



©2016 Capgemini. All rights reserved.
The information contained in this document is proprietary and confidential.
For Capgemini only.

Document History

Date	Course Version No.	Software Version No.	Developer / SME	Change Record Remarks
12-Jan-2012	1.0		Amit Sali	Content Creation
20-Jan-2012			CLS Team	Review
25-July-2016	1.1		Sachin Narandekar	Content Creation



Copyright © Capgemini 2015. All Rights Reserved. E

Course Goals and Non Goals

- Course Goals

- Learn various aspects of Oracle Forms and Reports
 - Develop Oracle Forms and Reports using Developer 2000

- Course Non Goals

- Registration and use Forms and Reports in Oracle Application.



Pre-requisites

- Knowledge of Structured Query Language (SQL)
- Knowledge of PL/SQL as programming language



Copyright © Capgemini 2015. All Rights Reserved. 8

ERP- Oracle Apps

▪ Programmers and Analysts



Capgemini
CONSULTING TECHNOLOGY SERVICES

Copyright © Capgemini 2013. All Rights Reserved.

Day Wise Schedule

■ Day 1

- Lesson 1: Introduction to Oracle Reports and Report Wizard
- Lesson 2: Types of Reports and Report Preview

■ Day2

- Lesson 3: Designing Report

■ Day3

- Lesson 4: Triggers in Oracle Reports
- Lesson 5: Built in Packages and Miscellaneous Settings

■ Day4

- Lesson 6: Introduction to Oracle Forms
- Lesson 7: Items



Copyright © Capgemini 2013. All Rights Reserved. 8

Day Wise Schedule

■ Day 5

- Lesson 8: Windows and Canvases
- Lesson 9: Triggers

■ Day 6

- Lesson 10: Flexible Code
- Lesson 11: Sharing Code



Copyright © Capgemini 2015. All Rights Reserved. T

Table of Contents

- Lesson 1: Introduction to Oracle Reports and Report Wizard
 - 1.1: Introduction to Reports
 - 1.2: Report Wizard
- Lesson 2: Types of Reports and Report Preview
 - 2.1: Types of Reports
 - 2.2: Running a Report
 - 2.3: Report Preview



Copyright © Capgemini 2013. All Rights Reserved. 8

Table of Contents

- Lesson 3: Designing Report
 - 3.1: Introduction
 - 3.2: Basic Components of Report Builder
 - 3.3: Data Model
 - 3.3.1: Queries
 - 3.3.2: Groups
 - 3.3.3: Links
 - 3.3.4: Columns
 - 3.3.5: Parameters
 - 3.4: Layout Model



Copyright © Capgemini 2015. All Rights Reserved. 8

Table of Contents

- Lesson 6: Introduction to Oracle Forms
 - 6.1: Introduction to Forms
 - 6.2: Form Design
 - 6.3: From Builder and its Attribute
- Lesson 7: Items
 - 7.1: Types of Items
 - 7.2: Input Items
 - 7.3: Non-Input Items
- Lesson 8: Windows and Canvases



Copyright © Capgemini 2015. All Rights Reserved. 10

Table of Contents

- Lesson 9: Triggers
 - 9.1: Introduction to Triggers
 - 9.2: Trigger Example
 - 9.3: Query Trigger
 - 9.4: Validation Trigger
 - 9.5: Navigation Trigger
 - 9.6: Transaction Trigger
 - 9.7: Messages and Alerts Trigger
- Lesson 10: Flexible Code
- Lesson 11: Sharing Code



Copyright © Capgemini 2015. All Rights Reserved. 11

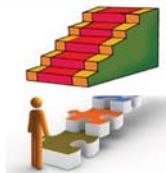
References

- Student Guide
 - All slides presented during lecture along with explanation
- Lab Guide
 - Hands-on lab exercises with sample solutions
- CDROM
 - Pre-requisite Files / Scripts for Assignments
- Reference Material



Next Step Courses (if applicable)

- Oracle Application as ERP



Copyright © Capgemini 2015. All Rights Reserved. 13

Other Parallel Technology Areas

- NA



Copyright © Capgemini 2015. All Rights Reserved. 14

ERP- Oracle Apps Forms and Reports

Lesson 1: Introduction to
Oracle Reports and Report
Wizard

Lesson Objectives

- To understand the following topics:
 - Understand the features of Developer 2000 Reports
 - Report Wizard
 - Tabular Report



1.1: Introduction to Reports

Features of Reports

- Report Wizard
- Template
- Template Editor
- Data Model and Layout
- Object Navigator
- Functions
- Support for fonts, colors and graphics
- Conditional printing capabilities
- Live Previewer

 Capgemini
Consulting Services for Oracle

Copyright © Capgemini 2010. All Rights Reserved.

Report Wizard

The Report Wizard enables you to quickly and easily define the report. Report Builder uses what you specify on each page of the wizard to create a data model and layout for the report. After the wizard has created the report, you can modify it as you would any other report.

Template

Templates are .TDF files, which have common characteristics and objects that you want to apply to multiple reports.

Template editor

Template editor is the work area in which you can select and edit chart and field templates. A display has only one Chart Template editor. Template Editor allows you to create, rename and delete chart templates, and set frame, field template, axis, and other properties. To customize chart templates, you use the Chart Template Editor.

Data Model

The data retrieved for a report is defined in the data model. The data model is composed of objects that define the data to be included in a report. Data model objects include Queries, Columns, Groups, Links, Parameters

Layout Model

The Layout Model view is a work area in which you can define the format of your report output. The style of a report is defined in the layout model. You can work with layout objects, such as frames, repeating frames, fields, boilerplate, anchors, and Graphics objects.

Object Navigator

The Object Navigator is a tool designed for customizing reports. It provides a high-level view of your current Report Builder session and provides a hierarchical view of all the objects contained in or referenced by your current report. You can use the Object Navigator to locate, inspect, create, copy, and delete objects. **From Object Navigator you can:** Select different nodes, Expand and collapse nodes ,Display editors (Data Model, Report editor), Choose between views (Ownership view and Object type views).

Live Previewer

The Live Previewer is a work area in which you can preview your report and manipulate the actual or live data at the same time. In the Live Previewer you can customize reports interactively, meaning that you can see the results immediately as you make each change. To activate buttons in the Live Previewer, you must display the report output in the Runtime Previewer. **Using Live Previewer you can:** Insert date and time, Insert page numbers, Edit text , Change colors, Change fonts ,Set format masks, Align columns Resize columns ,Move columns

1.1: Introduction to Reports Report Types

- The second important thing in report designing is to find out in which style the report is going to run
- The common report styles are
 - Tabular
 - Master-detail
 - Master and Multiple details
 - Matrix



Copyright © Capgemini 2010. All Rights Reserved 5

Tabular Report

A tabular report is the most basic type of report. Tabular layout displays information with a column heading at the top of the page and rows of data underneath the heading. It is a most common style, also they provide simple columnar lists of data. Each column corresponds to a column selected from the database.

Master-detail

A simple Master -Detail report contains two groups of data Master Group ,Detail Group. For each master record fetched, only the related detail records are fetched. In Master/Detail reports for every master group, related detail records are fetched. A relationship between the two queries must be established.

Matrix

A matrix (crosstab) report contains one row of labels, one column of labels, and information in a grid format that is related to the row and column labels.

1.2: Report Wizard

What is it?

- Report Wizard
 - To speed development and help you get started generating objects for your report, Report Builder provides a Report Wizard.
 - The wizard guides you step-by-step through the process of creating a report. You specify the options you want, and the Report Wizard does the rest, creating the appropriate data model and layout model, and displaying the report output automatically in the Live Previewer

 Capgemini
INNOVATIVE INFORMATION TECHNOLOGIES

Copyright © Capgemini 2010. All Rights Reserved. 6

Report Wizard

A powerful quality of the Report Wizard is its ability to operate in re-entrant mode. This means you can use Report Wizard to modify an existing report. To invoke the Report Wizard in re-entrant mode, open an existing report and choose Tools -> Report Wizard.

The Report Wizard provides six buttons to help you navigate your way through its pages:

1. Cancel - Cancel all changes you have made (if any) and exit the wizard
2. Help - Display online help text for the current page of the wizard
3. Back - Navigate to the previous page in the wizard
4. Next - Navigate to the next page in the wizard
5. Apply - Apply changes you have made (re -entrant wizard only)
6. Finish

Save changes you have made with the wizard (if any), and exit the wizard. If you click Next before you have entered all necessary information for a particular wizard page, the wizard prevents you from navigating to the next page. Similarly, if you have not entered all necessary information into the wizard when you click Apply or Finish, the wizard automatically places you onto the page where you can finish entering the required data.

Invoking Report Builder

To Invoke Report Builder

- Go to Programs □ Start □ Oracle Reports 6i □Report Builder
- Report Builder is a tool which makes it easy to design, publish, and distribute professional, production-quality reports in a variety of formats to meet any business need



Copyright © Capgemini 2010. All Rights Reserved 7

Invoking Report Builder

- To speed development and help you get started generating objects for your report, Report Builder provides a Report Wizard. You specify the options you want, and the Report Wizard does the rest, creating the appropriate data model and layout mode, and displaying the report output automatically in the Live Previewer.
- With Report Builder, you can create reports in practically any format, including tabular Master/detail, matrix, form-like, mailing label, form letter, or a combination of styles.
- Go to the Reports Builder from Start -> Programs -> Developer 2000 R 2.0 -> Reports Builder of Windows 95 machine. When first time you are visiting the report builder it will display the welcome screen of report builder.

Invoking Report Builder

- From Developer/2000 menu
 - From Project Builder's tool bar
 - Report Builder: Tools -> Report Wizard
- NOTE:
- Tabs are different for each report style
 - Wizard preserves all previous settings



Copyright © Capgemini 2010. All Rights Reserved.

Invoking Report Wizard

- Report Wizard leads you through the steps to build a report. It takes you from a blank report to a running report, allowing you to specify data source, which fields to display, group fields, and any totals you wish.
- When you invoke Report Builder for the first time from Developer 2000 V 2.0' then Report Builder displays the 'Welcome to the Report Wizard' dialog box.
- The option Use the report wizard is automatically selected for the first time, so click on the OK button. Now you will be placed in the Report Wizard's Welcome Screen.
- The Report Wizard is tabbed when you use it on an existing report, enabling you to quickly go to any page.
- You can use it on an existing report to modify settings as well as to build a new report from scratch.

Pages

- Report Style: Select tabular
- Data Query: Enter a select statement to retrieve the report data
- Fields: Select the fields that you want to display
- Totals: Select the fields that you want to summarize
- Labels: Alter the labels that appear for each field, and the width of each field
- Template: Template contains standard information such company logo, date etc. This enforces corporate standards



Copyright © Capgemini 2010. All Rights Reserved



Report Style

Oracle Reports provides eight different report-style formats by default. They include tabular, Form like, form letter, mailing label, group left, group above, and matrix with groups and matrix. The default report style is tabular.

Fields

Fields are the placeholders for parameters, columns and such values as the page the page number, current date etc. They define the formatting attributes for all columns displayed in the report. A field is one of the objects that can be located inside a frame or repeating frame. If the column in the report does not have an associated field, then its values will not appear in the report layout editor.

There are two types of fields

1. Default fields
2. User-defined Fields

Default Fields

Whenever user accepts default layout dialog box, Oracle Reports automatically creates one field for each column, and places each field inside of a repeating frame.

User-defined Fields

User can create a field in the layout editor by selecting the field tool and then you can set the property of that field for e.g. to display page numbers, to display current date etc

Totals

- Totals tab is used to create some computation columns or to perform some computations on the available fields.
- In the 'Totals' tab whenever you select any field and its related function then report builder will automatically create the summary column along with its properties.

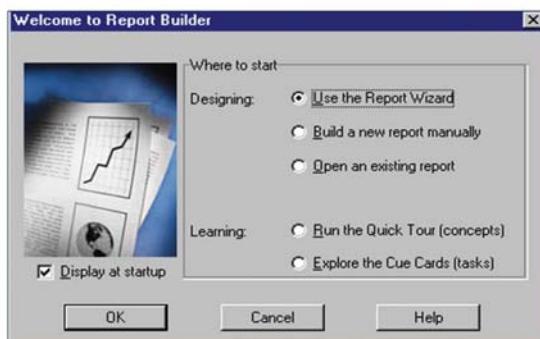
Labels

In the labels tab you can change the labels for the fields.

Template

A Template is an object that you can use to format a module with basic options or contents. If you have business standards and development guidelines, build them into templates that you can reuse in building modules. You can create a report that has all the standard settings, then developers can copy that module as a template to start developing their reports.

Creating a Tabular Report



Copyright © Capgemini 2010. All Rights Reserved 31

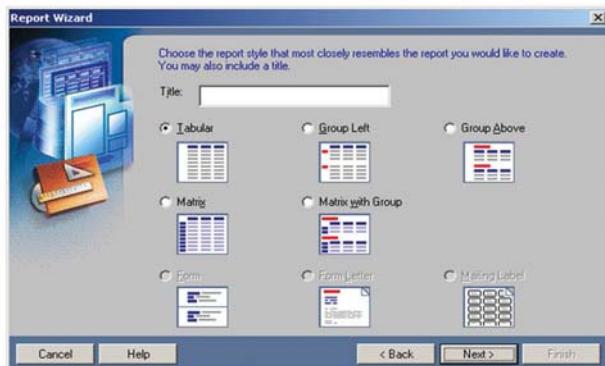
Tabular Report

A tabular report is the most basic type of report. Tabular layout displays information with a column heading at the top of the page and rows of data underneath the heading. It is a most common style, also they provide simple columnar lists of data. Each column corresponds to a column selected from the database.

Steps

- Go to the Reports Builder from Start → Programs → Developer 2000 R 2.0 → Reports Builder of Windows 95 machine.
- The option Use the report wizard is automatically selected for the first time, so click on the OK button.
- Click on the NEXT button. It will take you to the available styles of reports. By default the tabular style will be highlighted. Since you are creating the tabular report so click on the NEXT button

Creating a Tabular Report (contd..)



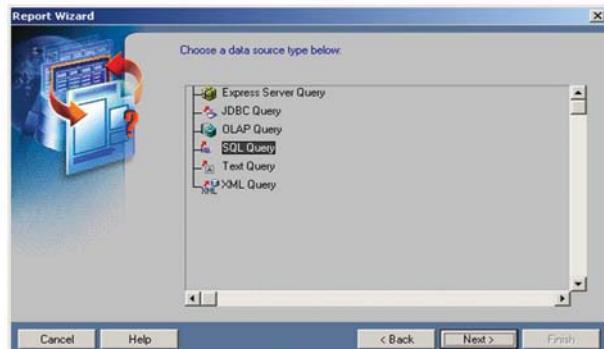
Creating a Tabular Report (contd..)

- The above dialog box is displayed.
 - Click the desired Report Style
 - Click Next to continue



Copyright © Capgemini 2010. All Rights Reserved 13

Creating a Tabular Report (contd..)



Copyright © Capgemini 2010. All Rights Reserved 34

Select SQL Statement. Click Next to continue.

Creating a Tabular Report (contd..)

- Select SQL Statement
- Click Next to continue. The following screen appears.



Copyright © Capgemini 2010. All Rights Reserved 15

Report Builder allows you to build the query by three ways:

1. SQL select statement box

You can define the query by writing a SQL statement in the SQL select statement box.

2. Query Builder

Query Builder is an easy-to-use data access tool designed for analysts, managers, and other business professionals.

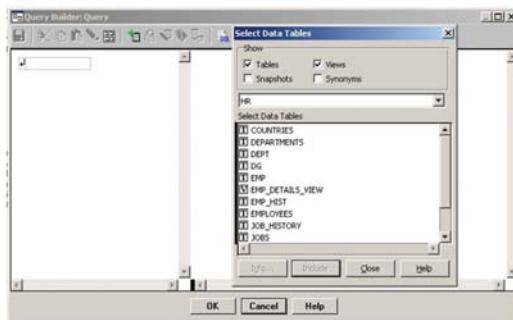
3. Import SQL query

If you have stored a query in a .SQL file into some directory then you can import SQL file by specifying the proper path through the 'BROWSE' button.

- Click the Query Builder button, the movement you click on the Query Builder button, Reports will ask you for connection because you are extracting the data from the database. If you are not already connected to the database then the connection dialog box will appear.
- Enter the username, password and connection string of the database in the appropriate position.

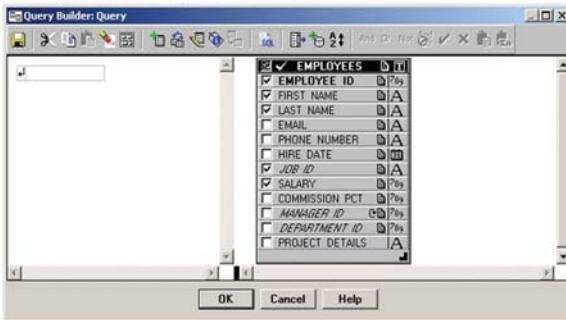
Creating a Tabular Report (contd..)

- Click the Query Builder button. Following screen appears.



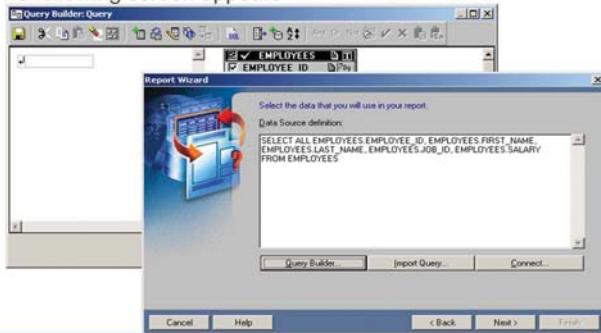
1.2: Report Wizard Creating a Tabular Report (contd..)

- Select the Customer_Master table and say Include and Close
- Following Screen appears



1.2: Report Wizard Creating a Tabular Report (contd..)

- Select the items required in the Report. Click OK
- The following screen appears



Copyright © Capgemini 2010. All Rights Reserved 10

Building a Query

- Query Builder is an easy user interface for writing SQL.
- Query Builder toolbar:
 - Column Sequence:
 - Defines the sequence of column names in the select clause
 - Define Column
 - Defines a new, derived or calculated column
 - Sort
 - Defines the Order By clause



Copyright © Capgemini 2010. All Rights Reserved 10

Query Builder

- Query Builder is an easy-to-use data access tool designed for analysts, managers, and other business professionals.
- It provides a logical and intuitive means to access information from your organization's databases for analysis and reporting.
- Query Builder is designed for professionals who do not have a computer programming or database background. Because of its powerful query features and its support of Structured Query Language (SQL) statements, experienced database users and programmers will find that Query Builder serves many of their needs as well.
- You can use Query Builder to define almost any query that you would build using a SQL SELECT statement.
- Query Builder automatically generates the appropriate SELECT FROM [table. column] clause based on columns displayed in the Query Builder workspace.

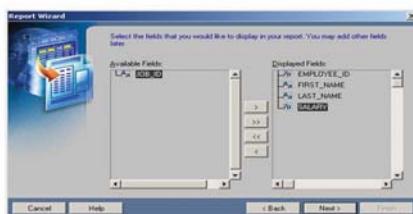
Building a Query (contd..)

- Query Builder toolbar:
- Logical operators for Where and Having clauses
 - Place the cursor in the Condition Box on the left side of the Query Builder window. Type a condition. Use the AND, OR and NOT buttons to create compound conditions

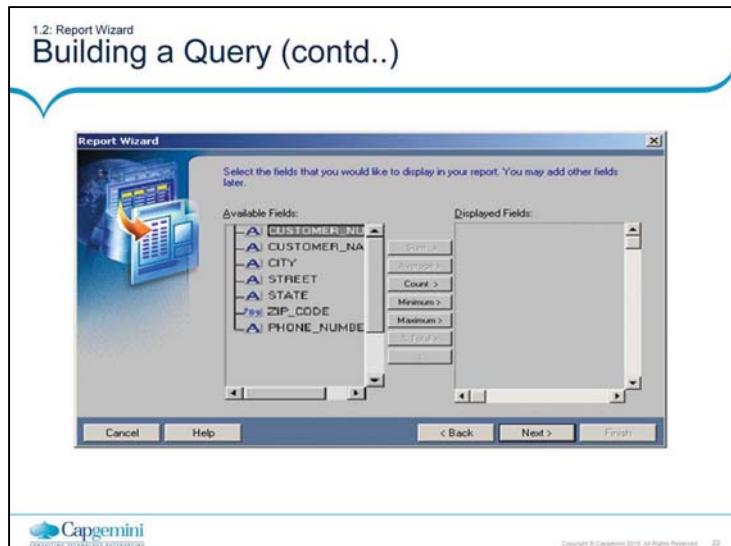


1.2: Report Wizard Building a Query (contd..)

- Click Next to continue.
- The following screen appears.
- Select the desired fields to be displayed in the report
- Click Next



- Click Next to continue.
- Now you will be placed in the field's screen i.e. you are able to choose the field from the specified table. Select all the available fields and put them into the displayed field box.



- Here Reports builder will allow you to create the computations for the report.
- If you want to perform this operation then click a field in the Available Fields list and click the appropriate button for the function you want to apply to the field. An entry with the field and function name appears in the Totals List.
- You can also remove a field from the Totals list, click it then click the left arrow button.

Building a Query (contd..)

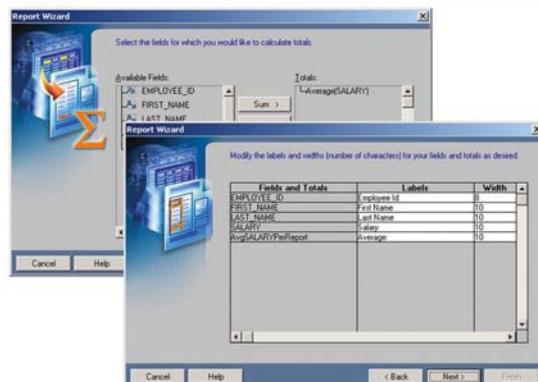
- The above Dialog Box is used to do the calculation part. You can choose any function on any of the items having the number data type
- Click Next to continue



Copyright © Capgemini 2010. All Rights Reserved 23

- Click Next to continue
- The next screen allows you to enter the labels and width for the field names

Building a Query (contd..)



Building a Query (contd..)

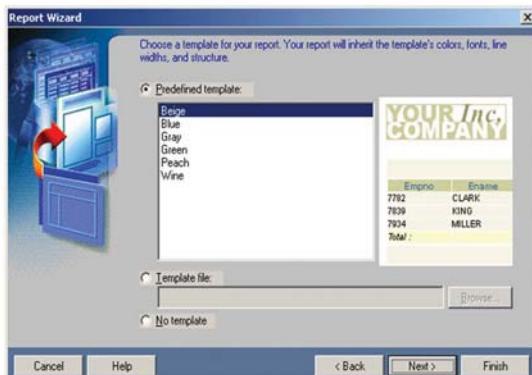
- In the above dialog box you can change the label name and width of the item
- Click next



Copyright © Capgemini 2010. All Rights Reserved 25

- Click on the Next button, the next screen allows you to define the templates for the report.

Building a Query (contd..)



Copyright © Capgemini 2010. All Rights Reserved 20

Templates

Templates are .TDF files, which as common characteristics and objects that you want to apply to multiple reports.

For example, you can define a template that includes the company logo and sets fonts and colors for selected areas of report.

- In this screen you have the option of applying a template to the report.
- When you choose a template, objects in the header and trailer pages, and margin area of a template are imported into the same location in the report.
- The characteristics (formatting, fonts, colors, etc.) of objects in the body area of the template are applied to objects in the body area of the report. You can also create your own user-defined templates and add through the browse button into your report.

Building a Query (contd..)

- In the above dialog box Select any of predefined templates or a template saved in the file system or click No Template option if you don't want to use any template
- Click Finish to view the report
- The report output is shown in the following page



Copyright © Capgemini 2010. All Rights Reserved. 27

- Now you are ready to exit the Report Wizard and can continue to work on the report.
- When you click Finish, Report Builder will close the Report Wizard and automatically displays the report in the Live Previewer.
- In the Live Previewer you can modify your report, or you can choose Tools -> Report Wizard to reenter the wizard and make further modifications there.

ERP- Oracle Apps Forms and Reports Introduction to Oracle Reports and Report Wizard

Preview

The screenshot shows a report titled "Counter New" in the "Report Editor - Paper Design" window. The report has a header section with the text "YOUR Inc COMPANY" in a green box. Below the header is a table with the following data:

Employee Id	First Name	Last Name	Salary
100	Steven	King	24000
101	Neena	Kochhar	17000
102	Lex	De Haan	17000
103	Alexander	Hunold	9000
104	Bruce	Ernst	6000
105	David	Austin	4800
106	Valli	Pataballa	4800
107	Diana	Lorentz	4200
108	Nancy	Greenberg	12000
109	Daniel	Faviet	9000
110	John	Chen	8200

Capgemini
ENTERPRISE INTELLIGENCE

Copyright © Capgemini 2010. All Rights Reserved. 28

Demo

- Create a Tabular Report using Report wizard



Copyright © Capgemini 2010. All Rights Reserved 29

Add the notes here.

Summary

- In this lesson, you have learnt:
 - Basic Components of Report Builder
 - Report Builder wizard
 - Tabular Report



Review Questions

- Question 1: _____ guides you step-by-step through the process of creating a report.
- Question 2: _____ is an easy user interface for writing SQL.



ERP- Oracle Apps Forms and Reports

Lesson 2: Types of Reports
and Report Preview

Lesson Objectives

- To understand the following topics:
 - Types of Reports
 - Running of Report
 - Preview Options in Report Builder



2.1: Types of Reports

Tabular/Master Detail Reports

Tabular report

List of Products

Product Number	Description	Price
NNNN	XXXXXXXXXX	9999.99



Master-detail report

Outstanding Customer Items

Customer Name:	Product	Price
XXXXXXXXXX	NNNN	9999.99
XXXXXXXXXX	NNNN	9999.99

Customer Name:	Product	Price
XXXXXXXXXX	NNNN	9999.99
XXXXXXXXXX	NNNN	9999.99
XXXXXXXXXX	NNNN	9999.99



 Copyright © Capgemini 2010. All Rights Reserved. 3

Tabular Report

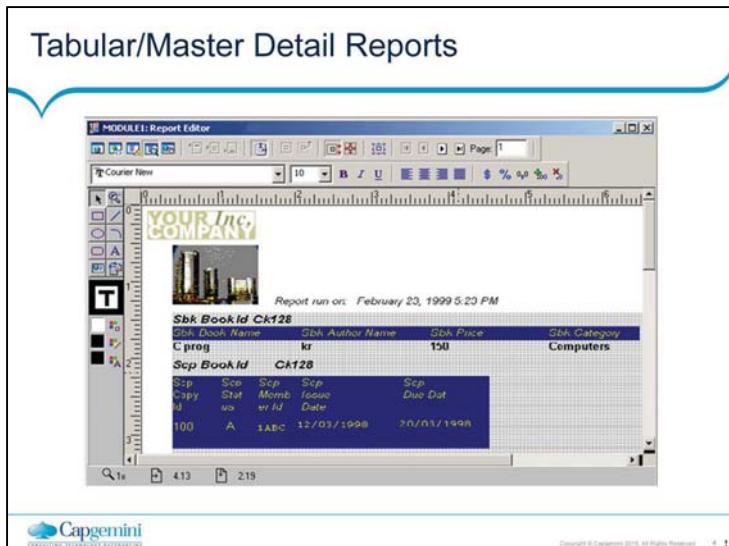
A tabular report is the most basic type of report. Tabular layout displays information with a column heading at the top of the page and rows of data underneath the heading. It is a most common style, also they provide simple columnar lists of data. Each column corresponds to a column selected from the database. A tabular style will be simple report in which a simple table of columns and rows with no groups will be displayed.

Master –Detail

A simple Master -Detail report contains two groups of data.

- Master Group
- Detail Group

For each master record fetched, only the related detail records are fetched. In Master/Detail reports for every master group, related detail records are fetched. A relationship between the two queries must be established.



Types Of Reports

- Oracle Reports provides eight different report-style formats by default.
- They include **tabular**, **Form like**, **form letter**, **mailing label**, **group left**, **group above**, and **matrix with groups and matrix**. The default report style is tabular.

Master with Two Detail Report

▪ Master with Two Detail Report

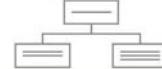
Customer Statistics

Customer Name: XXXXXXXXXX

Outstanding items		Orders in last six months		
Product	Price	Order	Date	Qty
NNNN	9999.99	xxxxx	ddmmyy	nn
NNNN	9999.99	xxxxx	ddmmyy	nn
		xxxxx	ddmmyy	nn

Customer Name: XXXXXXXXXX

Outstanding items		Orders in last six months		
Product	Price	Order	Date	Qty
NNNN	9999.99	xxxxx	ddmmyy	nn
NNNN	9999.99			.



 Capgemini
CONSULTING INTEGRATION CONSULTING

Copyright © Capgemini 2010. All Rights Reserved. 5

Group Left and Above

- Group Left

<u>Region</u>	<u>Department</u>	<u>Last Name</u>	<u>Title</u>
xxxxx	xxxxxxxx	xxxxxx xxxx	
		xxxxxxxx	xxx

- Group Above

Region	xxxxxxxx
Department	xxxx
<u>Last Name</u>	<u>Title</u>
xxxx	xxxxxxxx
xxxxxxxx	xxxx

DATAWAREHOUSE EDUCATION

Copyright © Capgemini 2010. All Rights Reserved.

Group left

A group left report divides the rows of a table into “sets”, based on a common value in one of the columns. Use this type of report to restrict a column from repeating the same value several times while values of related column change. A group left reports can be a tabular report with one or more group columns that the group frame displays on the left of the grouped data records.

Group above

A group above report contains two or more groups of data. For every value of the master group, the related values of the detail group(s) are fetched from the database. Group above reports are also similar to the tabular reports but with one or more group columns that the group frame displays above the grouped data records.

Group Left

MODULE: Report Editor

Report run on: February 24, 1999 12:31 PM

Category	Book Id	Book Name	Author Name
Computers	SO161	Tuning Guide	Oracle
	dk127	dbms	Koth
	Ck128	C prog	kr
	oo126	oracle dba	oracle press
Fiction	tc141	thimbirds	colleen mc culloch
	th121	to kill a mockingbird	harper lee
	me122	malory towers	enid blyton
	jR73	JLS	Richard Bach
	ir101	illusions	richard bach
	hf124	hardy boys	fmlyn w. dixon
	abc	All quiet on the western	bbbb

Capgemini
ENTERPRISE INTEGRATION EXPERTS

Copyright © Capgemini 2010. All Rights Reserved 7

Group Above

The screenshot shows the Oracle Report Editor interface with the title "MODULE1: Report Editor". The report is titled "BREAK REPORT" and is run on "February 24, 1999 12:07 PM" under the group "GROUP ABOVE". The report displays two sections of data:

Book Id	Category
Ck128	Computers
C prog	kr
Total:	150
Minimum:	150
Maximum:	150
% of Total:	3.59%
Book Id	JR73
JLS	Richard Bach
Total:	250
Minimum:	250

At the bottom of the report, there is a note: "This report was generated by Oracle Reports 10g".

Capgemini
ORACLE INTEGRITY EDUCATION

Copyright © Capgemini 2010. All Rights Reserved.

Matrix Report

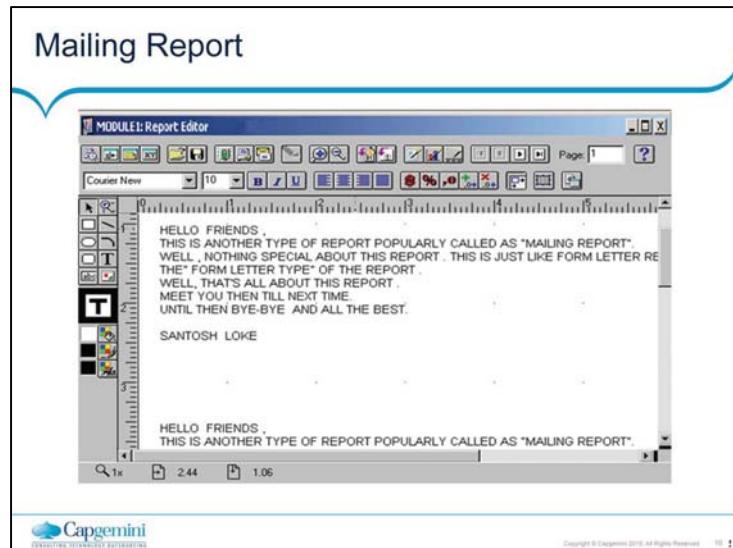
		Customer Matrix				
Product⇒ Customer ↓		NNNN	NNNN	NNNN	NNNN	NNNN
XXXXXXX		9999	9999	9999	9999	9999
XXXXXXX		9999	9999	9999	9999	9999
XXXXXXX		9999	9999	9999	9999	9999
XXXXXXX		9999	9999	9999	9999	9999
XXXXXXX		9999	9999	9999	9999	9999
XXXXXXX		9999	9999	9999	9999	9999
XXXXXXX		9999	9999	9999	9999	9999

Capgemini
CONSULTING INTEGRATED SERVICES

Copyright © Capgemini 2010. All Rights Reserved.

Matrix Report

- A matrix (crosstab) report contains one row of labels, one column of labels, and information in a grid format that is related to the row and column labels.
- A distinguishing feature of matrix reports is that the number of columns is not known until the data is fetched from the database.
- To create a matrix report, you need at least four groups: one group must be a crossproduct group, two of the groups must be within the cross-product group to furnish the “labels”, and at least one group must provide the information to fill the cells.
- The groups can belong to a single query or to multiple queries. Matrix layout is like a grid or a spreadsheet. It contains a row of headings, a column of heading. Matrix reports are sometimes referred to as cross-tabular reports.
- It includes layout fields as labels across the top and down the left side. User can also create the additional layout fields, where the column intersects.



Mailing Report

A mailing label report prints mailing labels in multiple columns on each page. You can print the labels across the page and then down, or down and then across. It organizes each record immediately following the preceding record. It does not include field labels. The fields are placed on the page along the left side and then down.

A mailing label consists of data displayed in a format suitable for use as address labels on envelopes. The labels can be printed in one or many columns, and can begin at any position. You can make the labels print down the page, across the page, across the page and then down, or down and then across. Mailing labels can be created using the simple, one-query reports with a special layout style. To fetch the data for a mailing label report, all you need to do is create a query to select it.

2.2: Running a Report

Ways to run a report

- There are many ways to run a report like:
 - A front end menu
 - A customized menu in a forms applications
 - A button in a forms application
 - Using report runtime in the windows menu
 - Through a web browser



Copyright © Capgemini 2010. All Rights Reserved 31

Output Destination

- Report Destinations:
 - Screen: format with screen fonts
 - Preview: format with printer fonts
 - File: choose file format
 - Mail: MAPI-compliant
 - Printer: output direct to printer



Copyright © Capgemini 2010. All Rights Reserved 12

2.3: Report Preview

Preview Options

The screenshot shows the Oracle Report Editor interface. A bar chart is displayed with the following data:

Author Name	Count
Korth	378
Richard Bach	250
JLS	900

Three numbered callouts point to specific features:

- 1: Zoom In/Out button
- 2: Page Traversing buttons (FIRST, NEXT, etc.)
- 3: Help: PREVIEWER button

Capgemini
ORACLE APPS FORMS AND REPORTS

Copyright © Capgemini 2010. All Rights Reserved. 13

Report Previewer

The report Previewer is a work area in which you can preview your report and manipulate the actual or live data at the same time. In the report Previewer you can customize reports interactively, meaning that you can see the results immediately as you make each change. To activate buttons in the report Previewer, you must display the report output in the Runtime Previewer.

Preview Options (Cont..)



- 1. LAYOUT MODEL
- 2. DATA MODEL
- 3. SAVE
- 4. ALIGNMENT

Summary

- In this lesson, you have learnt:
 - Types of Reports
 - Running of Report
 - Preview Options in Report Builder



Review Questions

- Question 1: Group Below is a type of Report.
 - True / False

- Question 2: A simple Master -Detail report contains which two groups of data.
 - A) Master Group
 - B) Child Group
 - C) Super Group
 - D) Detail Group



ERP- Oracle Apps Forms and Reports

Lesson 3: Designing Report

Lesson Objectives

- To understand the following topics:
 - Basic Design Steps
 - Components of Reports
 - Data Model
 - Layout Model



3.1: Introduction Basic Design Steps

- To describe the common styles required n a report
- To describe the structure of each report
- To identify the various report destinations
- To view the report in the REPORT PREVIEWER
- Acquaintance with the report concepts
 - e.g . Layout Model,DataModel etc.



Copyright © Capgemini 2010. All Rights Reserved 3 |

Basic Design Steps

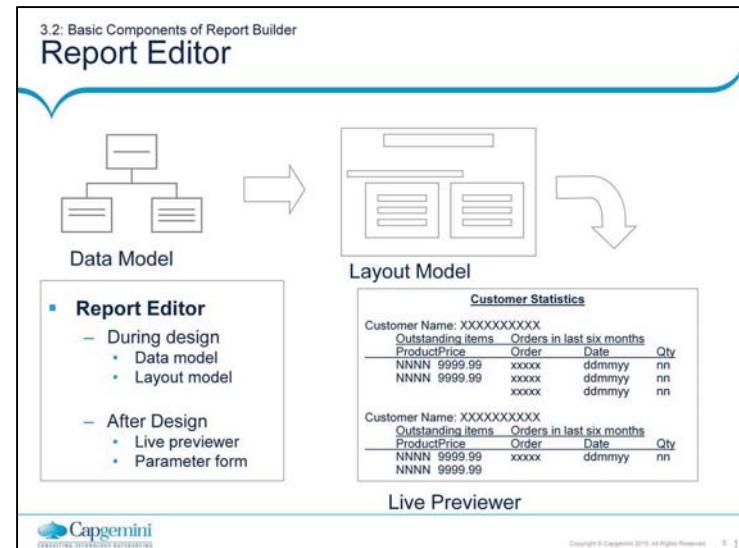
Before you start development, consider:

```
graph TD; Specification[Specification] --> Structure[Structure]; Structure --> Style[Style]; subgraph Cloud [ ]; direction TB; DR((Data retrieval)); CF((Common features)); end; Specification -.-> Cloud; Structure -.-> Cloud;
```

- Common Report Styles:
 - Tabular
 - Master-detail
 - Master and multiple details
 - Matrix

 Capgemini
CONSULTING INTEGRATED SERVICES

Copyright © Capgemini 2010. All Rights Reserved. 4



Report Editor

The Layout window also called as report editor, in which you create the Report Layout objects.

Report Level Settings

- Report
- Data Model
- Layout Model
- Parameter Form

- Properties: Define page dimensions and Previewer settings
- Triggers: Allow PL/SQL to be executed at different stages of the report execution
- PL/SQL program units: Contain functions, procedures that can be called from report-level objects in the same report

 Capgemini
ENTERPRISE INTEGRATION EXPERTS

Copyright © Capgemini 2010. All Rights Reserved.

Reports

The ‘Reports’ component of Developer 2000 Ver 2.0 is the part of the development environment in which you develop reports modules. Reports Builder is a tool, which is used for developing, displaying and printing quality report.

Triggers

Triggers are PL/SQL program units that “fire” at specific times during report execution.

Data Model

The Data Model is a central component of the report design. It defines the data to be included in the report and provides a way to organize and manipulate that data. The Data Model is composed of several objects:

- Report:** The main document that contains the data and presentation logic.
- Data Model:** The core object that defines the data structure and rules for the report.
- Layout Model:** The object that defines the visual layout and presentation of the data.
- Parameter Form:** The object that defines the input parameters for the report.

Query:

- Select the data for your report.

Group:

- Organize the data to form the required hierarchical structure.

Columns:

- Contain individual data values; database columns exist by default and contain the database columns or expressions defined in the query; you can also create formula, summary and placeholder column types.

Copyright © Capgemini 2016. All Rights Reserved 7

Data model

The data model is composed of objects that define the data to be included in a report. Data model objects include each of them is explained below:

- Queries
- Columns
- Groups
- Links
- Parameters

Data Model (Cont...)

Report

Data
Model

Layout
Model

Parameter
Form

- Data Link:

- Join queries for complex data relationships

- Parameter

- Provide for runtime defaults or user input; system parameters exist by default; you can also create user parameters



Copyright © Capgemini 2010. All Rights Reserved.

Layout Model

Report Data Model Layout Model Parameter Form

- Frames:
 - Contains other objects and prints only once
- Repeating frames:
 - Contains other objects and prints once for each record of the associated group
- Fields:
 - Contains data and other variable values and their formats

 Capgemini
CONSULTING INTEGRATED SOLUTIONS

Copyright © Capgemini 2010. All Rights Reserved. 9

Layout Model

The Layout Model view is a work area in which you can define the format of your report output. You can work with layout objects, such as frames, repeating frames, fields, boilerplate, anchors, and Graphics objects. When you run a report, Report Builder uses the Layout Model as a default template for the report output. This template will grow or shrink based on the data fetched for and the PL/SQL logic assigned to the layout objects. The layout model is composed of objects that define the positioning and appearance of data and other objects in a report.

Frames

A frame is a container for grouping related layout objects in the report. Frames may contain other frames, repeating frames, boilerplate, and fields. In the layout editor frames appear as rectangular lines with diamonds symbols on the edges to indicate that the frame can expand. A frame contains data and other report objects also. It prints only as often as the object by which it is enclosed or to which it is attached. A frame is not record-related.

There are two types of frames

1. Default Frame
2. User-defined Frame

Default frames

When you accept the 'default layout' dialog box, Oracle Reports automatically generates to surround objects like boilerplate and summary fields, that need to remain with the same object in the layout.

User-defined Frames

User can create a frame in the layout editor by clicking on the frame tool, dragging a region, then specifying its properties in its property sheet.

Repeating frame

Like a frame, a repeating frame is also a container for grouping related layout objects in the report, and like a frame, repeating frames may contain other frames, repeating frames, boilerplate, and fields. A repeating frames may contains other frames, repeating once for each record of a group, the data model object with which it is associated. Repeating frames contain the data owned by their corresponding groups. They are called as repeating frame because they repeat as many times necessary to display all values of the data they contain. In the Layout editor, a repeating frame appears as a box surrounding one or more fields. The arrow on its border indicates the direction in which the repeating frame repeats.

Fields

Fields are the placeholders for parameters, columns and such values as the page the page number, current date etc. They define the formatting attributes for all columns displayed in the report. A field is one of the objects that can be located inside a frame or repeating frame. If the column in the report does not have an associated field, then its values will not appear in the report layout editor.

There are two types of fields

1. Default fields
2. User-defined Fields

Default Fields

Whenever user accepts default layout dialog box, Oracle Reports automatically creates one field for each column, and places each field inside of a repeating frame.

User-defined Fields

User can create a field in the layout editor by selecting the field tool and then you can set the property of that field for e.g. to display page numbers, to display current date etc.

Layout Model (Cont...)

The diagram illustrates the four components of a Layout Model:

- Report
- Data Model
- Layout Model** (highlighted with a blue background)
- Parameter Form

Below the components, there are two sections with bullet points:

- Boilerplate:
 - Contains fixed text or graphics that may appear anywhere in the report
- OLE2:
 - Allows you to create an OLE2 container and define the contents using OLE2 tool

Capgemini
CONSULTING INTEGRATION CONSULTING

Copyright © Capgemini 2010. All Rights Reserved 31

Boilerplate

Boilerplate consists of text and graphics that appear in the report each time it is run.

For e.g. a label appearing above a column of data is boilerplate text. They are used to customize your report to creating heading pages, adding graphics etc. in the above diagram the parts of the layout is shown. Report Builder will create one boilerplate object for each label selected in the Report Wizard (it is named B_columnname). Also, one boilerplate object is sometime created for each report summary. There are two types of Boilerplate object: Default and User-Created. A user can create a boilerplate object in the layout editor by clicking any one of the drawing tools or by selecting a 'T' tool.

OLE2

Report Builder is an OLE2 container. You can include OLE2 objects in your reports through the data model or the layout model. An OLE2 report can display objects such as spreadsheets, graphics, and sound from OLE2-aware applications. Reports that contain OLE2 objects supported only on the Windows platform.

Parameter Form

Report

Data Model

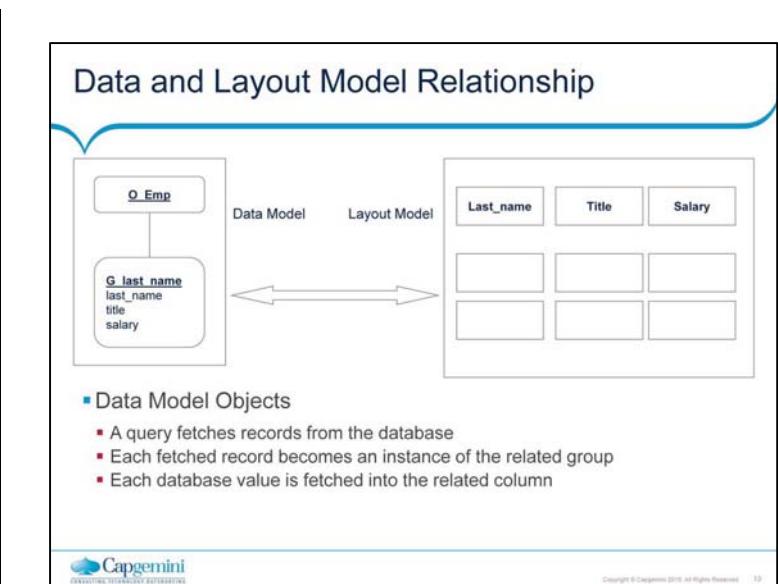
Layout Model

Parameter Form

- Fields:
 - Contains parameter values
- Boilerplate:
 - Contains constant text or graphics that may appear on the runtime parameter form

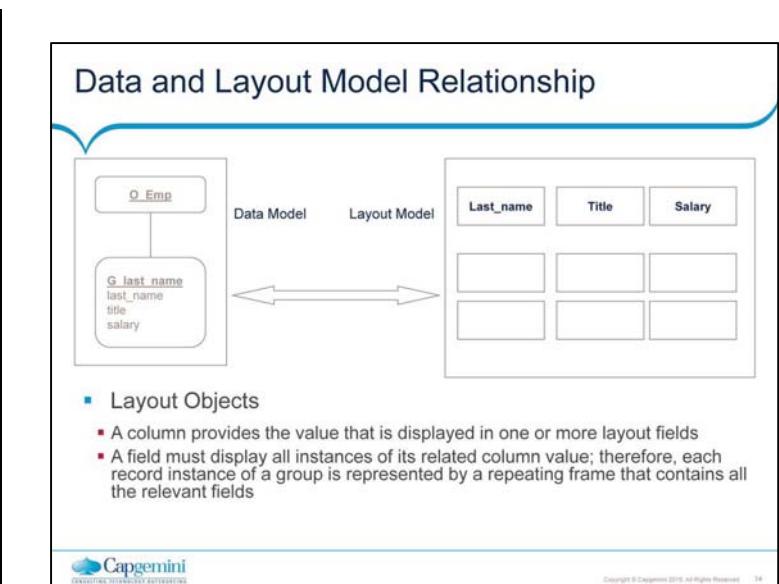


Copyright © Capgemini 2016. All Rights Reserved 12



Data model objects

Define the data to be included in a report, and include: queries, columns, groups, parameters, and data links.



Layout model objects

Define the positioning and appearance of data and other objects in a report, and include: frames, repeating frames, fields, boilerplate objects, anchors, buttons, charts/graphical displays, and OLE2 objects.

3.3: Data Model Steps to Setup Data Model

- Describe data model objects
- Create groups to modify the hierarchy
- Change the order of data in a group
- Use a group filter to eliminate records
- Create supplemental rows of data, using additional queries
- Create a data link to link data from different queries
- Describe various data model columns
- Display the contents of a file
- Recognize characteristics of user defined columns
- Create and populate a placeholder



Copyright © Capgemini 2010. All Rights Reserved 10 |

Objects

- Query
- Group
- Link
- Column
- Parameter



Copyright © Capgemini 2010. All Rights Reserved 10

Queries

Oracle Reports applications are based on SQL queries that return data from the database to your report. Queries are SQL SELECT statements that define which rows and columns from specified tables or views are to be fetched from the database. The data fetched by a query can be used in calculations as well as displayed in the report output. All queries are based on SQL select statement that are either external or local to the report. Once a query is accepted, a group containing its columns is created in the data model editor.

Groups

Groups determine the hierarchy of the data appearing in a report, and are used primarily to create breaks.

There are main two types of groups in Reports builder:

1. Default Groups
2. User-Created Groups

Groups can do two things:

- It separates a query's data into sets.
- Filter a Query's data

Default Groups

By default Oracle Reports generates a group for each query and assigns all of the query's columns to it.

User-Created Groups

You can drag a column outside of an existing query group, which becomes the userdefined group. For e.g. if you want to create a break group. You can create a new group in the data model and include column into it for which you want to create a break column. You can also create groups to define the Cross Product of Matrix Report. In the following chapters we will be creating the matrix reports using user-defined groups. In Developer ver 2 there are two group report styles are added.

Columns

Columns represent the data of your report. Whenever you create a query in the data model window, automatically Oracle Reports creates a group for all the columns specified in the query.

In Oracle Reports there are main two types of columns:

1. Default
2. User-Defined

Default Columns

By default, Oracle Reports generates one column for each item in a query's SELECT list, and assigns each column the datatype and width specified in the database data dictionary. These columns are placed in the default group. In addition to the traditional column types (date, number, char, varchar2) oracle reports also supports graphic columns, whose values can be graphics, and they are stored in the database such column's datatype is usually long or long Raw.

User-Defined columns

You can create your own columns to perform computations. There are three types of user-defined columns

1. Summary Column
2. Formula Column
3. Placeholder Column

Summary Columns

Summaries column computes their own values using function with Oracle Reports. A summary is a computation that is performed on all records of one column. (For e.g. the sum of all salaries) Summaries operates on one value over multiple records e.g. sum of sal).

Formula Columns

A formula is a computation performed on one or more columns of one record. A formula computes their own values using PL/SQL expressions. Occasionally, you may find that report requires a different computations. For e.g. (:ITEMTOT * .07) is a formula that performs a computation on one column, while (:SAL + :COMM) performs a computations using two column in a record.

Placeholder Columns

A placeholder is a column whose value and datatype are set via PL/SQL. Placeholder columns are useful when you want to selectively set the value of a column (e.g. each time the nth record is fetched, or each time a record containing a specific value is fetched etc.)

Link

Data links are used to establish Parent-Child relationship between queries and groups via column matching.

Parameter

Parameters are variables for your report that enable you to change selection criteria at runtime. Oracle Reports creates automatically a set of system parameters at runtime, but you can create your own as well. Parameters are especially useful for modifying SQL SELECT statements and setting PL/SQL variables at runtime.

There are two types of parameters

1. System parameters
2. User-defined parameters

System parameters

System parameters are parameters that Oracle Reports creates for every report.

User-defined parameters

User-defined parameters are parameters that you create for the report. Both user and system parameters can be assigned values at runtime (e.g. from the Runtime Parameter Form). You can delete or rename a user parameter but you cannot delete or rename a system Parameter.

3.3.1: Queries Modifying Properties

- Modify SQL Query Statement
 - Add, rename, or delete columns
 - Use column and table aliases.
 - Remove or modify schema name.

```
SELECT      d.id, d.name, e.title  
            e.first_name||'&'||e.last_name  
            employee_name,  
            e.salary*12 annual_salary  
FROM        s_emp e, s_dept d WHERE e.dept_id =d.id
```

- Syntax error checks occur when:
 - Existing SQL query statement
 - Compiling or executing a report



Updating Layout

- Update layout to reflect changes in Data Model
 - Choose Report Wizard.
 - After tab pages.
- Wizard destroys previous layout and creates new objects



Copyright © Capgemini 2010. All Rights Reserved 20 1

3.3.2: Groups

Introduction

- Groups determine hierarchy and frequency
 - Wizard creates
 - Default naming conventions
 - You can change query name
 - You can change group name
 - Developer-created groups for
 - Control break reports
 - Complex matrix reports

Copyright © Capgemini 2010. All Rights Reserved 21

Groups

Groups are created to organize the columns in your report. When you create a query, Report Builder automatically creates a group that contains the columns selected by the query. You create additional groups to produce break levels in the report, either manually or by using the Report Wizard to create a group above or group left report.

Creating a Group

To create a break group using the Data Wizard:

- In the Data Model view, click the query object, then choose ToolsData Wizard.
- On the Groups page, select one or more columns for break groups.

To create a break group manually:

- In the Data Model view, drag the title bar of the group object down.
- Drag the break column (e.g., DEPTNO) above the group to create a new group.

(Optional) Double-click the new group object to display the Property Palette and set the desired properties.

Packed Filter

- Reports provides two packaged filters:
 - First
 - Retrieves the first <n> records for the group
 - Second
 - Retrieves the last <n> records for the group



Copyright © Capgemini 2010. All Rights Reserved. 22

External Query

- Import an external query from a file
 - Import SQL Query button
- Create an external query file
 - Notepad
 - SQL*Plus
 - Report Builder
- Link to an external query source



Copyright © Capgemini 2010. All Rights Reserved. 23

Import SQL query

If you have stored a query in a .SQL file into some directory then you can import SQLfile by specifying the proper path through the 'BROWSE' button. Click on the Next button, the movement you click on the NEXT button, Reports will ask you for connection because you are extracting the data from the database.

3.3.3: Links

Data Link

- Compound join
 - Multi-column primary or foreign key
 - Define multiple equi-joins
- Nonequi-join
 - Primary key value between two other values
 - Create two links:
 - S_emp.salary >= salgrade.losal
 - S_emp.salary <= salgrade.hisal

 Capgemini
CONSULTING INTEGRATION CONSULTING

Copyright © Capgemini 2010. All Rights Reserved. 24

Data Link

Data links relate the results of multiple queries. A data link (or parent-child relationship) causes the child query to be executed once for each instance of its parent group. When you create a data link in the Data Model view of your report, Report Builder constructs a clause (as specified in the link's Property Palette) that will be added to the child query's SELECT statement at runtime.

Creating a Data link

In the Data Model view, single-click in the tool palette.

- Create a Query to Query Link To create a data link for each database constraint defined between the tables for two queries, click the parent query, then drag to the child query.
- Create a Group to Group Link To create a link between one query's group and another query's group, which is useful when you want the child query to know about the parent's data, click the parent group (avoiding the columns in the group) and drag a link to the child group.

Create a Column to Column Link To create a link between columns, click a column of the parent query and drag a link to a column of the child query. Double-click the new link object. In the Property Palette, set the desired properties.

Data Link

- Two other methods of forming a link:
 - Manual: using a WHERE clause to form the link
 - Automatic: using primary/foreign key constraints to form the link



Copyright © Capgemini 2010. All Rights Reserved 25 |

Data Link

A data link object is data model object which enable users to relate multiple queries. For a simple Master/Detail report the user need to relate two queries using the primary and foreign keys of the tables from which you are selecting data. The query with the primary key column i.e. the source for the master records is called Parent Query and the query with the foreign key column i.e. the source for the detail records is called the Child Query.

A primary key is a column for which each value uniquely identifies the record in which it is found. A foreign key is a column, which contains the same values as the primary key for another table, and is used to reference records in that table. Linking two tables via a primary and foreign key is similar to specifying a join condition.

In Report builder a join is defined by using the information in the data link object. The where clause for the join is added to the child query's SELECT statement at runtime.

The join defined by a data link is an outer join i.e. in addition to returning all rows that satisfy the link's condition, an outer join returns all rows for the parent query that do not match a row from the child query.

Demo

- Create a Group above Report using Report wizard



Copyright © Capgemini 2010. All Rights Reserved 20

Add the notes here.

3.3.4: Columns

Value and Frequency

- What type of value ?
 - Choose the correct column tool
- What frequency ?
 - Create in a group or a report level

 Capgemini
CONSULTING INTEGRATION SERVICES

Copyright © Capgemini 2010. All Rights Reserved. 27

Summary Column

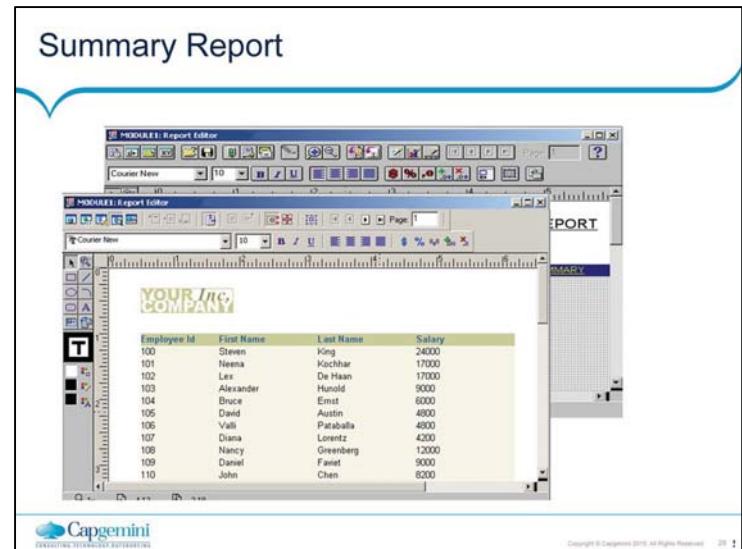
- Specific properties:
 - Function
 - Source
 - Reset At
 - Compute At
- Datatype depends on Source datatype
- Page summaries: NOT supported in the wizard



Copyright © Capgemini 2010. All Rights Reserved 29

Summary Columns

Summaries column computes their own values using function with Oracle Reports. A summary is a computation that is performed on all records of one column. (For e.g. thesum of all salaries) Summaries operates on one value over multiple records e.g. sum of sal).



Resetting Summary Values

Data Model Group	Reset AT		
	Report	G_DEPT	G_EMP
REPORT	Grand Total	XXXX	XXXX
G_DEPT	Running Total	Sub Total	XXXX
G_EMP	Running Total	Running Total	Record Total



Copyright © Capgemini 2010. All Rights Reserved 301

Formula Column

- Performs a user-defined computation
- Executes a PL/SQL function
- Must return a value
- Can be a Character, Number, Date
- Returned value must match datatype

```
function CF_SALCALCFormula return Number is
begin
return(my_function :salary)
end;
```



Copyright © Capgemini 2016. All Rights Reserved 31 |

Formula Columns

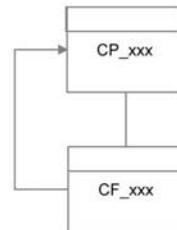
A formula is a computation performed on one or more columns of one record. A formula computes their own values using PL/SQL expressions. Occasionally, you may find that report requires a different computations for e.g. (:ITEMTOT *.07) is a formula that performs a computation on one column, while (:SAL + :COMM) performs a computations using two column in a record.

Formula Column

	Sbk Price	Discount	New Price
	100	5	105
	100	5	105
	100	5	105
	1000	50	1050
	400	20	420

Placeholder Column

- An empty container at design time
- Populated by another object at run time
 - Before report trigger
 - Formula column at report level
 - Formula column in same group or below placeholder



Copyright © Capgemini 2010. All Rights Reserved 33 |

Placeholder Columns

A placeholder is a column whose value and datatype are set via PL/SQL. Placeholder columns are useful when you want to selectively set the value of a column (e.g. each time the nth record is fetched, or each time a record containing a specific value is fetched etc.)

NOTE:

You cannot populate a placeholder by writing code in the placeholder's own Formula property.

The PL/SQL Formula button opens the program unit editor where you enter and edit your code. This is only applicable when you call a user exit.

Demo

- Edit same Report for various columns



Copyright © Capgemini 2010. All Rights Reserved 34

Add the notes here.

3.3.5: Parameters

Definitions

- To create and use the user parameters
- Lexical vs Bind references
- List of Values
- To use the system parameters
- Build a parameter form

 Capgemini
CONSULTING INTEGRATION CONSULTING

Copyright © Capgemini 2010. All Rights Reserved. 35

User parameters

The developer can create 'user-defined parameters'. They allow user to modify the report at runtime. The user parameters can also contain initial values as defined during development. The values of any of these parameters may be changed in the Report Builder via their property palettes or at runtime by placing on the Parameter Form Window. In addition 'user-defined parameters' may be based on a list of Static values or on a (SQL) 'SELECT' statement.

Bind Parameters

Use a bind reference when you want the parameter to substitute only one value at runtime. Bind parameters are referenced with a colon (:). With bind parameters forward referencing is permissible.

For e.g.

```
SELECT * FROM ORD
WHERE ORDNO = :ORDPARAM
```

Because :ORDPARAM can only be replaced by one expression (e.g., 20000), you precede the parameter reference with a colon. Queries with bind references are parsed or checked for errors once. The bind references themselves are re-evaluated for each execution of the query. With bind parameters one value, one word, is substituted in the parameter reference. These parameters can accept the number, character and date data types.

lexical parameters

When you want the parameter to substitute multiple values at runtime, then you can use the lexical parameters. Using lexical parameters you can substitute a string of characters. It is prefixed by an '&'. Lexical Parameter values can be only of type Character, and forward referencing is not allowed in this type of parameter. You can reference lexical parameters as elements of expressions in the following clauses of your SELECT statement: 'where', 'group by', 'having', 'order by', 'connect by', 'start with' etc. To reference a lexical parameter in a query, you must first create the parameter and assign default values to it in the 'Parameter Form Window'.

For example:

```
SELECT * FROM ORD  
WHERE_clause;
```

User Parameter

- Can be used to :
 - Restrict values in a WHERE clause

```
SELECT NAME, SALES_REP_ID  
FROM S_CUSTOMER  
WHERE ID = <a value>
```

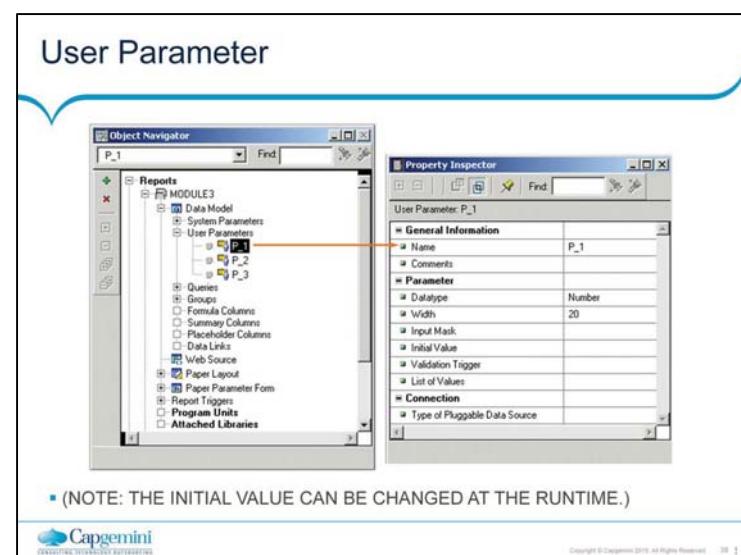
- Substitute any part of select statement

```
SELECT NAME, SALES_REP_ID  
FROM S_CUSTOMER  
<a where clause>
```

- Substitute a single column or expression

```
SELECT <a column/expression>  
FROM S_CUSTOMER
```





User Parameter

Bind reference replaces a value

- :parameter_name
 - parameter object is created by default

Lexical reference replaces a clause

- ¶meter_name
 - parameter object is never created by default



Copyright © Capgemini 2010. All Rights Reserved 30

Lexical Parameters and References

Lexical Parameters or Lexical References are used to dynamically alter the query used in the report based on the parameters chosen by the user when the report is run.

The uses of Lexical Parameters are as follows:

- This can be used to improve the performance of the report by not cluttering the query with AND clauses for parameters whose value is NULL.
- Lexical parameters can also be used to handle situations where a particular parameter has a set of defined values, and for each value, a totally different set of where conditions need to be executed.
- To enable passing of column names as a parameter to the report. (This rarely happens in Oracle Applications Reports).

Consider a scenario where a particular parameter, say :p_parameter is optional. Instead of writing:

AND column_name = NVL(:p_parameter, column_name)

Like we usually do its more efficient to have code in the Before Report or After Parameter trigger that checks if *p_parameter* is null and then removes that entire AND clause from the query instead of having the NVL.

Lexical parameters are implemented in the following manner:

Define a character variable of sufficiently large size (say 1000 characters) in the User Parameter section. Write code in the After Parameter/Before Report that generates the code to be inserted in the main query as a string. The lexical parameter is equated to this string containing dynamically generated code. The main query is updated to reference this lexical parameter using the ampersand sign followed by the parameter name, without a colon.

Eg:

```
SELECT * from table_a  
WHERE column_a = 5  
AND &I_lexical_parameter
```

Note:

The Query Builder in the Data Model will refuse to compile a query that includes lexical parameters if these lexical parameters do not have an initial value defined.

Consider this query:

```
Select column1, column2, &I_column from table1  
You plan on setting the lexical parameter, I_column, to some column  
name dynamically when the report is run.  
If you provide some default, valid column name as the initial value of this  
lexical parameter in the Property Palette of this parameter, it will work  
fine. <inside your after parameter trigger>
```

```
--  
L_column := 'column3';  
--
```

Procedures and Functions

You can usually avoid creating separate procedures or functions by using formula columns in your report to perform complex calculations. There are some situations, however, when certain functionality is not possible using a formula column. For example, format triggers on layout items cannot pass values directly to the rest of the report or to each other. If a value needs to be passed from one format trigger to another, you can create a separate procedure in Program Units that accepts a value and then sets a bind variable to this value. You can call this procedure from the format trigger. Your second format trigger will be able to reference the bind variable, effectively passing this value from one format trigger to the other

User Parameter

- Restrict values in a WHERE clause

```
SELECT NAME, SALES REP_ID  
FROM S.CUSTOMER  
WHERE ID > :P_CUST
```

- Substitute a single value or expression in the select statement

```
SELECT NAME, SALES REP_ID  
FROM S.CUSTOMER  
ORDER BY DECODE(:SORT, 1, NAME, 2,  
STATE,COUNTRY)
```



User Parameter

- Use to substitute any part of the query

```
SELECT NAME, SALES REP_ID  
FROM S_CUSTOMER  
&P WHERE_CLAUSE  
&P_ORD_CLAUSE  
  
SELECT NAME, SALES REP_ID  
FROM S_CUSTOMER  
&P WHERE_ORD_CLAUSE  
  
SELECT &P_CUST CUST, &P_SALESREP REP  
FROM &P_TABLE
```

- Ensure number of values and datatypes match at runtime



User Parameter Tips

- Always do the following
 - Specify column aliases when substituting column names
 - Create lexical parameters explicitly in the Object Navigator (Report Builder creates bind parameters if necessary)
 - Enter an initial value for parameters that affect query validation when NULL



Copyright © Capgemini 2010. All Rights Reserved 43

System Parameter

- BACKGROUND: Report running mode
- COPIES: Number of copies
- CURRENCY: Symbol for currency
- DECIMAL: Symbol for decimal
- DESFORMAT: Output device definition, ignored if it is screen/preview. (Example: dflt, pslan80, pdf)
- DESNAME: Destination name, file name, printer name, mail userid, ignored if it is screen/ preview
- DESTYPE: Destination type (file/ printer/ mail/ screen/ preview)
- MODE: Bitmapped or character mode
- ORIENTATION: Print direction of printer output
- PRINTJOB: Whether printjob dialog appears at runtime
- THOUSANDS: Symbol for thousands

Copyright © Capgemini 2010. All Rights Reserved A4

Steps to Create LOV

■ Steps to create static LOVs

- In the parameter property palette, choose the LOV property. The static values radio button is selected by default
- Enter a value in the value field and choose ADD
- Repeat for each value you want in the list
- Note: To remove a value, select the value in the list and choose REMOVE

■ Steps to create dynamic LOVs

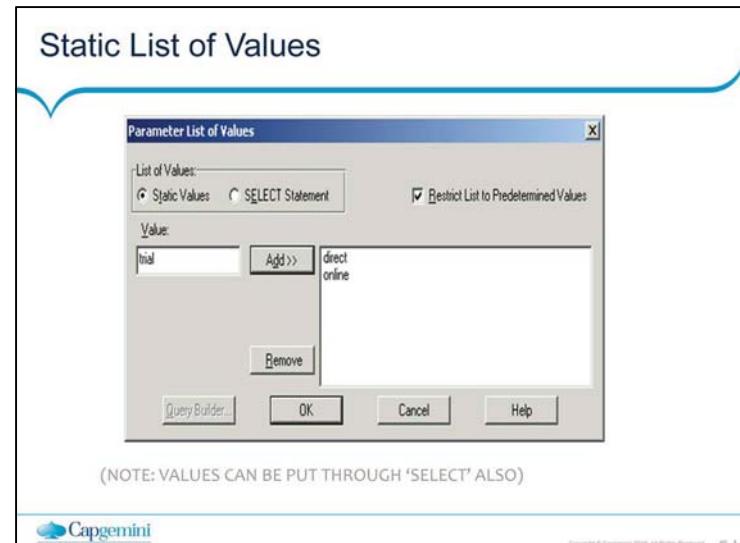
- In the parameter property palette, choose the LOV property
- Choose SELECT statement. The SQL query statement area displays
- Enter query to populate the list of values. You can include more than one column; the parameter takes its value from the first column in the list
- Set the restrict list to predetermined values property, as required



Copyright © Capgemini 2010. All Rights Reserved. 40

LOVs (List of Values)

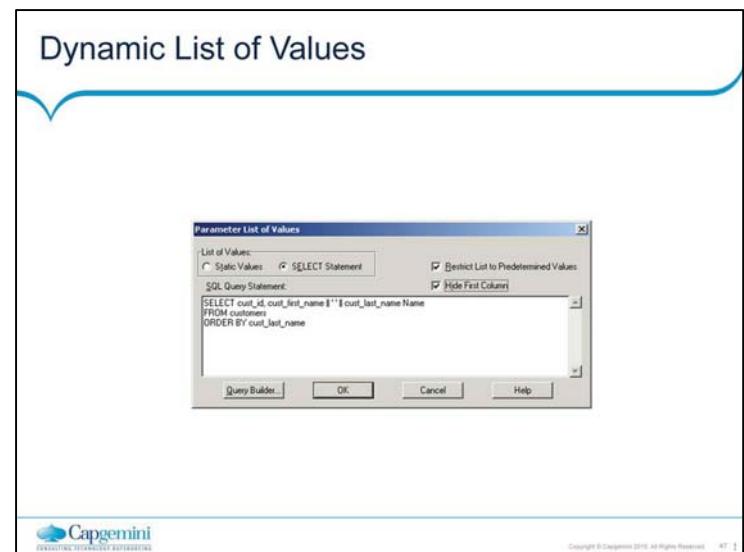
LOVs also called as List of Values provide the user with list of valid entries for a field. A list of Values presents data contained within an object called a record group whereby the user will select one value from the list to populate a form item. The list of values may also be used to validate the user input to ensure that a valid value is entered. In the 'List of Values' property you can enter static values or values retrieved with the help of Select statement.

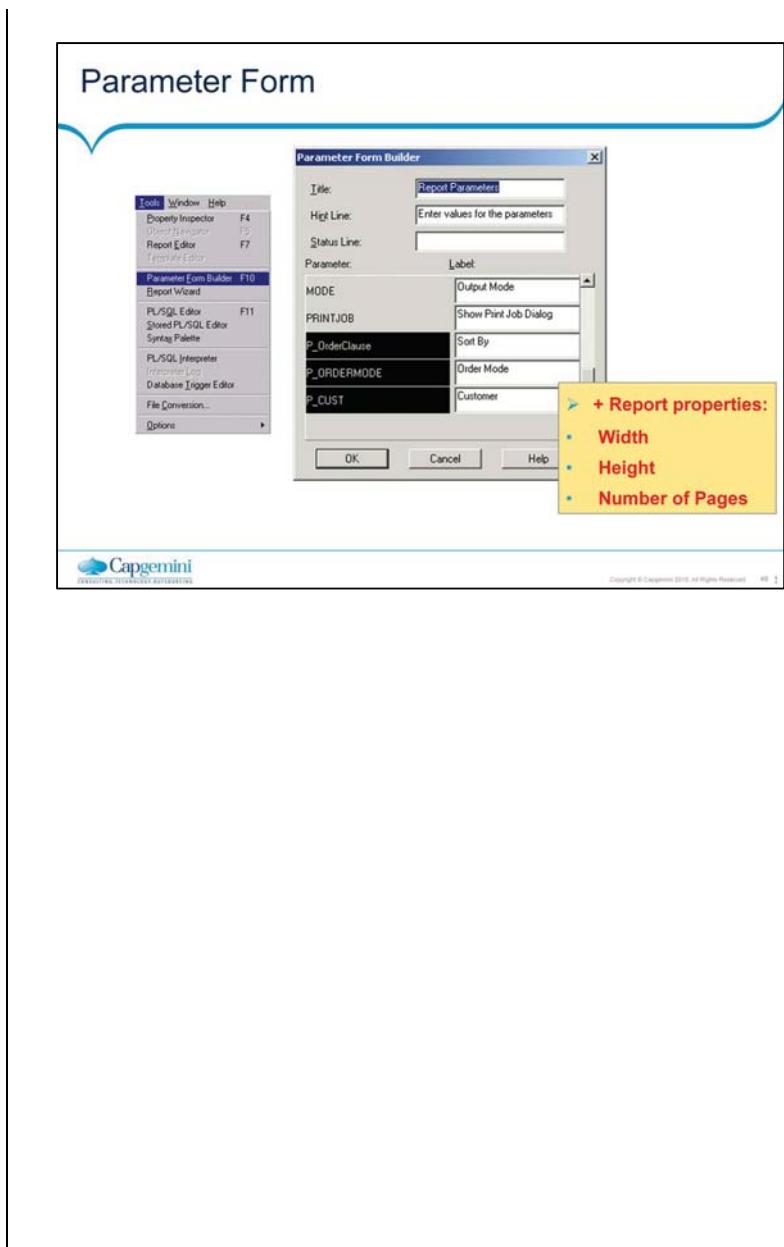


Copyright © Capgemini 2010. All Rights Reserved 40

Static LOVs

- In the List of Values property you can also provide the 'static values'.
- Click on the radio button 'Static values' and enter the value and then click on the 'Add>>' button.
- The 'Add>>' button will keep on adding the values.
- You can also remove the values from the box by clicking on the 'Remove' button.





Demo

- Create a Report using Report wizard with parameter form



Add the notes here.

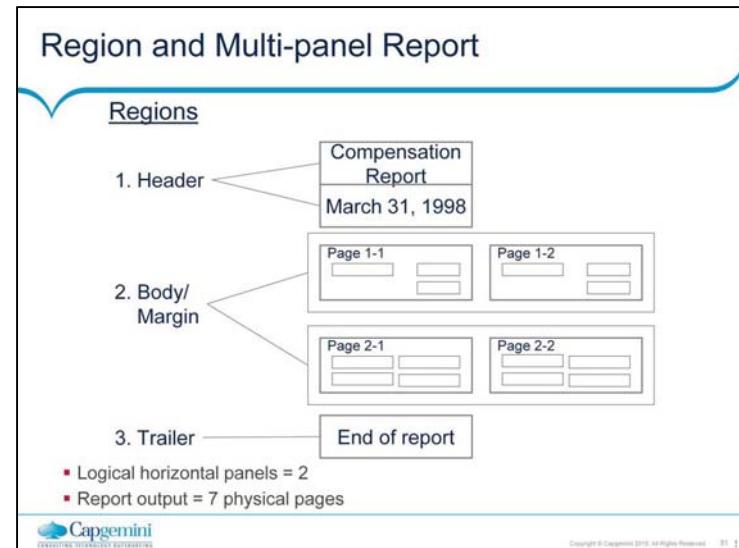
3.4: Layout Model

Basic Design Steps of Layout Model

- Identify the different regions of the layout model
- Toggle between modes in the layout model
- Understand the layout objects
- Create buttons in the layout model

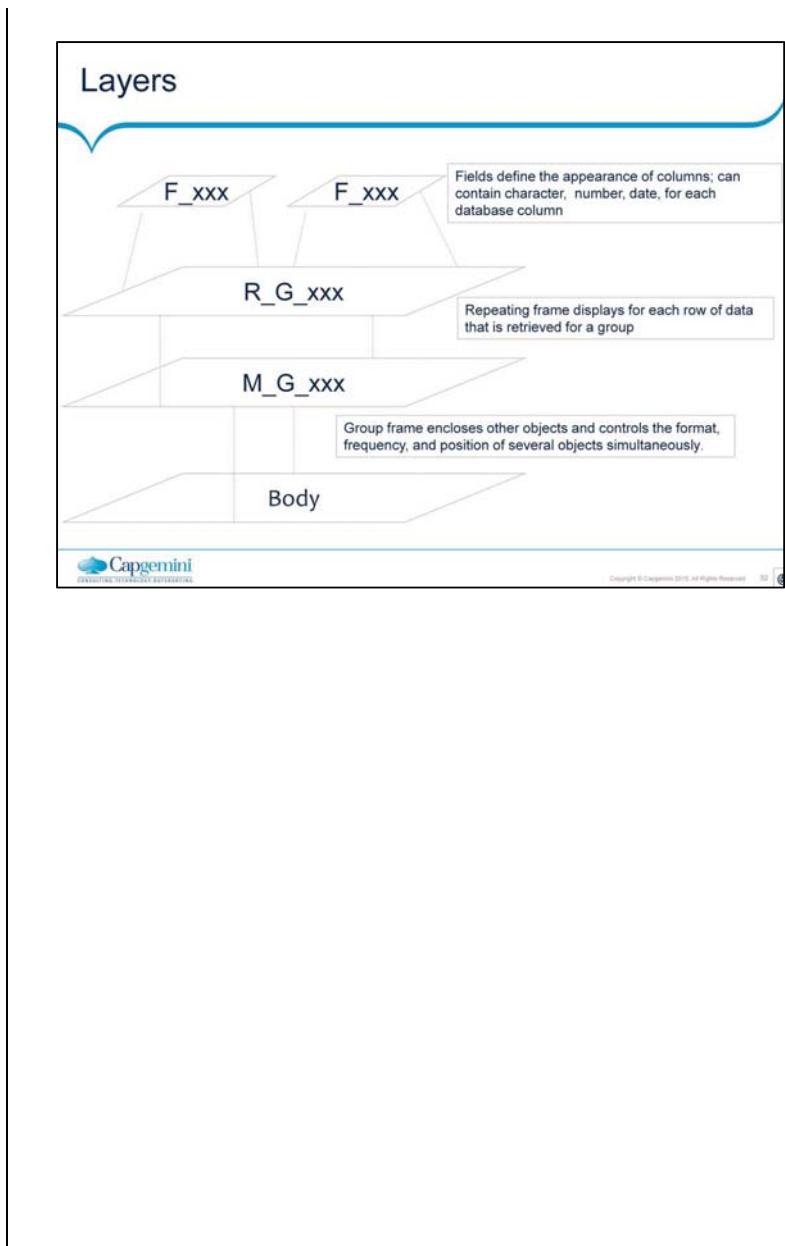


Copyright © Capgemini 2016. All Rights Reserved. 30



Margin

The black thick line that separates from the report body defined the margin area. In this area you can place the images, boilerplate text, fields and other objects.



Avoiding Layout Errors

- To avoid common hierarchy errors in your report layout always work using following modes
 - Confine Mode
 - Enabled (the closed padlock):
 - To avoid accidentally moving one object outside or below its current enclosing object
 - Disabled (open padlock):
 - To deliberately want to move one object outside its enclosing object
 - Flex Mode
 - Enabled:
 - To adjust all affected objects when you move or resize one object
 - Disabled:
 - To move an individual object without moving other objects



Copyright © Capgemini 2010. All Rights Reserved 53

When moving or resizing an object, you often want to ensure that the object maintains the same relationship to its parent object. For example, if you resize a field such that it no longer fits in its parent-repeating frame, you will get a frequency error when you run the report.

To ensure that objects maintain the correct relationship to their parent objects, Oracle Report's Layout editor provides the following modes:

1. Confine mode

2. Flex mode

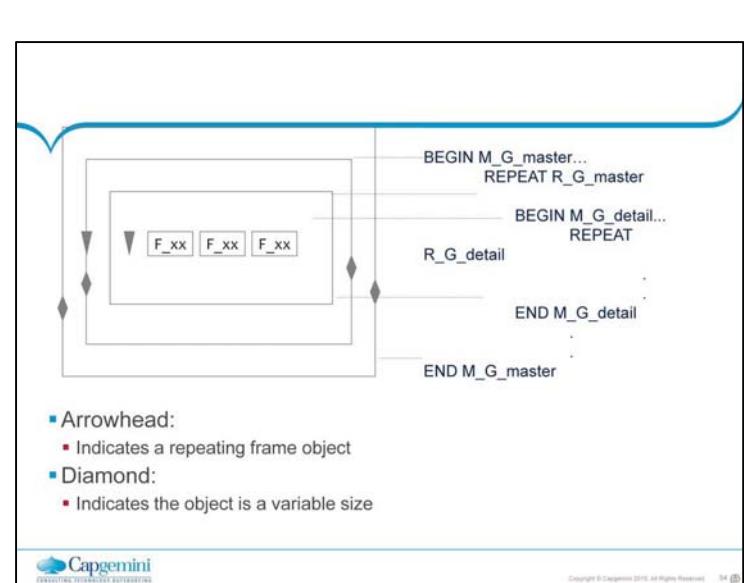
Confine mode

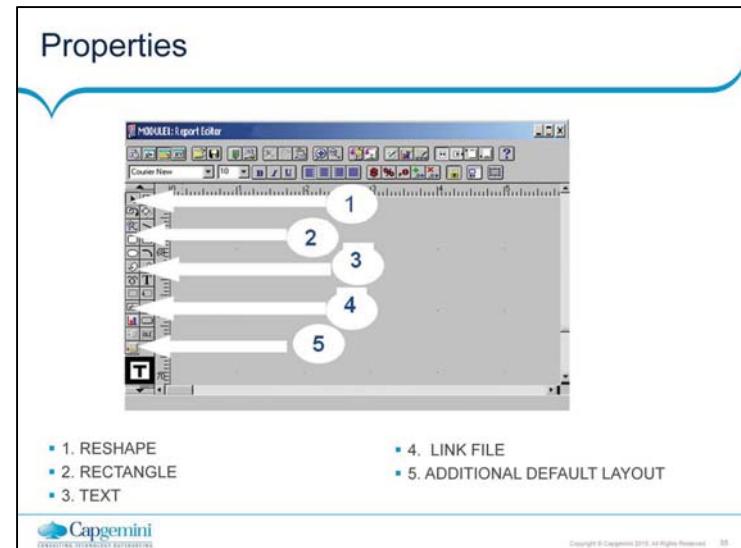
Confine mode prevents you from performing an operation that would cause the layout not to work. For example, in confine mode, Oracle Reports will not allow you to move a field outside of its parent-repeating frame. Confine mode is on by default in the Layout editor. To turn confine mode on or off, use the Confine tool in the toolbar.

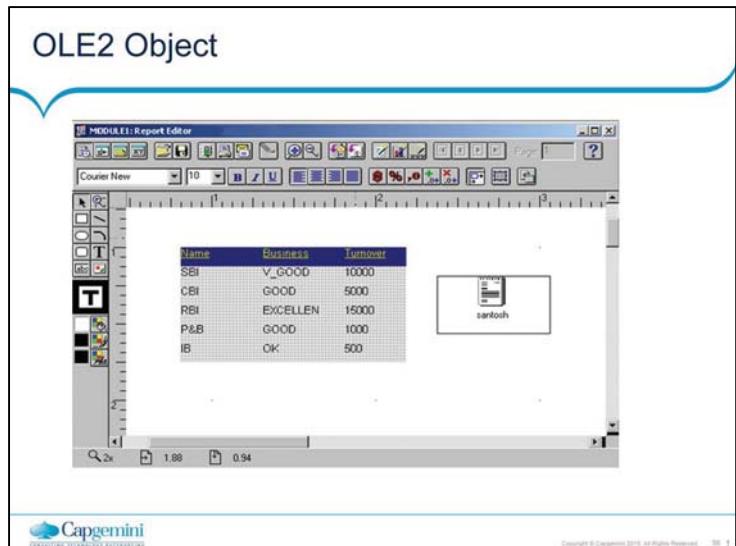
Flex mode

Flex mode moves or resizes the object, its enclosing objects, and objects in their push path simultaneously to maintain the same overall relationships in the layout.

For example, if you try to move a field outside of its parent repeating frame in flex mode, its repeating frame will grow as necessary to contain it and any objects around the repeating frame will also grow or be pushed as necessary. You can only move or resize objects in one direction at a time (vertically or horizontally, not diagonally). To use flex mode, select the Flex tool in the toolbar or Flex in the Layout Options dialog box, or hold down the Control key while moving or resizing the object.



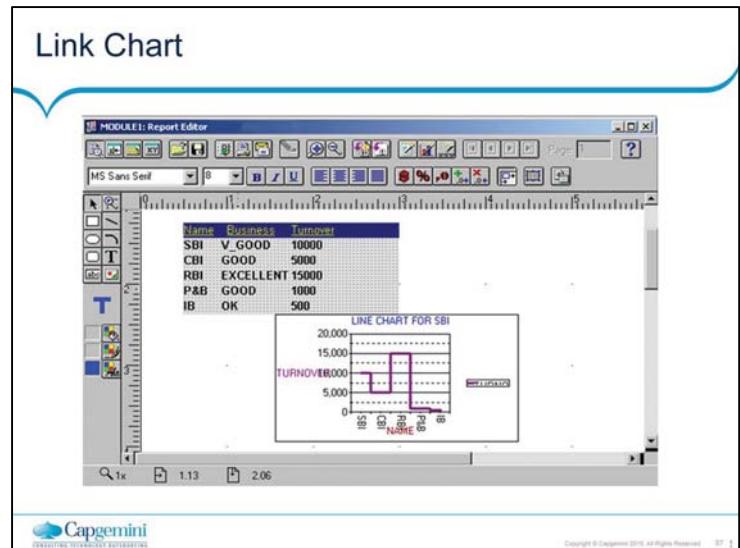




OLE2

(Object Linking and Embedding, V2) objects display objects, such as spreadsheets or drawings, from OLE2-aware applications. OLE2 objects can be static or dynamic (displayed based on report values).

OLE allows an editor to "farm out" part of a document to another editor and then re-import it. For example, a [desktop publishing](#) system might send some text to a [word processor](#) or a picture to a [bitmap editor](#) using OLE. The main benefit of using OLE is to display visualizations of data from other programs that the host program is normally unable to generate itself (e.g. a pie-chart in a text document), as well as to create a master file. References to data in this file can be made and the master file can then have changed data which will then take effect in the referenced document. This is called "linking" (instead of "embedding").



Creating Layout

- Creating Additional Layout

- Example:

- Create a letter to be sent to all customers and add a tabular layout to list all customers at the end of the report.
 - Both layouts use the same data
 - There is no need for two queries
- STEPS:
 - Select Additional Default Layout tool from the tool palette
 - Drag an area in the Layout Model editor defining where to place the new objects. (now report wizard opens)
 - Choose the group tab; choose the group you want and direction to print the records
 - Complete the wizard tabs as before



Copyright © Capgemini 2010. All Rights Reserved 38

Flex Lines

- Adjust to fit variable and repeating frames
- Make use of Live Previewer or Layout Editor for creating and modifying the flex line.



Copyright © Capgemini 2010. All Rights Reserved 30 |

Button

■ Buttons

- Provide runtime user interaction for Previewer reports only.
- Do not appear in printed output
- Can exist in any layout region
- Perform one of three actions:
 - Display multimedia information
 - Access a URL
 - Drill-down to a detail report



Copyright © Capgemini 2010. All Rights Reserved 80 |

Buttons

Buttons are used to execute the PL/SQL code or to activate a multimedia object. Buttons are layout objects on which users can click to do the following

- Display videos, sounds, or images that are fetched by a query
- Display videos, sounds, or images that are stored in files
- Execute PL/SQL to perform some action (such as launch a detail report for the current record in the Previewer). At runtime the button appears in the previewer, but does not get printed. Buttons may be given a text or iconic label via the button face option in the property sheet.

Anchor



- Objects in the push path have implicit anchors (not visible in layout model)
- An object below a vertical repeating frame
- An object to the right of a horizontal repeating frame
- An object below an object that has a variable vertical size
- An object to the right of an object that has a variable horizontal size
- Explicit anchors override implicit anchors (visible in layout model)
- All anchors appear in Object Navigator

Anchors

- Anchors are used to determine the vertical and horizontal positioning of a child object relative to its parent. The end of the anchor with a symbol on it is attached to the parent object.
 - Since the size of some layout objects may change when the report runs (and data is actually fetched), you need anchors to define where you want objects to appear relative to one another. An anchor defines the relative position of an object to the object to which it is anchored.
- Positioning is based on the size of the objects after the data has been fetched rather than on their size in the editor. It should also be noted that the position of the object in the Layout editor effects the final position in the report output. Any physical offset in the layout is incorporated into the percentage position specified in the Anchor property sheet.

There are two types of Anchors:

Implicit Anchors

Anchors can be used to keep objects aligned to one another. Oracle Reports creates implicit anchors automatically at runtime.

Explicit Anchors

You can create the explicit anchor to ensure that the two objects are organized in the layout according to your requirements.

Modifying Properties

- Create a link file
- Modify common layout properties
- Modify specific layout properties



Copyright © Capgemini 2010. All Rights Reserved. 62

Comparing Properties



- 1. UNFREEZED PROPERTY PALETTE
- 2. FREEZED PROPERTY PALETTE

Layout Object Properties

- Four objects with common properties:
 - Frame
 - Repeating Frame
 - Field
 - Boilerplate
- Some common properties affect:
 - Sizing
 - Pagination
 - Frequency of Display



Copyright © Capgemini 2010. All Rights Reserved 34

Property Palette for Items

- Try to give proper names to all layout items, including text fields when you can. It might help when designing complex layouts.
- If you want to sort a particular column in the query in descending order, use the Break Order Property; set it to *descending*.
- If you are going to do a summary of data found in the query, like a grand total, which is displayed even if no data is found, you should set *Value if Null* to 0, even if the query or formula column uses NVL. If no data is found and this is not set, the summary column may return NULL in the report.
- Page Break Before and Page Break After are used to format the data so that a section starts on a new page or prints subsequent data on a new page respectively.
- Page Protect tries to fit that item in a single page by breaking appropriately before that item if necessary.
- Minimum Window lines tells the report the minimum number of lines of an item that should print before the page breaks.
- Format trigger is a code that runs when an item is about to be printed. If the code returns TRUE, the item is actually printed, while if it returns FALSE, the item is not printed on the screen. Format triggers can be used to hide layout items that are required conditionally, like No Data Found sections. Conditional formatting does the same thing.

- Print Object On: If set to All Pages, prints the object on all logical pages where the object could be printed based on other constraints.
- Base Printing On: If the object is the last object in that section, its printing is always based on the Anchoring object, that is, the object directly above it. Enclosing object is set when the object is the first object in the section

After building the report in Report Builder, compile it and save the file as REPORT_NAME.rdf. Then place the rdf in appropriate directory in unix box depending on the application the report belongs to. If the report belongs to a GL application copy the report in \$GL_TOP/reports/US directory.

Adding Page Number

A running page number can be provided in the Layout Editor to be visible in the Live Previewer.

Steps involved.

Insert a new field in the Layout. Select 'Field' from the Tool Palette and click on the Layout.

Go to its property sheet.

Select the SOURCE drop down.

Select Page Number from it.

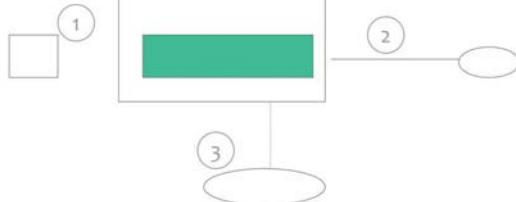
Sizing Objects

	Layout icons	
	Vertical	Horizontal
fixed		
expand		
contract		
variable		

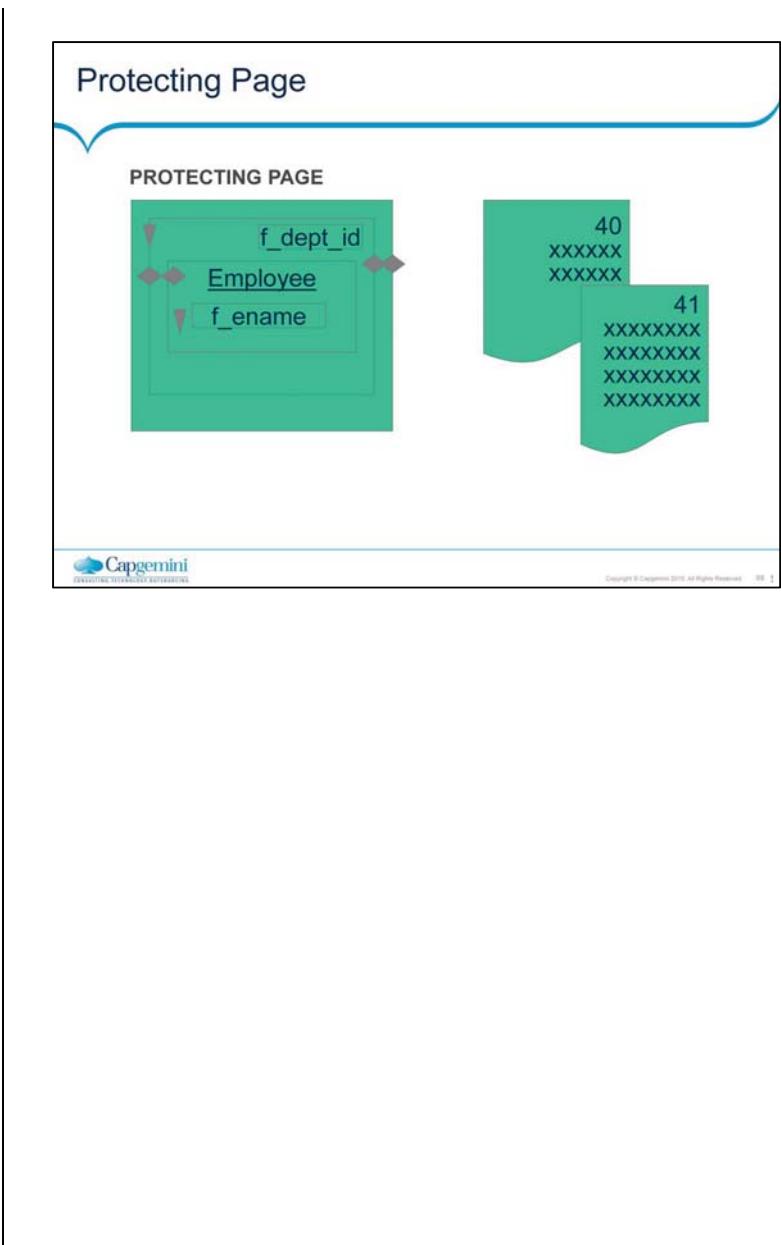
Capgemini
CONSULTING INTEGRATION CONSULTING

Copyright © Capgemini 2010. All Rights Reserved. 100

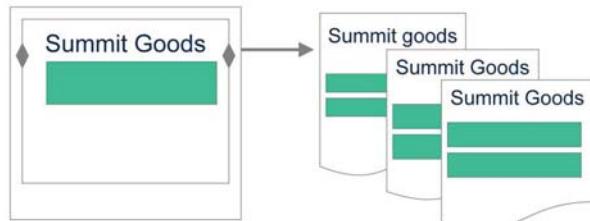
Layout Object Relationships



- No relationship, No anchor
- Explicit anchor to enforce relationship
- Implicit anchor because object is in the push path of a frame



Controlling Print Frequency



- Print Object On = All Pages
- Base Printing On = Enclosing Object

Demo

- Create Multiple Repeating Frames



Copyright © Capgemini 2010. All Rights Reserved 70

Capgemini
EXECUTIVE INFORMATION SYSTEMS

Add the notes here.

Summary

- In this lesson, you have learnt:
 - Basic Design Steps
 - Components of Reports
 - Data Model
 - Layout Model



Review Questions

- Question 1: The _____ is composed of objects that define the data to be included in a report.
- Question 2: The _____ view is a work area in which you can define the format of your report output.
- Question 3: _____ consists of text and graphics that appear in the report each time it is run



Review Questions – Match The Following

- a. are used to establish Parent-Child relationship between queries and groups via column matching
- b. SQL SELECT statements that define which rows and columns from specified tables or views are to be fetched from the database
- c. represent the data of your report
- d. variables for your report that enable you to change selection criteria at runtime
- e. determine the hierarchy of the data appearing in a report, and are used primarily to create breaks.

1. Queries
2. Groups
3. Columns
4. Link



ERP- Oracle Apps

Lesson 4: Triggers in Oracle Reports

Overview

- Topic Details - Line 1
- Topic Details - Line 2
 - Line 2.1
- Topic Details - Line 3
 - Line 3.1.
 - Line 3.1.1
- Topic Details - Line 4



Copyright © Capgemini 2014. All Rights Reserved.

Overview

- Topic Details - Line 1
- Topic Details - Line 2
 - Line 2.1
- Topic Details - Line 3
 - Line 3.1.
 - Line 3.1.1
- Topic Details - Line 4



Copyright © Capgemini 2014. All Rights Reserved.

Add the notes here.

Lesson Objectives

- To understand the following topics:
 - Types of trigger
 - Use of trigger
 - Formula triggers



Basic Types of Triggers

- Report-level:
 - Five triggers
 - Report Triggers node in Object Navigator
- Data Model:
 - formula (column)
 - filter (group)
 - parameter validation
- Layout:
 - format trigger on most objects



Copyright © Capgemini 2010. All Rights Reserved. 5

Report triggers

- Before Parameter Form
- After Parameter Form
- Before Report
- Between Pages
- After Report

Report Level Triggers

- Report Trigger
 - After Parameter Form
 - Example: Build dynamic 'where' clause

```
FUNCTION AfterPForm RETURN BOOLEAN IS
BEGIN
  IF :p_customer IS NULL THEN
    :p_where_clause := '';
  ELSE
    :p_where_clause := 'where id >= :p_customer';
  END IF;
  RETURN(TRUE);
END;
```

- Query Syntax

```
SELECT ID, NAME
  FROM CUSTOMER
&P_WHERE_CLAUSE
  ORDER BY NAME
```


Copyright © Capgemini 2010. All Rights Reserved.

Before Parameter Form

Fires before the Runtime Parameter Form is displayed.

You can access and change the values of parameters, PL/SQL global variables, and report-level columns

After Parameter Form

Fires after the Runtime Parameter Form is displayed.

You can access parameters and check their values.

Columns from the data model are not accessible from this trigger.

Before Report

Fires before the report is executed but after queries are parsed.

After Report

Fires after you exit the Previewer, or after report output is sent to a specified destination, such as a file, a printer.

This trigger can be used to clean up any initial processing that was done, such as deleting tables

Between Pages

Fires before each page of the report is formatted, except the very first page.

This trigger can be used for customized page formatting.

Data Model Level Trigger

■ Group Filter

- Restrict records in a group
- Perform PL/SQL for each record

```
FUNCTION G_empGroupFilter RETURN BOOLEAN IS
BEGIN
  IF :name = 'Operations' AND :salary > 3000 THEN RETURN(my_function);
  ELSE
    RETURN(TRUE);
  END IF;
END;
```

- PL/SQL filters result in ALL records being fetched.



Copyright © Capgemini 2010. All Rights Reserved 7 |

Group Filter

You can group together multiple filter conditions and nest filter condition groups within other filter condition groups by dragging and dropping. You are not limited in the number of groups you can use. You can create a filter condition group first or, if you prefer, you can add your filter conditions to the filter area, create a filter condition group, and then move the filter conditions into the group.

To create a filter condition group

- To open the **Filter Data** dialog box, click **Filter** on the **Report** menu.
- In the **Filter Data** dialog box, click **New Group**.

Select the option that you want to use. A new filter condition group is displayed in the filter area. The title of the filter condition group indicates the **New Group** option that you selected. When creating a new group, the new group is added to the end of the currently selected group, and becomes the new selected group.

Data Model Level Trigger

- Parameter validation
 - Example: do not allow report output to be sent directly to a printer
- ```
FUNCTION DESTYPEValidTrigger RETURN BOOLEAN IS
BEGIN
 IF :DESTYPE = 'Printer' THEN
 RETURN(FALSE);
 ELSE
 RETURN(TRUE);
 ENDIF;
END;
```
- You cannot reassign values to parameters or columns in this trigger



Copyright © Capgemini 2010. All Rights Reserved.

8 |

### Validation triggers

Validation Triggers are PL/SQL functions that are executed when parameter values are specified on the command line and when you accept the Runtime Parameter Form.

Note: For JSP-based Web reports, the Runtime Parameter Form displays when you run a report in Reports Builder, but does not display in the runtime environment. If parameters are not specified on the Runtime Parameter Form, the validation trigger returns false and generates error message rep-546 Invalid Parameter Input error. Validation triggers are also used to validate the Initial Value property of the parameter. Depending on whether the function returns TRUE or FALSE, the user is returned to the Runtime Parameter Form.

## Layout / Format Trigger

### Format Triggers:

- Exist on most layout objects
- Can suppress entire layout section (master group frame): no records fetched
- Can suppress display of individual records (repeating frame): all records fetched



Copyright © Capgemini 2010. All Rights Reserved 9

### Format Triggers

A format trigger is a PL/SQL function executed before an object is formatted. A trigger can be used to dynamically change the formatting attributes of the object. For example, you can use a format trigger to cause a value to display in bold if it is less than zero. Another example is to use a format trigger to use scientific notation for a field if its value is greater than 1,000,000.

A format trigger can fire multiple times for a given object, whenever Reports Builder attempts to format the object. Consider the case where Reports Builder starts to format the object at the bottom of a page. If the object does not fit on the page, Reports Builder stops formatting and reformats on the following page. In this case, the format trigger will fire twice. It is therefore not advisable to do any kind of "persistence" operation, such as logging.

## Example

The screenshot displays two windows side-by-side. On the left is the 'L13FIELD: Program Unit - P\_1VALID\_TRIGGER' window, which is an Oracle SQL editor showing the following PL/SQL code:

```
function F_SRK_BOOK_IDFormatText return boolean is
begin
 if user = 'nmidrad'
 then
 return FALSE;
 else
 return (TRUE);
 end if;
end;
```

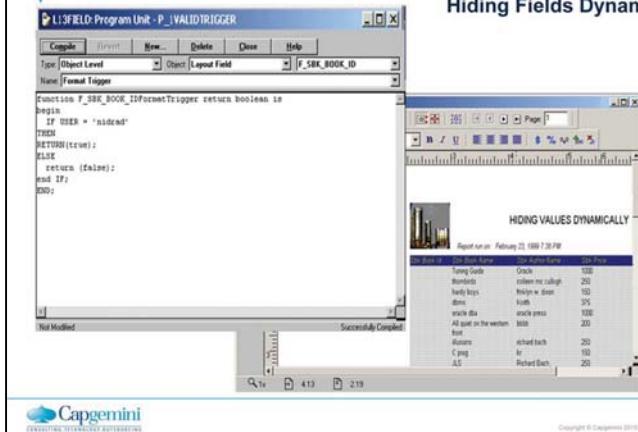
On the right is the 'MONKELL: Report Editor - Paper Design' window, showing a report layout with a title 'YOUR Inc. COMPUTER' and a section titled 'HIDING VALUES DYNAMICALLY'. Below this is a table with the following data:

| Order | Customer Name         | User/Author Name   | Unit Price |
|-------|-----------------------|--------------------|------------|
| sr140 | thousands             | colleen mc callagh | 250        |
| sr141 | thousands             | franklin w. stoen  | 100        |
| sr142 | handy boys            | franklin w. stoen  | 375        |
| sr143 | handy boys            | franklin w. stoen  | 375        |
| sr144 | apple dts             | oracle press       | 1000       |
| sr145 | apple dts             | oracle press       | 200        |
| sr146 | All about the western | oracle press       | 1000       |
| sr147 | true                  | richard bach       | 250        |
| sr148 | the wisdom            | richard bach       | 250        |
| sr149 | C ping                | kr                 | 150        |
| sr150 | ...etc                | richard Bach       | 250        |

Below the table, the text 'Report run on: Friday, 23, 1999 7:43 PM' is visible.

## Example

### Hiding Fields Dynamically



## Demo

- Create a Report using Report Wizard and withTriggers



Add the notes here.

## Summary

- In this lesson, you have learnt:
  - Types of trigger
  - Use of trigger
  - Formula triggers



## **ERP- Oracle Apps Forms and Reports**

Lesson 5: Built in Packages  
and Miscellaneous Settings

## Lesson Objectives

- To understand the following topics:
  - Built-in Package (SRW Package)
  - Bitmap and Character Mode
  - Fonts and Colors
  - Setting NLS Language Environment Variables



5.1: Built-in Package (SRW Package)

**Contents**

|                                                  |                                               |
|--------------------------------------------------|-----------------------------------------------|
| SRW.BREAK<br>SRW.SET_ATTR                        | SRW.DO_SQL<br>SRW.RUN_REPORT                  |
| SRW.REFERENCE<br>SRW.USER_EXIT<br>SRW.SET_MAXROW | SRW.MESSAGE<br>SRW TRACE<br>SRW.PROGRAM_ABORT |

Capgemini  
CONSULTING INTEGRATION CONSULTANCY

Copyright © Capgemini 2014. All Rights Reserved.

### SRW PACKAGE

The Report builder Built in package know as SRW Package This package extends reports ,Control report execution, output message at runtime, Initialize layout fields, Perform DDL statements used to create or Drop temporary table, Call User Exist, to format width of the columns, to page break the column, to set the colors Ex: SRW.DO\_SQL, It's like DDL command, we can create table, views , etc., SRW.SET\_FIELD\_NUM SRW. SET\_FIELD\_CHAR SRW. SET FILED \_DATE

#### Srw.Set\_maxrow

This procedure sets the maximum number of records to be fetched for the specified query. This is useful when your report formats (i.e., displays) fewer records than the query (or queries) that fetch them. Thus, with SRW.SET\_MAXROW, you can conditionally restrict data that is fetched for your report, enabling you to improve the report's performance.

#### Syntax

SRW.SET\_MAXROW (query\_name CHAR, maxnum PLS\_INTEGER);

### **Srw.Run\_report**

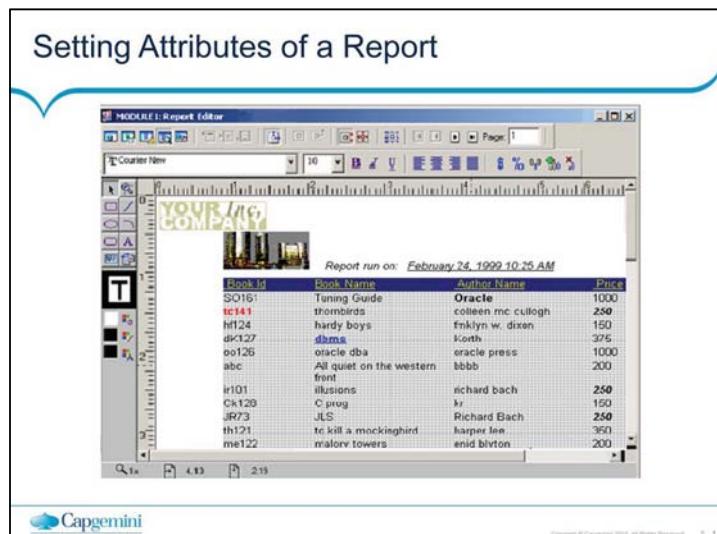
This procedure invokes RWRUN60 with the string that you specify.

This procedure is useful for:

- running drill-down reports (i.e., calling a report from a button's action trigger)
- sending parts of a report to different recipients (e.g., to send a report via e-mail to each manager with just his or her group's data)
- sending parts of a report to different printers (e.g., to send each manager's report to his or her printer)
- running multiple reports from a single "driver" report
- <!--[if !supportLists]-->  
SRW.RUN\_REPORT executes the specified RWRUN60 command.

### **Syntax**

SRW.RUN\_REPORT (command\_line CHAR);



#### Srw.User\_exit

This procedure calls the user exit named in user\_exit\_string. It is useful when you want to pass control to a 3GL program during a report's execution.

Syntax      SRW.USER\_EXIT (user\_exit\_string CHAR);

#### Srw References

This procedure causes Report Builder to add the referenced object to the PL/SQL construct's dependency list. This causes Report Builder to determine the object's value just before firing the PL/SQL construct. This is useful when you want to ensure that a column value passed to a user exit is the most recently computed or fetched value.

Syntax      SRW.REFERENCE (:object CHAR|DATE|NUMBER);

#### Srw.Program\_abort

This exception stops the report execution and raises the following error message:

REP-1419: PL/SQL program aborted.SRW.PROGRAM\_ABORT stops report execution when you raise it.

Syntax      SRW.PROGRAM\_ABORT;

## Outputting Message

- Warning

```
WHEN <exception> THEN
SRW.MESSAGE(999, 'Warning: report continues');
```

- Error

```
WHEN <exception> THEN
SRW.MESSAGE(999, 'Error: report terminated');
RAISE SRW.PROGRAM_ABORT;
```



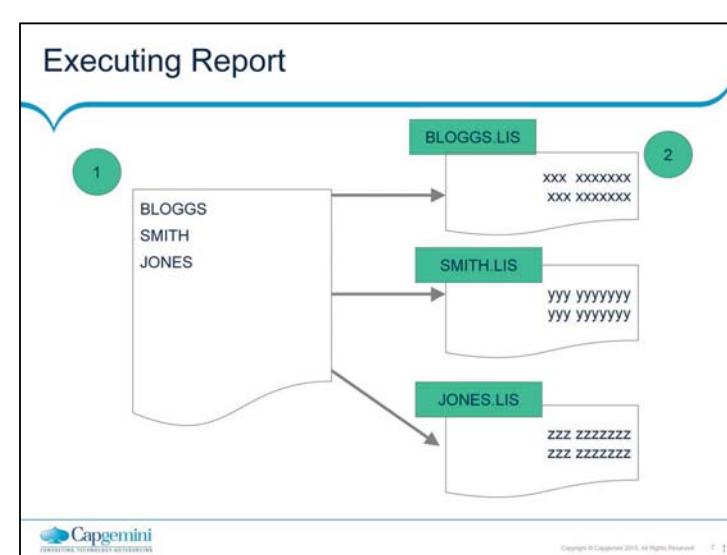
Copyright © Capgemini 2010. All Rights Reserved. 6 |

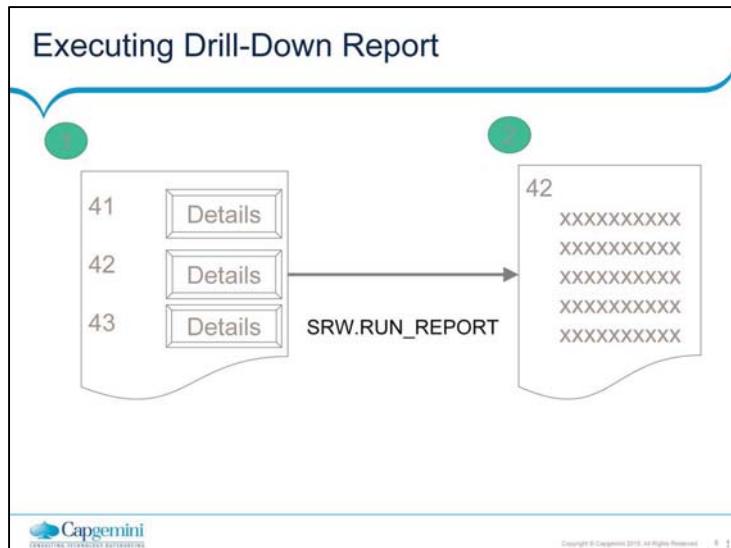
### Srw.Message

This procedure displays a message with the message number and text that you specify. The message is displayed in the format below. After the message is raised and you accept it, the report execution will continue.  
MSG-msg\_number: msg\_text.

### Syntax

```
SRW.MESSAGE (msg_number NUMBER, msg_text CHAR);
```





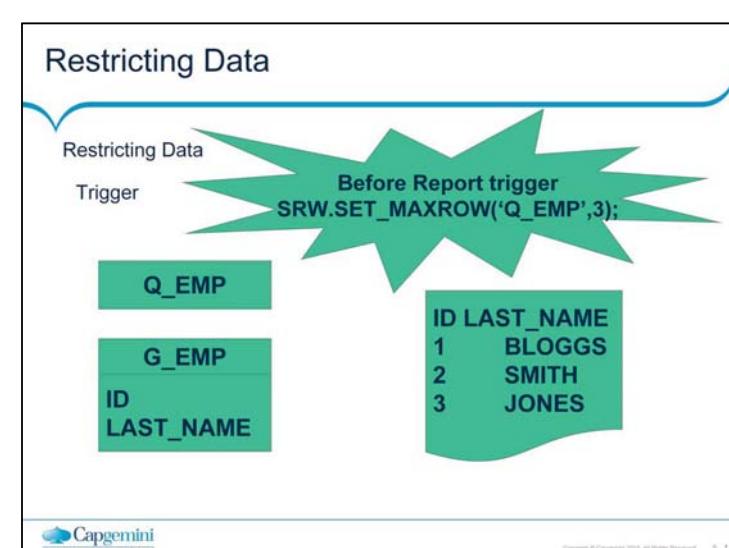
### Srw.Run\_report

This procedure invokes RWRUN60 with the string that you specify.  
This procedure is useful for:

- running drill-down reports (i.e., calling a report from a button's action trigger)
- sending parts of a report to different recipients (e.g., to send a report via e-mail to each manager with just his or her group's data)
- sending parts of a report to different printers (e.g., to send each manager's report to his or her printer)
- running multiple reports from a single "driver" report  
    <!--[if !supportLists]-->  
        SRW.RUN\_REPORT executes the specified RWRUN60 command.

### Syntax

SRW.RUN\_REPORT (command\_line CHAR)



#### Srw.Set\_maxrow

This procedure Srw.Run\_report. This is useful when your report formats (i.e., displays) fewer records than the query (or queries) that fetch them. Thus, with `SRW.SET_MAXROW`, you can conditionally restrict data that is fetched for your report, enabling you to improve the report's performance.

#### Syntax

```
SRW.SET_MAXROW (query_name CHAR, maxnum PLS_INTEGER);
```

## Initializing Fields

Output: logical

Layout editor

Page: F\_NEWPAGE

```
FUNCTION F_LOGICAL_PAGEFormatTrigger
 RETURN BOOLEAN IS
 my_page number;
BEGIN
 SRW>GET_PAGE_NUM(my_page);
 SRW.SET_FIELD_NUM(0,my_page+3);
 RETURN(TRUE);
END;
```

Page : 5

The diagram illustrates the initialization of a field. On the left, a 'Layout editor' window shows a field labeled 'Page: F\_NEWPAGE'. A green oval with the number '1' is positioned next to the field. An arrow points from this oval down to a green rectangular box containing PL/SQL code. This box contains a function named 'F\_LOGICAL\_PAGEFormatTrigger' that returns a boolean value. Inside the function, there is a call to 'SRW>GET\_PAGE\_NUM(my\_page)' and 'SRW.SET\_FIELD\_NUM(0,my\_page+3)'. The function ends with 'RETURN(TRUE);'. To the right of the code box, another green oval with the number '2' is shown. An arrow points from the code box up to this oval. Further up, another arrow points from the oval to a white box labeled 'Page : 5'. This visualizes how the function's logic is triggered by the field's value and updates the page number.

### Srw.Get\_page\_num

This procedure returns the current page number. This is useful when you want to use the page number in the field's Format Trigger property.

#### Syntax

SRW.GET\_PAGE\_NUM (page\_num);

## Performing DDL Statement

- Example

```
SRW.DO_SQL('CREATE TABLE SRW_LOG
 (RPT_NAME VARCHAR2(40),
 REC_NUM NUMBER,
 MSG_TEXT VARCHAR2(80))');
SRW.DO_SQL('INSERT INTO SRW_LOG
 (RPT_NAME REC_NUM, MSG_TEXT)
 VALUES
 ("PAY_REPORT", TO_CHAR(:ID),
 :LAST_NAME||"PAY REPORT RUN")');
```

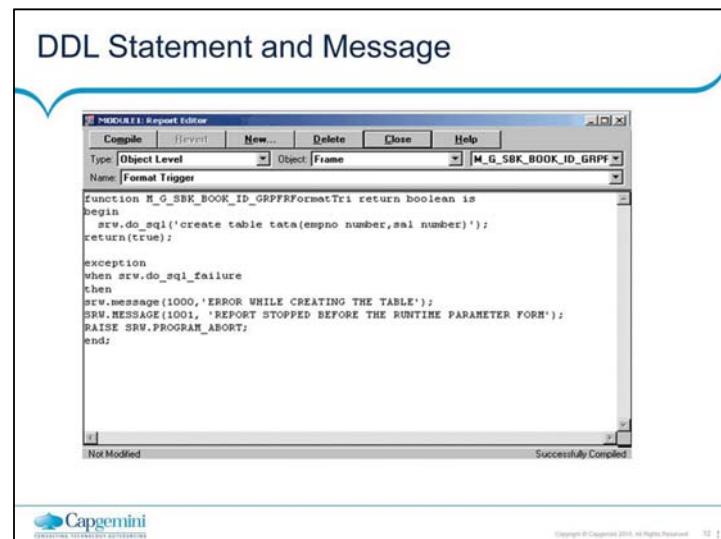
- Exception

- SRW.DO\_SQL\_FAILURE



Copyright © Capgemini 2014. All Rights Reserved.

## ERP- Oracle Apps Forms and Reports Built in Packages and Miscellaneous Settings



5.2: Miscellaneous Settings

## Bitmap and Character Mode

- Report Builder development
  - Bitmapped only
- Runtime
  - Bitmapped                            MODE = bitmap
  - Character Mode                    MODE = character

 Capgemini  
CONSULTING SERVICES & SYSTEMS

Copyright © Capgemini 2010. All Rights Reserved.

13

## Character Mode

- Output uses Fixed Font
  - Font type, size, or style does not hold any meaning.
  - Object gets mapped exactly to a set of character-cells without any overlapping (Character Cells)
  - Character mode reports run faster

### Bitmapped

- Output uses true Fonts.
  - Font type, size, or style does hold meaning
  - Objects gets mapped with overlapping (Pixels).

## Character Mode Reports

- Use predefined character mode template
- Modify report properties
- Avoid GUI objects:graphics,buttons,OLE2
- Avoid Borders
- Consider Fonts,Color,Fill Patterns



Copyright © Capgemini 2010. All Rights Reserved 34

Following are the mandatory steps for developing a character mode report

- Set the Report's(**Report -> Global Properties**) “Design in Character Units” property to “yes”
- Set the report width and height

E.g.

For Landscape 132 or 180 as width and 66 as height

For Portrait 102 as width and 85 or 116 as height

<< Embed the .rap file>>

- In the properties of system parameter “Mode”, set the initial value field to Character.
- In the layout editor’s view menu: switch *grid snap* on, and in *Format ~Layout options ~ ruler settings* set the number of *snap points per grid spacing* to one.
- Don’t use objects that are not consistent with character mode output e.g. images, colors , drawings, italics, ellipses, diagonal lines, drill down buttons, multimedia etc.

## Fonts and Colors

- Use common fonts
  - Or map fonts in uifont.ali
- Use common colors
- Beware of DPI (dots per inch)



Copyright © Capgemini 2014. All Rights Reserved.

## Setting NLS Language Environment Variables

- NLS\_LANG
- DEVELOPER\_NLS\_LANG,  
USER\_NLS\_LANG
- NLS\_LANG=French\_France.WE8DEC
- UNICODE
- NLS\_LANG=<lang>\_<territory>.AL24UTFSS



Copyright © Capgemini 2016. All Rights Reserved 10 |

A locale is a linguistic and cultural environment in which a system or program is running. Setting the NLS\_LANG parameter is the simplest way to specify locale behavior. It sets the language, territory, and character set used by the database for both the server session and the client application. Using this one parameter ensures that the language and territory environment for both the server and client are the same.

The NLS\_LANG parameter has three components (*language*, *territory*, and *charset*) in the form:

NLS\_LANG = language\_territory.charset Each component controls the operation of a subset of NLS features

Because NLS\_LANG is an environment variable, it is read by the client application at startup time. The client communicates the information defined by NLS\_LANG to the server when it connects.

**NLS\_LANGUAGE specifies the default conventions for the following session characteristics:**

- language for server messages
- language for day and month names and their abbreviations (specified in the SQL functions TO\_CHAR and TO\_DATE)
- symbols for equivalents of AM, PM, AD, and BC
- default sorting sequence for character data when ORDER BY is specified (GROUP BY uses a binary sort, unless ORDER BY is specified)
- writing direction
- affirmative/negative response strings

Demo

- Create a Report using Report Wizard and with all the above concepts



 Capgemini  
CONSULTING SERVICES

Copyright © Capgemini 2010. All Rights Reserved. 10

Add the notes here.

## Summary

- In this lesson, you have learnt:
  - Built-in Package (SRW Package)
  - Bitmap and Character Mode
  - Fonts and Colors
  - Setting NLS Language Environment Variables



Summary



Copyright © Capgemini 2014. All Rights Reserved. 18

## Review Questions

- Question 1: \_\_\_\_\_ procedure sets the maximum number of records to be fetched for the specified query
  
- Question 2: In \_\_\_\_\_ Mode Output uses Fixed Font  
Font type, size, or style does not hold any meaning.



Copyright © Capgemini 2014. All Rights Reserved. 10

## **ERP- Oracle Apps Forms and Reports**

Lesson 6: Introduction to  
Oracle Forms

## Lesson Objectives

- To understand the following topics:
  - Understand the features of Developer 2000 Forms
  - Basic Design Steps
  - Canvas, Data Blocks and Wizards



## 6.1: Introduction to Forms What is Form Builder?

- Form Builder is a powerful development tool for building client/server applications.
- Developers use Form Builder to create applications that provide end users access to information stored in a database.
- End users operate Form Builder applications to retrieve, enter, modify, and save information in the database.



Copyright © Capgemini 2015. All Rights Reserved.

## 6.1: Introduction to Forms What is Form Builder?

- To speed development and help you get started generating objects for your application, Form Builder provides wizards.
- The wizards' application-generation capabilities leave both new and experienced developers free to work on other aspects of a form module.



Copyright © Capgemini 2010. All Rights Reserved.

6.1: Introduction to Forms

## Key Features

- With Form Builder you can :
- Insert, update, delete, query data
- Present data as text, image etc
- Control forms across several windows
- Use customized menus
- Send data to Report Builder

 Capgemini  
Driving Methods for Businesses

Copyright © Capgemini 2010. All Rights Reserved.

6.1: Introduction to Forms

## Oracle Forms Consists of

- Form Builder
- Data Blocks, Frames, Text Items and LOV
- Input and Non-input Items
- Widows and Canvases
- Triggers and Functions
- Flexible code
- Sharing and Code
- Multi Form Applications



Copyright © Capgemini 2010. All Rights Reserved. 6 - 2

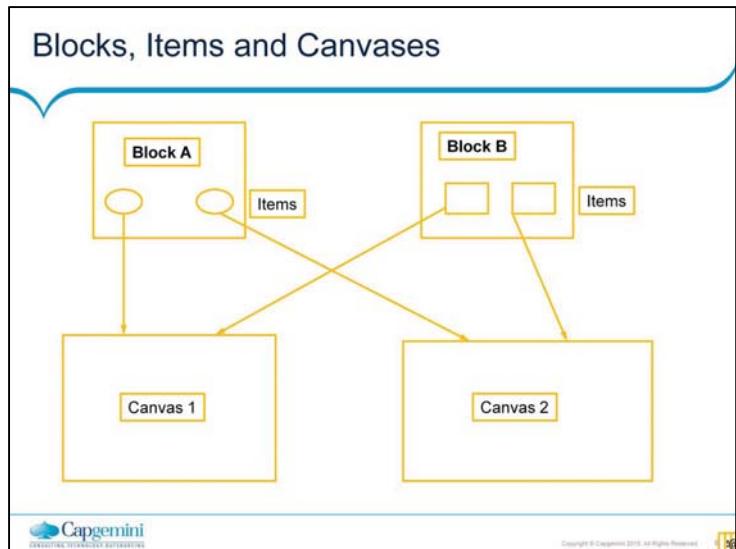
## 6.2: Form Design Basic Design Steps

- Form Builder
- Create a form module
- Create a data block
- Modify a data block using the data block Wizard
- Modify the layout using the layout wizard
- Creating a Master-Detail Form



Copyright © Capgemini 2010. All Rights Reserved

7



#### Blocks

Blocks are logical containers that have no physical representation only the items contained in a block are visible in the application interface. Each block can relate directly to a single database or view. Each item in a form (text item, image item, radio group, and so on) belongs to a block.

#### Canvas

A canvases is a surface -inside a Window Container -on which you place the interface items and boilerplate objects that end users interact With when they run the form. A canvas has an attribute called viewport. The viewport defines the area of the canvas, which is displayed in a Window at runtime, called the view. The operations that can be done on a canvas are showing and hiding the canva s, resizing the canvas and setting visual attributes.

#### Item

An item corresponds to a single data element of field. These items may contain database columns or may be used as containers for other related data.

Items are assigned to canvases; canvases are assigned to windows. Each window can have multiple canvases, and there can be multiple items on a single canvas.

## Navigation in a Block

Canvas1



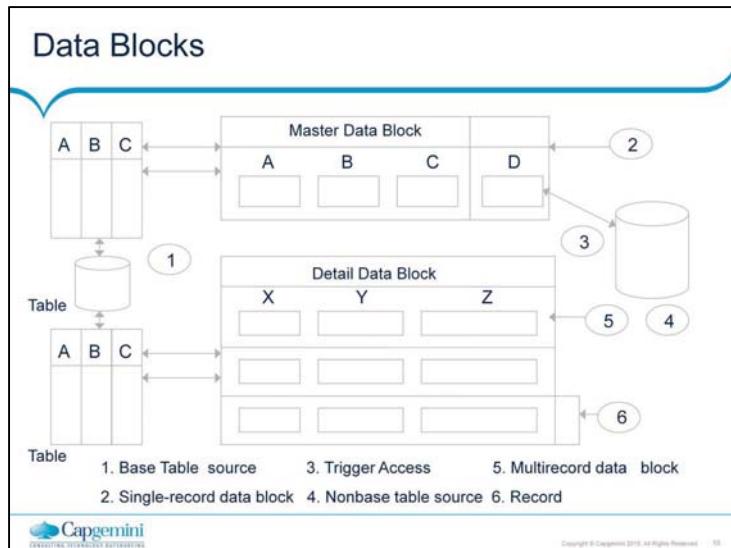
Canvas2



Copyright © Capgemini 2010. All Rights Reserved

### Navigation

The user interacts with the application by moving the focus from object to object. This movement is 'navigation'. By navigating from item to item, record to record, block to Process block or form to form the user controls the tasks that comprise the application Navigation Events (going to a different item, going to a different record, going to a different block, going to a different form) constitute the basic events of your form application.

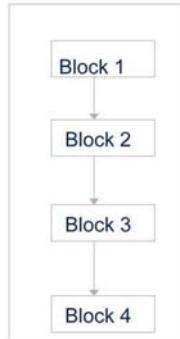


### Data Blocks

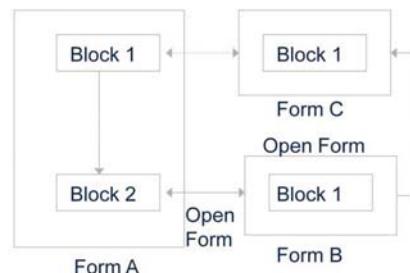
A Data Block is associated with a database table or view, or a set of stored procedures. Most often, data blocks are based on a single database table. A data block automatically includes the functionality to support querying, updating, inserting, and deleting rows in the table to which the block corresponds. A master- detail relationship is an association between two data blocks--a master and a detail block. The relationship between the blocks reflects a primary key foreign key relationship between the tables on which the blocks are based. The master block is related to the detail block through the join condition.

## Forms and Data Blocks

### ■ Data Blocks



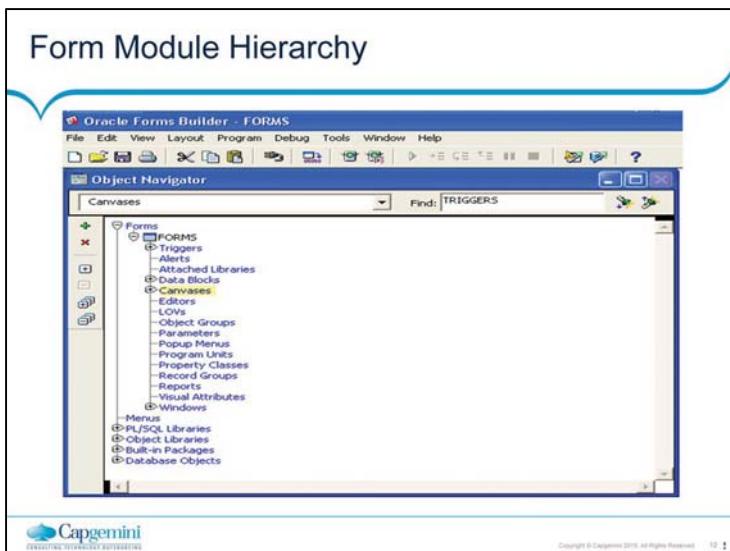
Single Form Module



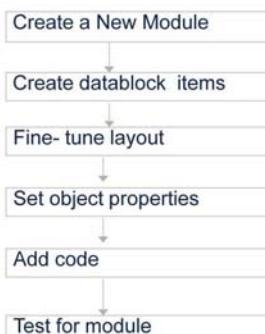
Multiple Form Modules



Copyright © Capgemini 2010. All Rights Reserved 31



## Creating a New Form Module

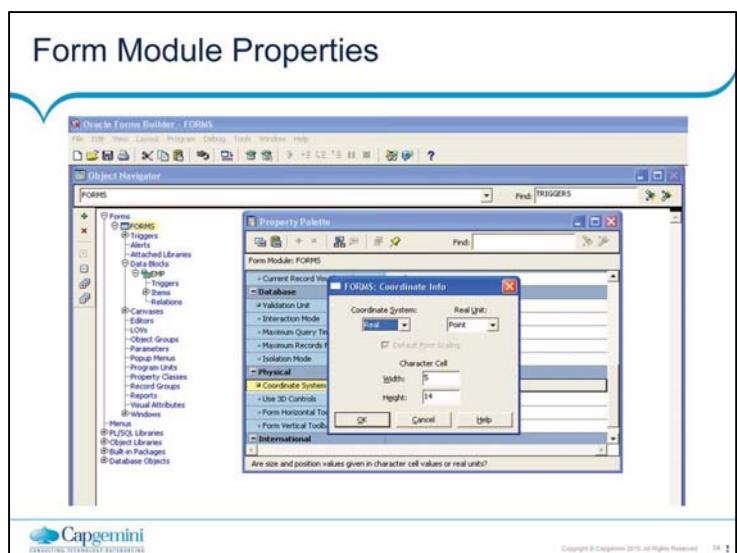


Copyright © Capgemini 2010. All Rights Reserved 13

The Form Module allows the programmer to create forms to facilitate data manipulation tasks, such as adding, reviewing, changing and removing data from the database. It consists of objects and code routines. In a form module you can define objects like windows, fields, check boxes etc.

The Data Block Wizard and Layout Wizard are powerful tools that allow you to quickly and easily complete the tasks of creating a data block (Using data block Wizard) and laying out its interface objects (Using Layout Wizard).

The Property Palette enables you to set the properties of objects you create in form and menu modules. When you select an object in an editor or in the Object Navigator, the Property Palette updates to show the properties of that object. When you need to compare the properties of two different objects, you can invoke additional Property Palettes as needed. Tip: Use the Freeze icon in the horizontal tool bar, in the Property Sheet.



## Creating a New Module

- Click on “Forms” 
- Click on the plus sign  of the tool palette
- “Module1” will appear
- A new module is now ready



Copyright © Capgemini 2010. All Rights Reserved 10 - 2

## Creating Data Block Items

- Click on the Module1(last created ) and click the plus sign of the tool palette
- A window showing the options would appear
- Select the option of using “wizard”
- Click “next” on every screen till you reach the “layout editor”(in between you will be asked to enter the table which you can do with the “browse “ button.
- Proceed further till you reach “layout editor”



Copyright © Capgemini 2010. All Rights Reserved 10 | Page

## Fine tune layout

- By now you can see the items on the layout editor screen , which are nothing but the database columns represented on the screen
- Try to resize the items single click on one of the item and then keeping the mouse on the item's boundary expand/contract it
- By going at the tool palette you can fill up the colours also in the items



Copyright © Capgemini 2010. All Rights Reserved 17 / 2

## Set object properties

- To alter the properties of any Module/DataBlock/Item ,click on any one of them and then go to the “Tool ” option of the menu
- Adjust the property sheet as you desire



Copyright © Capgemini 2010. All Rights Reserved 10 |

## Add Code

- To write a pl/sql code on any Module/DataBlock/Item , go to the "Program" of the menu structure and click it
- Write the code and click "compile"
- Click "close"(do not choose the delete option)



Copyright © Capgemini 2010. All Rights Reserved 10

## Test for Module



- Now go to the object navigator and click
- The runtime screen would appear
- You can enter the data into the database tables (through the items being displayed on the screen)
- Open the SQL\*PLUS to verify your data entry into the data base tables

## Saving a Form Module

- Save form module:
  - File -> Save
  - Save Icon
- Enter Filename
- Save to one of the following:
  - File System
  - Database



Copyright © Capgemini 2010. All Rights Reserved. 21

## Data Block Functionality

- Once you create a data block with the wizards, Form Builder automatically creates:
  - Form Module with database functionality
    - Query,insert,update,delete
  - Frame Object
  - Items in the data block (TRIGGERS)
  - Prompt of each item



Copyright © Capgemini 2010. All Rights Reserved 22 2

## Compiling the Form Module

- Compile Explicitly
  - File-->Administration -->Compile file
  - Form Compiler Component
  - Command line interface
- Compile implicitly
  - Tools -->Preferences
  - Set the build before running preference



Copyright © Capgemini 2010. All Rights Reserved 23

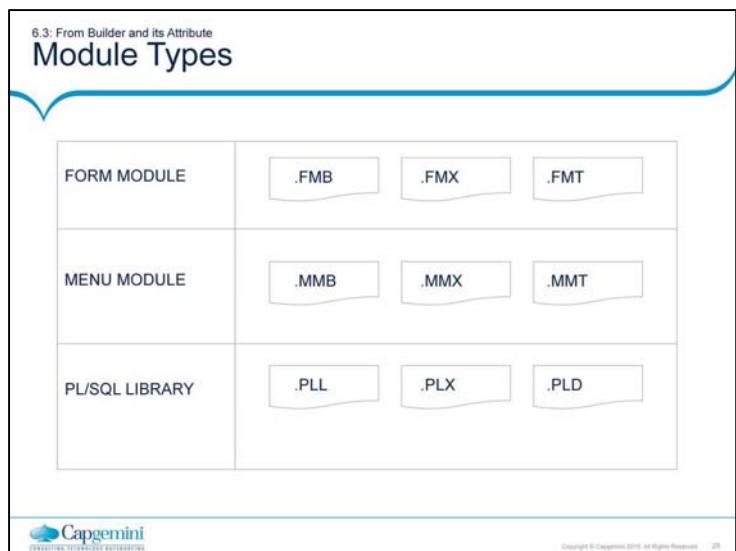
**Demo**

- Create a Form Manually and with wizard



**Capgemini**  
CONSULTING INTEGRATION SERVICES

Copyright © Capgemini 2010. All Rights Reserved. 24

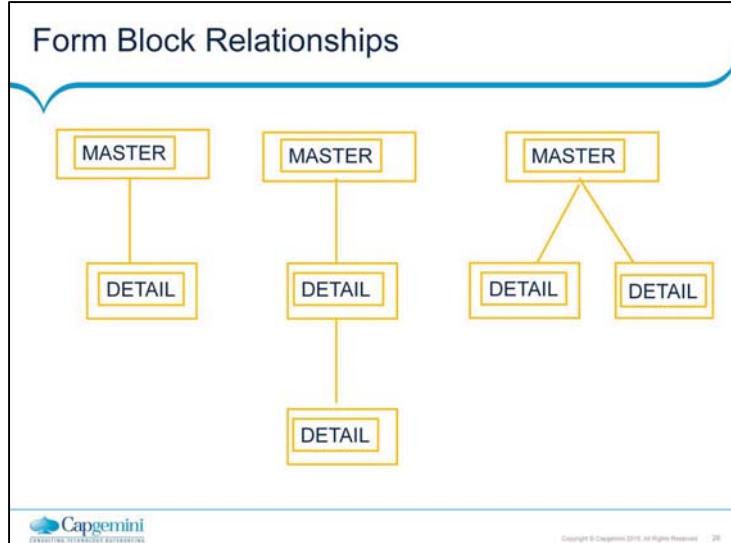
**.FMB and .FMX**

When you build an application in the Forms Builder, a .FMB (Form Module Binary) file gets created. Compiling a form module compiles all of its code objects and creates a .FMX (Form Module Executable) runfile.

**.PLL** : The library .PLL file contains both library source code and the compiled, executable code. The >PLL file is created or updated when you save your library module. When you save your library module, the changes are reflected in each module to which the library is attached.

**.PLX** : The library .PLX file is a platform-specific executable that contains no source. You can generate a .PLX file from the Designer or from the command line. When you attach a library to a module in the Designer, you attach the .PLL version of the library. At runtime, Form Builder looks for a .PLX file by that name in the default search path. If no .PPLX file is present, Form Builder looks for the .PLL file by that name.

**.PLD** : The >PLD file is a text format file, and can be used for source-controlling your library files.



#### Master-detail block

The master-detail relationship automatically does the following:  
Ensures that the detail block displays only those records that are associated with the current (master) record in the master block. Co-ordinates querying between the two blocks.

#### Master-with-dependent-details

A master-with-dependent-details relation includes a master block and n levels of detail blocks, such that the first detail block is itself a master for its own detail block. When you create this type of relation, consider the effect of the Delete Record Behavior and Coordination properties on the detail blocks. For example, if one relation in a chain of related blocks is set to deferred coordination, all subsequent blocks will also be deferred.

#### Master-with-independent-details

A master-with-independent-details relation involves two or more detail blocks, both sharing the same master block. This structure is useful when you want to display more than one set of detail records for a single master record.

## Relation Object

- New relation object in Object Navigator under master data block node
- Default name
  - MasterDataBlock\_DetailDataBlock
- Triggers and Program Units generated Automatically



Copyright © Capgemini 2010. All Rights Reserved 27 / 30

### Relation Object

You can define a master-detail relationship at any time during the Form development process. You define a master-detail relationship between blocks by creating and setting the properties of a relation object. A relation is a logical object that specifies the relationship between one master block and one corresponding detail block. In the Object Navigator, the relation object appears under the block that is the master block in the relation. When you create a relation, Form Builder generates the triggers and PL/SQL procedures required, to enforce co-ordination between the master and detail blocks.

## Running A Master-Detail Form Module

- Automatic Block linking for
  - Querying
  - Inserting
- Default Deletion rules
  - Cannot delete Master record if detail record exists



Copyright © Capgemini 2010. All Rights Reserved 28 2

- To maintain the master -detail relationship at runtime, Form Builder co-ordinates master and detail blocks to ensure that the records displayed in the detail block associated with the current record in the master block. Any event that makes a different record in the master block the current record is ordination-causing event. During the population phase, Form Builder issues a SELECT statement to repopulate the detail block with the detail records associated with the new master record. These operations are accomplished through the execution of triggers.
- The default setting prevents the deletion of a master record if associated detail records exist in the database.

## Working with Data Blocks and Frames

- Identify the components of the property palette
- Manipulate properties through the property palette.
- Control the behaviour and appearance of data blocks
- control frame properties
- create blocks that do not directly correspond to the database
- Delete data blocks and their components



Copyright © Capgemini 2010. All Rights Reserved 29

## Modifying the Appearance and Behavior of Data Blocks

- Reentrant Wizards
- Layout editor
- Data Block property palette
- Frame property palette



Copyright © Capgemini 2010. All Rights Reserved 30 2

### Layout Editor

The Layout editor enables you to quickly layout the items of a data block.

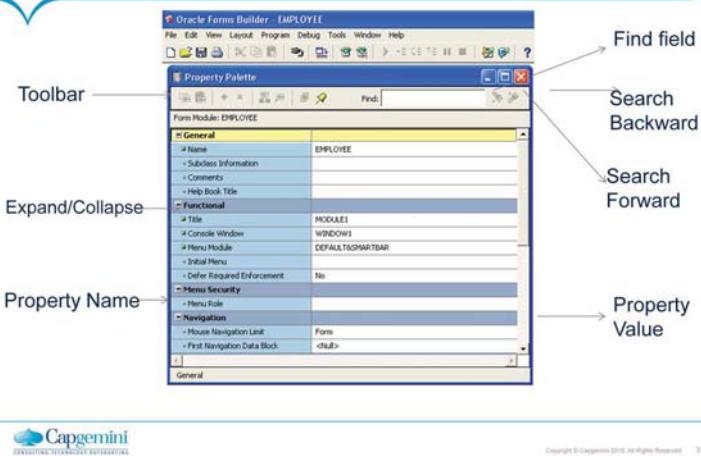
### Functions of the Layout Editor

- Resize the canvas and viewport
- Create, move and resize the items
- Invoke item property sheets
- Add boilerplate text and graphics to the canvas
- Set the font, color and font pattern attributes of the items and graphics
- Cut, copy and paste the items and graphics
- Import and export text, images and drawings to and from a canvas

### Property Palette

Each Oracle Developer 2000 provides a property palette to review and modify the properties of any object within the application. The properties are displayed in forms with a property window.

## Displaying the Property Palette



## Visual Attributes

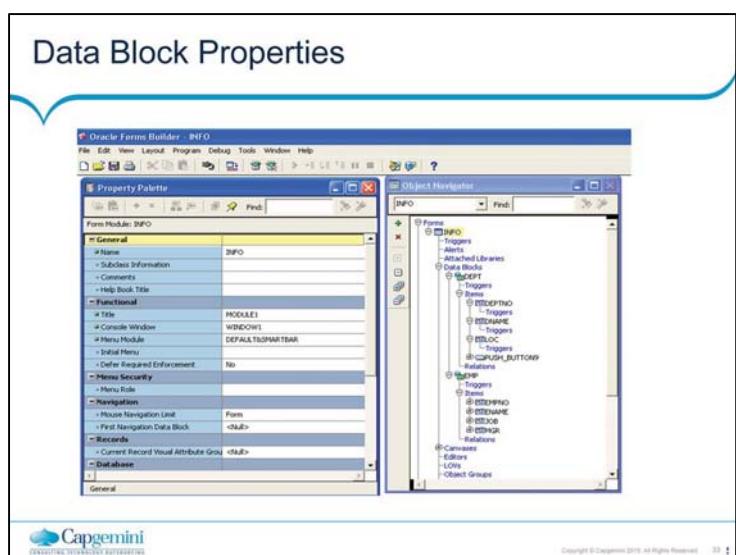
- Visual attributes :
  - are font, colour, and pattern properties
  - can be set for form and menu objects
- A visual attribute is a form object with font, colour, and pattern properties
- Set visual attribute group property to the visual attribute object



Copyright © Capgemini 2010. All Rights Reserved 32 2

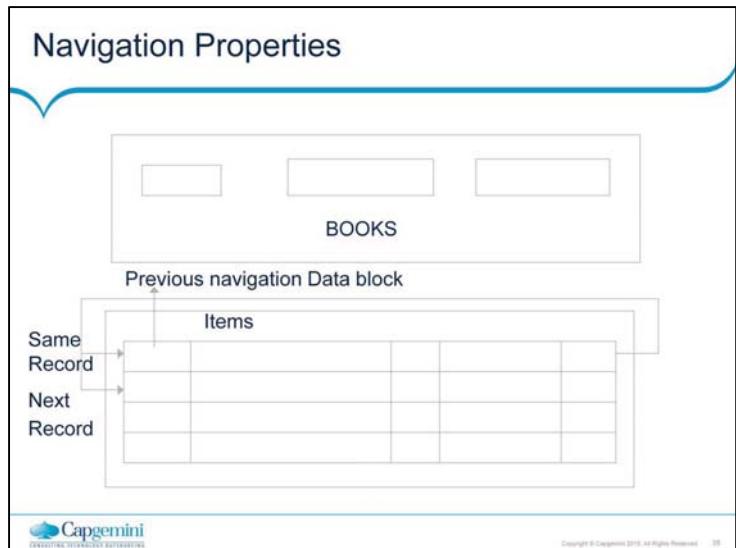
### Visual attributes

Visual attributes include font, color, and pattern properties that you set for form and objects that appear in your application. You can specify the visual properties of an item with the properties specified under the Fonts & Color node in the property palette of the item. You can specify the individual attribute setting like Font Name, Font Size etc. or attach a visual attribute group to the item. It determines object's individual Visual attribute settings, the look and feel (Font Size, Foreground Color, etc.).



## Data Block Property groups

- Each data block has several properties which may be divided into the following groups:
  - General
  - Navigation
  - Records
  - Database
  - Advanced Database
  - Scrollbar
  - Font and Colour
  - Character Mode
  - International



The user interacts with the application by moving the focus from object to object. This movement is 'navigation'. By navigating from item to item, record to record, block to Process block or form to form the user controls the tasks that comprise the application Navigation Events (going to a different item, going to a different record, going to a different block, going to a different form) constitute the basic events of your form application.

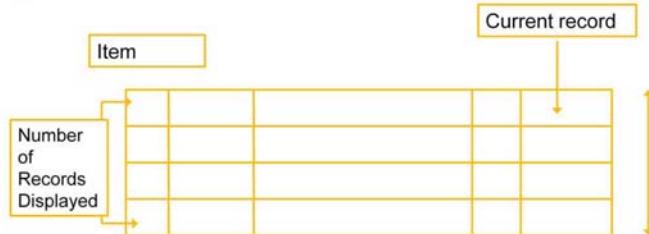
## Database Properties

- Type of Block - Data or Control Block
- Query, insert, update, and delete operations on the data block
- Data Block's Data source
- Query Search criteria and default sort order
- Maximum Query time
- Maximum Number of records fetched

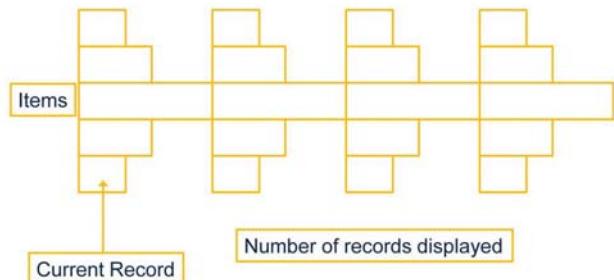


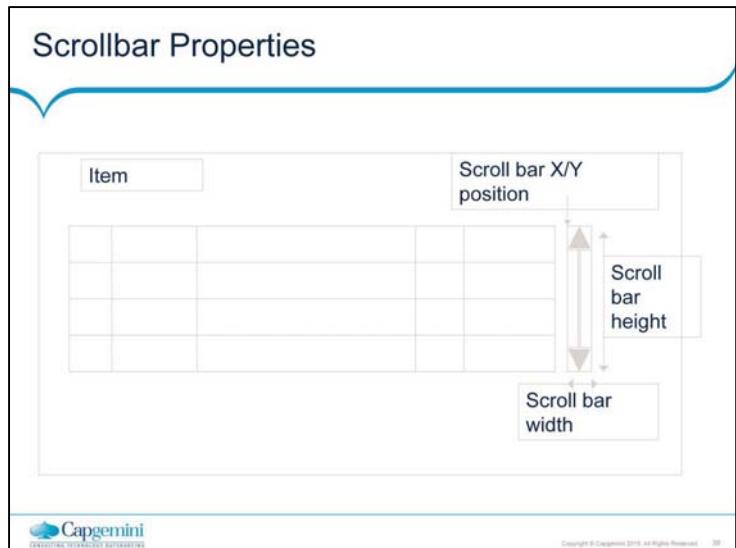
Copyright © Capgemini 2010. All Rights Reserved 30 2

## Vertical Record Orientation



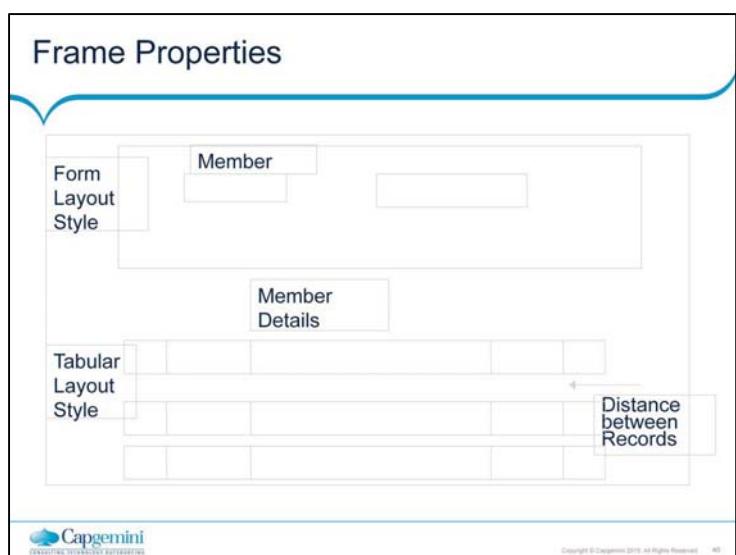
## Horizontal Record Orientation





### Scrollbar Properties

To display a scrollbar adjacent to the frame (both at design-time and runtime), set the Display Scroll bar check box to checked. Scrollbars are typically useful for multirecord.



### Frame

A frame is a graphic object that appears on a canvas. To display the Property Palette for the frame object; double click on the selected frame; or right click on the selected frame to see the popup menu and click on property palette or you can choose Tools -> Property Palette from main menu.

## Creating a Control Block

- Click Data Blocks node
- Click Create icon  
or  
Select Navigator --> Create
- Select the 'Build a new data block manually' option in the New Data Block dialog



Copyright © Capgemini 2010. All Rights Reserved. 41

## Deleting a Data block

- Select data block for deletion
- Click Delete icon  
or  
Navigator --> Delete
- Click Yes in alert box



Copyright © Capgemini 2010. All Rights Reserved A2 2

## Summary

■ In this lesson, you have learnt:

- Understand the features of Developer 2000 Forms
- Basic Design Steps
- Canvas, Data Blocks and Wizards



## Review Questions

- Question 1: With Form Builder you can :
  - 1) Insert, update, delete, query data
  - 2) Present data as text, image etc
  - 3) Control forms across several windows
  - 4) Use customized menus
- Question 2: A \_\_\_\_\_ is associated with a database table or view, or a set of stored procedures.
- Question 3: A \_\_\_\_\_ is associated with a database table or view, or a set of stored procedures. When you build an application in the Forms Builder, a \_\_\_\_\_ file gets created. Compiling a form module compiles all of its code objects and creates a \_\_\_\_\_ run file.



## **ERP- Oracle Apps Forms and Reports**

Lesson 7: Items

## Lesson Objectives

- To understand the following topics:
  - Various Items present on the Forms
  - Input Items
  - Non-Input Items



7.1: Types of Items

## Items

- Basically there are two types of items
  - INPUT ITEMS (text items, list items, push buttons)
  - NON INPUT ITEMS(LOV's , display items etc.)
- Note: The text items are the most often used for the data entry

 Capgemini  
CONSULTING INTEGRATION CONSULTING

Copyright © Capgemini 2010. All Rights Reserved 3 - 1

### Items

An item corresponds to a single data element of field. These items may contain database columns or may be used as containers for other related data.

#### Input items

#### Push Button

A rectangle with a text label or icon graphic inside.

#### Check Box

A text label with a graphic state indicator that displays the current value as either checked or unchecked. Selecting a check box toggles it to the opposite state.

#### Radio Groups

A group of radio buttons, one of which is always selected.

#### List Item

A list of choice displayed as either a Poplist or drop list, a Tlist or list box, and a Combo box. An example of how a list would appear during runtime.

**Text Item**

A single or multi-line text box, that displays a variety of datatypes, format masks, and editing capabilities. They are usually mapped to columns in the database.

**Non Input Items****Display item**

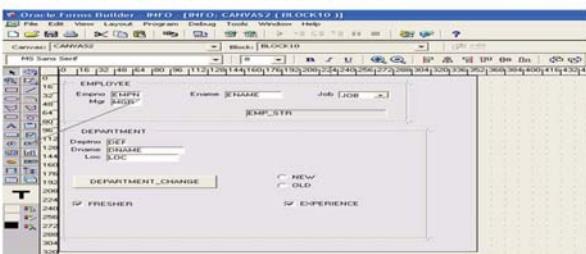
A read-only text box whose value must be fetched or assigned programmatically. Operators cannot navigate to a display item or edit text it contains.

**LOV's (List Of Values)**

Lovs also called as List of Values provide the user with list of valid entries for a field. A list of Values presents data contained within an object called a record group whereby the user will select one value from the list to populate a form item. The list of values may also be used to validate the user input to ensure that a valid value is entered.

## Text Items

- Default item type
- Interface object for
  - Querying, Inserting, Updating, Deleting
- Behavior defined in the property palette



## Modifying the Appearance of a Text Item



- Font and colour properties:
  - Visual attributes
  - Form name, size, weight, style, colour, pattern



## Prompts

- Text label that is associated with an item
- Several properties are available to arrange and manage prompts
- Use prompt properties of an item to change the prompt's display style, justification, alignment, colour and font



Copyright © Capgemini 2010. All Rights Reserved.

7

### Prompt

A prompt is the text label that appears beside and height / width for the items, creating a frame (including style and title) in which to display the items, and choosing the number of records to display in the frame and the distance between those records.

## Format Masks

- Standard SQL formats
  - Dates FXDD-MON-YY
  - Numbers L099G99D99
- Non-standard formats
  - Use double quotes for embedded characters ("099")"099"-0999
  - Allow for the format mask's embedded characters when defining the width property



## Navigational Behaviour of Text Items

- Established by:
  - Order of entries in Object Navigator
- Controlled by:
  - keyboard navigable
  - previous navigation item
  - next navigation item



## Database Properties

- Item's data source -- Base Table item or control item
- Query, Insert and Update operations on an item
- Maximum query length
- Query Case



Copyright © Capgemini 2010. All Rights Reserved

10 / 2

## Functional Properties

- Enabled
- Justification
- Multi-line
- Wrap style
- Case restriction
- Conceal data
- Keep cursor position
- Automatic skip
- Pop-up menu



Copyright © Capgemini 2010. All Rights Reserved

31

## Conceal Data Property

### LOG SCREEN

Enter Userid: scott

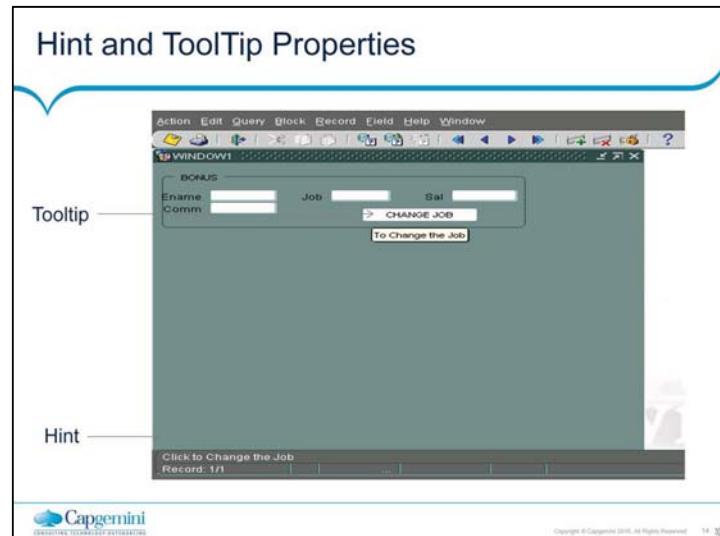
Enter Password: \*\*\*\*\*



Copyright © Capgemini 2010. All Rights Reserved 12 2

## Keyboard Navigable and Enabled Properties

- Set both properties to allow or disallow navigation and interaction with text item
- When enabled is set to YES, keyboard navigable can be set to YES or NO
- When enabled is set to NO, the icon is always non-navigable



## Creating Additional Input Items

- Identify the item types that allow input
- Create a check box
- create a list item
- create a radio group



## Other Input Items

- Item types that accept user input. These include:
  - Check Boxes
  - List Items
  - Radio Groups
- Can Allow:
  - Insert, Update, Delete and Query



## Check Boxes

- Two-State interface Object
  - Checked
  - Unchecked
- Not Limited to two values



Copyright © Capgemini 2010. All Rights Reserved 17

### Check box

A check box is a two-state control indicating whether a certain condition is True or False, On or Off, Checked or Unchecked. When you define a check box, you must specify which value you want the check box to represent as checked and which value you want it to represent as unchecked. Operators toggle between the state of a checkbox. Check boxes store characters, Numbers, and Date values.

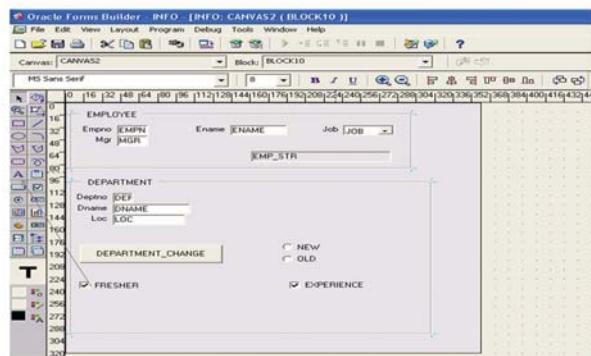
## Creating a Check Box

- Convert an existing item to a check box item
  - Use check Box tool in the layout editor
  - Use create icon in the object navigator
- 
- Check box Properties:
    - Data Type
    - Label
    - Access Key
    - Value when checked
    - Value when unchecked
    - Mapping of other values
    - Mouse Navigate



Copyright © Capgemini 2010. All Rights Reserved 10 |

## Creating a Check Box



Copyright © Capgemini 2010. All Rights Reserved 10

## List Items

- Set of mutually exclusive choices, each representing a different value
- Three list styles available
- Space\_saving alternative to a radio group
- Smaller scale alternative to an LOV

Copyright © Capgemini 2010. All Rights Reserved 20-1

### List Items ( Hard – coded Lists)

A list item is a list of elements of text type that can be displayed as a poplist, text list or a combo box. A list item displays a fixed number of elements maximum of 30 Characters long. Selecting a different element in a list deselects the previously selected elements. Values for every list elements can be populated either at design time or dynamically at runtime.

There are 3 styles for List Items

#### **Pop List**

Pop list appears as a single field. When the operator clicks on the list icon; a drop down list is displayed. User input is not allowed. User has to select from the available values in the list.

#### **Text List**

Text List appears as a rectangular box. It displays fixed values (Scro1lable). Does not allow user input. User has to select from the available values in the list.

#### **Combo List**

Combo List appears as a single field. When the user clicks on the list item, a drop down list is displayed. It allows for user input apart from selection from the list like poplist but allows user input.

## List Item Specific Properties

- Elements in List
  - List elements
  - List item value
- List style
- Mapping of other values
- Mouse Navigate



Copyright © Capgemini 2010. All Rights Reserved 21

### Radio Groups

Radio group is an interface control consisting of a group of buttons of which only one is selected (mutually exclusive). When you create a radio group, you associate a value with each radio button. A radio group can include any number of radio buttons. Operators can select a different button by clicking with the mouse. When you select anyone button from the group, another gets deselected. User can select only one button at runtime. When you create a radio group you associate a specific data value with each radio button in the group. When an operator selects a radio button, the value of the radio group changes to the value that you associated with the selected button. Radio buttons make up the group.

## List Styles

Excellent



Poplist

Excellent



Good

Poor

Tlist

Excellent



Combo Box



Copyright © Capgemini 2010. All Rights Reserved



## Radio Groups

- Set of mutually exclusive radio buttons, each representing a value

- **Use:**

- To display two or more static choices
- As an alternative for a list item
- As an alternative for check box



Copyright © Capgemini 2010. All Rights Reserved 23

## Creating a Radio Group

- Convert an existing Item
- Create a new radio group item in the layout editor
- Use the create icon in the object navigator
- Radio group and radio button properties
  - Radio Group
    - Datatype
    - Mapping of other values
    - Mouse Navigate
  - Radio Button
    - Access Key
    - Label
    - Radio Button Value

Copyright © Capgemini 2010. All Rights Reserved 24

## Creating Non-Input Items

- Identify Item types that do not allow input
- Create a display item
- Create an image item
- Create a button
- Create a calculated Field



Copyright © Capgemini 2010. All Rights Reserved 25.2

## Non-Input Items

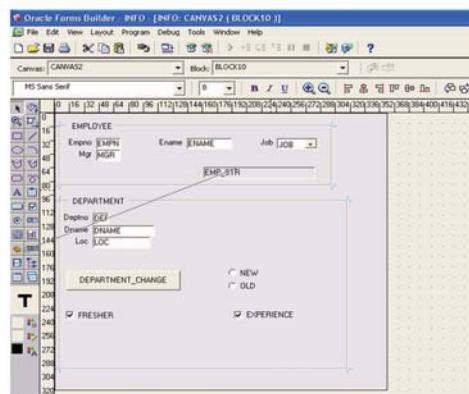
- Item types that do not accept direct user input include:
  - Display Items
  - Image Items
  - Calculated items
  - Buttons

## Display Items

- Display items are similar to text items
  - Display items cannot:
    - Be edited
    - Be queried
    - Be navigated to
    - Accept user input
  - Display Items can:
    - Display date
    - Conserve resources



## Creating a Display Item



## Image Items

- Interface Control
- Used to display bitmapped images
  - From file system-supported file type
  - From database-LONG RAW column



Copyright © Capgemini 2010. All Rights Reserved. 29

### Image items

Image items allow you to integrate images or pictures into your applications. Image item is a bordered rectangle of any size that can be used to store and display images fetched from the database or the file system. They can display images of different formats (.Bmp, .Tiff etc.). Image items can be mapped to columns in the database or read from the file system. Image items are dynamic and can change with records being displayed.

## Image Specific Item Properties

- Image Format
- Image Depth
- Compression Quality
- Display Quality
- Show palette
- Sizing Style
- Show Horizontal Scrollbar
- Show Vertical Scrollbar

## Buttons

- Interface control
- Cannot display/represent data
- Use to initiate an action
- Display as:
  - Text Button
  - Iconic



## Buttons

- Use Buttons to:
  - Move Input Focus
  - Display an LOV
  - Invoke an editor
  - Invoke another window
  - Commit data
  - Issue a query
  - Perform calculations



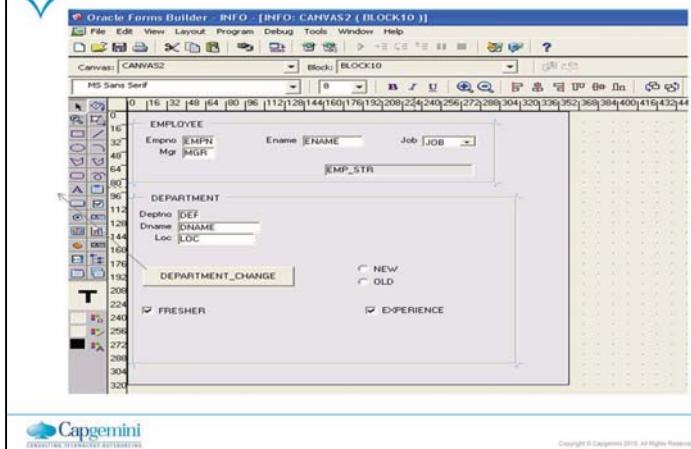
## Button Specific Item Properties

- Label
- Mouse Navigable
- Default Button
- Iconic
- Icon filename
- ToolTip
- ToolTip visual attribute group



Copyright © Capgemini 2010. All Rights Reserved 33 2

## Creating a Button



## Calculated Fields

- Accept item values that are based on calculation
- Are read only
- Can be expressed as:
  - Formula
  - Summary



Copyright © Capgemini 2010. All Rights Reserved 35 2

### Calculated fields

A common task in creating a form is to create an item that represents some calculated value. E.g. You could combine several database items into a concatenated string in a separate item that doesn't correspond to a column in the base table. Another common example is the Total field that sums the record in a block. An amount field calculated as Quantity multiplied by Rate.

## Calculation Modes

- Formula

- Calculated item value is the result of a horizontal calculation
- Involves bind variables

- Summary

- Calculated item's value is a vertical calculation
- Summary performed on values of a single item over all rows in a block



Copyright © Capgemini 2010. All Rights Reserved 30 2

## Item Properties Specific to the Calculated Field

- Formula
  - Calculation Mode
  - Formula
- Summary:
  - Calculation Mode
  - Summary Function (AVG, COUNT, MAX, MIN, STDDEV, SUM, VARIANCE)
  - Summarized block
  - Summarized Item

## Items Based on Formula

NVL(:s\_books.price \* :s\_books.qty),0)

| Books   |           |       |     |      |
|---------|-----------|-------|-----|------|
| Book id | Book name | Price | Cat | Qty  |
| 1       |           | 200   | 5   | 1000 |
| 2       |           | 120   | 4   | 480  |
| 3       |           | 50    | 9   | 450  |
| 4       |           | 25    | 3   | 75   |

Formula Item

### Rules:

- Must not invoke restricted built-ins
- Cannot execute any DML statements .
- Do not terminate PL/SQL expressions with a semicolon
- Do not enter complete PL/SQL statement in assignment expressions



## Items Based on Summary

- Rules for summary items
  - Summary Item must reside in:
    - The same block as the summarized item ,or
    - A control block with single record property set to yes
  - Summarized item must reside in:
    - A data block with query all records property or pre-compute summaries property set to YES, or
    - A control block
  - Data type of summary item must be number unless using MAX or MIN



## Items Based on Summary

Summarised Item

| Books   |           |       |              |       |
|---------|-----------|-------|--------------|-------|
| Book id | Book name | Price | No of copies | Total |
| 1       |           | 200   | 5            | 1000  |
| 2       |           | 120   | 4            | 480   |
| 3       |           | 50    | 9            | 450   |
| 4       |           | 25    | 3            | 75    |

Total Price 2,005

Summary Item



Copyright © Capgemini 2010. All Rights Reserved. 40

7.3: Non-Input Items

## Creating LOV and Editors

- Describe LOVs and Editors
- Design, create, and associate LOVs with text items in a form module
- Create editors and associate them with text items in a form module



Copyright © Capgemini 2010. All Rights Reserved. 41

## LOVs and Editors

### ■ LOVs

- List of Values for text items
- Dynamic list or static list
- Independent of single-text items
- Flexible and efficient

### ■ Editors

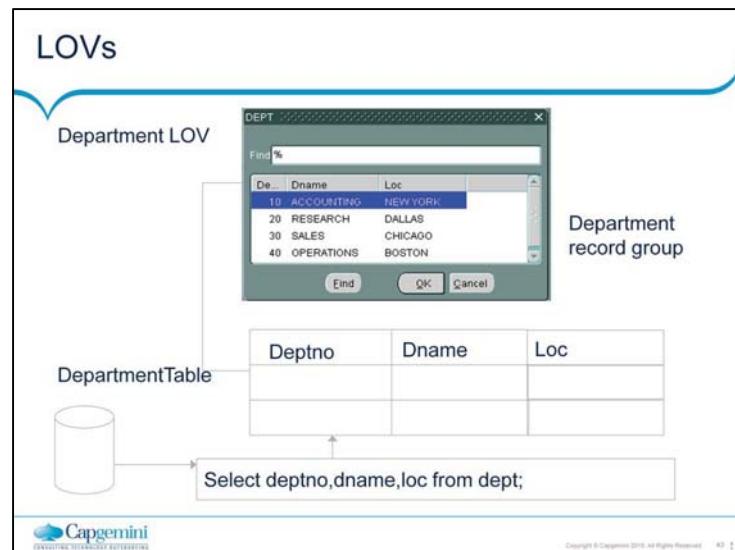
- Override default editor
- Used for special requirements like larger editing window, position, colour, and title
- System editor available as an option

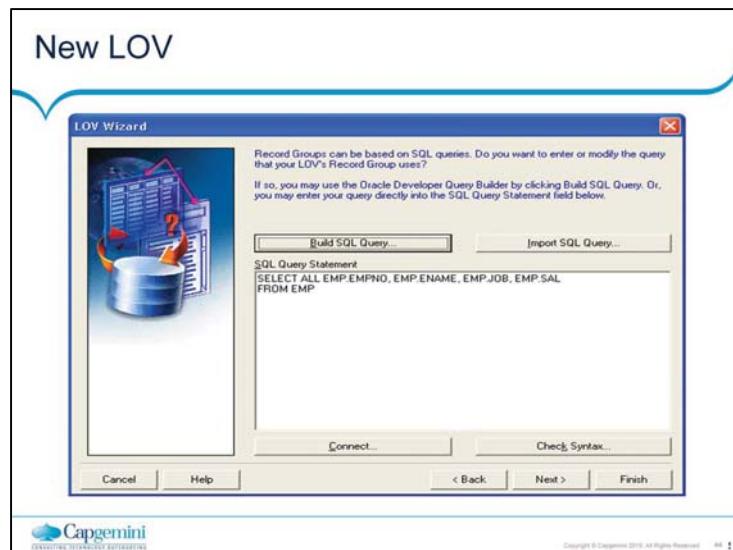


Copyright © Capgemini 2010. All Rights Reserved A2 31

### List of values

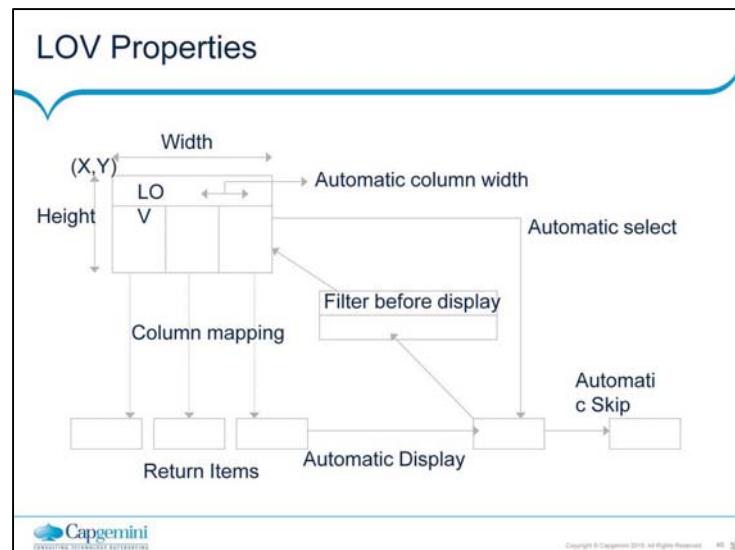
It is a scrollable window (Modal) that provides a single or multi-column list of value it can be displayed at users request or programmatically. LOV's are a special type of single selection lists that can be used with very little effort to display data to the users and allow them to pick the desired element. LOV's are combination of modal dialog boxes, lists and record groups, therefore they have characteristics from each of this objects. The user can select the desired value from the LOV and the value is assigned to the item or the object to which it is attached. It can be attached to one or more text items. Record Groups are internal data structures similar to that of database table when you create an LOV, the record group gets created automatically. A single record group can be attached to a number of LOVs. An LOV can be created at design only.





## LOV Queries

- Avoid very large queries -- use restrictions
- Use column in LOV to validate user input -- place this column first in SELECT list
- Define return items later, or use optional INTO clause
- Use optional WHERE , GROUP BY, and ORDER BY clauses



## Editors

- Defining an Editor -
  - A user defined or default Form Builder editor associated with text items.
- Associate one of three types of editors with a text item. Set text items editor property to one of the following:
  - Null (default form builder editor)
  - editor\_name (customized editor)
  - SYSTEM\_EDITOR (external editor)



Copyright © Capgemini 2010. All Rights Reserved 47

### Editors

An editor is a window for viewing and maintaining large data fields. These data fields are sometimes included for the entry of user comments or other information that would not normally fit in a displayed item. There are occasions when you need to create or maintain long strings of characters in your application. But it becomes difficult for the user to view the entire contents of the field and he has to scroll through it. In such cases he can make use of Editor, Which can also provide basic editing tools such as search and replace, or cut and paste.

#### Types of Editors

- Default
- System
- User named

#### Default Editor

This is Forms internal editor. It provides standard-editing features, is built into every form and is automatically available from every text item.

**System Editor**

If there is a system editor available you can specify that Form Builder use this editor in Windows environment this editor is the Notepad Editor.

**User Editors**

It is an object that you can create from the object navigator. Because it is a named object you can specify Editor attributes such as title, position, size, visual attribute. It can be displayed programmatically with the HOW\_EDITOR built-in. The Show\_Editor builtin can be used not only to specify the location at which you want to display the editor, but the source and destination of text can also be specified. You can invoke it from one or more text items.

## Demo

- Create a Form which contain various types of Items



## Summary

- In this lesson, you have learnt:
  - Various Items present on the Forms
  - Input Items
  - Non-Input Items



## Review Questions

- Question 1: There are two types of items
  - Input Items
  - Accept Items
  - Display Items
  - Non-Input Items
  
- Question 2: Item types that accept user input. These include:
  - Calculated items
  - Check Boxes
  - List Items
  - Radio Groups



## Review Questions

■ Question 3: Item types that do not accept direct user input include:

- Display Items
- Image Items
- Calculated items
- Button



## Review Questions

### ■ Question 4: Match The Following :

|            |                                                                          |
|------------|--------------------------------------------------------------------------|
| 1) Formula | A ) Calculated item value is the result of a horizontal calculation      |
|            | B) Calculated item's value is a vertical calculation                     |
| 2) Summary | C) Involves bind variables                                               |
|            | D) Summary performed on values of a single item over all rows in a block |



## **ERP- Oracle Apps Forms and Reports**

Lesson 8: Windows and  
Canvases

## Lesson Objectives

- To understand the following topics:
  - Windows
  - Types of Canvas
  - Viewports
  - Toolbars



## Creating Windows and Canvases

- Describe Windows and content canvases
- Describe the relationship between Windows and content canvases
- Identify window and content canvas properties
- Display a form Module in multiple windows
- Display a form module on multiple layouts



Copyright © Capgemini 2010. All Rights Reserved 3 - 1

## Creating Windows and Canvases

- Describe the different types of canvases and their relationships to each other
- Identify the appropriate canvas type for different scenarios
- Create an overlay effect using stacked canvases
- Create a toolbar
- Create a tabbed interface



Copyright © Capgemini 2010. All Rights Reserved. 4

## Windows and Canvases

- **Window**

- Container for form builder visual objects

- **Canvas**

- Surface on which you 'paint' visual objects

- To see a canvas and its objects display the canvas in a window



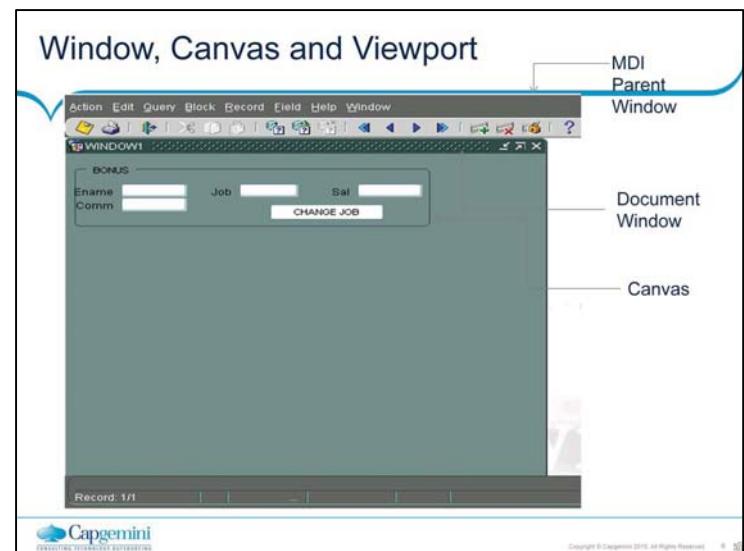
Copyright © Capgemini 2010. All Rights Reserved S-2

### Windows

A window is the frame within which a form appears on the user's screen. Each canvas is assigned to the specific window in the form and several canvases may be assigned to the same window. A single form can contain several windows or it may consist of only one window. The frame provides a title bar and a handle for interacting with the window. It is necessary to have atleast one content canvas on a window. Every new form contains a window named WINDOW1 by default.

### Canvas

- A canvases is a surface -inside a Window Container on which you place the interface items and boilerplate objects that end users interact With when they run the form. By default, any canvas you create at runtime is assigned to the window named WINDOW1. To explicitly associate a canvas to a specific window, set the canvas Window property accordingly. Each item refers to exactly one canvas in its Property Palette. You can divide a data block's items between different canvases.
- A canvas does not stand alone as an interface object. To see it and its items you must display the canvas in a window, a rectangular area of the application display surrounded by a frame. Developer/2000 makes these separate objects and lets you build windows that provide a View of the canvas. A View is a rectangle within the windows that covers all or part of the canvas. The part of the canvas that you can see through the window is the view.



## Content Canvas

- 'Base' canvas
- View occupies entire window
- Default Canvas type
- Each window should have at least one content canvas.



Copyright © Capgemini 2010. All Rights Reserved

7

### Content canvas

A content canvas-view is a base view, which occupies the entire window on which it is placed. There must be at least one content canvas-view for each window. A window can have more than one content canvas-view, but only one will be displayed at a time, during runtime.

## Windows

### ■ WINDOW1

- Created by default with each new form module
- Modeless
- You can delete, rename, or change its attributes

### ■ Use additional Windows to:

- Display two or more content canvases at once
- Switch between canvases without replacing the initial one

### ■ Two type of windows:

- Modal and Modeless



Copyright © Capgemini 2010. All Rights Reserved

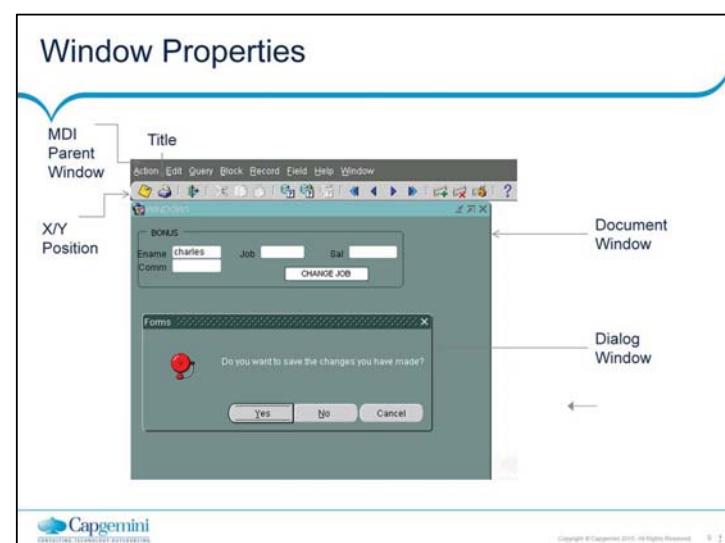
8 - 2

### Modeless Windows

1. More than one modeless window can be displayed at the same time
2. Modeless windows can also be layered to appear either in front of or behind other windows
3. Modeless windows can remain displayed until they are dismissed by the operation or hidden programmatically.

### Modal Windows

1. Modal windows are usually used dialogs
2. Modal windows cannot have scroll bars, and setting the Scroll Bar properties for a modal window has no effect
3. The Remove on Exit property does not apply to modal windows. By default, Form Builder prevents operators from navigating out of modal windows with the mouse, but does not allow them to navigate to another window with keyboard commands. When such navigation occurs, Oracle Forms always closes the modal window, unless the target window is itself a modal window.



## GUI Hints

- Recommendations about window appearance and functionality
- If the Window Manager supports a specific GUI Hint and its property is set to YES, it will be used
- Functional properties for GUI Hints
  - Close allowed
  - Move and Resize allowed
  - Maximize and Minimize allowed
  - Inherit Menu



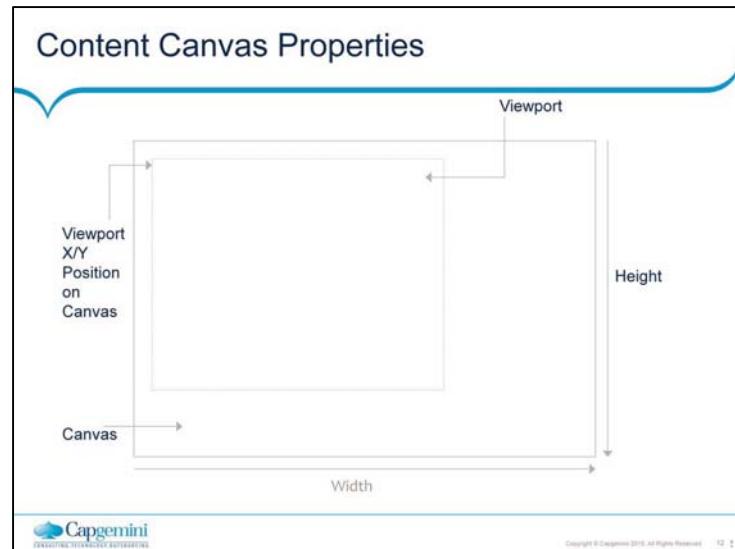
Copyright © Capgemini 2010. All Rights Reserved 10 / 2

## Creating a Content Canvas

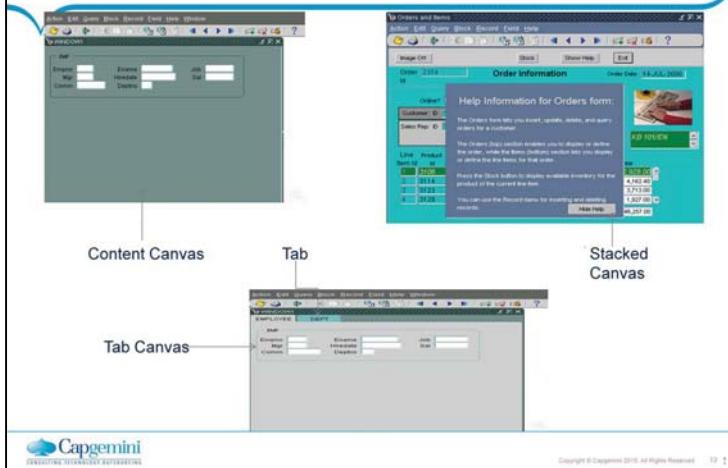
- Implicitly
  - Layout Wizard
  - Layout Editor
- Explicitly
  - Create icon in Object Navigator



Copyright © Capgemini 2010. All Rights Reserved 31 2



## Canvas Types



### There are five types of Canvas-Views

1. Content
2. Tab
3. Stacked
4. Horizontal Toolbar
5. Vertical Toolbar

## Stacked Canvas

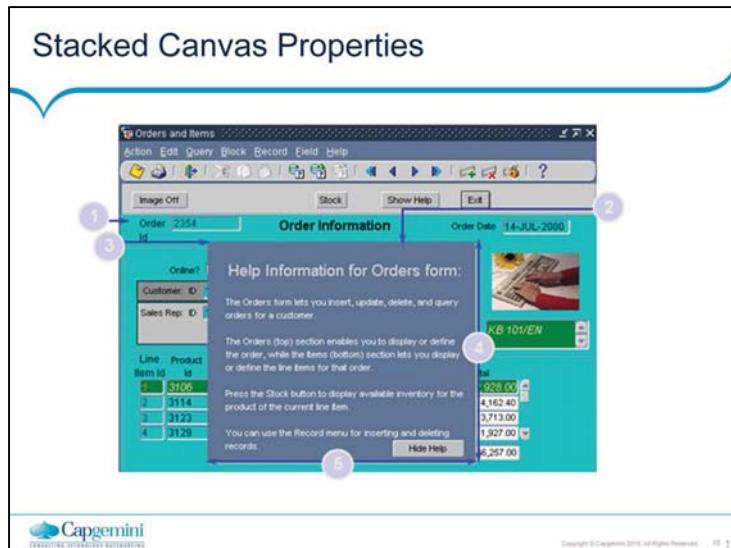
- Displayed on top of a content canvas
- Shares a window with a content canvas
- Size:
  - Usually Smaller than the content canvas in the same window.
  - Determined by viewport size
- Created in:
  - Object navigator
  - Layout editor



Copyright © Capgemini 2010. All Rights Reserved 34

### Stacked Canvas-Views

A canvas you display on top of another canvas, usually containing some items in a group separate from the items in the underlying content canvas. This means that, the content canvas-view lies below the stacked canvas-view and hence, that part of content canvas is not visible to the user at runtime. So stacked canvas-views are more often shown and hidden programmatically. We can display more than one stacked canvas-view in a window at same time.



### The Stacked Canvas (continued)

The stacked canvas displays on the content canvas. You can see the following elements depicted in the slide:

1. Content canvas
2. Stacked canvas
3. Viewport X/Y position
4. Viewport height
5. Viewport width

### Uses and benefits of stacked canvases

With stacked canvases you can achieve the following:

- Scrolling views as generated by Oracle Designer
- Creating an overlay effect within a single window
- Displaying headers with constant information, such as company name
- Creating a cascading or a revealing effect within a single window
- Displaying additional information
- Displaying information conditionally
- Displaying context-sensitive help
- Hiding information

## Toolbars

- Special type of canvas for tool items
- Three types:
  - Vertical toolbar
  - Horizontal toolbar
  - MDI toolbar
- Provide:
  - Standard look and feel
  - Alternative menu or function key operation



Copyright © Capgemini 2010. All Rights Reserved 10 | Page

### Horizontal / Vertical Toolbar Canvas -Views

Toolbar canvas-views are used to create toolbars for individual windows. This can be done by placing iconic control items in the canvas-views. Horizontal toolbars are displayed at the top of the window, just under its menu bar, while the vertical toolbars along the left side of the window.

## Toolbar Related Properties

- Canvas properties
  - Canvas type
  - Window
  - Width
  - Height
- Window properties
  - Horizontal toolbar canvas
  - Vertical Toolbar canvas
- Form Module Properties
  - Form Horizontal toolbar canvas
  - Form Vertical toolbar canvas



Copyright © Capgemini 2010. All Rights Reserved 17 / 2

## Tab canvas

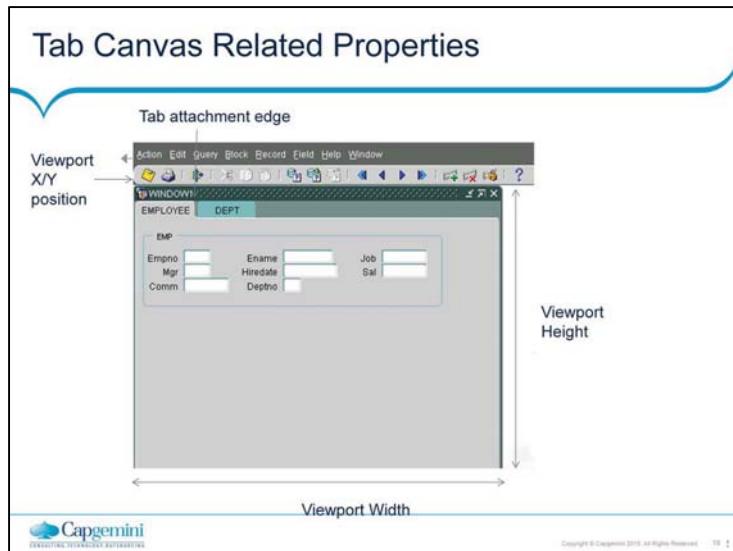
- Enables you to organize and display related information on separate tabs
- Consists of one or more tab pages
- Provides easy access to data
- Created in
  - Object navigator
  - Layout editor
- Items on a tab canvas
  - Place Items on a tab page for user interaction
  - Set item's canvas,tab page properties



Copyright © Capgemini 2010. All Rights Reserved 10 / 2

### Tab Canvas

- A tab canvas is made up of one or more tab pages, which allows you to group and display a large amount of related information on a single canvas object. Like stacked canvases, tab canvases are displayed on top of a content canvas. Tab pages each display a subset of the information displayed on the entire tab canvas.
- The tab canvas is a new type of canvas that enables you to organize and display related information on separate tabs. When the operator clicks the label area of a tab page at runtime, the tab page is brought to the front of the tab canvas, obscuring all other tab pages on the tab canvas.
- Tab pages have properties, and items placed on a tab canvas have tab canvas properties as well as tab page properties. You can define a canvas to have a series of tab pages.



## Demo

- Create a Form using various types of canvas



Copyright © Capgemini 2010. All Rights Reserved. 20

## Summary

- In this lesson, you have learnt:
  - Windows
  - Types of Canvas
  - Viewports
  - Toolbars



## Review Questions

- Question 1: \_\_\_\_\_ can remain displayed until they are dismissed by the operation or hidden programmatically



## **ERP- Oracle Apps Forms and Reports**

Lesson 9: Triggers

## Lesson Objectives

- To understand the following topics:
  - Triggers
  - Query Trigger
  - Validation Trigger
  - Navigation Trigger
  - Transaction Trigger
  - Messages and Alerts Trigger



9.1: Introduction to Triggers

## Form Builder Triggers

- Define Triggers
- Identify the different Trigger types
- Plan the type and scope of triggers in a form
- Describe the properties that affect the behaviour of a trigger
- Write trigger code
- Explain the use of built-in subprograms in D2K applications
- Describe the WHEN\_BUTTON\_PRESSED trigger

 Capgemini  
CONSULTING INTEGRATED SOLUTIONS

Copyright © Capgemini 2010. All Rights Reserved 3 / 1

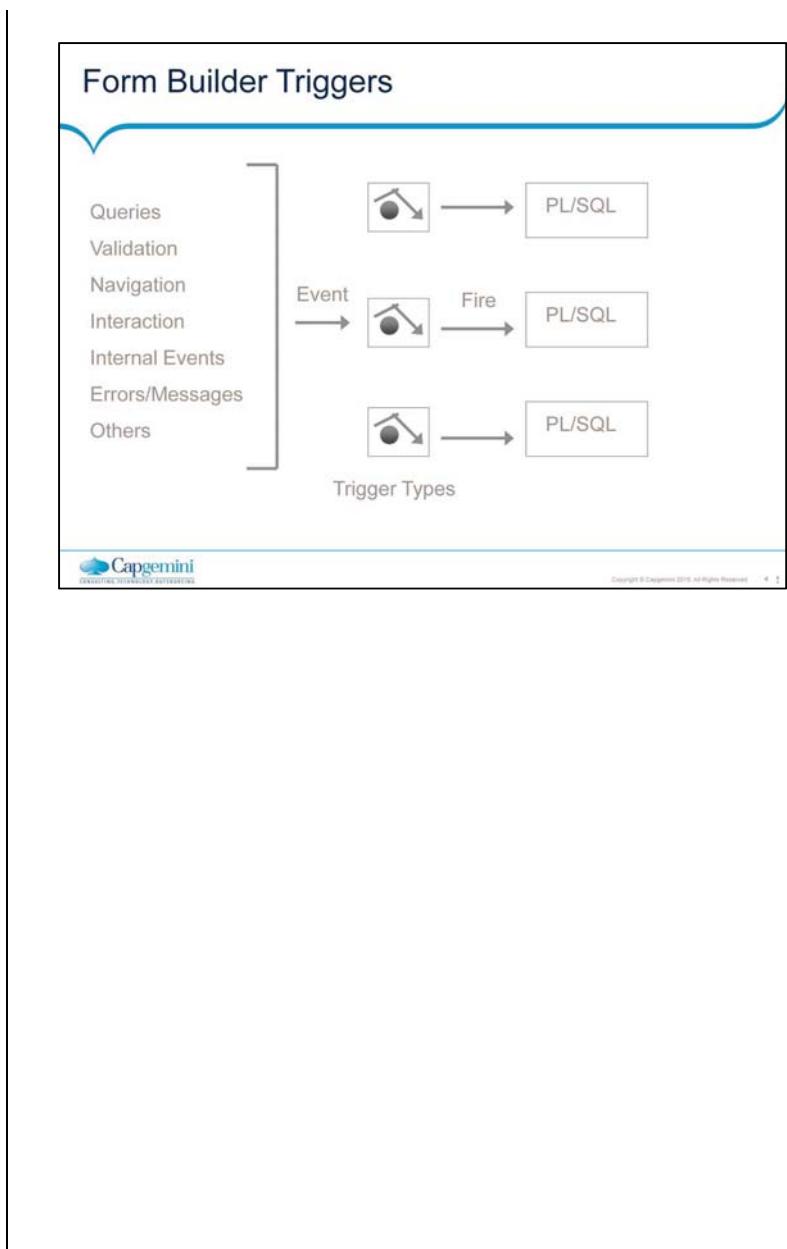
### Triggers

- Triggers are blocks of PL/SOL code you can attach to another object. : a form ,a data block ,or a data block item. But unlike procedures or functions these blocks of code fire or execute in response to Events. So, what are Events? Well, an event can be anything like 'Pressing any Key' or 'Clicking the mouse button' or 'Inserting or saving a record'. Whereas, your trigger is the code, which will be executed in response to the occurring of the event. Thus, we can write triggers to add functionality to a default application.
- Triggers are PL/SQL Code that will be executed based on specific activity or condition in the form. These activities are also called as events. Many of the trigger events will perform a particular default function. A trigger can be written to disable, modify, or enhance the default processing capabilities associated with the event. Triggers can be defined at the Form, Item, Block or Record level.

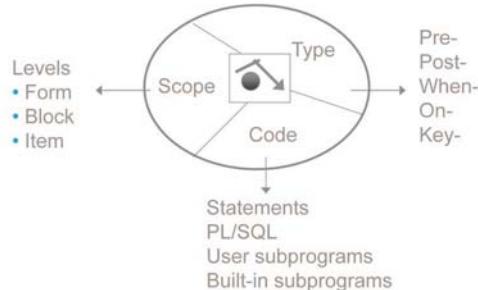
### Triggers associated with push buttons

#### When-Button-Pressed

- Fires when End users select a button by clicking it with the mouse, or select a button by navigating to it and pressing [Select]. The cursor remains in the button unless the button command moves it to a different item.
- Can be defined at form, block or item.



## Scope of a Trigger



## Scope of a Trigger

The screenshot shows a window titled "WINDOW1" with a menu bar and toolbar. Inside the window, there is a form containing fields for Empno, Ename, Job, Mgr, Comm, Hiredate, and Deptno. To the left of the window, three levels of trigger scope are defined:

- Form Level (ON-MESSAGE)
- Item Level (ON-MESSAGE)
- Block Level (ON-MESSAGE)

Below the window, the Capgemini logo is visible.

## Scope of a Trigger

- When the cursor is on the Book\_name item, a message fires the On-Message trigger of the Book\_name item
- When the cursor is elsewhere in the Books block, a message causes the block-level On-Message trigger to fire because it is outside the scope of the item level trigger
- When the cursor is outside the Books block, a message causes the form-level On-Message trigger to fire, because the cursor is outside the scope of the other two On-Message triggers



Copyright © Capgemini 2010. All Rights Reserved.

7 / 1

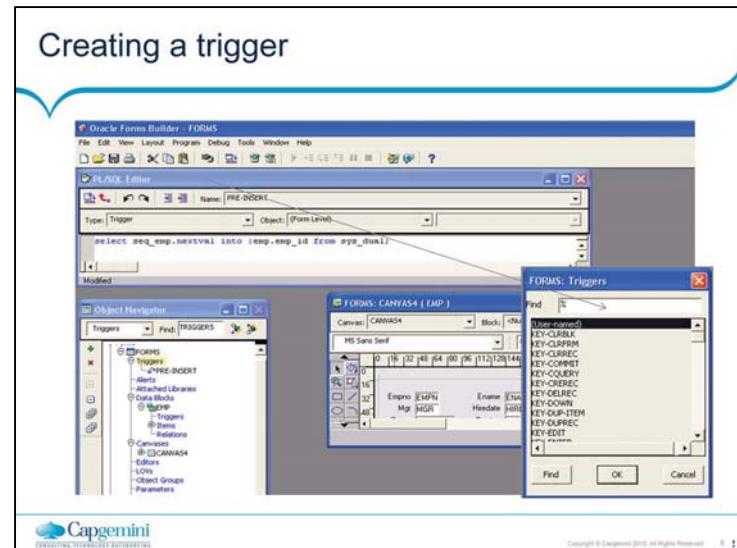
### Triggers

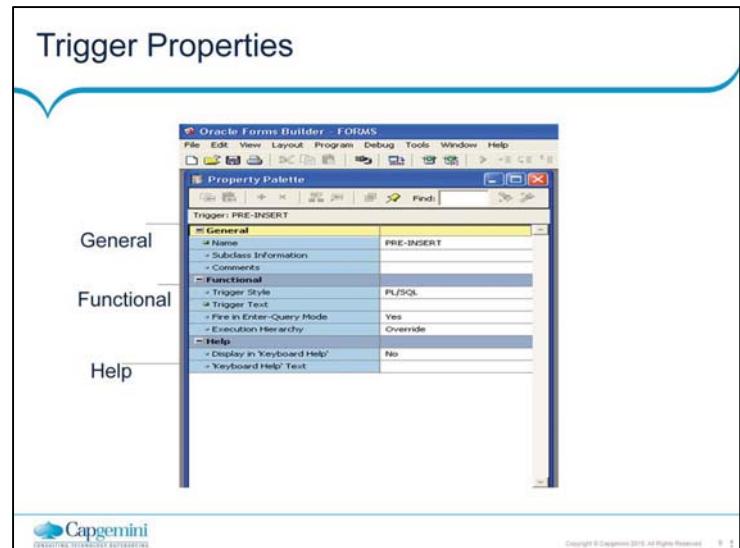
- When you create a trigger, you attach it to a specific object, either an item, a block, or the form itself. The trigger's scope is the set of objects that fire the trigger; it consists of the object that owns that trigger and any object belonging to that object.

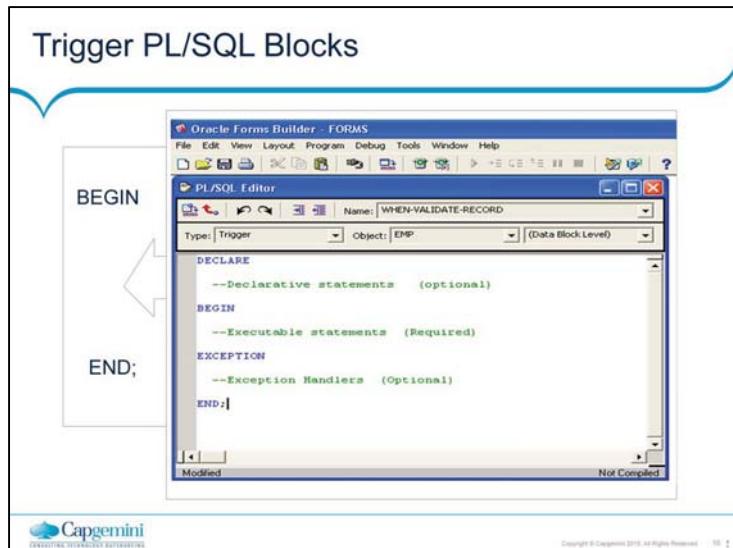
For e.g., if you attach a trigger to a block, events in all the items in that block fire that trigger. Triggers scope is defined at the trigger definition level. Thus, a block level trigger fires within that block, but not in another block.

- If there is more than one trigger with the same name in a particular scope by default developer/2000 fires the one attached to the object lowest in the hierarchy.

For e.g., if you have a When-New-Item-Instance trigger on both an item and that item's block. Developer/2000 fires the trigger on the item and ignores the one on the block . You can change this behaviour for a particular trigger by changing the Execution Hierarchy property of the trigger in the triggers property palette.







## Variables in Form Builder

- PL/SQL variables
  - must be declared in a trigger or defined in a package
- Form Builder variables
  - not formally declared in PL/SQL
  - need a colon prefix in reference



Copyright © Capgemini 2010. All Rights Reserved 31 2

## Form Builder Variables

- Items:
  - For presentation and user interaction
  - :block\_name.item\_name
- Global variables:
  - session wide character variable
  - :GLOBAL.variable\_name
- System variables:
  - form status and control
  - :SYSTEM.variable\_name
- Parameter variables:
  - passing values in and out of a module
  - :PARAMETER.name



Copyright © Capgemini 2010. All Rights Reserved 12 / 35

## Form Builder Built-in Sub-programs

- Built-ins belong to either
  - the standard extensions package
    - no prefix required
  - other Form Builder packages
    - prefix required
- Limits of Use
  - Unrestricted built-ins
    - any trigger or sub-program
  - Restricted built-ins
    - only allowed in certain triggers and sub-programs called from such triggers
  - Consult help system



Copyright © Capgemini 2010. All Rights Reserved 13-1

Restricted Package Procedures are built-in procedures in Oracle Forms that cause navigation to take place.

## Useful Built-ins

- EDIT\_TEXTITEM
- ENTER\_QUERY, EXECUTE\_QUERY
- EXIT\_FORM
- GO\_BLOCK, GO\_ITEM
- GET\_ITEM\_PROPERTY, SET\_ITEM\_PROPERTY
- MESSAGE
- SHOW\_ALERT, SHOW\_EDITOR, SHOW\_LOV
- SHOW\_VIEW, HIDE\_VIEW



Copyright © Capgemini 2010. All Rights Reserved 34

### **EDIT\_TEXTITEM**

Invokes the Runform item editor for the current text item and puts the form in edit mode. The input focus must be in a text item.

### **ENTER\_QUERY;**

Flushes the current block and puts the form in Enter Query mode. If there are changes to commit. Oracle Forms prompts the operator to commit them during the ENTER\_QUERY process.

### **EXIT\_FORM**

EXIT\_FORM (COMMIT\_MODE, ROLLBACK\_MODE);

The behavior of EXIT\_FORM depends on the current mode: in most case it navigates outside the form. If there are changes in current form, Oracle Forms prompts the operator to commit. If operator is in entry mode then it navigates out of the entry mode. During a CALL\_INPUT, EXIT\_FORM terminates the CALL\_INPUT function.

### **GO\_BLOCK**

Navigates to the block in the parameter passed in the function.

### **GO\_ITEM**

Takes the input focus to the indicated item It succeeds even if the target item has the Navigable property off.

**SHOW\_ALERT**

Displays the given alert and returns a numeric value when the operator selects one of three alert buttons.

**GET\_ITEM\_PROPERTY**

Returns information about a specified item. You may be able to get but not set certain object properties.

**SET\_ITEM\_PROPERTY**

Modifies all instance of an item in a block by changing a specified item property. You may be able to get but not set certain object properties.

**SHOW\_EDITOR**

Displays the given editor at the given coordinates and passes a string to the given editor, or retrieves an existing string from the editor. If coordinates are not supplied, the editor is displayed at the default position.

**SHOW\_LOV**

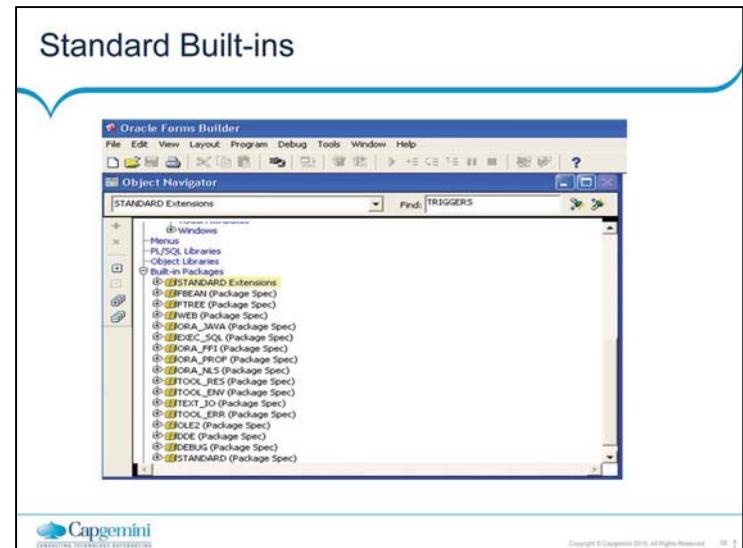
Displays an LOV at the given coordinates. If coordinates are not specified than it displays at default location.

**SHOW\_VIEW**

Displays the given view. If view is already displayed, SHOW\_VIEW raises the view in front of any other views in the same window.

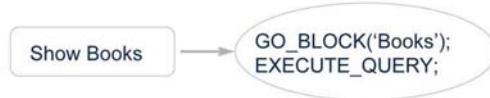
**HIDE\_VIEW**

Hides the given view.



## When-Button-Pressed trigger

- Fires when the operator clicks a button
- Allows restricted and unrestricted built-ins
- Used to provide convenient navigation, to display LOVs and many other frequently used functions
- Example:



### When-Button-Pressed:

Fires when the user selects a push-button, either by a mouse click or keyboard interaction. May be used to initiate an action like commit a form, exit a for, clear a block, perform a query etc.

## Functionality used in Triggers

- Description of the different types of Triggers with some examples
- Different styles of Messages and Alerts and their built-ins
- Runtime errors and built-ins
- A general overview on Query Array Processing
- Enter Query Mode
- Use of SYSTEM variables
- Validation at different levels
- Navigation



Copyright © Capgemini 2010. All Rights Reserved 10 |

9.2: Trigger Example

## Item Interaction Triggers

|                       |
|-----------------------|
| When-Button-Pressed   |
| When-Checkbox-Changed |
| When-Radio-Changed    |
| When-Image-Pressed    |
| When-Image-Activated  |
| When-List-Changed     |
| When-List-Activated   |

 Capgemini  
CONSULTING INTEGRATION CONSULTING

Copyright © Capgemini 2010. All Rights Reserved. 10

**When-CheckBox-Changed**

This trigger fires when the operator changes the state of the Check Box.

**When-Radio-Changed**

Fires when the user selects a different button in the radio group.

**When-List-Changed**

This trigger is used for Poplist and Combo List.

**When-List-Activated**

This trigger is used with only Tlist.

## Coding Item Interaction Triggers

- Valid commands
  - SELECT items
  - Standard PL/SQL constructs
  - All Built-in Subprograms
- Use When-Validate-"Object" to trap the operator during validation



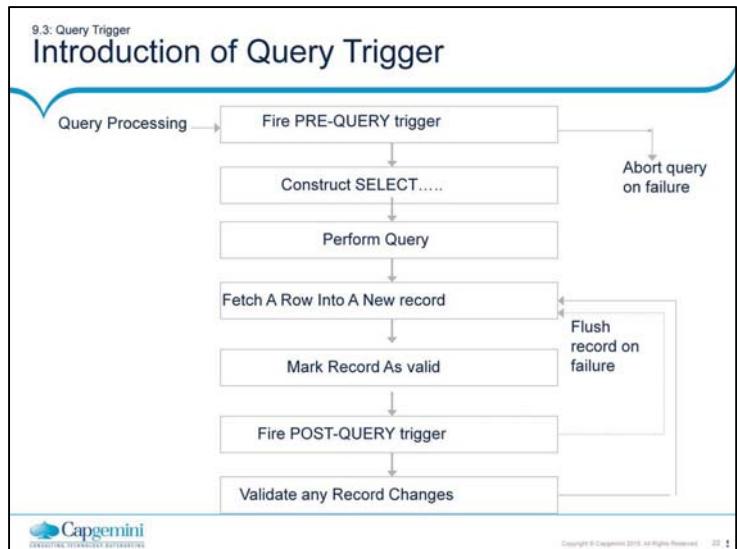
Copyright © Capgemini 2010. All Rights Reserved 20-1

### Example of When-Image Activated

```
BEGIN
READ_IMAGE_FILE
('C:\orant\tools\doc20\guide21\gifs\bookicon.gif'
'BEGIN.IMAGE');
END;
```



Copyright © Capgemini 2010. All Rights Reserved 21



## WHERE Clause

- Three sources for the WHERE Clause
  - Where clause block Property
  - Example Record
  - Query Where dialog
- WHERE Clauses are combined by the AND operator

## ORDER Clause

- Two sources of ORDER Clause
  - ORDER BY clause property
  - Query Where dialog
- Second source for ORDER BY clause overrides the first one



Copyright © Capgemini 2010. All Rights Reserved 24

## Pre\_Query, Post\_Query Triggers

### ■ Pre\_Query

- Define at block level
- Fires once, before query is performed

### ■ Post\_Query

- Fires for each fetched record
- (except during array processing)
- Use to populate non-database items and calculate statistics



Copyright © Capgemini 2010. All Rights Reserved 25 / 2

### Pre-Query

This trigger 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. It may be used to alter the example record that determines which row will be identified by the query. If the trigger fails, the query is cancelled.

### Post-Query

When a query is open in a block, this trigger fires each time Oracle Forms fetches a row into the block. It may be used to populate control items (non-based table items), calculate statistics about records retrieved and calculate running totals.

## Using SELECT Statements in Triggers

- Form Builder variables preceded by colon
- Query must return one row for success
- Code exception handlers
- INTO clause mandatory, with a variable for each selected column or expression
- ORDER BY not relevant

## Coding For ENTER-QUERY Mode

- Some triggers may fire in Enter-Query mode
- Set to Fire in ENTER-QUERY Mode
- Test mode during execution with :SYSTEM.MODE
  - NORMAL
  - ENTER-QUERY
  - QUERY
- Some built-ins are illegal
- Consult online Help
- You cannot navigate to another record in current form



Copyright © Capgemini 2010. All Rights Reserved 27 / 2

## Triggers firing in Enter-Query Mode

- 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



Copyright © Capgemini 2010. All Rights Reserved 28 2

## Overriding Default Query Processing

- On-Fetch continues to fire until:
  - It fires without executing  
CREATE\_QUIERIED\_RECORD
  - The query is closed by the user or by ABORT\_QUERY
  - It raises FORM\_TRIGGER\_FAILURE
- On-Select replaces open cursor,parse and execute phases



## Obtaining Query Information at Runtime

- SYSTEM.MODE to obtain the form mode
- SYSTEM.LAST\_QUERY to obtain the text of the base table SELECT statement that was last executed by Form Builder
  - Contains bind variables (ORD\_ID = :1) before SELECT\_RECORDS
  - Contains actual values (ORD\_ID=102)after SELECT\_RECORDS



Copyright © Capgemini 2010. All Rights Reserved 30 2

### **SYSTEM.MODE**

SYSTEM.MODE Indicates whether the form is in Normal, Enter Query or Fetch Processing mode.

### **SYSTEM.LAST\_QUERY**

SYSTEM.LAST\_QUERY Represents the query SELECT statement that Oracle Forms most recently used to populate a block during the current Runform session.

## Obtaining Query Information at Runtime

- GET\_BLOCK\_PROPERTY
- SET\_BLOCK\_PROPERTY
- Following block properties may be set for obtaining query information
  - Get and Set
    - DEFAULT WHERE
    - ORDER BY
    - QUERY\_ALLOWED
    - QUERY\_HITS
  - Get only
    - QUERY\_OPTIONS
    - RECORD\_TO\_FETCH



## Obtaining Query Information at Runtime

- GET\_ITEM\_PROPERTY
- SET\_ITEM\_PROPERTY
- Following block properties may be set for obtaining query information
  - Get and Set:  
CASE\_INSENSITIVE\_QUERY  
QUERYABLE  
QUERY\_ONLY
  - Get only:  
QUERY\_LENGTH



Copyright © Capgemini 2010. All Rights Reserved 32

### Get\_Item\_Property

#### Syntax:

```
Get_Item_Property(item_id, property);
Get_Item_Property(item_name, property);
```

It is used to get the current value of the given property for the item. The first parameter refers to the item, whereas the second property refers to the property of the item you want to know the property settings for the Required property, you can Get\_Item\_Property ('Empno' . Required).

### Set\_Item\_Property

#### Syntax:

```
Set_Item_Property(item_id, property,value);
Set_Item_Property(item_name, property,value);
```

It is used to set the properties of an item programmatically. The second parameter refers to the property name, which you want to set, whereas the value refers to the value to which you have to set the property to.

9.4: Validation Trigger

## Introduction of Validation Trigger

- Form Builder validates at the following levels:

```
graph TD; FormLevel[Form Level] --> BlockLevel[Block Level]; BlockLevel --> RecordLevel[Record level]; BlockLevel --> ItemLevel1[Item level]; ItemLevel1[Item level]
```

**Capgemini**  
CONSULTING INTEGRATION CONSULTING

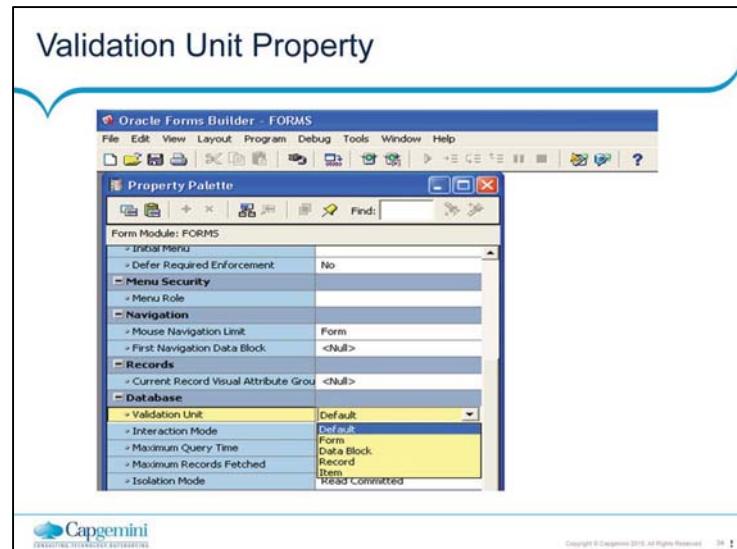
Copyright © Capgemini 2010. All Rights Reserved 33

### Validation

Validation is the process of making sure that an object satisfies all the constraints defined on it. Oracle Forms automatically validates Blocks, Records and Items. You can provide additional functionality by adding constraints as a part of the validation process (through the validation triggers).

### Form and Block Validation

Validating a Block means validating all the records in that block  
Validating a Form means validating all the Blocks in that Form.



## Validation Triggers

- Item level
  - When-Validate-item
- Block level
  - When-validate-record

```
if :s_issues.due_date < sysdate then
 message('enter due date correctly');
 raise FORM_TRIGGER_FAILURE;
End if;
```



Copyright © Capgemini 2010. All Rights Reserved 35 / 2

### **When-Validate-Item**

This trigger fires when the user attempts to navigate out of the item or as a part of the default sequence of posting. It fires for items with status as New or Changed.

### **When-Validate-Record:**

This trigger fires when the user navigates out of the record or as a part of the default sequence of posting. It will fire only for records whose status is Changed.

## Tracking Validation Status

- EW
  - When a record is created
  - Also for Copy Value from item or Initial Value
- CHANGED
  - When the item is changed by the user or a trigger
  - When any item in a new record is changed
- VALID
  - When Validation of the item has been successful
  - After records are fetched from database
  - After successful POST or COMMIT
  - Duplicated records inherit status of source



Copyright © Capgemini 2010. All Rights Reserved 30 / 2

An item can have three states

1. New
2. Changed
3. Valid

An item is created in any of the three ways

**By Creating a new record:**

Creating a record creates the items in it as New. Oracle Forms can fill values in it with a default value or copy it from another item.

**Duplicating a record:**

Duplicating a record copies the values from the previous record into the new record. The state of the items in the new record can be in any of the following states, New, Changed or Valid.

**Fetching a record from the database:**

Fetching the record immediately marks the item in the record as Valid. (If the item status is valid, then validation related triggers would not fire).

- When you modify an item in any way, Oracle Forms marks the item as Changed. All validation related triggers fire for an item whose status is New or Changed. If validation fails for some reason, then the cursor remains on the item itself forcing the entry of a valid value.
- After successful validation of an item depending on its status, Oracle Forms marks the item as Valid, so no future validation is needed if the user navigates out of the item without modifying it once again.

## Built-Ins for Validation

- CLEAR\_BLOCK,CLEAR\_FORM,EXIT\_FORM, EXIT\_FORM
  - ENTER
  - SET\_FORM\_PROPERTY
- (.....,VALIDATION)  
(.....,VALIDATION\_SCOPE)
- ITEM\_IS\_VALID item property
  - VALIDATE (VALIDATE\_SCOPE)



Copyright © Capgemini 2010. All Rights Reserved 37 / 2

## 9.5: Navigation Trigger Introduction of Navigation Trigger

- What is a navigation unit?
  - A Navigation Unit is an invisible, internal object that determines the navigational state of a form. It keeps track of the object that is currently the focus of a navigational process.
- Navigation unit can be an object in the hierarchy:
  - Outside the form
  - Form
  - Block
  - Record
  - Item
- Entering and Leaving Objects
  - What happens if navigation fails



Copyright © Capgemini 2010. All Rights Reserved 38 2

## Navigation Properties

- Form Module
  - Mouse navigation limit
  - First Navigation data block
- Block
  - Navigation style
  - Previous navigation data block
  - Next navigation data block
- Item
  - Enabled
  - Keyboard navigable
  - Mouse navigate
  - Previous navigation item
  - Next navigation item

Copyright © Capgemini 2010. All Rights Reserved 30 / 2

### Navigation

In Developer 2000 Forms, the navigational objects are arranged in a hierarchy of navigational units. The form contains a series of blocks, each of which contains a set of records and a sequence of items. Part of the navigational behaviour of forms is the automatic navigation that occurs because of user actions. For e.g. if the focus is on an item in a block and you click on an item in another block, this action navigates from item to item. Because the items are in different blocks, you also navigate from block to block at the same time. If the blocks each have different records, you navigate from record to record.

## Navigation Triggers

| <i>Pre- and Post-</i>                                          | <i>When New object Instance</i>                                  |
|----------------------------------------------------------------|------------------------------------------------------------------|
| Fire during navigation                                         | Fire after navigation                                            |
| Does not fire if validation unit is larger than trigger object | Does fire when validation unit is larger than the trigger object |
| Allows unrestricted built-ins                                  | Allow restricted and unrestricted built-ins                      |
| Handle failure by returning to initial object                  | Is not affected by failure                                       |



## When-New-'object'-Instance triggers

- When-New-Form-Instance
- When-New-Block-Instance
- When-New-record-Instance
- When-New-Item-Instance



Copyright © Capgemini 2010. All Rights Reserved 41

### **When-New-Form-Instance:**

It fires at form start-up after navigation to the form is complete, Restricted package procedures such as Go\_Item() etc. can be used in this trigger. It can be used to initialise the form, perform query and navigation at start-up.

### **When-New-Block-Instance:**

It fires after navigation to the block is complete. Fires each time a block is visited. It can be used to initialise the block and perform query. It supports restricted package procedures.

### **When-New-Record-Instance:**

It fires after navigation to the record is complete. Fires each time a record is visited. It can be used to initialise the record, set visual attributes. It supports restricted package procedures.

### **When-New-Item-Instance:**

It fires after navigation to the item is complete. Fires each time an item is visited. It can be used to initialise the item, invoke LOV or Editor. It supports restricted package procedures.

## The Pre- and Post-Triggers

- Pre/Post-Form
- Pre/Post-Block
- Pre/Post-Record
- Pre/Post-Text-Item

## Using Pre and Post Triggers

| Trigger Type   | Use to                                                                                                                                        |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| Pre-Form       | Validate - User, Time of day<br>Initialize control blocks –<br>Call another form to display messages                                          |
| Post-Form      | Perform housekeeping - erase global variables<br>Before exit - Display messages to user                                                       |
| Pre-Block      | Authorize access to the block                                                                                                                 |
| Post-Block     | Validate the last record that had input focus<br>Test a condition and prevent the user from leaving                                           |
| Pre-Record     | Set global variables                                                                                                                          |
| Post-Record    | Clear global variables - Set a visual attribute for an item as the user scrolls down through a set of records. Perform cross field validation |
| Pre-Text-Item  | Derive a complex default value.<br>Record the previous value of a text item                                                                   |
| Post-Text-Item | Calculate or change item values                                                                                                               |



## Post-Block Triggers Example

- Changing the status of the book to LOST

```
If (system.block_status =' new' or system.block_status
='changed') and :s_returns.lost =
'Y' then
 s_copy.status='L';
```



Copyright © Capgemini 2010. All Rights Reserved. 44

## Navigation in Triggers

| <i>Built-in routines for Navigation</i> | <i>Function</i>                                                                                             |
|-----------------------------------------|-------------------------------------------------------------------------------------------------------------|
| GO_FORM                                 | Navigates to an open form (in a multiple frm application)                                                   |
| GO_BLOCK                                | Navigates to an indicated block                                                                             |
| GO_ITEM                                 | Navigates to an indicated item                                                                              |
| GO_RECORD                               | Navigates to a specific record                                                                              |
| NEXT_BLOCK                              | Navigates to the next enterable block                                                                       |
| NEXT_ITEM                               | Navigates to the next enterable item                                                                        |
| NEXT_RECORD                             | Navigates to the first enterable item in the next record                                                    |
| NEXT_SET                                | Fetches another set of records from the database and navigates to the first record that the fetch retrieves |
| UP                                      | Navigates to the instance of the current item in the previous record                                        |



Copyright © Capgemini 2010. All Rights Reserved. 45

## Navigation in Triggers

| Built-in routines for Navigation | Function                                                                   |
|----------------------------------|----------------------------------------------------------------------------|
| DOWN                             | Navigates to the instance of the current item in the next record           |
| PREVIOUS_BLOCK                   | Navigates to the previous enterable block                                  |
| PREVIOUS_ITEM                    | Navigates to the previous enterable item                                   |
| PREVIOUS_RECORD                  | Navigates to the previous enterable record                                 |
| SCROLL_UP                        | Scrolls the block so that the records above the top visible one display    |
| SCROLL_DOWN                      | Scrolls the block so that the records below the bottom visible one display |



Copyright © Capgemini 2010. All Rights Reserved. 40

## 9.6: Transaction Trigger Introduction of Transaction Processing/Trigger

- Describe details of commit processing and commit triggers
- Supplement transaction processing by using triggers
- Allocate sequence numbers to records as they are applied to tables
- Implement Array DMLs



Copyright © Capgemini 2010. All Rights Reserved 47 |

### Transaction Processing

A forms transaction is a sequence of events, processes and triggers that ultimately results in either saving data to the database or rolling back changes.

#### The process of Posting to the database

##### Posting

Posting to the database means writing any pending changes in a form to the database through a series of INSERT, UPDATE and delete statements. Oracle Forms generates these statements as a part of default processing. Post does not commit changes to the database. A post is always followed by a commit or rollback as a part of default processing. During the process of Posting, Oracle Forms navigates to the form and validates it. IF there are no changes that require posting, the process stops right here. If there are changes that require posting, then Oracle Forms issues a savepoint.

##### Savepoint

A savepoint is a feature of Oracle that separates database transaction into segments. So that in case you want to rollback a part of your transaction, you can do so without rolling back the entire transaction. This permits Oracle Forms to post multiple times without rolling back all the posts except the one that failed.

##### Commit

After issuing a savepoint, Oracle Forms fires the Pre-Commit trigger, (this is the first trigger to fire as a part of the default processing of Post and Commit). For each block in the Form, Oracle Forms validates the block, then determines whether the block has any changes that affect the database. If the block is a control block, then it has no such changes.

## Transaction Processing

- Transaction processing includes two phases:
  - POST
    - Writes record changes to base tables
    - Fires transactional triggers
  - COMMIT
    - Performs database commit
  - Error result in:
    - Rollback of the database changes
    - Error messages displayed



Copyright © Capgemini 2010. All Rights Reserved 40 2

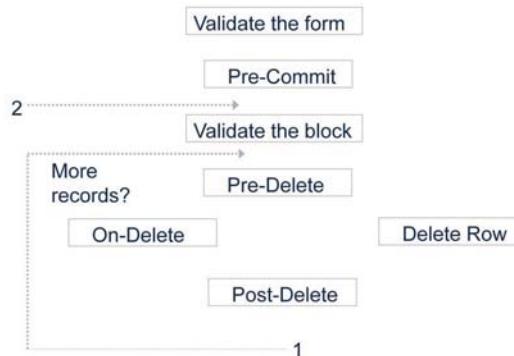
The database operation for posting is done, In the following sequence irrespective of the sequence in which the user perform the inserts, updates and deletes.

- Deletes take place first
- Next Updates and
- Lastly Inserts

Oracle Forms builds statements internally for each row of the block that is deleted, updated or inserted.

- During the Posting process it fires Pre-Event, On-Event and Post-Event triggers for each of the events of Insertion, Updation and Deletion as a part of the default processing.
- The On-Insert trigger calls a package procedure `Insert_Record` that actually builds the `Insert` statement internally and inserts a row into the table.
- The On-Update trigger calls a package procedure `Update_Record`, which actually builds the `Update` statement internally and updates a row in the table.
- The On-Delete trigger calls a package procedure `Delete_Record`, which actually builds the `Delete` statement internally and deletes a row in the table.
- After processing all the blocks in the form, it fires the Post-Forms-Commit trigger marking the end of the Posting process.
- At this point of time all changes are written to the tables, but not committed to the database. Hence a rollback is possible.
- Next, the database commit takes place when Oracle Forms fires the On-Commit which by default calls the `Commit_Form` package procedure.
- After the database commit, the Post-Database-Trigger fires and from here on it is not possible to rollback the changes committed.

## The Commit Sequence of Events

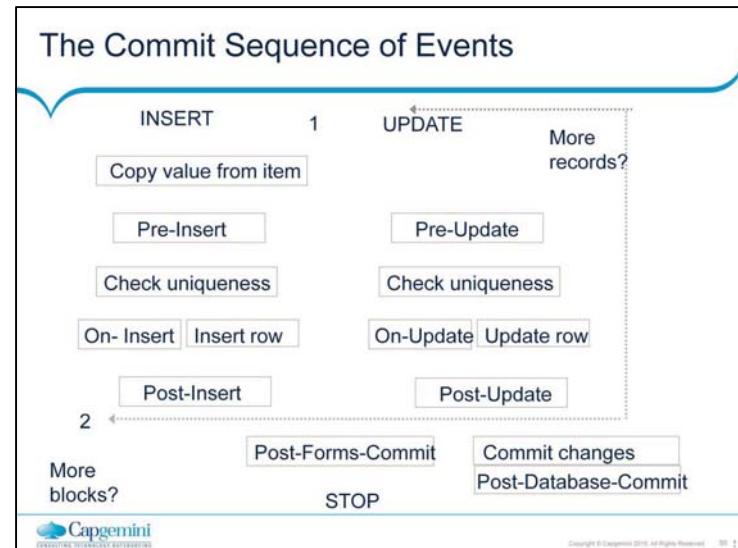


1

Contd.



Copyright © Capgemini 2010. All Rights Reserved. 40 / 2



## Characteristics of Commit triggers

- Pre-Commit: Fires once if form changes are made or uncommitted changes are posted
- Pre/Post-DML: Fires for each record selected for DML operations
- On-DML: Fires per record, replacing default DML on row
  - Use of DELETE\_RECORD , INSERT\_RECORD , UPDATE\_RECORD
- Post-forms-Commits : Fires once even if no changes are made
- Post-Database-Commit : Fires once even if no changes are made
- NOTE : A commit-Trigger failure causes a rollback to the savepoint



Copyright © Capgemini 2010. All Rights Reserved 31 |

### **Pre-Commit:**

This trigger fires once during the post and commit phase as apart of the default processing of posting and committing. When the user initiates a commit, it is the first trigger that marks the start of the posting phase.

### **Pre-Delete:**

This trigger fires during the post and commit process. It typically fires just before the On-Delete trigger. It fires once for every row deleted. It may be used to prevent deletion of the record if certain conditions exist.

### **On-Delete:**

This trigger fires during the post and commit process. Typically, it fires after the Pre-Delete trigger and before Post-Delete-Trigger. It fires for every row deleted.

### **Post-Delete:**

This trigger fires during the post and commit process. Typically, it fires soon after the On-Delete trigger. It fires for each row deleted.

### **Pre-Update:**

This trigger fires during the post and commit process. It fires just before the Update trigger. It fires once for each row updated.

### **On-Update:**

This trigger fires during the post and commit process. It fires once for every row updated.

**Post-Update:**

This trigger fires during the post and commit process. It fires typically soon after the On- Update trigger. It fires once for each row updated.

**Pre-Insert:**

This trigger fires during the post and commit process. It fires just before the On-Insert trigger. It fires for each row inserted.

**On-Insert:**

Fires during the post and commit process. It fires once for each row inserted in the default sequence. May be used to populate derived columns, to assign values to items not displayed in the form.

**Post-Insert:**

This trigger fires during the post and commit process. It fires typically soon after the On- Insert trigger. It fires once for each row inserted. May be used to update an audit trail.

**Post-Forms-Commit:**

This trigger fires after all changed rows are written (posted) to the database. It fires after the form levels post is complete but before the database commit. A rollback is possible from here. It fires before the On- Commit trigger.

**Post-Database-Commit:**

This trigger fires after the On-Commit trigger, after the transaction is finalized and a database commit is done. No rollback is possible from here. May be used to refresh record groups, re-initialize global variables.

## Commit Trigger Uses

|                         |                                                                                            |
|-------------------------|--------------------------------------------------------------------------------------------|
| Pre-Commit              | Check user authorization; set up special locking                                           |
| Pre-Delete              | Journaling; implement foreign-key delete rule                                              |
| Pre-Insert              | Generate sequence numbers; journaling; automatically generated columns; check constraints  |
| Pre-Update              | Journaling; implement foreign – key update rule; auto generated columns; check constraints |
| On-Insert/Update/Delete | Replace default block DML statements                                                       |
| Post-Forms-Commit       | Check complex multi-row constraints                                                        |
| Post-database-commit    | Test commit success test uncommitted posts                                                 |



Copyright © Capgemini 2010. All Rights Reserved 53

## Testing the result of Trigger DML

- SQL%FOUND
- SQL%NOTFOUND
- SQL%ROWCOUNT

- Example:

```
if SQL%NOTFOUND then
 message ('Book not available');
end if;
```



## DML Statements Issued During Commit Processing

- Rules

- DML statements may fire associated database triggers
- Using and retrieving ROWID
- Update Changed Columns and Column Security
- Locking Statements are not issued during default commit processing



Copyright © Capgemini 2016. All Rights Reserved. 35 / 2

## Overriding Default Transaction

- Additional Transactional Triggers

| Trigger            | Do-the-right-thing Built-in |
|--------------------|-----------------------------|
| On-Check-Unique    | CHECK_RECORD_UNIQUENESS     |
| On-Column-Security | ENFORCE_COLUMN_SECURITY     |
| On-Commit          | COMMIT_FORM                 |
| On-Rollback        | ISSUE_ROLLBACK              |
| On-Savepoint       | ISSUE_SAVEPOINT             |
| On-Sequence-Number | GENERATE_SEQUENCE_NUMBER    |



Copyright © Capgemini 2010. All Rights Reserved 38 / 2

### On-Check-Unique:

This trigger fires during a commit operation when Oracle Forms normally checks record that the Primary key values are unique in a base table. It fires once for each visual attributes, record updated or inserted. If a duplicate is found, Oracle forms displays the following message, FRM -40600 Record has already been inserted.

### On-Column-Security:

This trigger fires when the block property 'Column Security' is set to True.

### On-Commit:

This trigger fires when Oracle Forms would normally issue a database commit statement to finalize the transaction. It will fire once after all the Inserted, Updated and Deleted rows are posted in the default sequence.

### On-Sequence-Number:

This trigger fires when Oracle Forms normally perform the default action of generating sequence numbers for the default item values. (When Oracle Form interacts with the database to get the next value from a sequence object).

## Overriding Default Transaction

- Transactional Triggers for logging on and off

| Trigger     | Do-the-right-thing Built-in |
|-------------|-----------------------------|
| Pre-Logon   | -                           |
| Pre-Logout  | -                           |
| On-Logon    | LOGON                       |
| On-Logout   | LOGOUT                      |
| Post-Logon  | -                           |
| Post-Logout | -                           |

Copyright © Capgemini 2010. All Rights Reserved 37.1

### On-Logon

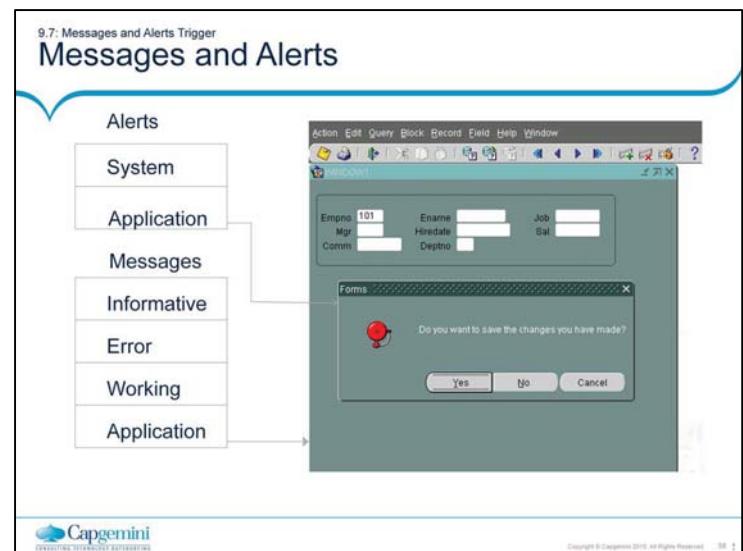
This trigger fires once per login when Oracle forms normally initiates the logon sequence.

Default Processing : LOGON

### On-Logout

This trigger fires once per log out when Oracle forms normally initiates the log out sequence.

Default Processing : LOGOUT



## Detecting Runtime Errors

- **FORM\_SUCCESS**
  - TRUE:Action successful
  - FALSE>Error/Fatal error occurred
- **FORM\_FAILURE**
  - TRUE:A non-fatal error occurred
  - FALSE>No error/No Fatal error
- **FORM\_FATAL**
  - TRUE:A fatal error occurred
  - FALSE>No error/No Fatal error



Copyright © Capgemini 2010. All Rights Reserved 30 / 1

### **FORM\_SUCCESS**

Returns a value that indicates the outcome of the action most recently performed. Use the routine to test outcome of a built-in to determine further processing within any trigger. Returns FALSE on fatal error and failure and TRUE on success.

### **FORM\_FAILURE**

Returns a value that indicates the outcome of the action most recently performed. Use the routine to test outcome of a built-in to determine further processing within any trigger. Returns FALSE on success and fatal error and TRUE on failure.

## Errors and Built-ins

- Built-in failure does NOT cause an exception
- Test built-in success with FORM\_SUCCESS function
  - IF FORM\_SUCCESS THEN ....
- What went wrong?
  - ERROR\_CODE,ERROR\_TEXT,ERROR\_TYPE
  - MESSAGE\_CODE,MESSAGE\_TEXT,MESSAGE\_TYPE



## Message Severity Levels

| Severity Level | Description                                        |
|----------------|----------------------------------------------------|
| 0              | All Messages                                       |
| 5              | Reaffirms an obvious condition                     |
| 10             | Procedural mistake by user                         |
| 15             | Form not designed for action attempted by user     |
| 20             | Cannot continue due to trigger problem             |
| 25             | Condition resulting in form performing incorrectly |
| >25            | Messages that cannot be suppressed                 |

System.MESSAGE\_LEVEL := severity level

System.SUPPRESS\_WORKING := 'TRUE'



Copyright © Capgemini 2010. All Rights Reserved. 81

## Error Triggers

### On-error

- Fires when a system error message is issued
- Used to trap form Builder and Oracle Server errors

### On-Message

- Fires when an informative system message is issued
- Used to suppress or customize specific message
- Built-in functions:
  - MESSAGE\_CODE
  - MESSAGE\_TEXT
  - MESSAGE\_TYPE



Copyright © Capgemini 2010. All Rights Reserved 62

### On-Error Trigger

It fires in response to any error. It can be used to trap errors. You can change the default error message given by Forms and replace it with your own message.

### On-Message Trigger

It fires in response to a default message by Oracle Forms. You can use it to change the default message to the custom message.

### MESSAGE\_CODE

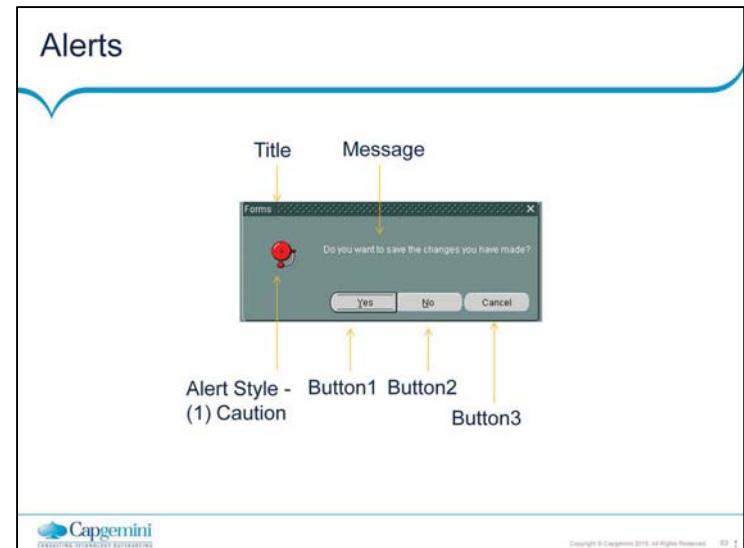
Returns a message number for the message that Oracle Forms most recently generated during the current Runform session.

### MESSAGE\_TEXT

Returns message text for the message that Oracle Forms most recently generated during the current Runform session.

### MESSAGE\_TYPE

Returns a message type for the message that Oracle Forms most recently generated during the current runform session.



### Alerts

Form Builder has many built-in alerts that display pre-defined messages. You can also create your own custom alerts that display in response to application-specific events. Alerts are internal Form Objects that when invoked at runtime are displayed as modal dialog boxes. They are used to notify the user of some application condition. They may be used to warn the user about some action, which might cause some undesirable situation. It is a kind of dialog box that displays a message with an icon. Every alert must have at least one push button and may have up to three buttons.

## Built-ins with Alerts

- SET\_ALERT\_PROPERTY
- SET\_ALERT\_BUTTON\_PROPERTY
- SHOW\_ALERT



Copyright © Capgemini 2010. All Rights Reserved 04

## Summary

- In this lesson, you have learnt:
  - Triggers
  - Query Trigger
  - Validation Trigger
  - Navigation Trigger
  - Transaction Trigger
  - Messages and Alerts Trigger



## Review Questions

- Question 1: Arrange the trigger levels in ascending order

- A) Item
- B) Form
- C) Block



- Question 2: \_\_\_\_\_ trigger gets fired when the user selects a push-button, either by a mouse click or keyboard interaction.

## **ERP- Oracle Apps**

Lesson 10: Flexible Code

## Lesson Objectives

- To understand the following topics:
  - Flexi Code
  - Getting and Setting Object Properties
  - Form Built in



## Flexible Code

- Is reusable code
- Is generic code
- Avoids hard-coded object names
- Makes maintenance easier
- Increases productivity



Copyright © Capgemini 2016. All Rights Reserved. 3

## System Variables for current Context

- Input focus
  - SYSTEM.CURSOR\_BLOCK
  - SYSTEM.CURSOR\_RECORD
  - SYSTEM.CURSOR\_ITEM
  - SYSTEM.CURSOR\_VALUE
- Trigger focus
  - SYSTEM.TRIGGER\_BLOCK
  - SYSTEM.TRIGGER\_RECORD
  - SYSTEM.TRIGGER\_ITEM



Copyright © Capgemini 2016. All Rights Reserved 4

### **SYSTEM.CURSOR\_BLOCK**

Contains the name of the block where cursor is located.

### **SYSTEM.CURSOR\_RECORD**

Represents the number of the recordscurrent physical order in the blocks list of records.

### **SYSTEM.CURRENT\_ITEM**

The value of item depends on the current navigation unit if it is item then the variable represents the name of the item else it contains NULL.

### **SYSTEM.CURSOR\_VALUE**

Represents the value of the item where the cursor is located.

### **SYSTEM.TRIGGER\_BLOCK**

Represents the name of the block where trigger is executed.

### **SYSTEM.TRIGGER\_ITEM**

Represents the name of the item where trigger is executed.

### **SYSTEM.TRIGGER\_RECORD**

Represents the name of the record where trigger is executed.

## System Status Variables

- Writing the trigger code for WHEN-BUTTON-PRESSED on SAVE button

- Example:

```
If :system.block_status='CHANGED' then
 Commit_form;
end if;
```



## GET\_<object>\_PROPERTY

- Built-in subprograms that provide the same type of runtime information as built-in variables
  - GET\_APPLICATION\_PROPERTY
  - GET\_FORM\_PROPERTY
  - GET\_BLOCK\_PROPERTY
  - GET\_RELATION\_PROPERTY
  - GET\_RECORD\_PROPERTY
  - GET\_ITEM\_PROPERTY
  - GET\_ITEM\_INSTANCE\_PROPERTY



Copyright © Capgemini 2016. All Rights Reserved. 6 / 2

**Get\_Item\_Property**

Syntax:

Get\_Item\_Property(item\_id, property);

Get\_Item\_Property(item\_name, property);

It is used to get the current value of the given property for the item. The first parameter refers to the item, whereas the second property refers to the property of the item you want to know the property settings for the Required property, you can Get\_Item\_Property('Empno' . Required).

**GET\_APPLICATION\_PROPERTY**

GET\_APPLICATION\_PROPERTY (PROPERTY);

The GET\_APPLICATION\_PROPERTY built-in returns information about the current Oracle Forms application.

**GET\_FORM\_PROPERTY**

GET\_FORM\_PROPERTY (FORM, PROPERTY);

Returns information about given form. It returns element according to information supplied in the property parameter.

**GET\_BLOCK\_PROPERTY**

GET\_BLOCK\_PROPERTY (BLOCK\_NAME, PROPERTY);

Returns information about given block. It returns element according to information supplied in the property parameter.

**GET\_RECORD\_PROPERTY**

GET\_RECORD\_PROPERTY (RECORD\_NUMBER,  
BLOCK\_NAME, PROPERTY);

Returns the value for the given property for the given record number in the given block. The three parameters are required. If you do not pass the proper constants, Oracle Forms issued an error.

**GET\_RELATION\_PROPERTY**

GET\_RELATION\_PROPERTY (RELATION\_ID, PROPERTY);

GET\_RELATION\_PROPERTY (RELATION\_NAME,  
PROPERTY);

Returns the state of the given relation property.

## GET\_<object>\_PROPERTY

- GET\_LOV\_PROPERTY
- GET\_RADIO\_BUTTON\_PROPERTY
- GET\_MENU\_ITEM\_PROPERTY
- GET\_CANVAS\_PROPERTY
- GET\_TAB\_PAGE\_PROPERTY
- GET\_VIEW\_PROPERTY
- GET\_WINDOW\_PROPERTY



Copyright © Capgemini 2010. All Rights Reserved 8 / 1

### **GET\_LOV\_PROPERTY**

GET\_LOV\_PROPERTY (LOV\_NAME, PROPERTY);

Returns information about a specified LOV.

### **GET\_RADIO\_BUTTON\_PROPERTY**

GET\_RADIO\_BUTTON\_PROPERTY (RADIO\_BUTTON, PROPERTY);

Returns information about a specified radio button.

### **GET\_MENU\_ITEM\_PROPERTY**

GET\_MENU\_ITEM\_PROPERTY (MENUITEM\_ID, PROPERTY);

GET\_MENU\_ITEM\_PROPERTY (MENU\_NAME.MENUITEM\_NAME, PROPERTY);

### **GET\_CANVAS\_PROPERTY**

GET\_CANVAS\_PROPERTY (CANVAS, PROPERTY);

Returns information about given canvas. It returns element according to information supplied in the property parameter.

### **GET\_VIEW\_PROPERTY**

GET\_VIEW\_PROPERTY (VIEW\_ID, PROPERTY);

GET\_VIEW\_PROPERTY (VIEW\_NAME, PROPERTY);

Returns the indicated view property for the indicated view.

### **GET\_WINDOW\_PROPERTY**

GET\_WINDOW\_PROPERTY (WINDOW\_ID, PROPERTY);

Returns information about given window. It returns element according to information supplied in the property parameter.

## SET\_“object”\_PROPERTY

- SET\_APPLICATION\_PROPERTY
- SET\_FORM\_PROPERTY
- SET\_BLOCK\_PROPERTY
- SET\_REALTION\_PROPERTY
- SET\_RECORD\_PROPERTY
- SET\_ITEM\_PROPERTY
- SET\_ITEM\_INSTANCE\_PROPERTY



Copyright © Capgemini 2010. All Rights Reserved.

### Set\_Item\_Property

#### Syntax

Set\_Item\_Property(item\_id, property,value);

Set\_Item\_Property(item\_name, property,value);

It is used to set the properties of an item programmatically. The second parameter refers to the property name, which you want to set, whereas the value refers to the value to which you have to set the property to.

#### SET\_FORM\_PROPERTY

SET\_FORM\_PROPERTY (FORM, PROPERTY, VALUE);

Sets a property of the given form.

#### SET\_BLOCK\_PROPERTY

SET\_BLOCK\_PROPERTY (BLOCK, PROPERTY, VALUE);

Sets the given block characteristics of the block.

#### SET\_RECORD\_PROPERTY

SET\_RECORD\_PROPERTY (RECORD\_NUMBER, BLOCK\_NAME, PROPERTY, VALUE);

Sets the specified record property to the specified value.

#### SET\_RELATION\_PROPERTY

SET\_RELATION\_PROPERTY (RELATION, PROPERTY, VALUE);

Sets the given relation property in a master-detail relationship.

## SET\_“object”\_PROPERTY

- SET\_LOV\_PROPERTY
- SET\_RADIO\_BUTTON\_PROPERTY
- SET\_MENU\_ITEM\_PROPERTY
- SET\_CANVASPROPERTY
- SET\_TAB\_PAGE\_PROPERTY
- SET\_VIEW\_PROPERTY
- SET\_WINDOW\_PROPERTY



Copyright © Capgemini 2016. All Rights Reserved 10

SET\_LOV\_PROPERTY  
SET\_LOV\_PROPERTY (LOV, PROPERTY, VALUE);  
SET\_LOV\_PROPERTY (LOV, PROPERTY, X, Y);  
Sets the property for a LOV.

SET\_RADIO\_BUTTON\_PROPERTY  
SET\_RADIO\_BUTTON\_PROPERTY (ITEM,  
BUTTON\_NAMR, PROPERTY, VALUE);  
SET\_RADIO\_BUTTON\_PROPERTY (ITEM,  
BUTTON\_NAMR, PROPERTY, X, Y);  
Sets the property for a radio button that is part of the given  
radio group specified by the item\_name.

SET\_MENU\_ITEM\_PROPERTY  
SET\_MENU\_ITEM\_PROPERTY (MENUITEM\_ID,  
PROPERTY, VALUE);  
SET\_MENU\_ITEM\_PROPERTY  
(MENU\_NAME.MENUITEM\_NAME, PROPERTY, VALUE);  
Modifies the given properties of a menu item.

SET\_CANVAS\_PROPERTY  
SET\_CANVAS\_PROPERTY (CANVAS, PROPERTY,  
VALUE);  
SET\_CANVAS\_PROPERTY (CANVAS, PROPERTY, X);  
SET\_CANVAS\_PROPERTY (CANVAS, PROPERTY, X, Y);  
Sets the given canvas characteristics of the canvas.

SET\_VIEW\_PROPERTY  
SET\_VIEW\_PROPERTY (VIEW, PROPERTY, VALUE);  
SET\_VIEW\_PROPERTY (VIEW, PROPERTY, X);  
SET\_VIEW\_PROPERTY (VIEW, PROPERTY, X, Y);  
Sets the given view characteristics of the view.

SET\_WINDOW\_PROPERTY  
SET\_WINDOW\_PROPERTY (WINDOW, PROPERTY,  
VALUE);  
SET\_WINDOW\_PROPERTY (WINDOW, PROPERTY, X);  
SET\_WINDOW\_PROPERTY (WINDOW, PROPERTY, X,  
Y);  
Sets the given window characteristics of the window.

## FIND\_Built-Ins

- The FIND\_ "object" built-in returns the id of the object being referenced by it.
  - FIND\_FORM
  - FIND\_BLOCK
  - FIND\_ITEM
  - FIND\_RELATION
  - FIND\_LOV
  - FIND\_WINDOW
  - FIND\_EDITOR
  - FIND\_VIEW
  - FIND\_CANVAS
  - FIND\_ALERT



## Using Object IDs

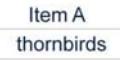
- Declare a PL/SQL variable of the same datatype
- Use a variable for any later reference to the object
- Use a variable within the current PL/SQL block only

- Example:

```
DECLARE
 item_var item;
BEGIN
 item_var := FIND_ITEM(:SYSTEM>CURSOR_ITEM);
 SET_ITEM_PROPERTY(item_var.position,30,35);
 SET_ITEM_PROPERTY(item_var.prompt_text,'Current');
END;
```



## Referencing Objects Indirectly



## Referencing Objects Indirectly

### ■ NAME\_IN Function

- Returns
  - The contents of the variable
  - Character string

### ▪ USE

- Conversion functions for NUMBER and DATE

### ■ COPY Procedure

- Allows

#### ▪ Direct Copy

- eg: COPY('thornbirds','S\_BOOKS.book\_name');
- eg: COPY('thornbirds',:GLOBAL.book\_name);

#### ▪ Indirect Copy

- eg: COPY('thornbirds',NAME\_IN('GLOBAL.customer\_name'));
- Assigns a value to an item whose name is stored in a global variable



## Summary

- In this lesson, you have learnt:
  - Flexi Code
  - Getting and Setting Object Properties
  - Form Built in



## **ERP- Oracle Apps Forms and Reports**

Lesson 11: Sharing Code

## Lesson Objectives

■ To understand the following topics:

- Describing various methods for reusing objects and code
- Inheriting properties from property classes
- Grouping related objects for reuse
- Reusing objects from an object library
- Reusing PL/SQL code
- Calling one form from another form module



## Sharing Code

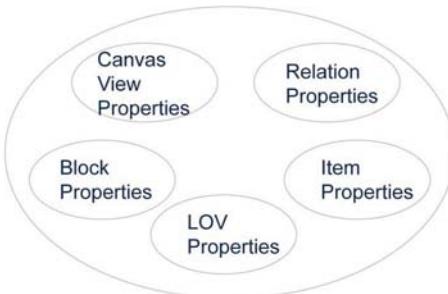
- Increases Productivity
- Decreases Maintenance
- Increases Modularity
- Maintains Standards



Copyright © Capgemini 2010. All Rights Reserved 3 - 1

## Property Classes

- A property class is a named object that contains a list of properties and their settings



Copyright © Capgemini 2010. All Rights Reserved

4

### Property Classes

- A property class is a named object belonging to a module that contains a set of properties and their settings. An object based on Property Class can inherit the setting of any property in the class that is valid for that object. When you make changes to the property class the changes are reflected across all the objects that have inherited that class.
- Thus we can say that Form Builder possesses Object Oriented features like inheritance Property class Inheritance is a powerful feature that allows you to quickly define objects that conform to your interface and functionality standards.

## Property Class Icons



Add Property



Delete Property



Property Class

Copyright © Capgemini 2010. All Rights Reserved S-2

## Inherited and Variant Properties

### ■ Inherited Property

- An inherited property is one that takes its value from the property class that you associated with the object.
- An inherited property displays with an arrow to the left of the property name

### ■ Variant Property

- A variant property is one that has a modified value although it is inherited from the property class associated with the object. You can override the setting of any inherited property to make that property variant.
- Variant properties display with a red cross over an arrow



Copyright © Capgemini 2010. All Rights Reserved. 6 / 31

### Inheritance

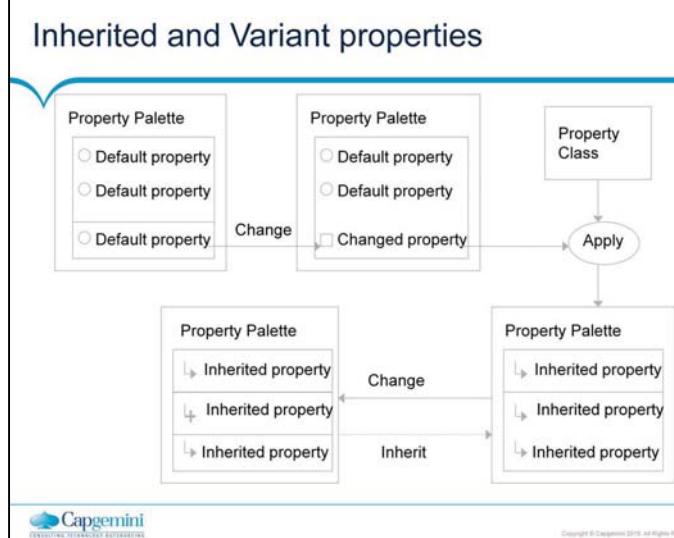
Inheritance is a relationship between objects such that the child object has all the property settings of the parent object plus whatever additional ones that makes it a different. A sign besides the property indicates that it has been inherited. After inheritance the properties of the object can be either in an Inherited state or in a Variant state.

### Inherited Property

- An inherited property is one that takes its value from the property class that you associated with the object.
- An inherited property displays with an arrow to the left of the property name.

### Variant Property

- A variant property is one that has a modified value although it is inherited from the property class associated with the object. You can override the setting of any inherited property to make that property variant.
- Variant properties display with a red cross over an arrow



## Inheriting Properties

- To inherit property values from a property class, set the Subclass Information Property
- To convert a Variant Property to an Inherited Property, change Inherited Property in the Property Palette to Variant Property
- To convert an Inherited Property to a Variant Property, overtype the Inherited value with a new value



Copyright © Capgemini 2010. All Rights Reserved

8 - 2

## Object Group

- An object group is a Logical Container for a set of Form Builder objects
- Enables you to
  - Group related objects
  - Copy multiple objects in one operation



Copyright © Capgemini 2016. All Rights Reserved

9 / 2

### Object Group

An object group is a container for a group of objects. Related objects are packed together in an object group, which can then be copied or referenced in another module. Thus, object groups enable us to implement reusability.

### For example

All the data entry forms have to be provided with common facilities like adding, deleting, saving a record etc. Instead of creating these buttons for these activities and writing various triggers on them, we can create push buttons for these activities in a separate canvas/block and then include this in the object group. This object group can then be referenced across the forms. Thus, with the help of object groups, we can reuse the block as well as the contents of the block (items, triggers) and maintain consistency.

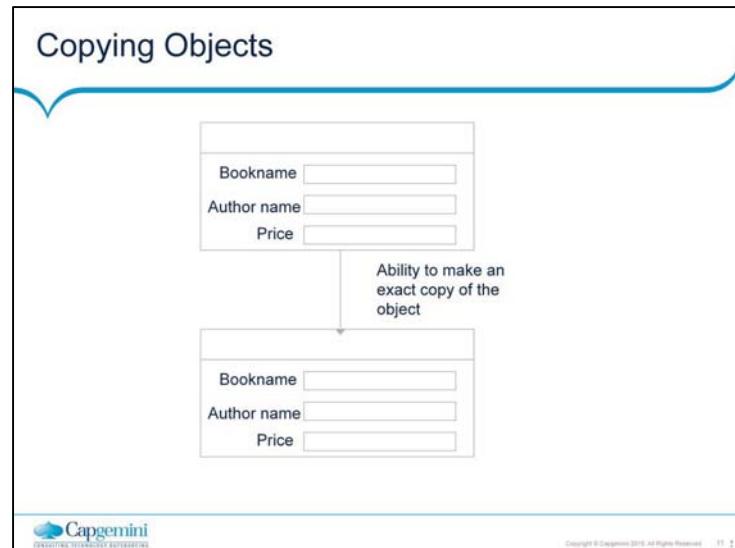
## Using Object Groups

- Including a block into an Object Group includes it's
    - Items
    - Item-level triggers
    - Block-level triggers
    - Relations
  - Other Object Groups cannot be included into an object group
- Deleting an
- Object Group does not affect objects in the form module
  - Object from a form module affects the object groups containing it

Copyright © Capgemini 2010. All Rights Reserved 10 / 2

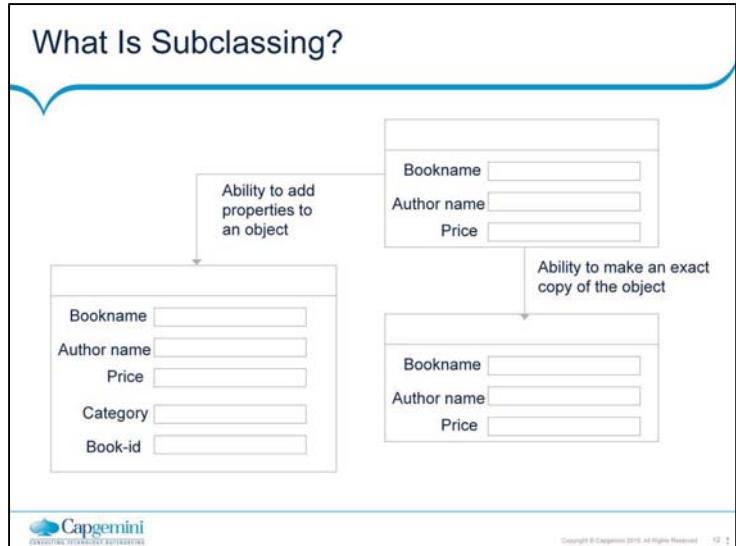
### Object Groups

- Program units cannot be included in an object group.
- The objects in a group must all be defined in the same module. You cannot place objects from two different forms in the same object group.
- An object group cannot contain another object group.
- When an object in an object group is deleted from a module, it is removed from the object group automatically.
- Deleting an object group from a module does not delete the objects it contains from the module.
- Object groups do not store copies of the objects they contain, but rather pointers to the objects. The size of the module does not increase substantially, even though object groups might be defined in it.



### Copying Objects

- One of the simplest ways to reuse an object is to copy it. When you copy an object, you create a new and separate instance of the object, and any objects, owned by the object being copied, are always copied automatically.
- For example, when you copy a block, the new block includes any items that were owned by the block and any triggers that were attached to the block or its items.
- When a block is dragged into an object group, all objects that are owned by the block (items, triggers, relations) are also included in that object group. But objects owned by blocks cannot be placed in an object group independently of the block in which they are defined. To include these objects in an object group, you must include the block itself.

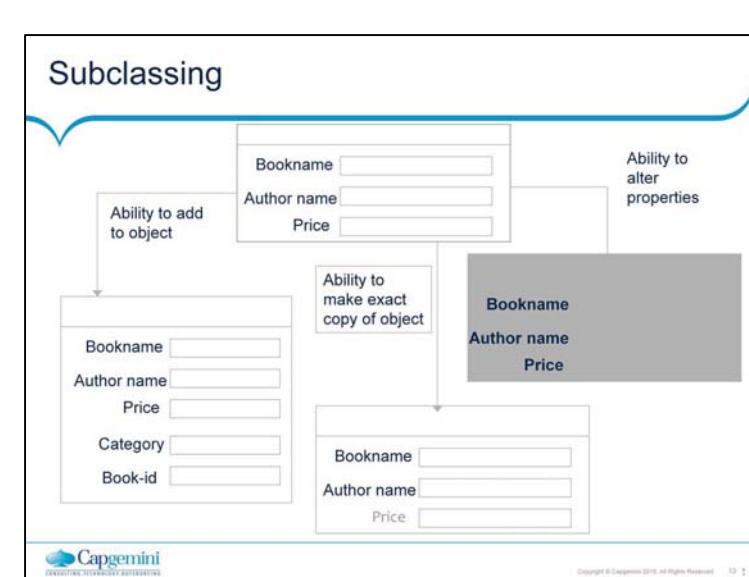


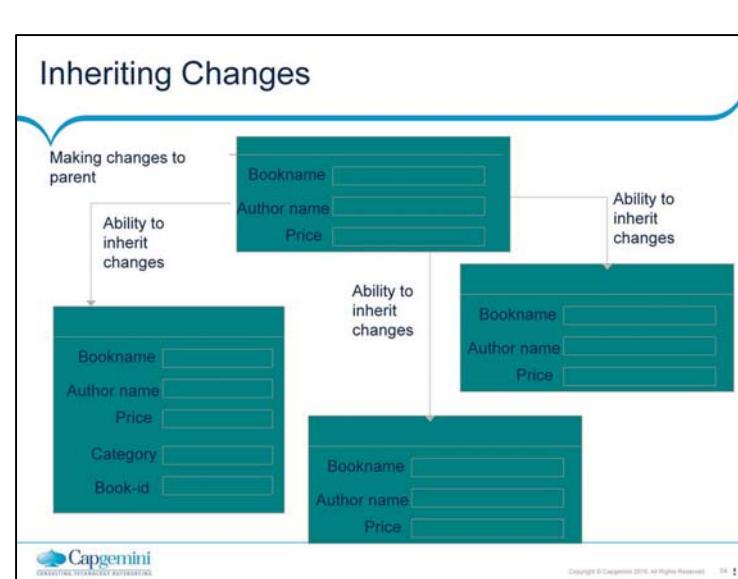
### Subclassing

Subclassing allows you to:

- Create an independent subclass of any object.
- Automatically propagate any changes made to the source object to each subclassed object.
- Override a subclassed object's properties and structure.

When you subclass an object, you create an object that inherits its functionality and appearance from another object. Subclassing and the resulting subclassed object maintains a link to its source object. A subclassed object automatically inherits any changes that have been made to the source object when you open or recompile the module that contains the subclassed object unless both modules are open. If both modules are open, changes are inherited immediately.





## What Is an Object Library?

- Convenient container of objects for reuse
- Simplifies reuse in complex environment
- Supports and provides controls for corporate, project, and personal standards
- Simplifies sharing of reusable components
- Eliminates the need to maintain multiple referenced forms
- Saved to '.olb' file or to the database



Copyright © Capgemini 2010. All Rights Reserved 10-2

### Object libraries

- Object libraries store form objects like blocks, items, alerts, editors etc. These objects can be dragged into the library and reused into other modules. We can store standard objects in the Object Libraries and use them through out the application to give it a uniform look and feel.
- The Object Library provides an easy method of reusing objects and enforcing standards.
- With the help of Object Libraries we can make use of powerful, object-oriented features to improve productivity through object reuse we can rapidly create applications by dragging and dropping predefined objects to your form.
- You can create template forms, reports, or charts for enforcing standards and rapidly develop applications. Whenever a new document is created using a template, all the objects in the template (e.g. standard toolbars, standard layout, and standard libraries etc.) are automatically included.

## SmartClass

- An object in an Object library that is frequently used as a Class is a Smart Class
  - Can be applied easily and rapidly to existing objects
  - Can be defined in many object libraries
  - Can have many SmartClasses of the given object type



Copyright © Capgemini 2010. All Rights Reserved 10 2

### Smart Class

Smart Classes are the objects that you define as being standard. They can be used to convert objects to standard objects. In other words, we can say that Object libraries can be used to specify the classes that are frequently used as the basis of other objects in forms. If you mark an object in the library as standard class, whenever developers create a new object of that type, they can quickly pick from a context sensitive list of standard classes known as Smart Classes.

## PL/SQL Libraries

- A library is a separate module, holding procedures, functions and packages
- Direct reference to bind variables are not allowed
- Use sub-program parameters for passing bind variables
- Use functions, where appropriate, to return values



Copyright © Capgemini 2010. All Rights Reserved 17 / 2

### PL/SQL libraries

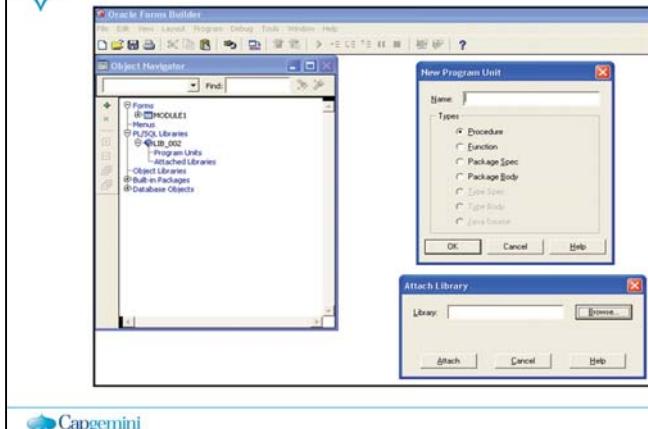
- A PL/SQL library is a collection of PL/SQL program units, including user-named procedures, functions and packages. The program units here can be used in triggers and procedures written in Form Builder if the library is attached to the form. Same library can be attached to multiple forms and menus. On the other hand, a single form or menu can have more than one library attached to it. Libraries are loaded into memory at the same time as the module to which they are attached.
  - There are three PL/SQL library file formats, .PLL, .PLX, and .PLD
1. Library modules being separate module are compiled independently of forms and menus. As a result, you cannot refer directly to form bind variables (the values of form items, system and global variables, and parameters) in a library program unit i.e. you cannot refer directly to the following variables:
    1. :block\_name.item\_name
    2. :GLOBAL.variable\_name
    3. :SYSTEM.variable\_name
    4. :PARAMETER.parameter\_name

Hence, we have to use the NAME\_IN built-in to refer to the values of bind variables.

You can say Name\_In('block\_name.item\_name') to get the name of the item.

2. To set the values of bind variables we have to use the COPY built-in. When COPY is used to set the value of a form item in a library routine, that item is marked as CHANGED. This behavior can affect subsequent validation and commit processing.
3. Whenever possible, you should write library program units that rely on IN and IN OUT parameters for passing data between the form or menu module and the library. You can also use functions with return values for this purpose. When you call a library procedure that uses parameters, you can pass bind variable references as actual parameters.

## Creating Library Program Units and attaching Libraries



## Multiple Form Applications

- Multi Form Applications
- New Chapter



Copyright © Capgemini 2010. All Rights Reserved 20 1

### Multiple-form application

- A multiple-form application is one that is designed to open more than one form during a single Runform session. When the runform session is invoked to run a form, Form Builder looks for the form in the appropriate directory and then loads it into memory form, which is being executed. These forms can, in turn, invoke still other forms. When one form programmatically invokes another, Form Builder looks for the new form in the appropriate directory and then loads it into memory.
- When invoking a new module from an existing form, you may need to pass values from the current environment to the new one. One way could be global variables, global Variables are visible across all modules. If you need to initialize some values upon new form's startup, you can pass these values using parameter lists.

## Multiple Form Applications

### ■ Behaviour

- Flexible navigation between windows
- Single or multiple database connections
- Transactions may span forms, if required
- Commits in order of opening forms, starting with current form

### ■ Links

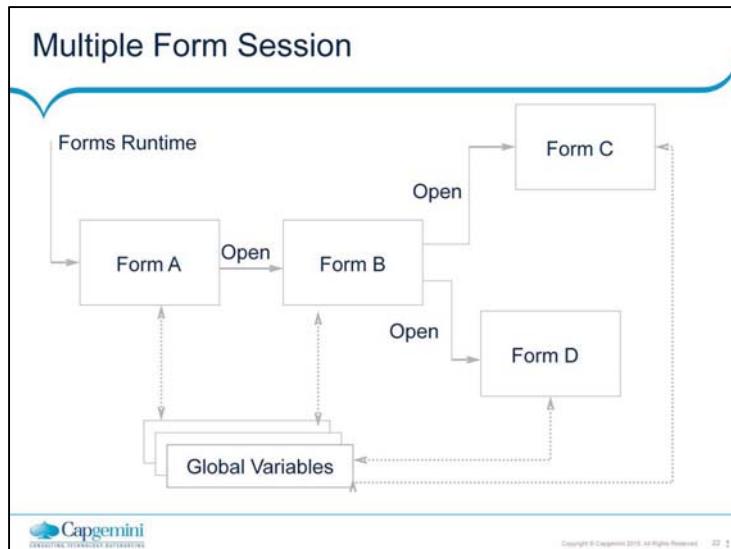
- Data exchanged by global variables or parameter lists
- Code shared as required, through libraries and databases



Copyright © Capgemini 2010. All Rights Reserved 21

### Multiple-form application

- A multiple-form application is one that is designed to open more than one form during a single Runform session. When the runform session is invoked to run a form, Form Builder looks for the form in the appropriate directory and then loads it into memory form, which is being executed. These forms can, in turn, invoke still other forms. When one form programmatically invokes another, Form Builder looks for the new form in the appropriate directory and then loads it into memory.
- When invoking a new module from an existing form, you may need to pass values from the current environment to the new one. One way could be global variables, global Variables are visible across all modules. If you need to initialize some values upon new form's startup, you can pass these values using parameter lists.



## Opening Another Form

### Notes:

- Control passes immediately to called form - no statements after open\_form are processed
- Activate mode NO\_ACTIVATE retains control in current form
- Transaction continues unless explicitly committed before



Copyright © Capgemini 2010. All Rights Reserved 23 / 1

There are three built-ins that can be used to programmatically invoke another form in a multi-form application:

1. Open\_Form
2. Call\_Form
3. New\_Form

### Open\_Form

- This is a procedure to open an independent form. When one form invokes another form by executing OPEN\_FORM, operators can navigate between the forms as desired. An opened form can share the same database session as the form from which it is invoked, or it can create a separate session of its own.
- Calling Open-Form leaves the current form accessible but replaces the current menu with the called forms menu.
- If you use Open\_Form, you can open the form in a new database session, and hence can initiate a different transaction, by using the SESSION Parameter. Default is to Open the form in the same session as the calling form. Opening separate forms in separate sessions keeps your transaction separate so that you can deal with them independently. On the other hand, if the forms use the same tables, you can get lock conflicts when both transactions try to access the same tables at once.

Separate sessions are most useful when the forms do not share tables.

NO\_SESSION opens the form in the current session.

#### Activate Parameter

- When you specify the ACTIVATE parameter, Open\_Form gives the called form the focus and ignores any statement following the Open\_Form call in the calling PL/SQL block. When you specify NO\_ACTIVATE parameter, Open\_Form keeps the focus on the calling form and continues executing the statements following the Open\_Form call.
- The Call\_Form and New\_Form subprograms issue an ORACLE savepoint, Open\_Form does not. A savepoint is a point in the transaction to which you may roll\_back without rolling back the entire transaction. Creating a savepoint when you move to a different form, gives you more control over what happens when the called form closes. If there is a rollback and there is a savepoint, ORACLE only rolls the changes back to the point before you called the new form, preserving all the changes you made and posted.

## Opening Another Form

- Example:

```
If book available open ISSUE form, else open DEMAND form
if book_status='A' then
 open_form('ISSUE');
Else
 open_form('DEMAND');
end if;
```



Copyright © Capgemini 2010. All Rights Reserved 25 |

## Closing a Form with EXIT\_FORM

- Default functionality same as [Exit] key
- Commit\_mode argument defines action on uncommitted changes
- Example:
  - EXIT\_FORM(DO\_COMMIT);
  - EXIT\_FORM(NO\_COMMIT);



Copyright © Capgemini 2010. All Rights Reserved 20 2

There are three ways to close a form:

Close\_Form : Close the Specified form equivalent to Exit\_Form

Exit\_Form : Close the current form corresponds to the Key-Exit trigger

Clear\_Form : Rolls back and resets all forms in memory. It is equivalent to Action-> Clear on default menu; corresponds to KeyClrFrm trigger.

## Explanation

- The other exceptions can be given user-defined names.
- Exceptions can be defined in the declarative part of any PL/SQL block, subprogram, or package. These are user-defined exceptions.



Copyright © Capgemini 2016. All Rights Reserved 27

## Summary

■ In this lesson, you have learnt:

- Describing various methods for reusing objects and code
- Inheriting properties from property classes
- Grouping related objects for reuse
- Reusing objects from an object library
- Reusing PL/SQL code
- Calling one form from another form module



## Review Questions

- Question 1: An \_\_\_\_\_ property is one that takes its value from the property class that you associated with the object.
- Question 2: An \_\_\_\_\_ is a Logical Container for a set of Form Builder objects.



# D2K / Reports

Lab Book

## Document Revision History

| Date        | Revision No. | Author    | Summary of Changes |
|-------------|--------------|-----------|--------------------|
| 11-Aug-2011 | 1.0          | Amit Sali | Content Creation   |
|             |              |           |                    |
|             |              |           |                    |
|             |              |           |                    |

## Table of Contents

|                                                             |    |
|-------------------------------------------------------------|----|
| <i>Document Revision History</i> .....                      | 2  |
| <i>Table of Contents</i> .....                              | 3  |
| <i>Getting Started</i> .....                                | 4  |
| <i>Overview</i> .....                                       | 4  |
| <i>Setup Checklist for EAS Oracle Application</i> .....     | 4  |
| <i>Instructions</i> .....                                   | 4  |
| <i>Problem Statement / Case Study</i> .....                 | 5  |
| <i>Lab 1. Wizard based report creation</i> .....            | 8  |
| <i>1.1: Create a tabular report</i> .....                   | 8  |
| <i>Lab 2. Group Report</i> .....                            | 22 |
| <i>2.1: Creating group report</i> .....                     | 22 |
| <i>Lab 3. BIND AND LEXICAL PARAMETERS</i> .....             | 26 |
| <i>3.1: Bind parameter with LOVs</i> .....                  | 26 |
| <i>Solution:</i> .....                                      | 34 |
| <i>Lab 4. Formula and Summary</i> .....                     | 39 |
| <i>4.1: Formula</i> .....                                   | 39 |
| <i>Lab 5. Triggers</i> .....                                | 46 |
| <i>5.1: Report with after parameter form trigger:</i> ..... | 46 |
| <i>Appendices</i> .....                                     | 55 |
| <i>Appendix A: Table of Figures</i> .....                   | 55 |

## Getting Started

### Overview

This lab book is a guided tour for learning EAS oracle application. It comprises 'To Do' assignments. Follow the steps provided and work out the 'To Do' assignments.

### Setup Checklist for EAS Oracle Application

Here is what is expected on your machine in order for the lab to work.

#### Minimum System Requirements

- Intel Pentium 90 or higher (P166 recommended)
- Microsoft Windows 95, 98, or NT 4.0, 2k, XP.
- Memory: 32MB of RAM (64MB or more recommended)
- Internet Explorer 6.0 or higher

### Instructions

- For all coding standards refer Appendix A. All lab assignments should refer coding standards.
- Create a directory by your name in drive <drive>. In this directory, create a subdirectory <eas orapps>\_assgn. For each lab exercise create a directory as lab <lab number>.

## Problem Statement / Case Study

Table descriptions and data considered in this lab book:

### EMP Table

SQL>DESC EMP

| Name     | Null?    | Type          |
|----------|----------|---------------|
| EMPNO    | NOT NULL | NUMBER(4)     |
| ENAME    |          | VARCHAR2(10 ) |
| JOB      |          | VARCHAR2(50 ) |
| MGR      |          | NUMBER(4)     |
| HIREDATE |          | DATE          |
| SAL      |          | NUMBER(7,2)   |
| COMM     |          | NUMBER(7,2)   |
| DEPTNO   |          | NUMBER(2)     |

SQL>SELECT \* FROM EMP

| EMPNO | ENAME  | JOB       | MGR  | HIREDATE  | SAL  | COMM | DEPTNO |
|-------|--------|-----------|------|-----------|------|------|--------|
| 7369  | SMITH  | CLERK     | 7902 | 17-DEC-80 | 800  |      | 20     |
| 7499  | ALLEN  | SALESMAN  | 7698 | 20-FEB-81 | 1600 | 300  | 30     |
| 7521  | WARD   | SALESMAN  | 7698 | 22-FEB-81 | 1250 | 500  | 30     |
| 7566  | JONES  | MANAGER   | 7839 | 02-APR-81 | 2975 |      | 20     |
| 7654  | MARTIN | SALESMAN  | 7698 | 28-SEP-81 | 1250 | 1400 | 30     |
| 7698  | BLAKE  | MANAGER   | 7839 | 01-MAY-81 | 2850 |      | 30     |
| 7782  | CLARK  | MANAGER   | 7839 | 09-JUN-81 | 2450 |      | 10     |
| 7788  | SCOTT  | ANALYST   | 7566 | 09-DEC-82 | 3000 |      | 20     |
| 7839  | KING   | PRESIDENT |      | 17-NOV-81 | 5000 |      | 10     |
| 7844  | TURNER | SALESMAN  | 7698 | 08-SEP-81 | 1500 | 0    | 30     |
| 7876  | ADAMS  | CLERK     | 7788 | 12-JAN-83 | 1100 |      | 20     |
| 7900  | JAMES  | CLERK     | 7698 | 03-DEC-81 | 950  |      | 30     |

|      |        |         |      |           |      |  |    |
|------|--------|---------|------|-----------|------|--|----|
| 7902 | FORD   | ANALYST | 7566 | 03-DEC-81 | 3000 |  | 20 |
| 7934 | MILLER | CLERK   | 7782 | 23-JAN-82 | 1300 |  | 10 |

14 rows selected.

**DEPT Table**  
SQL>DESC DEPT

| Name   | Null? | Type         |
|--------|-------|--------------|
| DEPTNO |       | NUMBER(2)    |
| DNAME  |       | VARCHAR2(14) |
| LOC    |       | VARCHAR2(13) |

SQL>SELECT \* FROM DEPT

| DEPTNO | DNAME      | LOC      |
|--------|------------|----------|
| 10     | ACCOUNTING | NEW YORK |
| 20     | RESEARCH   | DALLAS   |
| 30     | SALES      | CHICAGO  |
| 40     | OPERATIONS | BOSTON   |

**SALGRADE Table**

SQL&gt;DESC SALGRADE

| Name  | Null? | Type   |
|-------|-------|--------|
| GRADE |       | NUMBER |
| LOSAL |       | NUMBER |
| HISAL |       | NUMBER |

SQL&gt;SELECT \* FROM SALGRADE

| GRADE | LOSAL | HISAL |
|-------|-------|-------|
| 1     | 700   | 1200  |
| 2     | 1201  | 1400  |
| 3     | 1401  | 2000  |
| 4     | 2001  | 3000  |
| 5     | 3001  | 9999  |

**BONUS Table**

SQL&gt;DESC BONUS

| Name  | Null? | Type         |
|-------|-------|--------------|
| ENAME |       | VARCHAR2(10) |
| JOB   |       | VARCHAR2(9)  |
| SAL   |       | NUMBER       |
| COMM  |       | NUMBER       |

SQL&gt;SELECT \* FROM BONUS

no rows selected

## Lab 1. Wizard based report creation

|       |                                                                                 |
|-------|---------------------------------------------------------------------------------|
| Goals | <ul style="list-style-type: none"><li>• Create a report using wizard.</li></ul> |
| Time  |                                                                                 |

### 1.1: Create a tabular report

The steps in this lab will walk you through the wizard for creating report.

You will have to do the following tasks:

1. Create a report using wizard
2. Saving the report
3. Executing the report

#### Solution:

**Step 1: Start Report Builder** Open reports6i Application, by selecting

**Start → Programs → Oracle Reports 6i – OraHome62 → Report Builder.**

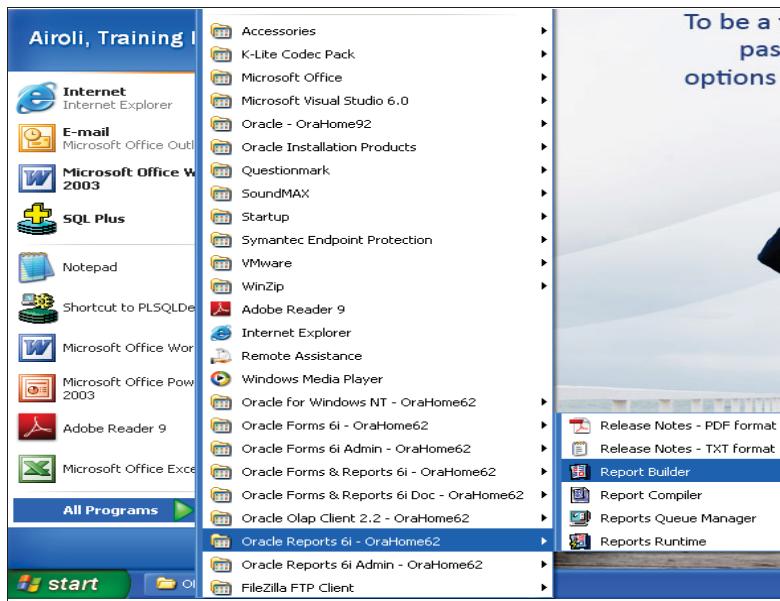


Figure 1: Start Report Builder

**Step 2:** The option 'Use the report wizard' is automatically selected for the first time, so click on the OK button.

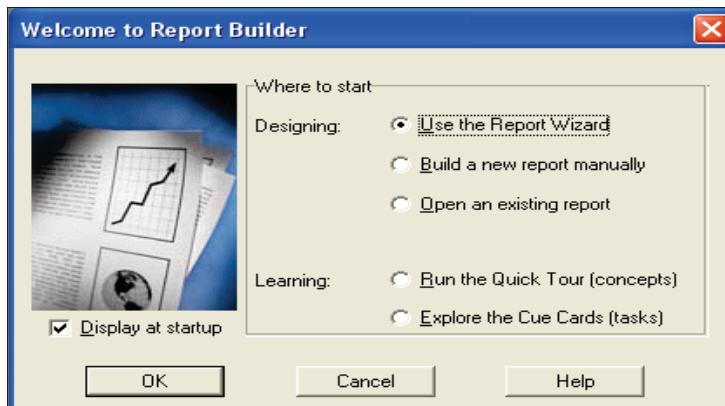
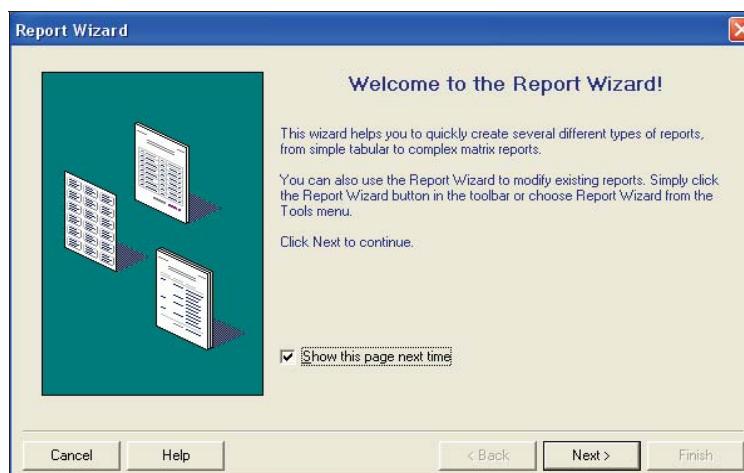


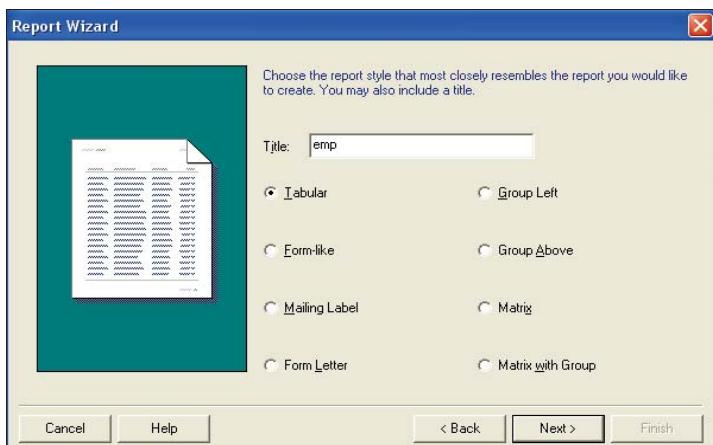
Figure 2: Welcome Screen of Report Wizard

Step 3: Next comes the welcome screen of report wizard. Click next.



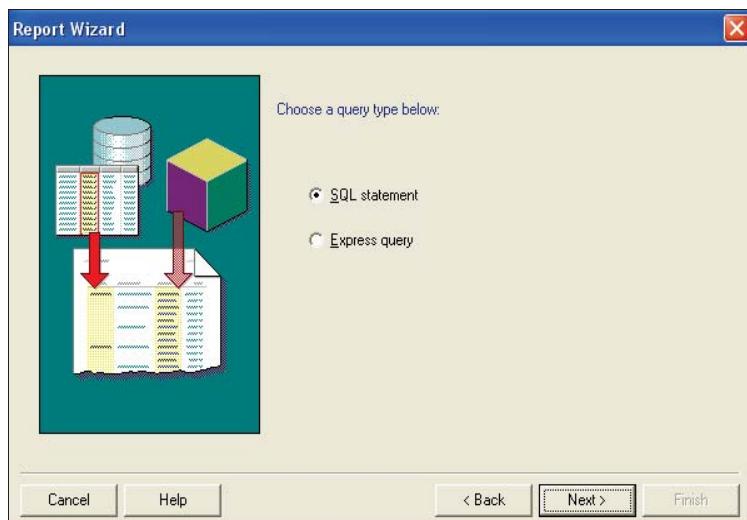
**Figure 3: Welcome Screen of Report Wizard**

**Step 4:** Choose tabular from the available styles of report and click next.



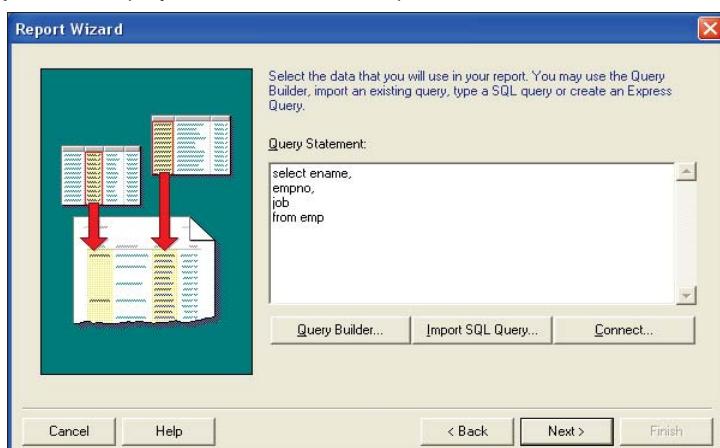
**Figure 4: Report Wizard Style Page**

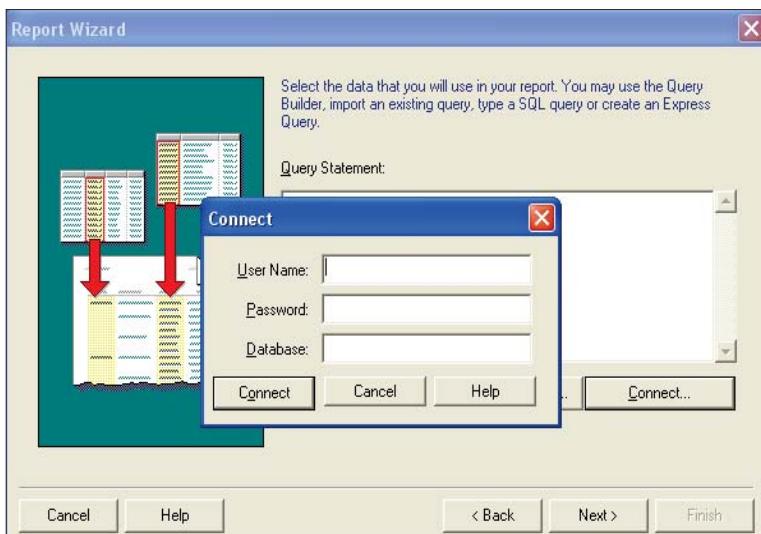
**Step 5:** Choose SQL statement option to write the query.



**Figure 5: Query page 1**

**Step 6:** Write the query to be executed in the area provided and click the connect button.





**Figure 6:** Query page 2

**Step 7:** Enter the valid username and password and also the database to which you want to get connected and click connect button. Click next.

**Figure 7:** Connection dialog box

**Step 8:** Select the fields which you want to display in your report from the available fields and put them into the displayed fields. To select all the fields use the double right arrowed button.

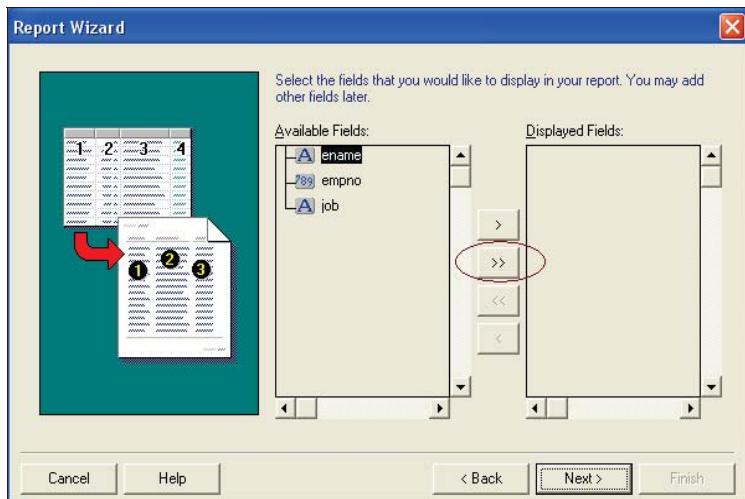
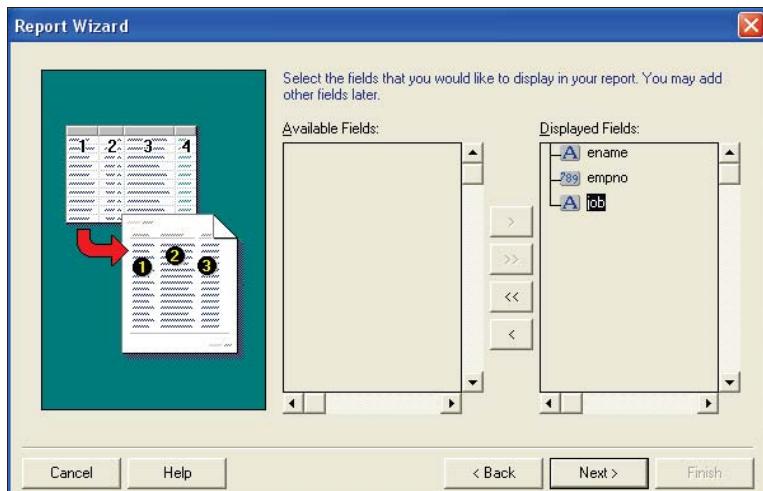


Figure 8: Field selection

**Step 9:** Select all the available fields and put them into the displayed field box.



**Figure 9: Field selection**

**Step 10:** Computations can be done in this screen.

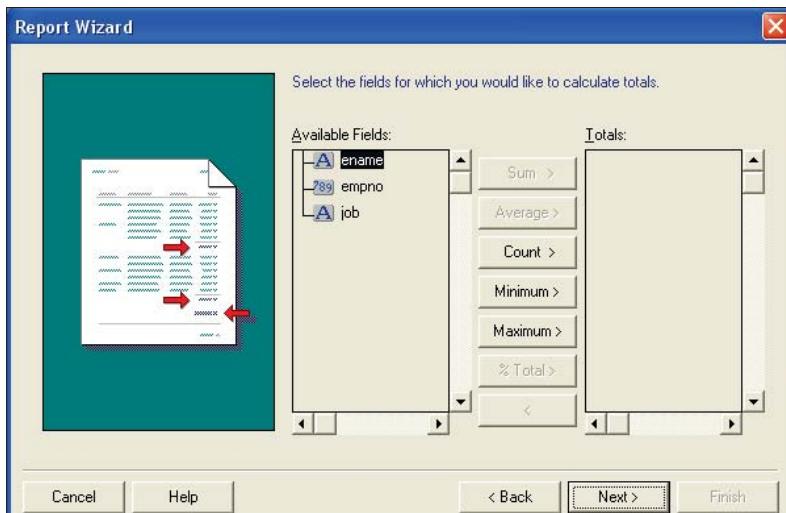
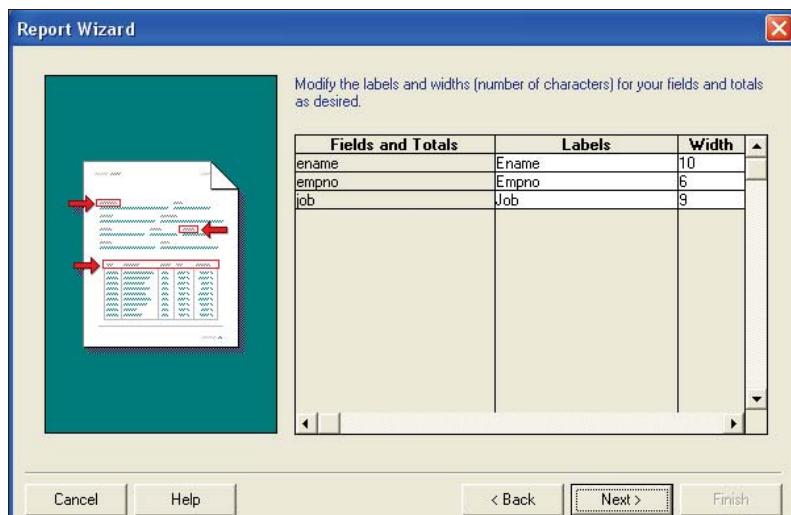


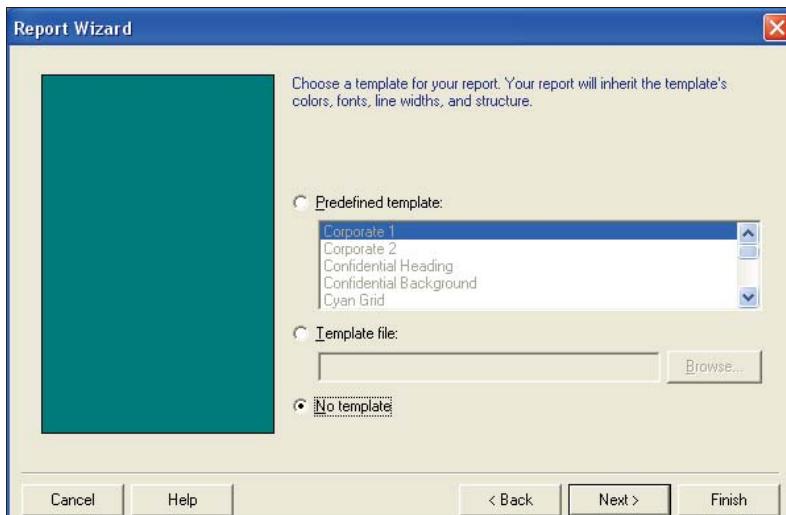
Figure 10: Total page

**Step11:** This screen allows you to enter the labels and width for the field names.



**Figure 11: Labels page**

**Step 12:** This screen allows you to define the templates for the report.



**Figure 12: Template page**

**Step 13:** You are now ready to exit the report wizard.

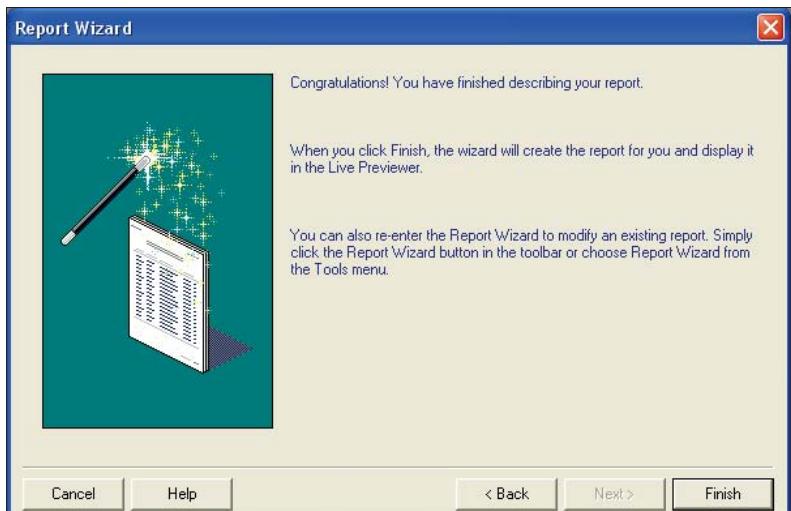


Figure 13: Finish page

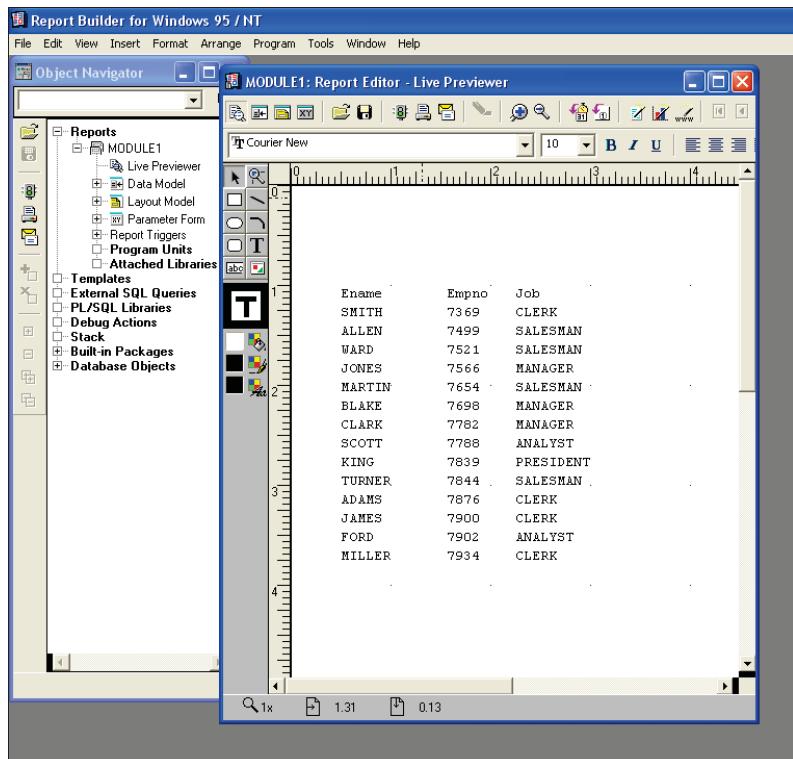


Figure 14: Report Editor Live Previewer

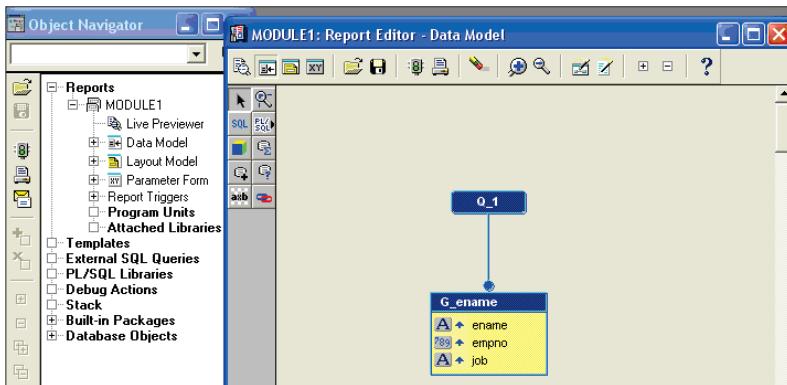


Figure 15: Report Editor Data Model

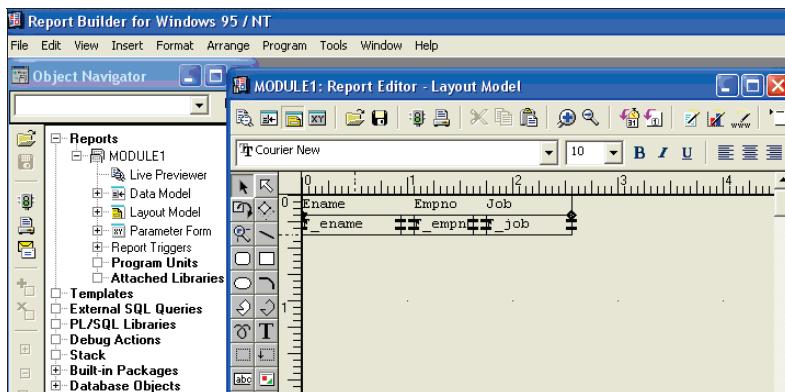


Figure 16: Report Editor Layout Model

**1.2.** Create a report which displays all the employee details using wizard (**To Do**).

**1.3.** Manually create a report which displays the employee name, employee ID and salary of all the employees present in the employee table (**To Do**).

## Lab 2. Group Report

|       |                          |
|-------|--------------------------|
| Goals | To create a group report |
| Time  |                          |

### 2.1: Creating group report

#### Solution:

Follow **steps 1-5** from **LAB 1**.

**Step 6:** Enter the query to be executed and connect to the database.

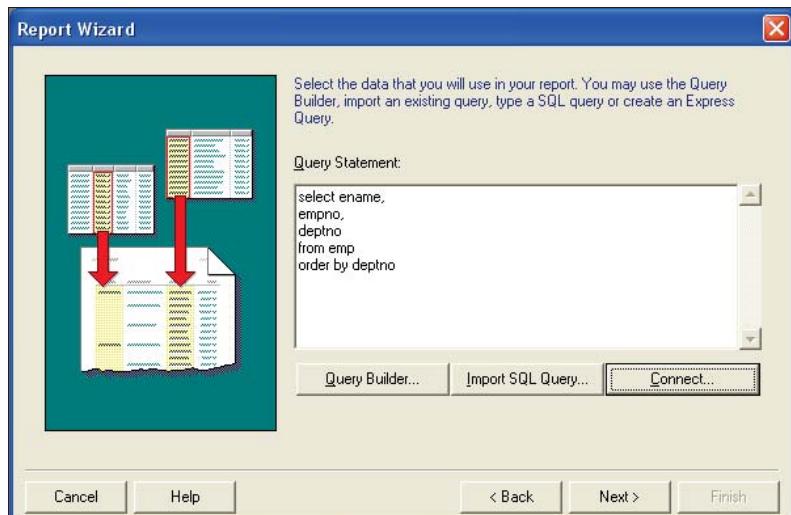
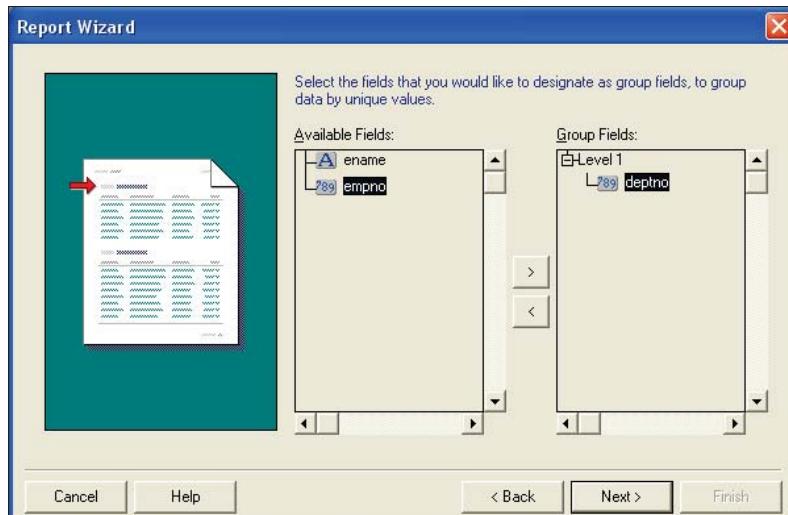


Figure 17: Query page

**Step 7:** Add the column which is supposed to be grouped to the group fields.



**Figure 18: Groups page**

Follow **Steps 8 to 13** from **LAB 1**.

**Step 14:**

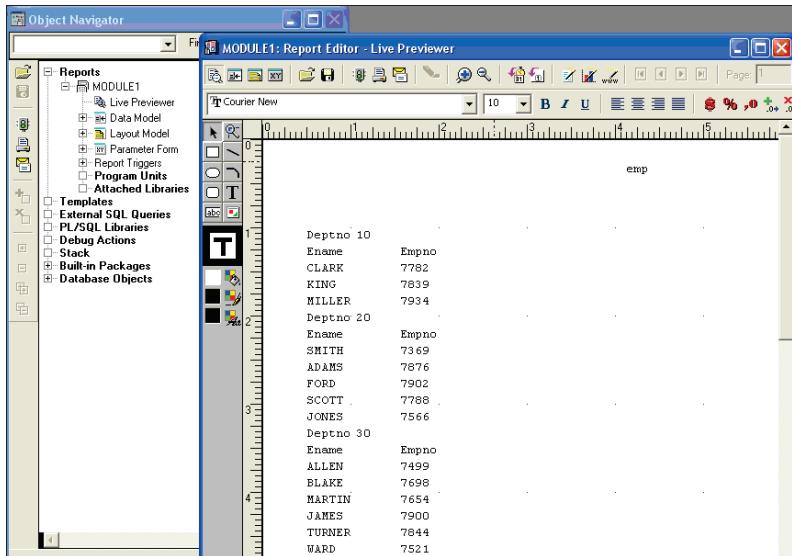


Figure 19: Group report created

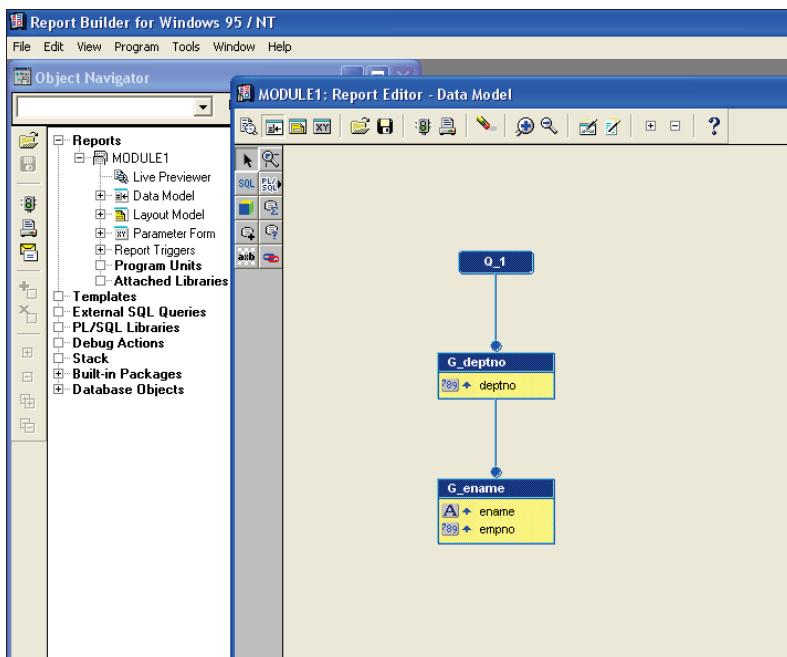


Figure 20: Data model window for Group report

- 2.2. Manually create a report that displays the employee details department wise (**To Do**).
- 2.3. Create a group left style report using wizard that displays the employee names in the hierarchical pattern with respect to their designations (**To Do**).

### Lab 3. BIND AND LEXICAL PARAMETERS

|       |                                                                                                                             |
|-------|-----------------------------------------------------------------------------------------------------------------------------|
| Goals | <ul style="list-style-type: none"><li>• To create bind variables with LOVs</li><li>• To create Lexical parameters</li></ul> |
| Time  |                                                                                                                             |

#### 3.1: Bind parameter with LOVs

Solution:

Follow Steps 1 to 5 from **LAB 1**.

**Step 6:** Enter the query with a bind parameter.

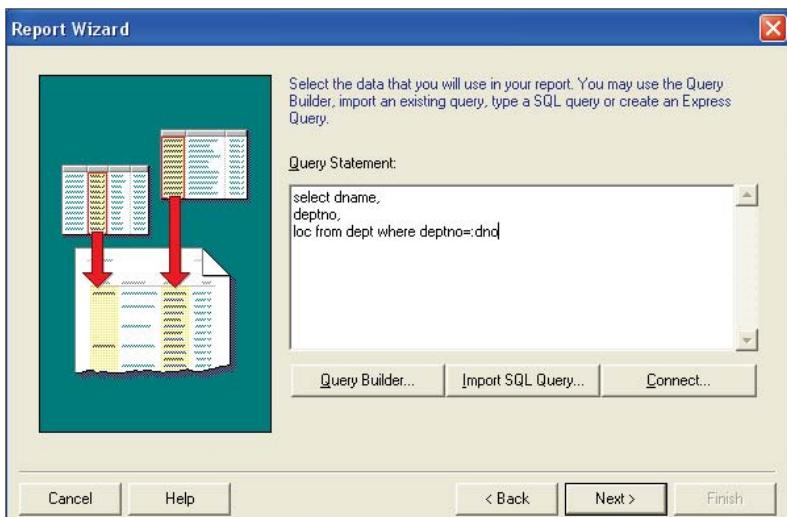
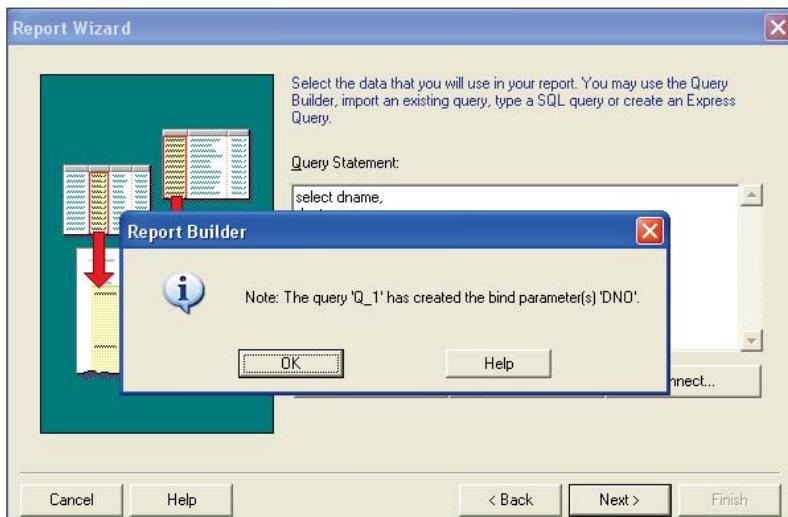


Figure 21: Query Page

**Message displayed confirming the creation of bind variable.****Figure 22: Bind Variable Indication Dialog Box**

Follow Steps 7 to 13 from LAB 1.

**Step 14:** Enter the value for the bind parameter.

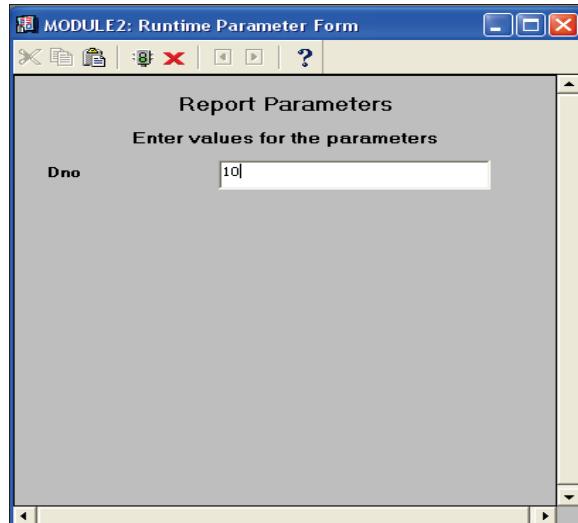


Figure 23: Parameter Box

**Step 14:** To create LOVs for bind variable.

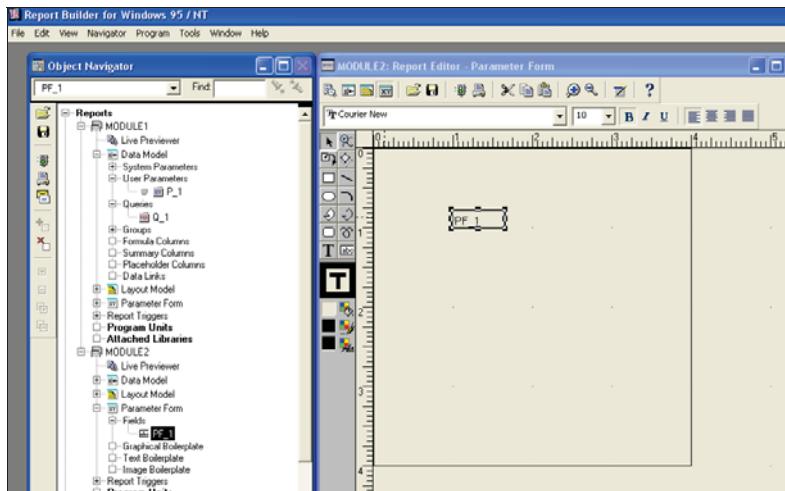


Figure 24: Parameter Window 1

**Step 15:**

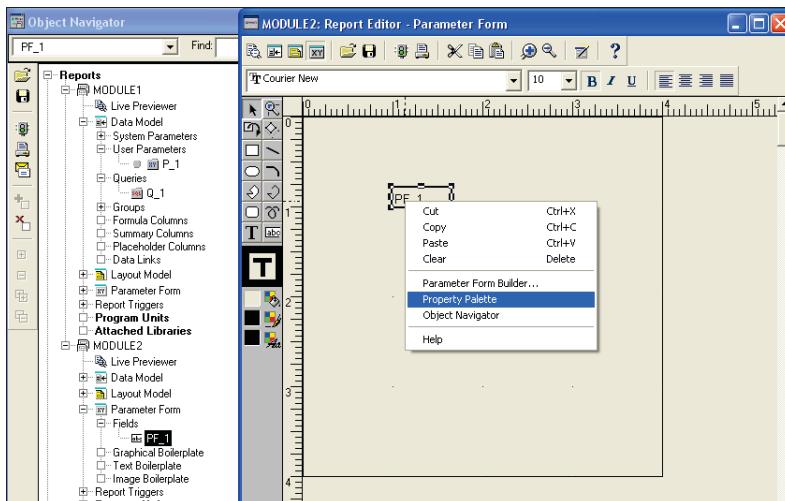


Figure 25: Parameter Window 2

Step 16:

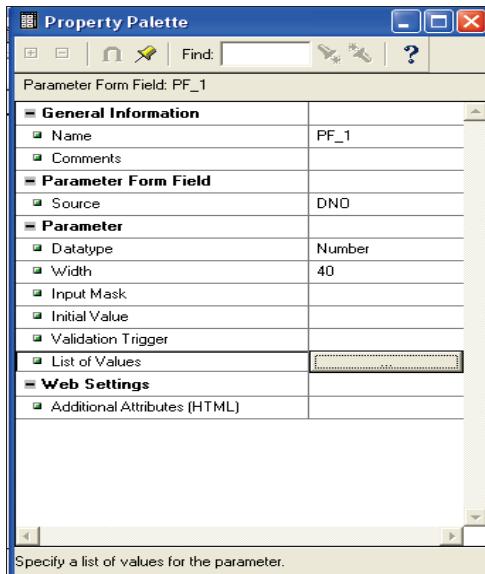


Figure 26: Property Palette

Step 17: Add values which are to be displayed in LOV list.

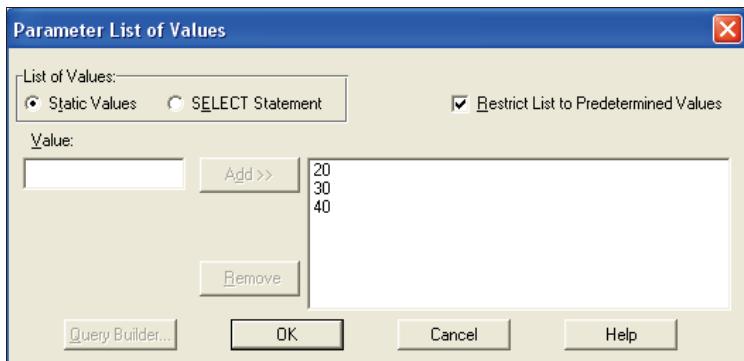


Figure 27: Adding LOVs

Step 18: The LOV getting displayed.

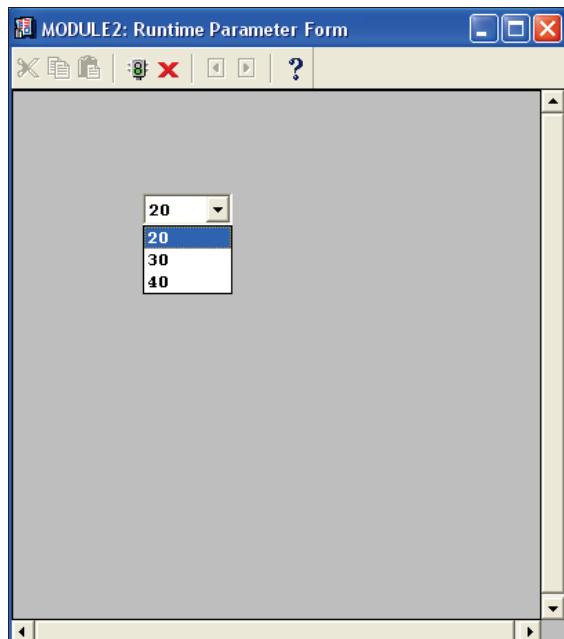
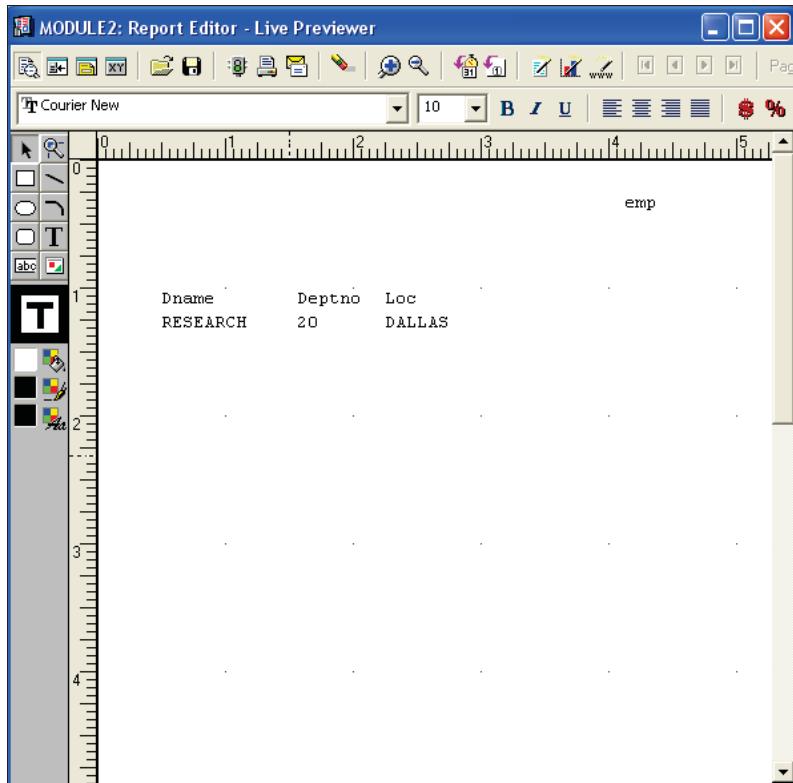


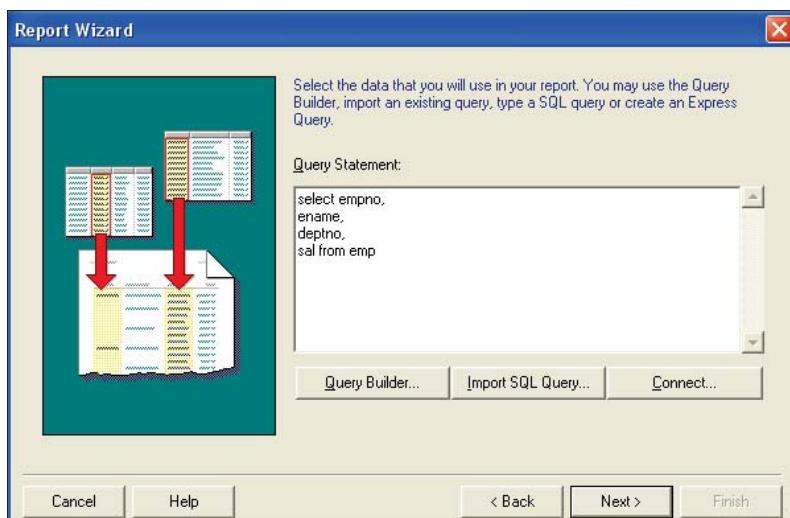
Figure 28: Parameter Window with LOV

**Step 19:** Output displayed after selecting a value from the LOV.



**Figure 29: Output Window**

## 3.2: Lexical parameter

**Solution:**Follow **Steps 1 to 5** from **LAB 1**.**Step 6:** Enter the query.**Figure 30: Query Page**Follow **Steps 7 to 13** from **LAB 1**.

**Step 14:** Create a user parameter with an initial value.

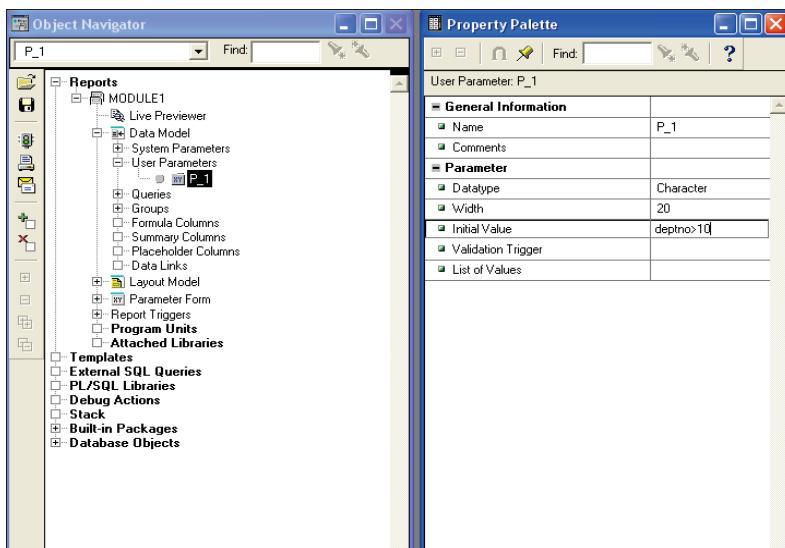
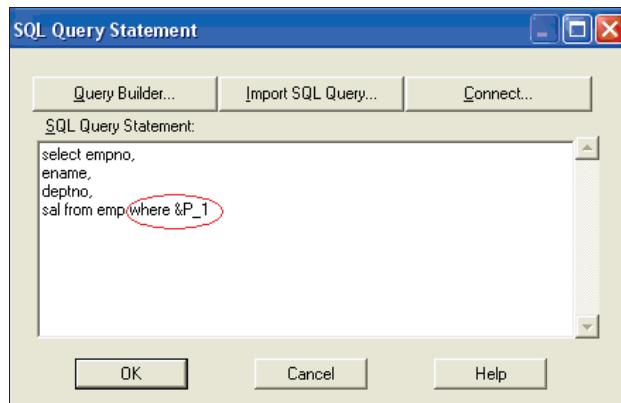


Figure 31: User Parameters Page

**Step 15:** Add the created parameter as lexical parameter in the query.



**Figure 32: Query Page 3**

**Step 16:** Parameter window which appears at the time of execution.

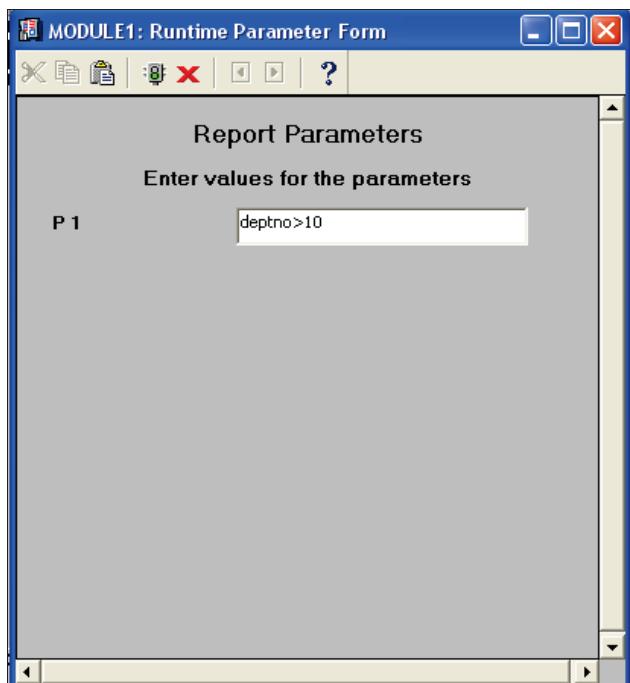
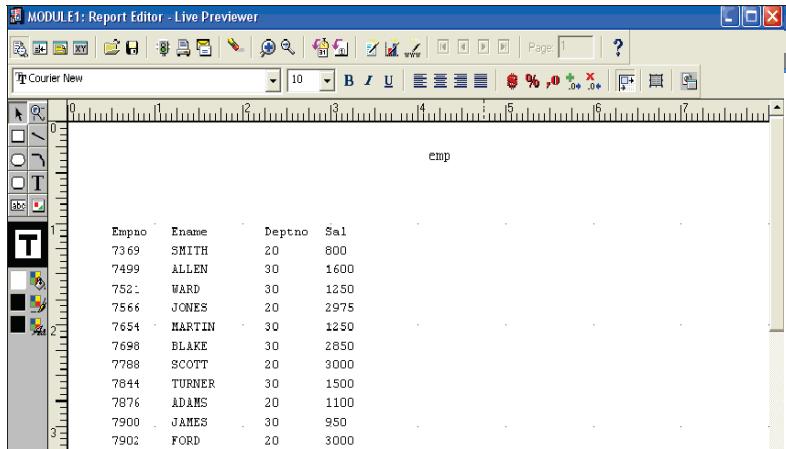


Figure 33: Parameter Window

**Step 14:** Executed output.

The screenshot shows a report editor interface with a toolbar at the top, a ruler, and a left-hand panel with various icons. The main area displays a table titled 'emp' containing the following data:

|    | Empno | Ename  | Deptno | Sal  |
|----|-------|--------|--------|------|
| 1  | 7369  | SMITH  | 20     | 600  |
| 2  | 7499  | ALLEN  | 30     | 1600 |
| 3  | 7521  | WARD   | 30     | 1250 |
| 4  | 7566  | JONES  | 20     | 2975 |
| 5  | 7654  | MARTIN | 30     | 1250 |
| 6  | 7698  | BLAKE  | 30     | 2850 |
| 7  | 7788  | SCOTT  | 20     | 3000 |
| 8  | 7844  | TURNER | 30     | 1500 |
| 9  | 7876  | ADAMS  | 20     | 1100 |
| 10 | 7900  | JAMES  | 30     | 950  |
| 11 | 7902  | FORD   | 20     | 3000 |

**Figure 34: Output Page**

**3.3.** Manually create a report using Lexical parameter, displaying the employees who earn more than Rs.3000 (**To Do**).

**3.4.** Manually create a report which displays employee details on selecting the employee number. Use Bind variable and LOVs which displays all the employee number (**To Do**).

## Lab 4. Formula and Summary

|       |                                                                                                                                                           |
|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Goals | <ul style="list-style-type: none"><li>• To create a report by using the Formula Column</li><li>• To create a report by using the Summary Column</li></ul> |
| Time  |                                                                                                                                                           |

### 4.1: Formula

Step 1: Choose the option to manually create the report.

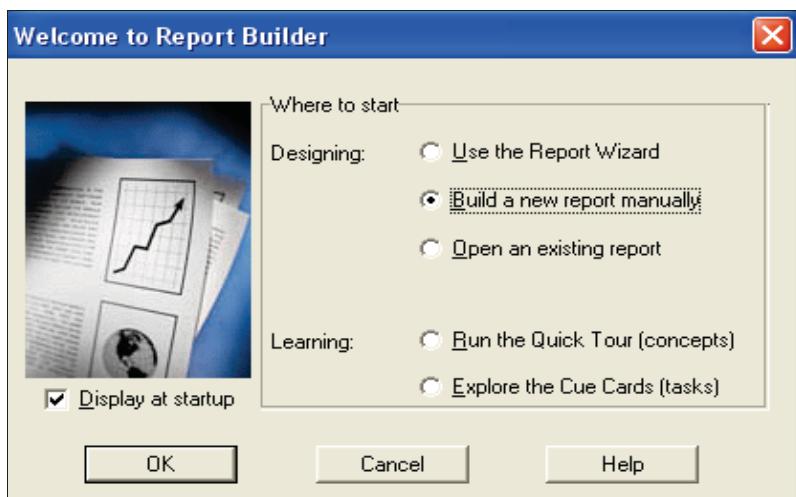
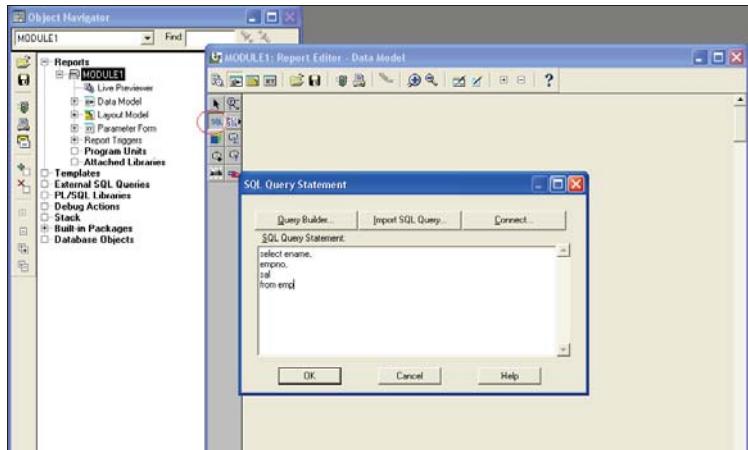


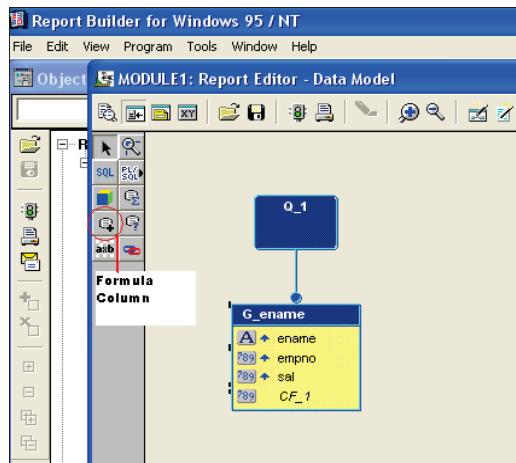
Figure 35: Welcome window

**Step 2:** Open the data model window by clicking the data model icon or view -> data model from the menu bar. Then click the SQL query tool from the tool bar to type the query statement.



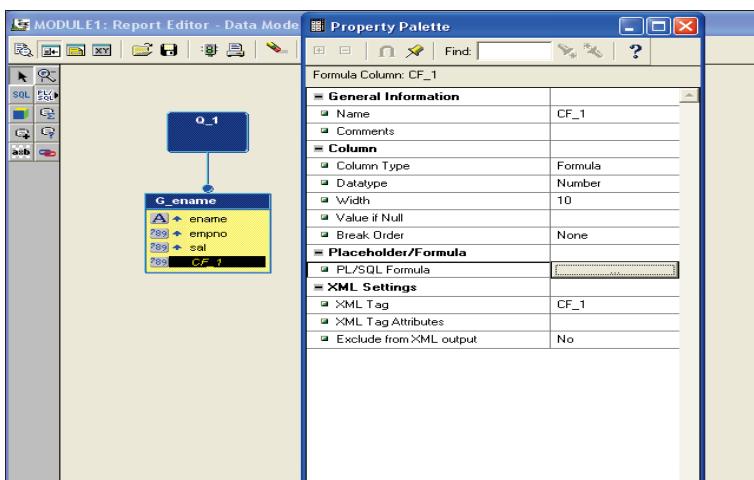
**Figure 36: SQL query window within data model window**

**Step 3:** Include a formula column (to do calculation) in the data model window by clicking the formula column tool.



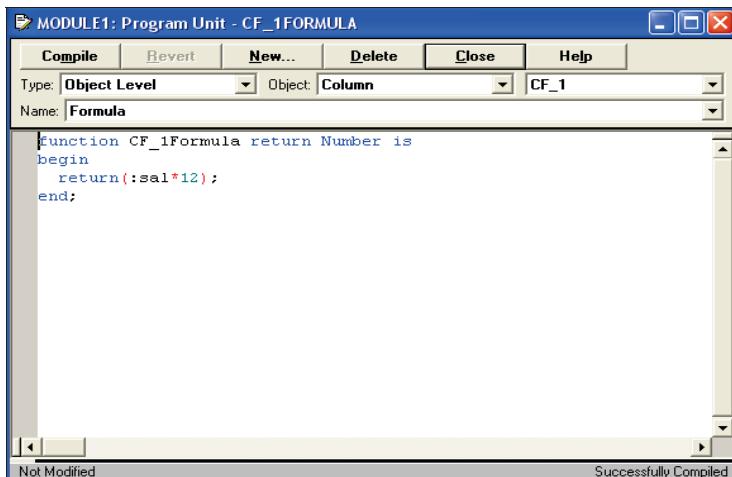
**Figure 37: Data model window after connecting to database**

**Step 4:** Include the formula which is to be calculated in the PL/SQL Formula block from the property palette of formula column.



**Figure 38: Property palette of formula column**

**Step 5:** Write the function to be calculated in the formula column.

**Figure 39: Function window**

The function written for calculating annual income of each employee.

```
function CF_1Formula return Number is
begin
 return(:sal*12);
end;
```

**Step 6:** Open the layout model window by clicking the layout model icon or view -> layout model from the menu bar. Then design the report with the help of the tools present in the tool bar.

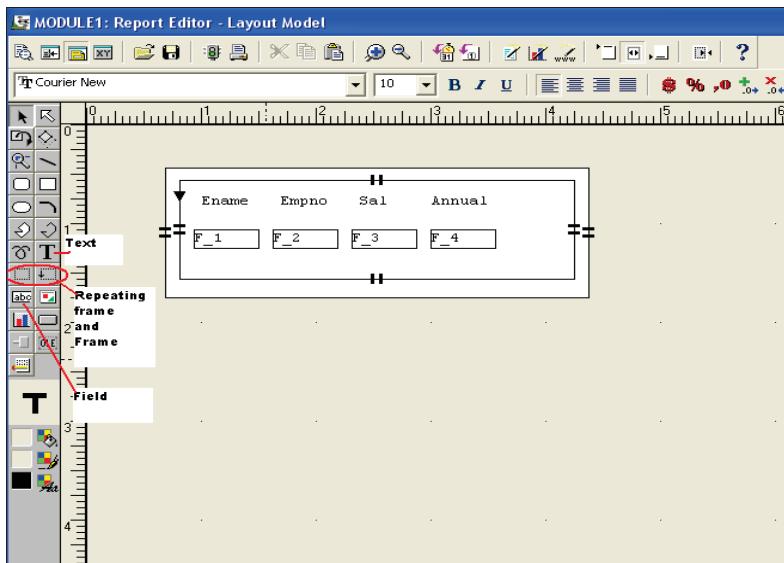


Figure 40: Layout model window with a report design

**Step 7:** Set the property of each item in the layout model using its respective property palette.

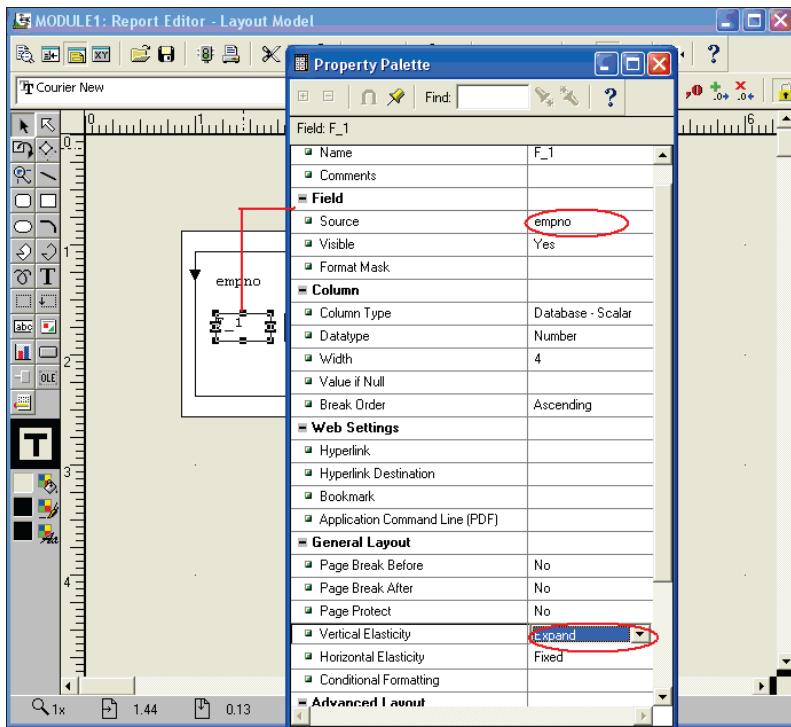
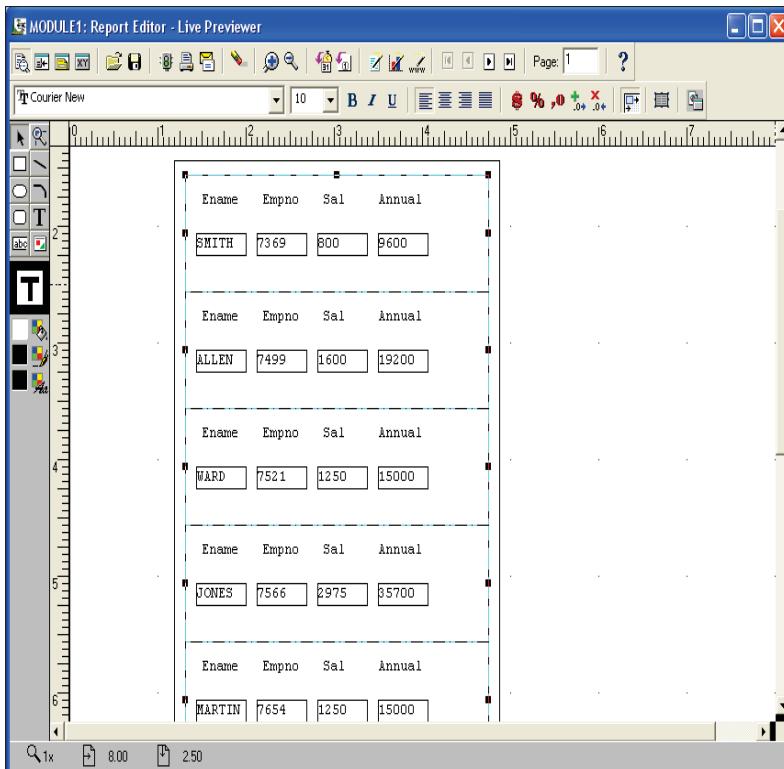


Figure 41: Property palette of a text box

**Step 8:** Execute the report to view the output.



**Figure 42: Output window**

**4.2.** Manually create a report to display the employee name, employee number, salary, commission and the total salary (salary + commission) for each employee. Also display the grand total of the employee's salary and total salary respectively (**To Do**).

## Lab 5. Triggers

|       |                                                                                                                                      |
|-------|--------------------------------------------------------------------------------------------------------------------------------------|
| Goals | <ul style="list-style-type: none"><li>• To create a report using Triggers</li><li>• To demonstrate the use of SRW packages</li></ul> |
| Time  |                                                                                                                                      |

### 5.1: Report with after parameter form trigger:

Solution:

Follow Steps 1 to 5 from LAB 1.

Step 6: Enter the query to be executed.

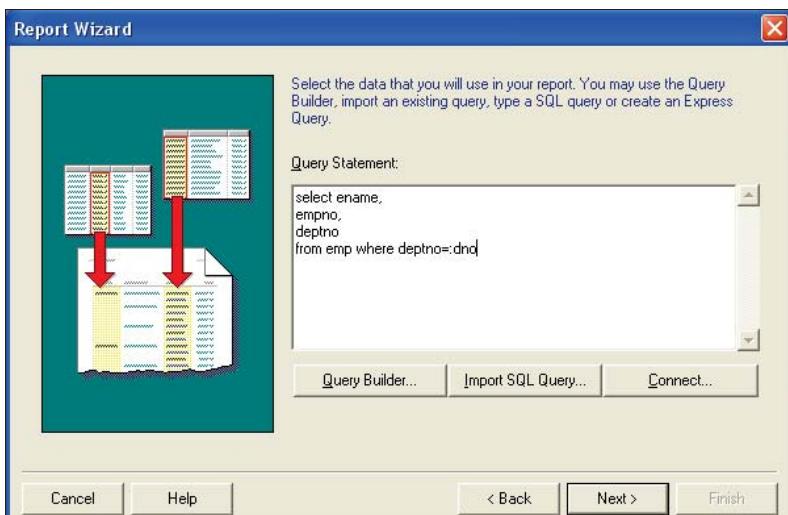


Figure 43: Query Page

Message displayed, confirming the creation of bind variable.

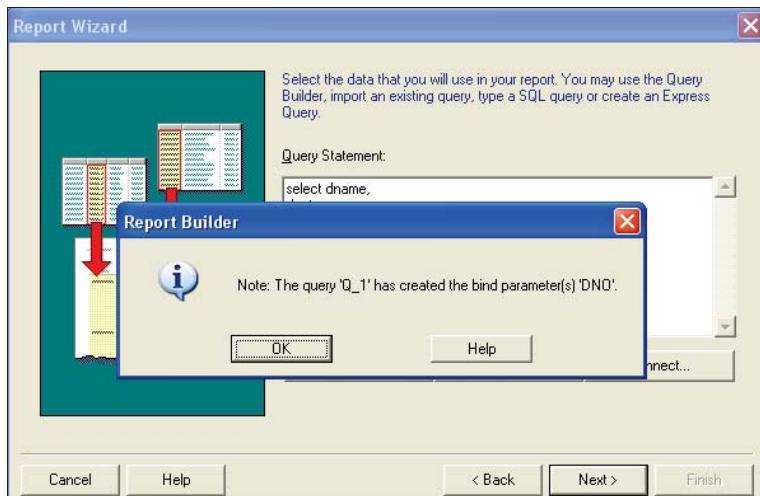


Figure 44: Bind Variable Indication Dialog Box

Follow Steps 7 to 13 from LAB 1.

**Step 14: Create an AFTER PARAMETER FORM trigger.**

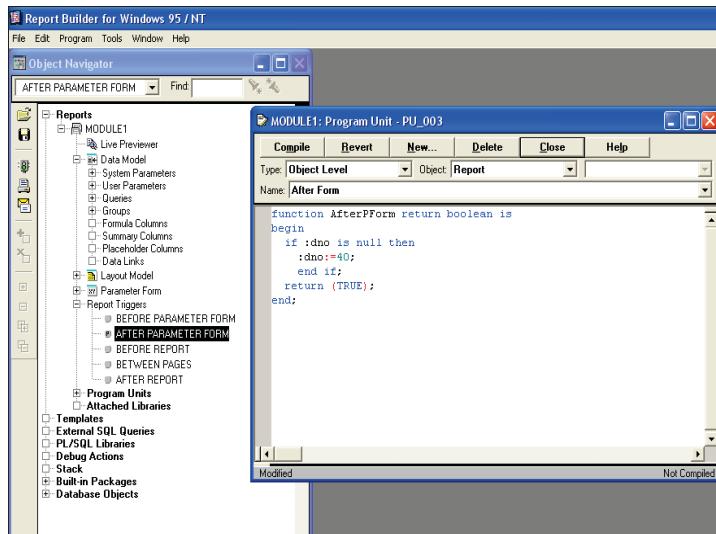


Figure 45: After Parameter Form Trigger Page

**Trigger Code:**

```

function AfterPForm return boolean is
begin
 if :dno is null then
 :dno:=40;
 end if;
 return (TRUE);
end;

```

Step 15: Parameter window at the time of execution.

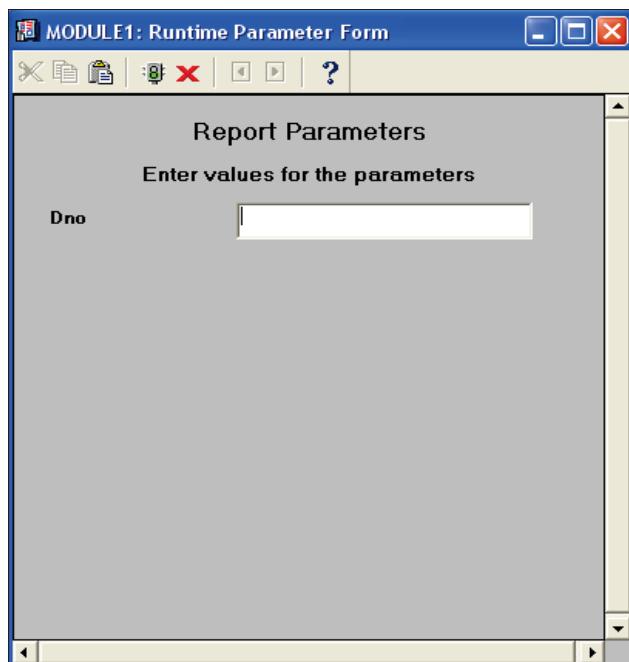


Figure 46: Parameter window

**Step 16:** Parameter window at the time of execution.

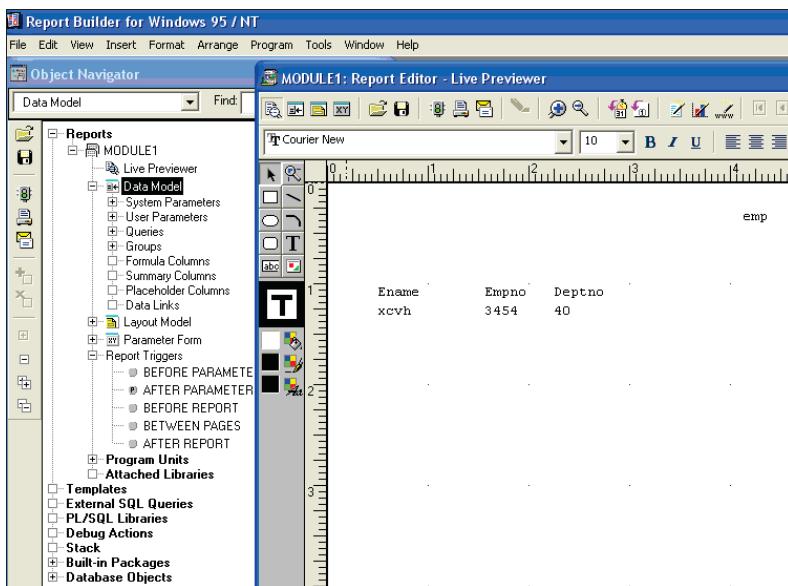


Figure 47: Output Page

## 5.2. Demonstrating SRW package:

**Step 17:** Write the SRW package in the format trigger from the property palette.

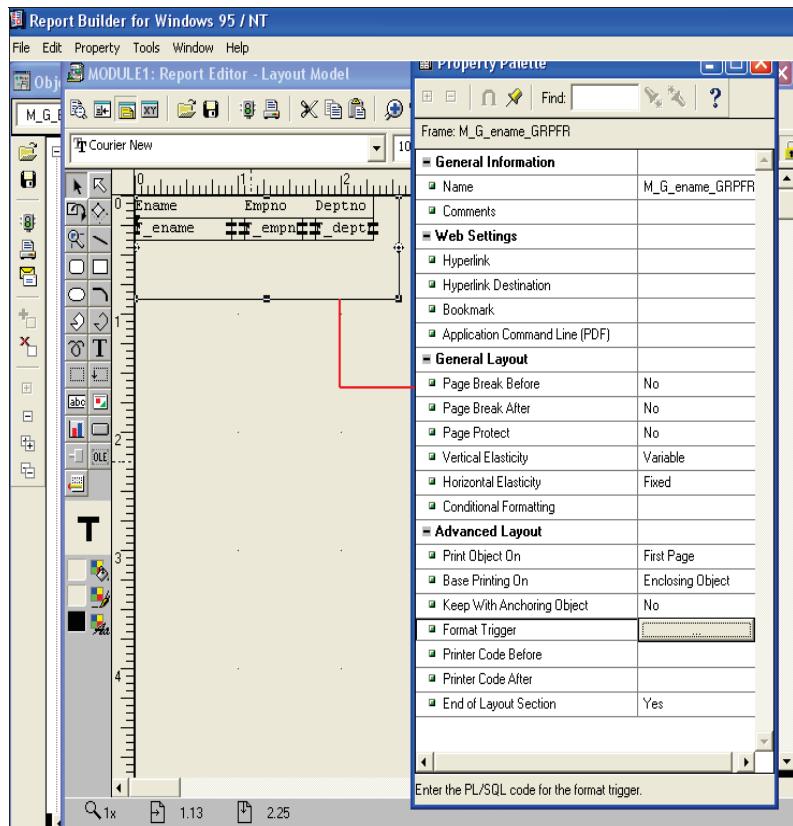


Figure 48: Property Palette Page

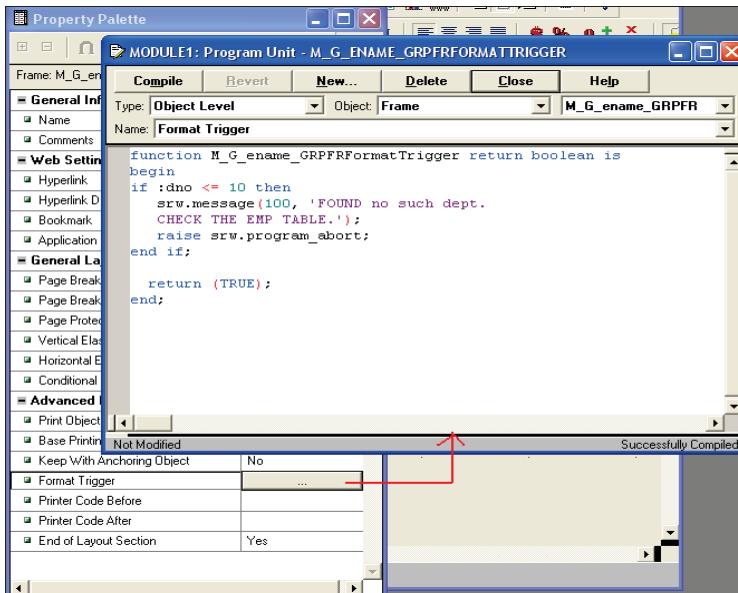


Figure 49: Format Trigger Page

**SRW Code:**

```
function M_Gename_GRPFRFormatTrigger return boolean is
begin
if :dno <= 10 then
 srw.message(100, 'FOUND no such dept.
 CHECK THE EMP TABLE.');
 raise srw.program_abort;
end if;

return (TRUE);
end;
```

**Step 18:** Parameter window at the time of execution.

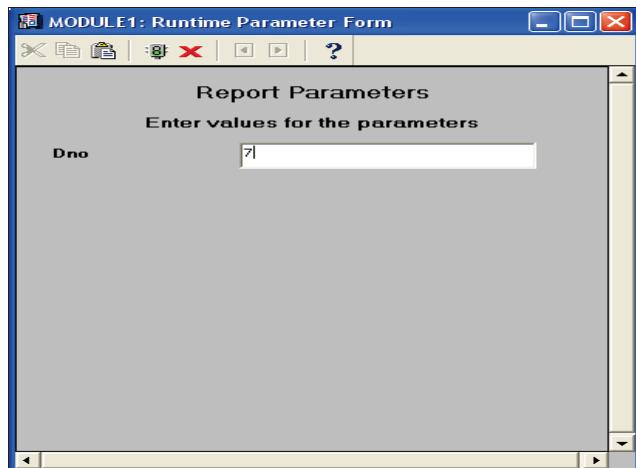


Figure 50: Parameter Window

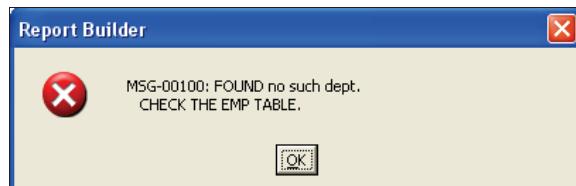


Figure 51: Error displayed when dept no is entered less than 10

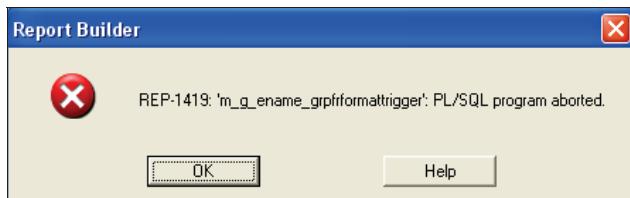


Figure 52: Abort message displayed

**5.3.** Create a report manually to display a department wise sum of salary along with commission, with the message displaying "This is the report displaying sum of sal and comm department wise". And also intimating the abort message after termination of the report.

## Appendices

### Appendix A: Table of Figures

|                                                                 |    |
|-----------------------------------------------------------------|----|
| Figure 1: Start Report Builder .....                            | 9  |
| Figure 2: Welcome Screen of Report Wizard .....                 | 10 |
| Figure 3: Welcome Screen of Report Wizard .....                 | 11 |
| Figure 4: Report Wizard Style Page.....                         | 11 |
| Figure 5: Query page 1.....                                     | 12 |
| Figure 6: Query page 2.....                                     | 13 |
| Figure 7: Connection dialog box.....                            | 13 |
| Figure 8: Field selection .....                                 | 14 |
| Figure 9: Field selection .....                                 | 15 |
| Figure 10: Total page .....                                     | 16 |
| Figure 11: Labels page.....                                     | 17 |
| Figure 12: Template page .....                                  | 18 |
| Figure 13: Finish page .....                                    | 19 |
| Figure 14: Report Editor Live Previewer .....                   | 20 |
| Figure 15: Report Editor Data Model .....                       | 21 |
| Figure 16: Report Editor Layout Model .....                     | 21 |
| Figure 17: Query page.....                                      | 22 |
| Figure 18: Groups page .....                                    | 23 |
| Figure 19: Group report created .....                           | 24 |
| Figure 20: Data model window for Group report.....              | 25 |
| Figure 21: Query Page .....                                     | 26 |
| Figure 22: Bind Variable Indication Dialog Box .....            | 27 |
| Figure 23: Parameter Box .....                                  | 28 |
| Figure 24: Parameter Window 1 .....                             | 29 |
| Figure 25: Parameter Window 2 .....                             | 30 |
| Figure 26: Property Palette .....                               | 30 |
| Figure 27: Adding LOVs.....                                     | 31 |
| Figure 28: Parameter Window with LOV .....                      | 32 |
| Figure 29: Output Window.....                                   | 33 |
| Figure 30: Query Page .....                                     | 34 |
| Figure 31: User Parameters Page .....                           | 35 |
| Figure 32: Query Page 3 .....                                   | 36 |
| Figure 33: Parameter Window .....                               | 37 |
| Figure 34: Output Page .....                                    | 38 |
| Figure 35: Welcome window .....                                 | 39 |
| Figure 36: SQL query window within data model window .....      | 40 |
| Figure 37: Data model window after connecting to database ..... | 41 |
| Figure 38: Property palette of formula column .....             | 42 |
| Figure 39: Function window .....                                | 42 |
| Figure 40: Layout model window with a report design .....       | 43 |
| Figure 41: Property palette of a text box.....                  | 44 |
| Figure 42: Output window .....                                  | 45 |
| Figure 43: Query Page .....                                     | 46 |

|                                                                      |    |
|----------------------------------------------------------------------|----|
| Figure 44: Bind Variable Indication Dialog Box .....                 | 47 |
| Figure 45: After Parameter Form Trigger Page .....                   | 48 |
| Figure 46: Parameter window.....                                     | 49 |
| Figure 47: Output Page .....                                         | 50 |
| Figure 48: Property Palette Page .....                               | 51 |
| Figure 49: Format Trigger Page .....                                 | 52 |
| Figure 50: Parameter Window .....                                    | 53 |
| Figure 51: Error displayed when dept no is entered less than 10..... | 53 |
| Figure 52: Abort message displayed.....                              | 53 |

## D2K / Forms Lab Book

### Document Revision History

| Date        | Revision No. | Author    | Summary of Changes |
|-------------|--------------|-----------|--------------------|
| 11-Aug-2011 | 1.0          | Amit Sali | Content Creation   |
|             |              |           |                    |
|             |              |           |                    |
|             |              |           |                    |

## Table of Contents

|                                                                                                            |    |
|------------------------------------------------------------------------------------------------------------|----|
| <i>Table of Contents</i> .....                                                                             | 3  |
| <i>Getting Started</i> .....                                                                               | 4  |
| <i>Overview</i> .....                                                                                      | 4  |
| <i>Setup Checklist for EAS Oracle Application</i> .....                                                    | 4  |
| <i>Instructions</i> .....                                                                                  | 4  |
| <i>Problem Statement / Case Study</i> .....                                                                | 5  |
| <i>Lab 1. Wizard base form creation</i> .....                                                              | 8  |
| <i>1.1: Create a form</i> .....                                                                            | 8  |
| <i>Lab 2. Wizard base Master- Child Form creation</i> .....                                                | 23 |
| <i>2.1: Create a form</i> .....                                                                            | 23 |
| <i>2.2. Create a form which shows all items and records of both emp and salgrade tables. (To-do)</i> ..... | 34 |
| <i>Lab 3. Working With Items</i> .....                                                                     | 35 |
| <i>3.1: Create a form</i> .....                                                                            | 35 |
| <i>Lab 4. Summary and Formula functions</i> .....                                                          | 51 |
| <i>4.1: Calculate the sum of salaries of a department</i> .....                                            | 51 |
| <i>Lab 5. Triggers</i> .....                                                                               | 54 |
| <i>Lab 6. Validation</i> .....                                                                             | 58 |
| <i>Appendices</i> .....                                                                                    | 61 |
| <i>Appendix A: Table of Figures</i> .....                                                                  | 61 |

## Getting Started

### Overview

This lab book is a guided tour for learning EAS oracle application. It comprises 'To Do' assignments. Follow the steps provided and work out the 'To Do' assignments.

### Setup Checklist for EAS Oracle Application

Here is what is expected on your machine in order for the lab to work.

#### Minimum System Requirements

- Intel Pentium 90 or higher (P166 recommended)
- Microsoft Windows 95, 98, or NT 4.0, 2k, XP.
- Memory: 32MB of RAM (64MB or more recommended)
- Internet Explorer 6.0 or higher

### Instructions

- For all coding standards refer Appendix A. All lab assignments should refer coding standards.
- Create a directory by your name in drive <drive>. In this directory, create a subdirectory <eas orapps>\_assgn. For each lab exercise create a directory as lab <lab number>.

## Problem Statement / Case Study

Table descriptions and data considered in this lab book:

### EMP Table

SQL>DESC EMP

| Name     | Null?    | Type         |
|----------|----------|--------------|
| EMPNO    | NOT NULL | NUMBER(4)    |
| ENAME    |          | VARCHAR2(10) |
| JOB      |          | VARCHAR2(50) |
| MGR      |          | NUMBER(4)    |
| HIREDATE |          | DATE         |
| SAL      |          | NUMBER(7,2)  |
| COMM     |          | NUMBER(7,2)  |
| DEPTNO   |          | NUMBER(2)    |

SQL>SELECT \* FROM EMP

| EMPNO | ENAME  | JOB       | MGR  | HIREDATE  | SAL  | COMM | DEPTNO |
|-------|--------|-----------|------|-----------|------|------|--------|
| 7369  | SMITH  | CLERK     | 7902 | 17-DEC-80 | 800  |      | 20     |
| 7499  | ALLEN  | SALESMAN  | 7698 | 20-FEB-81 | 1600 | 300  | 30     |
| 7521  | WARD   | SALESMAN  | 7698 | 22-FEB-81 | 1250 | 500  | 30     |
| 7566  | JONES  | MANAGER   | 7839 | 02-APR-81 | 2975 |      | 20     |
| 7654  | MARTIN | SALESMAN  | 7698 | 28-SEP-81 | 1250 | 1400 | 30     |
| 7698  | BLAKE  | MANAGER   | 7839 | 01-MAY-81 | 2850 |      | 30     |
| 7782  | CLARK  | MANAGER   | 7839 | 09-JUN-81 | 2450 |      | 10     |
| 7788  | SCOTT  | ANALYST   | 7566 | 09-DEC-82 | 3000 |      | 20     |
| 7839  | KING   | PRESIDENT |      | 17-NOV-81 | 5000 |      | 10     |
| 7844  | TURNER | SALESMAN  | 7698 | 08-SEP-81 | 1500 | 0    | 30     |
| 7876  | ADAMS  | CLERK     | 7788 | 12-JAN-83 | 1100 |      | 20     |
| 7900  | JAMES  | CLERK     | 7698 | 03-DEC-81 | 950  |      | 30     |

|      |        |         |      |           |      |  |    |
|------|--------|---------|------|-----------|------|--|----|
| 7902 | FORD   | ANALYST | 7566 | 03-DEC-81 | 3000 |  | 20 |
| 7934 | MILLER | CLERK   | 7782 | 23-JAN-82 | 1300 |  | 10 |

14 rows selected.

#### DEPT Table

SQL>DESC DEPT

| Name   | Null? | Type         |
|--------|-------|--------------|
| DEPTNO |       | NUMBER(2)    |
| DNAME  |       | VARCHAR2(14) |
| LOC    |       | VARCHAR2(13) |

SQL>SELECT \* FROM DEPT

| DEPTNO | DNAME      | LOC      |
|--------|------------|----------|
| 10     | ACCOUNTING | NEW YORK |
| 20     | RESEARCH   | DALLAS   |
| 30     | SALES      | CHICAGO  |
| 40     | OPERATIONS | BOSTON   |

**SALGRADE Table**

SQL&gt;DESC SALGRADE

| Name  | Null? | Type   |
|-------|-------|--------|
| GRADE |       | NUMBER |
| LOSAL |       | NUMBER |
| HISAL |       | NUMBER |

SQL&gt;SELECT \* FROM SALGRADE

| GRADE | LOSAL | HISAL |
|-------|-------|-------|
| 1     | 700   | 1200  |
| 2     | 1201  | 1400  |
| 3     | 1401  | 2000  |
| 4     | 2001  | 3000  |
| 5     | 3001  | 9999  |

**BONUS Table**

SQL&gt;DESC BONUS

| Name  | Null? | Type         |
|-------|-------|--------------|
| ENAME |       | VARCHAR2(10) |
| JOB   |       | VARCHAR2(9)  |
| SAL   |       | NUMBER       |
| COMM  |       | NUMBER       |

SQL&gt;SELECT \* FROM BONUS

no rows selected

## Lab 1. Wizard base form creation

|       |                                                                               |
|-------|-------------------------------------------------------------------------------|
| Goals | <ul style="list-style-type: none"> <li>Create a form using wizard.</li> </ul> |
| Time  | 2.5 hr                                                                        |

### 1.1: Create a form

The steps in this lab will walk you through the wizard for creating form.

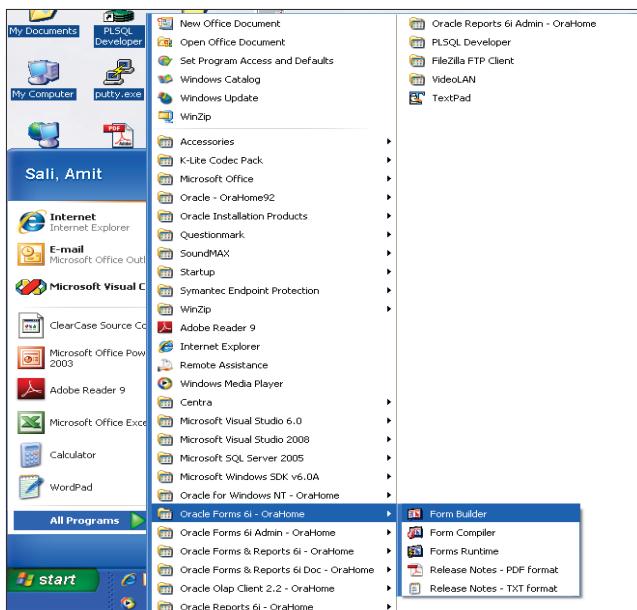
You will have to do the following tasks:

1. Create a using wizard
2. Saving the form
3. Executing the form

#### Solution:

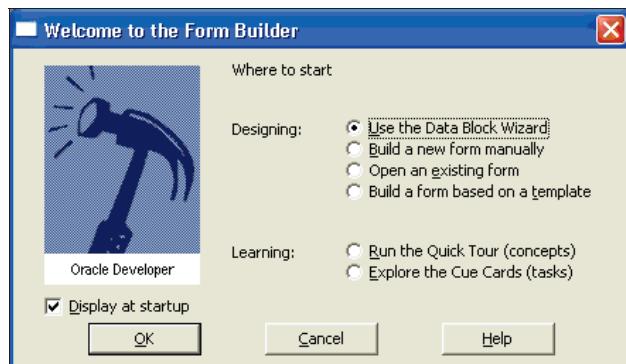
##### Step 1: Data block and Layout wizard:

Open form 6i Application, by selecting **Start → Programs → Oracle Forms 6i – OraHome → Form Builder**.



**Figure 1: All Programs menu**

**Step 2:** Select Data block wizard and press OK.



**Figure 2: Start of data block wizard**

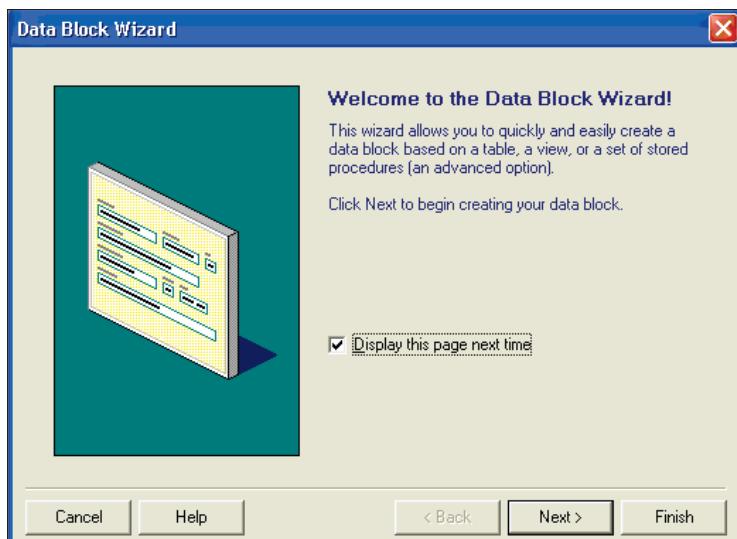


Figure 3: Data block wizard

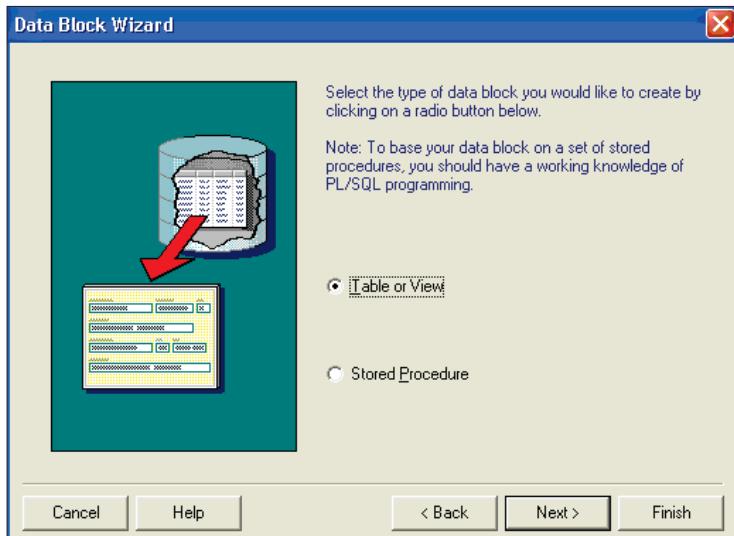


Figure 4: Data block wizard

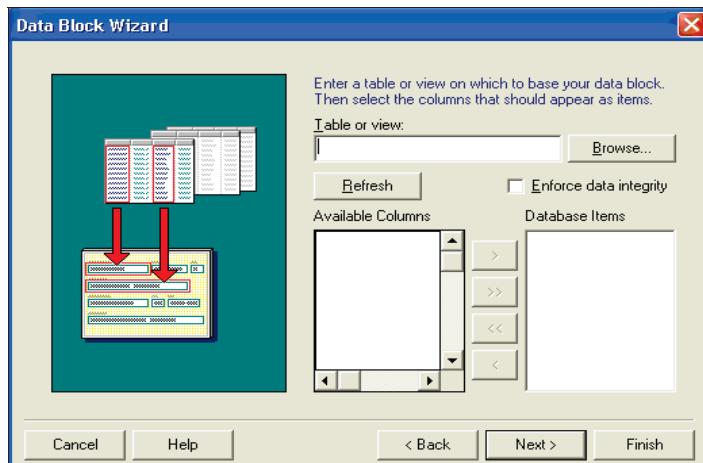


Figure 5: Data block wizard

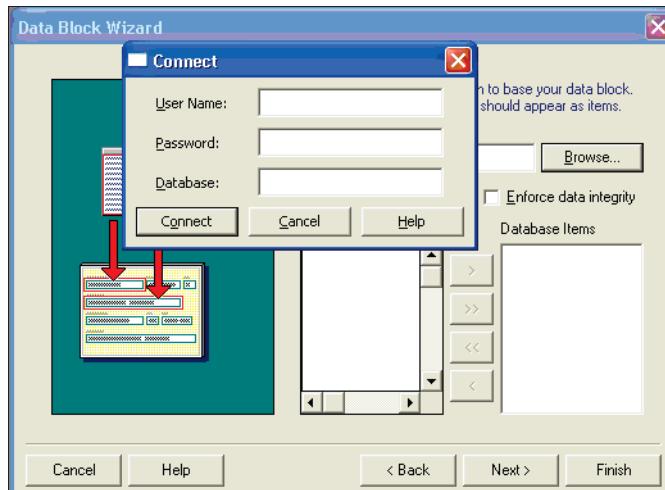


Figure 6: Data block wizard, Database connectivity

**Tables**

Display:

|                                                  |                                            |
|--------------------------------------------------|--------------------------------------------|
| <input checked="" type="checkbox"/> Current user | <input checked="" type="checkbox"/> Tables |
| <input type="checkbox"/> Other users             | <input type="checkbox"/> Views             |
|                                                  | <input type="checkbox"/> Synonyms          |

Table Owner

|               |  |
|---------------|--|
| AUDITLOG      |  |
| BANK_ACCOUNT  |  |
| BONUS         |  |
| CHK           |  |
| CUSTOMER      |  |
| DEPARTMENTS_1 |  |
| <b>:DEPT</b>  |  |
| DUMMY         |  |
| EMP           |  |
| EMP20         |  |
| EMPLOYEE      |  |
| EMPLOYEEAUDIT |  |
| EMP_LOC       |  |
| ERROR_LOG     |  |
| FINAL         |  |
| FUNCDEMO      |  |
| INV_01_2010   |  |
| INV_02_2010   |  |
| INV_03_2010   |  |

Figure 7: Data block wizard, Table selection

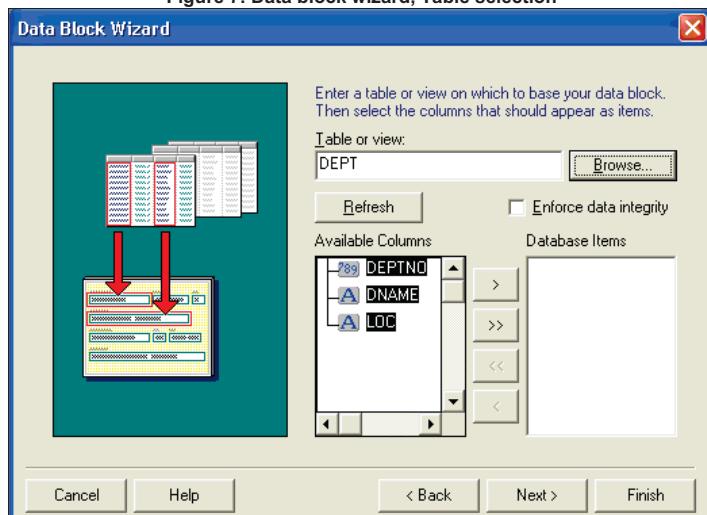


Figure 8: Data block wizard, column selection

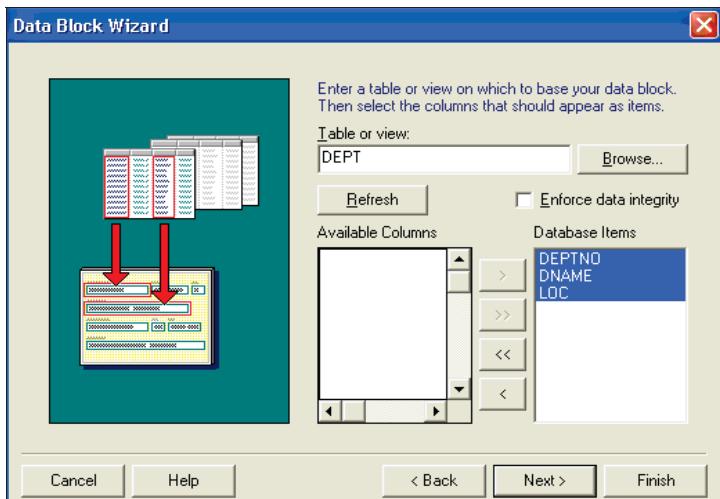


Figure 9: Data block wizard, column selection

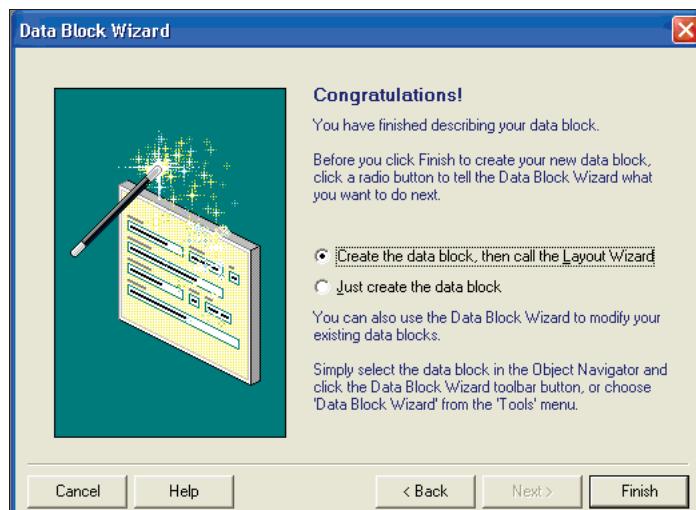


Figure 10: Data block wizard



Figure 11: Layout Wizard

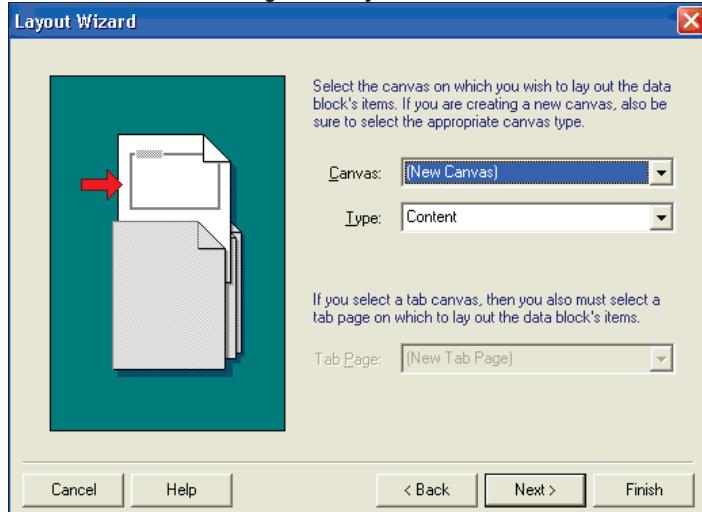


Figure 12: Layout Wizard, canvas type selection

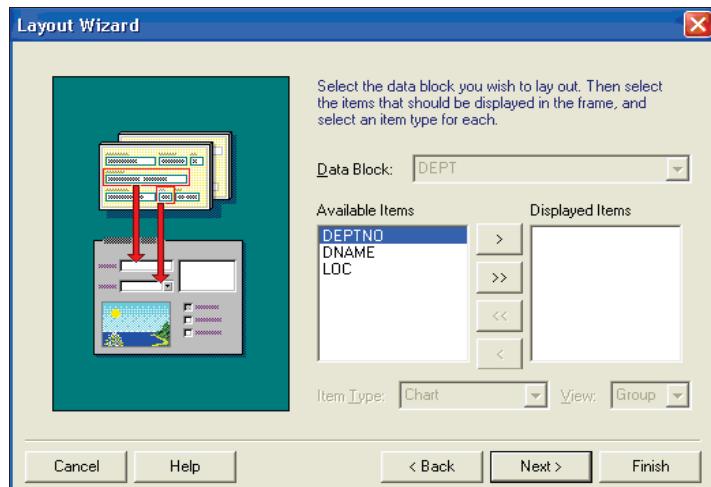


Figure 13: Layout Wizard, column selection

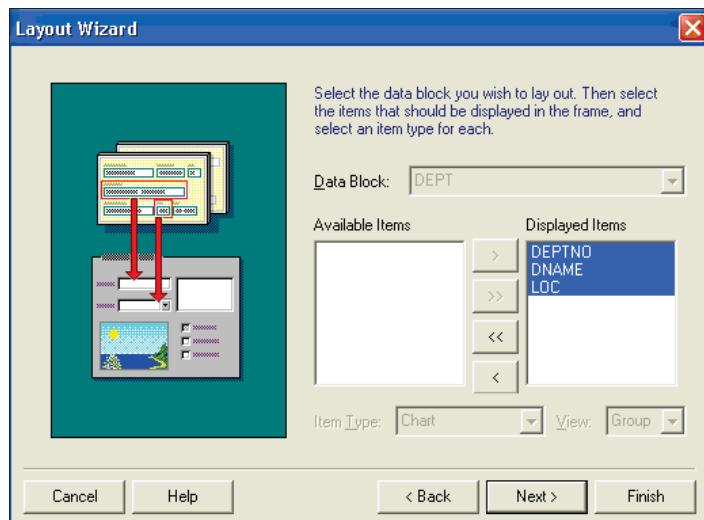


Figure 14: Layout Wizard, column selection

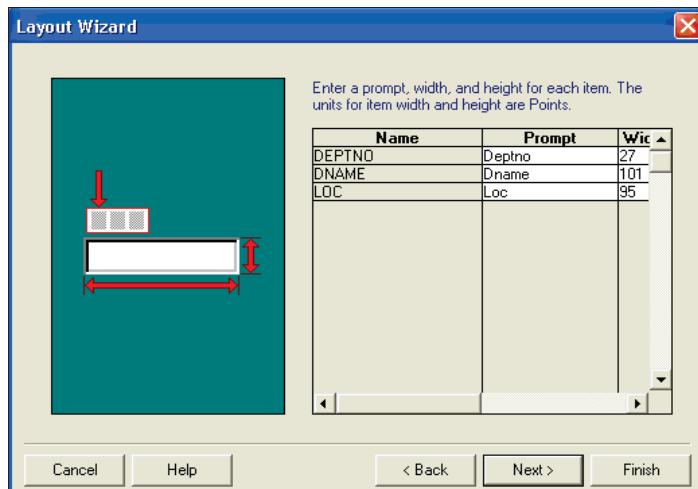


Figure 15: Layout Wizard, column selection

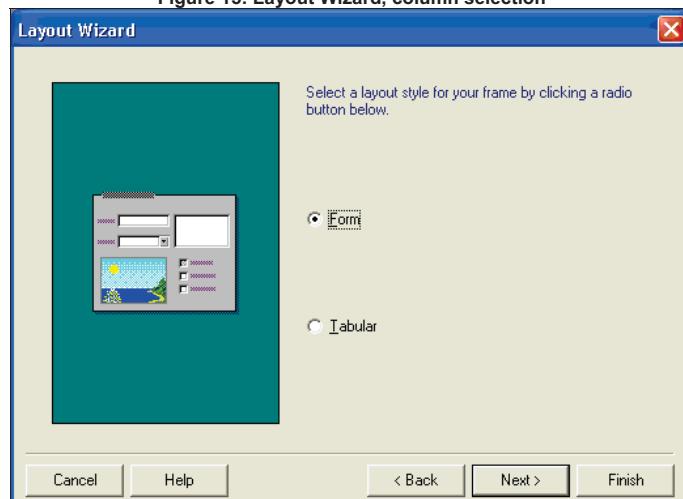


Figure 16: Layout Wizard, layout type selection

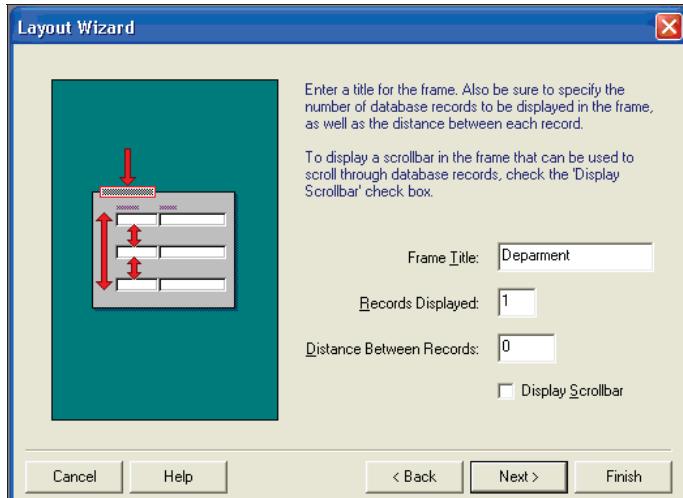


Figure 17: Layout Wizard

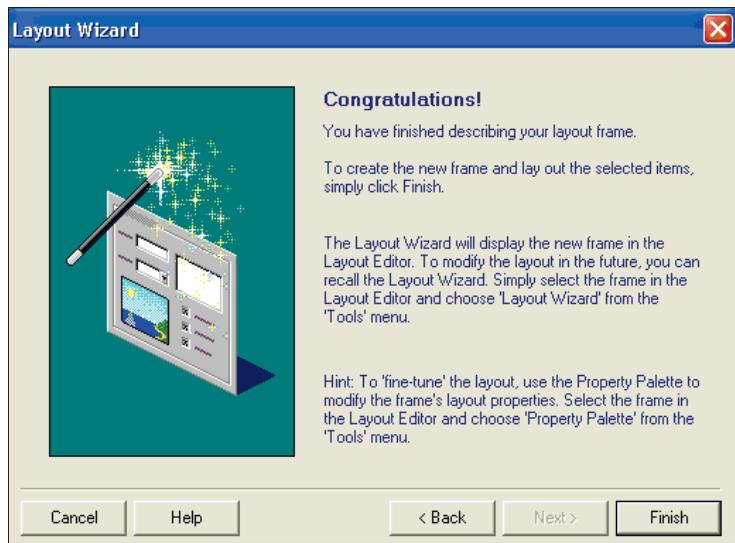
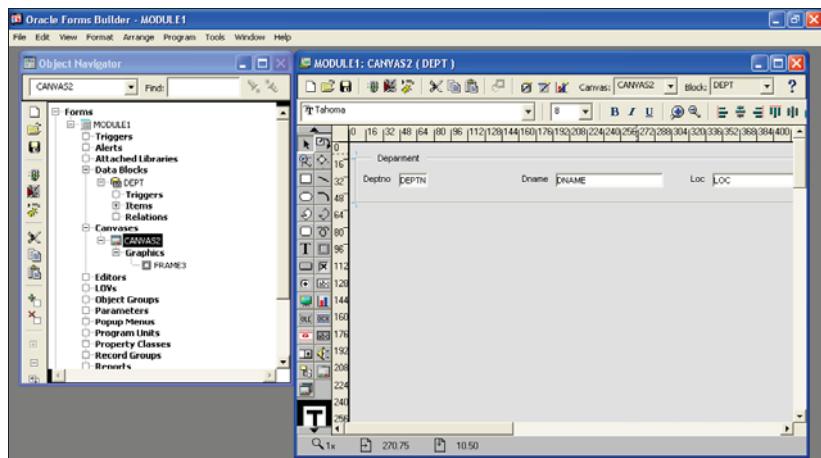
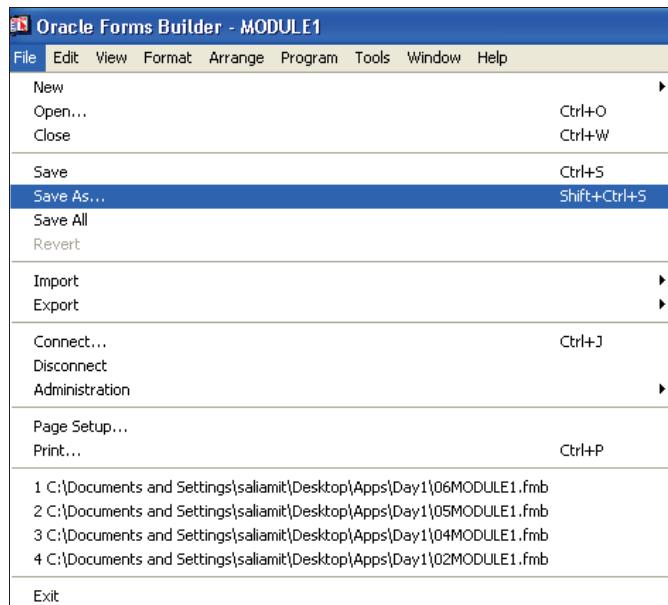


Figure 18: Layout Wizard



**Figure 19: Form Builder Application****Step 2: Saving the form**by selecting **File → Save As****Figure 20: Saving Form**

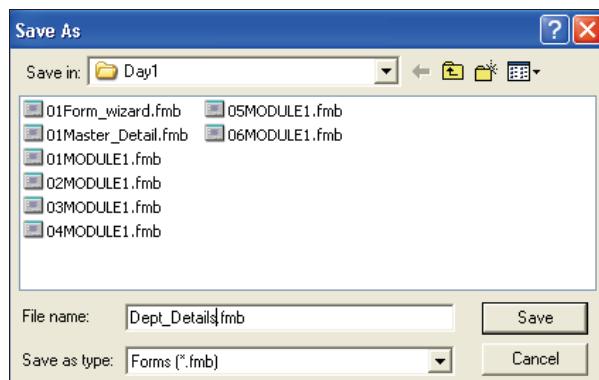


Figure 21: Saving Form

### Step 3: Executing the form

By selecting clicking icon mention in screen shot

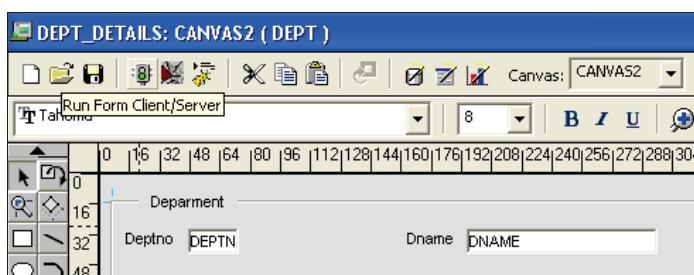


Figure 22: Executing form

Or

By selecting Program → Run Form→ Client Server

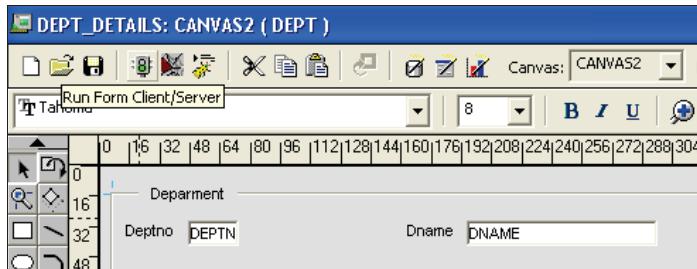


Figure 23: Executing form

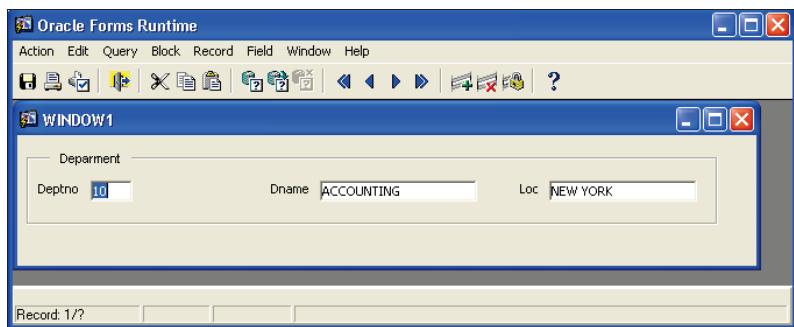


Figure 24: Executed form output

- 1.2. Create a form which displays all the items and records of salgrade table.(To-do)
- 1.3. Create a form which displays empno, sal, ename, deptno in emp table.(To-do)

## Lab 2. Wizard base Master- Child Form creation

- |       |                                                                                           |
|-------|-------------------------------------------------------------------------------------------|
| Goals | <ul style="list-style-type: none"><li>Create a Master- Child form using wizard.</li></ul> |
| Time  | 2.5 hr                                                                                    |



### 2.1: Create a form

The steps in this lab will walk you through the wizard for creating Master Child form.

You will have to do the following tasks:

1. Create a Master form using
2. Create a Child form using
3. Saving the form
4. Executing the form

#### Solution:

##### Step 1: Creating Master Form

Follow Set 1 of lab 1 to create master form.

##### Step 2: Creating Master Form

Follow Set 1 of lab 1 to create master form.

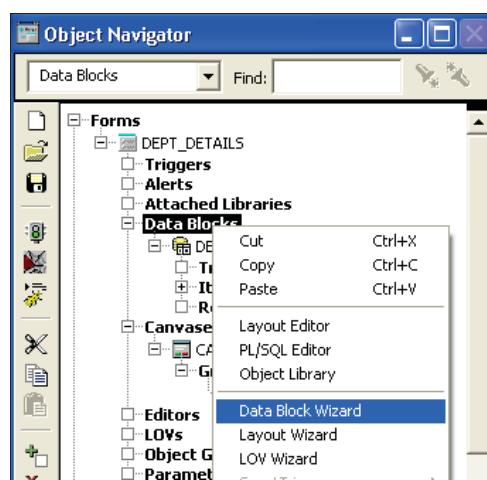


Figure 25: All Programs menu

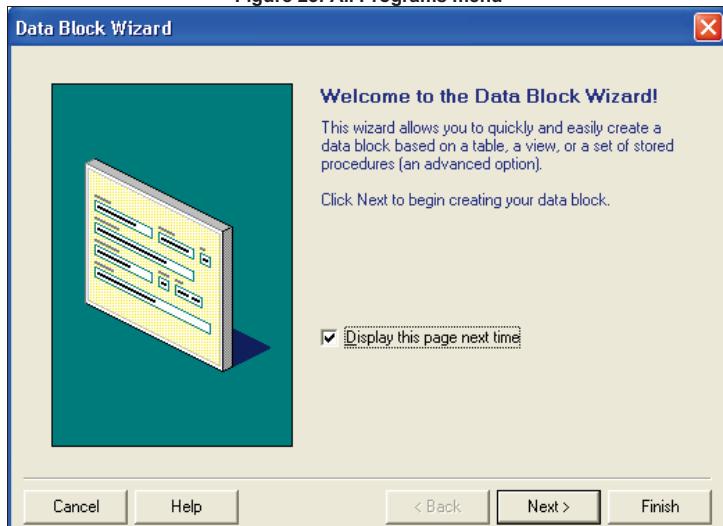


Figure 26: Data Block wizard



Figure 27: Data Block wizard

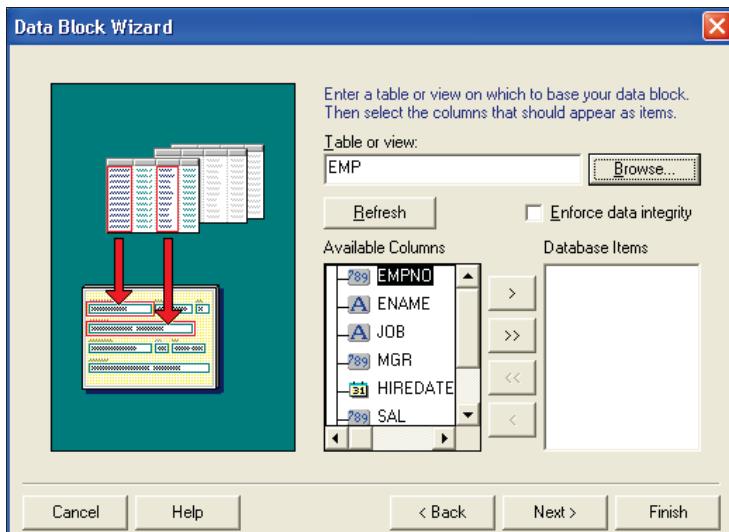


Figure 28: Data Block wizard

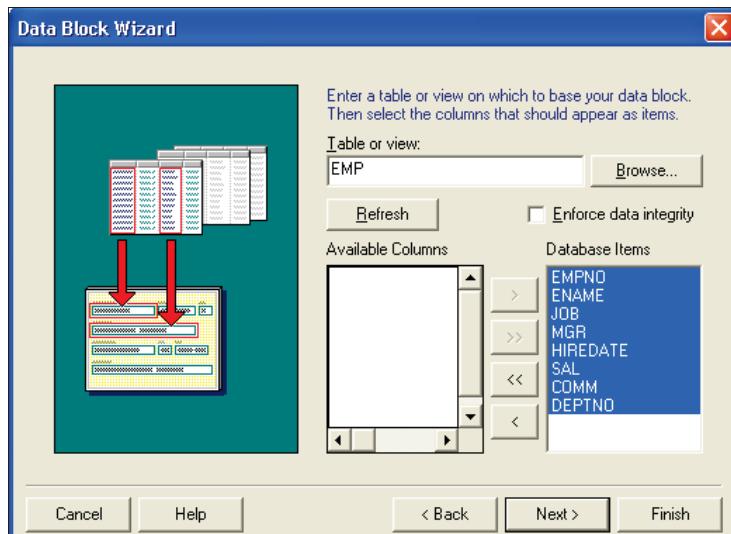


Figure 29: Data Block wizard

Click on Create relationship:

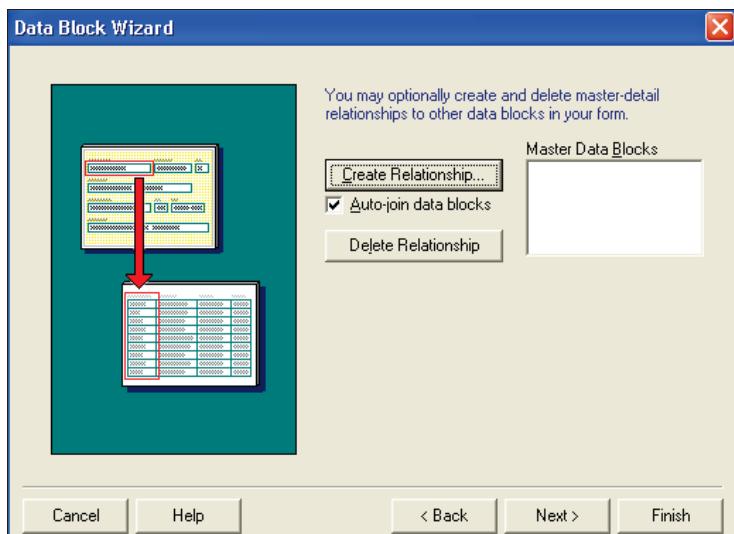


Figure 30: Data Block wizard, creating relationship between master Detail



Figure 31: Data Block wizard, creating relationship between master Detail

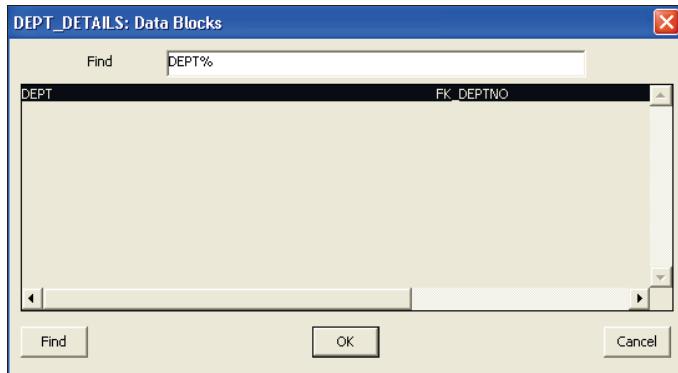


Figure 32: Data Block wizard, creating relationship between master Detail

So, we have created relationship between master block and child block using key Deptno.

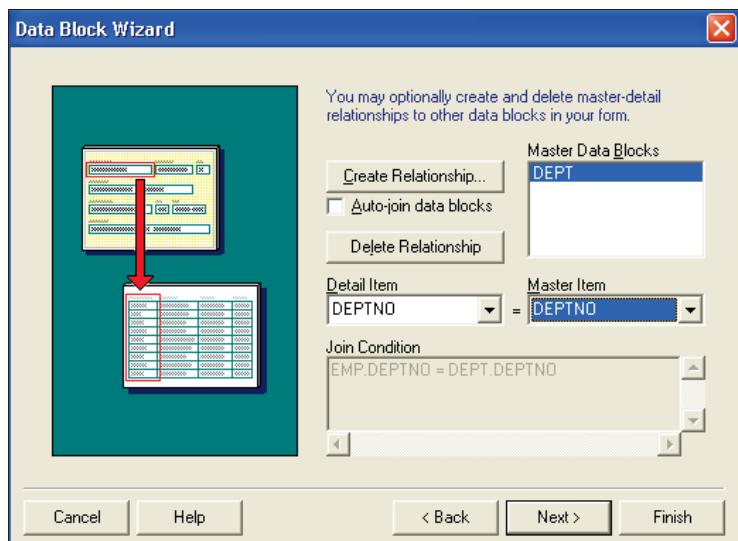


Figure 33: Data Block wizard, creating relationship between master Detail



Figure 34: Layout wizard

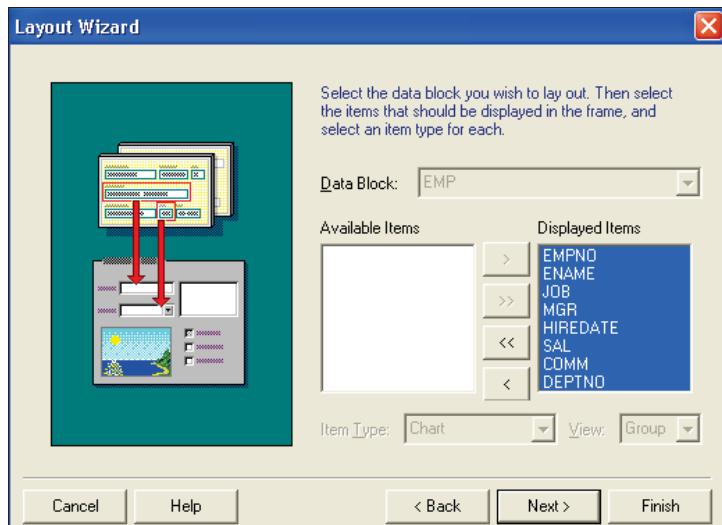


Figure 35: Layout wizard

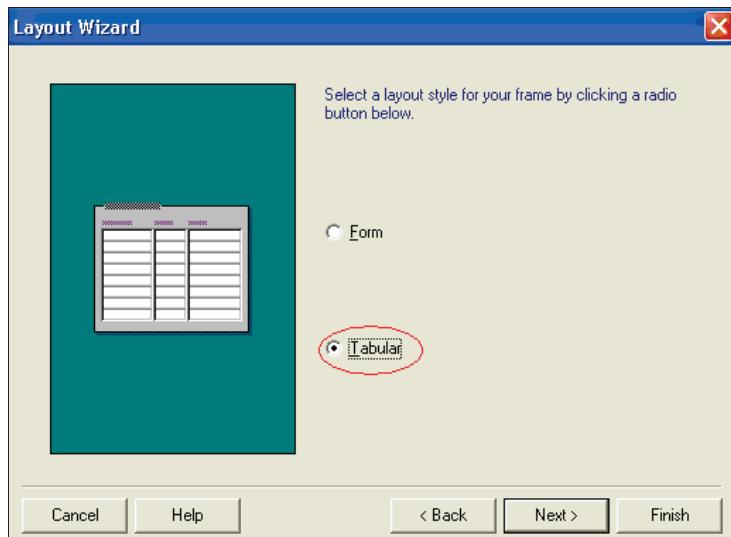


Figure 36: Layout wizard, select table as a layout

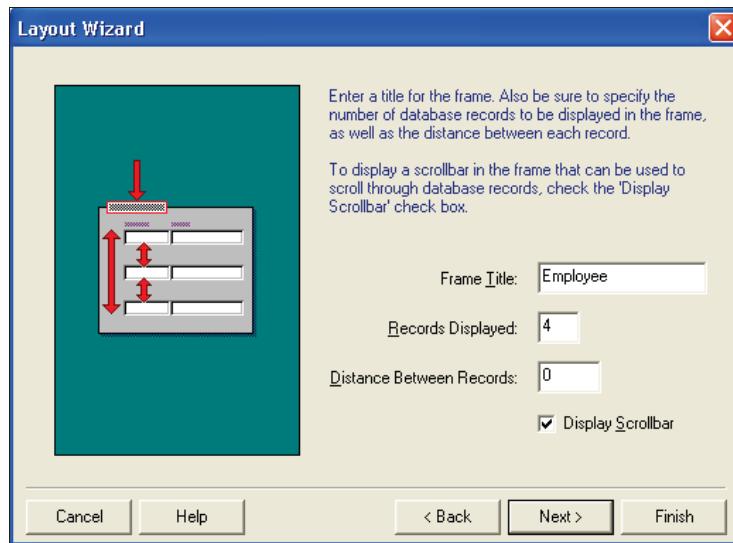


Figure 37: Layout wizard

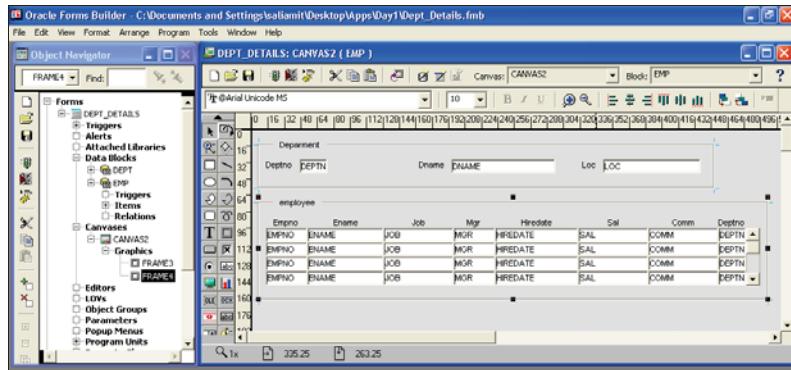


Figure 38: Report Builder

## Step 2: Saving the form

Same as lab 1 steps, Save the form as Master\_Details.fmb

**Step 3: Executing the form**

Same as lab 1 steps.

The Output is,

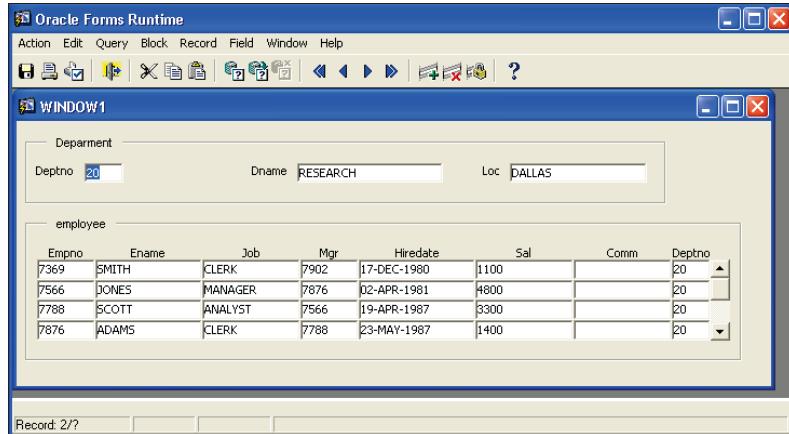


Figure 39: Master- Detail Form Output

**2.2. Create a form which shows all items and records of both emp and salgrade tables. (To-do)**

## Lab 3. Working With Items

|       |                                                                                              |
|-------|----------------------------------------------------------------------------------------------|
| Goals | <ul style="list-style-type: none"> <li>Working with various items present in Form</li> </ul> |
| Time  | 2.5 hr                                                                                       |

### 3.1: Create a form

The steps in this lab will walk you through the wizard for creating form.

You will have to do the following tasks:

1. Create a using wizard
2. Saving the form
3. Executing the form

**Solution:**

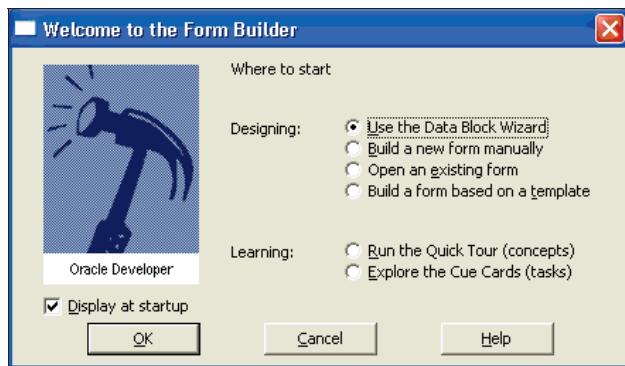
#### Step 1: Data block and Layout wizard:

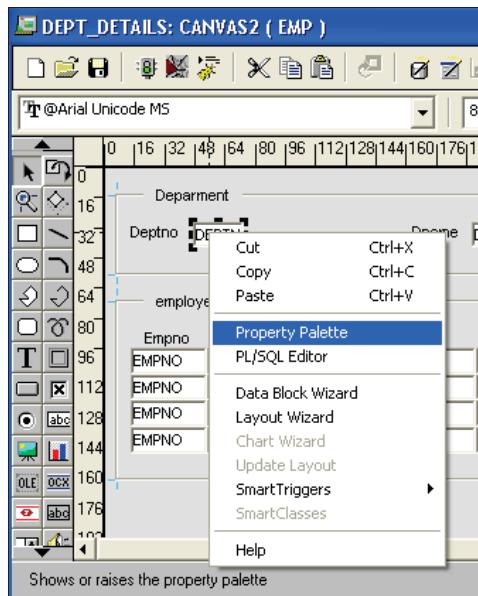
Open form 6i Application, by selecting **Start → Programs → Oracle Forms 6i – OraHome → Form Builder**.



**Figure 40:** All Programs menu

**Step 2:** Select Data block wizard and press OK.

**Figure 41:** Start of data block wizard



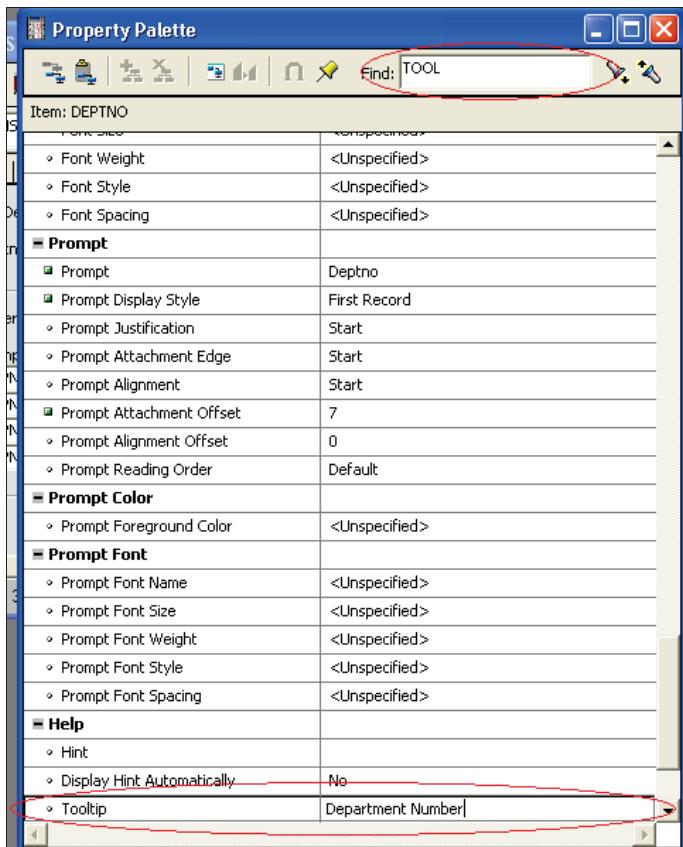
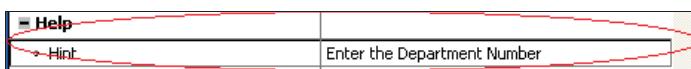


Figure 42: Data block wizard



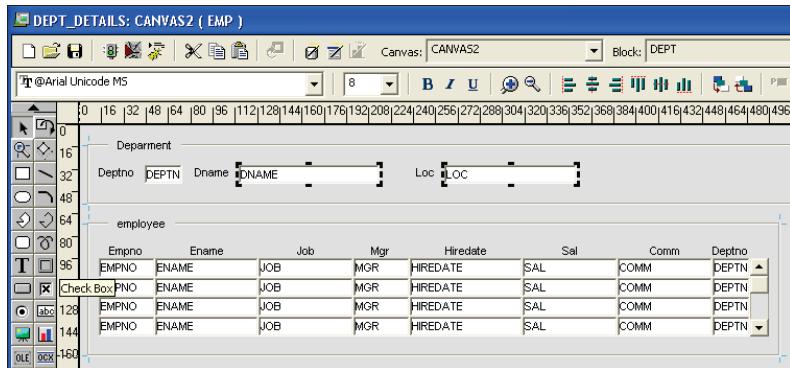


Figure 43: Selecting check box

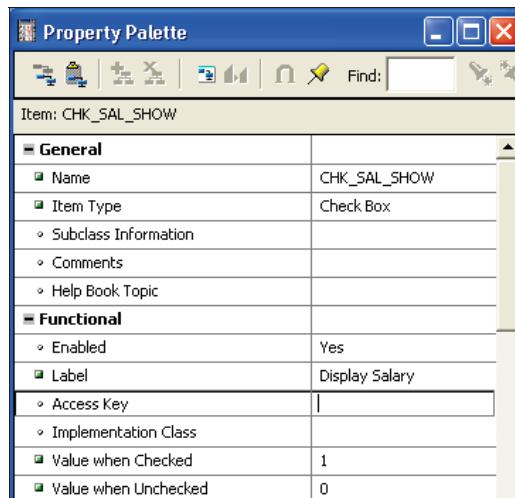
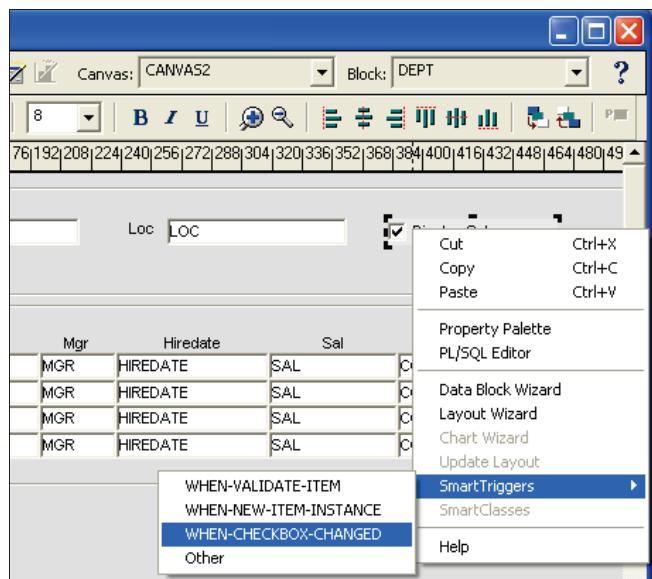


Figure 44: Property palette for check box item selected

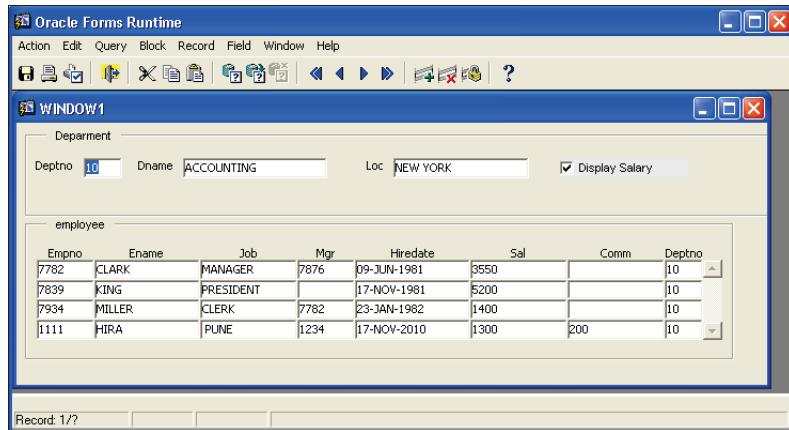
Say Database Item as No



```
begin
if (:dept.chk_sal_show = 1) then
 Set_Item_Property('emp.sal',VISIBLE,PROPERTY_TRUE);
else
 Set_Item_Property('emp.sal',VISIBLE,PROPERTY_FALSE);
end if;
end;

compile
```

Run

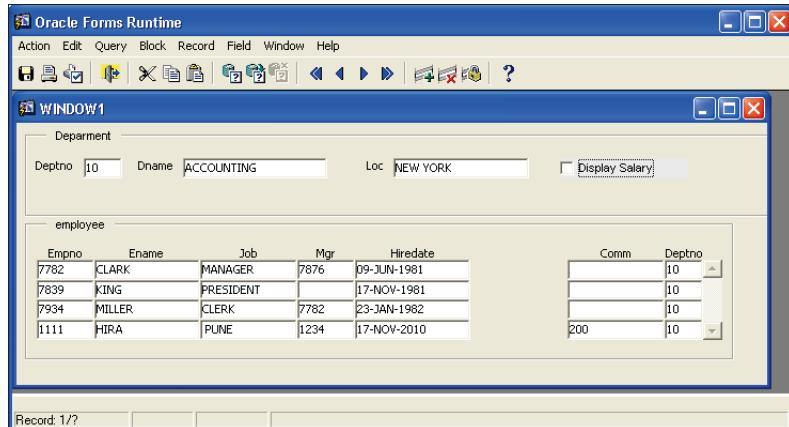


This screenshot shows the Oracle Forms Runtime interface. The main window title is "Oracle Forms Runtime". The menu bar includes Action, Edit, Query, Block, Record, Field, Window, Help. The toolbar contains various icons for file operations like Open, Save, Print, and Database. The window title is "WINDOW1". Inside, there's a "Department" section with fields for Deptno (10), Dname (ACCOUNTING), Loc (NEW YORK), and a checked checkbox labeled "Display Salary". Below this is an "employee" section containing a grid of data:

| Empno | Ename  | Job       | Mgr  | Hiredate    | Sal  | Comm | Deptno |
|-------|--------|-----------|------|-------------|------|------|--------|
| 7782  | CLARK  | MANAGER   | 7876 | 09-JUN-1981 | 3550 |      | 10     |
| 7839  | KING   | PRESIDENT |      | 17-NOV-1981 | 5200 |      | 10     |
| 7934  | MILLER | CLERK     | 7782 | 23-JAN-1982 | 1400 |      | 10     |
| 1111  | HIRA   | PUNE      | 1234 | 17-NOV-2010 | 1300 | 200  | 10     |

At the bottom, a status bar shows "Record: 1/?".

Figure 45: Execution of form with check box item checked



This screenshot shows the same Oracle Forms Runtime interface as Figure 45, but with a different state for the "Display Salary" checkbox. The checkbox is now unchecked. The rest of the interface and data grid remain the same.

Figure 46: Execution of form with check box item unchecked

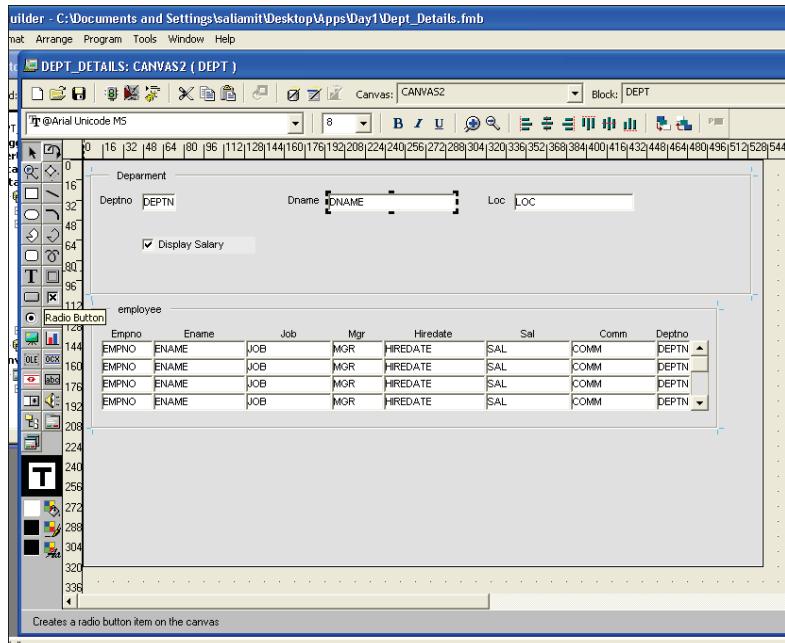
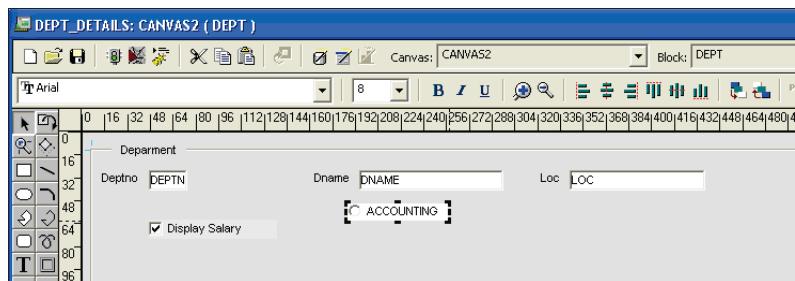


Figure 47: Selecting radio-group item



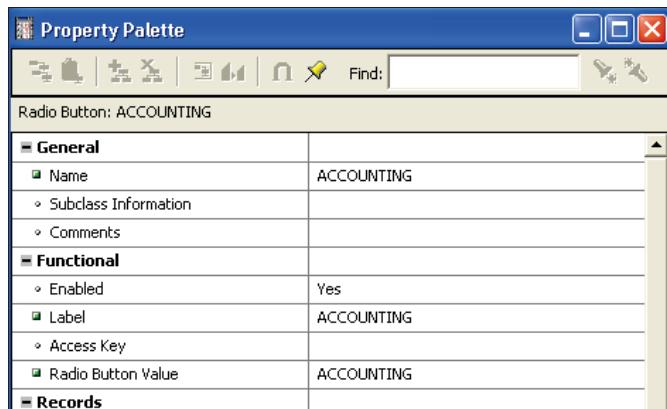


Figure 48: Property palette for the radio item selected



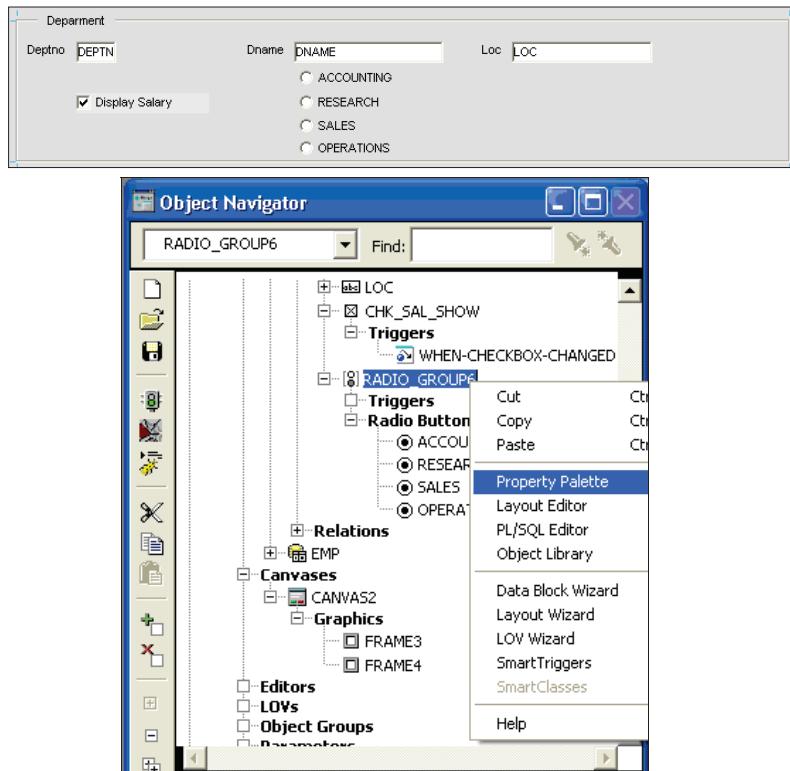


Figure 49: Selecting property palette of the radio group

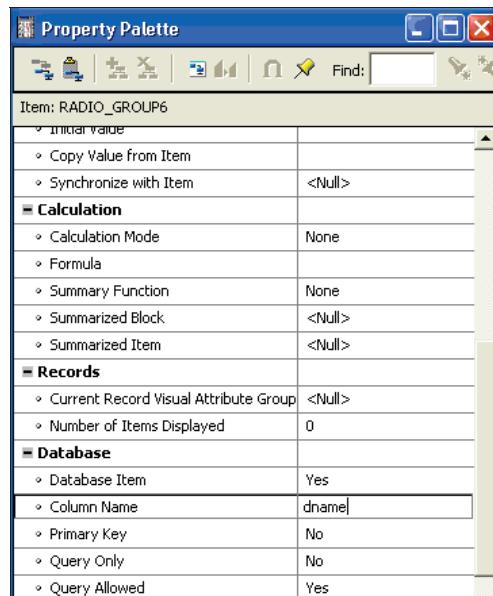


Figure 50: Property palette of radio group

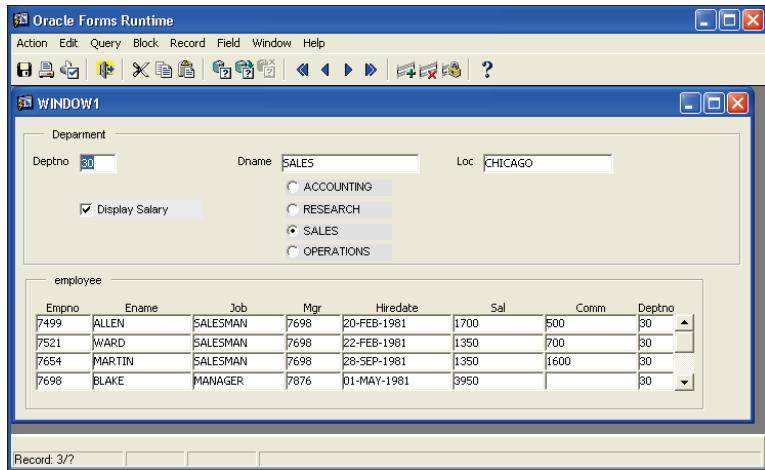


Figure 51: Execution of form with the radio group created

## BUTTONS

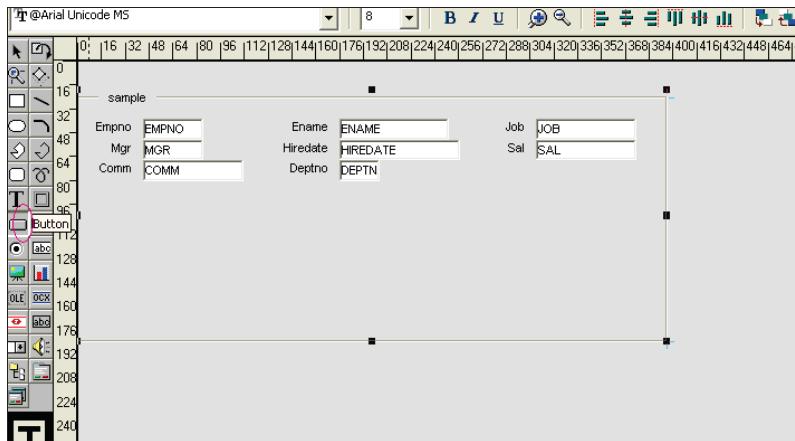


Figure 52: Selection of button

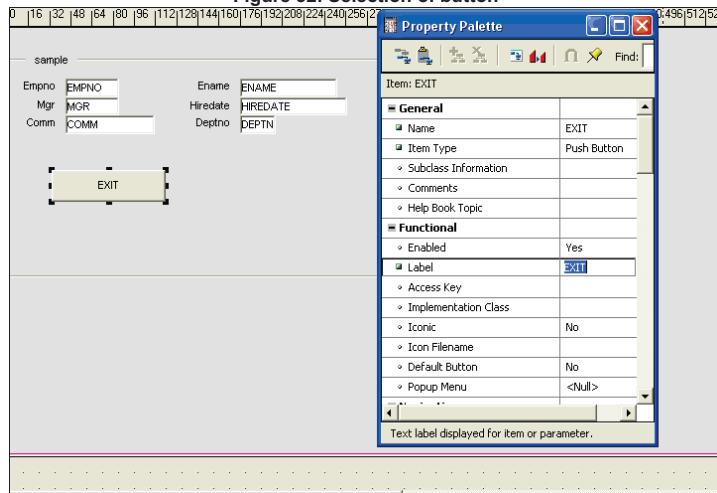


Figure 53: Property palette of button selected

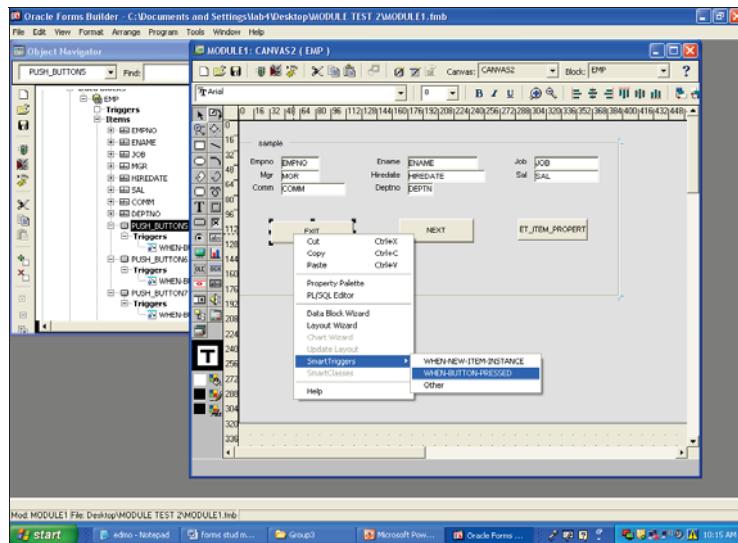


Figure 54: Selection of when button pressed trigger

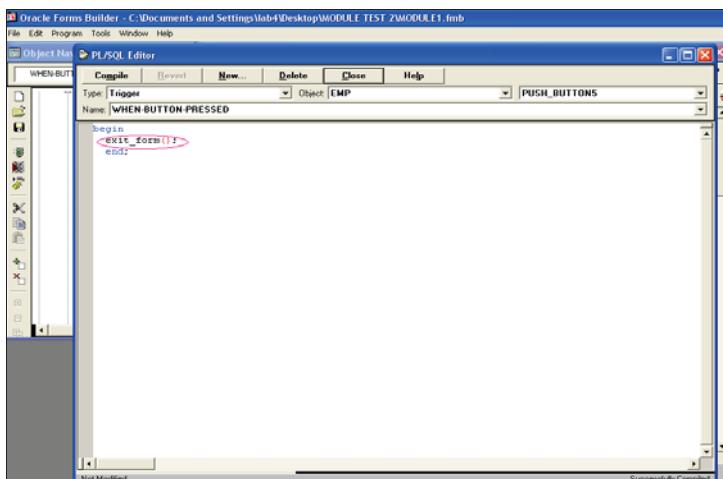


Figure 55: PL/SQL Code for when button pressed trigger

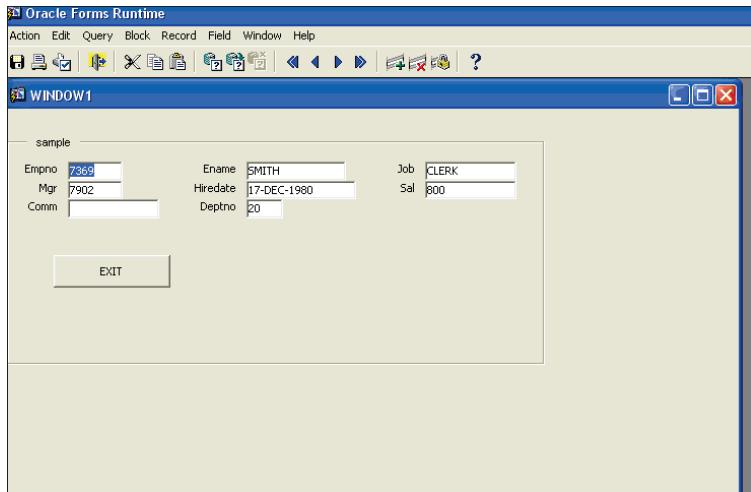
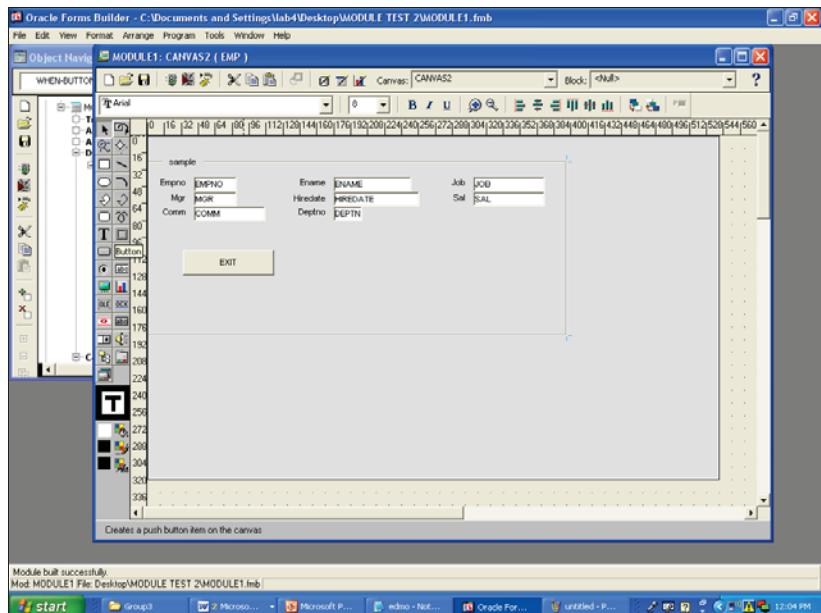


Figure 56: Execution of form with button



Create buttons to navigate between records. (To Do)  
Create a button which will save a new record. (To Do)

## Lab 4. Summary and Formula functions

|       |                                                                                                         |
|-------|---------------------------------------------------------------------------------------------------------|
| Goals | <ul style="list-style-type: none"><li>Create a form which uses summary and formula functions.</li></ul> |
| Time  | 2.5 hr                                                                                                  |

### 4.1: Calculate the sum of salaries of a department

Solution:

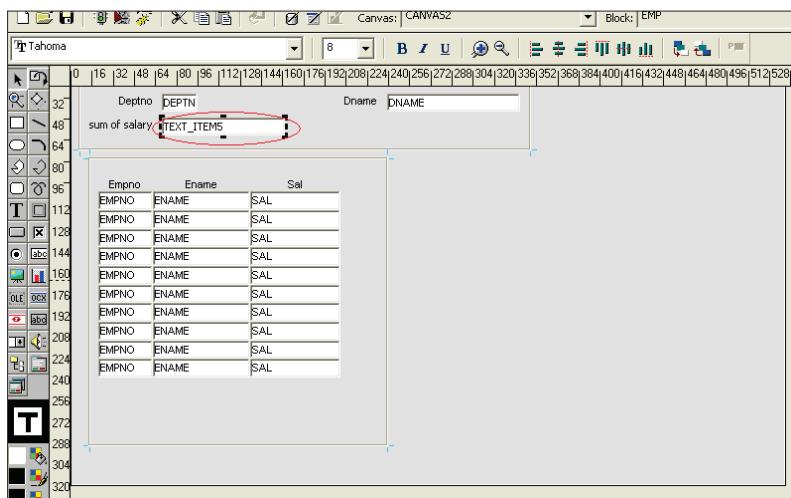


Figure 57: Text item selected which displays the total

| Item: TEXT_ITEMS                                              |         |
|---------------------------------------------------------------|---------|
| <b>* General</b>                                              |         |
| <b>* Functional</b>                                           |         |
| <b>* Navigation</b>                                           |         |
| <b>* Data</b>                                                 |         |
| <b>■ Calculation</b>                                          |         |
| <input checked="" type="checkbox"/> Calculation Mode          | Summary |
| » Formula                                                     |         |
| <input checked="" type="checkbox"/> Summary Function          | Sum     |
| <input checked="" type="checkbox"/> Summarized Block          | EMP     |
| <input checked="" type="checkbox"/> Summarized Item           | SAL     |
| <b>■ Records</b>                                              |         |
| » Current Record Visual Attribute Group                       | <Null>  |
| <input checked="" type="checkbox"/> Distance Between Records  | 14      |
| <input checked="" type="checkbox"/> Number of Items Displayed | 1       |
| <b>* Database</b>                                             |         |
| <b>* List of Values (LOV)</b>                                 |         |
| <b>* Editor</b>                                               |         |
| <b>* Physical</b>                                             |         |
| <b>* Visual Attributes</b>                                    |         |
| <b>* Color</b>                                                |         |

Figure 58: Property palette for the above selected text item

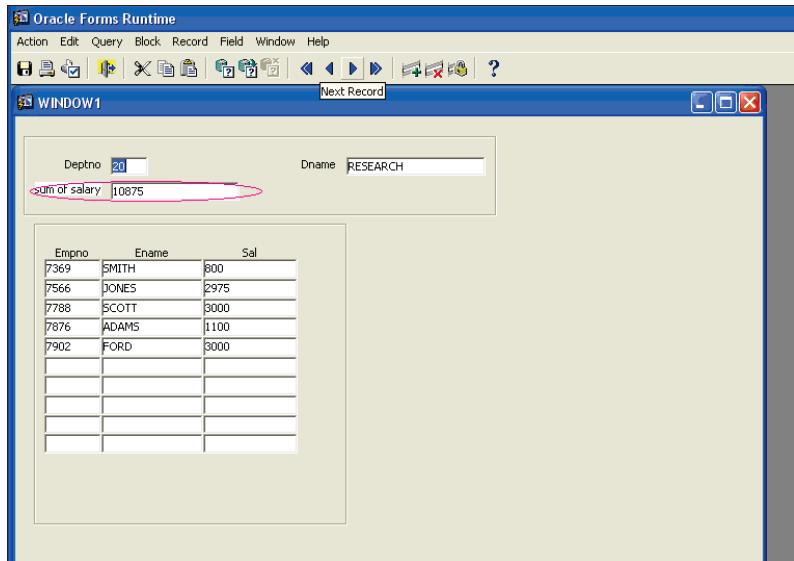


Figure 59: execution of the form shows the total

- 4.2. Calculate the maximum and minimum salaries of each department.(To Do)
- 4.3. Add commission to employees who presently has null commission where the commission should be 5% of salary.(To Do)

## Lab 5. Triggers

|       |                                                                                      |
|-------|--------------------------------------------------------------------------------------|
| Goals | <ul style="list-style-type: none"> <li>Create a form which uses triggers.</li> </ul> |
| Time  | 2.5 hr                                                                               |

The following example shows how to use a button pressed trigger.

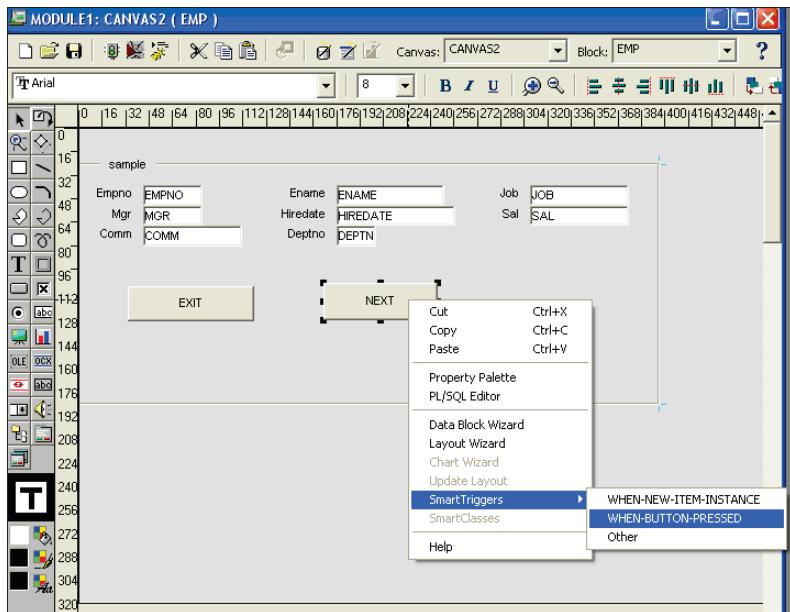


Figure 60: Selecting PL/SQL editor window for when button pressed trigger

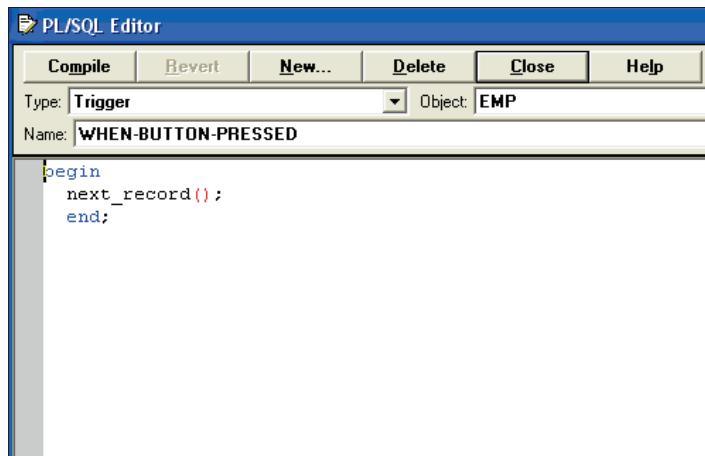


Figure 61: PL/SQL editor for when button pressed trigger

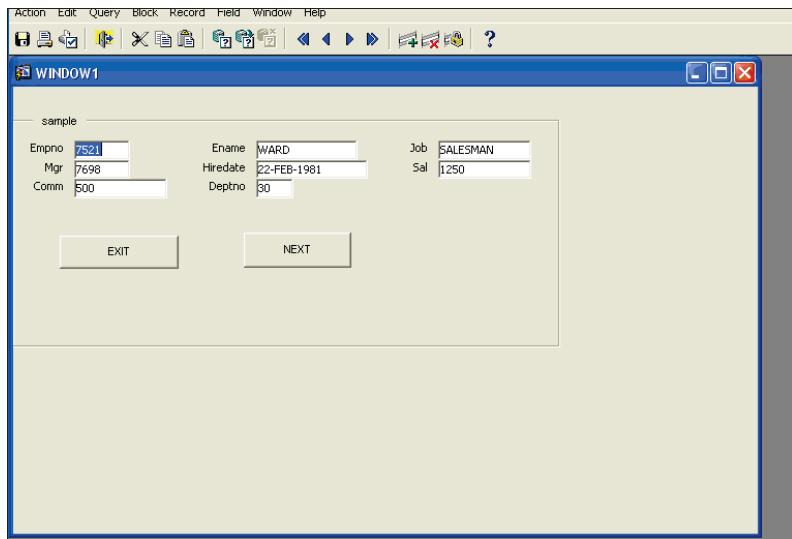


Figure 62: Execution of the form

**Built-in trigger:**

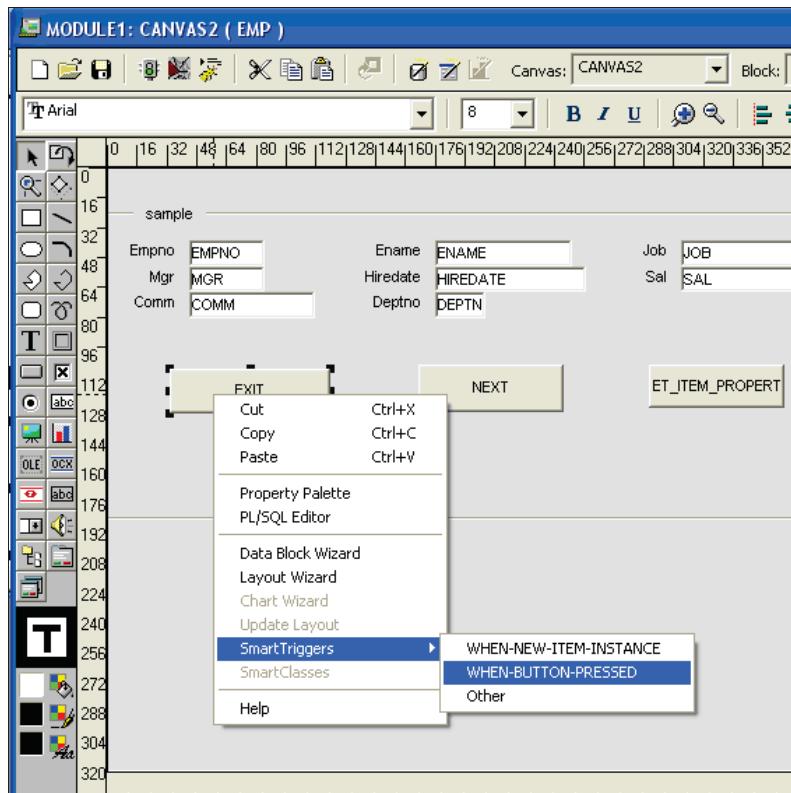
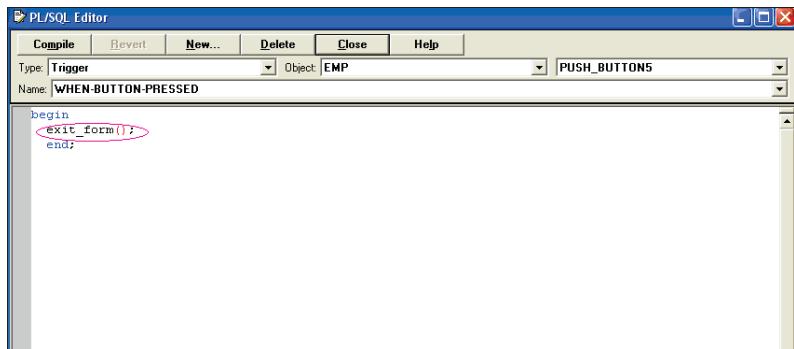


Figure 63: Selecting PL/SQL editor window for when button pressed trigger



The screenshot shows a PL/SQL Editor window with the following details:

- Toolbar: Compile, Revert, New..., Delete, Close, Help.
- Menu: Type (Trigger), Object (EMP), Name (PUSH\_BUTTON5).
- Code area:

```
begin
 exit_form();
end;
```

The line `exit_form();` is highlighted with a red oval.

To-do:

- 5.1. Create a form which shows details of a particular department by entering the dept no.
- 5.2. Create a form with a search button which while pressed will automatically execute the codes.

## Lab 6. Validation

|       |                                                                                        |
|-------|----------------------------------------------------------------------------------------|
| Goals | <ul style="list-style-type: none"> <li>Create a form which uses validation.</li> </ul> |
| Time  | 2.5 hr                                                                                 |

The following example validates the primary key constraint of employee number in employee table

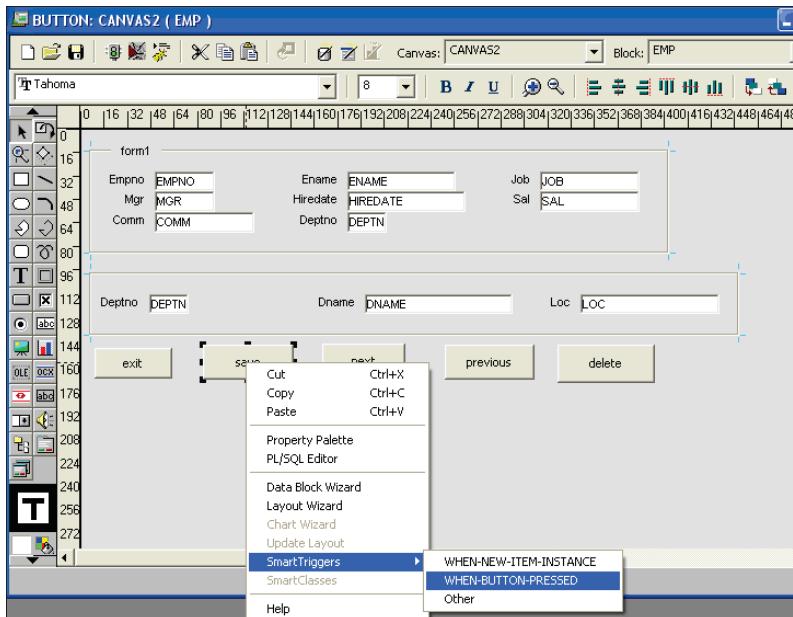


Figure 64: Selecting PL/SQL editor window for when button pressed trigger

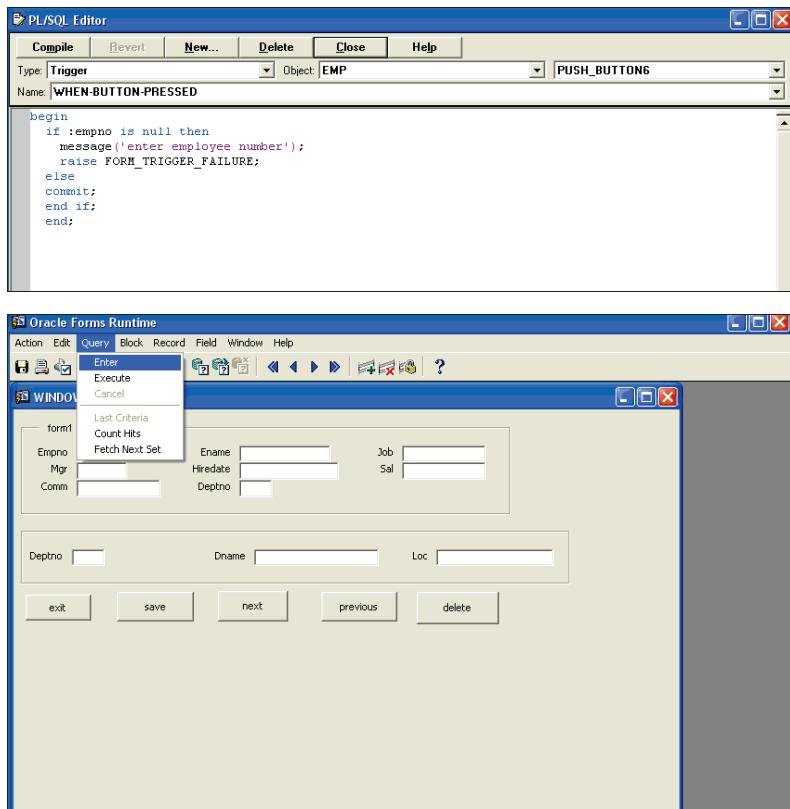
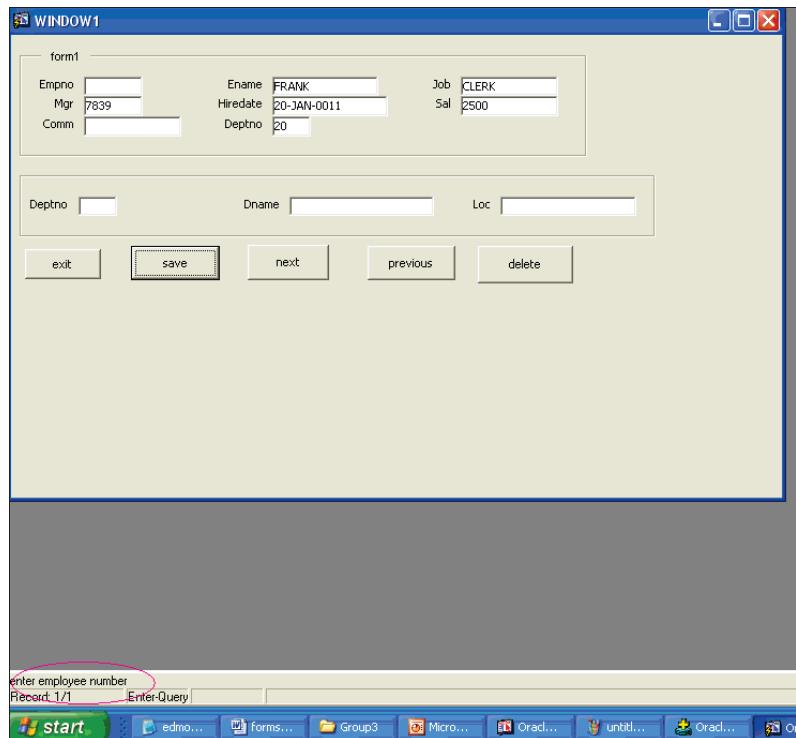


Figure 65: Execution window of the form



6.1. Insert record and show error message if hiredate is higher than sysdate. (To Do)

## Appendices

### Appendix A: Table of Figures

|                                                                                |           |
|--------------------------------------------------------------------------------|-----------|
| Figure 1: All Programs menu .....                                              | 9         |
| Figure 2: Start of data block wizard .....                                     | 9         |
| Figure 3: Data block wizard .....                                              | 10        |
| Figure 4: Data block wizard .....                                              | 10        |
| Figure 5: Data block wizard .....                                              | 11        |
| Figure 6: Data block wizard, Database connectivity.....                        | 11        |
| Figure 7: Data block wizard, Table selection.....                              | 12        |
| Figure 8: Data block wizard, column selection .....                            | 13        |
| Figure 9: Data block wizard, column selection .....                            | 13        |
| Figure 10: Data block wizard .....                                             | 14        |
| Figure 11: Layout Wizard .....                                                 | 15        |
| Figure 12: Layout Wizard, canvas type selection.....                           | 15        |
| Figure 13: Layout Wizard, column selection .....                               | 16        |
| Figure 14: Layout Wizard, column selection .....                               | 17        |
| Figure 15: Layout Wizard, column selection .....                               | 17        |
| Figure 16: Layout Wizard, layout type selection.....                           | 18        |
| Figure 17: Layout Wizard .....                                                 | 18        |
| Figure 18: Layout Wizard .....                                                 | 19        |
| Figure 19: Form Builder Application .....                                      | 20        |
| Figure 20: Saving Form .....                                                   | 20        |
| Figure 21: Saving Form .....                                                   | 21        |
| Figure 22: Executing form .....                                                | 21        |
| Figure 23: Executing form .....                                                | 22        |
| Figure 24: Executed form output .....                                          | 22        |
| Figure 25: All Programs menu .....                                             | 24        |
| Figure 26: Data Block wizard.....                                              | 24        |
| Figure 27: Data Block wizard.....                                              | 25        |
| Figure 28: Data Block wizard.....                                              | 26        |
| Figure 29: Data Block wizard.....                                              | 27        |
| Figure 30: Data Block wizard, creating relationship between master Detail..... | 28        |
| Figure 31: Data Block wizard, creating relationship between master Detail..... | 28        |
| Figure 32: Data Block wizard, creating relationship between master Detail..... | 29        |
| Figure 33: Data Block wizard, creating relationship between master Detail..... | 30        |
| Figure 34: Layout wizard .....                                                 | 30        |
| Figure 35: Layout wizard .....                                                 | 31        |
| Figure 36: Layout wizard, select table as a layout .....                       | 32        |
| Figure 37: Layout wizard .....                                                 | 33        |
| Figure 38: Report Builder .....                                                | 33        |
| Figure 39: Master- Detail Form Output.....                                     | 34        |
| Figure 40: All Programs menu .....                                             | 36        |
| Figure 41: Start of data block wizard .....                                    | 36        |
| Figure 42: Data block wizard .....                                             | 38        |
| <b>Figure 43: Selecting check box.....</b>                                     | <b>39</b> |
| <b>Figure 44: Property palette for check box item selected .....</b>           | <b>39</b> |

|                                                                                 |    |
|---------------------------------------------------------------------------------|----|
| Figure 45: Execution of form with check box item checked.....                   | 41 |
| Figure 46: Execution of form with check box item unchecked .....                | 41 |
| Figure 47: Selecting radio-group item.....                                      | 42 |
| Figure 48: Property palette for the radio item selected.....                    | 43 |
| Figure 49: Selecting property palette of the radio group .....                  | 44 |
| Figure 50: Property palette of radio group .....                                | 45 |
| Figure 51: Execution of form with the radio group created.....                  | 46 |
| Figure 52: Selection of button.....                                             | 47 |
| Figure 53: Property palette of button selected.....                             | 47 |
| Figure 54: Selection of when button pressed trigger .....                       | 48 |
| Figure 55: PL/SQL Code for when button pressed trigger.....                     | 49 |
| Figure 56: Execution of form with button .....                                  | 49 |
| Figure 57: Text item selected which displays the total .....                    | 51 |
| Figure 58: Property palette for the above selected text item.....               | 52 |
| Figure 59: execution of the form shows the total .....                          | 53 |
| Figure 60: Selecting PL/SQL editor window for when button pressed trigger ..... | 54 |
| Figure 61: PL/SQL editor for when button pressed trigger.....                   | 55 |
| Figure 62: Execution of the form .....                                          | 55 |
| Figure 63: Selecting PL/SQL editor window for when button pressed trigger ..... | 56 |
| Figure 64: Selecting PL/SQL editor window for when button pressed trigger ..... | 58 |
| Figure 65: Execution window of the form .....                                   | 59 |