



TRAINING WORKBOOK

Oracle Training

Oracle Forms 6i



Confidential Business Information

This documentation is proprietary information of SCT and is not to be copied, reproduced, lent or disposed of, nor used for any purpose other than that for which it is specifically provided without the written permission of SCT.

Prepared For: Release 6.x

Prepared By: SCT
4 Country View Road
Malvern, Pennsylvania 19355
United States of America

Issued: May 2004

©1992-1995, 1997, 1999-2002, 2003, 2004 Systems & Computer Technology Corporation. All rights reserved. The unauthorized possession, use, reproduction, distribution, display, or disclosure of this material or the information contained herein is prohibited.

In preparing and providing this publication, SCT is not rendering legal, accounting, or other similar professional services. SCT makes no claims that an institution's use of this publication or the software for which it is provided will insure compliance with applicable federal or state laws, rules, or regulations. Each organization should seek legal, accounting and other similar professional services from competent providers of the organization's own choosing.

“SCT,” the SCT logo, “Banner” and the Banner logo are trademarks of SCT. Third party hardware and software product names and trademarks are owned by their respective third party owners/providers, and SCT makes no claim to such names or trademarks.

Section A: Introduction

Overview

Intended audience Programmers with Oracle backgrounds who develop add-on modules for SCT Banner or modify the baseline SCT Banner forms.

Prerequisites Prerequisites include:

- Banner Navigation training
- Introduction to Oracle training
- Advanced PL/SQL and Database Objects training

In this section These topics are covered in this section.

Topic	Page
Introduction	A-4
Workbook contents	A-5

Introduction

Introduction

Oracle Forms is a development tool used for building client-server or web-based database applications that are portable to a variety of GUI and character mode platforms. All Banner forms have been written with this tool.

This course provides an in-depth discussion of how to utilize Oracle Forms 6i.

Objectives

Upon completion of this course, attendees will be able to:

- Run an Oracle Forms 6i application
- Describe the layout editor
- Create and customize a default form
- Explain canvasses, windows, menus, modules, blocks, items properties, relations, and visual attributes
- Describe alerts, record groups, lists, and triggers
- Explain stored procedures and shared libraries

Topics

- Oracle Forms 6i application
 - Layout editor
 - Default forms: creating and customizing
 - Canvasses, windows, menus, modules, block properties, relations, and visual attributes
 - Stored procedures, shared libraries, and images
 - SCT Banner design standards
-

Workbook contents

Workbook contents

This workbook contains the following sections:

- Section A: Introduction
 - Section B: Introduction to Oracle Forms
 - Section C: Creating a Form Module
 - Section D: The Layout Editor
 - Section E: Customizing Data Blocks
 - Section F: Text and Display Items
 - Section G: Additional Items
 - Section H: Introduction to Triggers
 - Section I: Navigation with Triggers
 - Section J: Validation Triggers
 - Section K: Query Triggers
 - Section L: Transaction Triggers
 - Section M: Creating Lists of Values and Editors
 - Section N: Canvases Part II
 - Section O: Runform Messages and Alerts
 - Section P: Sharing Objects and Code
 - Section Q: Multiple-Form Applications
 - Section R: Answer Guide
-

Section B : Introduction to Oracle Forms

Overview

Purpose

Oracle Forms is part of Developer 2000, a suite of application development tools, which includes:

- Forms
- Procedure Builder
- Reports
- Graphics
- Procedure Builder
- Query Builder
- Schema Builder
- Translation Builder

The development tools have been designed to work together, and share many components. This course will focus on developing Forms, the most robust development tool within the suite.

Objectives

This section will examine the following:

- What is Forms?
- What can Forms do?
- Forms components
- Starting the Builder
- Examining the Builder components

In this section

These topics are covered in this section.

Topic	Page
Overview of Oracle Forms	B-2
Forms components	B-3
Starting the Builder	B-4
The Object Hierarchy	B-5
The Layout Editor	B-6
The Property Palette	B-7

Overview of Oracle Forms

What is Forms? Oracle Forms is a development tool used for building client-server or web-based database applications that are portable to a variety of GUI and character mode platforms. All Banner forms have been written with this tool.

What can Forms do? Oracle Forms allows your users to insert, update, delete, and query data from the database through GUI items. These include:

- Buttons
- Checkboxes
- Lists
- Radio Buttons
- Text Items

As a developer, Forms allows you to quickly create applications which:

- Can use a number of data sources
 - Allow code and objects to be easily copied
 - Are portable across platforms, including GUI and character-mode environments
-

Forms components

Components

Component	Description
Forms Builder	The development environment.
Forms Compiler	Used to compile application files to create executable runfiles.
Forms Runtime	Runtime engine that is used to run a generated Oracle Forms application.

Forms modules

Oracle Forms applications include four types of modules:

Module type	Description
Forms	Collections of objects and data, which allow the user to interact with the database. Data items are arranged into records.
Menus	Collections of menu objects (main menu, pull-down menu, menu items) and menu command code.
PL/SQL Library	Collections of PL/SQL procedures, functions, and packages that can be called from other modules.
Object Library	Collections of form objects (items, data blocks, etc...) that can be called from other modules.

Starting the Builder

Starting the Builder

The Builder can be started in the following ways:

- Double-click the icon
- Locate the Forms Builder within the Windows menu system
- Enter the following command at the system prompt:
`ifbld60 [module] [userid/password] [parameters]`

Connecting to the database

- Select **File→ Connect**. The Connect dialog appears.
 - Enter a valid username, password, and database connect string in the appropriate fields.
 - Choose Connect. When you first start Oracle Forms Builder, you will see the Object Navigator.
 - Have everyone log into the Designer at this time.
-

The Object Hierarchy

Object Hierarchy

The Object Hierarchy provides a hierarchical display of the objects in all open modules.

- Objects are grouped under the appropriate node
- Objects and nodes in the Navigator are displayed with a + or - symbol to indicate whether they are currently expanded or collapsed

Form object types

Although we will be dealing with many types of objects within the Object Navigator during the class, the following are the major objects within a form:

Object	Description
Items	Interface objects that display information to operators and allow them to interact with your application.
Data Block	Each item in a form belongs to a data block: <ul style="list-style-type: none">• Logical containers that have no physical representation - only items are visible in the application interface• Provide a mechanism for grouping related items into a functional unit for storing, displaying, and manipulating records
Canvas	Area where you can "paint," or design, the layout of your form. A form can contain more than one canvas (known as a page in earlier Forms versions).

The Layout Editor

Layout Editor To view the layout of a canvas, open the Layout Editor window by selecting Tools→Layout Editor. When you begin adding objects to the form, you will be able to arrange the objects by dragging and dropping items with the mouse.

The Layout Editor will be discussed in detail in Section H.

The Property Palette

Property Palette	The Property Palette provides complete control over your form, block, item and other objects.
Components	<p>The Toolbar contains buttons giving convenient access to functions relevant to setting properties: copy, paste, add, delete, class create and inherit.</p> <p>The Context bar identifies which object is currently having its properties displayed by the Property Palette.</p> <p>The Property List is a two column display showing the property names in the left column and their current values in the right column. The properties are grouped by category. A + in front of a category name indicates that the category is expanded and that the properties within that category are all visible.</p>
Comparing objects	<p>Two or more objects can be compared by selecting all of the items you want to compare, then perusing the Property Palette. Properties having the same value in all selected objects will show the shared value, while properties that do not have the same value will display ***** for that property instead of a value.</p> <p>When you are showing the properties for multiple objects in a single palette, any property changed will be changed in all of the selected objects, overwriting whatever prior settings the objects had for that property.</p>

Section C: Creating a Form Module

Overview

Purpose This section is an introduction to creating and configuring basic form modules via using wizards.

Objectives This section will examine the following:

- Creating and naming form modules
- The Data Block Wizard
- The Layout Wizard
- Canvases

In this section These topics are covered in this section.

Topic	Page
Creating a form module	C-2
Naming a form module	C-3
How forms relate to tables	C-4
Data Block Wizard: Navigating the wizards	C-5
Layout Wizard	C-10
Canvases	C-15
Content Canvas	C-17


Creating a form module

Form options

When the Form Builder is initially opened, the following options appear:

- Begin building with the Data Block Wizard
- Begin building manually – New Form 'MODULE1' is already created
- Begin building by using an existing template
- Open an existing form

Create additional forms

To create additional forms, select **File**→**New**→**Form** or highlight 'Forms' in the Object Navigator and click the  icon on the toolbar.

Several modules can be opened at the same time.

Naming a form module

Changing the default name

By default, when a form is created, the form is named *MODULExx* , where *xx* stands for the next number available for the module names.

You can rename the module by doing one of the following:

Step	Action
1	Double-click the module name and edit the name, or:
2	Access the form's Property Palette via one of the following methods: <ul style="list-style-type: none">• Select Tools→Property Palette• Double-click the form module icon in the Object Navigator. The first property is the form module name• Right-click on the Form name and choose Property Palette

Module naming rules


- Must begin with a letter
- Can include up to 30 characters, including certain special characters (\$, _)
- Cannot include Oracle or Forms reserved words

Exercise 1

Create a new form module called SWAIDEN. The naming convention follows Banner standards. SWAIDEN stands for:

- Student
- Custom object
- Application form
- Identification

How forms relate to tables

Forms	A form is a group of related data blocks. Data Blocks are the linked between the form and the database; each data block can relate to one table in the database.
Data blocks	A data block is a logical container for interface items. All items, whether they come from a base table or not, must be in a data block.
Base table data blocks	<p>A base table data block is a data block that is associated with a table in the database.</p> <p>You may create base table data blocks with the Data Block Wizard, or by highlighting 'Data Blocks' in the Object Navigator and clicking the  icon on the toolbar.</p>

Data Block Wizard: Navigating the wizards

Buttons

- Cancel – Cancels any changes and exits the wizard
 - Help – Displays online help for the current page of the wizard
 - Back – Navigates to the previous wizard page
 - Next – Navigates to the next wizard page
 - Apply – Applies changes without exiting the wizard (Only available when the wizard is reentered)
 - Finish – Saves any changes and exits the wizard
-

Creating a Base Table Data Block

Step	Action
1	Select Tools→Data Block Wizard to display the Data Block Wizard, or right-click 'Data Blocks' and select the Data Block Wizard.
2	Enter the data block information for the Data Block Wizard (see below).
3	Choose Finish to create the data block and dismiss the Data Block wizard.
4	Enter the data block information for the Layout Wizard (see below).
5	Choose Finish to create the layout for the data block and dismiss the Layout wizard. <u>Note:</u> Do not click “Finish” until all the pages have been entered to your satisfaction.

Continued on the next page

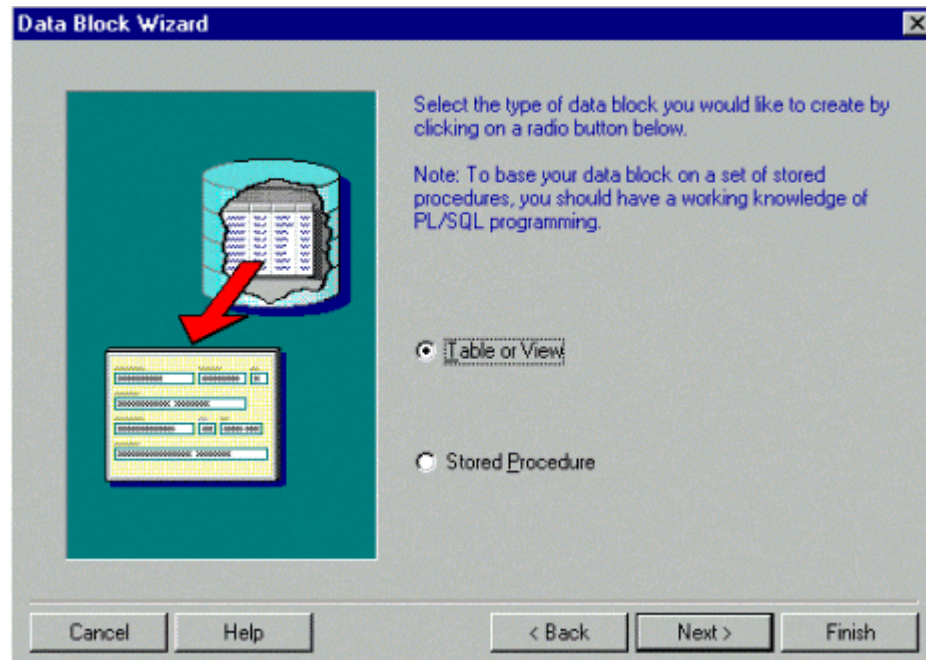
Data Block Wizard: Navigating the wizards, Continued

Type Page –
Data Block Step
#1

Choose one of the two data sources:

- Table or View
- Stored Procedure

Screen



Continued on the next page

Data Block Wizard: Navigating the wizards, Continued

Table Page - Data Block Step #2

Component	Usage
Table or View	Enter the name of the table or view you want the data block to be based on. You can choose the Browse button to the right of the field to invoke the Tables browser. Use the Refresh button to populate the Available Columns.
Enforce Data Integrity	Specifies whether Oracle Forms should enforce the table and column constraints defined in the data dictionary for the data block's base table.
Available Columns	Displays all the columns from the table that can be used within the data block.
Tools	> - Include only the selected column(s) >> - Include all columns << - Exclude all columns < - Exclude only the selected column(s)
Database Items	Displays the columns from the base table that will be included in the data block.

Screen

Continued on the next page

Data Block Wizard: Navigating the wizards, Continued

Exercise 2

In the new form module, create a data block based on the SWRIDEN table.

- Do not include the swriden_change_ind and swriden_activity_date columns

Continued on the next page

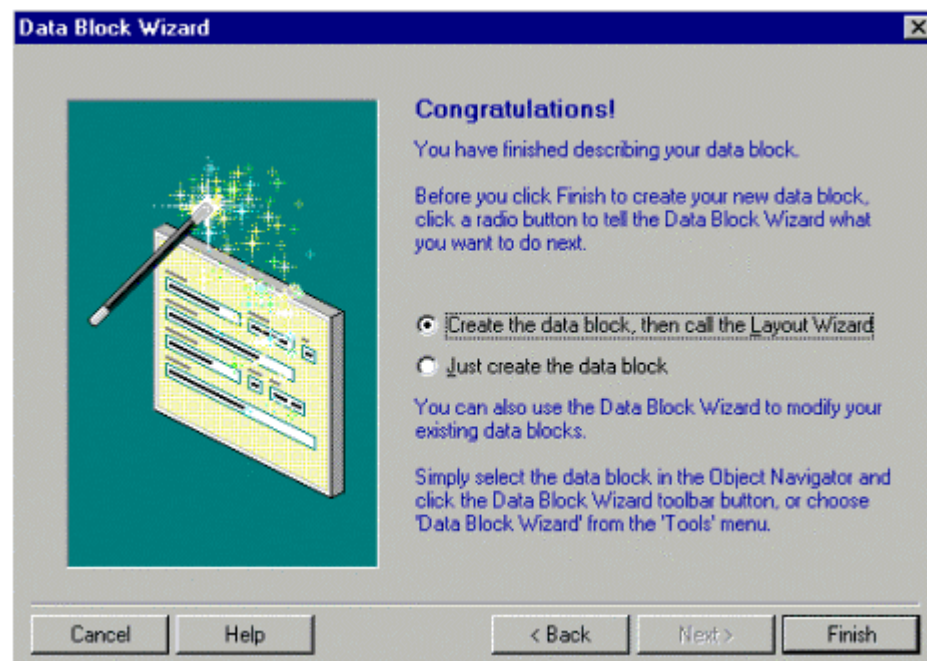
Data Block Wizard: Navigating the wizards, Continued

Finish Page - Data Block Step #3

The Data Block Wizard is now complete. The two options:

- Create the data block and then call the Layout Wizard
- Just create the data block

Screen

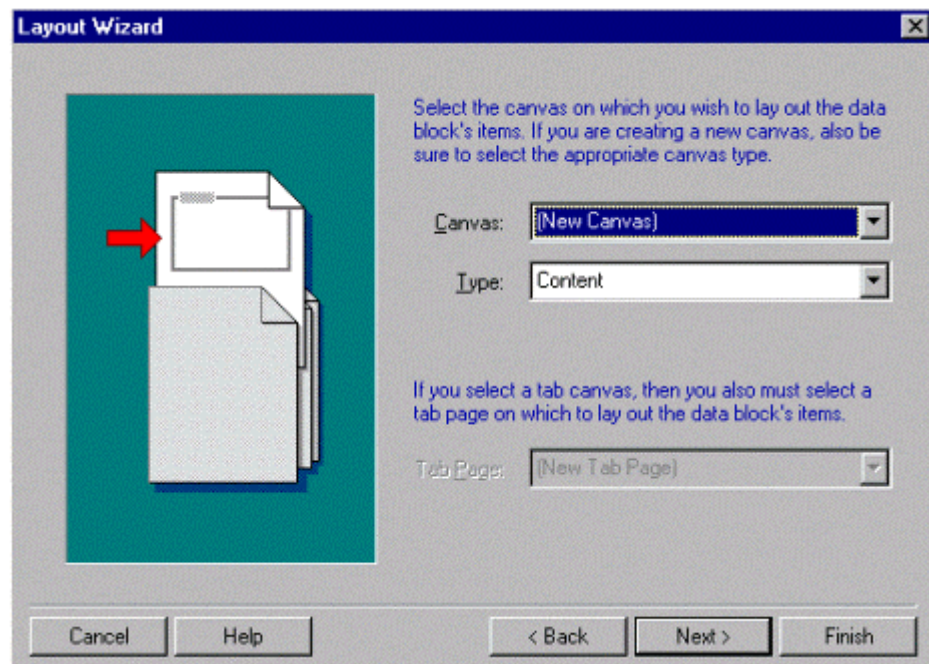


Layout Wizard

Canvas Page – Layout Step #1

Component	Usage
Canvas	Select the canvas on which you want Oracle Forms to place the items that will be created in the data block. If New Canvas is selected, Oracle Forms automatically will create a new canvas.
Type	Type of Canvas – (Content, Stacked, Tab, Vertical Toolbar, or Horizontal Toolbar).
Tab Page	Specific tab to place items – Only used for tab canvas.

Screen



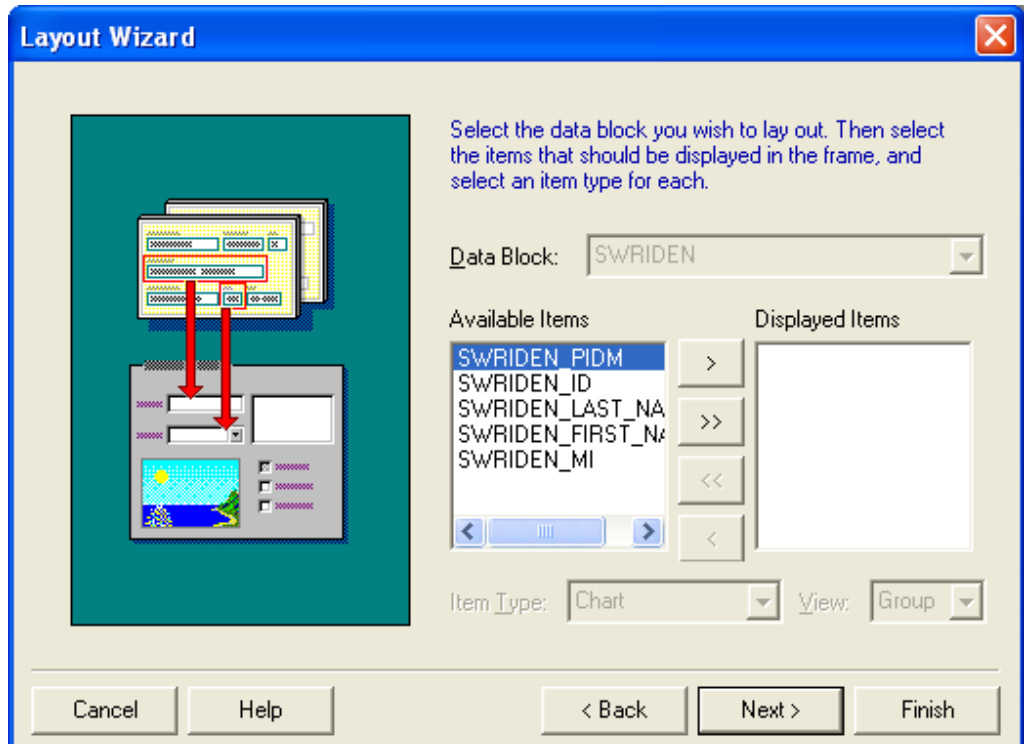
Continued on the next page

Layout Wizard, Continued

Data Block Page – Layout Step #2

Component	Usage
Data Block	The data block that is to be displayed.
Available Items	Specifies the data block columns that are available for display.
Tools	> - Include only the selected column(s) >> - Include all columns << - Exclude all columns < - Exclude only the selected column(s)
Displayed Items	Specifies the data block columns that will be displayed.
Item Type	Specifies the type of the item (text box, check box, list, etc.)

Screen



Continued on the next page

Layout Wizard, Continued

Items Page --
Layout Step #3

Component	Usage
Name	The name of the item to be displayed.
Prompt	Specifies the boilerplate text that Oracle Forms creates for the item.
Width	Specifies the width of the item's bounding box.
Height	Specifies the height of the item's bounding box.

Screen

Enter a prompt, width, and height for each item. The units for item width and height are Points.

Name	Prompt	Width
SWRIDEN_PIDM	Swriden Pidm	68
SWRIDEN_ID	Swriden Id	68
SWRIDEN_LAST_NAME	Swriden Last Name	176
SWRIDEN_FIRST_NAME	Swriden First Name	108
SWRIDEN_MI	Swriden Mi	108

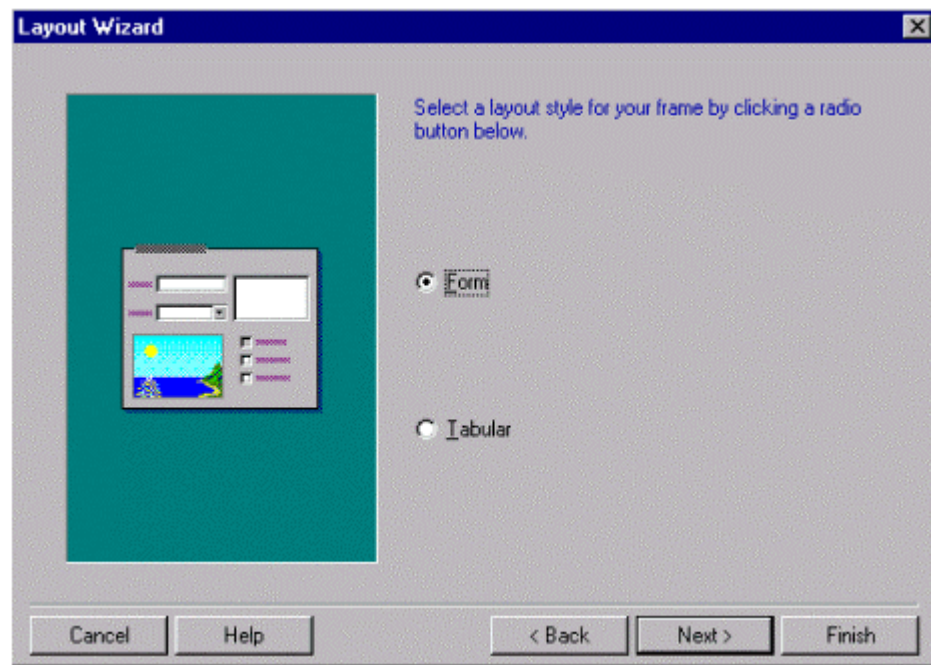
Continued on the next page

Layout Wizard, Continued

Style Page --
Layout Step #4

Component	Usage
Style	Form - Items in a two-column format, with boilerplate text labels positioned to the left of each item. Tabular - All items next to each other across a single row, with boilerplate labels above each item (similar to a spreadsheet).

Screen



Continued on the next page

Layout Wizard, Continued

Row Page --
Layout Step #5

Component	Usage
Frame Title	Title of the data block to be displayed.
Records Displayed	Specifies the number of records the data block displays.
Distance Between Records	Specifies the amount of space between each item. Based on coordinate system unit of the form.
Display Scrollbar	Specifies whether Oracle Forms should create a scrollbar for the base table data block.

Screen

Layout Wizard

Enter a title for the frame. Also be sure to specify the number of database records to be displayed in the frame, as well as the distance between each record.

To display a scrollbar in the frame that can be used to scroll through database records, check the 'Display Scrollbar' check box.

Frame Title: Identification

Records Displayed: 1

Distance Between Records: 0

Display Scrollbar

Cancel Help < Back Next > Finish

Finish Page --
Layout Step #6

The Layout Wizard is now complete. Click the Finish button to exit.

Canvases

Canvas and viewport

Think of the viewport as a rectangle positioned on the canvas. The area of the canvas that is within the viewport is what operators see displayed in the window at runtime.

- When the viewport is the same size as the canvas, all of the canvas is visible to the operator.
- When the viewport is smaller than the canvas, only the part of the canvas that is within the viewport is visible
- For a content or toolbar canvas, the viewport is defined by the window in which the canvas is displayed. Changing the size of the window at runtime (for example, by resizing it with the mouse) effectively changes the size of the viewport for that window's content canvas
- For a stacked or tab canvas, the size of the viewport can be specified at design time by setting the **Viewport Width** and **Viewport Height** properties

Canvas properties

For any type of canvas, you can set properties that specify the viewport's point of origin on the canvas; that is, to position the viewport rectangle at a specific location on its canvas. Moving the viewport, like resizing the viewport, changes the part of the canvas that operators see in the window at runtime.

When a content canvas is larger than its viewport (that is, larger than its window), the window can be scrolled to change the position of the viewport on the canvas, thus making a different part of the canvas visible.

Canvas creation methods

There are three ways to create a content canvas in Oracle Forms:


- Create a base table data block in the New Block window. A new canvas will be created if the specified name does not exist.
- Invoke the Layout Editor in a new form. (If there are no canvases in a form when you invoke the Layout Editor, Oracle Forms automatically creates a default canvas for you to work on.)
- Create a canvas in the Object Navigator. You can create a canvas of any type (Content, Stacked, Tab, Toolbar) in the Navigator.

Continued on the next page

Canvases, Continued

Create via Object Navigator

To create a canvas in the Object Navigator:

Step	Action
1	Highlight the Canvases node.
2	Select Navigator → Create or click the  icon. The default canvas is Content.
3	Highlight the canvas name and choose Tools → Property Palette to invoke the property palette.
4	Set the Window property to specify the window in which you want the canvas to be displayed. By default, new canvases are assigned to the first window listed under the Windows node in the Navigator.
5	To display the new canvas in the Layout Editor, double-click the canvases object icon in the Navigator.

Content Canvas

-
- Characteristics**
- Most canvases are content canvases
 - A content canvas is the "base" view that occupies the entire content pane of the window in which it is displayed
 - You must define at least one content canvas for each window you create
 - More than one content canvas can be assigned to the same window at design time, but at runtime, only one of them at a time is displayed in the window
-

Section D: The Layout Editor

Overview

Purpose Although your forms are functional, they probably do not have the appearance you wish. You may want to move items around, add color, change labels, increase or decrease item widths, etc. Within this section, you will be introduced to some basic layout tools that will dramatically improve the look of your forms.

- Objectives** This section will examine the following:
- Moving items with the mouse
 - Resizing objects
 - Resizing the canvas
 - Aligning objects
 - Grouping objects
 - Creating and modifying boilerplate text
 - Adding colors and borders
-

In this section These topics are covered in this section.

Topic	Page
Layout Editor overview	D-2
Moving objects	D-3
Resizing objects	D-4
Resizing the canvas in the Layout Editor	D-5
Aligning objects	D-6
Grouping objects	D-7
Boilerplate text	D-8
The color palette	D-9
Windows	D-10
Compiling your form	D-12
Module Access	D-14
Running your form	D-15
File types	D-16
Forms documentation	D-17

Layout Editor overview

What is the Layout Editor? The Layout Editor is a graphical design facility for creating and arranging interface items and boilerplate text and graphics in a form.

Invoking the Layout Editor

- In the Navigator, double-click the object icon for the canvas-view you want to edit
Or:
- In the Navigator, choose Layout Editor from the popup menu. (In Windows environments, right-click to display the popup menu.)
Or:
- Choose **Tools→Layout Editor**, then indicate the canvas-view you want to work on

You can open more than one Layout Editor at the same time, and can copy and paste objects between Layout Editors as needed.

Closing a Layout Editor To close a Layout Editor, double-click the **Close** box in the upper left corner of the window.

Restrictions You cannot clear, cut, copy, duplicate, or export the following objects in the Layout Editor:

- the canvas object (when the **View→Show Canvas** option is On)
- the view rectangle (when the **View→Show View** option is On)
- a data block scroll bar

Moving objects

Moving an object or objects	<p>To move an object, position the Select tool over it, then click and drag the object to the desired location.</p> <p>To move more than one object at a time, select the objects, then click and drag any selected object to move all of the objects in the selection.</p>
Moving a selection incrementally	<p>To move the current selection incrementally, press the appropriate arrow key:</p> <ul style="list-style-type: none">• [Up], [Down], [Left] or [Right] <p>When grid snap is turned off, the arrow keys move the current selection one pixel at a time in the direction indicated. When grid snap is turned on, the arrow keys move the current selection the distance of one snap point. Snap point increments are defined in the Ruler Settings dialog.</p>
Constraining a move	<p>To constrain a move to be vertical, diagonal, or horizontal, hold down [Constrain] (Shift on most platforms) while dragging the selected object.</p>
Note	<p>If you have used the Layout wizard and have created a frame for a data block, the frame will keep the items in the original order regardless of how you move them.</p>

Resizing objects

Resizing objects You can resize objects with the mouse, or by specifying precise dimensions in the Size Objects dialog.

To resize an object:

Step	Action
1	Select the object you want to resize.
2	<ul style="list-style-type: none">• Resize the object by dragging one of its selection handles. To constrain a resize operation, hold down the Shift key while resizing the selected object. For example, Shift-dragging constrains a rectangle to a square, and an ellipse to a circle. <p>Or:</p> <ul style="list-style-type: none">• Select Arrange→Size Objects to invoke the Size Objects dialog, then set Width and/or Height to Custom, and enter the desired setting. Once you have specified the size of an object, you can apply the same dimensions to other selected objects with the Arrange→Repeat Sizing command.

Making objects the same size

To make objects the same size:

Step	Action
1	Select the objects you want to make the same size.
2	Select Arrange→Size Objects to open the Size Objects dialog.
3	Specify whether you want the selected objects to be the same height and/or width as the smallest selected object, the largest selected object, or an average of all selected objects. If you want to enter a specific width or height value, choose Custom.
4	Choose OK to accept the size parameters and dismiss the dialog.

Resizing the canvas in the Layout Editor

Resizing the canvas

Step	Action
1	Make sure the canvas object is displayed by setting the View→Show Canvas option to On (the default).
2	Scroll the Layout Editor window until the lower right corner of the canvas is visible, then click the right edge or bottom edge of the canvas to select it.
3	When the canvas is selected, a black selection handle is displayed at its lower-right corner.
4	Click and drag the selection handle to resize the canvas as desired.

Aligning objects

Aligning objects

Step	Action
1	Select the objects you wish to align.
2	Select Arrange→Align Objects to display the Alignment Settings dialog, or use the icons on the horizontal toolbar.
3	Specify how you want the selected objects to be aligned. A single object can be aligned to the grid. Multiple selected objects can be aligned to the grid or to each other. You can specify horizontal and/or vertical alignment options.
4	Click OK to accept the settings and dismiss the dialog.

Grouping objects

Groups of objects

You can select multiple objects and create a group. The group can then be manipulated as a single object. For example, you might want to group several objects together and then move the group as a unit, without changing the relative positions of the objects in the group.

Grouping objects

To group objects:

Step	Action
1	Select two or more objects.
2	Select Arrange→Group . The new group is automatically selected, and you can manipulate it as you would a single object.

Ungrouping objects

To ungroup objects:

Step	Action
1	Select the group.
2	Choose Arrange→Ungroup .

Boilerplate text

Boilerplate graphics

Boilerplate graphics are the lines, circles, text, and images that you draw or import onto a canvas-view. Boilerplate graphics are associated with the canvas-view on which you place them; deleting a canvas-view deletes all of its boilerplate objects.

Unlike items, boilerplate graphics are not named objects. Boilerplate objects do not appear in the Object Navigator and their properties are not displayed in the Properties window.

Create boilerplate text

To create boilerplate text:

Step	Action
1	Select the Text tool in the Layout Editor's tool palette.
2	Click in the workspace where you want to place text.
3	Type the desired text.
4	Exit text mode by clicking in the workspace anywhere outside the text object's bounding box.

Edit boilerplate text

To edit boilerplate text:

Step	Action
1	Select the Text tool T in the Layout Editor's tool palette.
2	Click the boilerplate text object you want to edit.
3	Edit the text as desired.
4	Click in the workspace anywhere outside the text object's bounding box to cancel text mode.

The color palette

Color palette toolbar

The toolbar in the Layout Editor contains tools that allow you to easily modify colors for background fill, text, and lines.



Removing Boilerplate Lines

By default, a boilerplate text object has a line around its bounding box. To remove this line, select the boilerplate text object, then choose the No Line option in the Line Color palette.


Windows

Windows

A window is just a container for all the GUI objects that make up the form application. The window is a frame for the application.

A form can include any number of windows. Every new form automatically includes a default window named WINDOW1.

Create additional windows

Step	Action
1	Highlight the Windows node.
2	Select Navigator→Create or click the  icon. The default window is Modeless.
3	Double-click the Window icon to the left of the text name to display its property palette. For each window you create, you must also create at least one content canvas. You can associate the canvas with the window by setting the Primary Canvas Window property.

Display Properties

Property	Usage
X Position	X coordinate of the window
Y Position	Y coordinate of the window
Width	Width of the window
Height	Height of the window
Title	Window title in title bar

Continued on the next page

Windows, Continued

Functional Properties

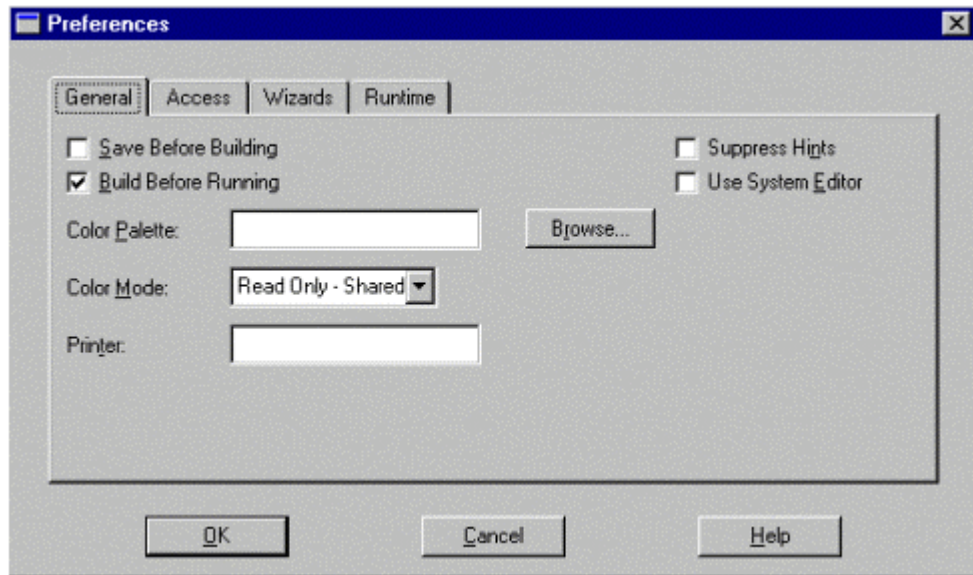
Property	Usage
Primary Canvas	The name of the canvas to display in the window when invoked programmatically.
Show Vertical Toolbar	Determines the presence of a vertical toolbar in a window.
Show Horizontal Toolbar	Determines the presence of a horizontal toolbar in a window.
Modal	Determines whether the window is modal or modeless.
Window Style	Specifies whether the window is a Document window or a Dialog window. Document Style windows are fixed and must remain inside the application frame. Dialog Style windows are free and can be moved outside the application frame.
Icon Filename	Specifies the icon filename used to represent the minimized window.
Hide on Exit	Whether a modeless window closes on exit.

Compiling your form

Compile your form

To compile your form, select **File→Administration→Compile File**. This will create a runtime executable (.fmx extension).

You can choose to always compile the executable before running the form from the Builder. To examine the setting, select **Tools→Preferences**. By default, the option will be set to build (compile) a new executable before running the form.



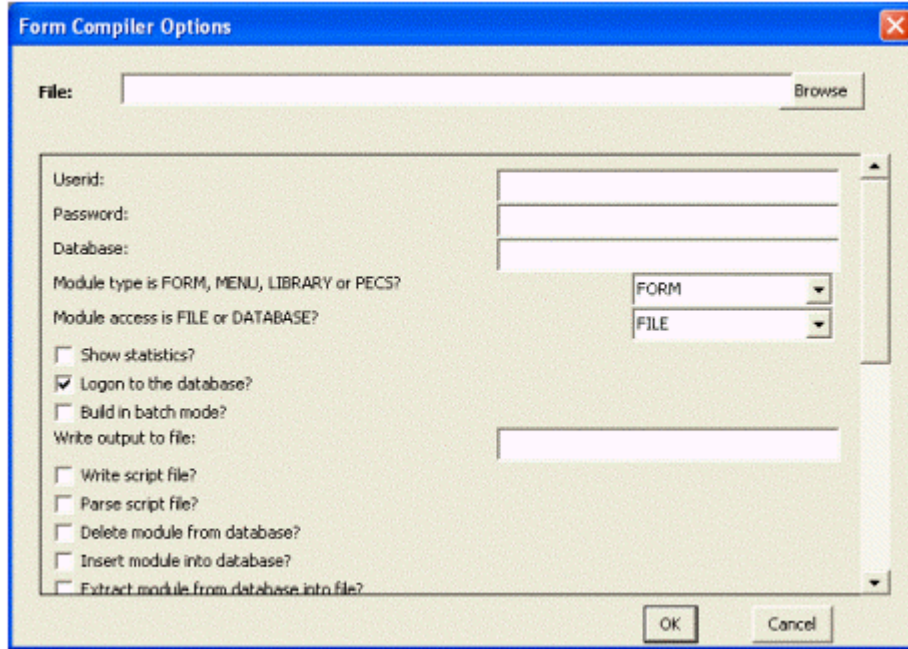
Continued on the next page

Compiling your form, Continued

Compiler

The Forms Compiler is the best compiler to use for Banner Forms.

Screen

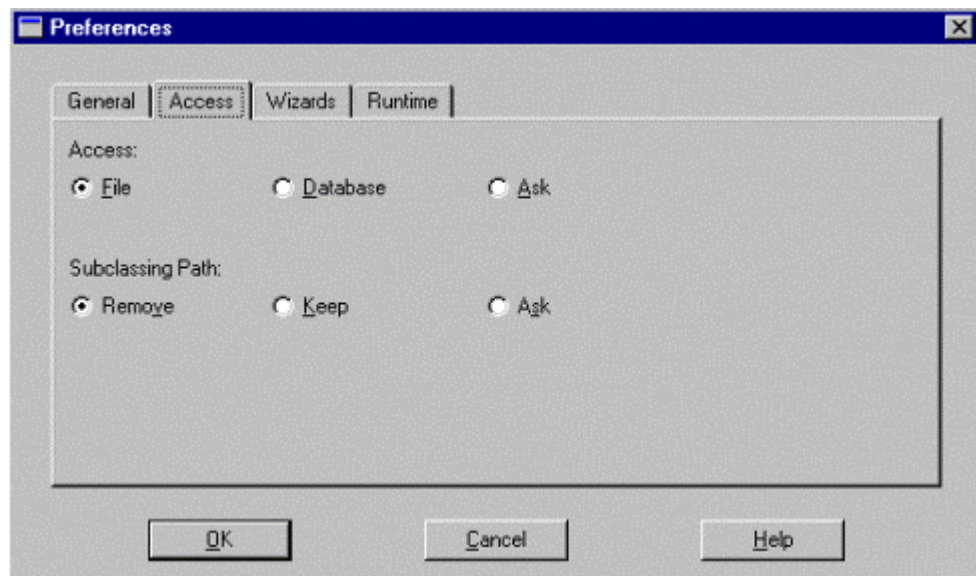


Module Access

Default save settings


You have the option to save forms, menus, and libraries to the database, rather than saving them to a file.

To change the default setting from File to Database, select **Tools→Preferences** (see below). The modules will be saved to Forms tables (whose filenames begin with FRM45). Because of performance issues, and the fact that database space is usually more precious than file space, all of Banner's modules are stored as files.



Running your form

Run your form

Select Program→Run Form→Client/Server or click  on the toolbar. The Runform component of Forms will be executed.

File types

File types

Form, menu, and library modules that you create in the Builder are:

- stored in binary format
- can be saved to files or to the database
- portable across platforms

When you generate a binary module, Oracle Forms creates a platform-specific runfile.

Module	Binary (Design)	Text	Executable Runfile
Form	.FMB	.FMT	.FMX
Menu	.MMB	.MMT	.MMX
PL/SQLLibrary	.PLL	.PLD	.PLX
Object Library	.OLB	.OLT	

Text versions

Text versions of the binary files can be created by selecting **File→Administration→Convert**.

Choose the module type (form, menu, library), the module name, and the conversion direction (binary to text, or text to binary). The text version can then be converted back, if necessary.

Forms documentation

Documentation Although the text versions of the modules can be opened in a word processor and examined, the text version of a module is not intended for documentation purposes. Instead, open the desired form up in the Object Navigator and select **File→Administration→Object List Report**. A .txt version of the form will be created in the same directory as the form.

The Forms documentation cannot be converted into an .fmb.

Section E: Customizing Data Blocks

Overview

Purpose Data block properties may need to be modified after the data block has been created. Within this section, we will take a look at the many data block properties you can set to enhance the functionality of your form.

Objectives This section will examine the following:

- Data block properties
- Setting properties on multiple objects

In this section These topics are covered in this section.

Topic	Page
Data block property categories	E-2
Specifying the default navigation sequence	E-7
Specifying the navigation style for a data block	E-8
Making items navigable and enabled	E-9
Creating a data block manually	E-10
Joining blocks	E-13

Data block property categories

Categories

The data block properties are divided into categories in order to find the properties faster. The categories are:

- General
- Navigation
- Records
- Database
- Advanced Database
- Scrollbar
- Visual Attributes
- Color and Font
- Character Mode
- International

General properties

Property	Description
Name	Name of the data block.
Subclass Information	Specifies the class that has passed its properties to the data block.
Comments	Describes the data block in detail.

Navigation Properties

Property	Description
Navigation Style	Once the user has navigated to the last item, where should the cursor go? Same Record, Next Record, Next Block.
Next/Previous Navigation Data Block	Usually, the next or previous data block is determined by the order of the data blocks within the Object Navigator, which is also the order that the form posts changes to the database. If this needs to be overridden, then specify the name of the data block within these properties.

Exercise 4

In the ID data block, allow the user to automatically navigate to the next record when tabbing, by modifying the Navigation Style on the data block level.

Continued on the next page

Data block property categories, Continued

Records Properties

Property	Description
Current Record Visual Attribute Group	Allows you to associate a visual attribute with the current record. Used to highlight the current record.
Query Array Size	Specifies the maximum number of records that Form Builder should fetch from the database at one time.
Number of Records Buffered	Specifies the minimum number of records buffered in memory during a query in the data block.
Number of Records Displayed	Number of records to be displayed for the data block on the canvas.
Query All Records	Specifies whether all the records matching the query criteria should be fetched into the data block when a query is executed.
Record Orientation	Whether the records are oriented vertically or horizontally. The default is vertical.
Single Record	Specifies that the control data block always should contain one record. Note: This differs from the number of records displayed in a data block.

Continued on the next page

Data block property categories, Continued

Database Properties

Property	Description
Database Data Block	Specifies that the data block is based on any of the following data source types: <i>Table</i> , <i>Procedure</i> , <i>Transactional Trigger</i> , or <i>Sub-Query</i> .
Enforce Primary Key	Indicates that any record inserted or updated in the data block must have a unique key in order to avoid committing duplicate rows to the data block's base table.
Delete Allowed, Insert Allowed, Update Allowed, Query Allowed	Specifies whether records can be deleted, inserted, updated, or queried within the data block. The default for these properties is <i>Yes</i> .
Query Data Source Type	Specifies the query data source type for the data block. A query data source type can be a <i>Table</i> , <i>Procedure</i> , <i>Transactional Trigger</i> , or <i>FROM clause query</i> .
Query Data Source Name	Specifies the name of the data block's query data source.
Query Data Source Columns	Specifies the names and datatypes of the columns associated with the data block's query data source. Only used when Query Data Source Type property is set to <i>Table</i> , <i>Sub-query</i> , or <i>Procedure</i> .
WHERE Clause/ ORDER BY Clause	These clauses are automatically appended to the <i>SELECT</i> statement that Oracle Forms constructs and issues whenever the operator or the application executes a query in the data block.
Optimizer Hint	Specifies a hint string that Oracle Forms passes on to the RDBMS optimizer when constructing queries. Using the optimizer can improve the performance of database transactions.
Update Changed Columns Only	When queried records have been marked as updates, specifies that only columns whose values were actually changed should be included in the <i>SQL UPDATE</i> statement that is sent to the database during a <i>COMMIT</i> . By default, Update Changed Columns Only is set to <i>No</i> , and all columns are included in the <i>UPDATE</i> statement.

Continued on the next page

Data block property categories, Continued

Exercise 5 In the SWRIDEN block, increase the number of records displayed to 5

Exercise 6 Ensure that the records retrieved in the SWRIDEN data block are current (the swriden_change_ind is NULL) by adding a WHERE clause.

Exercise 7 Sort the records in the SWRIDEN data block by last name.

Exercise 8 Make sure the user is unable to insert, update, or delete records in the Identification data block.

Continued on the next page

Data block property categories, Continued

Database Properties (cont.)

Property	Description
Enforce Column Security	Specifies when Oracle Forms should enforce update privileges on a column-by-column basis for the data block's base table. If an operator does not have update privileges on a particular column in the base table, Oracle Forms makes the corresponding item non-updateable for this operator only, by turning off the Update Allowed item property at form startup.
Maximum Query Time	Provides the option to abort a query when the elapsed time of the query exceeds the value of this property.
Maximum Records Fetched	Specifies the number of records fetched when running a query before the query is aborted.
Key Mode/Locking Mode Triggers	Default settings are appropriate for an Oracle database, but may need to be modified for other database types.

Scrollbar Properties

Property	Description
Show Scroll Bar	Specifies whether the scroll bar will be associated with the data block.

Visual Attribute Properties

Property	Description
Visual Attribute Group	Specifies Visual Attribute for the data block.

Specifying the default navigation sequence

Default navigation sequence

The default navigation sequence for items in a data block or for data blocks in a form is specified by arranging items and data blocks in the desired sequence in the Object Navigator.

Defining the default sequence

To define the default navigation sequence:

Step	Action
1	In the Object Navigator, arrange all of the data blocks in the form according to the desired navigation sequence.
2	Arrange all of the items in each data block according to the desired navigation sequence.
3	To move a data block or item, click and drag it to the desired location under the Data Blocks or Items nodes. The sequence of data blocks within a form also defines the default commit sequence; that is, the order in which Oracle Forms validates each data block during a save/commit operation.

Specifying the navigation style for a data block

Navigation Style property You can alter the default navigation sequence by setting the **Navigation Style** data block property. Navigation Style specifies how a Next Item or Previous Item operation is processed from the last navigable item and the first navigable item in the data block, respectively.

The following settings are valid for Navigation Style:

Setting	Description
Same Record	The default navigation style. A Next Item operation from a data block's last item moves the input focus to the first navigable item in the data block, in that same record.
Change Record	A Next Item operation from a data block's last item moves the input focus to the first navigable item in the data block, in the next record. If the current record is the last record in the data block and there is no open query, Oracle Forms creates a new record. If there is an open query in the data block (which means the data block contains queried records), Oracle Forms retrieves additional records as needed.
Change Data Block	A Next Item operation from a data block's last item moves the input focus to the first navigable item in the first record of the next data block. Similarly, a Previous Item operation from the first item in a block moves the input focus to the last item in the current record of the previous data block. Note that the Next Navigation Block and Previous Navigation Block properties can be set to redefine a data block's next or previous data block.

Making items navigable and enabled

Navigable items A navigable item is one that operators can navigate to with the **[Tab]** key during default navigation, or that Oracle Forms can navigate to by executing a navigational built-in procedure.

The following table shows the valid settings for these properties, and describes the resulting navigational behavior.


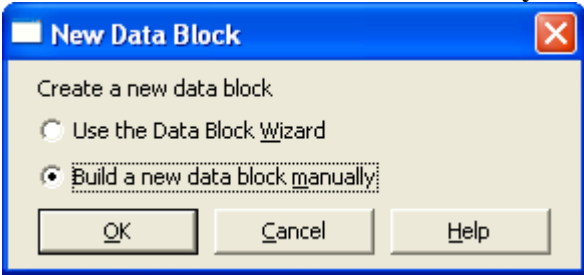

Keyboard navigable	Enabled	Resulting navigation behavior
Yes	Yes	Item is navigable, and Oracle Forms can move the input focus to the item during default navigation. Item is displayed normally.
No	Yes	Item is non-navigable. During default navigation, Oracle Forms skips over the item to the next navigable item in the sequence. Item is displayed normally, and operators can navigate to and manipulate the item with the mouse.
No	No	Item is non-navigable, and is displayed with reduced contrast to indicate that it is unavailable for input or mouse manipulation.

Creating a data block manually

Key Points

- Create the data block and assign the base table in the property palette
- Create items and associate column names
- Assign items to a canvas

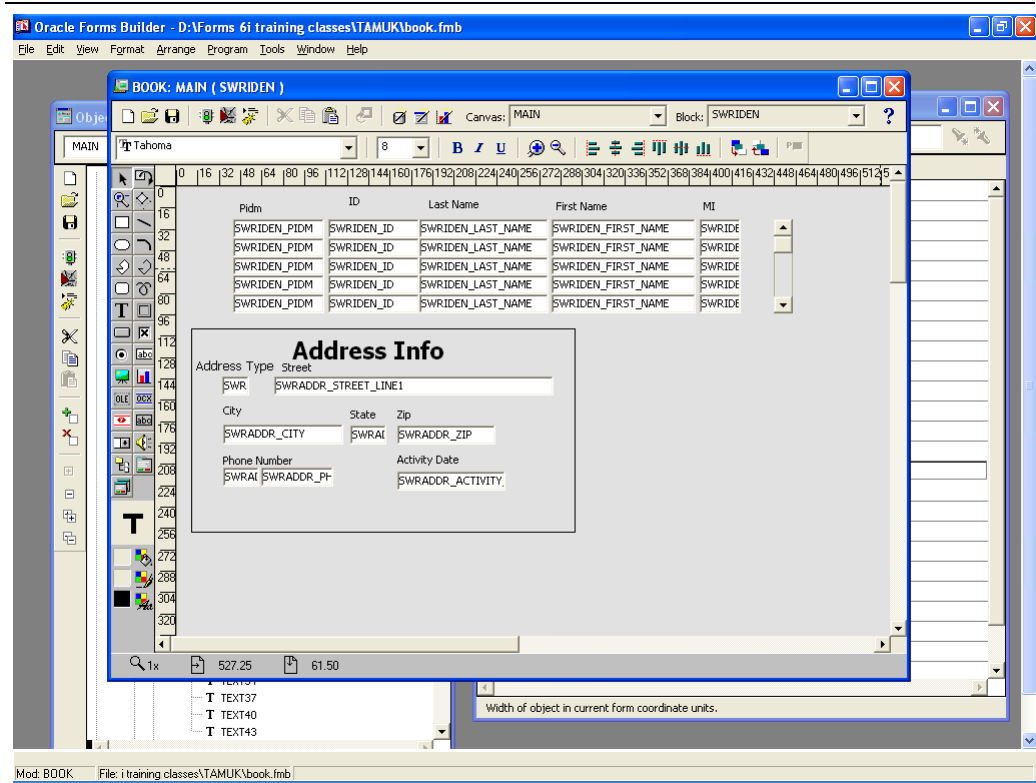
Process

Step	Action
1	Highlight Data Blocks.
2	Click the  icon.
3	Select Build a new data block manually . 
4	BLOCK# appears. Rename the block <i>SWRADDR</i> .
5	Open the property palette.
6	Set Query Data Source Name to <i>SWRADDR</i> .
7	Highlight items.
8	Click the  icon.
9	ITEM# appears. Go to the property palette and rename it to <i>swraddr atyp code</i> .
10	Set column name to <i>swraddr_atyp_code</i> .
11	Set canvas to <i>canvas1</i> .
12	Repeat the above steps so that your layout editor looks like this and includes the following columns: <ul style="list-style-type: none"> • <i>swraddr_phone_area</i> • <i>swraddr_phone_number</i> • <i>swraddr_street_line1</i> • <i>swraddr_city</i> • <i>swraddr_stat_code</i> • <i>swraddr_zip</i>

Continued on the next page

Creating a data block manually, Continued

Canvas 1



Continued on the next page

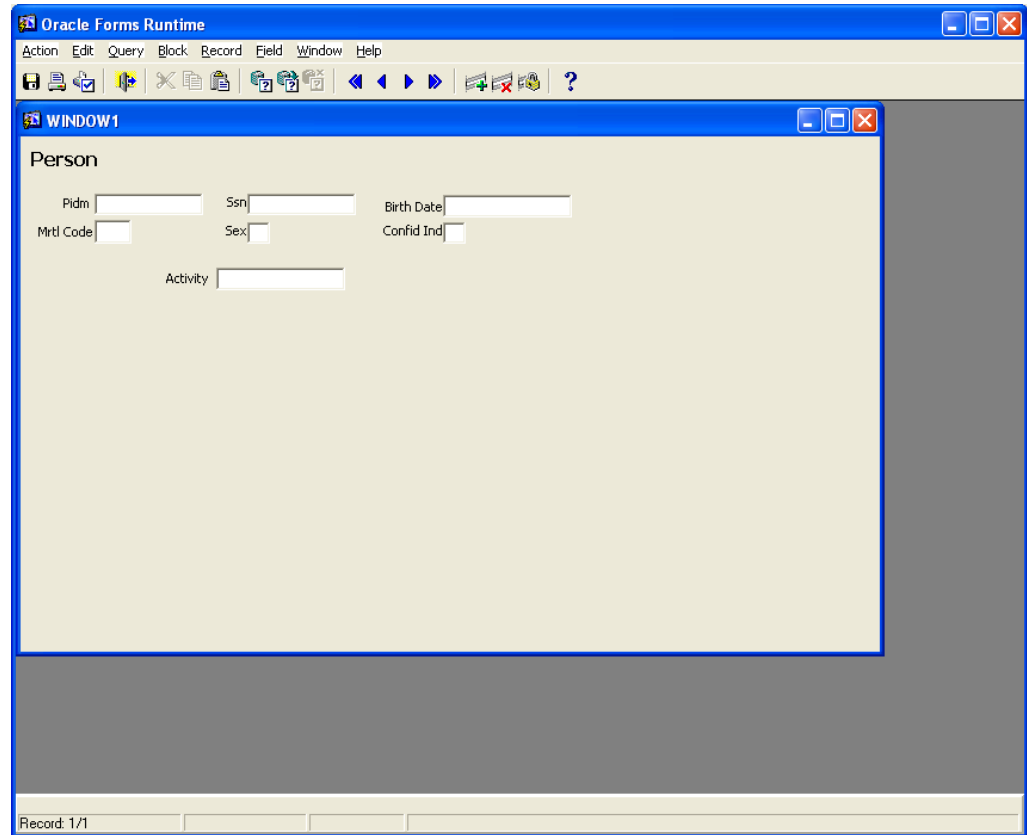
Creating a data block manually, Continued

Exercise 9

In the SWAIDEN form, create a data block based on the SWBPERS table.

- Include all columns
- Display the records on a new canvas called Canvas2

SWAIDEN – Canvas 2



The screenshot displays the Oracle Forms Runtime environment. The main window, titled 'WINDOW1', contains a form titled 'Person'. The form includes the following fields:

- Pidm:
- Ssn:
- Birth Date:
- Mrtl Code:
- Sex:
- Confid Ind:
- Activity:

The status bar at the bottom of the window indicates 'Record: 1/1'.

Joining blocks

Specifying a join condition The join specified is similar to an SQL join, except the specification is `Data_Block.item = Data_Block.item`, rather than `Table.column = Table.column`.

Join the address block and the person block together. The database column must be on the right and the current item (prefixed by :) is on the left.

Exercise 10 Join both the swraddr and swbpers blocks to the swriden pidm.

Section F: Text and Display Items

Overview

Purpose All items, just like data blocks, have individual property sheets which allow properties to be changed to enhance the functionality of the form. In this section, we will focus on text item properties.

Objectives This section will examine the following:

- Create a new text item
- Examine and modify text item properties
- Create a display item

In this section These topics are covered in this section.


Topic	Page
Creating a text item	F-2
General Properties	F-3
Physical Properties	F-4
Records Properties	F-5
Visual Attribute Properties	F-6
Color Properties	F-7
Font Properties	F-8
Prompt Properties	F-9
Data Properties	F-10
Initial values	F-11
Format masks	F-13
Navigation Properties	F-16
Database Properties	F-17
Functional Properties	F-18
Creating multi-line text items	F-19
Help Properties	F-20
Display items	F-21

Creating a text item


What is a text item?

A text item is an interface control that displays operator-enterable text.

Creating via the Navigator

Step	Action
1	Highlight the Items node.
2	Select Navigator → Create or click the  icon. The default item type is a Text Item.
3	Double-click the Text icon to the left of the text name to display its property sheet.

Creating via the Layout Editor

Step	Action
1	Click the Text Item  icon on the toolbar.
2	Click an area on the canvas to create the text item.
3	Double-click the text item to display its property sheet.

Exercise 11

In the SPRIDEN data block, set the canvas to null within the pidm item's property palette.

General Properties

General
properties

Property	Description
Item Type	Specifies the type of item.

Physical Properties

Physical Properties

Property	Description
Visible	Determines whether an item that is assigned to a canvas is shown or hidden at runtime.
Canvas	Specifies the canvas on which you want the item to be displayed.
Tab Page	Specifies the tab page on which the item is located. Tab Canvases only.
X Position	Specifies the position of the item's upper left corner relative to the upper left corner of the item's canvas. The values you specify are interpreted in the current form coordinate units (character cells, centimeters, inches, pixels, or points), as specified by the Coordinate System form property.
Y Position	Specifies the position of the item's upper left corner relative to the upper left corner of the item's canvas. The values you specify are interpreted in the current form coordinate units (character cells, centimeters, inches, pixels, or points), as specified by the Coordinate System form property.
Width	Sets the width of the item.
Height	Sets the height of the item.
Bevel	Specifies the appearance of the object border: either RAISED, LOWERED, INSET, OUTSET, PLAIN, or NONE.
Rendered	Specifies that the item is to be displayed as a rendered object when it does not have focus.
Show Vertical Scroll Bar	Specifies that a vertical scroll bar is to appear on the side of a canvas or window.

Records Properties

Records Properties

Property	Description
Current Record Visual Attribute Group	Specifies the named visual attribute used when an item is part of the current record.
Distance Between Records	Specifies the amount of space between instances of the item when the item is in a multi-record data block.
Number of Items Displayed	Specifies the number of item instances displayed for the item when the item is in a multi-record data block.

Visual Attribute Properties

Visual Attribute Properties

Property	Description
Visual Attribute Group	Specifies how the object's individual attribute settings (Font Name, Background Color, Fill Pattern, etc.) are derived.
Prompt Visual Attribute Group	Specifies the named visual attribute that should be applied to the prompt at runtime.

Color Properties

Color Properties

Property	Description
Foreground Color	Specifies the foreground color for the item.
Background Color	Specifies the background color for the item.
Fill Pattern	Specifies the pattern for the item.

Font Properties

Font Properties

Property	Description
Font Name	Specifies the name of the font to be used for the item.
Font Size, Font Style, Font Spacing, Font Weight	Specifies attributes for the font chosen.

Prompt Properties


Prompt Properties

Property	Description
Prompt	Specifies the text label that displays for an item.
Prompt Display Style	Specifies how the prompt is displayed: either First Record, Hidden, or All Records.
Prompt Justification	Specifies justification of the prompt: either Left, Right, Center, Start, or End.
Prompt Attachment Edge	Specifies which edge the prompt should be attached to: either Start, End, Top, or Bottom.
Prompt Alignment	Specifies how the prompt is aligned along the item's edge: either Start, End, or Center.
Prompt Attachment Offset	Specifies the distance between the item and its prompt.
Prompt Alignment Offset	Specifies the prompt's alignment offset.
Prompt Reading Order	Specifies the prompt's reading order: either Default, Left to Right, or Right to Left.

Prompts and Boilerplate

Prompts specify the text label that is associated with an item. When you move an item around in the Layout Editor, the Boilerplate Label will follow.

To associate boilerplate text with an item

Step	Action
1	Open the Layout Editor.
2	Select the item and the boilerplate text you want as the item's prompt.
3	Click the Associate Prompt  icon.

Data Properties

Data Properties

Property	Description
Data Type	Specifies what kinds of values Oracle Forms allows as input and how Oracle Forms displays those values (Examples: NUMBER, CHAR).
Maximum Length	Specifies the maximum length of the data value that can be stored in the item.
Fixed Length	When set to No, Fixed Length specifies that the item should be considered valid only when it contains the maximum number of characters allowed.
Initial Value	Specifies the default value that Oracle Forms should assign to the item whenever a record is created.
Required	When a new record is being entered, specifies that the item is invalid when its value is NULL.
Format Mask	Specifies the display format and input accepted for data in text items.
Lowest / Highest Allowed Value	Determines the maximum value or minimum value, inclusive, that Oracle Forms allows in the text item.
Copy Value From Item	Specifies the source of the value that Oracle Forms uses to populate the item. When you define a master-detail relation, Oracle Forms sets this property automatically on the foreign key item(s) in the detail data block.
Synchronize with Item	Specifies the name of the item from which the current item should derive its value. Setting this property synchronizes the values of the two items, so that they effectively mirror each other. Used for items within the same data block.

Initial values

Initial values

You can specify initial values for items by setting the Initial Value item property.

The value you specify must be compatible with the item's data type. For example, the initial value for a text item having a data type of DATE must be a value that can be displayed in a valid date format.

The initial value can be any of the following:

- Raw Value
100, 'Y'
- System Variables
 - Operating system current date/time:
\$\$DATE\$\$ DD-MON-YY
\$\$DATETIME\$\$ DD-MON-YYYY hh:mi[:ss]
\$\$TIME\$\$ Hh:mi[:ss]
 - Current database date/time:
\$\$DBDATE\$\$ DD-MON-YY
\$\$DBDATETIME\$\$ DD-MON-YYYY hh:mi[:ss]
\$\$DBTIME\$\$ Hh:mi[:ss]
- Form Item Value
:Address.atyp_code
- Global Variable
:GLOBAL.pidm
- Form Parameter
:PARAMETER.pidm
- Sequence
:SEQUENCE.pidm_sequence.NEXTVAL

Continued on the next page

Initial values, Continued

Exercise 12

In both the Address and Person data blocks:

- Alter the activity dates so that it initializes to the current database date for a new record. Try using an intersection to set the property for both items at the same time
- For both data blocks, do not allow the activity dates to be updated or inserted by the user
- Set the bubble help to **Activity Date**
- Set the format mask so that the date appears like the following: 01-JAN-1998
- Remember to alter the maximum length to allocate for the increase in characters

Format masks

Format masks for number values

Symbol	Description
9	Represents one numeric character. The number of 9's determines how many digits the text item can display.
0	Displays leading zeroes when present.
\$	Prefix number with a dollar sign.
B	Displays preceding zeroes as blank spaces.
MI	Displays “-“ after a negative value.
PR	Displays negative values in <angle brackets>.
, (comma)	Displays a comma in this position as required.
. (decimal)	Displays a decimal in this position.

Examples of Number Format Masks

You may need to increase the Maximum Length and Query Length properties to take into account the format mask.

Format mask	Number entered	Result
999	223.4	223
\$9,999.99	3445.34	\$3,445.34
99.99”%”	66.17	66.17%
999“-“ 99“-“ “9999	123456789	123-45-6789

Continued on the next page

Format masks, Continued

Format masks for date values

Symbol	Description
9	Represents one numeric character. The number of 9's determines the number of digits the text item can display.
MM	Month (1-12).
MON	Name of month, 3-letter abbreviation.
MONTH	Name of month, padded with blank spaces to length of 9 characters.
DD	Day of month (1-31).
DY	Name of day, 3-letter abbreviation.
DAY	Name of day, padded with blanks to length of 9 characters.
YYYY, YYY, YY, or Y	4,3,2 or 1-digit year.
HH or HH12	Hour of day (1-12).
HH24	Hour of day (1-24).
MI	Minute (0-59).
SS	Second (0-59).
AM or A.M. or PM or P.M.	Meridian indicator.
TH	Ordinal number (e.g. "DDTH" for "15TH")
SP	Spelled out number (e.g. "DDSP" for "FIFTEEN")
SPTH or THSP	Spelled out ordinal number (e.g., "DDSPTH" for "FIFTEENTH")
FM	Prefix used with symbols such as MONTH and DAY to suppress padding added by these symbols.

Exercise 13

In the Person data block, set the format mask for `swbpers_birth_date` so that it appears like the following: 01-JAN-1998. Set the bubble help to Birth Date.

Continued on the next page

Format masks, Continued

Exercise 14 In the Person data block, set the format mask for SSN so that it appears like the following at runtime: 123-45-6789. Set Fixed Length to Yes.

Exercise 15 In the Address data block, set the format mask for the phone number so that it appears like the following at runtime: 555-1212. Set Fixed Length to Yes.

Navigation Properties

Navigation properties

Property	Description
Keyboard Navigable	Determines whether the operator or the application can place the input focus in the item during default navigation.
Next Navigation Item	<p>Specifies the name of the item that is defined as the "next navigation item" with respect to this current item.</p> <p>By default, the next navigation item is the item with the next higher sequence as indicated by the order of items in the Object Navigator. However, you can set this property to redefine the "next item" for navigation purposes.</p>
Previous Navigation Item	Specifies the name of the item that is defined as the "previous navigation item" with respect to this current item.

Database Properties

Database properties

Property	Description
Database Item	Determines if the item value is stored in the data block base table.
Column Name	Specifies that an item corresponds to a column in the table associated with the data block.
Primary Key	Set automatically for Oracle. Indicates that the item is a base table item in a base table data block and that it corresponds to a primary key column in the base table.
Insert Allowed	Determines whether the item allows values to be inserted.
Query Only	Specifies that the item can be queried but not included in an INSERT or an UPDATE statement.
Query Allowed	Determines whether the item can be queried.
Query Length	Determines the maximum length of a query expression for the item. Should be at least as long as the Maximum Length value.
Case Insensitive Query	Determines whether case should be a factor when queries are performed.
Update Allowed	Determines whether the item can be updated.
Update Only if Null	Determines whether the item should only be updated if the value is null for the record.
Lock Record	Determines whether the record is locked when the item is changed; this property is only relevant to non-base table items.

Functional Properties

Functional properties

Property	Description
Enabled	Determines whether operators can navigate to an item and manipulate it with the mouse.
Justification	Determines the text justification of the value within the item.
Multi-Line	Determines whether the text item is a single-line or multi-line editing region.
Wrap Style	Specifies how text is displayed when a line of text exceeds the width of a text item or editor window, either None, Character, or Word.
Case Restriction	Automatically converts the case of the user's input to either Upper, Lower, or Mixed.
Conceal Data	Hides characters that the operator types into the text item. This setting is typically used for password protection.
Keep Cursor Position	Specifies that the cursor position be the same upon re-entering the text item as when last exited.
Auto Skip	Moves the cursor to the next navigable item when adding or changing data in the last character of the current item. The last character is defined by the Maximum Length property. Used in conjunction with the Fixed Length property.
Popup Menu	Specifies the popup menu to display with the item.

Creating multi-line text items

Multi-line text items

Multi-line text items are used for displaying large columns, such as comments.

Step	Action
1	In the Navigator, select the desired text item.
2	In the Properties window, set the Multi-line property to be True .
3	Change the Wrap Style property to either None , Character or Word .
4	(Optional) Increase the item Height , so that multiple lines on the form can be viewed at the same time.

Help Properties

Help properties

Property	Description
Hint	The message to be displayed in the console for the item. During runtime, select Help→Help to display the hint.
Automatic Hint	Determines if the help text specified by the item property, Hint, is displayed automatically when the cursor enters the item.
Tooltip	Specifies the help text that should appear in a small box beneath the item when the mouse enters the item. (Bubble Help)
Tooltip Visual Attribute Group	Specifies the named visual attribute that should be applied to the tooltip at runtime.

Display items

Display items Display items are similar to text items with the exception that display items only store and display fetched or assigned values. Operators cannot navigate to display items or edit display item values.


Display Item Property Sheet Display items have fewer properties than text items, because by definition the operator cannot manipulate them.

One advantage to using a display item is that it requires less memory than a text item, since there are fewer properties for Oracle Forms to keep track of.

Create via Object Navigator To create a display item from the Object Navigator:

Step	Action
1	Select the data block where you want to insert the display item, then select the Items node, and then choose Navigator→Create .
2	In the Properties window, set the Item Type property to Display Item .

Create via Layout Editor To create a display item from the Layout Editor:

Step	Action
1	Click the Display Item  icon on the toolbar.
2	Click the canvas to place the display item on the canvas.
3	Resize the display item if needed.
4	If needed, change the data block to which the display item is assigned using the data block pop-list.
5	Double-click the check box to view the Property Sheet.

- Exercise 16** In the Address data block, create a new display item named atyp_desc.
- Ensure that atyp_desc is not a base table item
 - Assign the item to Canvas1
 - Set the bubble help to **Address Description**
 - The display item will be populated by a trigger in a later exercise
 - On the canvas, place it to the right of the swraddr_atyp_code

Section G: Additional Items

Overview

Purpose Although text items alone would allow the user to manipulate data in the database, a form can be enhanced by adding GUI items, such as check boxes, radio buttons, list items, calculated items, hierarchical tree items, and push buttons.

- Objectives** This section will examine the following:
- Creating checkboxes
 - Creating radio buttons
 - Creating list items
 - Creating calculated items
 - Creating hierarchical tree items
 - Creating push buttons
-

In this section These topics are covered in this section.

Topic	Page
Defining check boxes	G-2
Converting a text item	G-4
Defining list items	G-5
Creating and defining radio groups	G-8
Creating and defining buttons	G-11

Defining check boxes

Check boxes

A check box is a two-state control that indicates whether a certain condition or value is on or off, true or false.

Operators toggle the state of a check box by clicking it with the mouse, or by navigating to the check box and pressing [Select].

In Enter Query mode, the operator can exclude a check box as query criteria by setting the check box state to undefined. When a check box is in the undefined state, it appears disabled. Note that the undefined state is valid only in Enter Query mode.


Creating via Object Navigator

To create a check box from the Object Navigator:

Step	Action
1	Select the data block where you want to insert the check box, select the Items node, and choose Navigator→Create .
2	In the Properties Palette, set the Item Type property to Check Box .
3	Create a label for the check box by entering the desired text in the Label property field.
4	Specify the values you want the check box to display as “checked” and “unchecked” in the Value When Checked and Value When Unchecked property fields.

Creating via Layout Editor

To create a check box from the Layout Editor:

Step	Action
1	Click the Check Box icon  on the toolbar.
2	Click the canvas to place the checkbox on the canvas.
3	Resize the check box if needed.
4	If needed, change the data block to which the check box is assigned using the data block pop-list.
5	Double-click the check box to view the Property Palette.

Continued on the next page

Defining check boxes, Continued

Check Box Properties

Property	Usage
Access Key	Specifies the character that will be the access key, allowing the operator to select or execute an item by pressing a key combination, such as Alt-C.
Label	Specifies the text label that displays for a check box.
Mouse Navigate	Specifies whether Oracle Forms should perform navigation to the check box when the operator activates the item with a mouse.
Value When Checked	Specifies the value that is stored in the table that should indicate a checked box. The value must be compatible with the datatype.
Value When Unchecked	Specifies the value that is stored in the table that should indicate an unchecked box. The value must be compatible with the datatype.
Check Box Mapping of Other Values	Determines how to handle any value that is not one of the values represented by the checked or unchecked states. Valid choices are NOT ALLOWED, CHECKED, and UNCHECKED.

Converting a text item

Process

Step	Action
1	Highlight the item in the Object Navigator.
2	Right-click to open the Property Palette.
3	Under item type , select <i>check box</i> .

Exercise 17

In the Person data block, convert the swbpers_confid_ind text item to a check box.

- Set the checked state to represent the base table value of Y and the unchecked state to represent N
- Ensure that new records are automatically assigned the value N
- Allow only those records with swbpers_confid_ind values of Y or N to display
- Resize the checkbox appropriately

Defining list items

List items

A list item displays a predefined set of choices that

- are mutually exclusive
- can be displayed as either a poplist, text list, or combo box

List item	Description
Poplist	Appears initially as a single field (similar to a text item field). When the operator selects the list icon, a list of available choices appears.
Text List	Appears as a rectangular box which displays a fixed number of values. When the text list contains values that cannot be displayed (due to the displayable area of the item), a vertical scroll bar appears, allowing the operator to view and select undisplayed values.
Combo Box	Combines the features found in list and text items. Unlike the poplist or the text list style list items, the combo box style list item will display fixed values and accept one operator-entered value. The combo box list item appears as an empty box with an icon to the right. The user can enter text directly into the combo field or click the list icon to display a list of available values.

List Item Properties

Property	Usage
Access Key	Specifies the character that will be the access key, allowing the operator to select or execute an item by pressing a key combination, such as Alt-C.
List Style	Choice of Poplist, T-List, or Combo Box
Mapping of Other Values	Determines how to handle any value that is not one of the values represented by the list elements.
Mouse Navigate	Specifies whether Oracle Forms should perform navigation to the list item when the operator activates the item with a mouse.
Elements in List	Opens List Item Elements window

Continued on the next page

Defining list items, Continued

Elements in List Properties

Property	Usage
List Elements	Specifies the text that appears in the list to the operator.
List Item Value	Specifies the table value that should be associated with the list element.


Defining from Object Navigator

To define a list item from the Object Navigator:

Step	Action
1	Select the data block where you want to insert the list item, then select the Items node, and then choose Navigator→Create .
2	In the Properties Palette, set the Item Type property to List Item .
3	Specify the display style for the list by setting the List Style property to Poplist , Text List , or Combo Box .
4	Double-click the Elements in List property to display the List Items Elements dialog, then enter the List Elements exactly as you want them to appear in the list item at runtime.
5	Associate a value with each List Element by entering the desired value in the List Item Value field, then choose OK .

Defining from Layout Editor

To define a list item from the Layout Editor:

Step	Action
1	Click the List Item icon  on the toolbar.
2	Click the canvas to place the list item on the canvas.
3	Resize the list item if necessary.
4	If needed, change the data block to which the list item is assigned using the data block poplist.
5	Double-click the list item to display the Property Palette.
6	Specify the display style for the list by setting the List Style property to Poplist , Text List , or Combo Box .
7	Double-click the List Elements property to display the List Items Elements dialog, then enter the List Elements exactly as you want them to appear in the list item at runtime.
8	Associate a value with each List Element by entering the desired value in the List Item Value field, then choose OK .

Continued on the next page

Defining list items, Continued

Exercise 18

In the Person data block, convert the `swbpers_mrtl_code` text item to a pop-list list item.

- Add list elements of Single, Married, Widowed, and Divorced to represent database values of S, M, W, and D
- Display any other values as Single
- Ensure that new records display the default value Single
- Resize the list item to see your choices at runtime

Creating and defining radio groups

Radio groups

A radio group is an interface control that displays a fixed number of mutually exclusive options. Each option is represented by an individual radio button.

A radio group can include any number of radio buttons. Radio buttons can be sized, positioned, and formatted independently of each other.

Radio group properties

Radio Group Property	Usage
Mapping of Other Values	Determines how to handle any value that is not one of the values represented by the radio buttons.
Access Key	Specifies the character that will be the access key, allowing the operator to select or execute an item by pressing a key combination, such as Alt-C.
Mouse Navigate	Specifies whether Oracle Forms should perform navigation to the list item when the operator activates the item with a mouse.

Radio button properties

Radio Button Property	Usage
Name	Specifies the name of the individual radio button.
Access Key	Specifies the character that will be the access key, allowing the operator to select this button by pressing a key combination, such as Alt-C.
Label	Specifies the text that should appear next to the radio button.
Radio Button Value	Specifies the table value that should be associated with the radio button.

Continued on the next page

Creating and defining radio groups, Continued

Define via Object Navigator

To define a radio group from the Object Navigator:

Step	Action
1	Select the data block where you want to insert the radio group, then select the Items node, and then choose Navigator→Create .
2	In the Property Palette, set the Item Type property to Radio Group .
3	Create the desired number of radio buttons.
4	In the property palette, create a label for the radio button by entering the desired text in the Label property field.
5	Enter a value for the currently selected radio button in the Radio Button Value property field. The values you assign to each radio button must be compatible with the datatype for the radio group.
6	Specify the display properties of the currently selected radio button.
7	Specify how you want the radio group to handle fetched or assigned values that are not one of the value associated with a specific radio button. To do so, set the Mapping of Other Values property for the radio group.
8	Set an initial value

Exercise 19

In the Person data block, convert the swbpers_sex text item into a radio group.


- Add radio buttons for Male, Female and Other to represent the database values of M, F and O
- Define access keys of M for male, F for female, and O for Other
- Define a default value of F for all new records

Continued on the next page

Creating and defining radio groups, Continued

Define via
Layout Editor

To define a radio button from the Layout Editor:

Step	Action
1	Click the Radio Button icon  on the toolbar.
2	Click the canvas to place the radio button on the canvas.
3	If a radio group exists, you will be asked if the radio button you are creating should belong to an existing radio group. If none exists, it will create a radio group for you.
4	Resize the radio button if necessary.
5	If needed, change the data block to which the radio button is assigned using the data block poplist.
6	Double-click the radio button to display the Property Palette.
7	In the property palette, create a label for the radio button by entering the desired text in the Label property field.
8	Enter a value for the currently selected radio button in the Radio Button Value property field. The values you assign to each radio button must be compatible with the datatype for the radio group.
9	Specify the display properties of the currently selected radio button.
10	Specify how you want the radio group to handle fetched or assigned values that are not one of the value associated with a specific radio button. To do so, set the Mapping of Other Values property for the radio group.

Creating and defining buttons

Buttons

Buttons are interface items that operators select to execute commands or initiate actions.

For example, buttons can be used to

- initiate navigation
- invoke an editor or window
- commit data in a form
- issue a query


Create via Object Navigator

To create a button in the Object Navigator:

Step	Action
1	Select the data block where you want to insert the button, then select the Items node, and then choose Navigator→Create .
2	In the Properties Palette, set the Item Type property to Push Button .

Create via Layout Editor

To create a button in the Layout Editor:

Step	Action
1	Click the Button icon  on the toolbar.
2	Click the canvas to place the button on the canvas.
3	Resize the button if necessary.
4	If needed, change the data block to which the button is assigned using the data block poplist.
5	Double-click the button to view the Property Palette.

Iconic buttons

To make a button an iconic button:

Step	Action
1	In the Navigator, select the desired button.
2	In the Property Palette, set the Iconic property to Yes.
3	Enter the name of the icon in the Icon Filename property field. Do not include the icon file extension.

Continued on the next page

Creating and defining buttons, Continued

Designating a button as the default button

You can designate one button per canvas as the default button. Operator can select default buttons implicitly by pressing a platform-specific key, without having to navigate to the button or activate it with the mouse.

Step	Action
1	In the Navigator, select the desired button.
2	In the Property Palette, set the Default Button property to Yes .

Button Properties

Property	Usage
Access Key	Specifies the character that will be the access key, allowing the operator to select or execute an item by pressing a key combination, such as Alt-C.
Label	Specifies the text label that displays for a button.
Mouse Navigate	Specifies whether Oracle Forms should perform navigation to the button when the operator activates the item with a mouse.
Default Button	Specifies whether this button is the default one for the data block.
Iconic	Specifies whether a button is to be an iconic button.
Iconic Filename	Specifies the name of the icon file. Do not include the icon file extension.

GO_BLOCK

Navigates to an indicated data block. If the target data block is non-enterable, an error occurs.

Syntax: `go_block('block_name');`

Continued on the next page

Creating and defining buttons, Continued

Exercise 20

Create a control data block.

Create three non-database items and convert all of them to push buttons:

- Button 1
 - Label: Person
 - Add trigger when-button-pressed for navigation to the Person block
 - Place on canvas1

 - Button 2
 - Label: Return
 - Add trigger when-button-pressed for navigation to the ID block
 - Place on canvas2

 - Button 3
 - Label: Exit
 - Add trigger when-button-pressed and enter exit_form;
 - Place on canvas1
-

Section H: Introduction to Triggers

Overview

Purpose Triggers are data blocks of code you write to add functionality to a default application. You can create a basic application without writing triggers, using only Oracle Forms' default processing to retrieve, add, delete, and change database records. However, you will usually need to write triggers to customize your application.

Objectives This section will examine the following:

- Purpose and types of triggers
- Rules for writing triggers

In this section These topics are covered in this section.

Topic	Page
Triggers overview	H-2
PL/SQL constructs	H-3
PL/SQL Editor	H-4
Create a new trigger	H-5
Trigger definition and scope	H-6
Trigger Properties	H-11

Triggers overview

What are triggers?

Triggers are data blocks of code you write to customize your application. The names of triggers correspond to runtime events, which in turn tell Oracle when the code should be executed.

What are triggers used for?

- Validate data entry
 - Protect the database from operator errors
 - Limit operator access to specified forms
 - Display related field data by performing table lookups
 - Compare values between fields in the form
 - Calculate field values and display the results of those calculations
 - Perform complex transactions, such as verifying totals
 - Display customized error and information messages to the operator
 - Alter default navigation
 - Display alert boxes
 - Create, initialize, and increment timers
-

Groups of triggers

Group	Function
When-triggers	Execute in addition to default processing
On-triggers	Replace default processing
Pre- and Post-triggers	Add processing before or after an event
Key-trigger	Change default processing assigned to a specific key

Writing trigger code

The code in Oracle Forms triggers and menu item commands is written in Oracle's PL/SQL language. PL/SQL is an extension to the SQL database language, and you can include both SQL statements and PL/SQL statements in an Oracle Forms trigger. You can also make calls to built-in Oracle Forms subprograms and to user-named PL/SQL subprograms you write yourself.

PL/SQL constructs

PL/SQL data blocks

The text of an Oracle Forms trigger is an anonymous PL/SQL data block. A data block can consist of three sections:

- a declaration section for variables, constants, cursors, and exceptions (optional)
- executable statements (required)
- exception handlers (optional)

PL/SQL Syntax

```
DECLARE
    -- declarative statements (optional)
BEGIN
    -- executable statements (required)
EXCEPTION
    -- exception handlers (optional)
END;
```

Without DECLARE section

In a trigger, only the executable section is required. When you write a trigger that does not have a DECLARE section, you do not need to include the BEGIN and END keywords, as they are added for you implicitly.

The following example shows such a trigger:

```
/* Key-CLRREC Trigger:          */
IF :System.Record_Status = 'CHANGED' OR
   :System.Record_Status = 'INSERT' THEN
  Commit_Form;
END IF;
Clear_Record;
```

With DECLARE section

If, however, your trigger will have a DECLARE section, you must include the BEGIN and END keywords so the compiler can detect the start of the executable section:

```
DECLARE
  Total_owed NUMBER(7,2);
BEGIN
  SELECT SUM(amount)
         INTO Total_owed
         FROM twraccd
         WHERE pidm = :Account.pidm
                AND paid_date IS NULL;
END;
```

Note

Transactional statements, such as COMMIT, SAVEPOINT, and ROLLBACK cannot be included directly in trigger code. Built-in subprograms can be called instead, which will be discussed later.

PL/SQL Editor

PL/SQL Editor The PL/SQL Editor is where you enter and compile code objects. Code objects in Oracle Forms include event triggers, subprograms (functions and procedures), menu item commands, menu startup code, and packages.

Invoking the editor To invoke the PL/SQL Editor:


- Choose **Program→PL/SQL Editor** from the menu
Or:
- In the Navigator, double-click the object icon for any code object
Or:
- In the Object Navigator, Menu Editor, or Layout Editor, select an object that can have code associated with it and choose **PL/SQL Editor** from the popup menu

Create a new trigger

Create a trigger To create a new trigger:

- Right-click an object in the Object Navigator or Layout Editor and highlight **Smart Triggers** from the pop-up menu. This will list common triggers that are appropriate for the selected object. Select the desired trigger

Or:

- In the Object Navigator, highlight the Triggers node of the form, data block, or item that needs the trigger. Select **Navigator→Create** or click the  icon on the toolbar

Or:

- Inside the PL/SQL Editor, click **New** to create a new trigger
-

Trigger definition and scope

Trigger definition

The object to which a trigger is attached determines the trigger's definition level in the object hierarchy. There are three levels in which a trigger can be defined:

- Form level
- Data block level
- Item level

Trigger scope

A trigger's definition level determines the trigger's scope. The scope of a trigger is its domain within the Oracle Forms object hierarchy, which determines where an event must occur for the trigger to respond to it.

A data block-level trigger fires if the trigger event occurs within that data block, but it does not fire if the same event occurs in some other data block.

Example

If you wanted a trigger to fire when Button1 in your form was pressed, you have the option to define the trigger at the item, data block, and form level.

Level	Result
Item Level	Fires only when Button1 is pressed.
Data Block Level	Fires when any button in the data block is pressed.
Form Level	Fires when any button in the form is pressed.

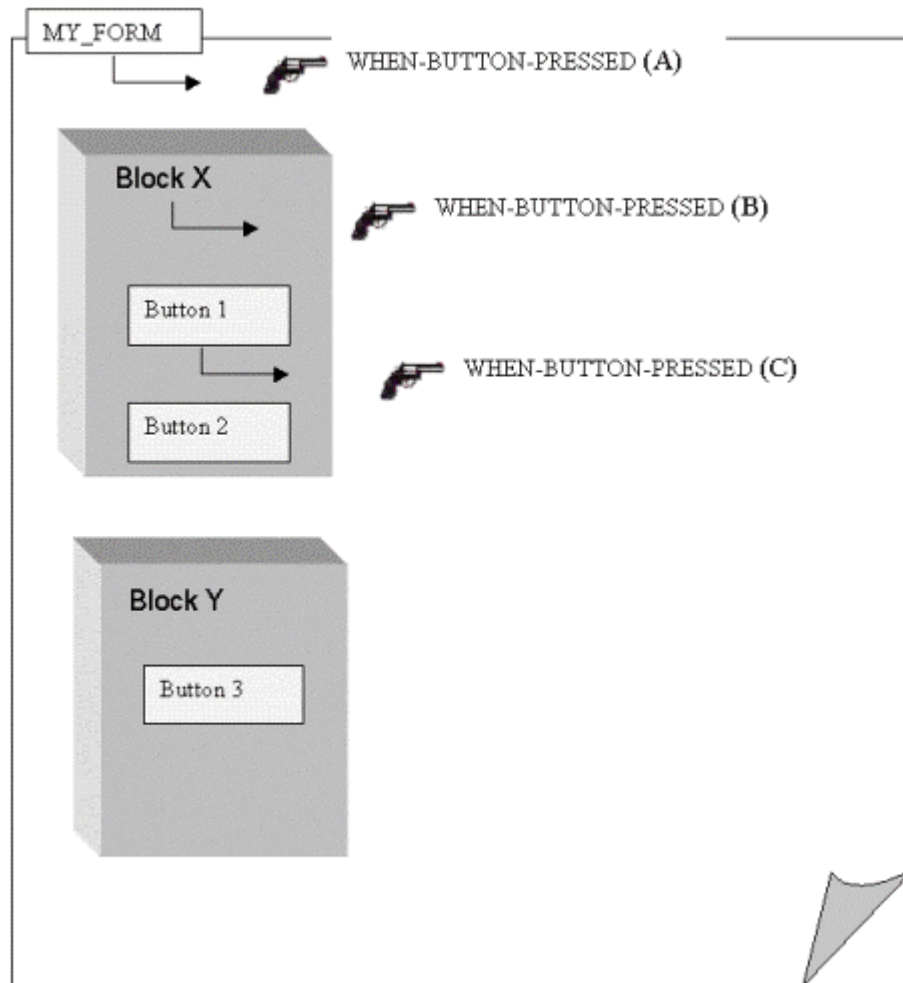
By default, only the trigger that is most specific to the cursor's current location will fire. Let's expand on the example above.

Continued on the next page

Trigger definition and scope, Continued

Diagram 1

Within form MY_FORM, three WHEN-BUTTON-PRESSED triggers are placed.

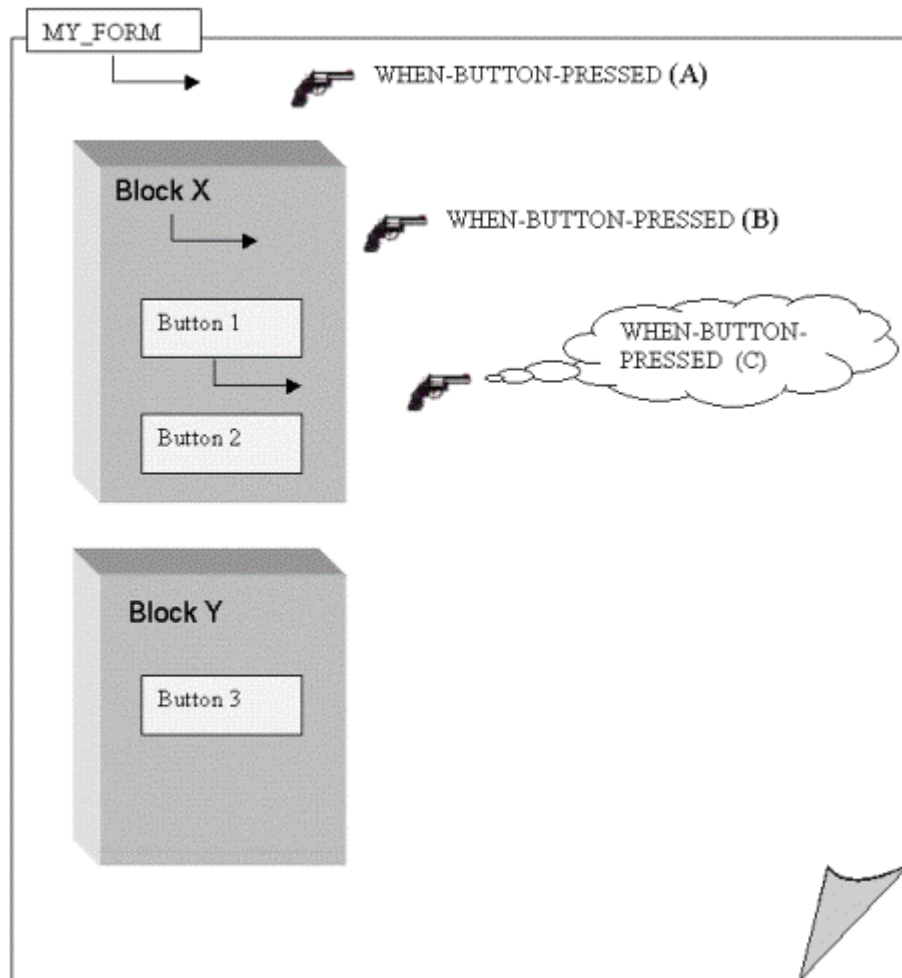


Continued on the next page

Trigger definition and scope, Continued

Diagram 2

Instance One: Button 1 is pressed. The trigger defined at the item level is fired (C).

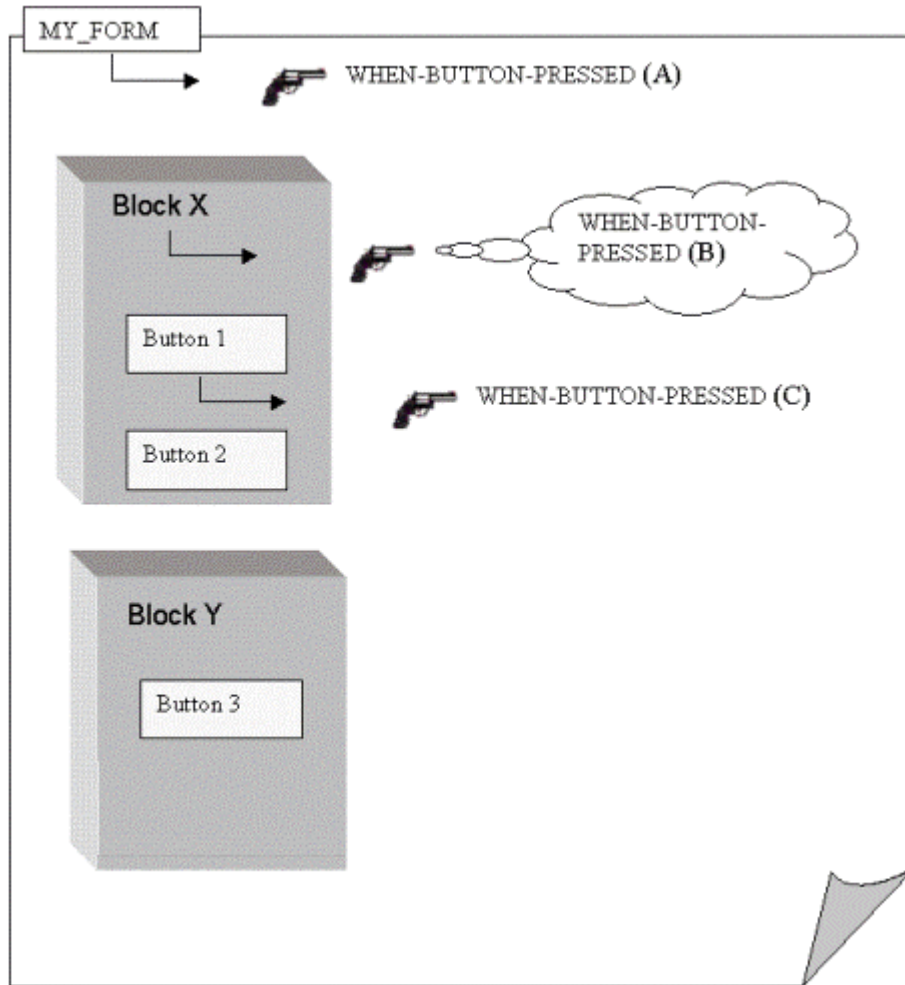


Continued on the next page

Trigger definition and scope, Continued

Diagram 3

Instance Two: Button 2 is pressed. Because no trigger is defined at the item level, Oracle Forms looks next at the Data block level. Since a WHEN-BUTTON-PRESSED trigger is defined at the data block level (B), it is fired.

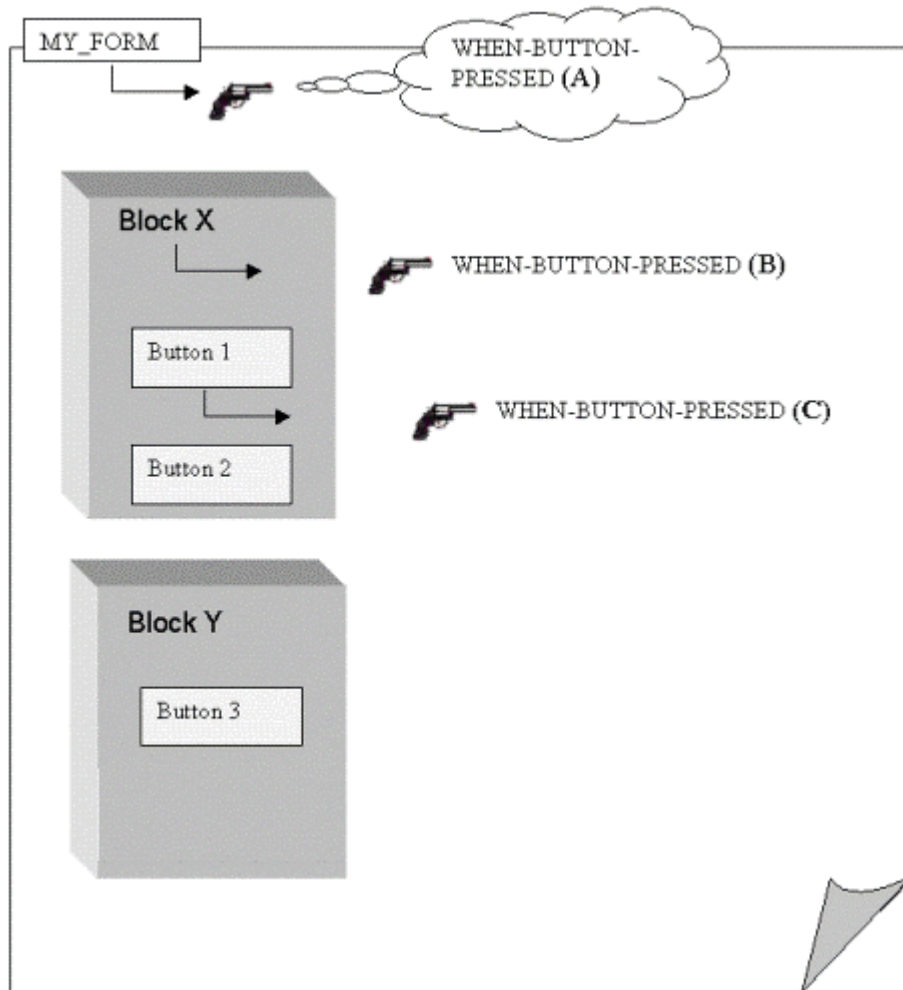


Continued on the next page

Trigger definition and scope, Continued

Diagram 4

Instance Three: User presses Button 3. Forms looks for a trigger at the item level. Since none exists, it looks at the data block level. Since that does not exist, the WHEN-BUTTON-PRESSED trigger is fired at the form level (A).



Trigger Properties

Execution Style Specifies how the current trigger code should execute if there is a trigger with the same name defined at a higher level in the object hierarchy.

The following settings are valid for this property:

Setting	Description
Override	Specifies that the current trigger fire instead of any trigger by the same name at any higher scope. This is known as "override parent" behavior. The default.
Before	Specifies that the current trigger fire before firing the same trigger at the next-higher scope. This is known as "fire before parent" behavior.
After	Specifies that the current trigger fire after firing the same trigger at the next-higher scope. This is known as "fire after parent" behavior.

FORM_ TRIGGER_ FAILURE_ exception

- A predefined PL/SQL exception available only in Oracle Forms
- You can raise this exception without having to first define it in the declarative section

When does a trigger fail?

- When a FORM_TRIGGER_FAILURE is raised
- When an unhandled exception occurs

If a trigger is not failed, then processing continues. In the example above, if the RAISE FORM_TRIGGER_FAILURE was omitted, the user would receive the message but would be allowed to leave the item.

Section I: Navigation with Triggers

Overview

Purpose As a user, you have been able to navigate around your form using the mouse, function keys, and the pull-down menu. Because a GUI environment offers so many ways of leaving or entering items, you may want to perform checks before allowing movement. In addition, you may want to automate more for the user, navigating to items for them programmatically.

- Objectives** This section will examine the following:
- Default navigation sequence
 - Controlling navigation through triggers
 - Built-in navigation subprograms
-

In this section These topics are covered in this section.

Topic	Page
WHEN-NEW-object-INSTANCE triggers	I-2
Built-in Navigation subprograms	I-5

WHEN-NEW-object-INSTANCE triggers

Trigger	Description
WHEN-NEW-ITEM-INSTANCE	Fires after the input focus successfully moves to an item.
WHEN-NEW-RECORD-INSTANCE	Fires after the input focus successfully moves to a record.
WHEN-NEW-BLOCK-INSTANCE	Fires after the input focus successfully moves to a data block.
WHEN-NEW-FORM-INSTANCE	Fires after the input focus successfully moves to a form.

Exercise 21 At the form level, create a trigger to immediately execute a query.

At both block levels for SWRADDR and SWBPERS when the block is entered also, execute a query.

Exercise 22 When navigating to the next record in the ID block, get the records in the address block to change as well.

Continued on the next page

Pre- and Post-triggers

PRE-FORM	Fires during the Enter the Form event, at form startup. Uses: <ul style="list-style-type: none">• Assign unique primary key from sequence• Restrict access to a form• Initialize global variables
POST-FORM	Fires during the Leave the Form process, when a form is exited. Uses: <ul style="list-style-type: none">• To clean up the form before exiting, such as global variables that the form no longer requires• To display a message to the operator upon form exit
PRE-BLOCK	Fires during the Enter the Data Block process, during navigation from one data block to another. Uses: <ul style="list-style-type: none">• Allow or disallow access to a data block• Set variable values
POST-BLOCK	Fires during the Leave the Data Block process. Uses: <ul style="list-style-type: none">• Validate the data block's current record• To test a condition and prevent the user from leaving a data block based on that condition

Continued on the next page

Pre- and Post-triggers, Continued

PRE-RECORD	Fires during the Enter the Record process, during navigation to a different record. Uses: <ul style="list-style-type: none">• Keep a running total
POST-RECORD	Fires during the Leave the Record process. Specifically, the Post-Record trigger fires whenever the operator or the application moves the input focus from one record to another. Uses: <ul style="list-style-type: none">• Perform an action whenever the operator or the application moves the input focus from one record to another
PRE-TEXT-ITEM	Fires during the Enter the Item process, during navigation from an item to a text item. Uses: <ul style="list-style-type: none">• Derive a complex default value, based on other items previously entered into the same record• Record the current value of the text item for future reference, and store that value in a global variable or form parameter
POST-TEXT-ITEM	Fires during the Leave the Item process for a text item. Specifically, this trigger fires when the input focus moves from a text item to any other item. Uses: <ul style="list-style-type: none">• Calculate or change item values

Exercise 23

Write a PRE-FORM trigger in SWAIDEN, which checks the username. If the name does not match your username, fail the trigger and display a message. Use the built-in subprogram GET_APPLICATION_PROPERTY. Attempt to run your form under a different training account. What occurred?

Built-in Navigation subprograms

Subprogram	Description
GO_FORM	In a multiple-form application, navigates from the current form to the indicated target form.
GO_BLOCK	Navigates to an indicated data block. If the target data block is non-enterable, an error occurs.
GO_RECORD	Navigates to the record with the specified record number.
GO_ITEM	Navigates to an indicated item. GO_ITEM succeeds even if the target item has the Navigable property set to False.
NEXT_BLOCK	Navigates to the first navigable item in the next enterable data block in the navigation sequence.
NEXT_RECORD	Navigates to the first enabled and navigable item in the record with the next higher sequence number than the current record.
NEXT_ITEM	Navigates to the navigable item with the next higher sequence number than the current item.
NEXT_SET	Fetches another set of records from the database. Then navigates to the first record that the fetch retrieves.
NEXT_KEY	Navigates to the enabled and navigable primary key item with the next higher sequence number than the current item.
DOWN	Navigates to the instance of the current item in the record with the next higher sequence number.
UP	Navigates to the instance of the current item in the record with the next lowest sequence number.
SCROLL_DOWN	Scrolls the current data block's list of records so that previously hidden records with higher sequence numbers are displayed.
SCROLL_UP	Scrolls the current data block's list of records so that previously hidden records with lower sequence numbers are displayed.
PREVIOUS_BLOCK	Navigates to the first navigable item in the previous enterable data block in the navigation sequence.
PREVIOUS_RECORD	Navigates to the first enabled and navigable item in the record with the next lower sequence number than the current record.
PREVIOUS_ITEM	Navigates to the navigable item with the next lower sequence number than the current item. If there is no such item, PREVIOUS_ITEM navigates to the navigable item with the highest sequence number.

Continued on the next page

Built-in Navigation subprograms, Continued

Example

```
/* WHEN-BUTTON-PRESSED-TRIGGER
   Navigate to Data Block A
*/
GO_BLOCK('BLOCKA');
IF NOT FORM_SUCCESS THEN
    MESSAGE('Could not navigate to Data Block A');
    RAISE FORM_TRIGGER_FAILURE;
END IF;
```

FORM_ SUCCESS

Returns the outcome of the action most recently performed during the current Runform session.

Use FORM_SUCCESS to test the outcome of a built-in to determine further processing within any trigger. To get the correct results, you must perform the test immediately after the action executes.

Cursor Position System Variables

CURSOR_BLOCK	The name of the data block where the cursor is located.
CURSOR_RECORD	The number of the record where the cursor is located.
CURSOR_ITEM	The name of the data block and item (data block.item) where the cursor is located.
CURSOR_POSITION	The value of the item where the cursor is located.

Section J: Validation Triggers

Overview

Purpose Validation triggers fire when Oracle Forms validates data in an item or record. Oracle Forms performs validation checks during navigation that occur in response to operator input, programmatic control, or default processing, such as commit operation.

Objectives This section will examine the following:

- When does forms validate?
- What are the types of validation triggers?

In this section These topics are covered in this section.

Topic	Page
Validating items during data entry	J-2
When-Validate-Item	J-3
Validating at the record level	J-4

Validating items during data entry

Setting properties

Many of the most common validation requirements can be handled by setting the following item-level properties:

- Data Type
- Maximum Length
- Fixed Length
- Required
- Range (Lowest Allowed Value / Highest Allowed Value)

When you need to add application-specific validation, you can write a When-Validate-Item trigger. This will fire **after** the standard checks listed above.

When does item validation occur?

- When the user tries to leave the item
 - When the form is saved
-

When-Validate-Item

Purposes

The WHEN-VALIDATE-ITEM trigger is used for two main purposes:

- validating an item
- populating non-base table items

Populate an item

To populate an item with values from another table:

Step	Action
1	Create an item of the desired type in the appropriate base table data block.
2	Make the item a display item by setting its Database Item property to No .
3	Set the item's Data Type , Maximum Length , and other properties to be compatible with the type of fetched values the item will display.
4	Write the necessary triggers to populate the control item at runtime.

Example

```
BEGIN
SELECT rtrim(swriden_last_name,' ')||', '||
       rtrim(swriden_first_name,' ')||' '||
       swriden_mi
INTO :name
FROM swriden
WHERE swriden_id = :swriden_id;
EXCEPTION
WHEN NO_DATA_FOUND THEN
  Message('Invalid ID. Please enter again. ');
  RAISE FORM_TRIGGER_FAILURE;
END;
```

Continued on the next page

Validating at the record level

WHEN_VALIDATE_RECORD

The WHEN-VALIDATE-RECORD is used mainly to enforce that a combination of item values is valid.

```
IF :DATE_ENROLLED > :DATE_GRADUATED THEN
    MESSAGE('The date of enrollment must occur
before the date of graduation');
    RAISE FORM_TRIGGER_FAILURE;
END IF;
```

When does record-level validation occur?

The When-Validate-Record trigger fires during the Leave the Record event, when Oracle Forms checks to ensure that all of the items in the current record are marked valid before navigating to the target record.

Exercise 24

In the Address data block, create a trigger to populate the address description item whenever validation occurs on swraddr_atyp_code (use STVATYP).

Fail the trigger and display a suitable message if the swraddr_atyp_code is not found.

Run the form, and enter a new address with an incorrect address type. Enter a correct address type to see if the display item populates.

Continued on the next page

Validating at the record level, Continued

Exercise 25 In the Address data block, create a trigger to populate the state description whenever validation occurs.

Exercise 26 Delete the swriden_last_name and swriden_first_name items and create one non-database item called name on the swriden block. Create a trigger that will populate the name after an ID is entered. If the ID is invalid, fail the trigger and display a suitable message.

Section K: Query Triggers

Overview

Purpose Query triggers allow you to control events just before and just after a query.

Objectives This section will examine the following:

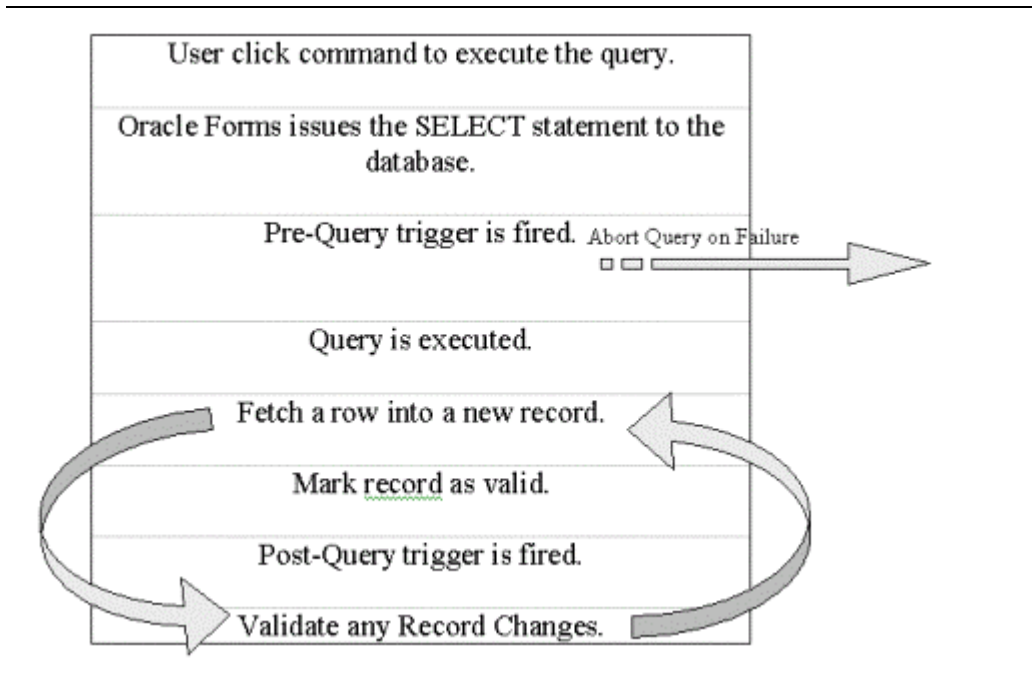
- Query processing flowchart
- Triggers which screen query triggers
- Triggers which supplement query results

In this section These topics are covered in this section.

Topic	Page
Query Processing	K-2
PRE-QUERY	K-3
POST-QUERY	K-4
Fire in Enter Query Mode	K-5
System Modes	K-6

Query Processing

Diagram



PRE-QUERY

Pre-Query triggers

Fires during Execute Query or Count Query processing, just before Oracle Forms constructs and issues the SELECT statement to identify rows that match the query criteria.

Use a Pre-Query trigger to modify the example record that determines which rows will be identified by the query.

On Failure

The query is canceled. If the operator or the application had placed the form in Enter Query mode, the form would remain in Enter Query mode.

Example

This example validates or modifies query criteria for a database data block query.

```
BEGIN
  IF :SWRIDEN_ID IS NULL THEN
    Message('Id must be entered for Query. ');
    RAISE Form_Trigger_Failure;
  END IF;
END;
```

POST-QUERY

Post-Query triggers

When a query is open in the data block, the Post-Query trigger fires each time Oracle Forms fetches a record into a data block. The trigger fires once for each record placed on the data block's list of records.

Use a Post-Query trigger to perform the following tasks:

- populate control items or items in other data blocks
- calculate statistics about the records retrieved by a query
- calculate a running total

On Failure

Oracle Forms flushes the record from the data block and attempts to fetch the next record from the database. If there are no other records in the database, Oracle Forms closes the query and waits for the next operator action.

Example

```
SELECT description
   INTO detc_desc
  FROM twvdetc
 WHERE detc_code = :Account.detc_code;
```

Exercise 27

Create a post-query trigger on the SWRADDR block to populate the atyp_desc field.

Fire in Enter Query Mode

Conditions

- Specifies that the trigger should fire when the form is in Enter Query mode, as well as in Normal mode.
 - Only applicable to the following triggers:
 - Key
 - On-Error
 - On-Message
 - When- triggers, except:
 - When-Database-Record
 - When-Image-Activated
 - When-New-Block-Instance
 - When-New-Form-Instance
 - When-Create-Record
 - When-Remove-Record
 - When-Validate-Record
 - When-Validate-Item
-

System Modes

Syntax

SYSTEM.MODE

Description

SYSTEM.MODE indicates whether the form is in Normal, Enter Query, or Fetch Processing mode. The value is always a character string.

Mode	Description
NORMAL	Indicates that the form is currently in normal processing mode.
ENTER-QUERY	Indicates that the form is currently in Enter Query mode.
QUERY	Indicates that the form is currently in fetch processing mode, meaning that a query is currently being processed.

Exercise 28

Ensure that the Exit button has no effect in Enter Query Mode.

Section L: Transaction Triggers

Overview

Purpose When a form's changes are saved during runtime, Oracle Forms enables you to fire triggers before and after events to control these actions.

Objectives This section will examine the following:

- What is a transaction
- What happens during transaction processing
- Learn triggers which can be added to enhance transaction processing

In this section These topics are covered in this section.

Topic	Page
Transactions	L-2
Data block-level transaction triggers	L-3
Form-level transactional triggers	L-6

Transactions

What is a transaction?

An Oracle Forms transaction is considered to be the set of all DML statements made between saves. If the user saves changes three times during the day, then three transactions occurred.

What occurs during transaction processing?

Two phases always occur once the user saves changes to a form.

- **Posting**
The changes that were made to the records in the forms are posted to the database in data block sequence order. For each data block, Deletes are posted first, followed by Updates and Inserts.
 - **Commit**
Performs the database commit, making all changes permanent.
-

Data block-level transaction triggers

Timing

Fire just before or after each DML statement is posted to the database.

Trigger	Description
PRE-DELETE	Fires before a row is deleted. It fires once for each record that is marked for delete. <u>Note:</u> Oracle Forms creates a Pre-Delete trigger automatically for any master-detail relation that has the Master Deletes property set to Cascading.
POST-DELETE	Fires after a row is deleted. It fires once for each row that is deleted from the database during the commit process.
PRE-UPDATE	Fires before a row is updated. It fires once for each record that is marked for update.
POST-UPDATE	Fires after a row is updated. It fires once for each row that is updated in the database during the commit process.
PRE-INSERT	Fires before a row is inserted. It fires once for each record that is marked for insert.
POST-INSERT	Fires just after a record is inserted. It fires once for each record that is inserted into the database during the commit process.

Pre-insert trigger

A pre-insert trigger is a good place to generate a one-up-number or to obtain the activity date for the record.

Example

```
SELECT swriden_pidm_sequence.NEXTVAL  
  INTO :swriden_pidm  
  FROM sys.dual;
```

Pre-update trigger

A pre-update trigger is a good place to update the activity date or the user who is making the change to the record.

```
:swraddr.activity_date := sysdate;
```

Continued on the next page

Data block-level transaction triggers, Continued

Exercise 29 In both the Address and Person data blocks, create triggers that populate the activity date and pidm when the user saves a new record.

Exercise 30 In both the Address and Person data blocks, create a trigger that populates the activity date when the user updates an existing record.

Continued on the next page

Data block-level transaction triggers, Continued

Pre-delete trigger

To ensure that detail records do not exist before a master record is deleted, you can use the pre-delete trigger.

Example

```
BEGIN
  SELECT 'X'
    INTO :global.dummy
    FROM swraddr
    WHERE swraddr.pidm = :swriden.pidm;
  WHEN SQL%FOUND THEN
    MESSAGE('Data found in address table.  Cannot
delete Identification record.');
```

```
    RAISE FORM_TRIGGER_FAILURE;
EXCEPTION
  WHEN NO_DATA_FOUND THEN NULL;
END;
```

Form-level transactional triggers

PRE-COMMIT trigger Fires once during the Post and Commit Transactions process, before Oracle Forms processes any records to change. Specifically, it fires after Oracle Forms determines that there are inserts, updates, or deletes in the form to post or commit.

The trigger does not fire when there is an attempt to commit, but validation determines that there are no changed records in the form.

Usage notes Use a Pre-Commit trigger to perform an action, such as setting up special locking requirements, anytime a database commit is going to occur.

On failure The Post and Commit process fails: No records are written to the database and focus remains in the current item.

POST-FORMS-COMMIT Fires once during the Post and Commit Transactions process. If there are records in the form that have been marked as inserts, updates, or deletes, the Post-Forms-Commit trigger fires after these changes have been written to the database but before Oracle Forms issues the database Commit to finalize the transaction.

If the operator or the application initiates a Commit when there are no records in the form have been marked as inserts, updates, or deletes, Oracle Forms fires the Post-Forms-Commit trigger immediately, without posting changes to the database.

Usage notes Use a Post-Forms-Commit trigger to perform an action, such as updating an audit trail, anytime a database commit is about to occur.

On failure Aborts post and commit processing: Oracle Forms issues a ROLLBACK and decrements the internal Savepoint counter.

Continued on the next page

Form-level transactional triggers, Continued

**POST-
DATABASE-
COMMIT**

Fires once during the Post and Commit Transactions process, after the database commit occurs.

Note that the Post-Forms-Commit trigger fires after inserts, updates, and deletes have been posted to the database, but before the transaction has been finalized by issuing the Commit. The Post-Database-Commit Trigger fires after Oracle Forms issues the Commit to finalize the transaction.

Usage notes

Use a Post-Database-Commit trigger to perform an action anytime a database commit has occurred.

On failure

There is no rollback, because at the point at which this trigger might fail, Oracle Forms has already moved past the point at which a successful rollback operation can be initiated as part of a failure response.

Section M: Creating Lists of Values and Editors

Overview

Purpose Lists of Values provide the user with choices of values for a given item. A List of Values can contain either suggestions or a validation list.

Objectives This section will examine the following:

- What can LOVs do?
- How do I create an LOV?

In this section These topics are covered in this section.

Topic	Page
What is a List of Values?	M-2
Creating an LOV using the LOV Wizard	M-3

What is a List of Values?

Definition


A List of Values (LOV) is a scrollable pop-up window that provides the operator with either a single or multi-column selection list.

LOVs provide the following functionality:

- Can be displayed by operator request when the operator navigates to a text item with an associated LOV, or programmatically, independent of any specific text items
 - LOV auto-reduction and search features allow operators to locate specific values
 - Values are selected by the operator can be assigned to form items according to the return items you designate
 - At design time, an LOV can be attached to one or more text items in the form
 - LOV values are derived from record groups
-

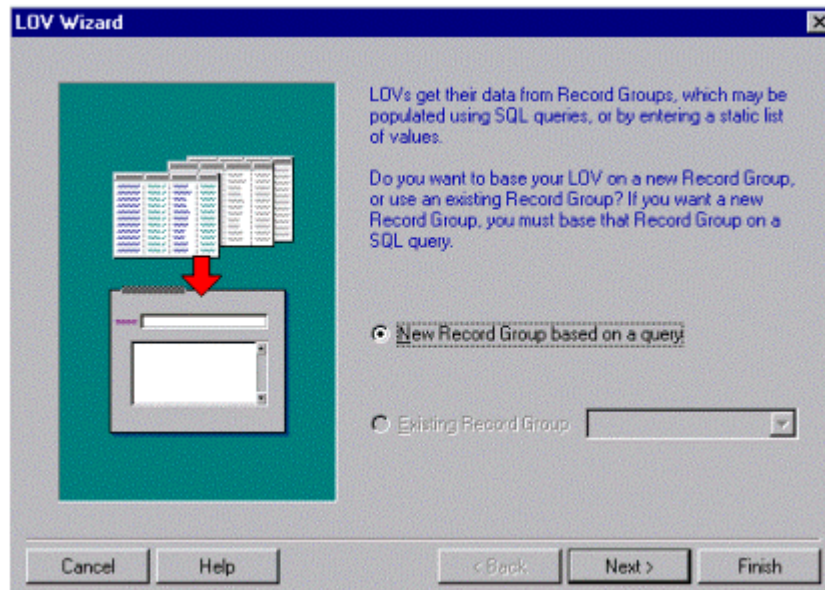
Creating an LOV using the LOV Wizard

Start the LOV Wizard

In the Navigator, create an LOV object by selecting the LOVs node. Then choose **Navigator**→**Create**, or click the  icon on the toolbar, or choose **Tools**→**LOV Wizard**.

Specify whether your LOV will be based on an existing record group or create a new query record group.

Screen



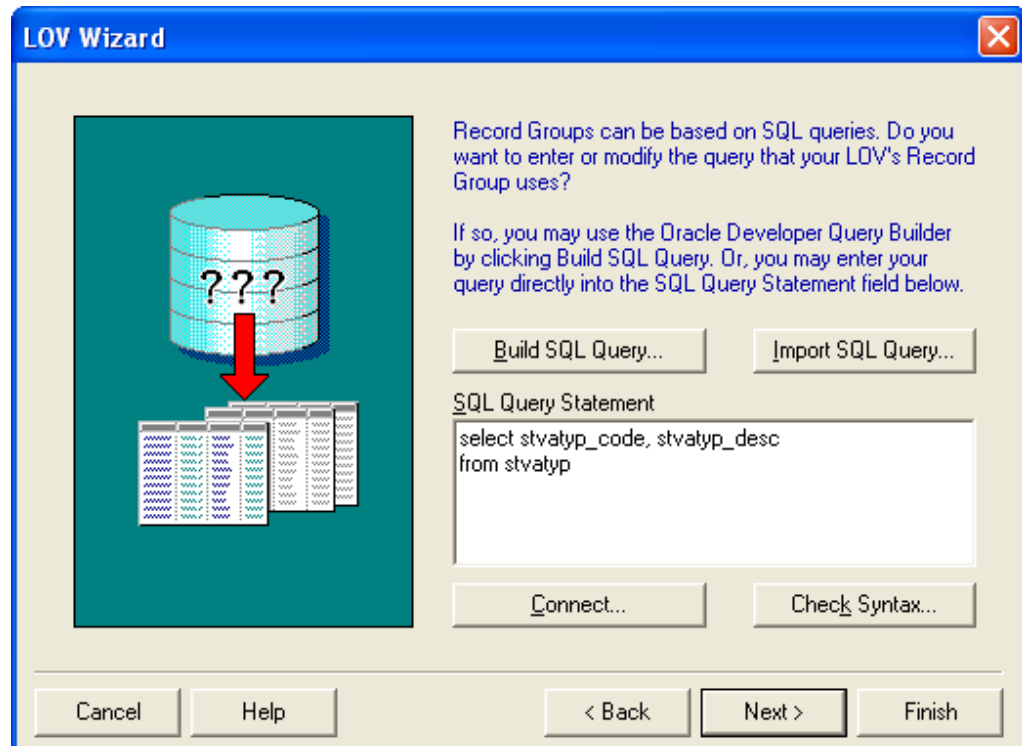
Continued on the next page

Creating an LOV using the LOV Wizard, Continued

Specify the query

On the SQL Query page, specify the query used to construct the record group. This page is only used for new record groups.

Screen



Build the Query

- Click the **Build SQL Query** button to invoke Query Builder.
- Click the **Import SQL Query** button to import an existing query from a file.
- To enter a query directly, just type the query into the SQL Query Statement field.

Additional Buttons

- Click the **Connect** button to connect with the database if you are not currently connected.
- Click the **Check Syntax** button to verify the validity of the query.

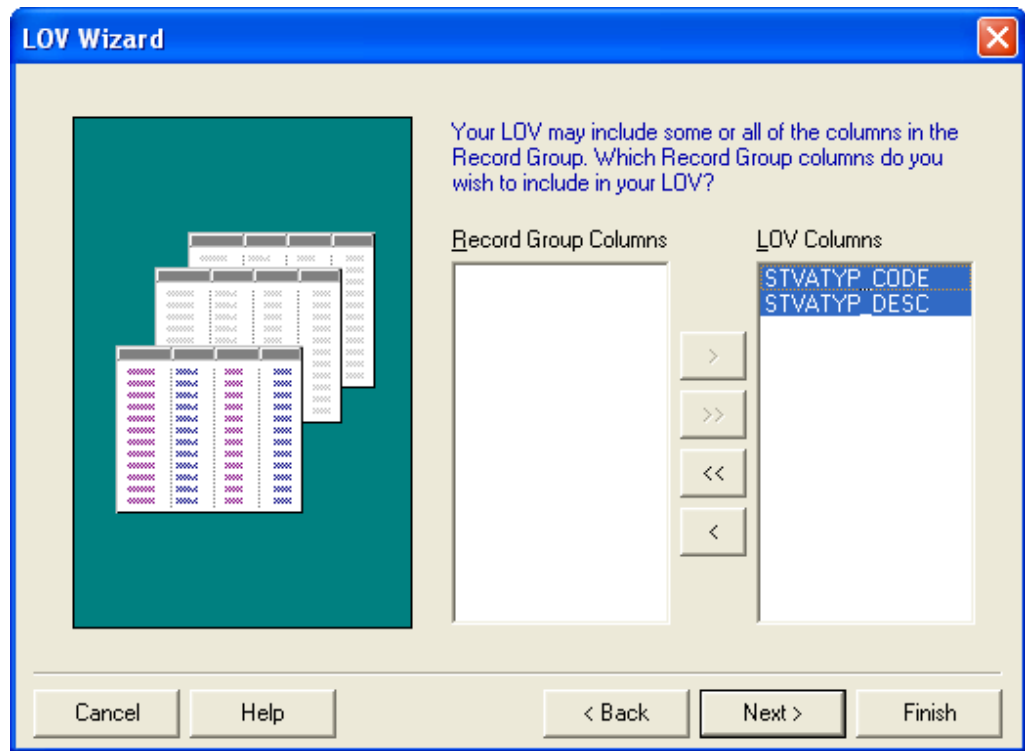
Continued on the next page

Creating an LOV using the LOV Wizard, Continued

Select Record Group columns

Choose the Record Group columns that will be included in the LOV.

Screen



Continued on the next page

Creating an LOV using the LOV Wizard, Continued

Define the LOV columns Define the look of the LOV columns, including which items will be assigned the returned LOV value.

Screen

If you wish to specify the LOV column properties, you may enter a title, width and return value for each LOV column. The units for the column width is Points.

Column	Title	Width	f
STVATYP_CODE	Stvatyp_Code	20	
STVATYP_DESC	Stvatyp_Desc	209	

Automatically size columns Look up return item...

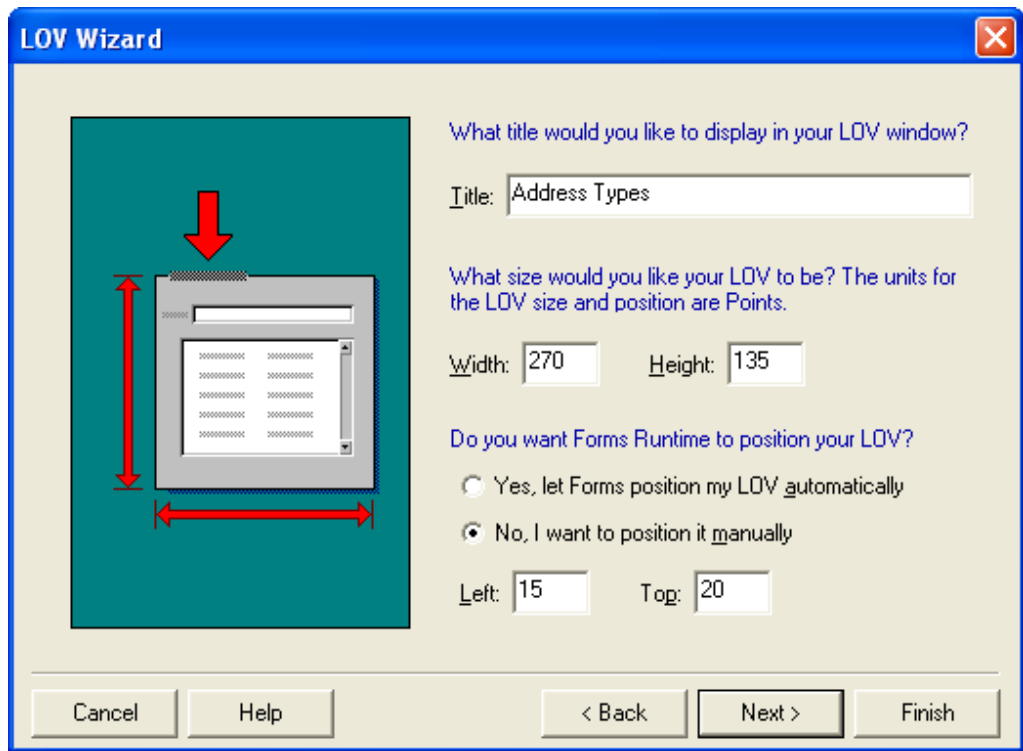
Cancel Help < Back Next > Finish

Continued on the next page

Creating an LOV using the LOV Wizard, Continued

Design the LOV window Design the look of the LOV window.

Screen



Continued on the next page

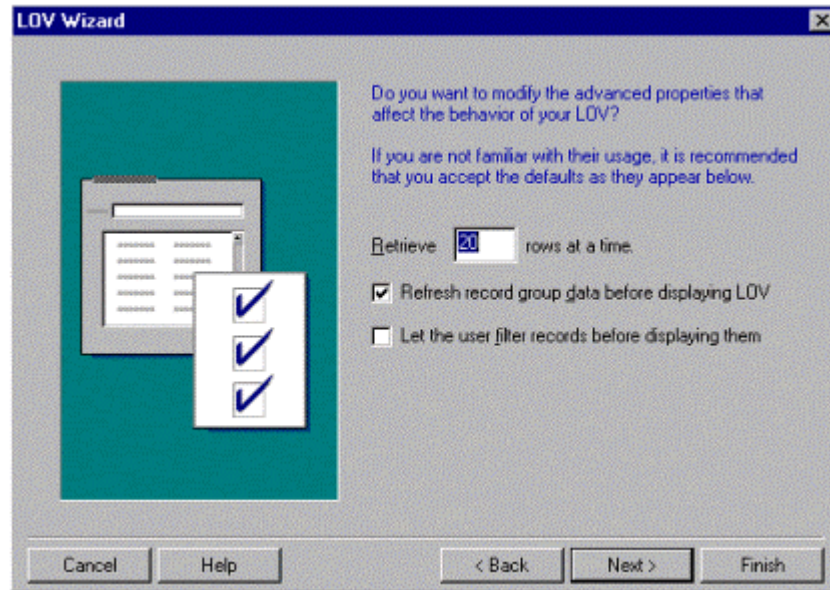
Creating an LOV using the LOV Wizard, Continued

Set up advanced options

Set up some advanced options:

- Number of rows fetched
- Whether the record group will be queried each time the LOV is invoked
- Whether the user will have to enter additional criteria before the LOV is displayed

Screen

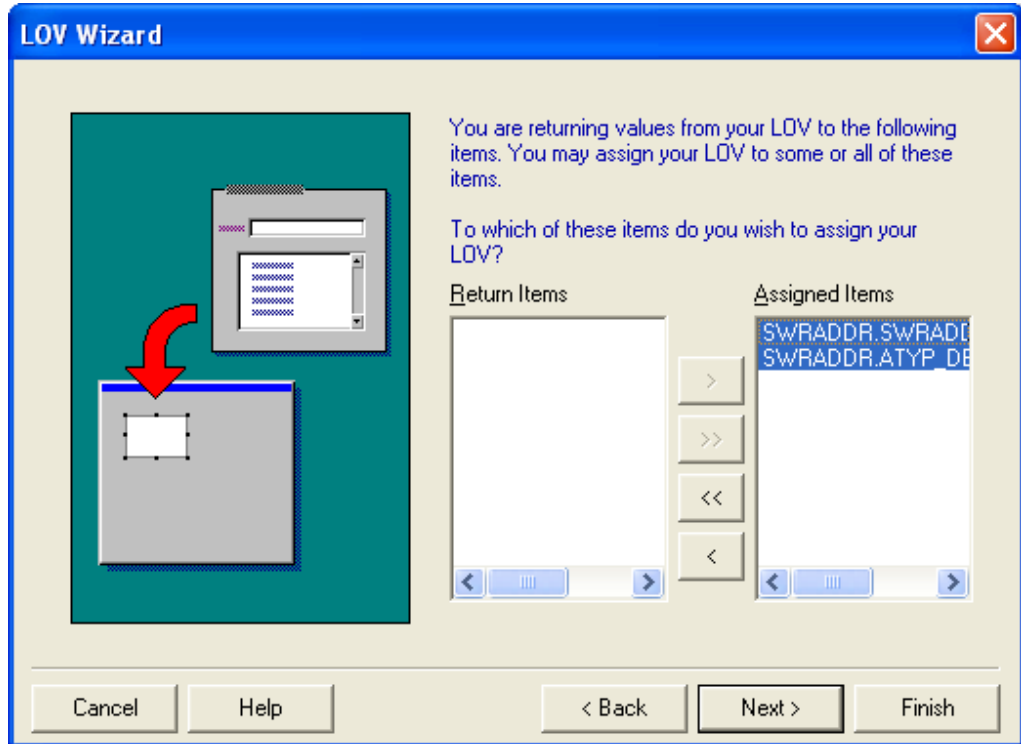


Continued on the next page

Creating an LOV using the LOV Wizard, Continued

Attach the LOV If there are LOV values being returned, then attach the LOV to a specific Form Item.

- In the finish page, Click Finish to complete the LOV creation process.



If the LOV is not currently attached, then attach it to a text item, or write a trigger to display the LOV programmatically.

Now under the LOV list is LOVxx and under the Record Group list is LOVxx. Rename the record group to STVATYP_RG and the LOV to STVATYP_LOV

Using LOV Values to Validate Text Items

Step	Action
1	In the Navigator, select the desired text item.
2	In the Property Palette window, set the Validate from List property to Yes.

Continued on the next page

Creating an LOV using the LOV Wizard, Continued

Creating a new LOV manually

Validate state codes via a push button.

Step	Action
1	Create a record group named <i>stvstat_rg</i> .
2	Create a LOV named <i>stvstat_LOV</i> .
3	Give the LOV a title.
4	Assign the RG.
5	Map the columns. In this case, we only want to map the state abbreviation because we do not have the full state name displayed on the form.
6	Attach the LOV to the item state.
7	Set the coordinates.
8	Validate from list property.

Invoking a LOV with a button

Step	Action
1	Create a button STATE_LBT in the control block. The extension for the button associated with a LOV should be LBT (list button).
2	Add a when-button-pressed trigger that will display the LOV.
3	Use the show_lov builtin to display the LOV show_lov('lov_name');
4	Remove the boilerplate text that says State and replace it with the button.

Exercise 31

Add an address type button to invoke the stvatyp_lov.

Section N: Canvases Part II

Overview

Purpose A GUI environment takes advantage of tiling, minimizing, cascading, etc. Oracle Forms allows you to take advantage of these GUI attributes.

Objectives This section will examine the following:

- Creating and modifying Content Canvases
- Types of canvases
- Windows and their properties

In this section These topics are covered in this section.

Topic	Page
About canvases	N-2
Stacked Canvas	N-3
Toolbar Canvas	N-7
Tab Canvas	N-9

About canvases

Types of canvases

-
- Content Canvas
 - Stacked Canvas
 - Toolbar Canvas
 - Tab Canvas
-


Stacked Canvas

- Characteristics** A stacked canvas is displayed in a window on top of, or "stacked" on the content canvas assigned to that same window.
- Stacked canvases obscure some part of the underlying content canvas, and are often shown and hidden programmatically
 - More than one stacked canvas can be displayed in a window at the same time

Create via Object Navigator To create a stacked canvas in the Object Navigator:

Step	Action
1	In the Object Navigator, position the cursor on the Canvases node, then choose Navigator→Create to insert a new canvas.
2	In the Properties Palette, set the Canvas Type property to Stacked .
3	Set the other properties of the canvas object as described on the next page.

Create via Layout Editor To create a stacked canvas in the Layout Editor:

Step	Action
1	Go to the Layout Editor and bring up the content canvas on which you are placing the stacked canvas.
2	Click the Stacked Canvas icon  on the toolbar.
3	Click and drag the mouse in the canvas to the position where you want to place the stacked canvas.
4	Open the Properties Palette and set the properties of the canvas object as described on the next page.

Continued on the next page

Stacked Canvas, Continued

Stacked Canvas Properties

Property	Usage
Visible	Set this to Yes if you want the stacked canvas to be visible when the window is invoked, or to No if you want it to be hidden until it is shown in response to navigation or programmatic events.
Width/Height	Specify the size of the canvas. A stacked canvas is usually smaller than the content canvas displayed in the same window.
Bevel	Determines whether the stacked view has a border. A border can visually separate a stacked view from other views displayed in the same window. To make a borderless view, set Bevel to None.
Viewport Width / Viewport Height	Specifies the size of the viewport for the stacked canvas. If you make the view smaller than the canvas, the stacked canvas can be scrolled at runtime.
Viewport X Position / Viewport Y Position	Specifies the x,y display coordinates of the viewport's upper-left corner relative to the upper-left corner of the content viewport currently displayed in the window. The default setting (0,0) displays the stacked viewport at the upper-left corner of the content viewport.
Show Horizontal Scrollbar / ShowVertical Scrollbar	Specifies that the stacked canvas should have a horizontal and/or vertical scroll bar. Operators can scroll the stacked canvas independently of the underlying content canvas.

Positioning

If the stacked canvas is not being shown programmatically or in response to navigation, make sure that its position in the canvas stacking order places it in front of the content canvas assigned to the same window. If it is not, it will be shown behind the content canvas, and will not be visible at runtime.

Continued on the next page

Stacked Canvas, Continued

Stacking order The stacking order of canvases in a window is defined by the sequence in which they are listed under the Canvases node in the Object Navigator.

Displaying a stacked view To display a stacked canvas in the Layout Editor, choose **View** → **Stacked View** and select the appropriate stacked canvas.

To hide the stacked canvas, hold the [**Shift**] key and select the appropriate stacked canvas.

Exercise 32 Create a new block and canvas called Comment and add items from the swrcmnt table.

Change the canvas type to stacked.

Don't forget the swrcmnt_cmtt_code LOV.

Exercise 33 Create a new window called Window2 in the SWAIDEN form, suitable for presenting Comment.

- Associate comment with WINDOW2
- Modify the height and width properties
- Add a title for the window
- Define a display position that is different than Window1, so that it appears below the name information displayed on CANVAS1
- Set the Hide on Exit property to Yes

Continued on the next page

Stacked Canvas, Continued

Exercise 34

Associate CANVAS2 (Personal Information) with Window2.

- Run the form to see the differences.
-

Toolbar Canvas

-
- Characteristics** Toolbar canvases are used to create toolbars for individual windows.
- Horizontal toolbars are displayed at the top of a window, just under its menu bar
 - Vertical toolbars are displayed along the left side of a window
 - MDI toolbars are used to avoid creating multiple toolbars for a Multi-Form Application. SCT Banner uses MDI Toolbars for its forms
 - Associate the toolbar with a window, and remember to update the window's toolbar property
-

Continued on the next page

Toolbar Canvas, Continued

Create a toolbar

Step	Action
1	In the Object Navigator, position the cursor on the Canvases node.
2	Choose Navigator → Create to insert a new canvas in the object hierarchy.
3	In the Property Palette, set the properties of the canvas as follows:

Property	Usage
Canvas Type	Set to Horizontal Toolbar or Vertical Toolbar.
Window	Specify the window on which you want the toolbar to display.
Width/Height	Oracle Forms will display whatever size toolbar you create, even one that completely obscures the window's content canvas-view. It is up to you to set the Width and Height properties to appropriate values, depending on how large you want the toolbar to be.

Step	Action
4	For the window to which you assigned the toolbar canvas, set the Horizontal Toolbar Canvas or Vertical Toolbar Canvas property by specifying the name of the toolbar canvas you created in step 2.
5	For the Form to which you want to assign a MDI Toolbar, set the Form Horizontal Toolbar Canvas or Form Vertical Toolbar Canvas property by specifying the name of the toolbar canvas you created in step 2.
6	Add items and boilerplate graphics to the toolbar canvas as you would for any other canvas.


Tab Canvas

-
- Characteristics** A tab canvas is displayed in a window on top of the content canvas assigned to that same window.
- Tab canvases enable you to organize and display related information on separate tabs
 - Tab canvases are made up of one or more tab pages, which have labeled tabs that comprise an equal amount of space on the tab canvas
 - Tab canvases can be used to display large amount of information on a single canvas and give the application a “Web” feel
-

Create via Object Navigator To create a tab canvas in the Object Navigator:

Step	Action
1	In the Object Navigator, position the cursor on the Canvases node, then choose Navigator→Create to insert a new canvas.
2	In the Property Palette, set the Canvas Type property to Tab .
3	Set the other properties of the tab canvas object as described below.
4	Expand the canvas node in the Object Navigator so it displays Tab Pages.
5	Highlight the Tab Pages node and then choose Navigator→Create to insert a new tab page.
6	Set the tab page properties as described below.
7	Create additional tab pages by repeating steps 5 and 6.

Creating via Layout Editor To create a tab canvas in the Layout Editor:

Step	Action
1	Go to the Layout Editor and bring up the content canvas on which you are placing the tab canvas.
2	Click the Tab Canvas icon  on the toolbar.
3	Click and drag the mouse in the canvas to the position where you want to place the tab canvas.
4	Open the Properties Palette and set the properties of the canvas object as described below.
5	Create additional tab pages in the Object Navigator.
6	Set the tab page properties as described below.
7	Create additional tab pages by repeating steps 5 and 6.

Continued on the next page

Tab Canvas, Continued

Tab Canvas Properties

Property	Usage
Visible	Set this to Yes if you want the tab canvas to be visible when the window is invoked, or to No if you want it to be hidden until it is shown in response to navigation or programmatic events.
Width/Height	Specify the size of the canvas. A tab canvas is usually smaller than the content canvas displayed in the same window.
Bevel	Determines whether the tab canvas has a border. A border can visually separate a stacked view from other views displayed in the same window. To make a borderless view, set Bevel to None.
Viewport Width / Viewport Height	Specifies the size of the viewport for the tab canvas.
Viewport X Position / Viewport Y Position	Specifies the x,y display coordinates of the viewport's upper-left corner relative to the upper-left corner of the content viewport currently displayed in the window. The default setting (0,0) displays the stacked viewport at the upper-left corner of the content viewport.
Corner Style	Specifies the shape of the label tabs. Choose from Chambered, Square, and Rounded.
Tab Attachment Edge	Specifies the location where the tabs are attached to the tab canvas.

Tab Page Properties

Property	Usage
Label	Specifies the text label for the tab page.

Continued on the next page

Tab Canvas, Continued

Stacking order If the tab canvas is not being shown programmatically or in response to navigation, make sure that its position in the canvas stacking order places it in front of the content canvas assigned to the same window. If it is not, it will be shown behind the content canvas, and will not be visible at runtime.

The stacking order of canvases in a window is defined by the sequence in which they are listed under the Canvases node in the Object Navigator.

Displaying a Tab Canvas To display a tab canvas in the Layout Editor, choose **View→ Stacked View** and select the appropriate tab canvas.

To hide the tab canvas, hold the [**Shift**] key and select the appropriate tab canvas.

Placing Items on a Tab Page In order to use the new tab canvas you must place individual items onto the tab pages. To accomplish this, do the following:

Step	Action
1	Open the item's Property Palette.
2	Set the item's Canvas Property and Tab Page property to the desired tab canvas and tab page.

Section O: Runform Messages and Alerts

Overview

Purpose By default, Oracle and Forms messages appear to the user during runtime. You can add your own messages, and replace the default error and warning messages that Forms displays.

Objectives This section will examine the following:

- Message types
- Replacing system messages
- Creating and displaying alerts

In this section These topics are covered in this section.

Topic	Page
What kinds of messages are automatically displayed?	O-2
Building your own messages	O-3
Built-in functions that detect success and failure	O-4
Triggers that intercept messages	O-5
Alerts	O-6
Displaying the alert	O-7
Changing the alert message	O-8
Setting a text item during runtime	O-9
Obtaining property values from items	O-10

What kinds of messages are automatically displayed?

Displayed messages

Oracle Forms communicates to the operator for:

- Error messages, such as:
FRM-10212: Login failed for this username and password.
ORA-00913: Too many values
 - Warning messages, such as:
FRM-10205: Menu <menu module name> not found.
 - Working messages, such as:
Attempting to reserve record for update or delete (CTRL-C to cancel)...
-

Building your own messages

**MESSAGE
function**

Displays specified text on the message line.

Syntax

```
MESSAGE (message_string) ;
```

To display the value of a variable with text:

```
message('message string'||variable);
```

Example

```
IF :Account.bill_date > :Account.paid_date THEN  
    MESSAGE('Bill date cannot be greater than date  
paid');  
END IF;
```

Built-in functions that detect success and failure

Functions

Functions used to indicate whether the last action in the form was successful.

Function	Description
FORM_SUCCESS	TRUE: Action was successful FALSE: A error or fatal error occurred
FORM_FAILURE	TRUE: A nonfatal error occurred FALSE: No error or a fatal error occurred
FORM_FATAL	TRUE: A fatal error occurred FALSE: No error or a nonfatal error occurred

Triggers that intercept messages

Trigger	Description
ON ERROR	Fires instead of displaying the system error message.
ON MESSAGE	Fires instead of displaying the informative system message.

Intercepting functions

To capture error details, you can use the built-in functions **ERROR_CODE**, **ERROR_TEXT**, **ERROR_TYPE**.

To capture informative message details, you can use the built-in functions **MESSAGE_CODE**, **MESSAGE_TEXT**, **MESSAGE_TYPE**.

ON-ERROR Example

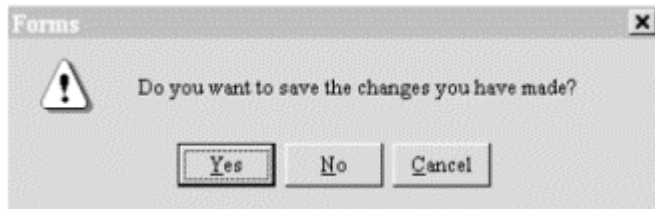
```
DECLARE
  errnum NUMBER          := ERROR_CODE;
  errtxt VARCHAR2(80)   := ERROR_TEXT;
  errtyp VARCHAR2(3)    := ERROR_TYPE;
BEGIN
  IF errnum = 40301 THEN
    Message('Your search criteria identified no
    matches...
           Try Again.');
```

```
    ELSE
      /*
      ** Print the Normal Message that would have
      appeared
      **
      ** Default Error Message Text Goes Here      */
      Message(errtyp || '-' || TO_CHAR(errnum) || ':
      ' || errtxt);
      RAISE Form_Trigger_Failure;
    END IF;
  END;
```

Alerts

Alerts

An alert is a modal window that displays a message notifying the operator of some application condition. The operator must respond to the alert's message by selecting one of the predefined alert buttons. Selecting any button immediately dismisses the alert.



Create an alert

To create an alert:

Step	Action
1	In the Navigator, select the Alerts node and then choose Navigator→Create .
2	In the Property Palette window, set the Alert Style property to the style that corresponds to the severity of your message. Valid choices are Stop , Caution , or Note . At runtime, an icon representing the style you select displays next to the message in the alert window.
3	Set the Message property by entering the message you want the alert to display at runtime. You can enter up to 200 characters.
4	Define one or more buttons for the alert by entering a text label in the Button 1 , Button 2 , and Button 3 fields. (The default text labels are “OK” for Button 1 and “Cancel” for Button 2 .) At least one button must have a label. Buttons that do not have labels are not displayed.
5	Choose the Default Alert Button , either Button 1, Button 2, or Button 3. The default button is the button that is selected implicitly when the operator presses [Accept]. On most window managers, the default button has a distinctive appearance.

Displaying the alert

SHOW_ALERT To display an alert, your application must execute the SHOW_ALERT built-in subprogram from a trigger or user-named subprogram. SHOW_ALERT is a function that returns a numeric constant.

The constant returned by the SHOW_ALERT function indicates which alert button the operator selected and is one of the following: ALERT_BUTTON1, ALERT_BUTTON2, ALERT_BUTTON3.

Example

```
DECLARE
    alert_button NUMBER;
BEGIN
    IF :Account.bill_date > :Account.paid_date THEN
        alert_button :=
SHOW_ALERT('invalid_billing_dates');
    END IF;
END;
```

Changing the alert message

Change an alert message You can change an alert message at runtime by executing the `SET_ALERT_PROPERTY` built-in procedure. Changing an alert's message allows you to reuse the same alert object, but display a different message each time it is invoked.

For example, it is common to have several alerts in an application that all use the same two buttons (for example, OK and Cancel) but that display different messages depending on application context.

Example The following example changes the message of an alert named *generic_alert*. This alert is informational only and has only one button, labeled "OK":

```
DECLARE
    alert_button NUMBER;
BEGIN
    IF :Account.bill_date > :Account.paid_date THEN
        Set_Alert_Property('Date_Alert',
            ALERT_MESSAGE_TEXT,
            'The Bill date cannot be after the date paid. ');
        alert_button := SHOW_ALERT('Date_Alert');
    END IF;
END;
```

Exercise 35 Create an alert called `HELP_ALERT`.

- The alert should be a Note type, with one OK button
- Briefly describe what the form is used for within the message property
- Create a button called help that will show the help alert message
- Assign the canvas to the main canvas

Setting a text item during runtime

Change property values

You may want to change an item's property value during runtime. To do so, use the SET_ITEM_PROPERTY built-in subprogram. To view the parameters, look in the Form Builder help.

```
/* Hide the item */  
SET_ITEM_PROPERTY('ACCOUNT.AMOUNT', DISPLAYED,  
PROPERTY_FALSE);
```

Obtaining property values from items

GET_ITEM_PROPERTY

The GET_ITEM_PROPERTY built-in function returns information about a specified item.

Syntax

```
GET_ITEM_PROPERTY(item_name, property);
```

To obtain more information about the parameters the function accepts, explore the Forms Builder Help.

Example

```
/* Hides the account button if it is displayed. */  
  
IF GET_ITEM_PROPERTY('Control.account_btn', DISPLAYED)  
  = 'TRUE' THEN  
  SET_ITEM_PROPERTY('Control.account_btn', DISPLAYED,  
    PROPERTY_FALSE);  
END IF;
```

WHEN-MOUSE-DOWN trigger

```
/* Obtains the name and coordinates of the item the  
mouse is on.  
*/
```

```
DECLARE  
  item_var VARCHAR2(30);  
  x_posn NUMBER;  
  y_posn NUMBER;  
BEGIN  
  item_var :=  
GET_ITEM_PROPERTY(:SYSTEM.MOUSE_ITEM, ITEM_NAME);  
  x_posn :=  
GET_ITEM_PROPERTY(:SYSTEM.MOUSE_ITEM, X_POS);  
  y_posn :=  
GET_ITEM_PROPERTY(:SYSTEM.MOUSE_ITEM, Y_POS);  
  . . .  
END;
```

Continued on the next page

Obtaining property values from items, Continued

Exercise 36

Alter the trigger so that Help button disappears when the form is in Enter Query Mode. Use the built-in subprogram SET_ITEM_PROPERTY. Examine the parameters it takes by opening the Forms Builder Help.

Section P: Sharing Objects and Code

Overview

Purpose Oracle Forms was designed so that objects and code could be reused and shared.

Objectives This section will examine the following:

- Property Classes
- Object Groups
- Copying and Subclassing Objects

In this section These topics are covered in this section.

Topic	Page
Property Classes	P-2
Creating Property Classes	P-3
Object groups	P-5
Copying vs. Subclassing	P-6

Property classes

Property classes A property class is a named object that contains a list of properties and their settings. Once you create a property class, you can base other objects on it. An object based on a property class can inherit the setting of any property in the class that makes sense for that object.

-
- Characteristics**
- Property class inheritance is a powerful feature that allows you to quickly define objects that conform to your own interface and functionality standards
 - Property classes also allow you to make global changes to applications quickly. By simply changing the definition of a property class, you can change the definition of all objects that inherit properties from that class
-

Creating Property Classes


Methods

There are two ways to create a property class:

- You can create a new property class in the Object Navigator and then add properties to it as desired.
- You can create a property class from an existing list of properties in the Property Palette window.




Create via Object Navigator

To create a property class in the Object Navigator:

Step	Action
1	In the Object Navigator, position the cursor on the Property Classes node and choose Navigator → Create , or click the Create  icon.
2	In the Property Palette window, add properties to the class as desired.

Create via Property Palette window

To create a property class in the Property Palette window:

Step	Action
1	In the Object Navigator or an editor, select one or more objects so that the Property Palette window displays the properties (and settings) that you want to be in the property class.
2	In the Property Palette window, click the Property Class  icon on the toolbar. An alert is displayed to confirm the name of the class being created. Add a property:  Delete a property: 

Continued on the next page

Creating Property Classes, Continued

Inherit property values To inherit property values from a Property Class:

Step	Action
1	Open up the Property Palette for the item you want to apply the properties from the property class.
2	Set the Subclass Information Property to the property class you want to use.


Inherited Property

- Takes its value from the property class that is associated with the object.
- Displayed in the Property Palette with an arrow.

Variant Property

- Has modified the inherited value from the property class that is associated with the object
- Displayed in the Property Palette with a cross over the arrow

Converting Variant to Inherited Properties

To convert a modified value back to the original inherited value, highlight the specific property and click the Inherit Property  icon on the Property Palette.

Exercise 37

Create a property class called DATE_CLASS

- Set the initial value to the database date
 - Set the format mask
 - Set the background color
 - Set **enabled** to *no*
 - Set **keyboard navigable** to *no*
 - Set the **datatype** to *type date*
 - Set the **maximum length** to *11*
-


Object groups

Object groups An object group is a container for a group of objects. You define an object group when you want to package related objects so you can copy or reference them in another module.

Object groups provide a way to bundle objects into higher-level building data blocks that can be used in other parts of an application and in subsequent development projects.

Once you create an object group, you can add and remove objects to it as desired.

Create an object group To create an object group:

Step	Action
1	In the Object Navigator, position the cursor on the Object Groups node and choose Navigator→Create , or click the Create  icon.
2	In the Object Navigator, drag the desired object(s) under the Object Group Children node.

Copying vs. Subclassing

- Characteristics** When you drag objects to copy them from one module to another, an alert is displayed that prompts you to specify whether you want to create a copy of the object or create a subclass object.
- Copying creates a new and separate instance of the object in the target module. Any objects owned by the copied object are also copied
 - Subclassing creates a new object that has true object-oriented capabilities:
 - Inherits from its parent object.
This includes changes to the parent object
 - Ability to alter or override inherited properties from the parent object
 - Ability to add new objects to itself that were not inherited
-

Section Q: Multiple-Form Applications

Overview

Purpose So far, you have run one form at a time to view and modify data. However, rarely would an application consist of one form. In this section, you connect your two forms together by having one call the other.

Objectives This section will examine the following:

- Methods of calling one form from another
- How to establish variables which can be used across forms

In this section These topics are covered in this section.

Topic	Page
About multiple-form applications	Q-2
OPEN_FORM	Q-3
CALL_FORM built-in	Q-4
NEW_FORM built-in	Q-7
Global variables	Q-8
DEFAULT_VALUE	Q-9
Preparing to move to Banner	Q-10
Conversion	Q-11

About multiple-form applications

Multiple-form applications	<p>A multiple-form application is one that is designed to open more than one form during a single Runform session.</p> <p>Every invocation of Runform begins the same way by starting a single form module. Once the first form is loaded into memory and begins execution, it can programmatically invoke any number of additional forms.</p>
How one form finds another	<p>When one form programmatically invokes another, Oracle Forms looks for the new form in the appropriate directory and then loads it into memory.</p> <p>When you deliver a multiple-form application to end users, all of the .FMX, .MMX, and .PLL (form, menu, and library) files that will be called during the session must reside in the working directory or search path defined for your system. In a 32-bit Windows environment, the Forms Path resides in the registry.</p>
Invoking forms	<p>There are three ways that one form can programmatically invoke another form:</p> <ul style="list-style-type: none">• Execute the OPEN_FORM procedure to open an independent form• Execute the NEW_FORM procedure to replace the current form with a different form• Execute the CALL_FORM procedure to call a modal form

OPEN_FORM

Procedure

The first form remains displayed, and operators can navigate between the forms as desired. The new form can share the same database session as the form from which it was invoked, or it can create a separate session of its own.

Syntax

```
OPEN_FORM(form_name [,activate_mode] [,session_mode]  
          [,data_mode] [,paramlist_id]);
```

Parameter	Description
form_name	Specifies the name of the form to open.
activate_mode	<ul style="list-style-type: none">• ACTIVATE: Sets focus to the form to make it the active form in the application.• NO_ACTIVATE: Opens the form but does not set focus to the form. The current form remains current.
session_mode	<ul style="list-style-type: none">• NO_SESSION: Specifies that the opened form should share the same database session as the current form. A COMMIT operation in any form will cause validation and commit processing to occur for all forms running in the same session.• SESSION: Specifies that a new, separate database session should be created for the opened form.
data_mode	<ul style="list-style-type: none">• NO_SHARE_LIBRARY_DATA: At runtime, Form Builder will not share data between forms that have identical libraries attached (At design time).• SHARE_LIBRARY_DATA: At runtime, Form Builder will share data between forms that have identical libraries attached (At design time).
paramlist_name	Specifies the CHAR name of a parameter list to be passed to the opened form.
paramlist_id	Specifies the unique ID that Oracle Forms assigns to the parameter list at the time it is created. Use the GET_PARAMETER_LIST function to return the ID to a variable of type PARAMLIST.

CALL_FORM built-in

Procedure

Runs an indicated form while keeping the parent form active. Oracle Forms runs the called form with the same Runform preferences as the parent form. When the called form is exited Oracle Forms processing resumes in the calling form at the point from which you initiated the call to `CALL_FORM`.

When you call a form, Oracle Forms issues a savepoint for the called form. If the `CLEAR_FORM` function causes a rollback when the called form is current, Oracle Forms rolls back uncommitted changes to this savepoint.

Continued on the next page

CALL_FORM built-in, Continued

Syntax

```
CALL_FORM(formmodule_name [, display] [, switch_menu]
          [, query_mode] [, data_mode] [, paramlist_name]);
```

Parameter	Description
formmodule_name	Specifies the form module name of the called form.
display	<ul style="list-style-type: none"> HIDE causes Oracle Forms to clear the calling form from the screen before drawing the called form. HIDE is the default parameter. NO_HIDE causes Oracle Forms to display the called form without clearing the calling form from the screen.
switch_menu	<ul style="list-style-type: none"> NO_REPLACE causes Oracle Forms to keep the default menu application of the calling form active for the called form. DO_REPLACE causes Oracle Forms to replace the default menu application of the calling form with the default menu application of the called form.
query_mode	<ul style="list-style-type: none"> NO_QUERY_ONLY causes Oracle Forms to run the indicated form in normal mode, allowing the operator to perform inserts, updates, and deletes from within the called form. QUERY_ONLY causes Oracle Forms to run the indicated form in Query Only mode, allowing the operator to query, but not to insert, update, or delete records.
data_mode	<ul style="list-style-type: none"> NO_SHARE_LIBRARY_DATA At runtime, Form Builder will not share data between forms that have identical libraries attached (At design time). SHARE_LIBRARY_DATA At runtime, Form Builder will share data between forms that have identical libraries attached (At design time).
paramlist_id	Specifies the unique ID Oracle Forms assigns when it creates the parameter list. You can optionally include a parameter list as initial input to the called form. The data type of the ID is PARAMLIST.
paramlist_name	The name you gave the parameter list object when you defined it.

Continued on the next page

CALL_FORM built-in, Continued

Example

```
/*
** Built-in:  CALL_FORM
** Example:  Calls a form in query-only mode.
*/
BEGIN
  Call_Form('SOAIDEN',NO_HIDE,NO_REPLACE,QUERY_ONLY);
END;
```

NEW_FORM built-in

Procedure

Exits the current form and enters the indicated form. The calling form is terminated as the parent form.

Oracle Forms runs the new form with the same Runform options as the parent form. If the parent form was a called form, Oracle Forms runs the new form with the same options as the parent form.

Syntax

```
NEW_FORM(formmodule_name [, rollback_mode]
          [, query_mode] [, data_mode] [, paramlist_name] );
```

Parameter	Description
formmodule_name	Specifies the form module name of the called form. The name must be enclosed in single quotes. The data type of the name is CHAR.
rollback_mode	<ul style="list-style-type: none">• TO_SAVEPOINT rolls back all uncommitted changes (including posted changes) to the current form's savepoint.• NO_ROLLBACK exits the current form without rolling back to a savepoint.• FULL_ROLLBACK rolls back all uncommitted changes (including posted changes) that were made during the current Runform session.
query_mode	<ul style="list-style-type: none">• NO_QUERY_ONLY runs the indicated form normally, allowing the operator to perform inserts, updates, and deletes in the form.• QUERY_ONLY runs the indicated form as a query-only form.
data_mode	<ul style="list-style-type: none">• NO_SHARE_LIBRARY_DATA At runtime, Form Builder will not share data between forms that have identical libraries attached (At design time).• SHARE_LIBRARY_DATA At runtime, Form Builder will share data between forms that have identical libraries attached (At design time).
paramlist_id	Specifies the unique ID Oracle Forms assigns when it creates the parameter list.
paramlist_name	The name you gave the parameter list object when you defined it. The data type of the name is CHAR. A parameter list passed to a form via NEW_FORM cannot contain parameters of type DATA_PARAMETER (a pointer to record group).

Global variables

Global variables A global variable is an Oracle Forms variable whose value is accessible to triggers and subprograms in any module that is active during the current session.

A global variable stores a character string of up to 255 characters in length.

Because global variables are accessible throughout an entire session, they are frequently used to keep track of variables used in a multiple-form application.

Declaration Global variables are not formally declared the way PL/SQL local variables are. Rather, you initialize a global variable the first time you assign a value to it:

```
:GLOBAL.pidm := TO_CHAR(:swriden_pidm);
```

Referencing To reference a global variable, prefix the variable name with the word GLOBAL and a colon.

Referencing a global variable that has not been initialized through assignment causes a runtime error.

Destruction To destroy a global variable and release its memory, use the ERASE built-in procedure:

```
Erase('GLOBAL.pidm');
```

DEFAULT_VALUE

Procedure

You can use the DEFAULT_VALUE built-in procedure to assign a value to a variable whose value is NULL. If the value of the indicated variable is not NULL, DEFAULT_VALUE does nothing. If the variable to which the value is being assigned is an undefined global variable, Oracle Forms creates the variable.

Example

The following example creates a global variable named pidm and initializes it to ":

```
Default_Value('','GLOBAL.pidm');
```

Preparing to move to Banner

Standard blocks Standard blocks on a Banner form:

- Form_header
- Key block
- Database datablocks

Form_header Non-database datablock

- Items:
 - workfld
 - callfld
 - workdate
 - current_institution
 - current_release
 - current_form
 - current_date
 - current_time

Key block The first block on most forms contains key information. (Some forms, especially validation forms and certain list forms, do not have a key block.)

The key block determines what is entered or displayed on the rest of the form. All information on the form refers to the key block.

The key block has at least one field and sometimes more. For example, a form that maintains population selection information may have key block fields for both an application and a selection ID.

The key block stays on the form as subsequent blocks appear. Occasionally, another window may appear on top of the key block if the window is unusually large or if the key block is not pertinent to the window.

When the cursor is in the key block, enterable fields in the key block are enabled. When you leave the key block, enterable fields in the key block are disabled.

Database datablocks

Conversion

Process

To begin conversion:

- Rename the swriden block to Key_block
- Change the block to a non-database data block
- Change all items to non-database items

Follow the instructions in Appendix B of the General 5.0 Release guide.

Section R: Answer Guide

Overview

Purpose This section contains answers for the workbook's exercises.

In this section These topics are covered in this section.

Topic	Page
Exercises 1-6	R-2
Exercises 7-11	R-3
Exercises 12-14	R-4
Exercises 15-17	R-5
Exercises 18-19	R-6
Exercises 20-22	R-7
Exercises 23-25	R-8
Exercises 26-27	R-9
Exercises 28-30	R-10
Exercises 31-33	R-11
Exercises 34-36	R-12
Exercise 37	R-13

Answer Guide

-
- Exercise 1** Create a new form module called SWAIDEN. The naming convention follows Banner standards. SWAIDEN stands for:
- Student
 - Custom object
 - Application form
 - Identification
- Select File->New-> Form. Double-click the form name to rename it to SWAIDEN.**
-
- Exercise 2** In the new form module, create a data block based on the SWRIDEN table. Do not include the swriden_change_ind and swriden_activity_date columns. The Identification data block should be displayed on Canvas1. Display Forms Style. Title the Frame Identification and only display one record.
- In the new form, select Tools->Data Block Wizard. When the dialog appears, enter the following properties:**
- **Base Table: SWRIDEN**
 - **Frame Name: Identification**
 - **Style: Form**
 - **Canvas: (New Canvas)**
 - **Records: 1**
- Deselect swriden_change_ind and swriden_activity_date columns so they do not appear.**
-
- Exercise 4** In the Address data block, allow the user to automatically navigate to the next record when tabbing, by modifying the Navigation Style on the data block level.
- Navigation Style: Change Record**
-
- Exercise 5** In the SWRIDEN block, increase the number of records displayed to 5
- Block property: records displayed 5**
-
- Exercise 6** Ensure that the records retrieved in the SWRIDEN data block are current (the swriden_change_ind is NULL) by adding a WHERE clause.
- Block property: where swriden_change_ind is null**
-

Continued on the next page

Answer Guide, Continued

Exercise 7 Sort the records in the SWRIDEN data block by last name.

Block property: order by swriden_last_name

Exercise 8 Make sure the user is unable to insert, update, or delete records in the ID data block.

Block properties insert, update and delete NO

Exercise 9 (No solution)

Exercise 10 Join both the swraddr and swbpers blocks to the swriden pidm

Block property

address block where clause :swriden_pidm = swraddr_pidm

person block where clause :swriden_pidm = swbpers_pidm

Exercise 11 In the SPRIDEN data block, set the canvas to null within the pidm item's property palette.

Pidm property canvas null

Continued on the next page

Answer Guide, Continued

Exercise 12

In both the Address and Person data blocks:

- Alter the activity date so that it initializes to the current database date for a new record. Try using an intersection to set the property for both items at the same time
- For both data blocks, do not allow the activity date to be updated or inserted by the user
- Set the bubble help to **Activity Date**
- Set the format mask so that the date appears like the following: 01-JAN-1998
- Remember to alter the maximum length to allocate for the increase in characters

To create an intersection: within the Object Navigator, highlight activity date in both the Address and Person data blocks.

To highlight items, click the first item, and then hold the [Ctrl] key and click the second item.

Enter the Property Palette by selecting Tools_Property Palette.

Set the following properties:

- **Format Mask: DD-MON-YYYY**
- **Initial Value: \$\$DBDATE\$\$**
- **Maximum Length: 11**
- **Update Allowed: No**
- **Insert Allowed: No**
- **Tooltip: Activity Date**

Exercise 13

In the Person data block, set the format mask for birth_date so that it appears like the following: 01-JAN-1998. Set the bubble help to Birth Date.

Format Mask: DD-MON-YYYY

Maximum Length: 11

Tooltip: Birth Date

Exercise 14

In the Person data block, set the format mask for swbpers_ssn so that it appears like the following at runtime: 123-45-6789. Set Fixed Length to Yes.

Format Mask: 999”-“99”-“9999

Fixed Length: Yes

Maximum Length: 11

Continued on the next page

Answer Guide, Continued

Exercise 15 In the Address data block, set the format mask for the phone number so that it appears like the following at runtime: 555-1212. Set Fixed Length to Yes.

Format Mask: 999”-“9999

Fixed Length: Yes

Maximum Length: 8

Exercise 16 In the Address data block, create a new display item named atyp_desc.

- Ensure that atyp_desc is not a base table item
- Assign the item to Canvas1
- Set the bubble help to **Address Description**
- The display item will be populated by a trigger in a later exercise
- On the canvas, place it to the right of the swraddr_atyp_code

In the Address data block, highlight the Items node and select Navigator->Create. In the property sheet, set the following properties:

- **Item Type: Display Item**
 - **Canvas: Canvas1**
 - **Database Item: No**
-

Exercise 17 In the Person data block, convert the swbpers_confid_ind text item to a check box.

- Set the checked state to represent the base table value of Y and the unchecked state to represent N
- Ensure that new records are automatically assigned the value N
- Allow only those records with swbpers_confid_ind values of Y or N to display
- Resize the checkbox appropriately

Set the following properties for swbpers_confid_ind:

- **Item Type: Check Box**
 - **Initial Value: N**
 - **Label: Confid**
 - **Value When Checked: Y**
 - **Value When Unchecked: N**
 - **Check Box Mapping of Other Values: Not Allowed**
-

Continued on the next page

Answer Guide, Continued

Exercise 18

In the Person data block, convert the `swbpers_mrtl_code` text item to a pop-list list item.

- Add list elements of Single, Married, Widowed, and Divorced to represent database values of S, M, W, and D
- Display any other values as Single
- Ensure that new records display the default value Single
- Resize the list item to see your choices at runtime

Within the `swbpers_mrtl_code` Property Palette, set the following properties:

- **Item Type: List Item**
- **Initial Value: S**
- **Mapping of Other Values: S**
- **List Style: Poplist**

Double-click List Elements, and enter:

List element	List item value
Single	S
Married	M
Widowed	W
Divorced	D

Exercise 19

In the Person data block, convert the `swbpers_sex` text item into a radio group.

- Add radio buttons for Male and Female to represent the database values of M and F
- Define access keys of M for male and F for female
- Define a default value of M for all new records

Enter the following properties for the `swbpers_sex` item:

- **Item Type: Radio Group**
- **Initial Value: M**

Create two radio buttons underneath the group, and set the following properties:

Name	Male	Female
Access Key	M	F
Label	Male	Female
Radio Button Value	M	F

Continued on the next page

Answer Guide, Continued

Exercise 20

Create a control data block.

Create three non-database items and convert all of them to push buttons:

- Button 1
 - Label: Person
 - Add trigger when-button-pressed and enter `go_block('SWBPERS');`
 - Place on canvas1

 - Button 2
 - Label: Return
 - Add trigger when-button-pressed and enter `go_block('SWRIDEN');`
 - Place on canvas2

 - Button 3
 - Label: Exit
 - Add trigger when-button-pressed and enter exit form;
 - Place on canvas1
-

Exercise 21

At the form level, create a trigger to immediately execute a query.

At both block levels for SWRADDR and SWBPERS when the block is entered also, execute a query.

```
Add trigger when-new-form-instance
go_block('swraddr');
execute_query;
go_block('swbpers');
execute_query;
go_block('swriden');
```

Exercise 22

When navigating to the next record in the ID block, get the records in the address block to change as well.

```
Add trigger when-new-record-instance
go_block('swraddr');
execute_query;
go_block('swbpers');
execute_query;
go_block('swriden');
```

Continued on the next page

Answer Guide, Continued

Exercise 23

Write a PRE-FORM trigger in SWAIDEN, which checks the username. If the name does not match your username, fail the trigger and display a message. Use the built-in subprogram GET_APPLICATION_PROPERTY. Attempt to run your form under a different training account. What occurred?

```
IF get_application_property(username) = 'TRAIN03' then
  message('You do NOT have access to this form!');
  raise form_trigger_failure;
end if;
```

Exercise 24

In the Address data block, create a trigger to populate the address description item whenever validation occurs on swraddr_atyp_code (use STVATYP).

Fail the trigger and display a suitable message if the swraddr_atyp_code is not found.

```
begin
  select stvatyp_desc into :atyp_desc;
  from stvatyp
  where stvatyp_code = :swraddr_atyp_code;

exception
  when no_data_found then
    message('Invalid Address Type Code!');
    message('Invalid Address Type Code!');
    raise form_trigger_failure;
end;
```

Exercise 25

In the Address data block, create a trigger to populate the state description on whenever validation occurs.

```
begin
  select stvstat_desc into :swraddr.stat_desc
  from stvstat
  where stvstat_code = :swraddr_stat_code;

exception
  when no_data_found then
    message('Invalid State Code!');
    message('Invalid State Code!');
    raise form_trigger_failure;
end;
```

Continued on the next page

Answer Guide, Continued

Exercise 26

Delete the last_name and first_name items and create one non-database item called name on the swriden block. Create a trigger that will populate the name after an ID is entered. If the ID is invalid, fail the trigger and display a suitable message.

- Delete first name item
- Change last name item to a non-database item
- Change the item name to name

```
BEGIN
SELECT rtrim(swriden_last_name,' ')||', '||
       rtrim(swriden_first_name,' ')||' '||
       swriden_mi
       INTO :name
       FROM swriden
       WHERE swriden_id = :swriden_id;
EXCEPTION
WHEN NO_DATA_FOUND THEN
    Message('Invalid ID. Please enter again.');
```

```
RAISE FORM_TRIGGER_FAILURE;
END;
```

Exercise 27

Create a post-query trigger on the SWRADDR block to populate the atyp_desc field.

```
declare
cursor c1 is
    select stvatyp_desc
    from stvatyp
    where stvatyp_code = :swraddr_atyp_code;

begin
    open c1;
    fetch c1 into :swraddr.addr_desc;

exception
when no_data_found then
    message('Invalid Address Type Code!');
    raise form_trigger_failure;

close c1;

end;
```

Continued on the next page

Answer Guide, Continued

Exercise 28

Ensure that the Exit button has no effect in Enter Query Mode.

Highlight the WHEN-BUTTON-PRESSED trigger underneath the Exit_Btn.

Enter the property sheet and set the following:

- **Fire in Enter Query Mode: No**

Exercise 29

In both the Address and Person data blocks, create triggers that populate the activity date and pidm when the user saves a new record.

Create a PRE-INSERT trigger defined under the Address and Person data blocks.

Enter the following code:

```
:swraddr_activity_date := sysdate;
:swraddr_pidm := :swriden_pidm;

:swbpers_activity_date := sysdate;
:swbpers_pidm := :swriden_pidm;
```

Exercise 30

In both the Address and Person data blocks, create a trigger that populates the activity date when the user updates an existing record.

Create a PRE-UPDATE trigger defined under the Address and Person data blocks.

Enter the following code:

```
:swraddr_activity_date := sysdate;
:swbpers_activity_date := sysdate;
```

Continued on the next page

Answer Guide, Continued

Exercise 31 Add an address type button to invoke the stvatyp_lov.

Create a push button in the control_block.

Add when-button-pressed-trigger

```
DECLARE
    misc boolean ;

BEGIN
    misc := show_lov ('STVATYP_LOV') ;
END ;
```

Exercise 32 Create a new block and canvas called Comment and add items from the sprcmtt table. Change the canvas type to stacked. Don't forget the swrcmnt_cmtt_code LOV.

- **Right click data blocks**
- **Select data block wizard**
- **Include all columns**
- **In the layout wizard**
- **Select new canvas type stacked**
- **Display all items**

Exercise 33 Create a new window called Window2 in the SWAIDEN form, suitable for presenting Comment.

- Associate comment with WINDOW2
- Modify the height and width properties
- Add a title for the window
- Define a display position that is different than Window1, so that it appears below the name information displayed on CANVAS1
- Set the Hide on Exit property to Yes

- **Title: Comment**
- **Primary Canvas: Comment**
- **Hid on Exit: Yes**
- **Physical X position: 0**
- **Physical Y position: 275**
- **Width: 530**
- **Height: 128**

Continued on the next page

Answer Guide, Continued

Exercise 34

Associate CANVAS2 (Personal Information) with Window2.

- Run the form to see the differences.

Exercise 35

Create an alert called HELP_ALERT.

- The alert should be a Note type, with one OK button
- Briefly describe what the form is used for within the message property
- Create a button called help that will show the help alert message assign the button to the main canvas

- **Highlight alerts**
- **Click the create icon**
- **Title the alert Help**
- **ID, address and personal information form**
- **Create an item in the control block and change the type to push button**
- **Create a when-button-pressed trigger on the button and add:**

```
declare
  alert_button number;

begin
  alert_button := show_alert('HELP');
end;
```

Exercise 36

Alter the trigger so that the Help button disappears when the form is in Enter Query Mode. Use the built-in subprogram SET_ITEM_PROPERTY. Examine the parameters it takes by opening the Forms Builder Help.

```
If system.mode = 'Enter-Query' then
    set_item_property('Help', displayed,
PROPERTY_FALSE);
end if;
```

Continued on the next page

Answer Guide, Continued

Exercise 37

Create a property class called DATE_CLASS

- Set the initial value to the database date
 - Set the format mask
 - Set the background color
 - Set enabled to no
 - Set keyboard navigable to no
 - Set the datatype to type date
 - Set the maximum length to 11
-