

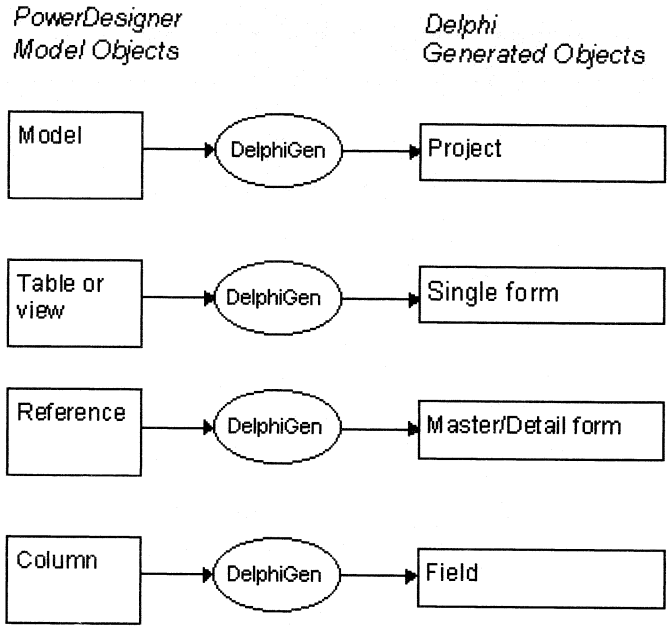
You start AppModeler for Delphi by double-clicking PDDL6.EXE or by clicking its icon in the Program Manager or taskbar.

What DelphiGen does

DelphiGen uses information from a PDM to instantiate predefined templates and generate Delphi applications. DelphiGen generates an entire Delphi application, including:

Delphi file type	Description
DPR	Main project file
DFM	Form files

The following schema shows the object transformations performed by DelphiGen:



Installed files and directories

AppModeler for Delphi includes the following disk files that you install with the Setup program:

Filename	Directory*	Description
PDDL6.EXE	PD6	Executable file
DELPHI.EXA	PD6\EXA	Default extended attribute import file

* Only short path names are shown in the table. Long names are PowerDesigner 6 (PD6) and Extended Attributes (EXA).

DelphiGen templates

AppModeler for Delphi ships with several predefined templates and resource files. The Setup program installs them in the APPLICATION GENERATORS\DELPHI 2 TEMPLATES directory (APPGEN\DL2TPL directory if you install AppModeler using short names).

References to template files are included in subkeys of the registry path HKEY_CURRENT_USER\ Software\Powersoft\ PowerDesigner 6\ AppModeler for Delphi\Delphi *version*.

Project and form templates

Project and form templates that ship with DelphiGen are:

Filename	Registry subkey	Template type
PROJECT.DPT	Project Templates	Project
MDIMAIN.DMT	Main Form Templates	Main form
SINGLE.DFT	Table Form Templates View Form Templates	Form for single data source
MSTRDETL.DFT	Reference Form Templates	Form for master /detail data source
STDTFLD.DTT	TField Objects Templates	TField objects visible at design time and runtime

Associated project and form files

The Setup program installs other project-related and form-related template files in the DelphiGen template directory. DelphiGen copies or generates and modifies these files in your project directory.

Filename	How DelphiGen uses it	Result file
PROJECT.DPR	Generates a project file	*NAME*.DPR* *NAME*.LOG*
PD_TOOLS.PAS	Copies for data control functions and project status bar	PD_TOOLS.PAS
MDIMAIN.PAS MDIMAIN.TXT	Generates a main form	*NAME*.DFM* *NAME*.PAS* *NAME*.TXT*
SINGLE.PAS SINGLE.TXT MSTRDETL.PAS MSTRDETL.TXT	Generates data forms	*FORMNAME*.DFM# *FORMNAME*.PAS# *FORMNAME*.TXT#
ABOUTBOX.DFM ABOUTBOX.PAS	Generates About box that you can open from the project Help menu	ABOUTBOX.DFM ABOUTBOX.PAS

* You enter this name on the General page of the Delphi Model Extended Attributes dialog box. The Project Name is used for the project files and the Application Form Name for the main form file.

You enter this name in the Form Filename box on the General page of the Delphi *Object* Extended Attributes dialog box.

Field templates

Field templates affect the way data is presented in forms that use bound controls. DelphiGen ships with the following field templates:

Filename	Registry subkey	Template type
COMPUTED.DBT	Computed Field Templates	Computed fields
CHECKBOX.DBT	CheckBox Field Templates	Checkbox
DBCOMBO.DBT CHKLIST.DBT ECHKLIST.DBT	DropDown Field Templates	Combination box
IMAGE.DBT	Image Field Templates	Bitmap
RADIOGRP.DBT	RadioButton Field Templates	Radio buttons
LABELBOX.DBT MLTXTBOX.DBT SPIN.DBT TEXTBOX.DBT	Edit Field Templates	Text area

Grid templates

The Setup program installs the grid template, GRID.DGT. You use this template to generate the standard Delphi grid control.

Ensuring that your project refers to an existing database

The projects you develop with DelphiGen use information from existing databases. You must ensure that the database exists and contains the fields and data types defined in your PDM before your project can work properly.

You must also assign an alias to your database using the Database Explorer or the BDE Configuration Utility that ships with Delphi.

For information on using the BDE Configuration Utility, see the Delphi online documentation.

❖ **To ensure that your project refers to an existing database:**

- Generate a database directly from the PDM.
  or
  Reverse engineer an existing database to create the PDM.

Importing extended attributes for Delphi

The DELPHI.EXA file contains default attribute values for object generation from DelphiGen. The DELPHI.EXA attribute values are loaded automatically into a new PDM. You can import these attributes and values into any PDM built with PowerDesigner or S-Designor.

❖ **To import extended attributes for Delphi:**

1. Select Dictionary➤Extended Attributes➤Import Attributes.
2. Select the DELPHI.EXA file.
3. Click OK.

   A message box tells you that the extended attribute import is successful.
4. Click OK.

Building a Delphi project

A project file manages a Delphi application. Project files contain addresses and filenames for Delphi forms and for other project resources.

A project file generated by DelphiGen includes information about project templates and project properties.

Selecting a project template and project properties

DelphiGen generates a project file by instantiating a project template with values from the PDM. You attach project and main form templates to the PDM and define certain properties:

Property	Description
Template set	Delphi 2.0 or customized template set*
Project template	Name of template you use to build your project
Project name	Filename you generate for the project
Project title	Title for the project
Project directory	Directory in which you generate project files
Application form template	Name of template you use to build your main form
Application form name	Filename you generate for the main form
Menu items position	Menu bar position for data form menu items

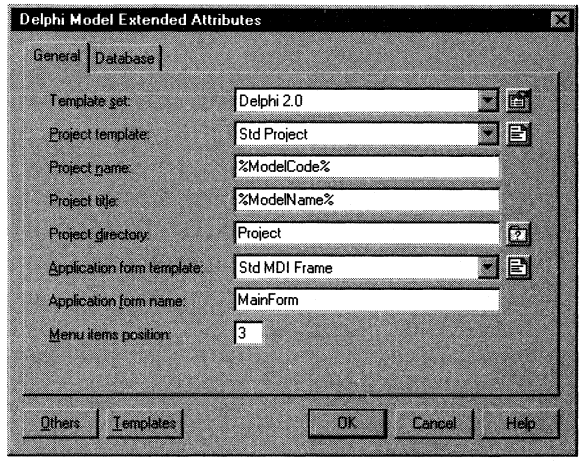
* The Template Set dropdown lists any template set that you name as a string value in the Delphi Template Sets registry key.

DelphiGen can create a new project directory if you type the directory name in the Project Directory box. If you do not type a complete path, DelphiGen looks for the directory in the PWRS\PD6 path, or in its long name equivalent.

❖ To select a project template and project properties:

1 Select Client➤Delphi Model Attributes.

The Delphi Model Extended Attributes dialog box opens to the General properties page.



2 Select a target template set from the Template Set dropdown listbox.

Only template sets listed in your registry are available in the dropdown listbox.

For information on adding template sets to your registry from DelphiGen, see "Adding a template set" on page 523.

3 Select a project template from the Project Template dropdown listbox. Select a main form template from the Application Form Template dropdown listbox.

The Setup program installs the Std Project and Std MDI Frame templates as the default selections.

4 (Optional) Type changes to Project Name and Application Form Name.

These are filenames. You do not need to add DPR or DFM extensions, because DelphiGen adds them automatically during generation.

5 Use the button to select a project directory.
or
Type a directory name in the Project Directory box.

The files you generate with DelphiGen will be saved in this directory.

6 Type changes to any of the other General page properties that you want to implement in your application.

7 Click OK.

Defining database properties

Connection information

To generate an application that displays information from a database, you must specify connection information. You must define a database alias in Delphi before you can assign it to the project you generate with DelphiGen.

If you select the Login Prompt checkbox, users of the project you generate will not be able to open the project without logging in. If you clear the Login Prompt checkbox, Delphi connects automatically to the data source using the user name and password that you specify.

> **Clearing the Login Prompt checkbox**
> If you clear this checkbox, you must be certain that the user name and password information you enter in DelphiGen is correct. Otherwise, users of your project will not be able to connect to the database.

When no data forms are open in the project you generate, Delphi closes the connection to the data source unless you select the Keep Connection checkbox.

Transaction isolation

Delphi supports three levels of transaction isolation:

Isolation level	Database data display
Dirty Read	Uncommitted changes
Read Committed	Real-time committed changes
Repeatable Read	Committed changes at time of connection for duration of project session

You designate an isolation level for your project in DelphiGen. If your DBMS does not support the isolation level you select, Delphi defaults to an isolation level that is supported.

SQL queries

If the table and column codes in your PDM contain special characters (blank spaces, slashes, and so on), DelphiGen ensures that quotation marks are generated around these names in the SQL queries of your Delphi project.

Using blank spaces in column codes
The use of blank spaces in column codes is not recommended. If you use column codes containing blank spaces in views, queries generated from these columns may not execute properly.

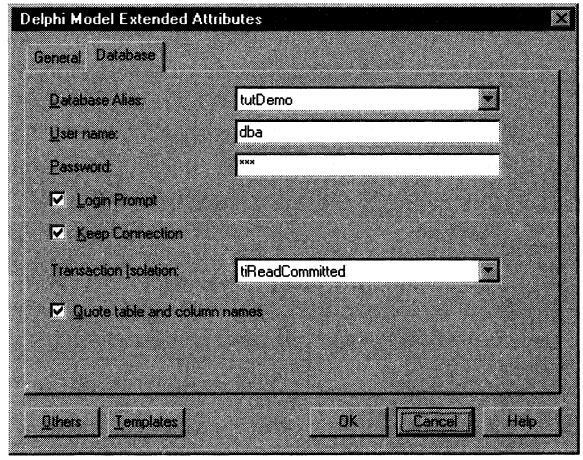
If you select the Quote Table and Column Names checkbox, DelphiGen generates quotation marks around every table and column name used in the SQL queries of your project.

❖ To define database properties for your project:

1 Select Client➤Delphi Model Attributes.

2 Click the Database tab.

The Database properties page opens.



3 Select the data source for your project.

The Database Alias dropdown listbox only includes data sources you have defined in the Database Explorer or the BDE Configuration Utility.

4 Type the name and password needed to access the database.

5 Select the Login Prompt checkbox.
or
Clear the Login Prompt checkbox.

6 Select the Keep Connection checkbox.
or
Clear the Keep Connection checkbox.

7 Select a level from the Transaction Isolation dropdown listbox.

8 Select the Quote Table and Column Name checkbox.
or
Clear the Quote Table and Column Name checkbox.

9 Click OK.

Defining Delphi forms and fields

Forms are the primary interface elements of Delphi projects. Each form must have its own DFM file that corresponds to an individual table, view, or reference.

The term **data form** is used to distinguish child forms from the main form (frame window) of a project. Data forms contain data that is presented in fields that you define with DelphiGen field templates.

Fields are form elements that correspond to database columns. In DelphiGen, field templates define a Delphi control or set of controls that can display data from the database.

Controls are form or field elements that you can use in applications to get user input or to display output. Textboxes, listboxes, and command buttons are examples of standard Delphi controls.

Selecting form templates and form properties

For each table, view, and reference in the PDM, you can define the following general properties:

Property	Description
Generate form	Select to generate a form
Form template	Name of template you use to build a form
Form name	Name of form in the Delphi project window
Form title	Title for a form
Form filename	Filename for a form (DelphiGen adds a DFM extension)
Menu option checkbox	Select to generate a menu item to open a form
Menu option textbox	Name of the menu item that will open a form

What is a menu option?

The Menu Option checkbox indicates whether or not a menu item will be generated for a form. The menu option name is the name of the menu item that opens the form in the project you generate. If you want DelphiGen to generate the menu option name as a menu item, you must select the Menu Option checkbox.

Menu hot key
You define a menu hot key by typing an ampersand (&) before a letter of the menu option name. Pressing the hot key opens the data form from the generated frame and sheet menus.

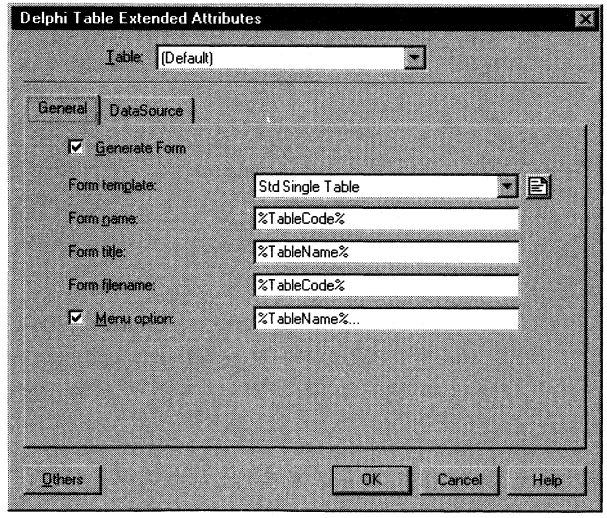
Defining general properties

You can use default general properties for a PDM object type or define form properties for each table, view, and reference.

❖ To define general properties:

1 Select Client➤Delphi Table Attributes.
or
Select Client➤Delphi View Attributes.
or
Select Client➤Delphi Reference Attributes.

The Delphi *Object* Extended Attributes dialog box appears.



2 Select a table from the Table dropdown listbox.
or
Select a view from the View dropdown listbox.
or
Select a reference from the Reference dropdown listbox.

If the dropdown listbox displays Default, any changes to object properties that you make in this dialog box apply to all objects of the same type for which these properties have not been previously modified. This includes all new objects that you add to the PDM.

3 Select a form template from the Form Template dropdown listbox.

4 Make changes to other properties, as needed.

5 Click OK.

Defining data source properties

The data source properties you select in DelphiGen are:

Property	Description
Updatable	Permits update of database from your Delphi project
Presentation Style	Sets display format of form
TField Objects Template	Generates TField objects for the columns of the data source for the form

You can assign or modify the data presentation style for generated forms. DelphiGen uses the following presentation styles:

Style	How it displays data	How it affects data presentation
Free Form	In bound controls	Different bound controls may be used for different fields in the same form
Grid	In a grid format	All fields in the form appear in a grid

You can display certain bound controls, such as the dropdown listbox, inside a grid. However, if you generate a form in the grid presentation style without selecting a TField Objects template, these controls do not appear in the grid.

Defining data source properties for a table or view

Default values

The DELPHI.EXA default presentation style for a table is free form. The default presentation style for a view is grid. The default grid template is GRID.DGT.

The DELPHI.EXA default for a TField objects template is the Std TField template that ships with AppModeler. At generation, DelphiGen assigns a TField type for each field of a form that you generate with the Std TField template.

Problem data types

If you use the default TField objects template, the TFields that DelphiGen generates may have a wrong type or size for certain column data types. DelphiGen can generate correct values for TFields if you assign these values to the TFieldType and TFieldSize attributes in the Delphi Column Extended Attributes dialog box.

One way to determine the correct TField values is to select None from the TField Objects Template dropdown listbox before you generate a form. In this case, Delphi will calculate the correct TField type and size for the generated objects.

At design time in Delphi, you can view the TField values for type and size in the Delphi TQuery object. You can then assign these values to column attributes in DelphiGen and generate the form a second time using the Std TField template.

For information on assigning attributes to columns, see "Attaching a field style and field template to a column" on page 496.

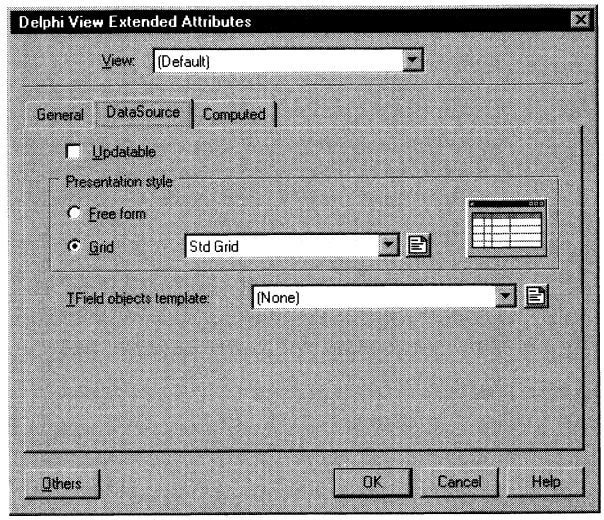
Selecting a TField objects template
The disadvantage of selecting None for a TField objects template is that DelphiGen does not generate any formatting attributes for TField objects.

❖ **To define data source properties for a table or view:**

1 Select Client➤Delphi Table Attributes.
or
Select Client➤Delphi View Attributes.

2 Click the Datasource tab.

The Datasource page opens.



3 Select a table or view from the Table or View dropdown listbox.

4 Select the Updatable checkbox.
or
Clear the Updatable checkbox.

5 Select a Presentation Style radio button.

6 Select a TField template from the TField Objects Template dropdown listbox.

7 Click OK.

Defining data source properties for a reference

A reference has a master data source and a detail data source. The data source can be a table or view. Depending on your selection, the objects you generate from a reference can use information from table or view columns.

Autoincrement field

For certain DBMS, if the primary key of a master data source includes an autoincrement field, you must generate TFields for that data source. In these cases, you must not select None for the TField objects template of the master data source.

TField objects template

If you use the DELPHI.EXA default TField objects template for a master or detail form, DelphiGen generates TFields for the form. However, you may still need to assign size and type attributes to certain columns in the form.

For more information on using the default TField objects template, see "Defining data source properties for a table or view" on page 490.

❖ To define the data source properties for a reference:

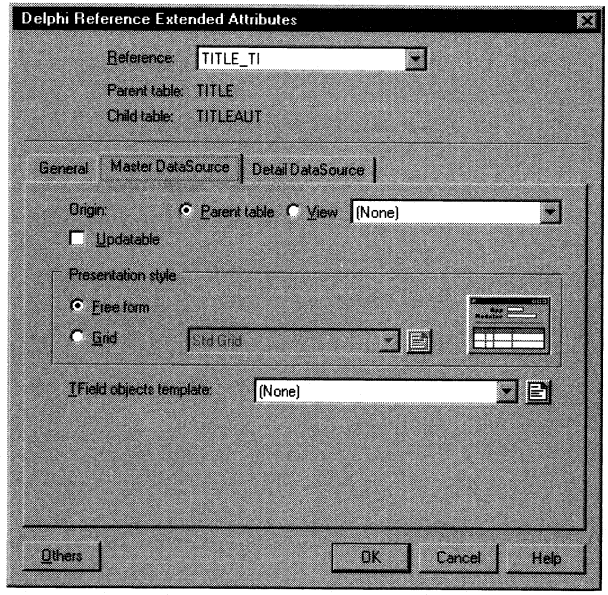
1 Select Client➤Delphi Reference Attributes.

The Delphi Reference Extended Attributes dialog box opens to the General page.

2 Select a reference from the dropdown listbox at the top of the Delphi Reference Extended Attributes dialog box.

3 Click the Master Datasource tab.

The Master Datasource page opens.



4 Click the Parent Table radio button.
or
Select a view from the View dropdown listbox.

Only views that contain the parent table primary keys are available from the dropdown listbox. Selecting a view from the dropdown listbox automatically selects the View radio button.

5 Select the Updatable checkbox.
or
Clear the Updatable checkbox.

6 Select a data presentation style for the master form.

Free form is the default presentation style for a master form.

7 Select a template from the TField Objects Template dropdown listbox.

If you select None, Delphi generates TField objects for your project at runtime.

8 Click the Detail Datasource tab.

The Detail Datasource page opens.

9 Click the Child Table radio button.
or
Select a view from the View dropdown listbox.

Only views that contain a foreign key of the selected reference are available from the dropdown listbox.

10 Select the Updatable checkbox.
or
Clear the Updatable checkbox.

11 Select a data presentation style for the detail form.

Grid is the default presentation style for a detail form. GRID.DGT is the default grid template.

12 Select a template from the TField Objects Template dropdown listbox.

If you select None, Delphi generates TField objects for your project at runtime.

13 Click OK.

Using computed fields

Computed fields template

If you use calculated expressions in a view, you must use a computed fields template to generate a form based on that view.

DelphiGen ships with a single computed fields template that enables you to display all calculated expressions. DelphiGen automatically assigns this template to all columns using these expressions.

If you create your own computed field templates, you can assign different computed fields templates to different views. However, you cannot assign different computed fields templates to different columns in the same view.

TField objects template

You can also generate TFields for columns containing calculated expressions, but you must assign a TField object type to the column in addition to a TField objects template.

If you assign a TField objects template to a computed field, you must also assign a TField objects template to other columns of the view that do not contain calculated expressions, and vice versa.

TField objects type

For a view containing two or more calculated expressions that have diffferent data types, you can modify the TField objects type of each column, but not from the Delphi View Extended Attributes dialog box. The TField objects type is a column extended attribute that you can change from the Application Generation window.

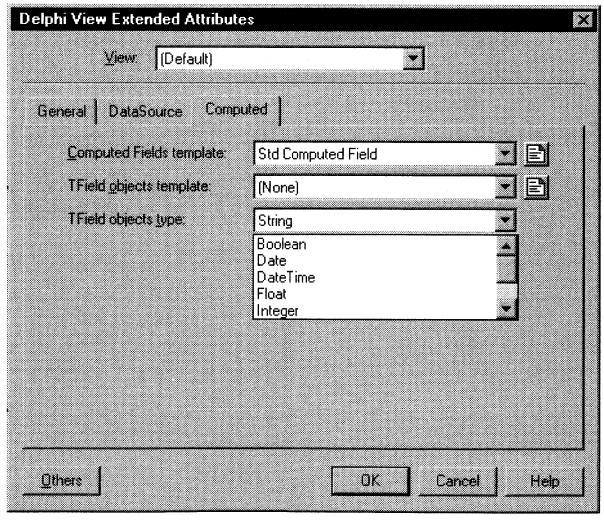
For information on assigning different TField objects types to different columns of the same view, see "Modifying extended attributes of a column" on page 507.

❖ To use computed fields in your project:

1 Select Client➤Delphi View Attributes.

2 Click the Computed tab.

The Computed page of the Delphi View Extended Attributes dialog box opens.



3 Select a template from the Computed Fields Template dropdown listbox.

4 Select a template from the TField Objects Template dropdown listbox.

If you select None, Delphi generates TFields at runtime for the columns containing calculated expressions.

5 Select a TField object type from the TField Objects Type dropdown listbox.

6 Click OK.

Defining field attributes

To use information from a database in forms generated with bound controls, you need to attach field styles and templates to columns or domains.

The default field style is Edit. TEXTBOX.DBT is the default field template for the Edit field style. The default field template for a field style is modifiable in the Field Styles subkey of the registry.

For information on adding field styles or changing default field templates, see "Adding a template or a field style" on page 521.

Attaching a field style and field template to a column

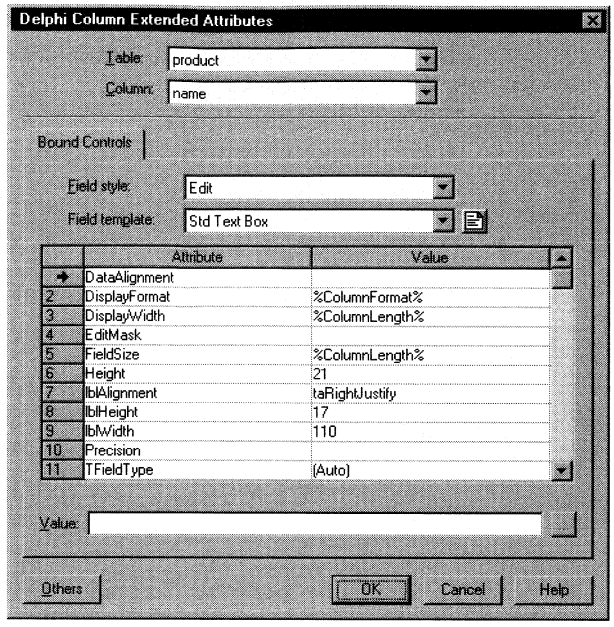
A field style indicates the type of Delphi controls to generate for a field in a data form that uses the free form presentation style. A field style can use any standard or personalized Delphi control that you define in a field template.

Each field template has a list of attributes. You can modify the values of these attributes from DelphiGen.

❖ **To attach a field style and field template to a column:**

1 Select Client➤Delphi Column Attributes.

The Delphi Column Extended Attributes dialog box appears.



2 Select a table from the Table dropdown listbox.

3 Select a column from the Column dropdown listbox.

4 Select a style from the Field Style dropdown listbox.

5 Select a template from the Field Template dropdown listbox.

6 Double-click an item in the Attribute column.

The attribute is selected and the cursor moves to the Value box.

7 Type a value for the field style attribute in the Value box.

8 Double-click another attribute and change its value if needed.

9 Click OK.

Attaching a field style and field template to a domain

You can attach a field style and field template to a domain. These apply automatically to any columns you attach to the domain. You can also update the values of field template attributes for columns attached to the domain.

❖ **To attach a field style and field template to a domain:**

1 Select Client➢Delphi Domain Attributes.

The Delphi Domain Extended Attributes dialog box appears.

2 Select a domain from the Domain dropdown listbox.

3 Select a field style from the Field Style dropdown listbox.

4 Select a template from the Field Template dropdown listbox.

5 Double-click an item from the Attribute column.

The attribute is selected and the cursor moves to the Value box.

6 Type a value for the field style attribute in the Value box.

7 Select another attribute and change its value if needed.

8 Click OK.

A message box asks if you want to update the columns attached to the domain.

9 Click Yes.
or
Click No.

Generating a project

You can generate a Delphi application from objects you select in the PDM.

Tree view items

The objects that you want to generate must be listed in a tree view in the Application Generation window. This window contains an object list in a tree view with the following items:

- Project object
- Main form object
- Forms for tables, views, and references you select from the PDM
  or
  Forms corresponding to filters you set in the last DelphiGen session
- Columns of each form

> **Your first DelphiGen session**
> If you do not select any objects in the PDM and you did not use a selection filter in a previous DelphiGen session, form objects are listed in the Application Generation window for every table, view, and reference.

Checking items to be generated

Each item in the tree view that you want to generate must have a checkmark in front of it. You can use the Select All button to check all form and project level items in the tree view, or you can use the Unselect All button to clear these checkmarks.

Generating a menu

If you use the MDIMAIN.DMT template to generate a main form, you also generate a main form menu bar containing three menu items: File, Topics, and Help. Two additional menu items, Edit and Window, are not visible until you open a data form.

If you generate your project using the menu option, and you do not change the menu position default (menu position = 3), the data forms you generate are accessible from the Topics menu.

The Edit menu allows a user of your project to cut, copy, and paste information in the data forms, or to undo a previous action. The Window menu provides access to any open data forms and allows you to arrange them in a tile or cascade display.

For more information on data form menu options, see "Selecting form templates and form properties" on page 488.

Generating a project from selected objects

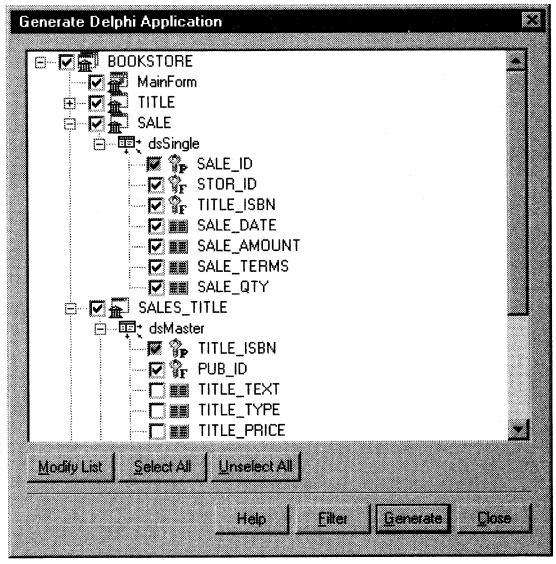
To create a list of objects to generate, you can select objects directly from the PDM or you can reuse objects you selected in the last DelphiGen session.

❖ To generate a project from selected objects:

1 Use selection tools to select objects in the PDM.
or
Verify that no objects are selected in the PDM to reuse objects selected during the previous DelphiGen session.

2 Select Client➤Generate Delphi Application.

The Application Generation window appears.



3 Click any plus signs to the left of tree view items.

All tree view items are now visible.

4 Verify that all objects you want to generate are checked.

5 Click the Generate button.

Reviewing the generation log

You can follow the progress of the generation from the log window that appears after the generation begins. DelphiGen indicates if there are any errors in the generation or if the generation is successful. Before DelphiGen attempts to generate a file with the same name as an existing file in the project directory, a message box asks you for confirmation.

The project log file, *MODELCODE*.LOG, is created automatically in your project directory. It contains model and generation information. You can read the log file with a text editor.

Modifying the generation selection

There are several ways to modify a preliminary object selection from the Application Generation window. To modify the object selection, you can:

- Set the generation flag for objects in the tree view
- Add objects to or remove objects from the tree view
- Apply a filter

Setting the object generation flag

A checkmark to the left of an item in the tree view indicates that its generation flag is selected.

Project and form items

By clicking the Select All button beneath the tree view, you can set the generation flags for all project and form items. By clicking Unselect All, you clear the generation flags of all project and form items in the tree view. You must expand the tree view to list the column items in the tree view.

Column items

The Select All and Unselect All buttons do not affect column items in the tree view. You can only clear the checkboxes of column items one by one.

You cannot clear the generation flag of a primary key column. The tree view does not display checkboxes for columns in a view or for columns in a reference that use a view as their data source. These columns are always generated when you generate their parent form.

Items sharing a data source

Two references can share the same data source (parent table or child table). Changes to column generation flags of the master or detail form of one of the references are automatically reflected in the columns of forms based on the other reference. However, changing the column generation flag for a table does not affect the generation flag for the column of a master or detail form that uses the table as its data source.

❖ **To set the generation flags for objects in the tree view:**

1 Click the plus signs in front of tree view items.

You expand the tree view.

2 Select checkboxes to the left of the objects you want to generate.

3 Clear checkboxes to the left of objects you do not want to generate.

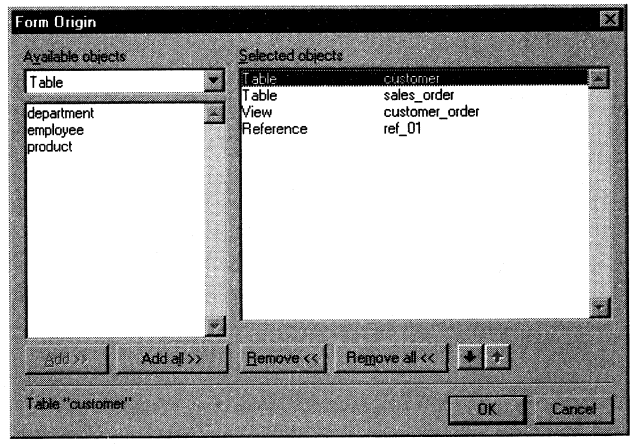
Adding or removing objects from the tree view

If you modify the list of objects in the tree view without using a filter, the modified selections you make are only valid for the current DelphiGen session. If you close the Application Generation window without generating a Delphi project, DelphiGen does not save your modified selections.

❖ **To add or remove objects from the tree view:**

1 Click the Modify List button below the tree view.

The Form Origin dialog box appears.



2 Select a PDM object type from the Available Objects dropdown listbox.

The Available Objects listbox displays all objects of the type selected that are not in the tree view. The Selected Objects listbox displays all objects that are in the tree view.

3 Select objects in the Available Objects listbox and click Add.
or
Select objects in the Selected Objects listbox and click Remove.

Selected objects move from one listbox to the other.

> **Using the Add All or Remove All buttons**
> You can click the Add All or Remove All buttons to move all objects of a given type from one listbox to the other.

4 Repeat steps 2 and 3 for other PDM object types.

5 Click OK.

You return to the Application Generation window. The objects you added to the Selected Objects listbox appear in the tree view. Objects you removed from this listbox disappear from the tree view.

Applying a filter to modify object selection

Filters help handle large amounts of information. A large PDM can include hundreds of objects. You can use a filter to augment or restrict the items included in the tree view in the Application Generation window.

After you apply a filter, only the matching items appear in the tree view. The selections you make with a filter are saved by DelphiGen, whether or not you generate your selections during the current session.

Types of filters

There are four types of filters:

Filter	Action
All objects	Selects every object in the PDM
Objects of submodel	Selects objects from a named submodel
Modified objects	Compares objects in the current PDM to objects in an archived PDM and selects objects that have changed
User-defined list	Lets you select a list of objects from the PDM

You can select or clear object type checkboxes with any of the filter types.

User-defined filter

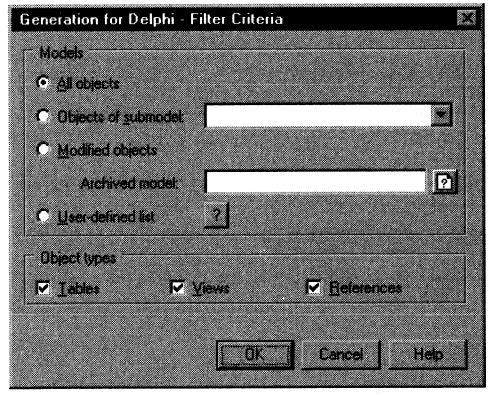
If you use the user-defined filter, you can add any object in the PDM to the tree view. You can use the buttons in the Form Origin dialog box to make customized filter selections:

Button	Function
Add	Transfers a selected object from the Available Objects listbox to the Selected Objects listbox and adds it to the tree view
Add All	Transfers all objects of the selected object type
Remove	Transfers a selected object from the Selected Objects listbox to the Available Objects listbox and removes it from the tree view
Remove All	Transfers all objects of the selected object type
	Moves the selected object up in the list order of Selected Objects and in the tree view
	Moves the selected object down in the list order of Selected Objects and in the tree view

❖ To select objects using user-defined filter criteria:

1 Click the Filter button in the Application Generation window.

The Filter Criteria dialog box appears.



2 Click the  button beside the User-Defined List option.

The Form Origin dialog box appears.

3 Select a PDM object type from the Available Objects dropdown listbox.

The Available Objects listbox displays all objects of the type selected that are not in the tree view. The Selected Objects listbox displays all objects that are in the tree view.

4 Use the buttons of the Form Origin dialog box to add, remove, or reposition objects in the listboxes and in the tree view generation list.

5 Click OK.

You return to the Filter Criteria dialog box.

6 Verify that checkboxes are selected for the object types that you want to appear in the tree view.

7 Click OK.

You return to the Application Generation window. The tree view displays the new selections you made with the user-defined filter.

Fine-tuning before and after generation

Before you generate or regenerate your application, you can fine tune and test it from the tree view of the Application Generation window as follows:

- Modify model properties
- Modify extended attributes
- Select columns to generate
- Include or remove menu options
- Open the generated project

Modifying model properties

You can display or modify model properties and extended attributes from the context menus of the project or main form tree view items.

❖ **To modify model properties from the tree view:**

1. Right-click the project item in the tree view.
   or
   Right-click the main form item in the tree view.

   The item context menu appears.
2. Select Model Properties.

   The Model Properties dialog box appears.
3. Make changes to the model properties as needed.
4. Click the Extended button.

   The Delphi Model Extended Attributes dialog box appears.
5. Make changes to the extended attributes as needed.
6. Click OK in each of the dialog boxes.

Modifying extended attributes

You can display and modify extended attributes from the context menus of data form items in the tree view. Column extended attributes are also accessible from the tree view.

Modifying extended attributes of a table, view, or reference

❖ **To modify extended attributes of a table, view, or reference:**

1 Right-click a data form item in the tree view.

The data form item context menu appears. The context menu selection depends on the data source origin of the data form.

2 Select Table Attributes.
or
Select View Attributes.
or
Select Reference Attributes.

The Delphi Extended Attributes dialog box appears.

3 Make changes to the object extended attributes as needed.

4 Click OK.

Modifying extended attributes of a column

For columns containing calculated expressions, you can modify the TField object type. You can assign different types to different columns in the same view. However, when you assign a TField object type to a column from the tree view, your selection is only valid for the current DelphiGen session.

❖ **To modify extended attributes of a column:**

1 Right-click a , , , or column of a table, view, or reference.

The column context menu appears.

2 If the column is a calculated expression, select a TField object type.

3 Select Column Attributes from the column context menu.

4 Make changes to the column extended attributes as needed.

5 Click OK.

Selecting columns to generate in a data form

From the tree view, you can select which columns of a table you want to generate in a data form. You can also select which columns you want to generate in a master or detail data form of a reference, if it uses a table for its data source.

You must expand a form and its data source item to list the columns of that form. You expand a tree view item by clicking a plus sign to the left of it.

For more information on selecting individual columns for generation, see "Setting the object generation flag" on page 501.

Including or removing menu options

DelphiGen generates an application menu based on a main form template. You can select menu options to add to the application menu in object Extended Attributes dialog boxes. These menu options open data forms that you generate with DelphiGen.

From the Application Generation window, you can choose to generate a menu without menu options. If you generate without menu options, the menu options selections in the object Extended Attributes dialog boxes do not change. These menu options are saved in the PDM and you can use them the next time you generate your application menu.

❖ To include or remove menu options from the tree view:

1 Right-click the main form item in the tree view.

The main form item context menu appears.

2 Select Include Menu Options.

Selecting this item alternatively places or removes a checkmark before the item name. A checkmark selects the menu options for generation with the main menu.

Opening the project from DelphiGen

If you generated a project with DelphiGen, you can test it from the tree view.

❖ To open the project from DelphiGen:

1 Right-click the project item in the tree view.

2 Select Run Project from the context menu.

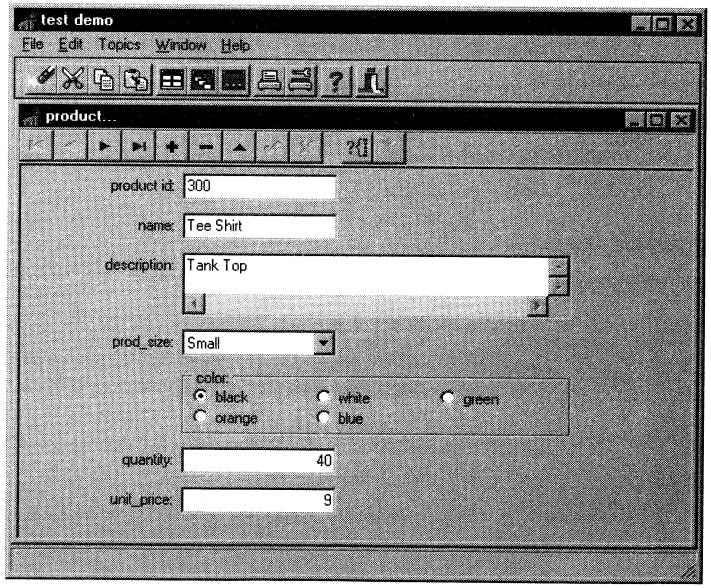
You open the project in Delphi, or a message box tells you that Delphi is already open.

> **Open Delphi file**
> If the open file in Delphi is an earlier version of your project file, close the file without saving it, or save it under a different filename. You can then open the project you generated using the File➤Open menu in Delphi.

Examples of forms based on predefined templates

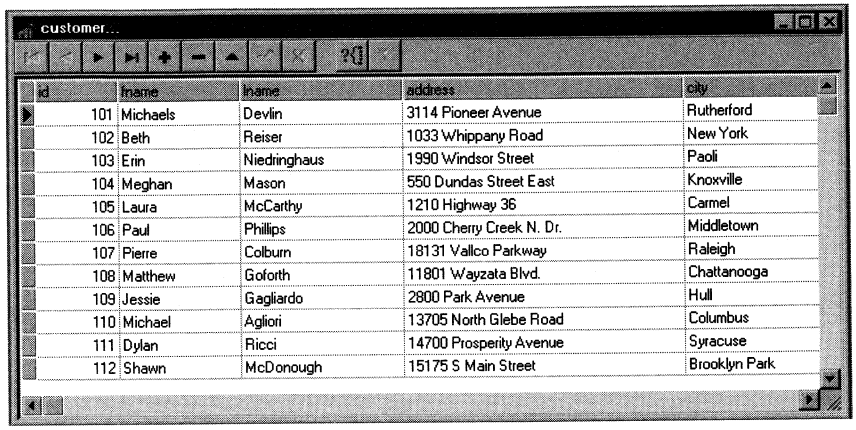
SINGLE.DFT in free form style

The following example shows a form generated using the SINGLE.DFT form template and the TEXTBOX.DBT, MLTXT.DBT, ECHKLIST.DBT, and the RADIOGRP.DBT field templates.



SINGLE.DFT + GRID.DGT

The following example shows a form generated using the SINGLE.DFT form template and the GRID.DGT grid template.



id	fname	lname	address	city
101	Michaels	Devlin	3114 Pioneer Avenue	Rutherford
102	Beth	Reiser	1033 Whippany Road	New York
103	Erin	Niedringhaus	1990 Windsor Street	Paoli
104	Meghan	Mason	550 Dundas Street East	Knoxville
105	Laura	McCarthy	1210 Highway 36	Carmel
106	Paul	Phillips	2000 Cherry Creek N. Dr.	Middletown
107	Pierre	Colburn	18131 Vallco Parkway	Raleigh
108	Matthew	Goforth	11801 Wayzata Blvd.	Chattanooga
109	Jessie	Gagliardo	2800 Park Avenue	Hull
110	Michael	Aglíori	13705 North Glebe Road	Columbus
111	Dylan	Ricci	14700 Prosperity Avenue	Syracuse
112	Shawn	McDonough	15175 S Main Street	Brooklyn Park

Using templates

You use project and form templates, along with information from PDM objects and extended attributes, to generate Delphi project and form files. Field templates define the controls that you can generate in data forms.

What project and form templates define

Project (DPT)

The project template has a DPT extension. Using information from the PDM, it generates a DPR file.

The project template can contain a list of files to copy during generation of a Delphi project. If the project template lists a file, DelphiGen copies it to projects generated with the template.

Main form (DMT)

The main form template has a DMT extension. It generates a DFR file that can contain menu items, database connection information, and a status bar. You can modify the menu from DelphiGen to add menu items that open data forms.

The MDIMAIN.DMT template that ships with AppModeler and its associated PAS and TXT files add an MDI main form window and a standard Delphi toolbar to your project.

Data form (DFT)

Data form templates have DFT extensions. They can provide offset values for controls added to generated DFM files. They have associated PAS and TXTfiles that add a Delphi navigation control to your project.

The navigation control defined by the data form templates that ship with AppModeler also contain two nonstandard buttons described below:

Button	Description
	Resets or refreshes data
	Applies an editable filter to data

If you click these buttons after starting your project from the Delphi developer environment, Delphi may display a message telling you that the process has stopped. If you get this message, you can select Run➤Run or Run➤Step Over from the Delphi menu bar to redisplay your project form.

You can avoid this message by clearing the Break on Exception checkbox at design time in the Delphi Environment Options dialog box (Tools➤Options from the Delphi menu bar). The message does not appear if you start your project from the EXE file that Delphi creates when you build your project.

What field templates define

Field templates define the controls that you generate in Delphi data forms that use the free form presentation style. Field templates usually define one bound control and a label control for identification of the data that the bound control displays. You attach field templates to columns or domains.

If you use the free form presentation style, DelphiGen generates column labels as tool tips for the Delphi control that displays column data. DelphiGen also generates longer hints for the status bar if you add the pipe character (|) to the column label. The information you type after the pipe becomes the status bar hint for the generated column.

Recommended field styles

Certain field styles are appropriate to columns of a specific data type, and vice versa. Recommended correspondences are:

Data type*	Recommended field style	Template
Boolean	Checkbox	CHECKBOX.DBT
Counter	Label box	LABELBOX.DBT
Integer	Spin control textbox	SPINNUM.DBT
OLE	Image	IMAGE.DBT
Text (limited)	Textbox	TEXTBOX.DBT
Text (unlimited)	Multiline	MLTXTBOX.DBT

* The data type name depends on the DBMS that you use.

Using the dropdown combination box template

The dropdown list field template DBCOMBO.DBT substitutes displayed information from one column with information from another column in a different table. The tables must be linked by a common key column.

The values of the replacement column are listed in the generated dropdown combination box control. You can assign the column code of the replacement column to the LookupExpression attribute or you can use an expression referring to several columns.

Using an expression

If you assign an expression to the LookupExpression attribute, it must correspond to SQL syntax supported by your DBMS. For example, if you use SQL Anywhere, the column codes must be separated by || concatenation marks. If you use Access as your DBMS, you must use the + sign as the concatenation operator.

For example, to list the color, name, and size columns in a combination box, you could use the following expression as a value for LookupExpression:

```
color || ' ' || name || ' (' || prod_size || ')'
```

The result of this expression is shown below in a combination box field with zoom button that was generated by DelphiGen:

Using a dropdown in a grid control

If you use a free form presentation style, you must assign a value to the Width attribute for display of the lookup expression. If you want to have a dropdown combination box in a grid presentation style, you must assign values to the LookupLength and LookupTFieldType attributes.

The DBCOMBO.DBT template includes default values for the Width and LookupLength attributes. In general it is recommended to keep values for both of these attributes regardless of the form presentation style.

The LookupTFieldType attribute does not have a default value. If you assign it a value you must also assign a TField objects template to the parent form.

Assigning a column label

You can assign the label for your replacement column by typing a value for the LookupLabel attribute. The column name of the replaced column is the DelphiGen default label.

DelphiGen instantiates the names of the common key and the source table from default variables that the template defines for the LookupKeyColumn and LookupSource attributes.

Zoom button

The LookupZoomForm attribute defines an optional zoom button positioned to the right of the dropdown listbox control. The default value for this attribute is the name of the data form generated from the table containing the replacement column. To open this form using the zoom button, the form must be listed in the project that you generate.

If you assign a null value to the LookupZoomForm attribute, the zoom button is not generated.

❖ **To use the dropdown combination listbox template:**

1 Select Client➤Delphi Column Attributes.

2 Select a table from the Table dropdown listbox.

3 Select a column from the Column dropdown listbox.

4 Select DropDown from the Field Style dropdown listbox.

5 Select DB Dropdown List from the Field Template dropdown listbox.

6 Double-click the LookupExpression attribute.

7 Type the name of the column you want to use to replace information from the current column.

8 Double-click the LookupTFieldType attribute.

The Value box becomes a dropdown listbox.

9 (Recommended) Select a constant from the Value dropdown listbox.

If you select a constant, you must also select a TField objects template for the table containing the column.

10 Click OK.

Using radio button templates

DelphiGen ships with the RADIOGRP.DBT field template that defines radio buttons enclosed in a frame.

You define the radio button labels in the List of Values in the Check Parameters dialog box for a column or a domain.You define the maximum number of radio button columns by assigning a value to the RadioColumns attribute. The template default value is 3.

❖ To use a radio button template:

1 Select Dictionary➢List of Columns.

2 Select the column for which you want to generate radio buttons.

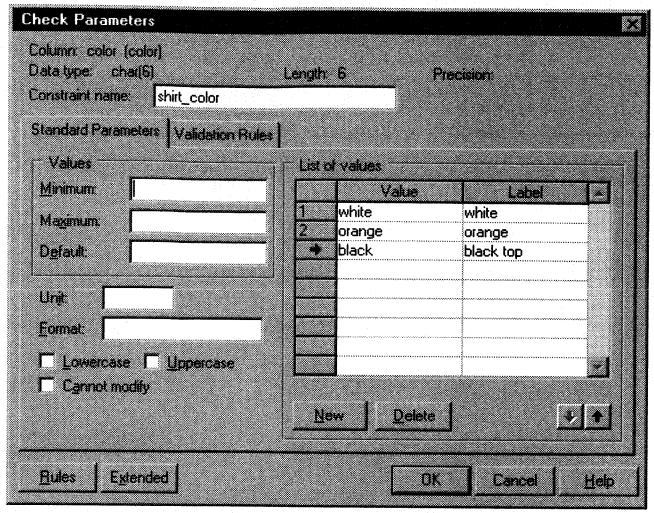
3 Click the Check button.

The Check Parameters dialog box appears.

4 Type the database values for this column in the Value column of the List of Values.

5 Type the labels you want to display in the Label column.

These become the radio button labels in the forms that you generate with radio button templates.



6 Click the Extended button.

7 Select RadioButtons from the Field Style dropdown listbox.

8 Select a radio button template from the Field Template dropdown listbox.

9 Click OK in each dialog box.

Using the standard dropdown template

DelphiGen uses the List of Values in the Check Parameters dialog box to instantiate controls it generates using the CHKLIST.DBT and ECHKLIST.DBT templates. You must type values and labels in this list to use these templates.

The values you type in the Value column become list entries in the dropdown listbox that you generate with the CHKLIST.DBT template. The labels you type in the Label column become list entries in the dropdown listbox that you generate with the ECHKLIST.DBT template.

Using the spin button template

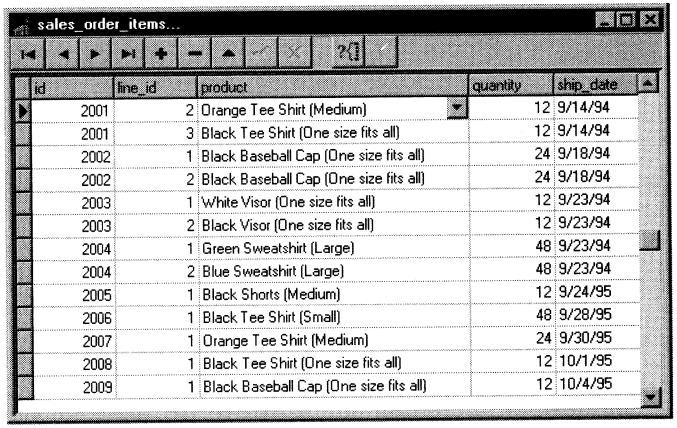
The SPINNUM.DBT spin button template uses default variables for the attributes HiRange and LowRange. DelphiGen instantiates these variables with the Maximum and Minimum values that you define in the Check Parameters dialog box for a column.

You must type values for these check parameters to use the SPINNUM.DBT template. Otherwise, you can replace the default variables with values in the Delphi Column Extended Attributes dialog box.

The SPINNUM.DBT template also uses the value you type in the Default Values box of the Check Parameters dialog box.

Using grid templates

Setup installs GRID.DGTas the default grid template. An example of a standard grid generated in a Delphi data form is shown below. One of the grid columns (product) uses a dropdown combination box control.



id	line_id	product	quantity	ship_date
2001	2	Orange Tee Shirt (Medium)	12	9/14/94
2001	3	Black Tee Shirt (One size fits all)	12	9/14/94
2002	1	Black Baseball Cap (One size fits all)	24	9/18/94
2002	2	Black Baseball Cap (One size fits all)	24	9/18/94
2003	1	White Visor (One size fits all)	12	9/23/94
2003	2	Black Visor (One size fits all)	12	9/23/94
2004	1	Green Sweatshirt (Large)	48	9/23/94
2004	2	Blue Sweatshirt (Large)	48	9/23/94
2005	1	Black Shorts (Medium)	12	9/24/95
2006	1	Black Tee Shirt (Small)	48	9/28/95
2007	1	Orange Tee Shirt (Medium)	24	9/30/95
2008	1	Black Tee Shirt (One size fits all)	12	10/1/95
2009	1	Black Baseball Cap (One size fits all)	12	10/4/95

Viewing template information

You can view template descriptions and other template information directly from DelphiGen. The type of information available depends on the type of template you select for viewing.

From DelphiGen you can access information for the following template types:

Template	Access in DelphiGen	Viewable sections
Project	Model Extended Attributes	Description Modules
Main form	Model Extended Attributes	Description MenuScript
Data form	Table Extended Attributes View Extended Attributes Reference Extended Attributes	Description Attributes
TField	Table Extended Attributes View Extended Attributes Reference Extended Attributes	Description Controls
Field	Column Extended Attributes Domain Extended Attributes	Description Controls Attributes
Grid	Table Extended Attributes View Extended Attributes Reference Extended Attributes	Description Controls

Modifying template descriptions

You can modify the template descriptions if you open the template file with a text editor.

> **Before you modify template files**
> Make a copy of all template files before you attempt to modify them.

If you change information in a template section that you can view from DelphiGen, the changes you make appear in the Show Template dialog box.

❖ **To view a template description:**

1 Click the  button in a Delphi Extended Attributes dialog box.

2 Select a template from the Template Name dropdown listbox.

3 Select a template section to view from the Show Section dropdown listbox.

The section you select is displayed in a scrollable window.

4 Click OK.

Customizing templates and template sets

You can create your own templates from existing projects and then add template path information directly to the registry from DelphiGen.

From DelphiGen, you can also create a customized template set and add the set name to the Template Sets registry subkey.

> **Using predefined templates**
> If your project uses the DelphiGen templates that ship with AppModeler, you do not need to add customized templates or template sets.

Creating project and form templates

You can create your own project templates and form templates to use with DelphiGen.You should save the template files that you create to a new template set directory.

With the AppModeler for Delphi Add-In, you can preview form templates that you create. You can also use the Add-In to create or modify templates.

For information on using the AppModeler for Delphi Add-In, see the chapter "Delphi Add-In."

❖ To create project and form templates:

1. Create and save a Delphi project.
2. Convert your project and form DFM files to TXT files using the Delphi CONVERT.EXE utility
3. Using a text editor, replace names and identifiers in your project and form files with AppModeler system variables.

   You use variables for names and identifiers that you want DelphiGen to instantiate from the PDM.

   For information on AppModeler system variables, see the appendix "Generation Variables and Attributes."
4. Copy and rename the PROJECT.DPT project template, keeping the DPT extension.
5. Make changes to the new project template as needed.
6. Rename your Delphi TXT and PAS project files using the filename prefix of your DPT project template.

7 Copy and rename the MDIMAIN.DMT main form template, keeping the DMT extension.

8 Make changes to the new main form template as needed.

9 Rename your main form TXT and PAS files with the filename prefix of your DMT template.

10 Copy and rename the SINGLE.DFT file or the MSTRDETL.DFT file, keeping the DFT extension.

11 Make changes to the new form template as needed.

12 Rename your form TXT and PAS files with the filename prefix of your DFT template.

Creating field templates

You create field templates by placing controls in an open Delphi form, saving the form as a text file and isolating the code for the controls in the text file. You must save the isolated code to a new file that you name with a DBT extension. The controls must be included in a @Controls section of your DBT field template.

You must also isolate and save code concerning the controls from the Delphi form PAS file. You place the code from the PAS file in the @AddCode and @InsertCode sections of your DBT field template.

Using variables in field templates

You can replace the names in a field template with system or user-defined variables. DelphiGen can instantiate an attribute that is defined by a variable in the @Attributes section of a field template. If you make reference to this attribute in other sections of the template, the attribute name must be surrounded by percent signs.

%FieldCode% variable

If you create your own field templates, you can name the template controls using the %FieldCode% variable. DelphiGen can instantiate the %FieldCode% variable of a control identifier with an index number that it concatenates to the column code. This ensures that each control is unique, even if the column code is used more than once in the same form.

For example, a label control description from the TEXTBOX.DBT field template uses the %FieldCode% variable:

```
object Label_%FieldCode%: TLabel
  Left = 3
  Top = 3
  Width = %lblWidth%
  Height = %lblHeight%
  AutoSize = False
  Alignment = %?lblAlignment%
  Caption ='%.Q:ColumnName%:'
  FocusControl = DBEdit_%FieldCode%
end
```

The label control name is Label_*%FieldCode%*. DelphiGen replaces %FieldCode% by the column code, and an index number, if necessary, to differentiate two labels for the same column (for example, in master/detail forms where the column names are the same for fields in the master and detail sections).

%ColumnWidth% variable

The %ColumnWidth% variable defines the width of a textbox control based on the length of the text declared for the column multiplied by a constant. The UnitsPerChar attribute in DelphiGen data form templates defines this constant as eleven Delphi units per character.

The TEXTBOX.DBT field template uses the %ColumnWidth% variable as the default value for the textbox control it defines:

```
lblWidth                  110 ' Description label width
lblHeight                  17 ' Description label height
lblAlignment taRightJustify ' Alignment of text in
label box
DataAlignment                 ' Alignmemt of text in
Edit box / Grid (empty => automatic)
Width          %ColumnWidth% ' (computed) width of
textbox
Height                     21 ' Height of textbox
TFieldType              (Auto) ' The TFieldType
DisplayFormat %ColumnFormat%' TField.DisplayFormat
DisplayWidth %ColumnLength% ' TField.DisplayWidth
FieldSize    %ColumnLength% ' TField.Size
EditMask                      ' TField.EditMask
Precision
```

Using switches in variable names

Undefined variables

If you type a **?** after the first percent sign in a variable name in a field template, DelphiGen only generates the line containing the variable if the variable is not null. The variable can be a system variable or a user-defined variable.

For example, the template description of a textbox control shown below uses the system variable %?ColumnLength%. If the PDM does not have a value for the length of a column you generate with this template, DelphiGen does not generate the MaxLength line that contains this variable.

```
object DBEdit_%FieldCode%: TDBEdit
  Left = > 6
  Top = 0
  Width = %Width%
  Height = %Height%
  DataSource = %DataSource%
  DataField = '%ColumnCode%'
  MaxLength = %?ColumnLength%
end
```

Formatting variables

You can use the special formatting switch **.Q:** in a variable to protect single quote marks or apostrophes in objects you generate with DelphiGen. As with other formatting switches, you insert it after the first percent sign of a variable name.

At the moment of instantiation, DelphiGen doubles the quote mark in text generated from the line containing the formatting switch. Delphi can then recognize quotation marks in the text as literal characters.

For information on other formatting switches that you can use with AppModeler variables, see the chapter "Database Creation and Modification."

Adding a template or a field style

Adding a template

You can add project, form, TField, field, or grid templates to the registry directly from DelphiGen. You must include a template name and filename for each template that you add.

You must select a template type for each template that you add, and you must type a template name and filename. The new template name is included as a string value in the registry under the template type subkey you select.

Adding a field style

For each field style that you add, you must include a field style name and a default template filename. After you validate the field style name, the new field style appears in the Type dropdown listbox for your current template set. You can then add other field templates for this field style, in addition to the template that you selected as the field style default.

For long lists of field styles, a scroll bar in the Type dropdown listbox permits you to view all available DelphiGen template types.

Your field style name appears as a string value in the registry under the Field Styles subkey. The default template filename appears as a data value for the new field style name.

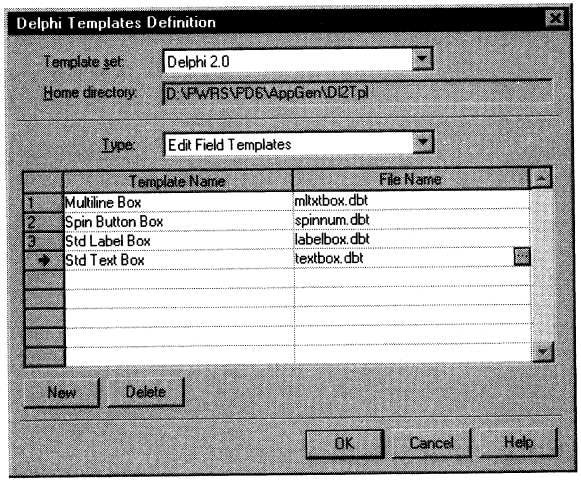
❖ To add a template or a field style:

1 Select Client➤Delphi Model Attributes.

The Delphi Model Extended Attributes dialog box appears.

2 Click the Templates button.

The Delphi Templates Definition dialog box appears.



3 Select the template set to which you want to add a template or field style.

Only template sets listed in your registry are included in the Template Set dropdown listbox.

For information on adding template sets to your registry from DelphiGen, see "Adding a template set" on page 523.

4 Select a template type from the Type dropdown listbox.
or
Select Field Styles from the Type dropdown listbox.

5 Click the New button.

An arrow appears at the beginning of the first blank line.

6 Type a name in the Template Name column.
or
Type a name in the Field Style Name column.

7 Click the File Name column for your new template.
or
Click the Default File Name column for your new field style.

A button appears in the column.

8 Click the  button to browse available drives.
Select a filename and click OK.
or
Type a filename in the column.

You can type the complete path and filename, or you can use the relative path from the template set home directory.

9 Click OK.

Adding a template set

The AppModeler Setup program installs a template set for Delphi 2.0. From DelphiGen, you can create your own template sets as repositories for custom-designed templates.

You can select a home directory for each template set and use relative addresses for your individual template files. When you add templates to a template set, you can always use a full address for the location of a template file that is not in the template set home directory.

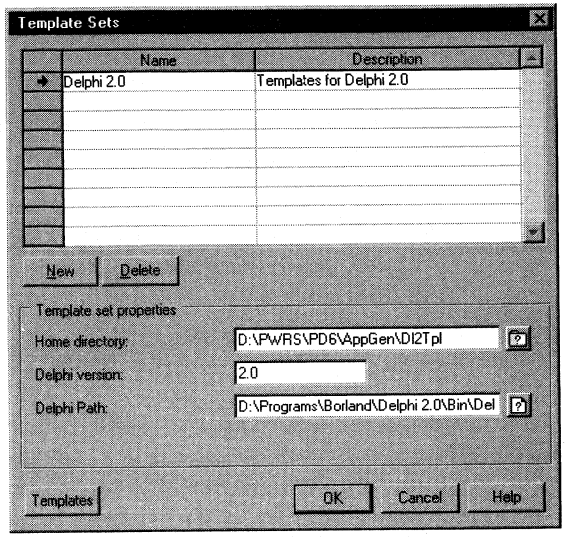
❖ **To add a template set:**

1 Select Client➤Delphi Model Attributes.

The Delphi Model Extended Attributes dialog box appears.

2 Click the  button next to the Template Set box.

The Template Sets dialog box appears.



3 Click the New button.

The arrow moves to a new row in the list of template sets.

4 Type a name for your new template set.

5 (Optional) Type a description for the template set.

6 Click the button next to the Home Directory box.
Select the Home directory for your template set and click OK.
or
Type the directory for your template set in the Home Directory box.

7 Type the Delphi version you want to use with your template set.

8 Click the button next to the Delphi program box.
Select your Delphi program and click OK.
or
Type the path and filename for your Delphi program.

9 Click OK.

Your template set is added as a string value to the Template Sets registry subkey, and as a new key under the AppModeler for Delphi subkey.

CHAPTER 19

Delphi Add-In

About this chapter

This chapter describes how to use the AppModeler for Delphi Add-In to create or modify DelphiGen templates.

Contents

Before you begin

The AppModeler Setup program copies the Add-In to your hard drive. You must register it with the Delphi Component manager before you can use it.

For more information on DelphiGen templates, see the chapter "Delphi Generator." For more information on DelphiGen variables and extended attributes for Delphi, see the appendix "Generation Variables and Attributes."

Using the AppModeler for Delphi Add-In

The AppModeler for Delphi Add-In appears in the Delphi Tools menu after you install it from the Delphi Component manager. You can display the Add-In as a tool bar or as a form containing a tree view.

Including the Add-In in the Delphi menu bar

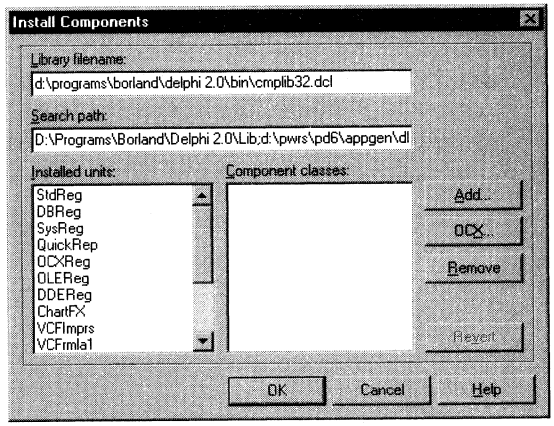
You must add the AppModeler for Delphi Add-In to the Delphi menu before you can use it to create or modify DelphiGen templates.

❖ To include the Add-In in the Delphi menu bar:

1 Open Delphi.

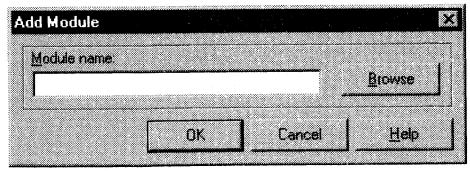
2 Select Component➤Install.

The Install Components dialog box appears.



3 Click the Add button.

The Add Module dialog box appears.



4 Click the Browse button.

A standard file selection dialog box appears.

5 Select Unit File (*.DCU) from the Files of Type box.

6 Select the PDDL2AD.DCU file.

7 Select OK.

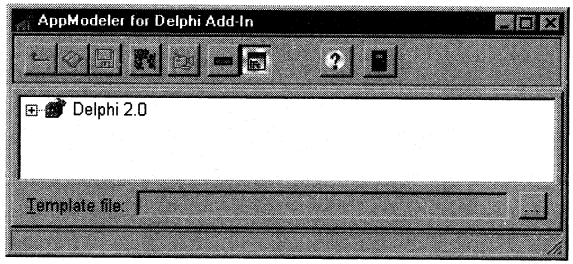
You return to the Install Components dialog box. The new item PDDL2AD appears in the Installed Units list and the Add-In path is added to the Search Path box.

8 Click OK.

After a few seconds the new component is registered and the AppModeler for Delphi menu item is added to the Delphi Tools menu.

9 Select Tools➤AppModeler for Delphi from the Delphi menu bar.

You open the AppModeler for Delphi Add-In.



Previewing DelphiGen templates

The AppModeler for Delphi Add-In can appear as a toolbar or as a form containing a tree view. The Add-In tree view shows the registry items listing DelphiGen templates.

Available template sets

Each DelphiGen template must belong to a template set. AppModeler ships with the Delphi 2.0 template set. From the Add-In, you can also create custom template sets.

For information on creating custom template sets, see "Adding or removing a template set" on page 529.

Templates with system variables

You can use the Add-In to preview form and field templates in Delphi. If the template contains system variables, you can assign values to these variables before you preview them.

For information on using the Add-In to assign values to system variables, see "Assigning a value to a system variable" on page 541.

❖ To preview DelphiGen templates from the Add-In:

1 Select Tools➤AppModeler for Delphi from the Delphi menu bar.

The AppModeler for Delphi Add-In displays as a form listing available template sets. The Setup program installs the Delphi 2.0 template set.

2 Double-click a template set.

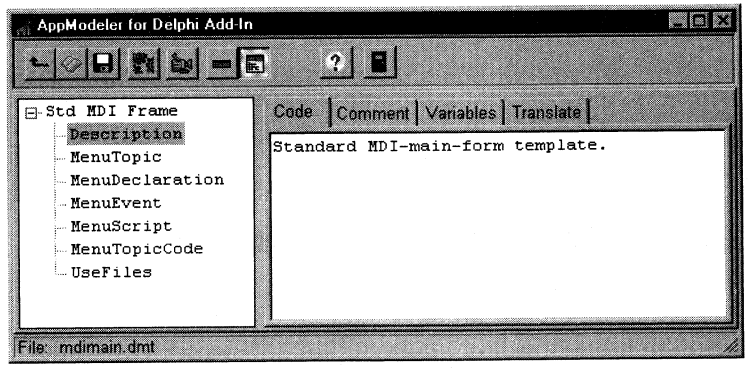
You display the second-level items in a tree view under the template set you selected.

3 Right-click the ⊞ signs under the template set.

You expand the tree view to display all templates in the template set.

4 Double-click a icon for the template you want to display.
or
Select a template and click the button in the Add-In toolbar.

The template definition window appears. The filename of the template you selected are displayed in the status bar.



The template sections pane at the left of the window lists the template name and the sections of the template. The tab control pane at the right shows the contents of a section you select in the template sections pane.

The template sections pane and the tab control pane are separated by a movable splitter bar.

5 Select a template section.

The template sections available in the listbox depend on the template you selected. After you select a template section, the code or text of that section appears in the tab control pane.

6 (Optional) Make changes to the template code.

7 (Optional) Click the  minimize form button.

You fold up the Add-In so it appears as a toolbar.

8 Click the preview  button.

Delphi creates a temporary form showing the template that you selected.

9 Close the form in Delphi.

The focus returns to the Add-In.

10 Click the  button.

If you made any changes to the template, a message box asks if you want to save the changes. You can click Yes to save the changes or click No to cancel the changes. You return to the Add-In tree view.

Using the Add-In to modify the registry

You can use the Add-In to add or remove AppModeler for Delphi templates, template sets, and field styles. The registry reflects the changes you make from the Add-In.

Adding or removing a template set

A template set is a top-level item in the Add-In tree view. AppModeler installs a template set for Delphi 2.0.

If you use the Add-In to create a template set, the Add-In creates a new registry key for the template set. The new registry key contains empty subkeys for project templates, form templates, field templates, TField object templates, computed field templates, and grid templates.

You can assign or change a template set home for each template set.

❖ **To add a template set:**

1 Right-click a  template set item from the Add-In tree view.

The template set context menu appears.

2 Select New Target from the context menu.

The Enter New Template Set Name dialog box appears.

3 Type a name for the template set and click OK.

You return to the Add-In tree view.

4 Right-click the new template set.

The template set context menu appears.

5 Select Properties from the context menu.

The Template Set Properties dialog box appears. The Add-In automatically fills in the Delphi Version and Delphi Path boxes.

6 Make changes to the Delphi Version and Delphi Path boxes, if necessary.

7 Type a home directory for the template set and click OK.

❖ **To remove a template set:**

1 Right-click a template set item from the Add-In tree view.

The template set context menu appears.

2 Select Remove from the context menu.

A confirmation box appears.

3 Click Yes.

The template set and all references contained in the template set are removed.

Adding or removing a field style

The Field Styles registry key lists the types of field templates available for generation of Delphi controls. From the Add-In tree view, you can add and remove field styles from the registry.

❖ **To add a field style:**

1 Double-click the icon to the left of a template set in the Add-In tree view.

You expand the tree view.

2 Right-click the Field Styles item in the tree view.

The Field Styles item is one level below a template set item. You open the Field Styles context menu.

3 Select New Field Style from the context menu.

A new field style item is added below the Field Templates item in the tree view. It is selected.

4 Type a name for your new field style.

Your new field style is entered in the registry.

5 Click the ⊞ sign that appears to the left of your new field style.
or
Double-click your new field style.

The tree view expands to show an empty default template.

6 Select the default template for your new field style.

7 Type a filename of a DBT template in the Template File box.
or
Click the ▪ button to open a file selection dialog box, select a DBT file for the default template, and click OK.

If you type a filename without a path, you save your template in the template set home directory.

❖ **To remove a field style:**

1 If the Add-In tree view is contracted, double-click the 💣 icon to the left of a top-level item in the tree view.

You expand the tree view.

2 Click the ⊞ sign to the left of the Field Styles item.

You display the field styles for your template target set.

3 Right-click the 📄 field style you want to remove.

4 Select Remove Field Style.

A confirmation box appears.

5 Click Yes.

You remove the field style and all its templates from the registry.

Adding, removing, or renaming a template

You can add a project template, form template, grid template, TField object template, or computed fields template from second-level items in the Add-In tree view. Second-level items are items one step below the template set item. You add field templates from third-level tree view items.

Creating a template file

The Add-In creates a new template file if the filename you type in the Template File box does not already exist. If you type a filename without a path, you save your template in the template set home directory.

Copying a template

You can copy an existing template to a new template file using the context menu of a template item in the Add-In tree view. You must assign a template filename to activate the Copy From context menu item. You cannot copy a template to the default template of a field style.

Template extensions for different template types are shown below:

Template type	Extension
Project template	DPT
Main form template	DMT
Table form template View form template Reference form template	DFT
Grid template	DGT
TField objects template	DTT
Field template Computed field template	DBT

❖ **To add a template:**

1. If the Add-In tree view is contracted, double-click a icon to the left of a template set item in the tree view.
2. Right-click a second level item to which you want to add a project, form, grid, TField objects or computed fields template.
   or
   Click the ⊞ sign to the left of the Field Styles item.
   Right-click a field style for which you want to add a field template.

   A context menu appears.
3. Select New Template from the context menu.

   The new template item appears in the tree view under the item that you right-clicked. It is selected.
4. Type a name for the new template.

   The new template name is entered in the registry.

5 Type a filename for the new template in the Template File box.
or
Click the button to open a file selection dialog box, select a template file with the correct extension, and click OK.

6 (Optional) Right-click the new template, select Copy From from the template context menu, select a template to copy, and click OK.

You can use an existing template as a prototype for your new template.

❖ **To remove or rename a template:**

1 If the Add-In tree view is contracted, double-click a template set item.

2 Click the sign to the left of second-level and third-level items.

3 Right-click the icon to the left of the template you want to remove or rename.

The template context menu appears.

4 Select Remove from the context menu to remove the template.
or
Select Rename from the context menu and type a new template name.

Editing template sections

The main purpose of the AppModeler for Delphi Add-In is to help edit templates for use with DelphiGen. You can type changes to a template on the Code page of the Add-In, or you can select controls in a Delphi form and add the controls and their attributes to the template. You can also add InsertCode and InsertColumnCode sections to certain templates.

With the Add-In, you can copy an existing template and use it as a starting point for creating a new template. The Add-In can display text and code in a color of your choice. You can view the list of AppModeler system variables and assign default values to these variables from the Add-In.

Creating controls and attributes in a field template

Field template sections

You can use the Add-In to add any controls that you select in a Delphi form. You add the controls to the Controls section of a DelphiGen field template. You can add control properties to the Attributes section of a DelphiGen field template.

The Add-In does not add code or procedures to the AddCode section or the InsertCode sections of a DelphiGen template. You can copy and paste in the Add-In using the CTRL+INS and SHIFT+INS keys.

From the Add-In you can view and copy declarations for objects and events:

InsertCode subsection	What to add
AddControls	Declarations for objects
AddProcedures	Declarations for object event procedures

For example, the following line is an object declaration:

```
DBEdit1%FieldCode%: TDBEdit;
```

For example, the following line is an object event procedure declaration:

```
procedure DBEdit1%FieldCode%Change(Sender: TObject);
```

Form name variable

You must copy object event procedures to the AddCode section of a DelphiGen template. You should rename the form listed in a procedure using the %FormName% variable. If you are integrating a field template with the template set that ships with AppModeler, you must use Tfrm%FormName% for the form name.

Control names

You can rename the controls using AppModeler system variables. If you use the %FieldCode% variable in a control name, DelphiGen can instantiate the name with an index number that it concatenates to the column code. This ensures that each control is unique, even if the column code is used more than once in the same form.

Control properties

You can select attributes to add to the Attributes section of your template. The attributes page of the List of Attributes dialog box lists properties of the Delphi controls that you selected. Only one attribute value is listed for a property that is shared by more than one control.

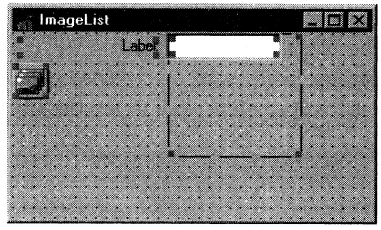
Controls and attributes that you add to an existing template replace any controls and attributes previously listed in the Controls section and Attributes section of the template.

❖ **To create controls and attributes in a field template:**

1 From Delphi, select controls from a form that you want to include in a template.

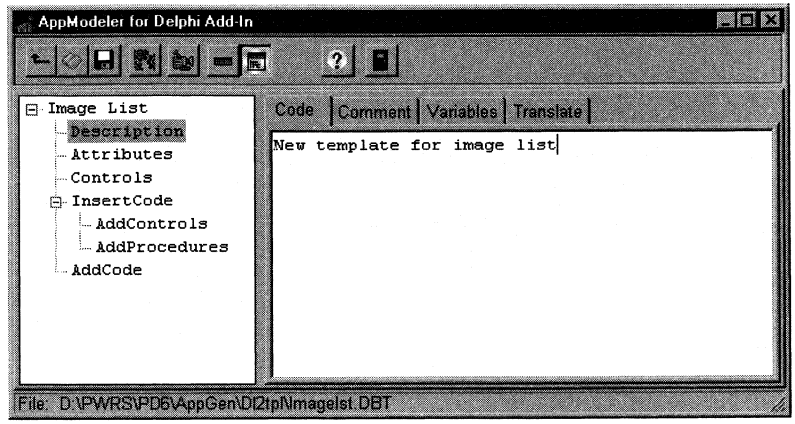
Handles appear around the controls in the form.

For example, the form below has four selected controls: a label box, a text box, an image box, and an image list. The textbox control has the property Visible=False, and is not visible at runtime.



2 From the Add-In tree view, double-click a new field template in which you want to place the selected controls.

The Add-In displays the sections of the template you selected.



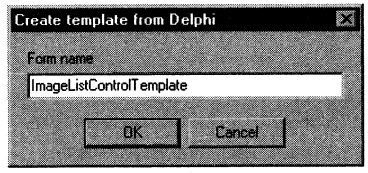
3 Right-click an item in the template sections pane.

A context menu appears.

4 Select the context menu item Create From Delphi➢Other.

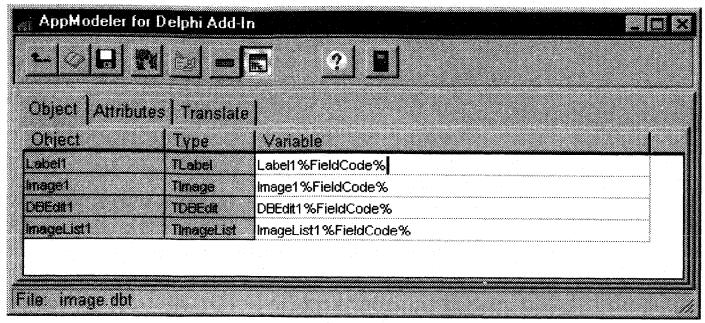
> **The controls you want to create are in a Delphi default form**
> The Create From Delphi context menu has submenu items, Form1, Form2, and Form3. If you are creating controls from these default forms, you can select the form directly in the Create From Delphi context menu and skip the next step of this procedure.

The Create Template from Delphi dialog box appears.



5 Type the name of the Delphi form containing the controls you want to add to your template.

A tab control dialog box opens to the Object page.



6 (Optional) Type a name for each control in the Variable column.

The Add-In renames controls using the %FieldCode% variable.

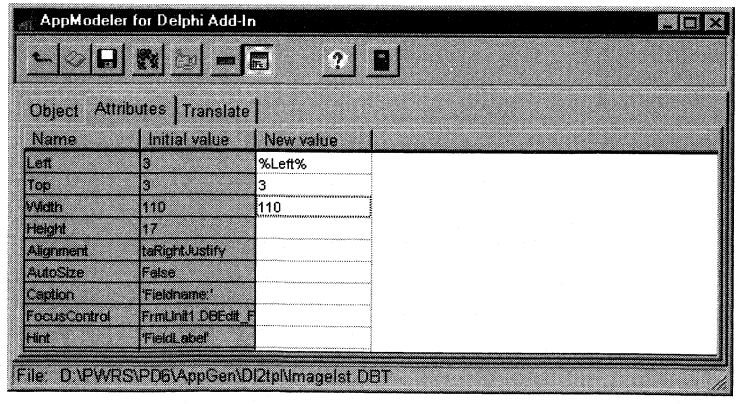
7 Click the Attributes tab.

The Attributes page opens. The attributes listen on the Attributes page include properties of the Delphi controls that you selected. If a property exists for more than one control, only one value is listed.

8 Type a new value in the New Value column for each attribute that you want to add to your template.

The value in the New Value column is the value you add to the field template Attributes section. You can use an AppModeler system variable for an attribute value.

Using the attribute value from Delphi
You can double-click an attribute in the list of attributes to enter its initial value in the New Value column.



9 Click the Translate tab.

You view the code that will be added to the Controls section of your field template.

10 Click the Pascal button.

You view the code as it should appear in the PAS file you create with your template. You must copy and paste this code to the AddCode and InsertCode sections of your template.

11 Click the  button.

You return to the template sections window of the Add-In. All the selected controls are added to the Controls section of your field template, but they are not yet saved to disk.

12 Click the  button.

A message box asks if you want to save the changes to your template.

13 Click Yes.

You return to the Add-In tree view.

Adding InsertCode and InsertColumnCode template sections

InsertCode

You can add an InsertCode section to all field templates, TField objects templates, computed field templates, and grid templates. If an InsertCode section already exists, you can create additional InsertCode subsections.

An InsertCode subsection specifies code in a field template that DelphiGen adds to the form template. The code is added to an InsertCode subsection having the same name as the InsertCode subsection in the field template, just before generation of a form.

InsertColumnCode

For grid templates, you can also add an InsertColumnCode section and subsections. DelphiGen adds code specified in an InsertColumnCode section to the form template, as many times as there are columns in the grid attached to the template. The code is added to the form InsertCode subsection that has the same name as the InsertColumnCode subsection of the grid template.

❖ **To add an InsertCode template section:**

1 Double-click a field template, TField objects template, computed field template, or grid template from the Add-In tree view.

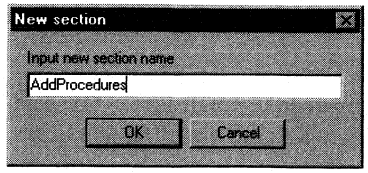
The template definition window appears.

2 Right-click an item in the template sections pane.

A context menu appears.

3 Select Create InsertCode from the context menu.

If an InsertCode section did not already exist in the template, an InsertCode section and subsection is created. A dialog box asks you for a name for the InsertCode subsection you are creating.



4 Type a name for the new InsertCode subsection.

5 Add code to the Code page for the new InsertCode subsection.

6 (Optional) Add a comment to the Comment page for the InsertCode subsection.

7 Click the button.

A message box asks you for confirmation of the changes.

8 Click Yes.

You return to the tree view. Your changes are saved.

❖ To add an InsertColumnCode template section:

1 Double-click a grid template from Add-In tree view.

The template definition window appears.

2 Right-click an item in the template sections pane.

A context menu appears.

3 Select Create InsertColumnCode from the context menu.

If an InsertColumnCode section did not already exist in the template, an InsertColumnCode section and subsection are added. A dialog box asks you for a name for the InsertColumnCode subsection you are creating.

4 Type a name for the new InsertColumnCode subsection.

5 Add code to the Code page for the new InsertColumnCode subsection.

6 (Optional) Add a comment to the Comment page for the InsertColumnCode subsection.

7 Click the  button.

A message box asks you for confirmation of the changes.

8 Click Yes.

You return to the tree view. Your changes are saved.

Selecting a color and font for template code and text

The AppModeler for Delphi Add-In can help you to verify the syntax of changes you make to template code. The Add-In can display variables, keywords, and normal text in different colors. You can also select colors for text on the comment and translation pages of the Add-In.

The default font for all Add-In text is Courrier New 10-point. The default colors for Add-In text parts are listed in the table below:

Syntax element	Default color (style)
Normal text	Black
Keyword	Black (bold)
Variables	Blue
System variables	Blue (bold)
Unknown variables	Red
Comment page	Green
Translation page	Teal

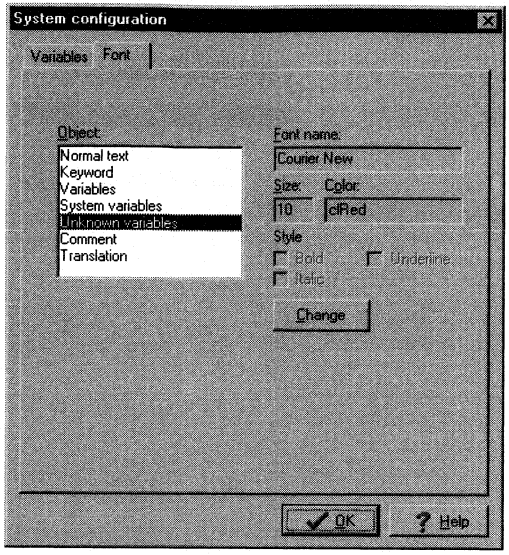
❖ **To select a color and font for template code and text:**

1 Click the  button from the Add-In tool bar.

The System Configuration dialog box appears.

2 Click the Font tab.

The Font page opens.



3 Select a text type or syntax element from the Object listbox.

4 Click the Change button.

A standard font format dialog box appears.

5 Select a color from the Color dropdown listbox.

6 Select a font, style, and size.

7 Click OK.

You return to the font format dialog box. The dialog box displays the color, font, style, and size selections you made.

8 Click OK.

Assigning a value to a system variable

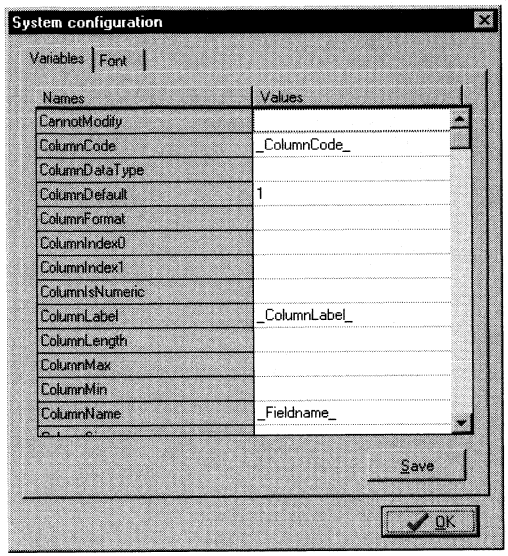
You can assign a value to an AppModeler system variable from the AppModeler for Delphi Add-In. The values you assign are only for use in previewing your templates from the Add-In. They are not transferred to DelphiGen.

For information on previewing your template from the Add-In, see "Previewing DelphiGen templates" on page 527.

❖ **To assign a value to a system variable:**

1 Click the button from the Add-In tool bar.

The System Configuration dialog box appears.



2 Type a value for the system variable in the Value column.

3 Click the Save button.

4 Click OK.

You return to the Add-In template description window.

Assigning user-defined variables to a template

If you substitute your own variables for fixed values in the Controls section of a template, you must also list these variables in the Attributes section.

Variable syntax

Variables in the Controls section are enclosed by percent signs (%). In the attributes section, you do not use the percent signs. However, if you assign a system variable as a value for a user-defined variable in the Attributes section, you must use percent signs around the system variable.

In AppModeler for Delphi, field template attributes appear in the list of attributes of the Delphi Column Extended Attributes dialog box.

❖ **To assign user-defined variables to a template:**

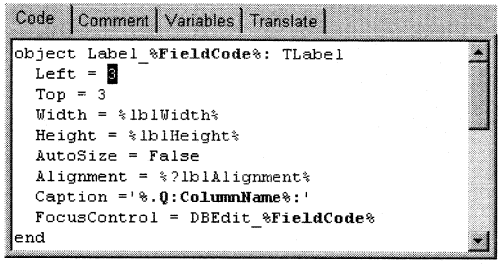
1 From the Add-In tree view, double-click the template to which you want to assign a variable.

The Add-In form displays the sections of the template you selected.

2 Select the Controls section.

3 On the Code page, select the value of the control property to which you want to assign a variable.

For example, in the code for the control Label_%FieldCode% below, select the value 3 for the Left property.



4 Type the variable name surrounded by percent signs.

You should use a variable name that indicates the control property. For example, you can use %lblLeft% to identify the Left property for a label control.

Because you have not yet defined the variable as an atttibute, the variable name receives the color defined for an unknown variable.

5 Select the Attributes section of your template.

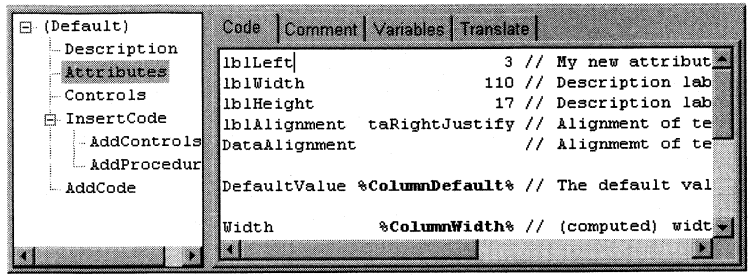
6 Type the variable name on a new line without percent signs.

The variable becomes a template attribute. If there are other attributes in this section, you can arrange the attributes in the order that you want them to appear in the list of attributes of the Delphi Column Extended Attributes dialog box in DelphiGen.

7 (Optional) Type several blank spaces and a default value after the attribute that you added.

If you use a system variable for a default value, you surround the value by percent signs.

8 (Optional) Type a double slash mark followed by a comment.



9 Select the  button.

A message box asks you if you want to save your template changes.

10 Click Yes.

Your changes are saved. You can view the comments from the Show Template dialog box that you access in DelphiGen.

How to use template sections and symbols

The AppModeler for Delphi Add-In displays all the sections of DelphiGen templates. Each section of a template is separated from the preceding section by ## signs. Each section starts with the @ sign.

Other special signs can affect the way AppModeler generates a project.

DelphiGen template sections

The table below lists the different possible sections of a DelphiGen template and gives a brief description.

Template section	Description
Include	Lists complementary tools that must be loaded for the Delphi project to function correctly*
Description	Explains the template function and behavior
Attributes	Lists the user-defined variables that DelphiGen can instantiate during generation of a Delphi project
Controls	Defines the appearance of controls in a form generated by DelphiGen
AddCode	Adds code defined in a field template to a form in which the field template is used. The code is added after any existing code
InsertCode	Adds code at precise locations, usually after a reference in a form template
InsertColumnCode	Repeats the InsertCode sequence for each column of a table, view, or reference
Master	Defines the appearance of a grid control in a master form
Detail	Defines the appearance of a grid control in a detail form
MasterField	Provides the master field qualifier for synchronizing master and detail forms
Screen	Defines offsets for the first fields added to a form
MenuScript	Defines the main form and sheet form menus
Modules	Lists modules to be copied to the generated project

* Uses the keyword Object, followed by an equals sign, and then by the complementary tools required. Multiple tools or OCX objects are separated by semicolons.

Using symbols in template code

Certain symbols have a special meaning in DelphiGen templates.

Symbol	DelphiGen meaning
>	Assigns a relative reference for the placement of a control
@	Defines the beginning of a template section
.Q:	Adds single quotation marks to the text generated, enabling Delphi to interpret an apostrophe in the text as a literal character*
%	Surrounds a system or user-defined variable
?	Indicates that a line of code will not be generated if the instantiated value of a system variable is null*
##	Separates template sections

* This sign must precede the variable name, but remains inside the percent signs that surround the variable. For example, %.Q:ColumnName% or %?ListValues%.

CHAPTER 20

Web Generator

About this chapter

This chapter provides an overview of the AppModeler Web Generator (WebGen) and how it works.

Contents

Before you begin

To use WebGen, you must install NetImpact Dynamo and configure a SQL Central Personal Server or Application Server.

The templates that ship with WebGen are designed to work with Netscape Navigator 3.0 and Microsoft Internet Explorer 3.0. If you use an earlier version of one of these browsers, or a browser from a different company, you may need to make modifications to some of the templates.

Generator basics

WebGen is the AppModeler generator for the Web. It generates Web projects, pages, and controls using predefined templates and information from objects and attributes that you define in a PDM.

You start AppModeler for the Web by double-clicking PDWEB6.EXE or by clicking its icon in the Program Manager or taskbar (Windows 95).

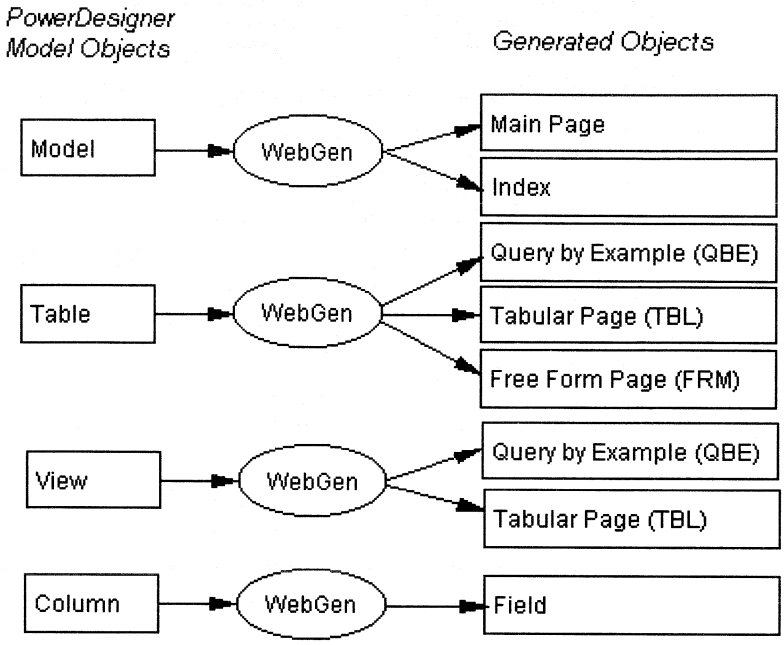
What WebGen does

WebGen uses data from a PDM to instantiate predefined templates and generate Web applications. WebGen can generate HTML pages and save them directly to a Web database that you configure with NetImpact Dynamo and that you can open and modify with a Web browser.

With the 32-bit version of WebGen, you can generate files to disk, instead of a database. WebGen generates STM files for objects you select in the PDM. STM files are NetImpact Dynamo files that contain HTML syntax.

You can import the STM files into a Web site that you configure with NetImpact Dynamo, and open them with a Web browser.

The following schema shows the object transformations performed by WebGen:



WebGen can generate references as hypertext links between the pages it generates from parent and child tables.

Using WebGen with NetImpact Dynamo and SQL Central

The applications you generate with WebGen require a Web site database. You can use the example database that ships with NetImpact Dynamo to store pages of an application. Otherwise, you must create and configure a Web site database for your application.

You can create different databases to store different Web applications that you generate with WebGen, but you must generate all pages of a single application to the same Web site database. WebGen generates an HTTP address for each page it creates in a Web database.

For more information about NetImpact Dynamo, see the *NetImpact Dynamo User's Guide*.

❖ **To use WebGen with NetImpact Dynamo and SQL Central:**

1 Create a SQL Anywhere database for your project pages.

NetImpact Dynamo uses a SQL Anywhere database to store HTML pages. You can use SQL Central to create a SQL Anywhere database.

If your application database is a SQL Anywhere database, you can use the same database to store your HTML pages. Otherwise, you must create a new SQL Anywhere database for your project Web site.

2 Define an ODBC data source for the Web site database.

You can define a data source for the SQL Anywhere Web site database using the ODBC Administrator.

3 Initialize the Web site database.

NetImpact Dynamo uses system tables to store HTML pages. If you generate your application directly to the database, you must create these tables before generating the application with WebGen.

You create the system tables by connecting to the Web site database as type NetImpact Dynamo. You can do this from the Tools➤Connect➤ NetImpact Dynamo menu of SQL Central.

4 Create a connection profile for the ODBC Web site data source.

The NetImpact Dynamo server uses a connection profile to identify a Web site database. From SQL Central, you type a connection name and select the ODBC data source that you defined for your Web site database.

5 Define a URL prefix for the connection profile.

The NetImpact Dynamo server uses a URL prefix to identify a Web site connection profile. You must define a unique URL prefix for each connection profile. The prefix must have the same name as the root folder in the Web site database. By default, the root folder for a NetImpact database is */Site*, but you can change this from the */Site* Properties context menu in SQL Central.

You define a URL prefix from the NetImpact Dynamo Configurator or a NetImpact Dynamo server. You must type a URL prefix and select your connection profile in a dialog box that you access by clicking the Add button in the NetImpact Dynamo Configurator, or by selecting Properties from the context menu of your NetImpact Dynamo server.

6 Define an ODBC data source for your application database.

Use ODBC Administrator to define a data source for your application database if it is different from your Web site database. You can connect to the data source from WebGen and reverse engineer your application database.

7 Generate a Web application with WebGen.

The rest of this chapter describes how to use WebGen to generate a Web application.

8 Start NetImpact Dynamo.

You must start your NetImpact Dynamo server before you can view the Web application you generate.

9 Test the generated application from a browser.

If you generate your application directly to your Web database, you can open it in a Web browser that you start from AppModeler.

Installed files and directories

WebGen includes the following disk files that you install with the Setup program:

Filename	Directory*	Description
PDWEB6.EXE	PD6	Executable file
NI.EXA	PD6\EXA	Default extended attribute import file

* Only short path names are shown in the table. Long names are PowerDesigner 6 (PD6) and Extended Attributes (EXA).

WebGen templates

WebGen ships with several templates and resource files. The Setup program installs them in the PD6\APPGEN\NITPL directory.

References to template files are included in subkeys of the registry path HKEY_CURRENT_USER\Software\Powersoft\PowerDesigner 6 \AppModeler for the Web.

Project and page templates

Project and page templates that ship with WebGen are:

Filename	Registry subkey	Template type
PROJECT.SWT	Main Page Templates	Project
INDEX.SIT	Index Page Templates	Index for project pages
SINGLE.SPT	Table Page Sets	Defines the page types and display order for pages of a table
VIEW.SPT	View Page Sets	Defines the page types and display order for pages of a view
QSINGLE.SQT QVIEW.SQT	QBE Page Templates	Query page for a table Query page for a view
TSINGLE.STT TVIEW.STT	Tabular Page Templates	Tabular page for a table Tabular page for a view
FSINGLE.SFT	Form Page Templates	Free Form page for a table

Associated project and page files

The project and page templates include associated STM files that the Setup program installs in your template directory. WebGen copies these files to your project directory or to a Web database. It instantiates them with information from the PDM, and renames them.

Template filename	Project filename	Description
PROJECT.STM	*ModelName*.STM	Main page
QSINGLE.STM QVIEW.STM	Q*PageName*.STM	QBE page
TSINGLE.STM TVIEW.STM	T*PageName*.STM	Tabular page
FSINGLE.STM	F*PageName*.STM	Free Form page

In addition to the STM files, WebGen copies and renames the following files:

Template filename	Project filename	Description
QSINGLE.QFM QVIEW.QFM	QFM*PageName*.STM	Frame for QBE page
TSINGLE.TFM TVIEW.TFM	TFM*PageName*.STM	Frame for Tabular page
FSINGLE.CFM	CFM*PageName*.STM	Commit page for data update
FSINGLE.FFM	FFM*PageName*.STM	Frame for Free Form page

Other project files that WebGen copies without renaming are:

Filenames	Description
fmSTART.STM	Defines the frame for the Index page
fsSTART.STM	Defines the screen for your main Web page
INDEX.STM	Index page for project

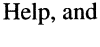
Project help files

WebGen copies help files to your project directory or Web database. You can add information to these files using a text or HTML editor. You open the pages generated from these files by clicking a Help or About button in other pages of your project.

Filenames	Access
hlpABOUT.STM	Main page About button
hlpCOMMT.STM	Commit page Help button
hlpFSTM.STM	Free Form page Help button
hlpMAIN.STM	Main page Help button
hlpQBE.STM	QBE page Help button
hlpTBL.STM	Tabular page Help button

Project toolbars

WebGen uses different toolbars for different pages of your project. The toolbars consist of buttons with descriptive lables.

Filenames	Available buttons
tbFFRM.STM	Insert, Back, Update*, Delete*, Home, Help
tbQBE.STM	Submit Query, New, Home, Help
tbSTART.STM	Help, About
tbTBL.STM	Change Query, New*, Home, Help, and  #

* Only available if you do not generate page with Read-Only property.

Tool buttons for First Record, Previous Record, Next Record, and Last Record.

Project script files

WebGen copies script files containing NetImpact Dynamo code. The script files encode functions that are common to each HTML page type.

Filenames	Page type
FSINGLE.SSC	Free Form
FCSINGLE.SSC	Commit
QSINGLE.SSC QVIEW.SSC	QBE
TSINGLE.SSC TVIEW.SSC	Tabular

Field templates

Field style templates that ship with WebGen include templates for computed fields and various controls, as follows:

Filename	Registry subkey	Template type
COMPUTED.SCT	Computed Field Templates	Computed fields
CHECKBOX.SCT	CheckBox Field Templates	Checkbox
CHKLIST.SCT DBCOMBO.SCT	DropDown Field Templates	Combination box
EMBEDDED.SCT	Advanced Object Templates	Pointer to embedded object
PERCENT.SCT	Advanced Object Templates	ActiveX pie chart
HTTPIMG.SCT	Image Field Templates	Pointer to image file
RADIO.SCT	RadioButton Field Templates	Radio buttons
MLTXTBOX.SCT TEXTBOX.SCT	Edit Field Templates	Text area

Field template prototypes

The AppModeler Setup installs text files that you can use to create your own ActiveX and Java Applet templates. You must insert references to the ActiveX or Java Applet control that you want to use with your generated project, and you must change the TXT extension to an SCT extension.

Filename	Field template prototype
AX_SCT.TXT	ActiveX
AP_SCT.TXT	Java Applet

Ensuring that your project refers to an existing database

A project you develop with WebGen uses information from an application database. You must ensure that the database exists and contains the fields and data types defined in your PDM before your project can work properly.

❖ **To ensure that your project refers to an existing database:**

- Generate a database directly from the PDM.
  or
  Reverse engineer an existing database to create the PDM.

Importing extended attributes

The NI.EXA file contains default attribute values for object generation by WebGen. The NI.EXA attribute values are loaded automatically into new PDMs. You can import these attributes and values into any PDM built with PowerDesigner or S-Designor.

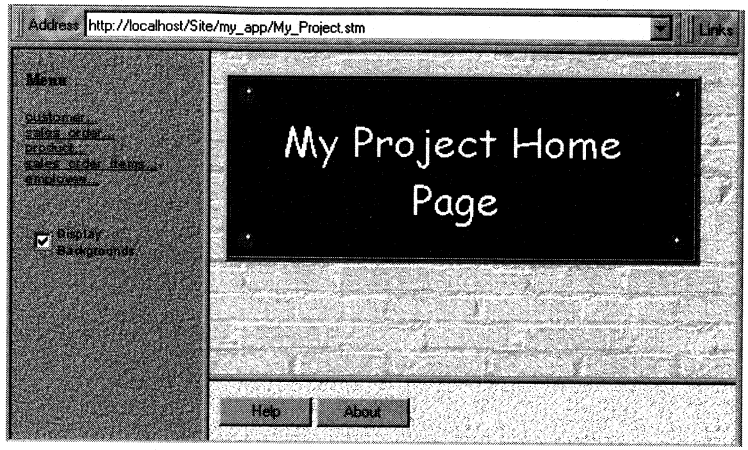
❖ **To import extended attributes:**

1. Select Dictionary➤Extended Attributes➤Import Attributes.
2. Select the NI.EXA file.
3. Click OK.

   A message box tells you if the extended attribute import is successful.

Building a project

WebGen generates a project home page to manage your Web application. The home page contains an index frame with links to pages you generate for tables and views.



Selecting project templates and project properties

WebGen generates project pages by instantiating templates with values from the PDM. You attach main page and index page templates to the PDM and define certain project properties on the General page of the NetImpact Dynamo Model Extended Attributes dialog box:

Property	Description
Template set	NetImpact DynamoTemplates or custom template set
Main page template	Name of template you use to build your main page
Main page name	Filename for the main page
Main page title	Title for the main page
Project directory	Directory in which you generate project files
Index page template	Name of template you use to build the index of pages
Index page name	Filename for the index
Generation mode	Specifies how you generate HTML forms

Generation mode

If you installed the 32-bit version of AppModeler, you can select the database or disk generation mode. If you installed the 16-bit version of AppModeler, you can only generate to a Web site database.

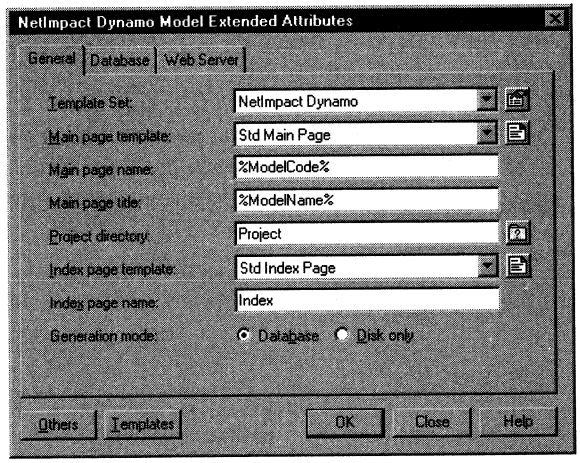
Mode	Description	Comment
Database	Project pages generated as objects in a Web database	The only file generated to your project directory is the *ModelCode*.LOG file
Disk	Project pages generated as files to a disk	You must import the project files in NetImpact Dynamo before you can view them in your Web browser

WebGen can create a new project directory if you type the directory name in the Project Directory box. If you do not type a complete path, WebGen looks for the directory in the PWRS\PD6 path, or in its long name equivalent.

❖ To select project templates and project properties:

1 Select Web➢Web Model Attributes.

The NetImpact Dynamo Model Extended Attributes dialog box opens to the General properties page.



2 Select a target template set from the Template Set dropdown listbox.

Only template sets listed in your registry are available in the dropdown listbox. NetImpact Dynamo Templates is the NI.EXA default template set.

3 Select a template from the Main Page Template dropdown listbox.

4 Select a template from the Index Page Template dropdown listbox.

5 (Optional) Type changes to Main Page Name and Index Page Name.

These are filenames. You do not need to add STM suffixes because WebGen adds them automatically during generation.

6 (Optional) Type changes to Main Page Title.

7 Click the  button to select a project directory and click OK.
or
Type a directory name in the Project Directory dropdown listbox.

8 Select a Generation Mode radio button.

9 Click OK.

Defining an application database connection

To generate a project that displays information from an application database, you must specify connection information by defining a data source. If you use a SQL Anywhere application database, you can generate your Web pages to the same database.

For information on selecting a Web site database, see "Selecting a Web site database" on page 560.

Quotation marks

If the table codes and column codes in your PDM contain special characters (blank spaces, slashes, and so on), WebGen ensures that quotation marks are generated around these names in the SQL queries of your Web project.

Using blank spaces in column codes
If a view contains column codes with blank spaces, queries generated from these columns may not execute properly.

If you select the Quote Table and Column Names checkbox, WebGen generates quotation marks around every table and column name used in the SQL queries of your project.

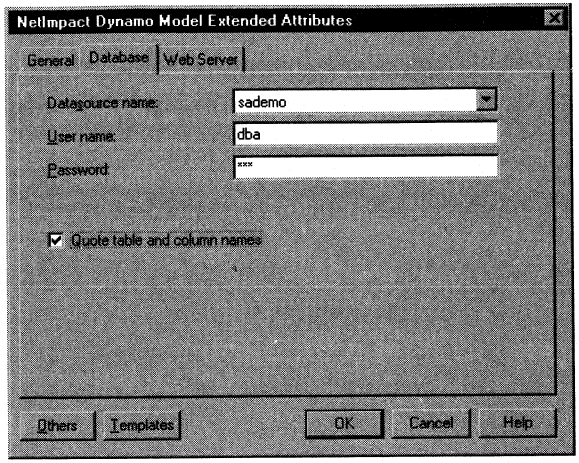
❖ To define an application database connection:

1 Select Web➤Web Model Attributes.

The NetImpact Dynamo Model Extended Attributes dialog box opens to the General page.

2 Click the Database tab.

The Database properties page opens.



3 Select the data source you want your application to access.

The Datasource Name dropdown listbox only includes data sources that you have already defined.

4 Type the name and password needed to access the data source.

5 Select the Quote Table and Column Names checkbox.
or
Clear the Quote Table and Column Names checkbox.

6 Click OK.

Selecting a Web site database

If you generate your project in a Web site database, you can view the project with a Web browser if the NetImpact Dynamo server is running.

Disk generation mode

With the 32-bit version of WebGen, you can generate your project to disk. If you generate a project to disk, you must import the generated files as objects in a NetImpact Dynamo database before you can view the project.

Web database requirements

The Web site database must have a connection profile supported by NetImpact Dynamo. You must also select or create a NetImact Dynamo folder for pages you generate in the database.

For information on importing files to a database and creating NetImpact Dynamo connection profiles, see the *NetImpact Dynamo User's Guide*. For a review of steps you can follow to configure a Web site database, see "Using WebGen with NetImpact Dynamo and SQL Central" on page 549.

Using SQL Central
If you open SQL Central before generating your project to a database, you must use the F5 key to refresh your display of the generated files. You do not need to use SQL Central to view your Web project.

Selecting a database for Web pages

You specify connection information for your Web site database on the Web Server page of the NetImpact Dynamo Model Extended Attributes dialog box. If you generate to disk, WebGen does not use information you enter on the Web Server page.

For more information on selecting or creating a Web site folder and on selecting a data source connection, see "Selecting a Web site folder" on page 562.

❖ To select a database for Web pages:

1 Select Web➤Web Model Attributes.

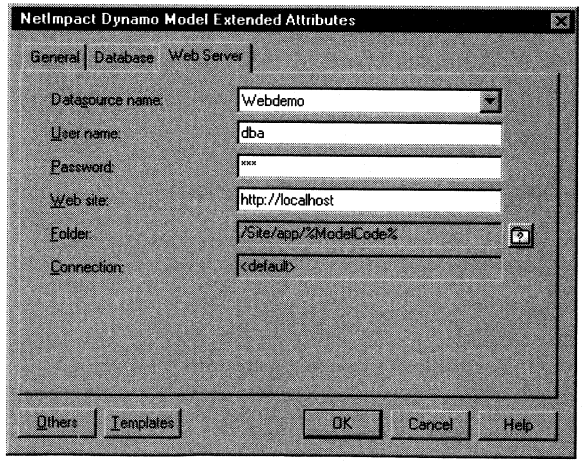
The NetImpact Dynamo Model Extended Attributes dialog box opens to the General page.

2 Select the Database radio button.

You must select this generation mode to generate to a database.

3 Click the Web Server tab.

The Web Server properties page opens.



4 Select the data source you want to contain your Web pages.

The Datasource Name dropdown listbox only includes data sources that you have already defined.

If the data source you selected is not a NetImpact Dynamo database, WebGen displays an error message after you select a folder Folder box.

5 Type the name and password needed to access the data source.

6 (Optional) Type changes in the Web Site box.

The default Web site is HTTP://LOCALHOST.

7 Select a Web site folder and data source connection.

8 Click OK.

Selecting a Web site folder

You can select or create a NetImpact Dynamo Web site folder from WebGen. The NI.EXA default folder is /Site/app/%ModelCode%. If you do not change the default value and the /Site/app folder exists in your Web site database, WebGen will create the *%ModelCode%* folder when it generates your project.

Web connection

The SQL Central connection profile for your Web site database includes a data source connection by default. If your application database is not a NetImpact Dynamo database, you cannot use the connection by default.

From SQL Central, you can add connections to other data sources for the same connection profile. You must define a connection in SQL Central before you can select it in WebGen.

The connection you select in the Web Connection dropdown listbox of the NetImpact Dynamo Web Site Folder dialog box must point to the same data source you select on the Database page of the NetImpact Dynamo Model Extended Attributes dialog box.

❖ **To select a Web site folder:**

1 Select Web➤Web Model Attributes.

The NetImpact Dynamo Model Extended Attributes dialog box opens to the General page.

2 Select the Database radio button.

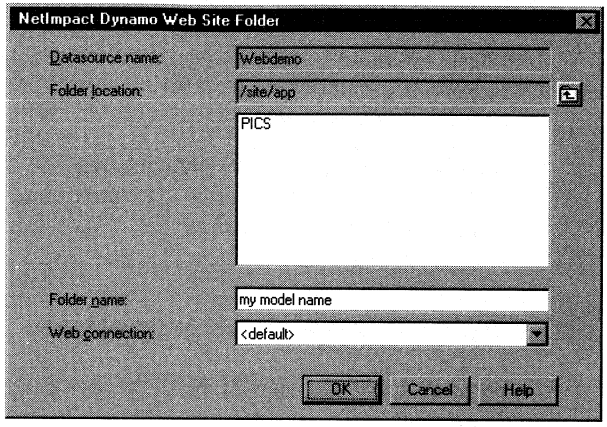
You must select this generation mode to generate to a database.

3 Click the Web Server tab.

The Web Server properties page opens.

4 Click the button next to the Folder box.

The NetImact Dynamo Web Site Folder dialog box appears.



5 Select a subfolder from the Folder Location listbox.
or
Click the button to display a list of subfolders of the parent folder.

6 (Optional) Select a subfolder in the Folder Location listbox.

The subfolder path appears in the Folder Location box.

7 Type a name in the Folder Name box.

8 Select a Web connection name from the Web Connection dropdown listbox.

Only connections defined in SQL Central are included in the dropdown listbox.

9 Click OK.

You return to the Web Server page of the NetImpact Dynamo Model Extended Attributes dialog box.

10 Click OK.

Defining pages and fields

Pages are the primary interface elements of Web projects. Each page is an STM document containing HTML script. WebGen generates STM documents from individual tables and views. With the 32-bit version of WebGen, you can generate the STM documents as disk files.

Fields are page elements that correspond to database columns. In WebGen, field templates are used to define a control or set of controls that can display data from database rows.

Controls are page or field elements that you can use in applications to get user input or to display output. Textboxes and command buttons are examples of standard controls.

Selecting page templates and page properties

For each table and view in the PDM, you can define the following general properties:

Property	Description
Generate page(s)	Select to generate HTML pages in an STM document or file
Page set template	Name of template you use to define the page types to build
Index option checkbox	Select to index the pages you generate
Index option textbox	Name of the index entry with a hypertext link to the page you generate. The name you type is only used if the Index option checkbox is selected

Defining general page properties

For each table or view, you can use default page properties, or you can define specific page properties.

AppModeler ships with separate table page set and view page set templates. These templates reference the following page type templates:

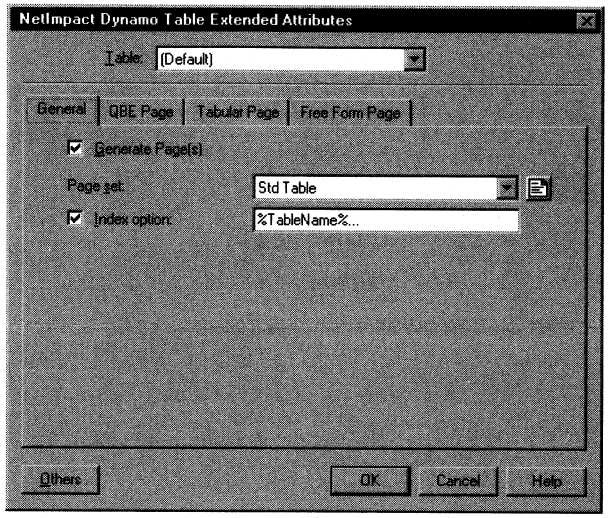
Page set template	Page types required
Table page set	QBE, Tabular, and Free Form
View page set	QBE and Tabular

The page set template also determines the entry point for the HTML pages based on the PDM object. The entry point indicates the type of the page that opens after you select an Index entry for the generated pages.

❖ To define general page properties:

1 Select Web➤Web Table Attributes.
or
Select Web➤Web View Attributes.

An extended attributes dialog box appears.



2 Select a table or view from the Table or View dropdown listbox.

If the dropdown listbox displays Default, any changes to object properties that you make in this dialog box apply to all objects of the same type for which these properties have not been previously modified. This includes all new objects that you add to the PDM.

3 Select a template from the Page Set dropdown listbox.

Only templates listed in the table or view page set registry subkey are available from the dropdown listbox.

4 Make changes to other properties, as needed.

5 Click OK.

Defining a QBE page

The page set templates that ship with AppModeler define the Query by Example (QBE) page as the entry point for retrieving information based on tables and views. The QBE templates generate pages with fields for user-defined database queries.

The QBE page has the following properties:

Property	Description
Generate page	Indicates if QBE page will be generated. The page set template determines if this property is modifiable
Page template	Name of QBE page template
Page name	Name of the file or database object you generate
Page title	Heading that appears at the top of the page

❖ To define a QBE page:

1 Select Web➤Web Table Attributes.
or
Select Web➤Web View Attributes.

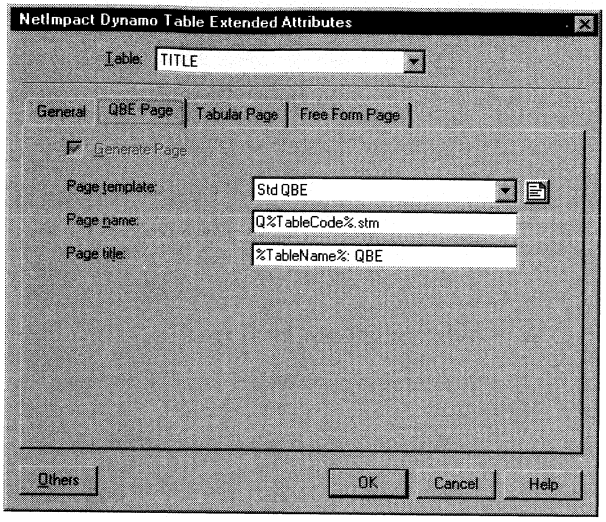
An extended attributes dialog box appears.

2 Select a table or view from the Table or View dropdown listbox.

If the dropdown listbox displays Default, any changes to object properties that you make in this dialog box apply to all objects of the same type for which these properties have not been previously modified. This includes all new objects that you add to the PDM.

3 Click the QBE Page tab.

The QBE Page page opens.



4 Select a template from the Page Template dropdown listbox.

Only templates listed in the QBE Page Templates registry subkey are available from the dropdown listbox.

5 Make changes to other properties, as needed.

6 Click OK.

Defining a Tabular page

A Tabular page lists information in a table. After you submit a query from the QBE page, the server processes the query and returns data on the Tabular page.

The table page set and view page set templates that ship with AppModeler always require a Tabular page. If you use these templates, you cannot clear the Generate Page checkbox.

The Tabular page properties are:

Property	Description
Generate page	Indicates if the Tabular page will be generated. The page set template determines if this property is modifiable
Page template	Name of Tabular page template
Page name	Name of the file or database object you generate
Page title	Heading that appears at the top of the page

❖ To define a Tabular page:

1 Select Web➤Web Table Attributes.
or
Select Web➤Web View Attributes.

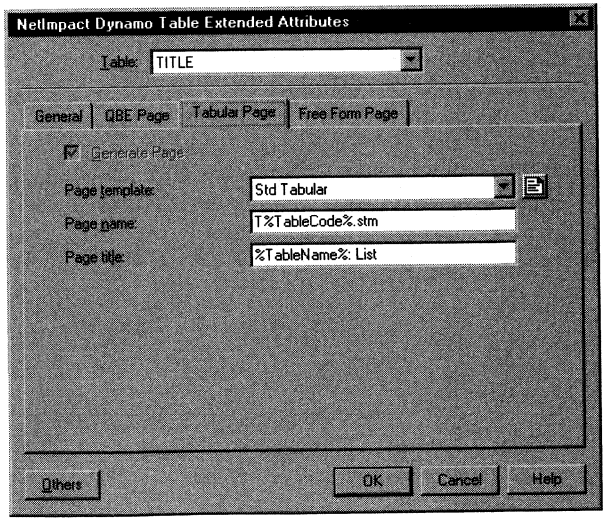
An extended attributes dialog box appears.

2 Select a table or view from the Table or View dropdown listbox.

If the dropdown listbox displays Default, any changes to object properties that you make in this dialog box apply to all objects of the same type for which these properties have not been previously modified. This includes all new objects that you add to the PDM.

3 Click the Tabular Page tab.

The Tabular Page page opens.



4 Select a template from the Page Template dropdown listbox.

Only templates listed in the Tabular Page Templates registry subkey are available from the dropdown listbox.

5 Make changes to other properties, as needed.

6 Click OK.

Defining a Free Form page

You can define a Free Form page for a table. The Free Form page lists information, record by record, from the table data source. You can only update an application database from the Free Form page. You cannot define a Free Form page for a view.

The table page set template that ships with AppModeler always requires a Free Form page. The Free Form page properties are:

Property	Description
Generate page	Indicates if the Free Form page will be generated. The page set template determines if this property is modifiable
Update	Select to allow updates to the application database
Page template	Name of Free Form page template
Page name	Name of the file or database object you generate
Page title	Heading that appears at the top of the page

❖ To define a Free Form page:

1 Select Web➤Web Table Attributes.

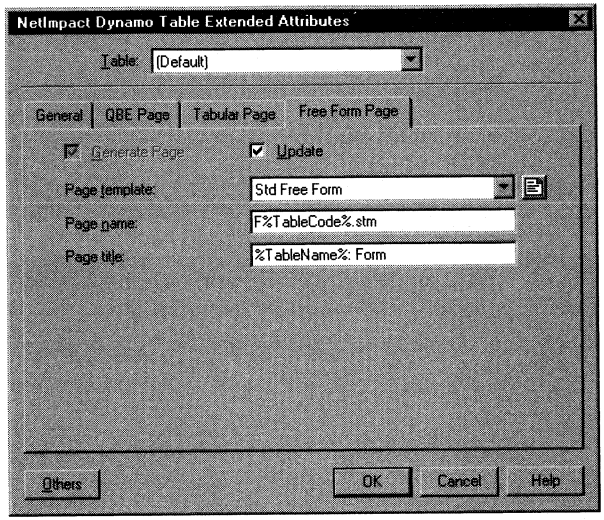
The NetImpact Dynamo Table Extended Attributes dialog box appears.

2 Select a table from the Table dropdown listbox.

If the dropdown listbox displays Default, any changes to table properties that you make in this dialog box apply to all tables for which these properties have not been previously modified. This includes all new tables that you add to the PDM.

3 Click the Free Form Page tab.

The Free Form Page page opens.



4 Select a template from the Page Template dropdown listbox.

Only templates listed in the Form Page Templates registry subkey are available from the dropdown listbox.

5 Make changes to other properties, as needed.

6 Click OK.

Assigning a computed fields template

If you use calculated expressions in SQL statements for a view, you must use a computed fields template to generate a page based on the view.

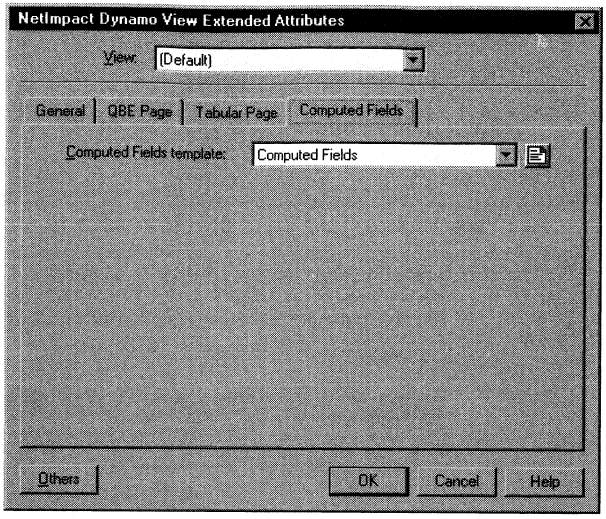
WebGen ships with a computed field template that enables you to display calculated expressions. By default, WebGen assigns this template to all calculated expressions.

❖ To assign a computed fields template:

1 Select Web➤Web View Attributes.

2 Click the Computed tab.

The Computed Fields page opens.



3 Select a template from the Computed Fields Template dropdown listbox.

4 Select a view from the View dropdown listbox.

You can only assign one computed fields template to each view. If the View dropdown listbox displays Default, the computed fields template that you select applies to all views for which you have not previously selected a different computed fields template.

5 Click OK.

Defining field attributes

To use information from a database in a generated application, you can attach field styles and templates to columns or domains.

The default field style is Edit. TEXTBOX.SCT is the default field template for the Edit field style. The default field template for a field style is defined in the Field Styles subkey of the registry.

For information on adding field styles or changing default field templates, see "Adding a template or a field style" on page 599.

Attaching a field style and field template to a column

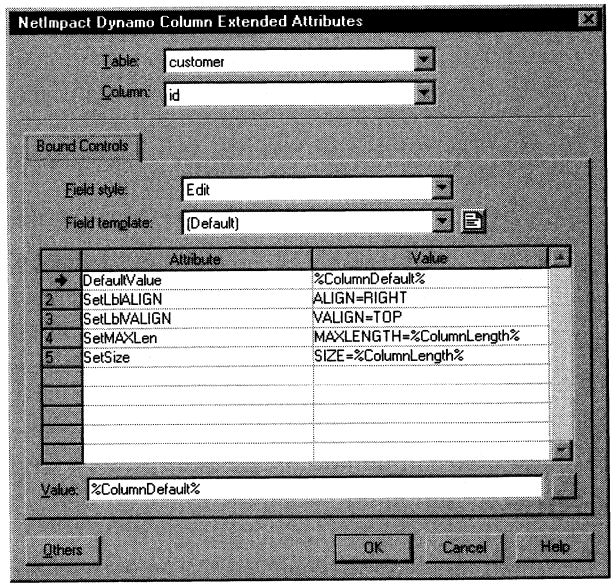
Field styles describe the types of controls you generate in your application. You can use any control that is supported by a Web browser, if you define the control in a field template.

Each field template has a list of attributes. You can modify the values of these attributes from WebGen.

❖ To attach a field style and template to a column:

1 Select Web➤Web Column Attributes.

The NetImpact Dynamo Column Extended Attributes dialog box appears.



2 Select a table from the Table dropdown listbox.

3 Select a column from the Column dropdown listbox.

4 Select a style from the Field Style dropdown listbox.

5 Select a template from the Field Template dropdown listbox.

6 Double-click an item in the Attribute column.

The attribute is selected and the cursor moves to the Value box.

7 Type a value for the field style attribute in the Value box.

8 Double-click another attribute and change its value if needed.

> **Using the keyboard**
> You can use the F6 key to switch between the list of attributes and the Value textbox.

9 Click OK.

Attaching a field style and template to a domain

You can attach field styles and templates to domains. The field styles and templates apply automatically to any columns you attach to the domain. You can also update extended attributes for columns already attached to the domain.

❖ To attach a field style and template to a domain:

1 Select Web➤Web Domain Attributes.

The Web Domain Extended Attributes dialog box appears.

2 Select a domain from the Domain dropdown listbox.

3 Select a field style from the Field Style dropdown listbox.

4 Select a template from the Field Template dropdown listbox.

5 Double-click an item from the Attribute column.

The attribute is selected and the cursor moves to the Value box.

6 Type a value for the field style attribute in the Value box.

7 Select another attribute and change its value if needed.

> **Using the keyboard**
> You can use the F6 key to switch between the list of attributes and the Value textbox.

8 Click OK.

A message box asks if you want to update the columns attached to the domain.

9 Click Yes.
or
Click No.

Selecting objects to generate

You can generate a Web application from objects you select in the PDM.

Tree view items

The objects that you want to generate must be listed in a tree view in the Generate NetImpact Dynamo Web Site window. This window contains an object list in a tree view with the following items:

- Main page object
- Index page object
- Page objects for selected tables and views
  or
  Page objects corresponding to filter definitions that you set in the previous WebGen session
- Columns of each page
- Reference links between parent and child tables

> **Your first WebGen session**
> If you do not select any PDM objects and you did not use a selection filter in a previous WebGen session, the Generate NetImpact Dynamo Web Site tree view lists page objects for every table and view.

Checking items to be generated

Each item in the tree view must be checked to generate it in a project. You can use the Select All button to place checkmarks to the left of all page level items in the tree view, and you can use the Unselect All button to clear these checkmarks.

Generating an index

If you use the PROJECT.SWT template to generate a main page, you must also use the INDEX.SIT index template. WebGen uses the index template to generate hypertext links to other pages in your project.

Generating a project from selected objects

To create a list of objects to generate, you can select objects directly from the PDM or you can reuse objects you selected in the last WebGen session.

If you generate to disk, WebGen asks for confirmation before it generates a file with the same name as an existing file in the project directory.

If you generate to a database, WebGen asks for confirmation before it generates an object that already exists in the database.

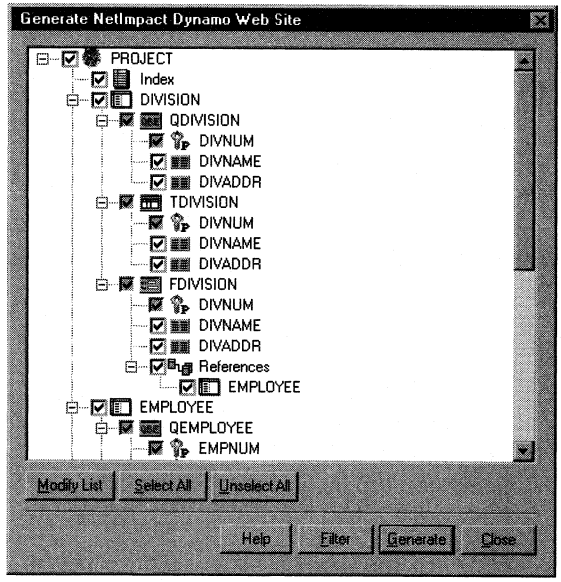
WebGen also asks for confirmation before it copies associated template files that already exist in the project directory or the Web Site database. If you have not made changes to any of these files, you can select No to All to generate the project quickly.

❖ To generate a project from selected objects:

1 Select the objects you want to generate.
or
Verify that no objects are selected in the PDM graph if you want to reuse objects selected during the previous WebGen session.

2 Select Web➤Generate NetImpact Dynamo Web Site.

The Generate NetImpact Dynamo Web Site window appears.



3 Click all plus signs to the left of tree view items.

All tree view items are now visible.

4 Verify that all objects you want to generate are checked.

5 Click the Generate button.

Reviewing the generation progress

You can follow the progress of the generation from the log window that appears after the generation begins. WebGen indicates if there are any errors in the generation or if the generation is successful.

The project log file, *MODELCODE*.LOG, is created automatically in your project directory. It contains model and generation information. You can read the log file with a text editor.

Modifying the generation selection

There are several ways to modify a preliminary object selection from the Generate NetImpact Dynamo Web Site window. To modify the object selection, you can:

- Set the generation flag for objects in the tree view
- Add objects to or remove objects from the tree view
- Apply a filter

Setting the object generation flag

A checkmark to the left of an item in the tree view indicates that its generation flag is selected.

Page items

By clicking the Select All button beneath the tree view, you can select the generation flags for all top level items. By clicking Unselect All, you clear the generation flags of all top level items in the tree view. If you do not generate a top level item, you cannot generate its dependent items, even when their generation checkboxes are selected.

Changes that you make to generation flags in the tree view are reflected automatically in the generation checkboxes of NetImpact Dynamo Extended Attributes dialog boxes.

Column items

You must expand the tree view to list the column items in the tree view. The Select All and Unselect All buttons do not affect the column items. You can only clear the checkboxes of column items one by one.

If the checkbox to the left of an item in the tree view is grayed, you cannot clear its generation flag. The tree view does not display checkboxes to the left of columns in a view. These columns are always generated when you generate their parent page.

❖ **To set the generation flags for objects in the tree view:**

1 Click the plus signs in front of tree view items.

You expand the tree view.

2 Select the checkboxes in front of the objects you want to generate.

3 Clear the checkboxes of objects that you do not want to generate.

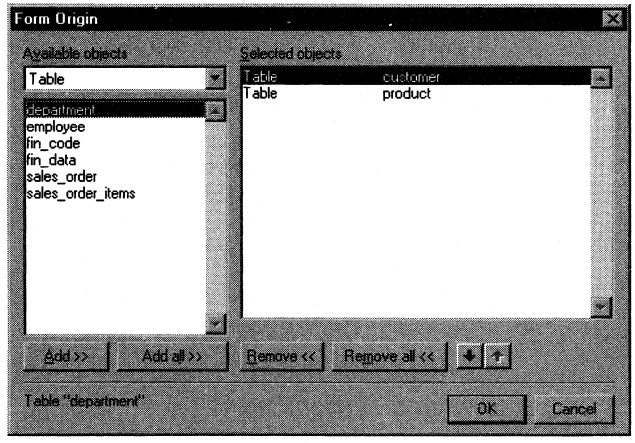
Adding or removing objects from the tree view

If you modify the list of objects in the tree view without using a filter, the modified selections you make are only valid for the current WebGen session. If you close the Generate NetImpact Dynamo Web Site window without generating a Web project, WebGen does not save your modified selections.

❖ **To add or remove objects from the tree view:**

1 Click the Modify List button below the tree view.

The Form Origin dialog box appears.



2 Select a PDM object type from the Available Objects dropdown listbox.

The Available Objects listbox displays all objects of the type selected that are not in the tree view. The Selected Objects listbox displays all objects that are in the tree view.

3 Select objects in the Available Objects listbox and click Add.
or
Select objects in the Selected Objects listbox and click Remove.

Selected objects move from one listbox to the other.

4 Repeat steps 2 and 3 for other PDM object types.

5 Click OK.

You return to the Generate NetImpact Dynamo Web Site window. The objects you added to the Selected Objects listbox appear in the tree view. Objects you removed from this listbox disappear from the tree view.

Applying a filter to modify object selection

Filters help handle large amounts of information. A large PDM can include hundreds of objects. You can use a filter to augment or restrict the items included in the tree view in the Generate NetImpact Dynamo Web Site window.

After you apply a filter, only the matching items appear in the tree view. The selections you make with a filter are saved by WebGen, whether or not you generate your selections during the current session.

Types of filters

There are four types of filters:

Filter	Action
All objects	Selects every object in the PDM
Objects of submodel	Selects objects from a named submodel
Modified objects	Compares objects in the current PDM to objects in an archived PDM and selects objects that have changed
User-defined list	Lets you select a list of objects from the PDM

You can select or clear object type checkboxes with any of the filter types.

User-defined filter

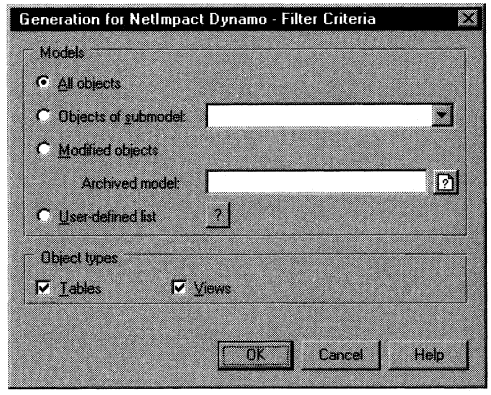
If you use the user-defined filter, you can add any table or view to the tree view. You can use the buttons in the Form Origin dialog box to make customized filter selections:

Button	Function
Add	Transfers a selected object from the Available Objects listbox to the Selected Objects listbox and adds it to the tree view
Add All	Transfers all objects of the selected object type
Remove	Transfers a selected object from the Selected Objects listbox to the Available Objects listbox and removes it from the tree view
Remove All	Transfers all objects of the selected object type
	Moves the selected object up in the list order of Selected Objects and in the tree view
	Moves the selected object down in the list order of Selected Objects and in the tree view

❖ To select objects using user-defined filter criteria:

1 Click the Filter button in the Generate NetImpact Dynamo Web Site window.

The Filter Criteria dialog box appears.



2 Click the ? button beside the User-Defined List option.

The Form Origin dialog box appears.

3 Select a PDM object type from the Available Objects dropdown listbox.

The Available Objects listbox displays all objects of the type selected that are not in the tree view.

The Selected Objects listbox displays all objects that are in the tree view.

4 Use the buttons of the Form Origin dialog box to add, remove, or reposition objects in the listboxes and in the tree view generation list.

5 Click OK.

You return to the Filter Criteria dialog box.

6 Verify that checkboxes are selected for the object types that you want to appear in the tree view.

7 Click OK.

You return to the Generate NetImpact Dynamo Web Site window. The tree view displays the new selections you made with the user-defined filter.

Fine-tuning before and after generation

Before you generate or regenerate your application, you can fine tune and test it from the tree view of the Generate NetImpact Dynamo Web Site window as follows:

- Modify model properties
- Modify extended attributes
- Select columns to generate
- Generate a hypertext link between pages based on parent and child tables
- Open the generated project in your Web browser

Modifying model properties

You can display or modify model properties and extended attributes from the context menus of the project or main page tree view items.

❖ **To modify model properties from the tree view:**

1. Right-click the main page item in the tree view.
   or
   Right-click the index page item in the tree view.

   The item context menu appears.
2. Select Model Properties.

   The Model Properties dialog box appears.
3. Make changes to the model properties as needed.
4. Click the Extended button.

   The NetImpact Dynamo Model Extended Attributes dialog box appears.
5. Make changes to the model extended attributes as needed.
6. Click OK in each of the dialog boxes.

Modifying extended attributes

You can display and modify extended attributes from the context menus of any page item in the tree view. Column extended attributes are also accessible from the tree view.

❖ **To modify extended attributes of a table or view:**

1 Right-click a , , , , or page icon in the tree view.

The page item context menu appears. The context menu selection depends on the object you select.

2 Select a page context menu item.

There is only one item in each page context menu. An extended attributes dialog box appears.

3 Make changes to the extended attributes as needed.

4 Click OK.

❖ **To modify extended attributes of a column:**

1 Right-click a , , or column of a table or view in the tree view.
or
Right-click a view column representing a calculated field.

A context menu appears.

2 Select Column Attributes.

The NetImpact Dynamo Column Extended Attributes dialog box appears.

3 Make changes to the column extended attributes as needed.

4 Click OK.

Selecting columns to generate

From the tree view, you can select which columns you want to generate on a particular page. When you expand page items displayed in the tree view, WebGen lists the columns to be generated. You expand a tree view item by clicking a plus sign to the left of it.

You can clear the generation flag of a column in one of the HTML pages that you generate from a table or view, and still generate the same column on a different page of the same table or view.

For more information on selecting individual columns for generation, see "Setting the object generation flag" on page 576.

Generate a hypertext link between pages

You can generate hypertext links between Web pages based on tables that are linked by a reference. You must generate a parent table and at least one of its child tables in order to generate this type of hypertext link.

❖ **To generate a hypertext link between pages:**

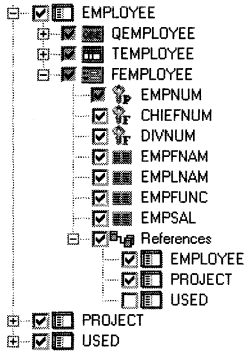
1 Select a parent and one or more child tables in the PDM workspace.

2 Select Web➤Generate NetImpact Dynamo Web Site.

The tree view opens in the Generate NetImpact Dynamo Web Site window.

3 Click the plus sign to the left of the parent table in the tree view.
Click the plus sign to the left of the Free Form page item.
Click the plus sign to the left of the References item.

You expand the tree view and display the reference links between parent and child tables. All child tables that appear in the tree view are also listed under the References item of their parent table.



4 Select the checkbox to the left of each child table which you want to generate.

5 Select the checkbox under References for each child table for which you want to generate a hypertext link.

This is the default selection. If you clear the checkbox, the link will not be generated. Although you can generate the child table without the hypertext link, you cannot generate the hypertext link without generating the child table.

6 Click the Generate button.

WebGen generates your project with the hypertext links you specified.

Opening the project from WebGen

If you generated a project to a Web site database with WebGen, you can test it from the tree view. You can only start the project from WebGen if your NetImpact Dynamo Personal Server or Application Server is already open. If your browser is already running, WebGen opens a new browser window.

For information on configuring a NetImpact Dynamo Personal Server or Application Server, see the *NetImpact Dynamo User's Guide*.

The AppModeler Setup program determines which Web browser is associated with HTML files in your registry. WebGen opens this browser by default for the projects you generate with WebGen templates. If your registry does not indicate a preferred Web browser, you must designate the browser you want to open for each template set you use to generate Web projects.

For information on designating a Web browser to open from WebGen, see "Adding a template set" on page 601.

❖ To open the project from WebGen:

1 Right-click the main page item in the tree view.

The main page context menu appears.

2 Select Run Project.

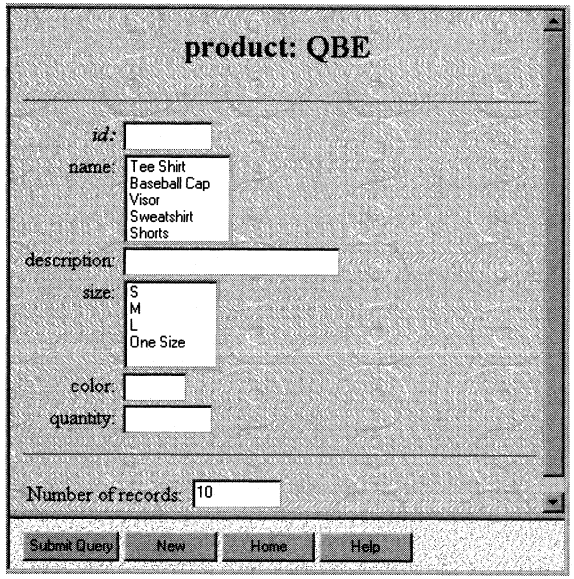
You open the project in your Web browser.

Example pages using WebGen templates

SINGLE.SPT + QSINGLE.SQT

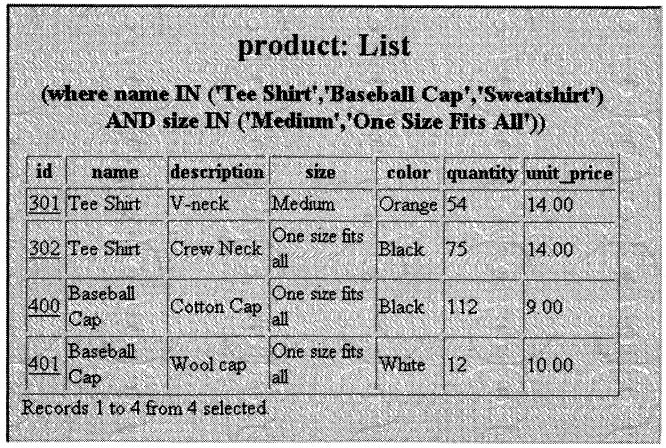
The following example shows a QBE page that was generated using the page set template SINGLE.SPT and the QBE template QSINGLE.SQT.

The Name field uses the field template CHKLIST.SCT, the Description field uses the field template MLTXTBOX.SCT, and the Size field uses the field template RADIO.SCT. Radio buttons only display on the Free Form page. They appear as a regular listbox on the QBE page.



SINGLE.SPT + TSINGLE.STT

The following example shows a Tabular page generated using the page set template SINGLE.SPT and the Tabular page template TSINGLE.STT.



product: List

(where name IN ('Tee Shirt','Baseball Cap','Sweatshirt') AND size IN ('Medium','One Size Fits All'))

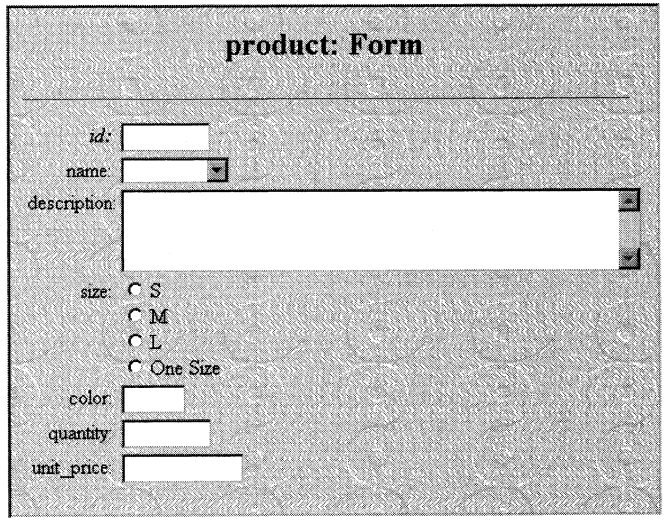
id	name	description	size	color	quantity	unit_price
301	Tee Shirt	V-neck	Medium	Orange	54	14.00
302	Tee Shirt	Crew Neck	One size fits all	Black	75	14.00
400	Baseball Cap	Cotton Cap	One size fits all	Black	112	9.00
401	Baseball Cap	Wool cap	One size fits all	White	12	10.00

Records 1 to 4 from 4 selected.

SINGLE.SPT + FSINGLE.SFT

You display the Free Form page by clicking a hypertext-linked primary key from the Tabular page or by clicking the New button from the QBE or Tabular page. If you cannot update the database from your Web project, QBE and Tabular pages do not include a New button.

The example below was generated using the SINGLE.SPT and FSINGLE.SFT page templates. The field templates used are the same as those described for the QBE page above.



The forms you generate may look slightly different, depending on the browser in which you view them.

Using templates

You use project and page templates, together with information from PDM objects and extended attributes, to generate Web project and page files. Field templates define the controls that you can generate in your HTML pages.

What page templates define

Main page (SWT)

The main page template has an SWT extension. It generates an STM file or database document for the main page.

This template can provide references to resource files that WebGen copies to your project directory or Web database. It uses the variables %ProjectName% and %ProjectTitle% to refer to values that you enter in the Main Page Name and Main Page Title textboxes of the NetImpact Dynamo Model Extended Attributes dialog box.

Index page (SIT)

The index page template has an SIT extension. It generates an index page with hypertext links to the pages you generate from tables and views. The INDEX.SIT template can also provide references to resource files that WebGen copies.

The index page that you generate from the INDEX.SIT template contains a Display Background checkbox that is selected by default. This checkbox refers to the background design and color of your HTML pages. You can clear this checkbox in the generated project if you need to conserve computer resources.

Page set (SPT)

Page set templates have SPT extensions. The page set templates provide the list of page types to be generated for tables and views. They also define the page type of the page that opens when you select a menu item in the index page.

The table page set template SINGLE.SPT must be used in conjunction with the page templates QSINGLE.SQT, TSINGLE.STT, and the FSINGLE.SFT. The view page set template VIEW.SPT is used in conjunction with the QVIEW.SQT and TVIEW.STT page templates.

QBE page (SQT)

A QBE page template is required to generate a Query By Example page for a table or view. You can use the QSINGLE.SQT template to generate the Query page for a table, and the QVIEW.SQT template to generate the Query page for a view.

The QBE help page that WebGen copies to your project explains how to use standard operators in generated fields for multiple-selection queries. You can add supplemental information to the help page by editing the hlpQBE.STM file in a text or HTML editor.

Tabular page (STT)

The Tabular page templates TSINGLE.STT and TVIEW.STT generate pages listing rows of data in a table format. The data match queries submitted from the corresponding QBE page. The primary key in each row of data has a hypertext link. It opens the database record in a Free Form page.

Free Form page (SFT)

The Free Form page template FSINGLE.SFT generates a page that can display complete data from a database record. If you can update your database, you can enter new records from the Free Form page.

What field templates define

Field templates define the controls that you generate in HTML pages. Field templates usually define one bound control and one label control for identification of the data that the bound control displays. You attach field templates to columns or domains.

For a list of field templates that ship with AppModeler, see "Field templates" on page 555.

Recommended field styles

Certain field styles are appropriate to columns of a specific data type, and vice versa. Recommended correspondences are:

Data type*	Recommended field style	Template
Boolean	Checkbox	CHECKBOX.SCT *or* CHKBOX2S.SCT
Counter	Label box	LABELBOX.SCT
Text (limited)	Textbox	TEXTBOX.SCT
Text (unlimited)	Multiline box	MLTXTBOX.SCT

* The data type name depends on the application DBMS that you use.

Using the dropdown combination box template

The dropdown list field template DBCOMBO.SCT substitutes displayed information from one column with information from a column in a different table. The tables must be linked by a common key column. The values of the replacement column are listed in the generated dropdown combination box control.

Using a lookup expression

You can assign an expression to the LookupExpression attribute that includes values from several columns of a table linked by a common key. The expression must correspond to SQL syntax supported by the DBMS of your data source.

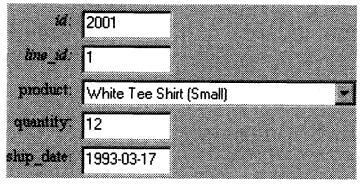
For example, if you use SQL Anywhere, the column codes must be separated by || concatenation marks. If you use Access as your DBMS, you must use the + sign as the concatenation operator.

QBE example

A dropdown listbox control containing information from three columns of the parent table is shown below as it appears on a QBE page.

Free Form example

The same control as it appears in a Free Form page is shown below for an existing database record.



This dropdown listbox was generated for a SQL Anywhere data source using the following expression as a value for the LookupExpression attribute:

```
color || ' ' || name || ' (' || size || ')'
```

This expression concatenates the values for the Color, Name, and Size columns of the parent table.

❖ **To use the dropdown combination listbox template:**

1 Select Web➤Web Column Attributes.

2 Select a table from the Table dropdown listbox.

3 Select a column from the Column dropdown listbox.

4 Select DropDown from the Field Style dropdown listbox.

5 Select DB Dropdown List from the Field Template dropdown listbox.

6 Double-click the LookupExpression attribute.

7 Type the name of the column you want to use to replace information from the current column.

8 Click OK.

Using the radio button template

The RADIO.SCT field template defines a radio button control.

RadioColumn attribute

This template has a RadioColumn attribute that indicates the number of columns of radio buttons. Whatever number you enter for this attribute, if you use the Internet Explorer 3.0 browser, you can only view a single column of radio buttons.

Netscape Navigator 3.0 uses the number you enter to perform a calculation. The actual number of columns Netscape displays also depends on the number of items in the list of values that you assign to radio buttons.

The number of columns attribute is only valid for the Free Form page.

SetListSize attribute

A QBE page generated with the RADIO.SCT template displays a regular listbox control instead of radio buttons. Changing the SetListSize attribute modifies the size of the listbox you generate on the QBE page.

Radio button labels

You define radio button labels in the List of Values in the Check Parameters dialog box for a column or a domain.

❖ **To assign radio button labels:**

1 Select Dictionary➤List of Columns.

2 Select the column for which you want to generate radio buttons.

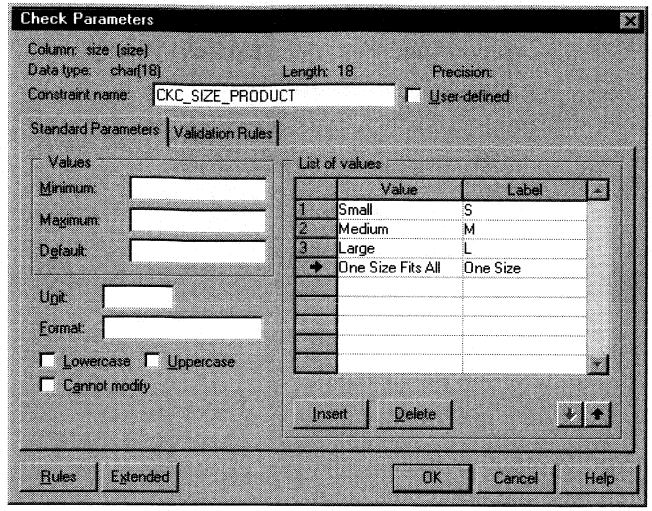
3 Click the Check button.

The Check Parameters dialog box appears.

4 Type the database values for this column in the Values column of the List of Values.

5 Type the labels you want to display in the Labels column.

These become the radio button labels in the forms that you generate with radio button templates.



6 Click the Extended button.

7 Select RadioButtons from the Field Style dropdown listbox.

8 Select a radio button template from the Field Template dropdown listbox.

9 (Optional) Type changes to the template attributes.

10 Click OK in each of the dialog boxes.

Using the standard dropdown template

The CHKLIST.SCT template generates a standard listbox control. To use this template you must type values and labels in the List of Values table of the Check Parameters dialog box. The labels you type become list entries in the listbox you generate.

Using the checkbox template

The CHECKBOX.SCT template generates a standard two-state checkbox on the Free Form page of your project. You can define the values that are saved in your application database by changing the OnValue and OffValue attributes of this template.

QBE page display

The field that you generate with the checkbox template appears as a set of three radio buttons on the QBE page. The radio buttons are useful for entering queries to the database.

The first radio button has the label Any. You can change the labels of the second and third radio buttons by modifying the OnLabel and OffLabel attributes of the template.

An example of a control generated with the CHECKBOX.SCT template on a QBE page is shown below:

Free Form page display

The same control on a Free Form page appears as a standard two-state checkbox:

Using the image template

You can add images to Free Form pages if you use the HTTPIMG.SCT template.

You assign the image template to a column with a text data type. The column data text must indicate an address for the image.

The address in the data text must conform to the syntax indicated below:

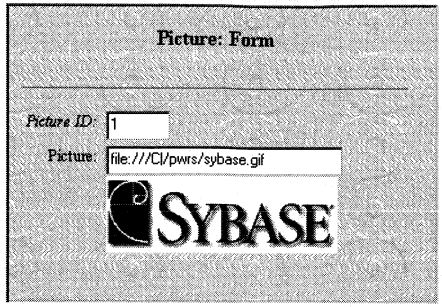
Image Location	Syntax	Example
Internet	http://*server name/path/filename*	http://localhost/pwrs/sybase.gif *or* http://www.powersoft.com/gifs/sybase.gif
Local to project server	*/path/filename*	/Site/Sybase.gif *or* Sybase.gif
Local file	file://local machine/*path/filename** *or* file:///*drivel/path/filename*#	file://chicago/temporary/xyz.gif *or* file:///cl/temporary/xyz.gif

* Availability of the image is limited to users having access to the local network.

This syntax is for use with PC complatible machines only. Availability of the image is limited to users having access to the named drive.

Images that you assign in GIF or JPEG format are supported by both the Internet Explorer and Netscape Navigator browsers. If the filename of an image is nonexistent or not accessible, the browser displays an icon indicating that the image is missing.

An example of an image generated with the HTTLIMG.SCT template is shown below:



Using the embedded object template

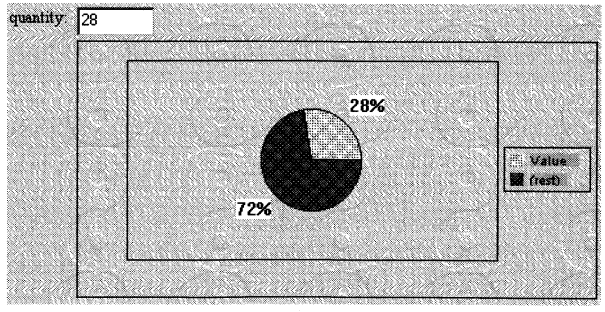
You can use the EMBEDDED.SCT template to add video and sound to your project. You assign this template to a column with a text data type. The column data text must indicate a file address that conforms to the same syntax as indicated for the HTTPIMG.SCT image template.

With the embedded object template, you can enhance the Free Form pages you generate with any object supported by the browsers for your project. The browser that you use may require a plug-in or ActiveX to view or play the embedded object.

Using the ActiveX pie chart template

You can assign the PERCENT.SCT template to a column with a numeric data type and a maximum upper limit. If the upper limit is not 100, you need to change the Maximum attribute.

You must have the Microsoft Chart ActiveX to use this template. You can only view the pie chart in a browser that supports ActiveX controls.



The ActiveX is only generated on Free Form pages. The PERCENT.SCT template also generates a textbox on the Free Form page that displays the current column data value.

If the database is updatable and if you generated your page with updating privileges, you can modify the number in the textbox. You must place the focus elsewhere on the Free Form page before the pie chart is redrawn using the new number that you enter.

Using the ActiveX and Java Applet template prototypes

AppModeler installs the AP_SCT.TXT template prototype for a Java Applet control, and the AX_SCT.TXT template prototype for an ActiveX control.

The description section in these template prototypes describes the minimal modifications you must make to the prototype files to change them into working templates. If you change the filename extension of the prototype files to SCT, you can view the description section in WebGen.

Viewing template information

You can view template descriptions and other template information directly from WebGen. The type of information available depends on the type of template you select for viewing.

From WebGen you can access information for the following template types:

Type	Accessibility in WebGen	Viewable sections
Main Page	Model Extended Attributes	Description Generated Objects
Index	Model Extended Attributes	Description Menu Topic
Table Page Set	Table Extended Attributes	Description
View Page Set	View Extended Attribute	Description
QBE Page	Table Extended Attributes View Extended Attributes	Description Attributes GeneratedObjects
Tabular Page	Table Extended Attributes View Extended Attributes	Description Attributes GeneratedObjects
Free Form Page	Table Extended Attributes	Description Attributes GeneratedObjects
Field	Column Extended Attributes Domain Extended Attributes	Description Attributes
Computed Fields	View Extended Attributes	Description

Modifying template information

You can modify template information using a text editor. If you change information in a template section that is viewable in WebGen, the modifications you make appear in the Show Template dialog box.

> **Before you modify template files**
> Save copies of template files before you modify them.

❖ To view a template description:

1 Click the  show template button in WebGen dialog boxes.

A Show Template dialog box appears.

2 Select a template from the Template Name dropdown listbox.

3 Select a template section to view from the Show Section dropdown listbox.

The section you select is displayed in a scrollable window.

4 Click OK.

Customizing templates and template sets

You can use existing projects to create your own templates and then add the template path infomation directly to the registry from WebGen.

From WebGen, you can also create a customized template set and add the set name to the Template Sets registry subkey.

Creating project templates and page templates

You can create your own project templates and page templates to use with WebGen.

❖ **To create project and page templates:**

1 Create a Web project using NetImpact Dynamo and SQL Central.

2 Export the project as STM files to a template set directory.

3 Replace names and identifiers in the STM files with AppModeler system variables.

You use variables for names and identifiers that you want WebGen to instantiate from the PDM.

For information on system variables, see the appendix "Generation Variables and Attributes".

4 Copy and rename the PROJECT.SWT file, keeping the SWT extension.

5 Change the new SWT file as needed.

For example, you can modify the resources that the template copies to a project directory or database.

6 Rename the project STM file you exported using the filename prefix you give to your the SWT template.

7 Copy and rename the INDEX.SIT file, keeping the SIT extension.

8 Change the new SIT file as needed.

9 Rename the index page STM file you exported using the filename prefix of the index SIT template.

10 Copy and rename the SINGLE.SPT or VIEW.SPT template, keeping the SPT extension.

11 Make changes to the new SPT template as needed.

The SPT template file must include a list of the types of pages it can support. You also can reference frame templates and script templates for WebGen to copy to your project directory or Web site database.

12 For pages based on tables, copy and rename the QSINGLE.SQT, TSINGLE.STT, and FSINGLE.SFT templates using the filename prefix of your SPT template, keeping the SQT, STT, and SFT extensions.

You can insert a letter in front of the filename prefix (Q, T, or F) to designate the type of page it generates.

13 For pages based on views, copy and rename the QVIEW.SQTand TVIEW.STT templates using the filename prefix of your SPT template, keeping the SQT and STT extensions.

14 If you exported an STM file for a QBE page in step 2, rename the QBE page you exported with the filename prefix of the new SQT template.

15 If you exported an STM file for a Tabular page in step 2, rename the Tabular page you exported with the filename prefix of the STT template.

16 If you exported an STM file for a Free Form page in step 2, rename the Free Form page you exported with the filename prefix of the SFT template.

Creating field templates

You can modify WebGen field templates with a text editor. You should replace field names and identifiers with system or user-defined variables. A field template filename must have an SCT extension.

Using variables in field templates

WebGen can instantiate an attribute that is defined by a variable in the @Attributes section of a field template. If you make reference to this attribute in other sections of the template, the attribute must be surrounded by percent signs.

%FieldCode% variable

If you create your own field templates, you can name the template controls using the %FieldCode% variable. WebGen can instantiate the %FieldCode% variable of a control identifier with an index number that it concatenates to the column code. This ensures that each control is unique, even if the column code is used more than once in the same page.

For example, the textbox control description from the Insert Code section of the TEXTBOX.SCT field template uses the %FieldCode% variable:

```
<INPUT TYPE="text" NAME="%FieldCode%"
```

The textbox control name is *%FieldCode%*. WebGen replaces %FieldCode% by the column code, and an index number, if necessary, to differentiate multiple instances of the same generated control.

Undefined variables

If you type a **?** after the first percent sign in a variable name in a field template, WebGen only generates the line of code containing the variable if the variable is not null. The variable can be a system variable or a user-defined variable.

An example from the RADIO.SCT template is shown below:

```
<MULTICOL COLS=%?RadioColumns%>
```

When the value of the RadioColumns attribute is null, WebGen does not add the MULTICOL COLS line to the generated form.

Protecting variables with formatting switches

Protecting variables for JavaScript

You can use the special formatting switch **.Q:** in a variable to protect quotation marks (single or double) that you generate with WebGen. As with other formatting switches, you insert it after the first percent sign of a variable name. An example from the InsertCode section of WebGen field templates is shown below:

```
var Hint_%FieldCode% = "%.Q:ColumnLabel%";
```

At generation, WebGen adds a backslash before the quotation mark in the text generated from the line containing the formatting switch. NetImpact Dynamo can then recognize the quotation mark (single or double) as a literal character.

Protecting variables for HTML

You can use the formatting switch **.H:** in a variable to protect certain special characters. If you use this formatting switch, WebGen replaces these characters with the corresponding HTML codes at generation. Your browser can then recognize them as special characters. The replacement values are shown below:

Character	HTML replacement text
'	&#39
"	&quot
<	&lt
>	&gt
&	&amp

Protecting variables for JavaScript and HTML

You may need to protect quotation marks (single and double) for JavaScript and for HTML. In this case you can combine the formatting switches, as in the following example from the INDEX.SIT template:

```
<A HREF="%IDXPage%?mode='updatable'"
Target="client_frame" OnMouseOver="status =
'%.QH:IDXTitle%'; return true;">
</A>
```

For information on other formatting switches that you can use with variables, see the chapter "Database Creation and Modification."

Adding a template or a field style

Adding a template

After you list a new main page template, index page template, page set template, page template, or field template in the NetImpact Dynamo Templates Definition dialog box, WebGen adds a reference to the template in the registry.

You must select a template type for each template that you add, and you must type a template name and filename. The new template name is included as a string value in the registry under the template type subkey you select.

Adding a field style

For each field style that you add, you must include a field style name and a default template filename. After you validate the field style name, the new field style appears in the Type dropdown listbox for your current template set. You can then add other field templates for this field style, in addition to the template that you selected as the field style default.

For long lists of field styles, a scroll bar in the Type dropdown listbox permits you to view all available WebGen template types.

Your field style name appears as a string value in the registry under the Field Styles subkey. The template default filename appears as a data value for the new field style name.

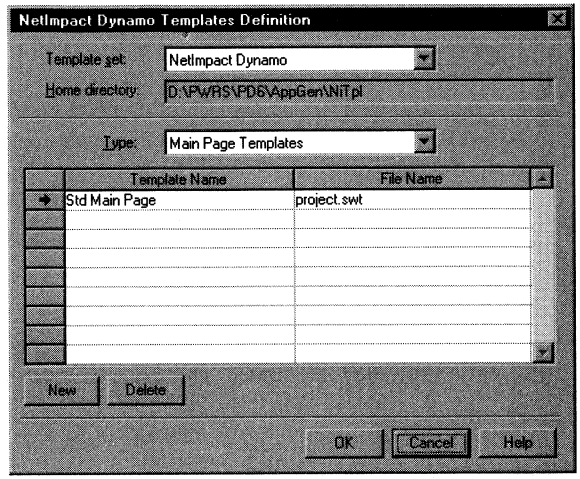
❖ To add a template or a field style:

1 Select Web➤Web Model Attributes.

The NetImpact Dynamo Model Extended Attributes dialog box appears.

2 Click the Templates button.

The Templates Definition dialog box appears.



3 Select the template set to which you want to add a template or field style.

Only template sets currently listed in your registry are included in the Template Set dropdown listbox.

4 Select a template type from the Type dropdown listbox.
or
Select Field Styles from the Type dropdown listbox.

5 Click the New button.

An arrow appears at the beginning of the first blank line.

6 Type a name in the Template Name column.
or
Type a name in the Field Style Name column.

7 Click the File Name column for your new template.
or
Click the Default File Name column for your new field style.

A ... button appears in the column.

8 Click the button, select a filename, and click OK.
or
Type a filename in the column.

You can type the complete path and filename, or you can use the relative path from the template set home directory.

9 Click OK.

Adding a template set

The AppModeler Setup program installs a template set for NetImpact Dynamo. From WebGen, you can add custom template sets to the registry. You can then select a custom template set from the NetImpact Dynamo Model Extended Attributes dialog box.

You can use relative addresses for template files that you add to a template set after you select a home directory for the set. You can still use full addresses for the location of template files that are not in the template set home directory.

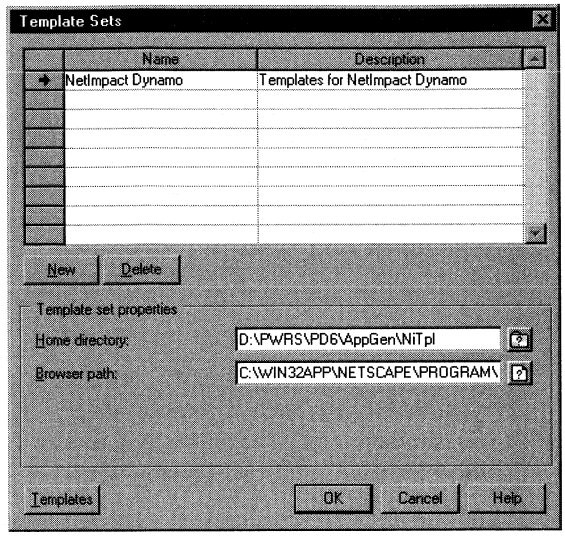
❖ To add a template set:

1 Select Web➤Web Model Attributes.

The NetImpact Dynamo Model Extended Attributes dialog box appears.

2 Click the button next to the Template Set box.

The Template Sets dialog box appears.



3 Click the New button.

The arrow moves to a new row in the list of template sets.

4 Type a name for your new template set.

5 (Optional) Type a description for the template set.

6 Click the button next to the Home Directory box.
Select the Home directory for your template set and click OK.
or
Type the directory for your template set in the Home Directory box.

7 Click the button next to the Browser Path box.
Select your browser path and filename and click OK.
or
Type the path and filename for your preferred Web browser.

8 Click OK.

Your template set is added as a new string value to the Template Sets registry subkey, and as a new key under the AppModeler for the Web subkey.

PART FOUR

Model Presentation

This part describes how to present models on screen and in print.

CHAPTER 21

Model Graphics

About this chapter

This chapter describes how to manipulate the graphical display of models.

Contents

Model display preferences

Model display preferences apply to the entire model as it appears in the workspace.

Selecting general display preferences

You can define the following general display preferences:

Preference	Result when selected
Auto-adjust to text	Sizes symbols to display all the text they contain
Keep center	When you resize symbols, maintains the position of the center of symbols (symmetrical sizing)
Keep aspect ratio	When you resize symbols, protects their proportions
Printer fonts	Displays what-you-see-is-what-you-get (WYSIWYG) fonts
Page scale	Enlarges the relative size of the page to the model graphic (for example, if you select 40%, the graphic takes up only 40% of the space that it would at 100%)
Name	Displays object names in symbols
Code	Displays object codes in symbols

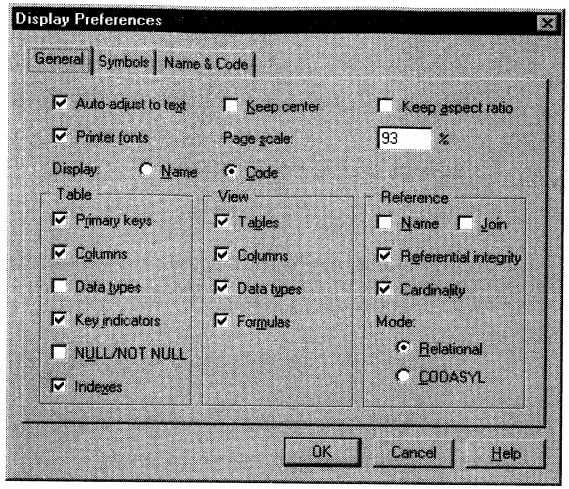
These preference are saved in the current model. The preferences Name and Code apply to the current model only. The other preferences apply to all new models of the same type.

For information on displaying text with object symbols, see chapters on defining models. For example, for information on displaying the primary key in a table symbol, see "Building a Physical Data Model."

❖ To select general display preferences:

1 Select File➤Display Preferences.

The Display Preferences dialog box opens to the General page.



2 Select or clear display preferences.

3 Click OK.

The model graphically reflects your choices.

Selecting symbol display preferences

Symbol display preferences apply to object symbols and their associated text. They do not apply to graphic shapes and lines, or to free text.

These preferences are saved in the current model and apply to all new models of the same type.

You can set the following symbol display preferences:

Preference	Description
Width	Horizontal size of a symbol containing text (measured in units of 1/7200 of an inch)
Height	Vertical size of a symbol containing text (measured in units of 1/7200 of an inch)
Font	Font, size, and style of text associated with a symbol
Foreground color	Color of borders or lines in a symbol
Background color	Color that fills a symbol
Text color	Color of text associated with a symbol

Selecting a size preference

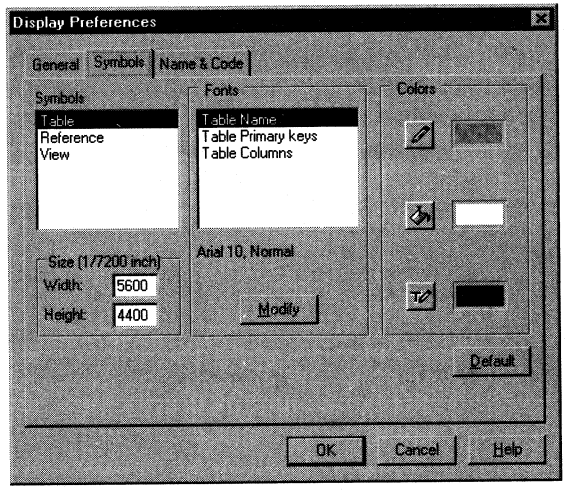
A preference for symbol size applies to all new symbols. If you change this preference, it does not automatically resize existing symbols

❖ To select a size preference:

1 Select File➤Display Preferences.

The Display Preferences dialog box opens to the General page.

2 Click the Symbols tab.



3 Select a symbol from the Symbols list.

4 Type changes to symbol size in the Width and Height boxes.

5 Click OK.

Selecting font preferences

Font preferences apply to each type of text individually. For example, you can user a larger font for primary keys than for other columns.

Font preferences apply to all existing and new symbols.

❖ To select font preferences:

1 Select File➤Display Preferences.

The Display Preferences dialog box opens to the General page.

2 Click the Symbols tab.

3 Select a symbol from the Symbols list.

4 Select a text from the Font list.

5 Click the Modify button.

A standard font selection dialog box appears.

6 Type changes to font preferences.

7 Click OK in each of the dialog boxes.

Selecting color preferences

Text color applies to all text related to a symbol. For example, even if you display primary keys larger than other columns, all columns are always the same color.

You can open a color selection dialog box by clicking the following tools:

Tool	For selection	Description
	Foreground color	Color of lines and borders in the symbol
	Background color	Color that fills the symbol
	Text color	Color of text associated with the symbol

You can choose to apply color preferences to symbols as follows:

Option	Applies color preferences to ...
All symbols	All existing symbols and all new symbols
Selected symbols	Selected symbols and all new symbols
New symbols	New symbols only (does not change the color of existing symbols in the model)

❖ **To select color preferences:**

1 Select File➤Display Preferences.

The Display Preferences dialog box opens to the General page.

2 Click the Symbols tab.

3 Select a symbol from the Symbols list.

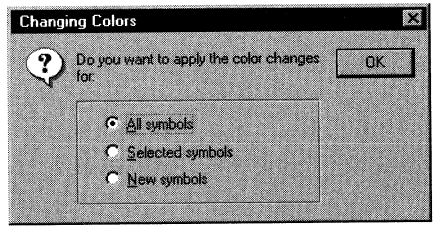
4 Click the foreground color tool, select a color, and click OK.

5 Click the background color tool, select a color, and click OK.

6 Click the text color tool, select a color, and click OK.

7 Click OK.

A confirmation box asks which symbols you want to modify.



8 Select a radio button and click OK.

Applying default symbol display preferences

You can apply default preferences for symbol display. In this case, you replace all symbol display preferences that you previously selected in the current model.

❖ **To apply default symbol display preferences:**

1 Select File➤Display Preferences.

The Display Preferences dialog box opens to the General page.

2 Click the Symbols tab.

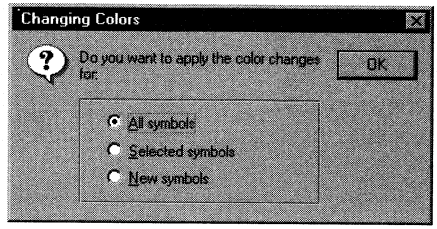
3 Select a symbol from the Symbols list.

4 Click the Default button.

Default preferences display in the dialog box.

5 Click OK.

A confirmation box asks which symbols you want to modify.



6 Select a radio button and click OK.

Selecting a display preference for name and code

You can define the following display preferences for name and code:

Preference	Description	Auto-adjust to text
None	Truncates name or code to the width of the symbol	Symbol fits entire name or code
Truncation	Truncates name or code at the indicated length	Symbol fits truncated name or code up to the indicated length
Word wrapping	Wraps name and code text onto additional lines after wrap character (up to the indicated length)	Symbol fits wrapped name or code up to the indicated length

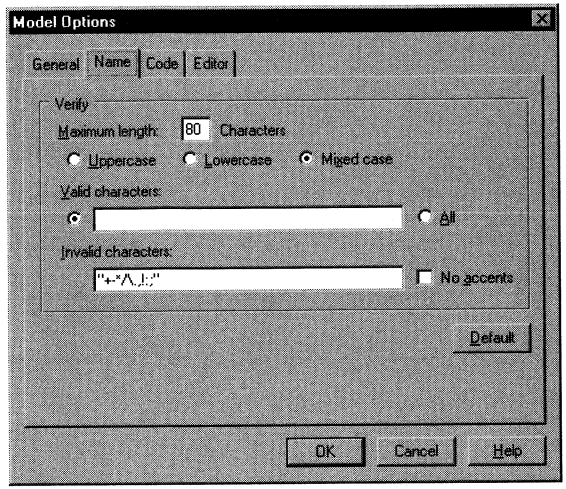
Name and code preference are saved in the current model and apply to the current model and all new models of the same type.

❖ To select display preferences for name and code:

1 Select File➢Display Preferences.

The Display Preferences dialog box opens to the General page.

2 Click the Name & Code tab.



3 Select the None radio button.
or
Select the Truncation radio button and type a length.
or
Select the Word Wrapping radio button, type a length, and type wrap characters.

4 Click OK.

The model graphically reflects your choices.

Modifying symbol appearance

You can modify the display of individual symbols in a model including object symbols, graphic shapes, and lines.

Selecting symbols

Selecting symbols is the first step in many operations that modify the appearance of a model. Before you can change the appearance of symbols in a model, you must select one or more symbols.

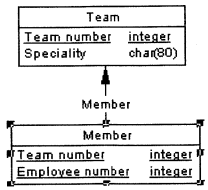
The following selection tools are available:

Tool	Selection	How to use
	One	Click the Pointer tool in the tool palette, then click a symbol in the model
	All	Click the Grabber tool, then click anywhere in the model
	All symbols of one type	Double-click the corresponding tool in the tool palette (for example, to select all tables, double-click the Table tool)
	Symbols in an area	Click the Pointer tool or the Lasso tool, then outline the area containing the symbols
	Symbols in different areas	Click the pointer tool, then click the first selection and press the SHIFT key while you click additional symbols

The following menu choices select objects:

Menu choice	Selection
Edit➤Select Connected Symbols	Symbols connected directly to one or more selected symbols (for example, tables and references connected to a selected table)
Edit➤Select All	All symbols

Selected symbols display handles. For example, the illustration below shows thc selection of the Member table.



Applying color to symbols

You have the following color choices:

Color choice	Applies color to ...
Foreground color	Lines and the outlines of shapes and of symbols containing text
Background color	Interior of shapes and of symbols containing text
Text color	Text associated with symbols and free text

❖ To apply a foreground color:

1 Use selection tools to select one or more symbols.

2 Select Format➤Foreground Color.

or

Click the Foreground Color tool.

A color selection box appears.

3 Select a color.

4 Click OK.

❖ To apply a background color:

1 Use selection tools to select one or more symbols.

2 Select Format➤Background Color.

or

Click the Background Color tool.

A color selection box appears.

3 Select a color.

4 Click OK.

❖ **To apply a text color:**

1 Use selection tools to select one or more symbols.

2 Select Format➤Text Color.

or

Click the Text Color tool.

A color selection box appears.

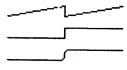
3 Select a color.

4 Click OK.

Applying a line style to symbols

You can modify the line style of any symbol in the model. Your choice of line style is saved in the current model and applies only to the current model.

Line styles apply to different symbols, as follows.

Styles	Applies style to ...	Description
Invisible ——— – – – – – – ····· –·–·–·–·– ––·––·––·	Lines and the outlines of shapes and of symbols containing text	Apply invisible, solid, dashed and dotted lines
——— ———► ———► ◄——— ◄———►	Lines drawn with Line or Freeform tools only	Position arrowhead
	All lines	Make angles jagged, right sharp, or right rounded

Your choice of line style applies to selected symbols. In addition, the choice of jagged, right sharp, or right rounded lines applies to all new lines.

❖ **To apply a line style:**

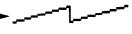
1 Use selection tools to select one or more symbols.

2 Select Format➤Line Style➤*line style*.

A checkmark appears next to the selected line style.

Bending and straightening lines

❖ **To bend a line:**

1 Select Format➤Line Style➤  .

2 Press CTRL while you click a point on the line where you want to insert an angle.

This point becomes a handle.

3 Release CTRL.

4 Drag and drop the handle.

❖ **To straighten a line:**

1 Click a line that has angles.

Handles appear on the line.

2 Press the CTRL key while you click a handle.

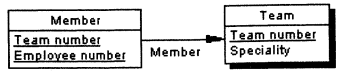
The handle and its angle disappear.

Adding and removing shadow

You can use shadow with the following selection of symbols:

Action	Selection
Add shadow to all selected symbols	At least one selected symbol does not have a shadow
Remove shadow	All selected symbols have shadows
Add shadow to all new symbols	No symbols are selected

In the model below, the table Team has shadow.



❖ **To add or remove shadow:**

1 Use selection tools to select one or more symbols.

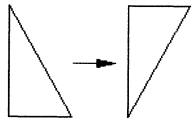
2 Select Format➤Shadow.
or
Click the Shadow tool.

Shadows appear or disappear behind the selected symbols.

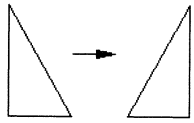
Flipping symbols

You can flip any asymmetrical symbols both horizontally and vertically.

For example, this diagram shows a triangle flipped horizontally:



This diagram shows a triangle flipped vertically:



❖ **To flip symbols:**

1 Use selection tools to select one or more asymmetrical symbols.

2 Select Arrange➤Disposition➤Flip Horizontal.
or
Select Arrange➤Disposition➤Flip Vertical.

Sizing symbols

❖ **To size symbols:**

1 Use selection tools to select one or more symbols.

2 Drag and drop any symbol handle to the size of an area to be filled by the selected symbols.

Unselected symbols in lasso area
If you first select symbols then select the lasso tool, eight handles mark the larger area that contains the selected symbols. By dragging these handles, you resize only the symbols previously selected. You do not necessarily resize all symbols in the lasso area.

Applying size preferences to symbols

After you define size preferences for height and width of symbols, these preferences apply to new symbols only. You can apply size preferences to existing symbols or to symbols that you have resized.

For information on how to define size preferences, see "Selecting a size preference" on page 608.

❖ To apply size preferences to symbols:

1 Use selection tools to select one or more symbols.

2 Select Arrange➤Normal Size.

Selected symbols are resized.

Adjusting symbol size to text

Adjusting symbols to text resizes symbols so that they display all the text they contain. For example, a rectangle adjusts its size to fit closely around its text. A table symbol adjusts its size to display full column names or column codes up to the truncation or word wrap length.

For information on how to adjust symbol size by default, see "Selecting general display preferences" on page 606.

❖ To adjust symbol size to text:

1 Use selection tools to select one or more symbols.

2 Select Arrange➤Adjust to Text.

or

Click the Adjust to Text tool.

The symbol adjusts to the length and height of the text it contains.

Hiding and showing symbols

You may want to hide symbols for the following reasons:

- Make a large model more understandable
- Show just the applicable section of the model graphic
- Prepare a partial printout

Showing hidden symbols returns all symbols to their original, visible state.

❖ **To hide symbols:**

1 Use selection tools to select one or more symbols.

2 Select Arrange➤Hide Symbols.

❖ **To show all hidden symbols:**

- Select Arrange➤Show Hidden Symbols.

> **Using lists to hide and show objects**
> From the list of entities, attributes, relationships, tables, columns, references, or views, you can show or hide objects by selecting or clearing the checkbox in the corresponding D column.

Grouping and ungrouping symbols

You manipulate grouped symbols as if they were a single symbol. For example, you can move and size grouped symbols in a single operation.

Ungrouping symbols returns them to their original, individually selectable state.

❖ **To group symbols:**

1 Use selection tools to select one or more symbols.

2 Select Arrange➤Group.

❖ **To ungroup symbols:**

1 Use selection tools to select a group of symbols.

2 Select Arrange➤Ungroup.

Protecting and unprotecting symbols

If you protect a symbol, you cannot select or modify it. Unprotecting symbols returns all symbols in the model to their original, modifiable state.

❖ To protect symbols:

1 Use selection tools to select one or more symbols.

2 Select Arrange➤Protect Symbols.

❖ To unprotect all symbols:

♦ Select Arrange➤Unprotect Symbols.

Moving symbols

You can change the arrangement of selected symbols as follows:

- Drag to a new location
- Align
- Set line disposition
- Center
- Arrange attach points
- Overlap

Dragging symbols to a new location

❖ **To move symbols:**

1 Use selection tools to select one or more symbols.

2 Drag the selection to a new location.

> **Unselected symbols in lasso area**
> If you first select symbols then select the lasso tool, eight handles mark the larger area that contains the selected symbols. By dragging this area, you move only the symbols previously selected. You do not necessarily move all symbols in the lasso area.

Aligning symbols

You align selected symbols in relation to each other.

You can align symbols from a menu or using a tool, as follows:

Align submenu item	Tool	Action
		Aligns left borders of selected symbols with leftmost selected symbol
		Aligns right borders of selected symbols with rightmost selected symbol
		Centers selected symbols on the most central selected symbol
	—	Stretches selected symbols to the width of the selection area (from the leftmost symbol to the rightmost symbol)
		Aligns tops of selected symbols with topmost selected symbol
		Aligns bottom selected symbols with lowest selected symbol
		Centers selected symbols on the most central selected symbol
	—	Stretches selected symbols to the height of the selection area (from the topmost symbol to the lowest symbol)

❖ **To align symbols:**

1 Use selection tools to select one or more symbols.

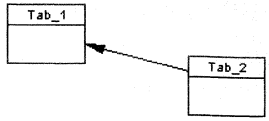
2 Select Arrange➤Align➤*alignment item.*
or
Click an alignment tool.

Setting line disposition

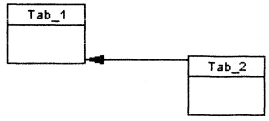
You can set the horizontal or vertical disposition of any line in the model. A change in disposition straightens lines and, where applicable, aligns their attach points (for example, aligning reference attach points).

Horizontal disposition

This model shows a reference before setting horizontal disposition.

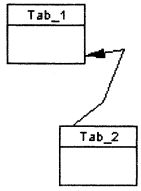


This is the same model after setting horizontal disposition.

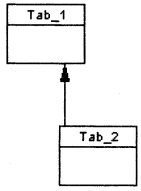


Vertical disposition

This model shows a reference before setting vertical disposition.



This is the same model after setting vertical disposition.



❖ **To set line disposition:**

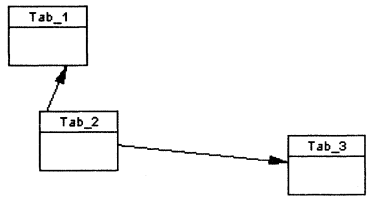
1 Use selection tools to select one or more lines.

2 Select Arrange➤Disposition➤Horizontal.
or
Select Arrange➤Disposition➤Vertical.

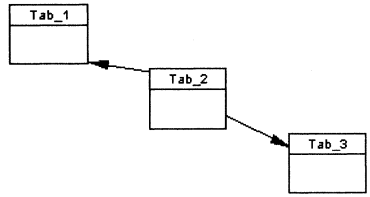
Centering a symbol

When you center a symbol, you move it to the center of gravity of the symbols to which it is connected.

This model shows the disposition of tables before centering on Tab_2.



This model shows the same tables after centering on Tab_2.



❖ **To center a symbol:**

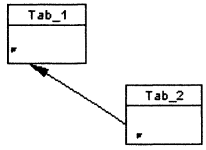
1 Use selection tools to select a symbol.

2 Select Arrange➤Disposition➤Arrange Symbols.

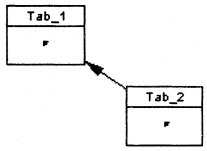
Arranging attach points

Arranging attach points moves the endpoints of a line to the center of objects on either end of a line.

This model shows a reference before arranging attach points:



This model shows the same reference after arranging attach points:



❖ **To arrange attach points:**

1 Use selection tools to select one or more lines.

2 Select Arrange➤Disposition➤Arrange Connectors.

Overlapping symbols

When you insert symbols in a model, they appear in overlapping layers, as follows:

- A new symbol that can contain text appears in front of all other symbols
- A new line appears behind all other symbols

The following items from the Arrange➤Order submenu change the layer of a selected symbol.

Submenu item	Description
Bring to front	Selected symbol appears in front of all other symbols
Send to back	Selected symbol appears behind all other symbols
Bring forward	Selected symbol moves forward one layer
Send backward	Selected symbol moves back one layer

❖ **To order overlapping symbols:**

1 Use selection tools to select a symbol.

2 Select Arrange➤Order➤*order item.*

Finding a symbol

You can locate the symbol that corresponds to an object in a list.

❖ **To find a symbol:**

1 Select Dictionary➤*List of* submenu.

2 Select an object in the list.

An arrow appears at the beginning of the line.

3 Click the Find button.

You return to the model workspace. The object and its synonyms are selected.

Inserting graphics

You can improve the visual appeal of your models by adding shapes and lines. For example, these graphic elements can surround parts of a model in order to mark off domains of activity.

Drawing shapes

❖ **To draw a rectangle:**

1 Click the Rectangle tool.

2 Drag and drop the far corner of a rectangle from any point in your model.

❖ **To draw a square:**

1 Click the Rectangle tool.

2 Press the CTRL key while you drag and drop the far corner of a square from any point in your model.

❖ **To draw an oval:**

1 Click the Oval tool.

2 Drag and drop the far end of an oval from any point in your model.

❖ **To draw a circle:**

1 Click the Oval tool

2 Press the CTRL key while you drag and drop the far end of a circle from any point in your model.

❖ **To draw a rounded rectangle:**

1 Click the Rounded Rectangle tool.

2 Drag and drop the far corner of a rounded rectangle from any point in your model.

❖ **To draw a rounded square:**

1 Click the Rounded Rectangle tool.

2 Press the CTRL key while you drag and drop the far corner of a rounded square from any point in your model.

❖ **To draw a polygon:**

1 Select Format➤Line Style➤ .

2 Click the Polygon tool.

3 Click the position of the start of the polygon.

4 Click each position where you want to add an angle in the polygon.

Straight lines join click positions.

5 Right click to close the polygon.

Drawing lines

❖ **To draw a straight line:**

1 Click the Line tool.

2 Drag and drop the endpoint of a line from any point in your model.

❖ **To draw a jagged line:**

1 Select Format➤Line Style➤ .

2 Click the Polyline tool.

3 Click the position of the start of the line.

4 Click each position where you want to add an angle in the line.

Straight lines join click positions.

5 Right-click to release the tool.

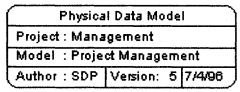
Inserting a title box

A title box displays the essential information about the model:

- Model type
- Project name
- Model name
- Author name
- Version number
- Date of last modification

This information comes from the property sheet for the model. The title box automatically takes into account any changes you make to model properties.

For example, the title box below displays information about the model Project Management.



❖ To insert a title box:

- Select Edit➤Add Title.

Importing images

A model can incorporate external graphic files in bitmap format (BMP) or Windows Metafile format (WMF). The imported images are stored in the file when the model is saved and are present the next time you open the model.

❖ To import an image into the current model:

1 Select Edit➤Import Image.

A standard Windows file selection dialog box appears.

2 Select a file

3 Click OK.

Using free text

Text may be associated with a symbol or not. **Free text** is text that is not associated with an object in the dictionary. For example, a note that you type in a rectangle is free text and a column name is not free text.

Inserting free text

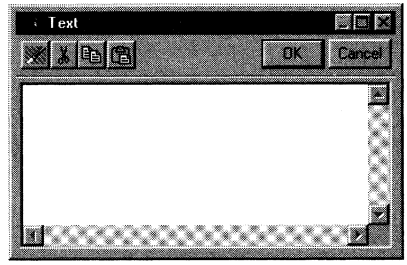
You can insert free text without any shape around it. You select and move this text like a symbol.

❖ **To insert free text:**

1 Click the text tool.

2 Click the model where you want to insert text.

A text input window appears.



3 Type the text inside the window.

4 Click OK.

Inserting text in a shape

You can insert text inside a rectangle, rounded rectangle, or oval. This text is attached to the shape. If you move the shape, the text moves with it. If you change the background color of the shape, the text changes, too.

You cannot insert text in a polygon.

❖ **To insert text in a shape:**

1 Double-click a shape in the model.

A text input window appears.

2 Type the text inside the window.

3 Click OK.

Selecting a color for free text

For information on selecting a color for text associated with objects in the dictionary, see "Selecting color preferences" on page 609.

❖ **To select a color to free text:**

1 Use selection tools to select free text in the model.

2 Select Format➤Text Color.

or

Click the Text Color tool.

A color selection box appears.

3 Select a color.

4 Click OK.

Selecting a style for free text

The following text styles are available:

- Normal
- Bold
- Italic
- Underline
- Strikeout

You cannot combine normal style with any other style. You can combine all other styles. For example, text can be bold and italic.

For information on selecting fonts for text associated with objects in the dictionary, see "Selecting font preferences" on page 608.

❖ **To select a style for free text:**

1 Use selection tools to select free text in the model.

2 Select Format➤Text Style➤*style*.

A checkmark appears next to active styles in the submenu.

Selecting a font for free text

A full range of font options are available for free text, including:

- Character font
- Size
- Style

You cannot cumulate normal style with any other style. You can cumulate all other styles. For example, text can be bold and italic.

For information on selecting fonts for text associated with objects in the dictionary, see "Selecting font preferences" on page 608.

❖ **To select a font for free text:**

1. Use selection tools to select free text in the model.
2. Select Format➢Fonts.
   or
   Click the Font tool.
3. Select a font name, size, and style from the corresponding lists.
4. Click OK.

Aligning free text

You can align free text vertically and horizontally. If text is in a shape, it aligns with the borders of the shape. If text is not in a shape, it aligns with text handles.

You cannot change the alignment of text associated with an object in the dictionary.

The following text alignment options are available:

Alignment	Text Alignment submenu item
Both horizontal and vertical	Middle
Horizontal	Right Left Centered
Vertical	Top Bottom Centered

❖ To align free text:

1 Use selection tools to select free text in the model.

2 Select Format➤Text Alignment➤*alignment item*.

When an alignment option is selected, a checkmark appears next to the submenu item.

Using zoom and centering

This section describes how to use zoom and centering to change the display of the model in the model workspace. These actions affect the visible portion of the model. They do not affect the actual size or placement of portions of the model in relation to the page grid.

Zooming in and out on the model

Zooming in on the model, magnifies the model without changing the center of the workspace.

Zooming out from the model, reduces the model without changing the center of the workspace.

❖ **To zoom in on the model:**

- Select View➤Zoom In.

or
Double-click the Zoom In tool.

❖ **To zoom out from the model:**

- Select View➤Zoom Out.

or
Double-click the Zoom Out tool.

Zooming to a specific scale

You can zoom in or out on a model from 25% to 400%. This does not change the center of the workspace.

❖ **To zoom to a specific scale:**

- Select View➤Scale➤*percentage*.

Zooming in and out on a point

When you zoom in on a point in the model, you magnify the model placing the point in the center of the workspace.

When you zoom out from a point in the model, you reduce the model placing the point in the center of the workspace.

❖ **To zoom in on a point:**

1 Click the Zoom In tool.

2 Click a point on the model.

The selected point appears in the center of the workspace.

❖ **To zoom out from a point:**

1 Click the Zoom Out tool.

2 Click a point on the model.

The selected point appears in the center of the workspace.

Zooming in on an area

When you zoom in on an area in the model, you magnify the model so that the selected area fills the entire workspace.

❖ **To zoom in on an area in the model:**

1 Click the Zoom tool

2 Drag an outline around an area in the model.

The model is magnified so that the selected area fills the entire workspace.

Displaying the entire model

You can display the entire model in the model workspace.

❖ **To display the entire model:**

♦ Select View➢Global View.

or

Click the Global tool.

Selecting pages to display

You can choose which pages to display in the workspace, as follows:

View menu	Tool	Workspace displays
Current Page		One page containing current selection only
Used Pages		Pages that contain symbols only
All Pages	—	All pages in the model

❖ **To select pages to display:**

- Select View➤ Current Page.
  or
  Select View➤Used Pages.
  or
  Select View➤All Pages.

Shifting workspaces

You shift the view of a model to make different portions visible in the workspace. This changes the position of models in relation to the workspace. It does not change their position on the page grid.

In addition to using the standard scrollbars, you can shift a view with the Grabber tool in relation to:

- A point in the model
- The workspace

❖ **To shift the display in relation to a point:**

1. Click the Grabber tool.
2. Press the SHIFT key, while you drag any point in the model to the position at which you want to display it.

❖ **To shift the display in relation to the workspace:**

1. Click the Grabber tool.
2. Press the CTRL key, while you drag the entire workspace.

   As you drag the workspace, a frame outlines the area you want to display.

Centering symbols in the workspace

❖ **To center symbols in the workspace:**

1 Use selection tools to select one or more symbols in the model.

2 Select View➤View Selection.

The selected symbols are centered in the workspace.

Returning to the previous view

You can display the previous view of the model taking into account the center of the view and its zoom.

Returning to the previous view allows you to toggle back and forth between a limited view and a global view of the model in the workspace.

❖ **To return to the previous view:**

♦ Select View➤Previous View.

Refreshing the display

You can refresh the display. This action does not change the center or the zoom scale of the workspace.

❖ **To refresh the display:**

♦ Select View➤Redisplay.

Showing and hiding the page grid

❖ **To show or hide the page grid:**

♦ Select Window➤Show Grid.

A checkmark appears next to the item when you show the grid.

Window display options

Window display options apply to the current model and to all models that you open subsequently.

You can set the following window display options:

Window option	Description
Color mode	Displays window in color or in black and white
Window color	Indicates the workspace color which is the background of the model

Activating and deactivating color mode

You activate color mode in order to display the model in color. You deactivate color mode to display the model in black-and-white.

❖ **To activate or deactivate color mode:**

♦ Select Window➢Color Mode.

When color mode is active, a checkmark appears next to menu item.

Selecting workspace color

The workspace color serves as a background color for the model.

❖ **To select a workspace color:**

1 Select Window➢Window Color.

A standard Windows color selection dialog box appears.

2 Click a color.

3 Click OK.

Printing model graphics

When you choose to print graphics, you print only the visual presentation of the model. You do not print background information about objects in the data dictionary.

Print options

The following print options are available:

Print option	What it prints
All	Entire model
Selected	Selected symbols only
Number of copies	Selected number of copies
Frame	Solid line border around graphic on all pages
Corner	Crop marks in corners on all pages. These marks make it possible to align model that print over several pages
Black&White	From a black-and-white printer
Color	From a color printer

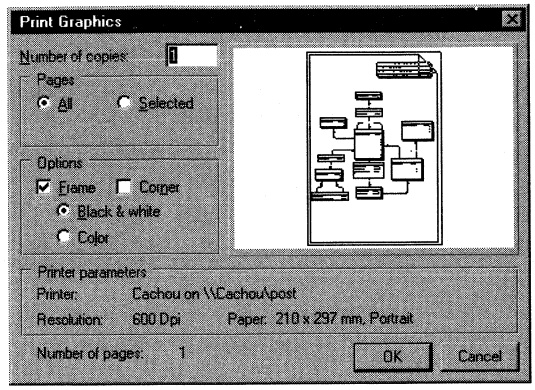
Printing a model

❖ **To print the current model:**

1 Select File➤Print Graphics.
or

Click the Printer tool.

The Print graphic dialog box displays default print options and the number of printed pages needed for the model.



2 Select print options.

3 Click OK.

Fitting the model on one printed page

You can reduce the print scale of a model graphic so that it prints on one page. By default, models are printed at 100% scale.

❖ **To fit the model on one printed page:**

- Select Arrange➤Fit to Page.

Exporting model graphics

You can export one or more selected symbols in Windows Metafile format (WMF). You decide whether the WMF file contains an image in black-and-white or in color.

❖ **To export symbols in WMF format:**

1 Use selection tools to select one or more symbols.

2 Select Edit➤Export in Color.

When color export is active, a checkmark appears next to the menu item.

3 Select Edit➤Export Image.

A standard Windows Save As dialog box appears.

4 Type the filename and click OK.

> **Copying to the clipboard in color**
> If you cut or copy selected symbols to the clipboard, you must select Edit➤Export in Color to copy the symbols in color.

CHAPTER 22

Report Generator

About this chapter

This chapter describes how to generate reports and report templates.

Contents

Using report templates

Reusable report templates indicate what information to include in reports.

You can use a report template to print a report for any model. You can also create your own templates.

Standard report templates

Standard report templates are installed in the REPORTPL sub-directory of your PowerDesigner directory. There are three types of standard templates:

Template type	File naming convention	Contents of resulting report
Full	**PDMFUL***language*.**TPL**	Table of contents All main model items
Standard	**PDMSTD***language*.**TPL**	Table of contents Model graphs Submodel graphs Most list items
List	**PDMLIS***language*.**TPL**	Title item All list items

For example, a PDM list report in English uses the template file PDMLISEN.TPL. A PDM standard report in French uses the template file PDMSTDFR.TPL.

Creating a report based on a template

You can create a report based on a template as follows:

- Print a report
- Save a report as an RTF file

If you use a template, you do not have to define report contents. However, you can select the objects to include in the report. You have the following selection options:

Option	Selection
All	All objects in the current model
Selected	Objects selected graphically in the current model
List	List of objects in the current model

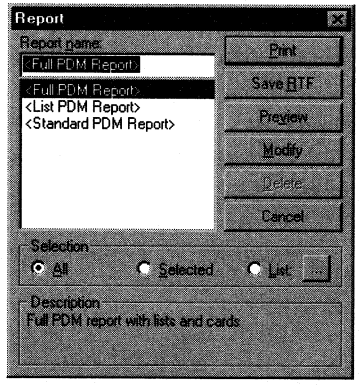
Printing a report based on a template

❖ To print a report based on a template:

1 Open a model.

2 Select File➤Create Report.

The Report dialog box appears.



3 Select a report template in the Report Name list.

4 Select the All radio button to include all objects in the report.
or
Select the Selected radio button to include graphically selected objects in the report.
or
Select the List radio button, click the button, select objects, and click OK.

5 Click the Print button.

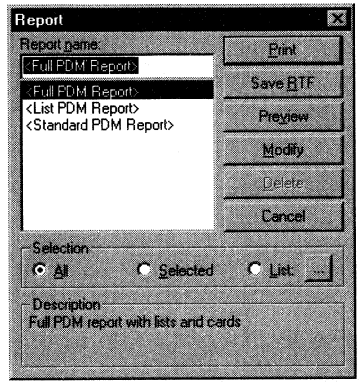
Saving an RTF file based on a template

❖ To save an RTF file based on a template:

1 Open a model.

2 Select File➤Create Report.

The Report dialog box appears.



3 Select a report template in the Report Name list.

4 Select the All radio button to include all objects in the report.
or
Select the Selected radio button to include graphically selected objects in the report.
or
Select the List radio button, click the ... button, select objects, and click OK.

5 Click the Save RTF button.

A Save As dialog box appears.

6 Type a filename for the RTF file.

7 Click OK.

A confirmation box indicates that the RTF file has been successfully generated.

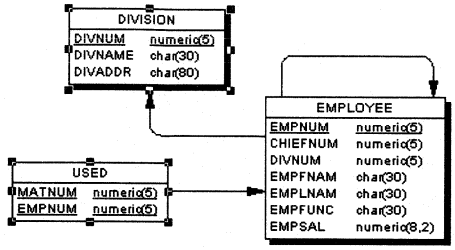
8 Click OK.

Including graphically selected objects in a report based on a template

❖ To include graphically selected objects in a report based on a template:

1 Use selection tools to select symbols in the model.

For example, the model below shows two tables selected.



2 Select File➤Create Report.

The Report dialog box opens.

3 Select a report template from the Report Name list.

4 Select the Selected radio button.

5 Click the Print button.
or
Click the Save RTF button.

Including listed objects in a report based on a template

A report based on a template can print information about objects coming from a global model and any of its related submodels.

Standard report templates that appear in brackets do not recall the previous list of selected objects.

❖ To include listed objects in a report based on a template:

1 Select File➤Create Report.

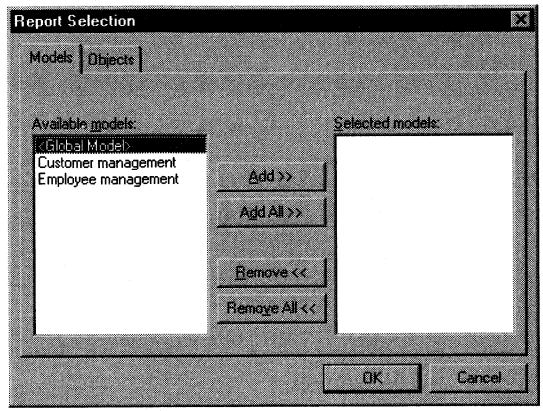
The Report dialog box opens.

2 Select a report template from the Report Name list.

3 Click the List radio button.

4 Click the button next to the List radio button.

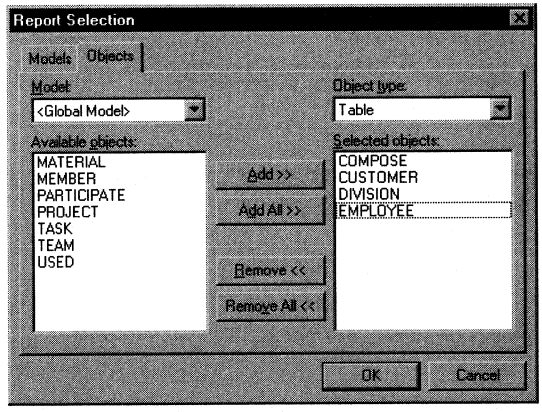
The Report Selection dialog box opens to the Models page which lists the current global model and its associated submodels.



5 Select one or more models in the Available Models list and click the Add button.

6 Click the Objects tab.

The Report Selection dialog box opens to the Objects page. The Selected Objects list includes all objects in the selected models.



7 In the Selected Objects list, select the objects that you do not want to include in the report.

8 Click the Remove button.

9 Click OK.

You return to the Report Dialog box.

10 Click the Print button.
or
Click the Save RTF button.

Creating a report template

You can create a template and save it with the current model.

❖ **To create a template:**

1 Select File➢Create Report.

The Report dialog box opens.

2 Type a name in the Report Name box.

3 Click the Create button.

The Report Editor window opens.

4 Build a report structure in the Contents pane.

For information on using the Report Editor, see "Using the Report Editor" on page 656.

5 Select Report➢Close.

A confirmation box asks you if you want to save changes.

6 Click Yes.

The next time you create a report for the current model, the template will appear in the Report Name list.

Saving a modified report template

If you modify a template, you can save it with the current model.

Unmodified versions of standard templates appear in brackets in the Report Name list. Templates that you have already modified appear without brackets.

❖ **To save a modified template:**

1 Select File➢Create Report.

The Report dialog box opens.

2 Select a report template from the Report Name list.

3 Click the Modify button.

The Report Editor window opens.

4 Modify the report structure in the Contents pane.

For information on using the Report Editor, see "Using the Report Editor" on page 656.

5 Select Report➤Close.

A confirmation box asks you if you want to save changes.

6 Click Yes.

The next time you create a report for the current model, the modified template will appear in the Report Name list.

Exporting a report template

You export a template so that you can use it with other models. If you do not export a modified template, it is only available in the current model.

When you export a template, you save its structure in an RTP file.

There are two ways to export templates:

Export option	Access to the exported template
Do not register	Import the template from the Report Editor
Register	Select template from the list of templates *or* import template from the Report Editor

Exporting a report template without registration

When you export a template without registration, you save its structure in an RTP file which you can then import from a different model.

❖ **To export a report template:**

1 Select File➤Create Report.

The Report dialog box opens. Registered templates appear in brackets in the Report Name list. Templates that you have modified appear without brackets.

2 Select a report template from the Report Name list and click the Modify button.
or
Type a name in the Report Name box and click the Create button.

The Report Editor window opens.

3 Define the report structure in the Contents pane.

For information on using the Report Editor, see "Using the Report Editor" on page 656.

4 Select Report➤Export Template.

A standard Save As dialog box appears.

5 Type a filename for the RTP file.

6 Click OK.

7 Select Report➤Close.

A confirmation box asks you if you want to save changes.

8 Click Yes.

Exporting a report template with registration

You can register an exported template. In this case, the template name and description appear in the list of templates for all models of the same type and language.

If you register a template, you provide the following information:

Summary information	Description	Required?
Name	Template name as it will appear in the Report Name list	✓
Author	Your name	—
Version	Version number	—
Description	Brief description as it will appear in the Report Name list	—

❖ To export a report template:

1 Select File➤Create Report.

The Report dialog box opens. Registered templates appear in brackets in the Report Name list. Templates that you have modified appear without brackets.

2 Select a report template from the Report Name list and click the Modify button.
or
Type a name in the Report Name box and click the Create button.

The Report Editor window opens.

3 Define the report structure in the Contents pane.

For information on using the Report Editor, see "Using the Report Editor" on page 656.

4 Select Report➤Export Template.

A standard Save As dialog box appears.

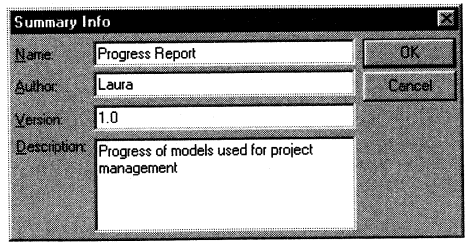
5 Type a filename for the RTP file.

> **Directory for report tempates**
> You must save registered templates in the REPORTPL subdirectory of your PowerDesigner directory. Templates saved in other directories are not visible from the Report dialog box.

6 Select the Register checkbox.

7 Click OK.

The Summary Info dialog box appears.



8 Type Summary Info for the exported template.

9 Click OK.

10 Select Report➤Close.

A confirmation box asks you if you want to save changes.

11 Click Yes.

The next time you select File➤Create Report, the name of the exported template will appear in brackets in the Report Name list.

Importing a report template

After you export a report template from one model, you can import the report template into another model.

You can import a template exported from another model type. For example, you can export a CDM template, and import it as a PDM template. The imported template automatically removes all objects that do not apply to the current model type.

❖ To import a report template:

1 Select File➤Create Report.

The Report dialog box opens.

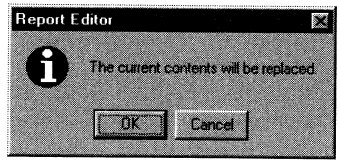
2 Type a name in the Report Name box.

3 Click the Create button.

The Report Editor window opens.

4 Select Report➤Import Template.

A confirmation box appears.



5 Click OK.

A standard file selection dialog box appears.

6 Select an RTP file.

7 Click OK.

The report structure appears in the Contents pane.

8 Select Report➤Close.

A confirmation box asks you if you want to save changes.

9 Click Yes.

The next time you create a report for the current model, the imported template will appear in the Report Name list.

Copying from a report template

When you copy from a report template, you replace the contents of the current report with the contents of a template that is attached to the same model.

❖ To copy from a report template:

1 Select File➤Create Report.

The Report dialog box opens.

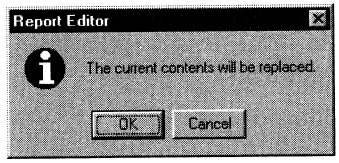
2 Type a name in the Report Name box.

3 Click the Create button.

The Report Editor window opens.

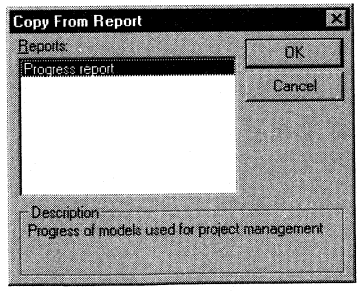
4 Select Report➤Copy From.

A confirmation box appears.



5 Click OK.

The Copy From Report dialog box displays a list of templates saved with the current model. It does not include standard templates.



6 Select a name in the Reports list.

7 Click OK.

The report structure appears in the Contents pane.

8 Select Report➤Close.

A confirmation box asks you if you want to save changes.

9 Click Yes.

The next time you create a report for the current model, the created template will appear in the Report Name list.

Using the Report Editor

You use the Report Editor to build report structures. A report structure takes the form of an indented outline that dictates what to print in a report.

From the Report Editor, you can:

- Create a report structure
- Modify a report structure
- Include listed objects in a report structure
- Preview a report
- Print a report
- Save a report to an RTF file

Opening the Report Editor

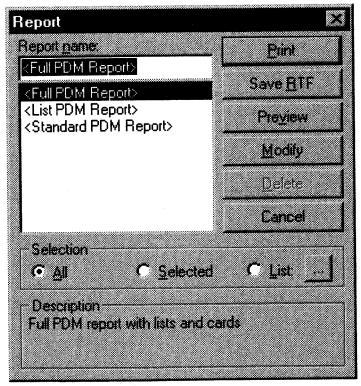
You open the Report Editor when you choose to create or modify a report template.

❖ To open the Report Editor:

1 Open a model.

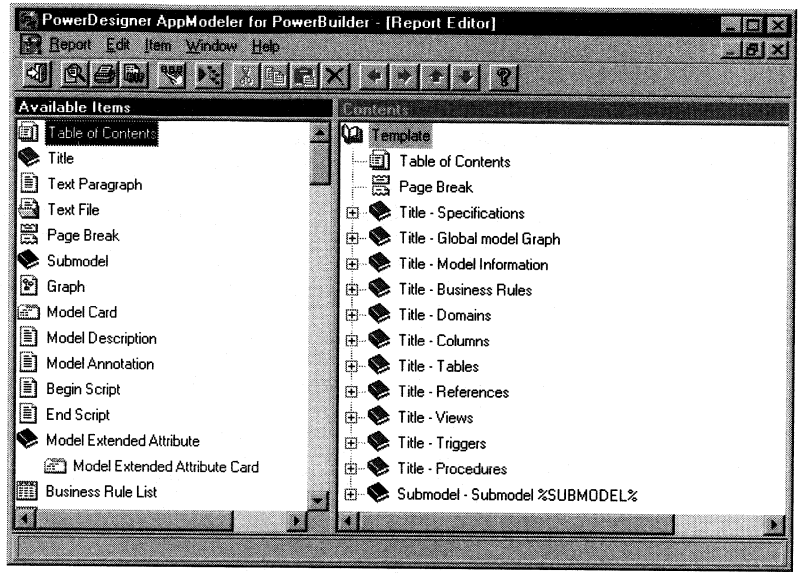
2 Select File➢Create Report.

The Report dialog box appears.



3 Select a report template from the Report Name list and click the Modify button.
or
Type a name in the Report Name box and click the Create button.

The Report Editor window opens.



Setting Report Editor options

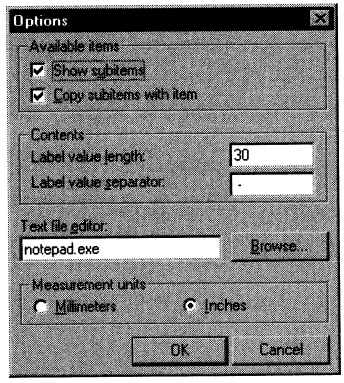
You can set the options for the Report Editor. These options are saved in the Windows registry or the PD6.INI file.

Option	Description
Show subitems	Display dependent items in the Available Items pane
Copy subitems with item	Add dependent items with node when you add a node to a report by dragging it from the Available Items pane to the Contents pane
Label value length	Maximum number of characters of the node title to display in the node label. The node label always displays the node type in its entirety
Label value separator	Character or characters in the node label that separate node type from node title
Text file editor	Name of text file editor
Measurement units	Displays column widths, tabs, and margins in inches or in millimeters in Report Editor dialog boxes

❖ To set Report Editor options:

1 Select Report➢Options.

The Options dialog box appears.



2 Select report options.

3 Click OK.

Printing a report from the Report Editor

❖ **To print a report from the Report Editor:**

♦ Select Report➤Print.
or

Click the Print tool.

Saving an RTF file from the Report Editor

❖ **To save an RTF file from the Report Editor:**

♦ Select Report➤Save RTF.
or

Click the Save RTF tool.

Selecting objects to include in a report

A report can include objects coming from a global model and any of its related submodels. If you save a report structure, you also save the selection of objects.

In the Report Editor, you select objects to include in a report by choosing one the following selections:

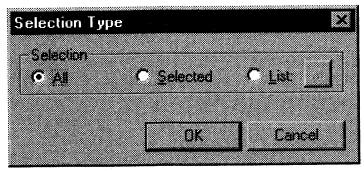
Selection	Description
All	Includes all objects in the current model
Selected	Includes objects that you selected graphically before you opened the Report Editor
List	Includes objects from a list that you define

❖ **To select objects to include in a report:**

1 Select Report➤Select Objects.
or

Click the Selection tool.

The Selection Type dialog box appears.

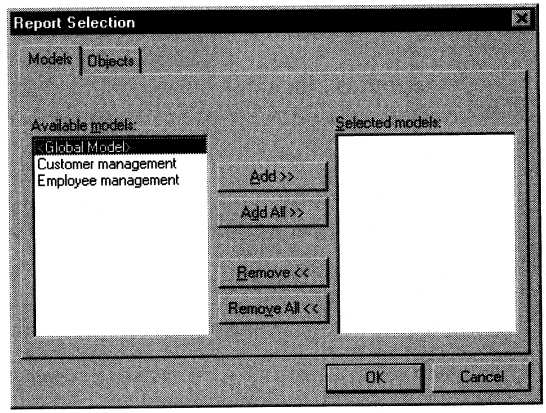


2 Select the All radio button and click OK.
or
Select the Selected radio button and click OK.
or
Select the List radio button.

If you selected the List radio, you define the list of objects as follows:

1 Click the ... button.

The Report Selection dialog box opens to the Models page which lists the current global model and its associated submodels.

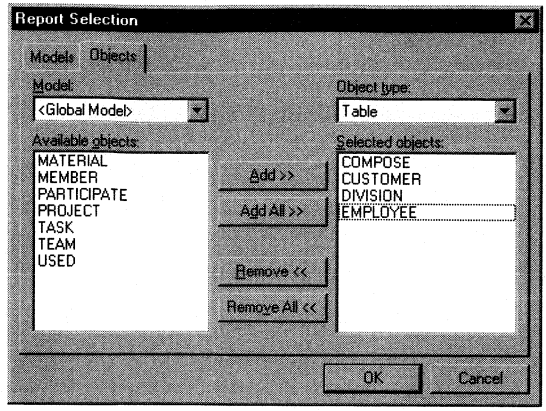


2 Select one or more models in the Available Models list.

3 Click the Add button.

4 Click the Objects tab.

The Report Selection dialog box opens to the Objects page. The Selected Objects list includes all objects in the selected models.



5 In the Selected Objects list, select the objects that you do not want to include in the report.

6 Click the Remove button.

7 Click OK in each of the dialog boxes.

Closing the Report Editor

When you close the Report Editor, you have the choice to save the current report. If you save the report, its template is always available with the current model. You also save any object selections in the report.

❖ To close the Report Editor:

1 Select Report➤Close.

A confirmation message box asks if you want to save changes to the report structure.

2 Click Yes.

> **If you select Report➤Exit**
> If you select Report➤Exit, you exit both the Report Editor and PowerDesigner. To remain in PowerDesigner, click Cancel in the confirmation message box.

Building reports

You build a report by designating what it contains including any of the following types of items:

Item	Description
Node	Contains dependent items at a lower level. Nodes appear as books in the Report Editor
Independent item	Can appear anywhere in the contents of a report (for example, a table of contents or a graph)
Model-dependent item	Contains information about a model or a submodel
Object list	Contains list of objects coming from a model or submodel
Object-dependent item	Contains information about a specific object type and can only appear under an object node

Adding items to a report

You build a report by adding report items to the Contents pane. When you add an item to the Contents pane, the item remains in the Available Items pane. You can insert the same item many times in the same report contents.

❖ **To add an item to a report:**

- Drag an item from the Available Items pane into the Contents pane.
  or
  Double-click an item in the Available Items pane.
  or
  
  Select one or more items in the Available Items pane and click the Add-to-report tool.

Drag-and-drop not possible
With the drag-and-drop method, the pointer becomes a barred circle when the mouse position is not an available drop target. You can only drop the item when the barred circle becomes a pointer again.

Dropping an item on a node
If you drop an item on a node, it is added below the node as a dependent item. To add the item at the same level as the node, press the ALT key while you drop the item.

Adding nodes to a report

You can add a node with or without its dependent items to the report contents. After you add a node without its dependent items, you can add its dependent items one-by-one.

❖ **To add a node with its dependent items to a report:**

1 Select Report➤Options.

The Options dialog box appears.

2 Select the Copy Subitems With Item checkbox.

3 Click OK.

4 Drag a node from the Available Items pane to the Contents pane.

The node and its dependent items appear in the Contents pane.

❖ **To add a node without its dependent items to a report:**

1 Select Report➤Options.

The Options dialog box appears.

2 Clear the Copy Subitems With Item checkbox.

3 Click OK.

4 Drag a node from the Available Items pane to the Contents pane.

The node appears without its dependent items in the Contents pane. You can drag dependent items one-by-one into the Contents pane and drop them on the node.

Inserting a table of contents

You can insert a table of contents item anywhere in the report contents. However, it always specifies a complete list of the node titles and their page numbers.

The table of contents item is included immediately under the Template item in the Standard and Full report templates. In reports based on these templates, the table of contents page prints on the page following the title page.

❖ To include a table of contents:

- Drag a table of contents item from the Available Items pane to the Contents pane.

Inserting a text paragraph

The text paragraph item contains free text that you type in an Editor dialog box.

❖ To insert a text paragraph:

1. Drag a text paragraph item from the Available Items pane to the Contents pane.
2. Double-click the text paragraph in the Contents pane.
   or
   Select the text paragraph in the Contents pane and select Item➤Edit.

   The Editor dialog box opens.
3. Type the text.
4. Click OK.

Text editor selection
You can select a text editor as a report option by selecting Report➤Options. The editor is NOTEPAD.EXE by default.

Inserting a text file

You can insert an ASCII format text file in a report structure.

❖ **To insert a text file:**

1 Drag a text file item from the Available Items pane to the Contents pane.

A standard Open dialog box appears.

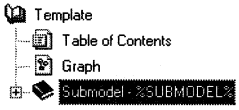
2 Select a text file.

3 Click OK.

Inserting graphs

You can add model graphs to a report by inserting the graph item in the Contents pane. If you insert a graph item under a submodel node, the report contains submodel graphs.

The following structure prints the graph of a global model:



The following structure prints the graphs of each submodel:

Repositioning an item

By repositioning items in the Contents pane, you change the order of items in the report.

When you reposition a node, you also reposition its dependent items.

The Template item is the first item in all reports. You cannot reposition or remove the Template item in the Contents pane.

❖ **To reposition an item in a report:**

♦ Select an item in the Contents pane and
click the arrow buttons in the Report Editor toolbar.
or
Drag an item to a new location in the Contents pane.

Changing the depth level of an item

You can position items at the same level or below nodes. You cannot position object-dependent items at or above the level of their object nodes.

❖ **To change the depth level of an item:**

1 Select an item in the Contents pane.

 2 Click the right or left arrow tools.
or
Select Item➤Raise Level.
or
Select Item➤Lower Level.

Copying an item in a report

You can copy items in the Contents pane.

When you copy a node, you also copy its dependent items.

You cannot copy the Template item.

❖ **To copy an item:**

♦ Hold down the CTRL key while you Drag an item to a second location.

Deleting an item from a report

In the Contents pane, you can delete any item except for the Template node.

When you delete a node, you also delete its dependent items.

❖ **To delete an item from a report:**

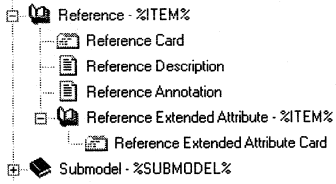
1 Select an item in the Contents pane.

 2 Click the Delete tool.

Using nodes

A node is an item that contains dependent items at a lower level. Nodes appear as books in the Report Editor.

For example, the Reference node contains items that describe a reference. It also contains another node.



Printing nodes

The report generator interprets nodes as print loops. For each object corresponding to a node, the report prints all the dependent items.

For example, if you apply the structure above to a model that contains two references MEMBER and USER, the report contains in the following order:

Reference card for MEMBER
Reference description for MEMBER
Reference annotation for MEMBER
Reference extended attribute card for each of MEMBER's extended attributes

Reference card for USER
Reference description for USER
Reference annotation for USER
Reference extended attribute card for each of USER's extended attributes

Node dependency

Nodes can depend on other nodes. The following table list all possible node dependencies.

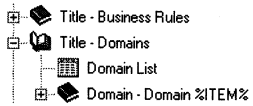
Node type	Can depend on	Can be depended on by
Template	—	All nodes and items in the report contents
Title	Template node Submodel node Object node	Title nodes Submodel nodes Object nodes Independent items Object-dependent items
Submodel	Template node Title node	Title nodes Object nodes Independent items
Object	Template node Title node Submodel node	Title nodes Object dependent nodes Independent items Object-dependent items
Object-dependent	Object node	Title nodes Independent items Object-dependent items

Expanding and collapsing nodes

You can expand and collapse the display of a node.

In an expanded node, all dependent items appear on a branch below the node and a minus sign precedes the node. In a collapsed node, all dependent items are hidden, and a plus sign precedes the node.

For example, here the Business Rules node is collapsed and the Domains node is expanded.



❖ **To expand or collapse a node:**

- Double-click the node in the Contents pane.

Using node titles

You can associate title text with nodes. In a report, a title prints just above the item that follows it in the Contents pane. A title contains free text.

By adding Title nodes to the Contents pane, you can insert titles anywhere and as often as you want. You can use title nodes to organize a report into chapters or sections.

Submodel, object, and object-dependent nodes have default titles that incorporate variable information:

Node type	Default title	Variable syntax
Title	None	—
Submodel	Submodel name	%SUBMODEL%
Object	Object name	%ITEM%
Object-dependent	Object name	%ITEM%

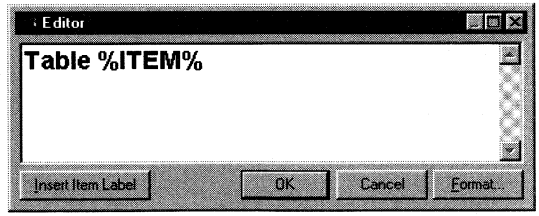
Modifying a node title

❖ **To modify a node title:**

1 Select a node in the Contents pane.

2 Select Item➢Edit.

The Editor dialog box opens and displays the current title text.



3 Type changes to the node title.

4 If you want to insert a variable for object name or submodel name, click the Insert Item Label button.

5 Click OK.

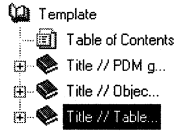
The first line of the title appears next to the node in the Contents pane.

Formatting node labels

In the Contents pane, each node has a label. This label indicates the type of the node and its title. For the display of labels in the Contents pane, you can select the following options:

Option	Description
Label value length	Number of characters from the node title displayed in the node label. The node label always displays the node type in its entirety
Label value separator	Character or characters in the node label that separate node type from node title

For example, the following tree displays five characters of the title and uses the separator character //.



❖ To format node labels:

1 Select Report➤Options.

The Options dialog box appears.

2 Type the number of characters that you want to display in the node label in the Label Value Length textbox.

3 Type one or more characters in the Label Value Separator textbox.

Spaces between label and separator
If you want to include spaces between label text and separator characters, you must type these spaces in the Label Value Separator box.

4 Click OK.

In the Contents pane, node labels change their appearance.

Using items

You can use different types of items in your report structure:

- Model-dependent items contain information about a model or a submodel
- Object-dependent items contain information about a specific type of object
- Independent items do not contain information about specific models or objects

Model-dependent items

Model-dependent items print information about a model or a submodel. If you insert a model-dependent item under a submodel node, the item contains information related to submodels.

You cannot insert the model-dependent items under object nodes.

The following are model-dependent items:

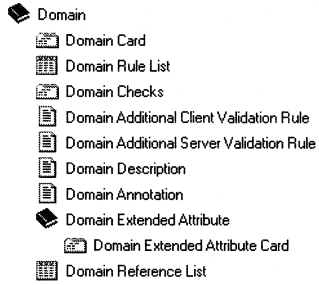
Dependent item	Print format	Content
Card	List of properties	Properties of a model
Description	Text paragraph	Description of model
Annotation	Text paragraph	Annotation of model
Begin script	Text paragraph	Text to add at the beginning of a script generated from this model
End script	Text paragraph	Text to add at the end of a script generated from this model
Object list	Table	Table that lists all objects of a certain type coming in this model

Object-dependent items

Object nodes contain information about objects in the dictionary. These nodes have a restricted set of dependent items called object-dependent items.

You can delete and change the order of object-dependent items in the Contents pane. You cannot move object-dependent objects under other object nodes.

For example, in the report structure below, the Domain object node precedes its dependent items.



Printing object-dependent items

The report generator prints each object-dependent item once for each occurrence of the object. If an item does not have information for a particular object, the item does not appear in the resulting report.

If you apply the outline above to a model that contains a domain that does not have any associated rules or check parameters, it prints the domain card following directly by the domain description.

The following object-dependent items are present under all object nodes:

Dependent item	Print format	Content
Card	List of properties	Properties of an object
Description	Text paragraph	Description of an object
Annotation	Text paragraph	Annotation of an object

In addition to those items mentioned above, each object node has dependent items related to the object type. For example, a business rule node has a dependent item for server expression and a column node has a dependent item for extended attributes.

Independent items

Independent items do not contain information about specific models or objects

You can insert the following independent items anywhere in the report structure:

Item	Printed result	Icon
Table of contents	Complete report table of contents	
Text paragraph	Paragraph of free text that you input	
Text file	Contents of a text file	
Page break	Prints next item on the following page	
Graph	Prints graph of model (or submodel if under a Submodel node)	

Formatting items

In a report, you have access to the following format options:

- Text format
- Graph format
- Table format

> **Context menus**
> You can access the commands that apply to any item by clicking it with the right mouse button.

Defining global format

Global formats apply to all new items that you add to a report structure. You apply global format to objects in the Available Items pane. After you define global format, it applies every time you drag that item to the Contents pane.

Global format does not apply to items already in the Contents pane.

❖ **To define a global format:**

1 Select an item in the Available Items pane.

2 Select Item➤Format.

A dialog box that corresponds to the type of item appears. For example, if you select a graph, the Graph Format dialog box appears.

3 Type format changes.

4 Click OK.

Every time you move that item to the Contents pane, it retains the global format.

Formatting individual items

You can change the format of an individual item already in the Contents pane.

❖ **To format an individual item:**

1 Select an item in the Contents pane.

2 Select Item➤Format.

A dialog box that corresponds to the type of item appears. For example, if you select a graph, the Graph Format dialog box appears.

3 Type format changes.

4 Click OK.

Your format choices apply to the selected object only.

Selecting text to format

Some items include more than one text selection. You can format each text selection differently.

The following text selections are available for card items (lists of properties).

Text selection	Description	Example
Item title	Title of the item	Table Card
Text title	Property name on a list of properties	Table Code
Text item	Value of a property	EMPLOYEE

The following text selections are available for model- and object-dependent items in text format.

Text selection	Description	Example
Text title	Title of the item	Table Description
Text item	Text of the description, annotation, or script	This table describes different employee characteristics

The following text selections are available for list items which print in table format.

Text selection	Description	Example
Item title	Title of the item	List of Indexes
Column title	Heading of column	Code
Column text	Value in the column	PK_EMPLOYEE

Selecting a font

You can select a font for any item, except for the graph and page break.

The following font choices are available:

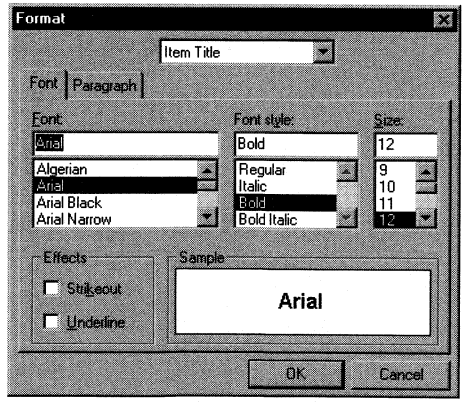
- Font
- Font style
- Size
- Underline
- Strikeout

❖ To select a font:

1 Select an item.

2 Select Item➤Format.

The Format dialog box opens to the Font page.



3 If the dialog box has a text selection dropdown listbox, select the text that you want to format.

4 Type font format choices.

5 Click OK.

Formatting a paragraph

You can select a paragraph format for all items, except for the graph and page break.

The following paragraph format choices are available:

- Indentation (left, right, first line)
- Spacing (before, after, internal line spacing)
- Frame
- Justification (alignment)
- Tab spacing

❖ To format a paragraph:

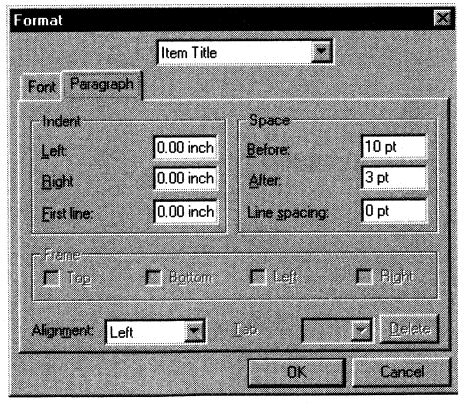
1 Select an item.

2 Select Item➤Format.

The Format dialog box appears.

3 Click the Paragraph tab.

The Format dialog box opens to the Paragraph page.



4 If the dialog box has a text selection dropdown listbox, select the text that you want to format.

5 Type paragraph format choices.

6 Click OK.

Editing text

❖ **To modify a text paragraph or a text file:**

1 Double-click the item in the Contents pane.
or
Select the item in the Contents pane and select Item➤Edit.

The Editor dialog box opens.

2 Type modifications to the text.

3 Click OK.

> **Text editor selection**
> You can select a text editor as a report option by selecting Report➤Options. The editor is NOTEPAD.EXE by default.

Formatting graphs

Graphs have the following formatting options:

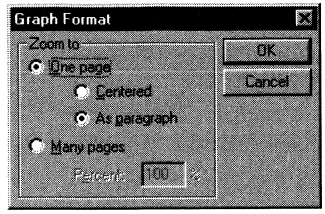
Format	Option	What it does
One Page	Centered	Centers the graph on its own page and, if necessary, reduces the graph to fit on one page
One Page	As Paragraph	Prints the graph on the next line in the report and, if necessary, reduces the graph to fit on one page
Many pages	Percent	Prints the graph on the number of pages that it occupies at a percentage print scale

> **Color graphs**
> You can print graphs in color by selecting Edit➤Export in Color from the main model window.

❖ To format a graph:

1 Double-click the graph item in the Contents pane.
or
Select a graph item and select Item➤Format.

The Graph Format dialog box opens.



2 Select the One Page radio button and the Centered radio button.
or
Select the One Page radio button and the As Paragraph radio button.
or
Select the Many Pages radio button and type a percentage print scale for the graph.

3 Click OK.

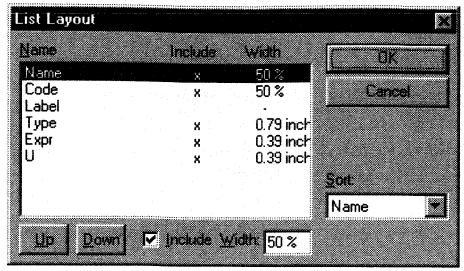
Selecting columns for a table

List items print in the form of tables. You can choose the columns to include in these tables.

❖ To select columns for a table:

1 Double-click a list item.
or
Select the list item and select Item➤Layout.

The List Layout dialog box appears. It displays the names of columns available for the table. An x in the Include column indicates a column to include in the table.



2 Select a column name.

3 Select the Include checkbox to include the column in the table.
or
Clear the Include checkbox to remove the column from the table.

4 Click OK.

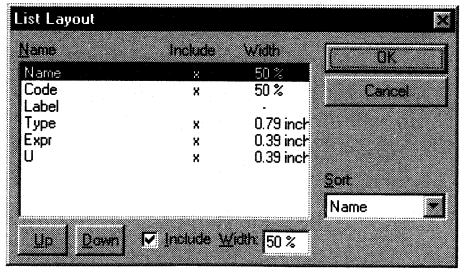
Changing column order in a table

You can change the order of columns in tables which list items generate.

❖ To change column order in a table:

1 Double-click a list item.
or
Select a list item and select Item➤Layout.

The List Layout dialog box appears. In this dialog box, the top-to-bottom order of columns corresponds to their left-to-right order in the printed table.



2 Select a column name.

3 Click the Up button to move the column up one place in the list.
or
Click the Down button to move the column down one place in the list.

4 Repeat the previous step until columns are in the order that you want.

5 Click OK.

Modifying column width in a table

You can indicate column width in one of the following ways:

Width	Abbreviation	Description
Inches	inch	Indicates column width in inches
Millimeters	mm	Indicates column width in millimeters
Percentage	%	Indicates column width as percentage of table space remaining after sizing other columns in inches and millimeters

❖ To modify column width in a table:

1 Double-click a list item.
or
Select a list item and select Item➤Layout.

The List Layout dialog box appears. The column width appears in the Width column.

2 Select a column in the list.

3 Type a width followed by **inch** for inches, **mm** for millimeters, or **%** for a percentage of remaining table space.

> **Report options Inches and Millimeters**
> If you select the report option Inches, widths that you type in millimeters appear as the equivalent in inches. If you select the report option Millimeters, widths that you type in inches appear as the equivalent in millimeters. You select report options by selecting Report➤Options.

4 Repeat the previous step until all columns have the widths that you want.

5 Click OK.

Setting up report pages

Before you print a report, you can set up the pages as follows:

- Include a header and footer
- Include a title page

These options do not correspond to items in the Contents pane.

Modifying a report header or footer

You can modify the header and footer of your report.

Headers and footers can include the following variable information:

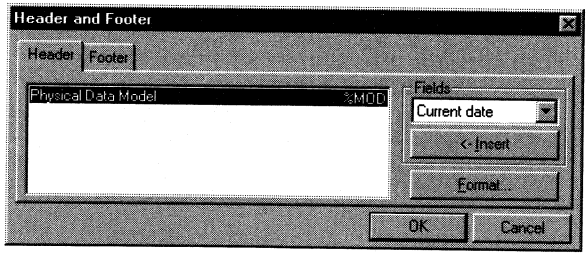
Current date
Current time
Page number
Model name
Model code

By default, the header shows the model type and the name of the model. By default, the footer shows the current date and the page number.

❖ To modify a report header:

1 Select Edit➤Header and Footer.

The Header and Footer dialog box opens to the Header page.



2 Type changes to header text.

> **Tabs in a header**
> Use the key combination CTRL + TAB to insert tabulations in a header.

3 Position the cursor in the header text where you want to add variable information.

4 Select a field from the Fields dropdown listbox.

5 Click the Insert button.

6 Click the Format button.

The format dialog box appears open to the Font page.

7 Select a font format for the header.

8 Click the Paragraph tab.

The format dialog box appears open to the Paragraph page.

9 Select a paragraph format for the header.

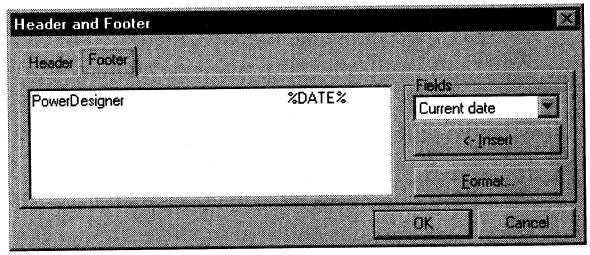
10 Click OK in each of the dialog boxes.

❖ To modify a report footer:

1 Select Edit➤Header and Footer.

The Header and Footer dialog box appears.

2 Click the Footer tab.



3 Type changes to footer text.

> **Tabs in a footer**
> Use the key combination CTRL + TAB to insert tabulations in a footer.

4 Position the cursor in the footer text where you want to add variable information.

5 Select a field from the Fields dropdown listbox.

6 Click the Insert button.

7 Click the Format button.

The format dialog box appears open to the Font page.

8 Select a font format for the footer.

9 Click the Paragraph tab.

The format dialog box appears open to the Paragraph page.

10 Select a paragraph format for the footer.

11 Click OK in each of the dialog boxes.

Indicating summary information

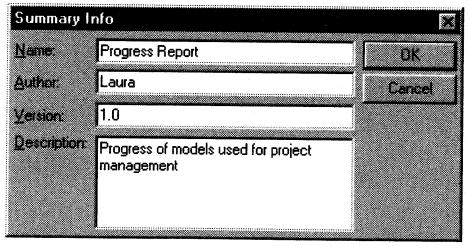
You can indicate summary information for a report. The summary information includes:

Information	Description	Appears where
Name	Name of the current report	Title page
Author	Name of the report's author	Title page (if selected)
Version	Version number of the report	Title page (if selected)
Description	Free text describing the report	Report dialog box

❖ To indicate summary information:

1 Click Edit➤Summary Info.

This opens the Summary Info dialog box.



2 Type summary information.

3 Click OK.

Including a title page

A title page prints as the first page in a report.

A title page can include any of the following information:

Information	Prints
Title	Free text, by default the name of the current report
Frame title	Frame around title text
Author	Name of the report author, as indicated in the report summary
Version	Version number of the report, as indicated in the report summary
Date	Current date when the report prints

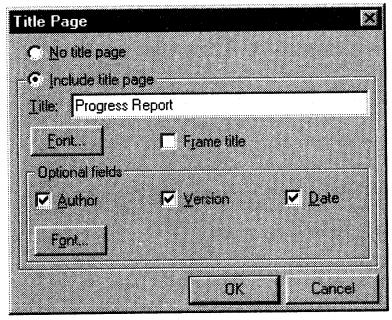
You can choose one font style for the title, and a different font style for all other text.

❖ To include a title page in a report:

1 Select the Template item.

2 Select Item➢Title Page.

This Title Page dialog box appears.



3 Click the Include Title Page radio button.

4 Type the Title text.

5 Click the Font button below the Title box.

The format dialog box appears open to the Font page.

6 Select a font format for the title.

7 Click OK.

8 Select the Frame Title checkbox to print a frame around the title.
or
Clear the Frame Title checkbox.

9 Select the checkboxes for the optional fields that you want to appear on the title page.

> **Including author and version**
> For author and version information to appear on a title page, you must type them in the Summary Info dialog box by selecting Edit➤Summary Info.

10 Click the Font button below the Author checkbox.

The format dialog box appears open to the Font page.

11 Select a font format for the optional fields.

12 Click OK in each of the dialog boxes.

Using print preview

From the Report Editor, you can display a print preview which shows what a report will look like if printed. Different navigation tools help you find sections of the report to preview.

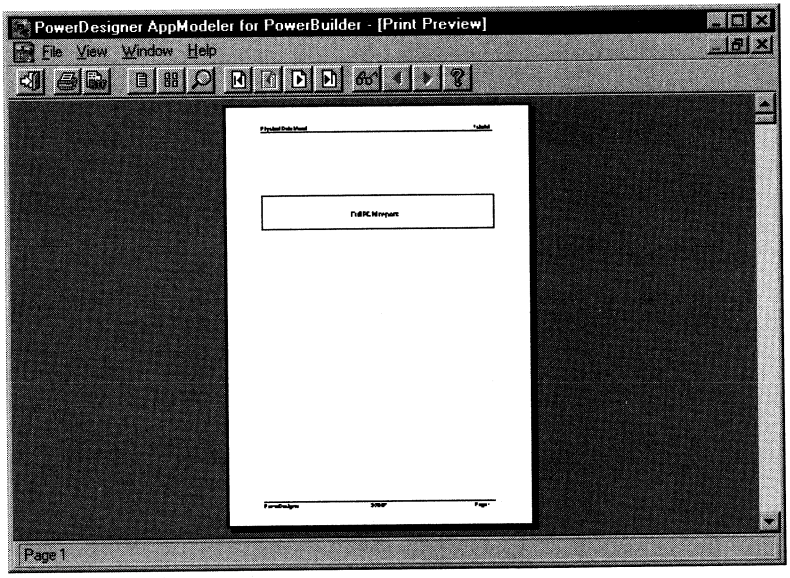
Opening print preview

You access print preview from the Report Editor.

❖ **To open print preview:**

- Select Report➤Print Preview.
  or
  Click the Print Preview tool.

The Print Preview window displays the first page of the report.



Quick view of an item
From the Report Editor, you can go directly to preview a particular item by selecting Quick View from its context menu.

Navigating in print preview

In print preview, you can use tools and menu items to navigate through the pages in the displayed report.

Tool	Menu item	Goes to preview
	View➤First Page	First page in report
	View➤Previous Page	Previous page in report
	View➤Next Page	Next page in report
	View➤Last Page	Last page in report
—	View➤Go To Page	Goes to a specific page number
	View➤Step Back	Previous page in preview
	View➤Step Forward	Next page in preview

❖ To navigate in print preview:

- Click a navigation tool in the toolbar.
  or
  Select View➤*Page* submenu.

Displaying an item in print preview

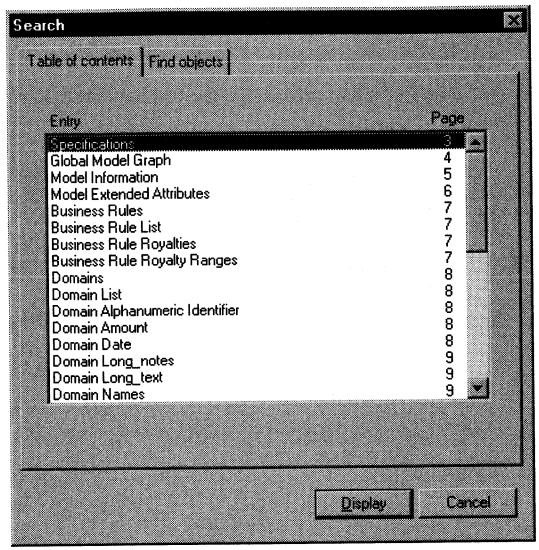
In print preview, you can go directly to any item listed in the table of contents of the current report.

❖ To display an item in print preview:

1 Select View➤Search.
or

Click the Search tool.

The Search dialog box opens to the Table of Contents page.



2 Select an item in the table of contents.

3 Click the Display button.

You return to the print preview window which displays the item you selected.

Finding an object in print preview

In print preview, you can find all references to an object. You can then go directly to any item that mentions the object in the current report.

❖ To find an object in print preview:

1 Select View➤Search.

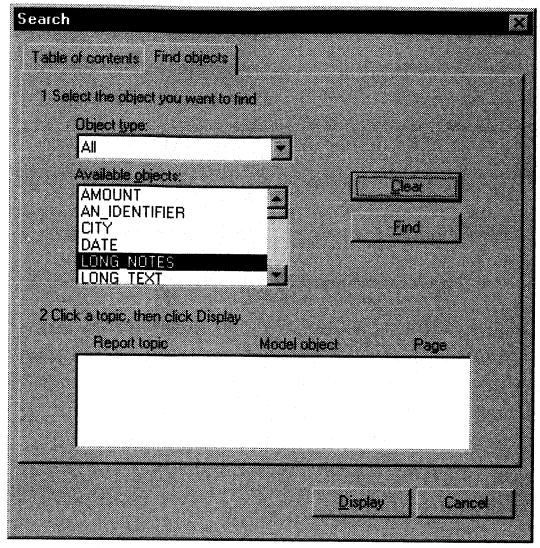
or

Click the Search tool.

The Search dialog box appears.

2 Click the Find Objects tab.

The Search dialog box opens to the Find Objects tab.



3 Select an object type from the Object Type dropdown listbox.

4 Select an object name from the Available Objects list.

5 Click the Find button.

The Report Topic box displays a list of items that mention the object.

6 Select an item in the list.

7 Click the Display button.

You return to the print preview window which displays the item you selected.

Zooming in print preview

You zoom in print preview by clicking inside the preview window. The cursor takes the form of a magnifying glass.

You toggle between the following two viewing scales:

Scale	Description
Whole page	Uses a scale that fits the entire page or multiple pages in the print preview window
Custom view	Displays the report in its actual size or in a scale you select

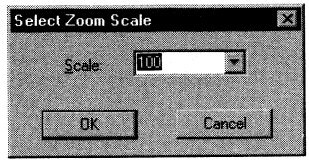
❖ **To zoom in print preview:**

- Click a page inside the preview window.

❖ **To select a custom view scale:**

1 Select View➢Zoom Scale.

The Select Zoom Scale dialog box appears.



2 Select a scale from the dropdown listbox.

3 Click OK.

Displaying multiple pages

If you toggle to Whole Page view, you can display multiple pages in the same view.

❖ **To display multiple pages:**

1 Position the cursor over the multiple page tool.

2 Press the left mouse button and drag the cursor down and to the right.

A multiple page grid appears.

3 Drag to select the number of pages you want to display.

4 Release the left mouse button.

Displaying one page at a time

You can display one page at a time.

❖ **To display one page at a time:**

♦ Click the one page tool.

Printing a report from print preview

❖ **To print a report from print preview:**

♦ Select File➢Print.
or
Click the Print tool.

Saving an RTF file from print preview

❖ **To save an RTF file from print preview:**

♦ Select File➢Save RTF.
or
Click the Save RTF tool.

Closing print preview

❖ **To close print preview:**

♦ Select File➢Close.
or
Click the Close tool.

You return to the Report Editor.

Defining a report language

You can choose to print a report in a limited number of languages. The language of a report applies to the following:

Item titles
Title boxes
Headers
Footers

When you save a report, you also save its choice of language.

Selecting the default report language

If the language property of a model is a language supported by the Report Editor, it is also the default report language.

If the language property of the model is not a language supported by the Report Editor, the default report language is English.

For example, the Report Editor supports French and does not support Finnish. For that reason, if the language property of a model is French, then French is the default report language. If the language property of a model is Finnish, then English is the default report language.

❖ **To select the language property for a model:**

1 Select Dictionary➢Model Properties.

The model property sheet appears.

2 Select a language from the language dropdown listbox.

3 Click OK.

Modifying the language of a report

In the Report Editor, you can modify the language of a report.

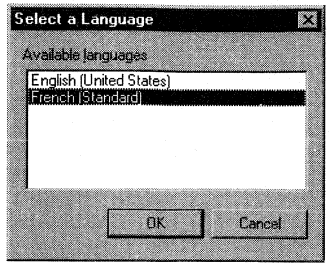
The language of a report applies to the following:

Item titles
Title boxes
Headers
Footers

❖ To modify the language of a report:

1 From the Report Editor, select Report➤Language.

The Select a Language dialog box appears.



2 Select one of the available languages.

3 Click OK.

The next time you print or preview the report, the selected language will apply to item titles, title boxes, headers, and footers.

Appendixes

The Appendixes give an overview of the DEF files and describe the extended attributes that you can generate or reverse engineer using the AppModeler 4GL interface.

APPENDIX A

DEF File Basics

About this chapter

This chapter describes the basics of consulting and using DEF files.

Contents

Customizing DEF files

A DEF file is a text file that contains specifications for a particular DBMS in a format understandable by PowerDesigner. The DEF file provides PowerDesigner with the syntax and guidelines for generating databases, triggers, and procedures appropriate for a particular DBMS or ODBC driver.

You can read a DEF file for details on DBMS implementation. You can also create your own DEF files to customize implementation. Avoid making changes to the standard DEF files installed with PowerDesigner.

> **DEF file modifications**
> If you modify a DEF file, you may damage its integrity and make it unusable. For this reason, before you modify a DEF file, you should copy and rename it. You can restore original DEF files by reinstalling PowerDesigner.

Creating a customized DEF file

You can create a customized DEF file, by copying, renaming, and modifying an existing DEF file.

❖ **To create a customized DEF file:**

1 Open a DEF file in the Notepad or another text editor.

2 Type a new value for the DBMS name.

```
DbmsName = Sybase SQL Server 11 (custom)
```

This name will appear in the list of target databases.

3 Type changes to the DEF file.

4 Select File➤Save As.

5 Type a new file name for the DEF file.

For example, for Sybase SQL Server 11 you could name the customized DEF file SY11CUST.DEF.

6 Click OK.

Modifying a customized DEF file

You should only modify customized DEF files which are renamed copies of DEF files delivered with PowerDesigner.

❖ **To modify a customized DEF file:**

1 Close any PDM that uses the customized DEF file.

2 Open the customized DEF file in the Notepad or another text editor.

3 Type changes to the DEF file.

4 Select File➤Save.

5 Select File➤Exit.

The next time you open any PDM that uses the customized DEF file, the PDM will take modifications into account. However, if you have previously modified the same options directly in the PDM, the values in the DEF file do not change these options.

DEF file syntax

All entries in the DEF file have the following syntax:

 # Comment
DEF option = Value

For example, the Sybase SQL Anywhere DEF file contains the following lines:

```
#  Constraint name template for Primary Keys
PKConstraintName = PK_%.U27:TABLE%
```

You must only modify values that follow the equals sign.

Formatting variables in DEF files

Variables have a syntax that can force a format on their values, as follows:

- Force values to lowercase or uppercase characters
- Truncate the length of values

When a variable does not specify formatting, its values have the same format as in the PDM.

Format code	Format of variable value in script
.L	Lower-case characters
.T	Removes blank spaces
.U	Upper-case characters
.*n*	Maximum length where *n* is the number of characters
.*n*J	Justifies to fixed length where *n* is the number of characters

You embed formatting options in variable syntax as follows:

%.*format*:*variable*%

You can combine format codes. For example, %.U8:CHILD% formats the code of the child table with a maximum of eight uppercase letters.

Defining constraint name templates

DEF files use variables and variable formatting to define constraint name templates. These templates define the format of default constraint names.

In PowerDesigner, you can define user-defined constraint names. Templates apply to all constraints for which you do not define user-defined constraint names.

DEF files define the following templates for constraint names:

DEF option	Description
PKConstraintName	Constraint name template of primary keys
FKConstraintName	Constraint name template for foreign keys
AKConstraintName	Constraint name template for alternate keys
CKCConstraintName	Constraint name template for column checks
CKTConstraintName	Constraint name template for table checks

PKConstraintName

PKConstraintName can accept the following variables:

Variable	Value
%OID%	Internal object identification number for the table
%TABLE%	Code of the table

FKConstraintName

FKConstraintName can accept the following variables:

Variable	Value
%OID%	Internal object identification number for the reference
%REFR%	Code of the reference
%NO%	Number of the reference in the order of references for the table
%PARENT%	Code of the parent table
%CHILD%	Code of the child table

AKConstraintName

AKConstraintName can accept the following variables:

Variable	Value
%OID%	Internal object identification number for the alternate key
%AKEY%	Code of the alternate key
%NO%	Number of the alternate key in the order of alternate keys for the table
%TABLE%	Code of the table

CKCConstraint Name

CKCConstraintName can accept the following variables:

Variable	Value
%OID%	Internal object identification number for the column
%COLN%	Code of the column
%NO%	Number of the column in the order of table columns
%TABLE%	Code of the table

CKTConstraint Name

CKTConstraintName can accept the following variables:

Variable	Value
%OID%	Internal object identification number for the table
%TABLE%	Code of the table

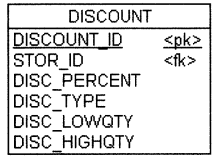
Example

The following example shows the use of constraint name templates for Sybase SQL Server 11.

SYSTEM11.DEF contains the following lines:

```
#  Constraint name template for Primary Keys
PKConstraintName = PK_%.U27:TABLE%

#  Constraint name template for Foreign Keys
FKConstraintName =
FK_%.U8:CHILD%_%.U9:REFR%_%.U8:PARENT%

#  Constraint name template for Alternate Keys
AKConstraintName = AK_%.U18:AKEY%_%.U8:TABLE%

#  Constraint name template for Check of Column
CKCConstraintName = CKC_%.U17:COLN%_%.U8:TABLE%

#  Constraint name template for Check of Table
CKTonstraintName = CKT_%.U26:TABLE%
```

You define the following table in a PDM:



The resulting script that you generate for Sybase SQL Server 11 declares constraint names for this table as follows:

```
create table DISCOUNT
(
    DISCOUNT_ID       T_IDENTIFIER           not null,
    STOR_ID           T_AN_IDENTIFIER        not null,
    DISC_PERCENT      T_PERCENT              not null,
    DISC_TYPE         T_SHORT_TEXT           null
        constraint CKC_DISC_TYPE_DISCOUNT check
(DISC_TYPE in ('High','Medium','Low')),
    DISC_LOWQTY       T_QUANTITY             null    ,
    DISC_HIGHQTY      T_QUANTITY             null    ,
    constraint PK_DISCOUNT primary key (DISCOUNT_ID)
)
go
```

Defining reserved keywords

For a domain or a column, standard check parameters can indicate minimum, maximum, and default values, as well as a lists of values.

In general, if the data type of domain or column is a string data type, quotation marks surround these values in the generated script. However, quotation marks are not generated in the following cases:

- You define a data type that is not recognized as a string data type by PowerDesigner
- Value is surrounded by tilde characters
- Value is a reserved keyword defined in the DEF file (for example, NULL)

In addition, if the value is already surrounded by quotation marks additional quotation marks are not generated.

The generation of single- or double-quotation marks depends on the target DBMS.

The DEF file lists reserved keywords under **ReservedDefault**.

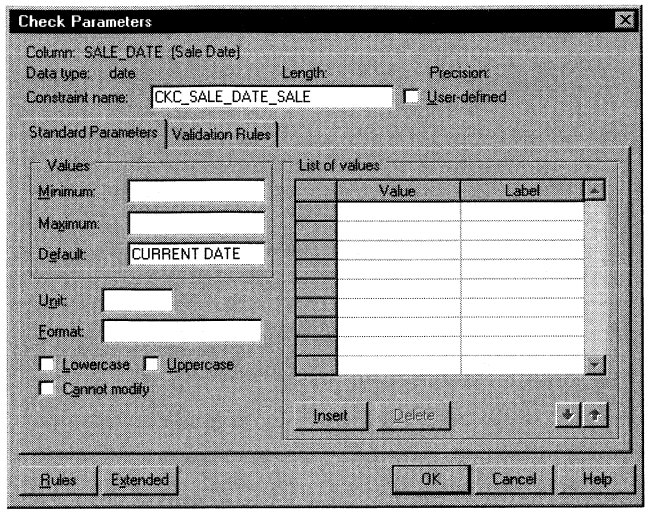
Example

The following example shows the result of a reserved keyword for Sybase SQL Anywhere.

SQLANY5.DEF contains the following lines:

```
ReservedDefault =
NULL
AUTOINCREMENT
CURRENT DATE
CURRENT PUBLISHER
CURRENT TIME
CURRENT TIMESTAMP
CURRENT USER
USER
```

You define CURRENT DATE as the default value of a column:



The resulting script contains the string CURRENT DATE without quotation marks.

```
create table SALE
(
    SALE_ID             T_IDENTIFIER            not null,
    STOR_ID             T_AN_IDENTIFIER         not null,
    TITLE_ISBN          char(12)                not null,
    SALE_DATE           T_DATE                  not null
        default CURRENT DATE,
    SALE_AMOUNT         T_AMOUNT                        ,
    SALE_TERMS          T_LONG_TEXT                     ,
    SALE_QTY            T_QUANTITY                      ,
    primary key (SALE_ID)
);
```

When you run this script, Sybase SQL Anywhere will recognize CURRENT DATE as a reserved default value.

Setting default script options for uppercase or lowercase characters

A DEF file can indicate whether a script is generated in uppercase or lowercase characters by default. The value in the DEF file only affects models for which you have not previously modified script options.

> **Script options for a PDM**
> You can consult and modify these options from a PDM by selecting Database➤Generation Options.

The following Boolean variables correspond to lowercase and uppercase options for a script:

DEF option	Script option in a PDM	Result of YES
UpperCaseOnly	Uppercase	Script contains all uppercase characters
LowerCaseOnly	Lowercase	Script contains all lowercase characters

Defining physical options

In a PDM, you can assign values to physical options for a database, tables, indexes, tablespace, and storage. In a DEF file, you can define physical options and their default values, using the following DEF options:

DEF option	Description
DefDatabaseOption	Default values for database options
DefTableOption	Default values for table options
DefIndexOption	Default values for index options
DefTablespaceOption	Default values for tablespace options
DefStorageOption	Default values for storage options

These options appear in the corresponding Options windows with the Use checkbox selected.

In a PDM, these options apply to all new objects and to all existing objects for which you have not previously modified the same options.

Example

The following example shows the generation of a default table option for Sybase System 11.

You create a customized DEF that defines table options on the following lines:

```
#  Default options for creating a table.
TableOption =
with max_rows_per_page = %d : default=0
on %s : category=storage

#  Default values for table options (customized)
DefTableOption = on my_segment
```

These options appear in the Options of the Table dialog box.

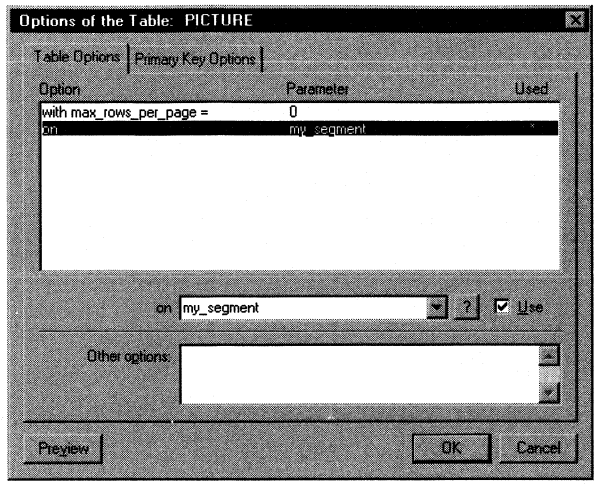


Table options in the PDM
You can consult or modify table options by clicking the Options button on the table property sheet.

The resulting script declares these options, as follows:

```
create table PICTURE
(
    PIX_ID          T_IDENTIFIER          not null,
    AU_ID           T_AN_IDENTIFIER       null     ,
    PIX_PICTURE     varbinary(255)        not null,
    constraint PK_PICTURE primary key (PIX_ID)
)
on my_segment
go
```

Consulting triggers, stored procedures, and functions

The DEF file contains the definition of templates for triggers, for stored procedures, and for functions.

You can consult and modify these templates directly in the DEF file. However, it is much easier to modify the templates from the PDM by selecting Dictionary➤Triggers and Procedures➤Trigger Templates and Trigger Template Items.

For more information on trigger templates, see the chapter "Triggers and Procedures."

APPENDIX B

Client Reference

About this appendix

This appendix describes extended attributes for different 4GL supported by PowerDesigner. Application generation attributes are described in the appendix, "Generation Variables and Attributes."

Contents

Progress extended attributes

This section lists all Progress extended attributes with their descriptions, data types, and default values. You import Progess extended attributes from PROGRESS.EXA. This file defines extended attributes for tables, columns, and indexes.

Progress table extended attributes

Extended attribute	Data type	Description and default
PG_ValMsg	String	Error message for table validation Default=Deletion forbidden in table %TableName%
PG_DumpName	String	Dump file name Default=%TableCode%

Progress column extended attributes

Extended attribute	Data type	Description and default
PG_HelpMsg	String	Help message when entering a field value Default=%ColumnName%
PG_ValMsg	String	Error message after column validation Default=Invalid value in field %ColumnName%
PG_ColumnLabel	String	Column label Default=%ColumnName%
PG_CaseSensitive	Boolean	Indicates if column is case-sensitive Default=FALSE
PG_Extent	Integer	Column size if it is declared in an array No default

Widget attributes

Widget attributes describe the appearance of a column on screen.

Extended attribute	Data type	Description and default
PG_WidgetType	PG_WidgetType	Screen representation of the column * No default
PG_Height	Integer	Field height (not applied to TEXT and TOGGLE-BOX widgets) No default
PG_Horizontal	Boolean	Horizontal alignment for RADIO-SET widget No default
PG_Max-Chars	Integer	Maximum number of field characters for EDITOR widget No default
PG_Max-Value	Integer	Maximum field value for SLIDER widget Default=%ColumnMaximum%
PG_Min-Value	Integer	Minimum field value for SLIDER widget Default=%ColumnMinimum%
PG_Multiple	Boolean	Multiple line selection for SELECTION-LIST widget Default=FALSE
PG_Native	Boolean	Activate NATIVE option in FILL-IN widget No default
PG_No-Drag	Boolean	Multiple line selection using drag for SELECTION-LIST widget Default=FALSE
PG_No-Word-Wrap	Boolean	No word wrap in EDITOR widget Default=FALSE
PG_Num-Chars	Integer	Number of visible characters per field for all widgets except TEXT and TOGGLE-BOX No default
PG_Num-Line	Integer	Number of visible characters per line for all widgets except TEXT and TOGGLE-BOX No default

Extended attribute	Data type	Description and default
PG_Scrollbar-Horizontal	Boolean	Horizontal scrollbar beneath field for EDITOR and SELECT-LIST widgets No default
PG_Scrollbar-Vertical	Boolean	Vertical scrollbar to the right of field for EDITOR and SELECT-LIST widgets No default
PG_SizeUnit	PG_SizeUnit	Unit of measurement for all widgets except TEXT and TOGGLE-BOX * No default
PG_Sort	Boolean	Alphabetical data sort in SELECTION-LIST widget No default
PG_Width	Integer	Field width for all widgets except TEXT and TOGGLE-BOX No default

* The list of accepted widget types appears in Value dropdown listbox. Certain widgets are only functional with particular data types.

Progress index extended attributes

Extended attribute	Data type	Description and default
PG_Active	Boolean	Indicates if the index is active Default=TRUE
PG_Word Index	Boolean	Indicates if the index is a Word Index Default=FALSE

Progress trigger extended attributes

Extended attribute	Data type	Description and default
PG_CreateTriggerLink	Boolean	Generate the create trigger link in the DF file Default=TRUE
PG_DeleteTriggerLink	Boolean	Generate the delete trigger link in the DF file Default=TRUE
PG_WriteTriggerLink	Boolean	Generate the write trigger link in the DF file Default=TRUE

Uniface extended attributes

This section lists all Uniface extended attributes with their descriptions, data types, and default values. You import Uniface extended attributes from UNIFACE.EXA. This file defines extended attributes for tables and columns.

Uniface table extended attributes

Entity definition

Extended attribute	Data type and code	Description and default
UN_DBMS	UN_DBMS #Z	Identification code (3 letters) of target database Default=%EntryDbms%
UN_InDatabase	Boolean #D	Table exists in database Default=TRUE
UN_NumberOfOccMax	Integer #P	Maximum number of occurrence of the table No default
UN_NumberOfOccMin	Integer #O	Minimum number of occurrence of the table No default
UN_Update	UN_Update #U	Table update (Yes, No, or Oprimistic locking) Default=Yes

Entity interface

Extended attribute	Data type and code	Description and default
UN_EntityFixedLength	Integer #I	Entity size in bytes No default
UN_FixedLengthOfOccurence	Integer #I	Size of entity occuerence in bytes No default

Uniface column extended attributes

Field definition

Extended attribute	Data type and code	Description and default
UN_DataType	UN_DataType #T	Generic Uniface data type * No default
UN_FieldDataPacking	UN_DataPacking #I	Uniface data packing code * No default
UN_InDatabase	UN_InDatabase #D	Field exists in database (Yes, No, or Boilerplate) Default=Yes

* The list of accepted values appears in Value dropdown listbox. You can display costants by clicking the Constants button.

Field syntax

Extended attribute	Data type and code	Description and default
UN_AllowedBold	Boolean #S	Allow bold characters TRUE=#S YBLD FALSE=#S NBLD
UN_AllowedItalic	Boolean #S	Allow italic characters TRUE=#S YITA FALSE=#S NITA
UN_AllowedUnderline	Boolean #S	Allow underlined characters TRUE=#S YUND FALSE=#S NUND
UN_AutoJump	Boolean #S	Auto jump Default=TRUE (#S JMP)
UN_BracketMatchCheck	Boolean #S	Check that brackets match TRUE=#S BRM
UN_CharactersAllowed	UN_CharactersAllowed #S	Characters allowed in field * No default
UN_DeleteControl	Boolean #S	Delete all control characters TRUE=#S YDCC
UN_DeleteLeadingControl	Boolean #S	Delete leading control characters TRUE=#S DLC
UN_DeleteLeadingSpaces	Boolean #S	Delete leading spaces TRUE=#S DLS
UN_DeleteLeadingZeros	Boolean #S	Delete leading zeros TRUE=#S DLZ

Extended attribute	Data type and code	Description and default
UN_DeleteReplaceSpaces	Boolean #S	Replace contiguous spaces with one space TRUE=#S RCS
UN_DeleteTextControl	Boolean #S	Delete all text control characters TRUE=#S YDCX
UN_DeleteTrailingControl	Boolean #S	Delete trailing control characters TRUE=#S DTC
UN_DisplayEditPrompt	UN_Edit #S	Edit/Display/Prompt * Default=display-edit
UN_EntryFormat	String #S	Entry field format Default=%Format%
UN_FieldLengthMax	Integer #S	Maximum field length No default
UN_FieldLengthMin	Integer #S	Minimum field length No default
UN_Overstrike	Boolean #S	Field in overstrike mode TRUE=#S OVS

* The list of accepted values appears in Value dropdown listbox. You can display costants by clicking the Constants button.

Field layout

Extended attribute	Data type and code	Description and default
UN_Alignment	UN_Alignment #L	Data alignment in field* Default=Left
UN_CursorVideo	Boolean #L	Activate video attributes No default
UN_VideoBlink	Boolean #L	Blinking data field TRUE=#L BLI FALSE=#L NBL
UN_VideoBorderLines	Boolean #L	Display field border TRUE=#L BOR FALSE=#L NBR
UN_VideoBright	Boolean #L	Data field bright TRUE=#L BRI FALSE=#L NBR

Extended attribute	Data type and code	Description and default
UN_VideoInverse	Boolean #L	Data field inverse video TRUE=#L INV FALSE=#L NIN
UN_VideoUnderline	Boolean #L	Data field underline TRUE=#L UND FALSE=#L NUN

* The list of accepted values appears in Value dropdown listbox. You can display costants by clicking the Constants button.

PowerHouse and Axiant extended attributes

This section lists all PowerHouse and Axiant extended attributes with their descriptions, data types, and default values. You import PowerHouse and Axiant extended attributes from COGNOS.EXA. This file defines extended attributes for the model and for columns.

PowerHouse and Axiant model extended attributes

Extended attribute	Data type	Description and default
PH_Database	String	Database name (and options) No default
PH_Dictionay	String	Dictionary file name (PDC file) No default
PH_SystemOptions	String	System options Default=%+%ASCII8 For a list of options, see PowerHouse and Axiant's *PDL and Utilities Reference Manual.*

PowerHouse and Axiant column extended attributes

Extended attribute	Data type	Description and default
PH_Character	String	Format when the data type is CHARACTER for ELEMENTS and USAGES Default=%Shift% %Picture%
PH_Date	String	Format when the data type is DATE for ELEMENTS and USAGES * Default=%Format%
PH_Description	Boolean	Generate column description for ELEMENT Default=YES
PH_Heading	String	Field header for ELEMENTS Default=%ColumnName%
PH_Help	String	Field help for ELEMENTS Default=%ColumnLabel%

Extended attribute	Data type	Description and default
PH_Item	String	Additional data type information for for ELEMENTS and USAGES * No default **Recommended use** PowerDesignor generates PH_Item directly after PH_Label. Insert %+% before the attribute value.
PH_Label	String	Field lable for ELEMENTS Default=%ColumnLabel%
PH_ListValues	Boolean	Declare list of values for ELEMENTS and USAGES Default=YES
PH_Numeric	String	Format when the data type is NUMERIC for ELEMENTS and USAGES Default=%Picture%
PH_Size	Integer	Field size for ELEMENTS and USAGES Default=%Size%
PH_Type	PH_Type	Data type for ELEMENTS and USAGES * Default=%Type%
PH_Usage	String	USAGE used Default=%DomainCode%
PH_Value	String	Initial, minimum, and maximum values for ELEMENTS and USAGES Default=%Initial% %MinMax%

* The list of accepted values appears in Value dropdown listbox. You can display constants by clicking the Constants button.

NS-Access extended attributes

This section lists all NS-Access extended attributes with their descriptions, data types, and default values. You import NS-Access extended attributes from NSACCESS.EXA. This file defines extended attributes for the model and for columns.

NS-Access model extended attributes

Target database (DB target)

Extended attribute	Data type	Description and default
NS_DBName	String	Name of target database No default
NS_DBDescription	String	Description of target database No default
NS_DBDatabase	NS_DBDatabase	NS-Access code for target database No default

* The list of accepted values appears in Value dropdown listbox. You can display constants by clicking the Constants button.

User interface (UI target)

Extended attribute	Data type	Description and default
NS_UIName	String	Name of target user interface No default
NS_UIDescription	String	Description of target user interface No default
NS_UIUserInterface	NS_UIUserInterface	NS-Access code for target user interface No default

* The list of accepted values appears in Value dropdown listbox. You can display constants by clicking the Constants button.

NS-Access column extended attributes

Check type

Extended attribute	Data type	Description and default
NS_Control	NS_Control*	Screen representation of the column No default
NS_DataType	NS_DataType*	Authorized data type in the check field No default
NS_Text	String	Field label appearing on the screen, corresponds to the TEXT variable in the NS-Access COLUMN creation statement For more information, see *NS-Access Class Definitions*. Default=%ColumnName%

* The list of accepted values appears in Value dropdown listbox. You can display costants by clicking the Constants button.

Check properties

Extended attribute	Data type	Description and default
NS_Characters	NS_Characters*	Legal characters for ENTRY and COMBBOXE No default
NS_ColorBackground	NS_Color*	Background color of the check field, except for VERTSCROLLBAR and HORZSCROLLBAR Default=default
NS_ColorForeground	NS_Color*	Foreground color of the check field, except for VERTSCROLLBAR and HORZSCROLLBAR Default=default
NS_Font	Font	Font of the check field, except for VERTSCROLLBAR and HORZSCROLLBAR No default
NS_ForceFill	Boolean	Fill field up to number of characters specified by NS_Length for ENTRY and COMBBOXE Default=FALSE
NS_Height	Integer	Check field height for ENTRY, LISTBOX, COMBBOX, and COMBBOXE No default

Extended attribute	Data type	Description and default
NS_HideText	Boolean	Characters typed in field are not visible for ENTRY and COMBBOXE Default=FALSE
NS_Justify	NS_Justify*	Justificiation in field for ENTRY and COMBBOXE Default=Left
NS_Length	Integer	Maximum number of characters in a field for ENTRY and COMBBOXE Default=31
NS_Margin	Boolean	Field frame on screen for ENTRY and COMBBOXE Default=FALSE
NS_MultiSel	Boolean	Allow multiple line selection for LISTBOX Default=TRUE
NS_Shadow	NS_Shadow*	Field shadow, for all except CHECKBOX, VERTSCROLLBAR, and HORZSCROLLBAR No default
NS_Margin	Boolean	Field frame on screen for ENTRY and COMBBOXE Default=FALSE
NS_SkipBlank	Boolean	Automatically delete leading and trailing blanks for ENTRY and COMBBOXE Default=FALSE
NS_SpinButton	Boolean	Field with a spin button for ENTRY and COMBBOXE Default=FALSE
NS_Width	Integer	Field length in pixels for all except RADIO Default=FALSE
NS_WordWrap	Boolean	If length exceeds NS_Width, wrap to next line to avoid truncation of MLE Default=FALSE

* The list of accepted values appears in Value dropdown listbox. You can display costants by clicking the Constants button.

APPENDIX C

Generation Variables and Attributes

About this appendix

This appendix lists variables and keywords recognized by AppModeler 4GL application generators. It also lists the extended attribute default values and definitions for 4GL object generation.

Contents

Before you begin

Variables specific to PBGen are included in the chapter "PowerBuilder Generator."

Common AppModeler system variables

Certain system variables are available with all AppModeler generators.

> **Entering variables in extended attribute definitions**
> When defining extended attribute values, you can enter standard variables, enclosed between percent signs, that correspond to PDM object identifiers. AppModeler replaces these variables with the object values.

Model-related variables

When you generate a project, AppModeler instantiates the following variables with values displayed in the Model Properties dialog box.

%ModelCode%
%ModelCreateDate%
%ModelCreateTime%
%ModelFile%
%ModelGenDate%
%ModelGenTime%
%ModelLabel%
%ModelModifyDate%
%ModelModifyTime%
%ModelName%
%ModelUser%
%ModelVersion%

Object-related variables

Other variables serve as placeholders for PDM object identifiers. You can use these variables with any of the AppModeler generators:

%ColumnCode%
%ColumnLabel%
%ColumnName%
%ObjectCode%
%ObjectName%
%ReferenceCode%
%ReferenceName%
%TableCode%
%TableName%
%ViewCode%
%ViewName%

> **Exception for reference variables**
> Reference variables are not instantiated in projects that you generate with AppModeler for Dynamo. WebGen generates references as links between the pages that it generates for parent and child tables.

PBGen generation attributes

This section lists PBGen generation attributes with their descriptions, data types, and default values.

Model extended attributes

Application extended attributes

Extended attribute	Data type	Description and default
PB_ApplicationLibrary	PB_Library	Application library No default
PB_ApplicationMenuName	String	Application menu name Default=m_%ModelCode%
PB_ApplicationMenu Processing	Boolean	Application menu processing Default=True
PB_ApplicationMenu Template	PBM_Template	Application menu template No default
PB_ApplicationName	String	Application name Default= a_%ModelCode%
PB_ApplicationTemplate	PBA_Template	Application template No default
PB_ApplicationTitle	String	Application title Default=%ModelName%

Additional library and menu extended attributes

Extended attribute	Data type	Description and default
PB_DataWindowLibrary	PB_Library	DataWindow library No default
PB_DefaultMenuPosition	Integer	Window access menu position Default=1
PB_MenuLibrary	PB_Library	Menu library No default
PB_MenuScript	String	Menu script code Default=ue_openwindow
PB_TemplateLibrary	PB_Library	Template library No default

Database profile extended attributes

Extended attribute	Data type	Description and default
PB_DatabaseProfile	PB_Profile	Database profile No default
PB_DatabaseProfileINI	File	PowerBuilder INI file No default
PB_UserDBProfileKeyReg	String	User database profile registry key (or INI file) Default=%ModelCode%
PB_UserDBProfileSubkey	String	User database profile registry subkey (or INI section) Default=Profile %db_profile%

Generation extended attributes

Extended attribute	Data type	Description and default
PB_GenerateApplication	Boolean	Generate application flag Default=True
PB_GenerateMainWindow	Boolean	Generate main window flag Default=True
PB_GenerateMenus	Integer	Menu generation mode (None=0, Generate=1, Update=2) Default=1
PB_GenerationMode	Integer	Generation mode Default=28*

* The PB_Generation Mode attribute has a 5-bit binary value. Bit 4 indicates whether to generate PDM references; Bit 3 indicates whether to generate PDM views; Bit 2 indicates whether to generate PDM tables; and Bits 0 and 1 indicate the type of filter used: 0=all, 1=submodel, 2=archive differences, 3=custom filter. The default value is 11100, which equals 28 (in base 10).

Window extended attributes

Extended attribute	Data type	Description and default
PB_MainWindowName	String	Main window name Default=w_%ModelCode%
PB_MainWindowTemplate	PBW_Template	Main window template No default
PB_WindowLibrary	PB_Library	Window library No default
PB_WindowMenuName	String	Window menu name Default= %MainMenu%_sheet
PB_WindowMenu Processing	Boolean	Window menu processing Default=True
PB_WindowMenuTemplate	PBM_Template	Window menu template No default

Table extended attributes

DataWindow extended attributes

Extended attribute	Data type	Description and default
PB_DataWindowLibrary	PB_Library	DataWindow target library Default=%DataWindowLibrary%
PB_DataWindowName	DataWin	DataWindow name Default=dt%TableCode%
PB_DataWindowStyle	PBDW_Style	DataWindow style Default=Form
PB_DataWindowUpdate	Boolean	DataWindow updatable Default=True

Generation extended attributes

Extended attribute	Data type	Description and default
PB_GenerateDataWindow	Boolean	Generate DataWindow flag Default=True
PB_GenerateMenuOption	Boolean	Generate menu option Default=True
PB_GenerateWindow	Boolean	Generate window flag Default=True

Menu and window extended attributes

Extended attribute	Data type	Description and default
PB_MenuOption	String	Menu option Default=%TableName%
PB_SelectWindow	Boolean	Select window Default=True
PB_TemplateType	Integer	Template type Default=128 (window type)
PB_WindowLibrary	PB_Library	Window target library Default=%WindowLibrary%
PB_WindowName	String	Window name Default=w_%TableCode%
PB_WindowTemplate	PBW_Template	Window template Default=%w_sys_single_dw%_
PB_WindowTitle	String	Window title Default=%TableName%

View extended attributes

DataWindow extended attributes

Extended attribute	Data type	Description and default
PB_DataWindowLibrary	PB_Library	DataWindow target library Default=%DataWindowLibrary%
PB_DataWindowName	DataWin	DataWindow name Default=dt%ViewCode%
PB_DataWindowStyle	PBDW_Style	DataWindow style Default=Grid
PB_DataWindowUpdate	Boolean	DataWindow updatable Default=True

Generation extended attributes

Extended attribute	Data type	Description and default
PB_GenerateDataWindow	Boolean	Generate DataWindow flag Default=True
PB_GenerateMenuOption	Boolean	Generate menu option Default=True
PB_GenerateWindow	Boolean	Generate window flag Default=True

Menu and window extended attributes

Extended attribute	Data type	Description and default
PB_MenuOption	String	Menu option Default=%ViewName%
PB_SelectWindow	Boolean	Select window Default=True
PB_TemplateType	Integer	Template type Default=128 (window type)
PB_WindowLibrary	PB_Library	Window target library Default=%WindowLibrary%
PB_WindowName	String	Window name Default=w_%ViewCode%
PB_WindowTemplate	PBW_Template	Window template Default=%w_sys_multi_dw%_
PB_WindowTitle	String	Window title Default=%ViewName%

Reference extended attributes

Master extended attributes

Extended attribute	Data type	Description and default
PB_MasterDataSource	Object	Master data source Default=0
PB_MasterDataWindow Library	PB_Library	DataWindow target library Default=%DataWindowLibrary%
PB_MasterDataWindow Name	DataWin	Master DataWindow name Default=dm%ReferenceCode%
PB_MasterDataWindow Style	PBDW_Style	Master DataWindow style Default=Form
PB_MasterDataWindow Update	Boolean	Master DataWindow updatable Default=True
PB_MasterGenerateData Window	Boolean	Generate master DataWindow Default=False

Detail extended attributes

Extended attribute	Data type	Description and default
PB_DetailDataSource	Object	Detail data source Default=0
PB_DetailDataWindow Library	PB_Library	DataWindow target library Default=%DataWindowLibrary%
PB_DetailDataWindow Name	DataWin	Detail DataWindow name Default=dt%ReferenceCode%
PB_DetailDataWindow Style	PBDW_Style	Detail DataWindow style Default=Grid
PB_DetailDataWindow Update	Boolean	Detail DataWindow updatable Default=True
PB_DetailGenerateData Window	Boolean	Generate detail DataWindow Default=True

Generation extended attributes

Extended attribute	Data type	Description and default
PB_GenerateMenuOption	Boolean	Generate menu option Default=True
PB_GenerateWindow	Boolean	Generate window flag Default=True

Menu and window extended attributes

Extended attribute	Data type	Description and default
PB_MenuOption	String	Menu option Default=%ReferenceName%
PB_SelectWindow	Boolean	Select window Default=True
PB_TemplateType	Integer	Template type Default=128 (window type)
PB_WindowLibrary	PB_Library	Window target library Default=%WindowLibrary%
PB_WindowName	String	Window name Default=w_%ReferenceCode%
PB_WindowTemplate	PBW_Template	Window template Default=%w_sys_mast_detl_dw%_
PB_WindowTitle	String	Window title Default=%ReferenceName%

Column extended attributes

Extended attribute	Data type	Description and default
PB_GenDetail	Boolean	Generate column in detail DataWindow Default=True
PB_GenMaster	Boolean	Generate column in master DataWindow Default=True
PB_GenSingle	Boolean	Generate column in DataWindow Default=True
PB_SkipColumn	Boolean	Temporary flag that is replaced by PB_GenDetail, PB_GenMaster, or PB_GenSingle column generation value Default=False

PowerBuilder repository attributes

When you generate an application or extended attributes, you transfer the values of catalog attributes to columns in PowerBuilder repository tables.

The following repository tables store values for PBGen catalog attributes:

Repository table	Contains description of
PBCatTbl	Table catalog attributes
PBCatCol	Column catalog attributes
PBCatFmt	Column display formats
PBCatVld	Column validation rules

Repository table PBCatTbl

Columns in the repository table PBCatTbl correspond to catalog attributes attached to tables.

PBCatTbl column	Catalog attribute	Description
Pbt_tnam	Table code	Table name
Pbt_cmnt	PB_Comment	Comment
Pbd_fhgt Pbd_fwgt Pbd_fitl Pbd_funl Pbd_ffce	PB_DataFont	Data font, size, and style
Pbh_fhgt Pbh_fwgt Pbh_fitl Pbh_funl Pbh_ffce	PB_HeaderFont	Header font, size, and style
Pbl_fhgt Pbl_fwgt Pbl_fitl Pbl_funl Pbl_ffce	PB_LabelFont	Description font, size, and style

Repository table PBCatCol

Columns in the repository table PBCatCol correspond to catalog attributes attached to columns.

PBCatCol column	Catalog attribute	Description
Pbc_tnam	Table code	Table name
Pbc_cnam	Column code	Column name
Pbc_labl	PB_Label	Label
Pbc_hdr	PB_Header	Header
Pbc_jtfy	PB_DataJustify	Data justification
Pbc_hpos	PB_HeaderJustify	Header justification
Pbc_lpos	PB_LabelJustify	Label justification
Pbc_case	Upper/Lower check parameters options	Character case
Pbc_hght	PB_Height	Height
Pbc_wdth	PB_Width	Length
Pbc_bmap	PB_BitMap	Picture
Pbc_init	Default value in check parameters	Initial value
Pbc_mask	PB_FormatCode	Format
Pbc_ptrn	PB_ValidCode	Validation rule
Pbc_edit	PB_EditCode	Edit style
Pbc_cmnt	PB_Comment	Comments

Repository table PBCatFmt

Columns in the repository table PBCatFmt correspond to display formats attached to columns.

PBCatFmt column	Catalog attribute	Description
Pbf_name	PB_FormatCode	Format code
Pbf_frmt	PB_FormatText	Format value

Repository table PBCatVld

Columns in the repository table PBCatVld describe validation rules attached to columns.

PBCatVld column	Catalog attribute	Description
Pbv_name	PB_ValidCode	Validation rule code
Pbv_vald	PB_ValidText	Validation rule value
Pbv_msg	PB_ValidMsg	Error message

Catalog attribute default values

Default values for catalog attributes are automatically loaded into a new PDM from the POWERBLD.EXA, POWERPFC.EXA, or POWERPP.EXA files. These EXA files have different values for several extended attributes concerning application generation, but not for catalog attributes.

Table extended attributes

Extended attribute	Data type	Description and default
PB_DataFont	Font	Table data font Default=Arial, 10, N
PB_HeaderFont	Font	Table header font Default=Arial, 10, B
PB_LabelFont	Font	Table label font Default=Arial, 10, I
PB_Comment	String	Table comment Default=%TableLabel%

Column extended attributes

Extended attribute	Data type	Description and default
PB_BitMap	Boolean	Indicates if column is an image Default=False
PB_Height	Integer	Column height No default
PB_Width	Integer	Column width No default
PB_DataJustify	Justify	Column data justification No default
PB_HeaderJustify	Justify	Column header justification No default
PB_LabelJustify	Justify	Column label justification No default
PB_EditCode	PB_Edit	Column edit style* No default

Extended attribute	Data type	Description and default
PB_FormatCode	PB_Format	Column format code No default
PB_FormatText	String	Column format value Default=%Format
PB_ValidCode	PB_Valid	Column validation rule code# No default
PB_ValidText	String	Column validation rule§ Default=@column between @min and @max
PB_ValidMsg	String	Error message for column validation rule No default
PB_Header	String	Column header Default=%ColumnName%
PB_Comment	String	Column comments Default=%ColumnLabel%
PB_Label	String	Column label Default=%ColumnName%

* Checks for this attribute verify code existence in the PowerBuilder dictionary and its coherence with the column data type. If the code does not exist, it creates the new format in the dictionary with the code PB_FormatCode and the value PB_FormatText.

Checks for this attribute verify code existence in the PowerBuilder dictionary and its coherence with the column data type. If the code does not exist, it creates a new rule in the dictionary with the code PB_ValidCode, the value PB_ValidText, and error message given by PB_ValidMsg.

§ Generated Rule: `@col=>Min and @col<=Max` if the minimum and maximum value of the column check parameters are defined. `@col=>Min` if only the minimum value of the column check parameters is defined. `@col<=Max` if only the maximum value of the column check parameters is defined.

VBGen variables and keywords

This section lists variables and keywords recognized by the AppModeler for Visual Basic application generator. VBGen variables complement the system variables listed in this appendix.

VBGen project variables

The following variables are used as placeholders for project identifiers:

Variable	PowerDesigner value
%ProjectTitle%	Project title
%ProjectName%	Project name
%Connect%	Database type
%Database%	Database name
%OdbcDSN%	ODBC data source name
%OdbcUSR%	ODBC user name
%OdbcPWD%	ODBC password

VBGen form variables

The following variables are used as placeholders for form identifiers and for database connection parameters:

Variable	PowerDesigner value
%FormName%	Form name
%FormTitle%	Form title
%MasterTableName%	Master table name
%MasterTableCode%	Master table code
%MasterViewName%	Master view name
%MasterViewCode%	Master view code
%MasterObjectName%	Master table or view name
%MasterObjectCode%	Master table or view code

Variable	PowerDesigner value
%DetailTableName%	Detail table name
%DetailTableCode%	Detail table code
%DetailViewName%	Detail view name
%DetailViewCode%	Detail view code
%DetailObjectName%	Detail table or view name
%DetailObjectCode%	Detail table or view code
%DataSource%	dsMain, dsMaster, or dsDetail
%ReadOnly%	Is the dsMain data source read-only: True or False
%MasterReadOnly%	Is dsMaster read-only: True or False
%DetailReadOnly%	Is dsDetail read-only: True or False
%EOFAction%	Action for dsMain end of file*
%MasterEOFAction%	Action for dsMaster end of file*
%DetailEOFAction%	Action for dsDetail end of file*
%FRX%	Path of FRX file, determined by VBGen

* If form data source is read-only, end of file action is Move Last. Otherwise, it is Add new.

VBGen keywords

These are reserved words used by VBGen templates. Replace the variable part of the keyword (*Form*) by Main, Master, or Detail depending on the form for which the keyword is used.

Keyword	Name for
cnt*Form*	Container, if used in the form*
ds*Form*	Data source for the form
grd*Form*	Grid control, if used in the form

* If cnt*Form* is included in a form template, the form template controls are generated inside the container.

VBGen field variables

General variables

Variable	PowerDesigner value
%ColumnLength%	Maximum number of characters
%ColumnFormat%	Column data mask
%ColumnType%	Column data type
%DataField%	Bound control identifier
%DataSource%	dsMain, dsMaster, or dsDetail
%DefaultValue%	Column default value
%FieldCode%	Field code
%GridName%	grdMain, grdMaster, or grdDetail
%ListValues%	Number of elements (Null through 'n')
%ListNumber%	Number of elements (0 through 'n')

List variables

Variable	PowerDesigner value
%ListData%	Data value
%ListDisplay%	Display value
%ListIndex0%	Zero-based index: 0 to (%ListNumber%-1)
%ListIndex1%	One-based index: 1 to %ListNumber%
%ListLeft%	Current *x* coordinate on screen
%ListTop%	Current *y* coordinate on screen
%ListWidth%	Current *dx* coordinate on screen
%ListHeight%	Current *dy* coordinate on screen
%ListCol%	Number of columns for radio button controls
%ListRow%	Number of rows for radio button controls
%ListHHeight%	Height of radio buttons with loop on many columns
%ListHWidth%	Width of radio buttons with loop on many columns
%ListVHeight%	Height of radio buttons with loop on many rows
%ListVWidth%	Width of radio buttons with loop on many rows

VBGen extended attributes

This section lists extended attribute that are necessary for Visual Basic object generation. These attributes and their default values are loaded automatically into your PDM from the VB#.EXA file.

The EXA file includes additional extended attributes that are specific to certain templates in order to restrict their values to a predefined set of constants. You can view the complete list of extended attributes from the Dictionary➤Extended Attributes➤List of Attributes menu in AppModeler.

Model extended attributes

Connection attributes

Attribute	Type	Description and default value
VB_Connect	String	Database connection Default=Access
VB_ConnectionMode	String	Connection mode Default=Jet
VB_Database	String	Database full filename
VB_DataSourceName	String	ODBC datasource name
VB_Password	String	ODBC password
VB_UseDataControl	Boolean	Use standard data control Default=True
VB_UserName	String	ODBC user name

Filter attributes

Attribute	Type	Description and default value
VB_FilterArchiveModel	String	Archived model path
VB_FilterMode	Integer	Model type (0=All, 1=Submodel, 2=Archived model, 3=User-defined) Default=0
VB_FilterObjectType	Integer	Object type binary flags (1=Table, 2=View, 4=Reference) Default=7*
VB_FilterSubModel	String	Submodel

* Table + View + Reference = 1 + 2 + 4 = 7

Project Identifier attributes

Attribute	Type	Description and default value
VB_FormName	String	Main form name Default = %ModelCode%
VB_FormTemplate	String	Main form template Default = Std MDI Frame
VB_ProjectDirectory	String	Project directory Default=Project Directory
VB_ProjectName	String	Project name Default=%ModelCode%
VB_ProjectTemplate	String	Project template Default= Std Project
VB_ProjectTitle	String	Project title Default=%ModelName%
VB_TemplateSet	String	Name of project template set Default=Determined at installation

Project generation attributes

Attribute	Type	Description and default value
VB_ComputedFieldsTemplate	String	Computed Fields Template Default=Std Computed Field
VB_GenerateProject	Boolean	Flag for project generation Default=True
VB_GenerateForm	Boolean	Flag for main form generation Default=True
VB_MenuPosition	Integer	Menu bar position for generated forms Default=3
VB_UpdateMenu	Boolean	Flag for generating menu update Default=True

Table extended attributes

Form identifier attributes

Attribute	Type	Description and default value
VB_FormFileName	String	Form filename Default=%TableCode%.FRM
VB_FormName	String	Form name Default=%TableCode%
VB_FormTemplate	String	Form template Default=Std Single Table
VB_FormTitle	String	Form title Default=%TableName%

Form generation attributes

Attribute	Type	Description and default value
VB_FormSelected	Boolean	Flag for inclusion in tree view Default=True
VB_GenerateForm	Boolean	Flag for form generation Default=True
VB_GenerateMenu	Boolean	Flag for menu generation Default=True
VB_MenuOption	String	Menu name for generated form Default=&%TableName%...
VB_ReadOnly	Boolean	Flag for read-only property Default=False

Form display attributes

Attribute	Type	Description and default value
VB_FormStyle	String	Form style Default=Free form
VB_GridTemplate	String	Grid template for form Default=Std Grid (Apex)
VB_LineSpacing	Integer	Line spacing in Visual Basic units Default=0
VB_XOffset	Integer	X offset (Left property in Visual Basic) Default=0
VB_YOffset	Integer	Y offset (Top property in Visual Basic) Default=0

View extended attributes

Form identifier attributes

Attribute	Type	Description and default value
VB_FormFileName	String	Form filename Default=%ViewCode%.FRM
VB_FormName	String	Form name Default=%ViewCode%
VB_FormTemplate	String	Form template Default=Std Single View
VB_FormTitle	String	Form title Default=%ViewName%

Form generation attributes

Attribute	Type	Description and default value
VB_FormSelected	Boolean	Flag for inclusion in tree view Default=True
VB_GenerateForm	Boolean	Flag for form generation Default=True
VB_GenerateMenu	Boolean	Flag for menu generation Default=True
VB_MenuOption	String	Menu name for generated form Default=&%ViewName%...
VB_ReadOnly	Boolean	Flag for read-only property Default=True

Form display attributes

Attribute	Type	Description and default value
VB_FormStyle	String	Form style Default=Grid
VB_GridTemplate	String	Grid template for form Default=Std Grid (Apex)
VB_LineSpacing	Integer	Line spacing in Visual Basic units Default=0
VB_XOffset	Integer	X offset (Left property in Visual Basic) Default=0
VB_YOffset	Integer	Y offset (Top property in Visual Basic) Default=0

Reference extended attributes

Form generation attributes

Attribute	Type	Description and default value
VB_DetailView	Object	Detail view Default=0
VB_MasterView	Object	Master view Default=0
VB_FormSelected	Boolean	Flag for inclusion in tree view Default=True
VB_GenerateForm	Boolean	Flag for form generation Default=True
VB_GenerateMenu	Boolean	Flag for menu generation Default=True
VB_MenuOption	String	Menu name for generated form Default=&%ReferenceName%...

Form identifier attributes

Attribute	Type	Description and default value
VB_FormFileName	String	Form filename Default=%ReferenceCode%.FRM
VB_FormName	String	Form name Default=%ReferenceCode%
VB_FormTemplate	String	Form template Default=Master/Detail
VB_FormTitle	String	Form title Default=%ReferenceName%

Master/Detail display attributes

Attribute	Type	Description and default value
VB_DetailStyle	String	Detail style Default=Grid
VB_DetailGridTemplate	String	Grid template for detail fields Default=Grid
VB_DetailReadOnly	Boolean	Flag for dsDetail read-only property Default=False
VB_MasterStyle	String	Master style Default=Free form
VB_MasterGridTemplate	String	Grid template for master fields Default=Std Grid (Apex)

Attribute	Type	Description and default value
VB_MasterReadOnly	Boolean	Flag for dsMaster read-only property Default=False
VB_LineSpacing	Integer	Line spacing in Visual Basic units Default=0
VB_XOffset	Integer	X offset (Left property in Visual Basic) Default=0
VB_YOffset	Integer	Y offset (Top property in Visual Basic) Default=0

Column extended attributes

Most of the column extended attributes in the EXA file are specific to Visual Basic field templates. Only those column extended attributes that are global in scope are listed below.

Field attributes

Attribute	Type	Description and default value
VB_FieldStyle	String	Field style Default=Edit
VB_FieldTemplate	String	Field template Default=(Default)
VB_Height	Integer	Column height Default=240
VB_Width	Integer	Column width Default=1000
VB_Left	Integer	Left value of first control

Group box attributes

Attribute	Type	Description
VB_colLeft	Integer	Distance from left edge of column to frame
VB_colWidth	Integer	Width of column
VB_fraHeight	Integer	Height of frame
VB_fraLeft	Integer	Distance from left edge of frame to form
VB_fraWidth	Integer	Width of frame
VB_rowHeight	Integer	Height of row
VB_rowTop	Integer	Distance from top of row to frame

P++Gen variables and keywords

This section lists variables and keywords recognized by AppModeler for Power++ (P++Gen). P++Gen variables complement the system variables listed in this appendix on page 724.

P++Gen project variables

The following variables are used as placeholders for project identifiers:

Variable	PowerDesigner value
%ProjectTitle%	Project title
%ProjectName%	Project name
%Database%	Database name
%OdbcDSN%	ODBC data source name
%OdbcUSR%	ODBC user name
%OdbcPWD%	ODBC password

P++Gen form variables

The following variables are used as placeholders for form identifiers and for database connection parameters:

Variable	PowerDesigner value
%FormName%	Form name
%FormTitle%	Form title
%DataWindowLibrary%	DataWindow library
%DataWindowName%	DataWindow name
%MasterTableName%	Master table name
%MasterTableCode%	Master table code
%MasterViewName%	Master view name
%MasterViewCode%	Master view code
%MasterObjectName%	Master table or view name
%MasterObjectCode%	Master table or view code

Variable	PowerDesigner value
%DetailTableName%	Detail table name
%DetailTableCode%	Detail table code
%DetailViewName%	Detail view name
%DetailViewCode%	Detail view code
%DetailObjectName%	Detail table or view name
%DetailObjectCode%	Detail table or view code
%DataSource%	dsMain, dsMaster, or dsDetail

P++Gen keywords

These are reserved words used by P++Gen templates.

Keyword	Name for
dsSingle	Data source used by the single form
dsMaster	Data source used by the master form
dsDetail	Data source used by the detail form

P++Gen field variables

General variables

Variable	PowerDesigner value
%ColumnLength%	Maximum number of characters
%ColumnFormat%	Column data mask
%ColumnType%	Column data type
%DataField%	Bound control identifier
%DataSource%	dsSingle, dsMaster, or dsDetail
%DefaultValue%	Column default value
%FieldCode%	Field code
%ListValues%	Number of elements (Null through 'n')
%ListNumber%	Number of elements (0 through 'n')

List variables

Variable	PowerDesigner value
%ListData%	Data value
%ListDisplay%	Display value
%ListIndex0%	Zero-based index: 0 to (%ListNumber%-1)
%ListIndex1%	One-based index: 1 to %ListNumber%
%ListLeft%	Current *x* coordinate on screen
%ListTop%	Current *y* coordinate on screen
%ListWidth%	Current *dx* coordinate on screen
%ListHeight%	Current *dy* coordinate on screen
%ListCol%	Number of columns for radio button controls
%ListRow%	Number of rows for radio button controls
%ListHHeight%	Height of radio buttons with loop on many columns
%ListHWidth%	Width of radio buttons with loop on many columns
%ListVHeight%	Height of radio buttons with loop on many rows
%ListVWidth%	Width of radio buttons with loop on many rows

P++Gen extended attributes

This section lists extended attribute that are necessary for Power++ or Optima++ object generation. The POWERPP.EXA file contains additional extended attributes that are specific to certain templates. They are included in the POWERPP.EXA file in order to restrict their values to a predefined set of constants.

You can view the complete list of extended attributes from the Dictionary➤Extended Attributes➤List of Attributes menu in AppModeler.

Model extended attributes

Connection attributes

Attribute	Type	Description and default value
PP_DataSourceName	String	ODBC datasource name
PP_Owner	String	Data source table owner Default=%OdbcUSR%
PP_Password	String	ODBC password
PP_UserName	String	ODBC user name
PP_UseOwner	PP_Boolean	Use owner name with tables Default=False
PP_UseQuotes	PP_Boolean	Use quotes for all tables Default=False

Filter attributes

Attribute	Type	Description and default value
PP_FilterArchiveModel	String	Archived model path
PP_FilterMode	Integer	Model type (0=All, 1=Submodel, 2=Archived model, 3=User-defined) Default=0
PP_FilterObjectType	Integer	Object type binary flags (1=Table, 2=View, 4=Reference) Default=7*
PP_FilterSubModel	String	Submodel

* Table + View + Reference = 1 + 2 + 4 = 7

Project identifier attributes

Attribute	Type	Description and default value
PP_FormName	String	Main form name Default = %ModelCode%
PP_FormTemplate	String	Main form template Default = Std MDI Frame
PP_ProjectDirectory	String	Project directory
PP_ProjectName	String	Project name Default=%ModelCode%
PP_ProjectTemplate	String	Project template Default=Std Project
PP_Project Title	String	Project title Default=%ModelName%
PP_TemplateSet	String	Name of template set for project Default=determined at installation

Project generation attributes

Attribute	Type	Description and default value
PP_GenerateProject	Boolean	Flag for project generation Default=True
PP_GenerateForm	Boolean	Flag for main form generation Default=True
PP_MenuPosition	Integer	Menu bar position for generated forms Default=3
PP_UpdateMenu	Boolean	Flag for generating menu update Default=True

Table extended attributes

Default values for PowerBuilder catalog attributes are automatically loaded into a PDM from the POWERPP.EXA file.

For information on catalog attributes, see "Catalog attribute default values" on page 735.

Form identifier attributes

Attribute	Type	Description and default value
PP_FormFileName	String	Form filename Default=%TableCode%.WXF
PP_FormName	String	Form name Default=%TableCode%
PP_FormTemplate	String	Form template Default=Std Single Table
PP_FormTitle	String	Form title Default=%TableName%

Form generation attributes

Attribute	Type	Description and default value
PP_FormSelected	Boolean	Flag for inclusion in tree view Default=True
PP_FormStyle	String	Form style Default=Free form
PP_GenerateForm	Boolean	Flag for form generation Default=True
PP_GenerateMenu	Boolean	Flag for menu generation Default=True
PP_GridTemplate	String	Grid template for form Default=Std Grid
PP_MenuOption	String	Menu name for generated form Default=%TableName%

DataWindow attributes

Attribute	Type	Description and default
PP_DWGen	Boolean	Generate DataWindow Default=False
PP_DWLib	Path	DataWindow library path
PP_DWName	String	DataWindow name
PP_DWTemplate	String	DataWindow template Default=DW1
PP_DWUpdate	Boolean	DataWindow update Default=True

View extended attributes

Form identifier attributes

Attribute	Type	Description and default value
PP_FormFileName	String	Form filename Default=%ViewCode%.WXF
PP_FormName	String	Form name Default=%ViewCode%
PP_FormTemplate	String	Form template Default=Std Single View
PP_FormTitle	String	Form title Default=%ViewName%

Form generation attributes

Attribute	Type	Description and default value
PP_ComputedFields Template	String	Computed Fields Template Default=Std Computed Field
PP_FormSelected	Boolean	Flag for inclusion in tree view Default=True
PP_FormStyle	String	Form style Default=Grid
PP_GenerateForm	Boolean	Flag for form generation Default=True
PP_GenerateMenu	Boolean	Flag for menu generation Default=True
PP_GridTemplate	String	Grid template for form Default=Std Grid
PP_MenuOption	String	Menu name for generated form Default=%ViewName%

DataWindow attributes

Attribute	Type	Description and default value
PP_DWGen	Boolean	Generate DataWindow Default=False
PP_DWLib	Path	DataWindow library path
PP_DWName	String	DataWindow name
PP_DWTemplate	String	DataWindow template Default=DW1
PP_DWUpdate	Boolean	DataWindow update Default=True

Reference extended attributes

Form generation attributes

Attribute	Type	Description and default value
PP_DetailView	Object	Detail view Default=0
PP_MasterView	Object	Master view Default=0
PP_FormSelected	Boolean	Flag for inclusion in tree view Default=True
PP_GenerateForm	Boolean	Flag for form generation Default=True
PP_GenerateMenu	Boolean	Flag for menu generation Default=True
PP_MenuOption	String	Menu name for generated form Default=%ReferenceName%...

Form identifier attributes

Attribute	Type	Description and default value
PP_FormFileName	String	Form filename Default=%ReferenceCode%.WXF
PP_FormName	String	Form name Default=%ReferenceCode%
PP_FormTemplate	String	Form template Default=Master/Detail
PP_FormTitle	String	Form title Default=%ReferenceName%

Presentation style attributes

Attribute	Type	Description and default value
PP_DDWTemplate	String	Detail DataWindow template Default=Grid3D
PP_DetailStyle	String	Detail style Default=Grid
PP_DetailGridTemplate	String	Grid template for detail fields Default=Std Grid
PP_MDWTemplate	String	Master DataWindow template Default=Form3D

Attribute	Type	Description and default value
PP_MasterStyle	String	Master style Default=Free form
PP_MasterGridTemplate	String	Grid template for master fields Default=Std Grid

DataWindow attributes

Attribute	Type	Description and default value
PP_DDWGen	Boolean	Generate detail DataWindow Default=False
PP_DDWLib	Path	Detail DataWindow library path
PP_DDWName	String	Detail DataWindow name
PP_DDWUpdate	Boolean	Detail DataWindow update Default=True
PP_MDWGen	Boolean	Generate master DataWindow Default=False
PP_MDWLib	Path	Master DataWindow library path
PP_MDWName	String	Master DataWindow name
PP_MDWUpdate	Boolean	Master DataWindow update Default=True

Column extended attributes

Most of the column extended attributes in the POWERPP.EXA file are specific to Power++ field templates. Only those column extended attributes that are global in scope are listed below.

Default values for PowerBuilder catalog attributes for columns are automatically loaded into a PDM from the POWERPP.EXA file.

For information on catalog attributes, see "Catalog attribute default values" on page 735.

Field attributes

Attribute	Type	Description and default value
PP_FieldStyle	String	Field style Default=Edit
PP_FieldTemplate	String	Field template Default=(Default)
PP_Height	Integer	Column height Default=240
PP_Width	Integer	Column width Default=1000
PP_Left	Integer	Left value of first control

DataWindow generation attributes

Extended attribute	Data type	Description and default
PB_GenDetail	Boolean	Generate column in detail DataWindow Default=True
PB_GenMaster	Boolean	Generate column in master DataWindow Default=True
PB_GenSingle	Boolean	Generate column in DataWindow Default=True
PB_SkipColumn	Boolean	Temporary flag that is replaced by PB_GenDetail, PB_GenMaster, or PB_GenSingle column generation value Default=False

DelphiGen variables and keywords

This section lists variables and keywords recognized by AppModeler for Delphi (DelphiGen). DelphiGen variables complement the system variables listed in this appendix on page 724.

DelphiGen project variables

The following variables are used as placeholders for project identifiers:

Variable	PowerDesigner value
%ProjectTitle%	Project title
%ProjectName%	Project name
%Database%	Database name
%OdbcDSN%	ODBC data source name
%OdbcUSR%	ODBC user name
%OdbcPWD%	ODBC password

DelphiGen form variables

The following variables are used as placeholders for form identifiers and for database connection parameters:

Variable	PowerDesigner value
%FormName%	Form name
%FormTitle%	Form title
%MasterTableName%	Master table name
%MasterTableCode%	Master table code
%MasterViewName%	Master view name
%MasterViewCode%	Master view code
%MasterObjectName%	Master table or view name
%MasterObjectCode%	Master table or view code
%DetailTableName%	Detail table name
%DetailTableCode%	Detail table code

Variable	PowerDesigner value
%DetailViewName%	Detail view name
%DetailViewCode%	Detail view code
%DetailObjectName%	Detail table or view name
%DetailObjectCode%	Detail table or view code
%DataSource%	dsMain, dsMaster, or dsDetail

DelphiGen keywords

These are reserved words used by DelphiGen templates.

Keyword	Name for
dsSingle	Data source used by the single form
dsMaster	Data source used by the master form
dsDetail	Data source used by the detail form

DelphiGen field variables

General variables

Variable	PowerDesigner value
%ColumnLength%	Maximum number of characters
%ColumnFormat%	Column data mask
%ColumnType%	Column data type
%DataField%	Bound control identifier
%DataSource%	dsSingle, dsMaster, or dsDetail
%DefaultValue%	Column default value
%FieldCode%	Field code
%ListValues%	Number of elements (Null through 'n')
%ListNumber%	Number of elements (0 through 'n')

List variables

Variable	PowerDesigner value
%ListData%	Data value
%ListDisplay%	Display value
%ListIndex0%	Zero-based index: 0 to (%ListNumber%-1)
%ListIndex1%	One-based index: 1 to %ListNumber%
%ListLeft%	Current *x* coordinate on screen
%ListTop%	Current *y* coordinate on screen
%ListWidth%	Current *dx* coordinate on screen
%ListHeight%	Current *dy* coordinate on screen

DelphiGen extended attributes

This section lists extended attribute that are necessary for Delphi object generation. The DELPHI.EXA file contains additional extended attributes that are specific to certain templates. They are included in the DELPHI.EXA file in order to restrict their values to a predefined set of constants.

You can view the complete list of extended attributes from the Dictionary➤Extended Attributes➤List of Attributes menu in AppModeler.

Model extended attributes

Connection attributes

Attribute	Type	Description and default value
DL_DatabaseAlias	String	Database alias
DL_DataSourceName	String	ODBC datasource name
DL_KeepConnection	DL_Boolean	Keep connection Default=True
DL_LoginPrompt	DL_Boolean	Login prompt Default=True
DL_TransIsolation	DL_TransIsolation	Transaction isolation level Default=Read Committed
DL_Password	String	ODBC password
DL_UseQuotes	DL_Boolean	Quote table and column codes Default=True
DL_UserName	String	ODBC user name

Filter attributes

Attribute	Type	Description and default value
DL_FilterArchiveModel	String	Archived model path
DL_FilterMode	Integer	Model type (0=All, 1=Submodel, 2=Archived model, 3=User-defined) Default=0
DL_FilterObjectType	Integer	Object type binary flags (1=Table, 2=View, 4=Reference) Default=7*
DL_FilterSubModel	String	Submodel

* Table + View + Reference = 1 + 2 + 4 = 7

Project identifier attributes

Attribute	Type	Description and default value
DL_FormName	String	Main form name Default = %ModelCode%
DL_FormTemplate	String	Main form template Default = Std MDI Frame
DL_ProjectDirectory	String	Project directory
DL_ProjectName	String	Project name Default=%ModelCode%.DXP
DL_ProjectTemplate	String	Project template Default= Std Project
DL_Project Title	String	Project title Default=%ModelName%
DL_TemplateSet	String	Name of template set for project

Project generation attributes

Attribute	Type	Description and default value
DL_GenerateProject	Boolean	Flag for project generation Default=True
DL_GenerateForm	Boolean	Flag for main form generation Default=True
DL_MenuPosition	Integer	Menu bar position for generated forms Default=3
DL_UpdateMenu	Boolean	Flag for generating menu update Default=True

Table extended attributes

Form identifier attributes

Attribute	Type	Description and default value
DL_FormFileName	String	Form filename Default=%TableCode%.DXF
DL_FormName	String	Form name Default=%TableCode%
DL_FormTemplate	String	Form template Default=Std Single Table
DL_FormTitle	String	Form title Default=%TableName%

Form generation attributes

Attribute	Type	Description and default value
DL_FormSelected	Boolean	Flag for inclusion in tree view Default=True
DL_FormStyle	String	Form style Default=Free form
DL_GenerateForm	Boolean	Flag for form generation Default=True
DL_GenerateMenu	Boolean	Flag for menu generation Default=True
DL_GridTemplate	String	Grid template for form Default=Std Grid
DL_MenuOption	String	Menu name for generated form Default=%TableName%
DL_TFieldsTemplate	String	TField objects template Default=Std TField
DL_Update	Boolean	Is form updatable? Default=True

View extended attributes

Form identifier attributes

Attribute	Type	Description and default value
DL_FormFileName	String	Form filename Default=%ViewCode%.DXF
DL_FormName	String	Form name Default=%ViewCode%
DL_FormTemplate	String	Form template Default=Std Single View
DL_FormTitle	String	Form title Default=%ViewName%

Form generation attributes

Attribute	Type	Description and default value
DL_FormSelected	Boolean	Flag for inclusion in tree view Default=True
DL_FormStyle	String	Form style Default=Grid
DL_GenerateForm	Boolean	Flag for form generation Default=True
DL_GenerateMenu	Boolean	Flag for menu generation Default=True
DL_GridTemplate	String	Grid template for form Default=Std Grid
DL_MenuOption	String	Menu name for generated form Default=%ViewName%
DL_TFieldsTemplate	String	TFields template Default=Std TFields
DL_Update	Boolean	Is form updatable? Default=True

Computed fields attributes

Attribute	Type	Description and default value
DL_ComputedFieldsTemplate	String	Computed fields template Default=Std Computed Fields
DL_ComputedTFieldsTemplate	String	Computed TFields template Default=Std TFields
DL_ComputedTFieldsType	DL_Comp TFieldType	TField object type Default=String

Reference extended attributes

Form generation attributes

Attribute	Type	Description and default value
DL_DetailView	Object	Detail view Default=0
DL_MasterView	Object	Master view Default=0
DL_FormSelected	Boolean	Flag for inclusion in tree view Default=True
DL_GenerateForm	Boolean	Flag for form generation Default=True
DL_GenerateMenu	Boolean	Flag for menu generation Default=True
DL_MenuOption	String	Menu name for generated form Default=%ReferenceName%

Form identifier attributes

Attribute	Type	Description and default value
DL_FormFileName	String	Form filename Default=%ReferenceCode%.WXF
DL_FormName	String	Form name Default=%ReferenceCode%
DL_FormTemplate	String	Form template Default=Master/Detail
DL_FormTitle	String	Form title Default=%ReferenceName%

Master/Detail display attributes

Attribute	Type	Description and default value
DL_DetailStyle	String	Detail style Default=Grid
DL_DetailGridTemplate	String	Grid template for detail fields Default=Std Grid
DL_DetailTFieldsTemplate	String	Detail TField objects template Default=Std TFields
DL_DetailUpdate	DL_Boolean	Is dsDetail updatable? Default=True

Attribute	Type	Description and default value
DL_MasterStyle	String	Master style Default=Free form
DL_MasterGridTemplate	String	Grid template for master fields Default=Std Grid
DL_MasterTFieldsTemplate	String	Master TField objects template Default=Std TFields
DL_MasterUpdate	DL_Boolean	Is dsMaster updatable? Default=True

Column extended attributes

Most of the column extended attributes in the DELPHI.EXA file are specific to DelphiGen field templates. Only those column extended attributes that are global in scope are listed below.

Generation attributes

Attribute	Type	Description and default value
DL_FieldStyle	String	Field style Default=Edit
DL_FieldTemplate	String	Field template Default=(Default)
DL_GenDetail	Boolean	Generate flag for column in a detail form Default=True
DL_GenMaster	Boolean	Generate flag for column in a master form Default=True
DL_GenSingle	Boolean	Generate flag for column in a single form Default=True
DL_Height	Integer	Column height Default=240
DL_Width	Integer	Column width Default=1000
DL_Left	Integer	Left value of first control

WebGen variables and keywords

This section lists variables and keywords recognized by AppModeler for Dynamo (WebGen). WebGen variables complement system variables listed in this appendix on page 724.

WebGen project variables

When you generate a project, WebGen instantiates project variables with values you enter in the WebGen Model Extended Attributes dialog box.

Variable	PowerDesigner value
%IndexPage%	Index page name
%ProjectName%	Main page name
%ProjectTitle%	Main page title

WebGen page variables

WebGen uses these variables as placeholders for project page identifiers.

Page variables for a table or view

Variable	PowerDesigner value
%FRMPage%	Free Form page name
%FRMTitle%	Free Form page title
%IDXPage%	Type of page to open from Index (QBE, Tabular, or Free Form)
%IDXTitle%	Title of page to open from Index
%PageTitle%	Index option
%PageType%	Page type (QBE, Tabular, or Free Form)
%QBEPage%	QBE page name
%QBETitle%	QBE page title
%TBLPage%	Tabular page name
%TBLTitle%	Tabular page title
%Updatable%	NI_Update extended attribute value

Page variables for a reference

Variable	PowerDesigner value
%ChildCode%	Table code of child table in a reference
%ChildName%	Table name of child table in a reference
%ChildPage%	Page name generated from child table
%ChildPageTitle%	Index option generated from child table
%ChildTitle%	QBE title, Tabular title, or Free Form title
%ChildUpdatable%	Updatable setting for page generated from child table
%FKColumnCode%	Column code of foreign key in child table
%FKFieldCode%	Unique column code of foreign key
%FKParentColumn%	Column code of foreign key in parent table
%FKParentForm%	Page based on parent table
%FKParentTable%	Parent table
%PKColumnCode%	Column code of primary key
%PKFieldCode%	Unique column code of primary key

WebGen keywords

These are reserved words used by WebGen templates.

Keyword	Description
FRM	Page type for individual record display
TBL	Page type for tabular list display
QBE	Page type for query entry
Mandatory	Page type with this indication must be generated
Start	Defines page type to open from Index menu link
Link	Defines page type to open from primary key link
Zoom	Defines page type to open from foreign key link

WebGen field variables

General variables

Variable	PowerDesigner value
%ColumnDefault%	Default value in column Check Parameters dialog box
%ColumnFormat%	Column data mask
%ColumnIsFK%	Column status: is column a foreign key
%ColumnIsMandatory%	Column status: is column mandatory
%ColumnIsNumeric%	Column status: does column have a numeric data type
%ColumnIsPK%	Column status: is column a primary key
%ColumnLength%	Maximum number of characters
%ColumnMax%	Maximum value in Check Parameters dialog box
%ColumnMin%	Minimum value in Check Parameters dialog box
%ColumnQuote%	Null for column with numeric data type, single quote for column with non-numeric data type
%ColumnSQLName%	Column expression
%ColumnType%	Column data type
%ColumnUnit%	Unit value in column Check Parameters dialog box
%DataField%	Bound control identifier
%FieldCode%	Field code

List variables

Variable	PowerDesigner value
%ListData%	Data value
%ListDisplay%	Display value
%ListIndex0%	Zero-based index: 0 to (%ListNumber%-1)
%ListIndex1%	One-based index: 1 to %ListNumber%
%ListNumber%	Number of values in Check Parameters List of Values
%IsListDefault%	Compares column List of Values with column Default value

WebGen extended attributes

This section lists extended attribute that are necessary for HTML object generation. These attributes and their default values are loaded automatically into your PDM from the NIDYNAMO.EXA file.

The EXA file includes additional extended attributes specific to certain templates in order to restrict their values to a predefined set of constants. You can view the complete list of extended attributes from the Dictionary➤Extended Attributes➤List of Attributes menu in AppModeler.

Model extended attributes

Connection attributes

Attribute	Type	Description and default value
NI_DataSourceName	String	ODBC data source name
NI_Password	String	ODBC password
NI_UserName	String	ODBC user name
NI_WebConnection	String	Web connection Default=default
NI_WebDataSourceName	String	Web ODBC data source name
NI_WebFolder	String	Web site folder Default=/Site/app/%ModelCode%
NI_WebPassword	String	Web server ODBC password
NI_WebSite	String	Web site Default=http://localhost
NI_WebUserName	String	Web ODBC user name

Filter attributes

Attribute	Type	Description and default value
NI_FilterArchiveModel	String	Archived model path
NI_FilterMode	Integer	Model type (0=All, 1=Submodel, 2=Archived model, 3=User-defined) Default=0
NI_FilterObjectType	Integer	Object type binary flags (1=Table, 2=View) Default=3*
NI_FilterSubModel	String	Submodel

* Table + View = 1 + 2 = 3

Project Identifier attributes

Attribute	Type	Description and default value
NI_IndexPageName	String	Index page name Default=Index
NI_IndexPageTemplate	String	Index page template Default=Std MDI Frame
NI_MainPageName	String	Main page name Default=%ModelCode%
NI_MainPageTemplate	String	Main page template Default=Std Main Page
NI_MainPageTitle	String	Main page title Default=%ModelName%
NI_ProjectDirectory	String	Project directory Default=Project Directory
NI_TemplateSet	String	Name of template set for project Default=SQL Anywhere 5.5

Project generation attributes

Attribute	Type	Description and default value
NI_GenerateIndexPage	Boolean	Flag for index page generation Default=True
NI_GenerateMainPage	Boolean	Flag for main page generation Default=True
NI_GenerateMenu	Boolean	Include menu option Default=True
NI_GenerationDisk	Boolean	Flag to indicate if generating as files to disk Default=False
NI_UpdateIndex	Boolean	Flag to allow Index page updates Default=True
NI_UseQuotes	NI_Boolean	Use quotes around table and column codes Default=True

Table extended attributes

Page identifier attributes

Attribute	Type	Description and default value
NI_FormPageName	String	Form page name Default=F%TableCode%.STM
NI_FormPageTemplate	String	Form page template Default=Std Free Form
NI_FormPageTitle	String	Form page title Default=%TableName%: Form
NI_QBEPageName	String	QBE page name Default=Q%TableCode%.STM
NI_QBEPageTemplate	String	QBE page template Default=Std QBE
NI_QBEPageTitle	String	QBE page title Default=%TableName%: QBE
NI_TabularPageName	String	Tabular page name Default=T%TableCode%.STM
NI_TabularPageTemplate	String	Tabular page template Default=Std Tabular
NI_TabularPageTitle	String	Tabular page title Default=%TableName%: List
NI_TemplateStyle	String	Page template type Default=Table Style

Page generation attributes

Attribute	Type	Description and default value
NI_FormGeneratePage	Boolean	Flag for generation of Form page Default=True
NI_GenerateIndex	Boolean	Flag for generation of page index entry Default=True
NI_GeneratePage	Boolean	Flag for page generation Default=True
NI_IndexOption	String	Name for page entry in index Default=%TableName%...
NI_PageSelected	Boolean	Selected in previous generation Default=True
NI_QBEGeneratePage	Boolean	Flag for generation of QBE page Default=True
NI_TabularGeneratePage	Boolean	Flag for generation of Tabular page Default=True
NI_Update	Boolean	Flag for update permission Default=True

View extended attributes

Page identifier attributes

Attribute	Type	Description and default value
NI_QBEPageName	String	QBE page name Default=Q%ViewCode%.STM
NI_QBEPageTemplate	String	QBE page template Default=Std View QBE
NI_QBEPageTitle	String	QBE page title Default=%ViewName%: QBE
NI_TabularPageName	String	Tabular page name Default=T%ViewCode%.STM
NI_TabularPageTemplate	String	Tabular page template Default=Std ViewTabular
NI_TabularPageTitle	String	Tabular page title Default=%ViewName%: List
NI_TemplateStyle	String	Page template type Default=Std View

Page generation attributes

Attribute	Type	Description and default value
NI_ComputedFieldsTemplate	String	Computed Fields Template Default=Std Computed Field
NI_GenerateIndex	Boolean	Flag for generation of index entry Default=True
NI_GeneratePage	Boolean	Flag for page generation Default=True
NI_IndexOption	String	Name for page entry in index Default=%ViewName%...
NI_PageSelected	Boolean	Selected in previous generation Default=True
NI_QBEGeneratePage	Boolean	Flag for generation of QBE page Default=True
NI_TabularGeneratePage	Boolean	Flag for generation of Tabular page Default=True

Column extended attributes

Most of the column extended attributes in the EXA file are specific to WebGen field templates. Only those column extended attributes that are global in scope are listed below.

Field attributes

Attribute	Type	Description and default value
NI_DefaultValue	String	Default value in column Check Parameters
NI_FieldStyle	String	Field style Default=Edit
NI_FieldTemplate	String	Field template Default=(Default)
NI_GenFRM	Boolean	Generate this column in Form page Default=True
NI_GenQBE	Boolean	Generate this column in QBE page Default=True
NI_GenTBL	Boolean	Generate this column in Tabular page Default=True

Glossary

alternate key — Column or columns whose values uniquely identify a row in the table and are not primary key columns

business rule — Written statement specifying what the information system must do or how it must be structured to support business needs

clustered index — Index in which the physical order and the logical (indexed) order is the same

column — Data structure that contains an individual data item within a row (record), model equivalent of a database field

constraint — Named check that enforces data requirements, default values, or referential integrity on a table or a column

data source — Identification of the data to access, its operating system, DBMS, and network platform

domain — Set of values for which a data item is valid

extended attribute — Additional information that completes the definition of an object for documentary purposes or for use by an external application such as a fourth-generation language (4GL)

foreign key — Column or columns whose values depend on and migrate from a primary key in another table

4GL — External application that uses a fourth-generation language, usually to generate a client/server application

index	Data structure that is based on a key and that speeds access to data and controls unique values
ODBC	Open Database Connectivity (ODBC) interface which gives PowerDesigner access to data in database management systems (DBMS)
ODBC driver	Part of the Open Database Connectivity (ODBC) interface that processes ODBC functions calls, submits SQL requests to a specific data source, and returns results to the application
Physical Data Model (PDM)	Table-reference diagram that models the information system including the details of physical implementation
primary key	Column or columns whose values uniquely identify a row in a table
property sheet	Window that displays the properties of an object
reference	Link between the primary key and the foreign key of different tables
referential integrity	Rules governing data consistency, specifically the relationships among primary keys and foreign keys of different tables
storage	Named partition that stores tables and indexes on a storage device
table	Collection of rows (records) that have associated columns (fields)
tablespace	Named partition that stores tables and indexes in a database
trigger	Special form of stored procedure that goes into effect when you insert, delete, or update a specified table or column
unique index	Index in which no two rows can have the same index value, including NULL
view	Alternate way of looking at the data in one or more tables. Usually created as a subset of columns from one or more tables

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z