Oracle Designer 6*i*: New Features

Volume 1 • Student Guide

This course supports:
Oracle Designer 6i

Course Code 20002GC10

Prod: 1.0

M0-12378

Authors

Angela Asmussen
Patrice Daux
Susan Duncan
Jeff Gallus
Sue Harper
Pete Jamieson
Tim Joy
Gail Wingate

Technical Contributors and Reviewers

Dave Brown
Simon Day
Kate Heap
Lucas Jellema
Mark Pirie

All the members of the
Designer Product
Development Team

Publishers

Don Griffin
Avril Price-Budgen
Fiona Simpson

Restricted Rights Legend

The information in this document is subject to change without notice. If you find any problems in the documentation, please report them in writing to Education Products, Oracle Corporation, 500 Oracle Parkway, Box 659806, Redwood Shores, CA 94065. Oracle Corporation does not warrant that this document is error-free.

Contents

Lesson 1: Introduction

Lesson 2: Creating and Maintaining Workareas and Containers

Lesson 3: Storing Files in the Repository

Lesson 6: Java and the Server Generator

Lesson 7: Oracle Forms Generation: Enhancing the End User Interface

Lesson 8: Modeling LOVs

Lesson 9: Additional Forms Generation Functionality

Lesson 10: Investigating Web PL/SQL Generation

Lesson 11: Placing an Object under Version Control

Appendix A: Solutions

1

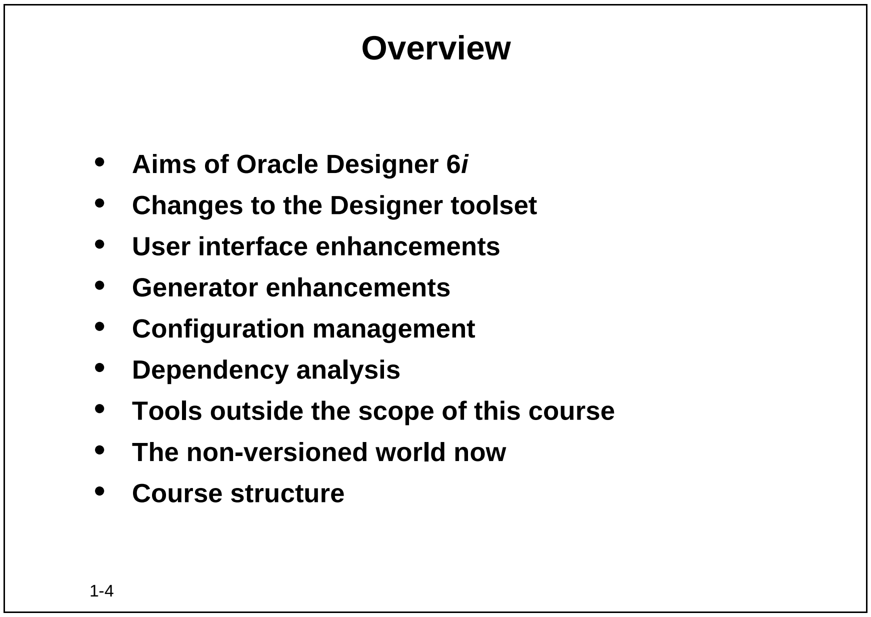
Introduction

Introduction

As applications become more complex and the number of users increase, there has been a growing need for the addition of configuration management to the Oracle Designer 6*i* product. To accommodate this need, Oracle Designer 6*i* incorporates significant changes to the repository.

This lesson gives an overview of the updated and restructured areas of Oracle Designer 6*i*. It also explains which areas are not covered in the course. Some topics are minor, generic user interface changes and these are covered in this lesson. Significant changes are addressed throughout the rest of the course.

Practices form an integral part of the course and are based on a complete business model.



Overview

- **Aims of Oracle Designer 6*i***
- **Changes to the Designer toolset**
- **User interface enhancements**
- **Generator enhancements**
- **Configuration management**
- **Dependency analysis**
- **Tools outside the scope of this course**
- **The non-versioned world now**
- **Course structure**

1-4

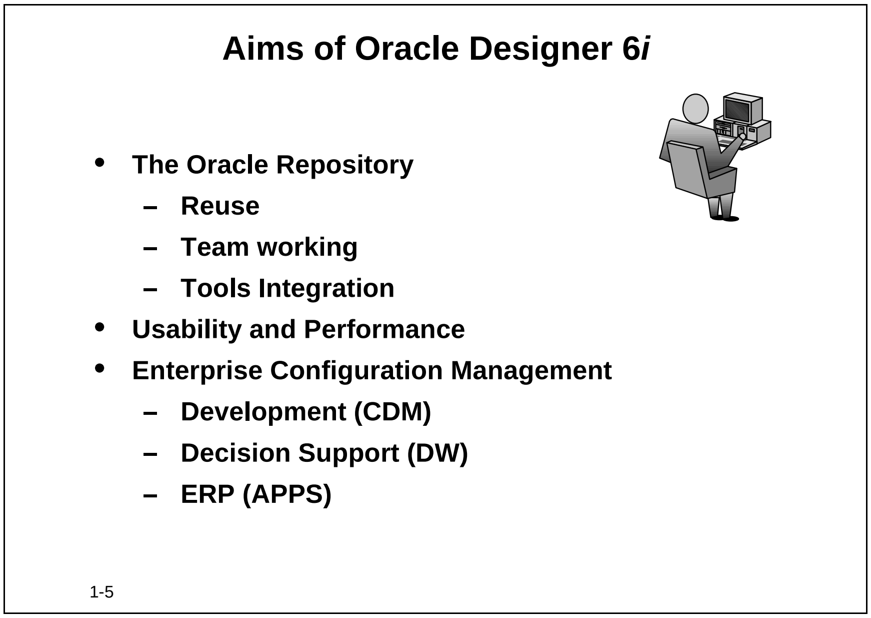
Course Objectives

At the end of this course, you should be able to:

- Create and use workareas within the Repository Object Navigator
- Store files in the repository
- Generate Oracle8*i* objects
- Enhance the layout of a generated form
- Create break reports and handle multi-row updates using the Web PL/SQL Generator
- Place objects under version control
- Modify versioned objects
- Create configurations of versioned objects
- Analyze object dependencies using the Dependency Manager

Aims of Oracle Designer 6*i*

Through integration with the Oracle Repository this major functional release adds significant enterprise configuration management capability.



Aims of Oracle Designer 6*i*

- **The Oracle Repository**
  - **Reuse**
  - **Team working**
  - **Tools Integration**
- **Usability and Performance**
- **Enterprise Configuration Management**
  - **Development (CDM)**
  - **Decision Support (DW)**
  - **ERP (APPS)**

1-5

A Shared Repository

The Oracle Repository will also be used by other internal groups, most notably the Common Warehouse Model (CWM) initiative within the Warehouse program office. This is aimed at integrating the various metadata sources within our business information and warehouse solutions.

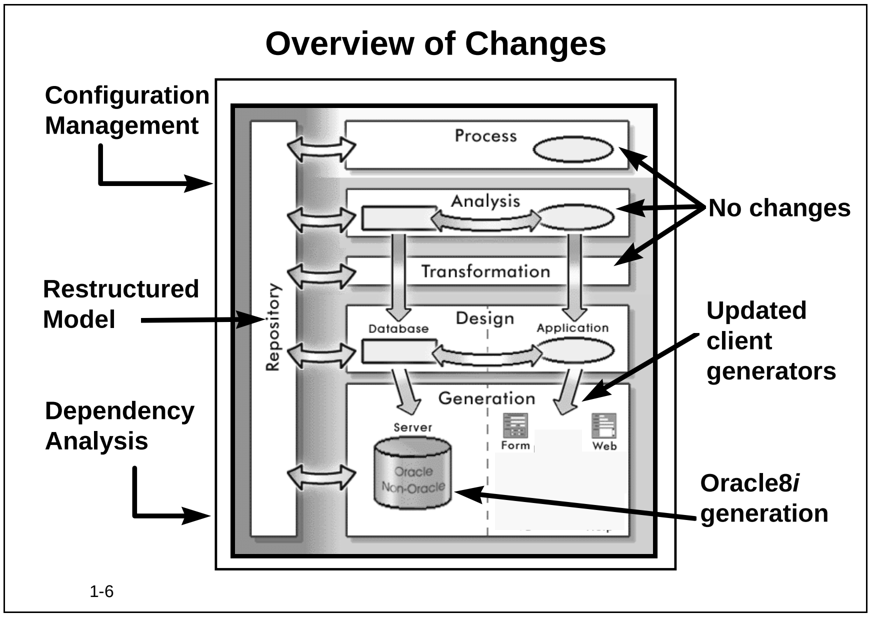
Usability and Performance

The new product set allows for software configuration management through fine grain versioning. In addition to storing the objects that the repository has always been able to handle, the new Oracle Repository functionality also allows files and folders, traditionally managed in the file system, to be managed and stored entirely within the repository.

Enterprise Configuration Management

Oracle Applications have started testing the technology with a view to adopting this as the standard configuration management solution.

Overview of Changes



Overview of Changes

Configuration Management

Restructured Model

Dependency Analysis

No changes

Updated client generators

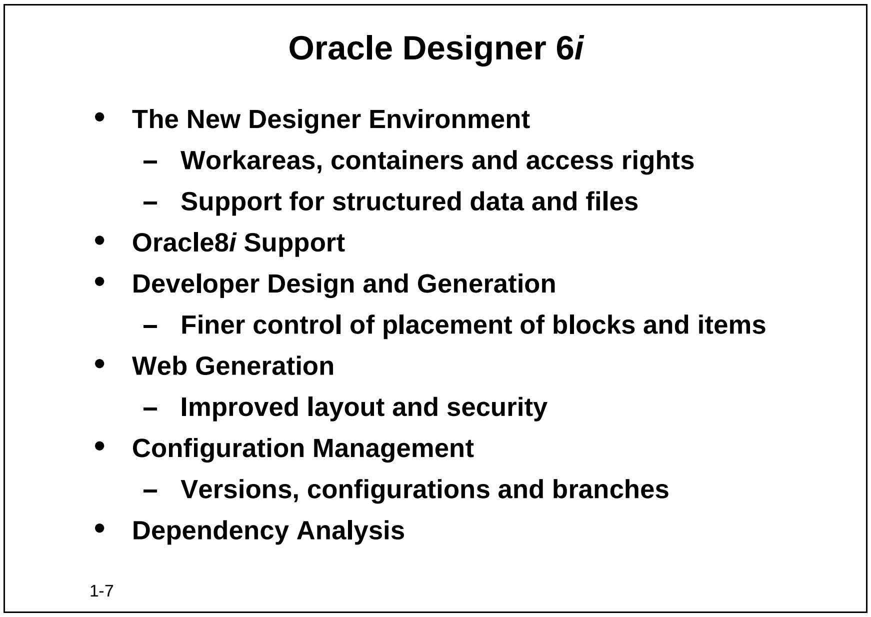
Oracle8*i* generation

1-6

Area of Oracle Designer 6*i*	Description of changes
Repository	The underlying model has been changed significantly to support configuration management.
	Dependency management has been significantly extended to support dependency analysis of repository data and files stored within the repository.
Client generation	The client generators have been modified to allow greater control over the end user interface.
Server generation	The server generators support generation for Oracle8 and Oracle8*i* objects.
Analysis Tools and Transformers	The analysis tools have had no functionality changes other than those required to support configuration management through workareas, and version control. A number of new matrixes have been added to the Matrix Diagrammer.

Topics Introduced During the Course

The Oracle Designer 6*i* toolset is vast and has many new and enhanced features.



Oracle Designer 6*i*

- **The New Designer Environment**
  - **Workareas, containers and access rights**
  - **Support for structured data and files**
- **Oracle8*i* Support**
- **Developer Design and Generation**
  - **Finer control of placement of blocks and items**
- **Web Generation**
  - **Improved layout and security**
- **Configuration Management**
  - **Versions, configurations and branches**
- **Dependency Analysis**

1-7

The course covers the most significant of the changes to the toolset, in particular:

- The Oracle Designer 6*i* Environment

  A significant change in the Oracle Designer 6*i* environment is the introduction of workareas and containers. Using containers, such as folders and application systems, you can store repository data and files in the repository.
- Oracle8*i* generation

  Oracle8*i* is central to the Oracle Internet Platform strategy, Oracle Designer 6*i* includes support for developing Oracle8*i* data servers.
- Form Generation

  The focus is on the changes to the Form generator that effect the generated user interface and at those affecting the underlying functionality.
- Web Generation

  The course looks at generating more flexible Web applications.
- Configuration Management

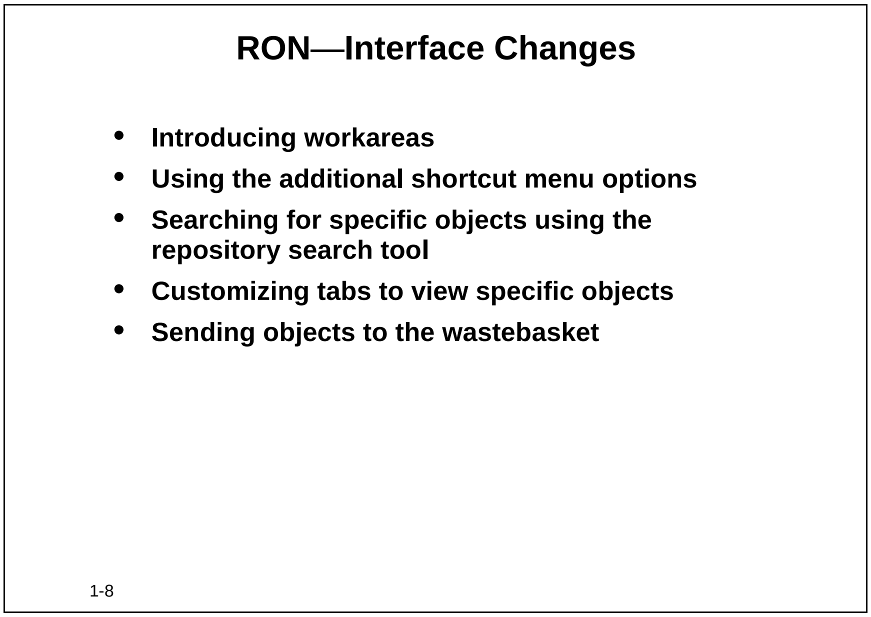
  You are introduced to different aspects of handling configuration management using the tool to do fine grain version control, create branches, compare and merge objects and to create and use configurations.

  The topics are dealt with in such a way as to serve as an introduction to those new to the subject area and as a starting point for those familiar to these topics.
- Dependency Management

  You look at the issues relating to determining the impact of changes during application development and tracking relationships.

Interface Changes

The introduction of workareas is a significant change to working in the repository. The concepts relating to workareas are covered in the next lesson. In addition to workareas, Oracle Designer 6*i* provides several new utilities for manipulating and viewing objects.



RON—Interface Changes

- **Introducing workareas**
- **Using the additional shortcut menu options**
- **Searching for specific objects using the repository search tool**
- **Customizing tabs to view specific objects**
- **Sending objects to the wastebasket**

1-8

Introducing Workareas

Workareas provide a view into containers and allow you to see and manipulate objects from more than one container at a time.

Shortcut Menu Options

Shortcut menus are available throughout the tool. These are restricted menus with options that are relevant to the object you have currently selected.

To invoke a shortcut menu, select an item and press the right mouse button. This displays a menu displaying the valid actions that can be performed for the selected item. We will refer to these as shortcut menus throughout the text.

Searching for Specific Objects

You use the Search Repository utility to perform searches for objects in a specific container, workarea, or across the entire repository. You can perform searches based on the characteristics of objects, such as:

- Type, for example, table definitions
- Name or short name
- Repository object identifier (IRID) or version identifier (IVID)

- Audit properties, such as date created or who created the object

The utility displays the SQL query executed to perform the search and the number of matches found.

Viewing Specific Object Types with Tabs

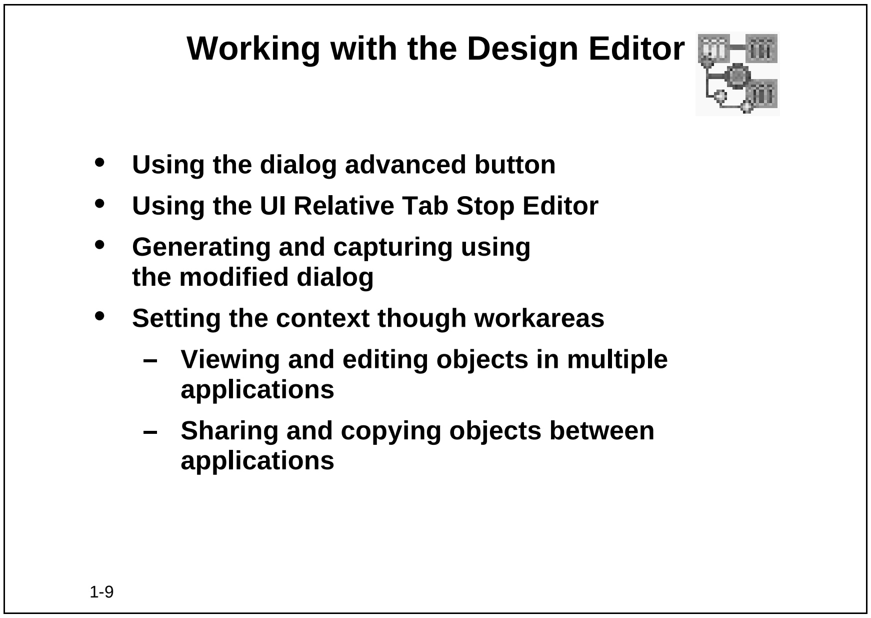
When you invoke the RON, the Navigator window groups objects under tabs that represent the different stages in the process of development. You can turn the tabs off or redefine your own groups.

Use **View –> Hide Group Tabs** to turn off the tabs. Use **Options –> Customize Navigator Groups** to modify the groups.

Wastebasket

Deleting objects does not remove them from the repository immediately. Instead, the objects are placed in a wastebasket and you can restore them or remove them completely from the repository. There is a separate wastebasket for each workarea in the repository.

Changes in the Design Editor



Working with the Design Editor

- Using the dialog advanced button
- Using the UI Relative Tab Stop Editor
- Generating and capturing using the modified dialog
- Setting the context though workareas
  - Viewing and editing objects in multiple applications
  - Sharing and copying objects between applications

1-9

Properties in the Advanced Tabs

The Property Dialog boxes provide you with an easy-to-use interface, grouping together related elements and properties. You can access any additional properties that do not appear in the dialog pages using an “advanced” button. You can also drill down into secondary objects from primary objects, and view their additional properties.

For example, if you are in the dialog box for a table definition, and select the **Columns** tab, you can then press the **Advanced** button to view the advanced properties for a column.

Relative Tab Stop Editor

The Relative Tab Stop Editor is a GUI editor that gives an approximate representation of how the final form is generated. Use the shortcut menu in the display view of a module component to invoke the relative tab stop GUI editor.

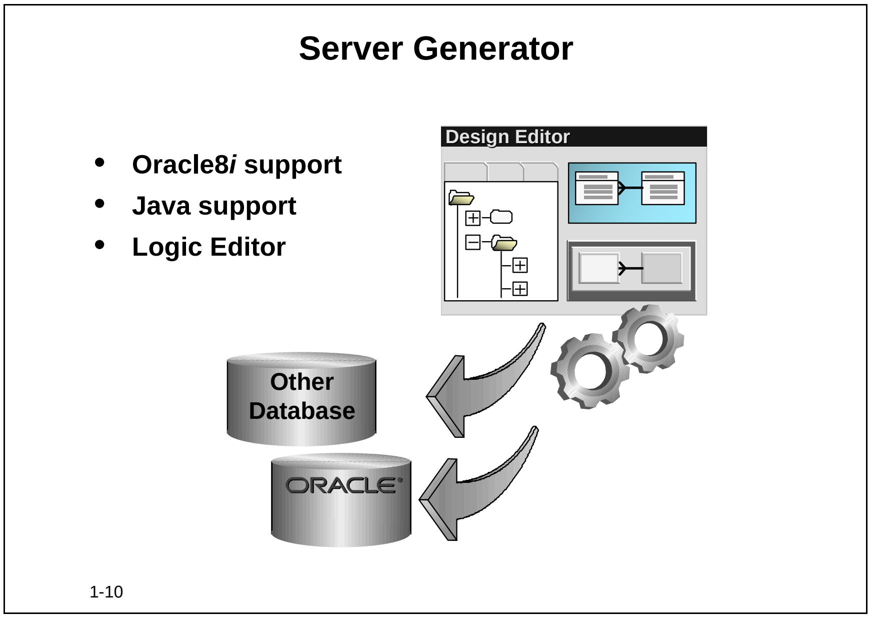
Using the Generator Dialogs

You have the option to set the commit or revert option before starting generation.

Workareas and Application Systems in the Design Editor

You always work in the context of a workarea. Because a workarea provides the context for the Design Editor session, it means that you can view and work in all the application systems visible in that workarea. It also means that you can copy and create short-cuts to objects in the Design Editor.

Using the Server Generator in Oracle Designer 6*i*



Server Generator

- **Oracle8*i* support**
- **Java support**
- **Logic Editor**

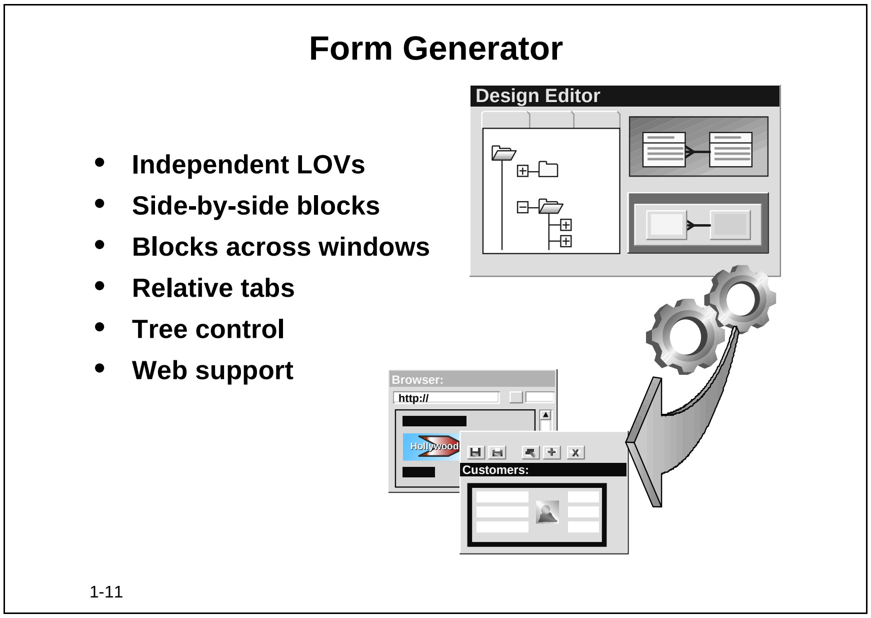
1-10

The capabilities of Oracle8*i* result in the addition of new element types to the Server Model in Oracle Designer 6*i*. In addition to generating database objects for these elements, the Server Generator also produces Oracle8 database objects from some new properties of existing element types.

Editing Code with the Logic Editor

Outwardly the Logic Editor has not changed. It still consists of a Construct Tree Window and Code Windows. The underlying technology of the Logic Editor has changed, allowing it to support languages such as Java, JavaScript, and PL/SQL.

Changes to the Client Generators

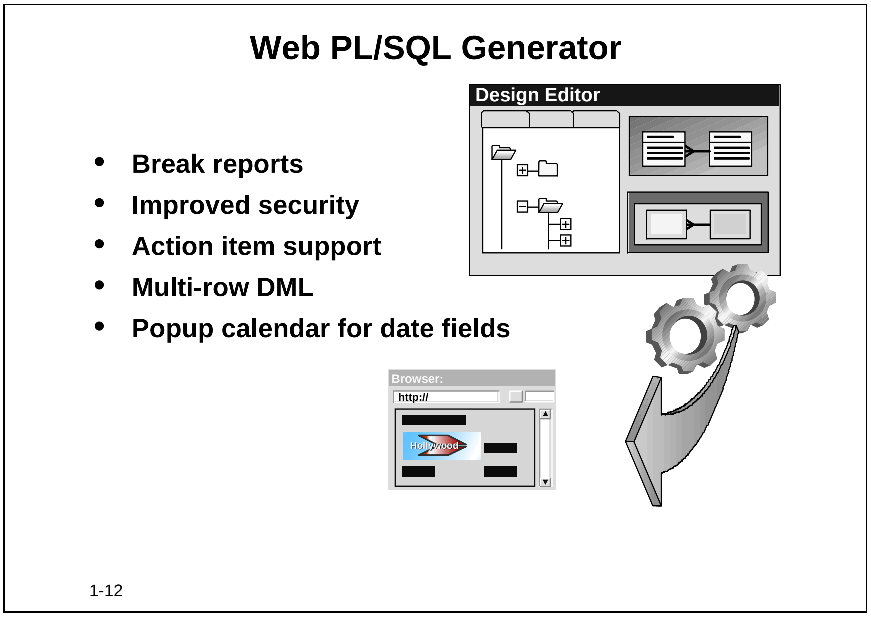


Form Generator

- Independent LOVs
- Side-by-side blocks
- Blocks across windows
- Relative tabs
- Tree control
- Web support

1-11

Significant changes to forms generation include greater control when placing blocks, items, and summary items. Also notable is the enhanced block navigation using navigator-style forms.



Web PL/SQL Generator

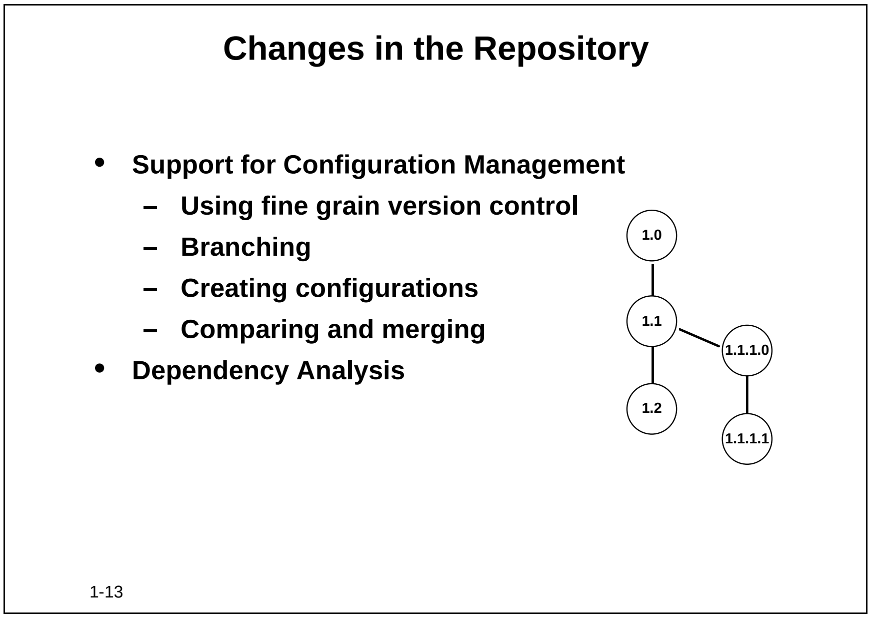
- Break reports
- Improved security
- Action item support
- Multi-row DML
- Popup calendar for date fields

1-12

Significant changes to the Web PL/SQL Generator include multi-row insert and update forms, break report style record lists, and an enhanced security API.

Supporting Configuration Management using Oracle Designer 6*i*

The toolset supports configuration management through fine grain version control in the repository.



Changes in the Repository

- **Support for Configuration Management**
  - **Using fine grain version control**
  - **Branching**
  - **Creating configurations**
  - **Comparing and merging**
- **Dependency Analysis**

1.0
1.1
1.1.1.0
1.2
1.1.1.1

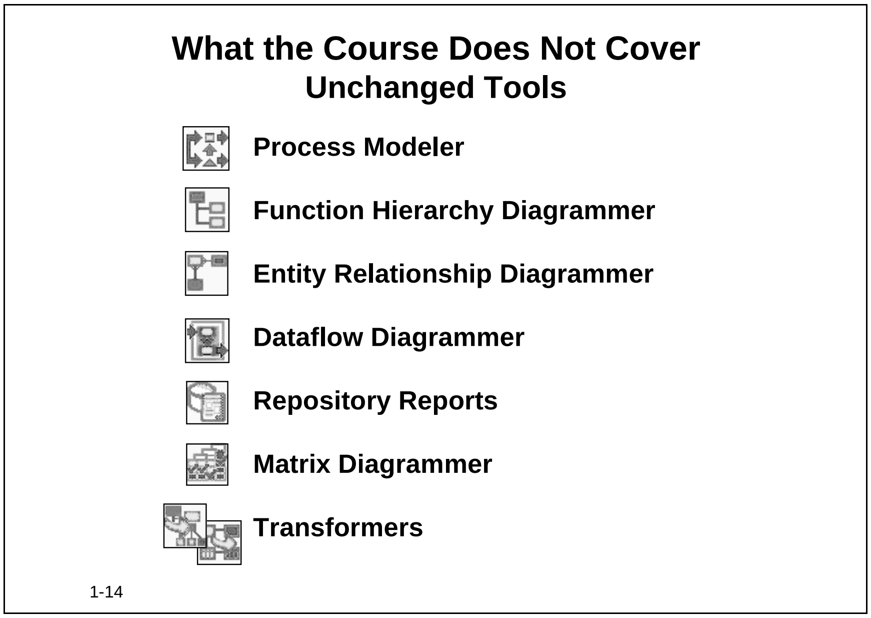
1-13

In this course you look at how to place objects under version control using checkin and checkout. You also go into the issues of creating branches and merging objects.

Dependency Analysis

You can analyze the dependencies on both repository object data and files in the repository using the Dependency Manager. You analyze dependencies during systems development to determine the impact of change on the application and to track relationships between parts of the design.

Tools Not Covered

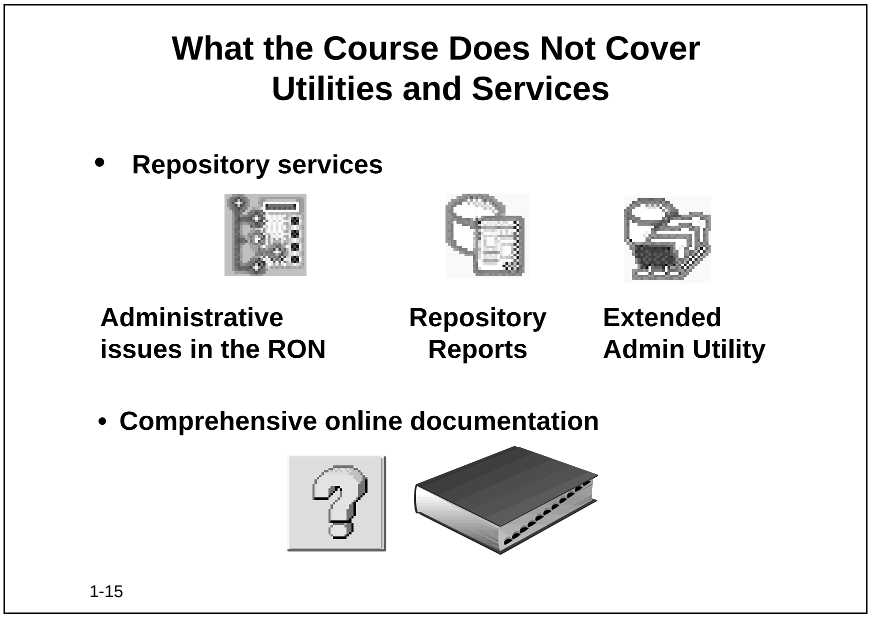


What the Course Does Not Cover
Unchanged Tools

- Process Modeler
- Function Hierarchy Diagrammer
- Entity Relationship Diagrammer
- Dataflow Diagrammer
- Repository Reports
- Matrix Diagrammer
- Transformers

1-14

The analysis tools have undergone no functionality changes other than those required to support configuration management through workareas, and version control.



What the Course Does Not Cover
Utilities and Services

- Repository services

Administrative issues in the RON

Repository Reports

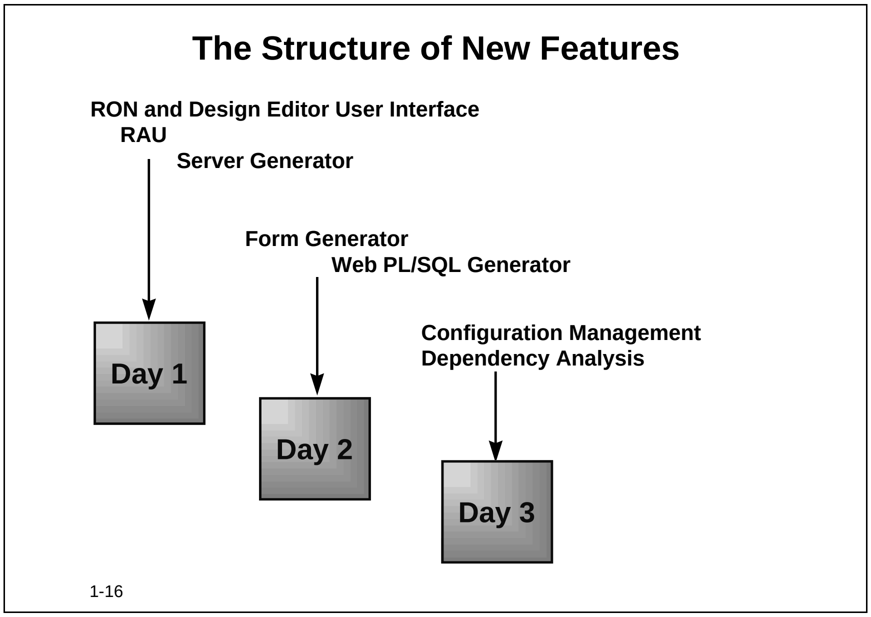
Extended Admin Utility

- Comprehensive online documentation

1-15

Course Structure

As always, the audience in a New Features course is varied. In an attempt to provide everyone with information in different areas of interest and skill levels, the course attempts to provide a large number of hands-on sessions.



The Structure of New Features

RON and Design Editor User Interface
RAU
Server Generator

Form Generator
Web PL/SQL Generator

Configuration Management
Dependency Analysis

Day 1

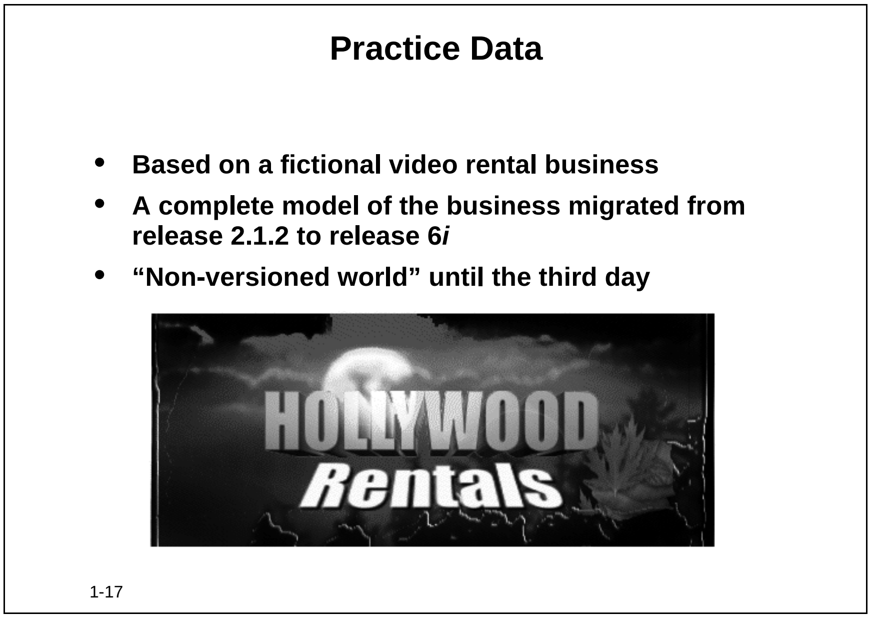
Day 2

Day 3

1-16

To avoid adding a level of complexity, the course first looks at the changes in the product since Oracle Designer R2.1.2, considering the topics without introducing versioning or introducing configuration management issues. You look at the new elements that are introduced and at the changes in the user interface of the supporting tools for these elements.

The Practices



HOLLYWOOD
Rentals

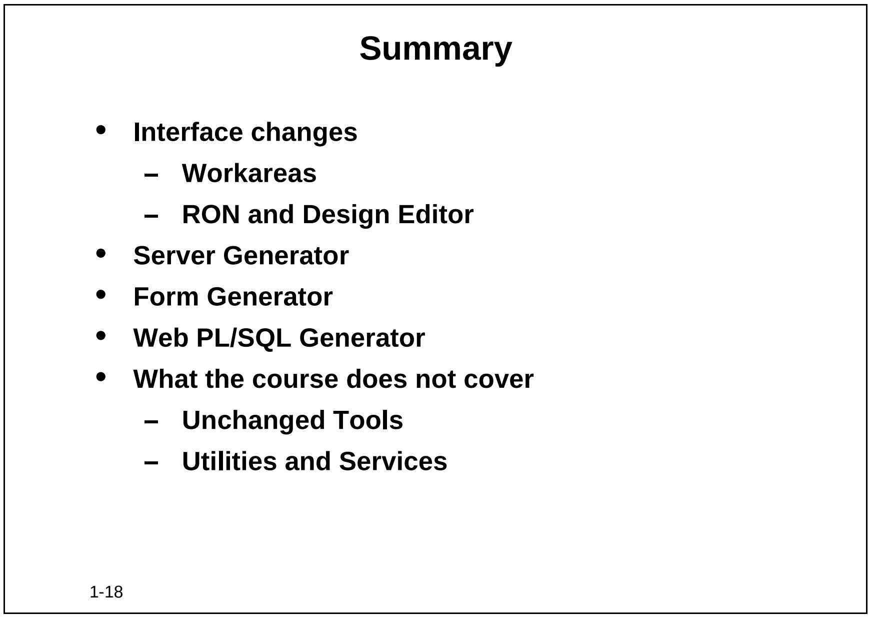
Practice Data

- **Based on a fictional video rental business**
- **A complete model of the business migrated from release 2.1.2 to release 6*i***
- **“Non-versioned world” until the third day**

1-17

On the third day you will start with the utilities and features that support configuration management.

Summary



Summary

- Interface changes
  - Workareas
  - RON and Design Editor
- Server Generator
- Form Generator
- Web PL/SQL Generator
- What the course does not cover
  - Unchanged Tools
  - Utilities and Services

1-18

Interface Changes to the RON

To enhance your use of the RON, you can:

- Perform complex searches using the Search Repository utility.
- View objects using tabs that group the objects according to the different stages in the process of development.

Design Editor Enhancements

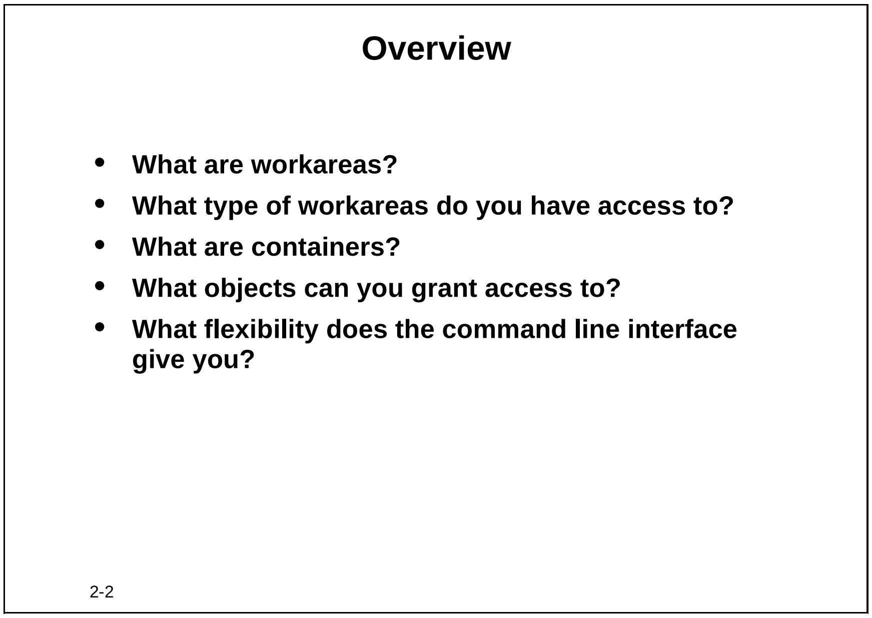
You can modify more than one application system at a time while working in the Design Editor. All properties for an object are now accessible from the dialog boxes.

2

Creating and Maintaining Workareas and Containers

Introduction

This lesson introduces the major new concepts in the repository: containers and workareas. There are different types of containers in the repository, application systems and folders, and these containers contain objects. For example, application systems could contain table definitions and modules. Workareas provide a view into these containers and allow you to see and manipulate objects from more than one container at a time.



Overview

- **What are workareas?**
- **What type of workareas do you have access to?**
- **What are containers?**
- **What objects can you grant access to?**
- **What flexibility does the command line interface give you?**

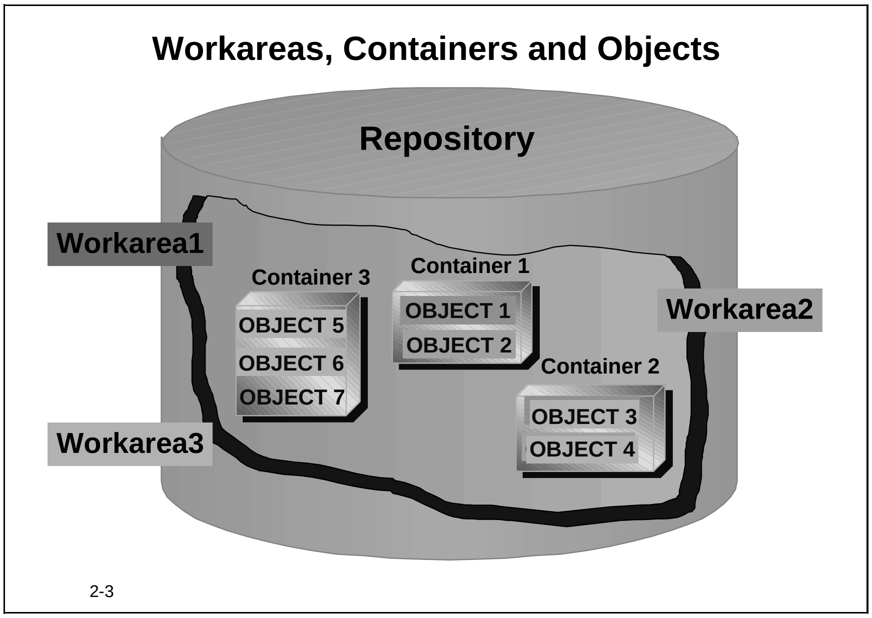
2-2

Topic	See Page
Practice 2 – 3: Manipulating the Repository Using the Command Line Tool	18
Practice 2 – Hints	20

Objectives

At the end of this lesson, you should be able to:

- Create workareas
- Define containers and grant users access to them
- Grant users access to workareas
- Use the Command Line Interface to perform operations on workareas

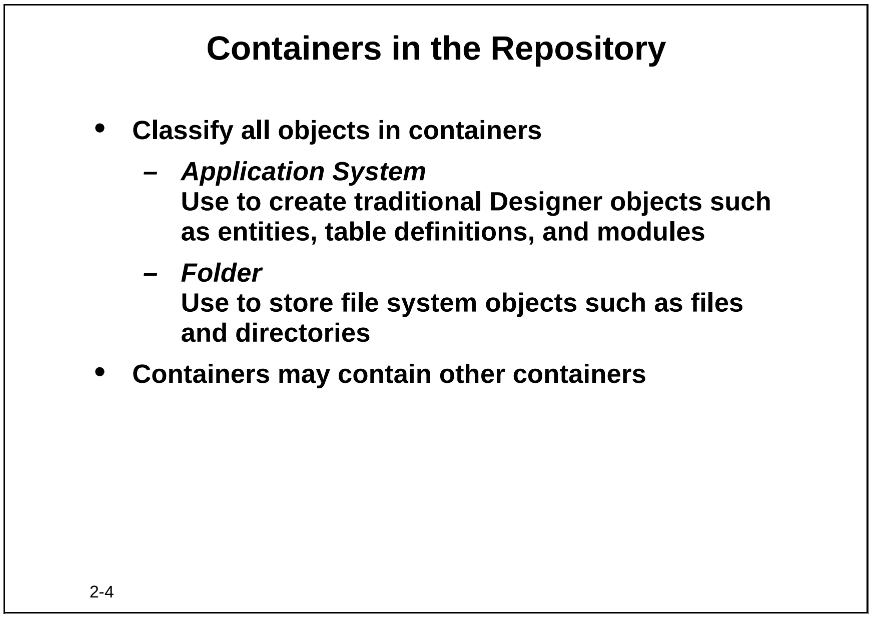


Workareas, Containers and Objects

2-3

Storing Objects in Containers

In earlier versions of Oracle Designer 6*i*, you use application systems to house or own the objects in the repository. In Oracle Designer 6*i* application systems are a type of container. In addition to application systems, a folder is another type of container.



Containers in the Repository

- **Classify all objects in containers**
  - ***Application System***
Use to create traditional Designer objects such as entities, table definitions, and modules
  - ***Folder***
Use to store file system objects such as files and directories
- **Containers may contain other containers**

2-4

Types of Containers

Containers own a collection of database objects. They can also own files uploaded from the operating system.

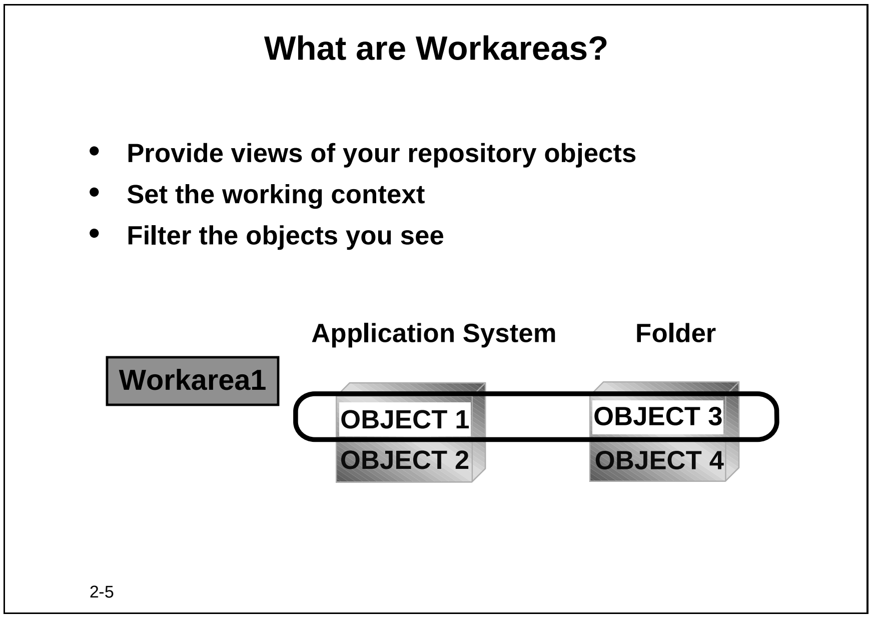
Container	**Description**
Application Systems	Use them as you would in releases of Designer earlier than Oracle Designer 6*i*.
Folders	Use them to store your operating system files. Note: The analysis tools, such as the ERD, do not recognize structured objects, such as entities, when they are owned by a folder.

Hierarchies of Containers

Containers may contain other containers in any combination. For example, you can create folders within application systems, or application systems within folders.

Workareas in the Repository

Workareas provide the only means through which you can both see and manipulate your objects. They are “views” to your repository objects.



What are Workareas?

- Provide views of your repository objects
- Set the working context
- Filter the objects you see

2-5

In the same way views on tables merely relay the data in a table, so workareas are views of your objects. The advantage of this is that you can create a workarea to work on a subset of available objects. Workareas do not own any objects, they just allow you to work with your objects. Unlike views and tables, however, you cannot manipulate objects outside a workarea.

Uses for Workareas

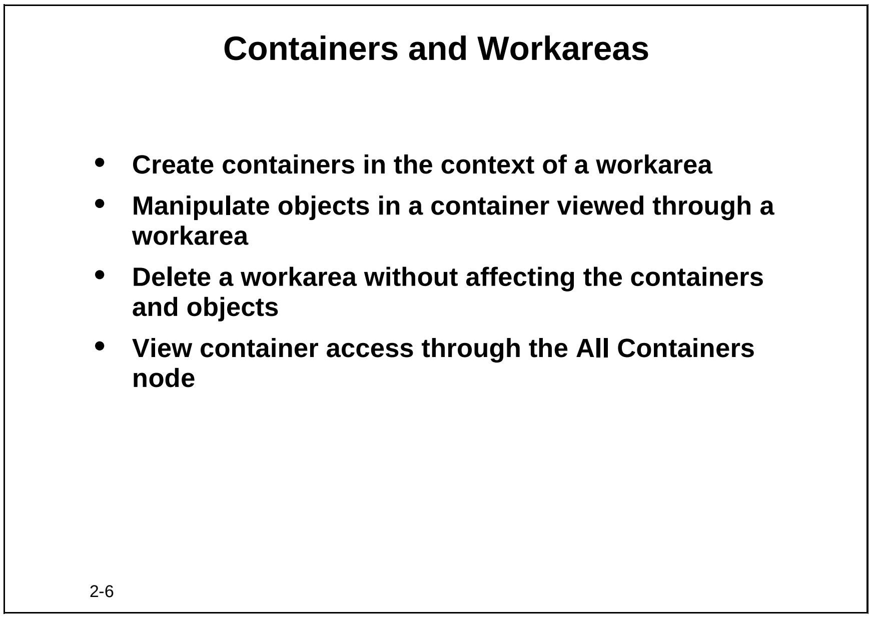
You could use workareas to:

- Present objects based on the stage of development. For example, you could present only analysis-level objects, such as entities and business functions.
- Restrict teams or organizations from viewing objects that do not pertain to their work.

Workareas set the context for tools such as the Design Editor, API scripts, and the Dependency Manager. When invoking and using these tools, you must specify a workarea to work within. You may recall that in prior versions of Oracle Designer 6*i* you had to work within the context of an application system; now you need to work in the context of a workarea.

Note: The main role of workareas is to support configuration management by allowing you to view specific versions of objects. You learn about object versions later in the course.

Comparing Containers and Workareas



Containers and Workareas

- **Create containers in the context of a workarea**
- **Manipulate objects in a container viewed through a workarea**
- **Delete a workarea without affecting the containers and objects**
- **View container access through the All Containers node**

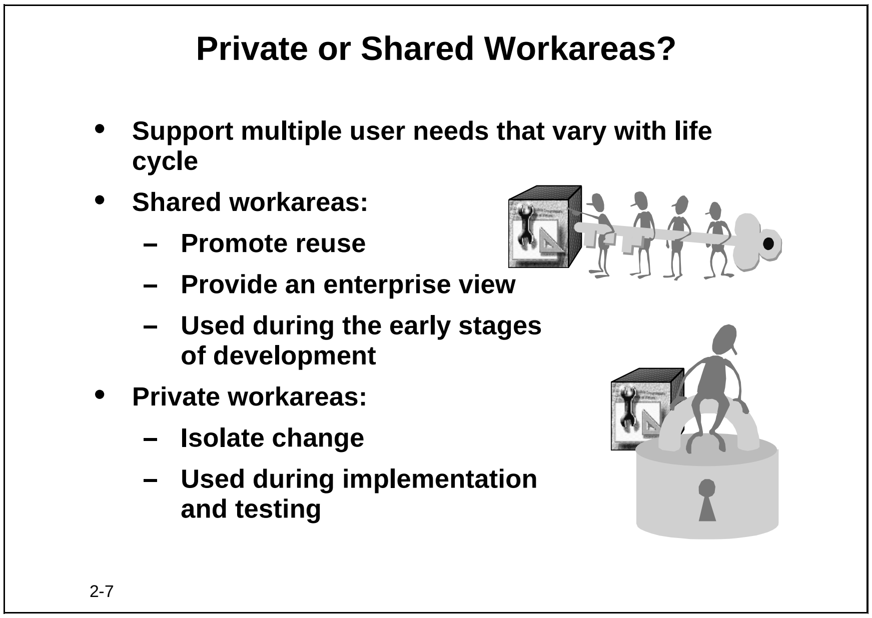
2-6

In comparing containers and workareas, you can:

- Create a container in the context of a workarea.
- Create and maintain objects in containers in the context of a workarea. While containers can exist on their own, you need a workarea to both view and manipulate the objects in the container.
- Delete a workarea. This does not delete the containers visible through the workarea.
- Use the **Shared Workareas** and **Private Workareas** nodes to view your workareas.
- Use the **All Containers** node to just view all containers that you have access to, independent of workareas. You cannot manipulate any object in the All Containers node.

Private and Shared Workareas

Some work you do may require collaboration with other colleagues. Other work efforts may not need this multi-user access. There are two kinds of workareas to meet these different needs.



Private or Shared Workareas?

- Support multiple user needs that vary with life cycle
- Shared workareas:
  - Promote reuse
  - Provide an enterprise view
  - Used during the early stages of development
- Private workareas:
  - Isolate change
  - Used during implementation and testing

2-7

Private workareas

A private workarea is only accessible to the user who creates the workarea. It is useful during the implementation and testing phases of development to work in private workareas, enabling changes to objects to be directed to a particular user.

If you grant another user access to a private workarea, it will not longer be visible under the Private Workareas node, instead it will appear under the Shared Workareas node.

Shared workareas

When more than one user has access to a workarea it is a shared workarea. The user who creates the workarea can grant access to the workarea to other users. It is convenient during the early stages of application development to work in shared workareas, allowing the reuse of objects among development team members.

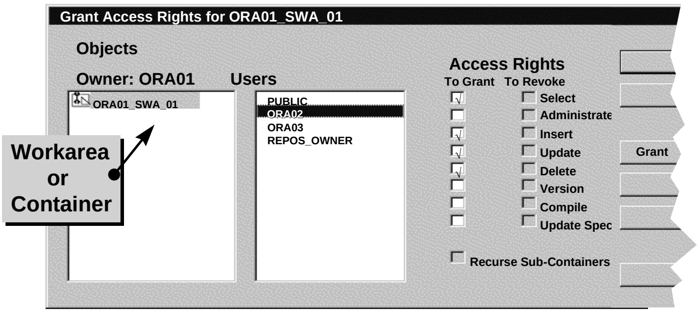
If you revoke all other user access to a shared workarea, it will not longer be visible under the Shared Workareas node, instead it will appear under the owners Private Workareas node.

Granting Access Rights to Multiple Users

You can grant access rights on a workarea or container to other repository users. You can grant one or more access rights to one or more users in a single operation. You can also transfer ownership of an object by clicking the **Ownership** button.

Who Has Access?

The owner or administrator grants access rights to other users



Grant Access Rights for ORA01_SWA_01

Objects

Owner: ORA01

ORA01_SWA_01

Users

PUBLIC
ORA02
ORA03
REPOS_OWNER

Access Rights

To Grant	To Revoke
☑	☐ Select
☐	☐ Administrate
☑	☐ Insert
☑	☐ Update
☑	☐ Delete
☐	☐ Version
☐	☐ Compile
☐	☐ Update Spec

☐ Recurse Sub-Containers

2-8

Before you can grant users access to workareas, you must have the privilege to allow the management of workareas. This privilege is assigned to you through the Repository Administration Utility (RAU).

How to Grant Access to Other Users

1 In the Navigator window of the RON, select a workarea or container to which you want to grant access.

2 Choose **File –> Access Rights –> Grant Access Rights.**

3 Choose one or more users from the **Users** list.

4 Choose one or more access rights from the **To Grant** column in the **Access Rights** list.

5 Click **Grant**.

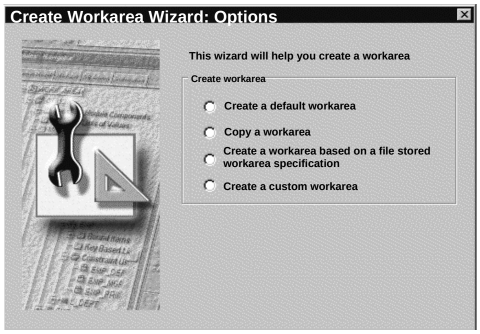
Privilege Restrictions

You can only grant access rights to workareas, containers and configurations. You cannot grant access rights directly to objects such as table definitions or modules.

Creating Workareas and Containers

Creating Workareas in the RON

- **Shortcut menu option Create Workarea**
- **Tools menu - Workarea Wizard...**
- **Create Object button**



Create Workarea Wizard: Options

This wizard will help you create a workarea

Create workarea

○ Create a default workarea

○ Copy a workarea

○ Create a workarea based on a file stored workarea specification

○ Create a custom workarea

2-9

How to Create Workareas in the RON

1 To create a workarea, click the Private Workareas node and using one of these options to invoke the Workarea Wizard:

– Click Create Workarea using the shortcut menu

– Use the **Tools –> Workarea Wizard...**

– Use the Create Object button.

2 Once you have invoked the wizard, use one of the following options:

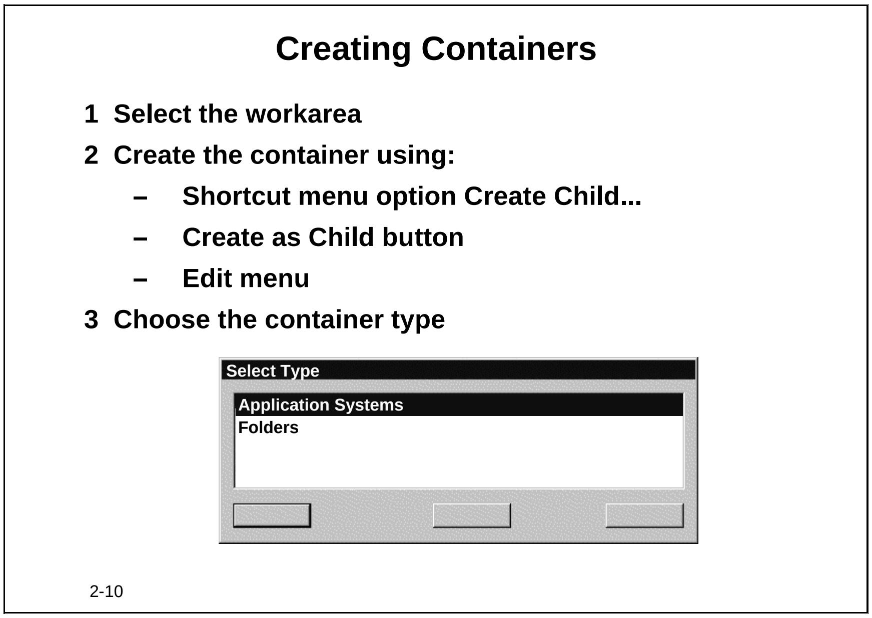
– Create a default workarea.

– Copy a workarea (this could be based on either private or a shared workarea and including the objects viewed through that workarea).

– Create a workarea based on a file stored workarea specification.

– Create a custom workarea. (This is discussed later in this course.)

How to Create Containers in the RON



Creating Containers

1 Select the workarea

2 Create the container using:

- Shortcut menu option Create Child...
- Create as Child button
- Edit menu

3 Choose the container type

2-10

1 Select the specific workarea to provide the context in which you will create the container.

2 Create the container using any of options:

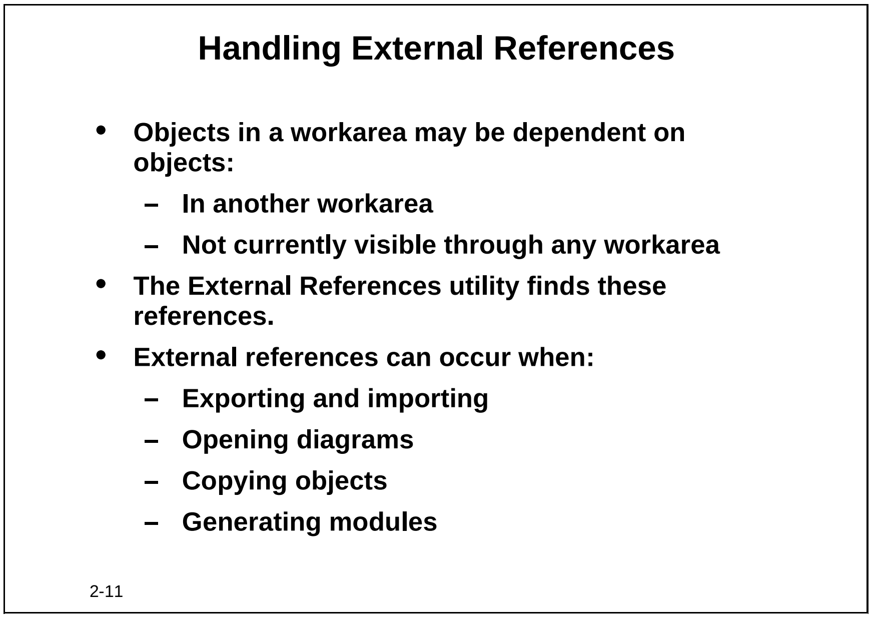
– **Create Child** shortcut menu

– **Create as Child** button

– **Edit –> Create Child** menu

3 Choose the container type to create.

Handling External References



Handling External References

- Objects in a workarea may be dependent on objects:
  - In another workarea
  - Not currently visible through any workarea
- The External References utility finds these references.
- External references can occur when:
  - Exporting and importing
  - Opening diagrams
  - Copying objects
  - Generating modules

2-11

Some of the objects visible in a workarea may be dependent on objects in containers visible in other workareas. For example, a module component visible in workarea A may be dependent on a table definition visible in workarea B. Workarea A is then said to have external reference to workarea B.

When to Consider External References

If you have created a workarea that contains a subset of a objects, you may have external references. You need to consider these when you are performing the following functions:

- Exporting objects and importing to another repository
- Opening diagrams
- Copying objects between application systems
- Generating modules

How to List and Include External References

Run the **External References** utility to look for those references to objects currently invisible through the current workarea and include the objects you need, such as elements missing from a diagram or table definitions required during generation of a module.

Exploring Workareas Using the Command Line Tool

The Command Line Tool is a “shell” alternative to a GUI for managing repository objects, both repository object data and repository-based files.

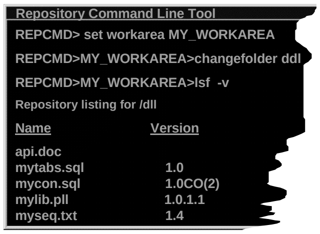
Maintaining Workareas with the Command Line Tool

R2.1.x/R6.0

- **Application system management only**

Designer 6*i*

- **Full access via the API, for example:**
  - **Make workarea (mkworkarea, MKWA)**
  - **Update workarea (updworkarea, UPDWA)**
  - **Remove workarea (rmworkarea, RMWA)**



Repository Command Line Tool

```
REPCMD> set workarea MY_WORKAREA
REPCMD>MY_WORKAREA>changefolder ddl
REPCMD>MY_WORKAREA>lsf  -v
Repository listing for /dll
Name                     Version
api.doc
mytabs.sql                 1.0
mycon.sql                  1.0CO(2)
mylib.pll                  1.0.1.1
myseq.txt                  1.4
```

- **Access using the RON or Start menu**

2-12

Commands Available from the Command Line Tool

You can perform a number of operations from the Command Line Tool including:

- Creating workareas and containers
- Deleting workareas and containers
- Listing the contents of workareas and containers

Commands consist of a command name, parameters, and options.

For example:

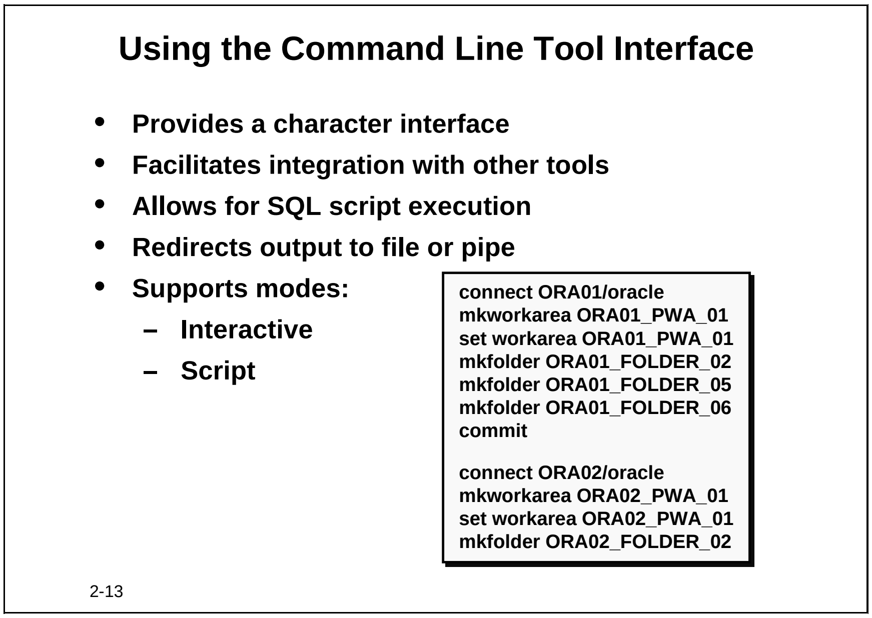
```
mkworkarea HOLLY_WA_MAIN
```

where the `mkworkarea` command creates the workarea named `HOLLY_WA_MAIN`.

Other examples are:

Command	**Description**
updwa	Update workarea
lswa	List accessible workareas
lsf	List accessible containers
cf	Change container
commit	Commit all changes

Using the Command Line Interface



Using the Command Line Tool Interface

- Provides a character interface
- Facilitates integration with other tools
- Allows for SQL script execution
- Redirects output to file or pipe
- Supports modes:
  - Interactive
  - Script

```
connect ORA01/oracle
mkworkarea ORA01_PWA_01
set workarea ORA01_PWA_01
mkfolder ORA01_FOLDER_02
mkfolder ORA01_FOLDER_05
mkfolder ORA01_FOLDER_06
commit

connect ORA02/oracle
mkworkarea ORA02_PWA_01
set workarea ORA02_PWA_01
mkfolder ORA02_FOLDER_02
```

2-13

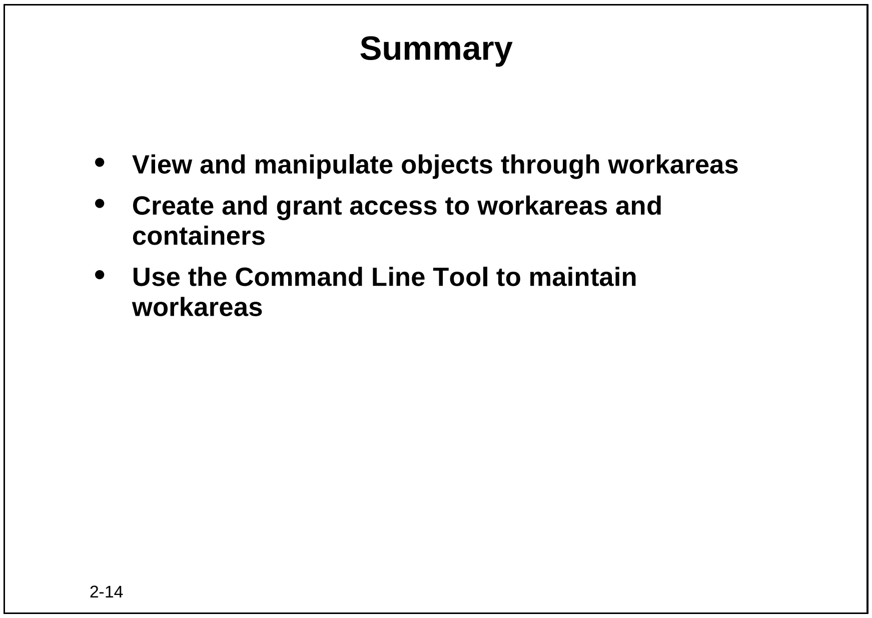
The Command Line Tool interface is similar in concept to SQL*PLUS in that you can enter commands from a command line to manipulate objects that exist in the repository. You can perform a wide variety of repository tasks from a command-line prompt. You can also perform bulk operations, such as creating releases or individual workareas for entire teams, using scripts run against the repository.

Performing Scheduled Batch Runs

One advantage of using the command line interface is that you have the ability to create batch files that can perform automatic updates of repository information. You can schedule these batch files to run at specific times without the need of intervention through repository access.

Summary

Workareas are a significant new feature to Oracle Designer 6*i* that can be seen throughout all of the tools.



Summary

- **View and manipulate objects through workareas**
- **Create and grant access to workareas and containers**
- **Use the Command Line Tool to maintain workareas**

2-14

Workareas and Containers in the Repository

A significant change to the repository is the addition of workareas and containers. In Oracle Designer 6*i* you can only both view and manipulate objects owned by containers in the context of a workarea. The advantage of this is that while a container may own thousands of objects, you can create a workarea to view and manipulate a subset of these objects.

User Access

You can explicitly grant users access to shared workareas and containers. The different container types are application systems and folders.

Practice 2 – 1: Creating a Workarea

Goal

In this practice you create a workarea and investigate the workareas in the RON.

Your Assignment

1. Invoke Oracle Designer 6*i* and connect to the repository using the information below as your starting point:

Username/password	ora<nn> /oracle
Connect string	

   where <nn> is the number your instructor has assigned to you.

2. Do not select an existing workarea from the initial window. Instead, create a new workarea using the workarea window to invoke the workarea wizard. Create a default workarea named *ORA<nn>_TEST_01,* where <nn> is the number assigned to you by your instructor.
3. Cancel the workarea window without selecting a workarea. Invoke the RON, and using the **Open Navigator** dialog window, navigate up a level to open the Repository. You should now see four nodes in the Navigator window in the RON.
4. Where is the workarea that you created? (Which node contains the workarea that you created?)
5. Do you have any ideas about how a workarea differs from an application system as you know it in releases earlier than Oracle Designer 6*i*?
6. Were any application systems created when you created the workarea?
7. How do you think you could create an application system?

Practice 2 – 2: Exploring and Working with Workareas

Goal

The purpose of this practice is to create and explore workareas in the RON.

Your Assignment

To introduce you to the Oracle Designer 6*i* environment, you:

- Use the Open Navigator Window in the RON
- Examine and grant access to workareas and containers
- Search the repository
- Customize the RON environment

Using the Open Navigator Window

1 Invoke the RON and select the Open Navigator option to invoke the Open Navigator window. (If you already have the RON open, **File –> Open**, will also invoke this window.) *Using the window*, answer the following questions:

a Is the workarea you created in Practice 2-1 a shared or private workarea?

b How many application systems are defined in the 20002_SWA_MAIN shared workarea?

c How many configurations do you have access to? (Do not worry about what they are, just where they can be found.)

d Double-click All Containers. What do you see?

e The workarea that you created in Practice 2-1 contains a folder, what folder is it?

f Move up to the top level and click once on the Repository node to open in the RON.

Note: As in Practice 2-1 question 3, you should now see the four nodes:

– Private Workareas

– Shared Workareas

– Configurations

– All Containers

Using the RON to Examine Private and Shared Workareas

2 Using the Workarea Wizard, create a default workarea. Use the following convention to name the workarea:

Workarea name	ORA<nn>_WA_02

where <nn> is the number assigned to you by the instructor.

3 Grant Select and Update access to the workarea to the student sitting next to you.

a Was the grant successful?

b What node does the workarea appear under? (You may need to requery to see this)

4 What access rights do you have on the shared workarea 20002_SWA_MAIN?

5 What access rights does NF_OWNER have on the shared workarea 20002_SWA_MAIN?

6 Who else has access to this 20002_SWA_MAIN workarea?

7 Create the application system container, ORA<nn>_APS_02, in the ORA<nn>_PWA_01 workarea.

8 Grant full access to the application system to the student sitting next to you.

a What do you gain by granting full access to the application system in the private workarea?

9 Create a folder container, ORA<nn>_FOLDER_01 in the ORA<nn>_PWA_01 workarea.

10 Grant Select access on the folder to the student sitting next to you. What do you achieve by granting select access to the folder in the private workarea?

11 Confirm the student next to you has access to the application system and folder containers created above.

If you have time...

Using Search Repository Tool.

12 Search the 20002_SWA_MAIN workarea, for all objects of the type ‘Domain’ with a name like “YES%”. How many did you find?

13 How many objects of the type ‘Sequence Definition’ exist in the workarea?

14 Start a new search. Set the context to be the entire repository. Use the **Audit...** button to restrict your search to find the number of objects the user, NF_OWNER, has created.

Customizing the Tabs in the RON.

15 Using the **Customize Navigator Groups** option, set the tabs to show the Server Model, Modules, Ref Data and Files.

16 What tab contains the Hollywood.con object, in the ORA<nn>_PWA_01 workarea, and in folder ORA<nn>_FOLDER_02?

Practice 2 – 3: Manipulating the Repository Using the Command Line Tool

Goal

The purpose of this practice is for you to use the Command Line Tool to perform some repository tasks such as making a workarea and listing the contents of a container.

Scenario

You need to view the contents of an application system in the repository and create a new workarea. You can perform these tasks without logging into Oracle Designer 6*i*. You need to take the following actions:

- Invoke the Command Line Tool and set your workarea
- List the contents of an application system
- Create a workarea

Your Assignment

Use the information below as your starting point:

Tool	Command Line Tool
Workarea	ORA<nn>_PWA_01
Application System	HOLLYWOOD

where <nn> is the number assigned to you by your instructor.

Setting the workarea using the Command Line Tool

1. Invoke the Command Line Tool.
2. Connect to the tool using your assigned username and password.
3. Using the following command, set the context of the workarea:

```
set workarea ORA<nn>_PWA_01
```

Note: This command is csae sensitive.

Changing to the folder and listing the contents

4. Using the following command, change to the folder:

```
cf ORA<nn>_FOLDER_02
```

5. List the contents of the folder. Use the following command:

```
lsf *.*
```

Creating a Workarea

6 Make a new workarea called ORA<nn>_PWA_CLT using the following command:

```
mkwa <workarea name>
```

7 Using the following command, set the context of the workarea:

```
set workarea ORA<nn>_PWA_CLT
```

8 List the contents of the workarea. Use the following command:

```
lsf *.*
```

9 Commit your changes and exit the tool. Invoke the RON and navigate to the workarea ORA<nn>_PWA_CLT and confirm your observations.

Practice 2 – Hints

To do this task	Follow these steps
Invoking the four nodes, Private Workareas, Shared Workareas, Configurations and All Containers.	**1** Use the RON menu option **File -> Open.** **2** Navigate **Up One Level**, using the button in the menu bar. **3** Select Repository. (Do not double-click on the word repository.) **4** Select OK.
Invoking the Repository Object Navigator	**1** Click the Repository Object Navigator icon in the Oracle Designer 6*i* front panel. **2** With the Open Navigator option chosen, click OK.
Creating a default workarea	**1** Click the Private Workareas node. **2** Use the shortcut menu, and select Create Workarea option. **3** Click the **Create a default workarea** option. **4** Enter a name and description. **5** Click the Finish button.
Granting access to workareas and containers	**1** Click the workarea or container to highlight it. **2** Choose **File –> Access Rights –> Grant Access Rights.** **3** Choose the repository user to whom you will grant access. **4** Select the available rights. **5** Click Grant. **6** Click OK.
Creating containers	**1** Click the workarea in which you will create the container. **2** Click the **Create as Child** toolbar button. **3** Choose the container type. **4** Click OK. **5** Name the container.
Searching for objects based on Name	**1** Choose **File –>Search Repository.** **2** Enter a value in the Name field. **3** Click the Search button.

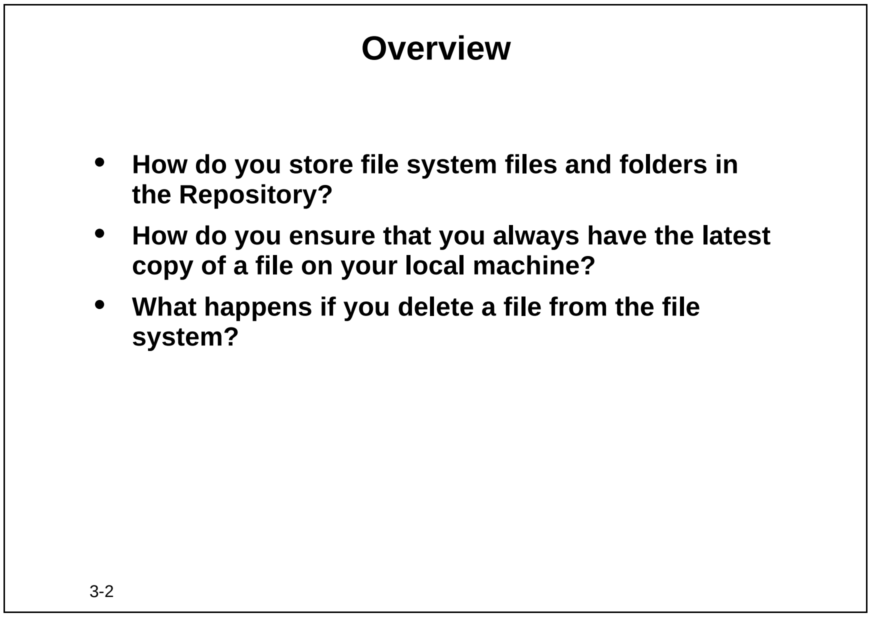
To do this task	Follow these steps
Counting the number of types of objects	**1** Choose **File –> Search Repository.** **2** Click the **Specific** radio button. **3** Use the browse button to select a value for Type names. **4** Click the **Count Hits** button.
Customizing the Tabs in the RON	**1** Use the **Customize Navigator Groups** option **2** Select the check boxes you want to display as tabs. **3** Display the tabs by selecting the menu option **View –> Show Group Tabs**
Invoking the Repository Command Line Tool	**1** Using your Windows Taskbar, choose **Start –> Programs –> Oracle Repository 6*i*–> Repository Command Line Tool.** **2** Enter the following command: `connect username/password` You can invoke the Command Line Tool from the RON. **Tools –> Command Line Tool.**

3

Storing Files in the Repository

Introduction

This lesson delves deeper into how to use folder containers to store files in the repository. You learn how to map folder containers to a file system directory and how to move files in and out of the repository.



Overview

- **How do you store file system files and folders in the Repository?**
- **How do you ensure that you always have the latest copy of a file on your local machine?**
- **What happens if you delete a file from the file system?**

3-2

Objectives

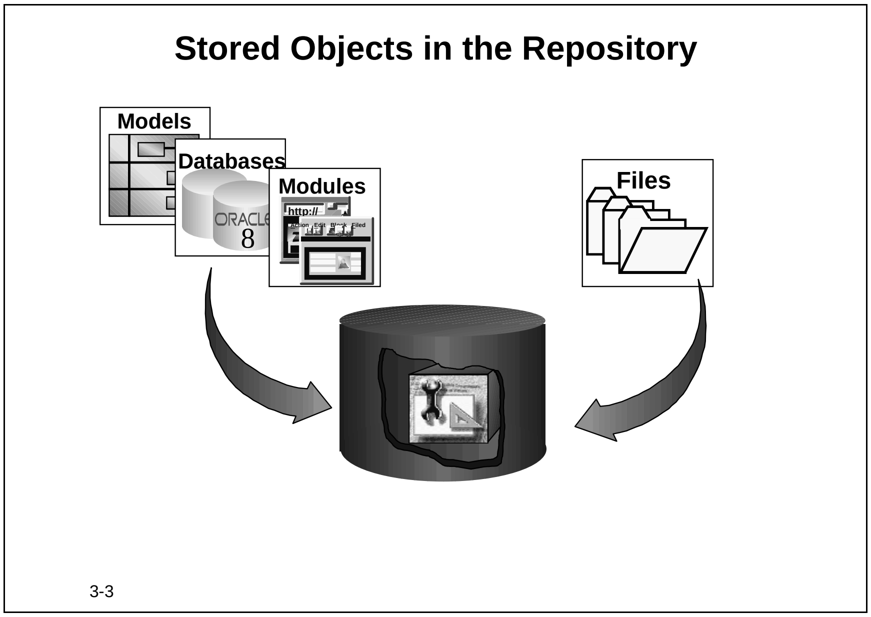
At the end of this lesson, you should be able to:

- Map files and folders to the Repository
- Upload and download files
- Edit files from the Repository Object Navigator
- Retrieve files that have been deleted from the file system
- Synchronize the file system with the file definitions in the Oracle Repository

Stored Objects in the Repository

Oracle Designer 6*i* allows you to store two types of objects in the repository:

- Object Data
- Files



Stored Objects in the Repository

3-3

Object Data

In earlier versions of Oracle Designer 6*i* and in the current version, you can store objects, such as entities, table definitions, modules, and diagrams.

Files

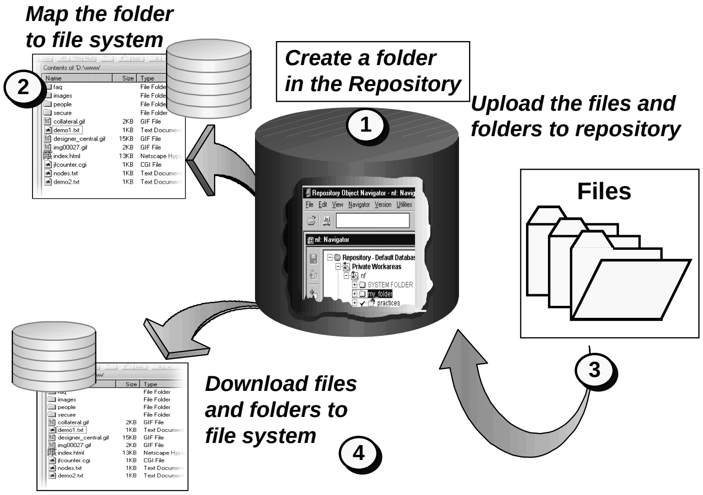
The Oracle Repository now allows you to store individual files or complete file systems. You upload these files into the repository or download them back to the file system. This allows you to store the files in a central area for access by different applications.

By allowing the inclusion of files and folders, it is possible to represent a collection of repository objects that are related in a significant way to an application. This allows an approach to managing an application system that focuses on all the objects that might affect that application, instead of just the objects that are needed to create the database or front-end applications.

The Repository and the File System

The first step to storing files and folders is to create a folder in the repository. Then you upload or download files from the file system within the context of a folder container.

Moving Files in and out of the Repository



Map the folder to file system

2

Create a folder in the Repository

1

Upload the files and folders to repository

Files

3

Download files and folders to file system

4

Contents of 'D:\www'
Name | Size | Type
faq | | File Folder
images | | File Folder
people | | File Folder
secure | | File Folder
collateral.gif | 2KB | GIF File
demo1.txt | 1KB | Text Document
designer_central.gif | 15KB | GIF File
img00027.gif | 2KB | GIF File
index.html | 13KB | Netscape Hyper
jfcounter.cgi | 1KB | CGI File
nodes.txt | 1KB | Text Document
demo2.txt | 1KB | Text Document
Repository Object Navigator - nf: Navig
File Edit View Navigator Version Utilities
nf: Navigator
Repository - Default Databas
Private Workareas
nf
SYSTEM FOLDER
my_folder
practices

3-4

Mapping Folders

When you map a folder to an operating system folder you associate a container in the repository with a directory in the file system. After that, if you select the folder to upload files to or download files from, the repository will use the mapped directory on the file system as a default.

Uploading Files and Folders

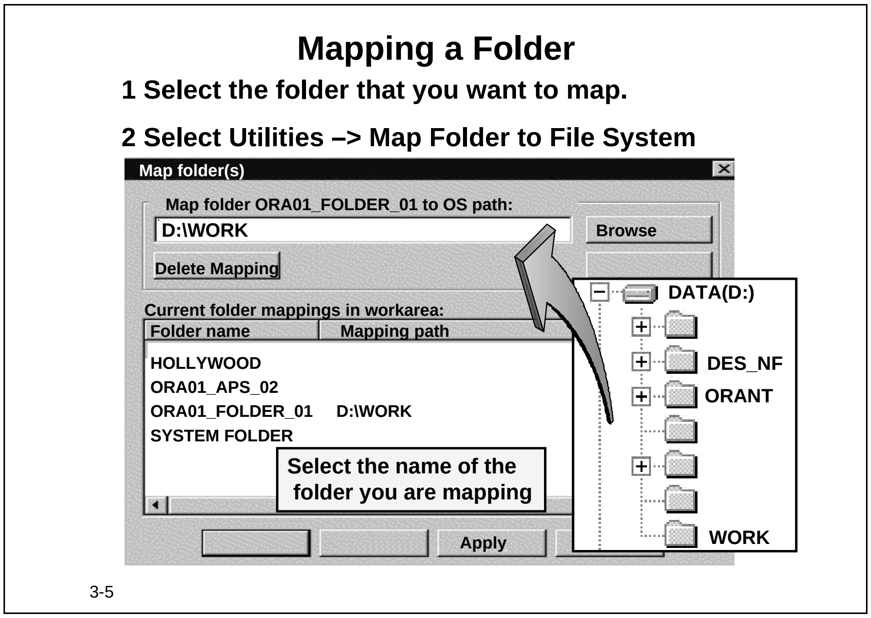
The files and folders are uploaded from the file system and placed in the folder that was defined in the **Destination** repository path. This process can only be performed from a client machine that can connect to the repository and has access to the machine or disk-drive from which files are being uploaded or downloaded.

Downloading Files and Folders

Downloading files and folders copies them from the repository back into the file system. If the file does not exist on the file system prior to download, a new file is created in the mapped directory with the same name as the file in the repository. If a file already exists on the file system in the mapped directory prior to downloading, that file is overwritten with the file from the repository. A message box prompts you if the file already exists.

Mapping to a Folder Container

Before uploading files and folders from the file system, you should map the folder in the repository to a directory on the file system.



Mapping a Folder

1 Select the folder that you want to map.

2 Select Utilities –> Map Folder to File System

Map folder(s)

Map folder ORA01_FOLDER_01 to OS path: D:\WORK

Current folder mappings in workarea:

Folder name	Mapping path
HOLLYWOOD	
ORA01_APS_02	
ORA01_FOLDER_01	D:\WORK
SYSTEM FOLDER	

Select the name of the folder you are mapping

Rules of Mapping

When you map folders, you should be aware of several rules.

- You map a repository folder to a specific file system directory to avoid having to specify a destination each time you upload or download files; however, you are not required to map a folder to a directory.
- You can map file system directories to repository folders visible in private workareas and those visible in shared workareas.
- You can only map files or folders from file systems to root folder containers. You cannot map files or folders from file systems to folder containers that are children of containers.
- You can map folder containers to directories on your local hard drive or floppy drive, or on a network drive to which you have access.

How to Map a Folder

To map a folder to a directory:

1. Select the folder that you want to map.
2. Choose **Utilities –> Map Folder to File System.**

   The dialog box for mapping a folder displays with the name of the folder already highlighted.

   You can click on another folder to map, if required.

3. Click the Browse button and navigate to the drive or directory that you want to map.
4. Click OK to exit the dialog box and save your changes.

Note: This mapping is user specific and is not stored in the repository. If the client is NT, then you would find the mapping in the NT registry.

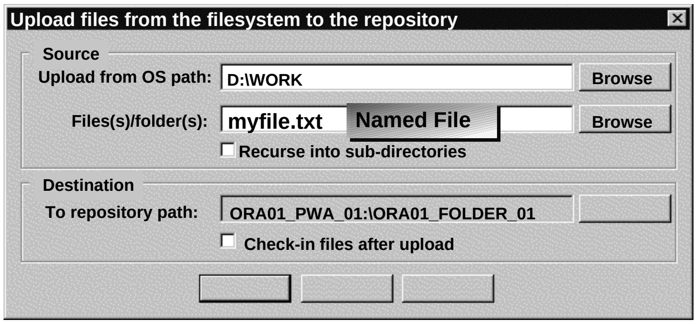
How to Upload Files and Folders

Uploading files and folders copies them from the file system into the repository tables that hold the file or folder. In essence, you are saving the file or folder definition to the repository.

Uploading Files and Folders

1 Select the folder to which you want to upload file(s).

2 Select Utilities –> Upload Files and Folders.



Upload files from the filesystem to the repository

Source

Upload from OS path: D:\WORK

Files(s)/folder(s): myfile.txt Named File

☐ Recurse into sub-directories

Destination

To repository path: ORA01_PWA_01:\ORA01_FOLDER_01

☐ Check-in files after upload

3-6

A file must exist on the file system prior to uploading. To perform an upload, in the RON:

1 Select the folder into which you want to load the files and folders.
2 Choose **Utilities –> Upload Files and Folders** (or the shortcut menu).

The dialog box for uploading files has several options:

Upload from OS path If you have selected a folder in the repository and then invoked the upload dialog box, the Upload from OS path will automatically default to the file system directory to which the folder is mapped. You can change this path by clicking the Browse button and choosing a different directory.

Files and Folders You can choose to either load a specific file or use a filter to refine the search for files that match a particular requirement.

Recurse into sub-directories Choosing the Recurse into sub-directories check box loads all sub-directories that exist in the source folder.

Destination You can set the destination repository path, if the upload operation is not performed from a specific folder.

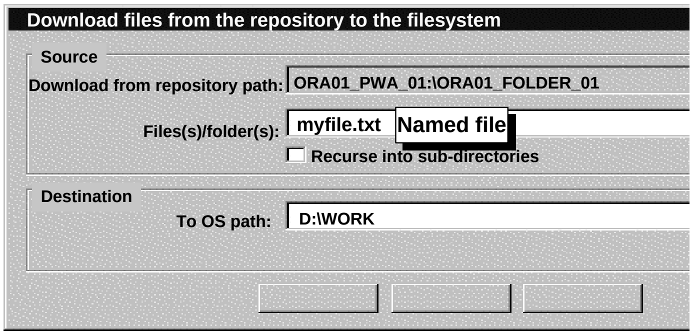
How to Download Files and Folders

If the file is not versioned, the last uploaded file to the repository is downloaded.

Download Files and Folders

1 Select the files that you want to download.

2 Select Utilities –> Download Files and Folders.



Download files from the repository to the filesystem

Source

Download from repository path: ORA01_PWA_01:\ORA01_FOLDER_01

Files(s)/folder(s): myfile.txt

☐ Recurse into sub-directories

Destination

To OS path: D:\WORK

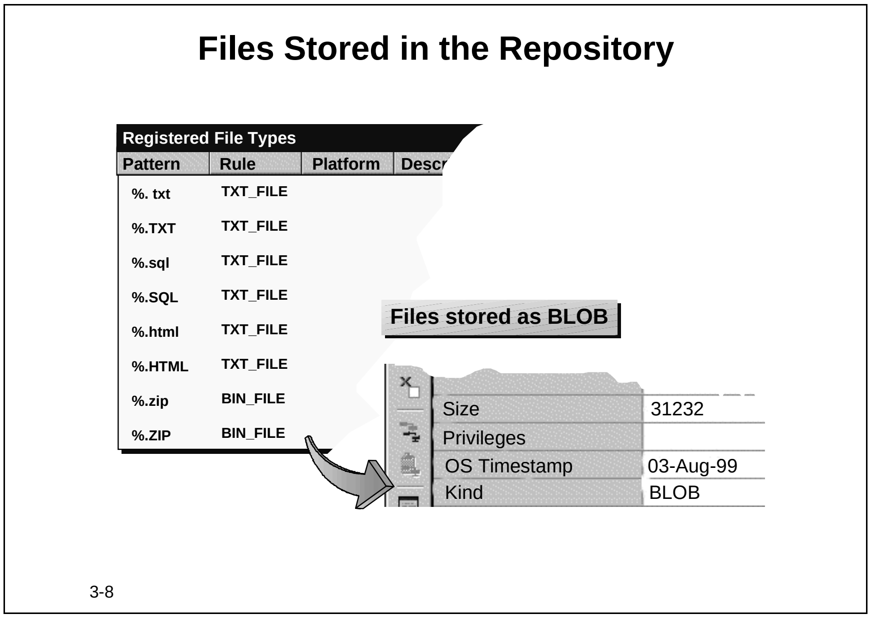
3-7

A file does not have to exist on the file system in order to download it. To download from the RON:

1. Select the file to be downloaded.
2. Choose **Utilities –> Download Files and Folders** (or use the shortcut menu).

Modifying Files in the Repository

Uploaded files stored in the repository are either binary or text files and they are stored in compressed or uncompressed format.



Files Stored in the Repository

Registered File Types

Pattern	Rule	Platform	Descr
%. txt	TXT_FILE		
%.TXT	TXT_FILE		
%.sql	TXT_FILE		
%.SQL	TXT_FILE		
%.html	TXT_FILE		
%.HTML	TXT_FILE		
%.zip	BIN_FILE		
%.ZIP	BIN_FILE		

Files stored as BLOB

Size	31232
Privileges	
OS Timestamp	03-Aug-99
Kind	BLOB

3-8

File Registry

The File Registry contains rules used to ensure that a particular type of file is stored as binary or text, or whether it is stored in compressed or uncompressed format. Each entry in the file registry defines one rule at a time. If you want to specify that a text file, '%.TXT', is stored as binary and compressed, you need to define two separate registry entries for the file type.

Viewing the File Registry

To view this registry, choose **Utilities –> Edit File Registry.**

You can add or delete file types from this registry as necessary. To find out how, choose **Help –> Index –> File Registry –> Editing the File Registry**.

Files Stored in the Repository

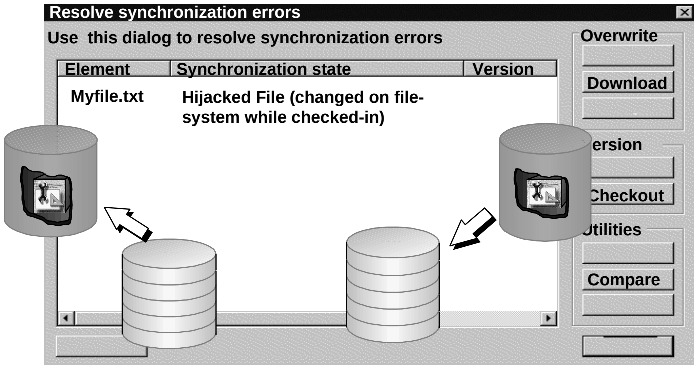
Files stored in the repository are stored in a BLOB (Binary Large Object) data type. The property palette specifies how the file is stored in the Kind property.

Synchronizing the Repository and the File System

Use synchronization to ensure that any changes you make in the repository are propagated to the file system and, similarly, changes made in the file system are propagated to the repository.

Synchronization

- **Synchronize repository and file system:**
  - **Uploads and downloads**
  - **Synchronize Files and Folders utility**



Resolve synchronization errors

Use this dialog to resolve synchronization errors

Element	Synchronization state	Version
Myfile.txt	Hijacked File (changed on file-system while checked-in)	

3-9

How Does Synchronization Work When Uploading and Downloading

Performing a file mapping implicitly activates synchronization. When you upload or download files, if there are changes to the files, a dialog is displayed and you are asked if you want to overwrite the existing file in the repository or the operating system, respectively. This synchronizes the files in the repository with those on the operating system.

How Does Synchronization Work When Using the Utility

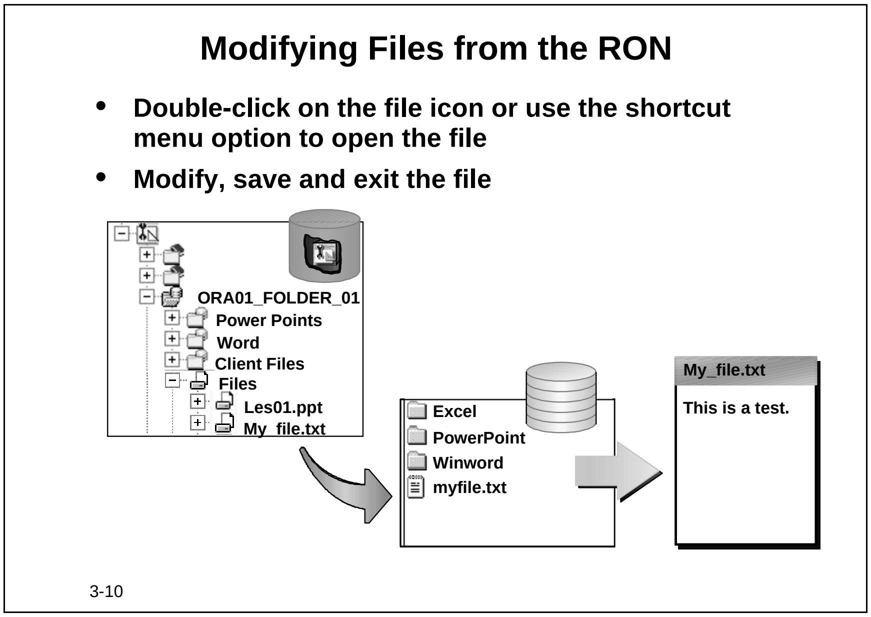
You can also explicitly synchronize the files using the **Utilities –> Synchronize Files and Folders** menu option. The synchronization utility chooses the direction to synchronize based on where the latest version is located. If one or both of the files have been changed, then the utility asks you to resolve the conflicts between the files. At that time you can choose to:

- Upload the file from the file system and overwrite the repository file
- Download the file from the repository and overwrite the operating system file
- Compare and merge the file system and repository files into one

Modifying Files from the Repository

To modify an existing file:

- Double-click on the file icon next to the filename
- Click on the shortcut menu to display a list of the possible operations.



Modifying Files from the RON

- **Double-click on the file icon or use the shortcut menu option to open the file**
- **Modify, save and exit the file**

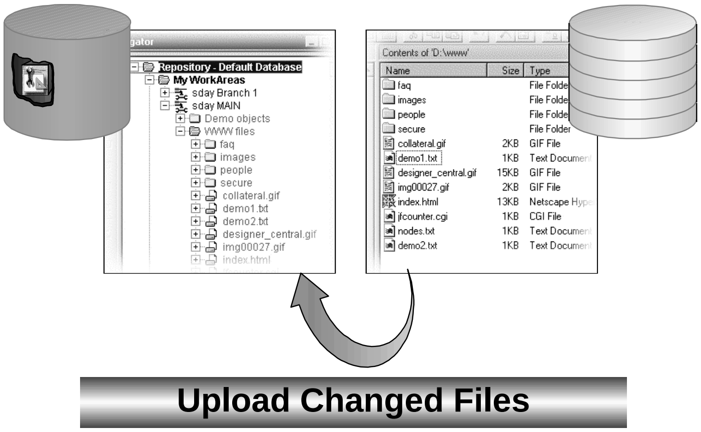
3-10

The file is opened using the program associated with that file type in the File Manager. When the file is saved, the changes are written to the file system. For more information on editing files from the RON, select **Help –> Index –> Files, Opening from the Repository**.

Storing Changes in the RON

Modifying a file by selecting and invoking it from the RON does not automatically save your changes to the repository. This opens the file in the file system and the changes are saved to the file system, but not in the repository.

Storing Changes in the Repository



Contents of 'D:\www'

Name	Size	Type
faq		File Folde
images		File Folde
people		File Folde
secure		File Folder
collateral.gif	2KB	GIF File
demo1.txt	1KB	Text Documen
designer_central.gif	15KB	GIF File
img00027.gif	2KB	GIF File
index.html	13KB	Netscape Hype
jfcounter.cgi	1KB	CGI File
nodes.txt	1KB	Text Documen
demo2.txt	1KB	Text Documen

Upload Changed Files

3-12

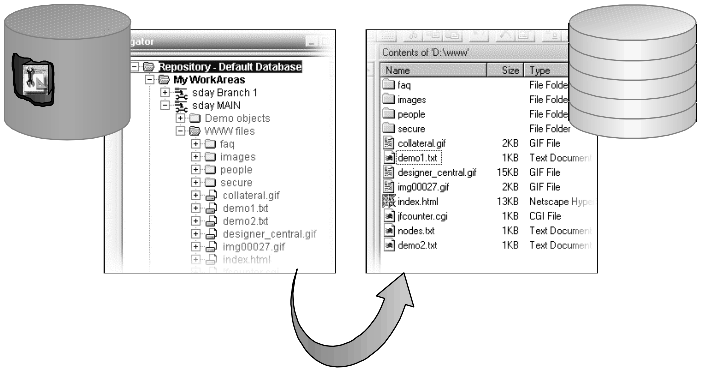
If the changes are not stored in the repository and the file is deleted or becomes corrupt, you lose your latest changes. There are a number of ways to save your changes to the repository.

- Explicitly perform a synchronization of files and folders.
- Upload a file.
- Check in the file to version it. This last method is covered later in this course.

Note: If you open a file from within the repository and the folder is mapped to the file system, an implicit synchronization occurs.

Restoring a File to the File System

Refresh File System



- Repository - Default Database
  - My WorkAreas
    - sday Branch 1
    - sday MAIN
      - Demo objects
      - WWW files
        - faq
        - images
        - people
        - secure
        - collateral.gif
        - demo1.txt
        - demo2.txt
        - designer_central.gif
        - img00027.gif
        - index.html

Contents of 'D:\www'

Name	Size	Type
faq		File Folde
images		File Folde
people		File Folde
secure		File Folder
collateral.gif	2KB	GIF File
demo1.txt	1KB	Text Documen
designer_central.gif	15KB	GIF File
img00027.gif	2KB	GIF File
index.html	13KB	Netscape Hype
jfcounter.cgi	1KB	CGI File
nodes.txt	1KB	Text Documen
demo2.txt	1KB	Text Documen

Download Changed Files

3-13

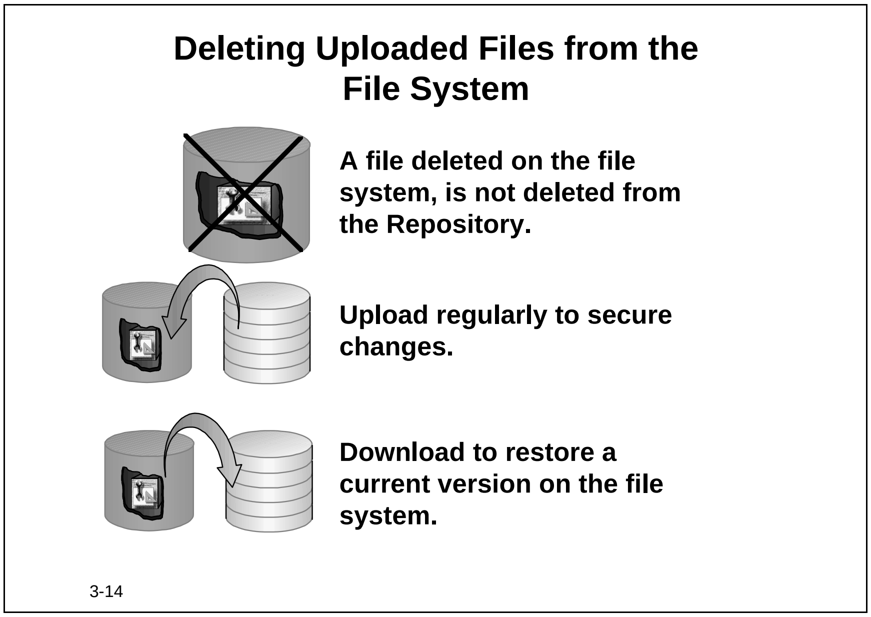
You may need to restore the files on the file system because you have:

- Deleted the file system files
- Made changes to the file system files that you want to discard.

To restore the previous version of a file to the file system, you download the file from the repository. The repository finds the latest version of the file that is stored and writes it to the file system. A dialog box asks if you are sure you want to overwrite the file on the file system. You should do this with caution as you may overwrite changes that you wanted to keep.

Retrieving Deleted Files from the Repository

Once a file is stored in the repository, removing it from the file system does not purge the file from the repository.



Deleting Uploaded Files from the File System

A file deleted on the file system, is not deleted from the Repository.

Upload regularly to secure changes.

Download to restore a current version on the file system.

3-14

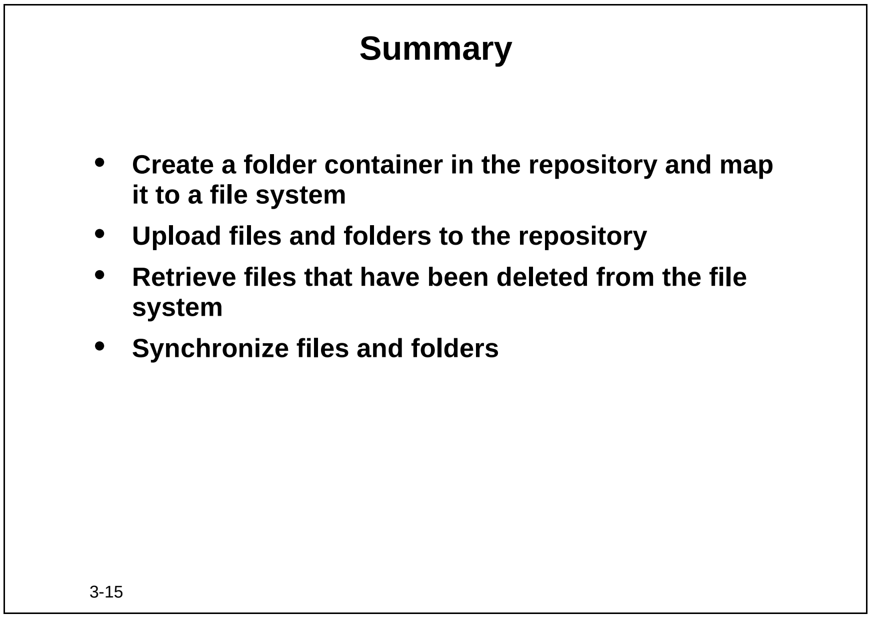
While it is generally recommended that you should modify files from the repository, you must modify some files from the file system. For example, you must enter Java or Visual Basic code from the file system because they work in an environment that cannot be opened from the RON.

To protect these files you should upload them to the repository or synchronize them with the repository every time you complete changes to the file.

If you mistakenly lose a file on the file system and have a current version stored in the repository, download or open the file to restore it to the file system.

Summary

The Oracle Repository now allows you to store individual files or complete file systems.



Summary

- **Create a folder container in the repository and map it to a file system**
- **Upload files and folders to the repository**
- **Retrieve files that have been deleted from the file system**
- **Synchronize files and folders**

3-15

You upload these files into the repository or download them back to the file system. This allows you to store the files in a central area for access by different applications. By allowing the inclusion of files and folders, it is possible to represent a collection of repository objects that are related in a significant way to an application.

Practice 3 – 1: Mapping Files to Folder Containers

Goal

The purpose of this practice is for you to load and modify a file in the repository and observe what happens to the file based on various operations that you perform. You will remove the file from the file system and retrieve it through the repository.

Scenario

The developers are working on bug fixes for the production release of the software. As each bug fix is completed, the developer responsible for fixing the bug documents a readme file. The readme file follows a standard format and the developer adds the documentation to the appropriate area within the file. Every Friday night, the completed bug fixes are implemented in production, and the readme file is placed in a directory for the users. Because of this process, the developers have made the decision to store and maintain the readme file in the repository. To achieve this, the following actions need to be taken:

- Upload a file into the repository and modify it by invoking it from the RON
- Store the latest changes to a file in the repository
- Retrieve a file that is inadvertently removed from the file system
- Download a file from the RON to a file system directory other than the one in which it was created

Your Assignment

Use the information below as your starting point:

Tool	Repository Object Navigator
Workarea	ORA<nn>_PWA_03

where <nn> is the number assigned to you by your instructor.

Uploading a File into the Repository and Modifying it by Invoking it from the RON

1. Create a folder named ORA<nn>_FOLDER_02 in the ORA<nn>_PWA_03 workarea.
2. Map the <working directory>/LES03 folder in the file system to the folder you just created. (Your instructor will provide you with the location of the working directory).
3. Upload the readme.txt file into the ORA<nn>_FOLDER_02 folder.
4. Open the readme.txt file from within the RON and place the following text next to the appropriate label:

LABEL	TEXT
BUG FIX OWNER	ORA<nn>
DATE COMPLETED	Current date
DESCRIPTION	The CALCULATE_LINE_ITEMS procedure is duplicating records when the total percentage for each budget line item results in an odd number.

5 Save and exit.

6 Open the file from the file system.

a Why are the changes you made, when opening the file from within the repository, automatically saved on the file system?

7 Exit the file without saving.

Storing the Latest Changes to a File in the Repository

8 In the previous task you saw how changes made to a file that you invoked from the RON are automatically saved to the file system. Are they automatically saved in the repository? Why or why not?

9 Which step do you perform if you want to ensure changes are stored in the repository? Perform this step.

10 Open the file again by invoking from the RON. Remove the description and save the file. (You can open the file from the file system to verify that your changes were saved. Exit without saving.)

11 Can you undo your latest changes without manually removing them from the file? How? Perform this step.

12 Open the readme.txt file from the RON.

a Which set of changes appear in the file? Why?

Retrieving a File that is Inadvertently Removed from the File System

13 Delete the readme.txt from the file system.

14 Will deleting the file from the file system also remove it from the repository? Why or why not?

15 Open the readme.txt file from the RON.

a Check the file system: Can you find the readme.txt file? Where did it come from, considering you deleted it in the previous step?

16 While still editing the readme.txt file, remove the date from DATE COMPLETED, save, and exit the file.

17 Delete the readme.txt file from the file system again.

18 When you open the file this time, will your latest changes be there?

19 Open the file.

- **a** Were the results as you expected? If not, what should you have performed to ensure that your latest changes appeared in the file?

Downloading a File from the RON to a Different File System Directory

20 From the RON, download readme.txt to <working directory>/LES03/README to test whether or not you can download a file to a different file system.

21 If you make additional changes to the readme.txt file in the RON, will those changes automatically update the file in the <working directory>/LES03/README?

Practice 3 – Hints

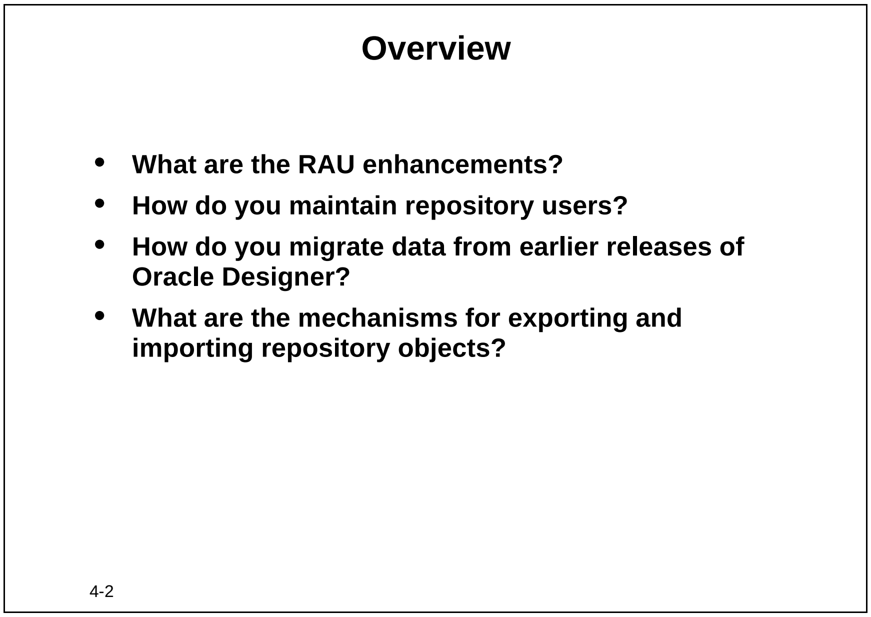
To do this task	Follow these steps
Mapping files to folder containers	**1** Select the folder that you want to map. **2** Use the shortcut menu and select **Map Folder to File System**. **3** Click **Browse** and navigate to the drive or directory that you want to map. **4** Click OK.
Uploading files to folder containers in the RON	**1** Select the folder into which you want to load the files and folders. **2** Use the shortcut menu and select **Upload Files and Folders**.
Downloading files from the RON	**1** Select the file to be downloaded. **2** Use the shortcut menu and select **Download Files and Folders**.

4

Investigating Administrative Issues and Migration to Oracle Designer 6*i*

Introduction

The purpose of this lesson is to allow you to become familiar with enhancements and changes to the Repository Administration Utility (RAU) and to the import and export utility in the RON.



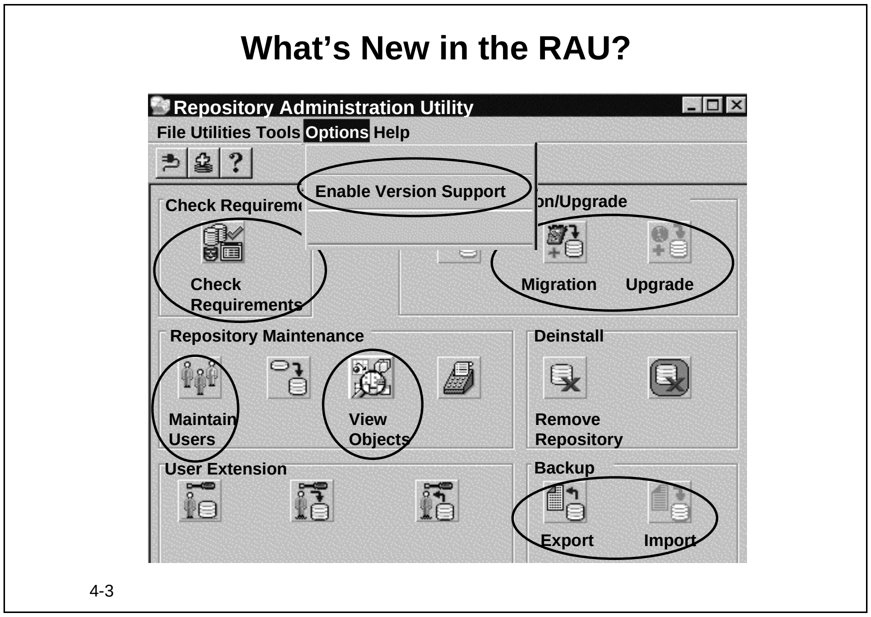
Overview

- **What are the RAU enhancements?**
- **How do you maintain repository users?**
- **How do you migrate data from earlier releases of Oracle Designer?**
- **What are the mechanisms for exporting and importing repository objects?**

4-2

Changes to the Repository Administration Utility

The changes to the RAU cover a wide range of topics.



What's New in the RAU?

4-3

Check Requirements

The information for verifying the system requirements in preparation for installation or upgrade has been consolidated into a single Check Requirements button. Use this Navigator-style interface to check:

- Parameter settings that the RAU uses
- System privileges and roles that you need to set for repository operations
- Tablespaces that the repository uses

You can modify the MS Windows Registry settings that apply to Oracle Repository from this requirements window.

Installation

You have a choice during installation of installing the full Oracle Designer 6*i* version with all of its object types, or installing a Core repository, which can be used if you are registering a schema or wish to use the Oracle Repository with a tool other than Oracle Designer 6*i*.

Migration and Upgrade

The Oracle Repository has been redesigned. An existing repository, created with earlier versions of Oracle Designer cannot be upgraded. You can, however, migrate the repository data in it to a new repository installed for Oracle Designer 6*i*.

- Early versions of Designer/2000 repositories must be upgraded to Oracle Designer

Release 2.1.2. or Release 6.0 before migration.

- Use the Migrate dialog box to move an Oracle Designer Release 2.1.2 or Release 6.0 repository to a Release 6*i* repository.
- The Upgrade dialog box allows you to upgrade 6i Limited Production to the production release of 6*i*.

Maintain Users

The functionality that can be assigned to a user has been extended to coincide with the more robust environment in which a user can work. In addition, the repository owner can specify the repository tools to which a user can have access.

View Objects

A Navigator-style interface allows you to view the status object types, and allows you to recompile or recreate them. You can compute statistics on selected repository tables and indexes from this window.

Backup

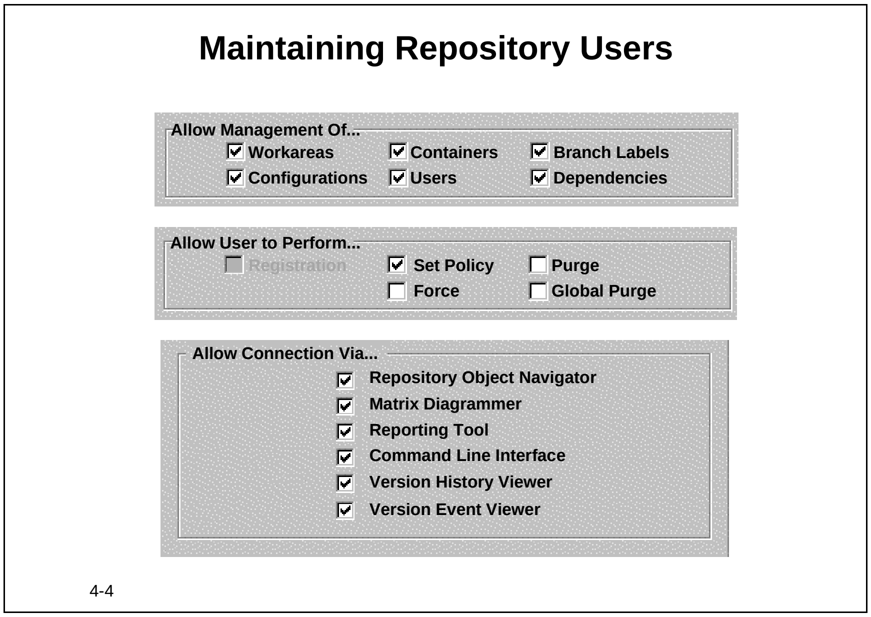
Backup and archive are no longer used. Instead, from the RAU you can export and import the full repository and from the RON you can import and export workareas, configurations, or groups of objects.

Enable Version Support

If you require versioning of repository objects, you must enable the versioning functionality from the Repository Administration Utility. This is not reversible.

Maintaining Repository Users

The facility to grant privileges to repository users is greatly extended.



Maintaining Repository Users

Allow Management Of...

☑ Workareas ☑ Containers ☑ Branch Labels
☑ Configurations ☑ Users ☑ Dependencies

Allow User to Perform...

□ Registration ☑ Set Policy □ Purge
□ Force □ Global Purge

Allow Connection Via...

☑ Repository Object Navigator
☑ Matrix Diagrammer
☑ Reporting Tool
☑ Command Line Interface
☑ Version History Viewer
☑ Version Event Viewer

4-4

Allow Management Of...

Set these privileges to specify whether users have the ability to manage environment objects, such as workareas, containers, and configurations.

Allow User to Perform...

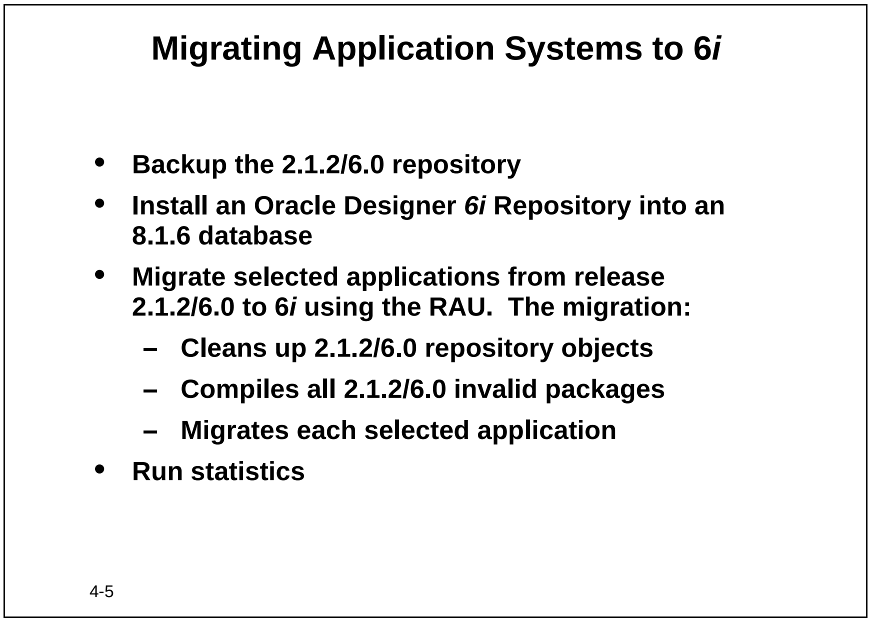
Set these privilege to specify whether users have the ability to perform repository-wide operations, such as set policies, purging, and force deleting objects.

Allow Connection Via...

Choose the tools to which a user will have access.

Migrating Application Systems to Oracle Designer 6*i*

To use a repository from an earlier release of Oracle Designer, you need to migrate the data from a release 2.1.2 or release 6.0 repository into a Oracle Designer 6*i* repository on a separate database.



Migrating Application Systems to 6*i*

- **Backup the 2.1.2/6.0 repository**
- **Install an Oracle Designer *6i* Repository into an 8.1.6 database**
- **Migrate selected applications from release 2.1.2/6.0 to 6*i* using the RAU. The migration:**
  - **Cleans up 2.1.2/6.0 repository objects**
  - **Compiles all 2.1.2/6.0 invalid packages**
  - **Migrates each selected application**
- **Run statistics**

4-5

Migrating Repositories

Moving from a release 2.1.2 or release 6.0 repository, you can migrate the full repository, with all its applications, or you can migrate individual applications.

Preparing for Migration

To prepare for the migration:

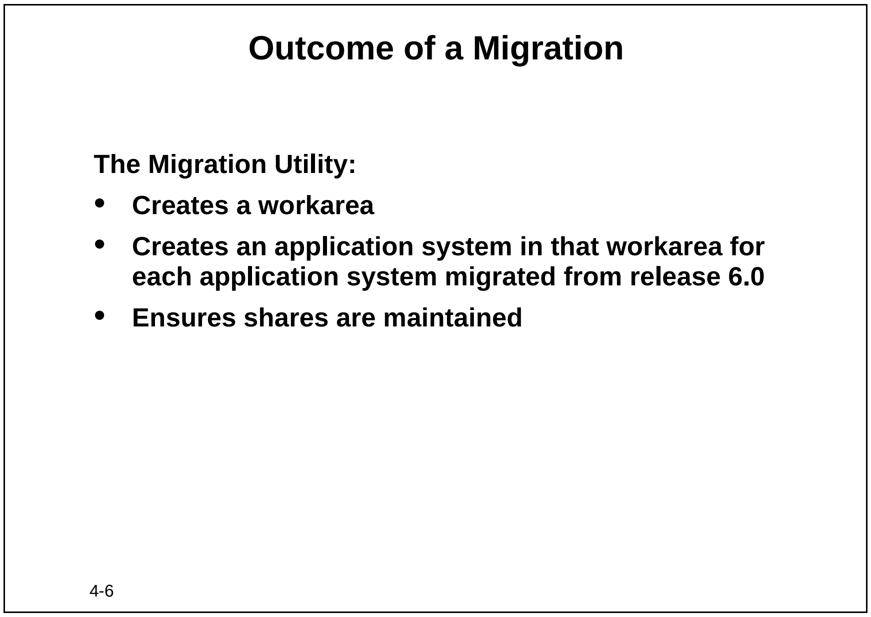
- Backup your release 2.1.2 or 6.0 repository.
- Install an Oracle Designer 6*i* Repository into an Oracle 8.1.6 database.

Migration

In Oracle Designer 6*i*, perform the following:

1. Install a repository.
2. Invoke the Migrate utility using the Migration button in the RAU.
3. Connect to the release 2.1.2 or 6.0 repository and selected the application systems you want to migrate and start the migration.
4. Compute repository statistics once the migration is complete.

The Results of the Migration



Outcome of a Migration

The Migration Utility:

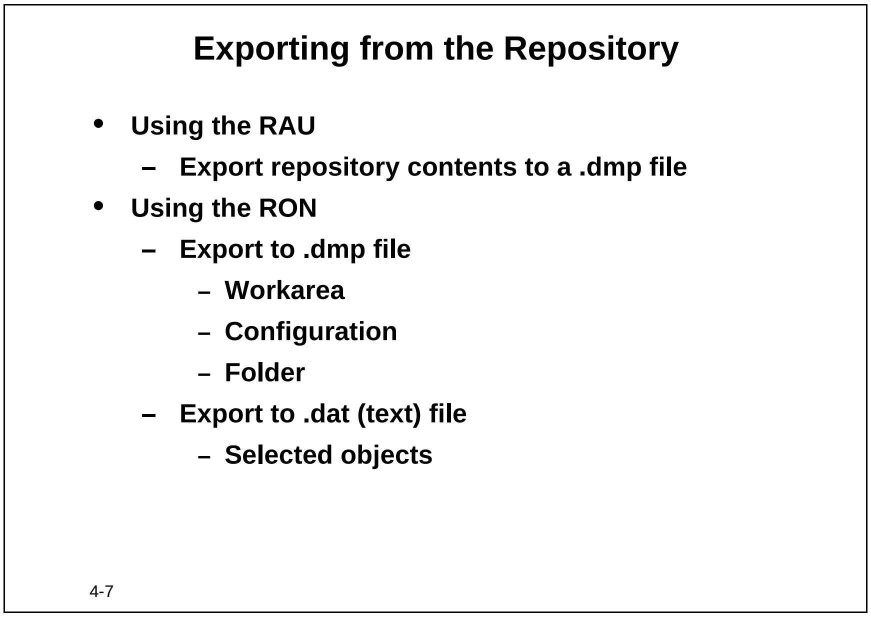
- **Creates a workarea**
- **Creates an application system in that workarea for each application system migrated from release 6.0**
- **Ensures shares are maintained**

4-6

All application systems are migrated into a single workarea in the new repository. All shares created on an application system in an earlier release of Oracle Designer are maintained when migrating. After migration users will see the same applications and have the same access as they did in the original release.

Exporting from the Repository

Use the import and export utilities in the RON or RAU to enable you to export entire repositories or move workareas, containers, individual objects, or groups of objects from one repository to another.



Exporting from the Repository

- Using the RAU
  - Export repository contents to a .dmp file
- Using the RON
  - Export to .dmp file
    - Workarea
    - Configuration
    - Folder
  - Export to .dat (text) file
    - Selected objects

4-7

Full Repository Export

In the RAU, the Export button allows you to archive (export) or restore (import) a complete repository. To use the export and import buttons you need to be connected as the repository owner.

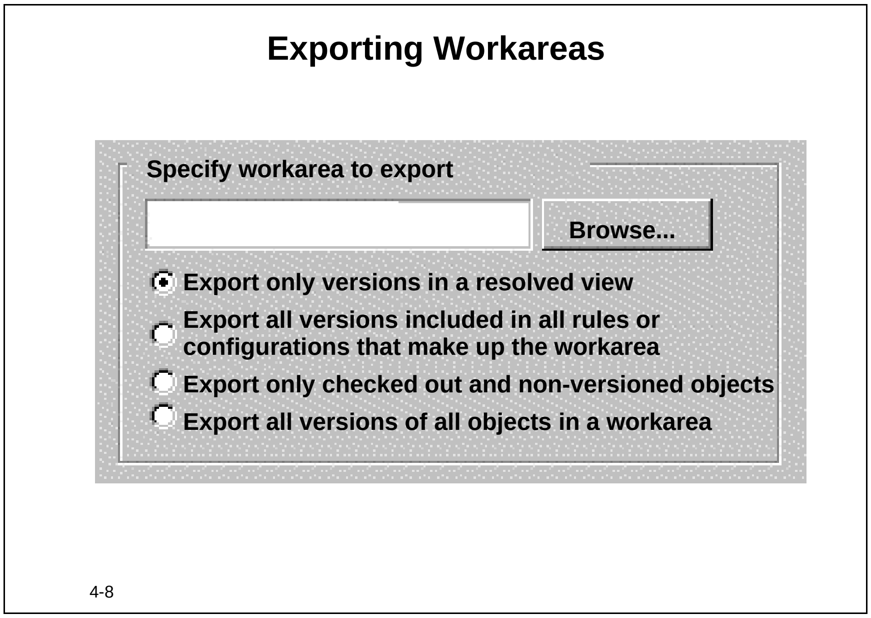
Using the Export Wizard

The Export Wizard is available from the RON and allows you to export objects at a finer level. In other words, you can export complete workareas, containers, or objects.

You can elect to export the data in Oracle database (.dmp) format or Repository Loader (.dat) format. You can only use the .dmp format to export workareas, configurations and containers. Use the .dat format to export specific objects or groups of objects such as tables and modules.

Exporting Workareas

There are a number of options available when you export workareas.



Exporting Workareas

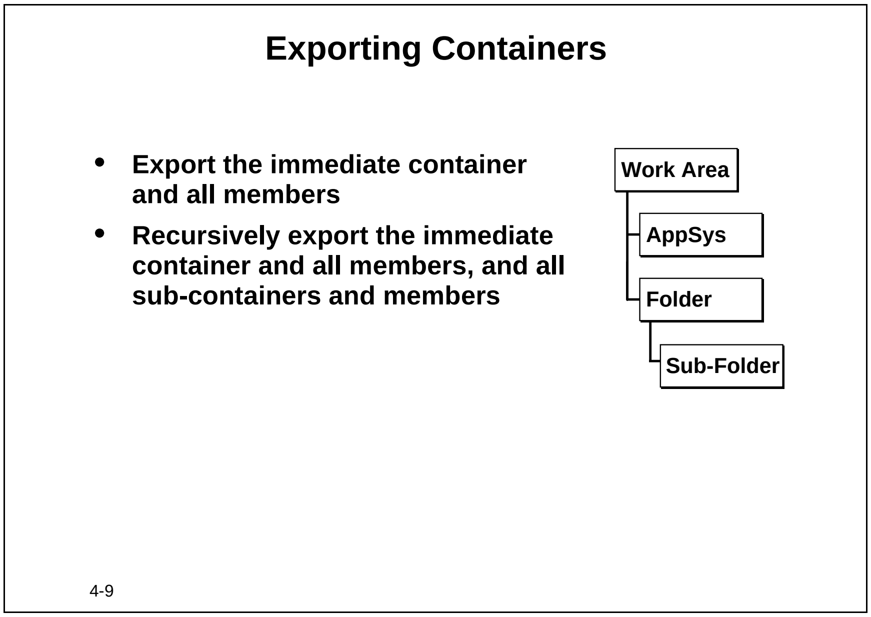
Specify workarea to export

● Export only versions in a resolved view
○ Export all versions included in all rules or configurations that make up the workarea
○ Export only checked out and non-versioned objects
○ Export all versions of all objects in a workarea

4-8

Option Available	Description
Export only versions visible in a resolved view	Exports only those object versions that are shown for this workarea in the Navigator window
Export all versions included in all rules or configurations that make up the workarea	Exports any object versions that are defined for this workarea by rules and configurations. So if a workarea specification has rules to include objects that are on two branches, the latest versions from both branches will be exported.
Export only checked out and non-versioned objects	Exports only checked out versions of objects in this workarea, together with any objects in the workarea that have never been checked in
Export all versions of all objects in a workarea	Exports every version of every object that is defined for this workarea

Exporting Containers



Exporting Containers

- **Export the immediate container and all members**
- **Recursively export the immediate container and all members, and all sub-containers and members**

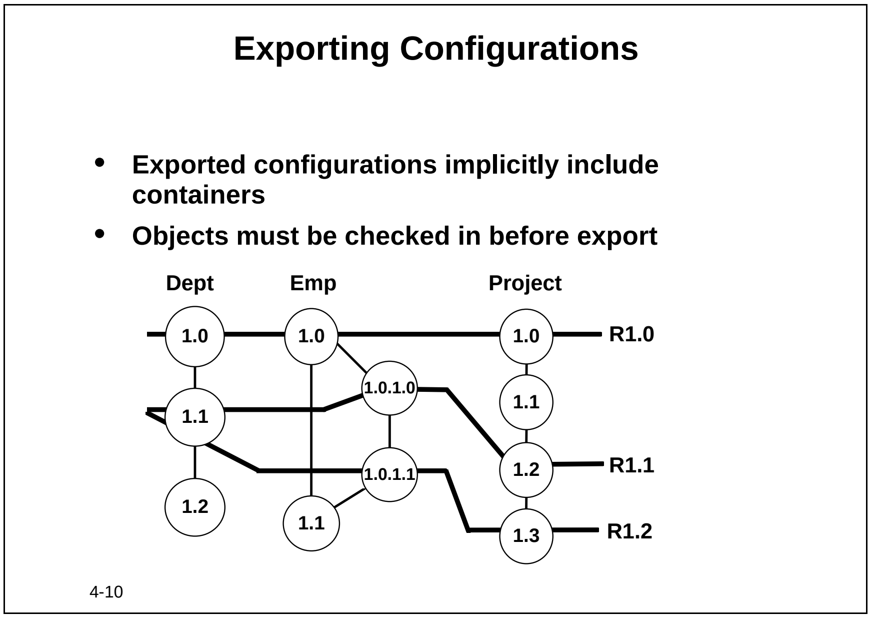
Work Area
AppSys
Folder
Sub-Folder

4-9

1. Specify the export format.
2. Specify workarea from which you want to export the folder.
3. Select the container whose contents you want to export.
4. Select the checkbox if you want to recurse though sub-directories to export the contents of any sub-containers of the specified container.

Exporting Configurations

You can export the entire contents of a configuration.



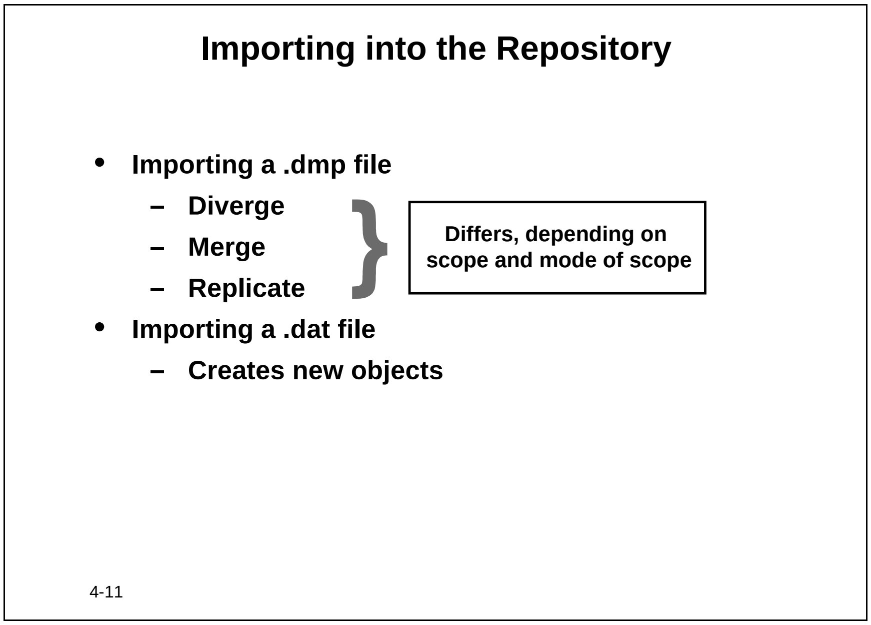
Exporting Configurations

- Exported configurations implicitly include containers
- Objects must be checked in before export

Dept
Emp
Project
1.0
1.1
1.2
1.0
1.0.1.0
1.0.1.1
1.1
1.0
1.1
1.2
1.3
R1.0
R1.1
R1.2

4-10

Importing into the Repository



Importing into the Repository

- Importing a .dmp file
  - Diverge
  - Merge
  - Replicate
- Importing a .dat file
  - Creates new objects

Differs, depending on scope and mode of scope

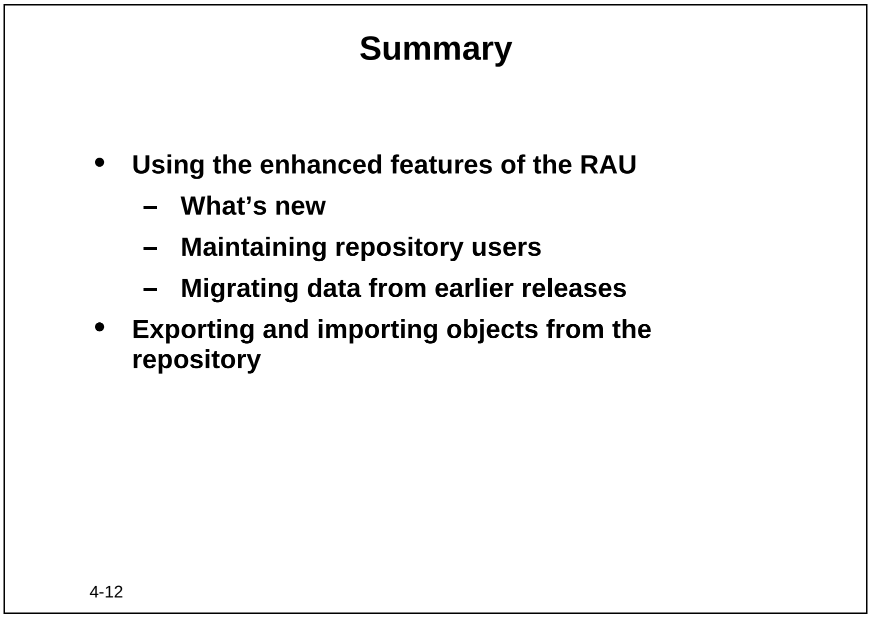
4-11

You can decide, during import, whether the tool should:

- Create brand new objects
- Create new versions of objects where the objects exist, otherwise create new objects
- Update existing object versions if the objects are checked in, otherwise create new versions of the objects. This option also creates new objects if the objects don’t exist.

Summary

This lesson briefly describes the enhancements to the administration areas in the repository.



Summary

- **Using the enhanced features of the RAU**
  - **What's new**
  - **Maintaining repository users**
  - **Migrating data from earlier releases**
- **Exporting and importing objects from the repository**

4-12

RAU Interface

A Navigator-style interface was added to many of the utilities to make them easier to use and more complete.

Users must migrate repositories from earlier releases of Oracle Designer to Oracle Designer 6*i*. Archive and backup are export and import, respectively, in Oracle Designer 6*i*.

User Maintenance

User privileges have been extended to allow users to manage their environment, perform repository-wide operations, and access tools.

Repository Migration

Migration to Oracle Designer 6*i* requires a live 2.1.2 or 6.0 repository. The results of the migration is a workarea containing the migrated applications. All shares are maintained.

Practice

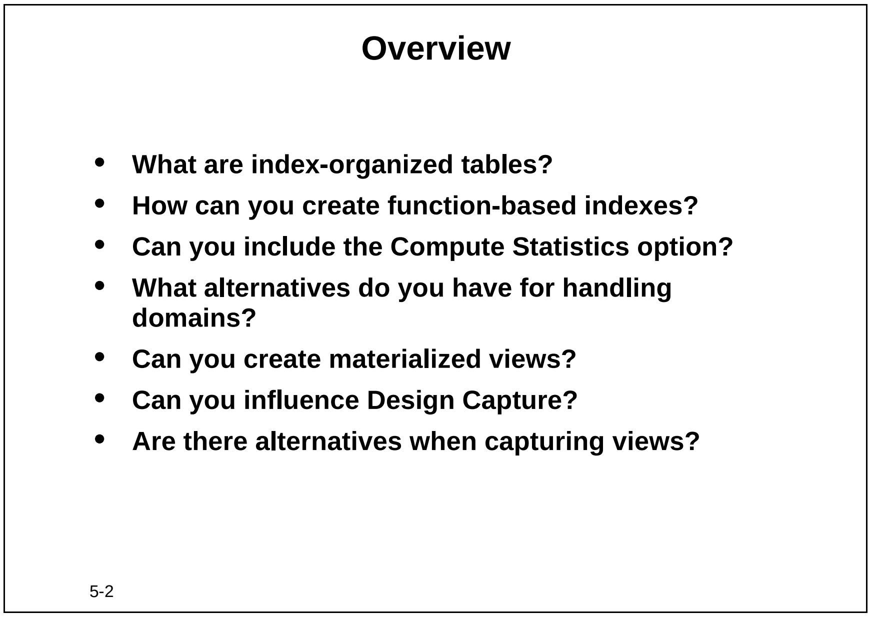
There is no practice for this lesson.

5

Exploring Data Design Features

Introduction

Some of the new features in Oracle8*i* have resulted in the addition of new element types to the Server Model in Oracle Designer 6*i*. As well as generating database objects for these elements, the Server Generator also produces Oracle8 database objects from some new properties of existing element types. In this lesson you define some of the objects that you can create and generate on the server side with Oracle8 and Oracle 8*i* databases.



Overview

- **What are index-organized tables?**
- **How can you create function-based indexes?**
- **Can you include the Compute Statistics option?**
- **What alternatives do you have for handling domains?**
- **Can you create materialized views?**
- **Can you influence Design Capture?**
- **Are there alternatives when capturing views?**

5-2

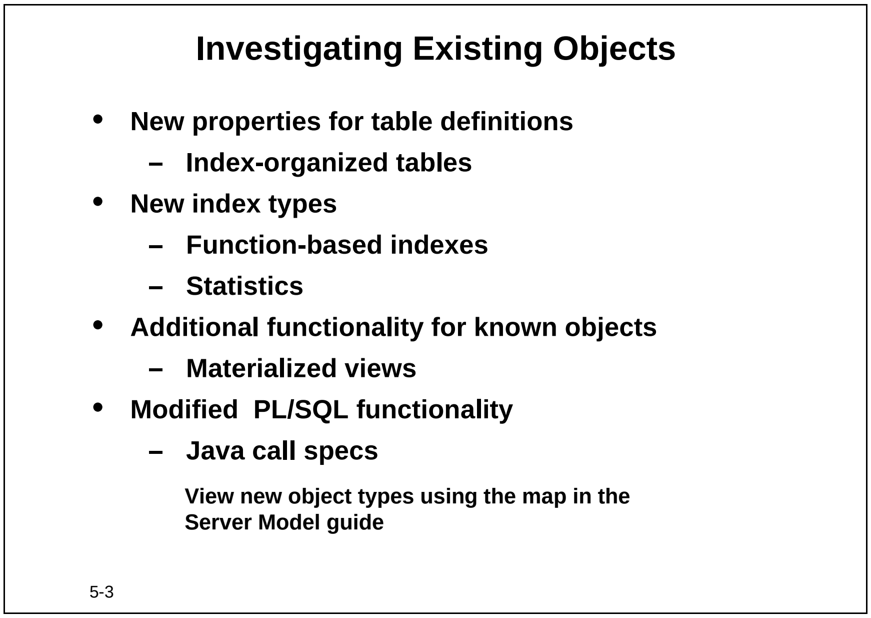
Objectives

At the end of this lesson, you should be able to:

- Create definitions of index-organized tables within your designs
- Implement index-organized tables
- Create definitions of function-based indexes for use in Oracle8*i* databases including the compute statistic option
- Develop and use domain key constraints
- Create definitions of materialized views and generate these
- Influence the behavior of the design capture process using preferences
- Capture views as declarative definitions with columns as secondary elements

Focus on the Server Model

In Oracle Designer 6*i* there are a number of new element types, and new properties for some of the existing elements available in earlier releases.



Investigating Existing Objects

- **New properties for table definitions**
  - **Index-organized tables**
- **New index types**
  - **Function-based indexes**
  - **Statistics**
- **Additional functionality for known objects**
  - **Materialized views**
- **Modified PL/SQL functionality**
  - **Java call specs**

View new object types using the map in the Server Model guide

5-3

This results in the Server Generator producing modified DDL, for tables and indexes, and a new type of PL/SQL database object for PL/SQL definitions when used with Java. The latter is discussed in the next lesson.

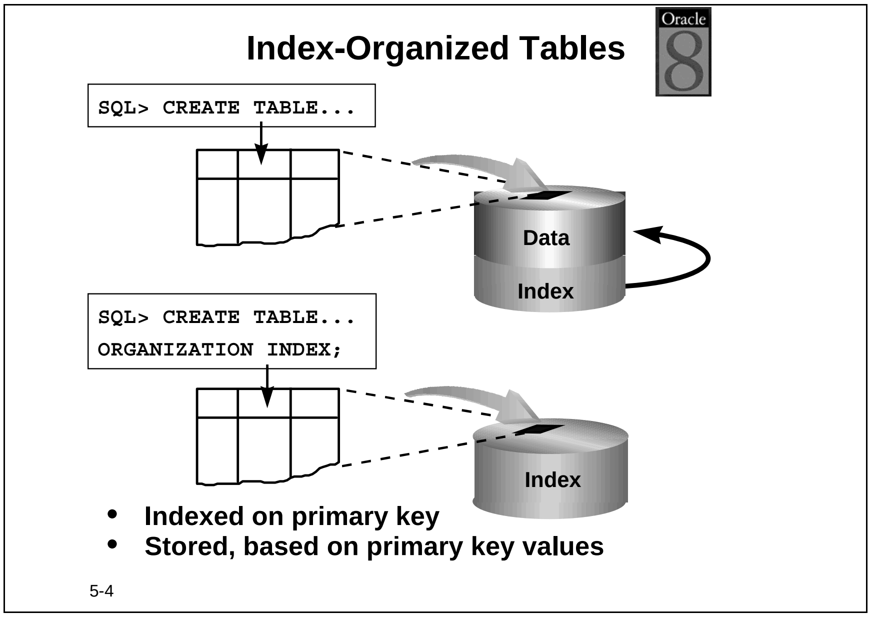
Viewing Objects

You can see the new element types in the map in the Server Model Guide, which illustrates all the element types of the Server Model. The new element types we discuss in the course are:

- Domain key constraints
- Materialized views
- Java database objects

Index-Organized Tables

An index-organized table is stored entirely as an index, based on the primary key.



Index-Organized Tables

```
SQL> CREATE TABLE...
```

```
SQL> CREATE TABLE...
ORGANIZATION INDEX;
```

- **Indexed on primary key**
- **Stored, based on primary key values**

5-4

If you have tables where the majority of columns form the primary key, there is a lot of redundant data stored, in the table and in the index. In Oracle8, instead of maintaining two separate storages for a table and its index, the database system only maintains the data in the associated index. If you add rows to the table, Oracle updates the index. The index holds both the encoded key value and the associated column values for the corresponding row.

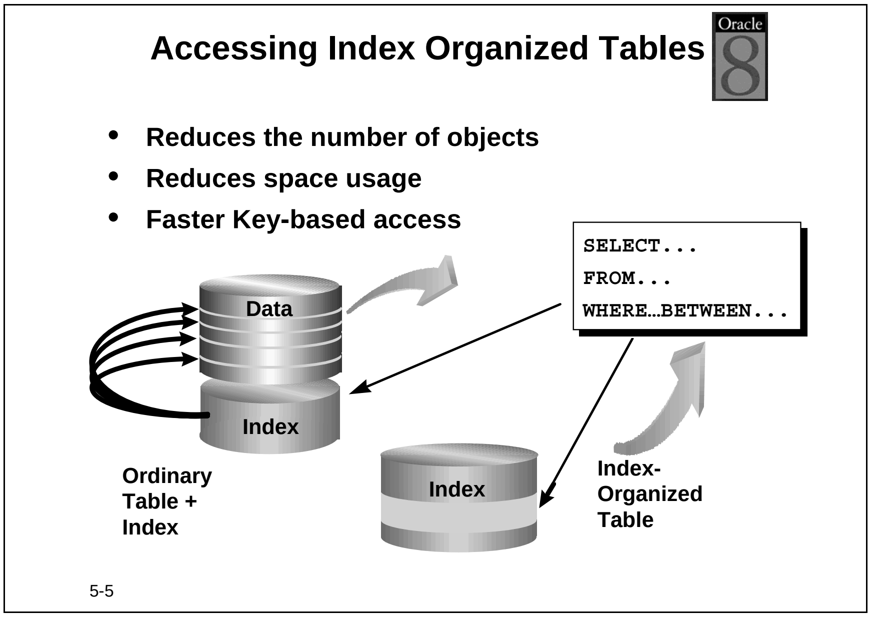
Uses

Index-organized tables are suitable for accessing data by the primary key or any key that is a valid prefix of the primary key.

Note

In Oracle8, you can build secondary indexes on index-organized tables to provide efficient access by other columns. Oracle Designer 6*i* does not support this.

Accessing Index-Organized Tables



Accessing Index Organized Tables

- Reduces the number of objects
- Reduces space usage
- Faster Key-based access

```
SELECT...
FROM...
WHERE…BETWEEN...
```

5-5

Performance

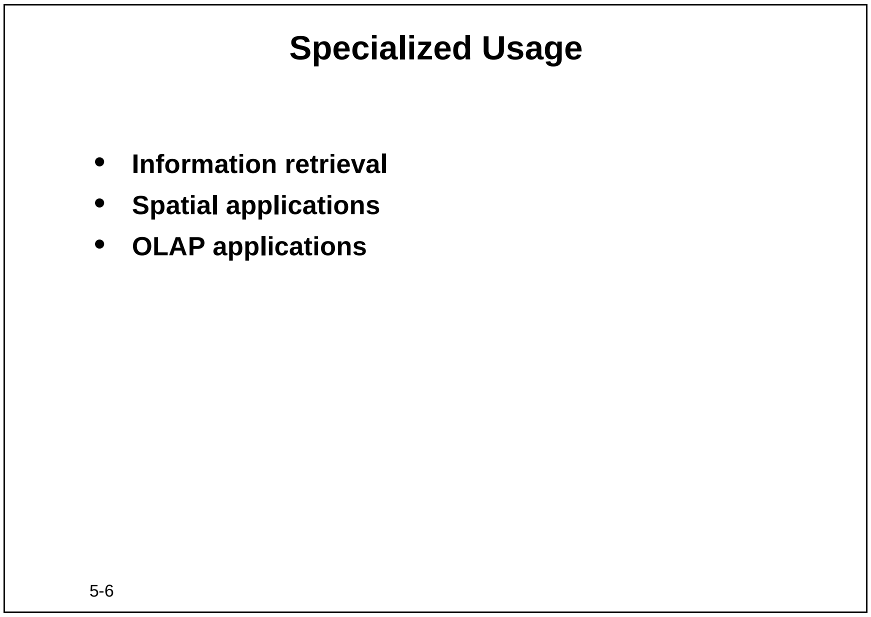
Because Oracle stores the data rows in the index, index-organized tables provide faster key-based access to table data for queries that involve:

- Exact match
- Range search
- A combination of an exact match and a range search.

Reduced Storage

Oracle stores index-organized tables in less space than a table and index because it does not duplicate key columns as is the case in an ordinary table and index. Also, because Oracle stores the data rows in the index, index-organized tables do not need to store the physical rowid to link the key values to corresponding rows in the table data.

Uses of Index-Organized Tables



Specialized Usage

- **Information retrieval**
- **Spatial applications**
- **OLAP applications**

5-6

Applications

Index-organized tables are best used in situations where the data is static. The reduced storage characteristics mean that index-organized tables are especially useful for the following types of applications:

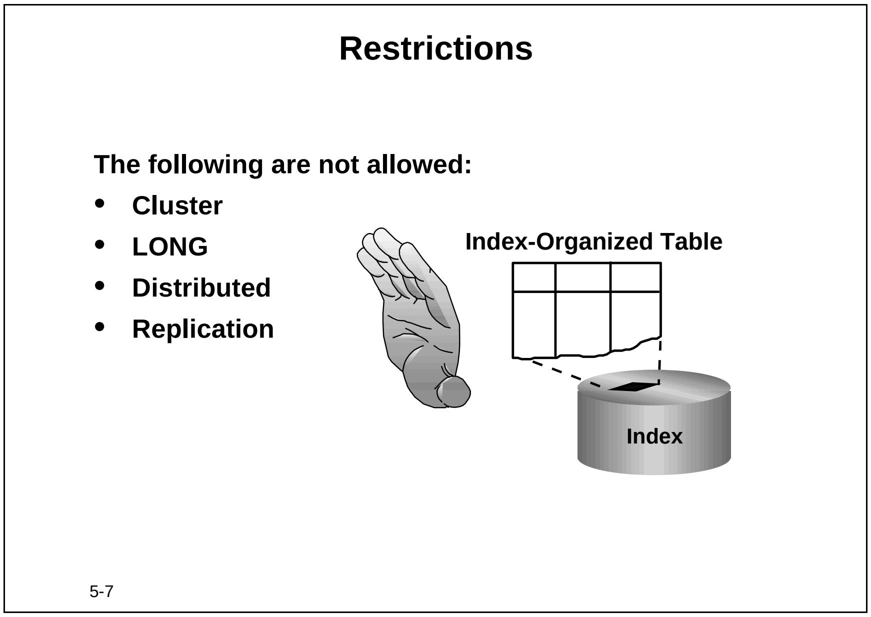
Application	**Description**
Information Retrieval (IR) applications	Support content-based searches on document collections. These typically maintain an index on each distinct word in a document.
Spatial applications	Typically maintain indexes for such things as objects residing in a collection of grids.
Online analytical processing (OLAP) applications	Typically manipulate multidimensional blocks and maintain an index to map a set of dimension values to a set of pages.

Example

An important use of an index-organized table occurs when the data in the index would be the same in the composite key and in the index table, that is, when the index and the table data maintain the same data. This occurs when you create a composite key in the intersection table that resolves a many-to-many relationship.

Restrictions

There are some restrictions on whether or not you can index-organize a table—not all tables are suitable.



Restrictions

The following are not allowed:

- **Cluster**
- **LONG**
- **Distributed**
- **Replication**

Index-Organized Table
Index

5-7

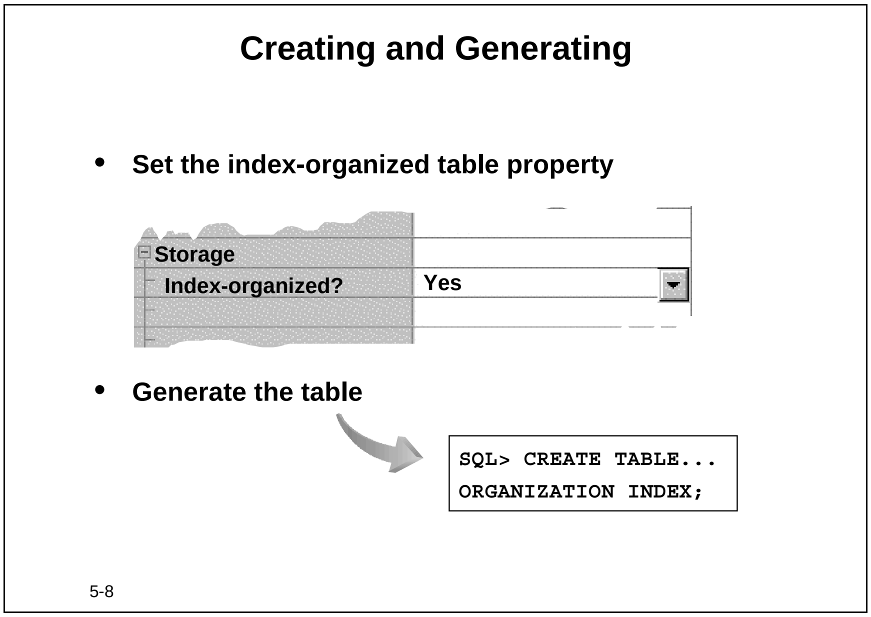
Types of Restriction

Index-organized tables:

- Cannot be stored in a cluster
- Can contain LOB columns but not LONG columns
- Do not support distribution, replication, and partitioning
- Cannot be object tables

Creating Index-Organized Tables

Using the property palette for a table definition, set the index-organized table property to Yes in the storage category.



Creating and Generating

- Set the index-organized table property

Storage

Index-organized? Yes

- Generate the table

```
SQL> CREATE TABLE...
ORGANIZATION INDEX;
```

5-8

Generating Index-Organized Tables

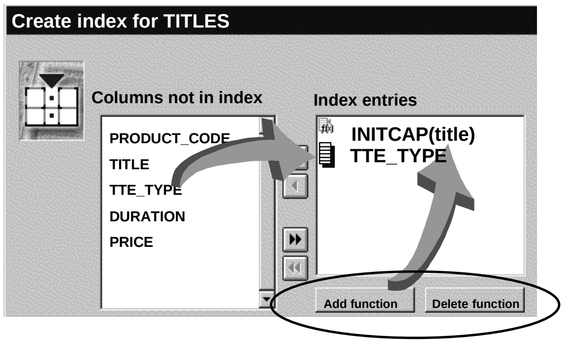
Generate the table as you would any table. The generated syntax includes the `ORGANIZATION INDEX` statement.

Defining Function-Based Indexes

A function-based index is an index on expressions. In Oracle 8*i* function-based indexes provide indexes for the database to use when evaluating queries that contain an expression, which could contain functions, and thereby improve performance.

Defining Function-Based Index

- **Single Entry Function-Based Indexes**



Create index for TITLES

Columns not in index

PRODUCT_CODE
TITLE
TTE_TYPE
DURATION
PRICE

Index entries

INITCAP(title)
TTE_TYPE

- **Multiple Entries Function-Based Index**

5-9

Creating a Function-Based Index

You can create a function-based index of functions and expressions that involve one or more columns in the table being indexed. The function-based index computes the value of the function or expression and stores it in the index. You can create a function-based index as either B*-tree or bitmap index.

The function you use for the index can be an arithmetic expression or an expression that contains a PL/SQL function, package function, C callout, or SQL function.

Restrictions

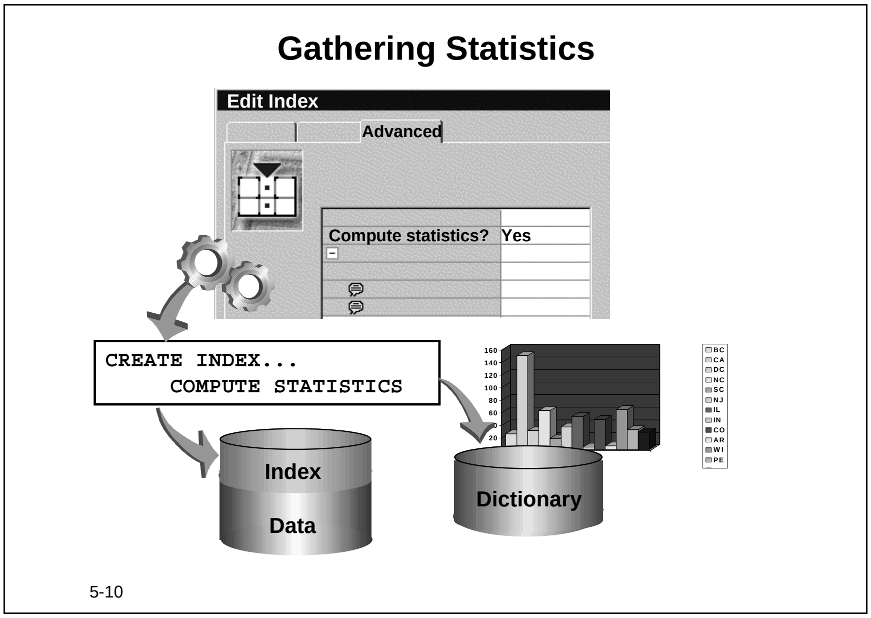
- The expression cannot contain any aggregate functions.
- Functions and procedures must produce repeatable values.
- The function must contain the `DETERMINISTIC` clause, in order to be used by function-based indexes.

How to Create a Function-Based Index

1. Use the property dialog and follow the process for creating an index.
2. At the **Create Index** dialog, instead of selecting columns you want included in the index, click the **Add Function** button and enter the function you require. These are not mutually exclusive, that is, you can include a function and a column in the index entry.

Gathering Statistics

You can specify that you want the Server Generator to include the compute statistics clause in the DDL it generates for an index.



Gathering Statistics

Compute statistics? Yes

```
CREATE INDEX...
     COMPUTE STATISTICS
```

The COMPUTE STATISTICS clause of the CREATE INDEX statement of the SQL language enables Oracle to collect statistics at relatively little cost during the creation of an index. Oracle stores these statistics in the data dictionary for ongoing use by the optimizer in choosing a plan of execution for SQL statements.

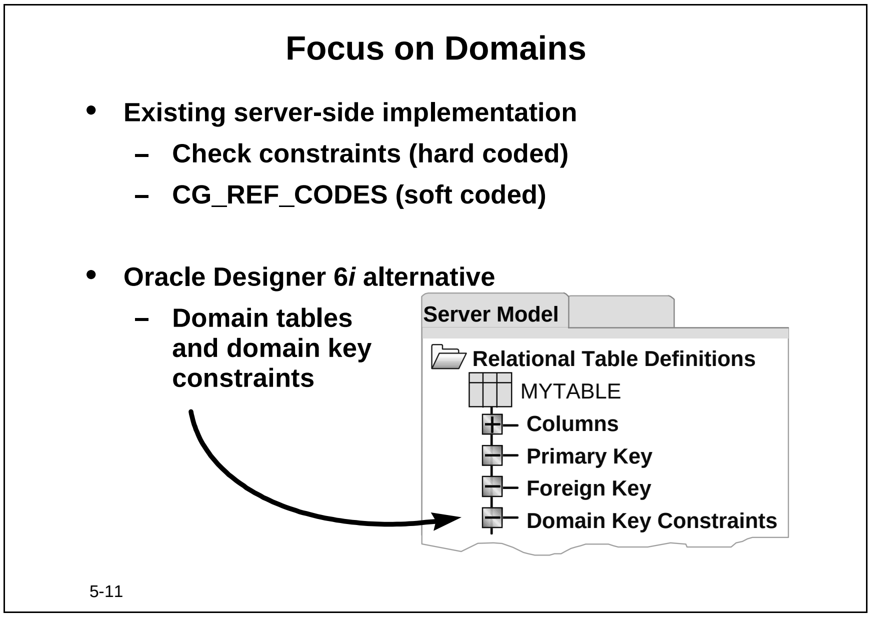
Including the Compute Statistics clause

You can specify that you want the Server Generator to include the compute statistics clause in the DDL it generates for an index by setting the Compute Statistics? property of the index to Yes.

This option is available for Oracle 8*i* only.

Focus on Domains

Oracle Designer 6*i* now provides a number of methods for specifying allowable values for columns. Domain key constraints in Oracle Designer 6*i* ensure that values entered in domain key columns in one table match the values in an associated domain table.



Focus on Domains

- **Existing server-side implementation**
  - **Check constraints (hard coded)**
  - **CG_REF_CODES (soft coded)**
- **Oracle Designer 6*i* alternative**
  - **Domain tables and domain key constraints**

5-11

Server Side Validation

There are a number of ways you can implement server-side validation.

- Existing methods include:
  - Check constraints
  - CG_REF_CODES
- Using Oracle Designer 6*i* you can build domain tables and domain key constraints.

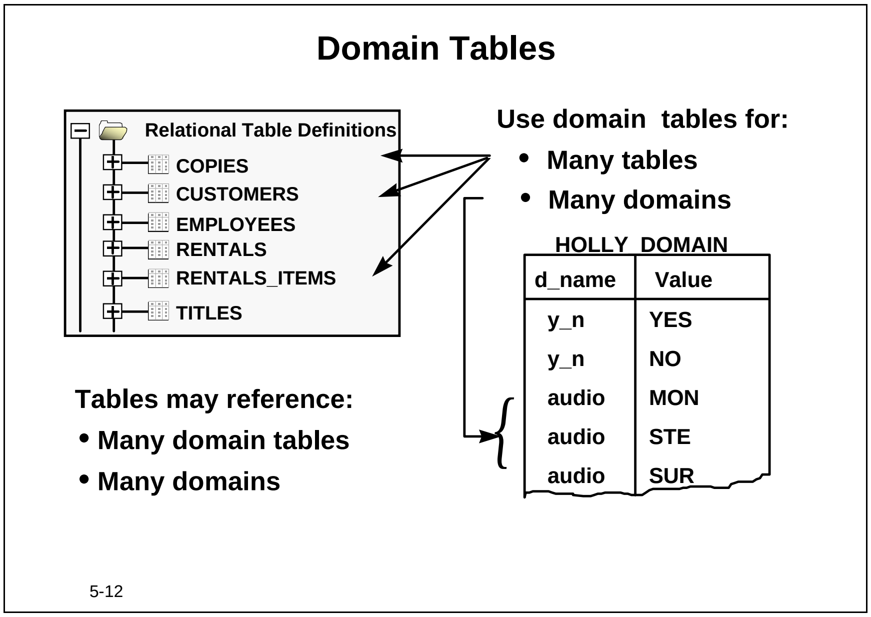
  Use the Server Generator to:
  - Generate one or more tables that will hold domain values.
  - Generate code to access the tables and to perform validation against the domains stored in the tables.

Defining Domain Key Constraints

To create a Domain Key Constraint you create a secondary element type “Domain Key Constraint” within a table definition.

Domain Constraint Tables

A domain table allows you to model domains using a normal table rather than the predefined reference code table, such as CG_REF_CODES.



Domain Tables

Relational Table Definitions
- COPIES
- CUSTOMERS
- EMPLOYEES
- RENTALS
- RENTALS_ITEMS
- TITLES

Use domain tables for:
- Many tables
- Many domains

HOLLY_DOMAIN

d_name	Value
y_n	YES
y_n	NO
audio	MON
audio	STE
audio	SUR

Tables may reference:
- Many domain tables
- Many domains

5-12

- A domain table can contain one or more sets of valid values.
- You can model one or more tables as domain tables or design capture existing tables which hold values you wish to use.

Using domain tables

You can use a domain table to:

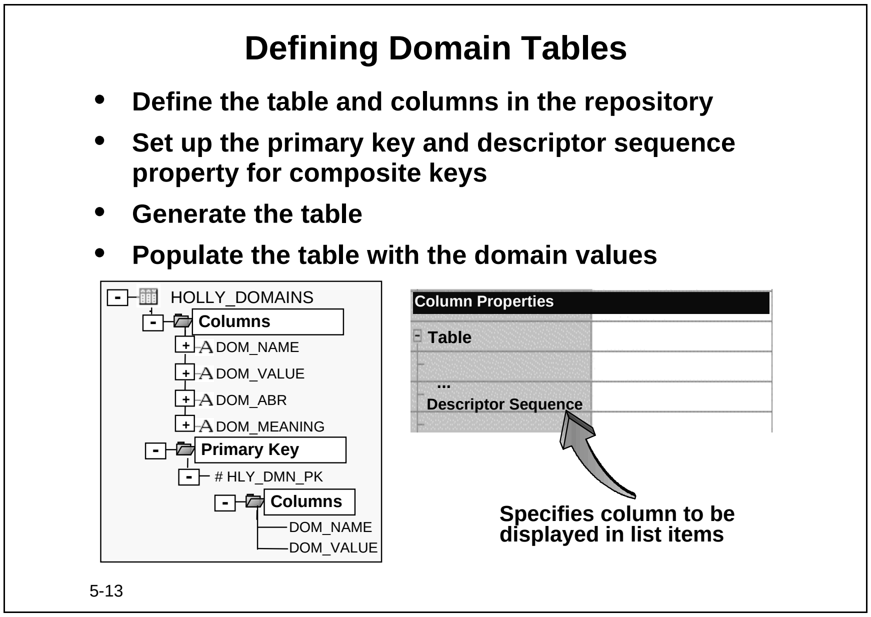
- Populate:
  - Pop-lists
  - Text lists
  - Combo boxes
  - LOVs
- Perform item validation

Items are populated and validated in a similar way to the dynamic implementation of allowable values modeled in the repository.

Are domain tables associated with defined allowable values?

A domain table has no connection with allowable values defined (either in a domain or directly against a column) in the repository. The main difference between a domain table and a reference code table is that it is the data that is modeled in the repository for a reference code table, rather than the table itself. A domain table is not linked to other tables via standard foreign keys.

Creating a Domain Table



Defining Domain Tables

- **Define the table and columns in the repository**
- **Set up the primary key and descriptor sequence property for composite keys**
- **Generate the table**
- **Populate the table with the domain values**

Specifies column to be displayed in list items

5-13

Defining a Domain Table

You can use any table as a domain table. For instance, you may have an existing table which holds valid values applicable to your application such as a table of read-only postal codes. You can capture the design of this table into the repository.

The columns in a domain table do not need to follow any predefined model although you may wish to include columns which can map to the following generic format:

Column	**Description**	**Example**
DOM_NAME	Name of the domain	Movie_Type
DOM_VALUE	Value of the domain	Drama
DOM_ABR	Abbreviation of domain value	DRA
DOM_MEANING	Meaning of the domain value	Drama Movies

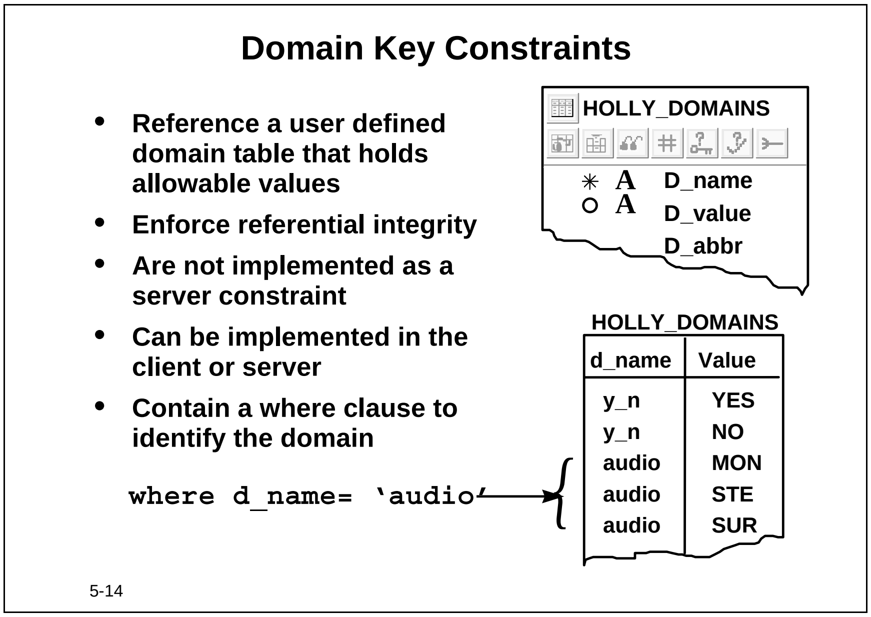
- If the domain table holds more than one domain, you need to create a composite primary key of the domain name and value.
- If you are going to use the domain table to populate a GUI item, you need to set the Descriptor Sequence column property to specify which column should be displayed.

Implementing a Domain Table

If you want to create a domain table in the repository it should be generated into the database using the Server Generator. You are responsible for entering the values into the table. You may decide to incorporate a form into your application to enable users to input the data.

Domain Key Constraints

A domain key constraint joins an application table with values in a domain table. It is created on a column in the application table and restricts the values available to that column to values held in the domain table, as such enforcing the integrity of the data in the related tables.



Domain Key Constraints

- Reference a user defined domain table that holds allowable values
- Enforce referential integrity
- Are not implemented as a server constraint
- Can be implemented in the client or server
- Contain a where clause to identify the domain

```
where d_name= 'audio'
```

HOLLY_DOMAINS

d_name	Value
y_n	YES
y_n	NO
audio	MON
audio	STE
audio	SUR

5-14

Using Domain Key Constraints

Model domain constraint tables as you would any other table.

Domain key constraints:

- Allow the application to access a range of domain values
- Are not supported natively on the server

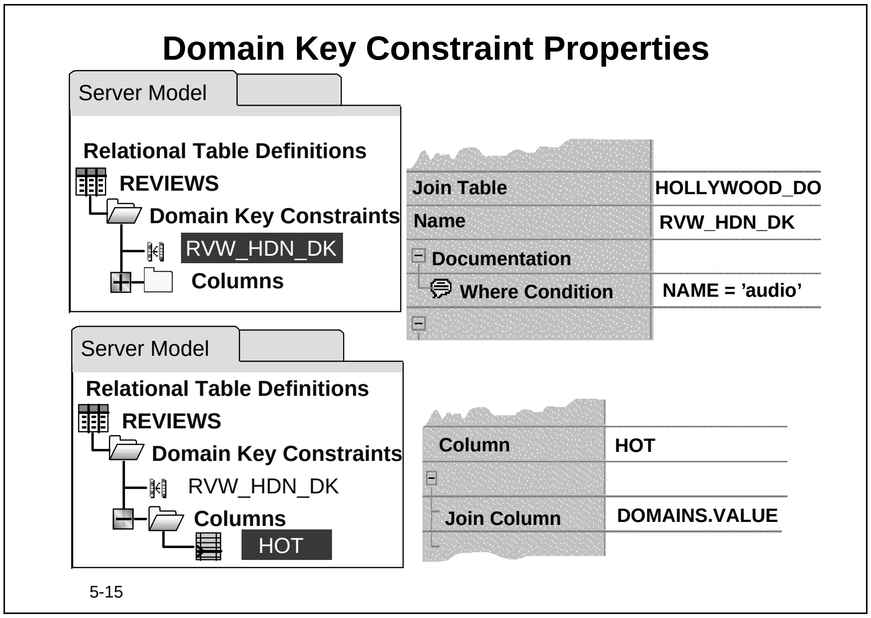
Implementing Domain Key Constraints

- On the client: During generation, the client generators create code for DML in the same way they generate code to implement foreign keys on the client.
- On the server: Generate the table API to implement the code on the server.

Each domain in a multiple domain table can be identified using a `WHERE` clause in the Domain Key Constraint.

Using Domain Key Constraints

In order to use domain key constraints there are several tasks to perform.



Domain Key Constraint Properties

Server Model

Relational Table Definitions
REVIEWS
Domain Key Constraints
RVW_HDN_DK
Columns

Join Table	HOLLYWOOD_DO
Name	RVW_HDN_DK
Documentation	
Where Condition	NAME = 'audio'

Server Model

Relational Table Definitions
REVIEWS
Domain Key Constraints
RVW_HDN_DK
Columns
HOT

Column	HOT
Join Column	DOMAINS.VALUE

5-15

How to Define a Domain Key Constraint

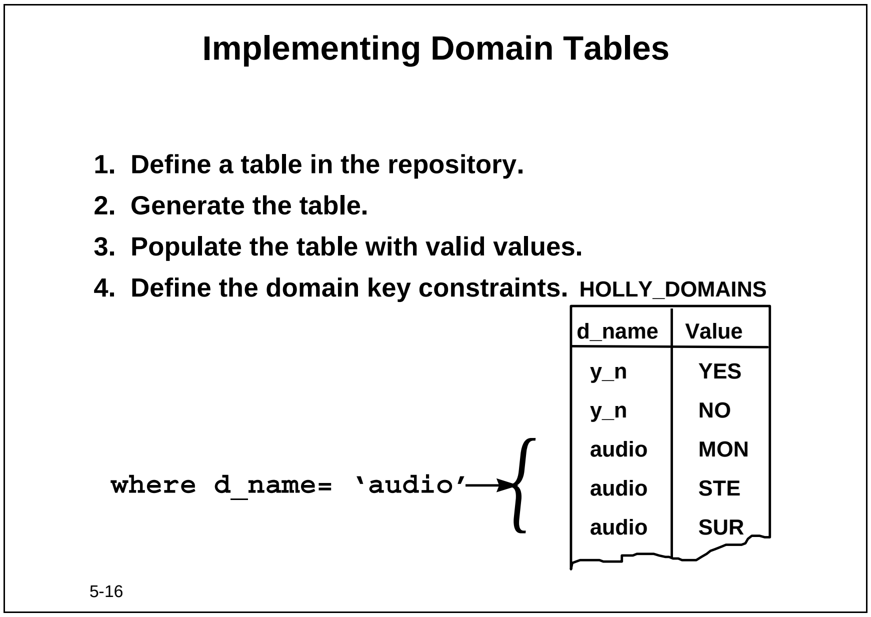
1. In the navigator expand the node, such as a relational table or view, in which you want to create the domain key constraint, and then select the **Domain Key Constraints** node.
2. Create a new domain key constraint, setting the required properties:

Property	Description
Join table	The table that contains the domain values.
Constraint name	This is automatically populated if you use the property dialog.
Mandatory	This forces a value to be specified for the key column.
WHERE clause	This identifies the specific domain in a domain table that contains multiple domains.
Validate In	Specify where the validation is carried out; on the client, server, or both.
Error Message	These to be displayed if constraint violated

3. Add a column to the domain key constraint:

Property	Description
Column(s)	Column(s) in the current table. This will be validated using the domain key constraint.
Join column	This is the column in the domain constraint table that you want to join to.

How to Implement Domain Tables



Implementing Domain Tables

1. Define a table in the repository.
2. Generate the table.
3. Populate the table with valid values.
4. Define the domain key constraints.

HOLLY_DOMAINS

d_name	Value
y_n	YES
y_n	NO
audio	MON
audio	STE
audio	SUR

```
where d_name= ‘audio’
```

5-16

1 Define the table in the repository.

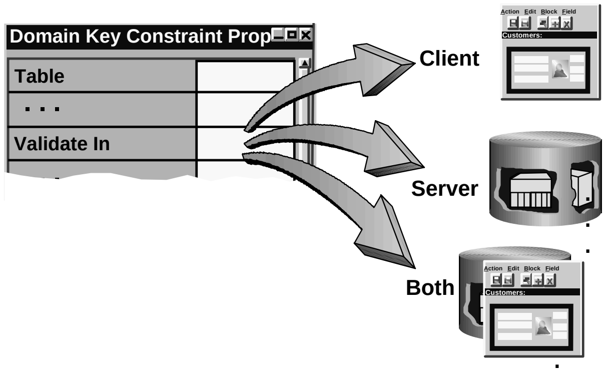
2 Generate and populate the table with valid values.

3 In the definition of each table where you want to use the domain constraint tables, create a domain key constraint definition.

4 Name the domain in each domain key constraint definition. You must also specify for which columns the domain key constraint is intended.

Enforcing Domain Key Constraints

Where and When are they enforced?



5-17

Where and When are Domain Key Constraints Enforced?

As with primary, foreign and other constraint keys, you can choose, on a constraint by constraint basis, where to enforce the Domain Key Constraint using its Validate In property.

Client Side Validation

If you implement domain key constraints on the client, the constraints are enforced for the application only.

Server-side Validation

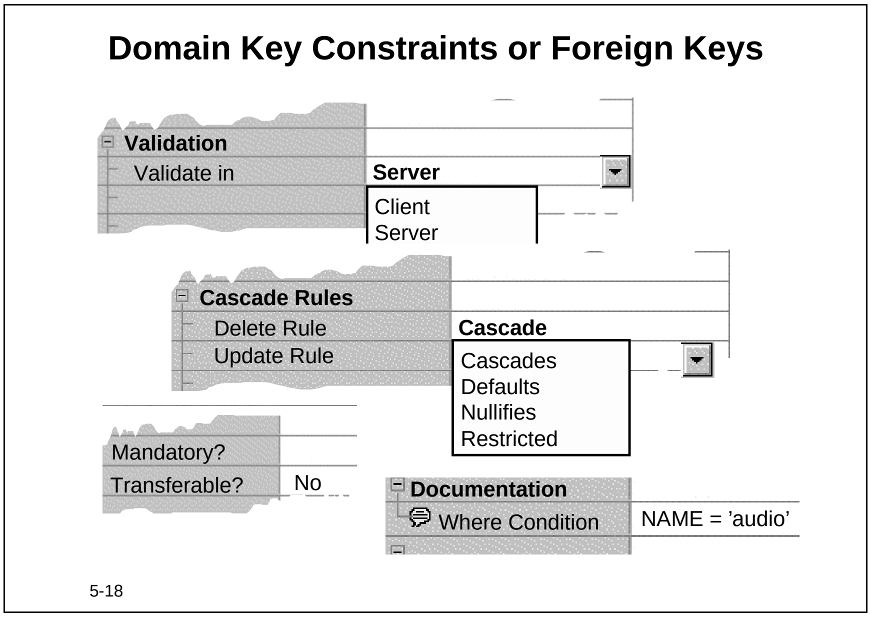
You need to generate the table API if you choose server-side validation for domain key constraints. The Server Generator generates:

- Code in the table API to implement validation
- Cascade rules (Cascade, Nullify, or Default)
- Mandatory and non transferability characteristics for the domain key constraint.

The Mandatory? property tells the Server Generator whether to create a mandatory (NOT NULL) table column or an optional table column.

Domain Key Constraints and Foreign Key Constraints

Domain key constraints and foreign key constraints are very similar. Both types of constraint enable you to join tables.



Domain Key Constraints or Foreign Keys

Validation

Validate in: Server

Cascade Rules

Delete Rule: Cascade

Update Rule

Mandatory?

Transferable?: No

Documentation

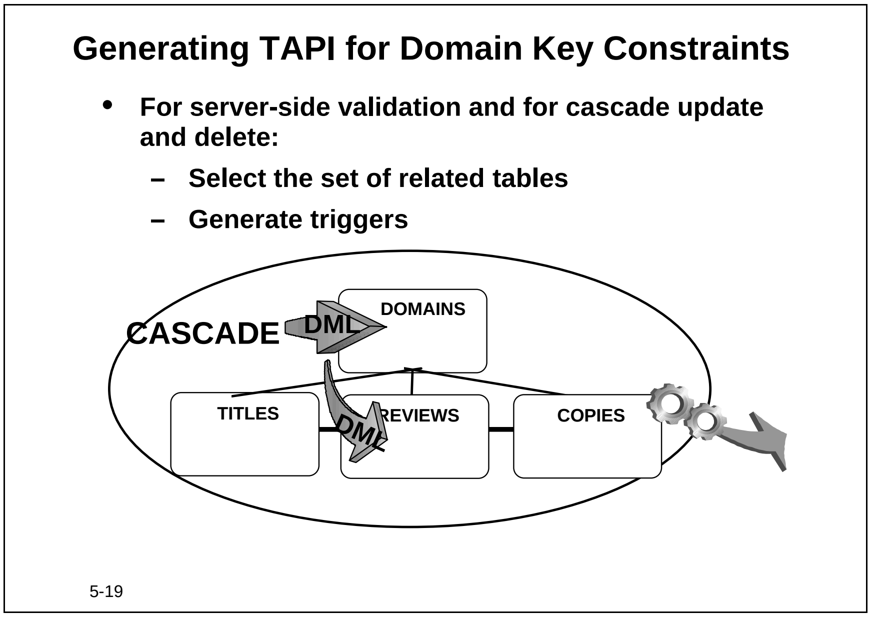
Where Condition: NAME = 'audio'

5-18

Although domain key constraints are primarily designed to support joins to domain tables, you can use a domain key constraint in the same way as a foreign key constraint to join module component table usages.

One difference is that domain key constraints include a WHERE Condition to enable you to restrict the values returned from the table at the other end of the domain key, allowing you to have multiple domains in one domain table.

Generating TAPI for Domain Key Constraint



Generating TAPI for Domain Key Constraints

- For server-side validation and for cascade update and delete:
  - Select the set of related tables
  - Generate triggers

5-19

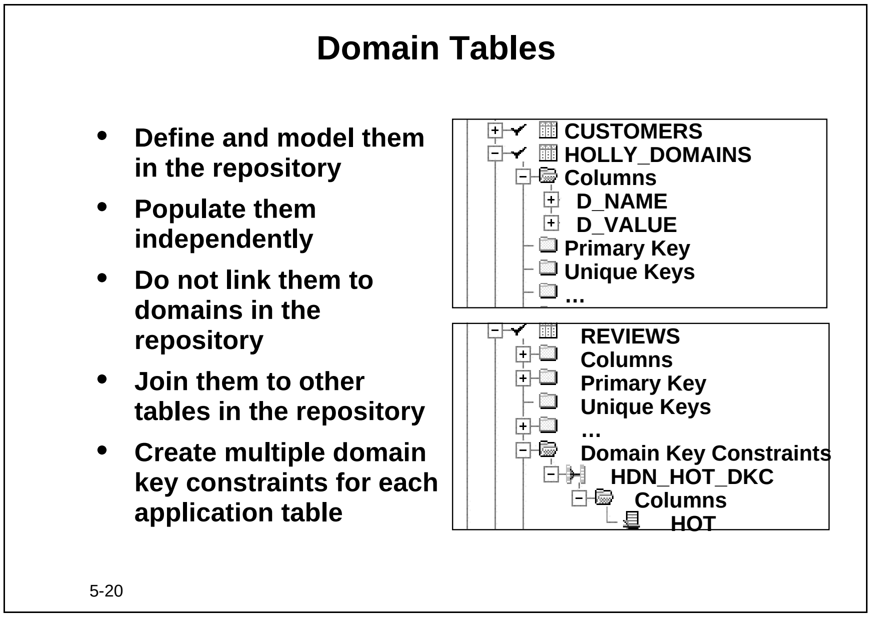
Using the Table API to Implement Domain Key Constraints on the Server

The main difference between domain and foreign key constraints is that domain key constraints are not natively supported on the server. Domain key constraints enable you to model a non-foreign key based join, and generate code into an application to enforce the constraint. If you want to enforce a domain key constraint on the server, you must generate the Table API.

Note

When you generate the table API for a domain key constraint, for the run-set you must select all the tables that use the domain constraint table. The reason for this is that the table API puts into single triggers the code necessary for all the tables involved. Selecting all the tables that use the domain constraint table gives the Server Generator the information about which tables are involved.

Advantages and Disadvantages of Domain Tables



Domain Tables

- **Define and model them in the repository**
- **Populate them independently**
- **Do not link them to domains in the repository**
- **Join them to other tables in the repository**
- **Create multiple domain key constraints for each application table**

5-20

Using domain tables instead of a single reference code table, such as CG_REF_CODES, allows you to:

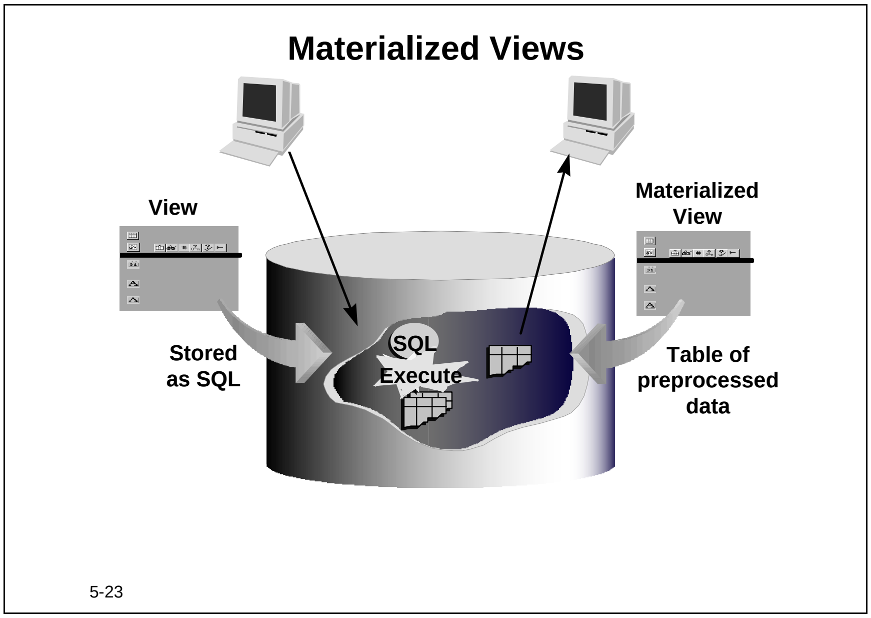
- Define the domain table in the repository
- Define the columns you want in the domain table

You need to populate the domain table yourself. This means that you are responsible for entering and maintaining the data in the table as you would be if you modeled and generated code tables.

Note: Instead of using a single CG_REF_CODES reference code table, many developers create multiple individual code tables to hold valid values. They use foreign keys to maintain the data integrity. By creating domain tables and domain key constraints instead, you need not create as many code tables, so avoiding the many foreign keys that you would need to create between the code tables and the tables they reference.

Defining Materialized Views

A materialized view in Oracle8*i* is not a new object, but it is equivalent to a snapshot in earlier versions of the database.



Materialized Views

5-23

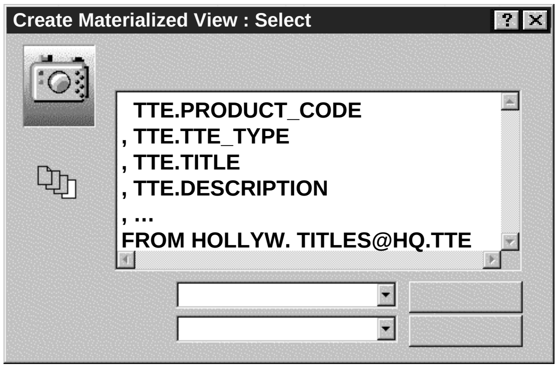
Comparing Views and Materialized Views

An important difference between a materialized view and an ordinary view is that a materialized view is a table of data that has been preprocessed or materialized. Ordinary views comprise stored SQL that must be executed whenever the view is accessed.

Creating Materialized Views

Defining a Materialized View

- Use the Design Editor Server Model tab
- Use free format to define the query



Create Materialized View : Select

```
  TTE.PRODUCT_CODE
, TTE.TTE_TYPE
, TTE.TITLE
, TTE.DESCRIPTION
, ...
FROM HOLLYW. TITLES@HQ.TTE
```

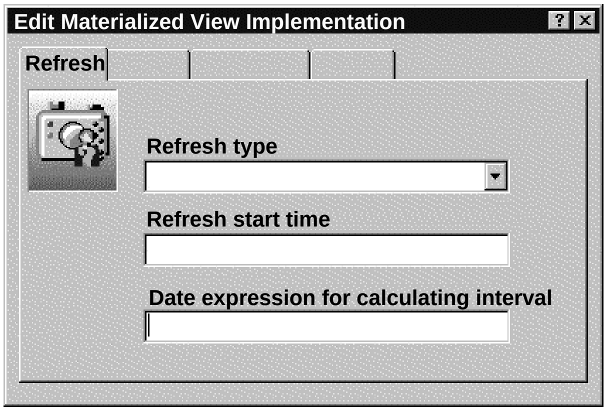
5-24

You can create materialized views in the Design Editor using the Server Model tab. You can use either a declarative definition or a free-format definition. In general you need to use the free-format definition if you want to specify the remote master table fully. Remember that you omit the SELECT when creating a free-format definition.

Materialized View Implementation

Define the materialized view refresh parameters in the Materialized View Implementation. You can specify multiple implementations of the materialized view at different locations and specify different refresh parameters for each if you wish.

Materialized View Implementation



Edit Materialized View Implementation

5-25

Defining Materialized View Logs

Define Materialized View Log

- **Define materialized view log on master table**
- **Set property in Table Implementation**

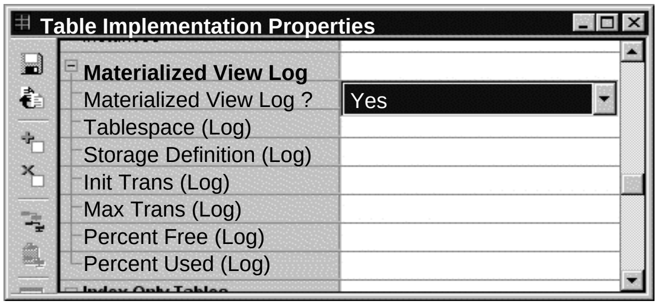


Table Implementation Properties

Materialized View Log
- Materialized View Log ?: Yes
- Tablespace (Log)
- Storage Definition (Log)
- Init Trans (Log)
- Max Trans (Log)
- Percent Free (Log)
- Percent Used (Log)

Index Only Tables

5-26

Define a materialized view log on the master table implementation for the materialized view that you have defined. The materialized view log enables fast updates on simple materialized views.

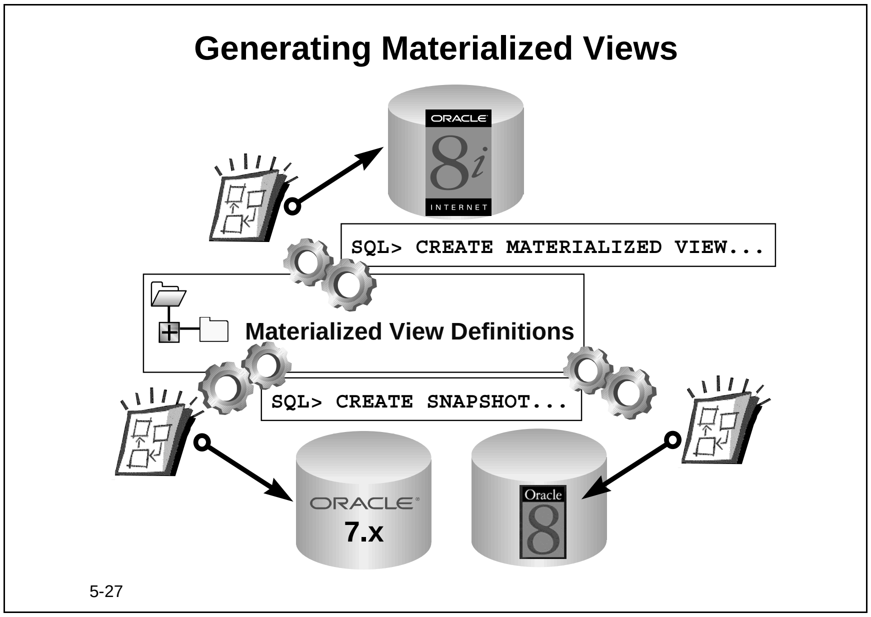
A simple materialized view is defined as one based on a single master table which does not have any of the following features in the defining query:

- GROUP BY clause
- CONNECT BY clause
- Distinct or aggregate functions
- Joins (other than allowed restricted subqueries)
- Set operators

A materialized view log records the changes in the master table that need to be propagated to the materialized views on the next refresh. These refreshes are not required to take place simultaneously.

Generating Materialized Views

When you generate a materialized view, the Server Generator produces different DDL depending on the target database.



Generating Materialized Views

```
SQL> CREATE MATERIALIZED VIEW...
```

```
SQL> CREATE SNAPSHOT...
```

If you selected Oracle8*i* as the target database the generator produces a `CREATE MATERIALIZED VIEW` statement. If you select Oracle8 or versions of Oracle7, the Server Generator produces a `CREATE SNAPSHOT` statement.

In Oracle8*i* materialized views and snapshots are synonymous.

Table API Changes

Triggers in the table API are optional.

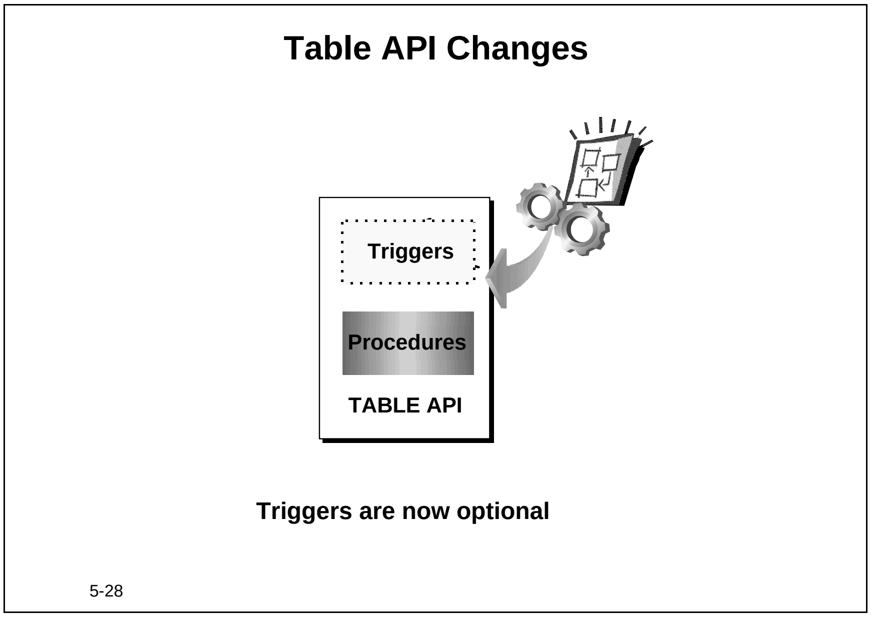


Table API Changes

Triggers

Procedures

TABLE API

Triggers are now optional

5-28

What is Meant by Optional?

Packaged stored procedures perform the main processing of the table API. To provide additional validation on the server you can also create a number of database triggers as part of the table API. These were always created in earlier releases (except Oracle Designer 2.1) but their generation is now optional.

Choosing Triggers

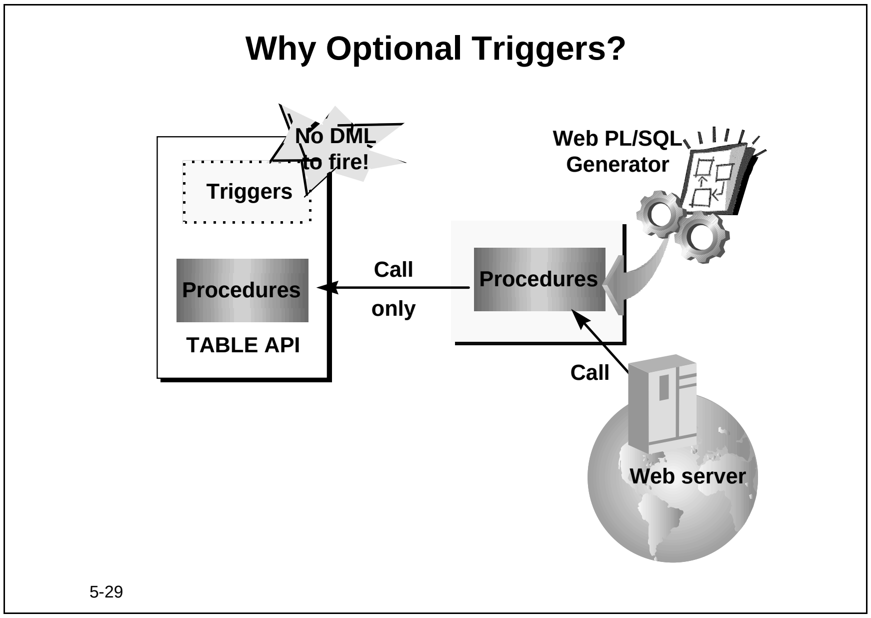
You can specify whether you want Server Generator to produce database triggers as part of a table API. You do this by selecting the Generate Table API Triggers check box in the Target tab of the Generate Table API dialog.

Invoking the Generate Table API Dialog

To invoke the Generate Table API dialog you use the Generate menu option:
Generate –> Generate Table API.

Why Optional Triggers?

The main purpose of the table API database triggers is to trap any direct DML against the table and to call the packaged stored procedures. This is to ensure that the table API procedures are always executed. However, for some types of application the table API triggers may not be necessary.



Why Optional Triggers?

5-29

Applications that May not Require Table API Triggers

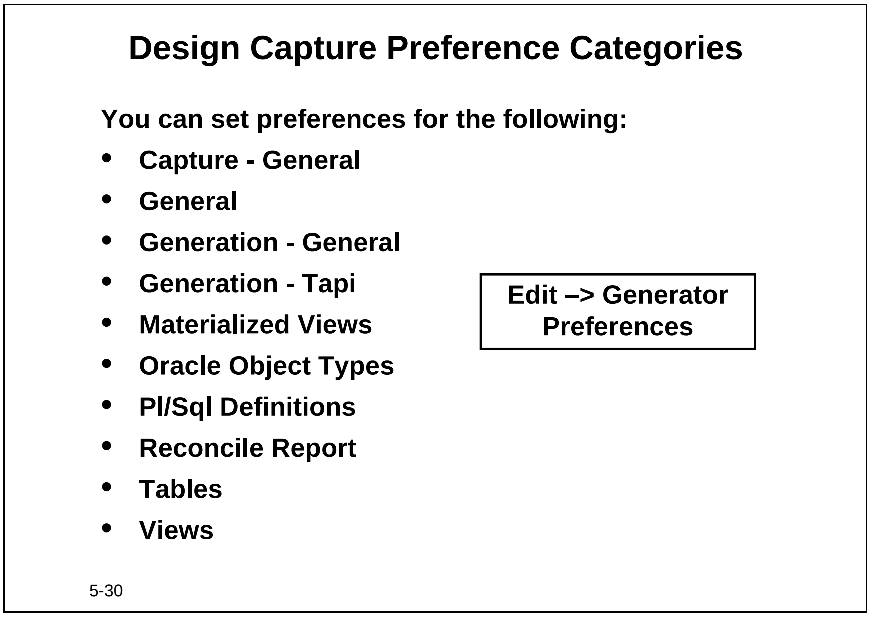
Webserver applications generated by Oracle Designer 6*i* perform all processing by means of calls to the table API procedures. If the application environment is controlled, so that you know that nobody can access the database using other than your application, (for example, no SQL*Plus access) you know there will be no direct DML and you may not need the table API triggers.

Note

Oracle Designer 6*i* does use the table API triggers to perform some special operations such as validating allowable values, implementing cascade rules, and validating arcs. You will still need the triggers if your application uses these mechanisms.

Design Capture Preferences

Design Capture now has a wide range of preferences available.



Design Capture Preference Categories

You can set preferences for the following:

- **Capture - General**
- **General**
- **Generation - General**
- **Generation - Tapi**
- **Materialized Views**
- **Oracle Object Types**
- **Pl/Sql Definitions**
- **Reconcile Report**
- **Tables**
- **Views**

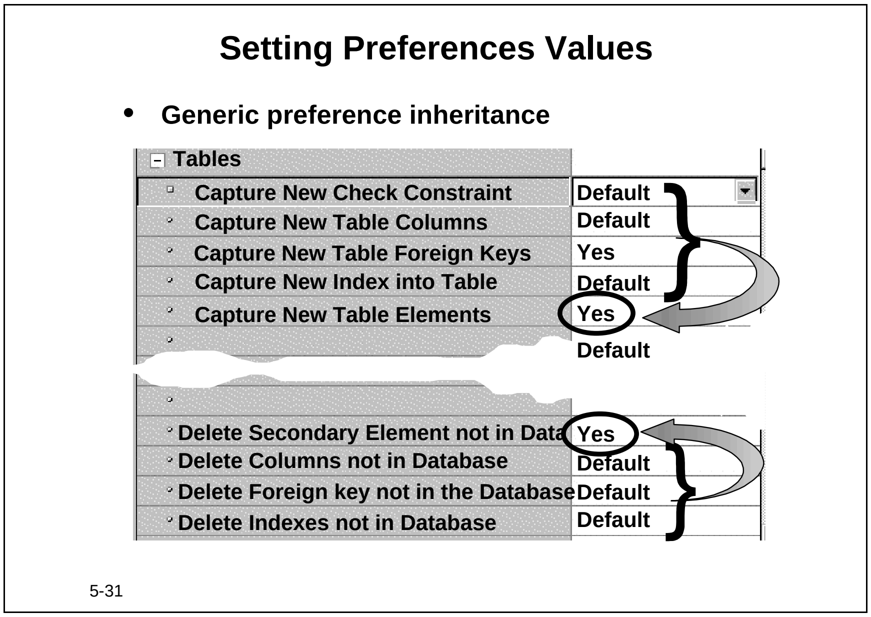
Edit –> Generator Preferences

5-30

Design Capture preferences are arranged into ten categories. There are over a hundred preferences in all. With these you can influence the behavior of Design Capture to give the results you require.

To set these preferences use the Edit menu **Edit –> Generator Preferences**.

Setting Preferences Values



Setting Preferences Values

- **Generic preference inheritance**

Tables	
Capture New Check Constraint	Default
Capture New Table Columns	Default
Capture New Table Foreign Keys	Yes
Capture New Index into Table	Default
Capture New Table Elements	Yes
	Default
Delete Secondary Element not in Data	Yes
Delete Columns not in Database	Default
Delete Foreign key not in the Database	Default
Delete Indexes not in Database	Default

5-31

These are some of the twenty-nine preferences available to influence capture of tables.

Some preferences may take a default value. This default value refers then, to a more generic property that dictates the behavior of this type of element. For example, check constraints, columns, and foreign keys are secondary elements of a table. Setting the preference **Capture New Table Elements (DADNST)** to Yes, becomes the default behavior for all related properties.

Table Capture Example

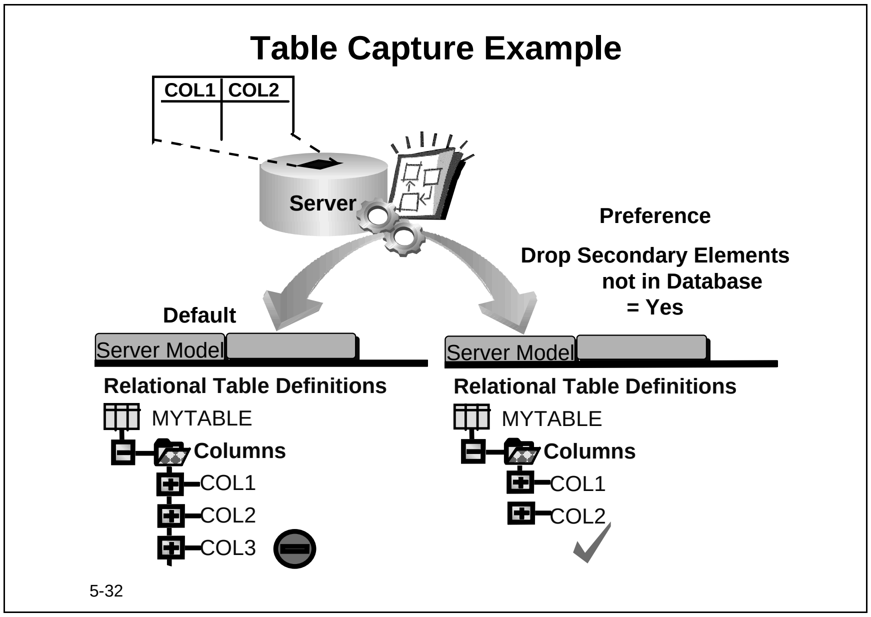


Table Capture Example

Preference

Drop Secondary Elements
not in Database
= Yes

Example

Consider running Design Capture to capture a table for which there is an existing definition in the repository. Assume there is a column defined in the repository but that the column does not exist in the database table. Assume the database table is correct and therefore you want to update the repository definition.

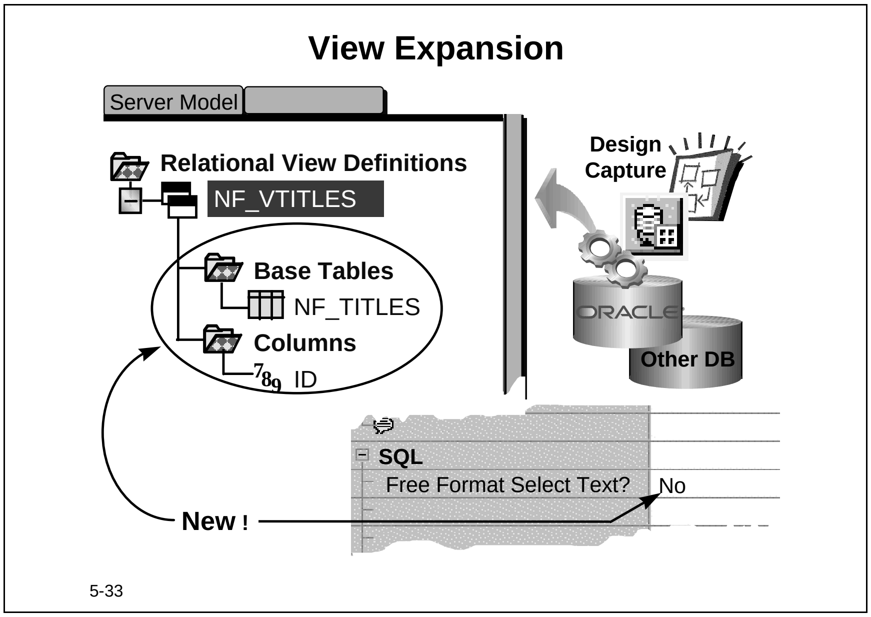
With default preference settings, Design Capture gives precedence to the repository definition and does not remove the column definition.

By setting the preference **Delete Columns not in the Database (DECOFT)**, the Server Generator deletes any columns not in the database, during design capture.

If you change the preference, **Delete Secondary Elements not in the Database (DDESFT)** to Yes, it alters the behavior of Design Capture and Oracle Designer 6*i* removes the secondary elements such as columns, check constraints and the foreign keys that are no longer associated with the table definition.

View Expansion

Design Capture can now create declarative definitions of views.



You should recall that you can define views by two methods:

- Declarative: Where you pick the tables and columns on which you want to base your view in the Table Selection and Columns dialogs. This method has limited complexity but handles most cases. You should use this method if possible.
- Free Format: Where you enter the full text of the SQL statement.

Providing the view is capable of being defined declaratively, Design Capture will create a declarative definition. Previous releases always created a Free Format definition.

Views that Oracle Designer 6*i* cannot capture declaratively include commands such as:

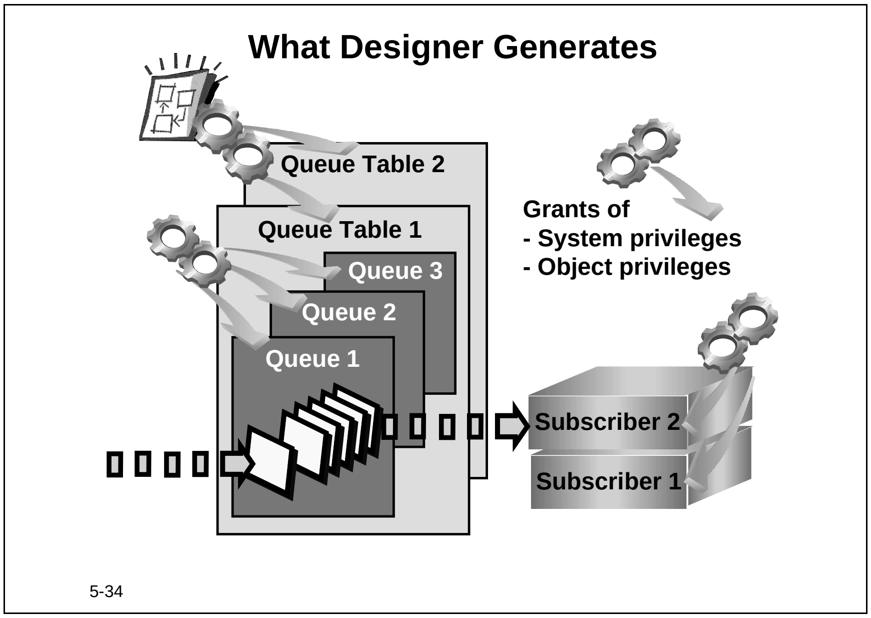
- Unions
- Functions
- From clause queries

Oracle8*i* Advanced Queuing

Oracle8*i* integrates a message queuing system with the Oracle server. Message queuing is becoming popular for distributed processing because it enables asynchronous communication. Oracle Advanced Queuing allows you to store messages in queues for deferred retrieval and processing. Because Oracle8*i* integrates the message queuing system in the server it enables message queuing without additional software and it provides the operational benefits of a database.

Oracle Designer 6*i* Support for Oracle Advance Queuing

Oracle Designer 6*i* generates the server-side components of the Oracle Advanced Queuing infrastructure.



What Designer Generates

Queue Table 2

Queue Table 1

Queue 3

Queue 2

Queue 1

Grants of
- System privileges
- Object privileges

Subscriber 2

Subscriber 1

5-34

In Oracle Designer 6*i* you can define and generate:

- Queue tables
- Queues
- Subscribers
- Grants of Oracle AQ specific system privileges and object privileges

Queue Tables

Queue tables store queues. Each queue table is a database table and can contain one or more queues.

Queues

A queue is the repository for messages. A message in a queue consists of payload data plus control properties. Oracle AQ uses the message control properties to manage the message through message queuing process.

Subscribers

A subscriber is a rule-based recipient.

Oracle AQ Privileges

There are several Oracle privileges that a user must be granted in order to be able to use Oracle AQ. You can generate the commands using Oracle Designer 6*i*. The commands are not the usual GRANT statements. Instead the commands use a special package for Oracle AQ named DBMS_AQADM. For example

```
EXECUTE DBMS_AQADM.GRANT_SYSTEM_PRIVILEGE(
   privilege         =>    'ENQUEUE_ANY',
   grantee           =>    'Jones',
   admin_option       =>     FALSE);
```

Oracle Designer 6*i* generates these statements for you:

System Privileges

Privilege	**Description**
ENQUEUE_ANY	Users granted this privilege are allowed to enqueue messages to any queues in the database.
DEQUEUE_ANY	Users granted this privilege are allowed to dequeue messages from any queues in the database.
MANAGE_ANY	Users granted this privilege are allowed to execute DBMS_AQADM calls on any schemas in the database.

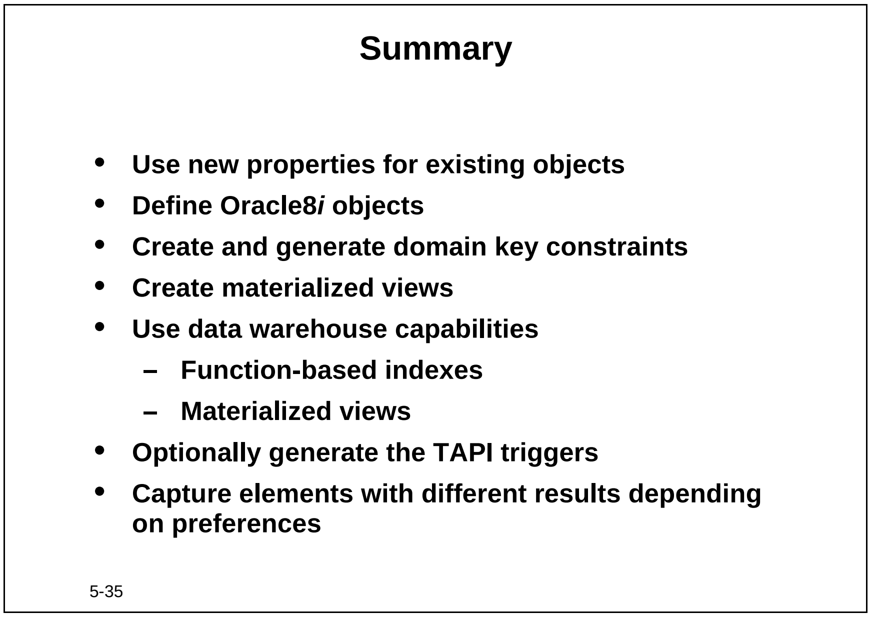
Object Privileges

Privilege	**Description**
ENQUEUE	Users granted this privilege are allowed to enqueue messages to a specific queue.
DEQUEUE	Users granted this privilege are allowed to dequeue messages from a specific queue.

Further Information

A full description of message queuing and the capabilities of Oracle AQ is beyond the scope of this course. You should consult your Oracle8*i* documentation for full coverage.

Summary



Summary

- **Use new properties for existing objects**
- **Define Oracle8*i* objects**
- **Create and generate domain key constraints**
- **Create materialized views**
- **Use data warehouse capabilities**
  - **Function-based indexes**
  - **Materialized views**
- **Optionally generate the TAPI triggers**
- **Capture elements with different results depending on preferences**

5-35

Index-organized Tables

Oracle stores an index-organized table entirely as an index based on the primary key. Index-organized tables give faster key-based access. Storage is also reduced compared with an ordinary table. There are some restrictions.

Function-based Indexes

Function-based indexes can improve performance by enabling access through indexes for statements having functions in their `WHERE` clauses.

Compute Statistics

You can specify that you want Oracle to collect statistics during the creation of an index.

Domain Key Constraints

You can now create domain key constraints to control allowable values for columns. You can use the table API to generate domain key constraint implementation mechanisms. For the run-set you must select all the tables involved.

Table API

Table API triggers are optional.

Materialized Views

Server Generator generates a materialized view for Oracle8*i* and a snapshot for earlier releases. In Oracle8*i* the terms materialized view and snapshot are synonymous.

Design Capture Preferences

There are new preferences for controlling the behavior of Design Capture.

View Expansions

Design Capture can now create declarative definitions for captured views.

Practice 5 – 1: Defining Oracle8*i* Objects in the Repository

Note: Ensure you switch to the Design Editor for this practice

Goal

The purpose of this practice is for you to define and generate some of the objects that you can create on the server-side with an Oracle8*i* database.

Scenario

The developers have planned to migrate to an Oracle8*i* version of the database. They have already revised the server definitions and have incorporated the changes to use the new features of the kernel.

They have decided on several additional actions:

- To improve performance by changing one of the tables to an index-organized table, add a function-based index to allow more flexible requests on part of customers' last names, and implement statistics on the index
- To make use of domain key constraints
- To create a materialized view

Your Assignment

Use the information below as your starting point:

Tool	Design Editor
Workarea	ORA<nn>_PWA_01
Application System	NF_HOLLYWOOD

where <nn> is the number assigned to you by your instructor.

Improving database performance

1 Change the properties of the MEMBERSHIP_PERIODS table definition so that it becomes an index-organized table.

2 Generate the index-organized table MEMBERSHIP_PERIODS to file and examine the DDL.

   a What is different about the DDL compared with the DDL for an ordinary table, that is, a non-index-organized table?

3 On the CUSTOMERS table definition, create a function-based index definition named I_SHORT_LASTNAME using the function upper(substr (...)) on the first two characters of the LAST_NAME column. Change the property of the function-based index in order to create statistics at generation time.

4 Generate the DDL for the CUSTOMERS table definition to file. Open the file which contains the function-based index I_SHORT_LASTNAME of the CUSTOMERS table definition and investigate the code.

Creating a domain key constraint

5 In the REVIEWS table, identify how the allowable values for the column HOT are implemented.

6 Add a domain key constraint to the column MONOCHROME in the TITLES table using the same approach that was used for the column HOT in REVIEWS.

7 Identify in the CUSTOMERS table definition which column could use a domain key constraint and in which table it should be implemented.

8 Generate to file the table in which you want to store the domain key constraint values. Connect to the database with SQL*PLUS (the domain key constraint table has already been created for you in the database). Investigate the values in the domain key constraint table.

9 Generate the table API for the domain constraint table and for the related tables.

10 Why do you need to generate the related tables within the same set?

Creating a materialized view

11 Using the Free Format option create a materialized view (M_HIT_TITLES) for Data Warehouse summary usage based on the following statement:

```
SELECT titles.title, titles.movie_category,
titles.duration,
count(*) nb_of_rents
FROM titles, copies, rental_items
WHERE titles.product_code=copies.ti_product_code
and copies.cop_id = rental_items.cop_id
GROUP BY titles.title, titles.movie_category,
titles.duration
```

Note: The file MAT_VIEW.txt exists in the repository in the folder ORA<nn>_FOLDER_05. It can also be found in <working directory>/LES05.

12 Generate the DDL for the materialized view M_HIT_TITLES that you have defined, to use in an Oracle8*i* database.

13 Generate the DDL for M_HIT_TITLES again, this time for use in an Oracle 7.3 database. Examine the DDL and compare it with the DDL for Oracle8*i*. What is the difference between the two and why are the two different?

Practice 5 – 2: Controlling the Design Capture Process

Goal

The purpose of this practice is for you to control the design capture process by using preferences while capturing the design of server model objects.

Scenario

Somebody has changed the database directly but not maintained the design in the repository. This means updating the repository to bring it in line with the database using the Server Generator preferences to capture the design. You want to familiarize yourself with the options.

You take the following actions to:

- Capture a table using default preferences
- Capture the same table with one of the preferences changed
- Capture a view and look at the results of the new view expansion

Your Assignment

Use the information below as your starting point:

Tool	Design Editor
WorkArea	ORA<nn>_PWA_01
Application System	NF_HOLLYWOOD

where <nn> is the number assigned to you by your instructor.

Capturing a table using the default design capture preferences

1 Compare the existing BOOKINGS table with the definition of the BOOKINGS table definition in the repository and identify the differences. You can use the Database Navigator to do this.

2 Capture the design of the BOOKINGS table and examine the table definition. Are the repository and the database now the same?

Capturing a table, having modified preferences

3 Add a column to the TITLES table in the repository.

4 Capture the TITLES table, setting one of the Server Generator preferences so that the design capture will drop the column definition in the repository in order to match the repository definition with the database table.

5 Run Design Capture and check that the column is dropped.

6 Attempt to capture the BOOKINGS table, setting one of the Server Generator preferences so that the design capture will drop the column definition in the

repository in order to match the repository definition with the database table. What issues do you run into?

Capturing a view and investigating the results

7 Capture the view VMOVIES and look at the results of the view expansion by looking at the element definition.

8 Why is the Free Format Select Text property set to No?

9 When would you expect this property to be set to True?

Practice 5 – 3: Navigating in the Design Editor (Optional)

Goal

The purpose of this practice is to navigate within workareas and application systems in the Design Editor.

Scenario

You want to review new functionality within the Design Editor. To do so you:

- Investigate objects in the context of an application system
- Investigate the Advanced tab, in the Edit property dialog

Your Assignment

Use the information below as your starting point:

Tool	Design Editor
WorkArea	ORA<nn>_PWA_01
Application System	NF_HOLLYWOOD

where <nn> is the number assigned to you by your instructor.

Investigating objects in the context of an application system in the Design Editor

1. In the Server Model Guide, set the context container to HOLLYWOOD.
2. What domains are contained in this application system?
3. What domain is assigned to the EMPLOYEE.EMAIL column?

Investigating the Advanced tab, in edit property dialog

4. Invoke the dialog box for the EMPLOYEES table definition, and view the Advanced tab in the dialog.
   a. Update the description and user help text for the table.
   b. What are the allowable values for the Journal table property?
5. In the Columns tab, access the advanced properties for the POSITION column.
   a. What are the values for the Denormalization - Using Operator *column* property?
   b. What are the allowable values for a column’s Derivation Expression Types?

Practice 5 – Hints

To Do This Task	Follow These Steps
Changing the properties of a table definition	**1** Select the table definition and double click to invoke the property palette or property dialog. **2** Complete all properties as required in the practice. **3** If you invoke the property dialog and do not find the property you want, look under the advanced tab or switch to the property palette.
Creating an index-organized table	**1** Invoke the property palette for the table definition. **2** Set the index-organized? property to Yes in the storage category.
Generating a table to file	**1** Select the table. **2** Select the menu option **Generate – > Generate Database from Server Model** **3** Select Target for Generation in order to generate to file, select DDL Files only. **4** Verify you have the required objects selected under the Objects tab.
Creating a function-based index definition	**1** Select the table and expand the node to expose the indexes node **2** Use the property dialog and follow the process for creating an index. **3** At the Create Index dialog, instead of selecting columns you want to include the index, click the **Add Function** button and enter the function you require. If you select a single table to generate, you may get warnings about additional keys that are not processed, because the associated tables were not included in the run set.
Examining the generated DDL	**1** Click the **List Actions** button in the Message Window to invoke the list of files that are created during generation. **2** Double-click the file to read the contents.
Identifying how the domain for a column is implemented	**1** Invoke the property palette for the column you want to investigate. **2** Is the Domain Property set? If it is, then the allowable values for the column are implemented through a traditional domain. **3** If the Domain Property is not set, expand the Domain Key Constraint node. Are there any Domain Key Constraints implemented? If there are, does the definition include the column you are investigating?

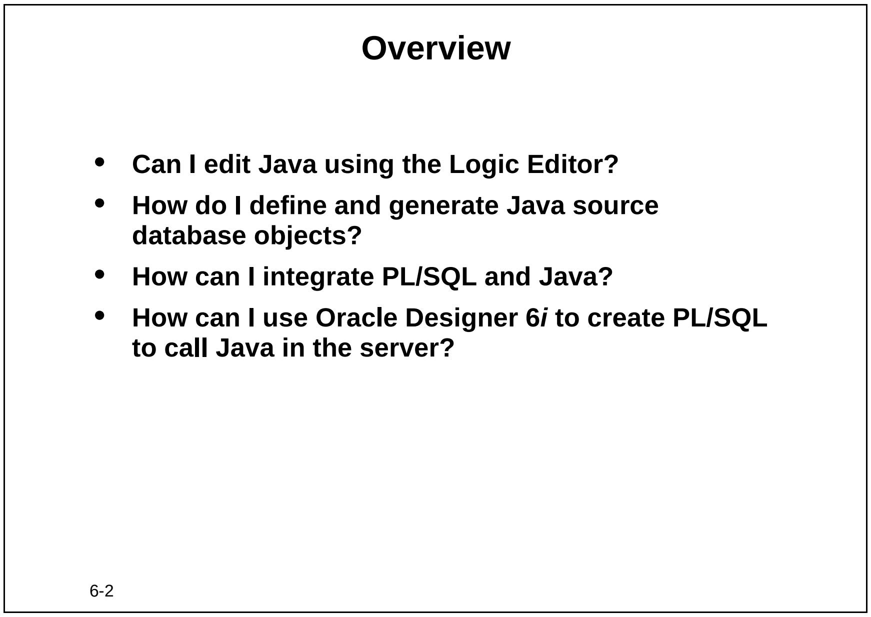
To Do This Task	**Follow These Steps**
Generating the table API	**1** Select the table definition. **2** Select the menu option **Generate – > Generate Table API**.
Using the Free Format option create a materialized view	Using the Property Palette, set the SQL property for Free Format Select Text to Yes and complete the materialized view definition.
Comparing the definitions of objects defined in a database with those defined in the repository	**1** Invoke SQL*PLUS and describe the object or use the Database Navigator.
Doing Design Capture	**1** Using the Menu option, **Generate – > Capture Design Of – > Server Model** to invoke the dialog. **2** Complete the information about the database user that owns the objects you want to capture. **3** Under the Options tab, select the objects for design capture.

6

Java and the Server Generator

Introduction

In this lesson you look at how you can model Java objects using Oracle Designer 6*i*. The lesson covers objects that the Server Generator generates for the purpose of integrating Java and PL/SQL, and objects that allow you to call generated Java objects.



Overview

- **Can I edit Java using the Logic Editor?**
- **How do I define and generate Java source database objects?**
- **How can I integrate PL/SQL and Java?**
- **How can I use Oracle Designer 6*i* to create PL/SQL to call Java in the server?**

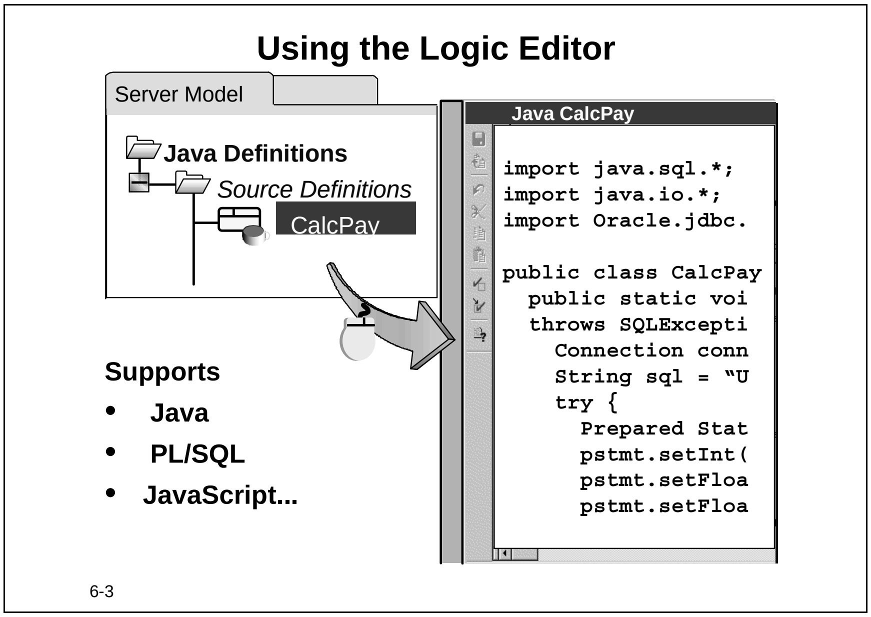
6-2

Objectives

At the end of this lesson you should be able to:

- Define and generate Java database objects
- Implement server-side procedures with Java
- Create server-side PL/SQL to integrate with Java in Oracle8*i*
- Generate call specs for Java stored procedures
- Describe how you can access SQL data from Java stored procedures
- Describe PL/SQL Java integration in Oracle8*i*

Using the Logic Editor



Using the Logic Editor

```
import java.sql.*;
import java.io.*;
import Oracle.jdbc.

public class CalcPay
  public static voi
  throws SQLExcepti
    Connection conn
    String sql = "U
    try {
      Prepared Stat
      pstmt.setInt(
      pstmt.setFloa
      pstmt.setFloa
```

Supports

- **Java**
- **PL/SQL**
- **JavaScript...**

6-3

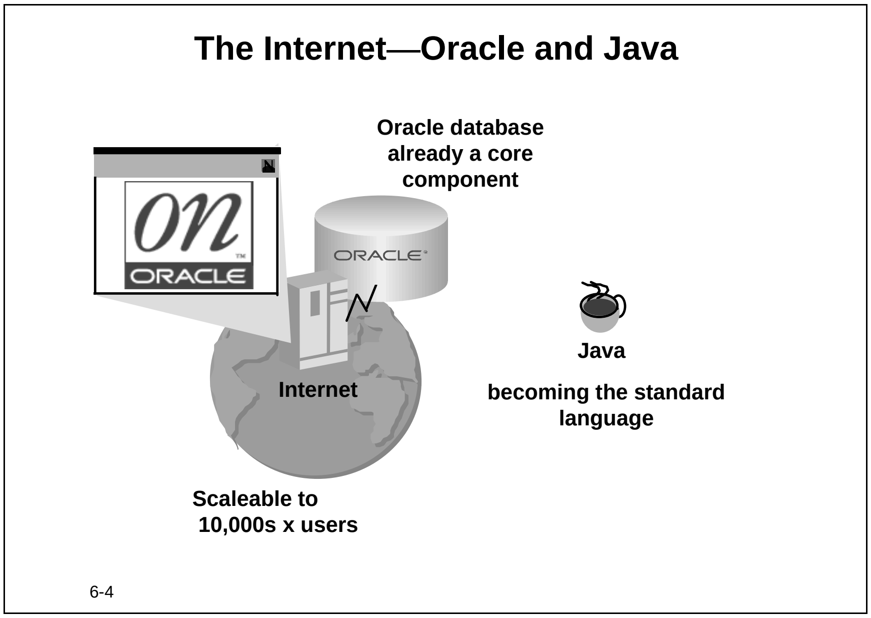
The source code for the Logic Editor has been replaced and the new code editor handles Java source code in addition to PL/SQL and JavaScript. The editor highlights Java key words and constructs in color.

Text Editing Features of the Logic Editor

Feature	**Description**
Editing using drag and drop	The process of defining logic is simplified to include drag-and-drop predefined constructs from the Construct Tree. This allows you to build code without the need for full knowledge of the syntax. In addition, you can copy design level data (for example, tables and views) into the code to supply data values in the program code.
Checking Syntax	A Check Syntax utility is available for checking PL/SQL code and Java stored in the repository.
Formatting Text	Features such as automatic indentation and color syntax highlighting of keywords, comments, and strings are applied to code typed directly into the Logic Editor Code Window.
Creating keyboard maps	You can customize keyboard mappings and add or remove predefined ones.

Feature	Description
Building keystroke macros	You can record a series of keystrokes and assign a keystroke to play back the keystrokes repeatedly. You can record up to ten macros.
Exporting and importing code	This enables you to reuse or modify code which was defined using other applications. You can import code stored in text files into the Logic Editor. You can then either store it in the repository, or, having edited the code using the text editing facilities, write the code back to a text file.

The Internet—Oracle and Java



The Internet—Oracle and Java

Oracle database already a core component

Internet

Scaleable to 10,000s x users

Java

becoming the standard language

6-4

Oracle database

The Oracle database is already a core component of the internet. The Oracle database provides the scalability and performance to service up to tens of thousands of users. Most of the world’s biggest web sites run on Oracle.

Java

Java is becoming the standard language of the internet. One of the reasons for this is that Java compiles to a standard, platform-independent set of bytecodes that can run on any machine. All the machine requires is a software component known as the Java virtual machine (JVM). Since the JVM is built into web browsers any machine running a web browser can use an internet site written in Java. This concept is known as “write once, run anywhere”.

Java is also truly object-oriented, efficient for application-level programs, and is easily learned by the huge body of existing C programmers.

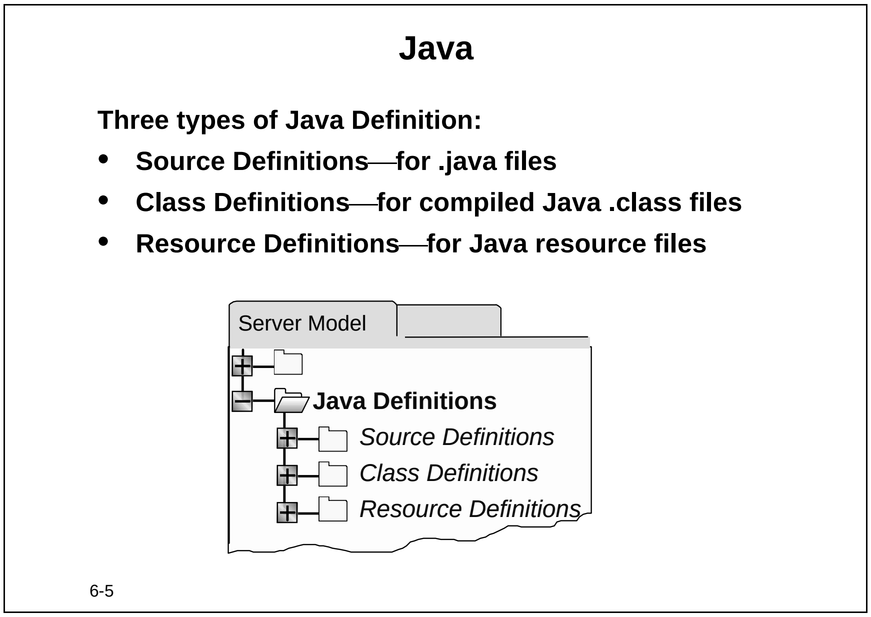
Integrating the JVM into the database provides Oracle8*i* levels of scalability and performance for Java. Along with the provision of support tools it is part of Oracle’s strategy to provide the total end-to-end Java solution for Internet Computing.

Internet Computing

Oracle believes that the convergence of the Oracle8*i* database with Java and the internet represents the maturation of the computer industry in a new wave of computing— Internet Computing. The internet changes priorities in the computer industry. Through enabling e-business it changes the nature of business itself.

Java Elements in Oracle Designer 6*i*

You can store Java source, Java classes and Java resources in an Oracle8*i* database as Java schema objects. Use Oracle Designer 6*i* to model these server-side Java objects and to generate `CREATE JAVA` statements to load the Java objects into the Oracle8*i* database.



Java

Three types of Java Definition:

- **Source Definitions—for .java files**
- **Class Definitions—for compiled Java .class files**
- **Resource Definitions—for Java resource files**

Server Model
Java Definitions
Source Definitions
Class Definitions
Resource Definitions

6-5

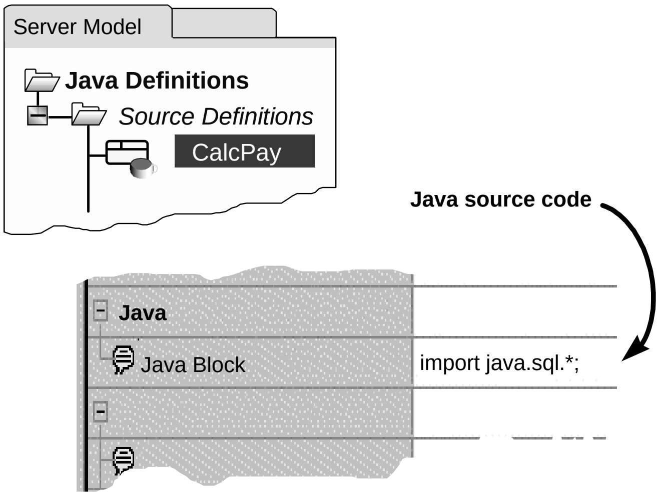
Java Definitions

Types	**Description**
Java Source	Java source definitions contain Java source code. The Java source code is the readable form of Java code stored in the database. The Oracle 8*i* server contains a compiler that compiles the source code to is compiled to class format when you invoke the Java object.
Java Class	Java class definitions record the use of a compiled Java class, in the standard binary class format. This is the compiled version of the source.
Java Resource	Java resource definitions contain Java data file content that is used or loaded at runtime.

Creating Java Definitions

You can create Java definitions in the repository in the same way as you create a PL/SQL definition. In addition, you need to specify how the Java object is going to derive the Java code. The Java code may be located in a file in an external file system, or within a LOB column on the database. Alternatively, Java source code can be stored within the definition itself. This last option is only available for Java source definitions.

Creating Java Source Definitions



6-6

How to Create a Java Source Definition using the Java Block property

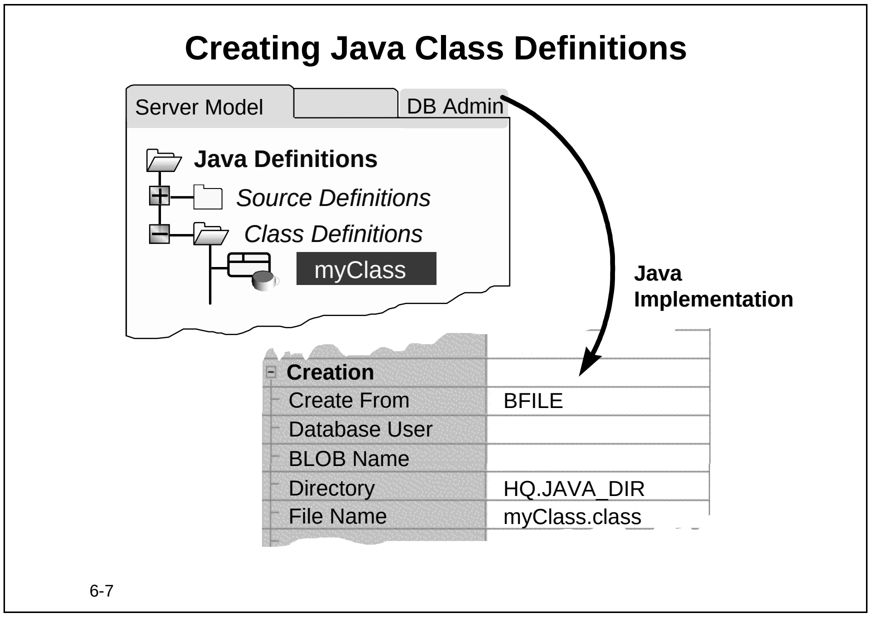
1. In the Design Editor, expand the Java Definitions node.
2. Select the Source Definitions node and click the Create button.
3. Specify the name and short name to create the Java definition.
4. Populate the Java Block property with the Java source code.

Technical Note:

Your source code may exist within an external file, or a LOB column on the database. To create a Java source definition based on existing code, leave the Java Block property empty. Instead, proceed as for Java Class or Resource by specifying the source code location using Java source implementation properties.

Creating Java Class and Resource Definitions

To define a compiled Java class, in the standard binary class format, or a Java resource, in standard portable format, create either a Java class definition or a Java resource definition.



Creating Java Class Definitions

Creation	
Create From	BFILE
Database User	
BLOB Name	
Directory	HQ.JAVA_DIR
File Name	myClass.class

6-7

How to Create a Java Class or Resource Definition

1. In the Design Editor, expand the Java Definitions node.
2. Select the appropriate node and click the Create button.
3. Specify the name and short name to create the Java definition.
4. Specify the location of the Java code.

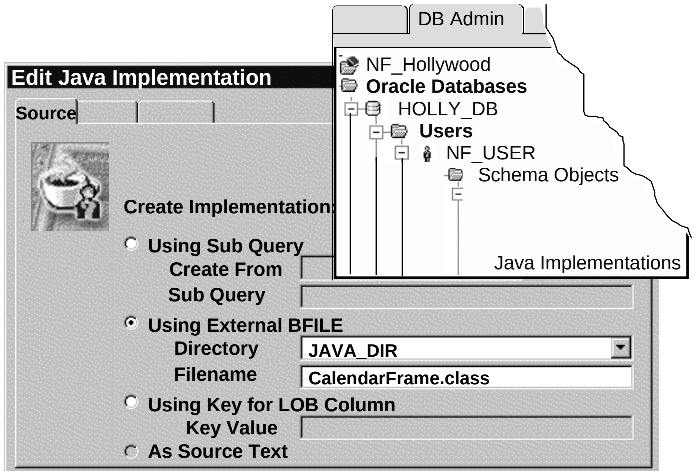
   You need to use the DB Admin tab to specify the location of the Java definition.

   You cannot store the code in the definition as you can with a Java source definition, instead, store the Java class or resource code in an external file, or within a LOB column on the database.

   In order to complete the definition, specify the source code location of your Java class or Java resource by creating an implementation of the Java object.

Defining Implementation Details

Creating Java Implementations



Create Implementation:

○ Using Sub Query

Create From

Sub Query

● Using External BFILE

Directory: JAVA_DIR

Filename: CalendarFrame.class

○ Using Key for LOB Column

Key Value

○ As Source Text

6-8

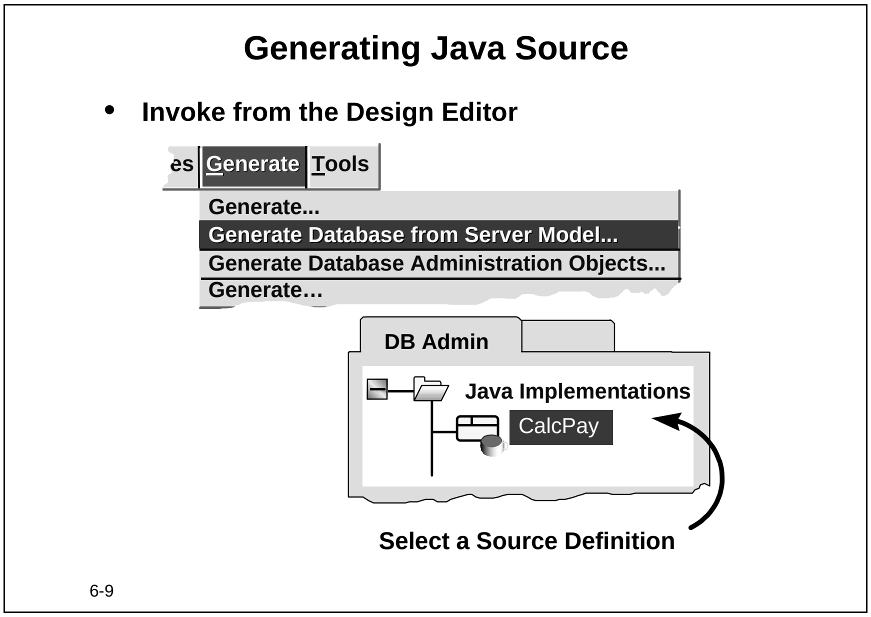
How to Define the Implementation Details

Use the **DB Admin** tab to define implementation details.

1. Expanding the Oracle Databases and Users nodes to display Schema Objects.
2. Select Java Implementations and click the Create button.
3. Choose the name of your Java definition.
4. If you have defined the source code in the Java definition then the definition is complete, if not, then you have a number of options:

Using Property	Java Object Stored in...	Value
Sub Query	CLOB, BLOB, or BFILE column	Enter a sub query to return the source code
External BFILE	An external BFILE	Specify the Directory and the name of the file. Note: Define the Directory under the Storage node in the repository.
Key for LOB Column	A column named LOB in the table CREATE$JAVA$LOB$TABLE	Enter the value in the LOB column to retrieve the source from the Name column in this table

Generating Java Definitions



Generating Java Source

- **Invoke from the Design Editor**

es
Generate
Tools
Generate...
Generate Database from Server Model...
Generate Database Administration Objects...
Generate…
DB Admin
Java Implementations
CalcPay

Select a Source Definition

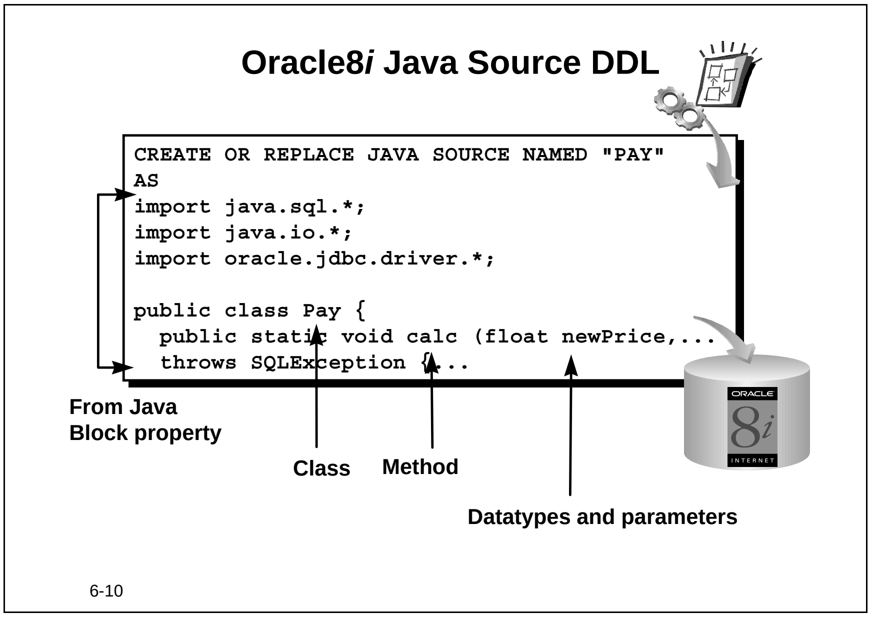
6-9

Generating Java Source Definitions

To generate the Java database objects from the Design Editor, choose **Generate –> Generate Database from Server Model.** The Server Generator produces very different DDL for Java definitions depending of the nature and the location of the Java object.

Generated DDL for Java Source

The Server Generator produces DDL that creates a Java source database object for Oracle8*i*. The generator obtains all the Java code from the Java block property of the Java source definition.



Oracle8*i* Java Source DDL

```
CREATE OR REPLACE JAVA SOURCE NAMED "PAY"
AS
import java.sql.*;
import java.io.*;
import oracle.jdbc.driver.*;

public class Pay {
  public static void calc (float newPrice,...
  throws SQLException {...
```

Oracle8*i* Java Source DDL

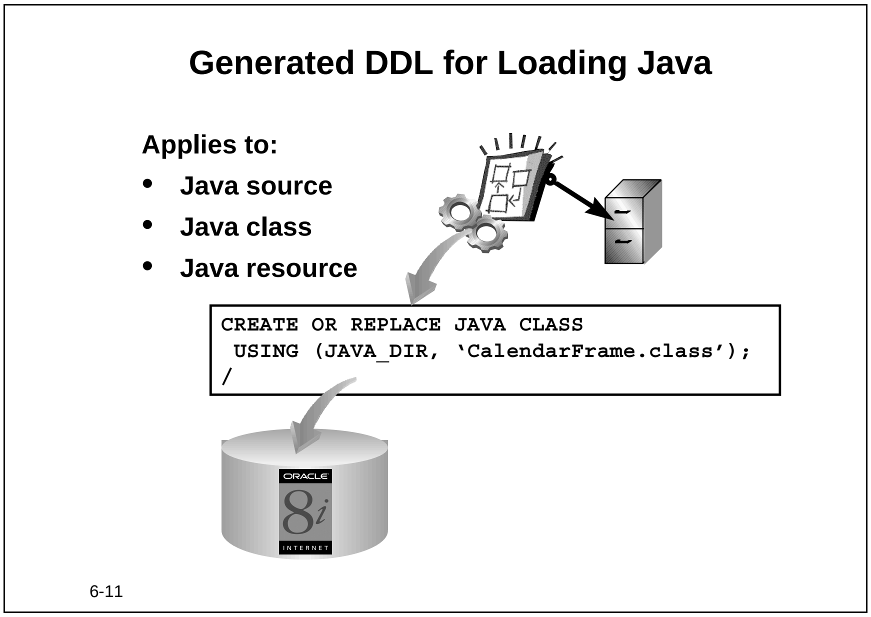
After the `AS` keyword the DDL for the Java source database object consists of pure Java code. The table below has an excerpt of generated Java source code.

```
CREATE OR REPLACE JAVA SOURCE NAMED "CalcPurch"
AS
   import java.sql.*;
   import java.io.*;
   import oracle.jdbc.driver.*;
public class CalcPurch { ...
```

If you set the Resolve property to Yes, when you execute the DDL that creates a Java source database object, Oracle8*i* attempts to resolve (or compile) the Java object if the CREATE JAVA command succeeds.

Generated Output

The Server Generator produces DDL that creates a Java source, Java class or Java resource database objects for Oracle8*i*.



Generated DDL for Loading Java

Applies to:

- **Java source**
- **Java class**
- **Java resource**

```
CREATE OR REPLACE JAVA CLASS
 USING (JAVA_DIR, ‘CalendarFrame.class’);
/
```

ORACLE
8i
INTERNET

6-11

Generated DDL for Loading Java

If you do not define the source text in the Java block property, you can generate Java source in the same way as you would for Java class or Java resource definitions, by using Sub query, external BFILE or LOB column.

The Server Generator produces different DDL for the Java source, Java class and Java resource objects depending where the objects are stored. The CREATE statement contains the directory and filename of the external file, or the LOB column in the database, that you define in the implementation.

Example

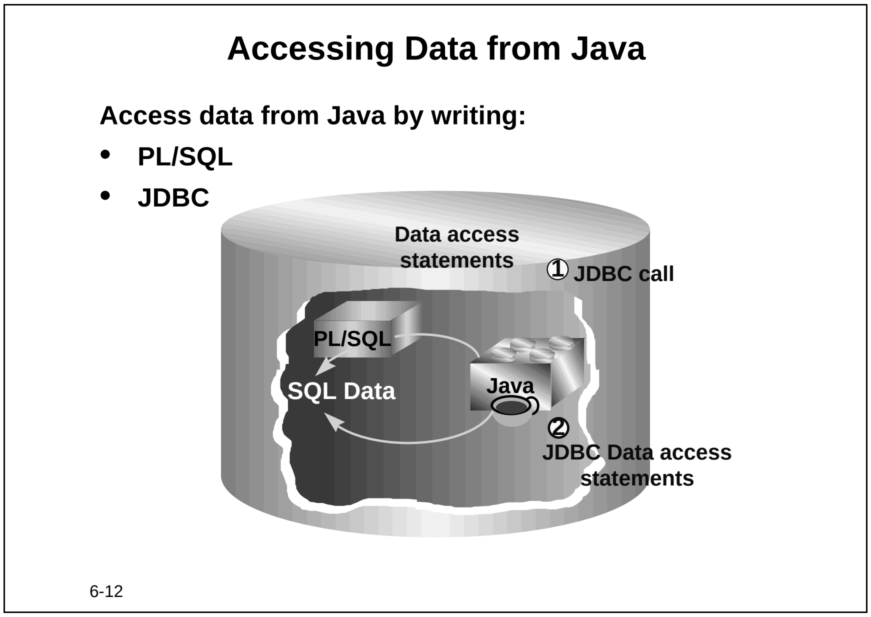
```
CREATE OR REPLACE JAVA CLASS
 USING BFILE (JAVA_DIR, ’CalendarFrame.class’);
```

In this example, you need to define the Directory JAVA_DIR in the repository and the CalendarFrame.class exists in the directory in the file system.

When you execute this `CREATE` statement in an Oracle8*i* database, Oracle loads the Java into the server where it can be executed by the Oracle Aurora Java virtual machine (JVM).

Accessing Data from Java

There are two methods available for accessing data from Java. You can either call PL/SQL database objects that contain your data access statements or you can write data access statements using the JDBC API.



Accessing Data from Java

Access data from Java by writing:

- **PL/SQL**
- **JDBC**

6-12

PL/SQL Access

Oracle8*i* provides you with the interface between PL/SQL and Java. PL/SQL provides the easiest and most efficient way to write data access statements. You can call PL/SQL objects from Java source database objects by writing simple JDBC call statements. This is particularly useful if you are familiar with PL/SQL. Java programmers who want access to Oracle data generally favor writing data access statements in JDBC.

JDBC

JDBC (Java Database Connectivity) is a standard Java interface for connecting to relational databases from Java. The JDBC standard was defined by Sun Microsystems. JDBC is based on the X/Open SQL Call Level Interface, and complies with SQL92.

JDBC Access

JDBC is flexible and powerful. Its syntax is correspondingly comprehensive. To use JDBC you write JDBC API calls in the Java language.

Simplifying JDBC Access with SQLJ

Depending on the nature of your programs you may be able to use the Oracle SQLJ layer to simplify JDBC data access. SQLJ embeds static SQL in Java programs and has a shorter, SQL-like syntax. For many applications you can perform data access more conveniently with SQLJ than with direct JDBC.

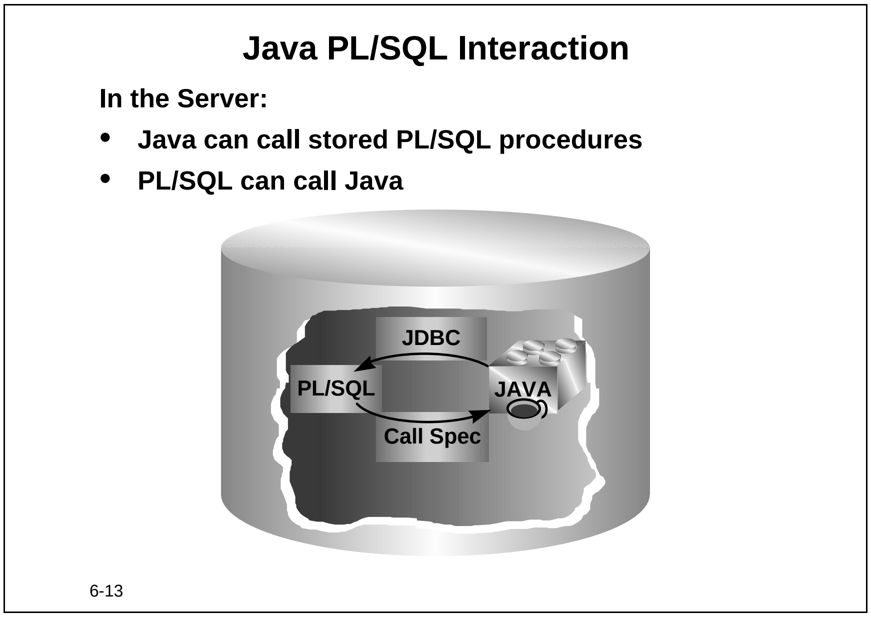
Java and PL/SQL

There are a number of issues to consider when deciding whether to use PL/SQL or Java. Because of the tight integration between PL/SQL and SQL in Oracle8*i*, PL/SQL provides more efficient access to SQL data than does Java, where there is an overhead in translating between the languages. PL/SQL is also familiar to a large group of existing developers. Java on the other hand offers faster raw processing speed for computations. The Java based data access method is only likely to be the preferred choice of Java programmers.

Any procedure written in PL/SQL can be written in server-side Java, so you can use either language. The choice is most likely to be based on the preference of the programmer or on an organization’s standards. However, in terms of efficiency PL/SQL is faster for DML intensive work, Java is faster for computation work.

Java PL/SQL Integration

Oracle8*i* allows a high degree of interoperability between Java and PL/SQL. You can call Java database objects from server-side PL/SQL and you can call PL/SQL database objects from server-side Java. This integration is handled for you by Oracle8*i*.



Java PL/SQL Interaction

In the Server:

- **Java can call stored PL/SQL procedures**
- **PL/SQL can call Java**

6-13

Oracle8*i* allows you to write and execute programs such as stored procedures and triggers in Java, to invoke existing PL/SQL programs from Java and to invoke Java programs from PL/SQL. Java applications can call PL/SQL stored procedures using an embedded JDBC driver. PL/SQL applications can call Java stored procedures directly

Java Stored Procedures

Java stored procedures are Java methods published to SQL and stored in an Oracle8*i* database for general use. To publish Java methods, you write call specifications, generally referred to as call specs.

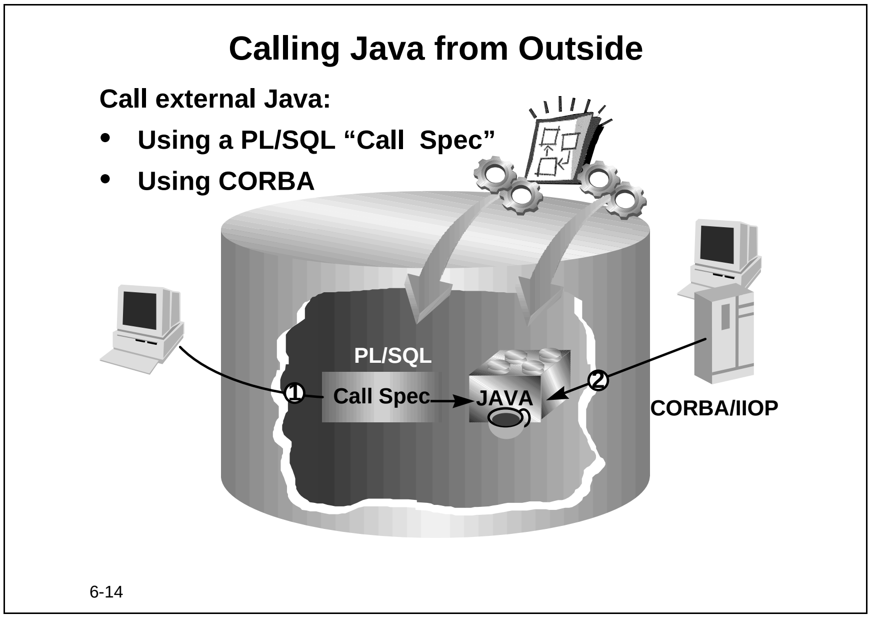
Call Specs

A call spec maps a Java method name, parameter types, and return types to their SQL counterparts. Unlike a wrapper, which adds another layer of execution, a call spec simply publishes the existence of a Java method. So, when you call the method through its call spec, the run-time system dispatches the call with minimal overhead.

Oracle8*i* supports a special type of PL/SQL procedure or function for this purpose. This special PL/SQL object uses a Java method for its main logic. In Oracle Designer 6*i* you can set up PL/SQL definitions that the Server Generator generates as call specs.

Calling Java from Outside

You can call Java database objects from outside the server using one of two different methods.



Calling Java from Outside

Call external Java:

- **Using a PL/SQL “Call Spec”**
- **Using CORBA**

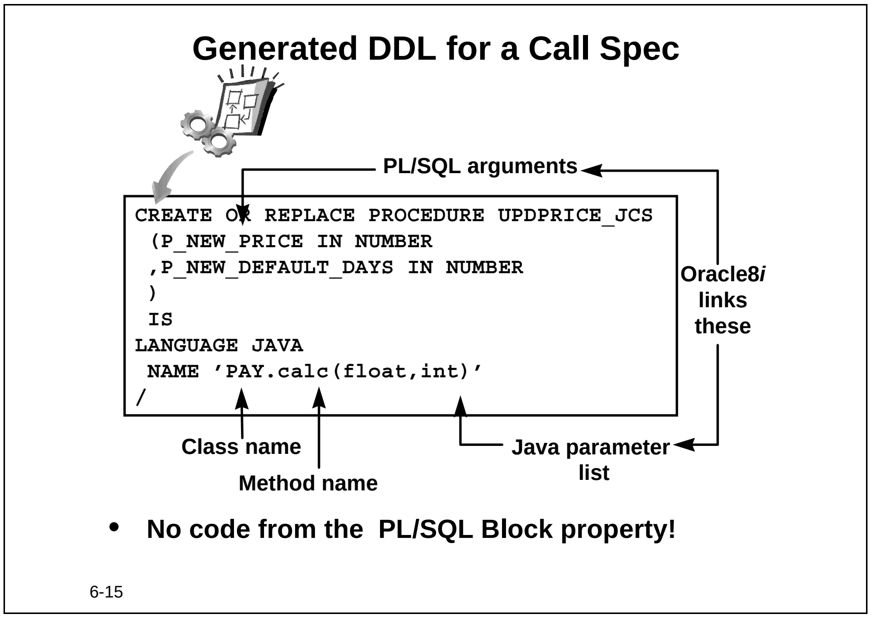
6-14

PL/SQL

The easiest method for calling Java from outside the server is by calling server-side PL/SQL and making use of the integration between Java and PL/SQL that Oracle8*i* provides, that is, by using a call spec.

Generated DDL for a Call Spec

Oracle Designer 6*i* generates call spec DDL for you. The DDL comes from the PL/SQL definition that you define.



Generated DDL for a Call Spec

```
CREATE OR REPLACE PROCEDURE UPDPRICE_JCS
 (P_NEW_PRICE IN NUMBER
 ,P_NEW_DEFAULT_DAYS IN NUMBER
 )
 IS
LANGUAGE JAVA
 NAME 'PAY.calc(float,int)'
/
```

- **No code from the PL/SQL Block property!**

6-15

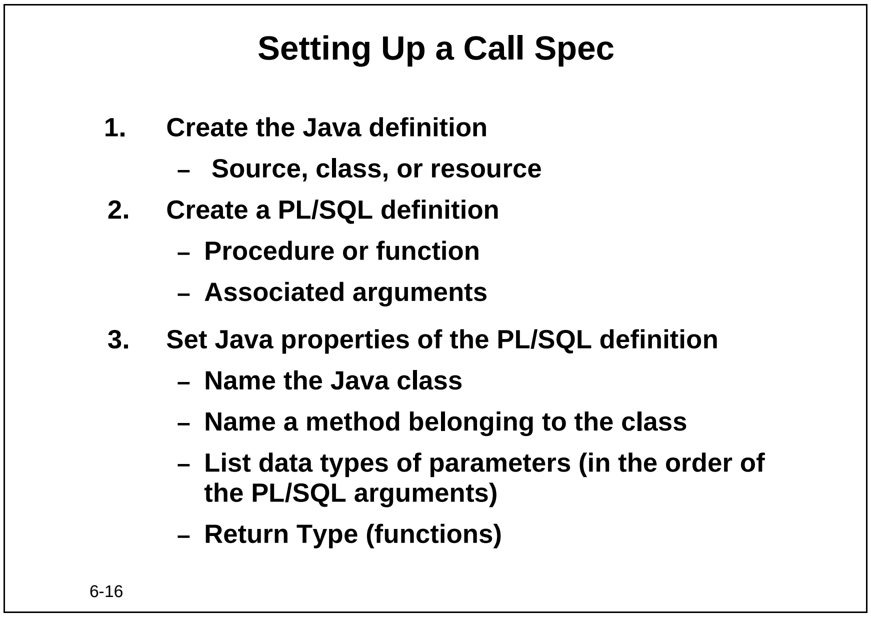
When you generate your call spec, the Server Generator produces a `CREATE PROCEDURE` statement for a PL/SQL database object. However, the generated code is different to a normal PL/SQL object. The `CREATE` statement includes the clause `IS LANGUAGE JAVA` and names:

- The Java class database object
- The method
- Any Java datatypes for the parameters you specified in the Java Parameter List property.

There are separate properties for each of these in the PL/SQL definition element type.

Setting up a Call Spec

You can generate call specs from Oracle Designer 6*i*. There is no repository element of type “call spec”. Instead, you define a call spec by a PL/SQL definition that has certain Java properties set.



Setting Up a Call Spec

1. **Create the Java definition**
   - **Source, class, or resource**
2. **Create a PL/SQL definition**
   - **Procedure or function**
   - **Associated arguments**
3. **Set Java properties of the PL/SQL definition**
   - **Name the Java class**
   - **Name a method belonging to the class**
   - **List data types of parameters (in the order of the PL/SQL arguments)**
   - **Return Type (functions)**

6-16

The Java Method

A call spec publishes a Java method to PL/SQL. Before defining a call spec you must first create a Java definition for the Java class that contains the method. The Java definition can be any one of Java source, Java class, or Java resource.

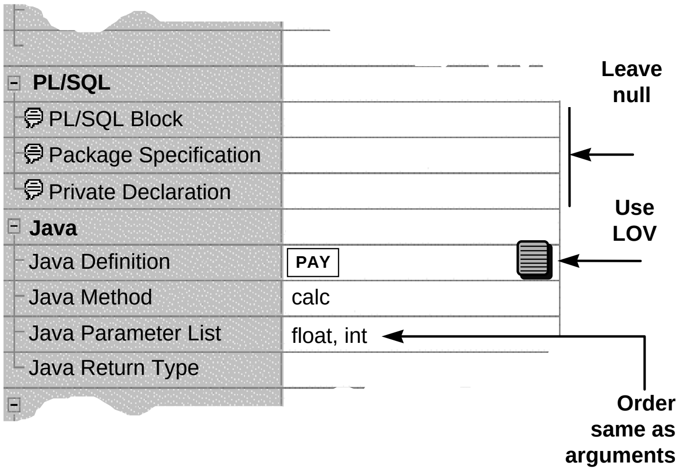
How to Set up a Call Spec

1. Define a PL/SQL element such as a procedure.
2. Instead of populating the PL/SQL Block property of the element with PL/SQL code as you would for a normal PL/SQL element, populate the Java properties.
3. Add any PL/SQL arguments that map to Java parameters to the definition.

When you have populated the Java properties of a PL/SQL element, Oracle Designer 6*i* recognizes the element as a call spec and generates database objects that are very different from normal PL/SQL objects.

Additional PL/SQL Definition Properties

PL/SQL Definition Properties for Call Specs



⊟ **PL/SQL**	
PL/SQL Block	
Package Specification	
Private Declaration	
⊟ **Java**	
Java Definition	PAY
Java Method	calc
Java Parameter List	float, int
Java Return Type	

Leave null

Use LOV

Order same as arguments

6-17

Java Class Property

To populate the Java definition property, select the appropriate Java element. You can select it from a drop-down list. The drop-down list will show all the Java definitions in the repository, that is, all Java source, all Java class and all Java resource definitions.

Java Methods

Java language programs always consist of one or more methods internally. With Java you always invoke a method, referring to the class and method using a “class dot method” notation, for example, CalcPay.Add. A call spec corresponds to a single method. You must name the appropriate method that you wish the call spec to invoke when executed.

Java Method Property

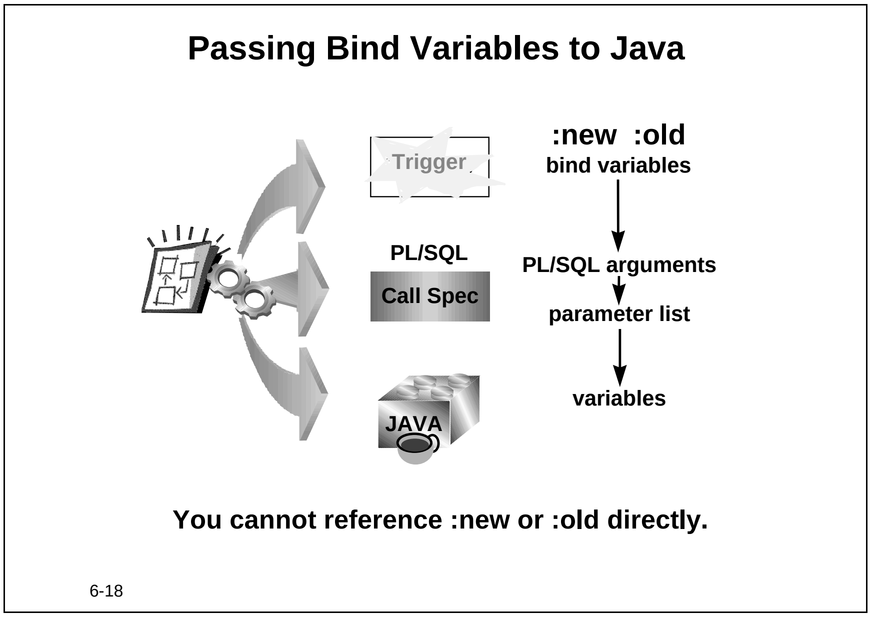
Name the appropriate method by populating the Java Method property with the method name. You must enter the name of the method yourself. There is no drop-down list available for this property because the methods are defined within the Java source code itself and are not separate elements in the repository.

Java Parameter List Property

Populate the Java Parameter List property with a list of Java data types for the PL/SQL arguments, if there are any. You must list the parameter data types in the same order as the corresponding PL/SQL arguments. Once you create your objects in the database, Oracle8*i* matches the arguments to the parameters based on their order.

Passing Bind Variables to Java

You can pass DML transaction bind variables of PL/SQL to Java source objects.



Passing Bind Variables to Java

Trigger

PL/SQL

Call Spec

JAVA

:new :old

bind variables

PL/SQL arguments

parameter list

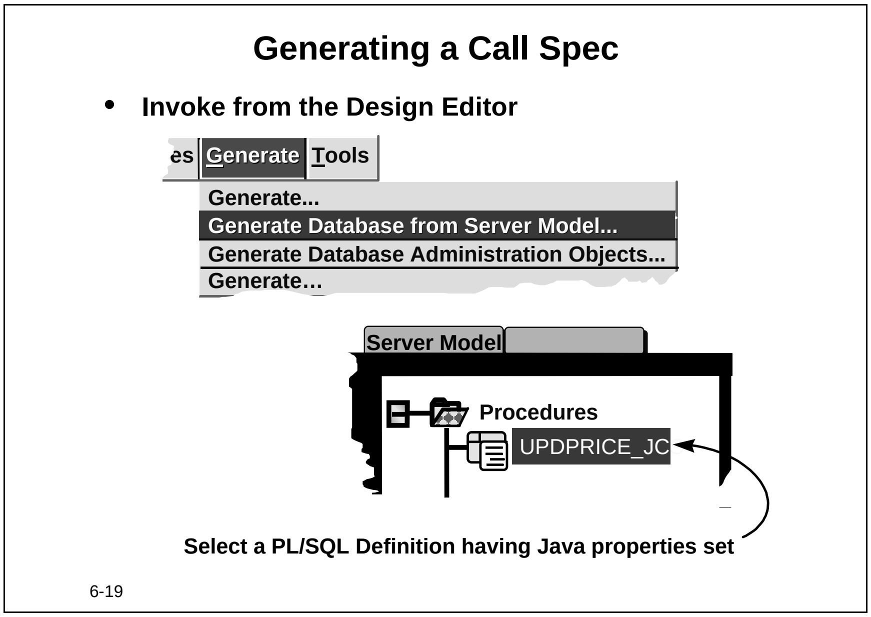
variables

You cannot reference :new or :old directly.

6-18

In PL/SQL, unless you specify alternative names these bind variables use **:new** and **:old** prefixes.

Generating a Call Spec



Generating a Call Spec

- **Invoke from the Design Editor**

es
Generate
Tools
Generate...
Generate Database from Server Model...
Generate Database Administration Objects...
Generate…
Server Model
Procedures
UPDPRICE_JC

Select a PL/SQL Definition having Java properties set

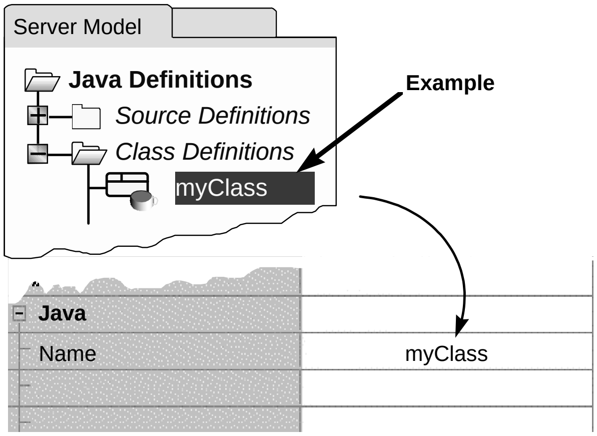
6-19

How to Generate a Call Spec

In the Design Editor, choose **Generate –> Generate Database from Server Model.** Make sure that you select the correct PL/SQL definition and that the definition has the Java properties set appropriately.

Capturing Java

Capturing External Java



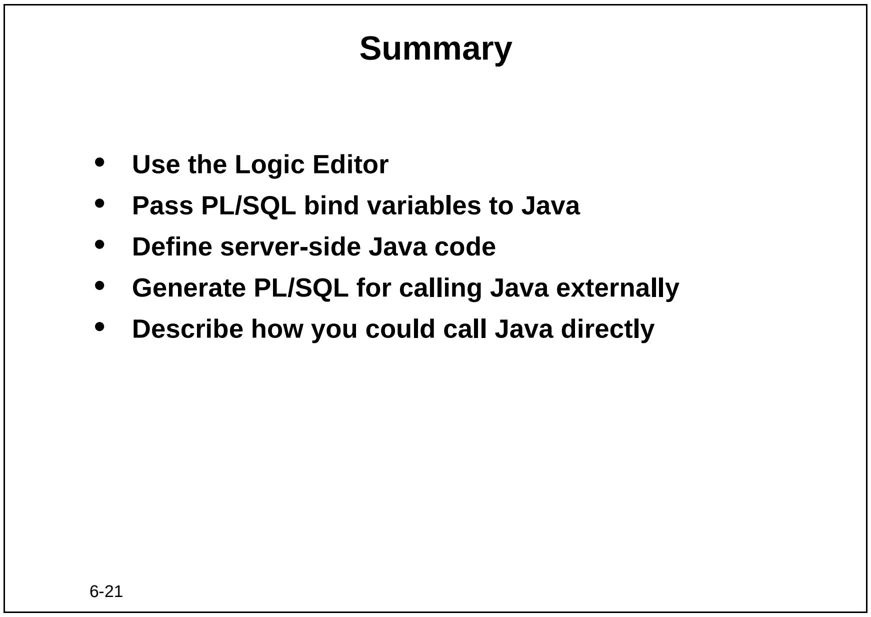
Java

Name myClass

6-20

The only information that you can capture from Java objects is the name. The database stores only a Short Name version (30 characters) of the Java object. For access to the Long Name, use the package DBMS_JAVA.longname

Summary



Summary

- Use the Logic Editor
- Pass PL/SQL bind variables to Java
- Define server-side Java code
- Generate PL/SQL for calling Java externally
- Describe how you could call Java directly

6-21

Java

You can now generate Java database objects. The logic editor supports Java source code and highlights Java key words and constructs.

Java Objects

You can generate all the new Java database objects of Oracle 8*i* using Oracle Designer 6*i* —Java source, Java class and Java resource. Oracle8*i* integrates PL/SQL and Java.

You can call server-side Java from outside the database using a new type of PL/SQL object referred to as a call spec. Oracle8*i* automatically links PL/SQL arguments with Java parameters. You can pass PL/SQL bind variables to Java as parameters.

PL/SQL or Java

PL/SQL is faster for DML intensive work, Java is faster for computation work.

Logic Editor

There is a new code editor that handles Java source code in addition to PL/SQL.

Call Spec

You use a call spec to publish a Java stored procedure to PL/SQL.

Practice 6 – 1: Implementing Oracle8*i* Objects in the Repository

Goal

The purpose of this practice is for you to implement server-side objects for an Oracle8*i* database.

Scenario

The developers have made changes to the server definitions for the Oracle8*i* database. They have defined new objects which now need implementing. One of the developers on the project has written Java code in the repository to implement a database trigger.

They have decided on the following actions:

- Use the Logic Editor to view the pre-written Java source code
- Create and generate a call spec
- Replace existing trigger logic with a call to the Java call spec

Your Assignment

Use the information below as your starting point:

Tool	Design Editor
WorkArea	ORA<nn>_PWA_01
Application System	NF_HOLLYWOOD

where <nn> is the number assigned to you by your instructor.

Using the Logic Editor to view the pre-written Java source code

1 The code required for your Java trigger already exists in the Java source element JS_UPDPRICE. Investigate the Java source definition.

a What is the short name of the Java source definition?

b What is the implementation name of the Java source definition?

2 Drag the Java source element JS_UPDPRICE and drop it onto the work surface in the Design Editor. Examine the code as presented by the Logic Editor. This is the logic that you want to use for the “after row” trigger on table PRICE_HISTORIES.

3 Examine the existing trigger, UPD_PRICE, and the associated PL/SQL trigger logic on the PRICE_HISTORIES table.

Defining and generating the call spec

4. Use a property dialog to create a PL/SQL procedure definition declaratively, named CALL_JS_UPDPRICE. Use the following table to assist you:

Property	Value
Short Name	CALL_JS_UPDPRICE
Implementation Name	CALL_JS_UPDPRICE

5. Add PL/SQL arguments to your call spec:

Argument	Datatype	Maximum length
P_NEW_PRICE	NUMBER	8,2
P_NEW_DEFAULT_DAYS	NUMBER	2
P_NEW_CODE	VARCHAR2	3

6. Make this procedure a call spec by specifying that you want the Java source schema object JS_UPDPRICE to supply the trigger logic. Use the following table to assist you:

Property	Value
Java Definition	JS_UPDPRICE (short name)
Java Method	pchPrice
Parameter List	float, int, char

7. Generate the call spec CALL_JS_UPDPRICE to file and examine the generated DDL and notice the PL/SQL arguments and the Java parameters.

Replacing the existing trigger logic with a call to the Java call spec

8. In the database trigger definition for the PRICE_HISTORIES table, replace the PL/SQL block with the following call statement:

```
CALL_JS_UPDPRICE(:NEW.price, :NEW.default_days,
:NEW.pl_price_code)
```

9. Generate the table PRICE_HISTORIES to file and examine the generated DDL for the associated trigger.

Practice 6 – Hints

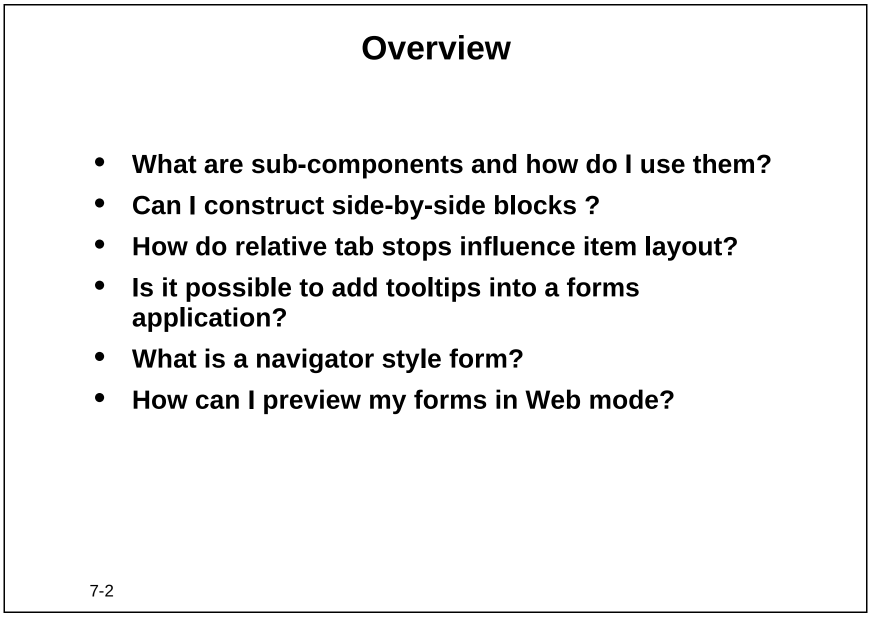
To Do This Task	Follow These Steps
Finding the short name of the Java source definition.	1 Expand the Java Definitions node. 2 Select the Source Definition node. 3 Select the required Java definition and invoke the property palette.
Examining code using the Logic Editor	1 Expand the Java Definitions node. 2 Select the Source Definition node. 3 Select the required Java definition drag it onto the work surface to invoke the Logic Editor.
Examining a trigger and its associated PL/SQL trigger logic	1 Expand the Relation Table Definitions node. 2 Select the required table and expand the node. 3 Expand the Trigger node. 4 Select the required trigger drag it onto the work surface to invoke the Logic Editor.
Creating a PL/SQL procedure definition	1 Expand the PL/SQL Definitions node. 2 Select the Procedure Definition node and click the Create button. 3 Complete the required properties.
Making a procedure a call spec	1 Expand the PL/SQL Definitions node. 2 Select the Procedure Definition node and click the Create button. 3 Complete the required properties. 4 Ensure you complete the Java properties and not the PL/SQL properties

7

Oracle Forms Generation: Enhancing the End User Interface

Introduction

The aim of this lesson is to introduce Forms Generator functionality to enhance the end user interface, allowing for more direct placement of blocks, items, and summary items. This lesson also introduces enhanced block navigation using navigator style forms.



Overview

- **What are sub-components and how do I use them?**
- **Can I construct side-by-side blocks ?**
- **How do relative tab stops influence item layout?**
- **Is it possible to add tooltips into a forms application?**
- **What is a navigator style form?**
- **How can I preview my forms in Web mode?**

7-2

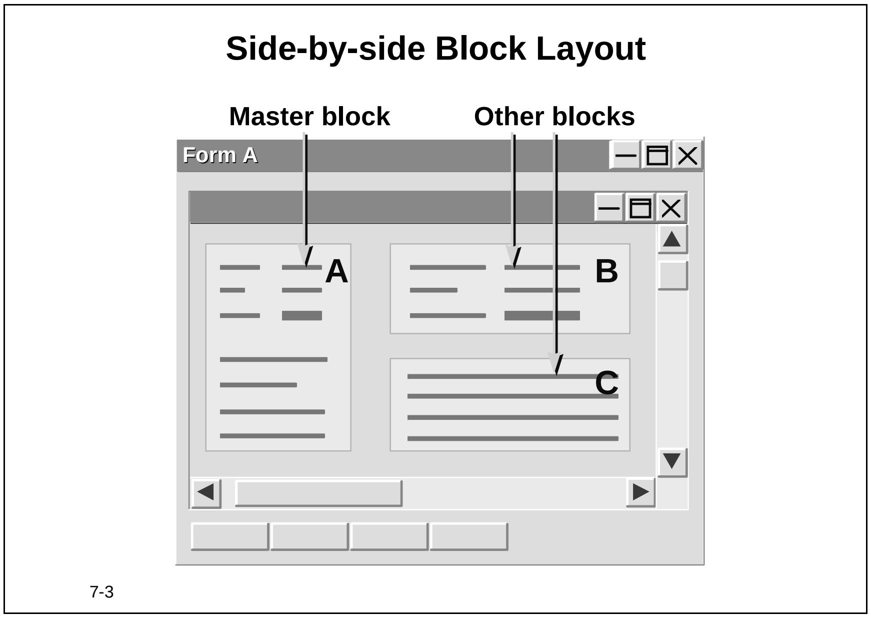
Objectives

At the end of this lesson, you should be able to:

- Construct side-by-side blocks
- Develop and incorporate sub-components within modules
- Create multi-region blocks using module components and sub-components
- Use relative tab stops to influence layout
- Define and generate navigator style forms
- Generate tooltips into a forms application
- Preview forms in Web mode

Side-by-side Blocks

A block placed to the right of another block in the same window is a side-by-side block.



Side-by-side Block Layout

7-3

What is a Side-by-side Block?

Information from a detail block is traditionally displayed beneath the master block on the generated form. However, there are times when placing information to the right or left of other blocks would be advantageous, such as to avoid scrolling down the canvas that extends off the screen, or when changing to different windows.

In the diagram above, block C could be placed to the right of block B. Alternatively, blocks B and C could be placed side-by-side, and both in an area below block A. Blocks B and C above do not have to be detail blocks of the master block.

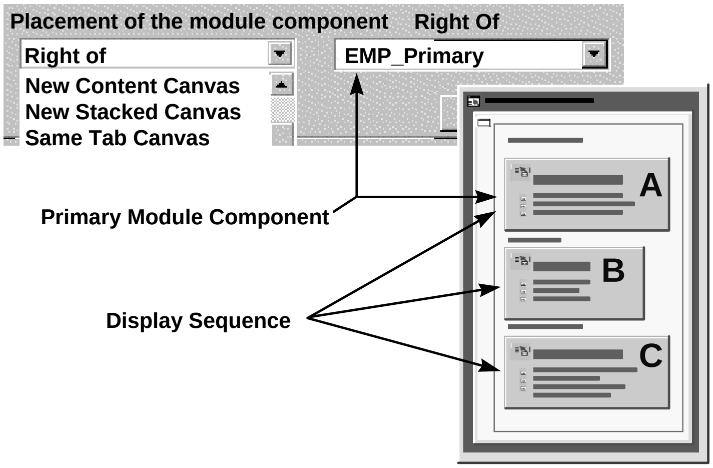
Side-by-side Blocks and Sequence Numbers

You can place any block beside another block, thus creating side-by-side blocks, provided that the sequence number of the module component on the right is higher than the sequence number of the module component on the left.

In the diagram above, module component A must have a sequence number lower than both B and C. If you placed C to the right of B, then C must have a higher sequence number than B.

Implementing Side-by-side Blocks

Implementing Side-by-side Blocks



Placement of the module component Right Of

Right of

EMP_Primary

New Content Canvas

New Stacked Canvas

Same Tab Canvas

7-4

How to Implement Side-by-side Blocks

Side-by-side block layout is activated in four steps.

1. Ensure the blocks to be laid out side-by-side are from module components on the same canvas in the same window.
2. Choose the module component you wish to place on the right and edit this module component using the dialog.
3. Select the Right of value in Placement of the module component property.
4. Select the name of the module component which will generate on the left.

The resulting layout places the module component being edited in step 2 above to the right of the module component chosen in step 4 above.

Limits to Side-by-side Block Placement

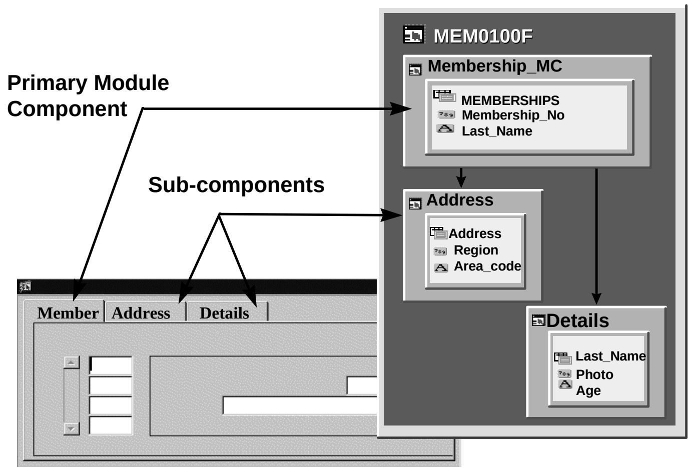
There is no limit to the number of side-by-side blocks you can create for one form. The Form Generator increases the horizontal width of the canvas to accommodate them all. By increasing the height of the first block generated, you can position two or more blocks, one under the other, to the right of this first block.

You cannot have tabbed blocks to the right of another block using the methods described here. The options of Same Tab Canvas Page and Right of are mutually exclusive.

Working in a Multi-region Block

A module component with one or more sub-components, all using the same base table usage and related lookup table usages, is generated as a multi-region block. This allows you to split blocks across tabbed canvases and in different windows.

Working in a Multi-region Block



7-5

Multi-region Blocks and Sub-components

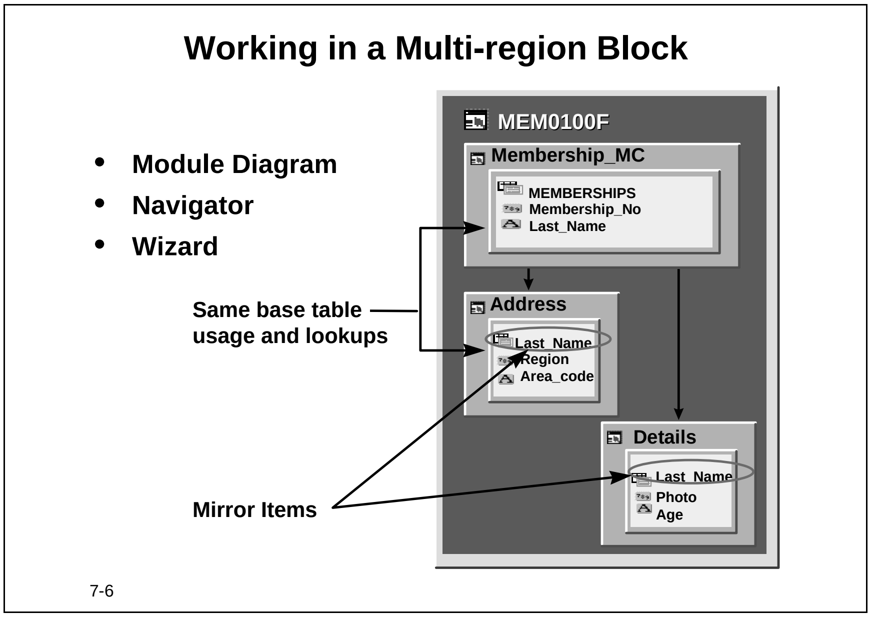
A multi-region block comprises a number of different rectangular layout regions. Form Generator creates a rectangular layout region for the primary module component and for each module sub-component in a single block in the generated form.

Items in a Multi-region Block

Sub-components allow items based on columns from the same table to be repeated on two or more regions, or different items from the same base table to be shown on different regions.

- The data in different regions in a multi-region block is always synchronized, that is, the regions all display data for the same current record.
- The different layout regions can display different numbers of rows.
- The different regions can be on different tabs and different windows.

Creating Sub-components



Working in a Multi-region Block

- **Module Diagram**
- **Navigator**
- **Wizard**

Same base table usage and lookups

Mirror Items

7-6

How to Create a Sub-component Using the Wizard

1. Open the module diagrammer in the Data View.
2. Click the Create Sub-component button.
3. Move the cursor onto the module component to which the sub-component will connect.
4. Click again to invoke the sub-component wizard.

You cannot select any other base table usage, but you may optionally choose any lookup table usage.

Renaming Second and Subsequent Items

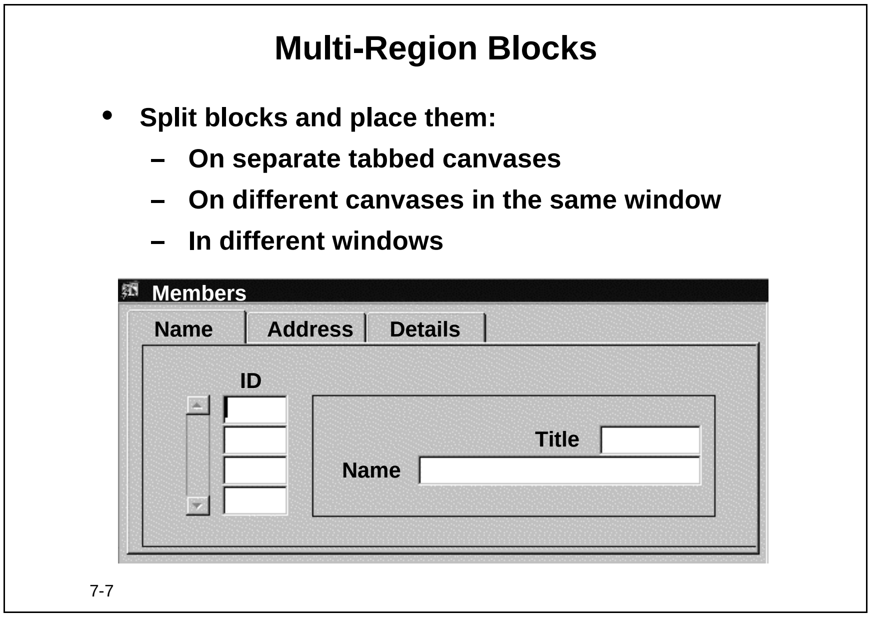
Items that are placed more than once across the components or sub-components are generated as mirror items. The standard source objects for these mirror items will be the same as for the first bound item. It will not be CGSO$CONTEXT, which is the usual standard source object for mirror items. You can also set column usages for second and subsequent items. These items will remain synchronized during updates.

Give different names to bound items based on the same column. If you do not do so, Form Generator will generate unique names for second and subsequent items based on the same column. It does this by appending a suffix string to the generated item names.

Properties of Sub-components

Most of the properties of a sub-component derive from the primary module component. You can set item properties for items in the sub-component, but block properties such as insert, update, and delete are set in the primary module component.

Placing Components in a Multi-region Block



Multi-Region Blocks

- **Split blocks and place them:**
  - **On separate tabbed canvases**
  - **On different canvases in the same window**
  - **In different windows**

7-7

Layout of a Multi-region Block

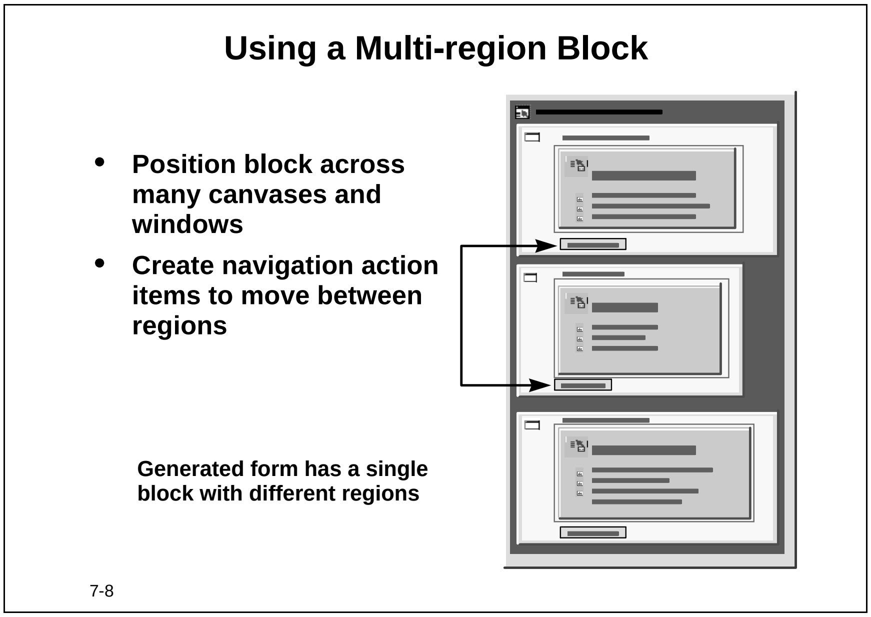
You can place the layout regions in a multi-region block:

- On different places on the same canvas
- On different canvases in the same window
- On different canvases in different windows

Using Scrollbars

You can generate multiple module sub-components as multi-row layout regions. However only one multi-row region in a multi-region block can have use of the block’s scrollbar. Use the Multiregion preference (MRBVSB) to specify which multi-row region has the scrollbar.

Navigation in a Multi-region Block



Using a Multi-region Block

- **Position block across many canvases and windows**
- **Create navigation action items to move between regions**

Generated form has a single block with different regions

7-8

If you generate multi-region blocks, navigation between the regions will depend upon whether the regions are on canvases, windows, or tabbed canvases.

If the multi-region is across different canvases in one window, then perform one of the following:

- Set the Restrict item navigation to items on the current canvas (NAVPAG) preference to Yes.
- Add navigation action items.

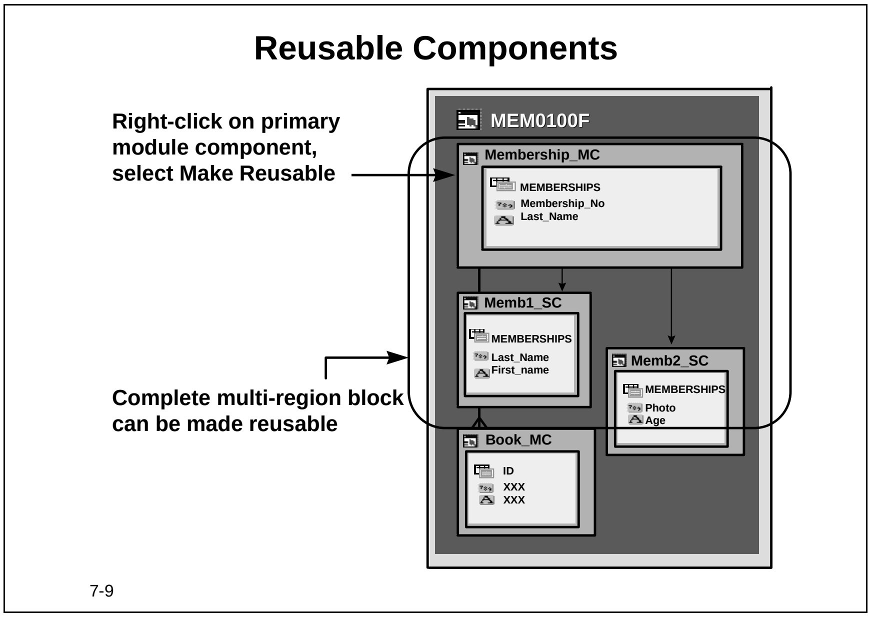
By setting the NAVPAG preference to No, navigation is to the first item in the layout region on the next canvas. If you wish more direct user control, then add navigation action items to move from one canvas or window to the next.

If the multi-region is across different non-modal windows, then you can also access each window (and thus each region) through normal mouse-clicks within the GUI.

If the multi-region is across tabbed canvases, then you can click the tab to make it active and navigate to it.

Reusable Components

You can make module components and module sub-components reusable.



Reusable Components

Right-click on primary module component, select Make Reusable

Complete multi-region block can be made reusable

7-9

How to make a Component Reusable

In the Data View:

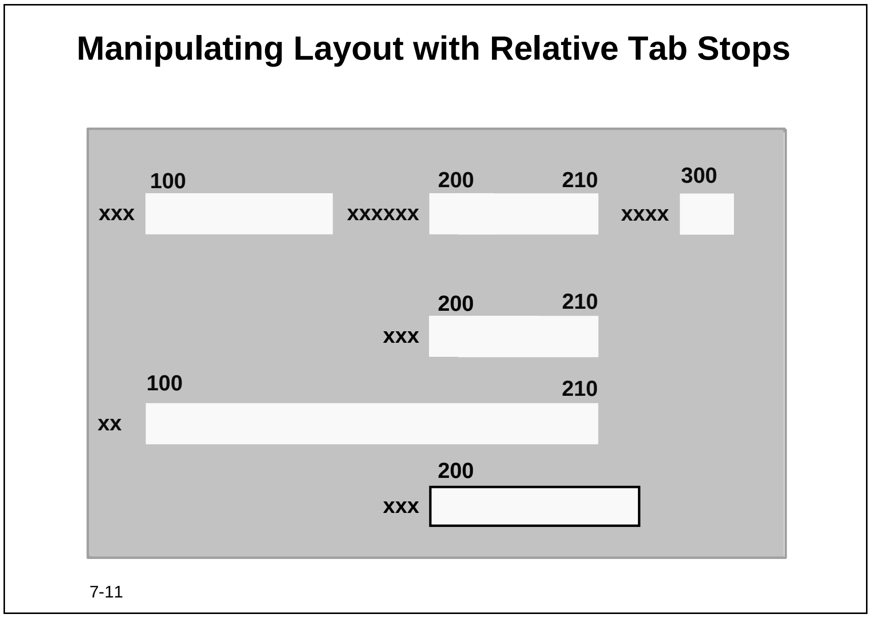
1. Right-click the module component.
2. Select Make Reusable.

Including a Reusable Module Component

When you include a reusable module component you also include all of its sub-components. You are not permitted to add further sub-components to an inclusion of a reusable module component. However, you can add further sub-components to the original reusable module component. These new sub-components are not added to any of the present inclusions of that reusable module component, but will appear in any future inclusions.

Relative Tab Stops

Relative Tab Stops are numeric values that enable you to position and align items and item groups relative to each other, that is, next to each other or below each other.



Manipulating Layout with Relative Tab Stops

7-11

Understanding Relative Tab Stops

Relative tab stops are not fixed tab stops, and any fixed tabs set previously are ignored when the new relative tab stops are invoked.

Each item may have a start relative tab stop and an end relative tab stop. All the items with the same start relative tab stop will left align with each other, and all the items with the same end relative tab stop will right justify with each other.

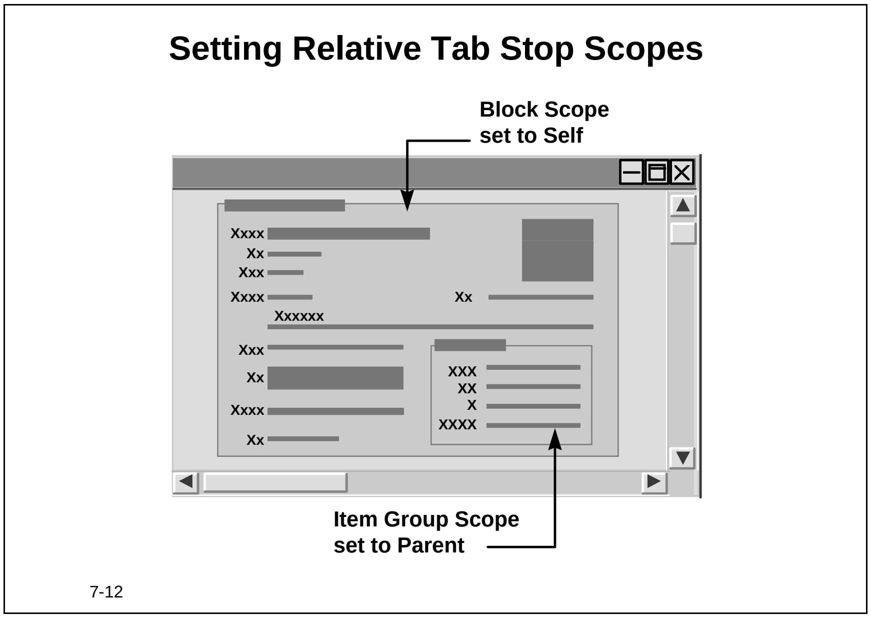
What is the Effect of a Particular Number?

The number you provide for a relative tab stop has no intrinsic value, but does have a purpose. The lowest number you provide will be the leftmost relative tab stop, and the highest value you provide will be the rightmost relative tab stop. You may enter any number provided that, for any given item, its start tab stop number is lower than its end tab stop number, and that each number is an integer greater than zero.

- Items stay in the same order in which they are placed on the form.
- Items with a tab number higher than the previous tab are placed to the right of the last item.
- Items with a tab number lower than the previous tab are placed on the next line.
- Items that have no tab number will simply go to the right of the previous item.
- Items are laid out below previous items aligned to the same tabs.
- No tab is placed to the left of a tab with a lower number.

Invoking Relative Tab Stops

Relative tab stops only work when you invoke them by setting a block or group scope.



Setting Relative Tab Stop Scopes

Block Scope set to Self

Item Group Scope set to Parent

7-12

Setting the Scope

To turn on relative tab stops, set the block or item group Tab Stop Scope property to Parent or Self from the property palette.

Scope Setting	**Description**
Self	Allows the generator to dynamically allocate tabs only according to items within that block or item group.
Parent	Allows interaction between the settings of blocks and groups if set in each of two blocks (or of Self in a block and Parent in an item group)
None	Turns off relative tab stops for the block or group

Freeing Item Groups from the Effects of a Parent Block Scope Setting

Consider creating an item group that:

- Has no tab scope set
- Exists within a block that has a tab scope set to Self or Parent
- Contains at least one item with relative tab stops set

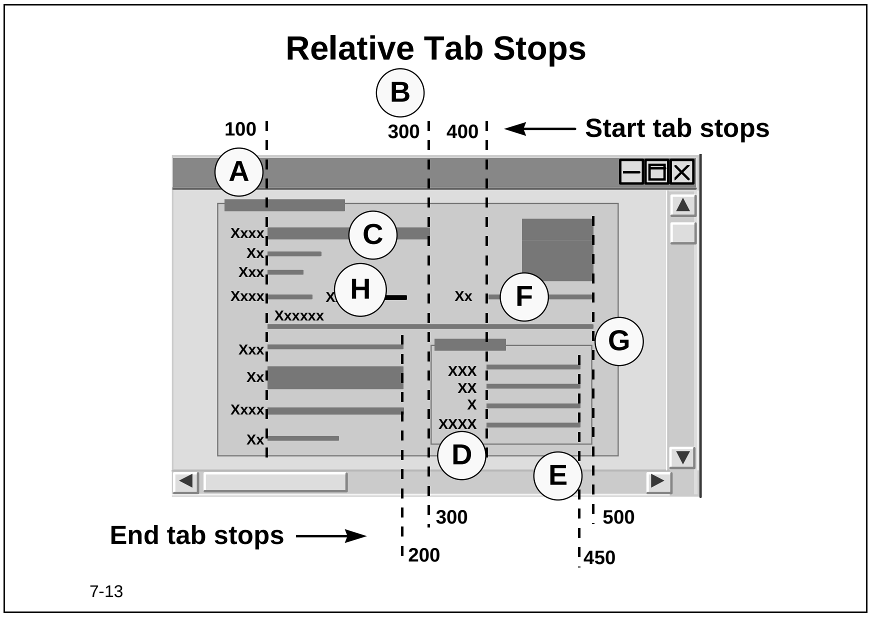
Then, the tool will attempt to align the items within the item group to the relative tab stops of the parent block. If you do not want to have alignment between the block and the item group, remove the relative tab stops of the items within the item group.

Forgetting to Set a Tab or a Scope

If you set scope to Parent or Self, and have no relative tab stops set, all the items will align to the right of the previous item, and the canvas will expand horizontally.

If you have no scope set, all relative tab stops will be ignored (although the caveat in the Freeing Item Groups still applies).

Dynamic Effects of Relative Tab Stops

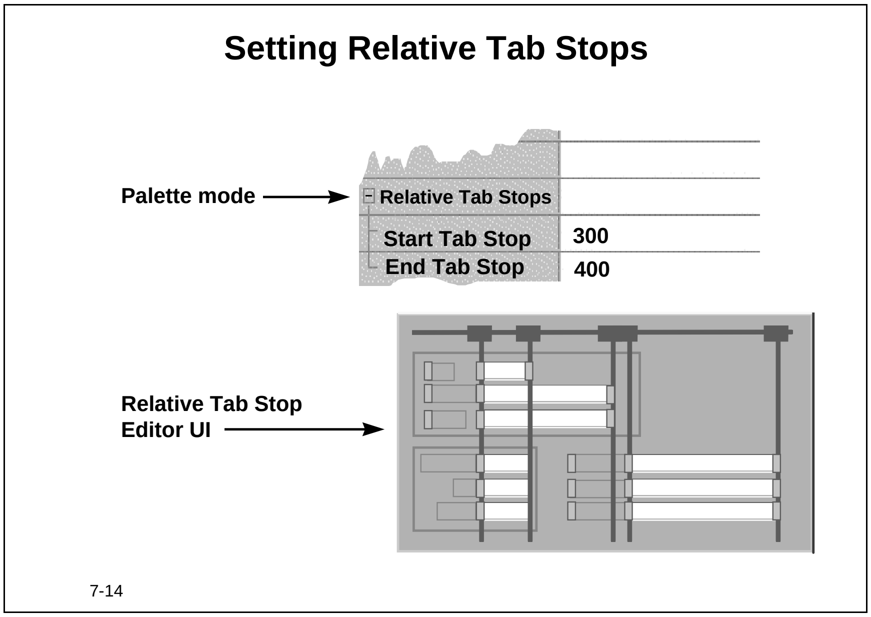


In the diagram above, the Block Vertical Fill preference (BLKVFL) is set to Yes. Form Generator utilizes the vertical space next to tall items, such as the space to the left of the item group D.

A	Tab scope is Self for the block and Parent for the item group. Tab stop 100 is the lowest number used, so all items with this as a start tab stop begin a new line and are left adjusted. Three items with an end tab stop of 200 align right.
B	Tab stop 300 is used as both a start tab stop on item group D and an end tab stop on item C.
C	This item, although only 8 characters wide, has a start tab stop of 100 and an end tab stop of 300. The width has extended because, after iterative passes, the generator found the start of the item group D had to be placed to the right of any items with an end tab stop of 200. Item C was extended accordingly.
D	The item group uses 300 as a start tab stop, but the items within the group have a start tab stop of 400. Because the tab scope is set to Parent, the items within this group will align to the 400 tab start stop in the parent block, which is item F.

E	All items in the item group are aligned right to end tab stop of 450.
F	The item marked F has a start tab stop of 400, making it left align with the item group items below it, and an end tab stop of 500, making it align right with the item group bounding box.
G	The item group bounding box, and three other items in the parent block all align right with end tab stop of 500, which is the highest number used.
H	The item and prompt marked H has no tab stops set at all, and defaults to the same line as the previous item, moving as far to the left as it can.

Using Relative Tab Stops



Setting Relative Tab Stops

Palette mode

Relative Tab Stops

Start Tab Stop 300

End Tab Stop 400

Relative Tab Stop Editor UI

7-14

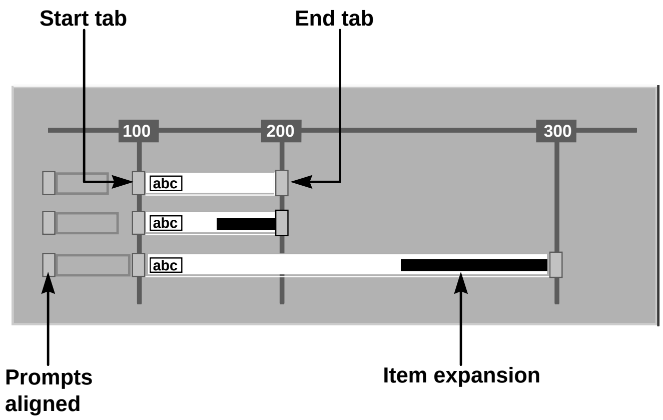
Interface Choices for Setting Relative Tab Stops

You may set relative tab stops using any of three methods:

Palette	At block level choose to set the tab scope; at item level, set the start and end tab stops, and choose to align prompts.
Dialog	Fill in the start and end boxes, and set the Align Prompt? box to Yes.
Relative Tab Stop Editor	A GUI editor which gives an approximate representation of how the final form is generated. Use the shortcut menu on the display view of a module component to invoke the GUI editor.

Using the Relative Tab Stop Editor

Relative Tab Stop Editor



7-15

The Relative Tab Stop Editor allows you to graphically set tab stops for items, item groups and their prompts. To invoke the Relative Tab Stop Editor:

1. Turn the module component tab stop scope to Self or Parent
2. Right click on the module component in the Display View and select the Relative Tab Stop Editor.

You can:

- Set the tab stop scope for a component and its item groups using the shortcut menu
- Set the Block Vertical Fill (BLKVFL) preference
- Enter the start tab for a selected item or group
- Specify the end tab for a selected item or group
- Align the item prompts

Graphically you can:

- Move items using drag and drop
- Select an area containing many items, item groups or components
- Select all items, item groups or components
- Drag the start and end tab stop indicators

Note: The Relative Tab Stop Editor does not display action items.

Determining the Final Output

Relative tab stops provide a very quick way to:

- Align items left and vertically.
- Justify items right and vertically.
- Move items to a new line.

The Role of the Generator

Create relative tab stops using iterative or dynamic allocation of vertical alignment.

Process	**Description**
Iterative	The generator will make a number of passes of the settings during generation, feeding back information to update the settings on each pass.
Dynamic	The physical position of the tabs is not determined until all the tab settings of all the items are compared with each other

Allocation of vertical alignment

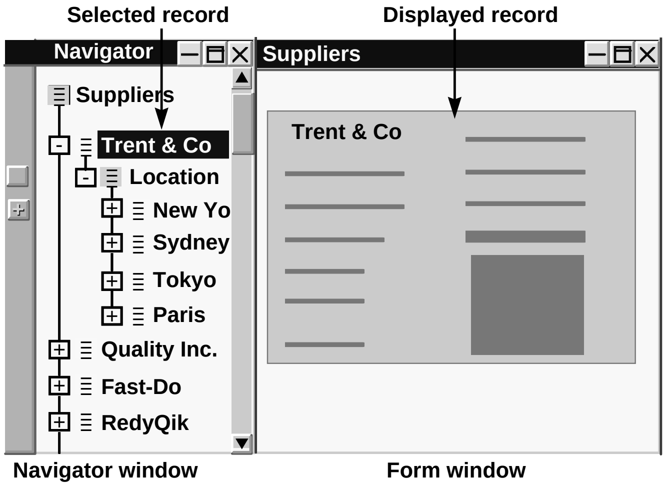
The items are placed on the canvas according to:

- Items always competing for leftmost position
- Width of item and prompt, and best fit, determining horizontal spacing
- Tab scope setting, aligning items to other relative tabs within the same block, other blocks, and item groups
- Line-by-line layout of items, top to bottom
- Vertical fill preference setting
- Prompts, whether set to display to the left of item or above item, aligned or not.

Navigator Style Forms

A Navigator Style form provides a tree view onto data using block nodes and record nodes.

Using Navigator Style Forms



7-16

Purpose of Navigator Style Forms

The hierarchical tree structure of the Navigator window provides easy access to records in detail blocks at any level. Each block node and record node expands and collapses to allow you to view each record or just the block containing records.

Clicking on the expand buttons drills down to reveal further blocks or records if they exist.

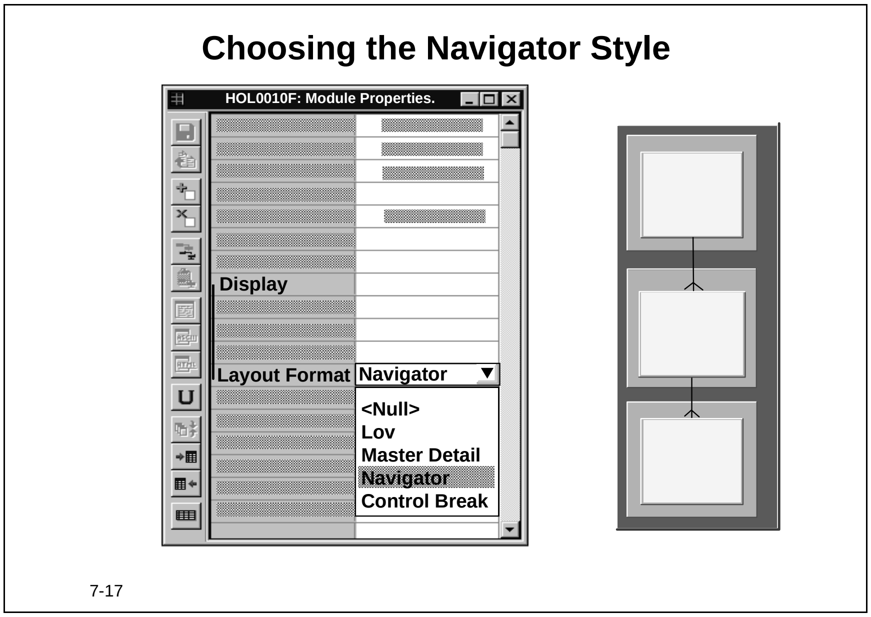
Using Navigator Style Forms

The first block of the form is generated onto a new canvas in a new window called the navigator window. This window is placed vertically left of screen, and when initially opened displays all the master blocks as unexpanded block nodes.

When you expand one of these block nodes, the records in the block are presented as record nodes.

If the master block has links to one or more detail blocks, then you can expand these record nodes to display a block node for the detail block. Similarly, you can further expand these detail block nodes to reveal their records as further record block nodes.

Choosing the Navigator Style



Choosing the Navigator Style

HOL0010F: Module Properties.

Display

Layout Format: Navigator

7-17

How to Change the Navigation Style to a Hierarchical Tree

When you want to generate a navigator style form, set the Layout Format property of the module to Navigator.

1. Open the property palette for the module.
2. Select Navigator from the Layout Format property.

To stop using a navigator style form, reset the Layout Format property to Master Detail, and regenerate the form.

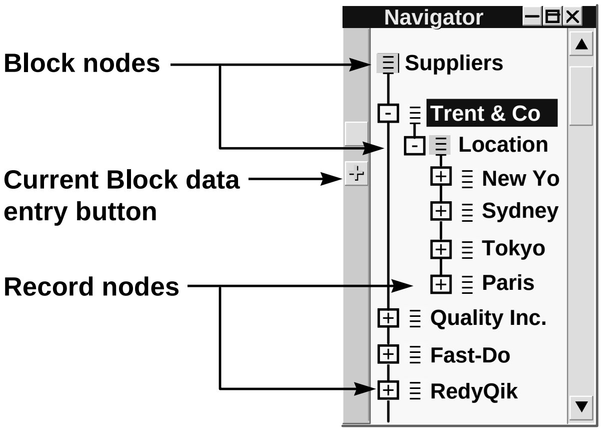
How to Define Labels for Hierarchical Trees

You can set meaningful block labels and record labels using generator preferences.

- Set the Navigator block node label preference (NVRBLB) at module component level to give a block node label. By default, the Form Generator assigns the name of the block to the label.
- Set the record node context by setting the Context? property to Yes at item level to use the item to identify records in the navigator window. By default, the Form Generator uses the primary key items.

Viewing and Editing Records Using Navigator Style Forms

Data Manipulation Using Navigator Style Forms



7-18

Locating, Viewing, and Editing Particular Records

Clicking on either the icon for a record or the name of a record brings up the generated form in another window. You can choose where to place that window by setting the X and Y coordinates in the Placement property of the window.

By default, the coordinates are (0,0).However, this will bring the new window up under or over the first window, obliterating one or other window. Setting a coordinate of, say, (35,0) will move the window 35 units to the right, and allow the navigator window to be seen.

The settings for the Navigator window are derived from CGSO$_WINDOW_NVGTR in the Object Library.

Entering New Data

You can update records and save the changes using navigator style forms.

Clicking on the name of a block removes the generated form. Only clicking on the name of a record, or its icon, presents the form for viewing or updating.

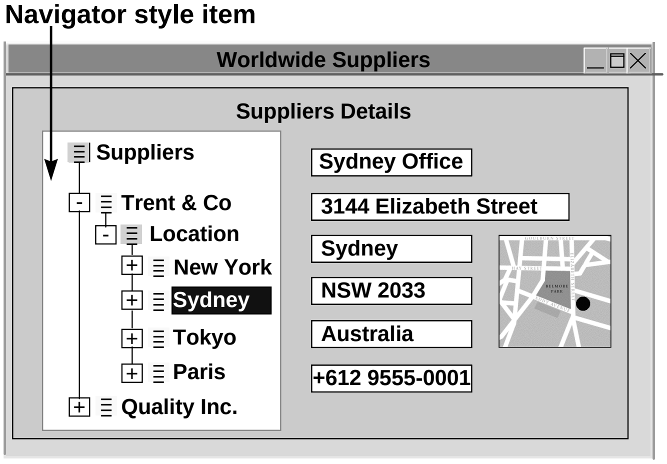
If you highlight a block or record, and then click the **Current Block Data Entry** Button, this opens a new window with a clear form ready for data entry.

All insert, update, and delete operations set for the block and record are available in Navigator style forms.

Using Navigator Style Items

Using Navigator Style Items

Navigator style item



Worldwide Suppliers

Suppliers Details

- Suppliers
  - Trent & Co
    - Location
      - New York
      - Sydney
      - Tokyo
      - Paris
  - Quality Inc.

Sydney Office

3144 Elizabeth Street

Sydney

NSW 2033

Australia

+612 9555-0001

7-19

Using an Unbound Item to Present Data in Navigator Style

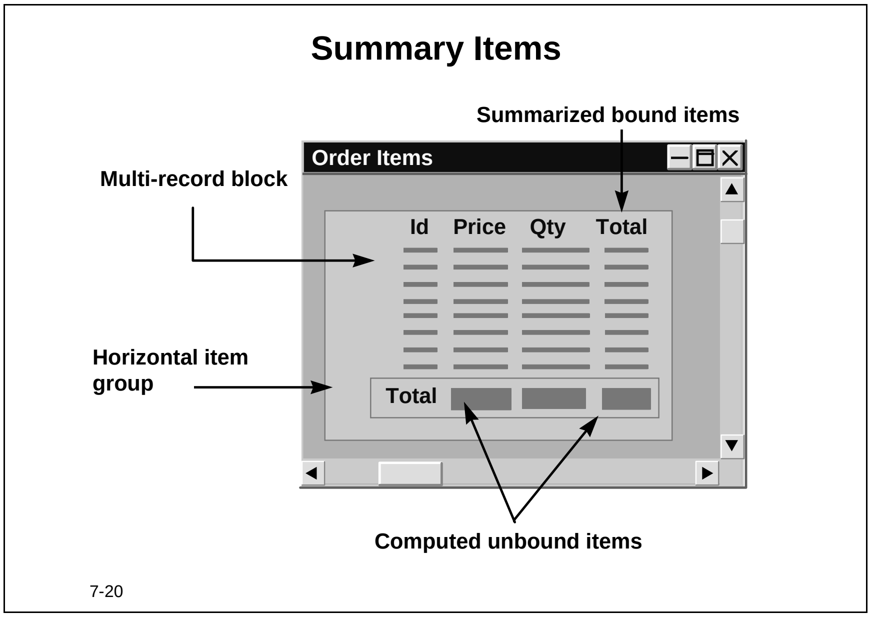
Navigator style items are a type of repository definition. You create an unbound item and make it of display type navigator style, then place it wherever you want.

A navigator style unbound item provides a generated hierarchical tree populated with data with all blocks shown as block nodes, and when expanded, the records in the block appear as record nodes.

How to Create a Navigator Style Item

1. Create an unbound item
2. Set the Unbound Type property to Custom
3. Set the Display Type property to Navigator Style Item
4. Set any WHERE clauses or ORDER BY clauses in the Query property to restrict and order returned records
5. Set the preference to automatically query and display the record selected at the record node (NVRWAS) to Yes to allow auto-select functionality.

Summary Items



Summary Items

7-20

Summarizing Items in Blocks

You can choose to display a summary item for any unbound item of type Computed. Oracle Designer 6*i* generates and places the summary item beneath the summarized bound items within multi-record blocks in which the lines do not wrap. It may also place an unbound summary item in a different block to the item being summarized.

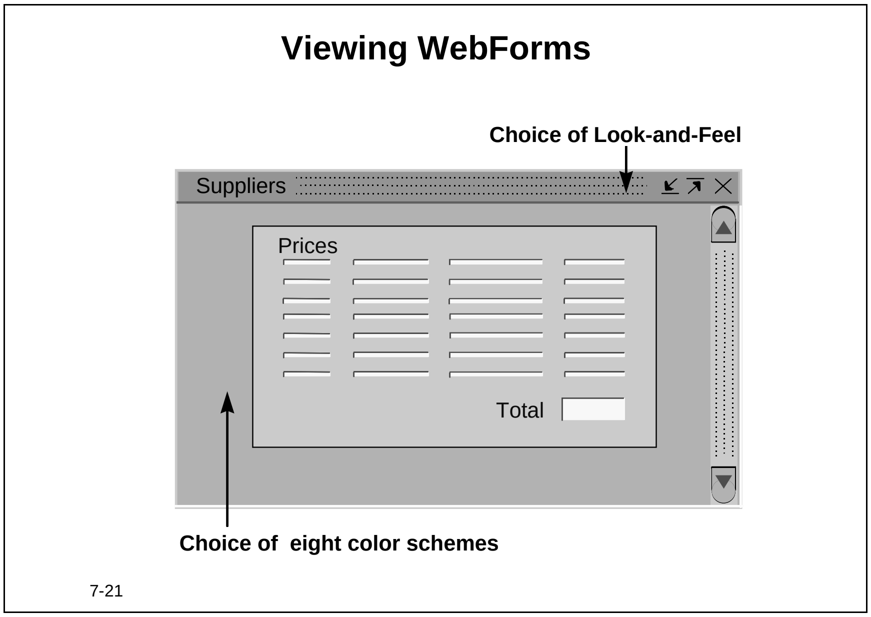
Placing Summary Items in Item Groups

You can choose to display summary items in vertical or horizontal item groups.

- When placed in a horizontal item group, the summary items are automatically below the items they are summarizing, provided only summary items are in the horizontal item group.
- When placed in a vertical item group, the item group is left-justified and the summary items are not placed below the items they summarize.

Running WebForms from Oracle Designer 6*i*

You can run a WebForm either using the Forms Web Previewer or over a web browser.



Viewing WebForms

Choice of Look-and-Feel

Choice of eight color schemes

7-21

There are a number of steps to take and preferences to set for running WebForms directly from Oracle Designer 6*i*.

Running a WebForm using the Web Previewer

The Web Previewer enables you to run forms as though they are being deployed from a web server, without the need for any Web browser or a Web server. This means that you can use the Web Previewer as a tool to test your web forms without having to actually deploy the forms on a server.

If you choose to view the form in Web preview mode, a component called the Web Previewer that runs a Java Applet called the WebForms Applet. This is a browser independent applet.

Setting Web Previewer Preferences

The Form Generator creates an HTML file containing runtime parameters to pass to the Web Previewer. This is stored in the temporary directory (e.g. c:\temp). You can set a number of preferences to control the HTML file parameters:

Preference	**Description**
WEBCOL	Preview mode color scheme
WEBHGT	Preview window height
WEBLAF	Preview mode look-and-feel style
WEBWID	Preview window width

Running the Web Previewer

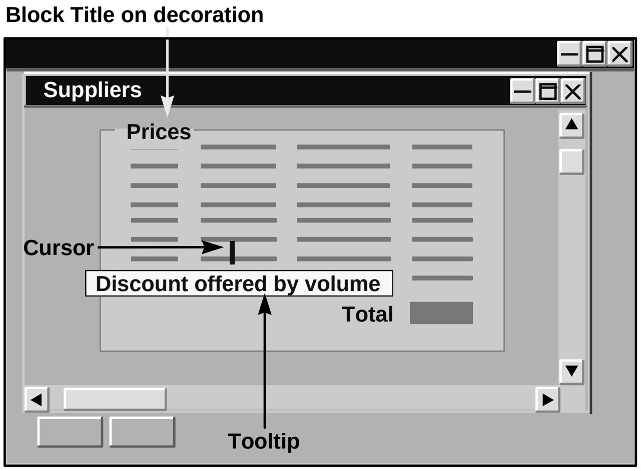
To run the WebForms applet automatically after generation:

1. Go to the Forms Generator Options dialog.
2. Select **Run as a Web Form** from the Run tab.
3. Ensure the **Run Form Automatically** check box is also selected.

Alternatively, at any time after generation you can select **Run the form in Web preview mode** from the List Actions box in the message window.

Refining Forms using Preferences

Refining Forms Using Preferences



7-22

Generating Form Tooltips

You can provide item-sensitive tooltips for your users which are activated as the mouse pauses above an item field.

Enter the text into the item Hint property and set the Provide tooltips for items preference (ITMTIP) at any level from application system to item. The options are:

All	Any item with text in the Hint property displays that text as a tooltip.
Button	Only buttons with text in the Hint property display that text as a tooltip.
None	Items with text in the Hint property do not display that text as a tooltip.

Manipulating Block and Item Decorations

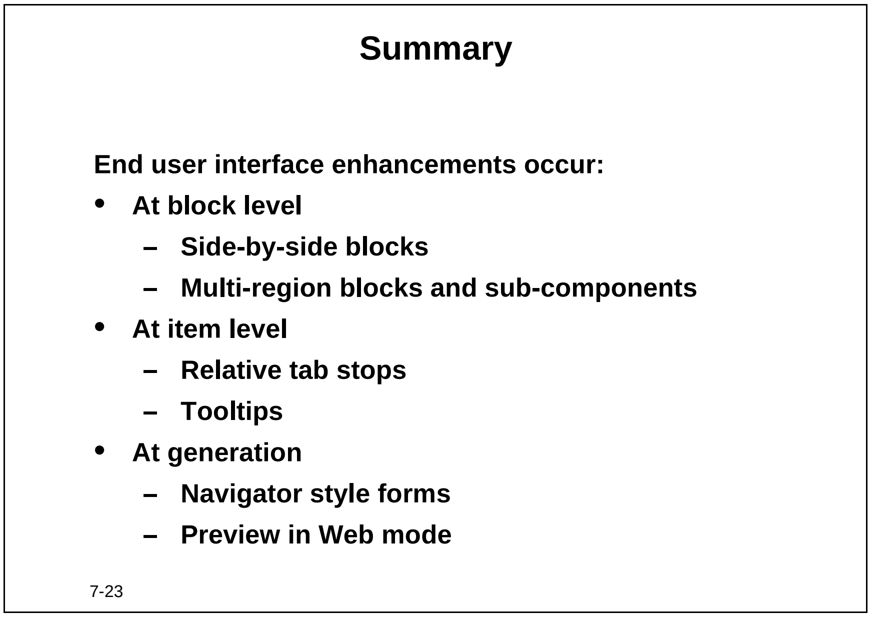
You can now place Block titles on the block decoration, and not just below it or above it. Set this property using the Block title position preference (BLKTLP) with On Decoration selected.

If the block is surrounded by a raised or lowered bevel, then the title will be placed inside the decoration.

Real Units for Decoration Preferences

You can declare all units used to enter length and size in any real unit (pixel, point, decipoint, centimeters, or inches). If you specify values for a preference in units different to those used in the form, Form Generator converts values as necessary.

Summary



Summary

End user interface enhancements occur:

- **At block level**
  - **Side-by-side blocks**
  - **Multi-region blocks and sub-components**
- **At item level**
  - **Relative tab stops**
  - **Tooltips**
- **At generation**
  - **Navigator style forms**
  - **Preview in Web mode**

7-23

The end user interface enhancements provide many new options to allow more exact placement of items and blocks on the generated form.

Web Preview Mode

Web preview permits developers to test out a Web application without the need of Web browsers or application servers.

Navigator Style Forms

Navigator style forms provide windows of a hierarchical tree view. You can access data in a tree view by drilling down. Navigator style items provide a tree view within an item.

Practice 7 – 1: Focus on Sub-components and Side-by-Side Blocks

Goal

The purpose of this practice is for you to manipulate form modules based on tables with many columns using sub-components, side-by-side blocks, or a mixture of both in order to clearly present the data.

Scenario

The system designers have found it necessary to place many columns in a single table. This table contains important personal information about each client. The designers have asked you to modify the form based on the table so that the data is easily accessible.

They have decided on an action which involves demonstrating three alternatives:

- Investigate and prepare the module
- Place some of the items in another window
- Place some of the items in a second block beside the first block
- Place the items on a series of tabbed canvases where one or more of the canvases contains a side-by-side block

Your Assignment

Use the information below as your starting point:

Tool	Design Editor
Workarea	ORA<nn>_PWA_01
Application System	HOLLYWOOD
Module	HOL0010F

where <nn> is the number assigned to you by your instructor.

Investigating and Preparing the Module

1 Using the Design Editor, create a diagram of the module HOL0010F. (You need not save this diagram.)

  a How many module components are there?

  b What table usages does this module have?

  c What is the overflow style of the module component?

  d Which columns will the generated form display?

2 Generate and run the module.

  a Are all the items easily accessible?

  b What problems do you foresee for a user with this form?

3 Remove all the address items and the EMAIL_ADDRESS and HOME_PHONE items from the primary module component.

4 Using a property dialog, create a module sub-component, within HOL0010F, using the following information:

Property	**Value**
Sub-component Name	CUSTOMER_SC
Title	Customer Home Details
Base Table Usage	Same as the primary module component
Columns	All the address columns, EMAIL_ADDRESS, HOME_PHONE

5 Ensure the sub-component is on new content canvas in a new window.

6 Generate and run the form. Execute a query in the form and tab through the items.

- a What problem have you encountered in viewing all the items?
- b How could you overcome this?

7 Return to the Design Editor and place each of the components onto new tab canvases in the same window.

8 Confirm the size of the content/stacked canvas of primary module component is set to 80 by 40.

9 Add LAST_NAME as a mirror item on the sub-component. Position it to be displayed first and give it an appropriate Prompt.

10 Generate and run the form.

- a What difference has this made to the way the user gains access to the information on the form?

11 Create another sub-component in the module HOL0010F, using the following information:

Category	**Property**	**Value**
Display	Name	CUSTOMER2_SC
Display	Title	Work Details
Size	Columns	WORK_PHONE, WORK_EXTENSION

12 Ensure component CUSTOMER_MC does not display the work phone and the work extension.

13 Place CUSTOMER2_SC on a new tab canvas in the same window.

14 Generate and run the form.

- **a** Does this provide a clearer user interface for the user?

15 If you chose to place each of the components into separate windows, what additional issues would you need to consider?

Displaying Side-By-Side Blocks (If you have time)

16 Investigate the data and display views of the module HOL0050F.

17 Move the REVIEWS_MC onto the same canvas in the same window as EMP_MC.

18 Choose to display REVIEW_MC to the right of EMP_MC.

19 Generate and run the form.

Practice 7 – 2: Generating Navigator Style Forms

Goal

The purpose of this practice is for you to develop and utilize a new way of navigating from parent to grandchild records.

Scenario

To more readily access details of the items that are rented by customers, the developers have discovered that Oracle Designer 6*i* implements a new navigator style.

They have decided on the following actions:

- Implement the new navigator style
- Add a summary item to the Rental Item form
- Add some tooltips to specific items

Your Assignment

Use the information below as your starting point:

Tool	Design Editor
Workarea	ORA<nn>_PWA_01
Application System	HOLLYWOOD
Module	HOL0030F

where <nn> is the two-digit number assigned to you by your instructor

Implement the New Navigation Style

1 Look at the structure of the module HOL0030F.

a How many module components do you have and what are they?

b How many windows do you have?

2 Generate and run the form HOL0030F.

3 Find the rental items booked out or returned by customers Fermum and Brandt.

a What difficulties do you have in finding and presenting the great-great-grandchild detail records?

4 Implement the new navigator style by changing the module layout format to Navigator for HOL0030F.

5 Generate and run the form HOL0030F.

a What labeling problems do you observe?

b What layout problems did you find?

c How easy is it to find Brandt?

6 Refine the layout of the module by using the Edit Window dialog for each of the windows in the HOL0030F module. Set the following:

Property	**Value**
X-position	45
Y-position	0

7 Set the context property for the items listed in the following table, in order to display the record node labels:

Module Component Inclusion that contains the item	**Record node label for the ITEM**	**Value**
CUST_MC	Last_Name	Yes
MEMB_MC	Cus_Id	Yes
RENT_MC	Rental_Date	Yes
RENTITEM_MC	Return_Date	Yes

a What is the impact of not setting the context property?

8 Use NVRBLB to set the block node labels at the module component inclusion level as follows:

Module Component Inclusion	**Block node label**
CUST_MC	Customers
MEMB_MC	Memberships
RENT_MC	Rentals
RENTITEM_MC	Rental Items

9 Generate and run the form.

10 Find the customers Fermum and Brandt, using navigator style form.

a Where do the canvases now display?

b Is the information now presented in the navigator useful? Explain.

Practice 7 – 3: Refining the Layout

Goal

The purpose of this practice is for you to decide the best layout for items on a single block and across multiple blocks using the Relative Tab Stops feature.

Scenario

Some early forms for the new system have been given to you, but the layout is quite unhelpful and needs to be modified for usability. You are accustomed to updating the forms within Form Builder. However the design team want you to make the modifications within Designer.

You need to take the following action:

- Use the relative tab stops functionality to improve the appearance of the layout.

Your Assignment

Use the information below as your starting point:

Tool	Design Editor
Workarea	ORA<nn>_PWA_01
Application System	HOLLYWOOD
Module	HOL0020F

where <nn> is the two-digit number assigned to you by your instructor.

Using the relative tab stops functionality to improve the appearance of the layout

1. Generate and run the basic, unrefined form HOL0020F.
   a. What problems do you see with this layout?
2. Lay out the module, working on the Video Games block and using Relative Tab Stops to set items and item groups. Use the following information to assist you, placing the items in display order as shown in this table:

Block or Item Group	Column	Start Tab Stop	End Tab Stop
	TITLE	100	
	PRICE	300	
	DESCRIPTION	100	
	DURATION	300	
	TI_TYPE		

Block or Item Group	Column	Start Tab Stop	End Tab Stop
	PREVIEW		
Movie Information			
	MOVIE_CATEGORY		
	AUDIO		
	MONOCHROME		
Game Information			
	MEDIUM		
	GAME_CATEGORY		
	MINIMUM_MEMORY		
	DEFAULT_DAYS	100	
	PRODUCT_CODE		
	PL_PRICE_CODE		
	AGE_RATING	200	

3 Ensure the BLKVFL, Block Vertical Fill, in the module level generator preference is set to Yes.

4 Set the Tab Scopes as appropriate.

a What tab scope settings must be set, and where should you set these?

5 Generate and run the form.

6 Tab through the generated items.

a Why is PRICE not positioned closer to TITLE?

b Why do items remain to the right of DURATION instead of wrapping around to the next line?

c Which setting would you change in order for TI_TYPE and the rest of the information to wrap on to the next line?

d Which relative tab stops can you set to align the column PREVIEW and the item groups MOVIE INFORMATION and GAME INFORMATION, to utilize less screen space?

7 Using the module HOL0020F, modify the following settings:

:

Item Group or Block	Column	Start Tab Stop	End Tab Stop
	TITLE	100	
	PRICE	300	400
	DESCRIPTION	100	400
	DURATION	300	400
	TI_TYPE	300	400
	PREVIEW	100	
Movie Information		300	400
	MOVIE_CATEGORY		350
	AUDIO		350
	MONOCHROME		350
Game Information		300	400
	MEDIUM		350
	GAME_CATEGORY		350
	MINIMUM_MEMORY		350
	DEFAULT_DAYS	100	
	PRODUCT_CODE		
	PL_PRICE_CODE		
	AGE_RATING	200	

- **a** What settings do you need to set at the module component and item group level?

8 Generate and run the form.

- **a** Does the layout now provide a clearer and easier screen for the data user? Why?
- **b** What causes the width of the layout to be so large?

9 Change the DESCRIPTION column to display height 3 and width 30.

10 Open the Relative Tab Stop Editor and fine tune the relative tab stops to match the following settings, then set the width of the columns TITLE, DURATION, and DEFAULT_DAYS as noted.

Item Group or Block	**Column**	**Start Tab Stop**	**End Tab Stop**	**Width**
	TITLE	100	150	30
	PRICE	300	400	
	DESCRIPTION	100	150	
	DURATION	300	400	25
	TI_TYPE	300	400	
	PREVIEW	100		
Movie Information		300	400	
	MOVIE_CATEGORY		350	
	AUDIO		350	
	MONOCHROME		350	
Game Information		300	400	
	MEDIUM		350	
	GAME_CATEGORY		350	
	MINIMUM_MEMORY		350	
	DEFAULT_DAYS	100		25
	PRODUCT_CODE			
	PL_PRICE_CODE			
	AGE_RATING		400	

11 Generate and run the form

- **a** The display is still too wide. What single item width would you change to minimize the width substantially further?
- **b** Within what range of numbers would you need to set Start Tab Stops for item TI_TYPE and all the items within the two item groups in order to left align them with each other?

12 Return to Designer and make those changes.

13 Generate and run the form. Confirm that your tab stop number choice has performed as you thought.

Practice 7 – 4: Adding Tooltips (If you have time)

Goal

The purpose of this practice is for you to improve the usability of your forms.

Scenario

To make using the forms easier for end users, the developers have decided on the following actions:

- Add a summary item to the Rental Item form
- Add some tooltips to specific items
- View their forms as WebForms

Your Assignment

Use the information below as your starting point:

Tool	Design Editor
Workarea	ORA<nn>_PWA_01
Application System	HOLLYWOOD
Module	HOL0030F

where <nn> is the number assigned to you by your instructor

Add a Summary Item

1 Add an unbound summary display item called TOTCHRG to the RENTITEM_MC module component.

2 Use it to Sum the price paid for any particular rental.

3 Give it the prompt “Total Charge for Rental”.

4 Generate and run the form.

5 Navigate to, and determine, the Total Charge For Rentals for both Bork and Fermum.

Add Tooltips to Specific Items

6 Update the text as a tooltip for the Rental Item form using the following details:

Item	**Value**
Rental Period	Duration of Rental
Return Date	Required return date before overdue penalties begin
Price Paid	Amount for the rental of this item

7 Ensure tooltips preference is set to ALL at module level.

8 Test the tooltips by generating the form and running the form.

Run a WebForm

9 Select any of the forms you have generated and choose to run it as a WebForm with color scheme of Olive.

Practice 7: Hints

To Do This Task	Follow These Steps
Creating a sub-component using the wizard	**1** Open the module diagrammer in data view. **2** Click the **Create Sub-component** button. **3** Move the cursor onto the module component to which the sub-component will connect. **4** Click again to invoke the sub-component wizard. **5** Complete all properties as required in the practice.
Adding mirror items	**1** Create a bound item from which Form Generator is to generate the item that will be 'mirrored' by the mirror item **2** Create a second bound item based on the same column as the first bound item. **3** This is the item from which Form Generator will generate the mirror item. Note that you can create further bound items based on the same column as the first bound item. **4** Set properties for both items as required, using object library objects, preferences and repository information.
Creating tab canvases	**1** Create the module components on the same content canvas. **2** Set the Placement property of the module components to New tab canvas page.

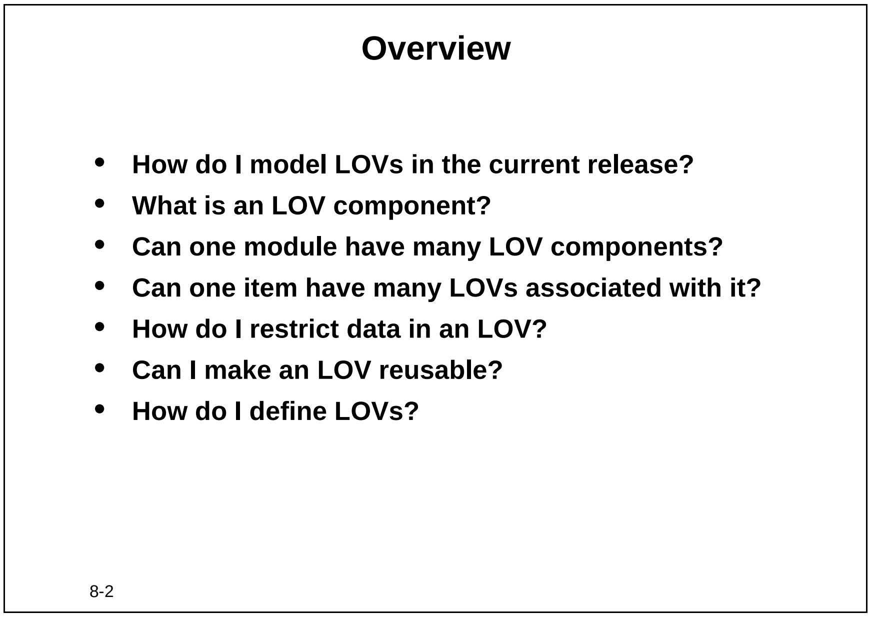
To Do This Task	Follow These Steps
Setting preferences	**1** Display the Generator Preferences Palette. **2** From the Generator Preferences Palette, select the preference that you want to set and change the value. **3** Save the preference values.
Creating an unbound summary item	**1** Create an unbound item in the module component that contains the group in which you want the computed item to appear. **2** Set the type of the unbound item to Computed. **3** Set the function and the item to use as the source of the summary: – if you are using the property dialog, use the drop-down lists to select the function and item – if you are using the property palette, enter the function and item in the Derivation Text property **4** Set the level at which the summary will be calculated – if you are using the property dialog, select one of the options – if you are using the property palette, use the Reset Level property

8

Modeling LOVs

Introduction

In this lesson you learn about generating Lists of Values. Lists of Values are now explicitly modeled and are decoupled from lookup table usages. You learn how to define and use new repository LOV elements and attach LOVs to items in modules.



Overview

- **How do I model LOVs in the current release?**
- **What is an LOV component?**
- **Can one module have many LOV components?**
- **Can one item have many LOVs associated with it?**
- **How do I restrict data in an LOV?**
- **Can I make an LOV reusable?**
- **How do I define LOVs?**

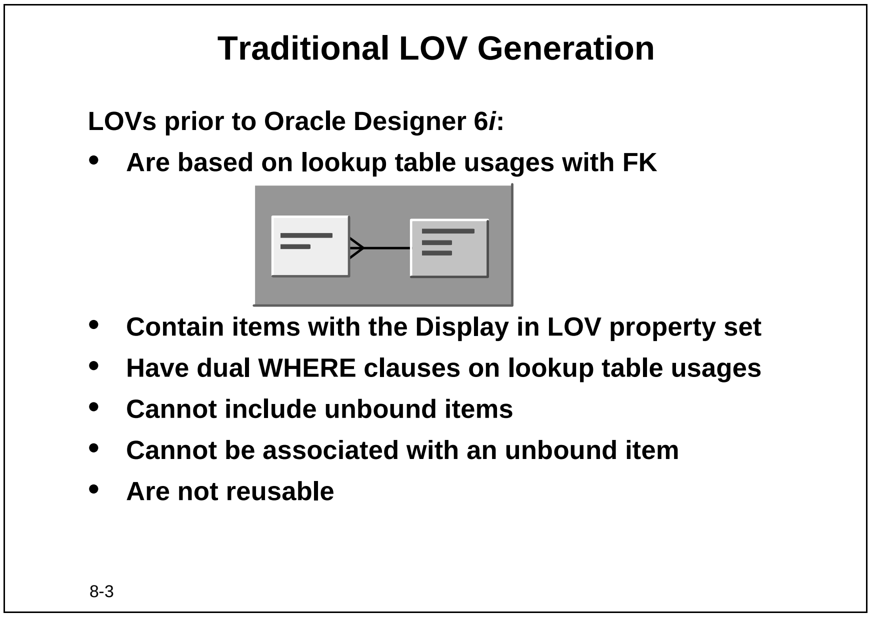
8-2

Objectives

At the end of this lesson, you should be able to:

- Create a List of Values component using the default utility
- Create LOVs manually and refine them
- Explain the difference between specific and reusable LOVs
- Associate LOVs with multiple items
- Develop queries to restrict the rows used by an LOV
- Attach an LOV to both bound and unbound items
- Display an LOV as a poplist
- Use a domain table to populate an LOV
- Construct a Subquery for an LOV
- Construct a Single Row Summary item for an LOV

Traditional LOV Generation



Traditional LOV Generation

LOVs prior to Oracle Designer 6*i*:

- **Are based on lookup table usages with FK**
- **Contain items with the Display in LOV property set**
- **Have dual WHERE clauses on lookup table usages**
- **Cannot include unbound items**
- **Cannot be associated with an unbound item**
- **Are not reusable**

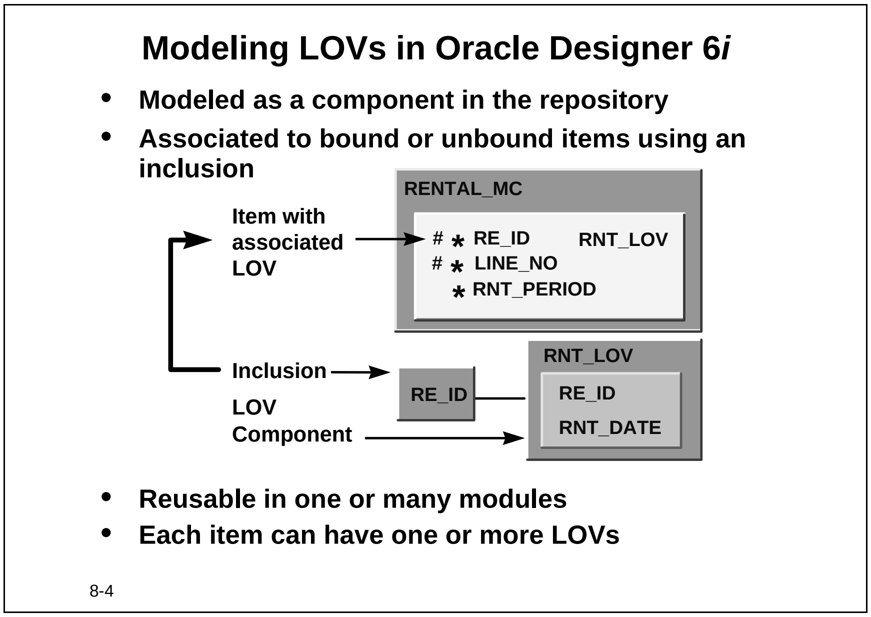
8-3

In releases of Designer prior to Oracle Designer 6*i* there are several limitations to generating lists of values. Below are some of the limitations:

- An LOV cannot contain unbound items.
- You cannot define an LOV on an unbound item.
- The Foreign Key to a lookup table usage is required.
- The use of Lookup table usages in the creation of LOVs can be confusing. You can use the lookup table usages both to define items that display in the form and to specify items for display in the LOV. To do this you need to set different properties on the items. To display items on the LOV but not on the form, include the items in the lookup table usage and set the Display in LOV property of the items to Yes and ensure the Display property is set to No.
- To restrict the values that appear in the list of values, you need to set a combination of settings in the WHERE Clause of Query and the Lookup Validation WHERE Clause of the lookup table usage.

Modeling LOVs in Oracle Designer 6*i*

In Designer 6*i*, the way you implement LOVs has changed.



Modeling LOVs in Oracle Designer 6*i*

- **Modeled as a component in the repository**
- **Associated to bound or unbound items using an inclusion**

RENTAL_MC

Item with associated LOV

* RE_ID RNT_LOV

# * LINE_NO

* RNT_PERIOD

Inclusion

RE_ID

LOV Component

RNT_LOV

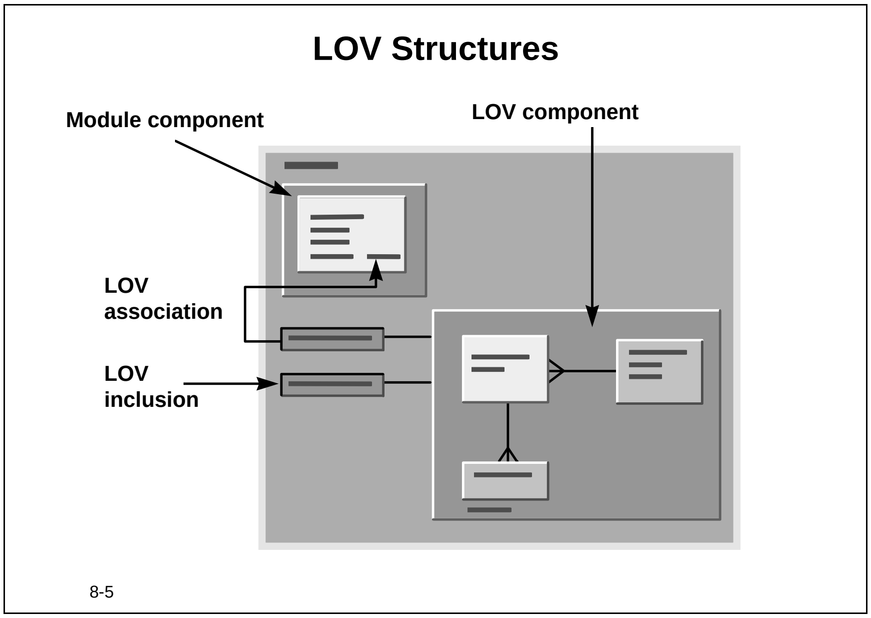
RE_ID

RNT_DATE

- **Reusable in one or many modules**
- **Each item can have one or more LOVs**

8-4

- You can create an LOV as an object without relying on Foreign Key and lookup table usages being present in a module.
- You can include unbound items in LOVs.
- You can associate LOVs with unbound items.
- You can restrict values using WHERE clauses in LOV itself.



LOV Structures

Module component

LOV component

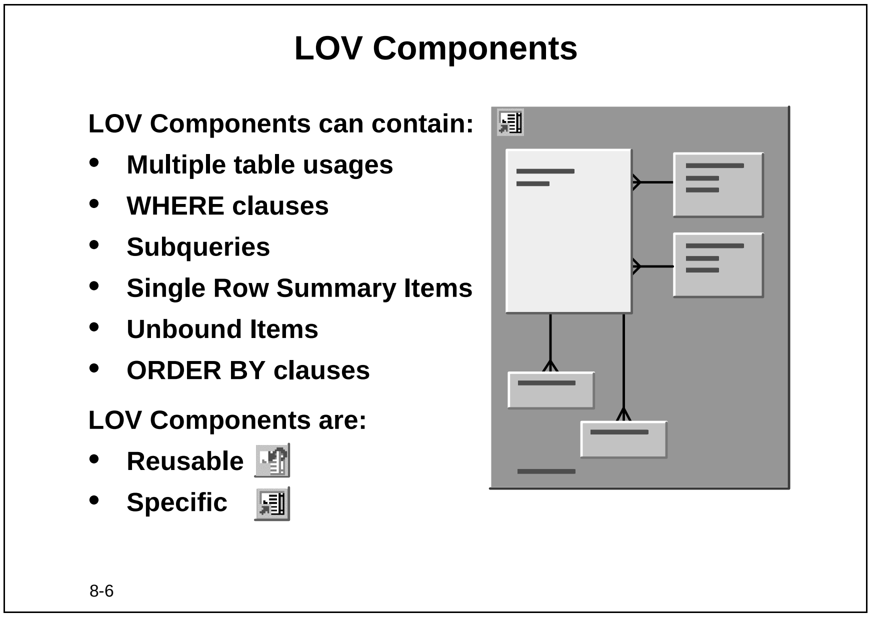
LOV association

LOV inclusion

8-5

What is an LOV Component?

There is a new object called an LOV component that you can use to explicitly model an LOV that is independent from a module component lookup table usage. An LOV component is based on one table and any associated lookup table usages required to form part of the LOV.



LOV Components

LOV Components can contain:

- **Multiple table usages**
- **WHERE clauses**
- **Subqueries**
- **Single Row Summary Items**
- **Unbound Items**
- **ORDER BY clauses**

LOV Components are:

- **Reusable**
- **Specific**

8-6

When creating an LOV component, you can:

- Create the LOV component based on one table and any associated lookup table usages.
- Use any user-created domain table as the basis for the LOV component.
- Incorporate unbound items in the LOV. This would allow you to define SQL expressions in the LOV.
- Define WHERE clauses for any or all of the table usages in the LOV component. Each clause you add is used in one of two ways:
  - When the form is in query mode
  - When the LOV is being used for validation
- Define Subqueries and SQL Aggregate Summary Item—SRSA as part of the LOV component.

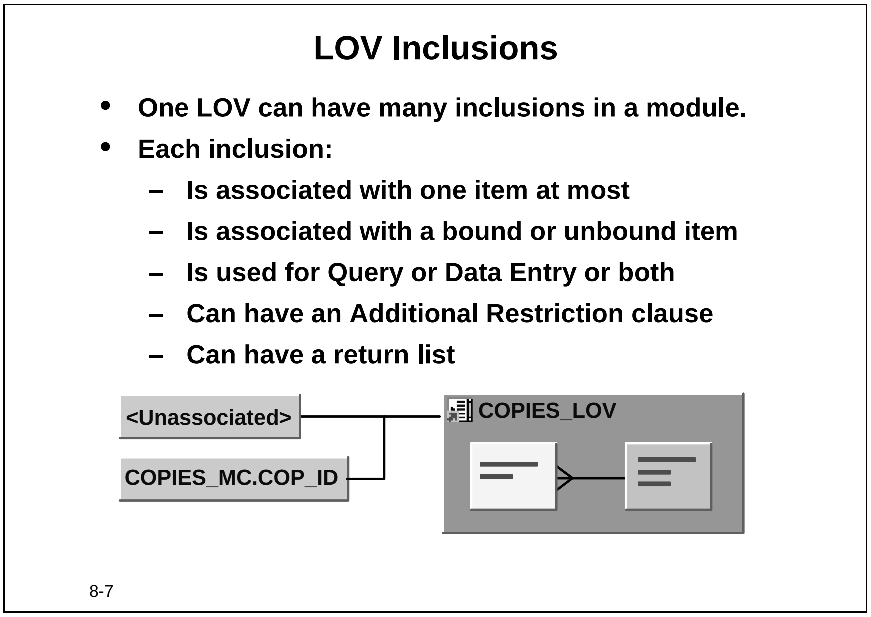
Types of LOV Component

There are two types of LOV component:

A Reusable LOV component	This is modeled independently from a module and you can use it in many modules
A Specific LOV component	This is created within a module where you can use it zero or more times. Note that this differs from a specific module component which is only used once within the module where it is created.

You can change the type of an LOV component from reusable to specific.

LOV Inclusions and Associations



LOV Inclusions

- One LOV can have many inclusions in a module.
- Each inclusion:
  - Is associated with one item at most
  - Is associated with a bound or unbound item
  - Is used for Query or Data Entry or both
  - Can have an Additional Restriction clause
  - Can have a return list

<Unassociated>
COPIES_MC.COP_ID
COPIES_LOV

8-7

What is an LOV Inclusion?

An LOV inclusion describes one instance of the use of an LOV component in a module. One LOV can have many different inclusions in a module. For each LOV inclusion in a module an LOV is generated into the form. When a specific LOV component is created or a reusable LOV component is included in a module, an inclusion is automatically created. A reusable LOV component can have inclusions across different modules.

What is an LOV Association?

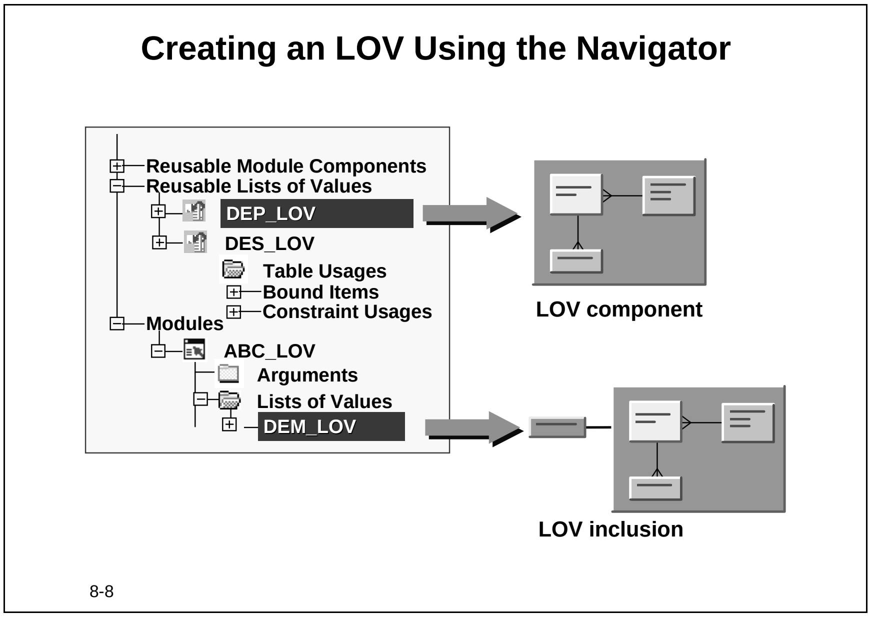
You can associate an LOV inclusion with an item in a module component. If you do not associate an inclusion with any item in the module component, the LOV is still generated into the form, but it will not be displayed unless you write application logic to call it.

Example of an LOV Inclusion and Association

The table that follows shows how the LOV component DEP_LOV is used within a module. The LOV shows three inclusions, meaning that three separate LOVs will be generated. (The Form Generator generates three unique LOV names by adding a numeric suffix to the name DEP_LOV.) Two of the inclusions have been associated with items and will be defined in the LOV property of the respective item in the generated form. The remaining LOV will be generated into the Form but will not be called unless you write application code to call it under specific circumstances.

LOV Component Inclusion	Associated Item
DEP_LOV	DEP_NO
DEP_LOV	DEP_NAME
DEP_LOV	

Creating an LOV



Creating an LOV Using the Navigator

8-8

New Elements in the Navigator

The change in implementing LOVs has resulted in two new elements in the navigator:

Reusable List of Values	This itemizes LOV components available within the application system. You can also drag a reusable list of values onto the worksurface to edit it.
List of Values	This lists each association of an LOV in a module and can contain both specific and reusable LOVs. It displays the LOV once for each association defined within the module. For easier reading, customize the display of the List of Values element to display the Item property as well as the LOV name.

Creating Reusable and Specific LOV Components

You can use the navigator to create reusable and specific LOV components.

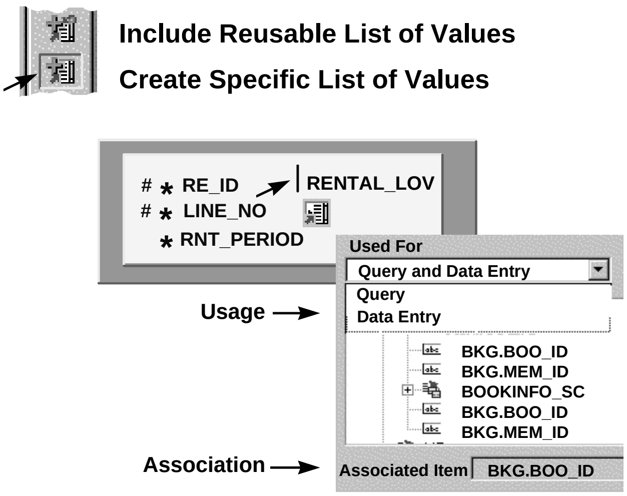
A List of Values Data Wizard steps you through the LOV creation process:

Reusable LOV component	Specify the name, title, template library object, size and other properties of the LOV component. Select the table usages and columns necessary for the LOV component and specify any WHERE clauses to be applied.
Specific LOV	You can only create a specific LOV in the context of a module. You need to create the LOV inclusion in addition to the LOV component. You can optionally specify the item you want the LOV inclusion to be associated with, when it is to be used (Query, Data Entry, or Both), any additional query conditions, and modify the return list mappings.

Adding an LOV to a Module

You can use the module data view diagram to create new LOVs for a module.

Including an LOV in a Module



Include Reusable List of Values

Create Specific List of Values

Usage

Association

8-9

Once you have created the LOV component you can:

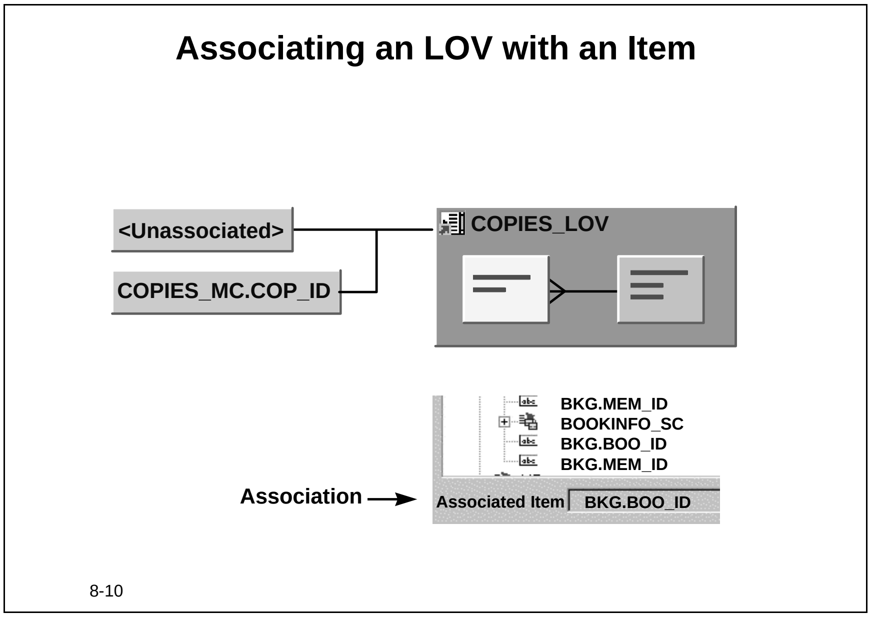
- Include the LOV in modules.
- Associate the LOV with module items.

Including an LOV Component in a Module

To create an LOV component inclusion in a module, two new buttons are available in the toolbar of the module diagram data view:

Button	Description
Include Reusable List of Values	Use this button to create an instance of an existing reusable or module-specific LOV in the module.
Create Specific List of Values	Use this button to create a new specific LOV component for the module.

Associating an LOV with an Item



Associating an LOV with an Item

8-10

If you move the cursor to a particular item in the module and click the item, you create an LOV inclusion associated with that item. The dialog that is displayed depends on whether you have specified one of the following:

Button	**Description**
Include Reusable List of Values	A dialog will allow you to select one LOV component from all the reusable LOVs available and all the specific LOVs you have already defined for the module. It will also allow you to specify the properties of the association, such as whether the generated LOV is to be used as Query, Data Entry, or Both, whether there are any additional query conditions, and to modify the return list mappings.
Create Specific List of Values	The List of Values Data Wizard will step you through the LOV component creation to select the table and column usages required. Additionally you can specify the association properties as described above.

Including an LOV with No Association

If you move the cursor onto the module, outside of any module component, you can create an LOV inclusion that is not associated with an item. There are differences here depending on which button you used:

Button	Description
Include Reusable List of Values	The dialog will allow you to select any reusable LOV component available to you that has not already been included in the module. This creates an unassociated inclusion.
Create Specific List of Values	The List of Values Data Wizard will step you through the LOV component creation but you will not be able to associate the LOV with an item at this stage.

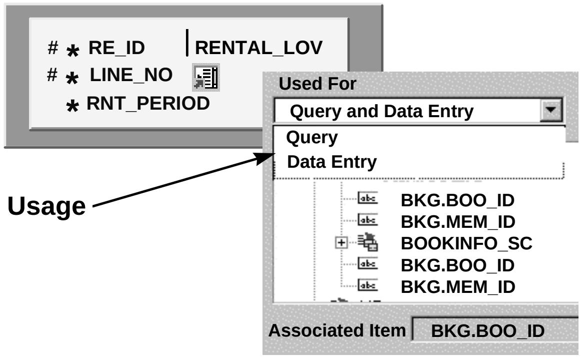
Editing an LOV Inclusion

You can edit an LOV inclusion to update any or all of the inclusion properties or to create or amend an association with an item.

Beware of inadvertently deleting a specific LOV component. If it has only one inclusion, and that inclusion is deleted from the repository, this also deletes the LOV component. To avoid inadvertently deleting an LOV, create an unassociated inclusion before deleting a single invalid association with an item or a sole unassociated inclusion.

Using the LOV for Data Entry or Data Query

Use an LOV for Data Entry or Date Query?



Used For: Query and Data Entry

Associated Item: BKG.BOO_ID

8-11

Is an Associated LOV used for Query or Data Entry or Both?

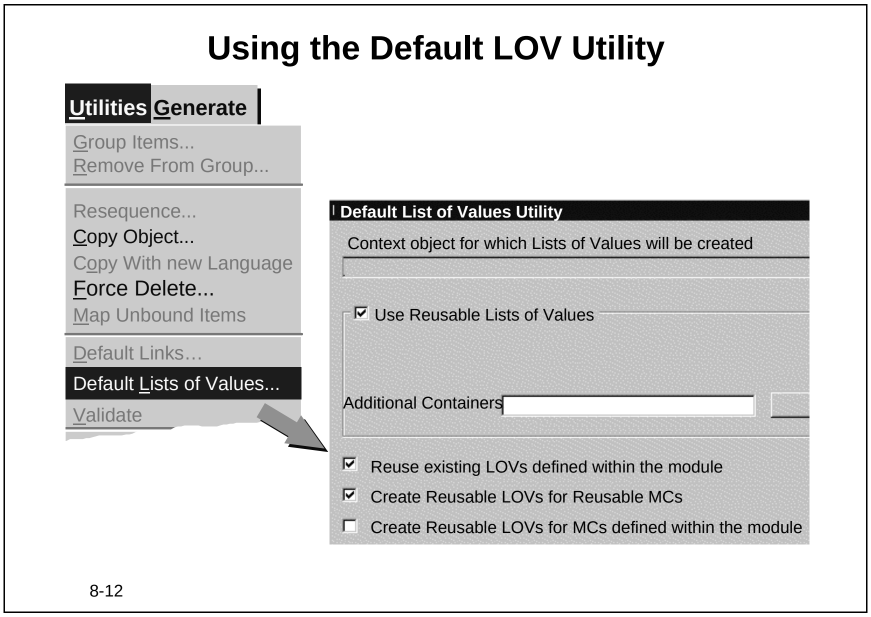
You can specify which operations an associated LOV is used for using the LOV association properties:

Query	When querying data
Data Entry	When entering or modifying data
Query and Data Entry	When querying or entering or modifying data

If you specify different LOVs for query and data entry the generator will create code to call the appropriate LOV. Alternatively you can write application logic to call a particular LOV under certain circumstances.

Default List of Values Utility

Use the utility to create an LOV component and associate it with an item.



Using the Default LOV Utility

Default List of Values Utility

Context object for which Lists of Values will be created

☑ Use Reusable Lists of Values

Additional Containers

☑ Reuse existing LOVs defined within the module

☑ Create Reusable LOVs for Reusable MCs

☐ Create Reusable LOVs for MCs defined within the module

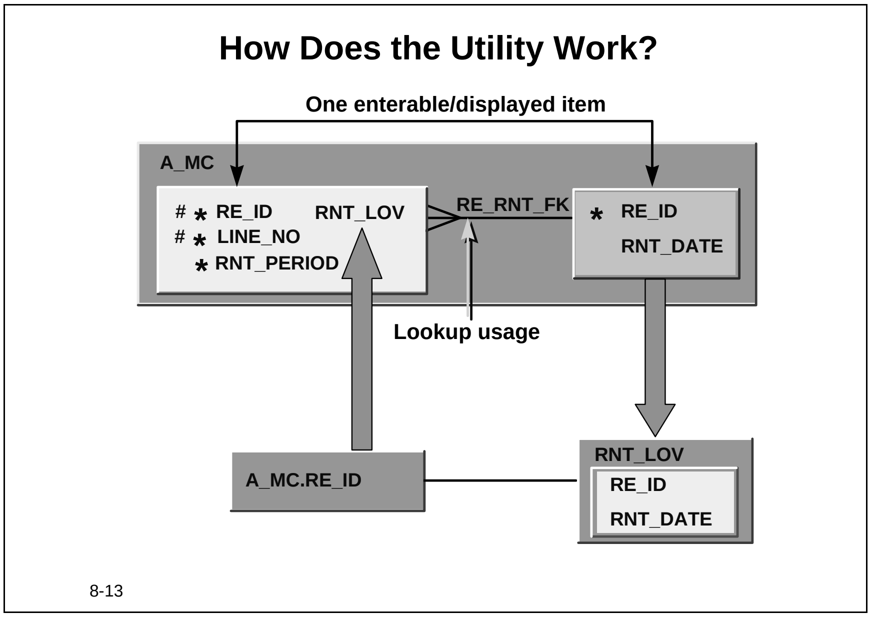
8-12

You can run the Default List of Values utility against a module, module component, sub-component, or table usage. Access the utility by selecting **Utilities —> Default List of Values.** Your selection is shown in the Context Object item of the utility dialog box. You can influence how the utility works against the context object by selecting combinations of the four check boxes.

Check box	**Behavior**
Use Reusable List of Values (1)	Creates associations from any suitable Reusable LOVs
Reuse existing LOVs defined within the module (2)	Creates associations from any suitable LOV defined within the module
Create Reusable LOVs for Reusable module components	Creates reusable LOV components and associates with suitable items for any reusable module components included in your context object
Create Reusable LOVs for module components defined within the module	Creates reusable LOV components and associates with suitable items for any specific module components included in your context object

You can select combinations of the boxes. If checkboxes marked as 1 and 2 (above) are selected, the utility will first check for any suitable LOV components defined within the module (2) and then check for any other reusable LOVs that have been defined but not yet included in the module (1).

Using the Default LOV Utility



How Does the Utility Work?

8-13

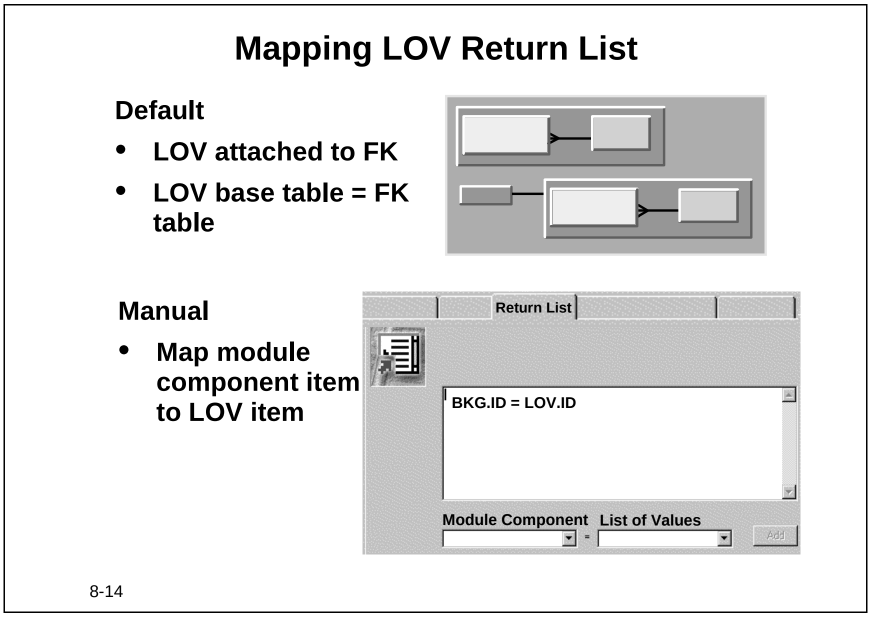
For the Default LOV utility to be successful the following must be correct:

- The module component must have a table lookup usage defined.
- One item in the table lookup usage, or the foreign key item in the base table, must be enterable and displayed.

The Default LOV utility creates an LOV component as detailed below:

- It is named using the naming convention <table alias>_LOV.
- It includes all the lookup table items in the module component.
- It is associated with the first enterable and displayed item in the module component that it finds. It achieves this by taking each item in the module component in its display sequence order, checking whether it is part of the LOV, and associating the LOV with the first enterable and displayed LOV item it finds.

Creating the Return List



Mapping LOV Return List

Default

- **LOV attached to FK**
- **LOV base table = FK table**

Manual

- **Map module component item to LOV item**

BKG.ID = LOV.ID

Return List
Module Component
List of Values
=
Add
8-14

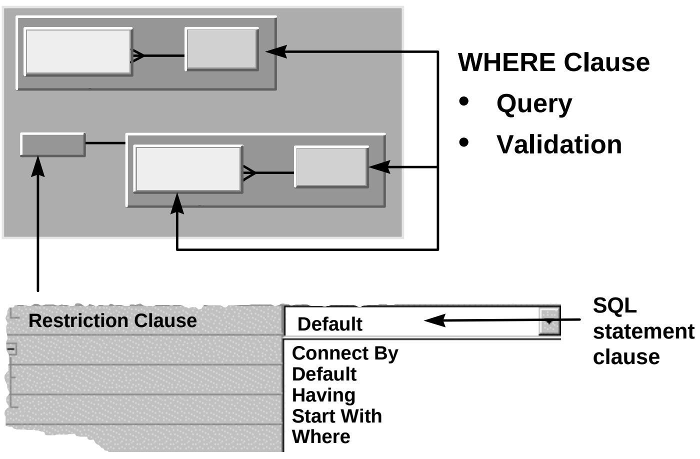
To create a return list, the generator attempts to map items in the LOV with corresponding items in the associated module component. If the LOV is attached to the foreign key item in the base table and the base table of the LOV component is a lookup table usage of that foreign key, the generator matches the items.

If the generator is unable to create a mapping, a warning is issued on generation. For example, the generator is not able to create a mapping when the domain table is attached to an unbound item.

You can manually create the return list through the LOV association properties as illustrated above. The property dialog view allows you to select a module component item and link it directly to an LOV item.

Refining an LOV

Restricting the Values in the LOV



8-15

Using a WHERE clause

You can manipulate the `WHERE` clause property of an LOV component to restrict the records displayed in the LOV under certain circumstances:

- A Query `WHERE` clause will form part of the LOV record group `WHERE` clause when the form is in enter-query mode.
- A Validation `WHERE` clause will form part of the LOV record group `WHERE` clause when the form is not in enter-query mode.

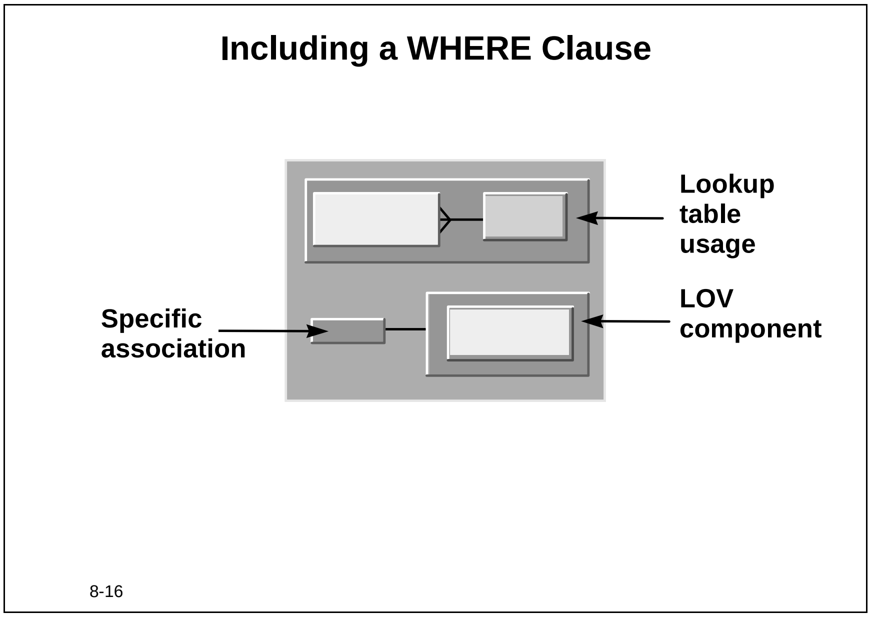
Using the Additional Restriction

You can set an additional restriction which is incorporated into the SQL statement defining the LOV usage to further manipulate the records in an LOV. The value of the Restriction Clause property determines how the clause is used in the SQL statement. The default additional restriction clause adds your SQL to the WHERE clause of the LOV SQL statement.

Restriction Clause Property Value	**Text of Additional Restriction**	**SQL clause in which the Restriction is placed**
`DEFAULT`	Contains word “Prior”	`CONNECT BY`
`DEFAULT`	References a summary item	`HAVING`
`DEFAULT`		`WHERE`

Restriction Clause Property Value	**Text of Additional Restriction**	**SQL clause in which the Restriction is placed**
`CONNECT BY`		`CONNECT BY`
`HAVING`		`HAVING`
`START WITH`		`START WITH`
`WHERE`		`WHERE`

Building up the WHERE Clause



Including a WHERE Clause

Lookup table usage

LOV component

Specific association

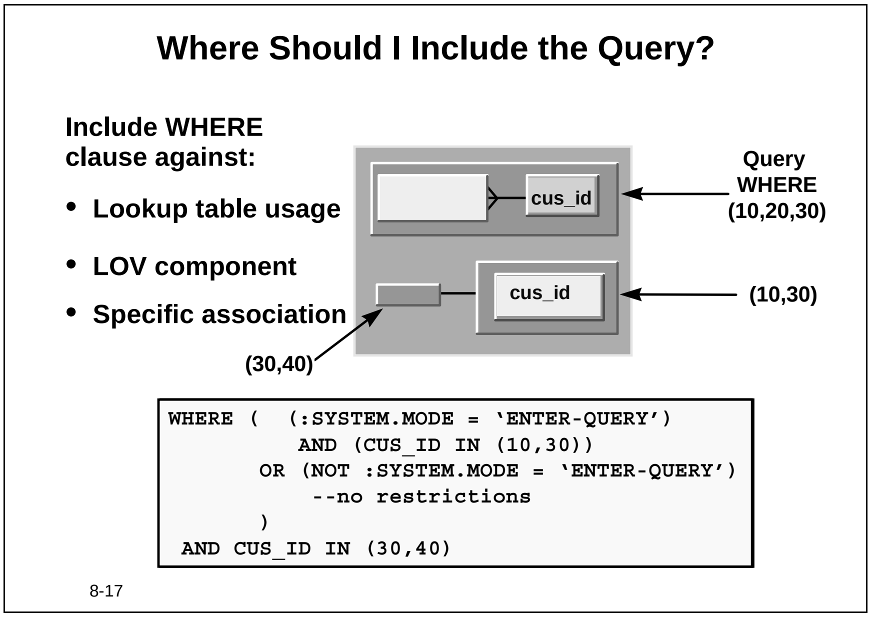
8-16

You can add WHERE clause restrictions to your LOV in three places:

Module component lookup table usage	The generator incorporates WHERE clause restrictions into the WHERE clause of the LOV, only if the lookup table usage of the module component is the same as the base table of the LOV. If you use the Additional Restriction property of the LOV association, the generator always ignores the WHERE clause of the lookup table usage.
LOV component base table or lookup table usage	If you add WHERE clause restrictions here, they are incorporated into every LOV association created from this LOV component.
Additional Restriction clause of LOV association with item	If you add a WHERE clause restriction here, it only applies to the use of the LOV against the individual item association.

For the module component and the LOV component, you can further refine the WHERE clause condition for use in enter-query or nonenter-query mode. The Additional Restriction works slightly differently. It is used in both enter-query and when the form is not in enter-query mode.

Placing the WHERE Clause



Where Should I Include the Query?

Include WHERE clause against:

- **Lookup table usage**
- **LOV component**
- **Specific association**

```
WHERE (   (:SYSTEM.MODE = 'ENTER-QUERY')
          AND (CUS_ID IN (10,30))
       OR (NOT :SYSTEM.MODE = 'ENTER-QUERY')
           --no restrictions
       )
 AND CUS_ID IN (30,40)
```

8-17

If you wish to restrict your LOV records ask yourself a number of questions:

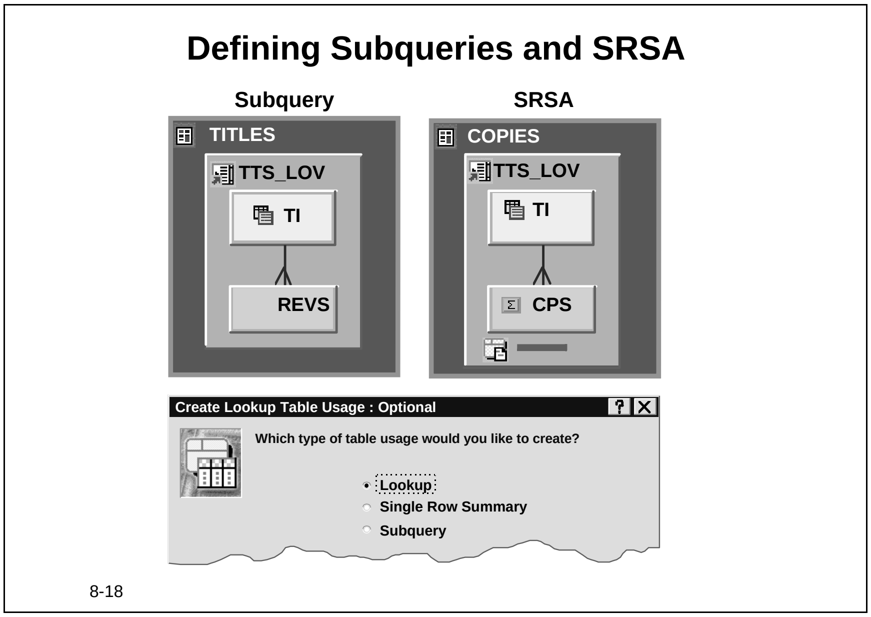
- Am I using the additional restriction clause (and therefore my lookup table usage `WHERE` clause definition will not be in my LOV)?
- Will I need to use my LOV for a number of different items where I need different records?
- Do I want to use the additional restriction clause for anything other than setting the `WHERE` clause?
- If I include a reusable LOV in my module, which has been designed to follow a company standard, will I subvert that standard by adding any additional restriction through my item association?

MC lookup table usage WHERE clause	**LOV Query WHERE clause**	**Additional Restriction WHERE Clause**	**LOV WHERE clause**

CUS_ID IN (10,20,30)			`WHERE (`
			`(:SYSTEM.MODE = 'ENTER-QUERY')`
			`AND (CUS_ID IN (10,20,30))`
			`OR (NOT :SYSTEM.MODE = 'ENTER QUERY')`
			`--no restrictions)`

MC lookup table usage WHERE clause	LOV Query WHERE clause	Additional Restriction WHERE Clause	LOV WHERE clause
CUS_ID IN (10,20,30)	CUS_ID IN (10,30)		WHERE ((:SYSTEM.MODE = ‘ENTER-QUERY’) AND (CUS_ID IN (10,20,30) AND (CUS_ID IN (10,30)) OR (NOT :SYSTEM.MODE = ‘ENTER QUERY’) --no restrictions)
CUS_ID IN (10,20,30)	CUS_ID IN (10,30)	CUS_ID IN (30,40)	WHERE ((:SYSTEM.MODE = ‘ENTER-QUERY’) AND (CUS_ID IN (10,30)) OR (NOT :SYSTEM.MODE = ‘ENTER QUERY’) --no restrictions) AND CUS_ID IN (30,40)

Additional Table Usage Types for LOV Components



Defining Subqueries and SRSA

Create Lookup Table Usage : Optional

Which type of table usage would you like to create?

● Lookup
○ Single Row Summary
○ Subquery

8-18

Table Usage Types

You can create two additional table usage types in your LOV component:

- A subquery enables you to add restrictions to the main query.
- A SQL aggregate summary item (SRSA) facilitates the use of group functions.

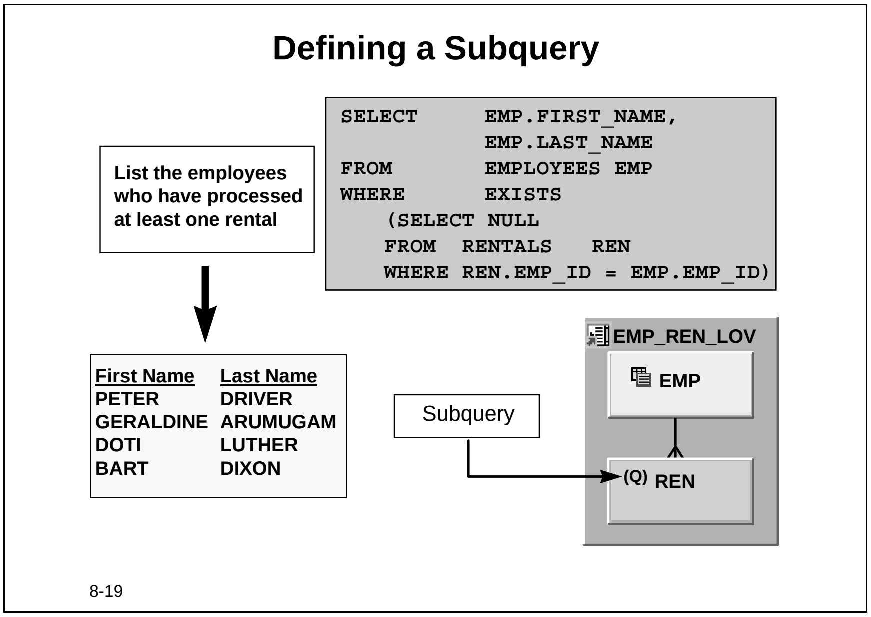
Table usages of type subquery and SRSA were introduced to the Report Generator in Oracle Designer Release 2. These table usage types give you more direct control over what you generate in your LOV. You can also now use them in Web PL/SQL modules.

Where to Define Subquery and SRSA Table Usages

When defining either type of table usage, include the usage in the same LOV component as the table usage to which it is linked.

You can create a separate condition for query and validation usages of the LOV through one subquery table usage.

Defining a Subquery



Defining a Subquery

List the employees who have processed at least one rental

```
SELECT      EMP.FIRST_NAME,
            EMP.LAST_NAME
FROM        EMPLOYEES EMP
WHERE       EXISTS
    (SELECT NULL
    FROM  RENTALS   REN
    WHERE REN.EMP_ID = EMP.EMP_ID)
```

First Name	Last Name
PETER	DRIVER
GERALDINE	ARUMUGAM
DOTI	LUTHER
BART	DIXON

8-19

A subquery table usage must conform to all of the following conditions:

- It must be linked to another table usage in the same LOV component. It can be linked to any other table usage, including another subquery.
- It must be at the “crowsfoot” end of the link.
- It must not have any displayed, bound items.

How to Define a Subquery

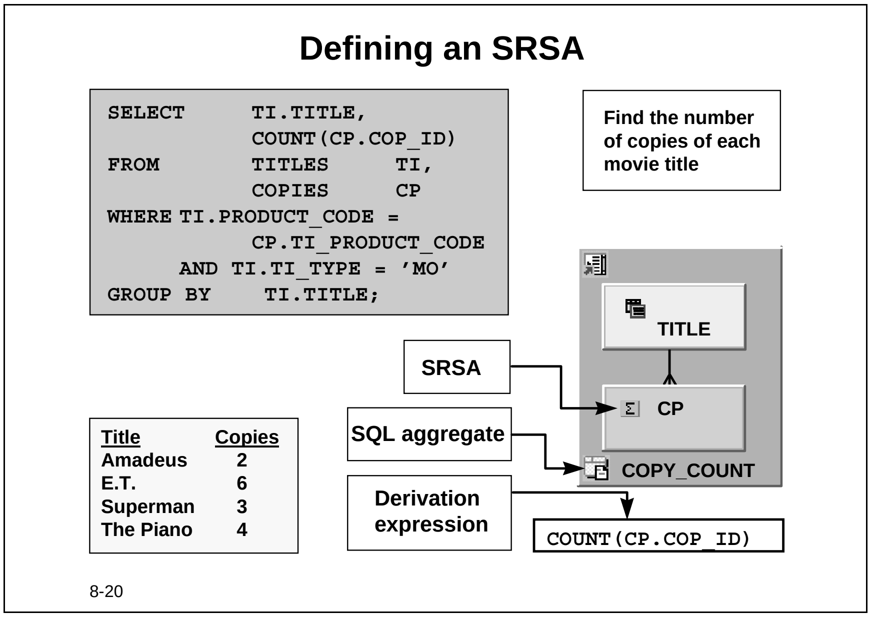
The example above shows how to define a subquery to refine the rows returned in the base table of the LOV component. You can use the Create Table Usage property dialog to define the subquery:

1. Open the property dialog for the LOV component.
2. Select the Subquery check box.
3. Select the table usage for the subquery.
4. Select the type of condition (query or validation).
5. Add any additional `WHERE` clause.

Reversing the Logic

If you want to reverse the logic of the subquery and generate a `NOT EXISTS` operator, then set the Sub Query: Not Exists property on the subquery table usage to Yes.

Defining an SRSA



Defining an SRSA

```
SELECT      TI.TITLE,
            COUNT(CP.COP_ID)
FROM        TITLES      TI,
            COPIES      CP
WHERE TI.PRODUCT_CODE =
            CP.TI_PRODUCT_CODE
      AND TI.TI_TYPE = 'MO'
GROUP BY     TI.TITLE;
```

Find the number of copies of each movie title

Title	Copies
Amadeus	2
E.T.	6
Superman	3
The Piano	4

8-20

If you want to summarize information from a table other than the base or lookup tables in an LOV, then you must define an SRSA table usage for it.

How to Define an SRSA

The example above shows how to define a group function that summarizes a column in the base table of the LOV component. You can use the Create Table Usage property dialog to define the SRSA:

1. Open the property dialog for the LOV component.
2. Select the Single Row Summary check box.
3. Select the table usage containing the item to be summarized.
4. Select the type of condition (query or validation).
5. Add any additional `WHERE` clause.
6. Enter the name of the unbound item of type SQL aggregate you want.
7. Add the group function and the column required.

When you use an SRSA, the query returns a summarized value for each row in the base table and displays it in the LOV.

Allowable Links for SRSAs

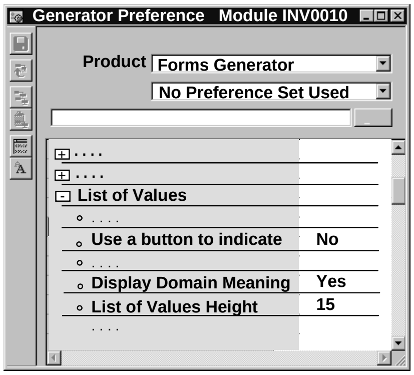
You can link an SRSA to a base, lookup, or other SRSA table usage, but not to a subquery.

A Few LOV Preferences

There are a number of preferences which can be set to influence the generation of an LOV.

Other LOV Preferences

- **LOV Display**
- **Button**
- **Positioning**



Product: Forms Generator

No Preference Set Used

⊞ . . . .

⊞ . . . .

⊟ List of Values

- . . . .
- Use a button to indicate: No
- . . . .
- Display Domain Meaning: Yes
- List of Values Height: 15
- . . . .

8-21

LOV Display

You can control the display of a generated LOV through a number of preferences specifying such things as the width and height of the LOV.

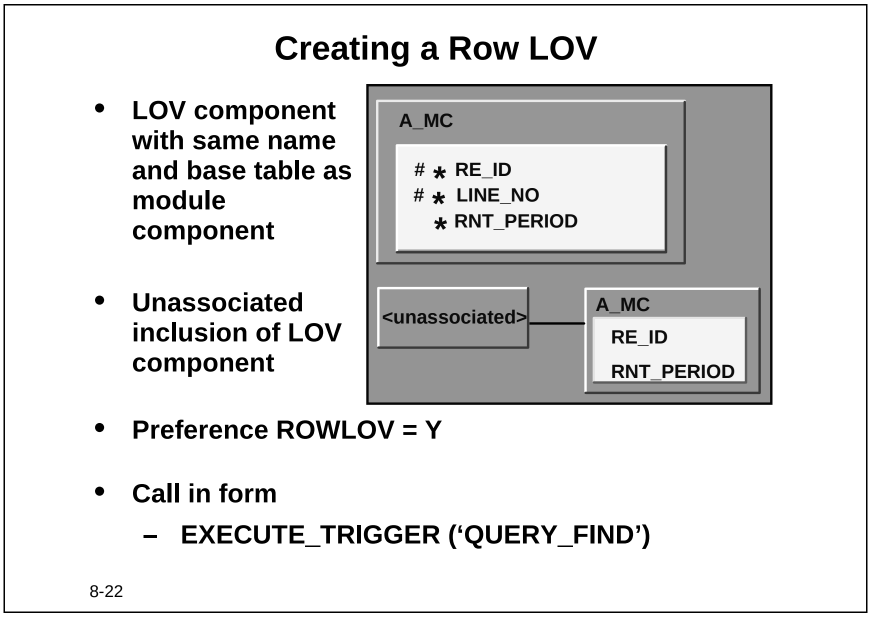
Button

You can choose to generate a button next to an item which has an LOV attached. Code is generated in the button to call the LOV when pressed.

Positioning

You can control such things as the vertical and horizontal position of a displayed LOV.

Implementing ROW LOVs with LOV Components



Creating a Row LOV

- **LOV component with same name and base table as module component**
- **Unassociated inclusion of LOV component**
- **Preference ROWLOV = Y**
- **Call in form**
  - **EXECUTE_TRIGGER (‘QUERY_FIND’)**

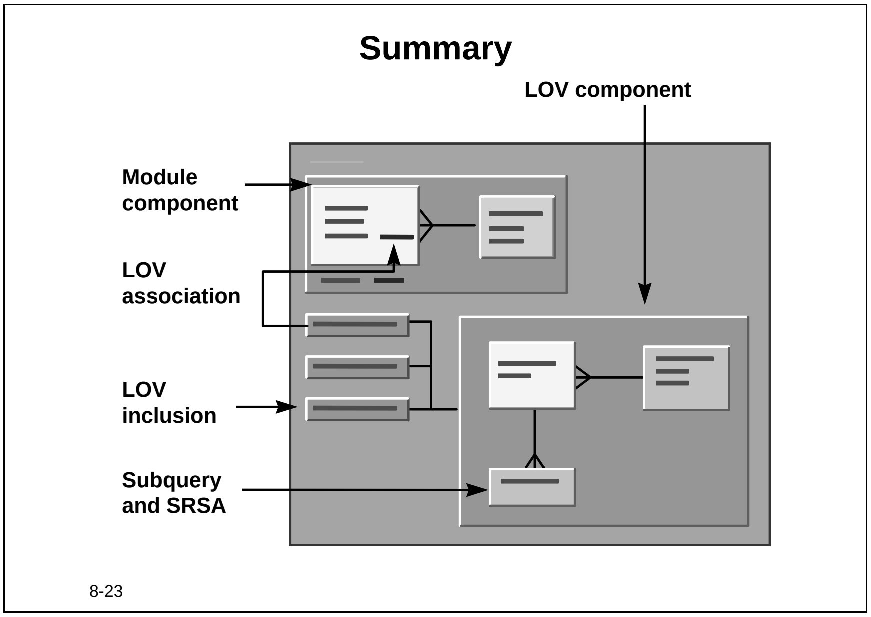
A_MC
# * RE_ID
# * LINE_NO
* RNT_PERIOD
<unassociated>
A_MC
RE_ID
RNT_PERIOD

8-22

For the Form Generator to generate a row LOV correctly a number of conditions must be met:

- An LOV component must exist, with the same name as the module component to which it is to be attached. The base table of the module component and the LOV component must be the same.
- An unassociated inclusion must exist against this LOV component. You can use the LOV component as a ‘normal’ LOV against additional items in the module, by creating other associated inclusions against it.
- The ROWLOV preference must be set to Yes.
- The Row LOV is called by the firing of the QUERY_FIND trigger created by the Form Generator. Provide the user with a way to fire the trigger. This can be a button on the form or a menu item.

Summary



Summary

8-23

Specific and Reusable LOV Components

An LOV component can be specific or reusable. It can incorporate columns from base and lookup tables and user-created domain tables.

Inclusions

Each use of an LOV in a module is an inclusion. An inclusion is associated with, at most, one bound or unbound item in the module. An unassociated inclusion needs application logic to invoke it in the generated form.

Refining Table WHERE Clauses

Each association can further refine the table `WHERE` clauses through the use of the Additional Restriction.

You can define subqueries and SRSAs for an LOV component.

Practice 8 – 1: Creating LOVs

Goal

In this practice you create and use LOVs to enhance the functionality of a form.

Scenario

The developers have created a number of modules with no LOV inclusions. Availability of LOVs will enhance the generated form’s usability.

They have decided on the following actions:

- Make use of the default LOV utility
- Incorporate existing LOVs
- Use query conditions to filter LOVs
- Modify LOV associations
- Include an LOV link using an action item
- Include an LOV link using a pop-up list

Your Assignment

Use the information below as your starting point:

Tool	Design Editor
WorkArea	ORA<nn>_PWA_01
Application System	NF_HOLLYWOOD
Module	HOL0080F

where <nn> is the number assigned to you by your instructor.

Making use of the default LOV utility

1 Use the Design Editor navigator to answer the following:

a Are there any Reusable Lists of Values defined for the application?

b Are there any LOV components defined within the module HOL0080F?

2 Invoke the Default List Of Values Utility with the MEMBERS_MC module component selected.

a How can you ensure that any pre-existing LOV components in the application are used by the Default List Of Values Utility?

b How can you influence whether any new LOV components created by the utility are reusable across the application?

c Can you run the default LOV utility for module component MEMBERS_MC?

3 Select the BOOKINGS_MC module component and run the default LOV utility, using the default settings.

a What is the name of the new LOV?

b Which item is the LOV associated with? Why?

4 Generate and run the form.

a What LOVs are available on the form?

Incorporate Existing Lists of Values

5 Use the Design Editor navigator to include a list of values for the IDENTIFIER bound item in the MEMBERS_MC module component. Ensure the LOV is for Query mode only.

6 Use the module diagram to include the reusable LOV EMPLOYEE_LOV. Ensure you associate it with the BOOKED_BY item in the BOOKINGS_MC. The LOV is for data entry and data query.

7 Edit the LOV inclusion for the MEMBERS_LOV associated with the IDENTIFIER bound item in the MEMBERS_MC module component.

8 Create a return list, mapping the IDENTIFIER, MT_CODE and VALID_FLAG items in the Module Component with the items of the same name in the List of Values.

9 Generate and run the form.

a Execute a query on the Members block. Can you invoke the LOV on the Member Id item?

b Invoke enter-query mode on the Members block. Can you use the LOV on the Member Id item?

c Select a member from the list. What items are populated in the form?

d Enter a new Member. Is the LOV available?

e Create a new booking for an existing member. Can you use the LOV available on Product Code and Booked By to help with the insert?

Use query conditions to filter LOVs

10 Include a Query WHERE clause to the TITLES lookup table usage.

```
TI_TYPE = ‘MO’
```

a What effect does having a Query WHERE clause on a lookup table usage have on LOV components when they are based on the same table as the lookup table usages? (In this example, the TITLES table)

b What effect will entering the same Query WHERE clause against the TITLES LOV component?

11 Generate and run the module.

- **a** Execute a query. Scroll through the members records, are any bookings of type GA (Game) returned in the Booking details?
- **b** Invoke enter-query mode in the Bookings block and invoke the List of Values on Product Titles. Why are these Titles returned?

Including an LOV link using an action item

12 Include the CANCELLED_BY item in the BOOKINGS_MC.

13 Create a button and a WHEN_BUTTON_PRESSED event to invoke the EMPLOYEE_LOV. Use the following piece of code to assist you:

```
DECLARE
    LOV boolean;
BEGIN
    LOV:= show_lov(’EMPLOYEE_LOV’);
END;
```

14 Generate and run the module.

- **a** Cancel an existing booking. Use the button you created to assist you in finding a employee to do the cancellation.

Including an LOV link using a Pop-up list

15 Create an LOV based on the CUSTOMERS table. Include the CUS_ID and LAST_NAME items. (The order is important, because the CUS_ID is the returned value.)

16 Associate the LOV with the CUS_ID item on the MEMBERS_MC.

17 Set the display type of CUS_ID to Poplist and the display width to 30.

18 Generate and run the module. The CUS_ID LOV should display in the form of a poplist.

Practice 8 – 2: Create an SRSA

Goal

In this practice you create an SRSA for an LOV

Scenario

The developers have created a module with a number of LOV inclusions. The users would like to see the number of bookings made by the employees, displayed in the EMPLOYEE_LOV.

To achieve this the developers need to:

- Create a Single Row Summary Item

Your Assignment

Use the information below as your starting point:

Tool	Design Editor
WorkArea	ORA<nn>_PWA_01
Application System	NF_HOLLYWOOD
Module	HOL0080F
Reusable LOV	EMPLOYEE_LOV

where <nn> is the number assigned to you by your instructor.

Creating a SQL aggregate summary item for an LOV

1. Create a diagram for the reusable list of values EMPLOYEE_LOV.
2. Right mouse click on the EMPLOYEE_LOV and invoke the Create Table Usage wizard.
3. Set the following properties:

Property	**Value**
Table usage type	Single Row Summary
Table usage	BOOKINGS linked via BOO_EMP_FK
Item Name	NO_OF_BOOKS
Function	COUNT
Table	BOOKINGS
Column	BOO_ID

   a. What has been added to the EMPLOYEE_LOV?

4. Invoke the property dialog for the item NO_OF_BOOKS and set the following:

Property	**Value**
Prompt	No of Bookings
Datatype	Text
Width	3
Height	1

5. Create a diagram for the module HOL0080F.
   a. Have these modifications been transferred to the LOV in the HOL0080F module?
6. Generate the module.
7. Invoke the LOV for item BOOKED BY and examine the new item.

Practice 8 – Hints

To Do This Task	Follow These Steps
Finding Reusable Lists of Values	**1** Expand the Reusable Lists of Values node in the Design Editor navigator under the modules tab
Invoking the Default List Of Values Utility	**1** Select a module or module component. **2** Invoke the utility through the menu using **Utilities -> Default List of Values...**
Including a list of values using the navigator	**1** Expand the Navigator to find the item you want to associate the LOV with **2** Use the short cut menu to: – Include List of Values to use an existing LOV – Create List of Values to create a new LOV.
Setting Query mode for the LOV using the property dialog	**1** Double click on the associated item and invoke the property dialog. **2** Select the **Usage** tab and update the Used for property.
Including an LOV using a module diagram	**1** Click on the **Include Reusable List of Values** button **2** Click on the item you want to associate the LOV with, to invoke a selection of available LOVs **3** Select the required LOV and click OK
Include a Query WHERE clause on an LOV	**1** Invoke the property palette for the Table usage for the LOV. **2** Include the WHERE clause condition.
Editing the return list	**1** Double click on the associated item to invoke the property dialog. **2** Select the **Return List** tab. **3** Match the Module Component items with the LOV Items to create the return list values.
Setting the display type of an item to Poplist.	**1** Double click on the item that has the LOV associated with it, to invoke a property palette or property dialog. **2** Set the display type to Pop List

To Do This Task	Follow These Steps
Creating a subquery	**1** Invoke the Create Table Usage property dialog by using the short cut menu with the LOV component selected in the diagram. **2** Select the Subquery radio group. **3** Select the table usage for the subquery. **4** Select the type of condition (query or validation). **5** Add any additional `WHERE` clause.
Creating a single row summary	**1** Invoke the Create Table Usage property dialog by using the short cut menu with the LOV component selected in the diagram. **2** Select the single row summary radio group. **3** Select the table usage for the summary. **4** Select the type of condition (query or validation). **5** Add the summary items.