Oracle Forms Developer 10*g*: Build Internet Applications

Electronic Presentation

D17251GC10
Edition 1.0
June 2004
D39562

Author

Pam Gamer

Technical Contributors and Reviewers

Alena Bugarova
Purjanti Chang
Laurent Dereac
Punita Handa
Mark Pare
Jasmin Robayo
Bryan Roberts
Divya Sandeep
Raza Siddiqui
John Soltani
Lex van der Werff

Editors

Nishima Sachdeva
Elizabeth Treacy

Publisher

Giri Venugopal

1

Introduction to Oracle Forms Developer and Oracle Forms Services

Objectives

After completing this lesson, you should be able to do the following:

- Define grid computing
- Explain how Oracle 10*g* products implement grid computing
- Describe the components of Oracle Application Server 10*g* and Oracle Developer Suite 10*g*
- Describe the features and benefits of Oracle Forms Services and Oracle Forms Developer
- Describe the architecture of Oracle Forms Services
- Describe the course application

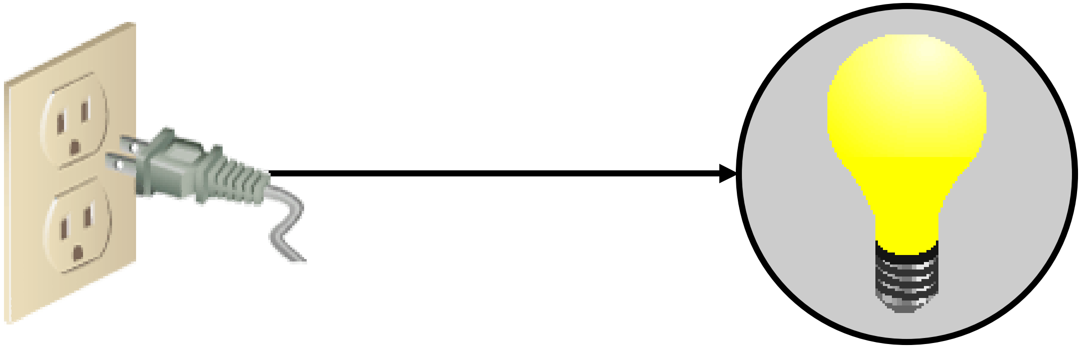
Internet Computing Solutions

Application Type and Audience	Product Approach	Oracle Products
Enterprise applications, Business developers	*Repository-based modeling & generation, Declarative*	*Oracle Designer, Oracle Forms Developer, & Oracle Forms Services*
Java components, Component developers	Two-way coding, Java and JavaBeans	Oracle JDeveloper Oracle Application Server 10*g*
Self-service applications & content management, Web site developers	Browser-based, Dynamic HTML	Oracle Portal Oracle Database Server
Reporting and analytical applications, MIS & business users	Dynamic Web reporting, Drill, Analyzing, Forecasting	Oracle Reports Developer, Oracle Reports Services, Oracle Discoverer, & Oracle Express

Plugging into the Grid

Grid computing is:

- Software infrastructure that uses low-cost servers and modular storage to:
  - Balance workloads
  - Provide capacity on demand
- Made possible by innovations in hardware
- Powered by software

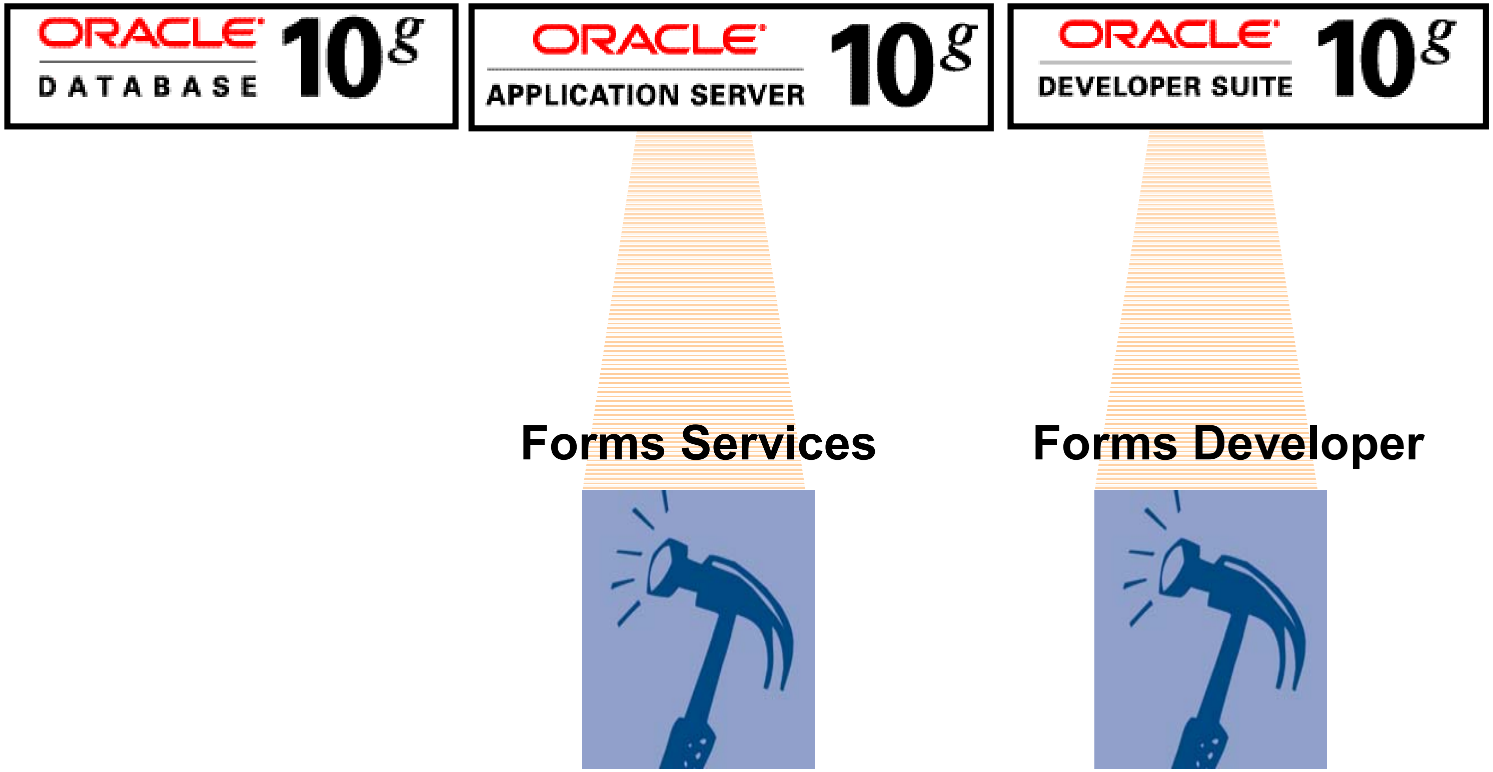


Oracle Enterprise Grid Computing

Oracle's grid infrastructure products:

- Oracle Database 10*g*
- Oracle Application Server 10*g*
- Oracle Enterprise Manager 10*g* Grid Control

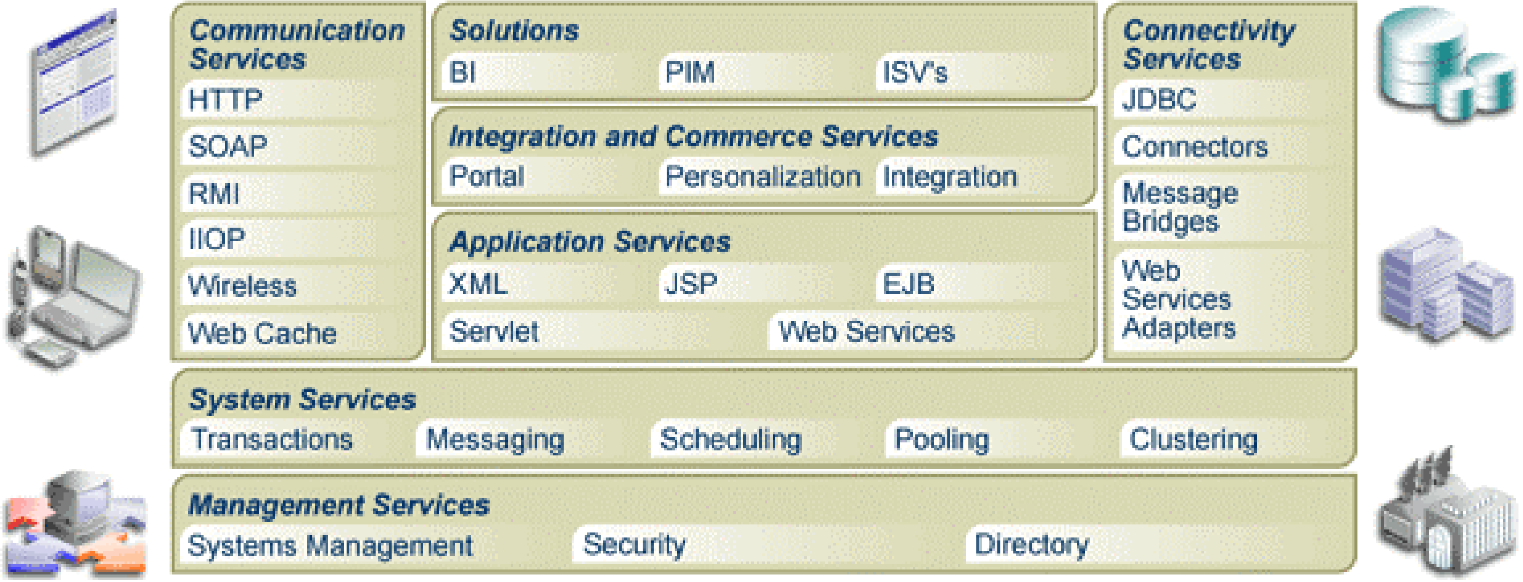
Oracle 10*g* Products and Forms Development



Forms Services

Forms Developer

Oracle Application Server 10*g* Architecture



Communication Services
HTTP
SOAP
RMI
IIOP
Wireless
Web Cache

Solutions
BI
PIM
ISV's

Integration and Commerce Services
Portal
Personalization
Integration

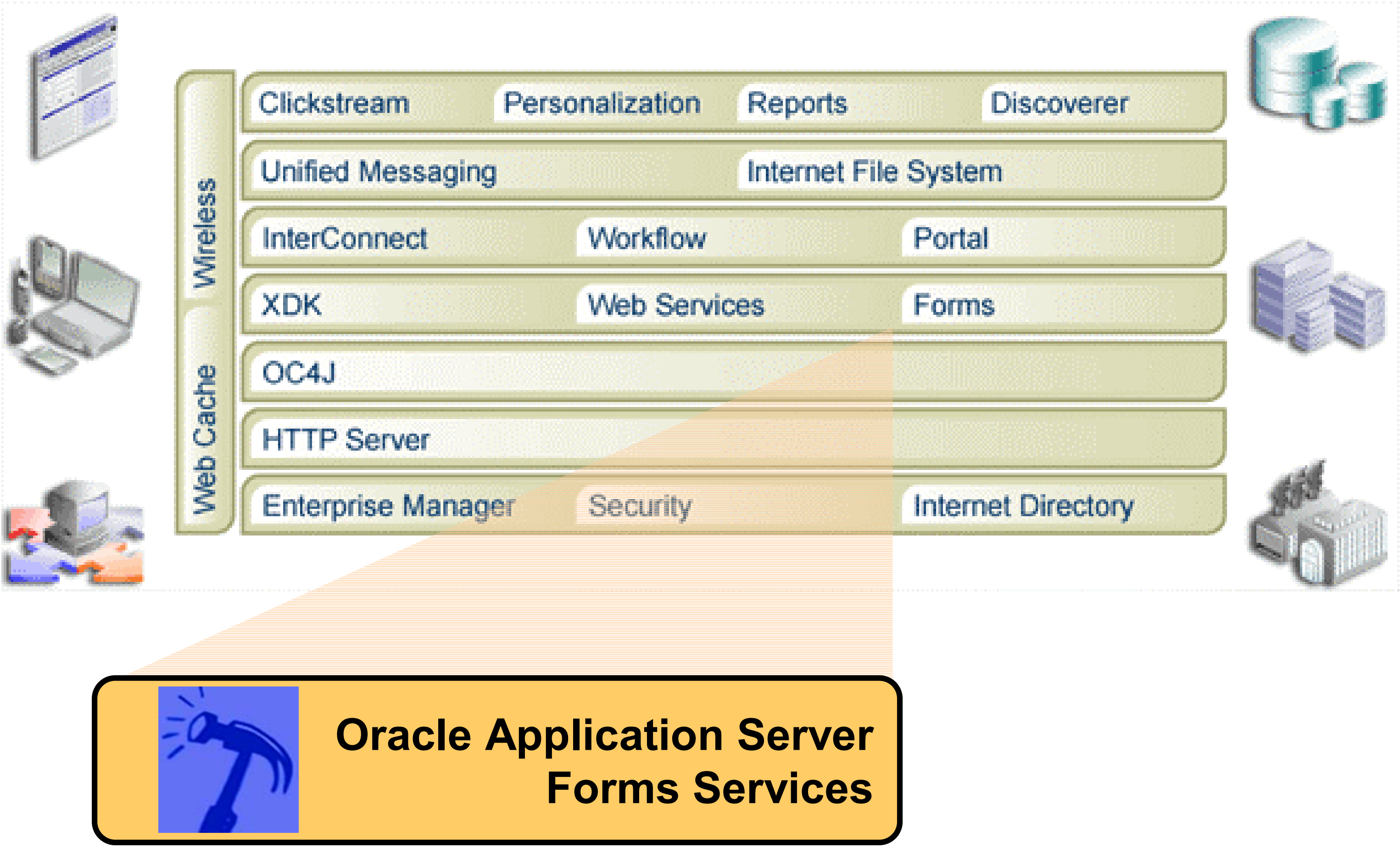
Application Services
XML
JSP
EJB
Servlet
Web Services

Connectivity Services
JDBC
Connectors
Message Bridges
Web Services Adapters

System Services
Transactions
Messaging
Scheduling
Pooling
Clustering

Management Services
Systems Management
Security
Directory

Oracle Application Server 10*g* Components



Wireless

Web Cache

Clickstream Personalization Reports Discoverer

Unified Messaging Internet File System

InterConnect Workflow Portal

XDK Web Services Forms

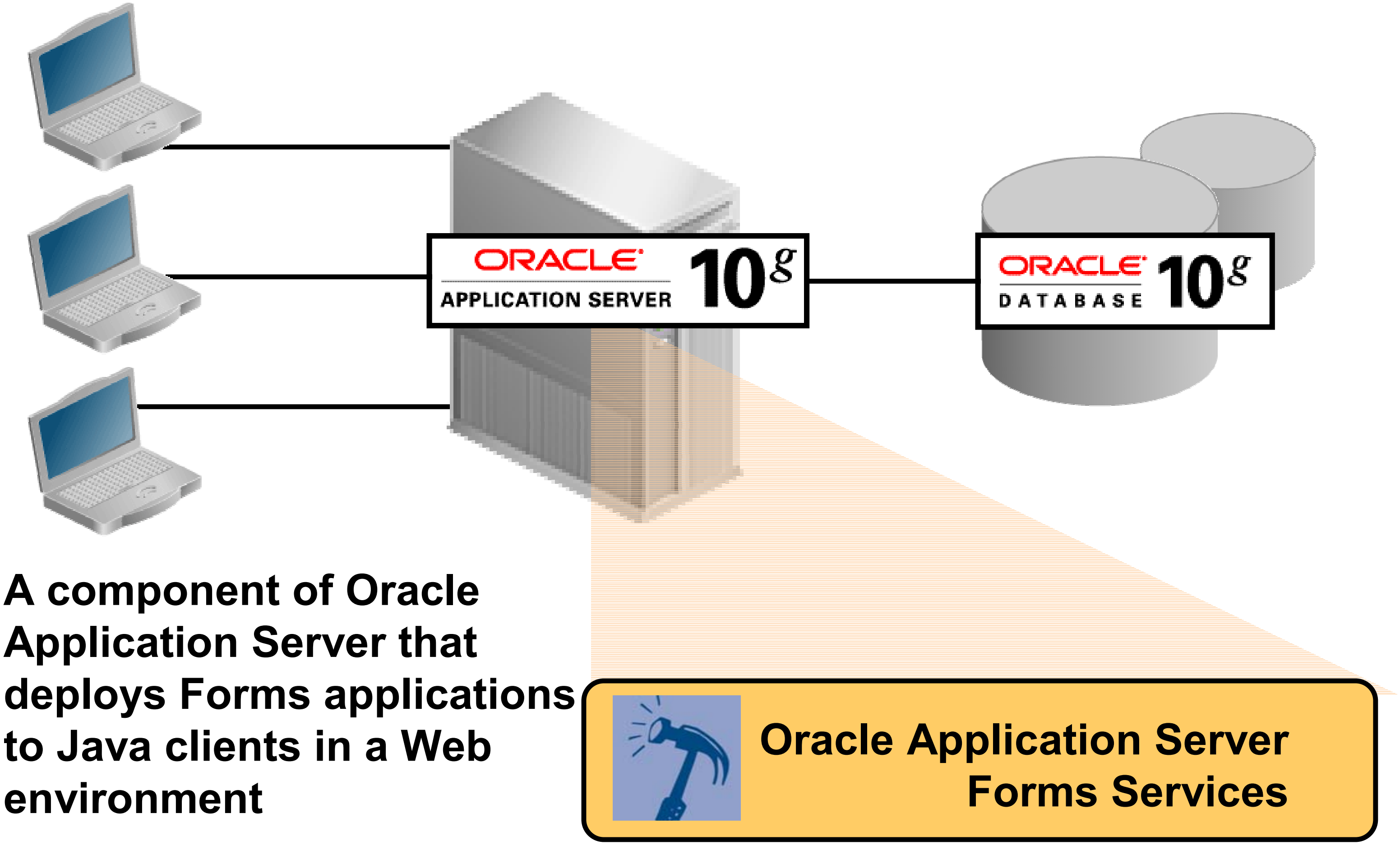
OC4J

HTTP Server

Enterprise Manager Security Internet Directory

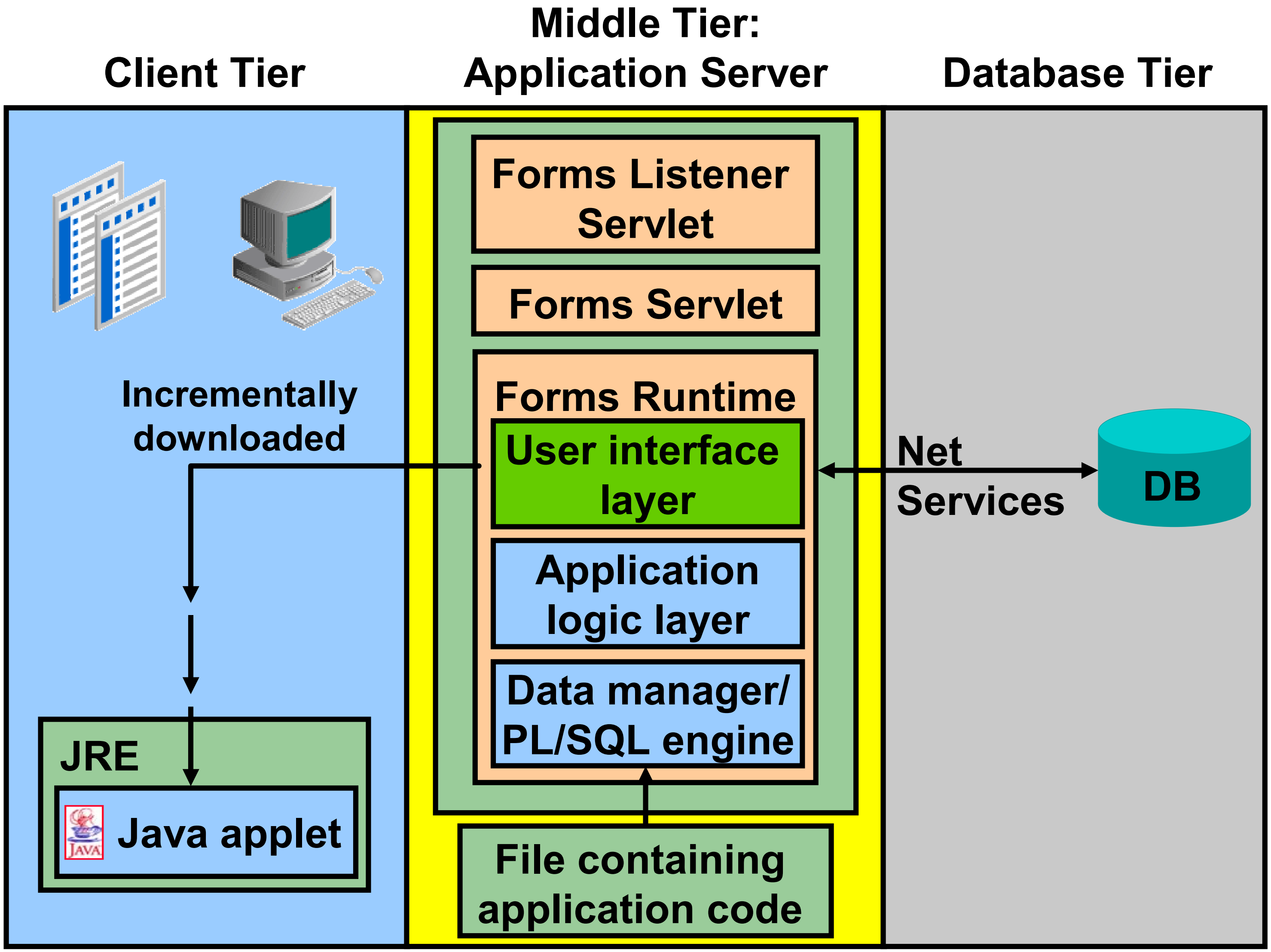
Oracle Application Server Forms Services

Oracle Forms Services Overview

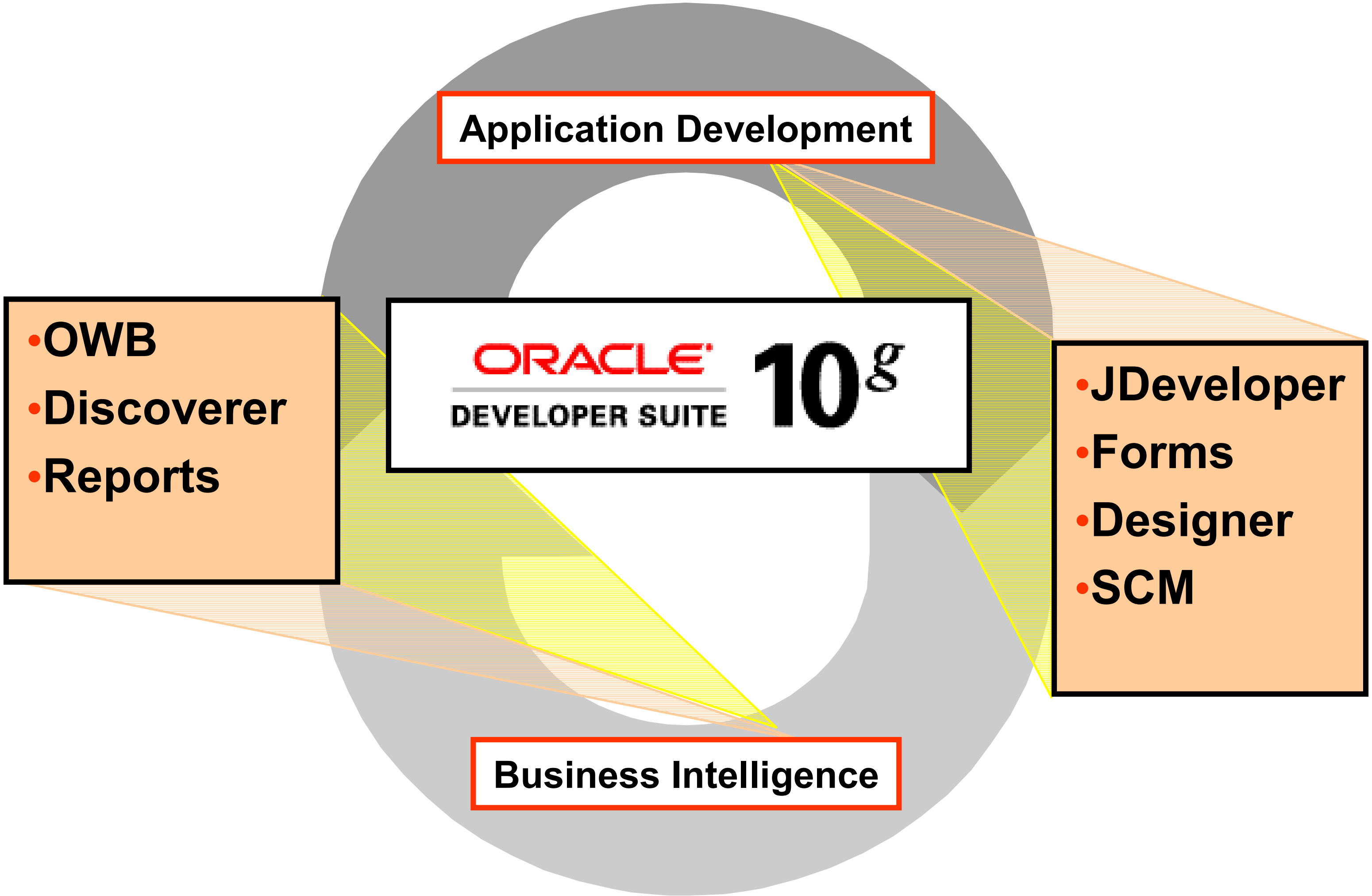


A component of Oracle Application Server that deploys Forms applications to Java clients in a Web environment

Forms Services Architecture



Benefits and Components of Oracle Developer Suite 10*g*



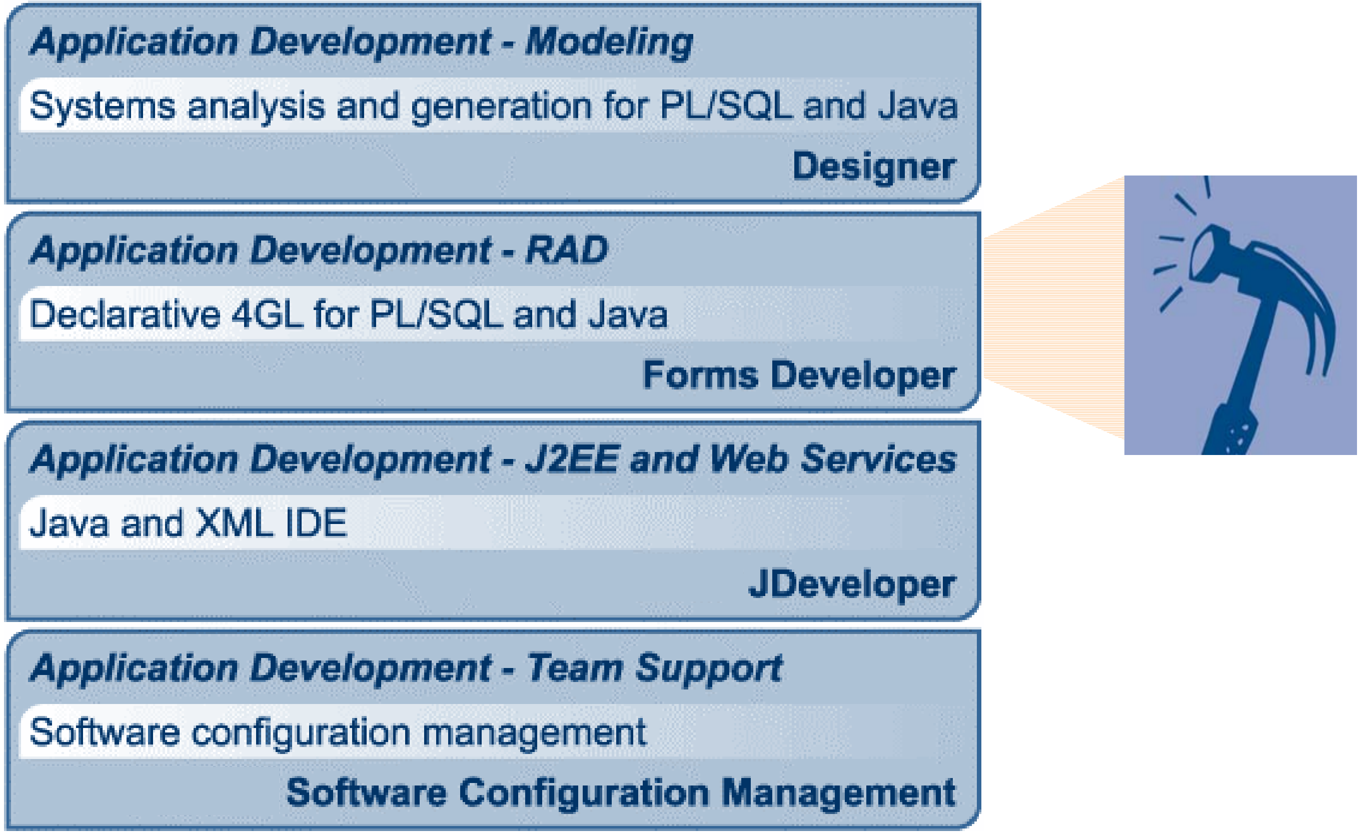
Application Development

- JDeveloper
- Forms
- Designer
- SCM

Business Intelligence

- OWB
- Discoverer
- Reports

Oracle Developer Suite 10*g* Application Development



Application Development - Modeling

Systems analysis and generation for PL/SQL and Java

Designer

Application Development - RAD

Declarative 4GL for PL/SQL and Java

Forms Developer

Application Development - J2EE and Web Services

Java and XML IDE

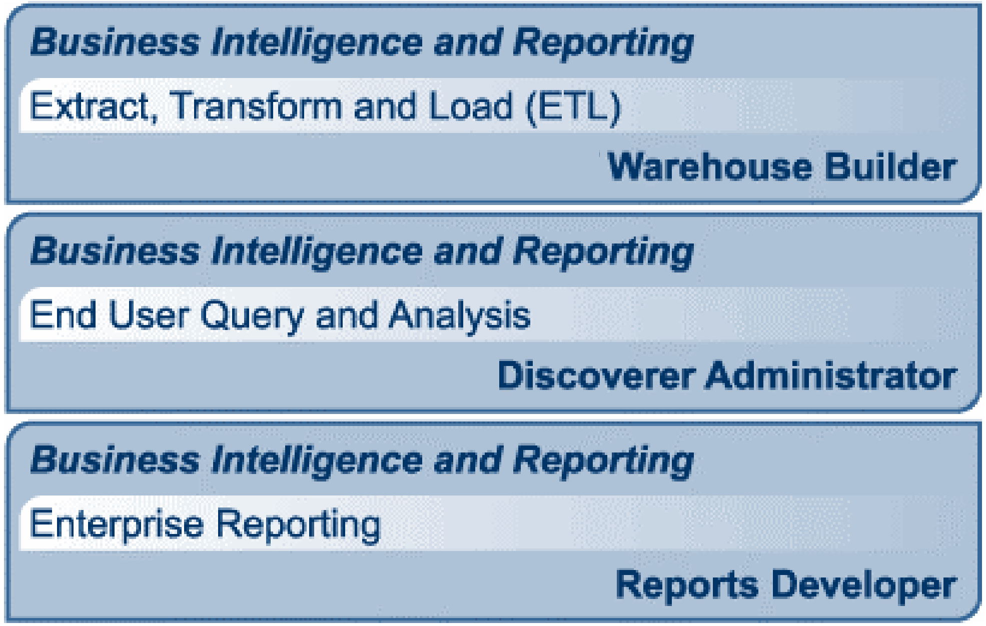
JDeveloper

Application Development - Team Support

Software configuration management

Software Configuration Management

Oracle Developer Suite 10*g* Business Intelligence



Business Intelligence and Reporting

Extract, Transform and Load (ETL)

Warehouse Builder

Business Intelligence and Reporting

End User Query and Analysis

Discoverer Administrator

Business Intelligence and Reporting

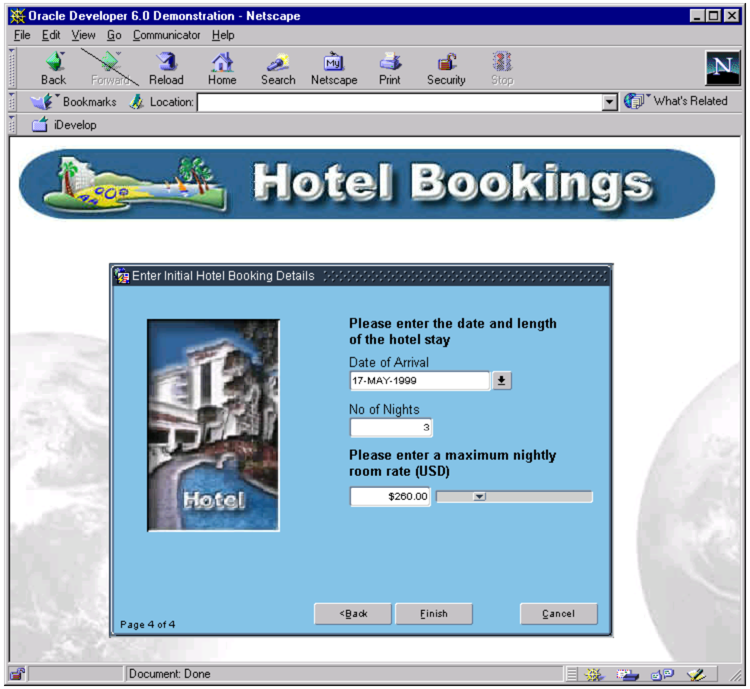
Enterprise Reporting

Reports Developer

Oracle Forms Developer Overview

Oracle Forms Developer:

- Is a productive development environment for Internet business applications
- Provides for:
  - Data entry
  - Queries


Oracle Developer 6.0 Demonstration - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: What's Related
iDevelop
Hotel Bookings
Enter Initial Hotel Booking Details
Hotel
<Back Finish Cancel
Document: Done

Please enter the date and length of the hotel stay

Date of Arrival: 17-MAY-1999

No of Nights: 3

Please enter a maximum nightly room rate (USD)

$260.00

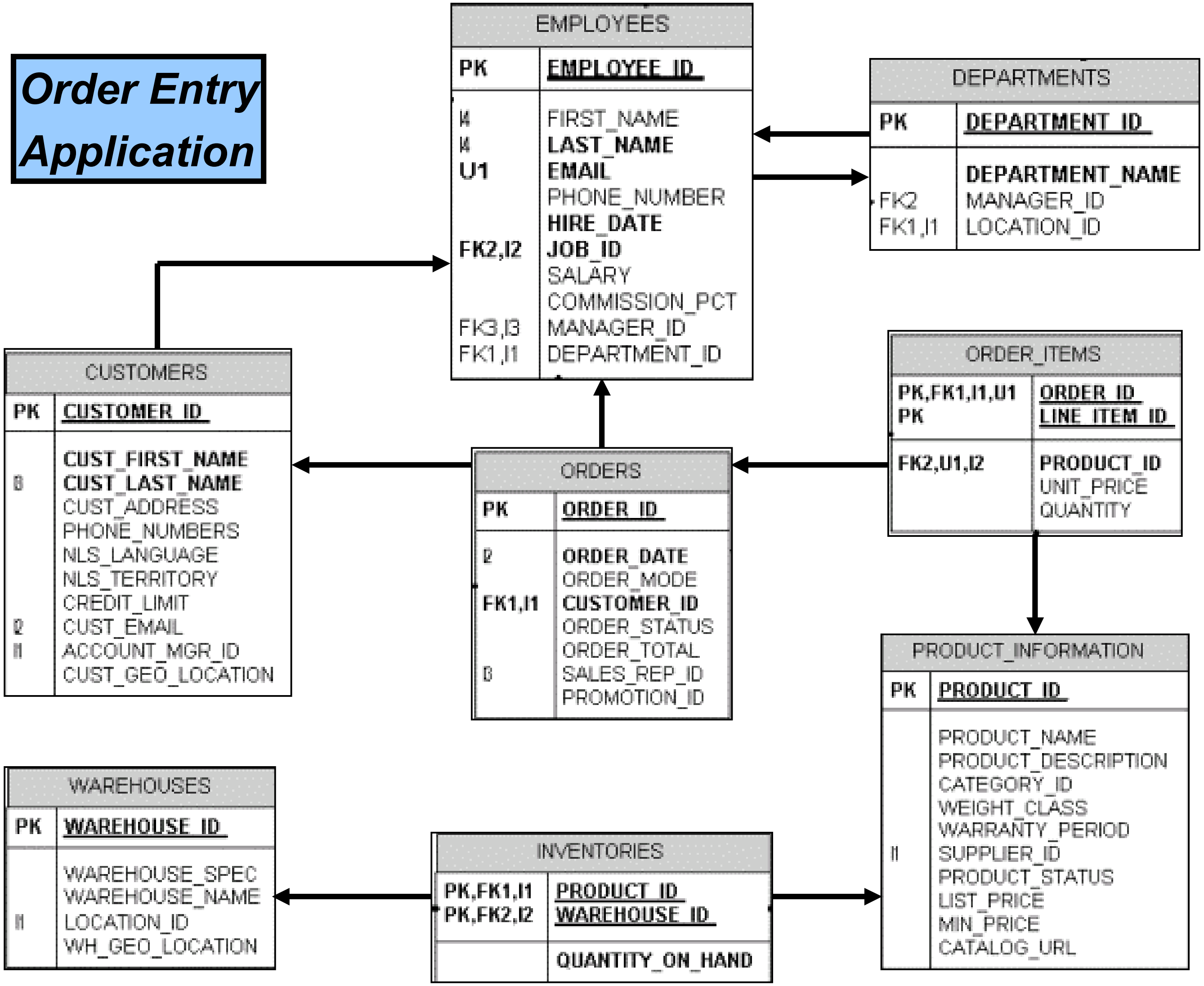
Page 4 of 4

Oracle Forms Developer: Key Features

- Tools for rapid application development
- Application partitioning
- Flexible source control
- Extended scalability
- Object reuse

Summit Office Supply Schema

Order Entry Application



	EMPLOYEES
PK	**EMPLOYEE_ID**
I4	FIRST_NAME
I4	**LAST_NAME**
U1	**EMAIL**
	PHONE_NUMBER
	HIRE_DATE
FK2,I2	**JOB_ID**
	SALARY
	COMMISSION_PCT
FK3,I3	MANAGER_ID
FK1,I1	DEPARTMENT_ID

	DEPARTMENTS
PK	**DEPARTMENT_ID**
	DEPARTMENT_NAME
FK2	MANAGER_ID
FK1,I1	LOCATION_ID

	CUSTOMERS
PK	**CUSTOMER_ID**
	CUST_FIRST_NAME
I3	**CUST_LAST_NAME**
	CUST_ADDRESS
	PHONE_NUMBERS
	NLS_LANGUAGE
	NLS_TERRITORY
	CREDIT_LIMIT
I2	CUST_EMAIL
I1	ACCOUNT_MGR_ID
	CUST_GEO_LOCATION

	ORDERS
PK	**ORDER_ID**
I2	**ORDER_DATE**
	ORDER_MODE
FK1,I1	**CUSTOMER_ID**
	ORDER_STATUS
	ORDER_TOTAL
I3	SALES_REP_ID
	PROMOTION_ID

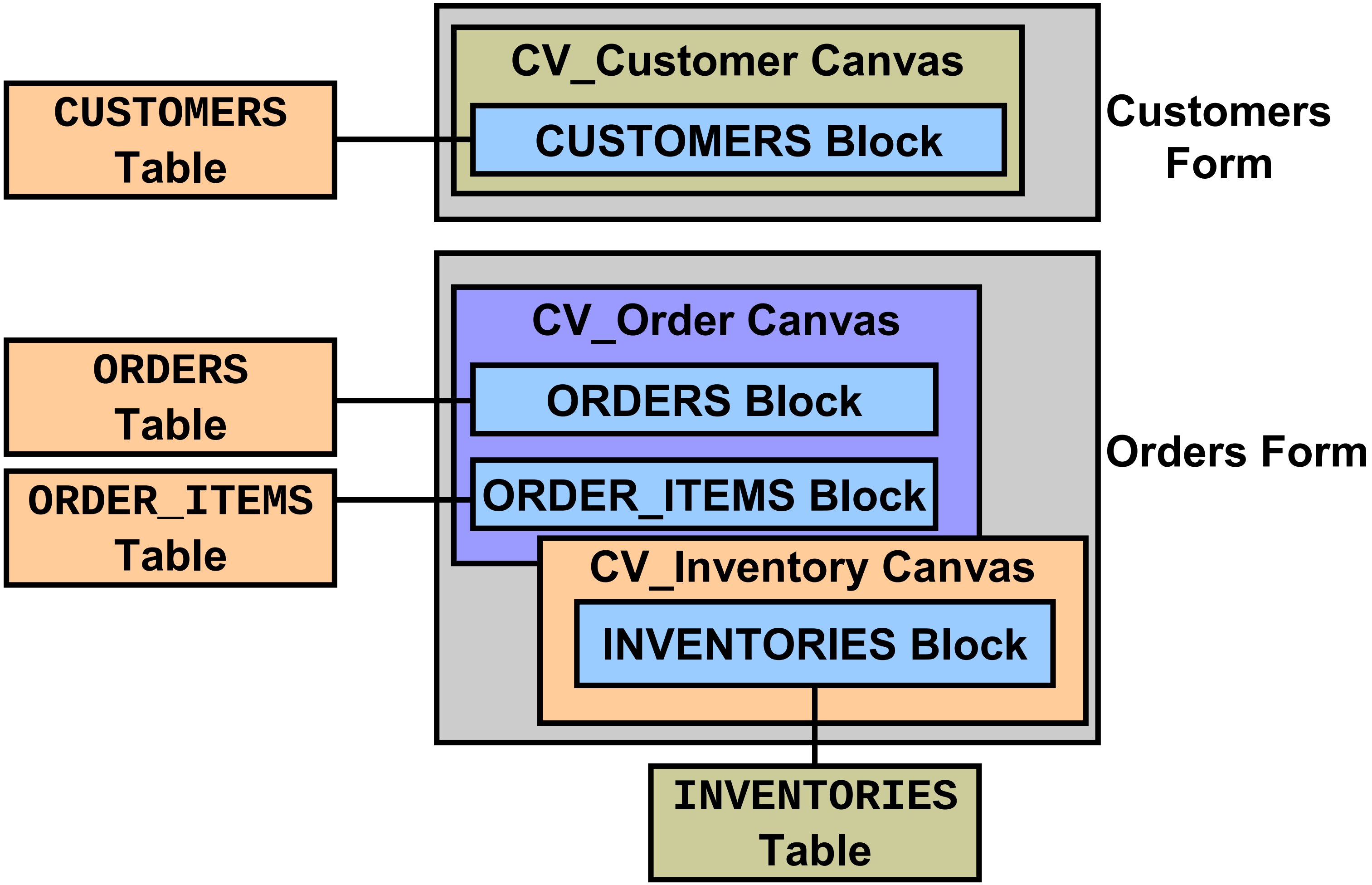
	ORDER_ITEMS
PK,FK1,I1,U1	**ORDER_ID**
PK	**LINE_ITEM_ID**
FK2,U1,I2	**PRODUCT_ID**
	UNIT_PRICE
	QUANTITY

	PRODUCT_INFORMATION
PK	**PRODUCT_ID**
	PRODUCT_NAME
	PRODUCT_DESCRIPTION
	CATEGORY_ID
	WEIGHT_CLASS
	WARRANTY_PERIOD
I1	SUPPLIER_ID
	PRODUCT_STATUS
	LIST_PRICE
	MIN_PRICE
	CATALOG_URL

	WAREHOUSES
PK	**WAREHOUSE_ID**
	WAREHOUSE_SPEC
	WAREHOUSE_NAME
I1	LOCATION_ID
	WH_GEO_LOCATION

	INVENTORIES
PK,FK1,I1	**PRODUCT_ID**
PK,FK2,I2	**WAREHOUSE_ID**
	QUANTITY_ON_HAND

Summit Application



Summary

In this lesson, you should have learned that:

- Grid computing makes computing power available without regard to its source
- Oracle 10*g* products provide the software to implement enterprise grid computing
- Oracle Application Server 10*g* provides services for building and deploying Web applications
- Oracle Developer Suite 10*g* includes components for application development and business intelligence

Summary

- Benefits of Oracle Forms Services include:
  - Optimized Web deployment of Forms applications
  - Rich Java UI without Java coding
  - Generic Java applet to deploy any Forms application
- Oracle Forms Services consists of the Forms client, the Forms Servlet, the Forms Listener Servlet, and the Forms Runtime Engine.
- Benefits of Oracle Forms Developer include rapid application development, application partitioning, flexible source control, extended scalability, and object reuse.
- The course application is a customer and order entry application for Summit Office Supply.

2

Running a Forms Developer Application

Objectives

After completing this lesson, you should be able to do the following:

- Start OC4J
- Describe the run-time environment
- Describe the elements in a running form
- Navigate a Forms application
- Describe the two main modes of operation
- Run a form in a Web browser
  - Retrieve both restricted and unrestricted data
  - Insert, update, and delete records
  - Display database errors

Testing a Form: OC4J Overview

Oracle Application Server Containers for J2EE (OC4J) is:

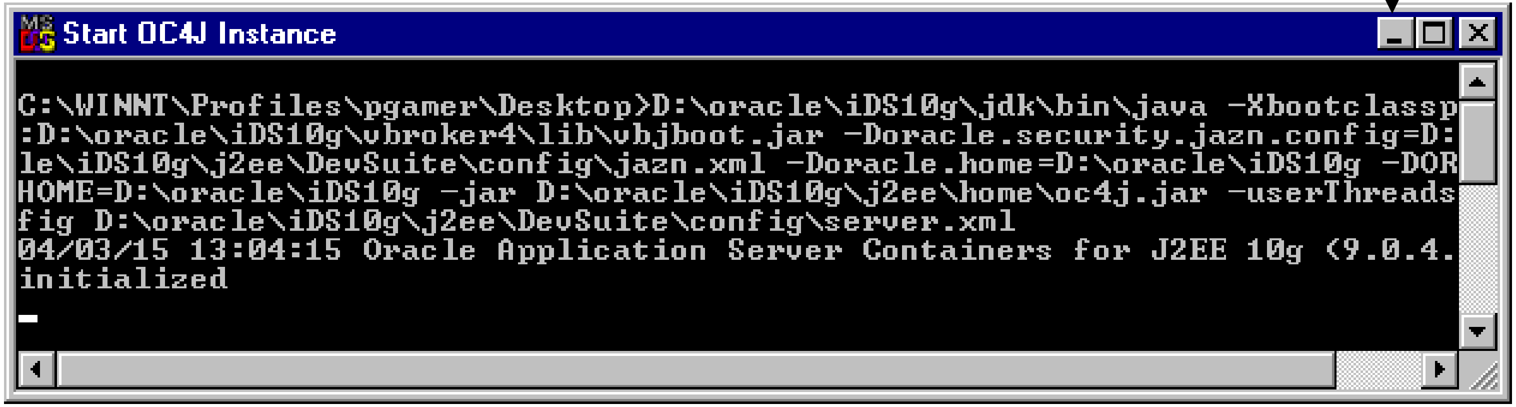
- Preferred to run Forms applications
- Included with Oracle Developer Suite to enable testing

Testing a Form: Starting OC4J

- On NT, run batch file to start OC4J: `startinst.bat`.
- OC4J starts in DOS window:
  - Minimize window
  - Closing window aborts OC4J
- Run batch file to stop OC4J: `stopinst.bat`.

Shortcut to startinst.bat

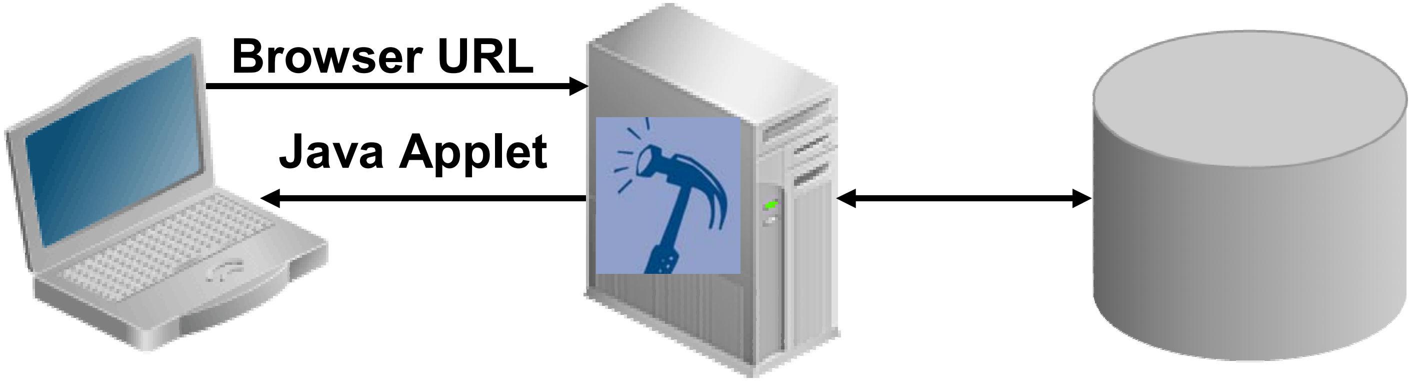
Shortcut to stopinst.bat



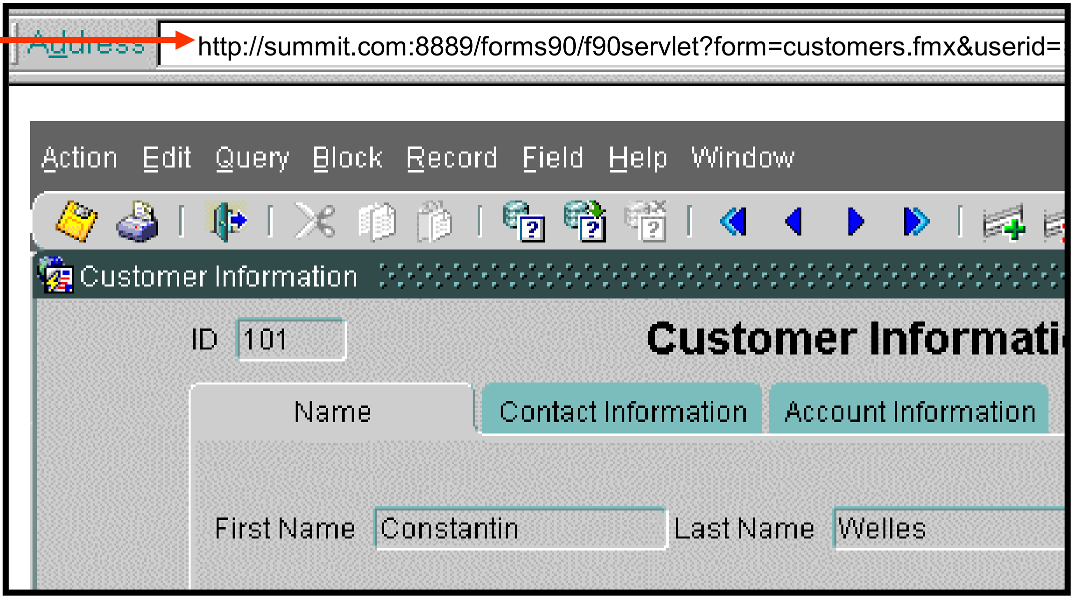
```
C:\WINNT\Profiles\pgamer\Desktop>D:\oracle\iDS10g\jdk\bin\java -Xbootclassp
:D:\oracle\iDS10g\vbroker4\lib\vbjboot.jar -Doracle.security.jazn.config=D:
le\iDS10g\j2ee\DevSuite\config\jazn.xml -Doracle.home=D:\oracle\iDS10g -DOR
HOME=D:\oracle\iDS10g -jar D:\oracle\iDS10g\j2ee\home\oc4j.jar -userThreads
fig D:\oracle\iDS10g\j2ee\DevSuite\config\server.xml
04/03/15 13:04:15 Oracle Application Server Containers for J2EE 10g (9.0.4.
initialized
_
```

Running a Form

Oracle Forms Services deployment:



Running a Form: Browser



http://summit.com:8889/forms90/f90servlet?form=customers.fmx&userid=

Customer Informati

ID 101

First Name Constantin

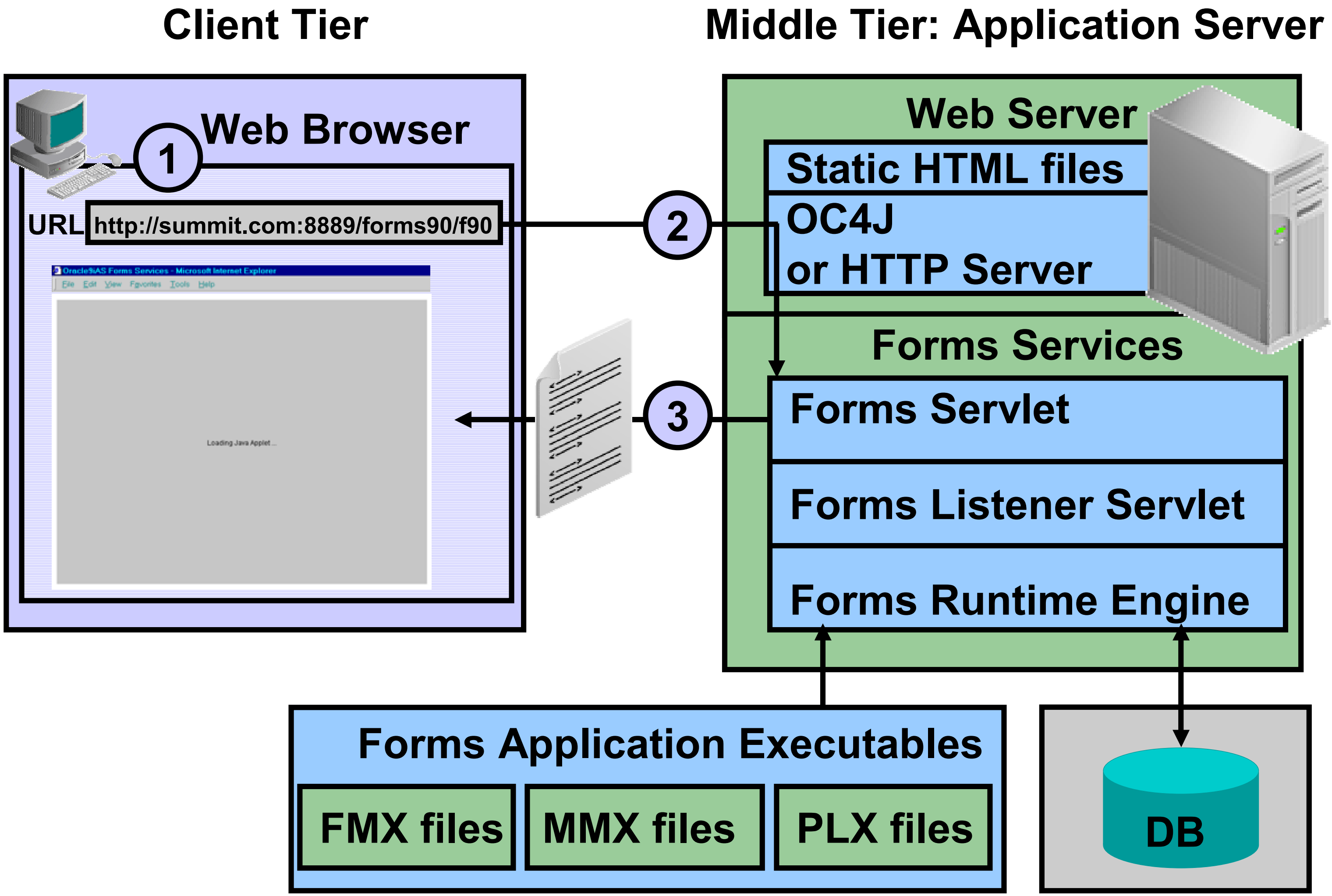
Last Name Welles

```
http://summit.com:8889/forms90/f90servlet
?form=customers.fmx&userid=username/password@database
&buffer_records=NO&debug_messages=NO&array=YES
&query_only=NO
```

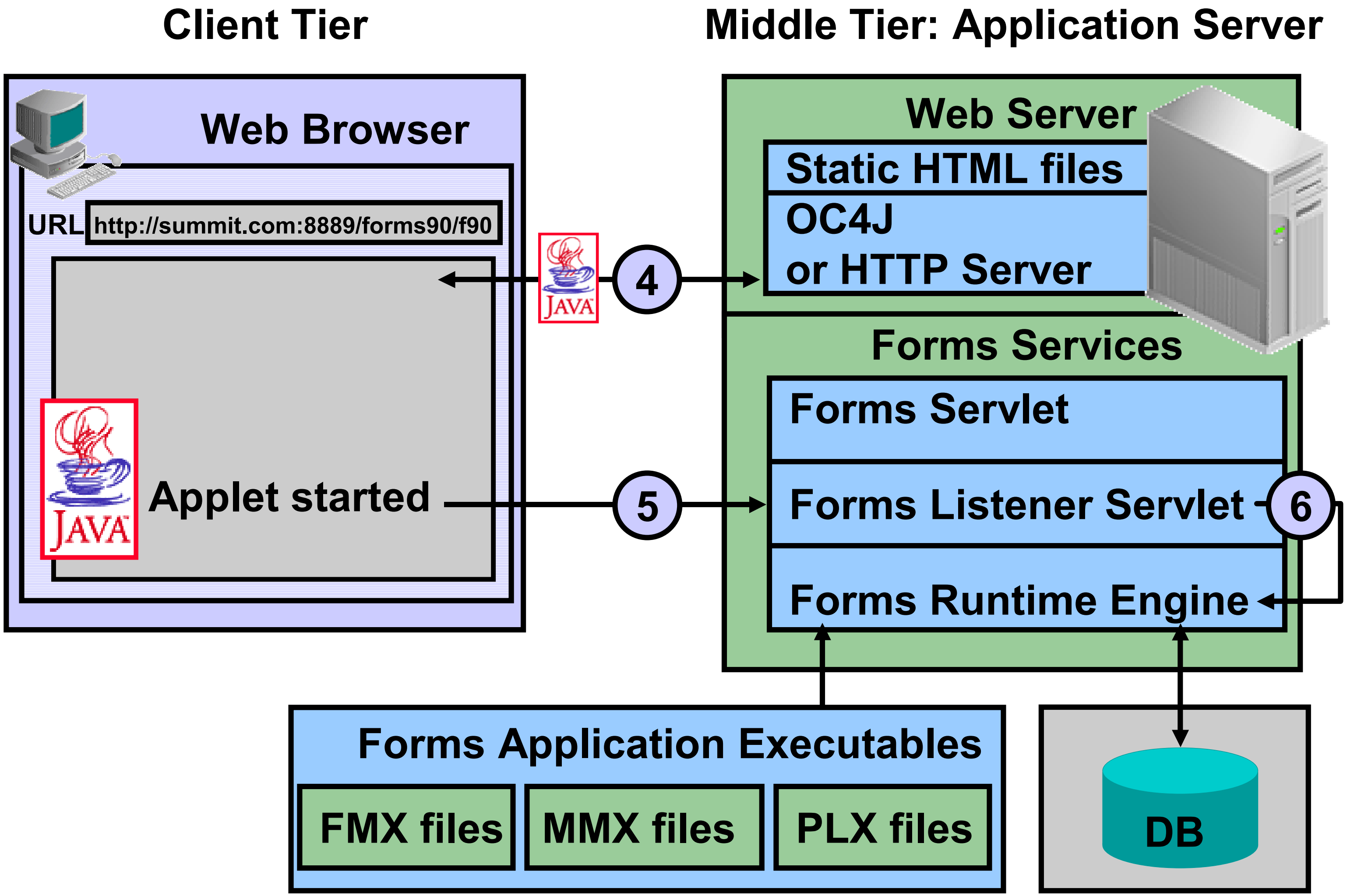
The Java Runtime Environment

- The Forms applet runs in a Java Runtime Environment (JRE) on the client machine.
- Types of JREs:
  - Java-enabled browser (native)
  - JInitiator (Oracle-supplied plug-in to Web browser) that provides:
    Incremental Java archive (JAR) file downloading
    JAR file caching
    Applet instance caching
    Automatic Java security configuration

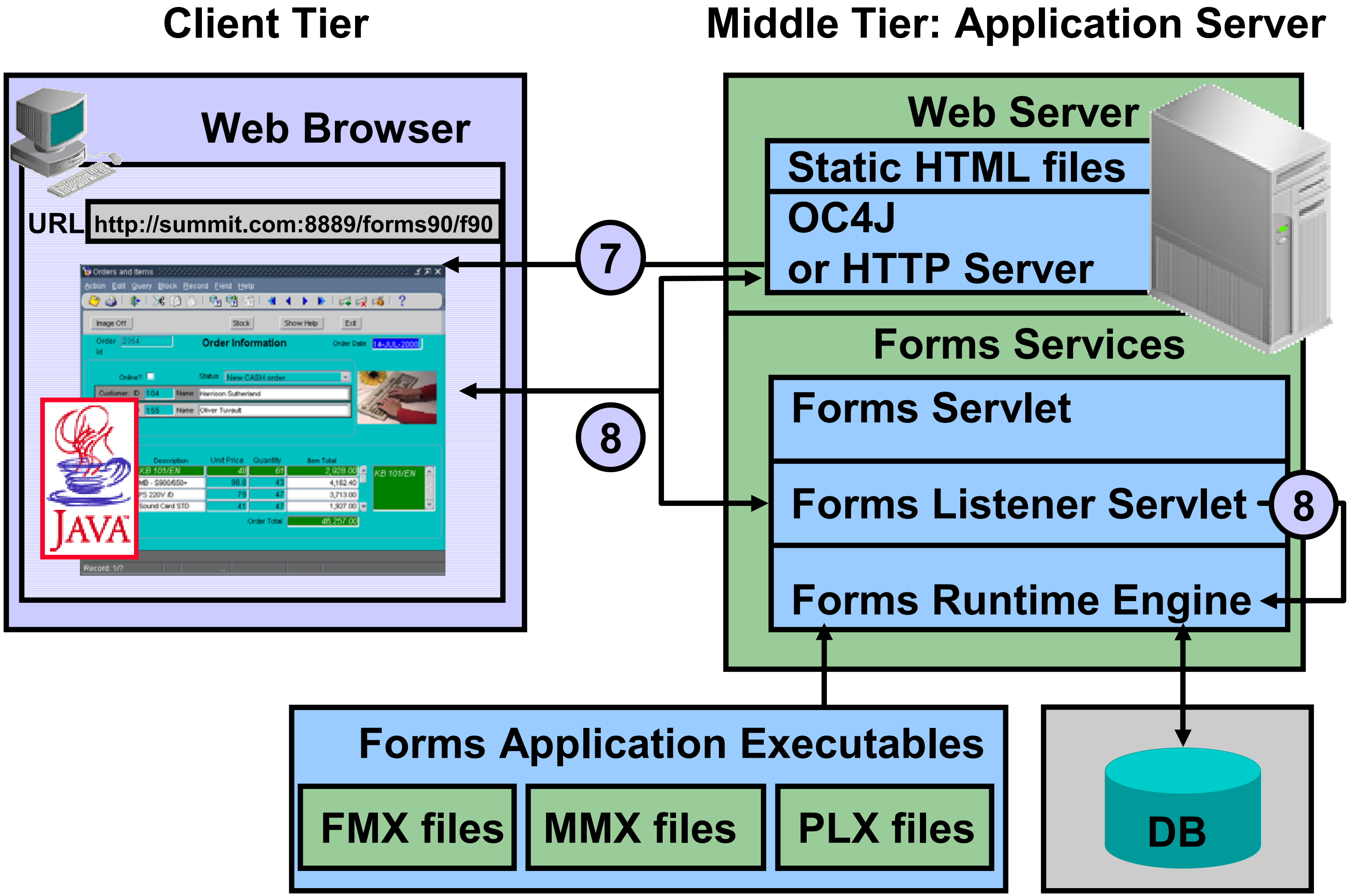
Starting a Run-Time Session



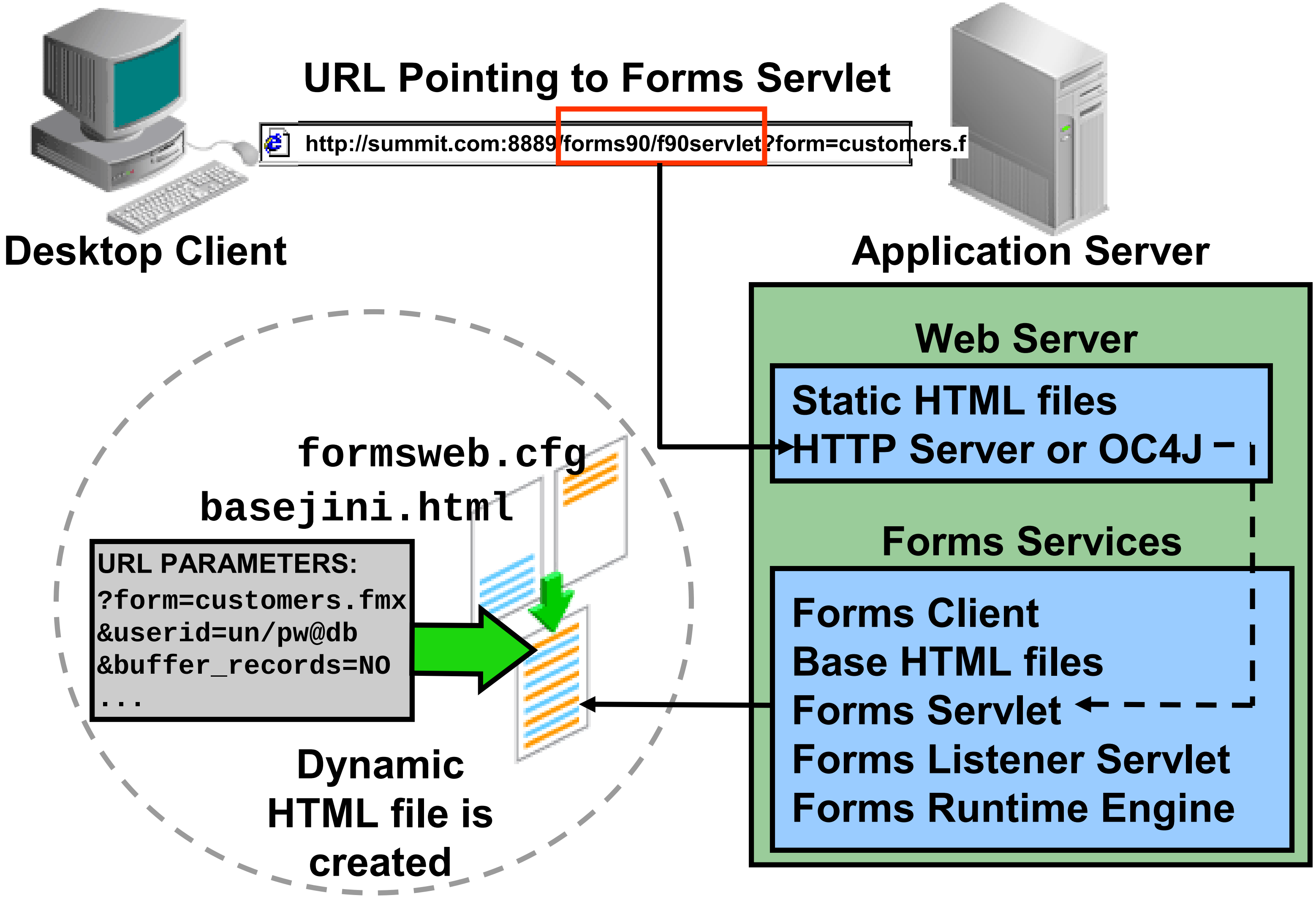
Starting a Run-Time Session



Starting a Run-Time Session



The Forms Servlet



URL Pointing to Forms Servlet

http://summit.com:8889/forms90/f90servlet?form=customers.f

Desktop Client

Application Server

Web Server

Static HTML files

HTTP Server or OC4J

Forms Services

Forms Client

Base HTML files

Forms Servlet

Forms Listener Servlet

Forms Runtime Engine

formsweb.cfg

basejini.html

URL PARAMETERS:

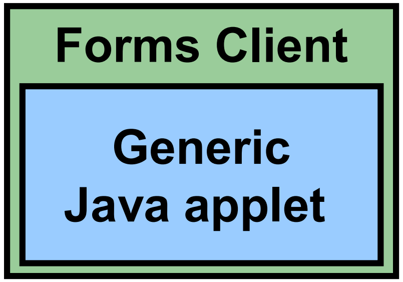
```
?form=customers.fmx
&userid=un/pw@db
&buffer_records=NO
...
```

Dynamic HTML file is created

The Forms Client

- Generic Java applet
- Responsibilities:
  - Displays the form’s user interface
  - Processes user interaction back to Forms Services
  - Processes incoming messages from Forms Services

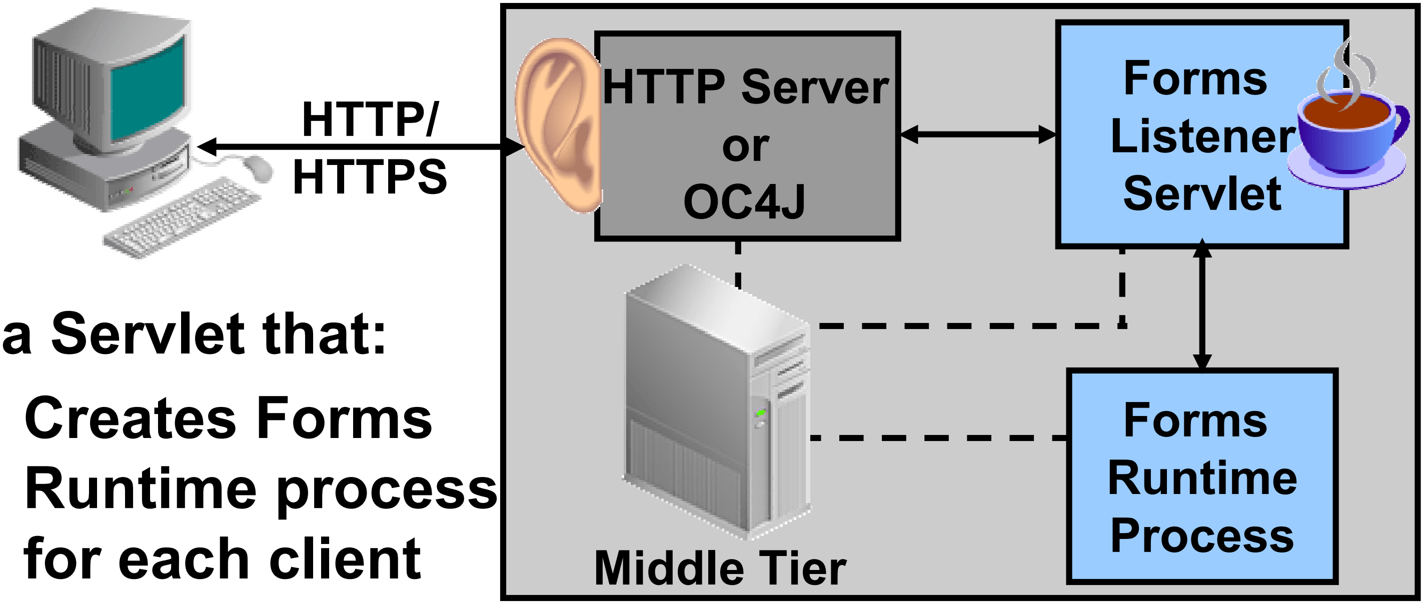
Desktop Client



Forms Client

Generic Java applet

The Forms Listener Servlet



Java Servlet that:

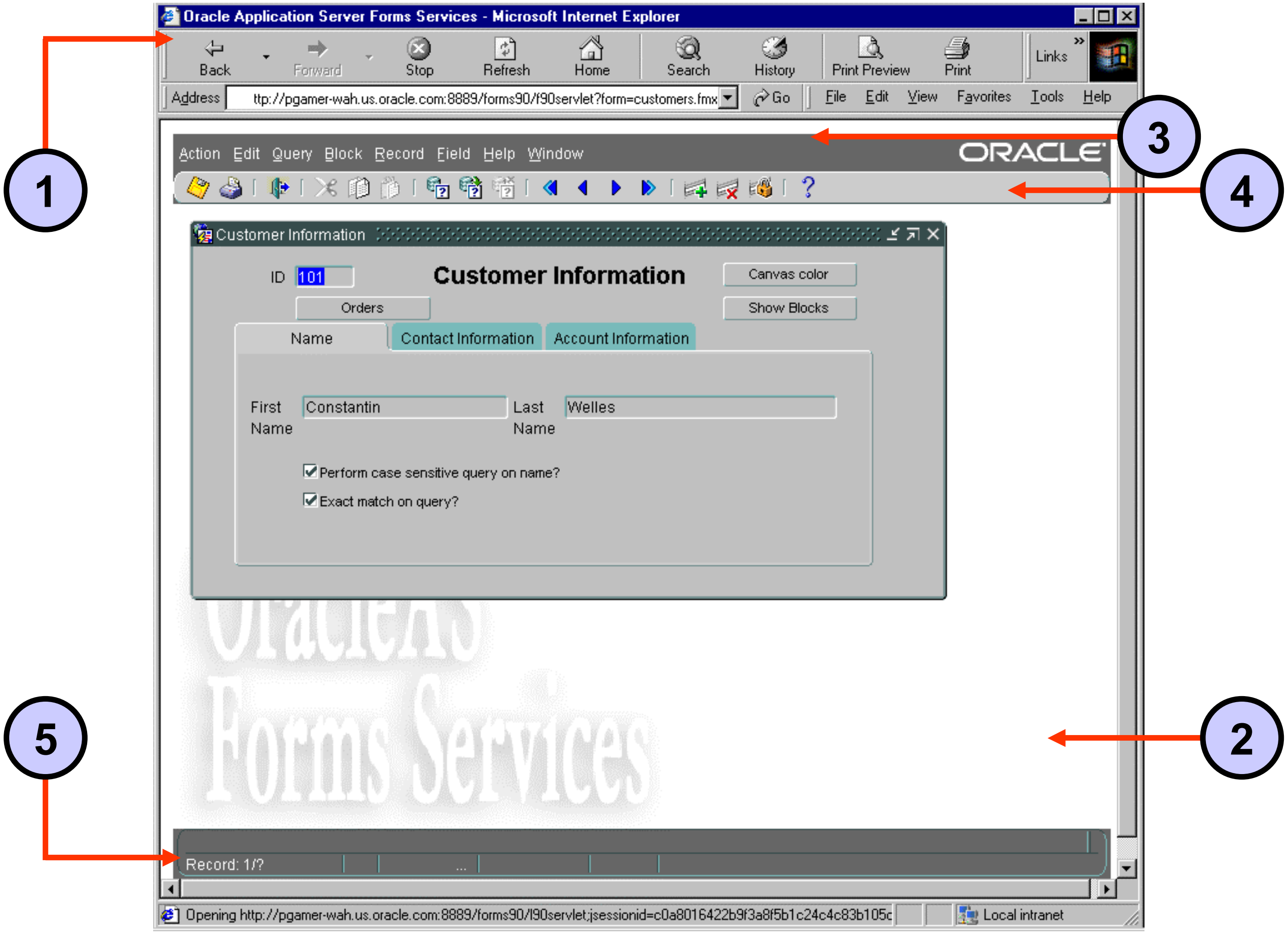
- Creates Forms Runtime process for each client
- Stops the Runtime process at session end
- Manages network communications between client and Forms Runtime process
- Communicates through Web server process

The Runtime Engine

The Forms Runtime Engine:

- Is a process (`ifweb90`) that runs on the Application Server
- Manages application logic and processing
- Communicates with the client browser and the database

What You See at Run Time



Customer Information

ID 101

First Name: Constantin

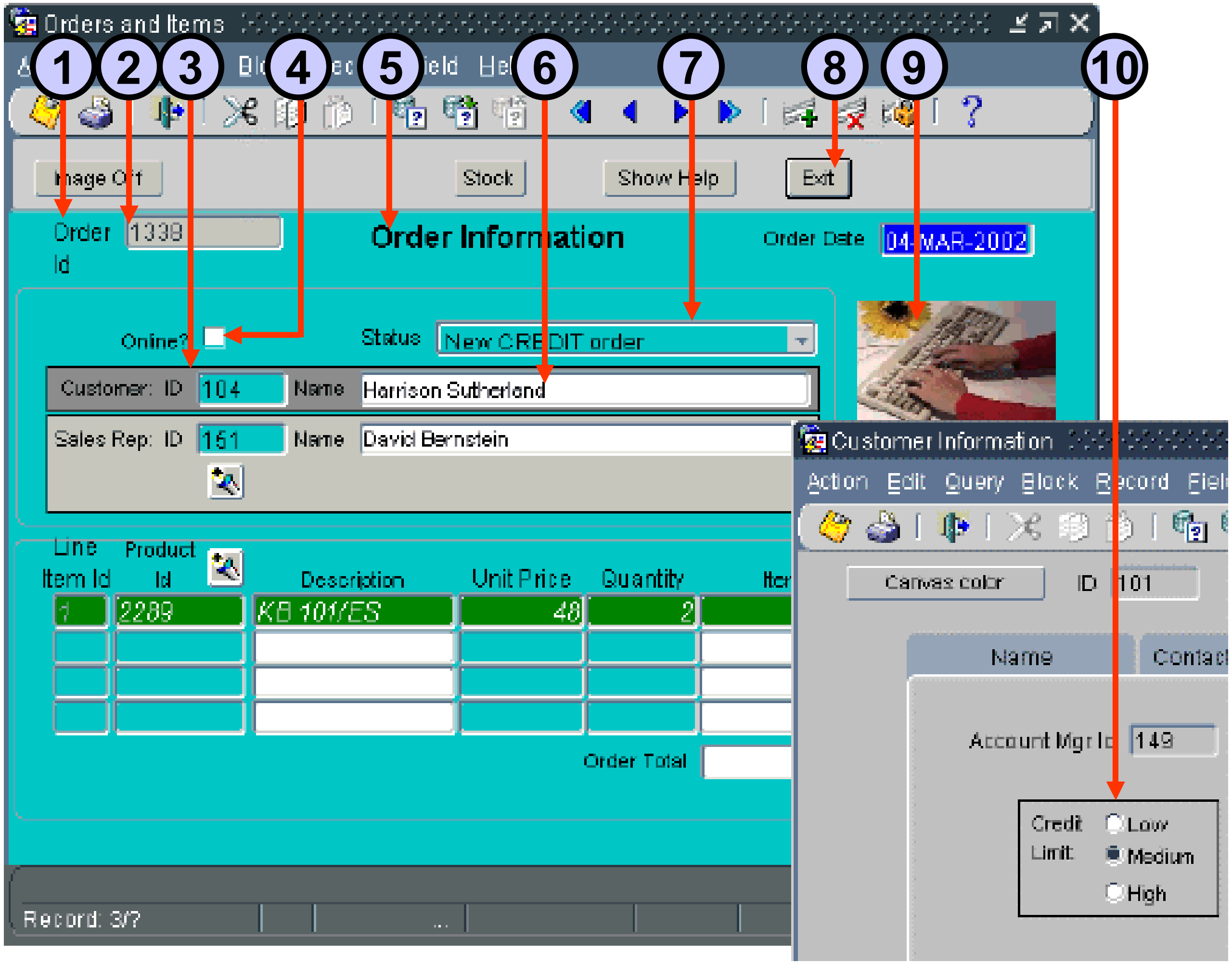
Last Name: Welles

☑ Perform case sensitive query on name?

☑ Exact match on query?

Record: 1/?

Identifying the Data Elements



Order Id 1339

Order Date 04-MAR-2002

Online? □

Status New CREDIT order

Customer: ID 104 Name Harrison Sutherland

Sales Rep: ID 151 Name David Bernstein

Line Item Id	Product Id	Description	Unit Price	Quantity
1	2289	KB 101/ES	48	2

Record: 3/?

ID 101

Account Mgr Id 149

Credit Limit ○ Low ● Medium ○ High

Modes of Operation: Enter-Query Mode

Allows:

- Unrestricted and restricted queries
- Query/Where dialog box
- Record count by using Query > Count Hits

Does not allow:

- Navigation out of current data block
- Exiting run-time session
- Certain functions
- Insert, update, delete

Modes of Operation: Normal Mode

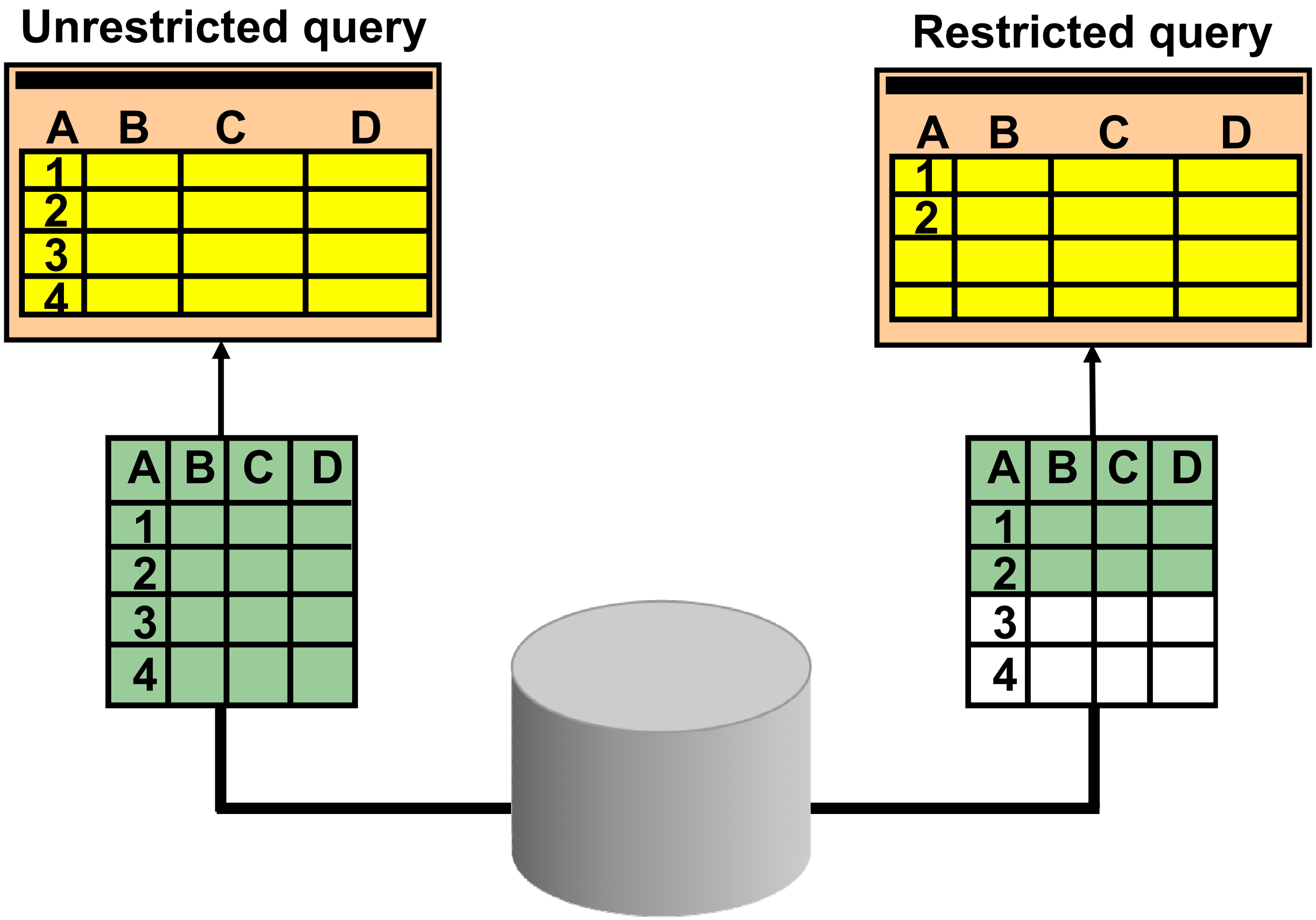
Allows:

- Unrestricted queries
- Insert, update, delete
- Commit (Save)
- Navigation out of current data block
- Exiting run-time session

Does Not Allow:

- Restricted queries
- Query/Where dialog box

Retrieving Data



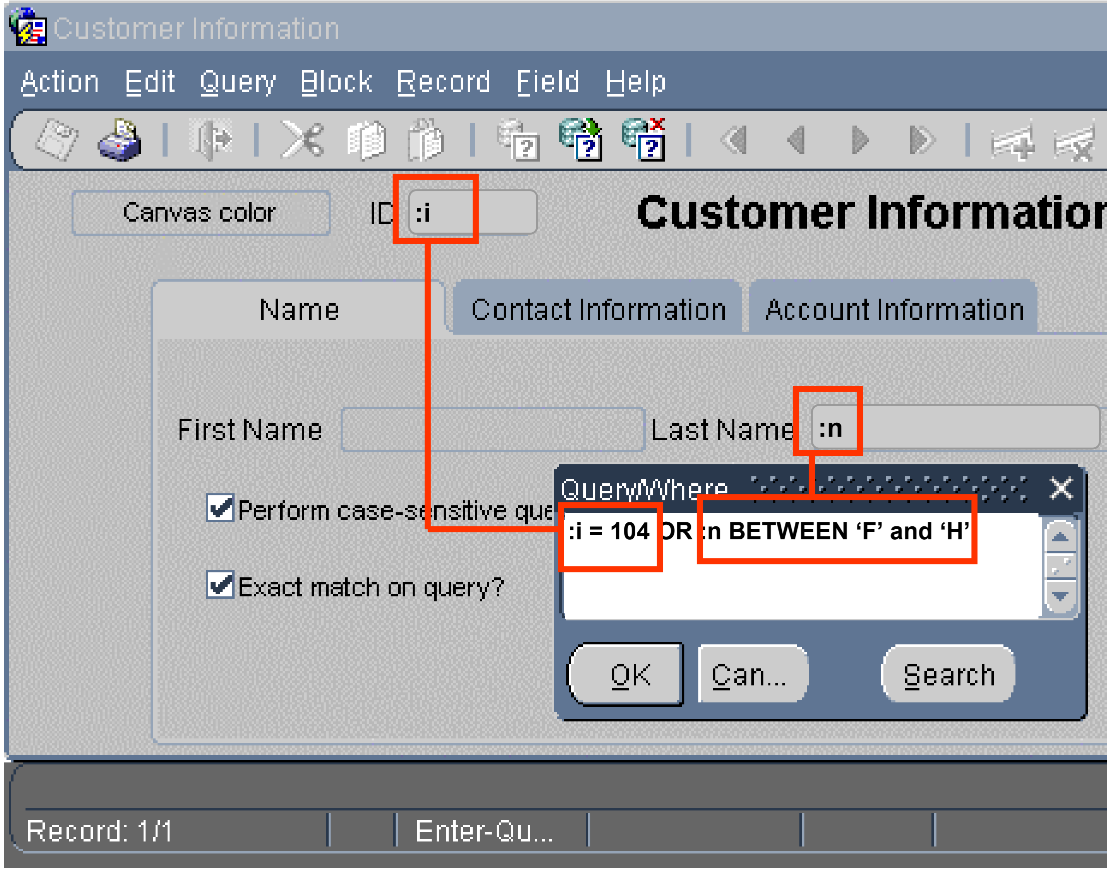
Retrieving Restricted Data

- Do not use quotation marks with character and date items.
- The `LIKE` operator is implied with % or _.
- Use hash (#) in front of SQL operators.
- Use Query/Where for complex query conditions.
- Use default date format (DD-MON-RR) in Query/Where.
- Use quotes around literals in Query/Where.

Query/Where Dialog Box

- Invoke by:
  - Entering `:variable_name`
  - Executing query
- Used to write:
  - Complex search conditions
  - Queries with `OR` predicates
  - `ORDER BY` clause

Query/Where Dialog Box



ID :i

First Name

Last Name :n

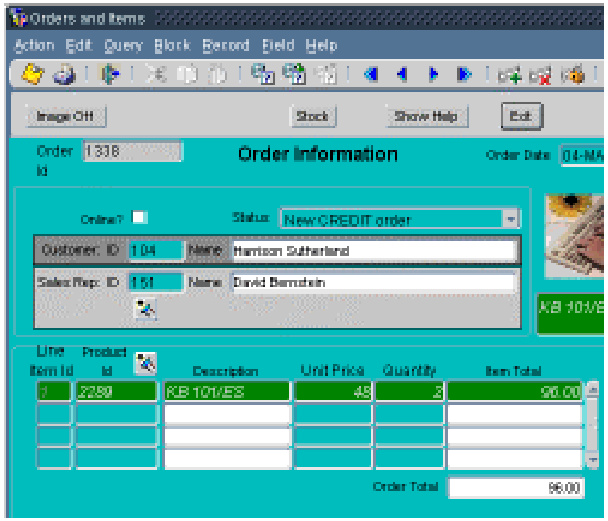
☑ Perform case-sensitive que

☑ Exact match on query?

:i = 104 OR :n BETWEEN ‘F’ and ‘H’

Inserting, Updating, and Deleting

Form module



Order Id 1338

Order Information

Order Date 04-MA

Online? □

Status: New CREDIT order

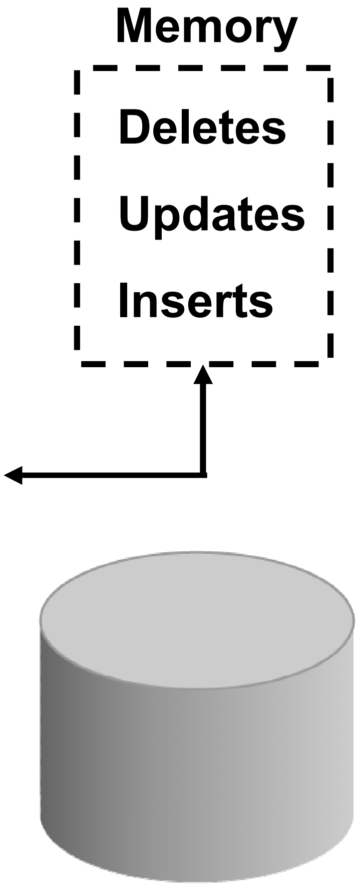
Customer: ID 104 Name: Harrison Sutherland

Sales Rep: ID 151 Name: David Bernstein

KB 101/E

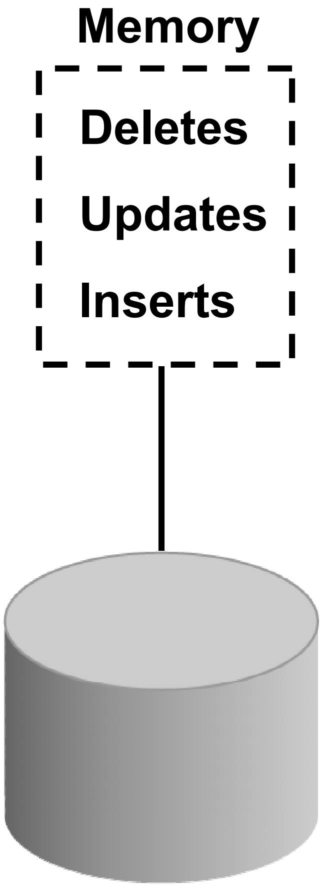
Line Item Id	Product Id	Description	Unit Price	Quantity	Item Total
1	2289	KB 101/ES	48	2	96.00

Order Total 96.00



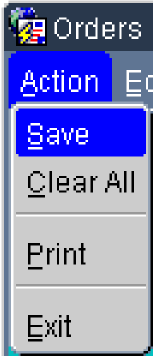
Making Changes Permanent

- Select Action > Save to make changes permanent.
- Select Action > Clear All to discard changes.

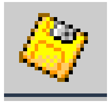


To commit or rollback:

Menu

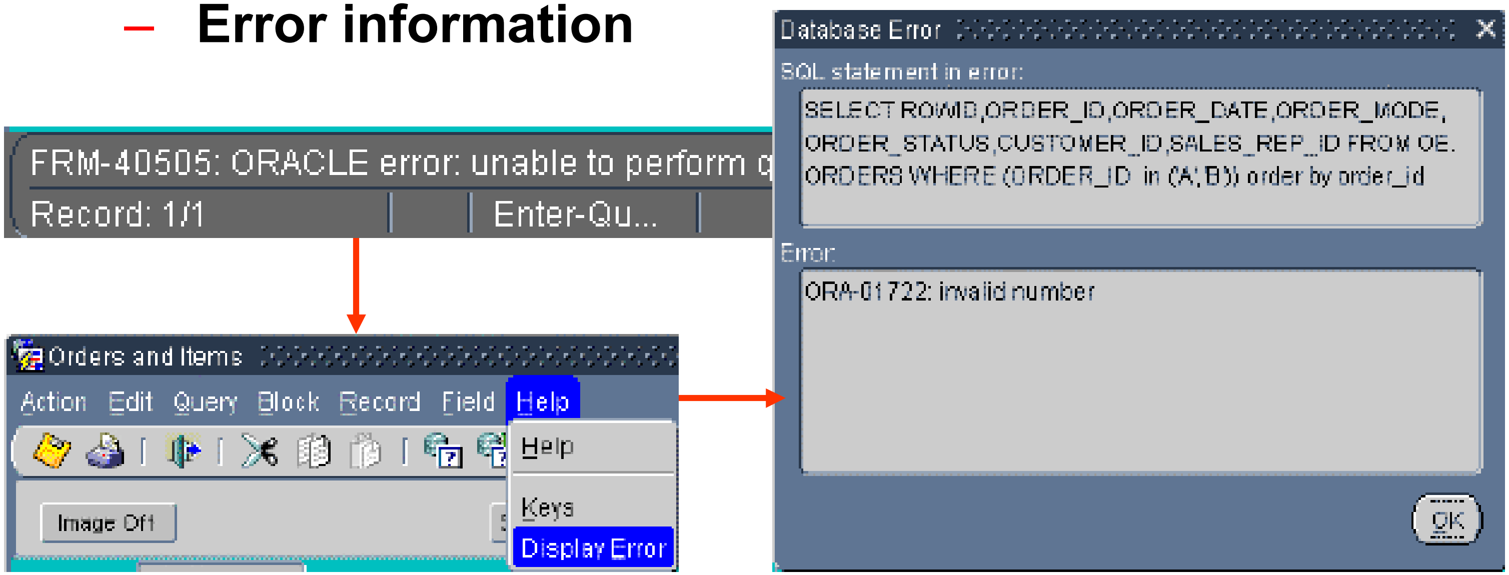


or
Toolbar



Displaying Errors

- Use to view Oracle errors
- Select Help > Display Error
- Shows Database Error window:
  - SQL statement
  - Error information



FRM-40505: ORACLE error: unable to perform q

Record: 1/1 | | Enter-Qu... |

SQL statement in error:

```
SELECT ROWID,ORDER_ID,ORDER_DATE,ORDER_MODE,
ORDER_STATUS,CUSTOMER_ID,SALES_REP_ID FROM OE.
ORDERS WHERE (ORDER_ID  in ('A','B')) order by order_id
```

Error:

```
ORA-01722: invalid number
```

Summary

In this lesson, you should have learned that:

- You can use OC4J on the development machine to run a Forms application in a Web browser
- At run time:
  - The Forms Client is downloaded
  - The Forms Servlet creates a start HTML file
  - The Forms Listener Servlet starts a run-time session and maintains communication between it and the Forms Client
  - The Runtime Engine carries out application logic and maintains a database connection on behalf of the Forms Client

Summary

- When you run a form you see a Java applet running in a browser and displaying a menu, menu toolbar, console, and several kinds of data elements.
- Users navigate a Forms application using the menu, toolbar, the mouse, buttons, or function keys.
- The two main modes of operation are Normal mode and Enter-Query mode.
- Executing a query returns all records, unless the query is restricted by search criteria.

Summary

- In normal mode you can insert, update, and delete records and commit changes to the database.
- You display database errors from the menu (Help > Display Error)

Practice 2 Overview

This practice covers the following topics:

- Starting OC4J
- Running the course application:
  - Querying records
  - Inserting a record
  - Updating a record
  - Deleting a record
  - Displaying a database error

3

Working in the Forms Developer Environment

Objectives

After completing this lesson, you should be able to do the following:

- Describe Forms Builder components
- Navigate the Forms Builder interface
- Identify the main objects in a form module
- Customize the Forms Builder session
- Use the online help facilities
- Identify the main Forms executables
- Describe the Forms module types
- Set environment variables for design and run time
- Run a form from within Forms Builder

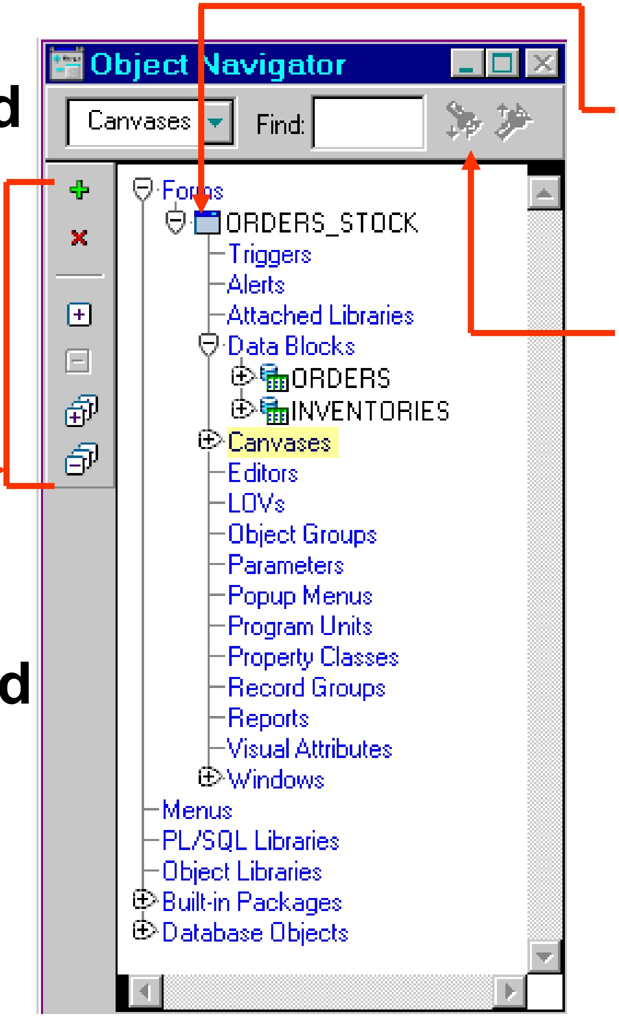
Forms Builder Key Features

With Forms Builder you can:

- Provide an interface for users to insert, update, delete, and query data
- Present data as text, image, and custom controls
- Control forms across several windows and database transactions
- Use integrated menus
- Send data to Oracle Reports

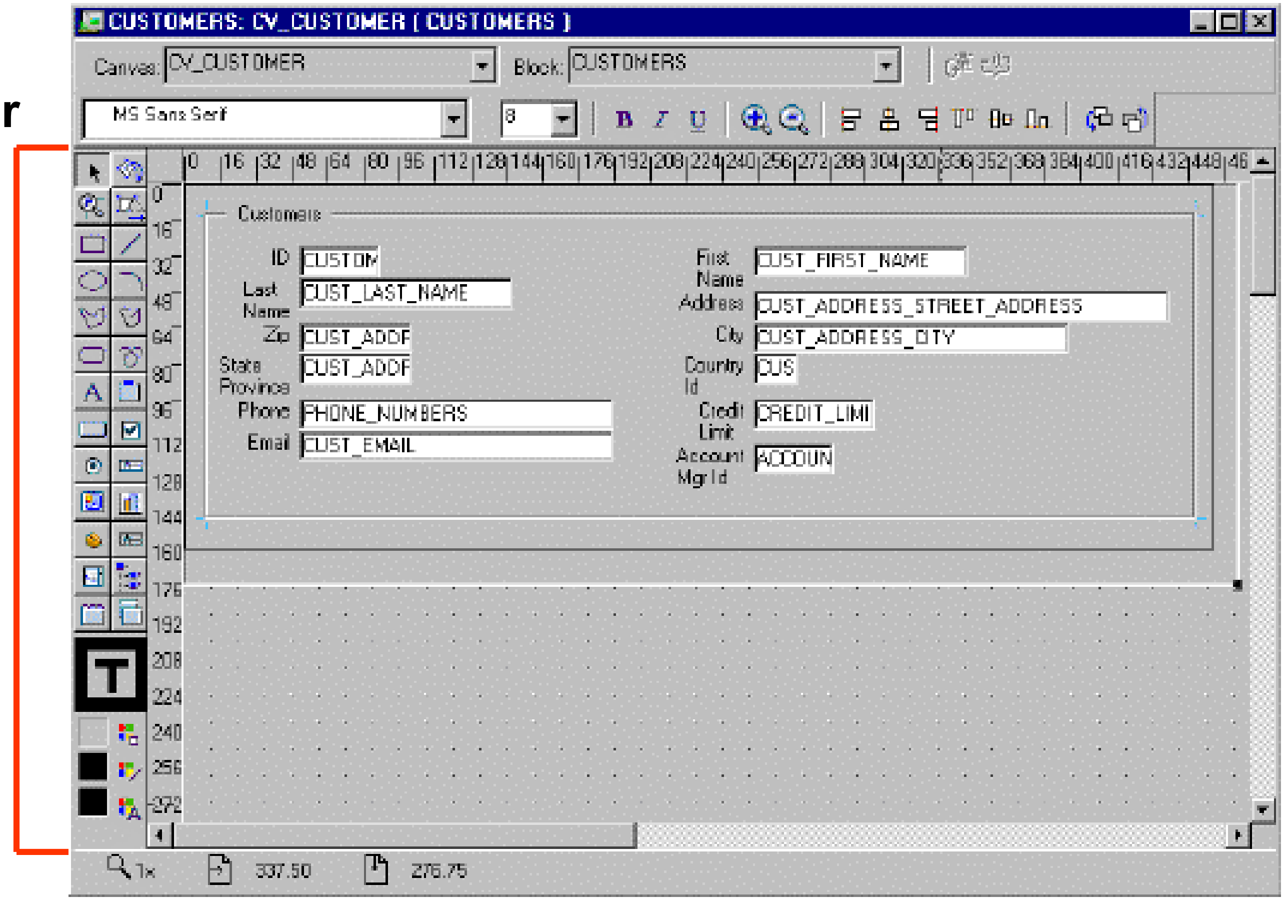
Forms Builder Components: Object Navigator

- Client-side and server-side objects displayed hierarchically
- Toolbar to create, delete or unload, expand or contract



- Icons to represent objects
- Fast search feature

Forms Builder Components: Layout Editor



Toolbar

Tool palette

CUSTOMERS: CV_CUSTOMER (CUSTOMERS)

Canvas: CV_CUSTOMER

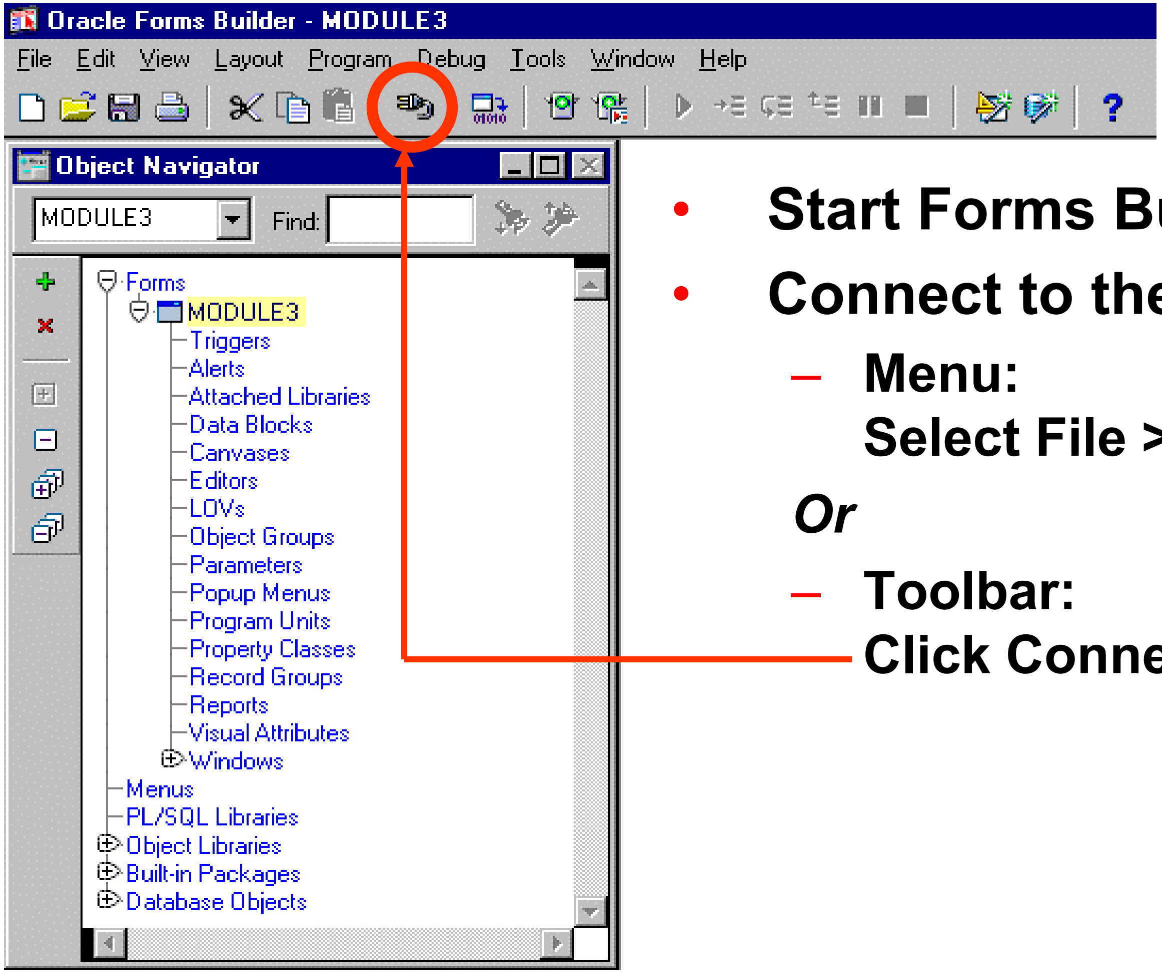
Block: CUSTOMERS

MS Sans Serif 8

Customers

ID	CUSTOM	First Name	CUST_FIRST_NAME
Last Name	CUST_LAST_NAME	Address	CUST_ADDRESS_STREET_ADDRESS
Zip	CUST_ADDF	City	CUST_ADDRESS_CITY
State Province	CUST_ADDF	Country Id	CUS
Phone	PHONE_NUMBERS	Credit Limit	CREDIT_LIMI
Email	CUST_EMAIL	Account Mgr Id	ACCOUN

Getting Started in the Forms Builder Interface

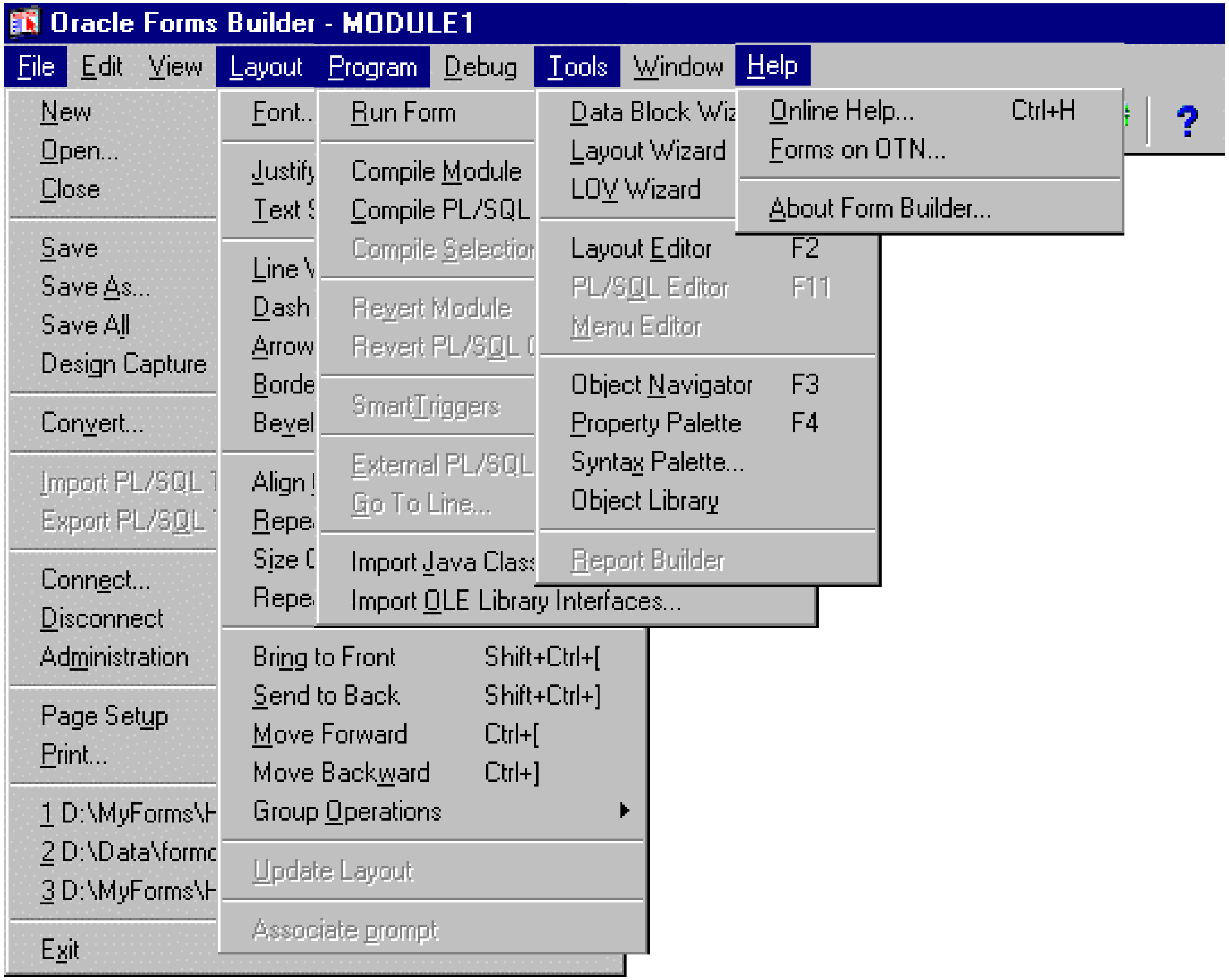


- Start Forms Builder
- Connect to the database:
  - Menu:
    Select File > Connect

  Or

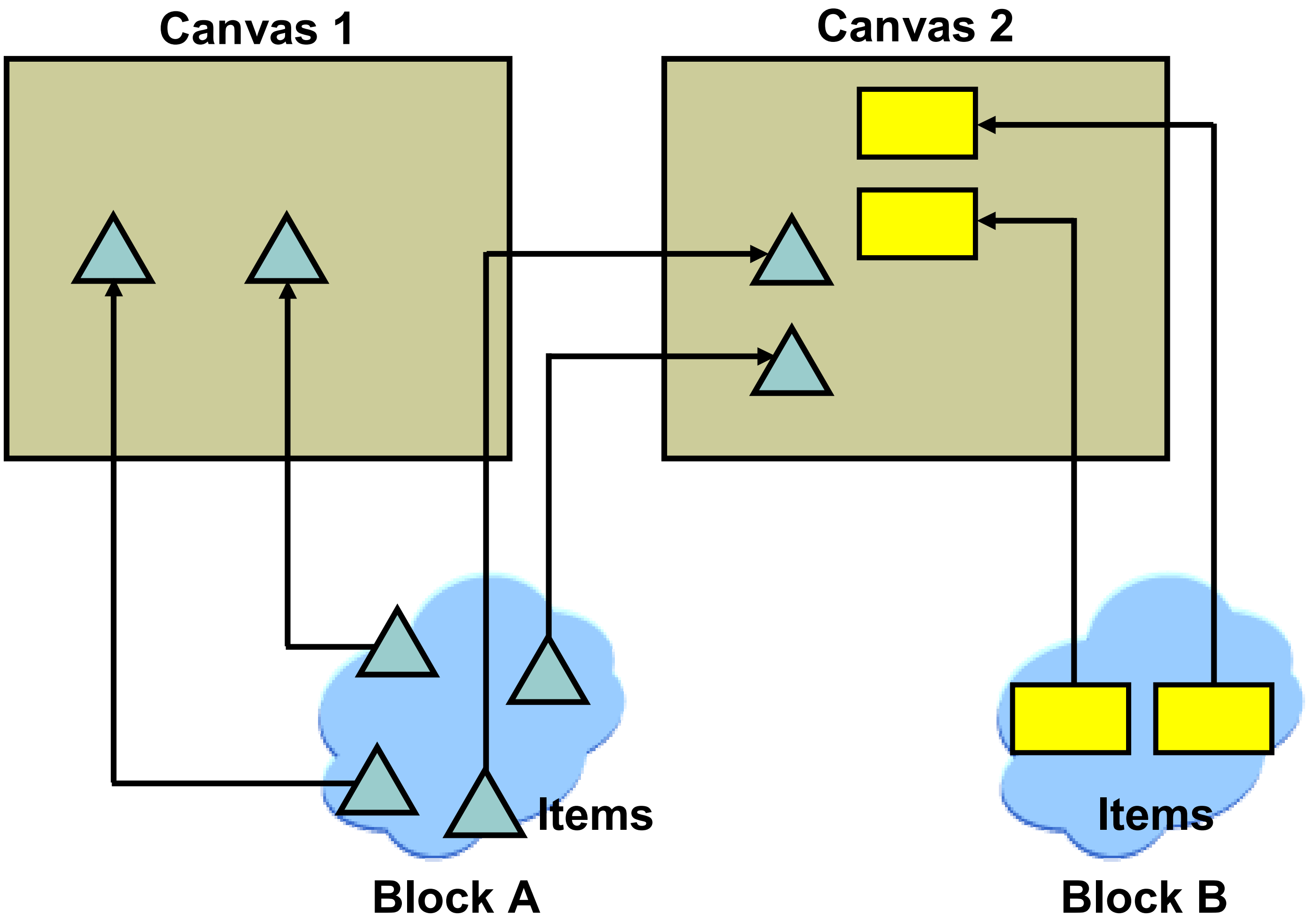
  - Toolbar:
    Click Connect

Forms Builder: Menu Structure

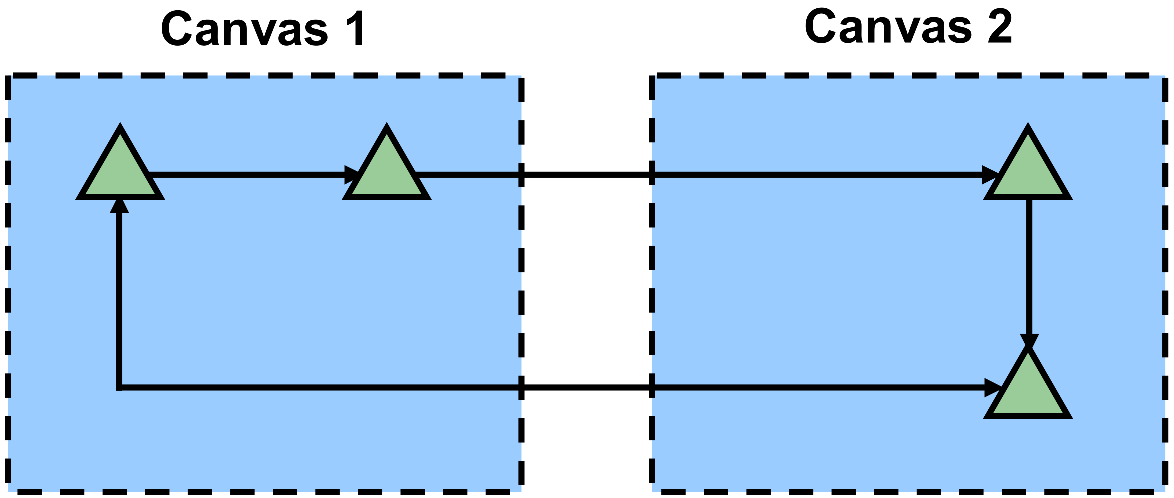


ORACLE

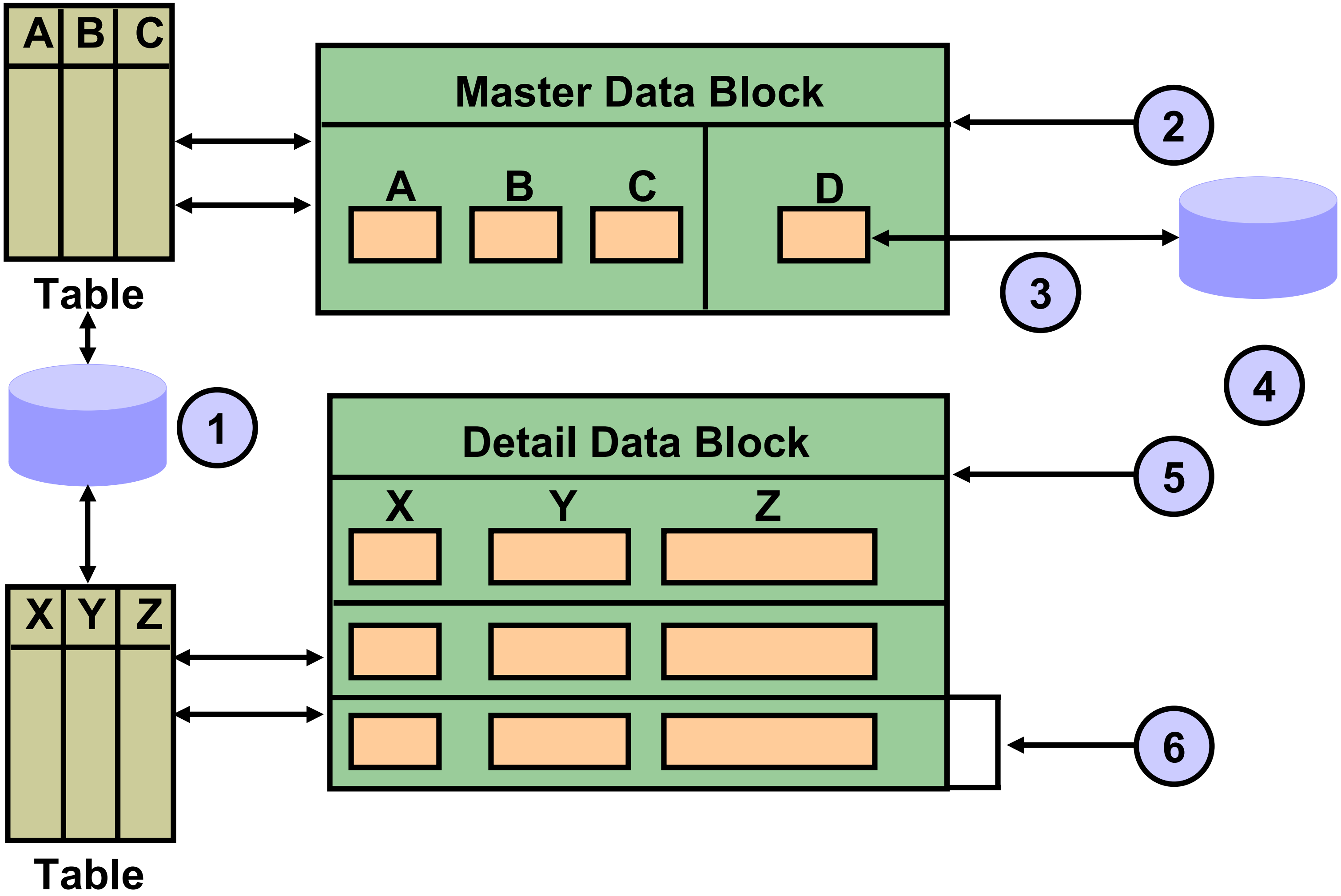
Blocks, Items, and Canvases



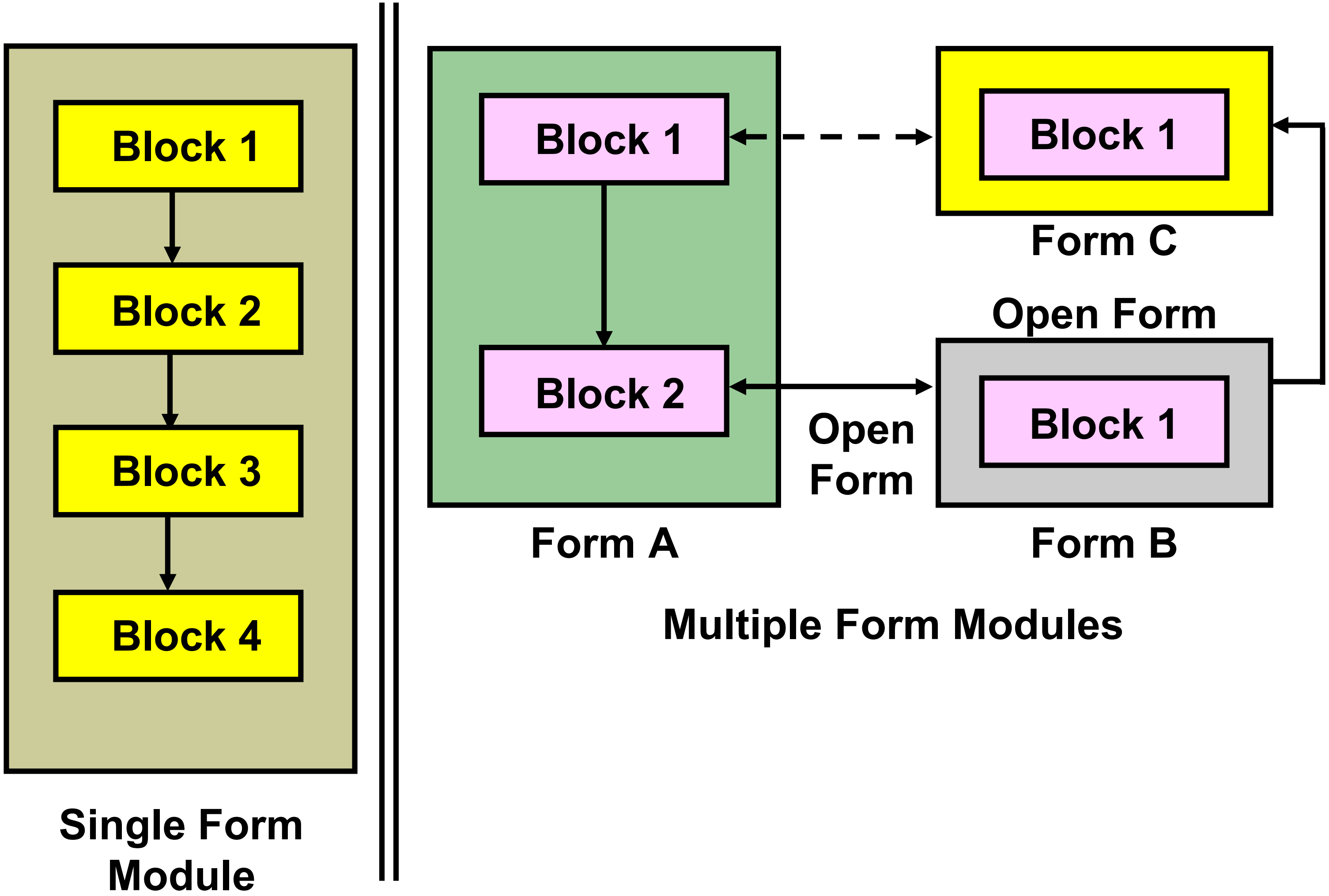
Navigation in a Block



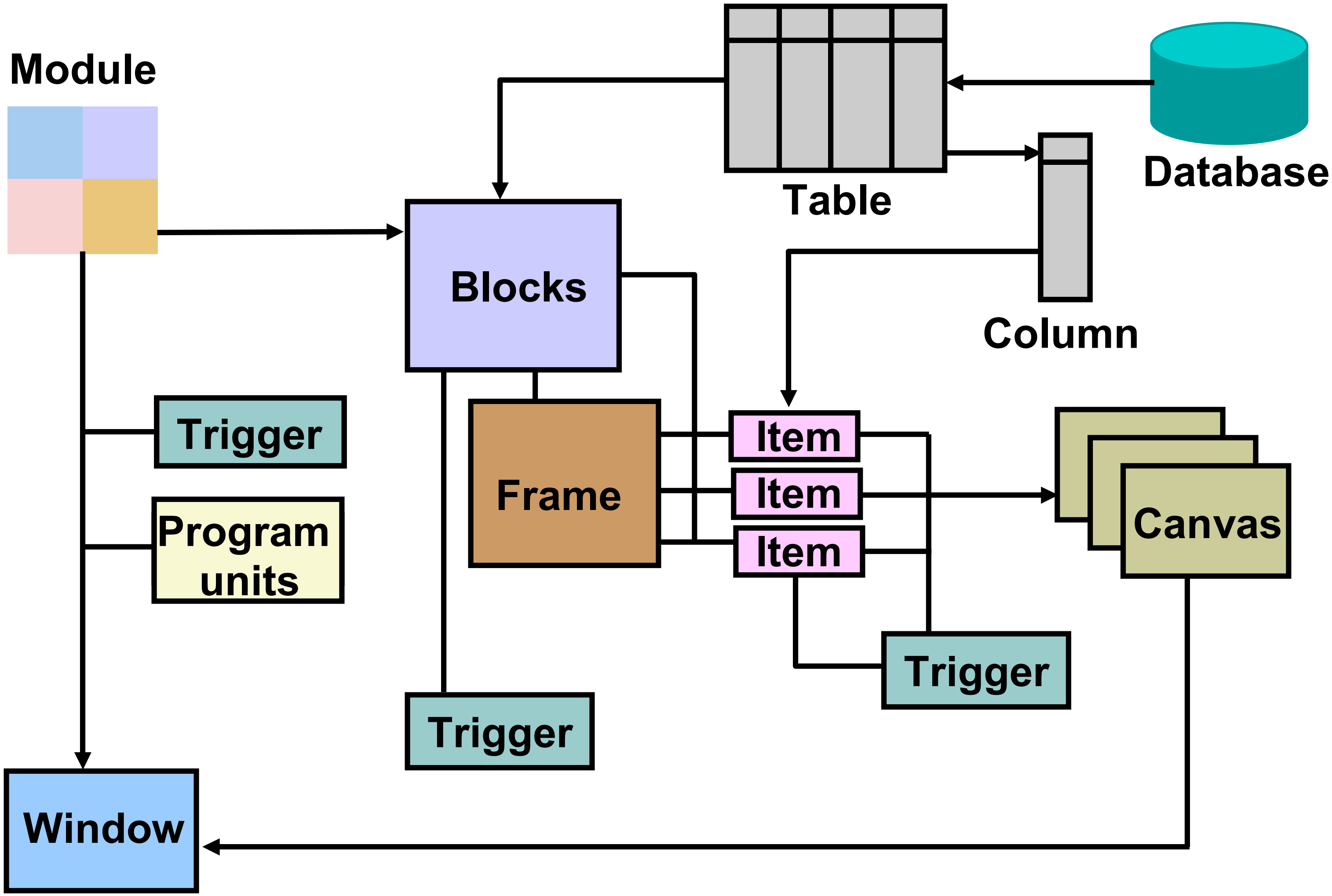
Data Blocks


A
B
C
Table
Master Data Block
A
B
C
D
2
3
4
1
Detail Data Block
X
Y
Z
5
6
X
Y
Z
Table

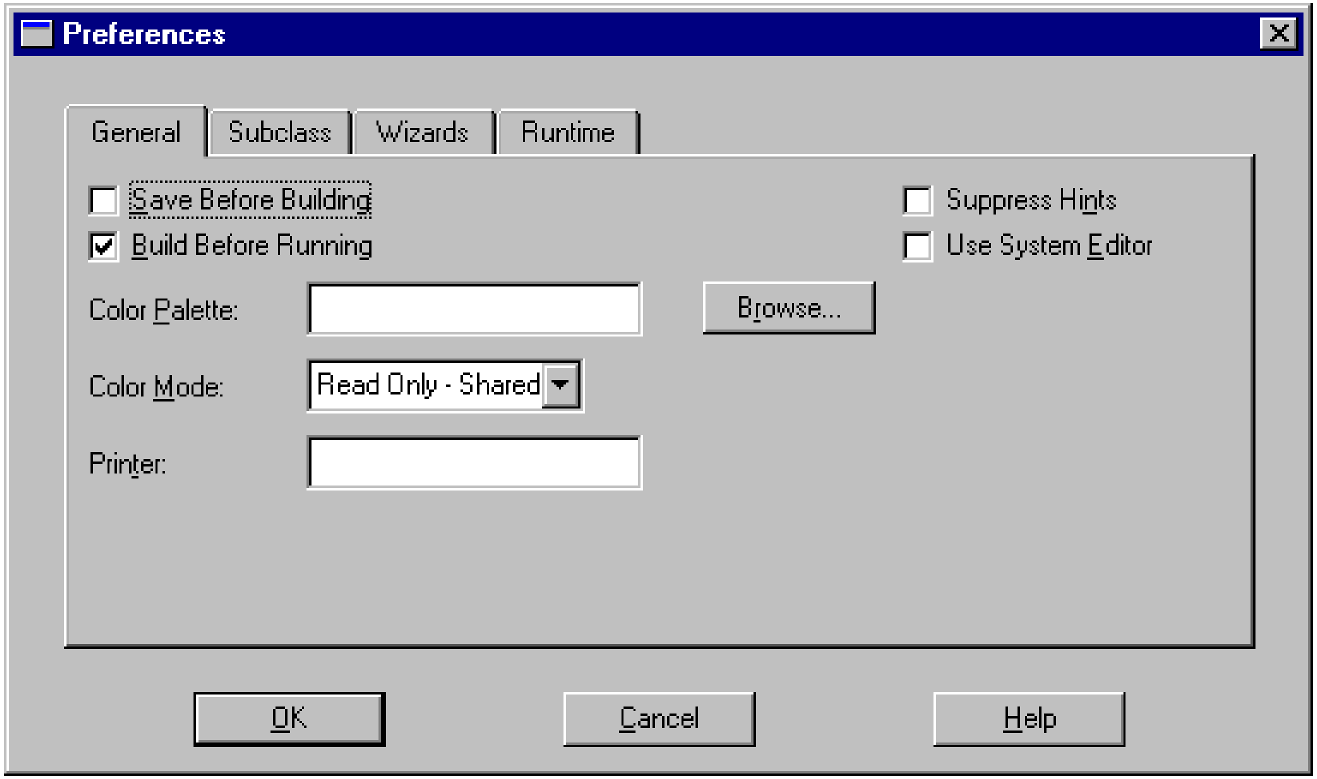
Forms and Data Blocks



Form Module Hierarchy



Customizing Your Forms Builder Session



Preferences

☐ Save Before Building

☑ Build Before Running

☐ Suppress Hints

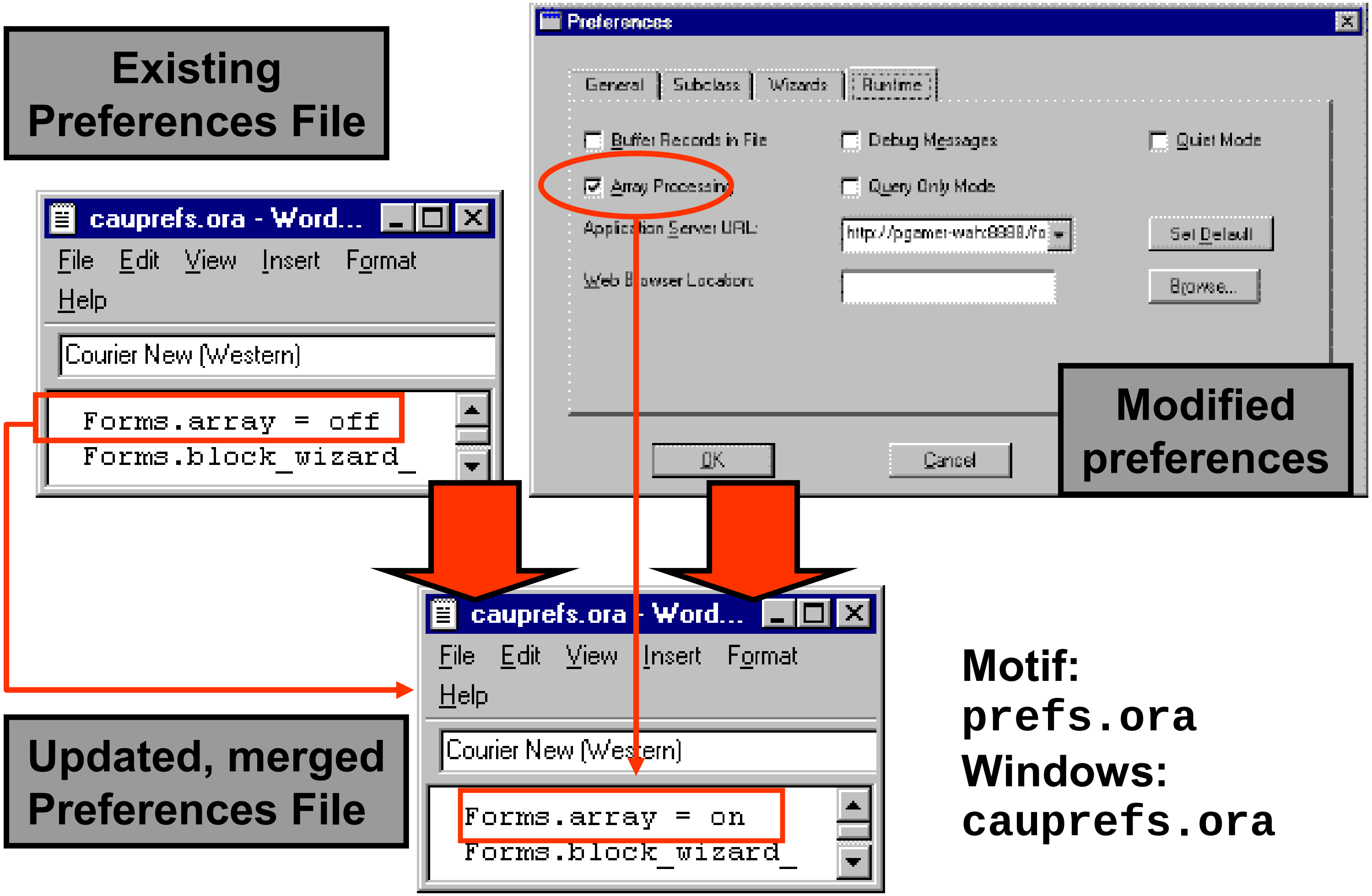
☐ Use System Editor

Color Palette:

Color Mode: Read Only - Shared

Printer:

Saving Preferences


Existing Preferences File

File Edit View Insert Format Help

```
Forms.array = off
Forms.block_wizard_
```

General Subclass Wizards Runtime

□ Buffer Records in File
□ Debug Messages
□ Quiet Mode
☑ Array Processing
□ Query Only Mode

Application Server URL: http://pgamer-wah:8888/fo

Web Browser Location:

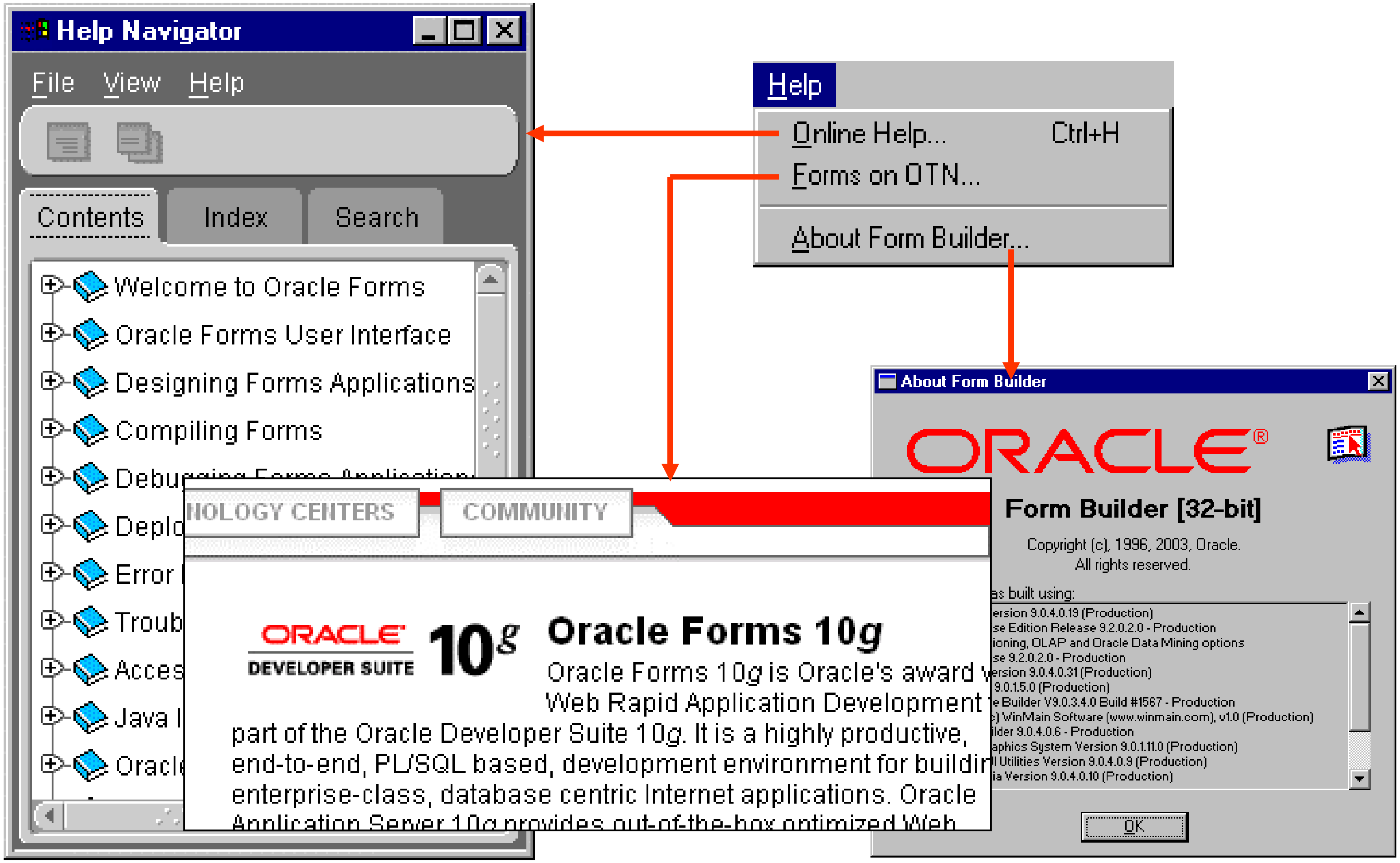
File Edit View Insert Format Help

```
Forms.array = on
Forms.block_wizard_
```

Updated, merged Preferences File

Motif:
`prefs.ora`
Windows:
`cauprefs.ora`

Using the Online Help System



Oracle Forms 10*g*

Oracle Forms 10*g* is Oracle's award w
Web Rapid Application Development
part of the Oracle Developer Suite 10*g*. It is a highly productive,
end-to-end, PL/SQL based, development environment for buildir
enterprise-class, database centric Internet applications. Oracle
Application Server 10*g* provides out-of-the-box optimized Web

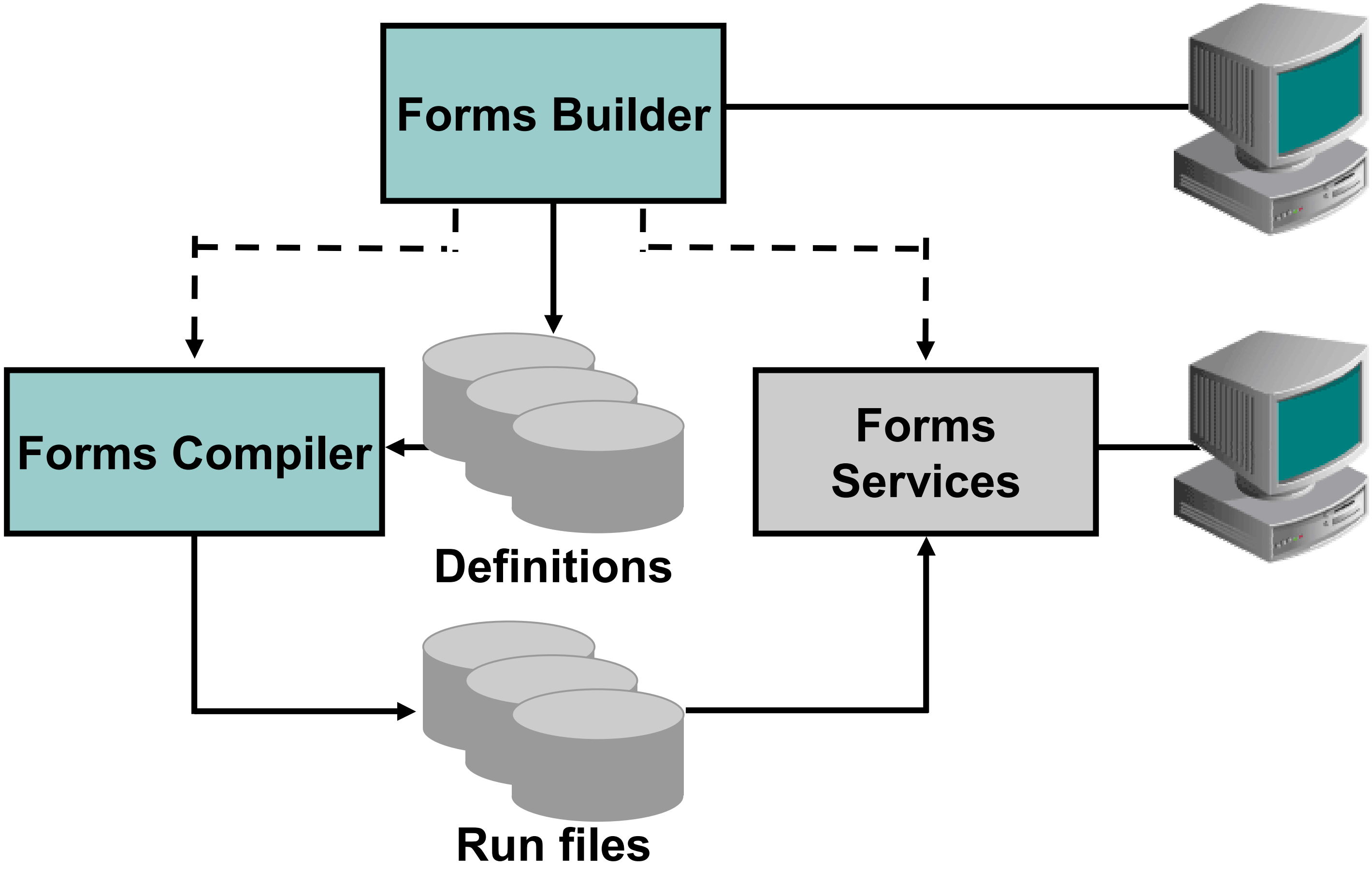
Form Builder [32-bit]

Copyright (c), 1996, 2003, Oracle.
All rights reserved.

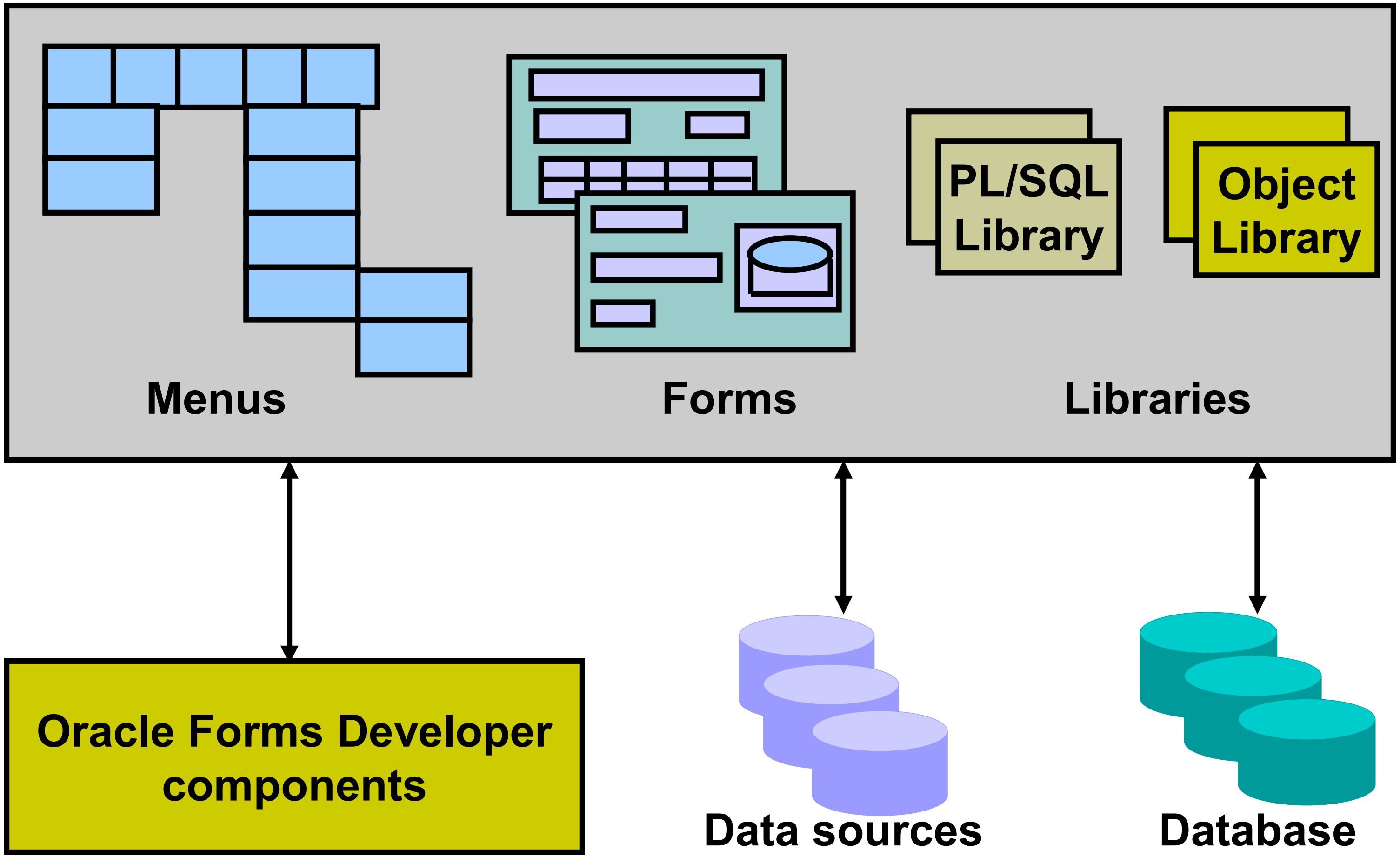
as built using:

```
ersion 9.0.4.0.19 (Production)
se Edition Release 9.2.0.2.0 - Production
ioning, OLAP and Oracle Data Mining options
se 9.2.0.2.0 - Production
version 9.0.4.0.31 (Production)
9.0.1.5.0 (Production)
e Builder V9.0.3.4.0 Build #1567 - Production
c) WinMain Software (www.winmain.com), v1.0 (Production)
ilder 9.0.4.0.6 - Production
aphics System Version 9.0.1.11.0 (Production)
l Utilities Version 9.0.4.0.9 (Production)
ia Version 9.0.4.0.10 (Production)
```

Forms Developer Executables



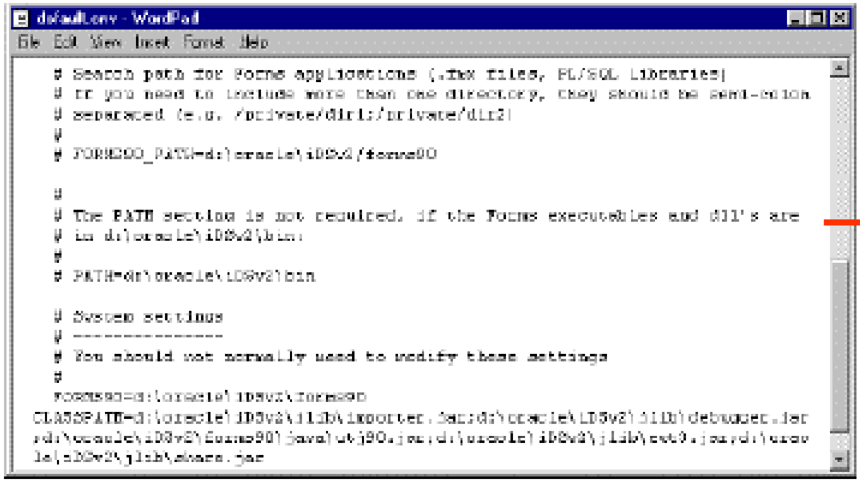
Forms Developer Module Types



Defining Forms Environment Variables for Run Time

Set on middle-tier machine (used at run time):

- `FORMS90_PATH`
- `ORACLE_PATH`
- `CLASSPATH`



```
    # Search path for Forms applications (.fmx files, PL/SQL libraries)
    # If you need to include more than one directory, they should be semi-colon
    # separated (e.g. /private/dir1;/private/dir2)
    #
    # FORMS90_PATH=d:\oracle\iDSv2/forms90

    #
    # The PATH setting is not required, if the Forms executables and dll's are
    # in d:\oracle\iDSv2\bin:
    #
    # PATH=d:\oracle\iDSv2\bin

    # System settings
    # ---------------
    # You should not normally need to modify these settings
    #
    FORMS90=d:\oracle\iDSv2\forms90
  CLASSPATH=d:\oracle\iDSv2\jlib\importer.jar;d:\oracle\iDSv2\jlib\debugger.jar
  ;d:\oracle\iDSv2\forms90\java\utj90.jar;d:\oracle\iDSv2\jlib\ewt3.jar;d:\orac
  le\iDSv2\jlib\share.jar
```

For Forms deployment, the settings in the environment control file override system settings.

Defining Forms Environment Variables for Design Time

Set on Developer Suite machine (used by Forms Builder):

- `FORMS90_BUILDER_CLASSPATH`
- `UI_ICON`
- `UI_ICON_EXTENSION`
- `FORMS90_HIDE_OBR_PARAMS`

Windows: Modify in Registry (`REGEDIT.EXE` or `REGEDT32.EXE`)

Environment Variables and Y2K Compliance

- `NLS_DATE_FORMAT`
- `FORMS90_USER_DATE_FORMAT`
- `FORMS90_USER_DATETIME_FORMAT`
- `FORMS90_OUTPUT_DATETIME_FORMAT`
- `FORMS90_OUTPUT_DATETIME_FORMAT`
- `FORMS90_ERROR_DATE_FORMAT`
- `FORMS90_ERROR_DATETIME_FORMAT`

Forms Files to Define Run-Time Environment Variables

Environment control file:

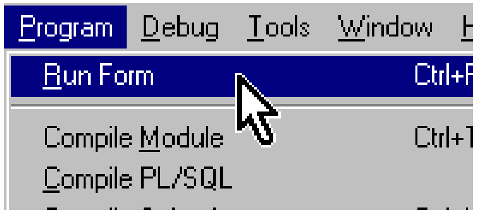
- `\forms90\server\default.env` *or*
- Other file specified in Forms configuration file

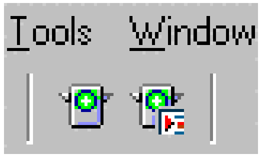
Forms configuration file:

- `\forms90\server\formsweb.cfg` or other
- Used to specify:
  - System parameters, such as `envFile` and `workingDirectory`
  - User parameters, such as form and user ID
  - Settings for the Java client
  - Other settings

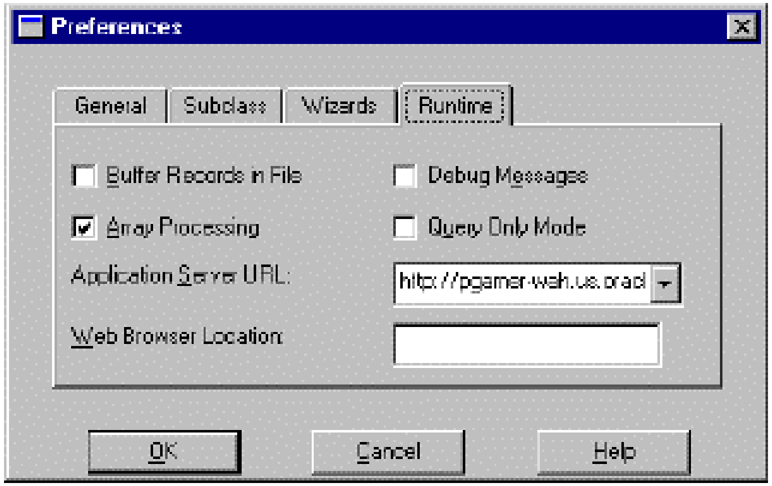
Testing a Form: The Run Form Button

- With the Run Form menu command or button, you can:
  - Run a form from Forms Builder
  - Test the form in a three-tier environment





- The Run Form command takes its settings from Preferences:
  - Edit > Preferences
  - Runtime tab
  - Set Web Browser Location if desired
  - Set Application Server URL to point to Forms Servlet: http://127.0.0.1:8889/forms90/f90servlet



□ Buffer Records in File

□ Debug Messages

☑ Array Processing

□ Query Only Mode

Application Server URL: http://pgamer-wah.us.oracl

Web Browser Location:

Summary

In this lesson, you should have learned that:

- Forms Builder includes the Object Navigator, the Property Palette, the Layout Editor, and the PL/SQL Editor
- You can use the Object Navigator or the menu and its associated toolbar icons to navigate around the Forms Builder interface
- The main objects in a form module are blocks, items, and canvases
- The Edit > Preferences dialog box enables you to customize the Forms Builder session

Summary

- The Help menu enables you to use the online help facilities to look up topics, or you can invoke context-sensitive help
- The Forms Developer executables are the Forms Builder and the Forms Compiler
- The Forms Developer module types are forms, menus, and libraries
- You can set environment variables in the Forms environment file (for run time) or on the development machine (for design time).
- You can use the Run Form button to run a form from within Forms Builder

Practice 3 Overview

This practice covers the following topics:

- Becoming familiar with the Object Navigator
- Setting Forms Builder preferences
- Using the Layout Editor to modify the appearance of a form
- Setting run-time preferences to use OC4J to test applications
- Running a form application from within Forms Builder
- Setting environment variables so the Layout Editor in Forms Builder displays `.gif` images on iconic buttons

4

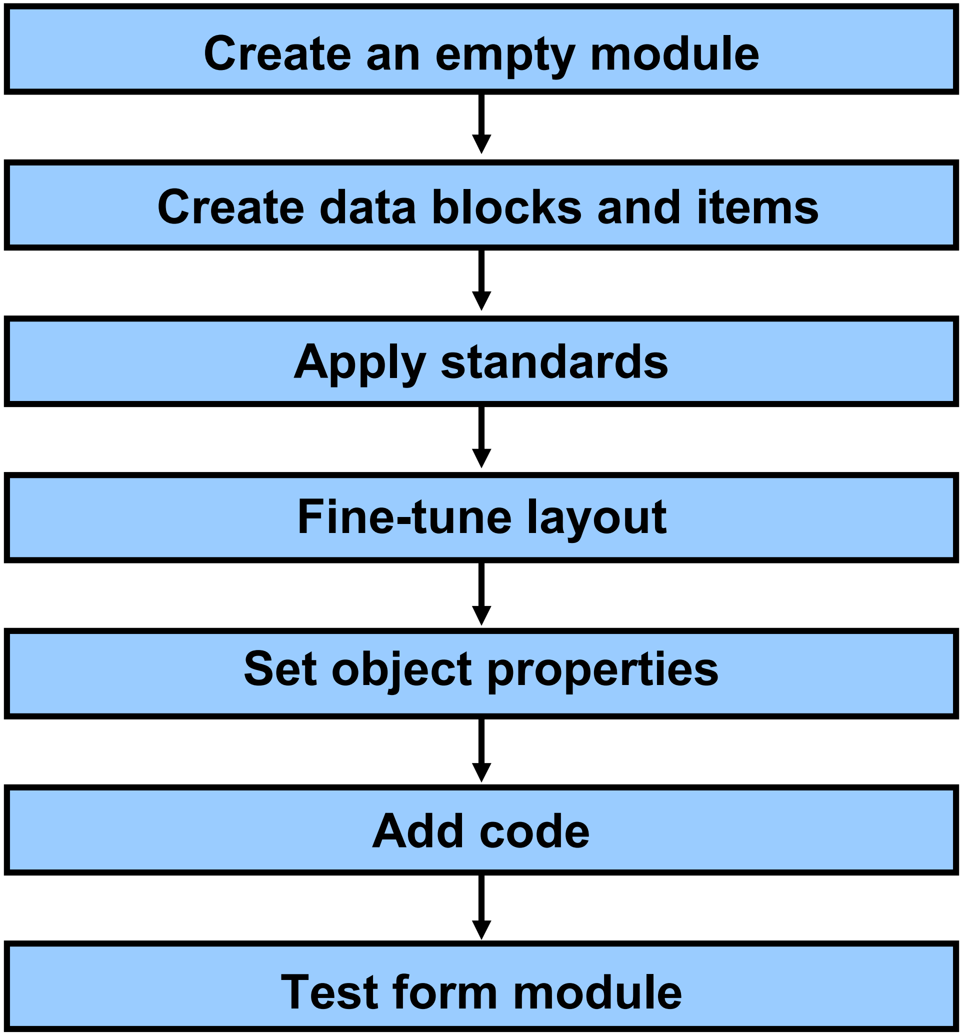
Creating a Basic Form Module

Objectives

After completing this lesson, you should be able to do the following:

- Create a form module
- Create a data block
- Save and compile a form module
- Identify Forms file formats and their characteristics
- Describe how to deploy a form module
- Explain how to create documentation for a Forms application

Creating a New Form Module



Create an empty module

Create data blocks and items

Apply standards

Fine-tune layout

Set object properties

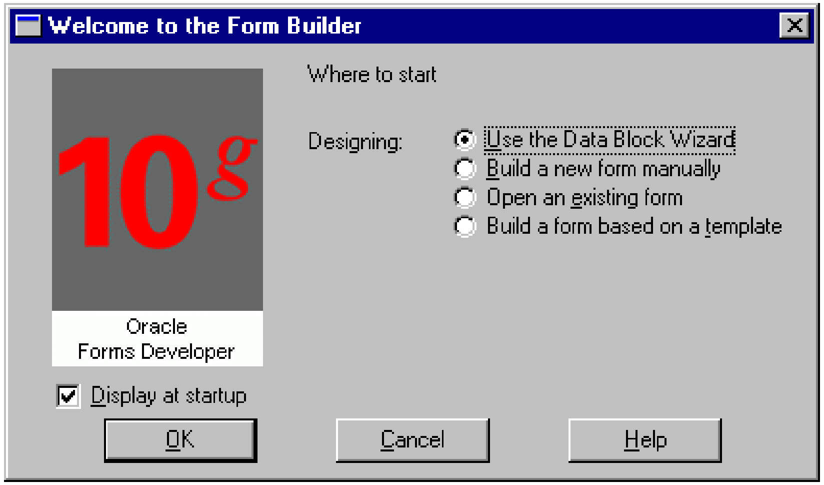
Add code

Test form module

Creating a New Form Module

Choose one of the following methods:

- Use wizards:
  - Data Block Wizard
  - Layout Wizard
- Build module manually
- Use template form



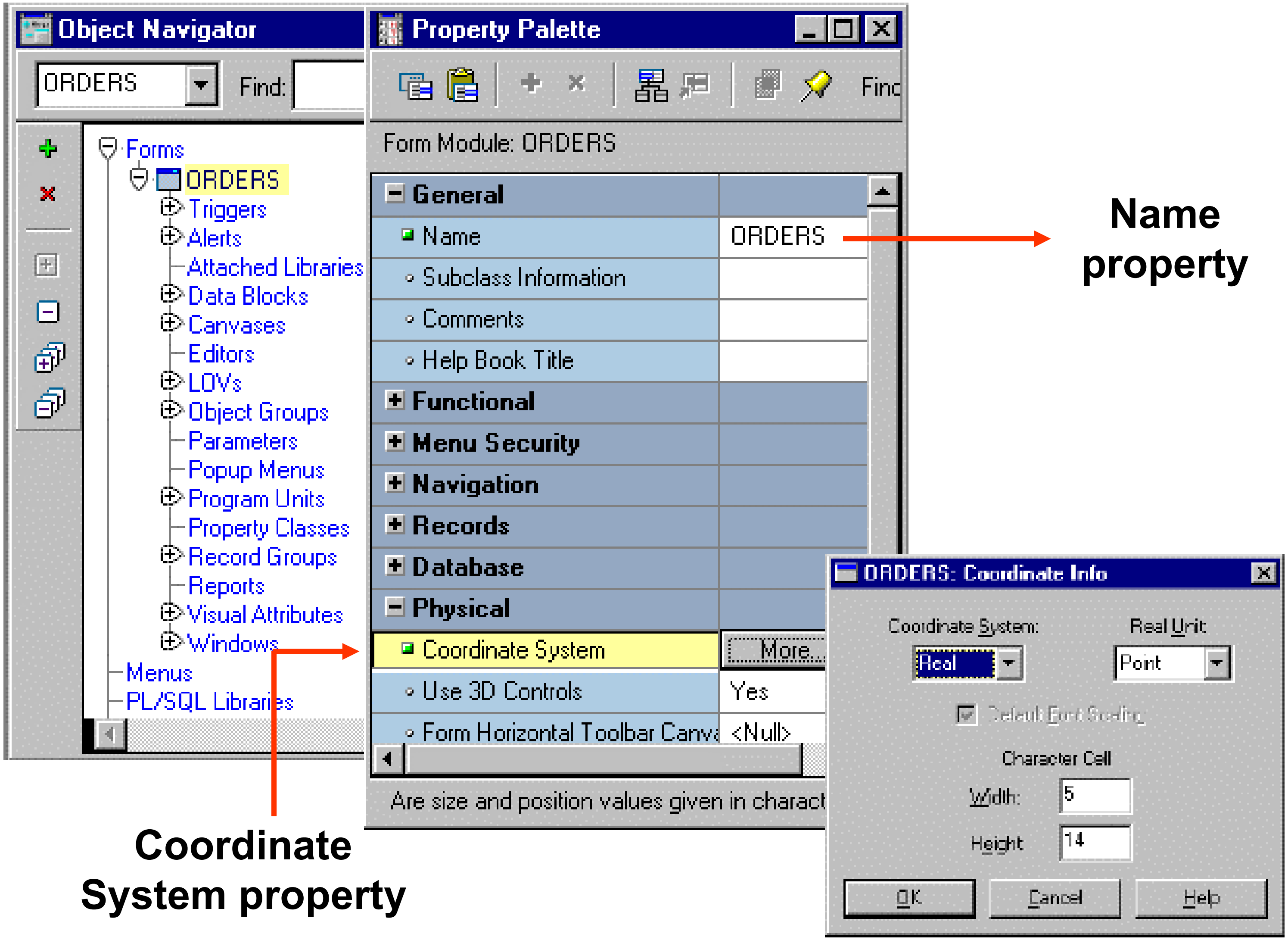
Welcome to the Form Builder

Where to start

Designing:
● Use the Data Block Wizard
○ Build a new form manually
○ Open an existing form
○ Build a form based on a template

☑ Display at startup

Form Module Properties



Form Module: ORDERS

General	
Name	ORDERS
Subclass Information	
Comments	
Help Book Title	
Functional	
Menu Security	
Navigation	
Records	
Database	
Physical	
Coordinate System	More...
Use 3D Controls	Yes
Form Horizontal Toolbar Canva	<Null>

Are size and position values given in charact

ORDERS: Coordinate Info

Coordinate System: Real

Real Unit: Point

☑ Default Font Scaling

Character Cell

Width: 5

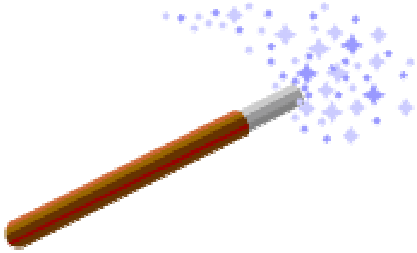
Height: 14

Name property

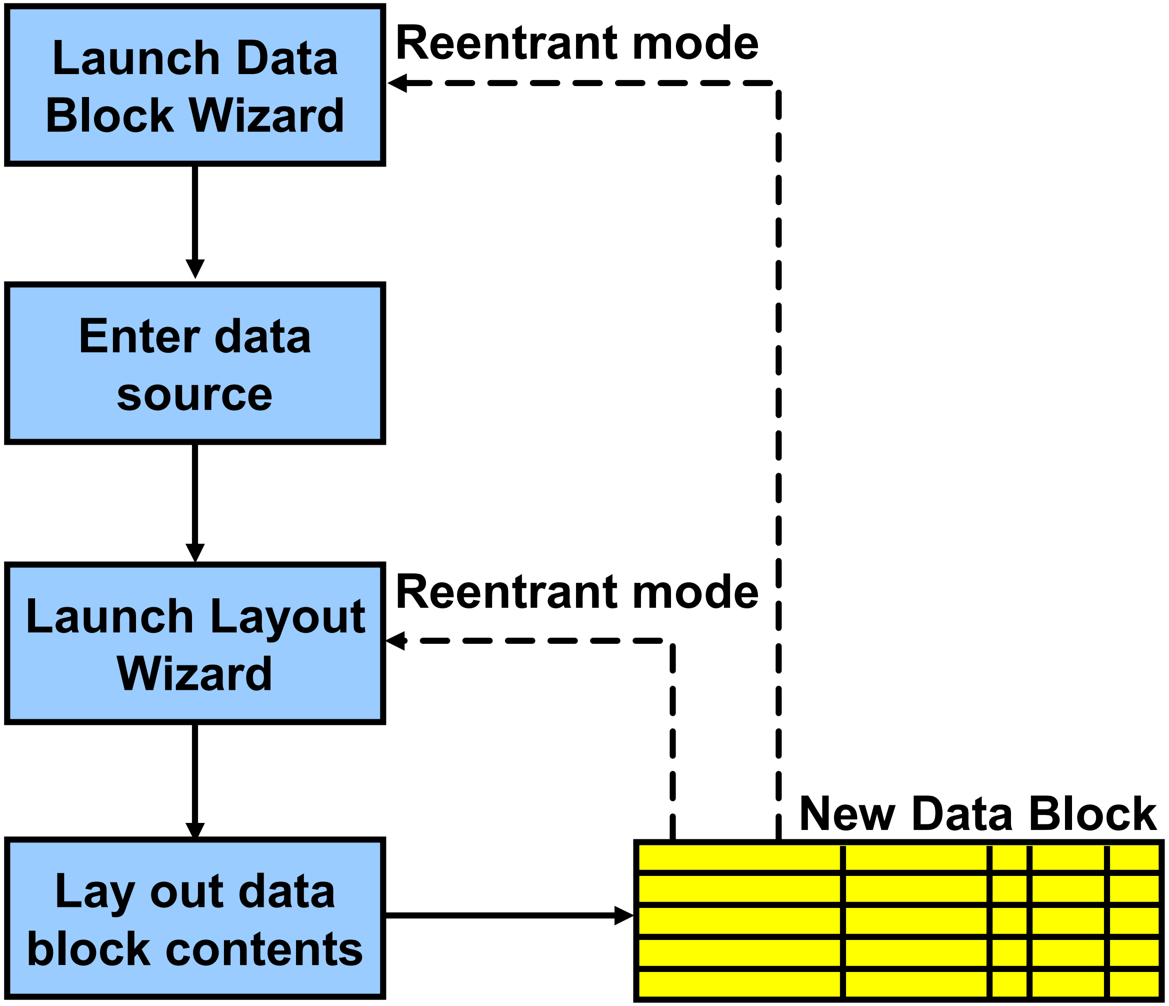
Coordinate System property

Creating a New Data Block

- Use Forms Builder Wizards:
  - Data Block Wizard: Create a data block with associated data source quickly and easily
  - Layout Wizard: Lay out data block contents for visual presentation
- Create manually



Creating a New Data Block



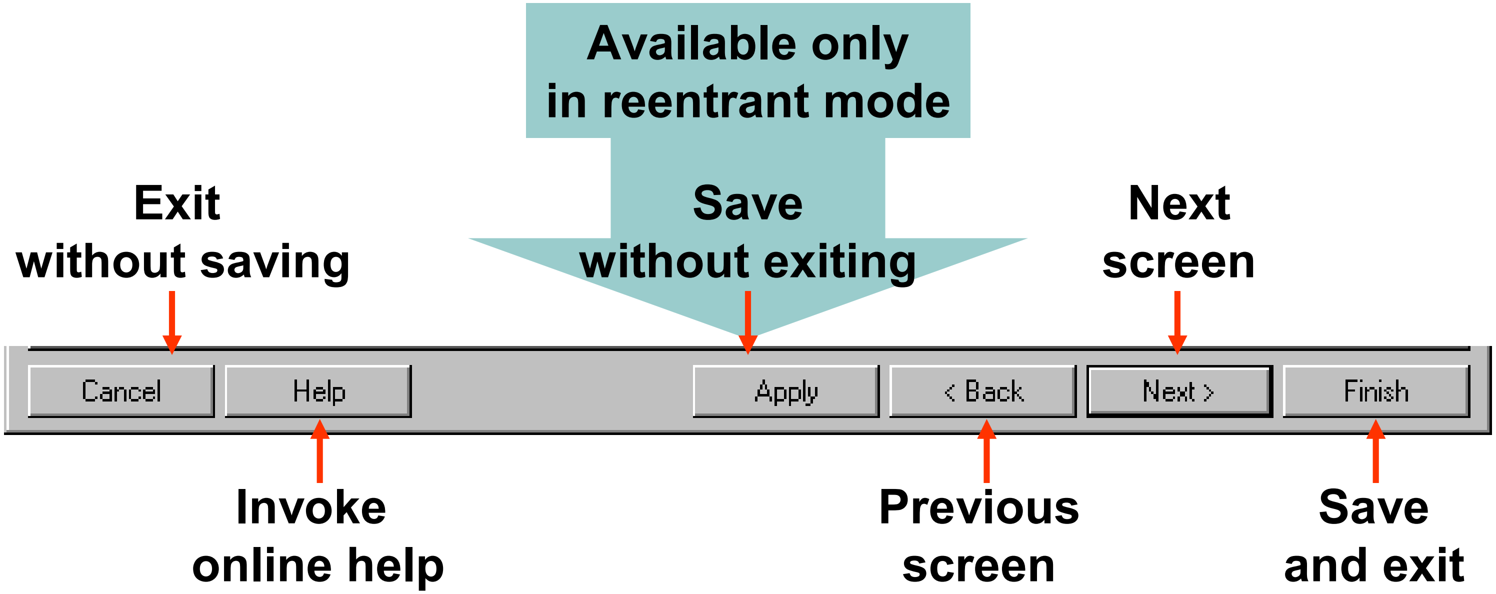
Launch Data Block Wizard

Enter data source

Launch Layout Wizard

Lay out data block contents

Navigating the Wizards



Available only in reentrant mode

Exit without saving

Save without exiting

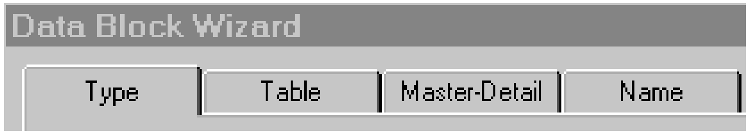
Next screen

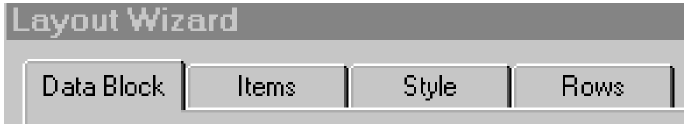
Invoke online help

Previous screen

Save and exit

Tabbed Interface:
Available only in reentrant mode

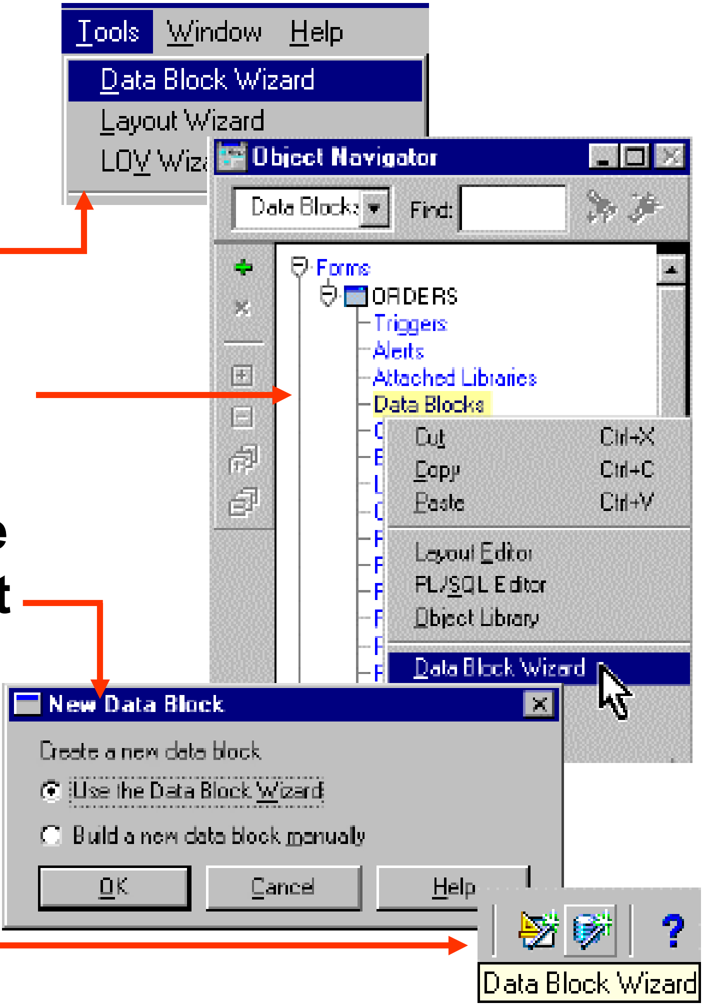




Launching the Data Block Wizard

In Forms Builder, do one of the following:

- Select Tools > Data Block Wizard.
- Right-click and select Data Block Wizard.
- Select the Data Blocks node and click Create icon; select Use the Data Block Wizard option.
- Use the Data Block Wizard button on the toolbar in the Layout Editor.

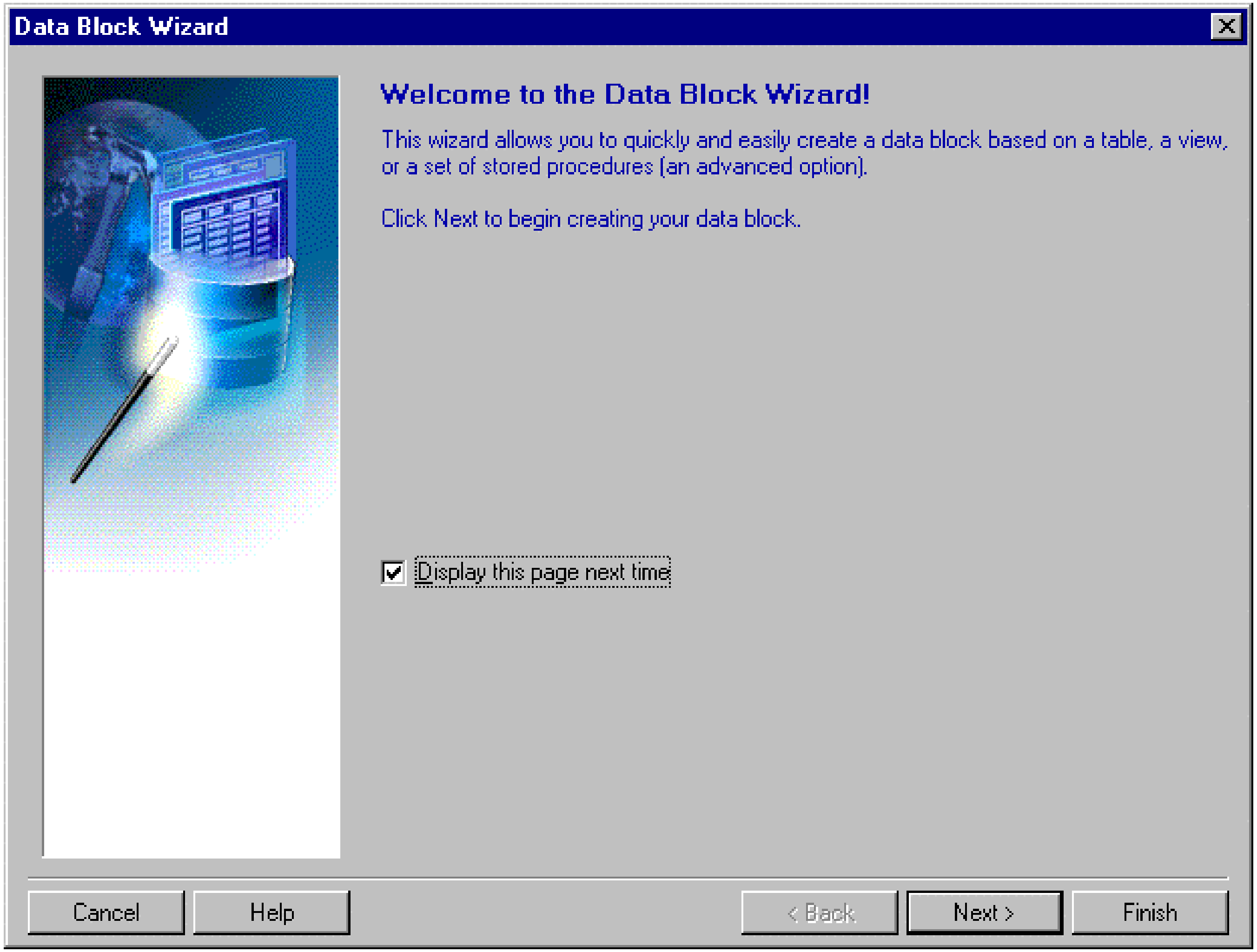


Create a new data block

● Use the Data Block Wizard

○ Build a new data block manually

Data Block Wizard: Type Page



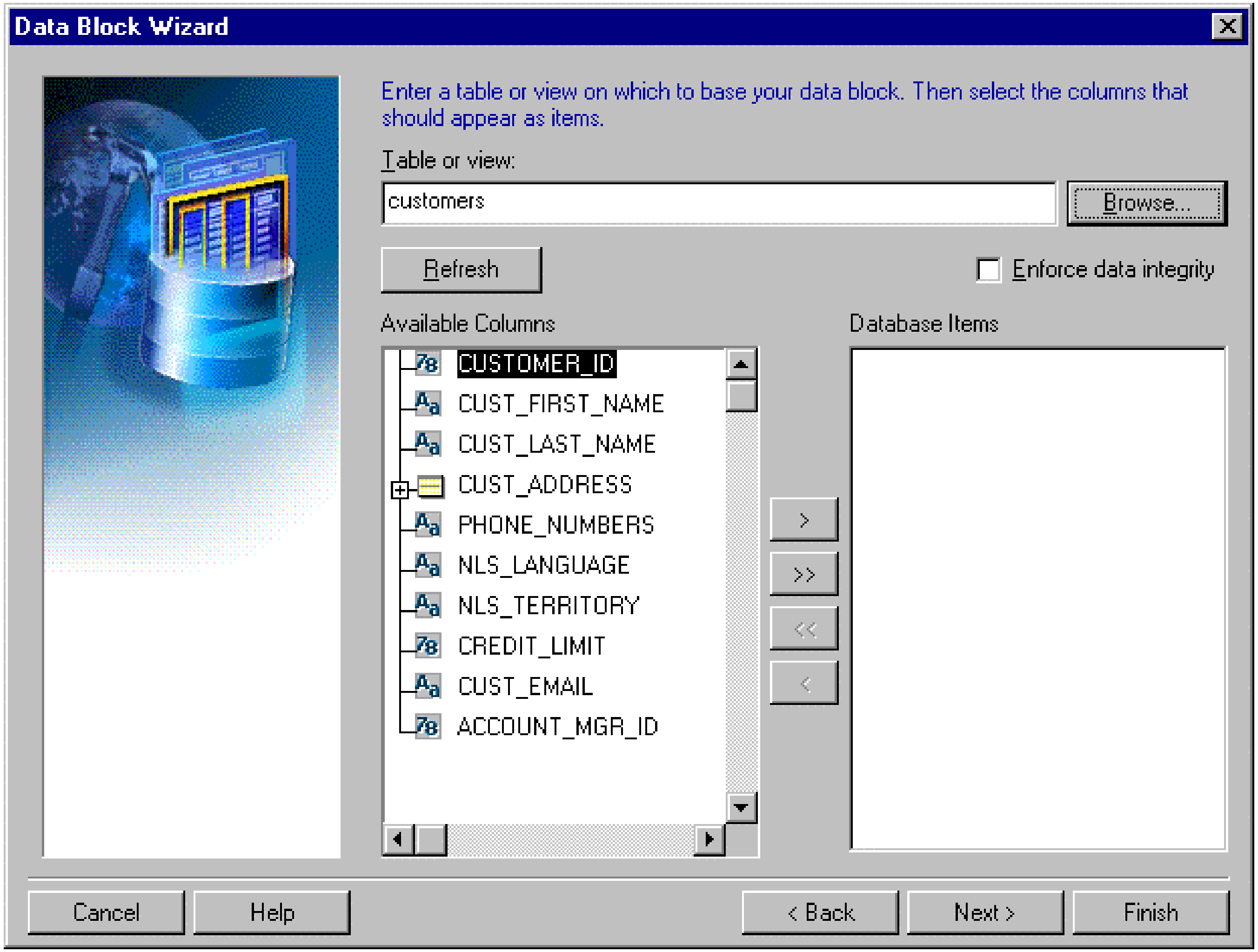
Welcome to the Data Block Wizard!

This wizard allows you to quickly and easily create a data block based on a table, a view, or a set of stored procedures (an advanced option).

Click Next to begin creating your data block.

☑ Display this page next time

Data Block Wizard: Table Page

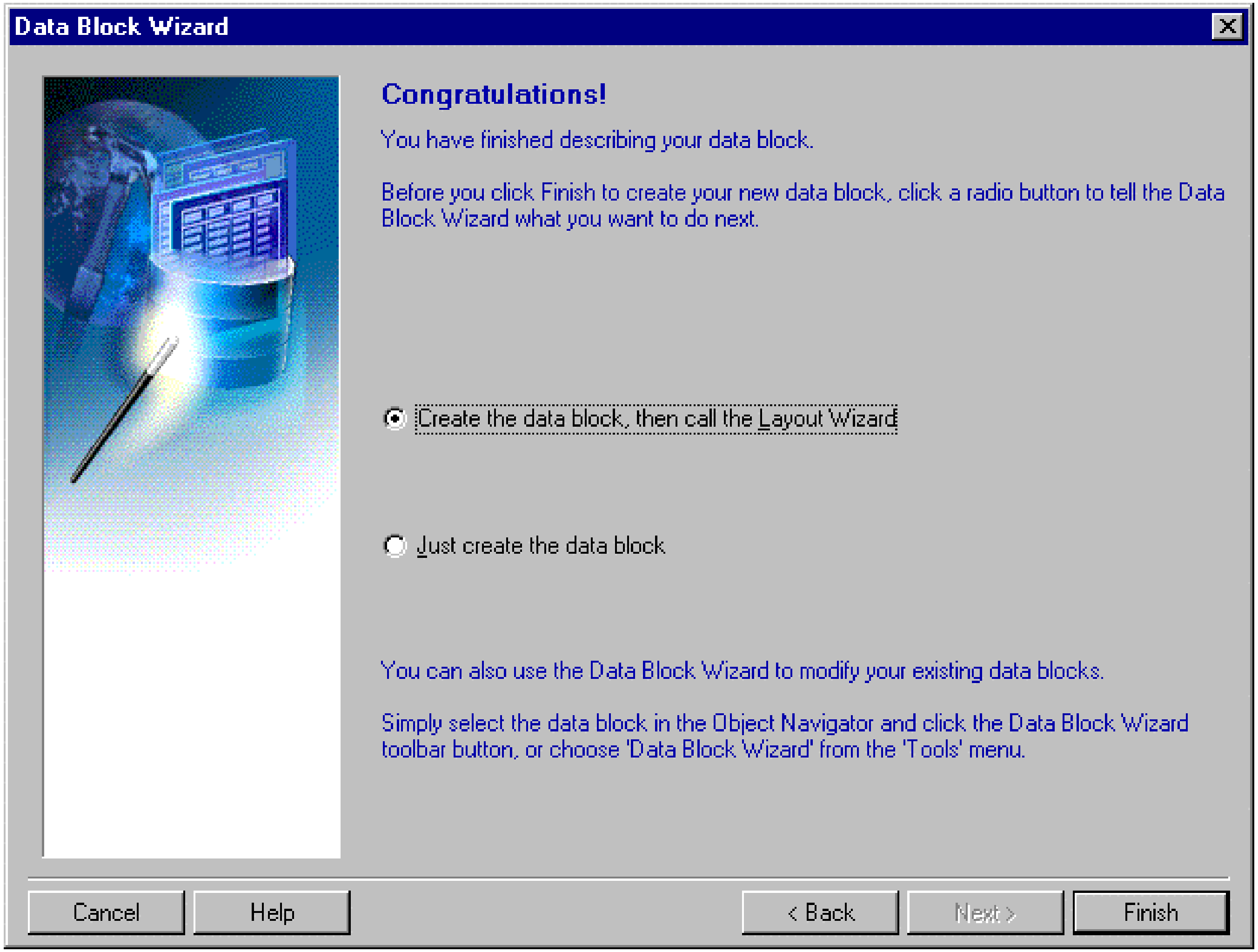


Data Block Wizard

Enter a table or view on which to base your data block. Then select the columns that should appear as items.

Table or view: customers

Data Block Wizard: Finish Page



Congratulations!

You have finished describing your data block.

Before you click Finish to create your new data block, click a radio button to tell the Data Block Wizard what you want to do next.

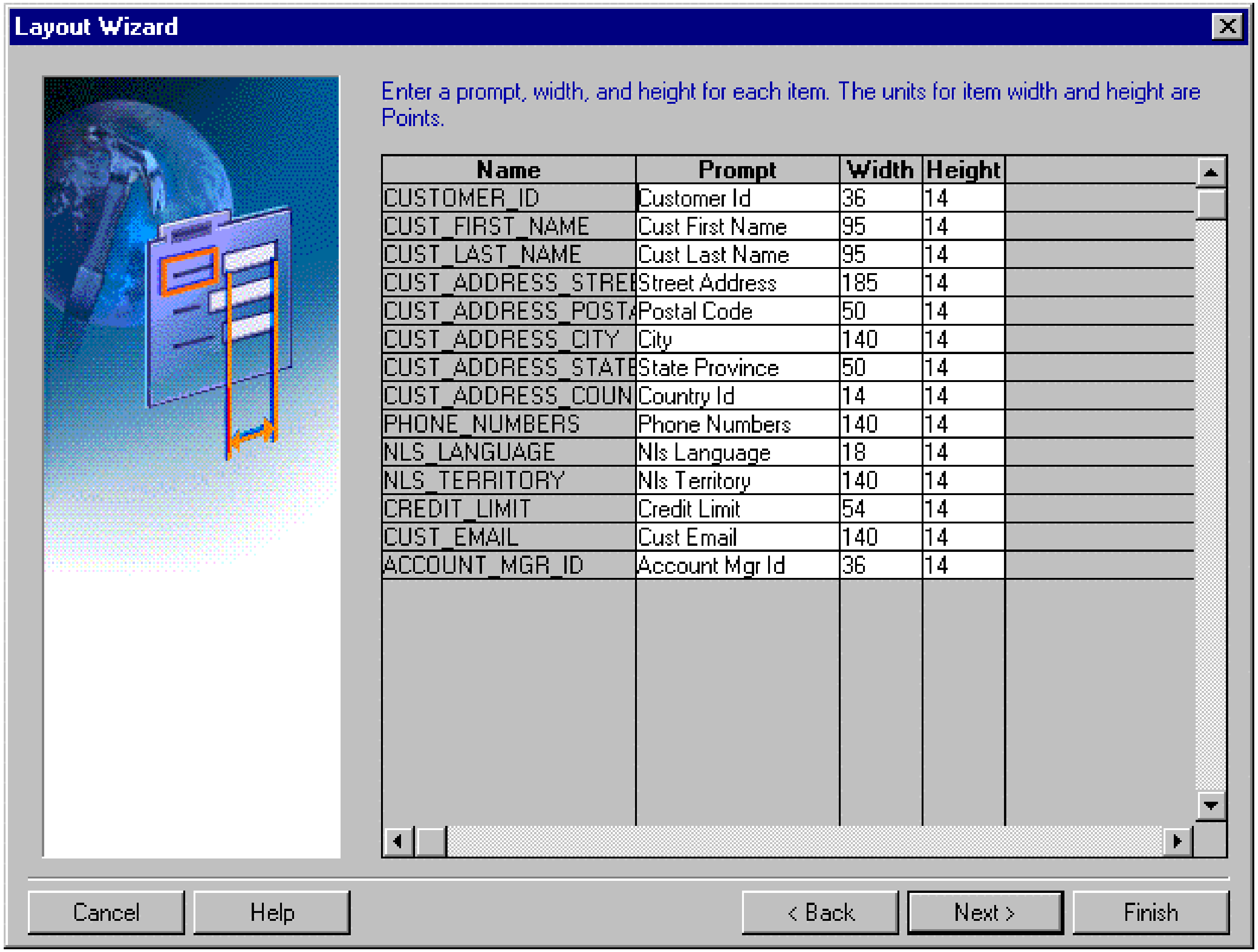
● Create the data block, then call the Layout Wizard

○ Just create the data block

You can also use the Data Block Wizard to modify your existing data blocks.

Simply select the data block in the Object Navigator and click the Data Block Wizard toolbar button, or choose 'Data Block Wizard' from the 'Tools' menu.

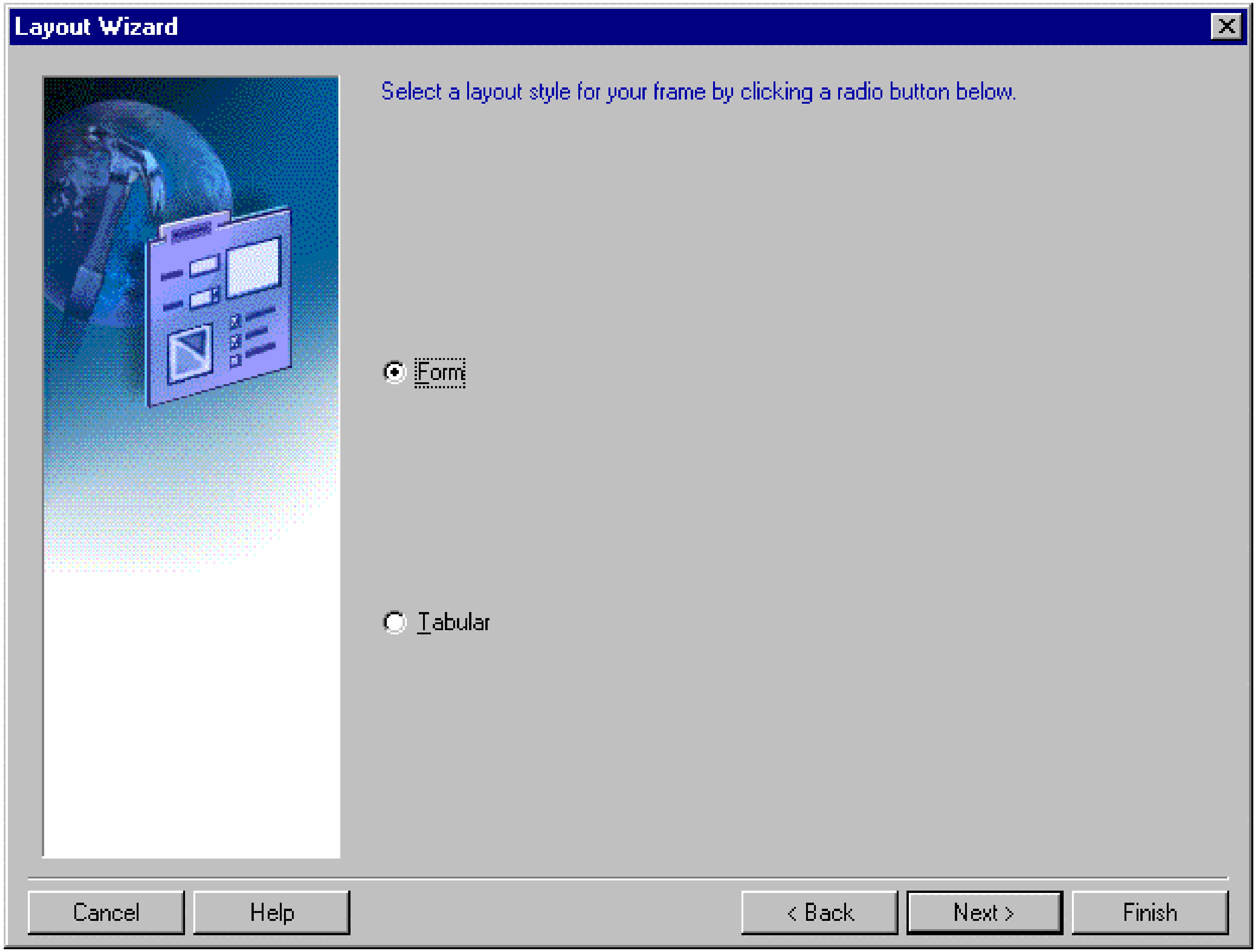
Layout Wizard: Items Page



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

Name	Prompt	Width	Height
CUSTOMER_ID	Customer Id	36	14
CUST_FIRST_NAME	Cust First Name	95	14
CUST_LAST_NAME	Cust Last Name	95	14
CUST_ADDRESS_STREE	Street Address	185	14
CUST_ADDRESS_POST/	Postal Code	50	14
CUST_ADDRESS_CITY	City	140	14
CUST_ADDRESS_STATE	State Province	50	14
CUST_ADDRESS_COUN	Country Id	14	14
PHONE_NUMBERS	Phone Numbers	140	14
NLS_LANGUAGE	Nls Language	18	14
NLS_TERRITORY	Nls Territory	140	14
CREDIT_LIMIT	Credit Limit	54	14
CUST_EMAIL	Cust Email	140	14
ACCOUNT_MGR_ID	Account Mgr Id	36	14

Layout Wizard: Style Page

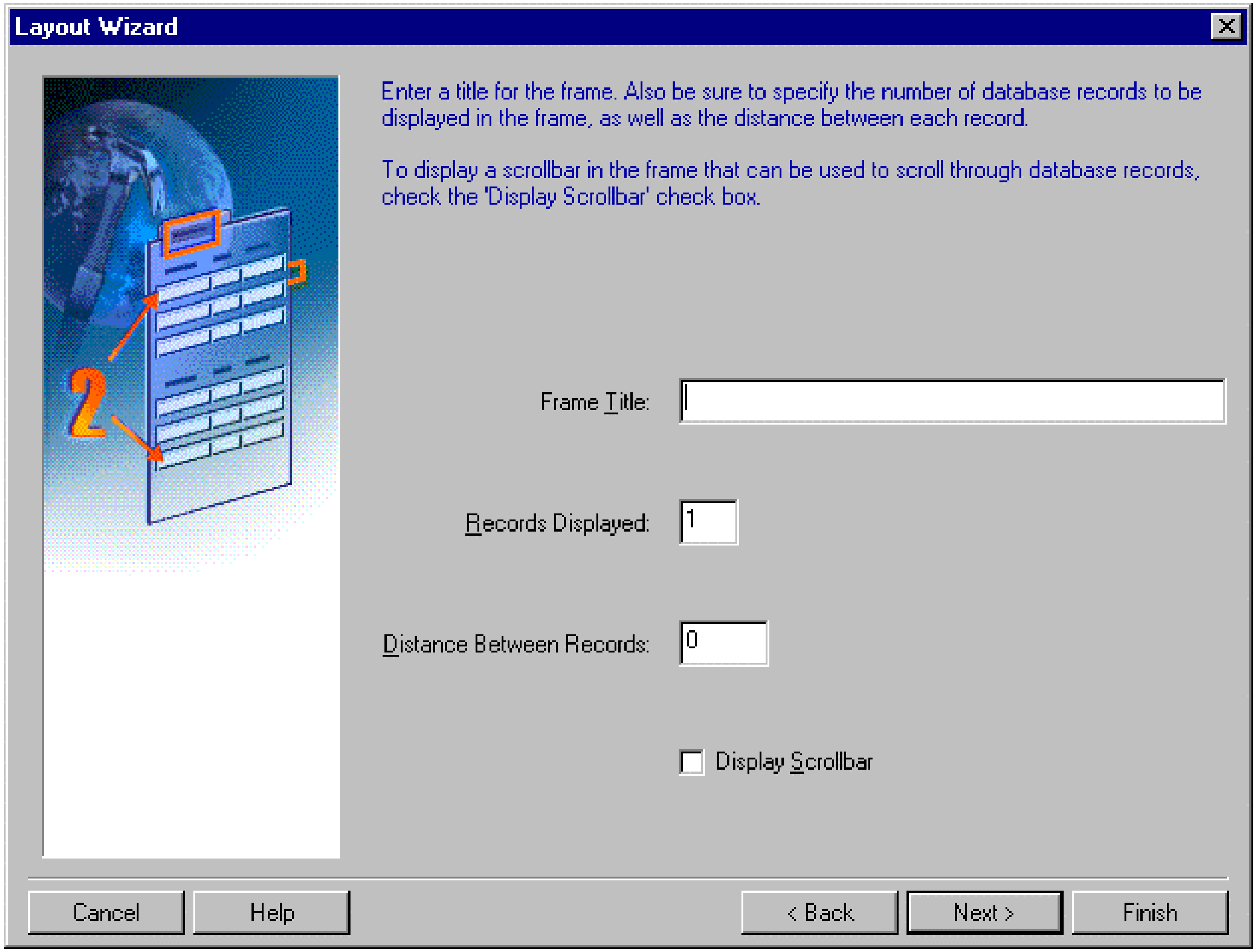


Select a layout style for your frame by clicking a radio button below.

● Form

○ Tabular

Layout Wizard: Rows Page



Layout Wizard

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

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

Frame Title:

Records Displayed: 1

Distance Between Records: 0

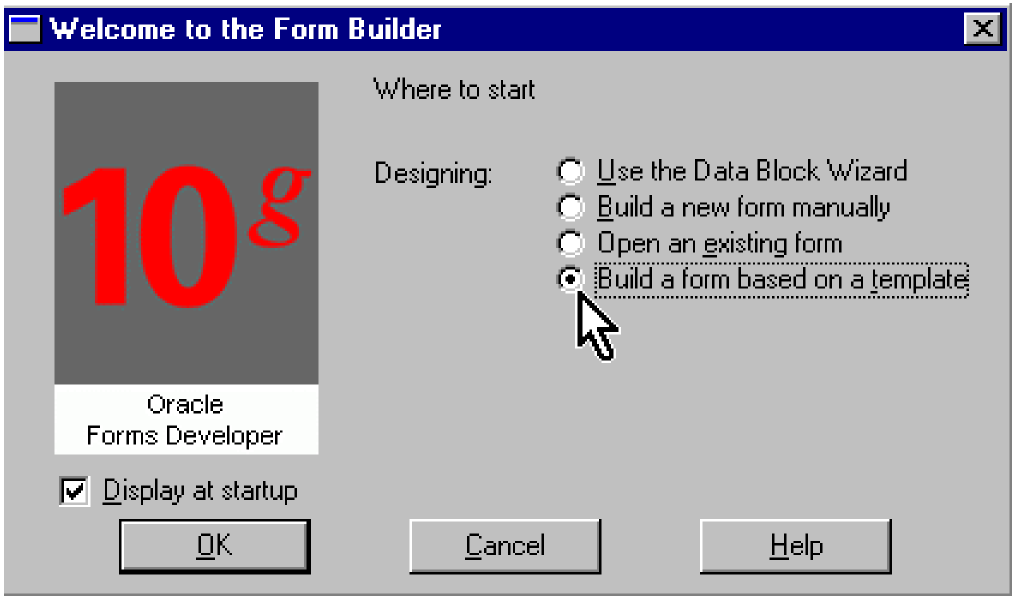
□ Display Scrollbar

Data Block Functionality

Once you create a data block with the wizards, Forms Builder automatically creates:

- A form module with database functionality including query, insert, update, delete
- A frame object
- Items in the data block
- A prompt for each item
- Triggers needed to enforce database constraints if “Enforce data integrity” is checked

Template Forms



Welcome to the Form Builder

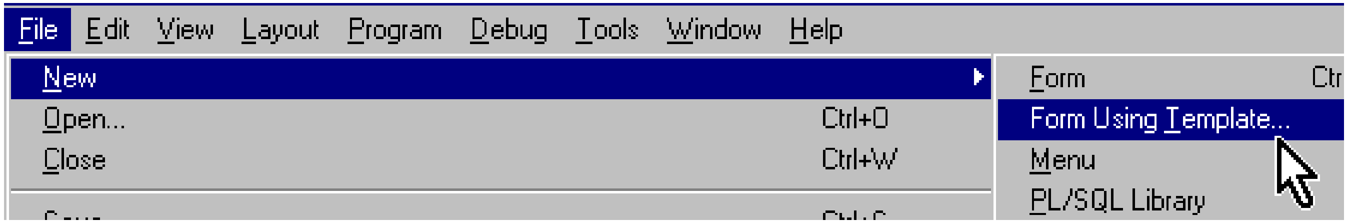
Oracle Forms Developer

Where to start

Designing:

○ Use the Data Block Wizard
○ Build a new form manually
○ Open an existing form
● Build a form based on a template

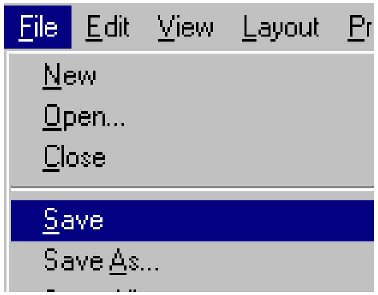
☑ Display at startup

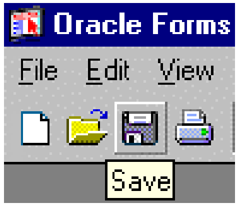


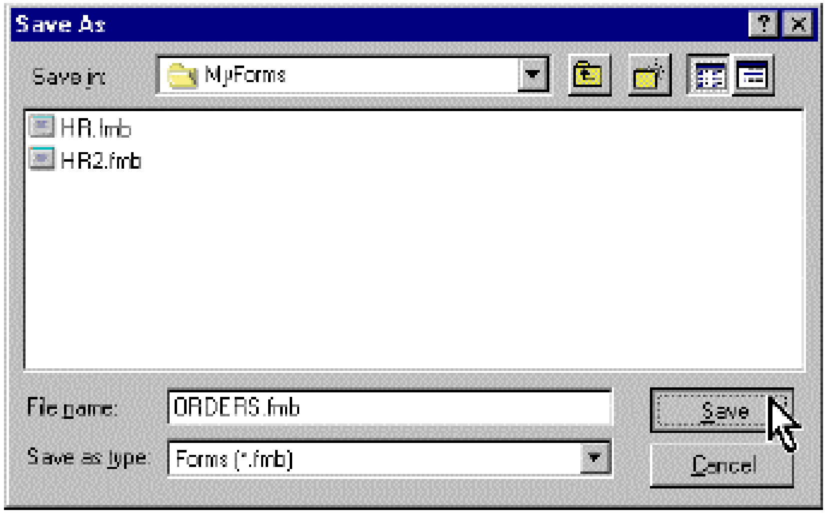
Saving a Form Module

To save the form module:

- Select File > Save
  OR
  Click the Save icon
- Enter a filename
- Navigate to desired location
- Click Save





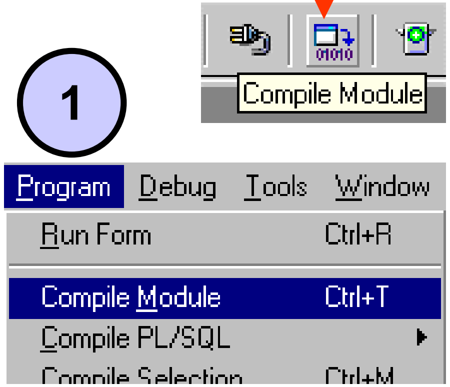


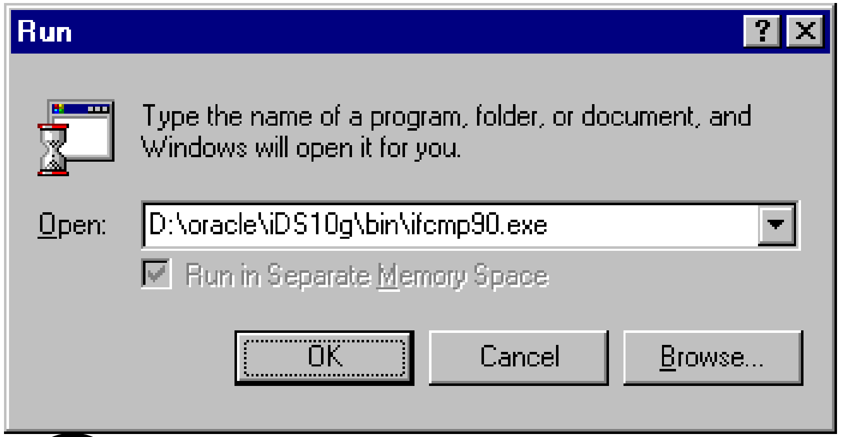
File name: ORDERS.fmb

Save as type: Forms (*.fmb)

Compiling a Form Module

1





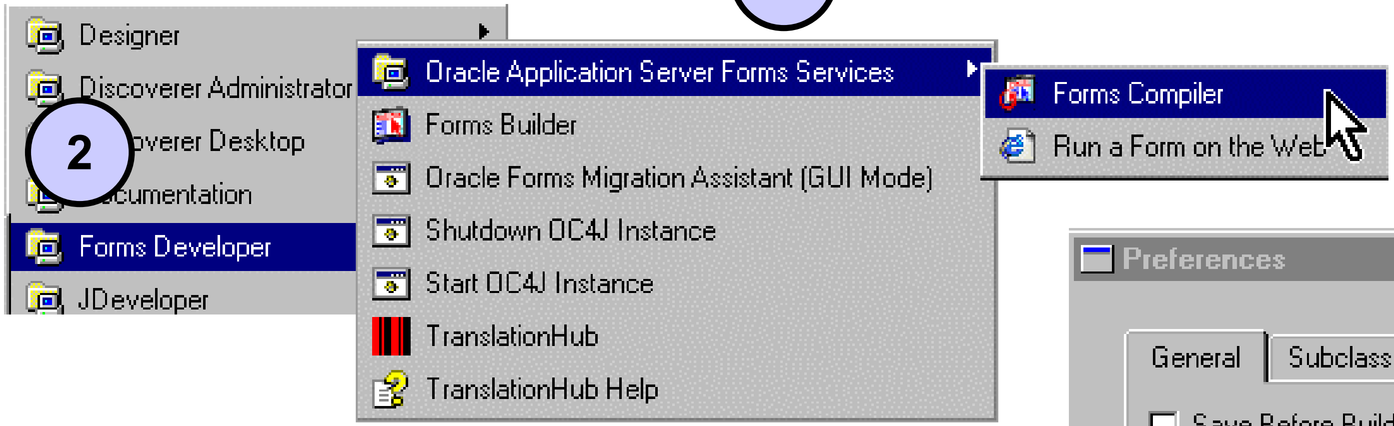
Type the name of a program, folder, or document, and Windows will open it for you.

Open: D:\oracle\iDS10g\bin\ifcmp90.exe

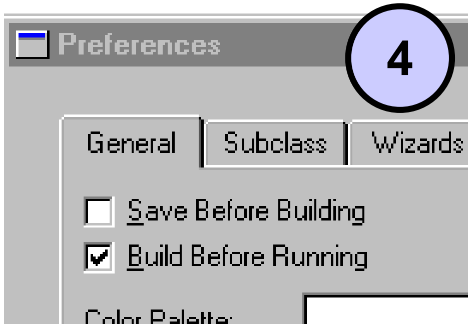
☑ Run in Separate Memory Space

3

2



4



□ Save Before Building

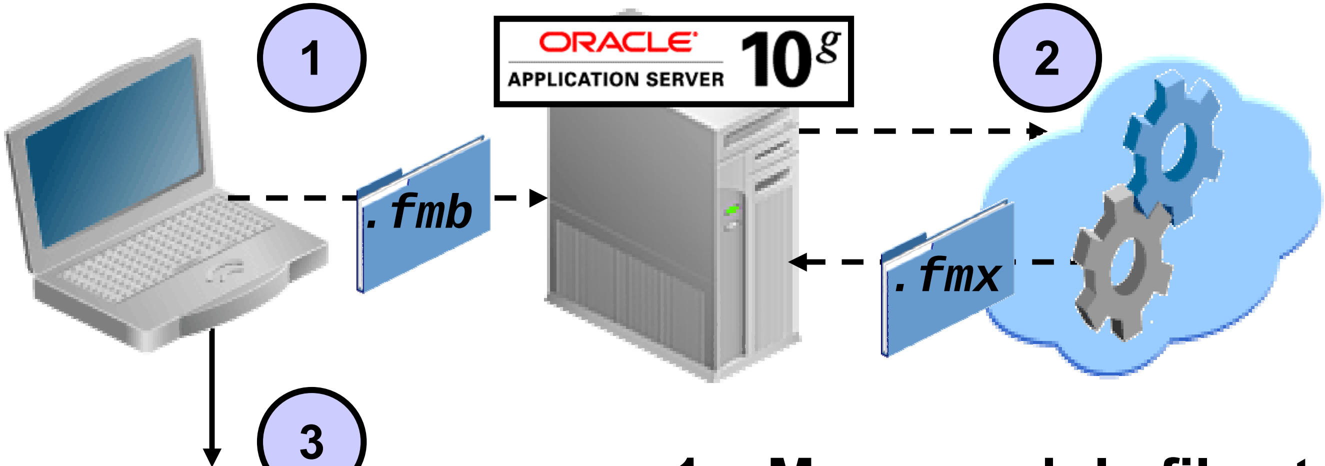
☑ Build Before Running

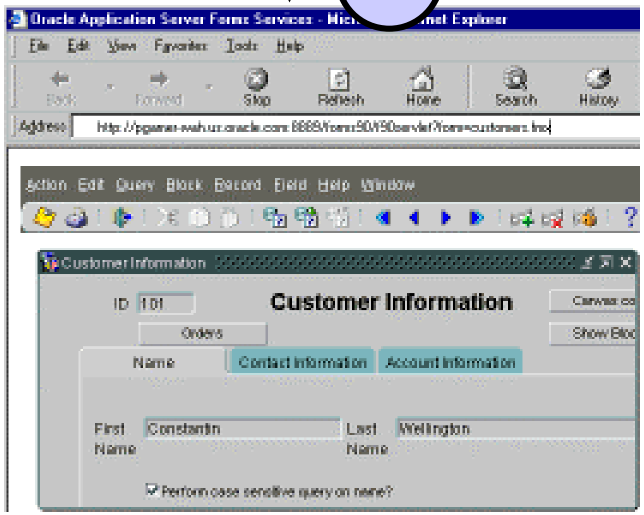
Color Palette:

Module Types and Storage Formats

Form Module	.fmb	.fmx	.fmt
Menu Module	.mmb	.mmx	.mmt
PL/SQL Library	.pll	.plx	.pld
Object Library	.olb		.olt

Deploying a Form Module




Oracle Application Server Forms Services - Micr... ...rnet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search History
Action Edit Query Block Record Field Help Window
Customer Information
Canvas co
Show Bloc
Orders
Name
Contact Information
Account Information

Address: http://pgamer-wah.us.oracle.com:8889/forms90/f90servlet?form=customers.fmx

Customer Information

ID 101

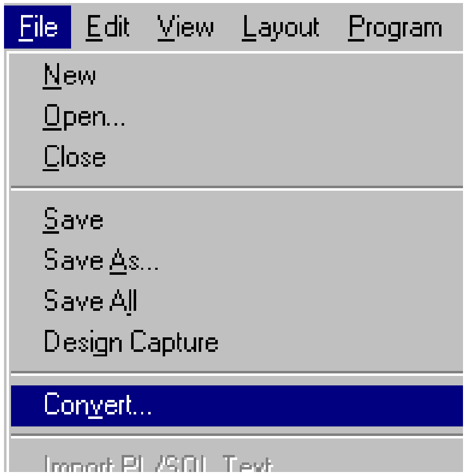
First Name Constantin

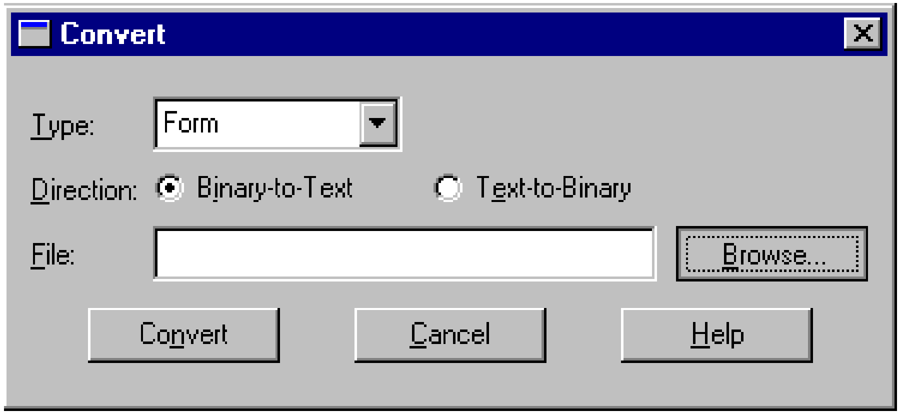
Last Name Wellington

☑ Perform case sensitive query on name?

1. Move module files to middle tier
2. Generate module on middle tier
3. Run in browser using Forms Services on middle tier

Text Files and Documentation





Type: Form

Direction: ● Binary-to-Text ○ Text-to-Binary

File:

- Convert a binary file to a text file.
- Create an ASCII file for a form module.

Summary

In this lesson, you should have learned that:

- To create a form module, you create an empty module, then add data blocks and other elements
- You can create a data block manually or with the Data Block Wizard and Layout Wizard
- You can save and compile a form module using the File and Program menus or from the toolbar
- You can store form, menu, and library modules in text format (useful for documentation), in a portable binary format, or a non-portable binary executable format
- To deploy a form module, you move it to the application server machine and generate it

Practice 4 Overview

This practice covers the following topics:

- Creating a new form module
- Creating a data block by using Forms Builder wizards
- Saving and running the form module

5

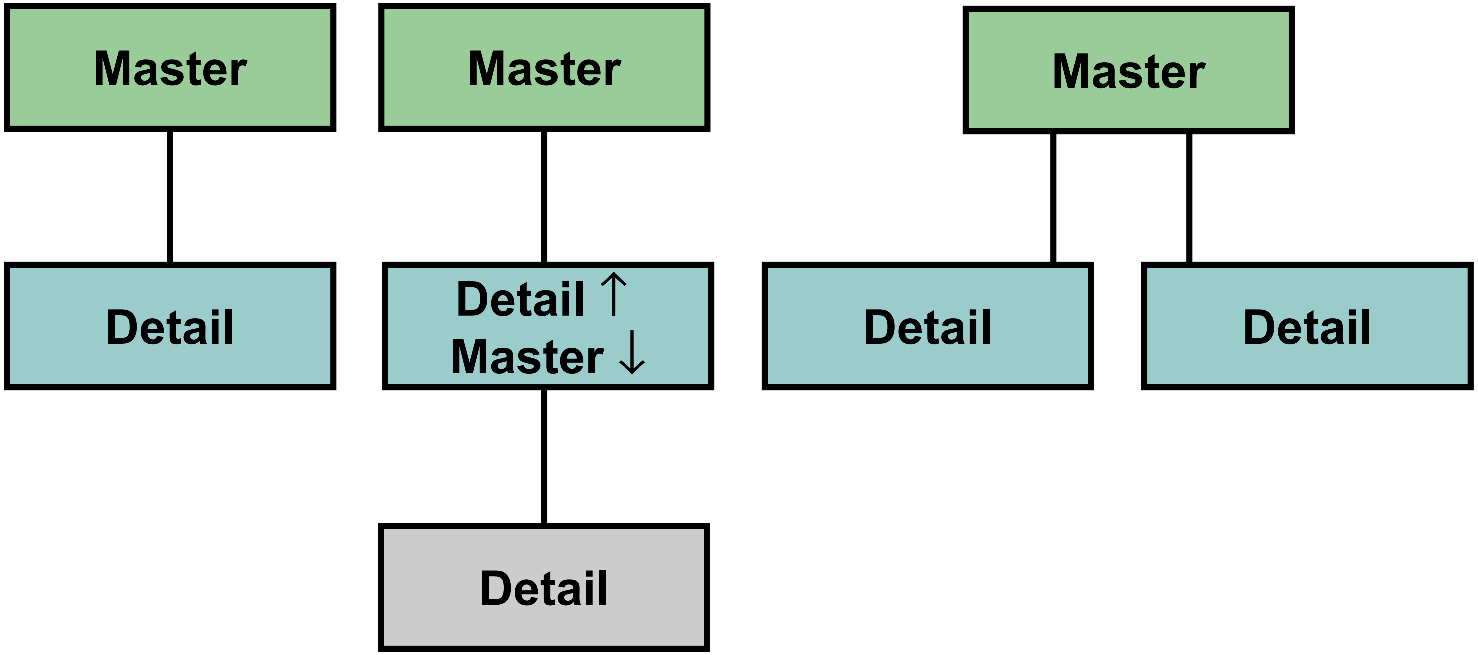
Creating a Master-Detail Form

Objectives

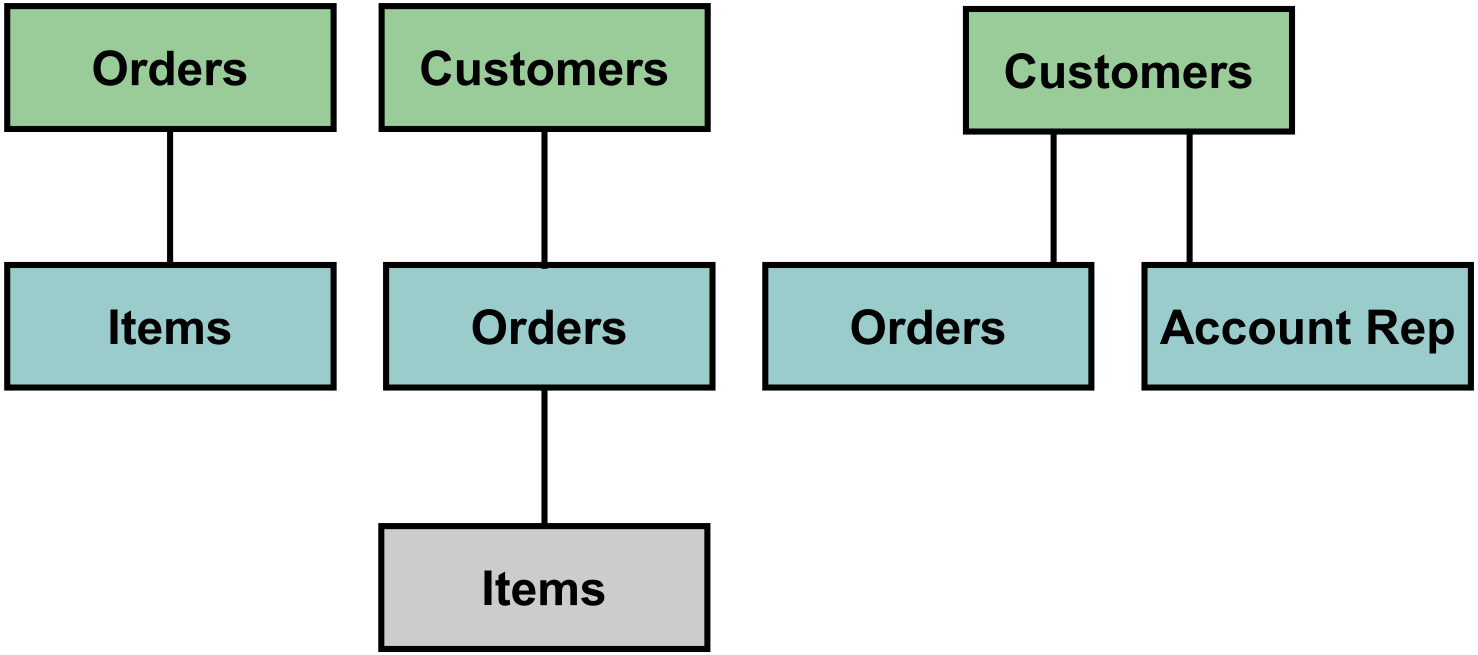
After completing this lesson, you should be able to do the following:

- Create data blocks with relationships
- Modify a data block
- Modify the layout of a data block
- Run a master-detail form

Form Block Relationships



Form Block Relationships


Orders

Items

Customers

Orders

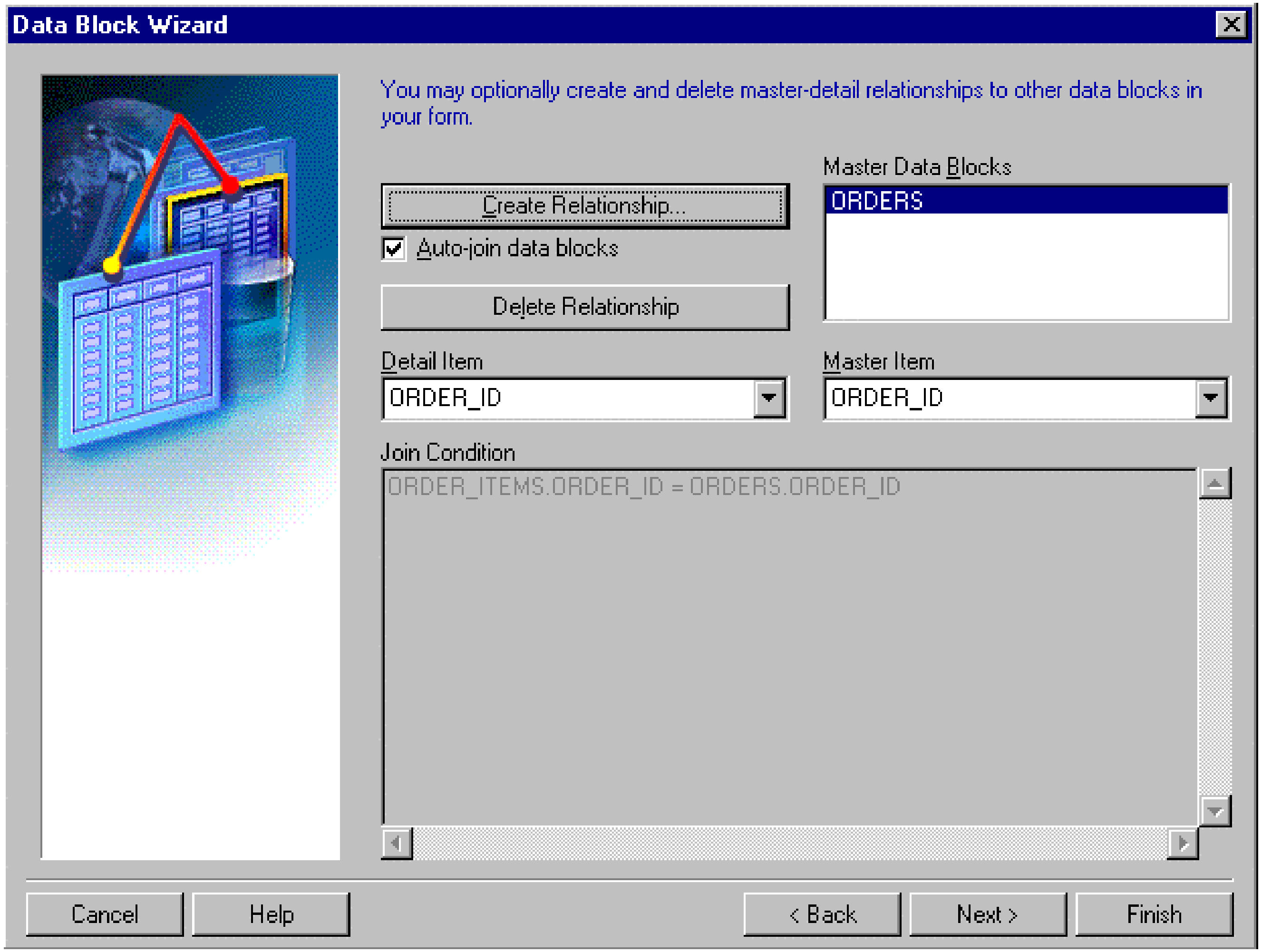
Items

Customers

Orders

Account Rep

Data Block Wizard: Master-Detail Page



You may optionally create and delete master-detail relationships to other data blocks in your form.

☑ Auto-join data blocks

Master Data Blocks: ORDERS

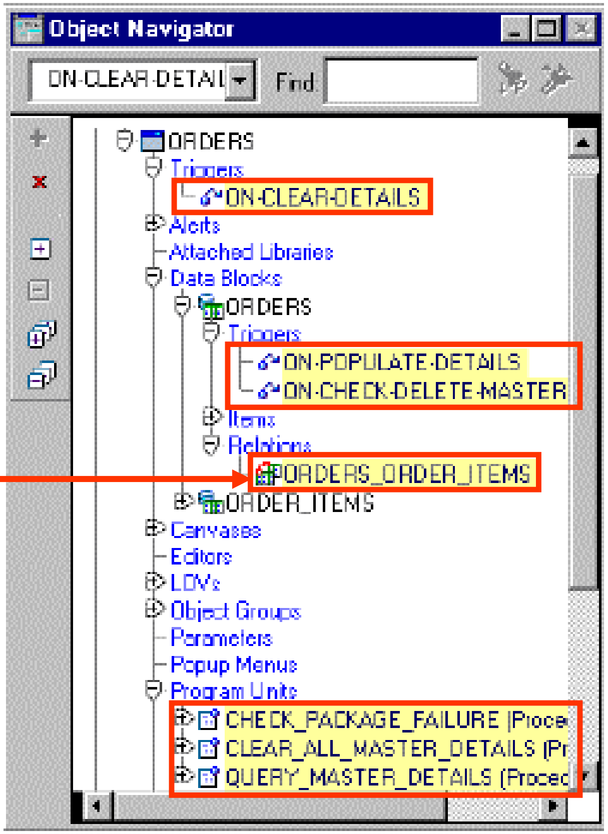
Detail Item: ORDER_ID

Master Item: ORDER_ID

Join Condition: ORDER_ITEMS.ORDER_ID = ORDERS.ORDER_ID

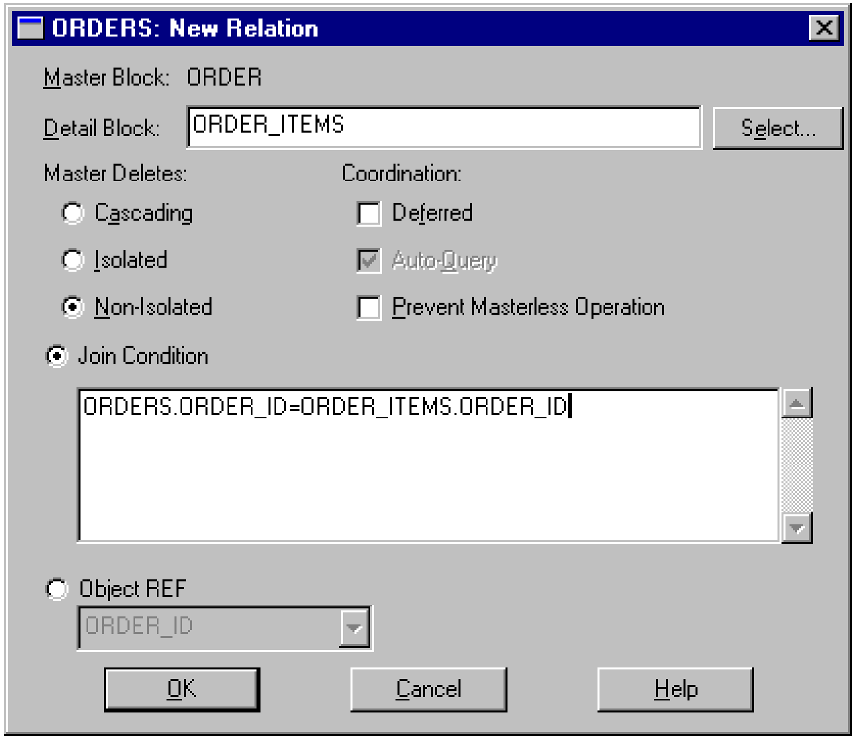
Relation Object

- New relation object created in Object Navigator under master data block node
- Default name assigned: `MasterDataBlock_DetailDataBlock`
- Triggers and program units generated automatically



```
ORDERS
  Triggers
    ON-CLEAR-DETAILS
  Alerts
  Attached Libraries
  Data Blocks
    ORDERS
      Triggers
        ON-POPULATE-DETAILS
        ON-CHECK-DELETE-MASTER
      Items
      Relations
        ORDERS_ORDER_ITEMS
    ORDER_ITEMS
  Canvases
  Editors
  LOVs
  Object Groups
  Parameters
  Popup Menus
  Program Units
    CHECK_PACKAGE_FAILURE (Proce
    CLEAR_ALL_MASTER_DETAILS (Pr
    QUERY_MASTER_DETAILS (Procec
```

Creating a Relation Manually



ORDERS: New Relation

Master Block: ORDER

Detail Block: ORDER_ITEMS

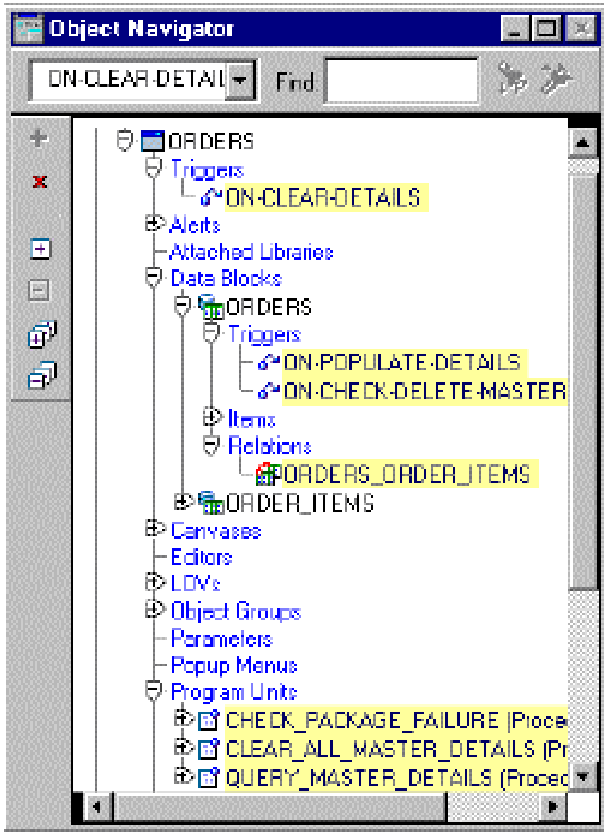
Master Deletes:
- ○ Cascading
- ○ Isolated
- ● Non-Isolated

Coordination:
- □ Deferred
- ☑ Auto-Query
- □ Prevent Masterless Operation

● Join Condition

```
ORDERS.ORDER_ID=ORDER_ITEMS.ORDER_ID
```

○ Object REF: ORDER_ID



Object Navigator

ON-CLEAR-DETAIL

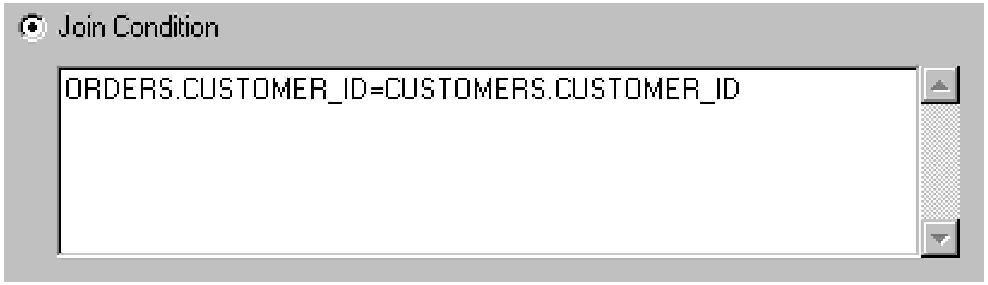
Find:

```
ORDERS
  Triggers
    ON-CLEAR-DETAILS
  Alerts
  Attached Libraries
  Data Blocks
    ORDERS
      Triggers
        ON-POPULATE-DETAILS
        ON-CHECK-DELETE-MASTER
      Items
      Relations
        ORDERS_ORDER_ITEMS
    ORDER_ITEMS
  Canvases
  Editors
  LOVs
  Object Groups
  Parameters
  Popup Menus
  Program Units
    CHECK_PACKAGE_FAILURE (Proce
    CLEAR_ALL_MASTER_DETAILS (Pr
    QUERY_MASTER_DETAILS (Proced
```

ORACLE

Join Condition

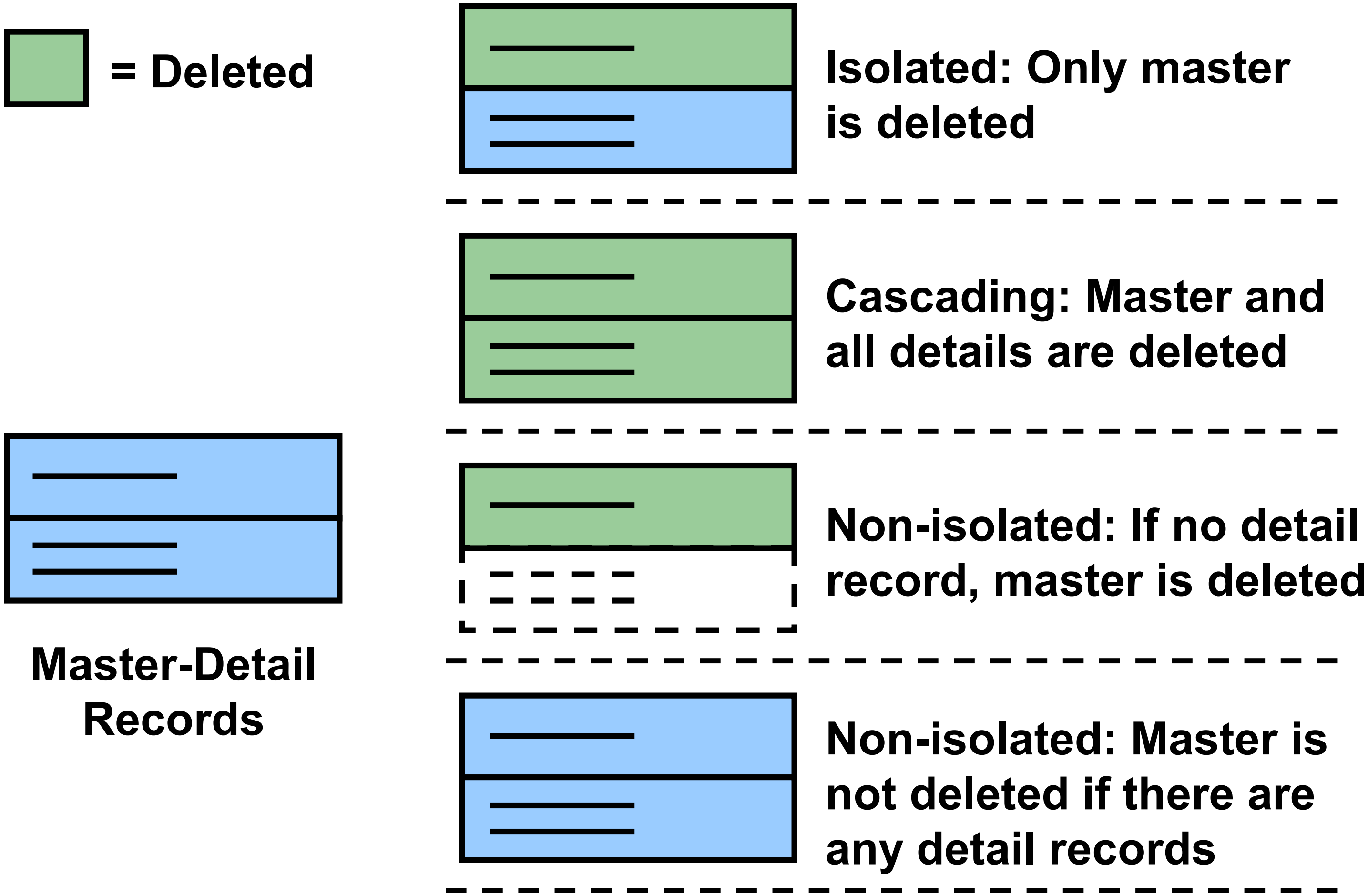
- The join condition creates primary-foreign key link between blocks
- Define a join condition using:
  - Block and item names (not table and column names)
  - Do not precede names with colon
  - SQL equijoin syntax



● Join Condition

```
ORDERS.CUSTOMER_ID=CUSTOMERS.CUSTOMER_ID
```

Deletion Properties



🟩 = Deleted

Master-Detail Records

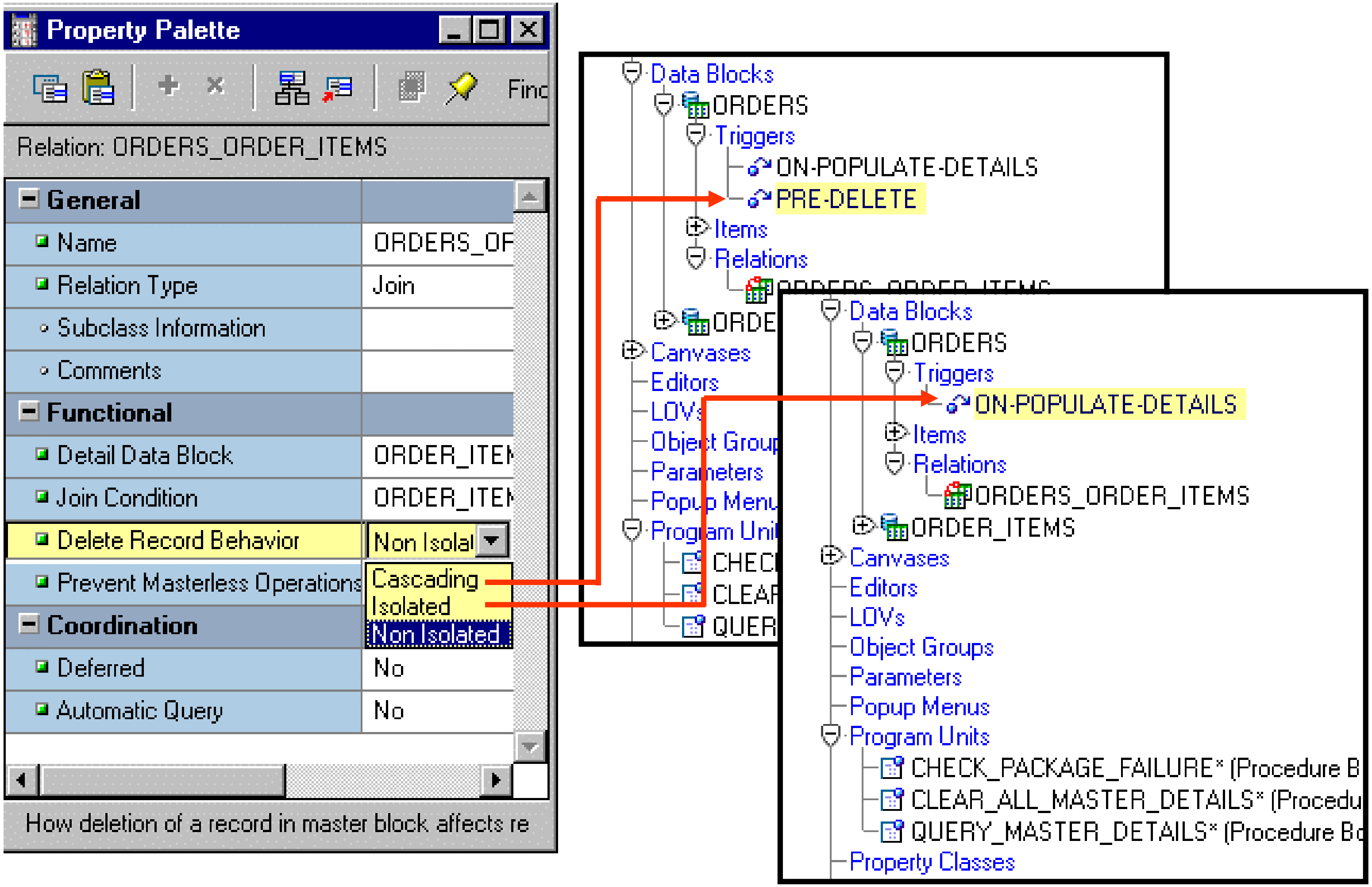
Isolated: Only master is deleted

Cascading: Master and all details are deleted

Non-isolated: If no detail record, master is deleted

Non-isolated: Master is not deleted if there are any detail records

Modifying a Relation



Property Palette

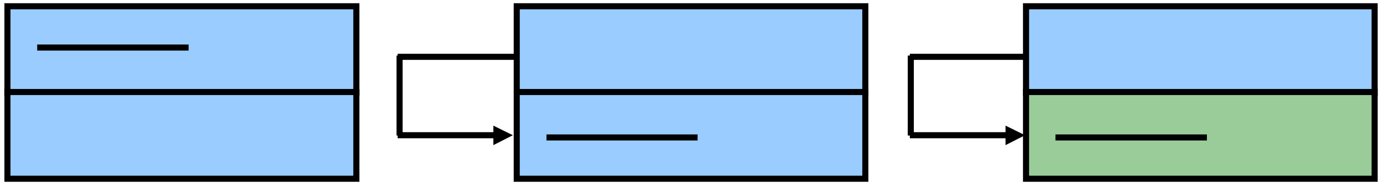
Relation: ORDERS_ORDER_ITEMS

General	
Name	ORDERS_OF
Relation Type	Join
Subclass Information	
Comments	
Functional	
Detail Data Block	ORDER_ITEM
Join Condition	ORDER_ITEM
Delete Record Behavior	Non Isolat
Prevent Masterless Operations	
Coordination	
Deferred	No
Automatic Query	No

Cascading
Isolated
Non Isolated

How deletion of a record in master block affects re

Coordination Properties



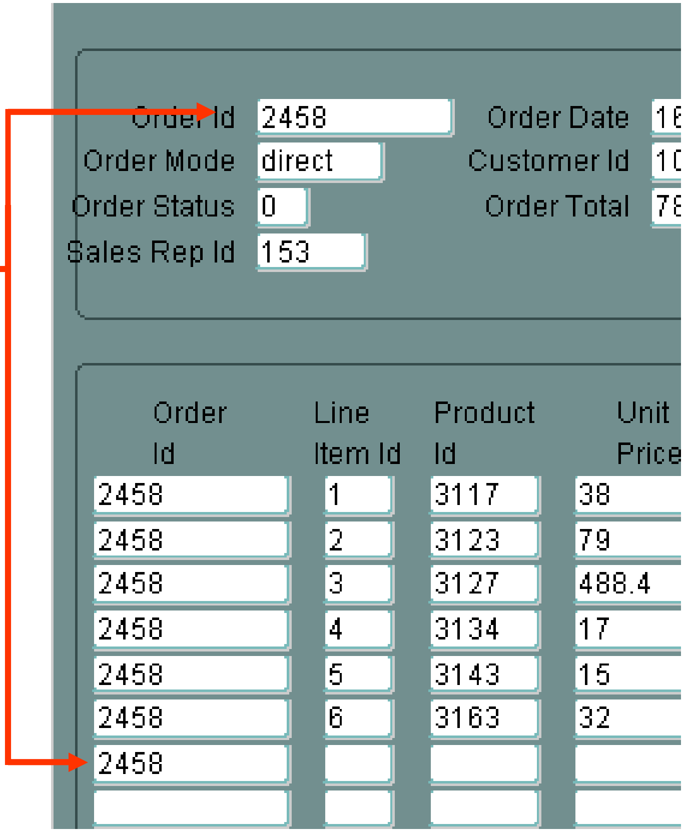
Default

Deferred
with auto query

Deferred
without
auto query

Running a Master-Detail Form Module

- Automatic block linking for:
  - Querying
  - Inserting
- Default deletion rules: Cannot delete master record if detail records exist

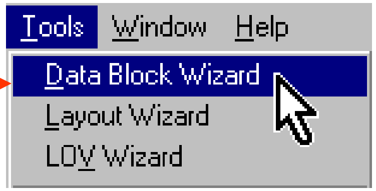


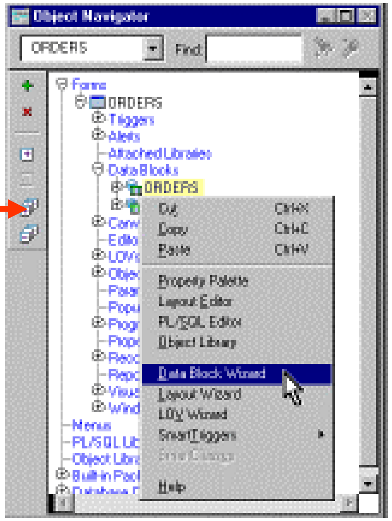
Order Id	2458	Order Date	16
Order Mode	direct	Customer Id	10
Order Status	0	Order Total	78
Sales Rep Id	153		

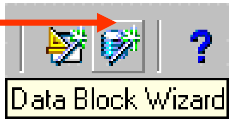
Order Id	Line Item Id	Product Id	Unit Price
2458	1	3117	38
2458	2	3123	79
2458	3	3127	488.4
2458	4	3134	17
2458	5	3143	15
2458	6	3163	32
2458			

Modifying the Structure of a Data Block

- Reentrant Data Block Wizard:
  1. Select frame or object in Layout Editor, or data block or frame in Object Navigator
  2. Select Tools > Data Block Wizard OR
     Right-click and select Data Block Wizard OR
     Click Data Block Wizard
- Object Navigator:
  – Create or delete items
  – Change item properties
- Block Property Palette: Change property values

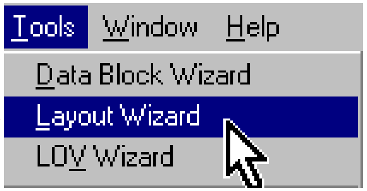


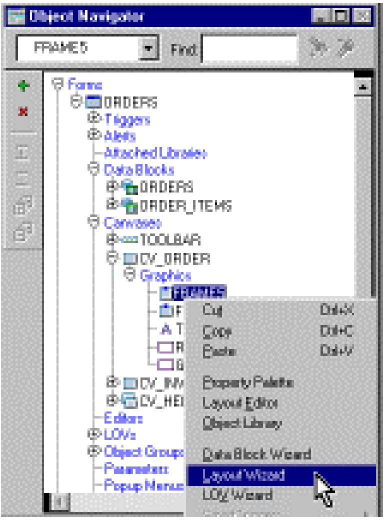


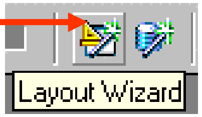


Modifying the Layout of a Data Block

- Reentrant Layout Wizard:
  - Select frame in Object Navigator or Layout Editor
  - Select Tools > Layout Wizard
  - OR
  - Right-click and select Layout Wizard OR
  - Click Layout Wizard
- Layout Editor:
  - Select Tools > Layout Editor
  - Make changes manually
- Frame Property Palette: Change property values







Summary

In this lesson, you should have learned that:

- You can create data blocks with relationships by using the Data Block Wizard or by manually creating a Relation object
- When you run a master-detail form, block coordination is automatic depending on properties of the Relation object
- You can modify a data block manually or with the Data Block Wizard in reentrant mode
- You can modify the layout manually or with the Layout Wizard in reentrant mode

Practice 5 Overview

This practice covers the following topics:

- Creating a master-detail form module
- Modifying data block layout by using the Layout Wizard in reentrant mode
- Saving and running the form module

6

Working with Data Blocks and Frames

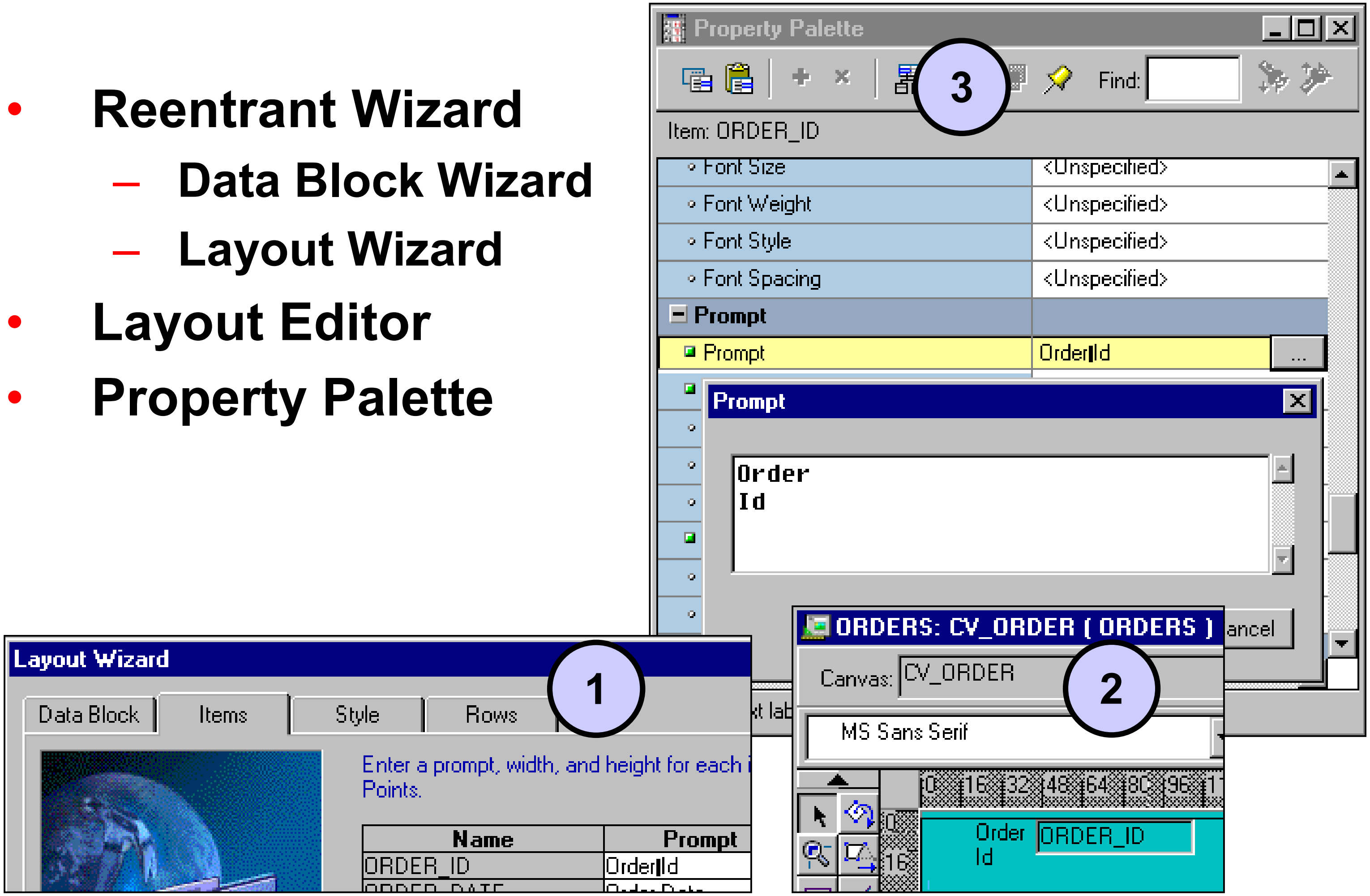
Objectives

After completing this lesson, you should be able to do the following:

- Identify the components of the Property Palette
- Manage object properties
- Create and use Visual Attributes
- Control the behavior and appearance of data blocks
- Control frame properties
- Create blocks that do not directly correspond to database tables
- Delete data blocks and their components

Managing Object Properties

- Reentrant Wizard
  - Data Block Wizard
  - Layout Wizard
- Layout Editor
- Property Palette



Item: ORDER_ID

Font Size	<Unspecified>
Font Weight	<Unspecified>
Font Style	<Unspecified>
Font Spacing	<Unspecified>
Prompt	
Prompt	OrderId

Prompt

Order
Id

Enter a prompt, width, and height for each i
Points.

Name	Prompt
ORDER_ID	OrderId

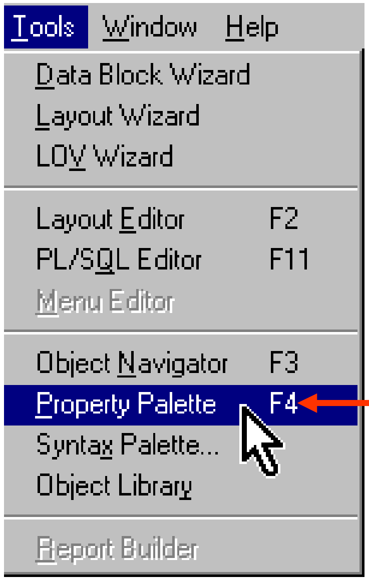
ORDERS: CV_ORDER (ORDERS)

Canvas: CV_ORDER

MS Sans Serif

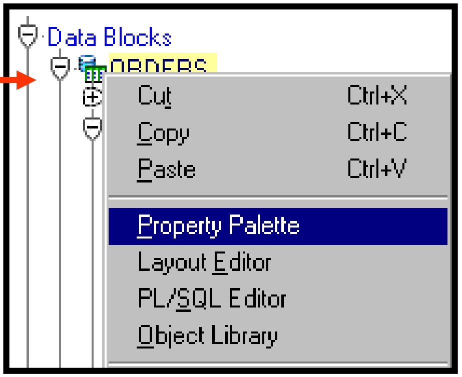
Order Id ORDER_ID

Displaying the Property Palette

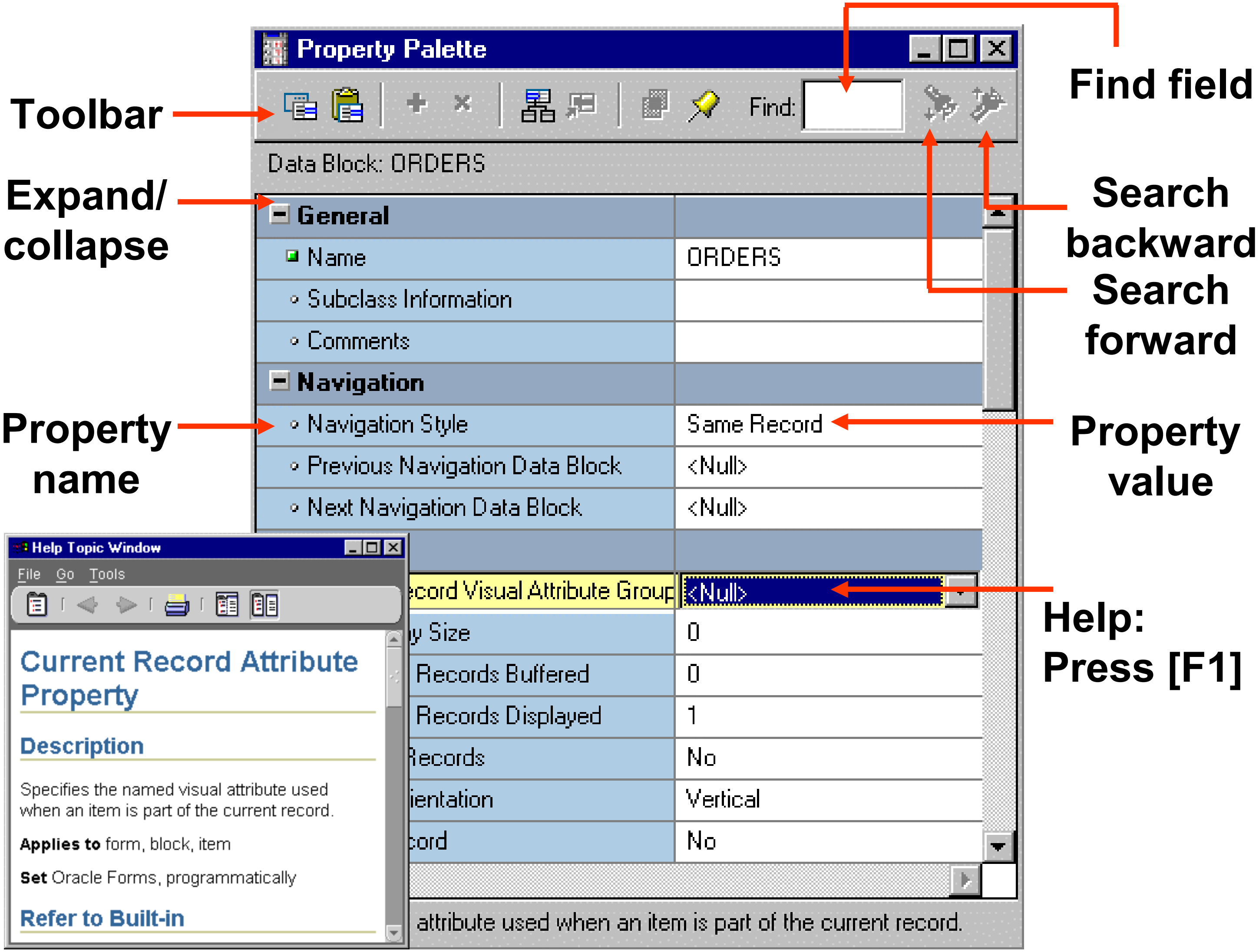


To display the Property Palette, use one of the following methods:

- Select Tools > Property Palette (or use the shortcut key).
- Double-click the object icon in the Object Navigator.
- Double-click the object in the Layout Editor.
- Right-click the object icon in the Object Navigator.
- Right-click the object in the Layout Editor.



Property Palette: Features



Data Block: ORDERS

General	
Name	ORDERS
Subclass Information	
Comments	
Navigation	
Navigation Style	Same Record
Previous Navigation Data Block	<Null>
Next Navigation Data Block	<Null>
ecord Visual Attribute Group	<Null>
y Size	0
Records Buffered	0
Records Displayed	1
ecords	No
ientation	Vertical
cord	No

attribute used when an item is part of the current record.

Current Record Attribute Property

Description

Specifies the named visual attribute used when an item is part of the current record.

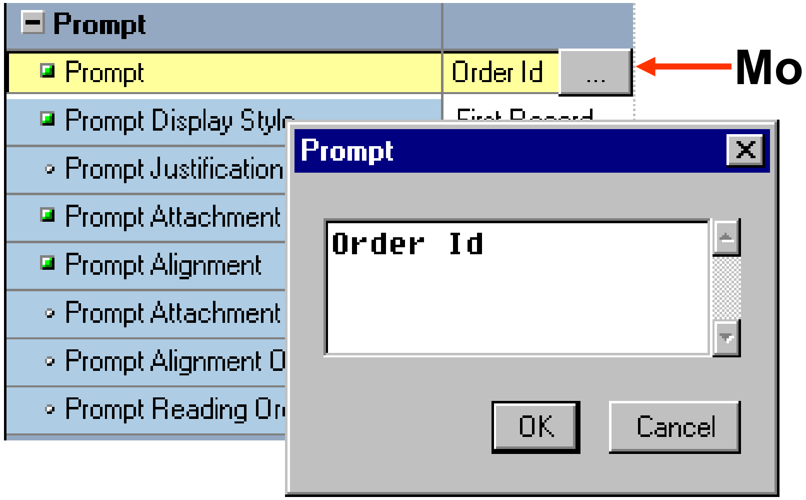
Applies to form, block, item

Set Oracle Forms, programmatically

Refer to Built-in

Property Controls

Text field



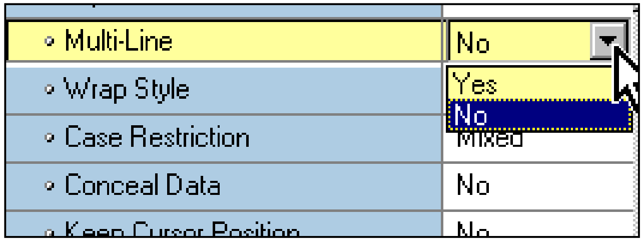
Prompt
- Prompt: Order Id ...
- Prompt Display Style: First Record
- Prompt Justification
- Prompt Attachment
- Prompt Alignment
- Prompt Attachment
- Prompt Alignment O
- Prompt Reading Or

Prompt

```
Order Id
```

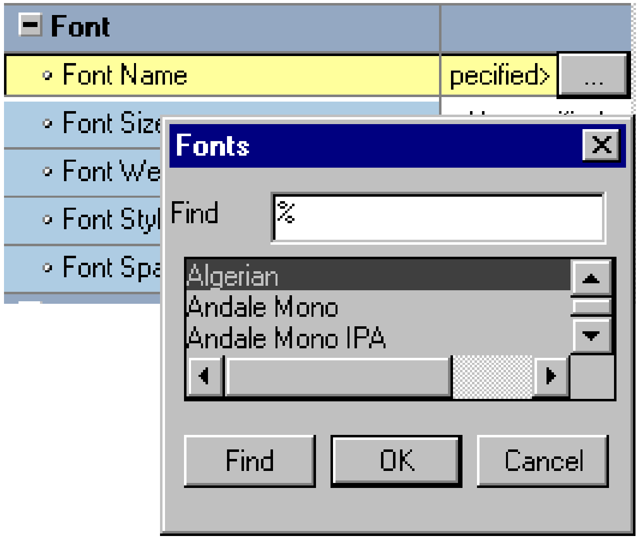
More button

Pop-up list



Multi-Line	No
Wrap Style	Yes No
Case Restriction	Mixed
Conceal Data	No
Keep Cursor Position	No

LOV window



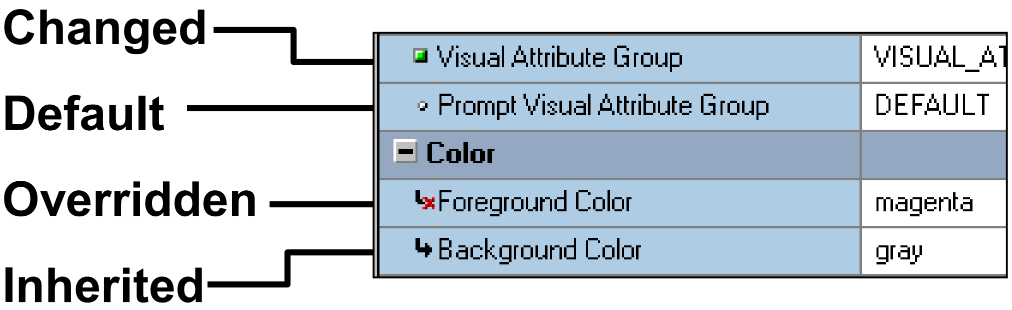
Font
- Font Name: pecified> ...
- Font Size
- Font We
- Font Styl
- Font Spa

Fonts

Find: %

- Algerian
- Andale Mono
- Andale Mono IPA

Property Controls



Changed

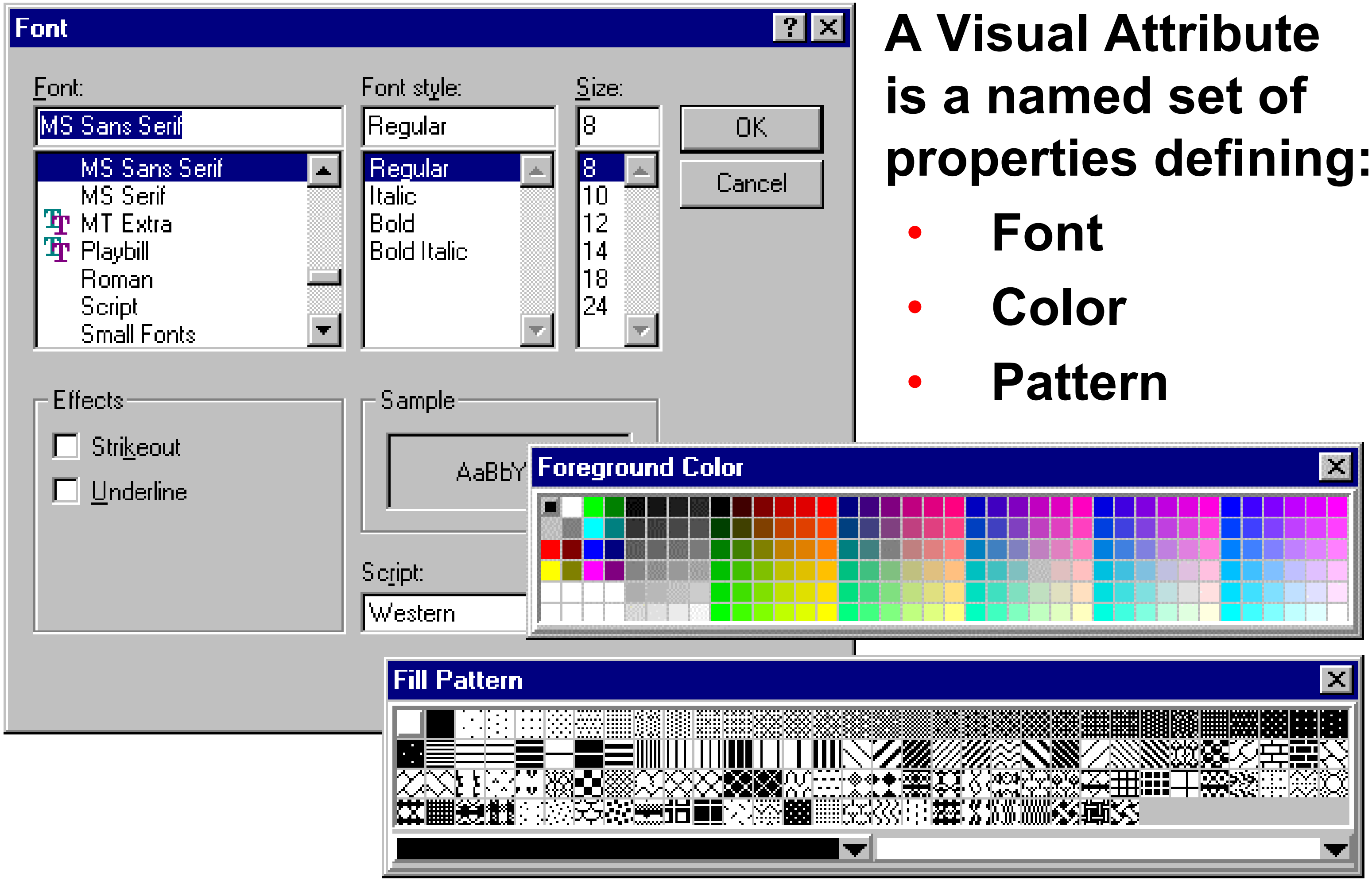
Default

Overridden

Inherited

🟩 Visual Attribute Group	VISUAL_AT
○ Prompt Visual Attribute Group	DEFAULT
⊟ **Color**	
↳✗ Foreground Color	magenta
↳ Background Color	gray

Visual Attributes



Font: MS Sans Serif

- MS Sans Serif
- MS Serif
- MT Extra
- Playbill
- Roman
- Script
- Small Fonts

Font style: Regular

- Regular
- Italic
- Bold
- Bold Italic

Size: 8

- 8
- 10
- 12
- 14
- 18
- 24

Effects

☐ Strikeout

☐ Underline

Sample: AaBbY

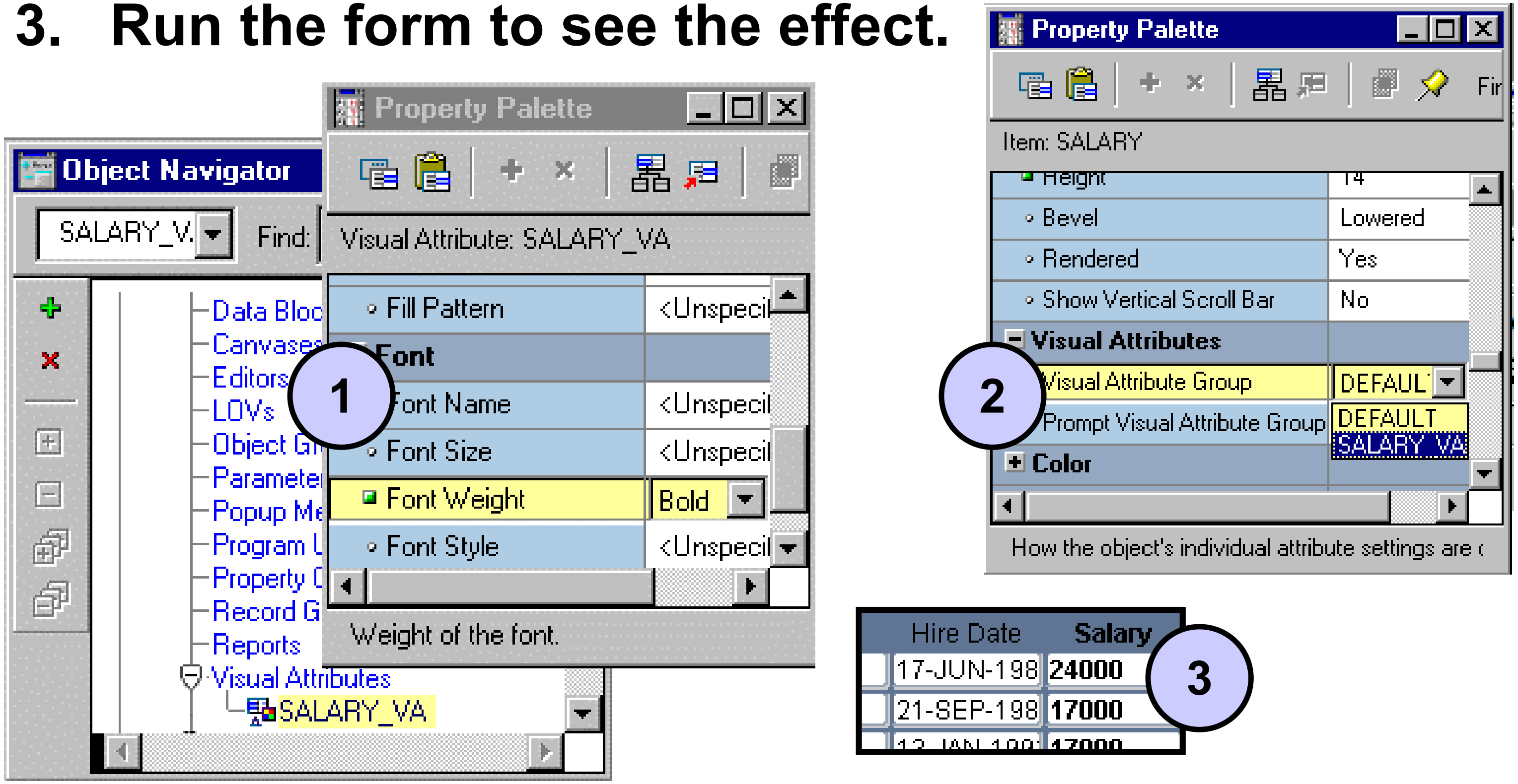
Script: Western

A Visual Attribute is a named set of properties defining:

- Font
- Color
- Pattern

How to Use Visual Attributes

1. Create a Visual Attribute.
2. Set the Visual Attribute–related property of an object to the desired Visual Attribute.
3. Run the form to see the effect.



Property Palette

Visual Attribute: SALARY_VA

Fill Pattern	<Unspecil
Font	
Font Name	<Unspecil
Font Size	<Unspecil
Font Weight	Bold
Font Style	<Unspecil

Weight of the font.

Property Palette

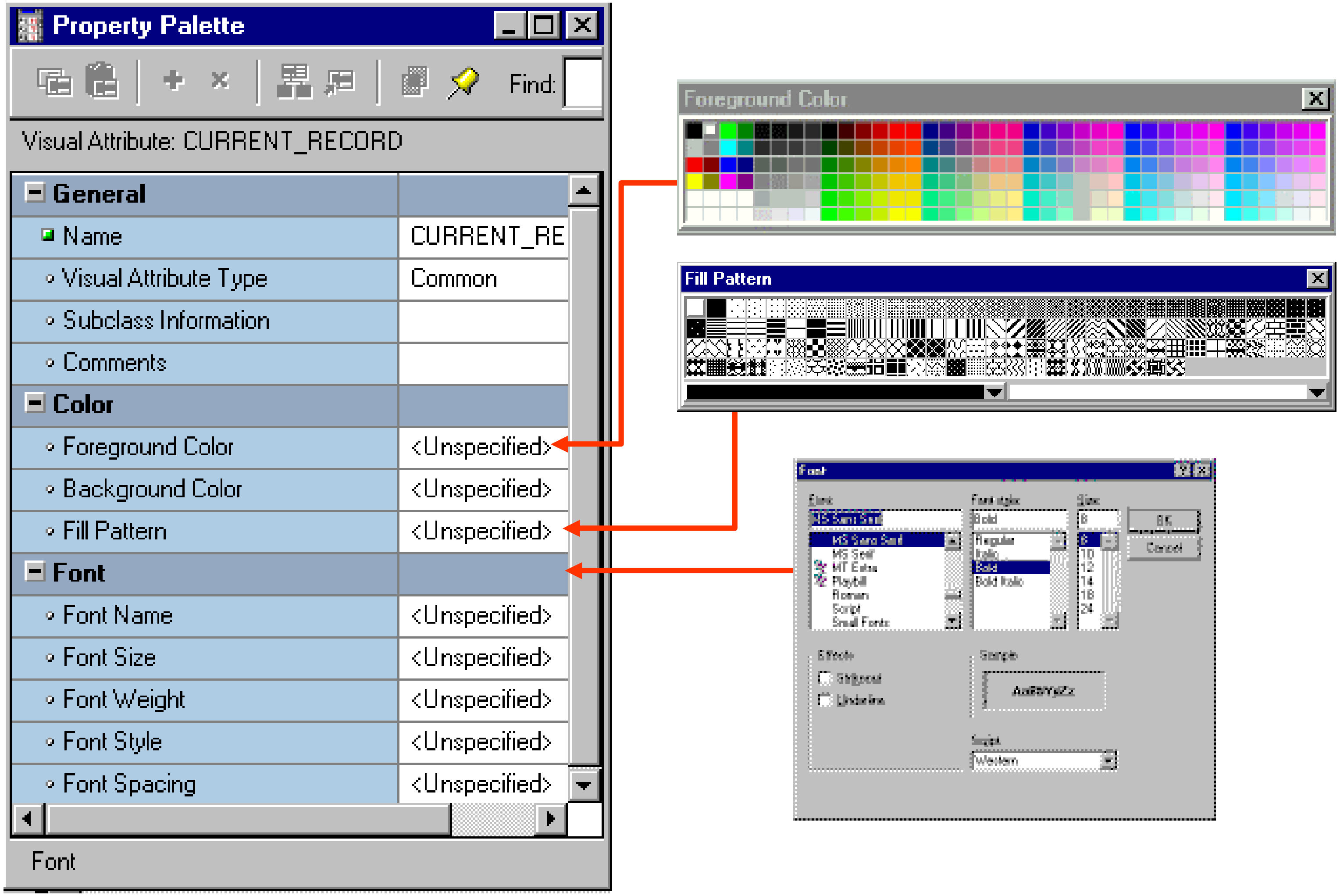
Item: SALARY

Height	14
Bevel	Lowered
Rendered	Yes
Show Vertical Scroll Bar	No
Visual Attributes	
Visual Attribute Group	DEFAUL
Prompt Visual Attribute Group	DEFAULT SALARY_VA
Color	

How the object's individual attribute settings are (

Hire Date	Salary
17-JUN-198	24000
21-SEP-198	17000
13-JAN-199	17000

Font, Pattern, and Color Pickers



Property Palette

Find:

Visual Attribute: CURRENT_RECORD

General	
Name	CURRENT_RE
Visual Attribute Type	Common
Subclass Information	
Comments	
Color	
Foreground Color	<Unspecified>
Background Color	<Unspecified>
Fill Pattern	<Unspecified>
Font	
Font Name	<Unspecified>
Font Size	<Unspecified>
Font Weight	<Unspecified>
Font Style	<Unspecified>
Font Spacing	<Unspecified>

Font

Font: MS Sans Serif

Font style: Bold

Size: 8

Effects

□ Strikeout

□ Underline

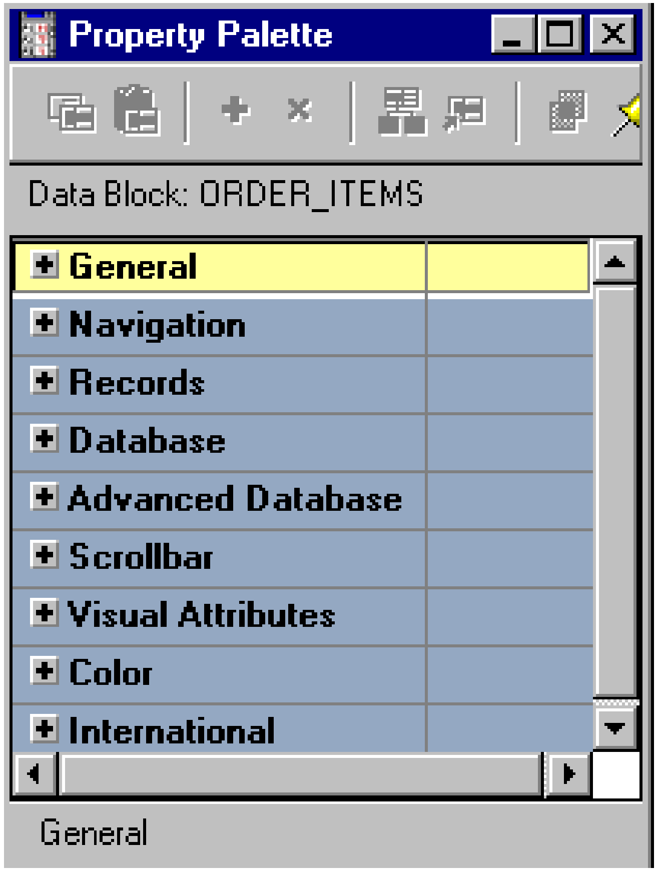
Sample: AaBbYyZz

Script: Western

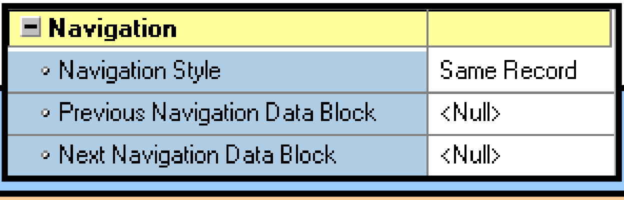
Controlling Data Block Behavior and Appearance

Data Block Property Groups:

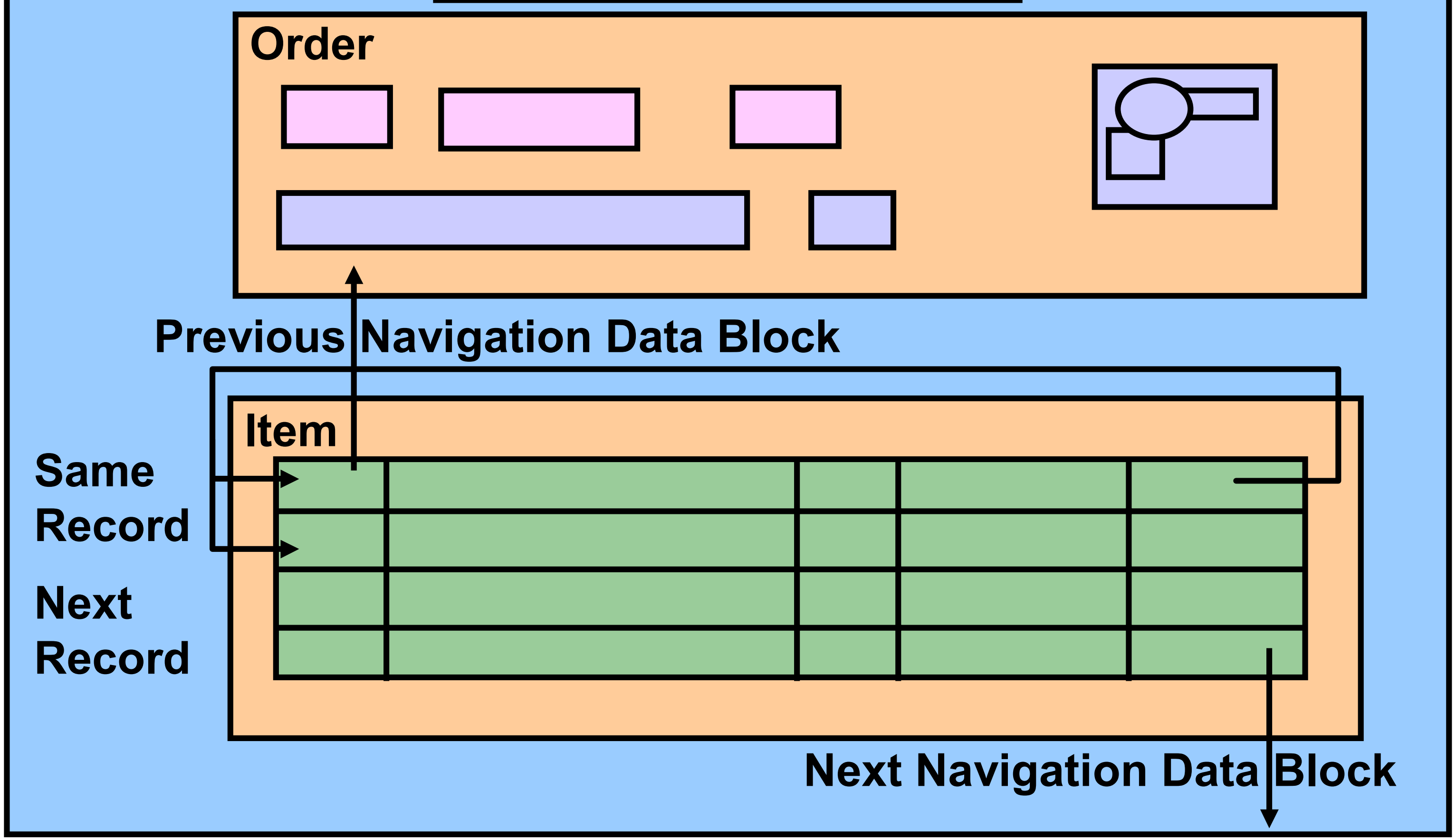
- General
- Navigation
- Records
- Database
- Advanced Database
- Scrollbar
- Visual Attributes
- Color
- International



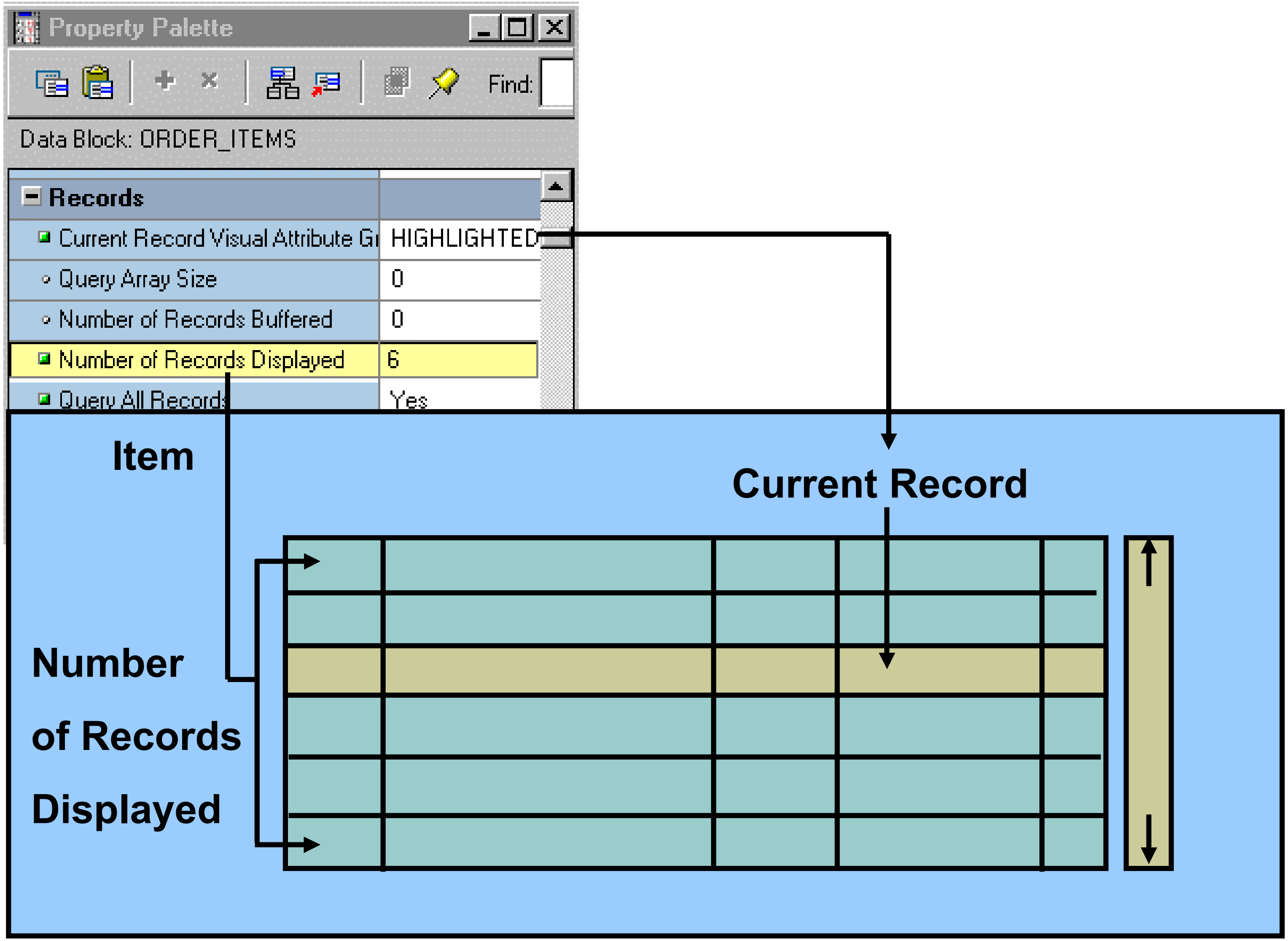
Navigation Properties



⊟ Navigation	
∘ Navigation Style	Same Record
∘ Previous Navigation Data Block	<Null>
∘ Next Navigation Data Block	<Null>



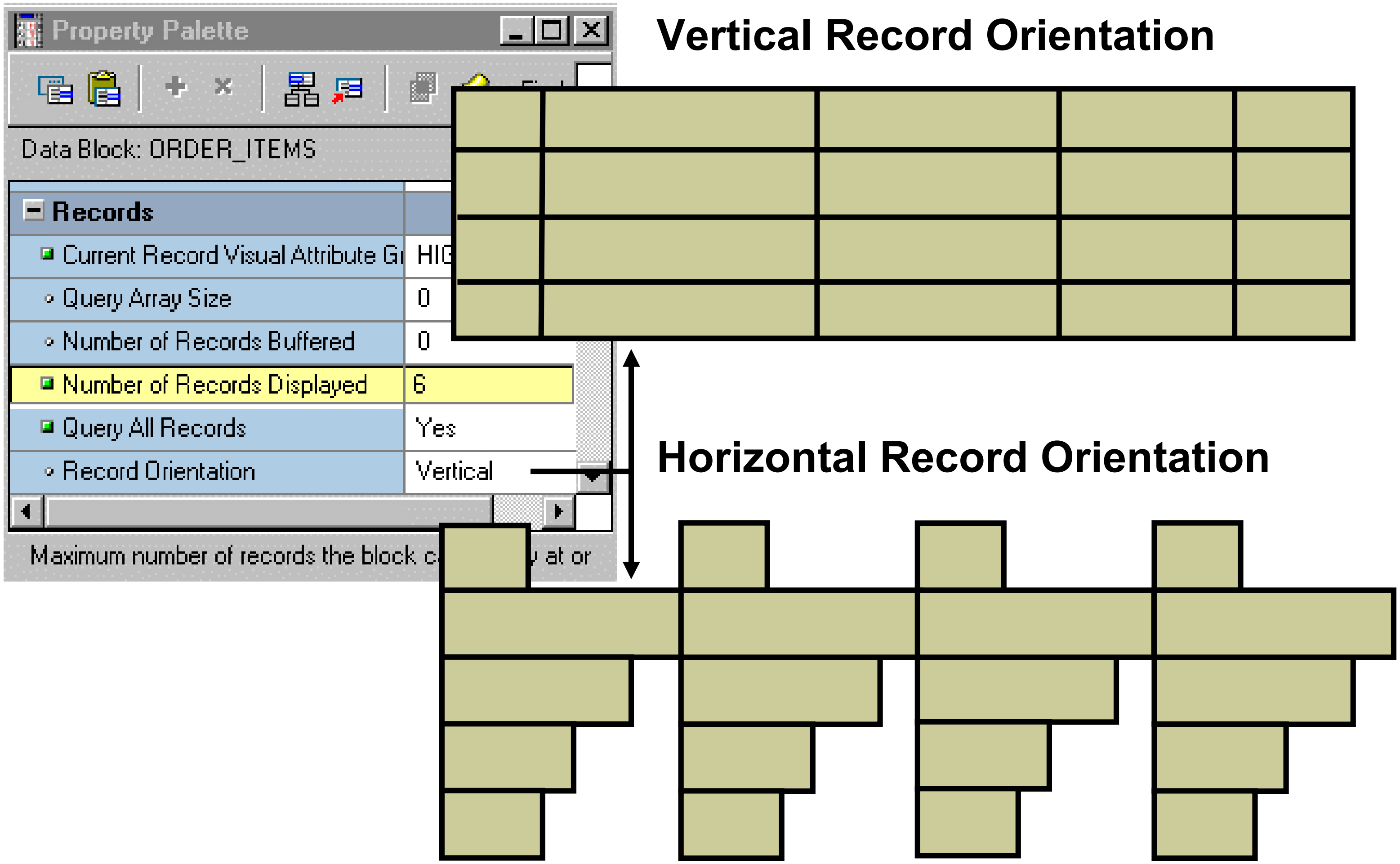
Records Properties



Data Block: ORDER_ITEMS

Records	
Current Record Visual Attribute Gı	HIGHLIGHTED
Query Array Size	0
Number of Records Buffered	0
Number of Records Displayed	6
Query All Records	Yes

Records Properties



Data Block: ORDER_ITEMS

Records	
Current Record Visual Attribute Gı	HIG
Query Array Size	0
Number of Records Buffered	0
Number of Records Displayed	6
Query All Records	Yes
Record Orientation	Vertical

Maximum number of records the block c y at or

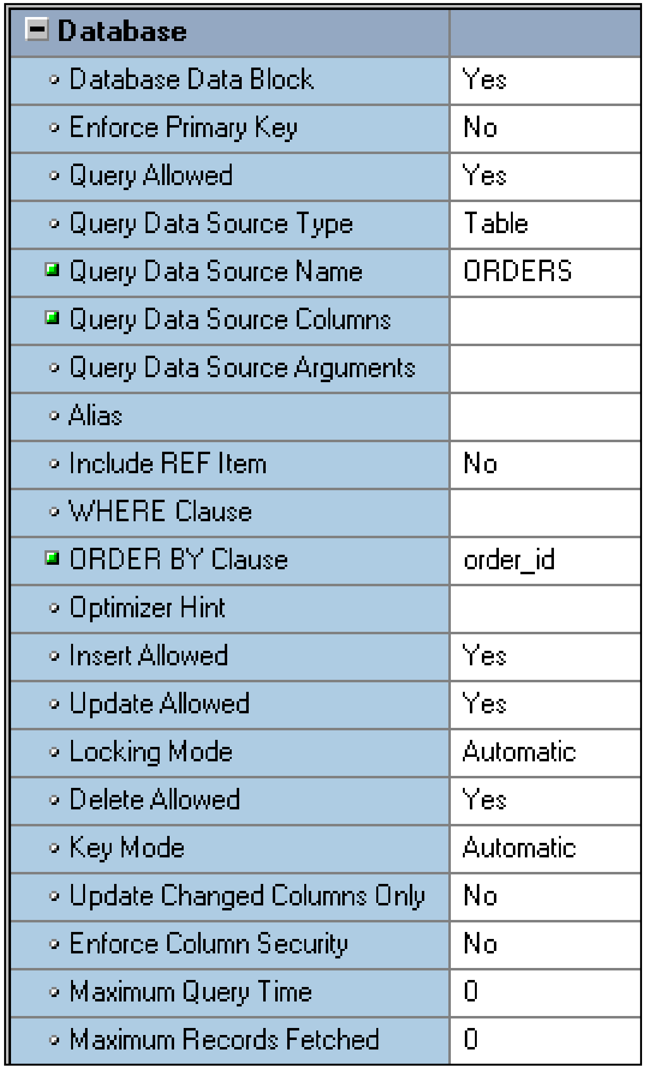
Vertical Record Orientation

Horizontal Record Orientation

Database Properties

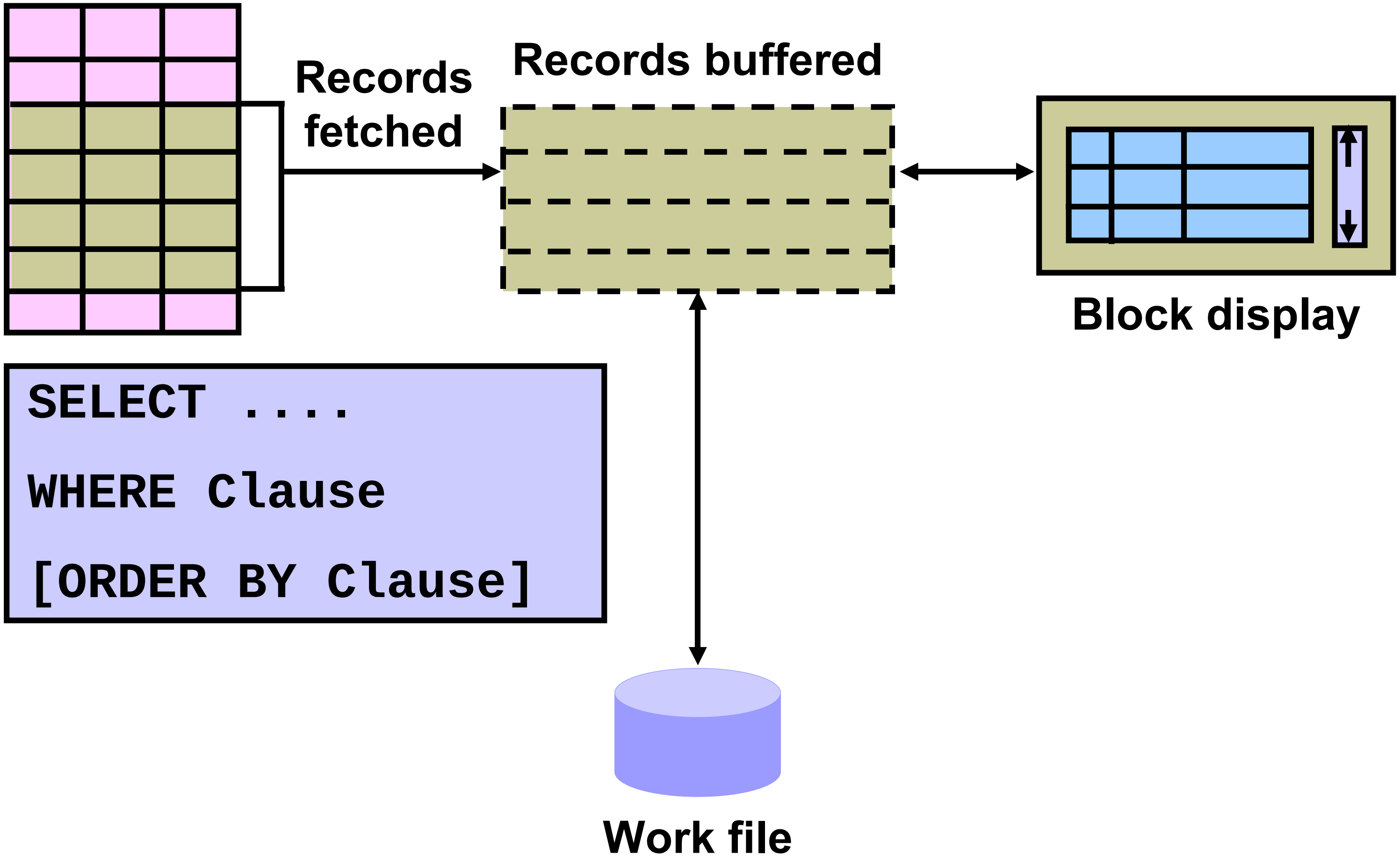
Use properties in the Database group to control:

- Type of block—data or control block
- Query, insert, update, and delete operations on the data block
- Data block’s data source
- Query search criteria and default sort order
- Maximum query time
- Maximum number of records fetched



Database	
Database Data Block	Yes
Enforce Primary Key	No
Query Allowed	Yes
Query Data Source Type	Table
Query Data Source Name	ORDERS
Query Data Source Columns	
Query Data Source Arguments	
Alias	
Include REF Item	No
WHERE Clause	
ORDER BY Clause	order_id
Optimizer Hint	
Insert Allowed	Yes
Update Allowed	Yes
Locking Mode	Automatic
Delete Allowed	Yes
Key Mode	Automatic
Update Changed Columns Only	No
Enforce Column Security	No
Maximum Query Time	0
Maximum Records Fetched	0

Database Properties



Records fetched

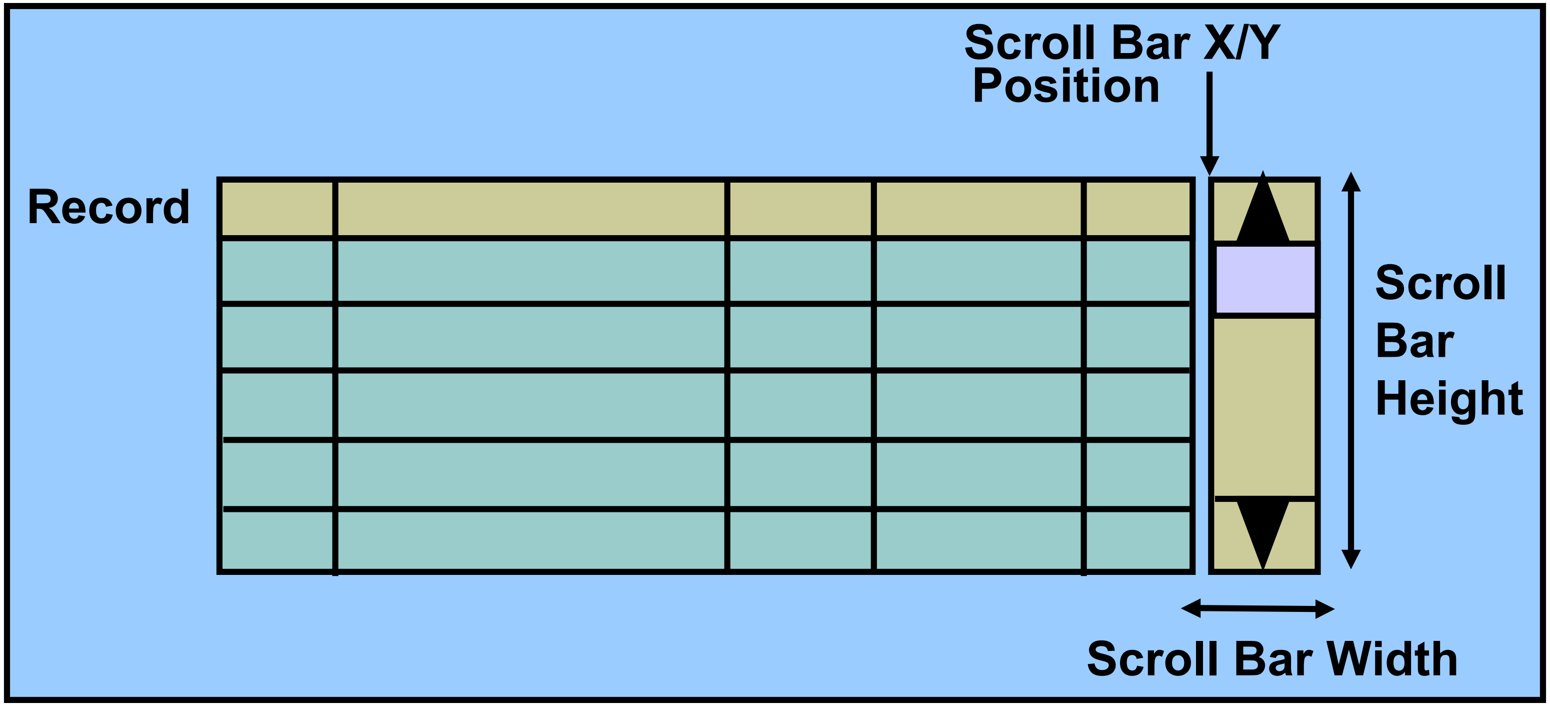
Records buffered

Block display

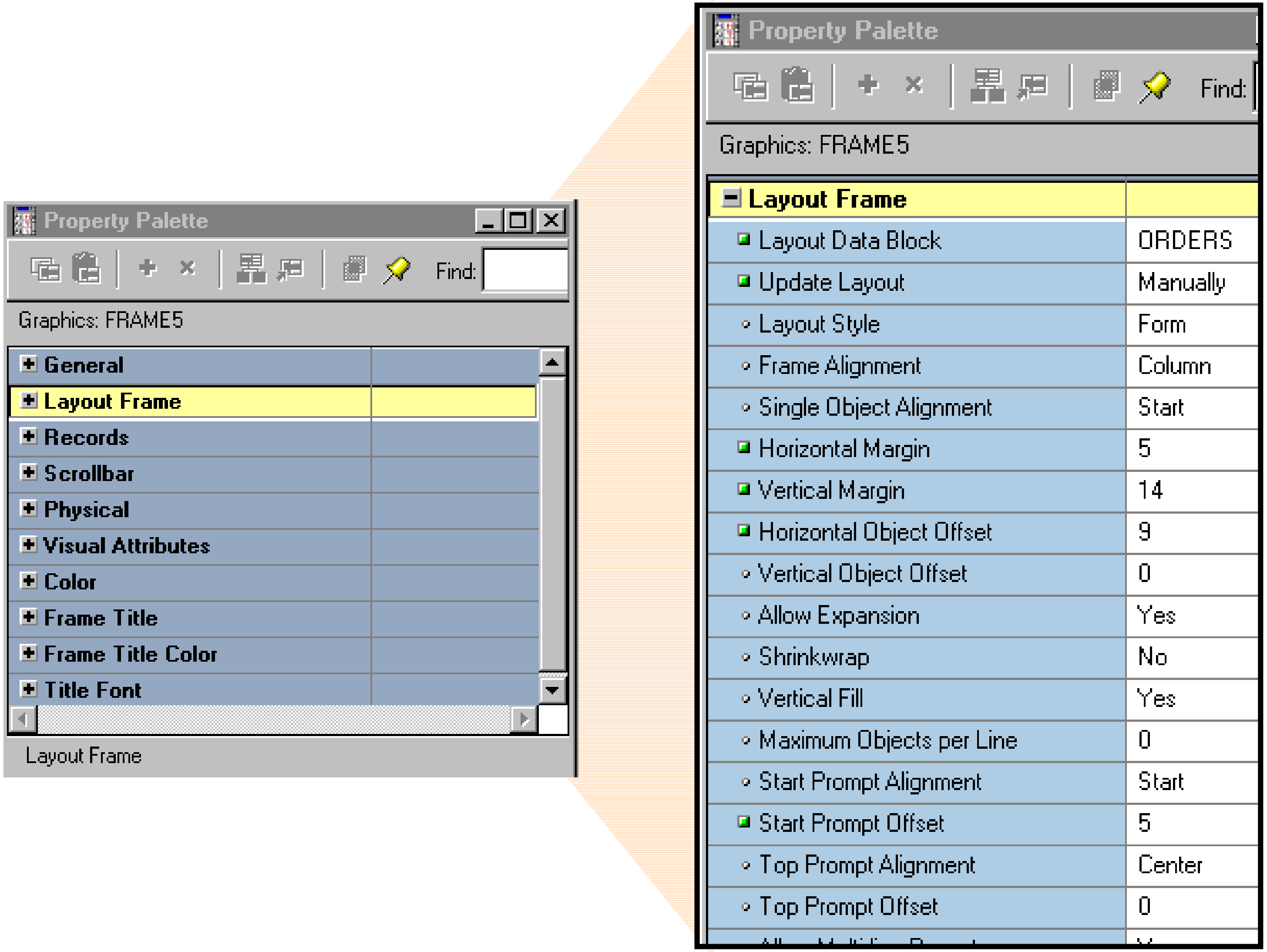
```
SELECT ....
WHERE Clause
[ORDER BY Clause]
```

Work file

Scroll Bar Properties



Controlling Frame Properties



Graphics: FRAME5

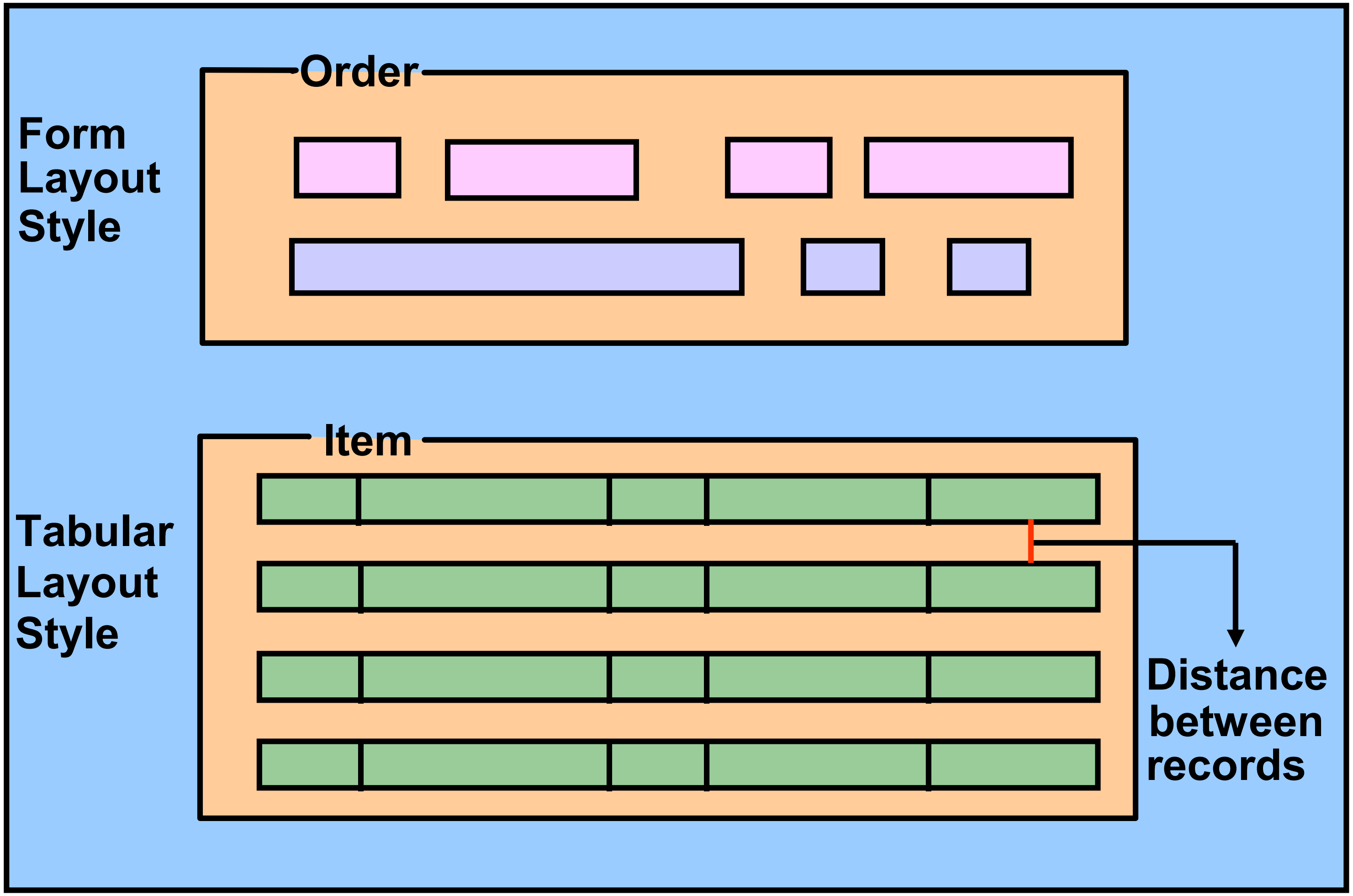
- General
- Layout Frame
- Records
- Scrollbar
- Physical
- Visual Attributes
- Color
- Frame Title
- Frame Title Color
- Title Font

Layout Frame

Graphics: FRAME5

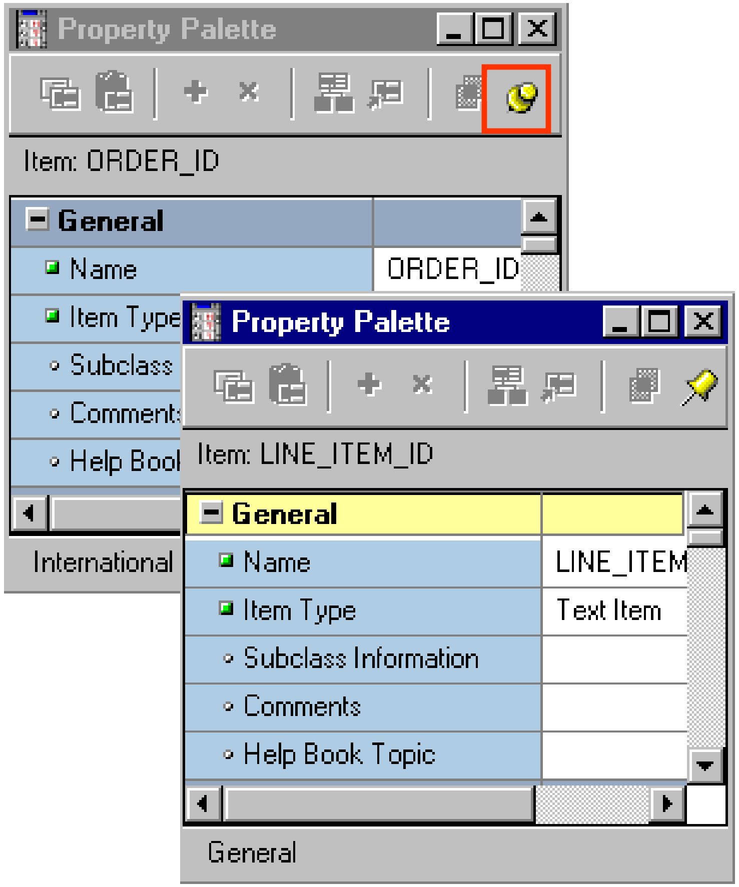
Layout Frame	
Layout Data Block	ORDERS
Update Layout	Manually
Layout Style	Form
Frame Alignment	Column
Single Object Alignment	Start
Horizontal Margin	5
Vertical Margin	14
Horizontal Object Offset	9
Vertical Object Offset	0
Allow Expansion	Yes
Shrinkwrap	No
Vertical Fill	Yes
Maximum Objects per Line	0
Start Prompt Alignment	Start
Start Prompt Offset	5
Top Prompt Alignment	Center
Top Prompt Offset	0

Controlling Frame Properties



Displaying Multiple Property Palettes

Two Palettes for Two Items:



Item: ORDER_ID

General	
Name	ORDER_ID
Item Type	
Subclass	
Comments	
Help Book	

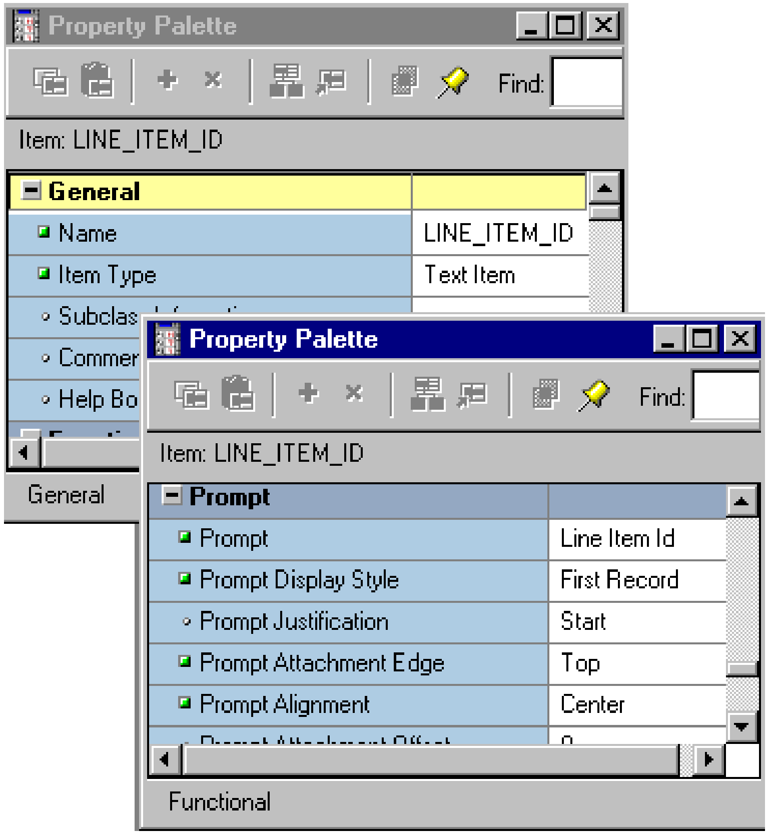
International

Item: LINE_ITEM_ID

General	
Name	LINE_ITEM
Item Type	Text Item
Subclass Information	
Comments	
Help Book Topic	

General

Two Palettes for One Item:



Item: LINE_ITEM_ID

General	
Name	LINE_ITEM_ID
Item Type	Text Item
Subclas	
Commer	
Help Bo	

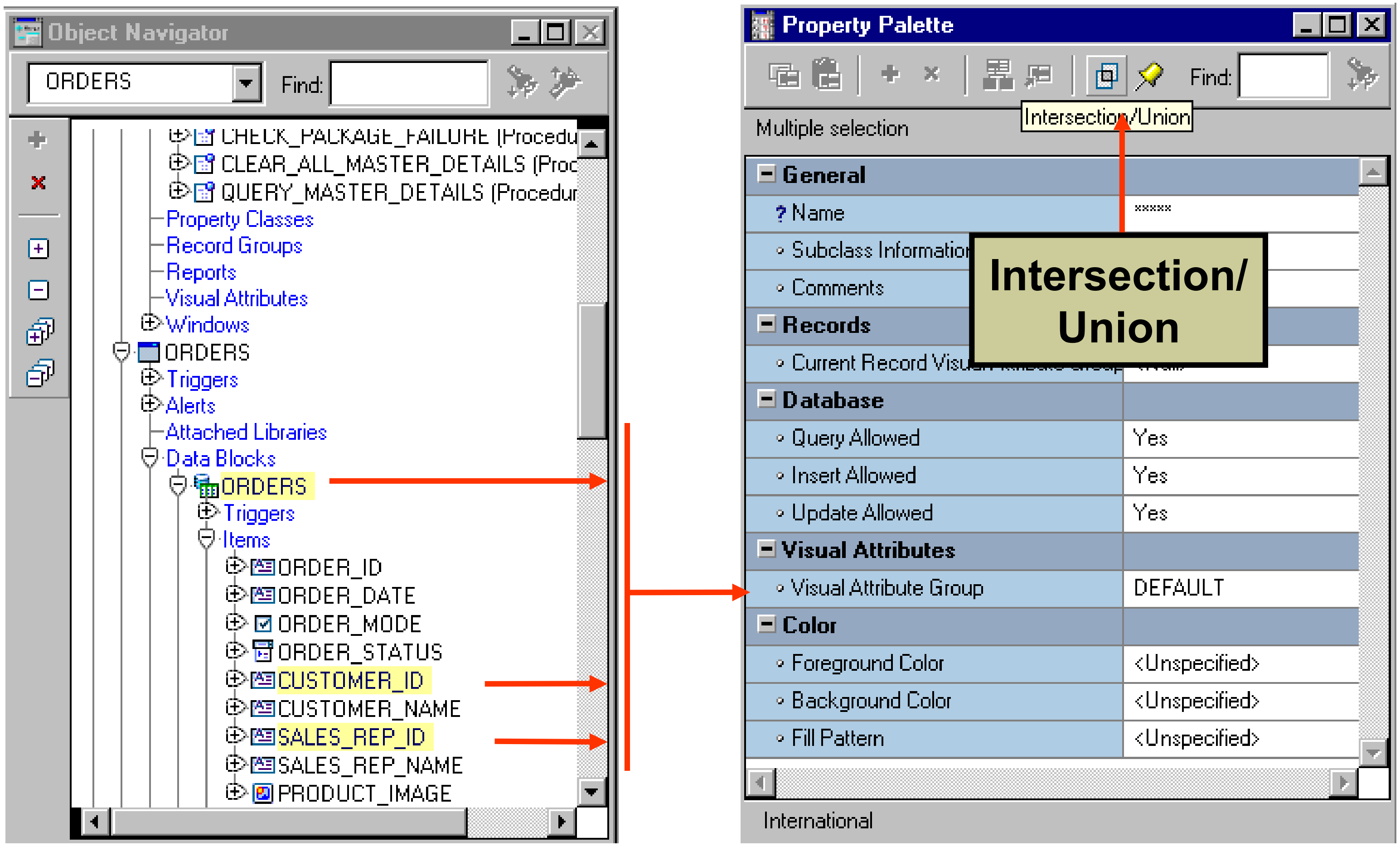
General

Item: LINE_ITEM_ID

Prompt	
Prompt	Line Item Id
Prompt Display Style	First Record
Prompt Justification	Start
Prompt Attachment Edge	Top
Prompt Alignment	Center

Functional

Setting Properties on Multiple Objects



Object Navigator

ORDERS Find:

```
CHECK_PACKAGE_FAILURE (Procedu
CLEAR_ALL_MASTER_DETAILS (Proc
QUERY_MASTER_DETAILS (Procedur
Property Classes
Record Groups
Reports
Visual Attributes
Windows
ORDERS
  Triggers
  Alerts
  Attached Libraries
  Data Blocks
    ORDERS
      Triggers
      Items
        ORDER_ID
        ORDER_DATE
        ORDER_MODE
        ORDER_STATUS
        CUSTOMER_ID
        CUSTOMER_NAME
        SALES_REP_ID
        SALES_REP_NAME
        PRODUCT_IMAGE
```

Property Palette

Find:

Intersection/Union

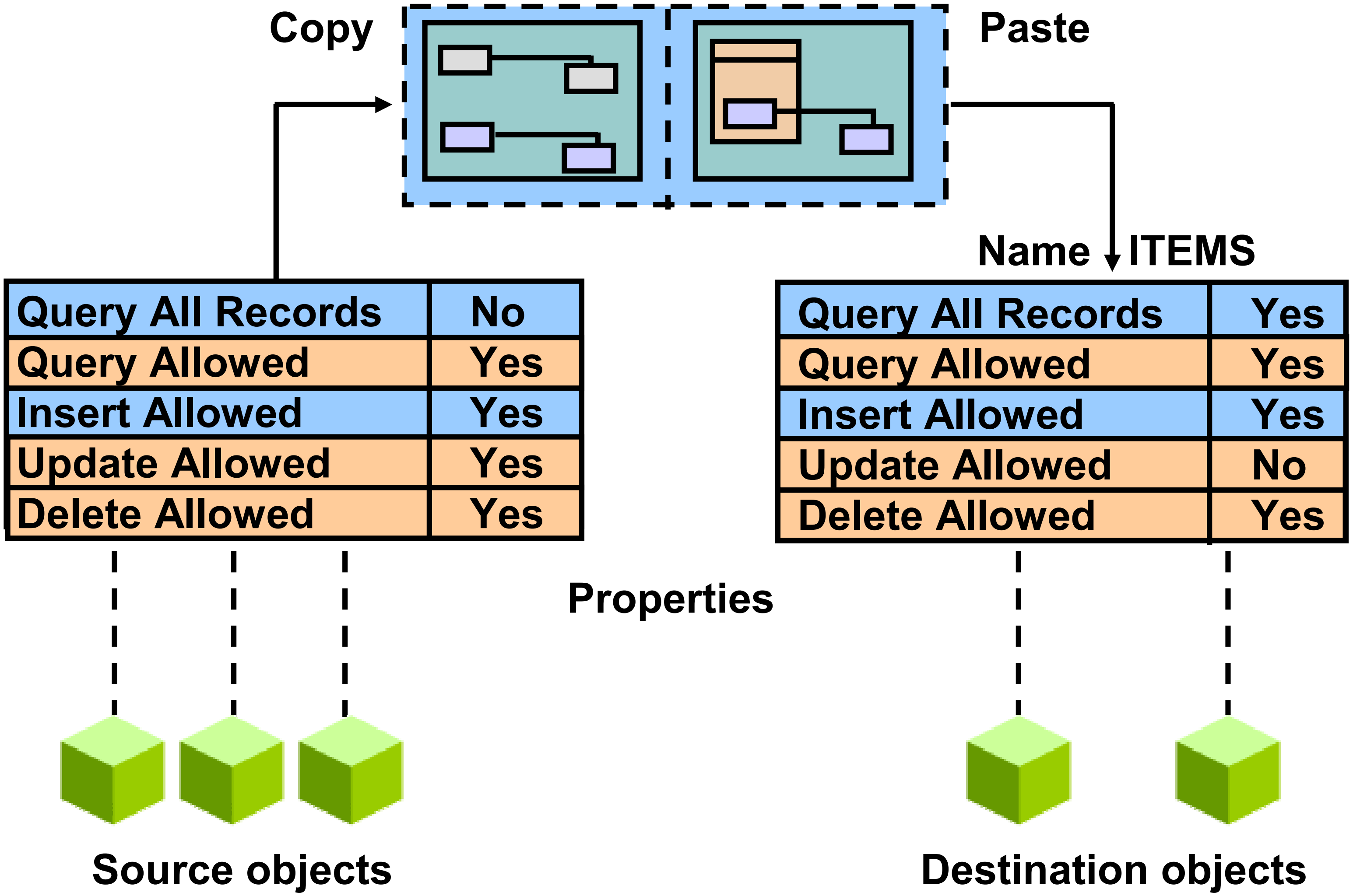
Multiple selection

General	
Name	*****
Subclass Information	
Comments	
Records	
Current Record Visual Attribute Group	<Null>
Database	
Query Allowed	Yes
Insert Allowed	Yes
Update Allowed	Yes
Visual Attributes	
Visual Attribute Group	DEFAULT
Color	
Foreground Color	<Unspecified>
Background Color	<Unspecified>
Fill Pattern	<Unspecified>

International

Intersection/ Union

Copying Properties



Copy

Paste

Name ITEMS

Query All Records	No
Query Allowed	Yes
Insert Allowed	Yes
Update Allowed	Yes
Delete Allowed	Yes

Query All Records	Yes
Query Allowed	Yes
Insert Allowed	Yes
Update Allowed	No
Delete Allowed	Yes

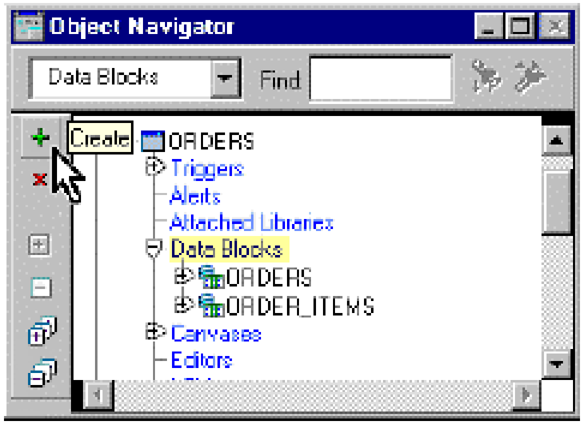
Properties

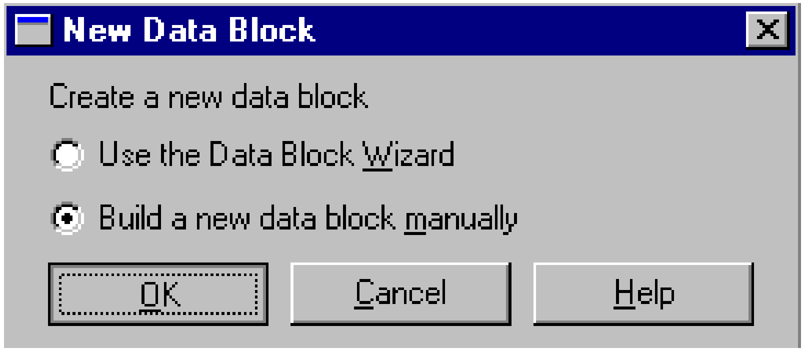
Source objects

Destination objects

Creating a Control Block

- Click the Data Blocks node
- Click the Create icon
  OR
  Select Edit > Create.
- Select the “Build a new data block manually” option in the New Data Block dialog box.




New Data Block

Create a new data block

○ Use the Data Block Wizard

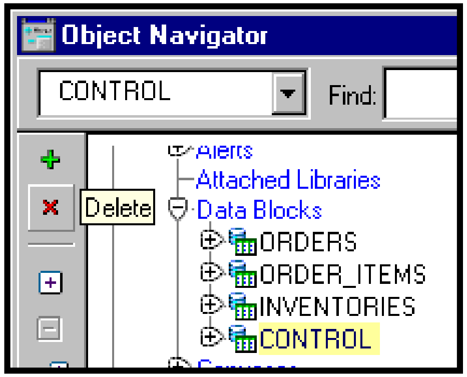
● Build a new data block manually

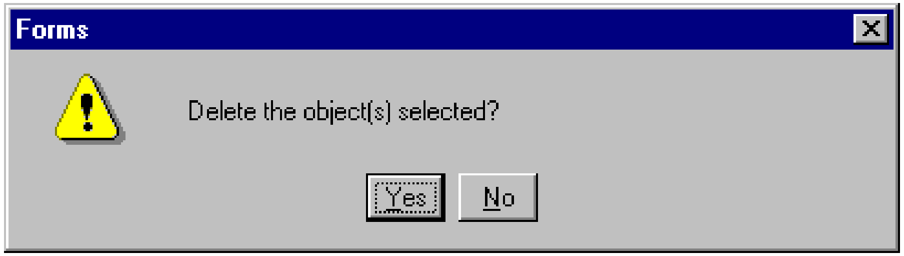
Deleting a Data Block

- Select a data block for deletion
- Click the Delete icon

  OR

  Press [Delete]
- Click Yes in the alert box.





Delete the object(s) selected?

Summary

In this lesson, you should have learned that:

- The Property Palette:
  - Contains property names and values that enable you to modify Forms objects
  - Has tools to search for properties, inherit properties, expand or collapse property categories, and pop up lists and dialog boxes for various properties
  - Shows different icons for default, changed, inherited, and overridden properties
- Block properties control the behavior and appearance of data blocks
- Frame properties control how block items are arranged
- You can create blocks that do not directly correspond to database tables by choosing to create the block manually rather than using the Data Block Wizard
- Deleting a data block deletes all of its components

Practice 6 Overview

This practice covers the following topics:

- Creating a control block
- Creating a Visual Attribute
- Invoking context-sensitive help from the Property Palette
- Modifying data block properties
- Modifying frame properties

7

Working with Text Items

Objectives

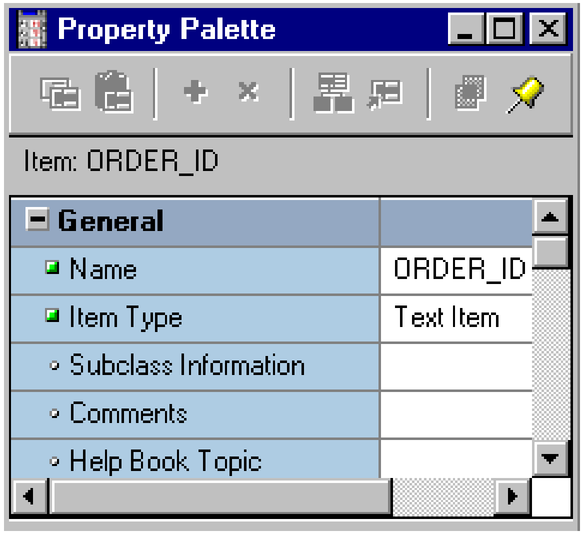
After completing this lesson, you should be able to do the following:

- Describe text items
- Create a text item
- Modify the appearance of a text item
- Control the data in a text item
- Alter the navigational behavior of a text item
- Enhance the relationship between the text item and the database
- Add functionality to a text item
- Display helpful messages

Text Item Overview

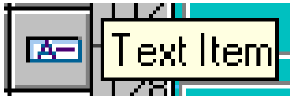
What is a text item?

- Default item type
- Interface object for:
  - Querying
  - Inserting
  - Updating
  - Deleting
- Behavior defined in the Property Palette



Item: ORDER_ID

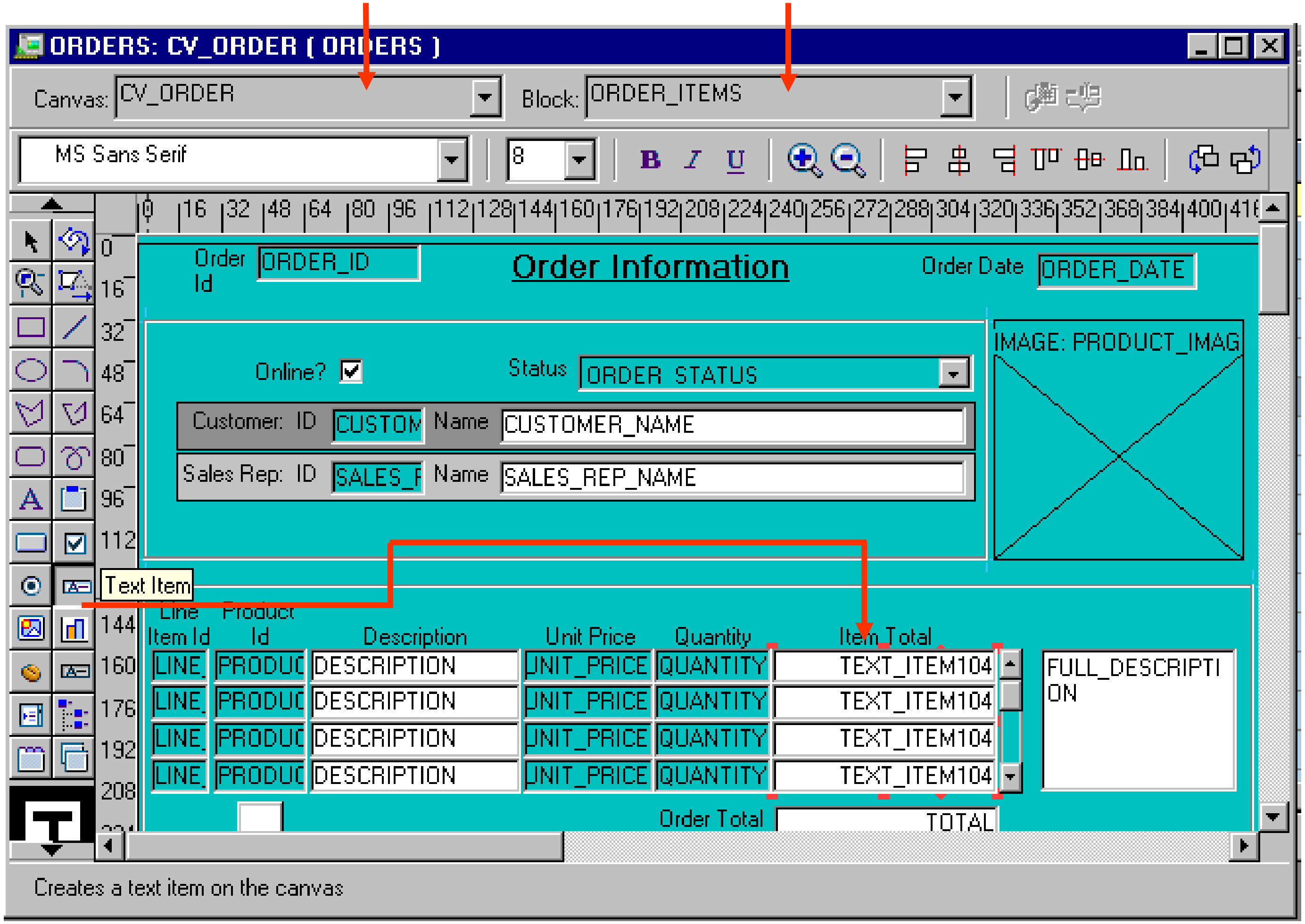
General	
Name	ORDER_ID
Item Type	Text Item
Subclass Information	
Comments	
Help Book Topic	



Creating a Text Item

Canvas selection

Block selection



ORDERS: CV_ORDER (ORDERS)

Canvas: CV_ORDER

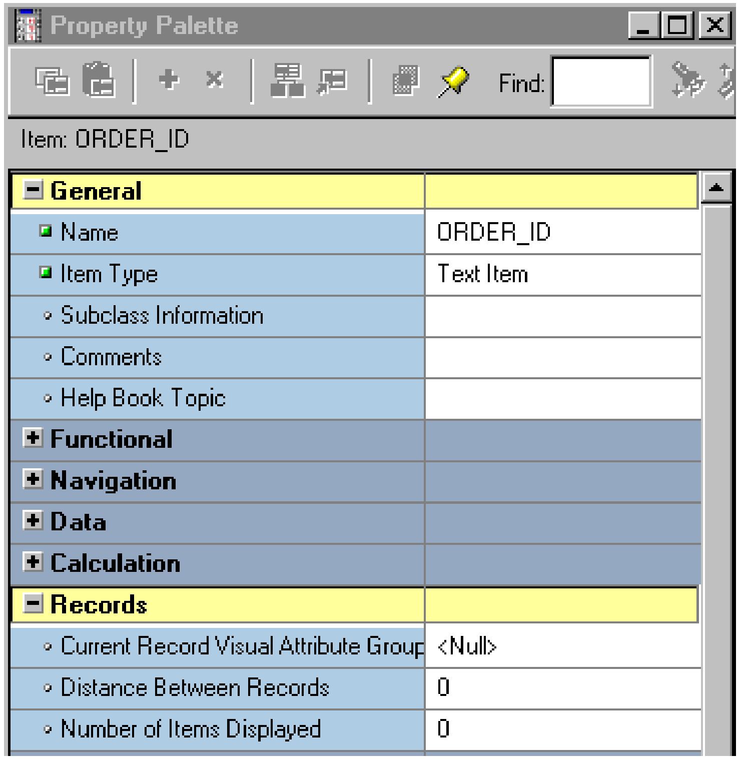
Block: ORDER_ITEMS

MS Sans Serif 8

Text Item

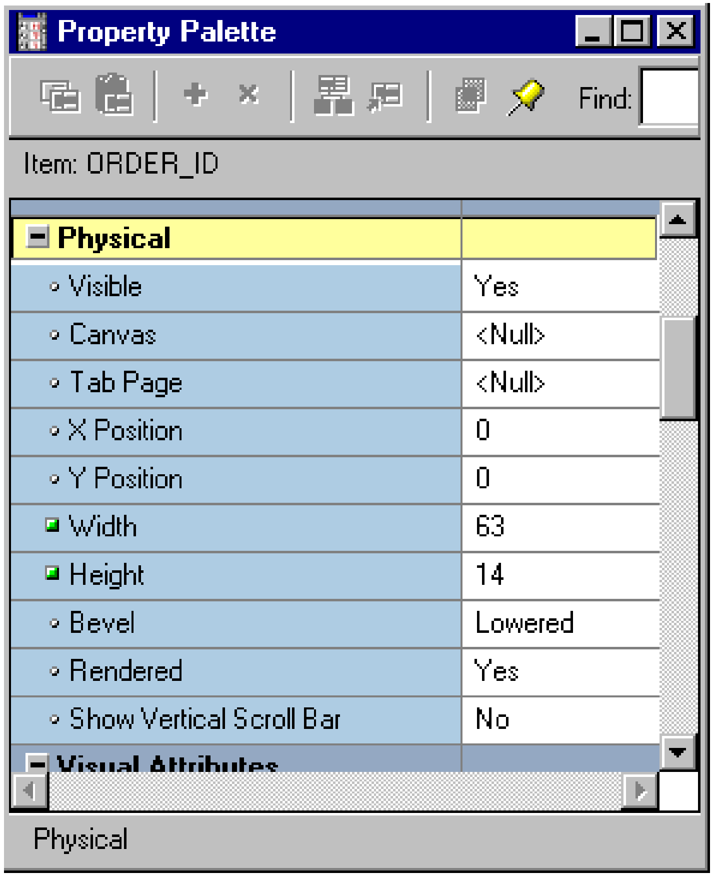
Creates a text item on the canvas

Modifying the Appearance of a Text Item: General and Physical Properties



Item: ORDER_ID

General	
Name	ORDER_ID
Item Type	Text Item
Subclass Information	
Comments	
Help Book Topic	
Functional	
Navigation	
Data	
Calculation	
Records	
Current Record Visual Attribute Group	<Null>
Distance Between Records	0
Number of Items Displayed	0

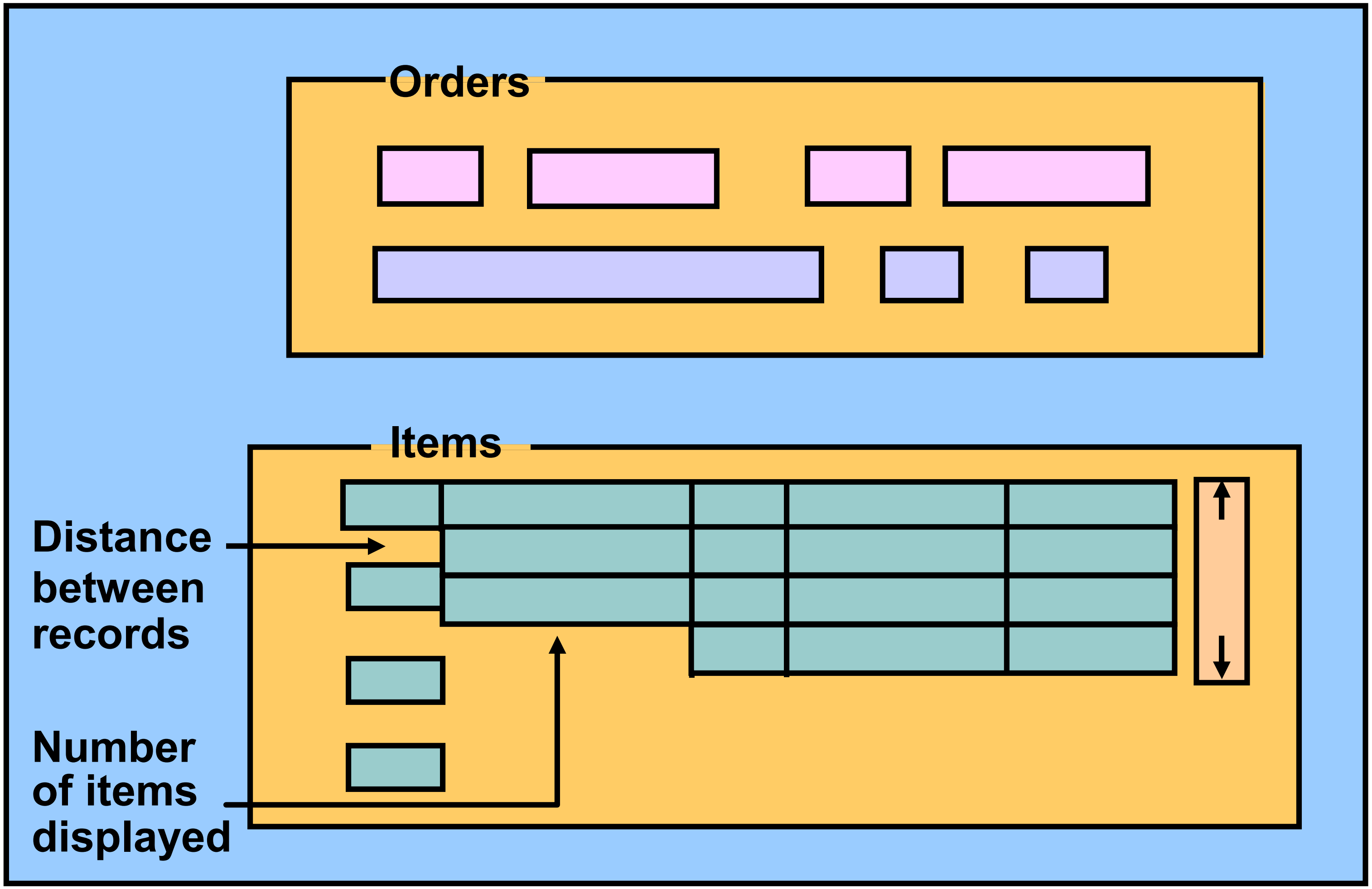


Item: ORDER_ID

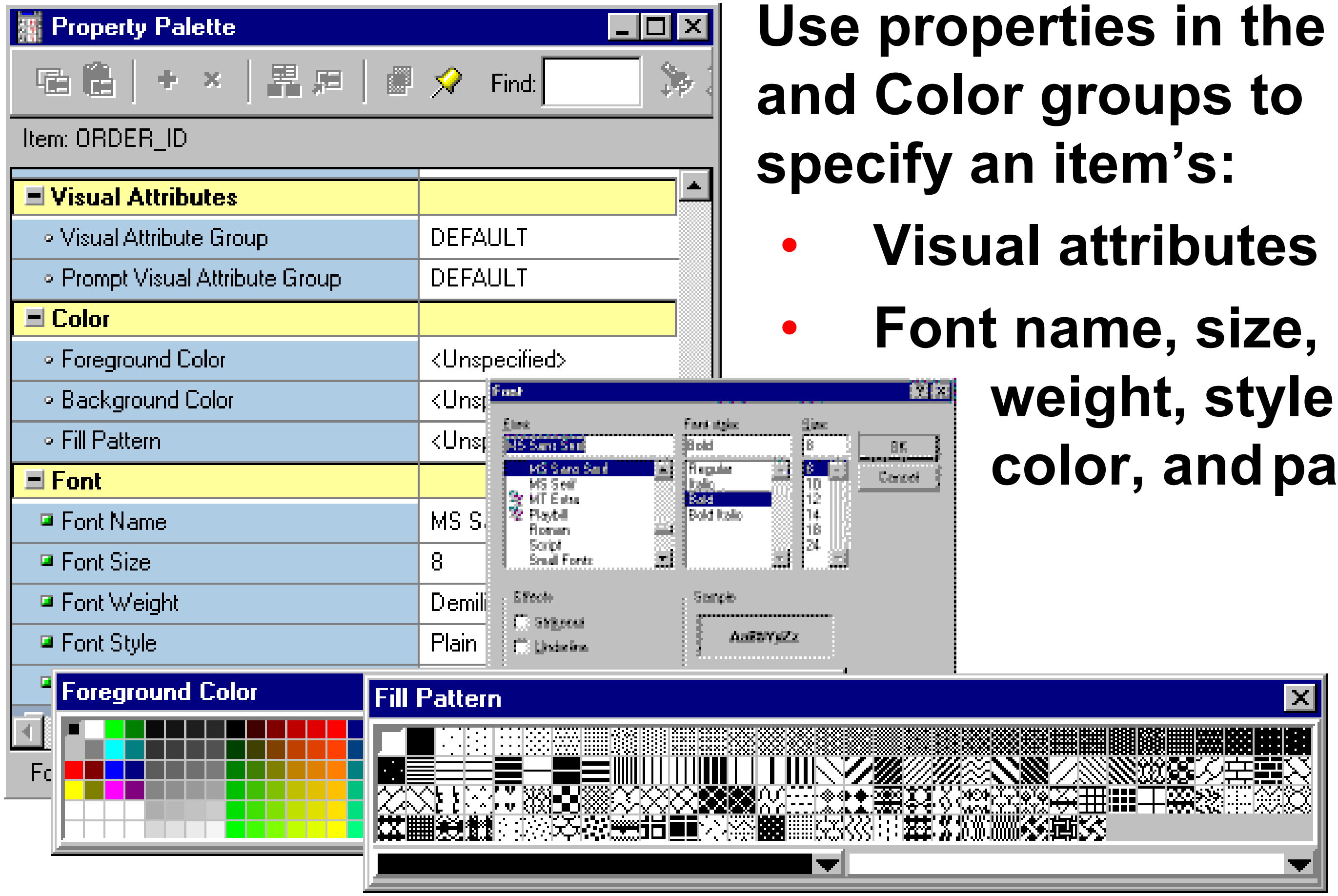
Physical	
Visible	Yes
Canvas	<Null>
Tab Page	<Null>
X Position	0
Y Position	0
Width	63
Height	14
Bevel	Lowered
Rendered	Yes
Show Vertical Scroll Bar	No
Visual Attributes	

Physical

Modifying the Appearance of a Text Item: Records Properties



Modifying the Appearance of a Text Item: Font and Color Properties



Item: ORDER_ID

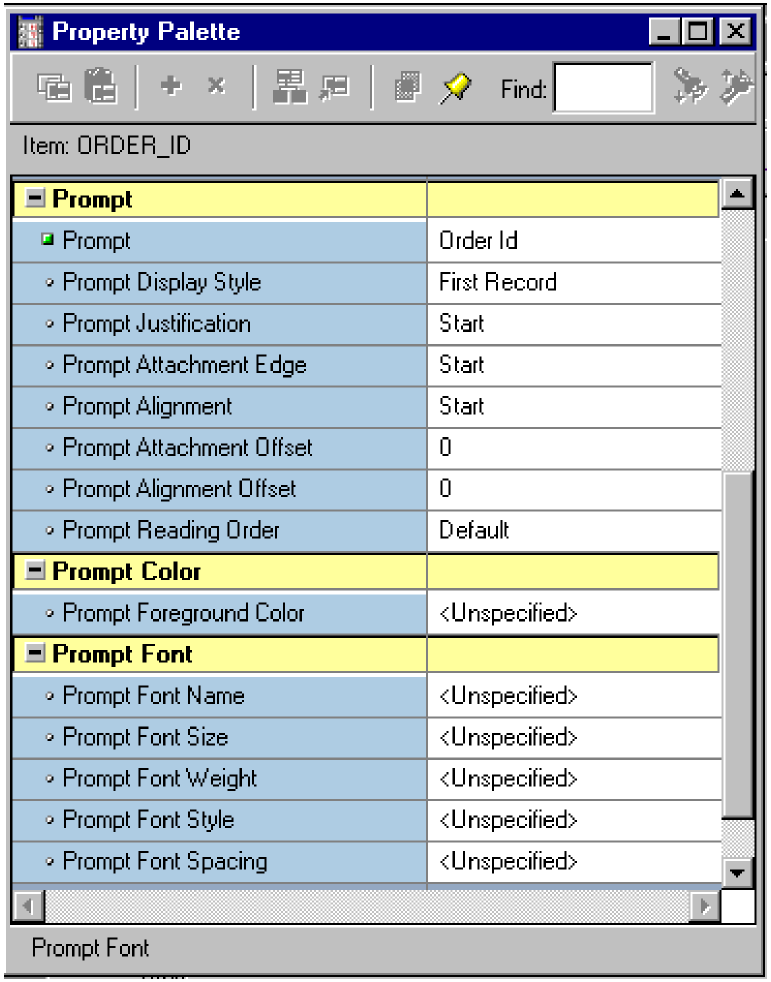
Visual Attributes	
Visual Attribute Group	DEFAULT
Prompt Visual Attribute Group	DEFAULT
Color	
Foreground Color	<Unspecified>
Background Color	<Unsp
Fill Pattern	<Unsp
Font	
Font Name	MS S
Font Size	8
Font Weight	Demili
Font Style	Plain

Font: MS Sans Serif
Font style: Bold
Size: 8

Use properties in the Font and Color groups to specify an item’s:

- Visual attributes
- Font name, size, weight, style, color, and pattern

Modifying the Appearance of a Text Item: Prompts



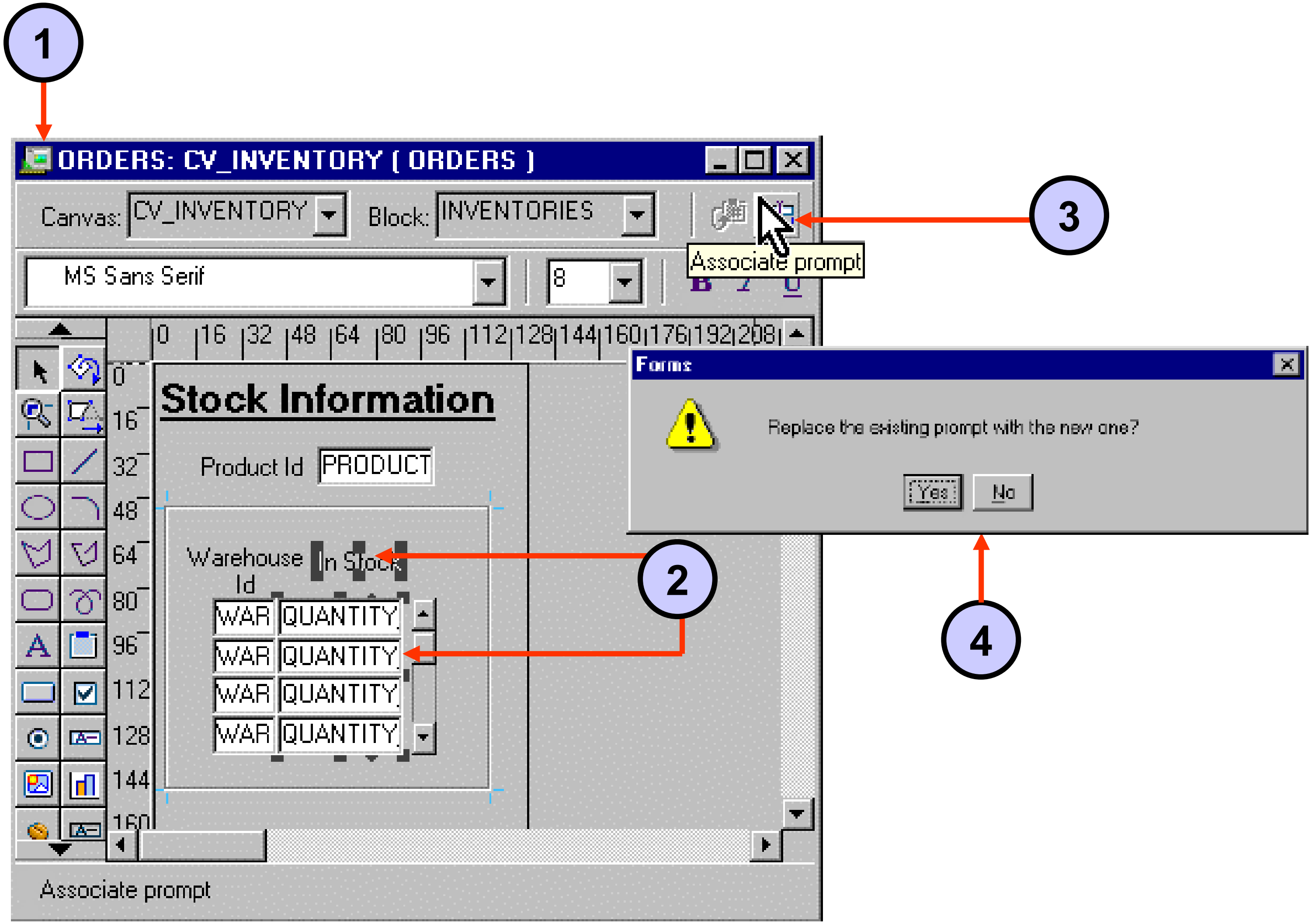
Item: ORDER_ID

Prompt	
Prompt	Order Id
Prompt Display Style	First Record
Prompt Justification	Start
Prompt Attachment Edge	Start
Prompt Alignment	Start
Prompt Attachment Offset	0
Prompt Alignment Offset	0
Prompt Reading Order	Default
Prompt Color	
Prompt Foreground Color	<Unspecified>
Prompt Font	
Prompt Font Name	<Unspecified>
Prompt Font Size	<Unspecified>
Prompt Font Weight	<Unspecified>
Prompt Font Style	<Unspecified>
Prompt Font Spacing	<Unspecified>

Prompt Font

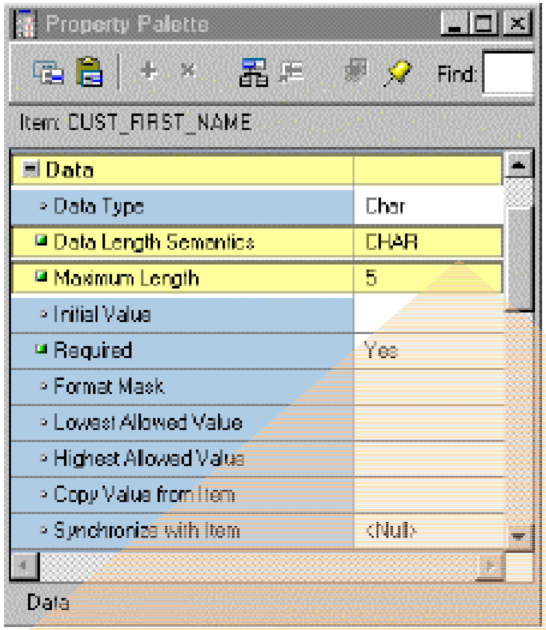
- A prompt specifies the text label that is associated with an item.
- Several properties are available to arrange and manage prompts.
- Use prompt properties to change the appearance of an item prompt.

Associating Text with an Item Prompt



Replace the existing prompt with the new one?

Controlling the Data of a Text Item



Property Palette

Find:

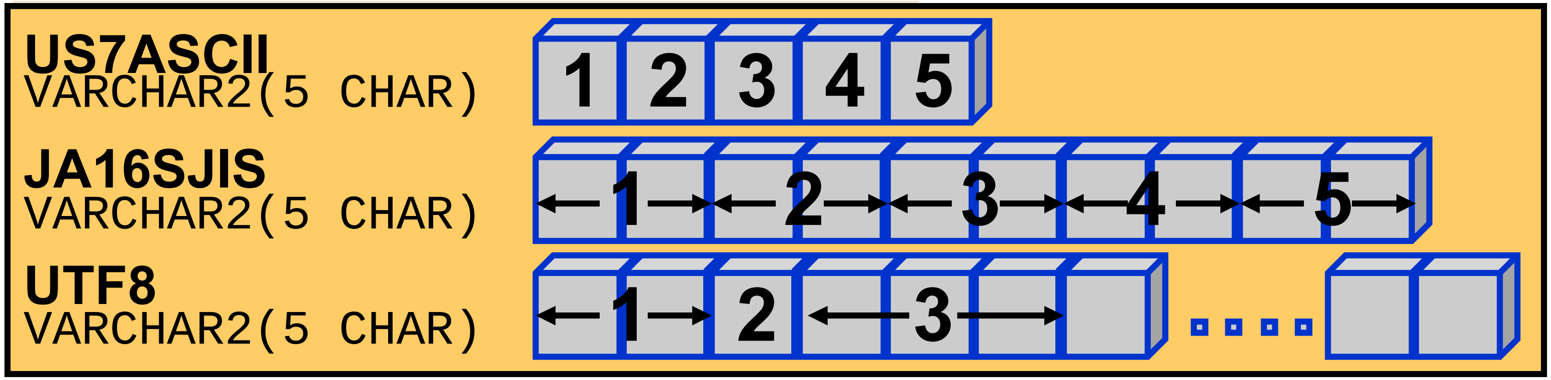
Item: CUST_FIRST_NAME

Data	
Data Type	Char
Data Length Semantics	CHAR
Maximum Length	5
Initial Value	
Required	Yes
Format Mask	
Lowest Allowed Value	
Highest Allowed Value	
Copy Value from Item	
Synchronize with Item	<Null>

Data

Use properties in Data group to control the data:

- Type
- Length
- Format
- Value



US7ASCII
VARCHAR2(5 CHAR)

JA16SJIS
VARCHAR2(5 CHAR)

UTF8
VARCHAR2(5 CHAR)

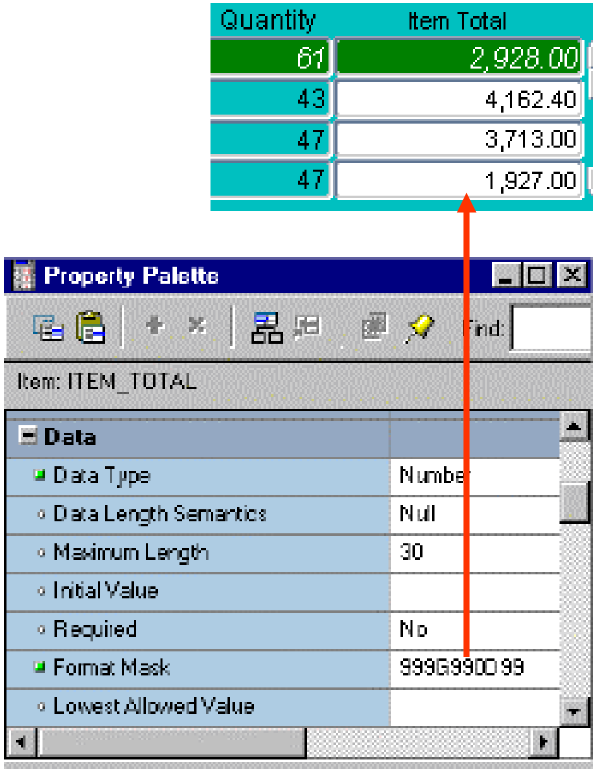
Controlling the Data of a Text Item: Format

Format masks:

- Standard SQL formats
  - Dates FXDD-MON-YY
  - Numbers L099G990D99
- Nonstandard formats

  Use double quotes for embedded characters
  “(“099”)”099“-”0999

Note: Allow for format mask’s embedded characters when defining Width property.



Quantity	Item Total
61	2,928.00
43	4,162.40
47	3,713.00
47	1,927.00

Property Palette

Find:

Item: ITEM_TOTAL

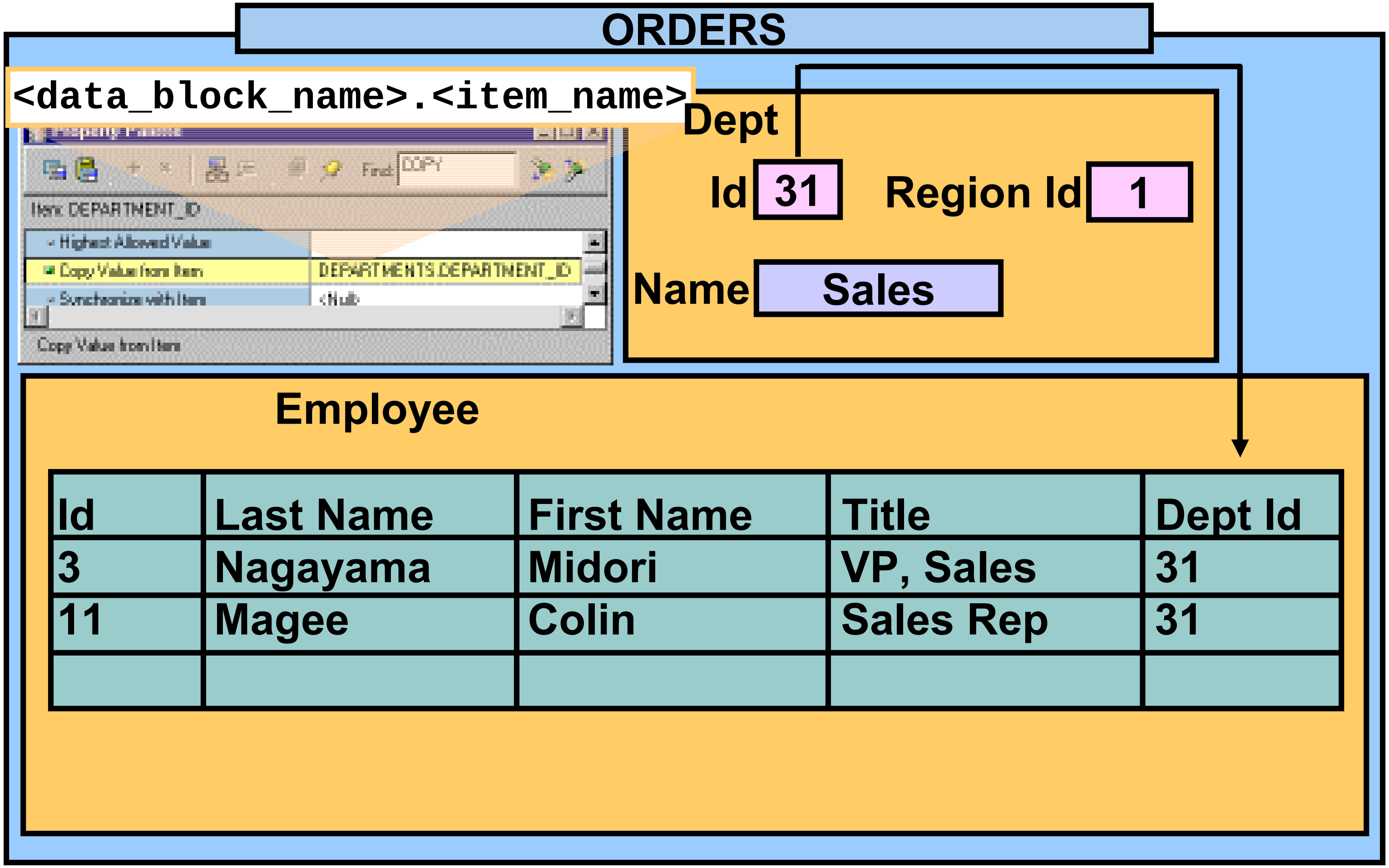
Data	
Data Type	Numbe
Data Length Semantics	Nul
Maximum Length	30
Initial Value	
Required	No
Format Mask	999G990D99
Lowest Allowed Value	

Controlling the Data of a Text Item: Values

Initial Values:

- Are used for every new record
- Can be overwritten
- Must be compatible with item’s data type
- Use:
  - Raw value
  - System variable
  - Global variable
  - Form parameter
  - Form item
  - Sequence

Controlling the Data of a Text Item: Copy Value from Item



`<data_block_name>.<item_name>`

Item: DEPARTMENT_ID

Highest Allowed Value	
Copy Value from Item	DEPARTMENTS.DEPARTMENT_ID
Synchronize with Item	<Null>

Copy Value from Item

Dept

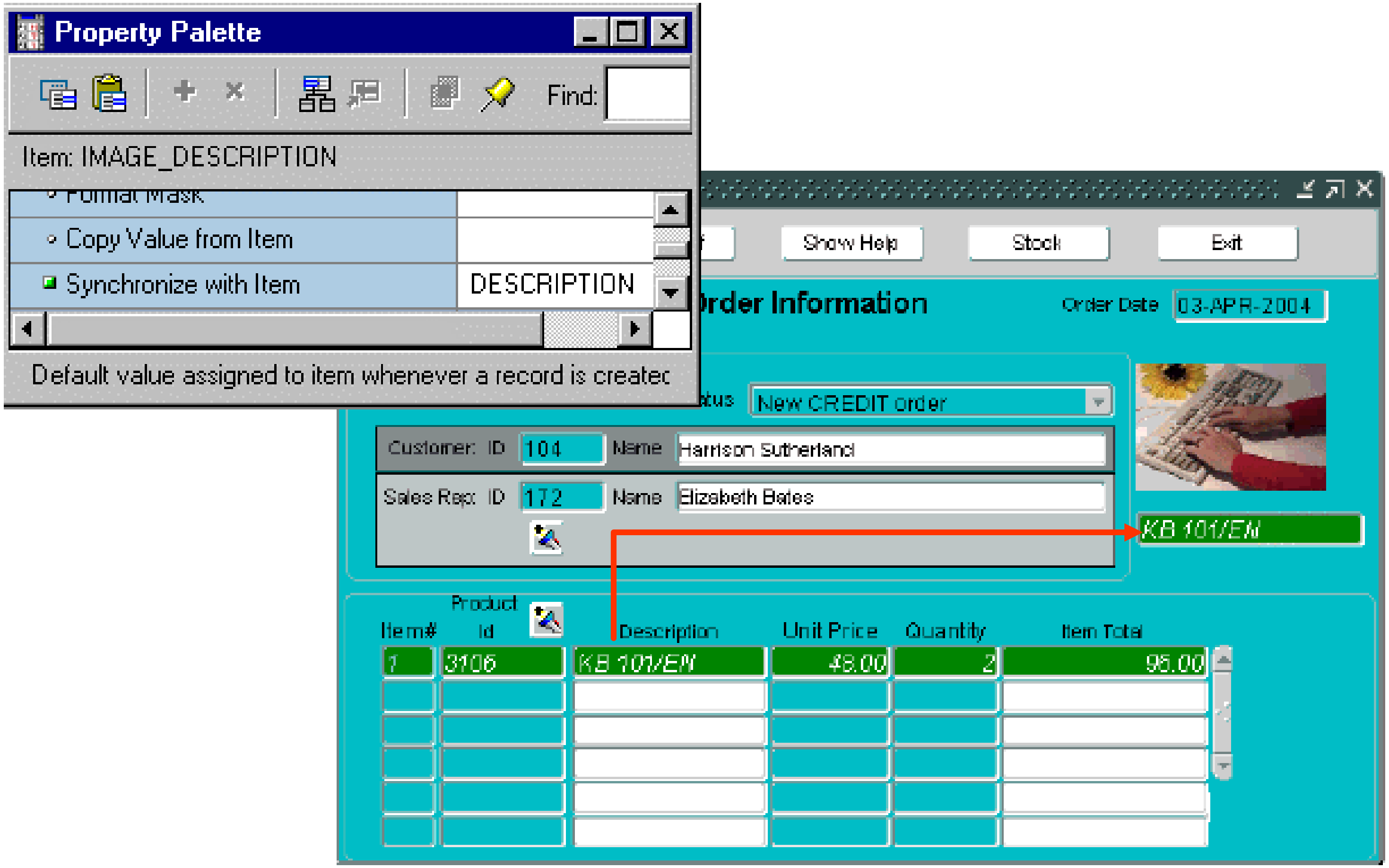
Id: 31 Region Id: 1

Name: Sales

Employee

Id	Last Name	First Name	Title	Dept Id
3	Nagayama	Midori	VP, Sales	31
11	Magee	Colin	Sales Rep	31

Controlling the Data of a Text Item: Synchronize with Item



Item: IMAGE_DESCRIPTION

Format Mask	
Copy Value from Item	
Synchronize with Item	DESCRIPTION

Default value assigned to item whenever a record is createc

rder Information

Order Date 03-APR-2004

atus New CREDIT order

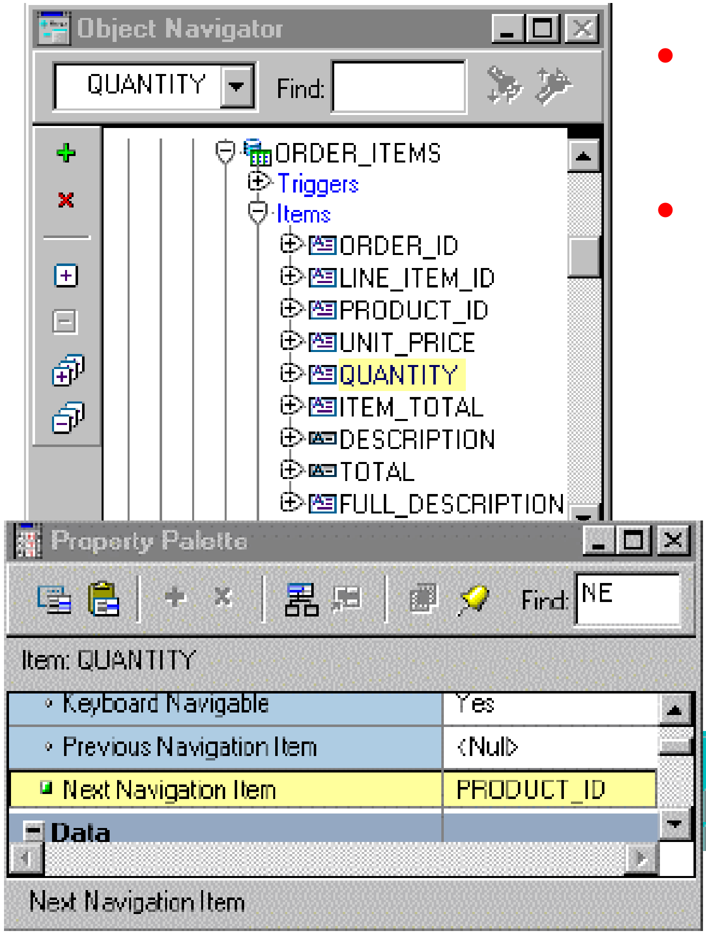
Customer ID 104 Name Harrison Sutherland

Sales Rep ID 172 Name Elizabeth Bates

KB 101/EN

Item#	Product Id	Description	Unit Price	Quantity	Item Total
1	3106	KB 101/EN	48.00	2	96.00

Altering Navigational Behavior of Text Items



Object Navigator

QUANTITY Find:

- ORDER_ITEMS
  - Triggers
  - Items
    - ORDER_ID
    - LINE_ITEM_ID
    - PRODUCT_ID
    - UNIT_PRICE
    - QUANTITY
    - ITEM_TOTAL
    - DESCRIPTION
    - TOTAL
    - FULL_DESCRIPTION

Property Palette

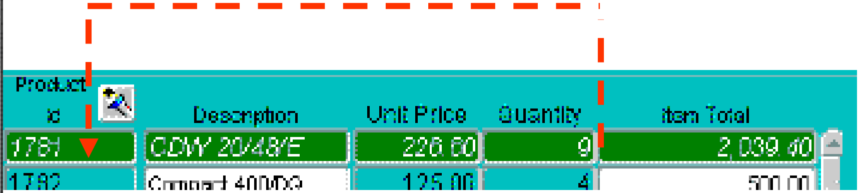
Find: NE

Item: QUANTITY

Keyboard Navigable	Yes
Previous Navigation Item	<Null>
Next Navigation Item	PRODUCT_ID
Data	

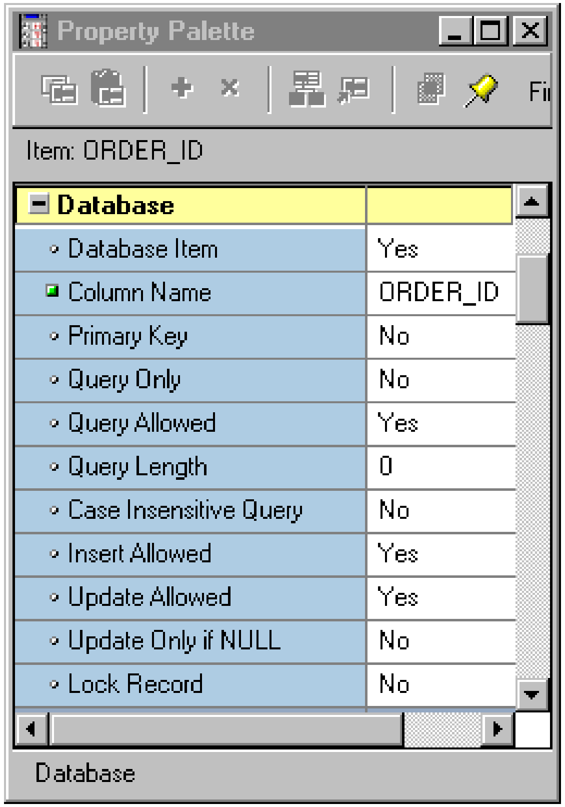
Next Navigation Item

- Established by order of entries in Object Navigator
- Alter by:
  - Keyboard Navigable
  - Previous Navigation Item
  - Next Navigation Item



Product Id	Description	Unit Price	Quantity	Item Total
1781	CDW 20/48/E	226.60	9	2,039.40
1782	Compact 400/DQ	125.00	4	500.00

Enhancing the Relationship Between Text Item and Database



Item: ORDER_ID

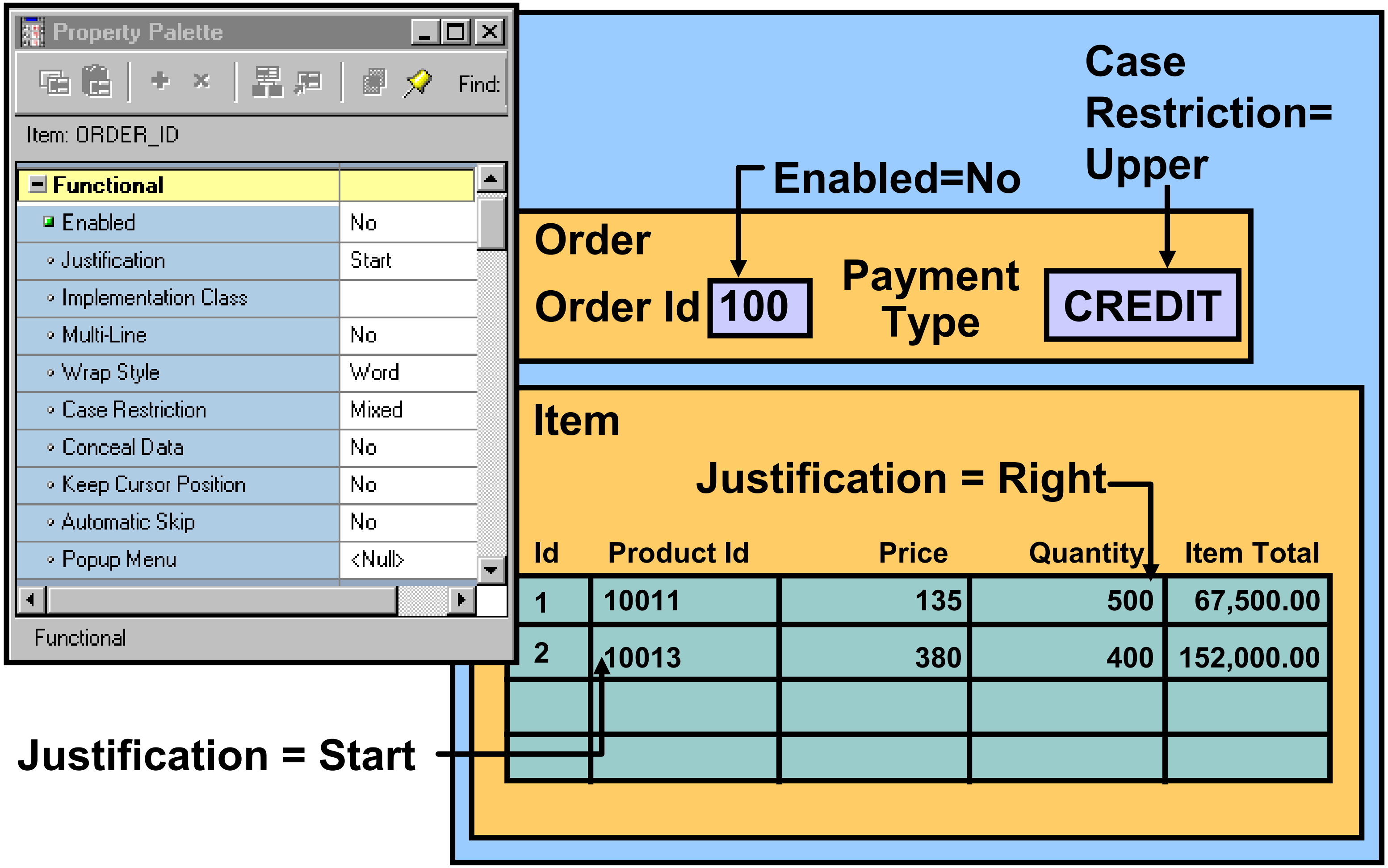
Database	
Database Item	Yes
Column Name	ORDER_ID
Primary Key	No
Query Only	No
Query Allowed	Yes
Query Length	0
Case Insensitive Query	No
Insert Allowed	Yes
Update Allowed	Yes
Update Only if NULL	No
Lock Record	No

Database

Use properties in the Database group to control:

- Item’s data source—base table item or control item
- Query, insert, and update operations on an item
- Maximum query length
- Query case

Adding Functionality to a Text Item



Property Palette

Item: ORDER_ID

Functional	
Enabled	No
Justification	Start
Implementation Class	
Multi-Line	No
Wrap Style	Word
Case Restriction	Mixed
Conceal Data	No
Keep Cursor Position	No
Automatic Skip	No
Popup Menu	<Null>

Functional

Case Restriction= Upper

Enabled=No

Order

Order Id 100

Payment Type CREDIT

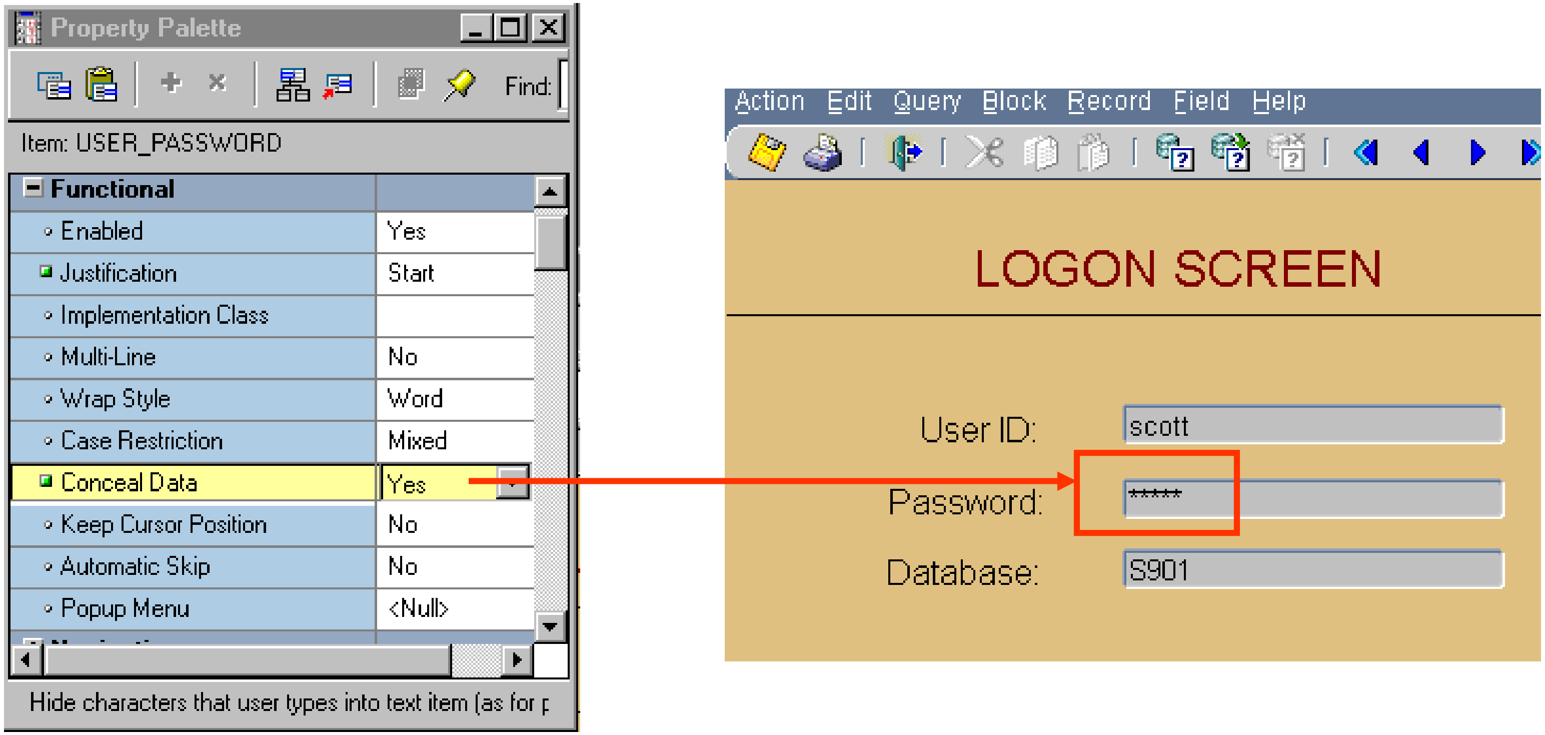
Item

Justification = Right

Id	Product Id	Price	Quantity	Item Total
1	10011	135	500	67,500.00
2	10013	380	400	152,000.00

Justification = Start

Adding Functionality to a Text Item: Conceal Data Property



Item: USER_PASSWORD

Functional	
Enabled	Yes
Justification	Start
Implementation Class	
Multi-Line	No
Wrap Style	Word
Case Restriction	Mixed
Conceal Data	Yes
Keep Cursor Position	No
Automatic Skip	No
Popup Menu	<Null>

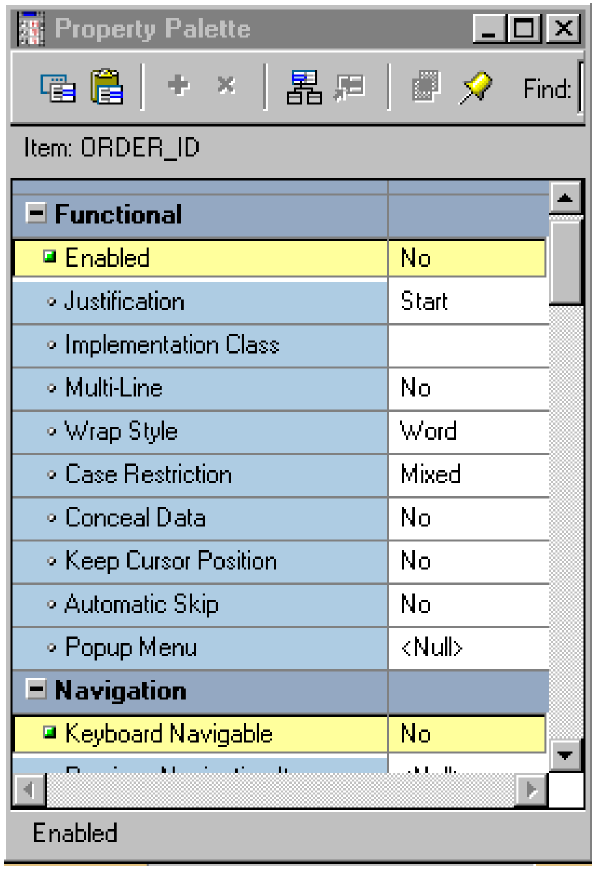
Hide characters that user types into text item (as for p

User ID: scott

Password: *****

Database: S901

Adding Functionality to a Text Item: Keyboard Navigable and Enabled



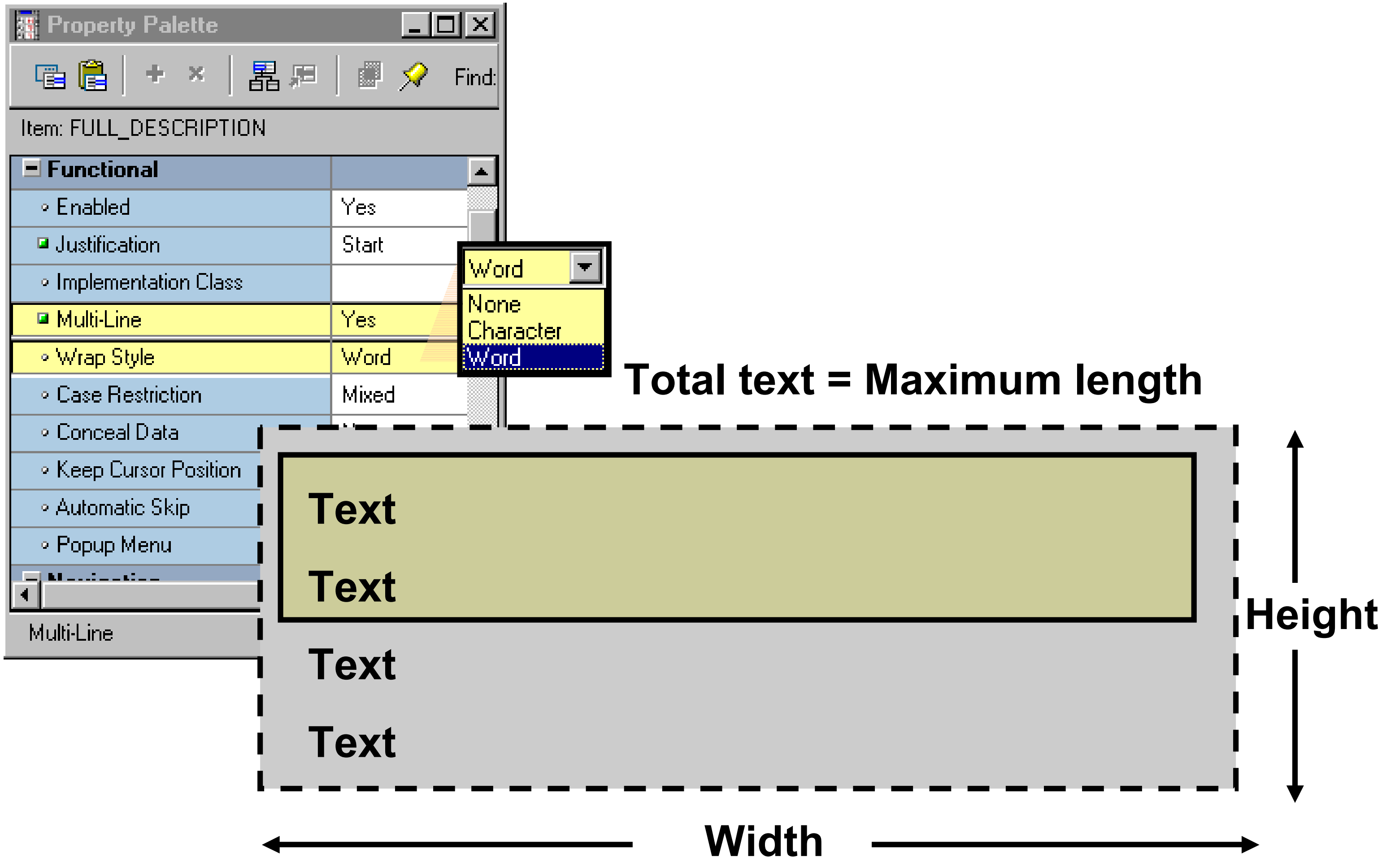
Item: ORDER_ID

Functional	
Enabled	No
Justification	Start
Implementation Class	
Multi-Line	No
Wrap Style	Word
Case Restriction	Mixed
Conceal Data	No
Keep Cursor Position	No
Automatic Skip	No
Popup Menu	<Null>
Navigation	
Keyboard Navigable	No

Enabled

- Set both properties to allow or disallow navigation and interaction with text item.
- When Enabled is set to Yes, Keyboard Navigable can be set to Yes or No.
- When Enabled is set to No, the item is always nonnavigable.

Adding Functionality to a Text Item: Multi-line Text Items



Property Palette

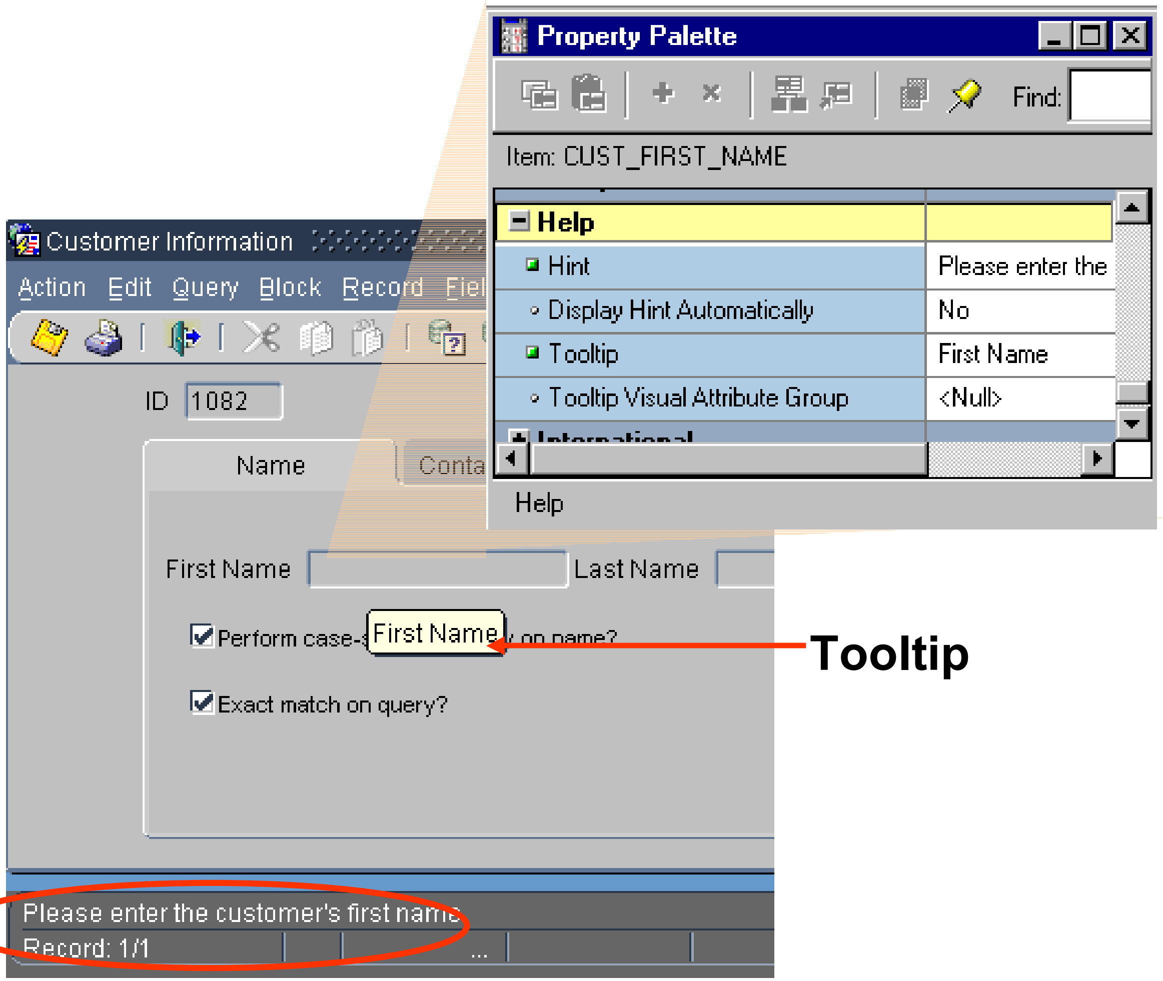
Item: FULL_DESCRIPTION

Functional	
Enabled	Yes
Justification	Start
Implementation Class	
Multi-Line	Yes
Wrap Style	Word
Case Restriction	Mixed
Conceal Data	
Keep Cursor Position	
Automatic Skip	
Popup Menu	

Multi-Line

Total text = Maximum length

Displaying Helpful Messages: Help Properties



Item: CUST_FIRST_NAME

Help	
Hint	Please enter the
Display Hint Automatically	No
Tooltip	First Name
Tooltip Visual Attribute Group	<Null>
International	

ID 1082

First Name

Last Name

☑ Perform case-s First Name y on name?

☑ Exact match on query?

Please enter the customer's first name

Record: 1/1

Tooltip

Hint

Summary

In this lesson, you should have learned that:

- Text items are interface objects that usually correspond to database columns
- You can create a text item with:
  - The Text Item tool in the Layout Editor
  - The Create icon in the Object Navigator
  - The Data Block Wizard

Summary

- You can modify a text item in its Property Palette:
  - General, Records, and Physical properties control the appearance of the text item
  - Data properties control the length, datatype, format, and other aspects of the data.
  - Navigation properties control how to navigate to and from a text item.
  - Database properties specify the relationship between the text item and its corresponding database column.
  - Functional properties control how the text item functions.
  - Help properties specify the display of helpful messages.

Practice 7 Overview

This practice covers the following topics:

- Deleting text items
- Modifying text item properties
- Creating text items

8

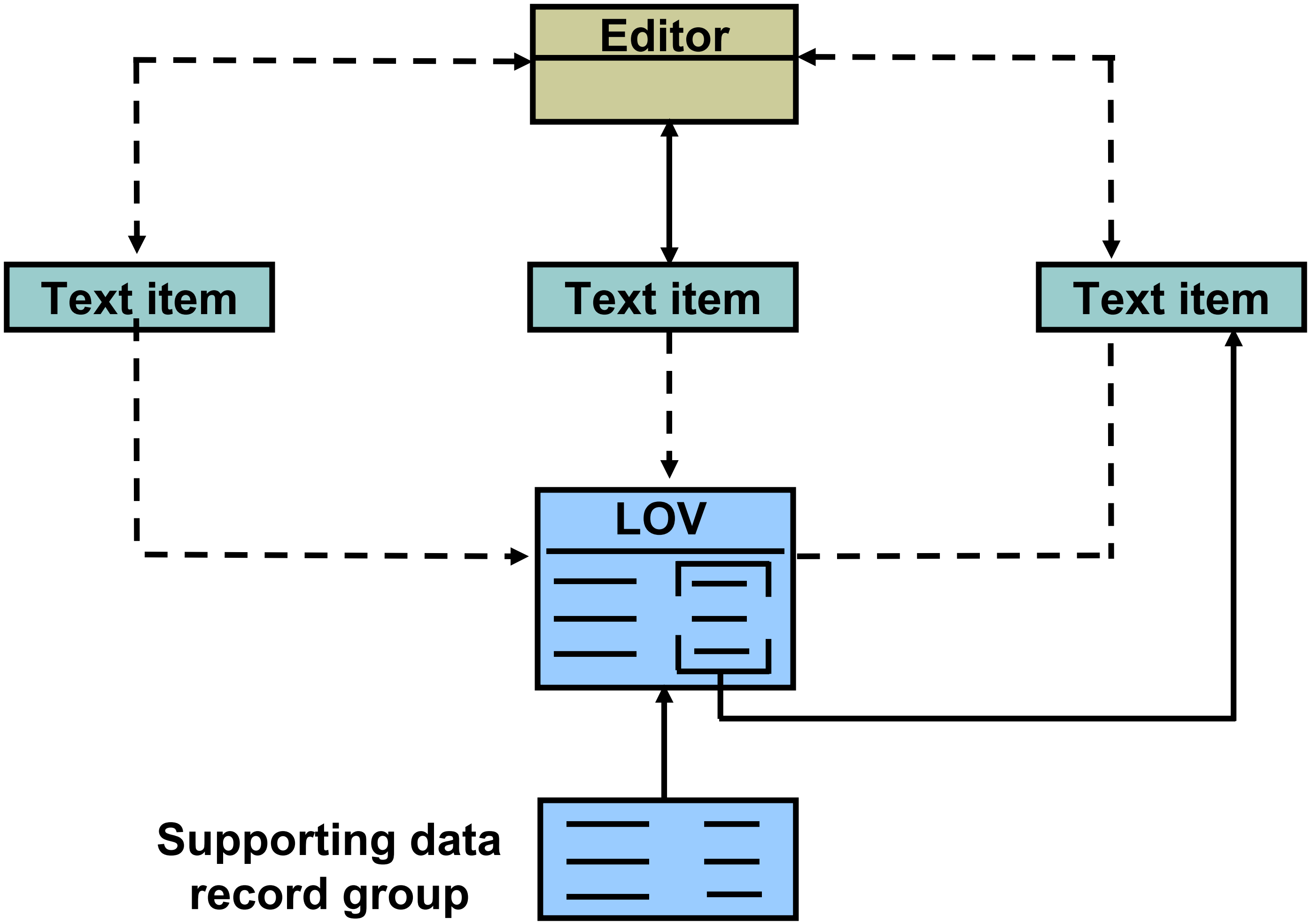
Creating LOVs and Editors

Objectives

After completing this lesson, you should be able to do the following:

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

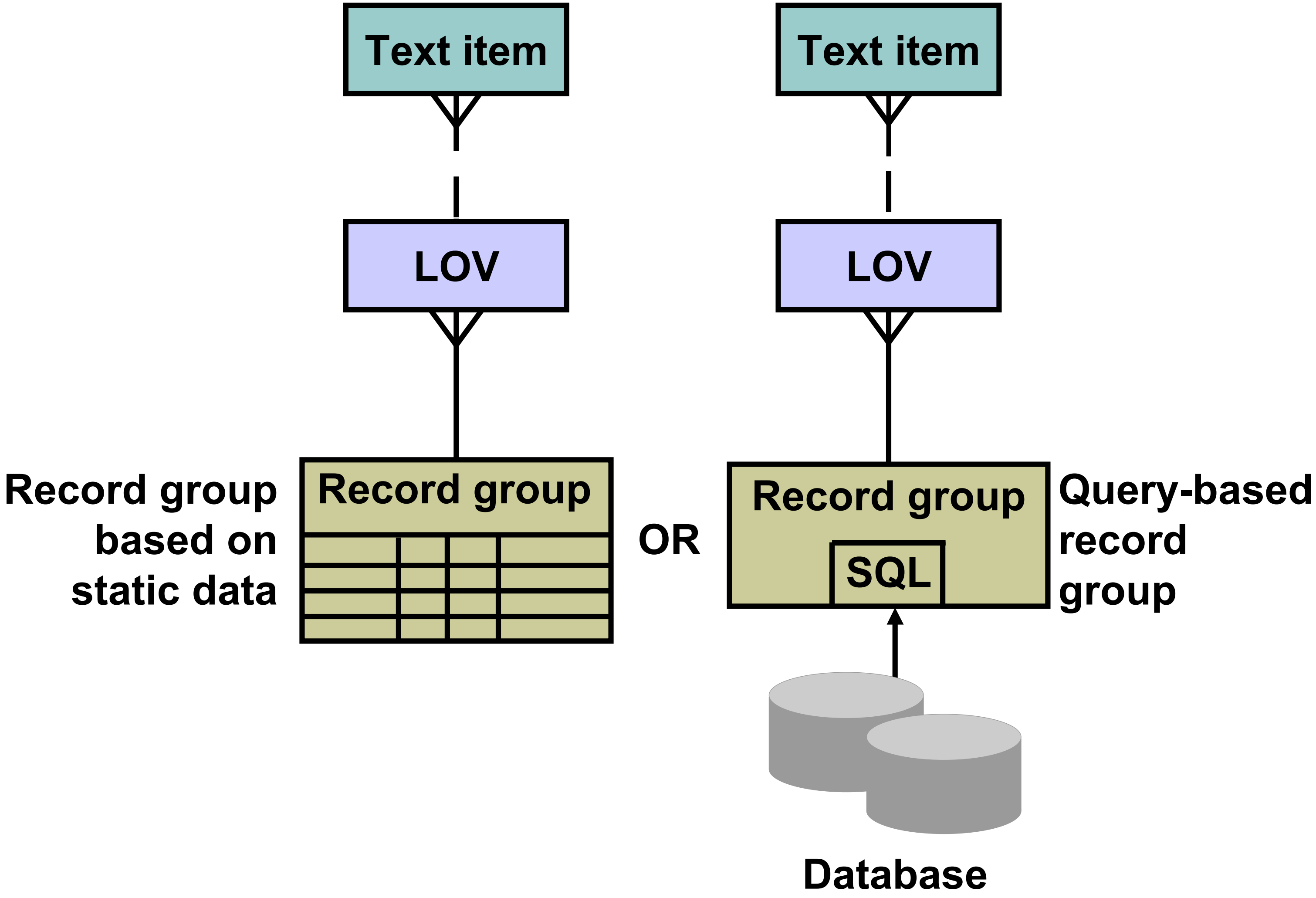
Overview of LOVs and Editors



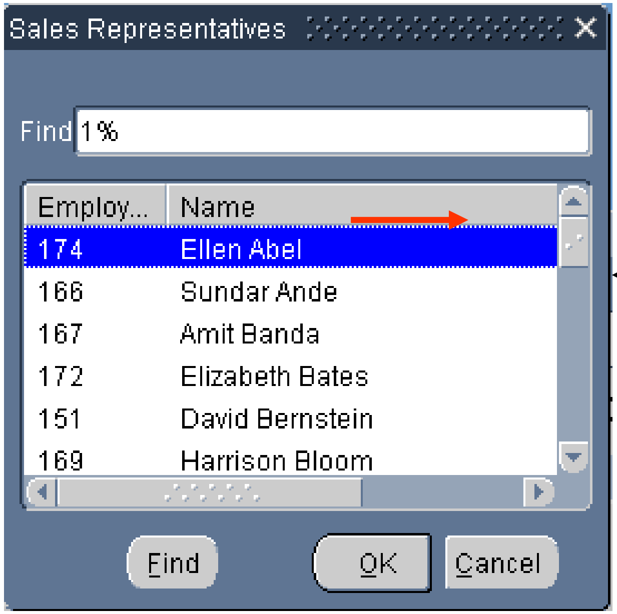
Overview of LOVs and Editors

- LOVs
  - List of values for text items
  - Dynamic or static list
  - Independent of single text items
  - Flexible and efficient
- Editors
  - Override default editor
  - Used for special requirements such as larger editing window, position, color, and title

LOVs and Record Groups



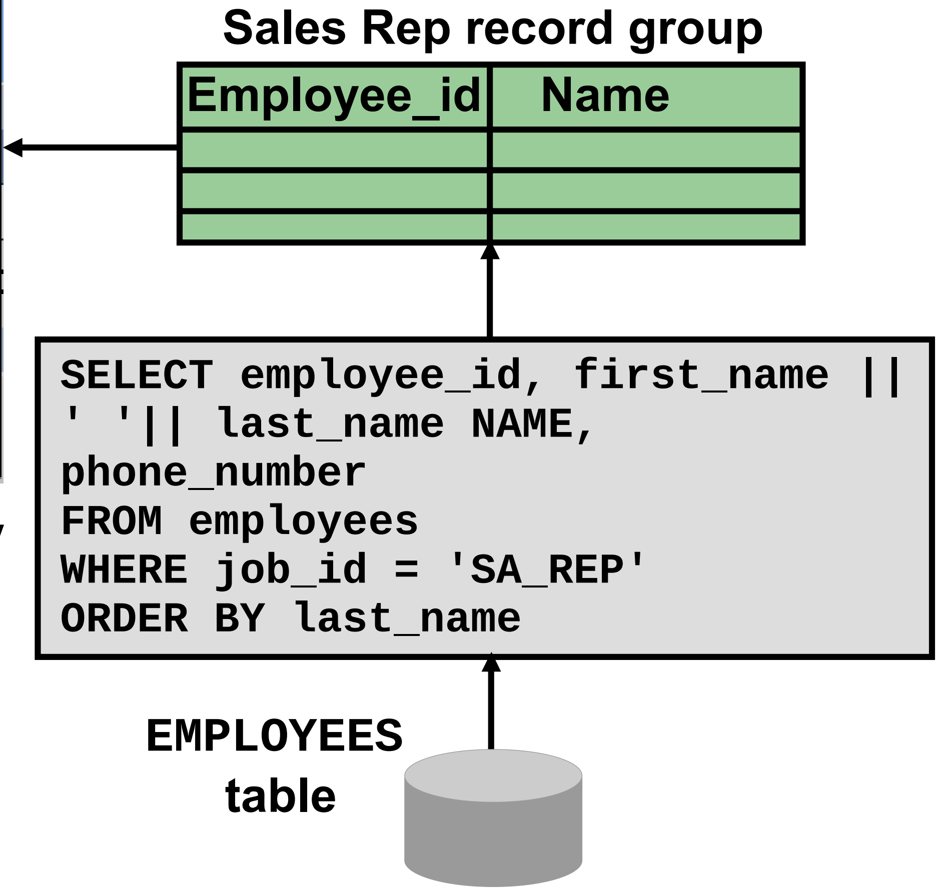
LOVs and Record Groups



Find 1%

Employ...	Name
174	Ellen Abel
166	Sundar Ande
167	Amit Banda
172	Elizabeth Bates
151	David Bernstein
169	Harrison Bloom

Sales Representatives LOV



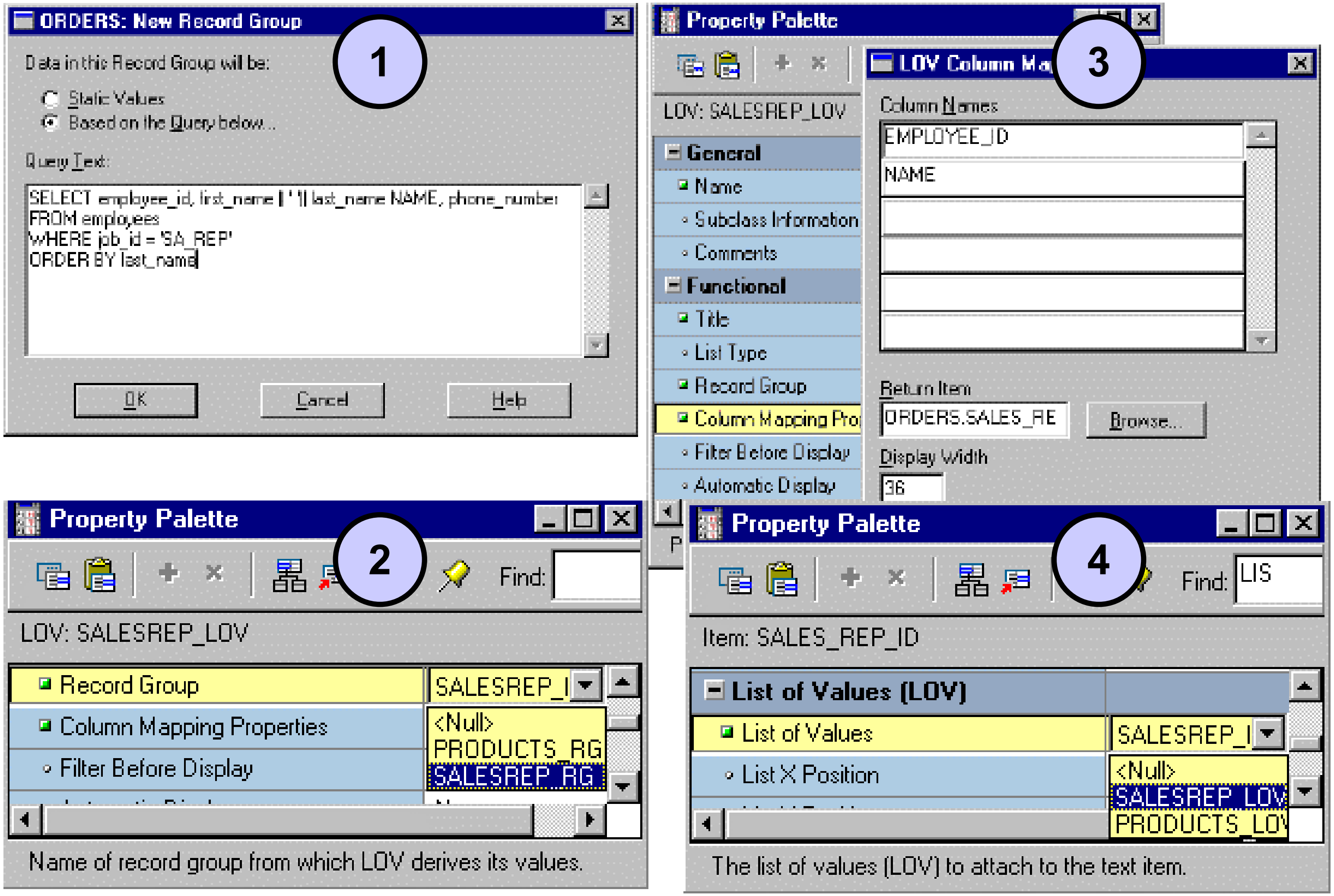
Sales Rep record group

Employee_id	Name

```
SELECT employee_id, first_name ||
' '|| last_name NAME,
phone_number
FROM employees
WHERE job_id = 'SA_REP'
ORDER BY last_name
```

EMPLOYEES
table

Creating an LOV Manually



1. ORDERS: New Record Group

Data in this Record Group will be:

○ Static Values
● Based on the Query below...

Query Text:

```
SELECT employee_id, first_name || ' '|| last_name NAME, phone_number
FROM employees
WHERE job_id = 'SA_REP'
ORDER BY last_name
```

2. Property Palette

LOV: SALESREP_LOV

Record Group	SALESREP_I
Column Mapping Properties	<Null> PRODUCTS_RG SALESREP_RG
Filter Before Display	

Name of record group from which LOV derives its values.

3. Property Palette

LOV: SALESREP_LOV

- General
  - Name
  - Subclass Information
  - Comments
- Functional
  - Title
  - List Type
  - Record Group
  - Column Mapping Pro
  - Filter Before Display
  - Automatic Display

LOV Column Ma

Column Names

EMPLOYEE_ID
NAME

Return Item: ORDERS.SALES_RE

Display Width: 36

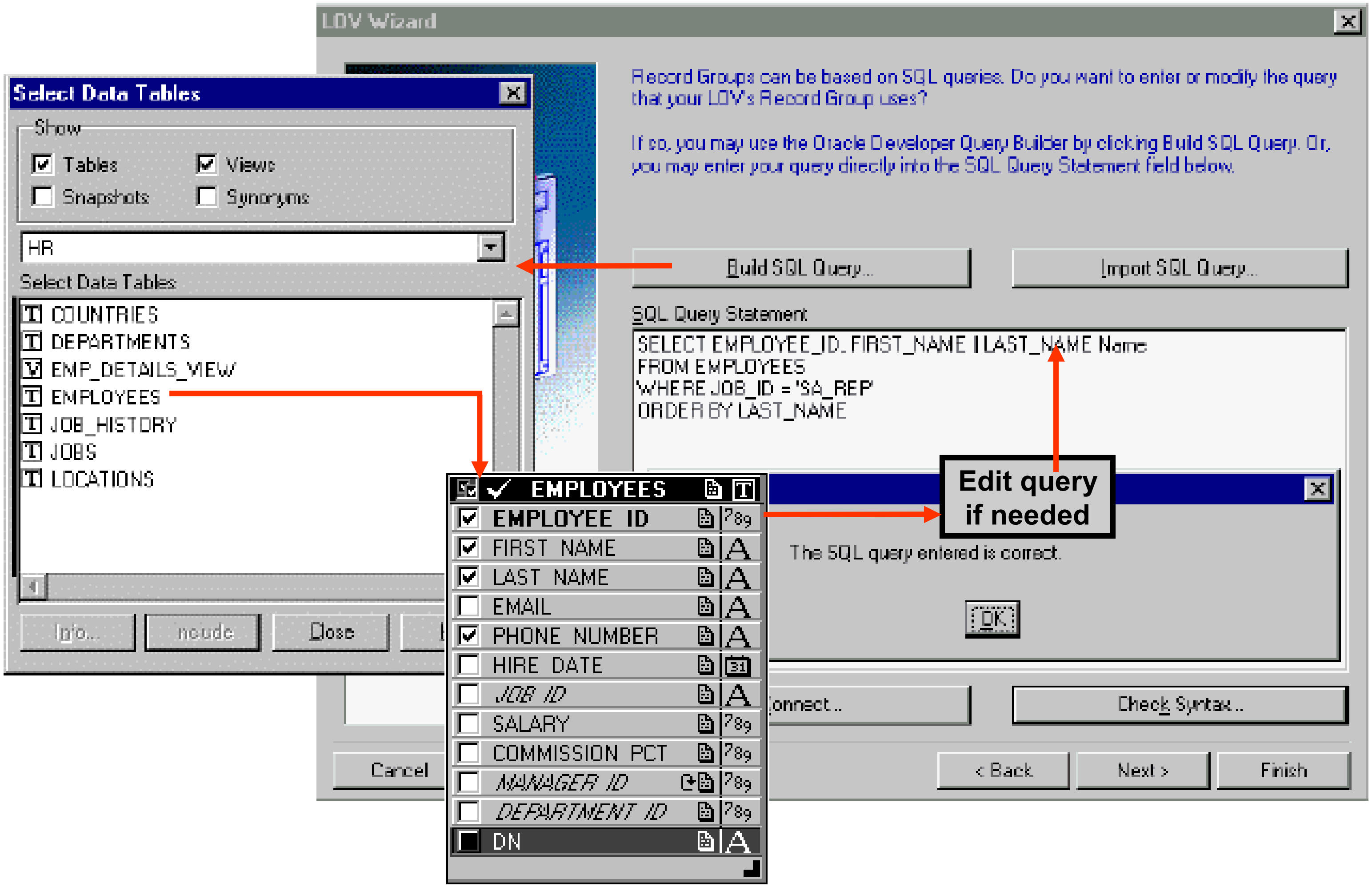
4. Property Palette

Item: SALES_REP_ID

List of Values (LOV)	
List of Values	SALESREP_I
List X Position	<Null> SALESREP_LOV PRODUCTS_LOV

The list of values (LOV) to attach to the text item.

Creating an LOV with the LOV Wizard: SQL Query Page



Record Groups can be based on SQL queries. Do you want to enter or modify the query that your LOV's Record Group uses?

If so, you may use the Oracle Developer Query Builder by clicking Build SQL Query. Or, you may enter your query directly into the SQL Query Statement field below.

SQL Query Statement

```
SELECT EMPLOYEE_ID, FIRST_NAME ||LAST_NAME Name
FROM EMPLOYEES
WHERE JOB_ID = 'SA_REP'
ORDER BY LAST_NAME
```

The SQL query entered is correct.

Show: ☑ Tables ☑ Views ☐ Snapshots ☐ Synonyms

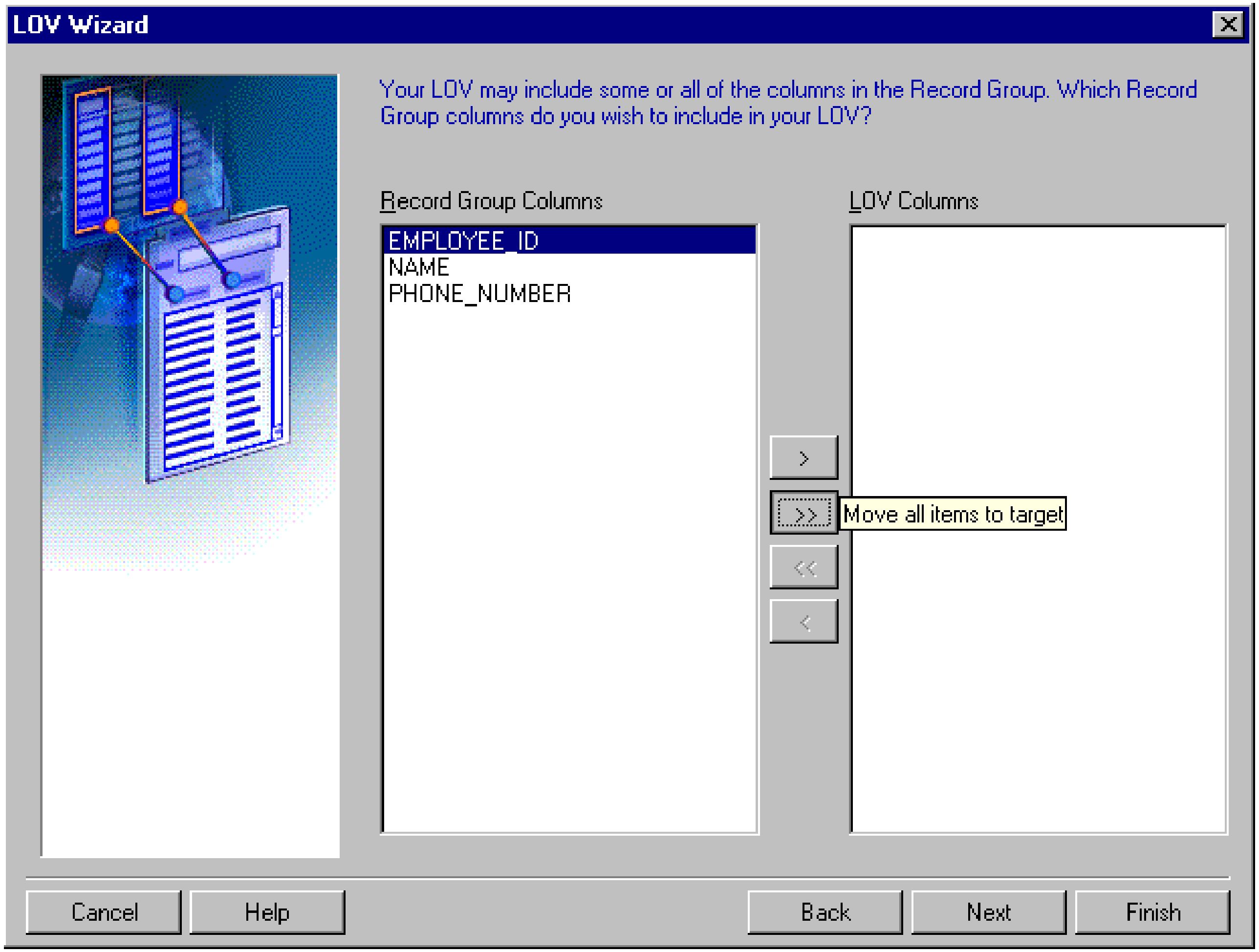
HR

Select Data Tables: COUNTRIES, DEPARTMENTS, EMP_DETAILS_VIEW, EMPLOYEES, JOB_HISTORY, JOBS, LOCATIONS

	EMPLOYEES
☑	EMPLOYEE_ID
☑	FIRST_NAME
☑	LAST_NAME
☐	EMAIL
☑	PHONE_NUMBER
☐	HIRE_DATE
☐	JOB_ID
☐	SALARY
☐	COMMISSION_PCT
☐	MANAGER_ID
☐	DEPARTMENT_ID
☐	DN

Edit query if needed

Creating an LOV with the LOV Wizard: Column Selection Page



LOV Wizard

Your LOV may include some or all of the columns in the Record Group. Which Record Group columns do you wish to include in your LOV?

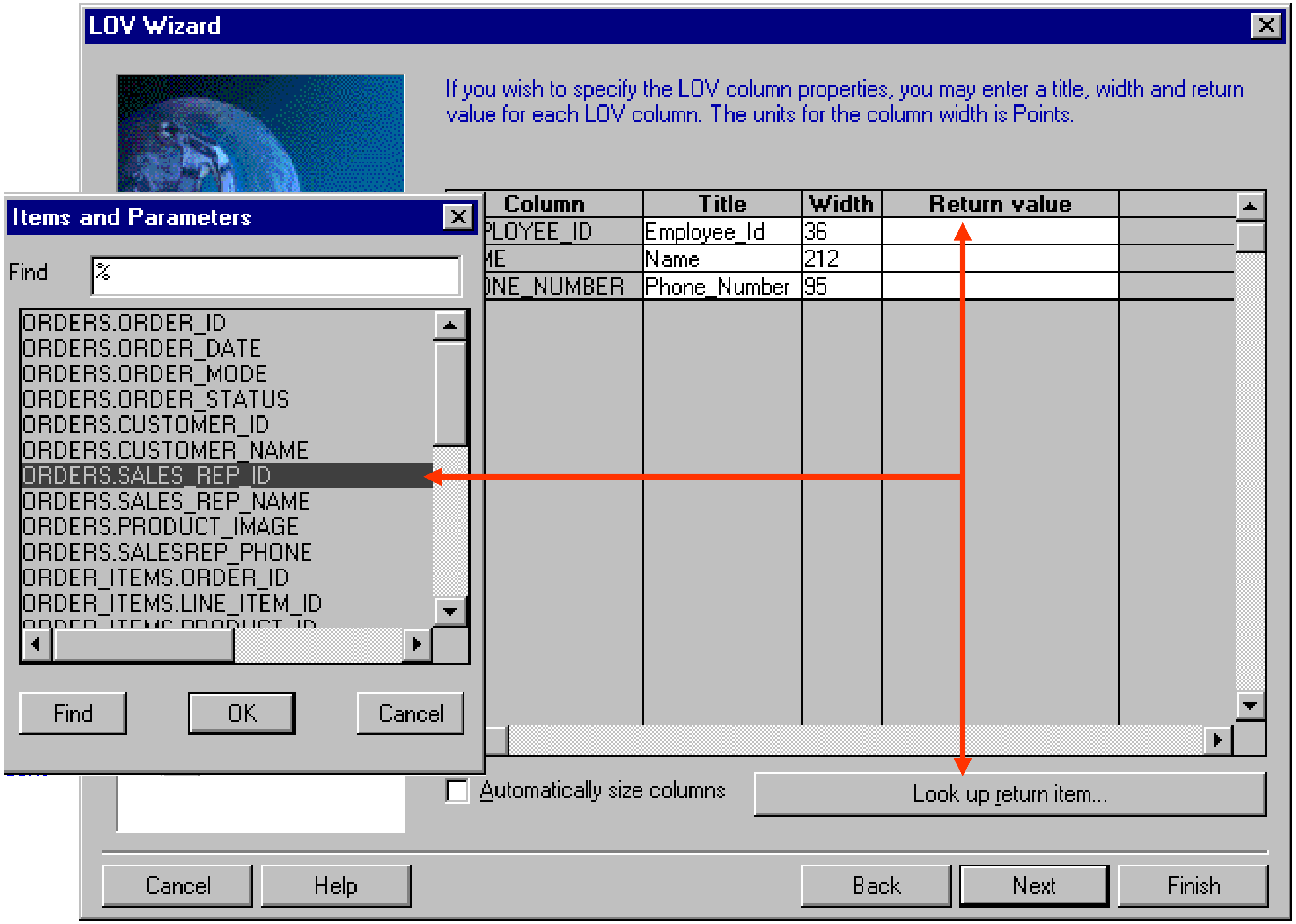
Record Group Columns

EMPLOYEE_ID
NAME
PHONE_NUMBER

LOV Columns

Move all items to target

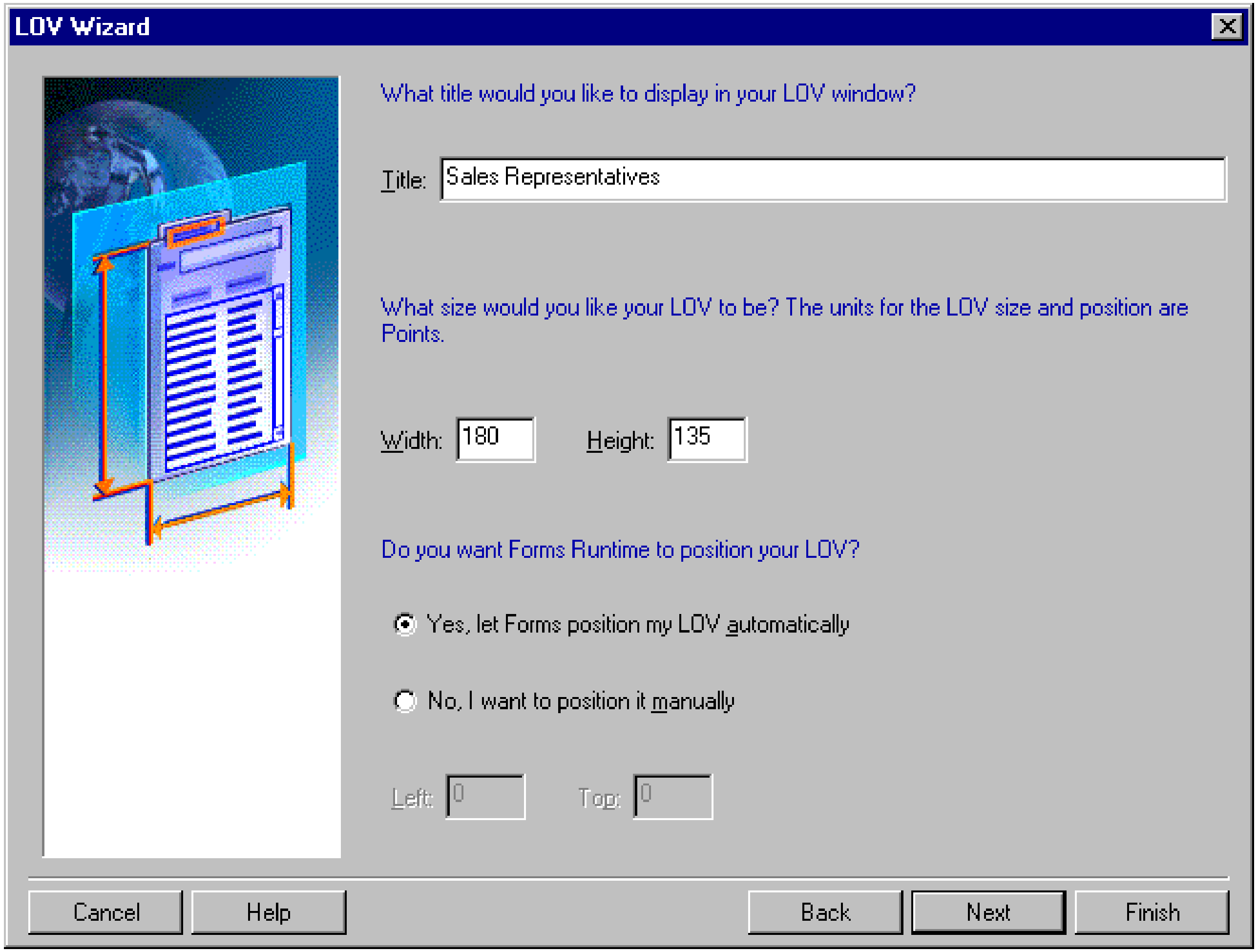
Creating an LOV with the LOV Wizard: Column Properties Page



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

Column	Title	Width	Return value
PLOYEE_ID	Employee_Id	36	
IE	Name	212	
NE_NUMBER	Phone_Number	95	

Creating an LOV with the LOV Wizard: Display Page



What title would you like to display in your LOV window?

Title: Sales Representatives

What size would you like your LOV to be? The units for the LOV size and position are Points.

Width: 180 Height: 135

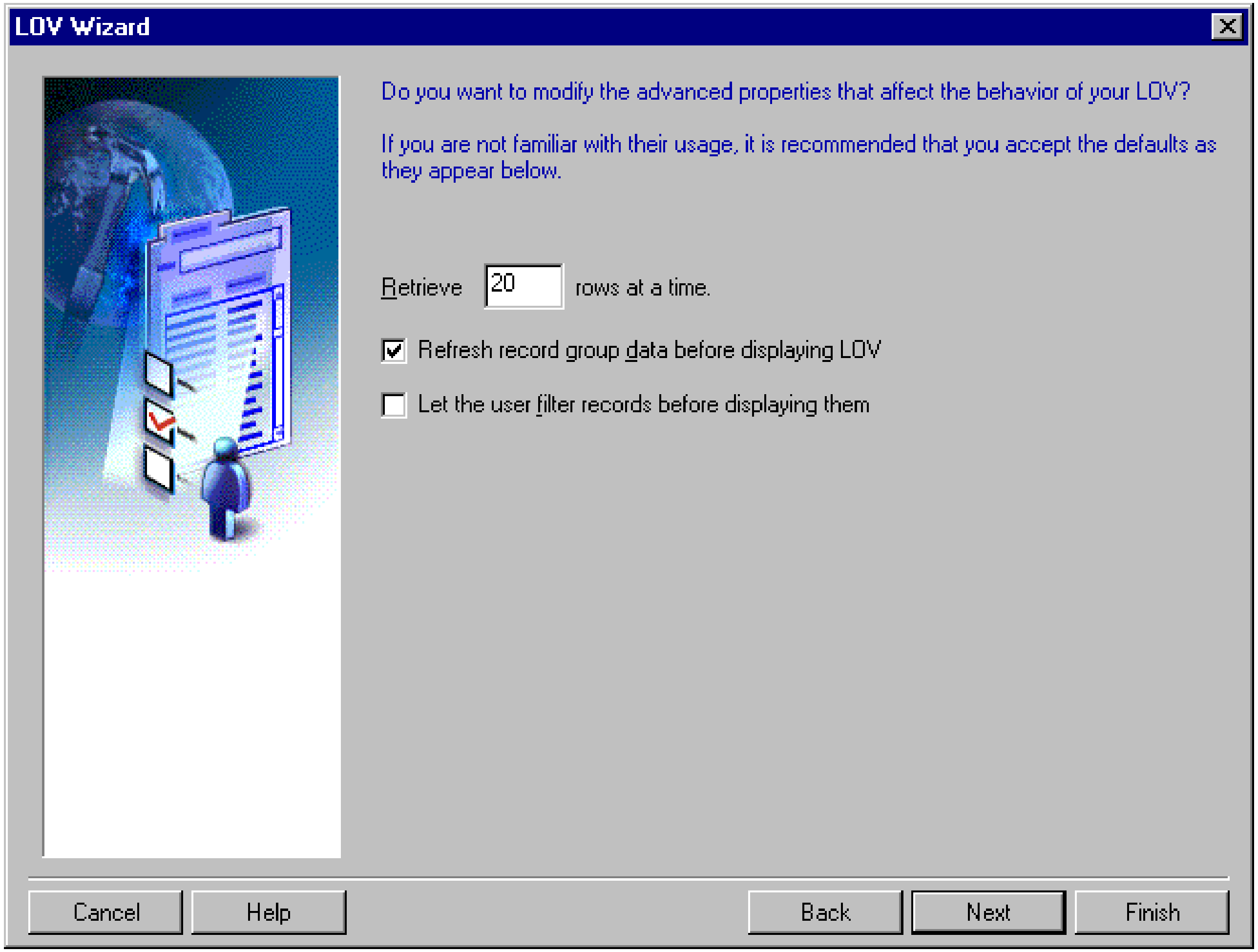
Do you want Forms Runtime to position your LOV?

● Yes, let Forms position my LOV automatically

○ No, I want to position it manually

Left: 0 Top: 0

Creating an LOV with the LOV Wizard: Advanced Properties Page



Do you want to modify the advanced properties that affect the behavior of your LOV?

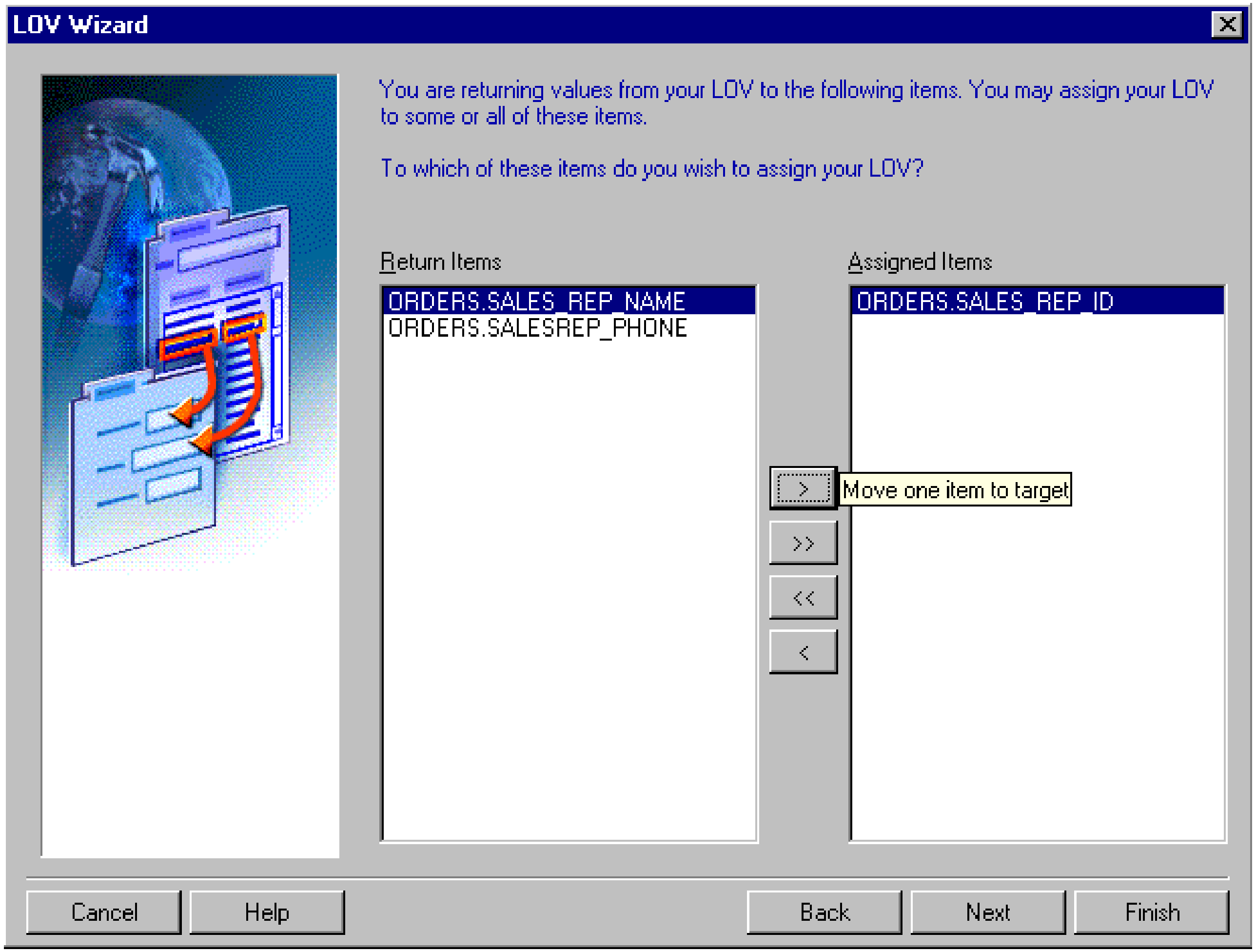
If you are not familiar with their usage, it is recommended that you accept the defaults as they appear below.

Retrieve 20 rows at a time.

☑ Refresh record group data before displaying LOV

☐ Let the user filter records before displaying them

Creating an LOV with the LOV Wizard: Assign to Item Page



LOV Wizard

You are returning values from your LOV to the following items. You may assign your LOV to some or all of these items.

To which of these items do you wish to assign your LOV?

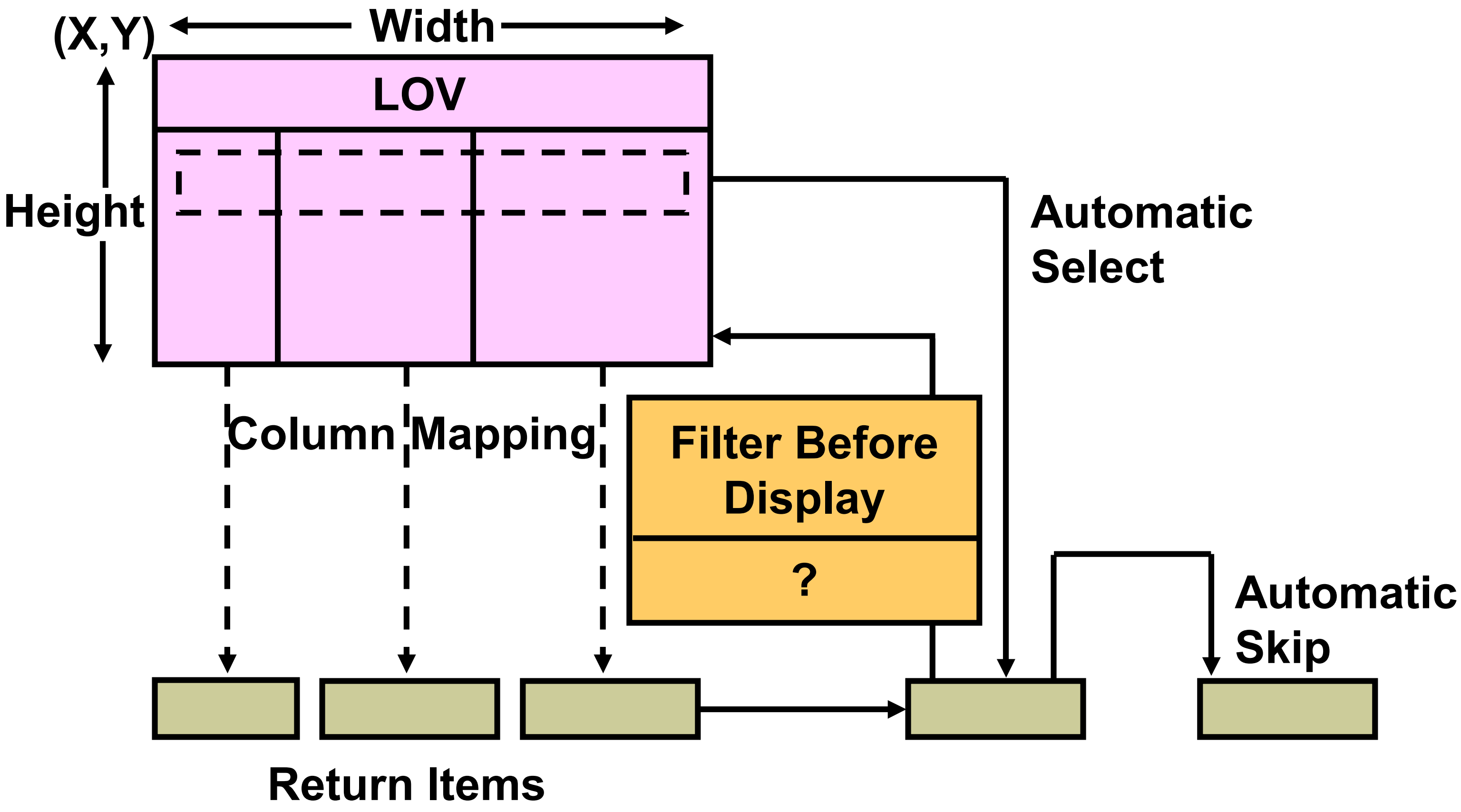
Return Items

- ORDERS.SALES_REP_NAME
- ORDERS.SALESREP_PHONE

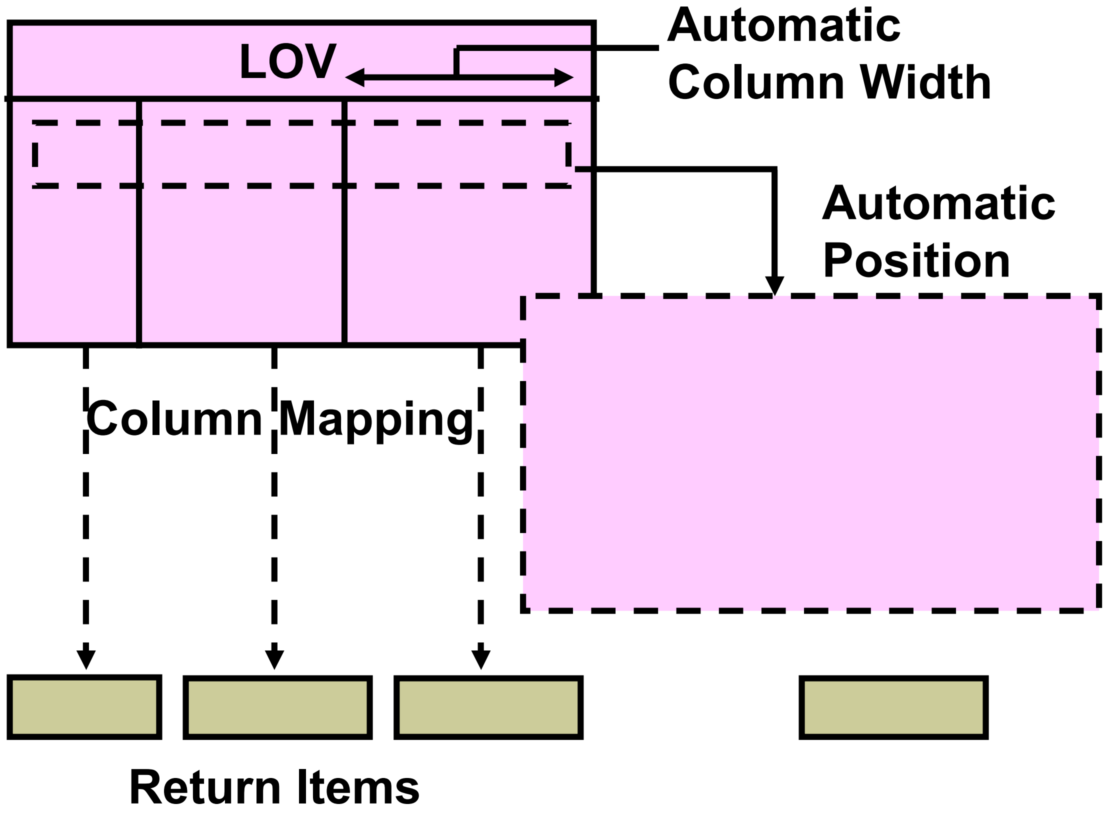
Assigned Items

- ORDERS.SALES_REP_ID

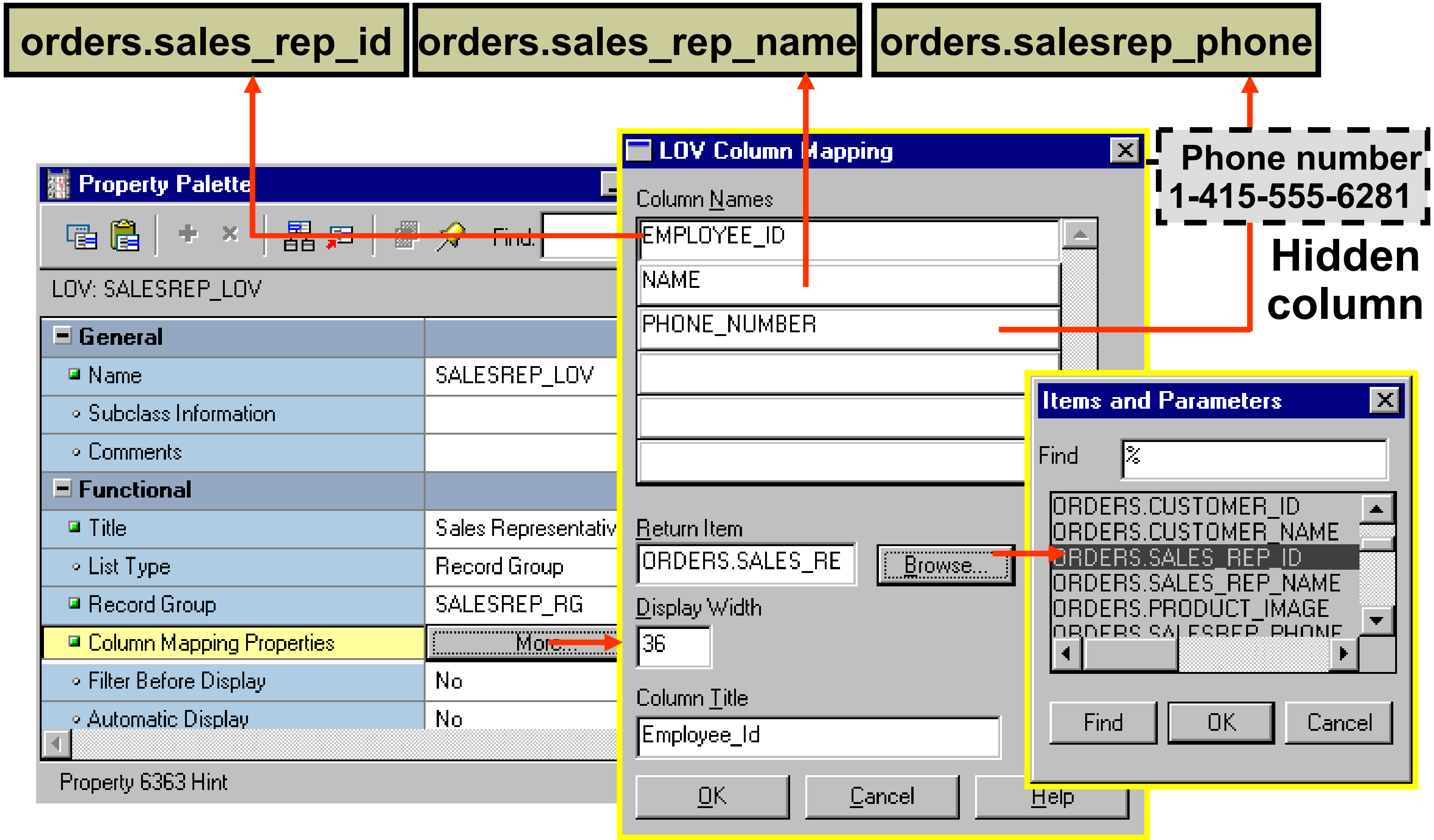
LOV Properties



Setting LOV Properties



LOVs: Column Mapping



LOV: SALESREP_LOV

General	
Name	SALESREP_LOV
Subclass Information	
Comments	
Functional	
Title	Sales Representativ
List Type	Record Group
Record Group	SALESREP_RG
Column Mapping Properties	More...
Filter Before Display	No
Automatic Display	No

Property 6363 Hint

Column Names: EMPLOYEE_ID, NAME, PHONE_NUMBER

Return Item: ORDERS.SALES_RE

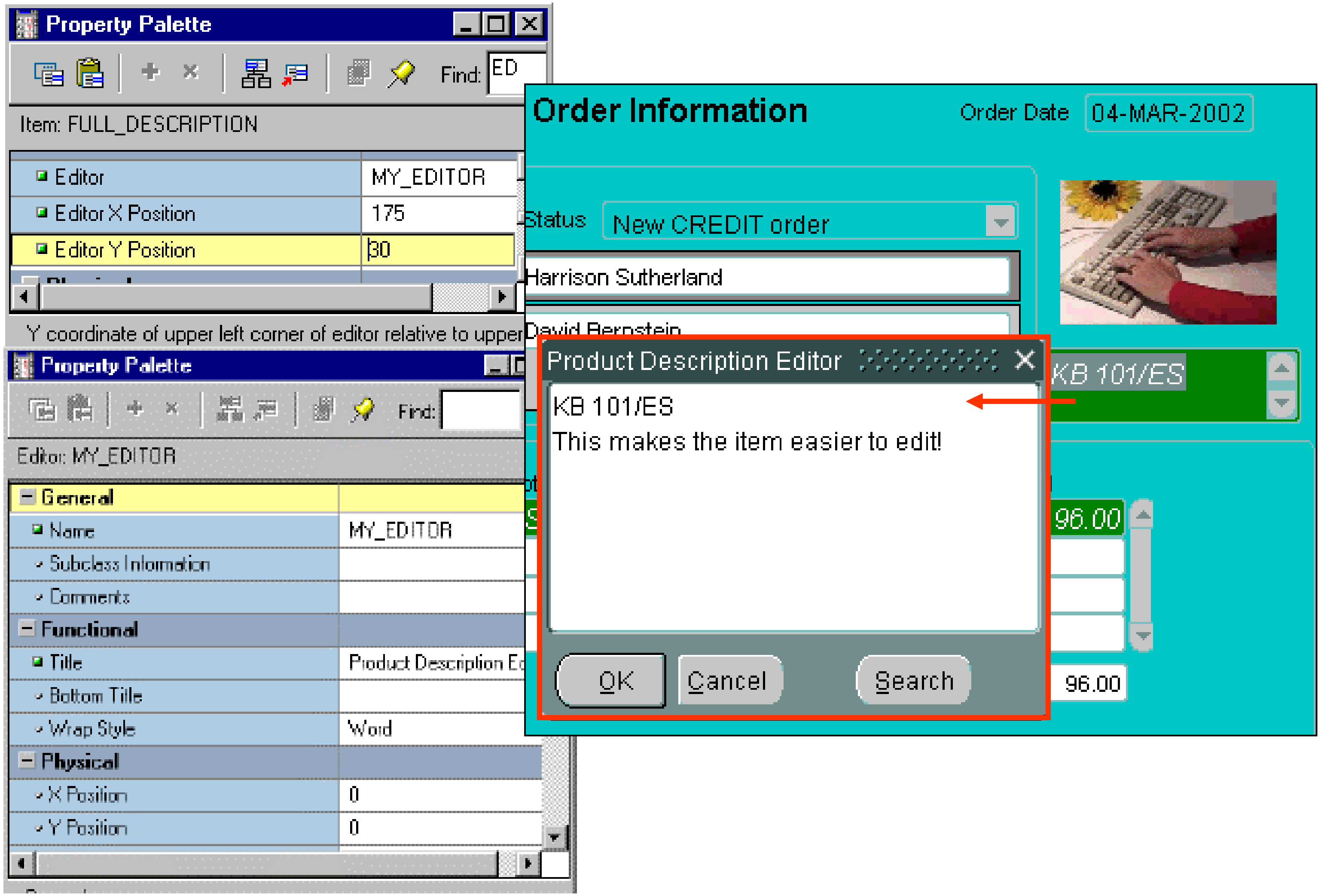
Display Width: 36

Column Title: Employee_Id

Find: %

ORDERS.CUSTOMER_ID
ORDERS.CUSTOMER_NAME
ORDERS.SALES_REP_ID
ORDERS.SALES_REP_NAME
ORDERS.PRODUCT_IMAGE
ORDERS.SALESREP_PHONE

Defining an Editor



Property Palette

Find: ED

Item: FULL_DESCRIPTION

Editor	MY_EDITOR
Editor X Position	175
Editor Y Position	30

Y coordinate of upper left corner of editor relative to upper

Property Palette

Editor: MY_EDITOR

General	
Name	MY_EDITOR
Subclass Information	
Comments	
Functional	
Title	Product Description E
Bottom Title	
Wrap Style	Word
Physical	
X Position	0
Y Position	0

Order Information

Order Date 04-MAR-2002

Status New CREDIT order

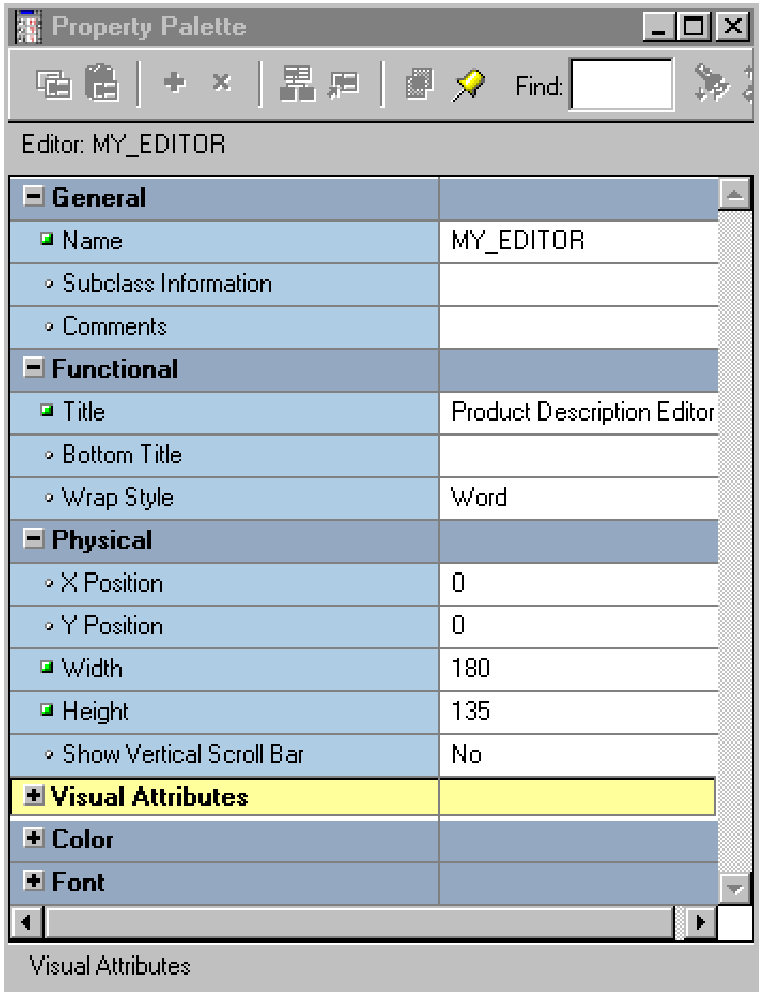
Harrison Sutherland

David Bernstein

Product Description Editor

KB 101/ES
This makes the item easier to edit!

Setting Editor Properties

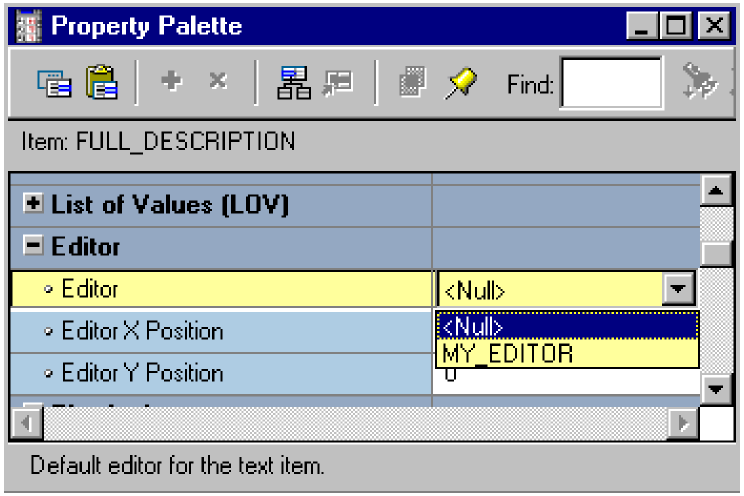


Editor: MY_EDITOR

⊟ General	
Name	MY_EDITOR
Subclass Information	
Comments	
⊟ Functional	
Title	Product Description Editor
Bottom Title	
Wrap Style	Word
⊟ Physical	
X Position	0
Y Position	0
Width	180
Height	135
Show Vertical Scroll Bar	No
⊞ Visual Attributes	
⊞ Color	
⊞ Font	

Visual Attributes

Associating an Editor with a Text Item



Item: FULL_DESCRIPTION

- List of Values (LOV)
- Editor
  - Editor: <Null>
    - <Null>
    - MY_EDITOR
  - Editor X Position
  - Editor Y Position: 0

Default editor for the text item.

- Associate one of two types of editors with a text item.
- Set text item's Editor property to one of the following:
  - Null (default Forms Builder editor)
  - Editor name (customized editor)

Summary

In this lesson, you should have learned that:

- An LOV is a scrollable pop-up window that enables a user to pick the value of an item from a multicolumn dynamic list
- The easiest way to design, create, and associate LOVs with text items is to use the LOV Wizard
- An Editor is a separate window that enables the user to view multiple lines of a text item simultaneously, search and replace text in it, and modify the text
- You create editors in the Object Navigator and associate them with text items in the item's Property Palette

Practice 8 Overview

This practice covers the following topics:

- Creating an LOV and attaching the LOV to a text item
- Creating an Editor and attaching it to a text item

9

Creating Additional Input Items

Objectives

After completing this lesson, you should be able to do the following:

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

Input Items Overview

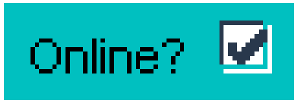
What are input items?

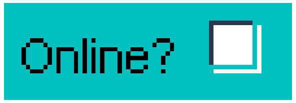
- Item types that accept user input include:
  - Check boxes
  - List items
  - Radio groups
- Input items enable insert, update, delete, and query.

Check Boxes Overview

What Are Check Boxes?

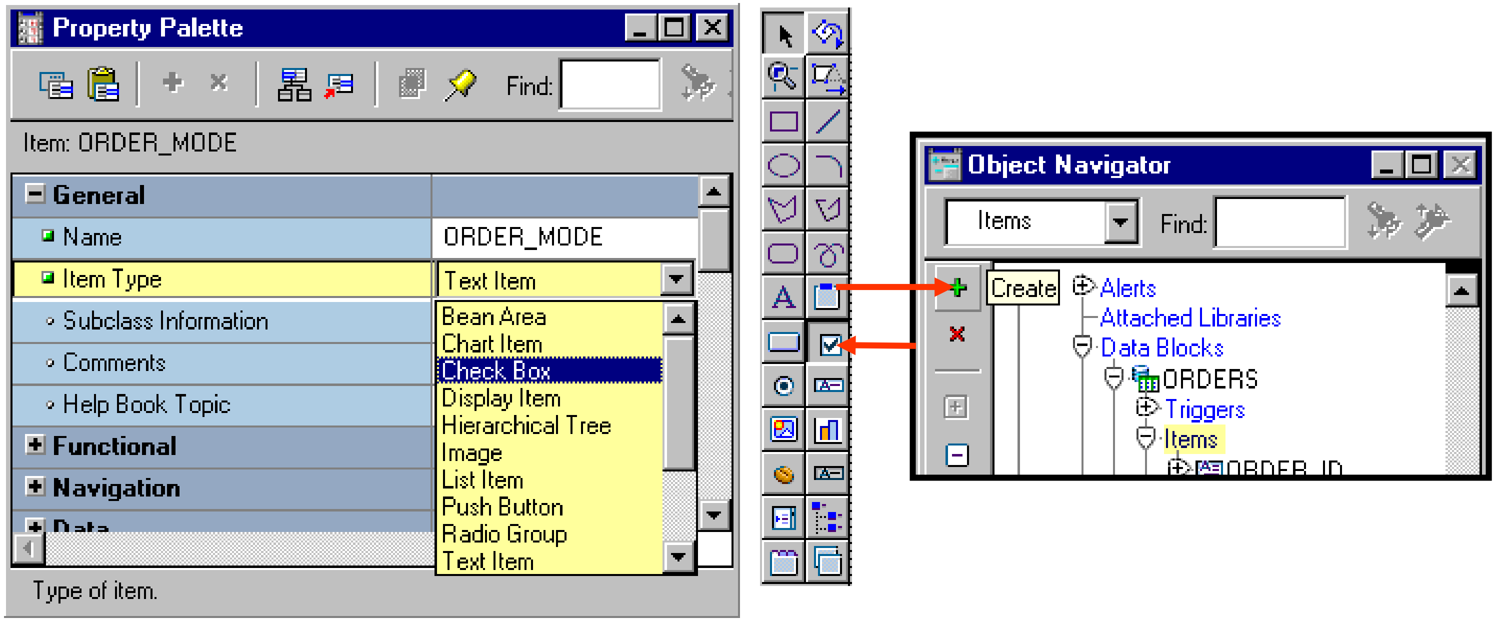
- Two-state interface object:
  - Checked
  - Unchecked
- Not limited to two values


Online? ☑


Online? □

Creating a Check Box

- Convert an existing item.
- Use the Check Box tool in the Layout Editor.
- Use the Create icon in the Object Navigator.



Property Palette

Find:

Item: ORDER_MODE

Name: ORDER_MODE

Item Type: Text Item

Type of item.

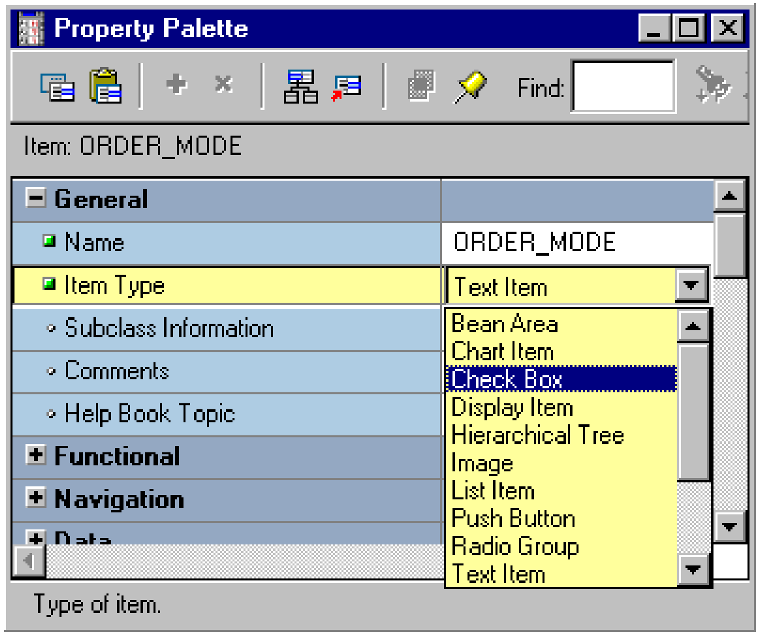
Object Navigator

Items

Find:

Converting an Existing Item into a Check Box

Convert text item to check box



Property Palette

Find:

Item: ORDER_MODE

General

Name: ORDER_MODE

Item Type: Text Item

Subclass Information

Comments

Help Book Topic

Functional

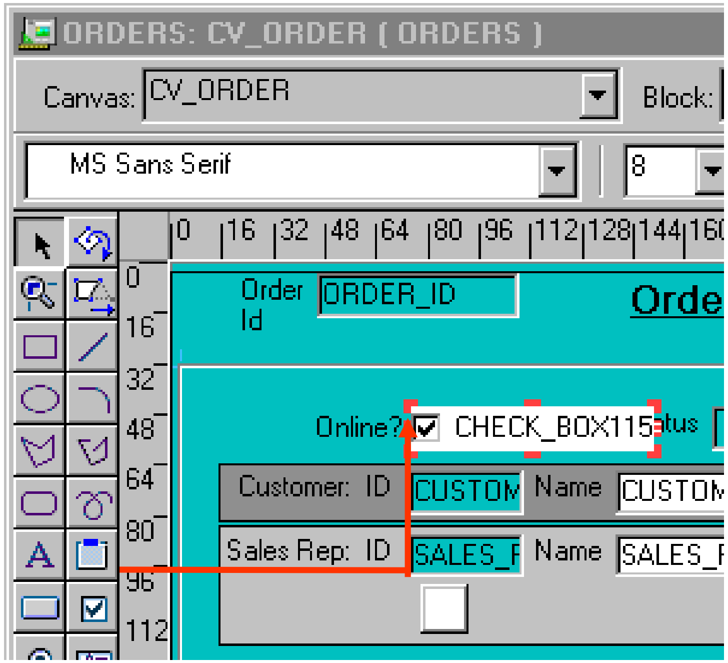
Navigation

Data

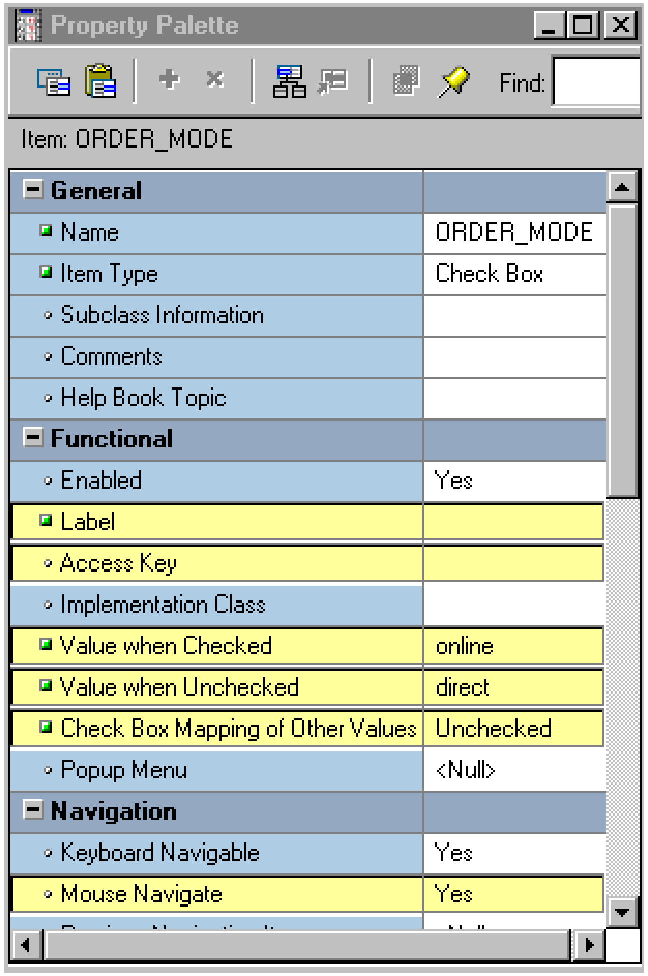
Type of item.

Creating a Check Box in the Layout Editor

Use check box tool in Layout Editor



Setting Check Box Properties

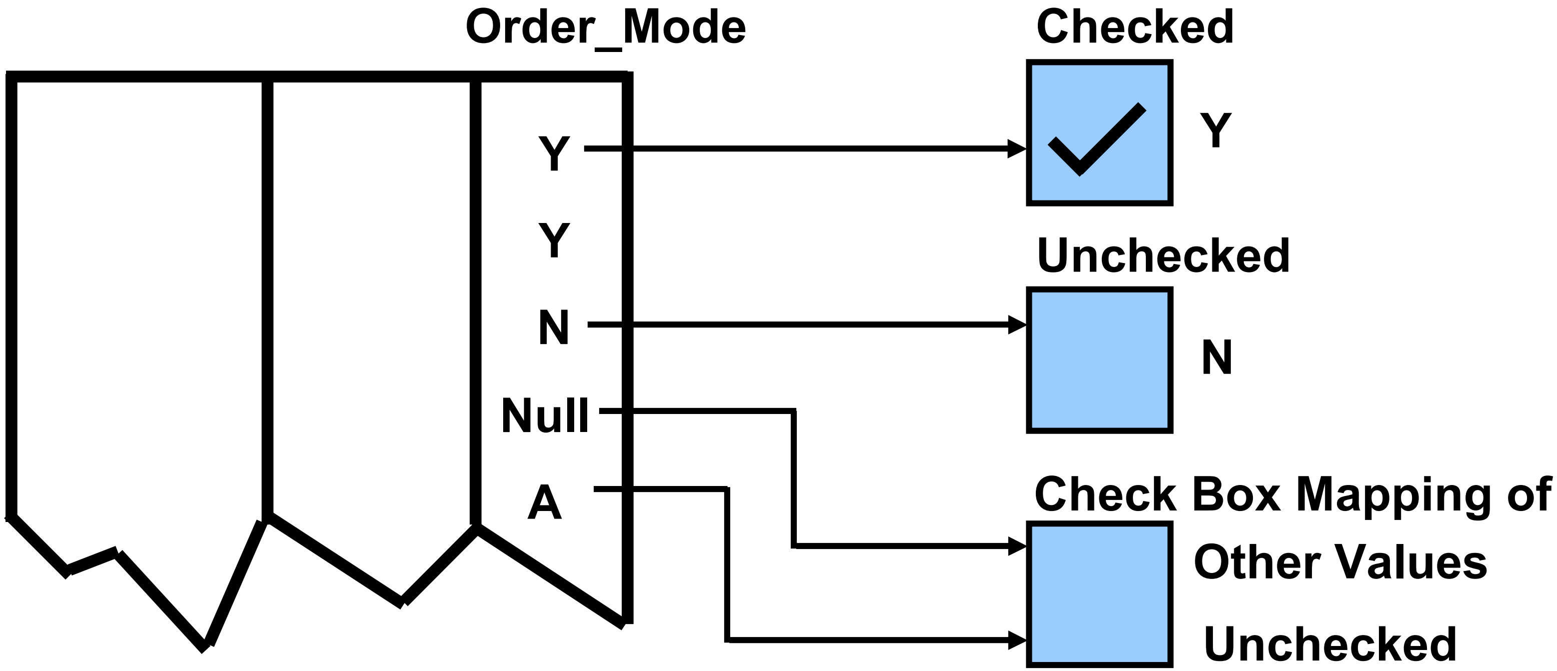


Item: ORDER_MODE

General	
Name	ORDER_MODE
Item Type	Check Box
Subclass Information	
Comments	
Help Book Topic	
Functional	
Enabled	Yes
Label	
Access Key	
Implementation Class	
Value when Checked	online
Value when Unchecked	direct
Check Box Mapping of Other Values	Unchecked
Popup Menu	<Null>
Navigation	
Keyboard Navigable	Yes
Mouse Navigate	Yes

- Data Type
- Label
- Access Key
- Value When Checked
- Value When Unchecked
- Check Box Mapping of Other Values
- Mouse Navigate

Check Box Mapping of Other Values

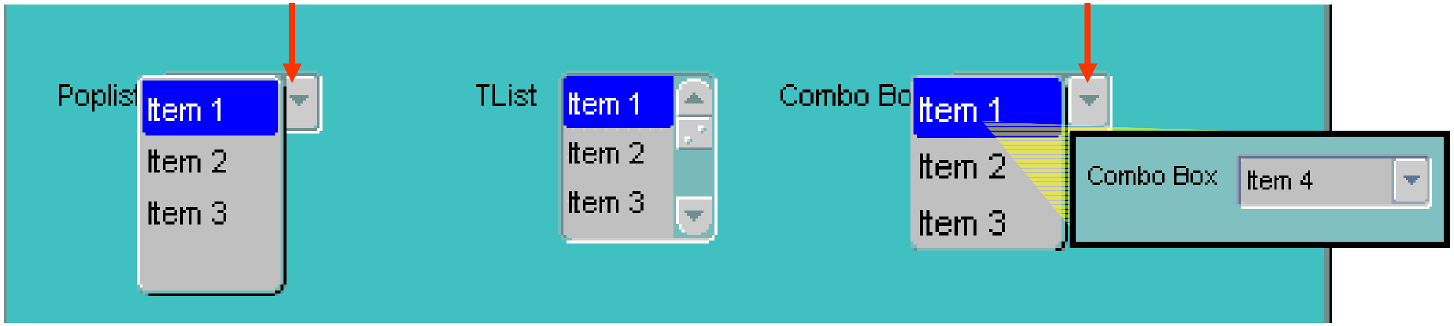


List Items Overview

What Are List Items?

- Set of mutually exclusive choices, each representing a different value
- Three list styles available:

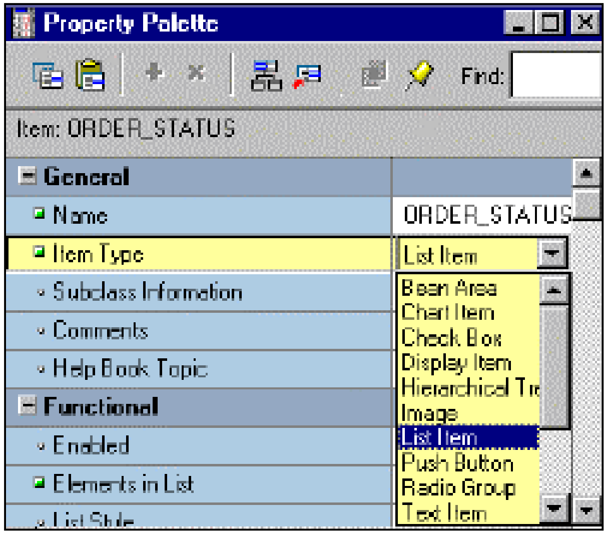
Poplist Tlist Combo Box



- Space-saving alternative to a radio group
- Smaller-scale alternative to an LOV

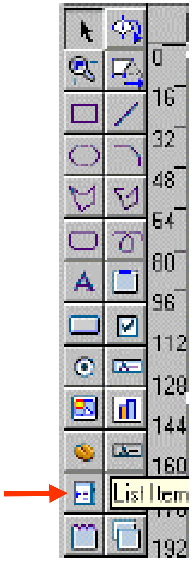
Creating a List Item

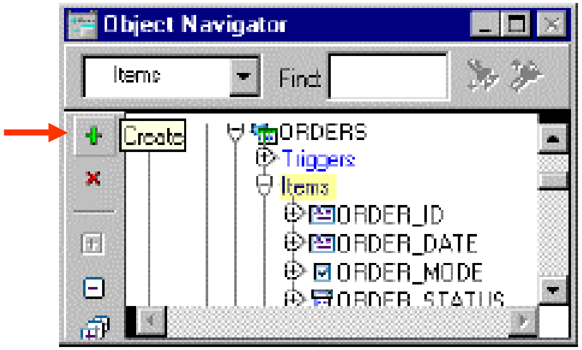
- Convert an existing item.
- Use the List Item tool in the Layout Editor.
- Use the Create icon in the Object Navigator.



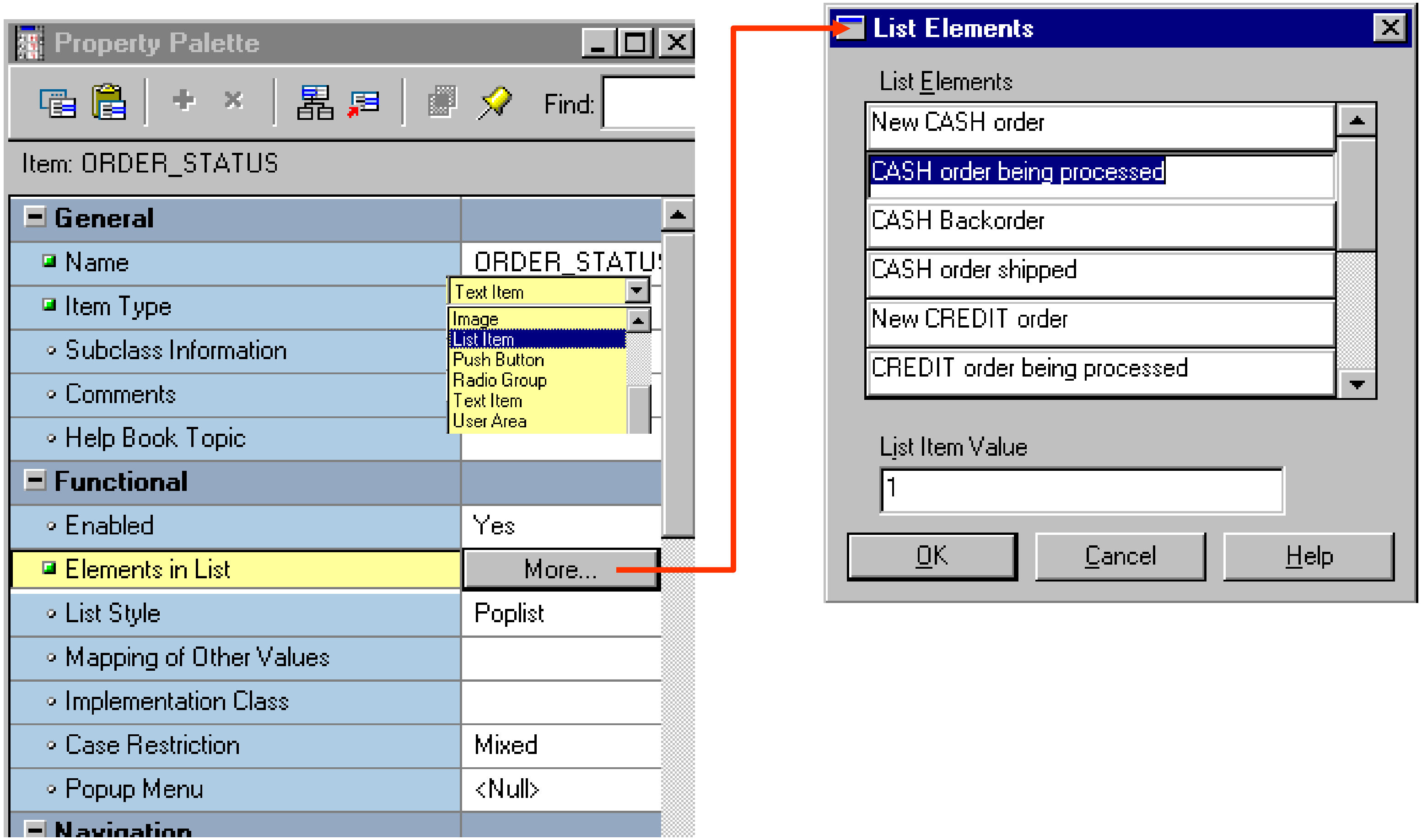
Item: ORDER_STATUS

- General
  - Name: ORDER_STATUS
  - Item Type: List Item
  - Subclass Information
  - Comments
  - Help Book Topic
- Functional
  - Enabled
  - Elements in List
  - List Style





Converting an Existing Item into a List Item



Item: ORDER_STATUS

General

Name	ORDER_STATU
Item Type	Text Item
Subclass Information	
Comments	
Help Book Topic	

Functional

Enabled	Yes
Elements in List	More...
List Style	Poplist
Mapping of Other Values	
Implementation Class	
Case Restriction	Mixed
Popup Menu	<Null>

Navigation

List Elements

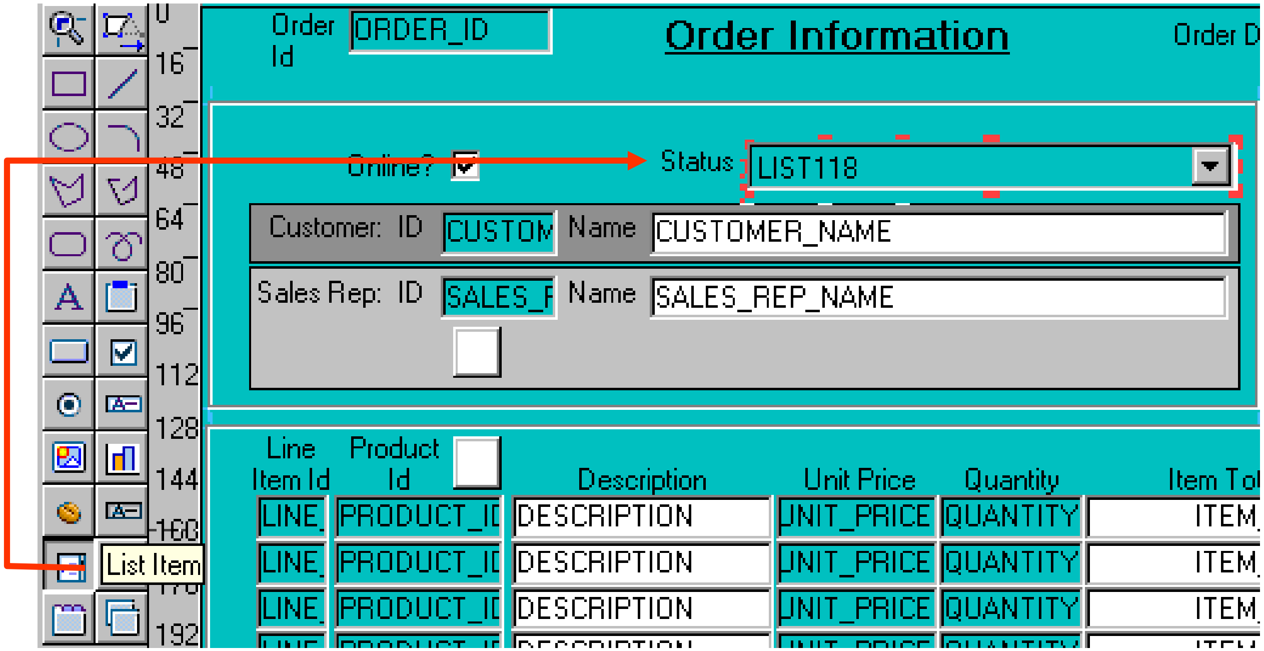
List Elements

- New CASH order
- CASH order being processed
- CASH Backorder
- CASH order shipped
- New CREDIT order
- CREDIT order being processed

List Item Value: 1

Creating a List Item in the Layout Editor

Use list item tool
in Layout Editor



Order Id ORDER_ID

Order Information

Order D

Online? ☑ Status LIST118

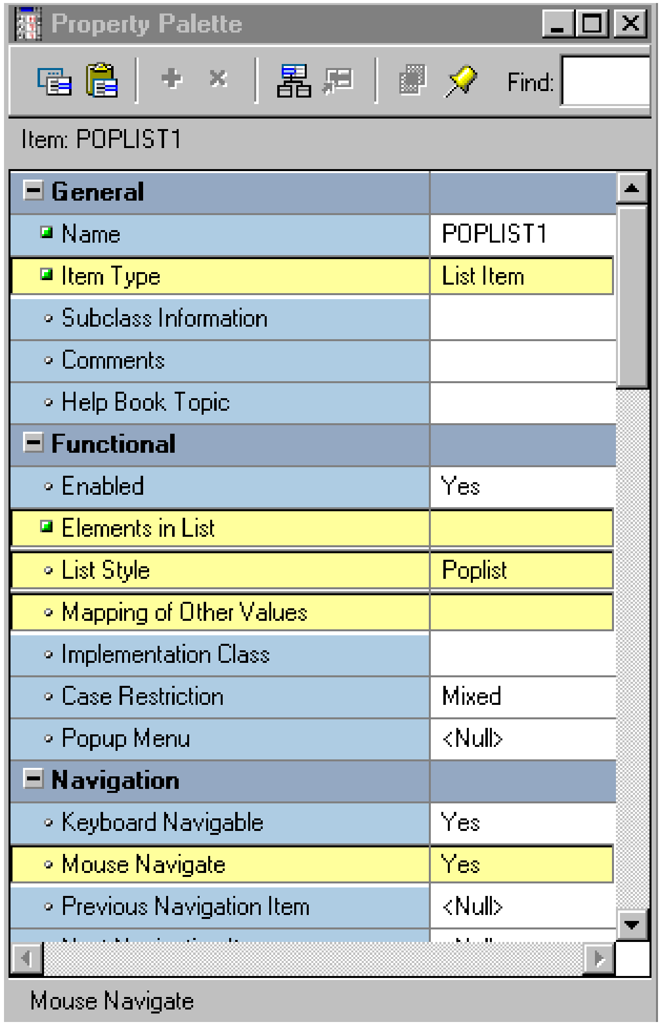
Customer: ID CUSTOM Name CUSTOMER_NAME

Sales Rep: ID SALES_F Name SALES_REP_NAME

Line Item Id	Product Id	Description	Unit Price	Quantity	Item Tot
LINE	PRODUCT_ID	DESCRIPTION	UNIT_PRICE	QUANTITY	ITEM
LINE	PRODUCT_ID	DESCRIPTION	UNIT_PRICE	QUANTITY	ITEM
LINE	PRODUCT_ID	DESCRIPTION	UNIT_PRICE	QUANTITY	ITEM

Setting List Item Properties

- Elements in List:
  - List elements
  - List item value
- List Style
- Mapping of Other Values
- Mouse Navigate

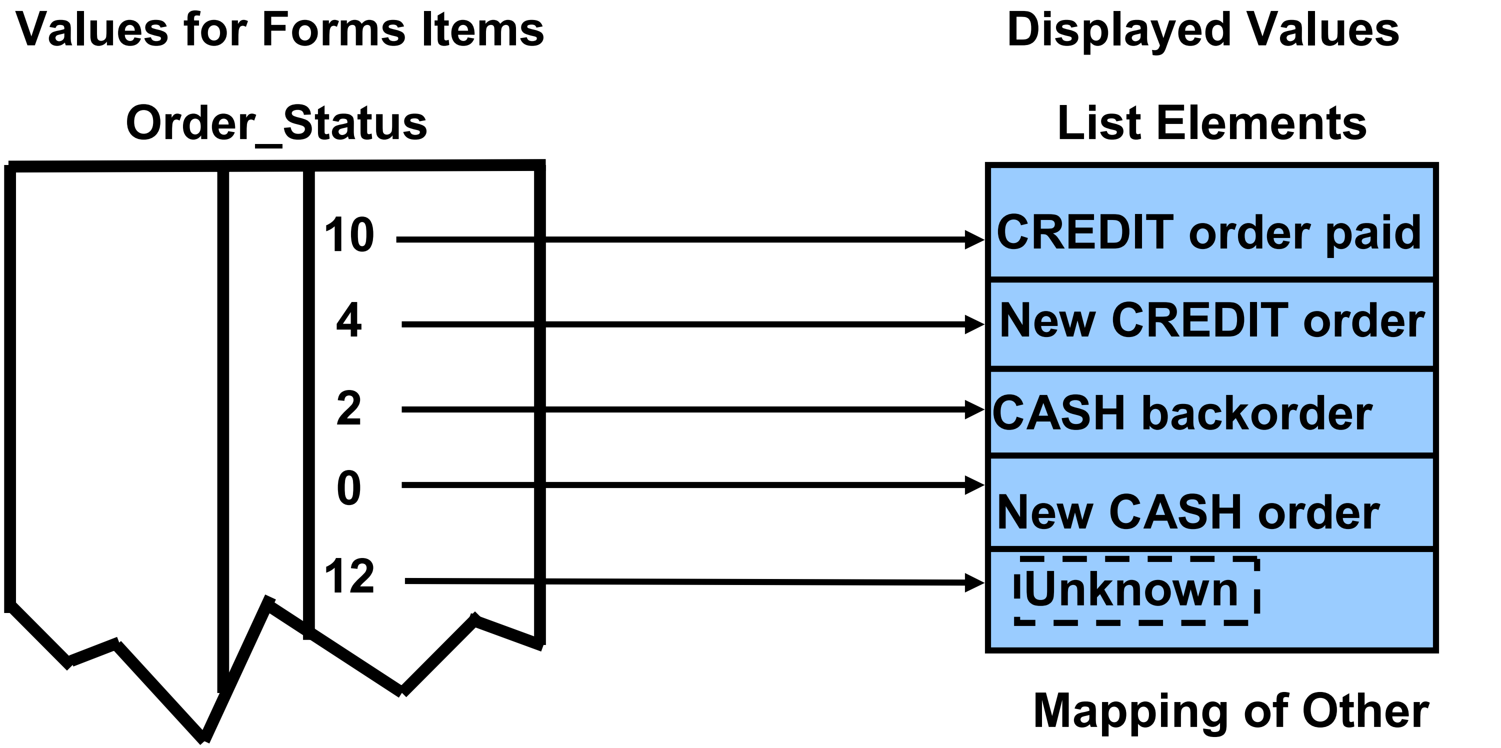


Item: POPLIST1

General	
Name	POPLIST1
Item Type	List Item
Subclass Information	
Comments	
Help Book Topic	
Functional	
Enabled	Yes
Elements in List	
List Style	Poplist
Mapping of Other Values	
Implementation Class	
Case Restriction	Mixed
Popup Menu	<Null>
Navigation	
Keyboard Navigable	Yes
Mouse Navigate	Yes
Previous Navigation Item	<Null>

Mouse Navigate

List Item Mapping of Other Values

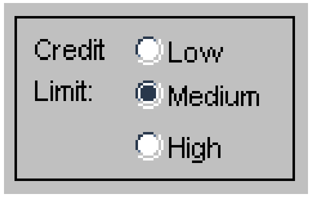


Mapping of Other
Values = 11 (Unknown)

Radio Groups Overview

What are radio groups?

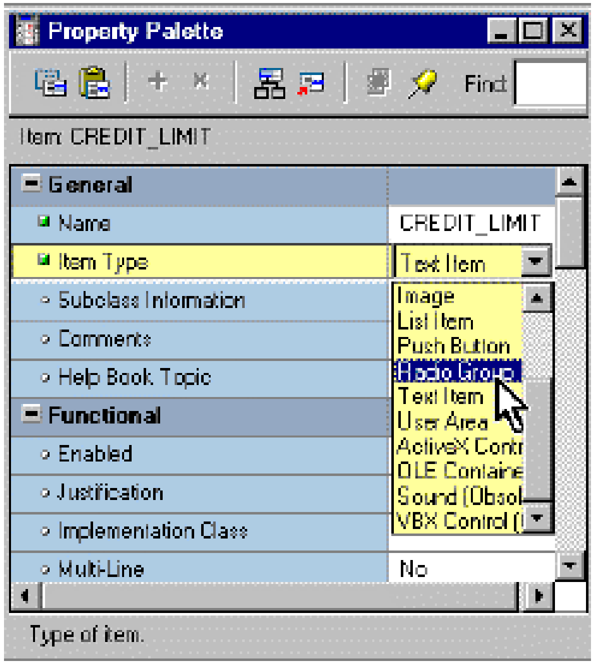
- Set of mutually exclusive radio buttons, each representing a value
- Use:
  - To display two or more static choices
  - As an alternative to a list item
  - As an alternative to a check box



Credit Limit: ○ Low ● Medium ○ High

Creating a Radio Group

- Convert an existing item.
- Create a new radio button in the Layout Editor.
- Use the Create icon in the Object Navigator.



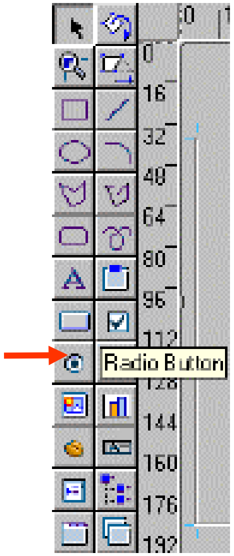
Item: CREDIT_LIMIT

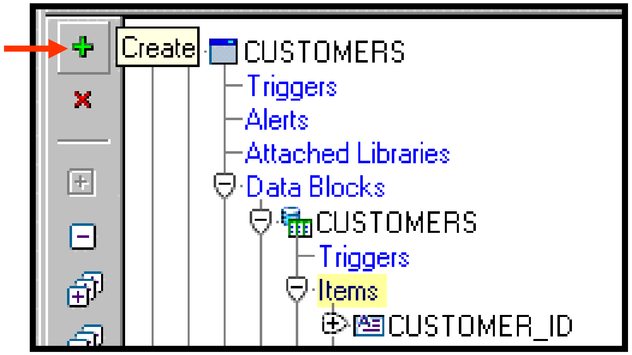
Name: CREDIT_LIMIT

Item Type: Text Item

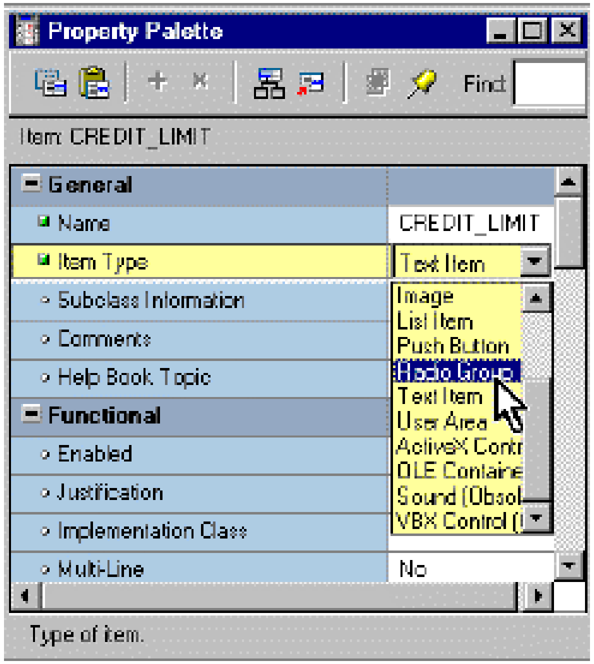
Multi-Line: No

Type of item.





Converting Existing Item to Radio Group

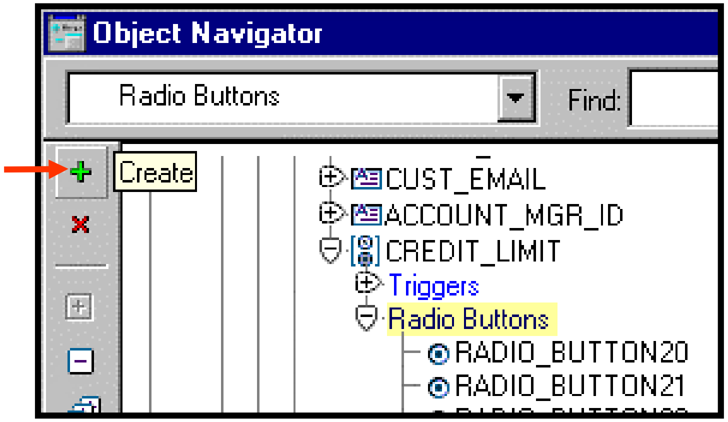


Item: CREDIT_LIMIT

General	
Name	CREDIT_LIMIT
Item Type	Text Item
Subclass Information	
Comments	
Help Book Topic	
Functional	
Enabled	
Justification	
Implementation Class	
Multi-Line	No

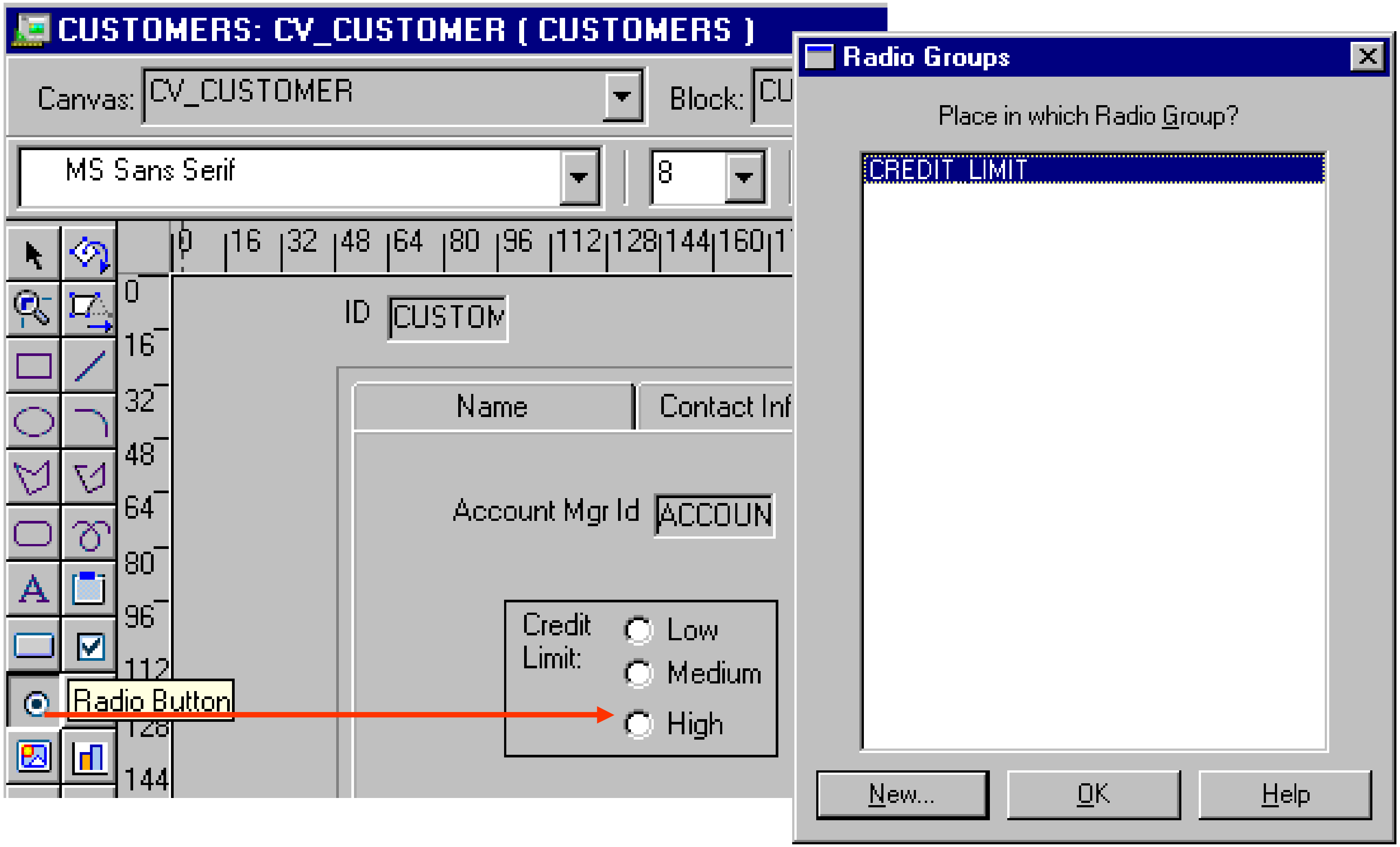
Type of item.

Change Item Type and set other properties



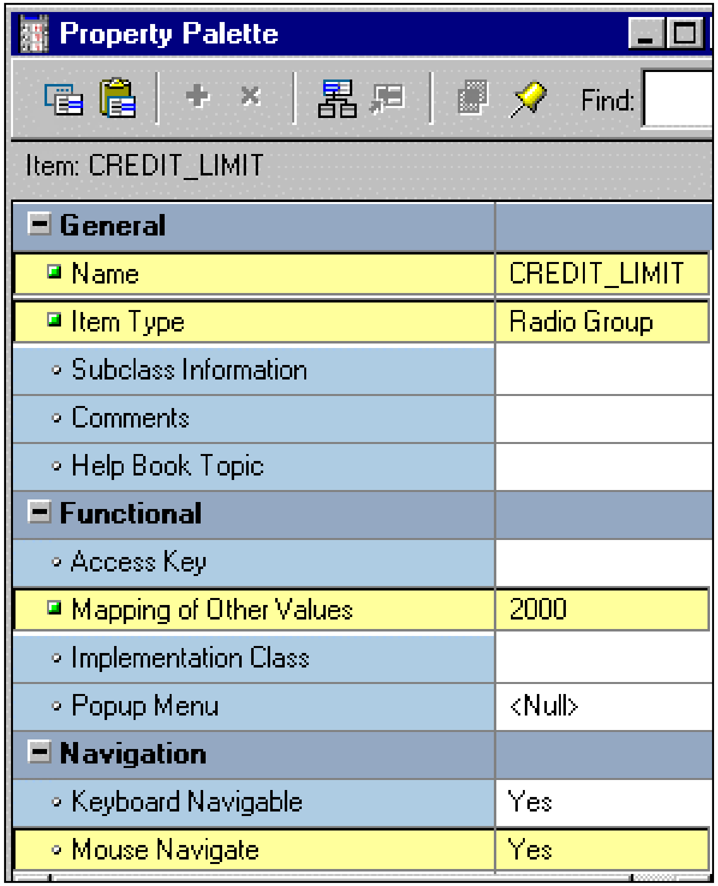
Create radio buttons for the radio group

Creating Radio Group in Layout Editor



Setting Radio Properties

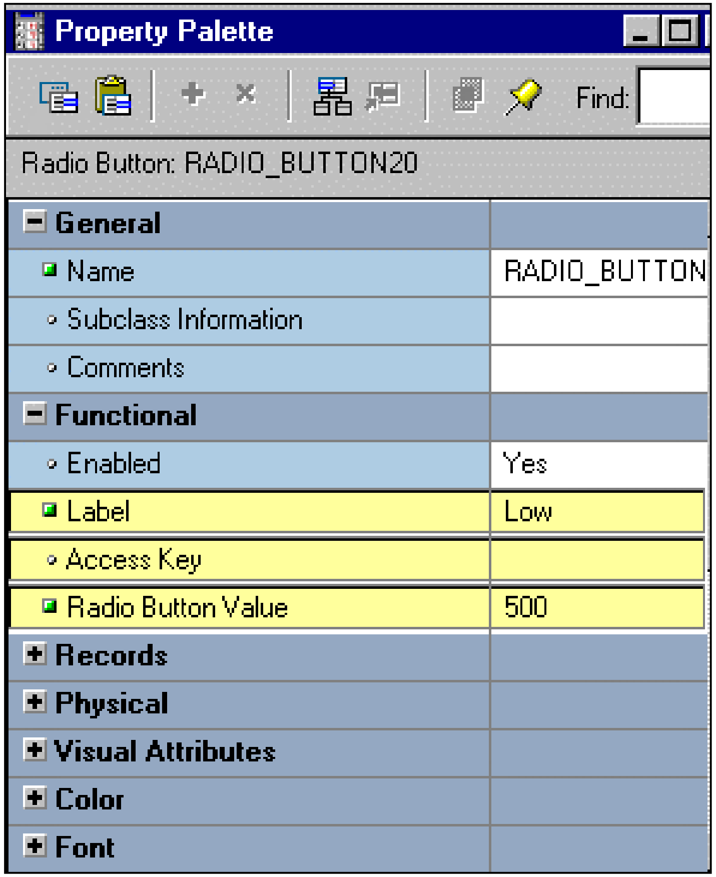
Radio group:



Item: CREDIT_LIMIT

⊟ General	
Name	CREDIT_LIMIT
Item Type	Radio Group
Subclass Information	
Comments	
Help Book Topic	
⊟ Functional	
Access Key	
Mapping of Other Values	2000
Implementation Class	
Popup Menu	<Null>
⊟ Navigation	
Keyboard Navigable	Yes
Mouse Navigate	Yes

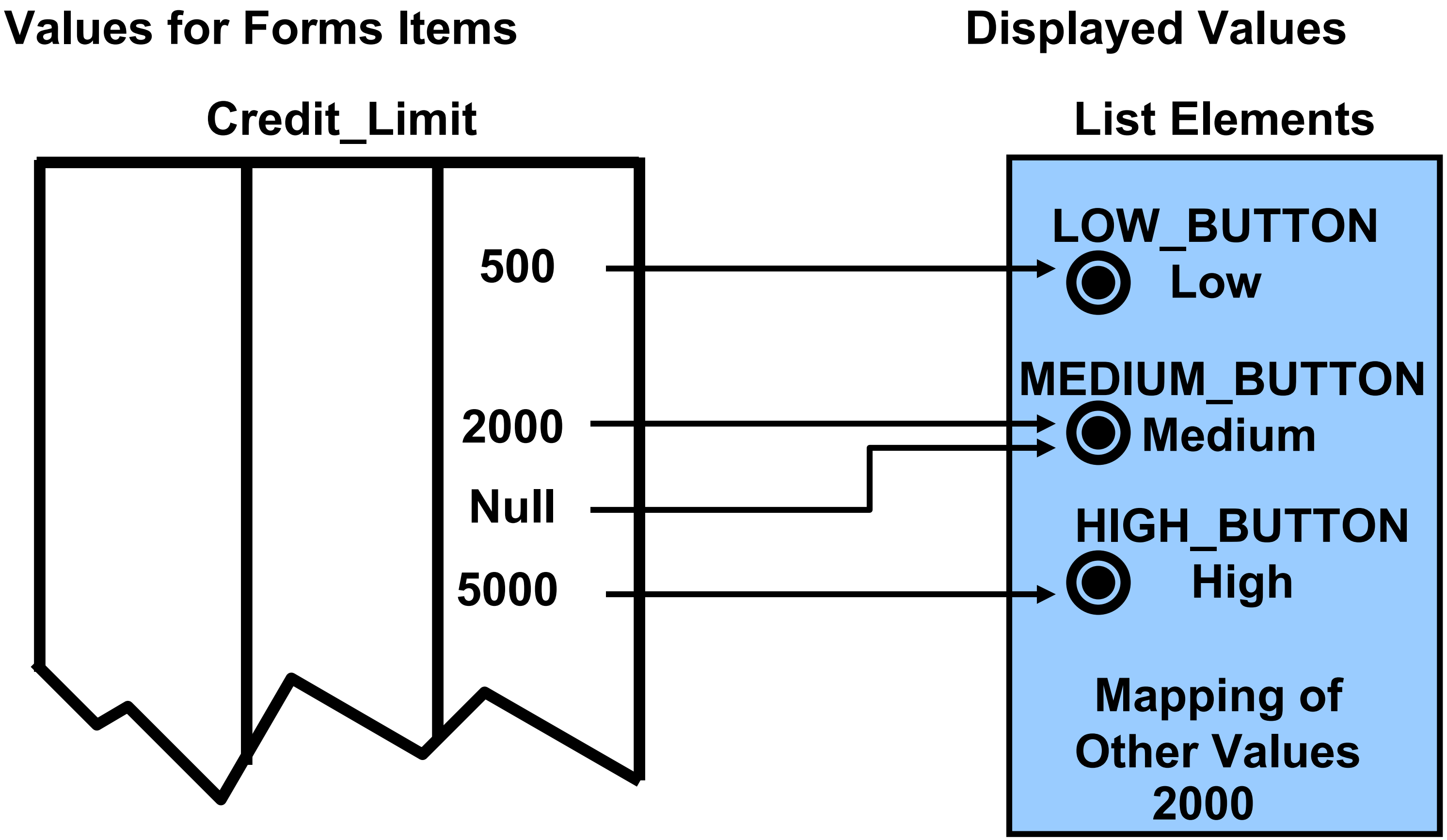
Radio button:



Radio Button: RADIO_BUTTON20

⊟ General	
Name	RADIO_BUTTON
Subclass Information	
Comments	
⊟ Functional	
Enabled	Yes
Label	Low
Access Key	
Radio Button Value	500
⊞ Records	
⊞ Physical	
⊞ Visual Attributes	
⊞ Color	
⊞ Font	

Radio Group Mapping of Other Values



Summary

In this lesson, you should have learned that:

- Check boxes, list items, and radio groups are the item types that allow input
- You create these items by:
  - Changing the item type of an existing item
  - Using the appropriate tool in the Layout Editor
- You can use a check box for items that have only two possible states
- You can use a list item to enable users to pick from a list of mutually exclusive choices
- You can use a radio group for two or three mutually exclusive alternatives

Practice 9 Overview

This practice covers the following topics:

- Converting a text item into a list item
- Converting a text item into a check box item
- Converting a text item into a radio group
- Adding radio buttons to the radio group

10

Creating Noninput Items

Objectives

After completing this lesson, you should be able to do the following:

- Identify item types that do not allow input
- Create a display item
- Create an image item
- Create a button
- Create a calculated item
- Create a hierarchical tree item
- Create a bean area item

Noninput Items Overview

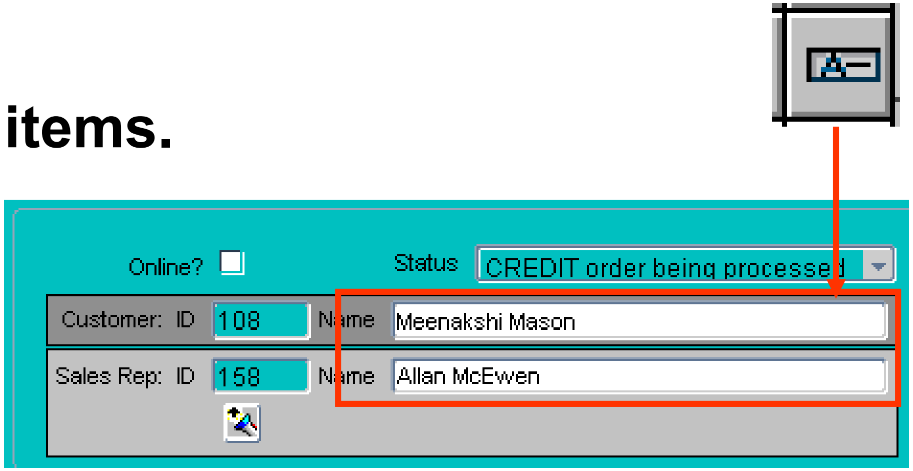
Item types that do not accept direct user input include:

- Display items
- Image items
- Buttons
- Calculated items
- Hierarchical tree items
- Bean area items

Display Items

Display items:

- Are similar to text items.
- Cannot:
  - Be edited
  - Be queried
  - Be navigated to
  - Accept user input
- Can display:
  - Nonbase table information
  - Derived values

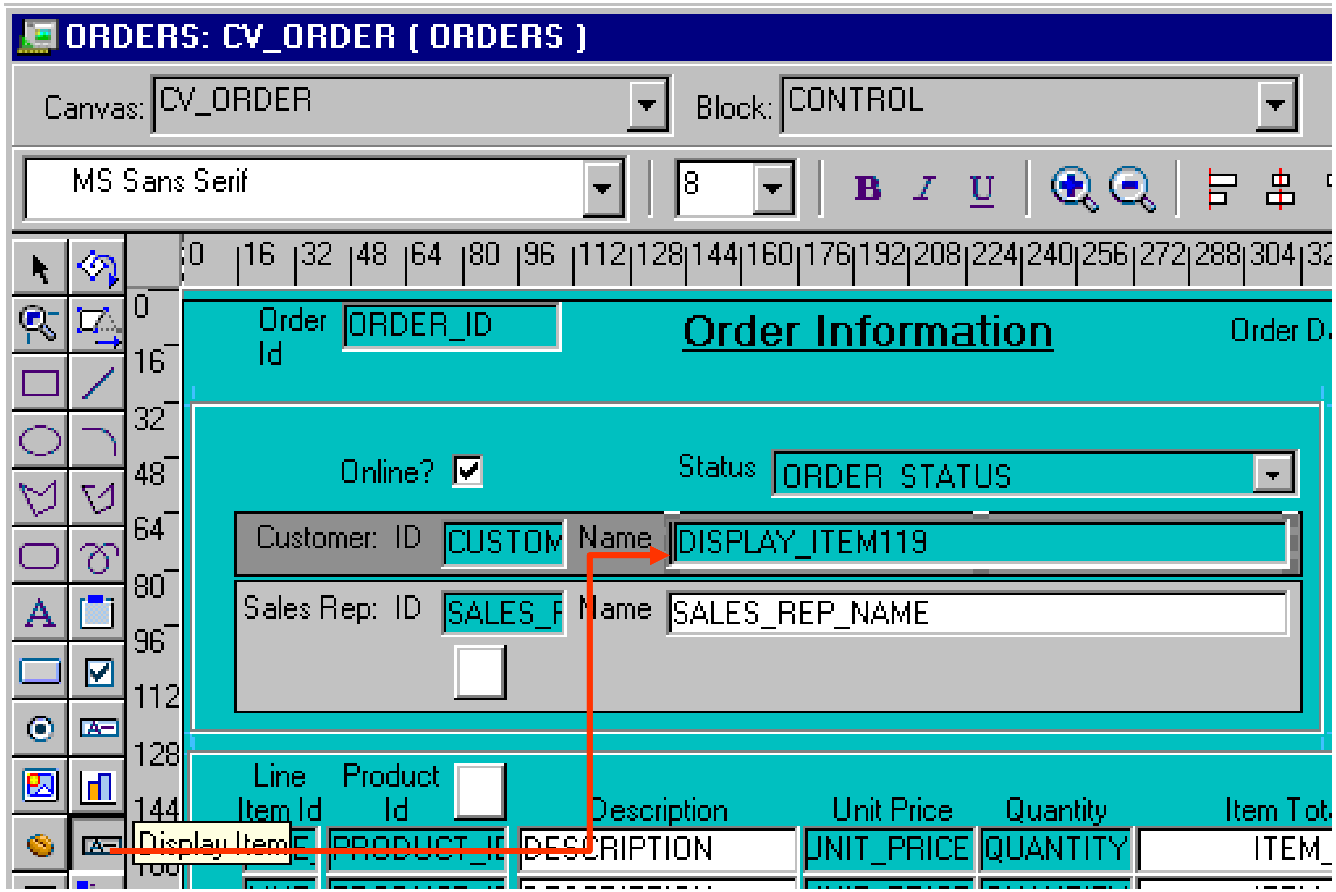


Online? □ Status CREDIT order being processed

Customer: ID 108 Name Meenakshi Mason

Sales Rep: ID 158 Name Allan McEwen

Creating a Display Item



ORDERS: CV_ORDER (ORDERS)

Canvas: CV_ORDER

Block: CONTROL

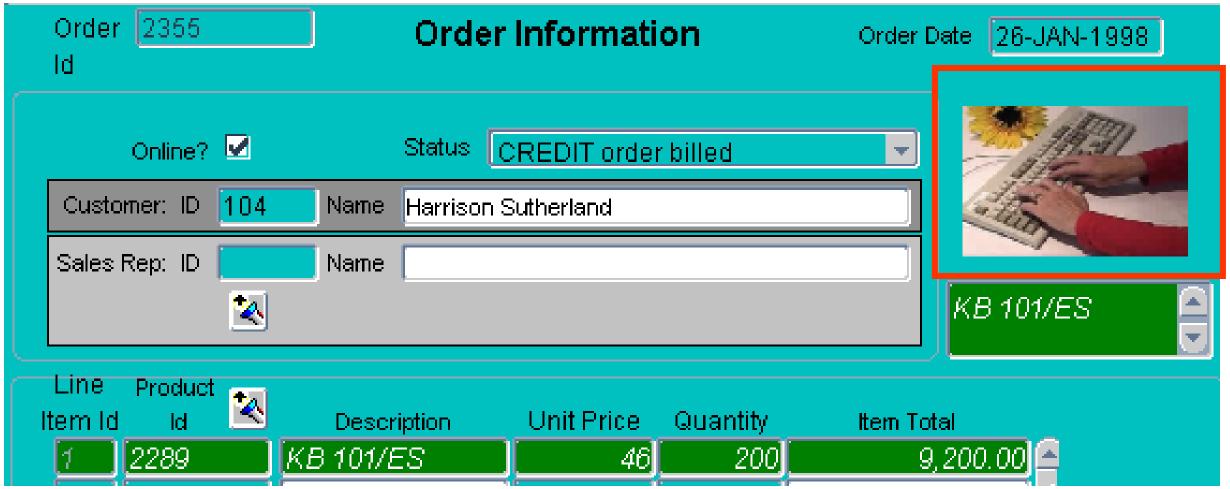
MS Sans Serif

8

Image Items

Use image items to display images:

- From file system—supported file type
- From database—LONG RAW column or a BLOB column



Order Id 2355

Order Information

Order Date 26-JAN-1998

Online? ☑

Status CREDIT order billed

Customer: ID 104 Name Harrison Sutherland

Sales Rep: ID Name

KB 101/ES

Line Item Id	Product Id	Description	Unit Price	Quantity	Item Total
1	2289	KB 101/ES	46	200	9,200.00

Image File Formats

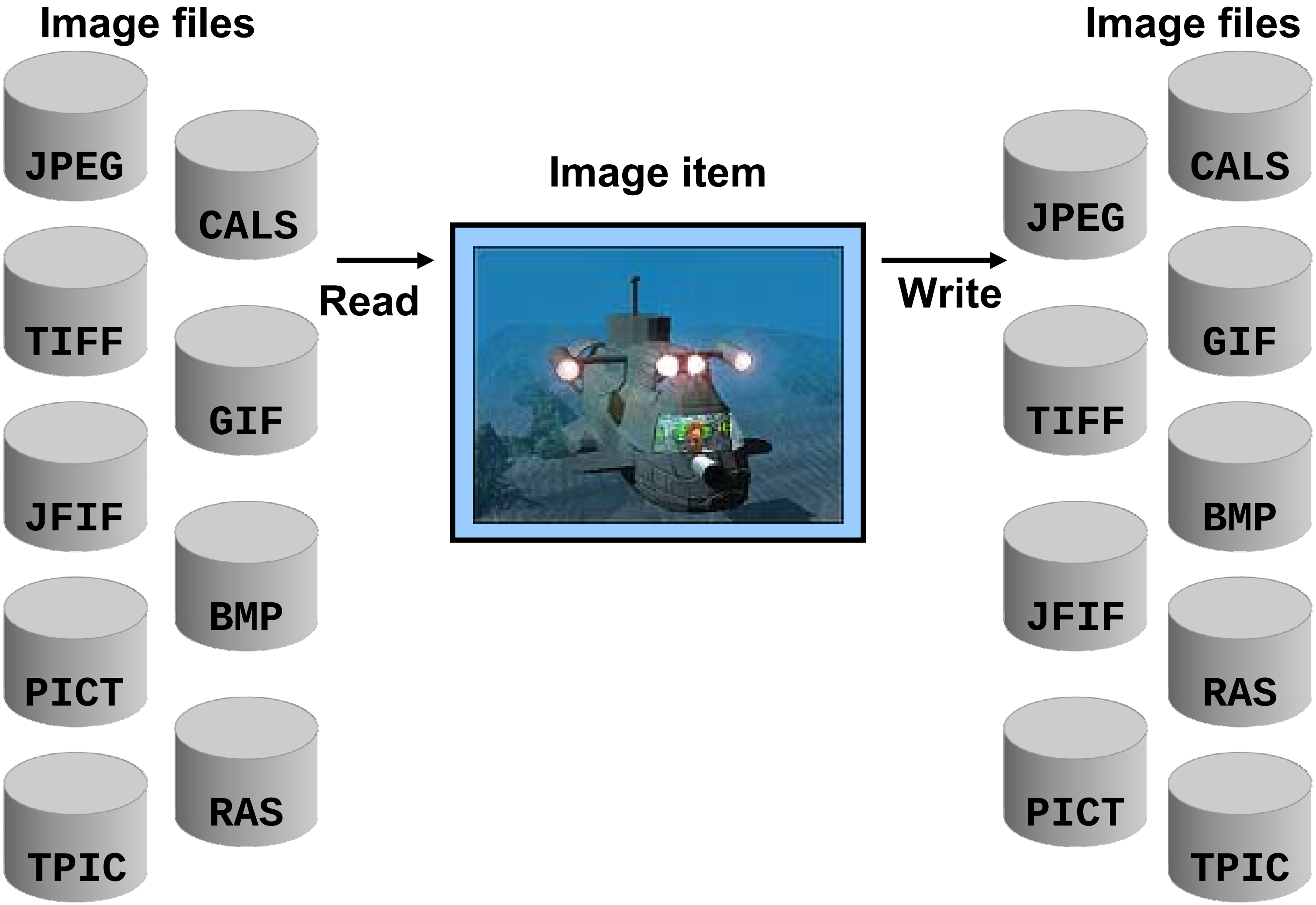


Image files

JPEG

CALS

TIFF

GIF

JFIF

BMP

PICT

RAS

TPIC

Read

Image item

Write

Image files

JPEG

CALS

TIFF

GIF

JFIF

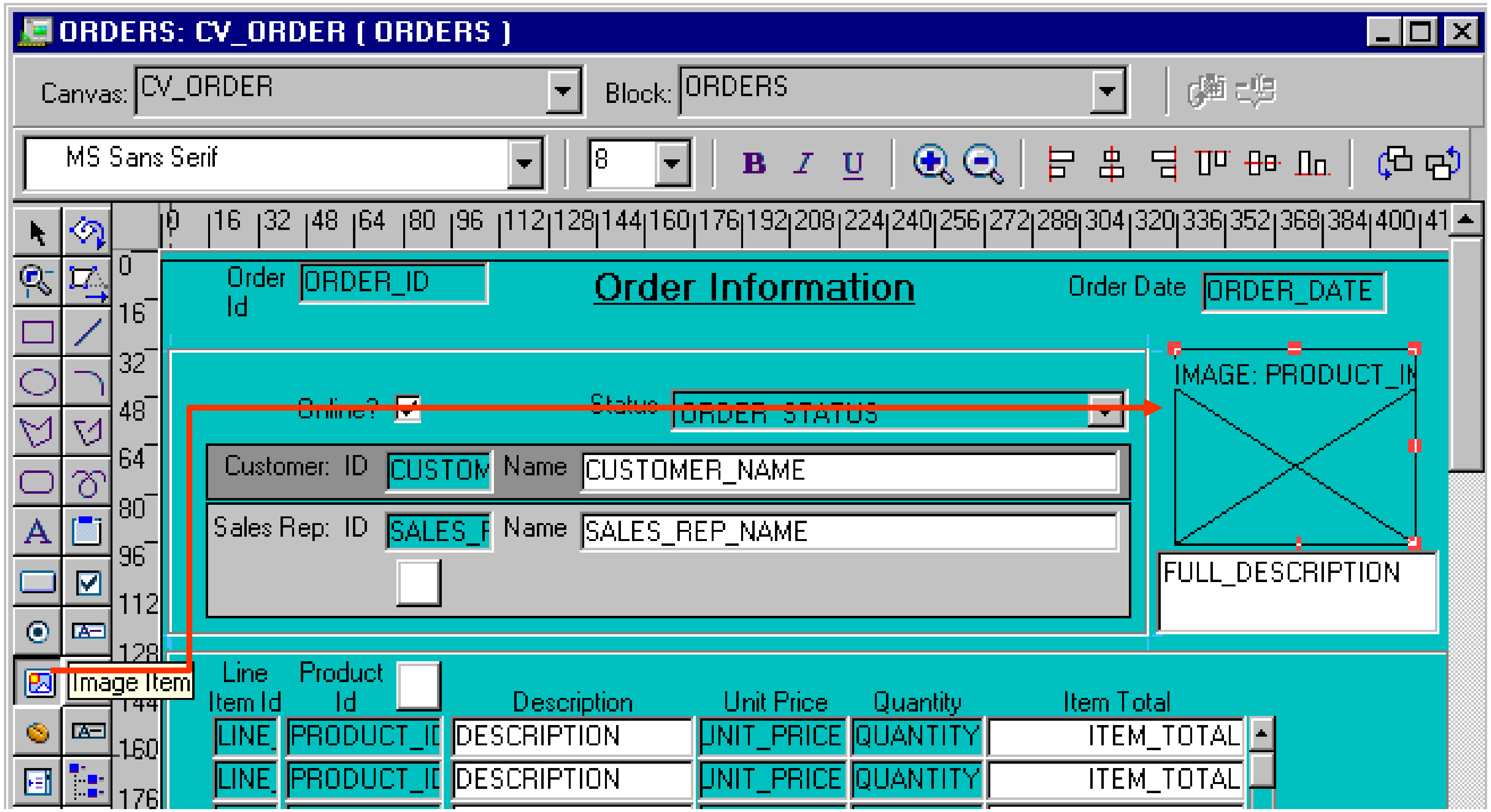
BMP

PICT

RAS

TPIC

Creating an Image Item



ORDERS: CV_ORDER (ORDERS)

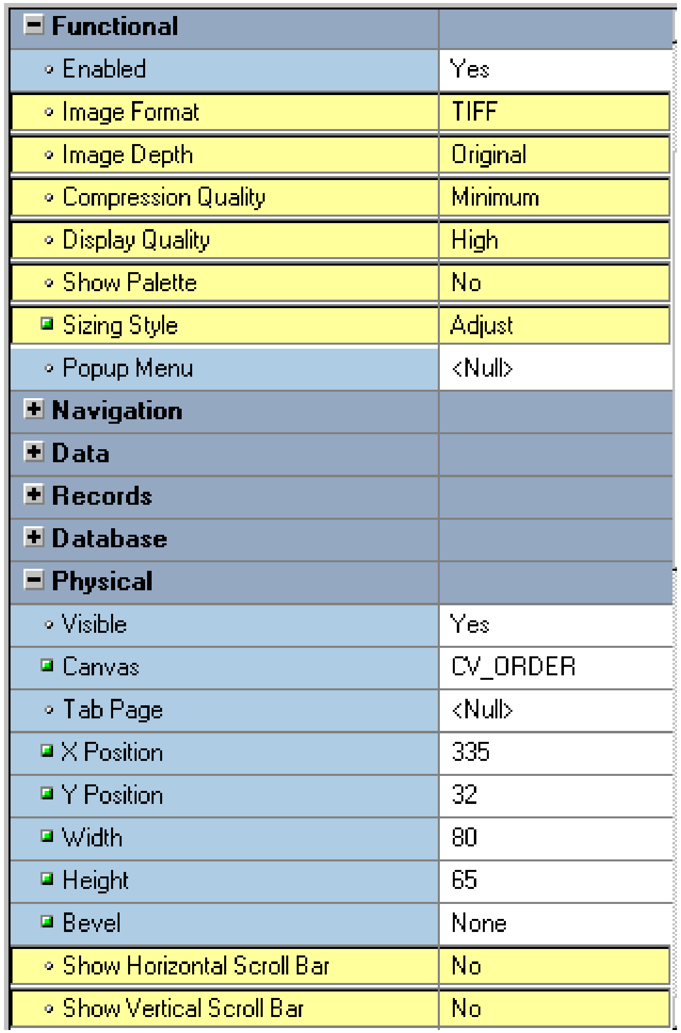
Canvas: CV_ORDER

Block: ORDERS

MS Sans Serif 8

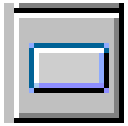
Setting Image-Specific Item Properties

- Image Format
- Image Depth
- Compression Quality
- Display Quality
- Sizing Style
- Show Horizontal Scroll Bar
- Show Vertical Scroll Bar



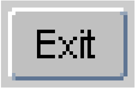
⊟ **Functional**	
Enabled	Yes
Image Format	TIFF
Image Depth	Original
Compression Quality	Minimum
Display Quality	High
Show Palette	No
Sizing Style	Adjust
Popup Menu	<Null>
⊞ **Navigation**	
⊞ **Data**	
⊞ **Records**	
⊞ **Database**	
⊟ **Physical**	
Visible	Yes
Canvas	CV_ORDER
Tab Page	<Null>
X Position	335
Y Position	32
Width	80
Height	65
Bevel	None
Show Horizontal Scroll Bar	No
Show Vertical Scroll Bar	No

Push Buttons



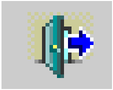
Push buttons:

- Cannot display or represent data
- Are used to initiate an action
- Display as:
  - Text button



Exit

  - Iconic

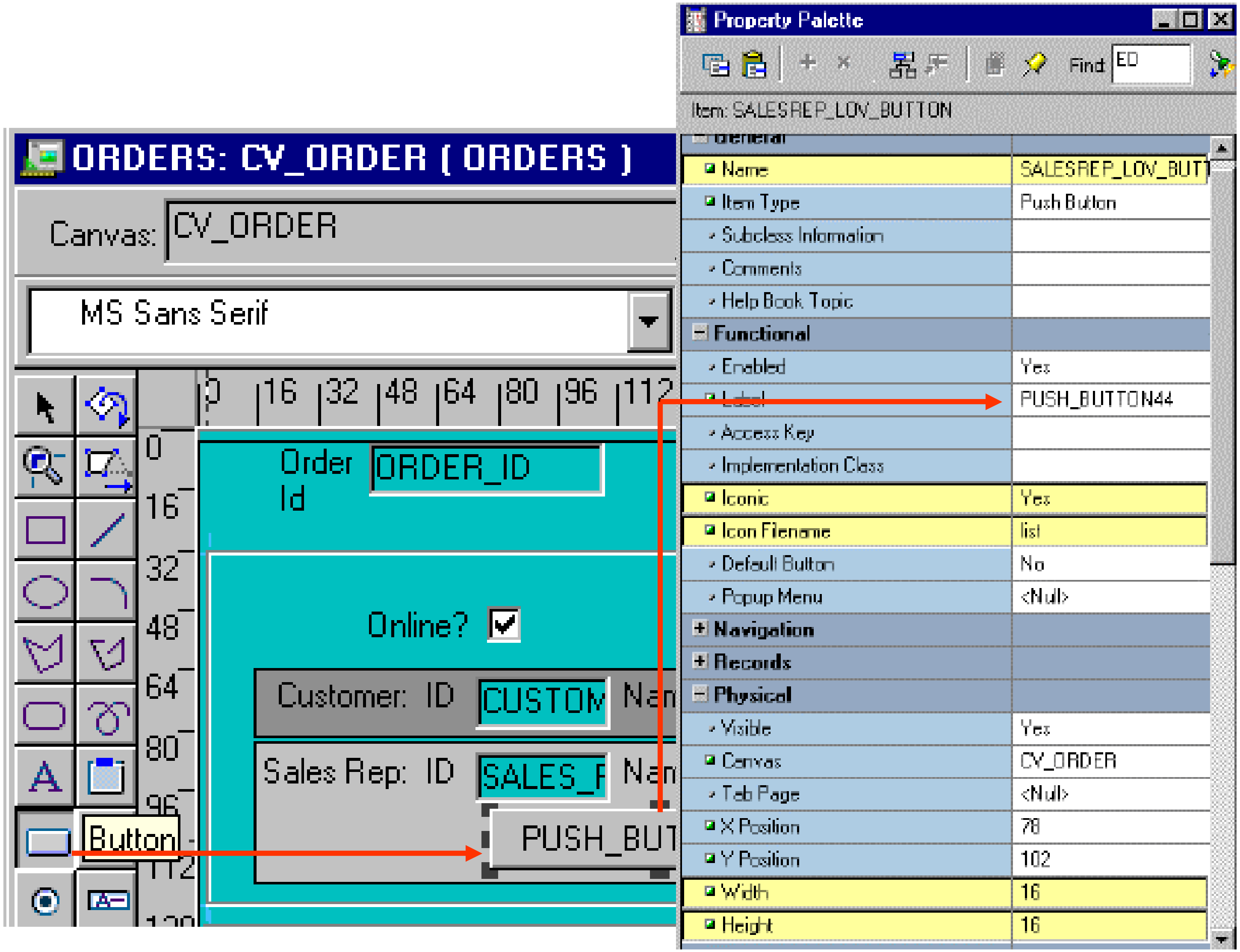


Push Button Actions

Use buttons to:

- Move input focus
- Display an LOV
- Invoke an editor
- Invoke another window
- Commit data
- Issue a query
- Perform calculations

Creating a Push Button

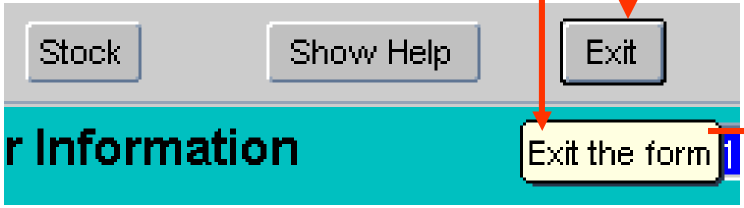


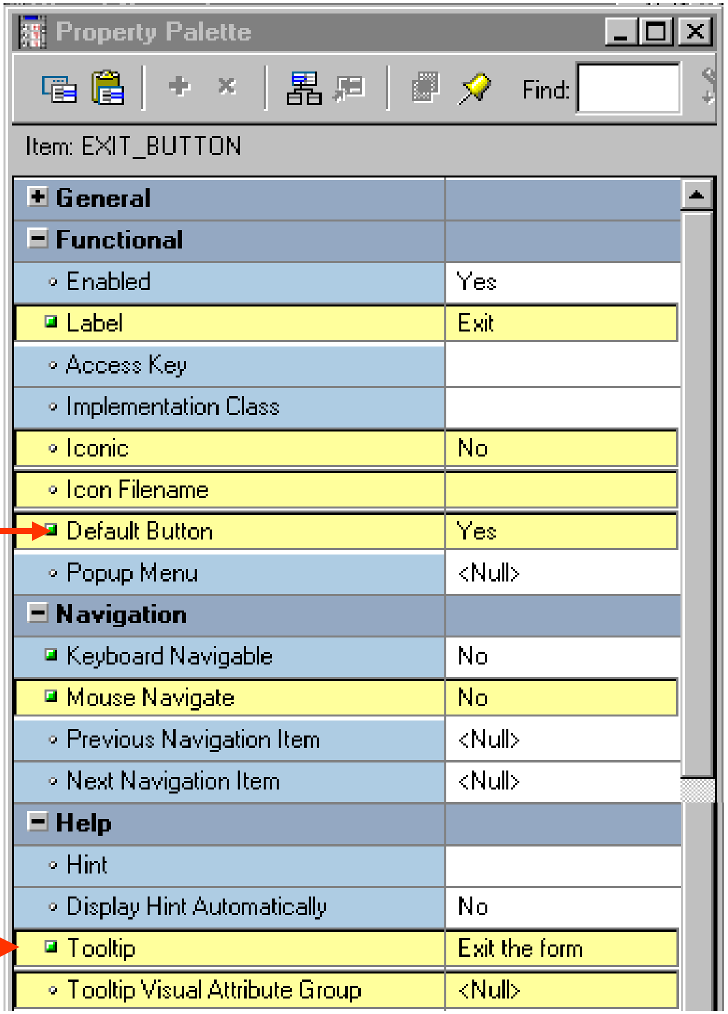
Item: SALESREP_LOV_BUTTON

General	
Name	SALESREP_LOV_BUTT
Item Type	Push Button
Subclass Information	
Comments	
Help Book Topic	
Functional	
Enabled	Yes
Label	PUSH_BUTTON44
Access Key	
Implementation Class	
Iconic	Yes
Icon Filename	list
Default Button	No
Popup Menu	<Null>
Navigation	
Records	
Physical	
Visible	Yes
Canvas	CV_ORDER
Tab Page	<Null>
X Position	78
Y Position	102
Width	16
Height	16

Setting Push Button Properties

- Label
- Iconic
- Icon Filename
- Default Button
- Mouse Navigate
- Tooltip
- Tooltip Visual Attribute Group





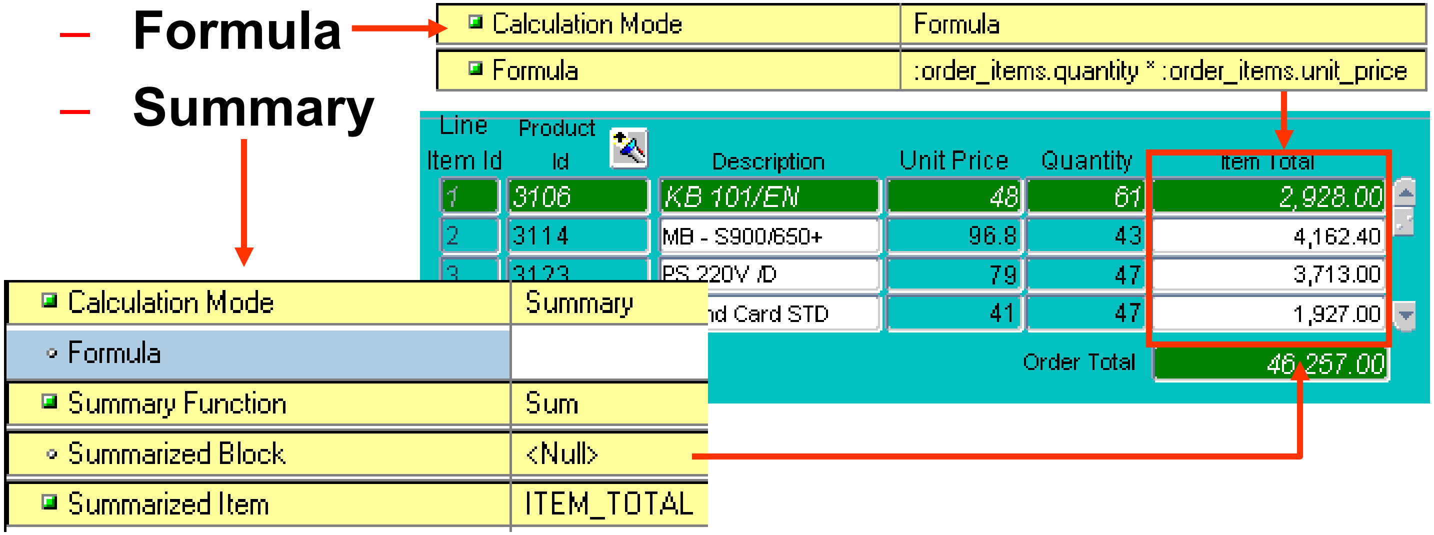
Item: EXIT_BUTTON

⊞ General	
⊟ Functional	
Enabled	Yes
Label	Exit
Access Key	
Implementation Class	
Iconic	No
Icon Filename	
Default Button	Yes
Popup Menu	<Null>
⊟ Navigation	
Keyboard Navigable	No
Mouse Navigate	No
Previous Navigation Item	<Null>
Next Navigation Item	<Null>
⊟ Help	
Hint	
Display Hint Automatically	No
Tooltip	Exit the form
Tooltip Visual Attribute Group	<Null>

Calculated Items

What are calculated items?

- They accept item values that are based on calculations.
- They are read-only.
- They can be expressed as:
  - Formula
  - Summary

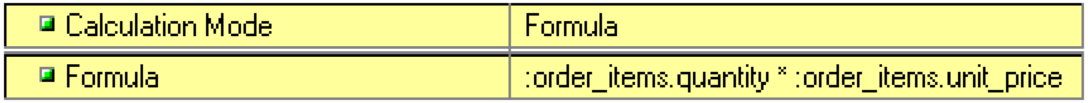


■ Calculation Mode	Formula
■ Formula	:order_items.quantity * :order_items.unit_price

Line Item Id	Product Id	Description	Unit Price	Quantity	Item Total
1	3106	KB 101/EN	48	61	2,928.00
2	3114	MB - S900/650+	96.8	43	4,162.40
3	3123	PS 220V /D	79	47	3,713.00
		nd Card STD	41	47	1,927.00
				Order Total	46,257.00

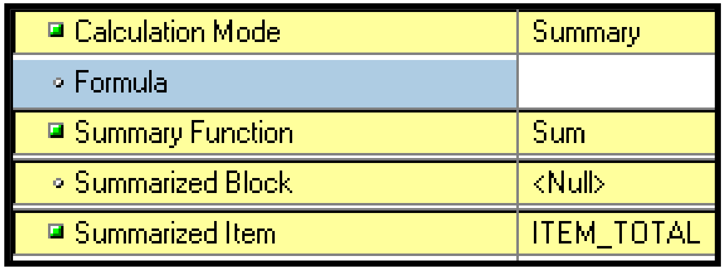
■ Calculation Mode	Summary
○ Formula	
■ Summary Function	Sum
○ Summarized Block	<Null>
■ Summarized Item	ITEM_TOTAL

Creating a Calculated Item by Setting Properties



Calculation Mode	Formula
Formula	:order_items.quantity * :order_items.unit_price

- Formula
  - A calculated item value is the result of a horizontal calculation.
  - It involves bind variables.
- Summary
  - A calculated item value is a vertical calculation.
  - A summary is performed on values of a single item over all rows in a block.

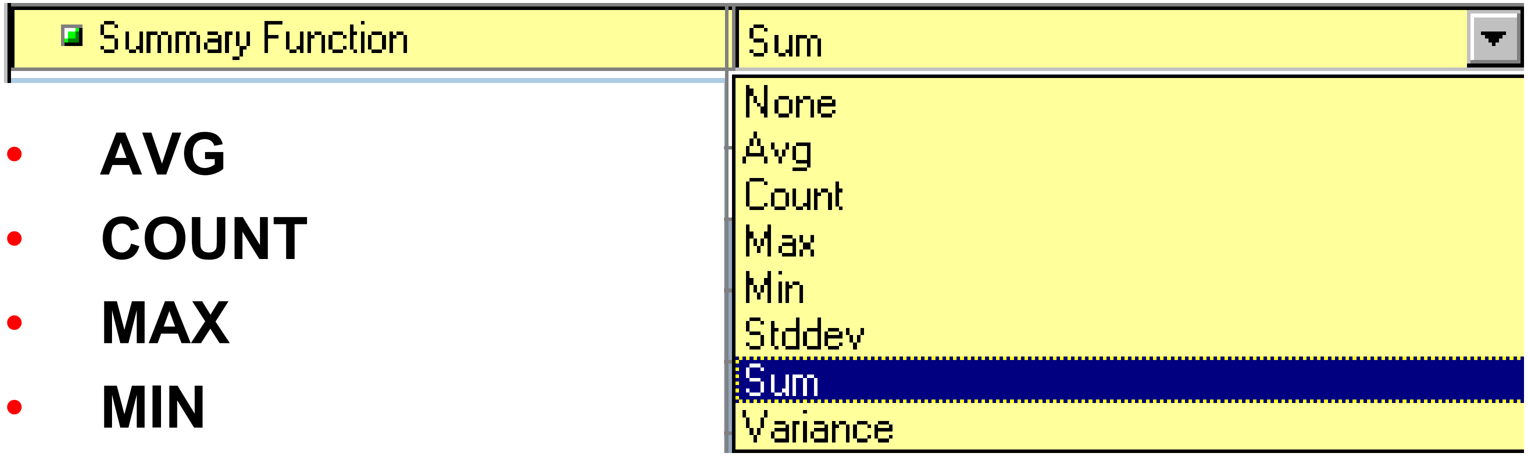


Calculation Mode	Summary
Formula	
Summary Function	Sum
Summarized Block	<Null>
Summarized Item	ITEM_TOTAL

Setting Item Properties for the Calculated Item

- Formula
  - Calculation Mode
  - Formula
- Summary
  - Calculation Mode
  - Summary Function
  - Summarized Block
  - Summarized Item

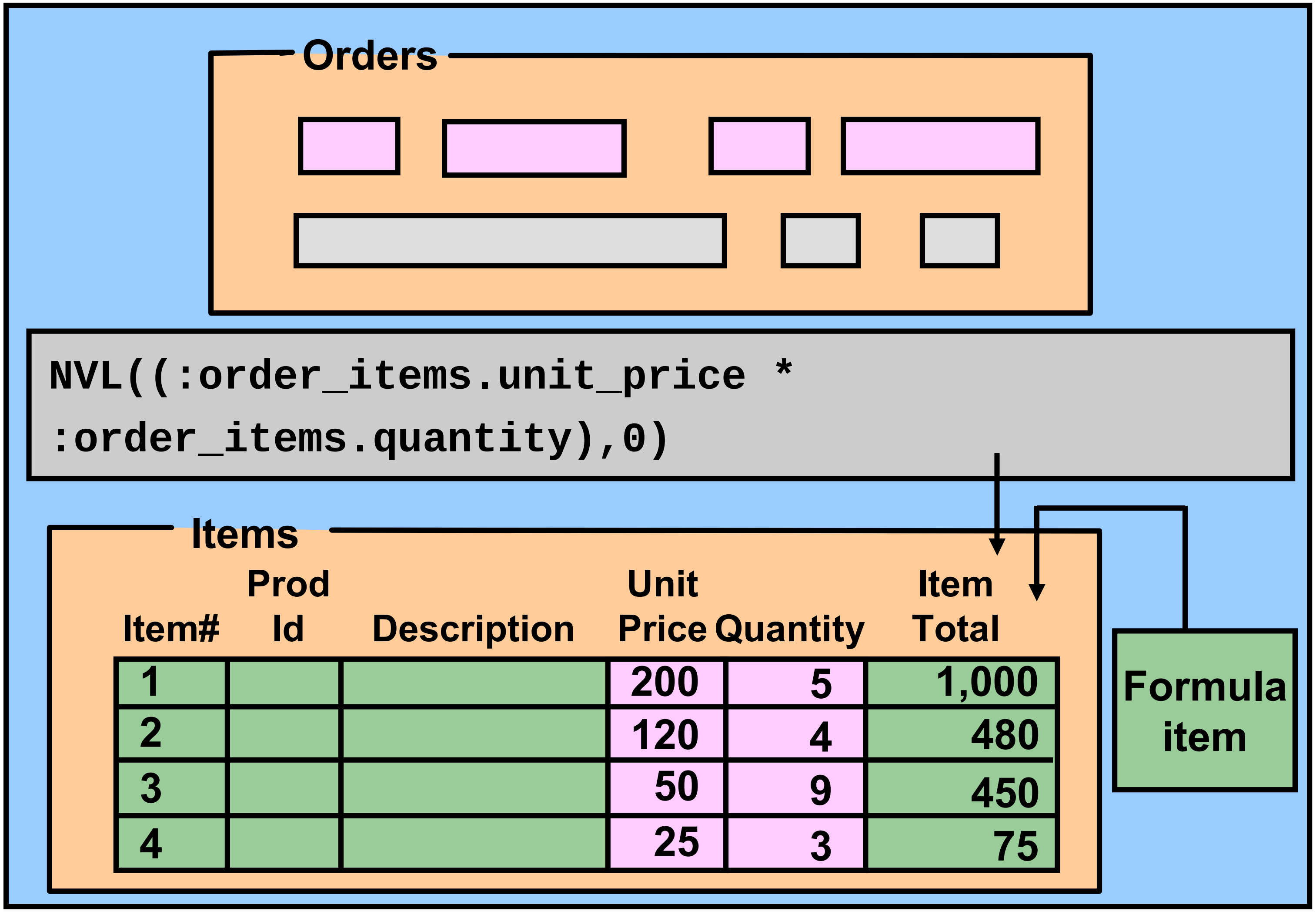
Summary Functions



Summary Function: Sum

- AVG
- COUNT
- MAX
- MIN
- STDDEV
- SUM
- VARIANCE

Calculated Item Based on a Formula



```
NVL((:order_items.unit_price *
:order_items.quantity),0)
```

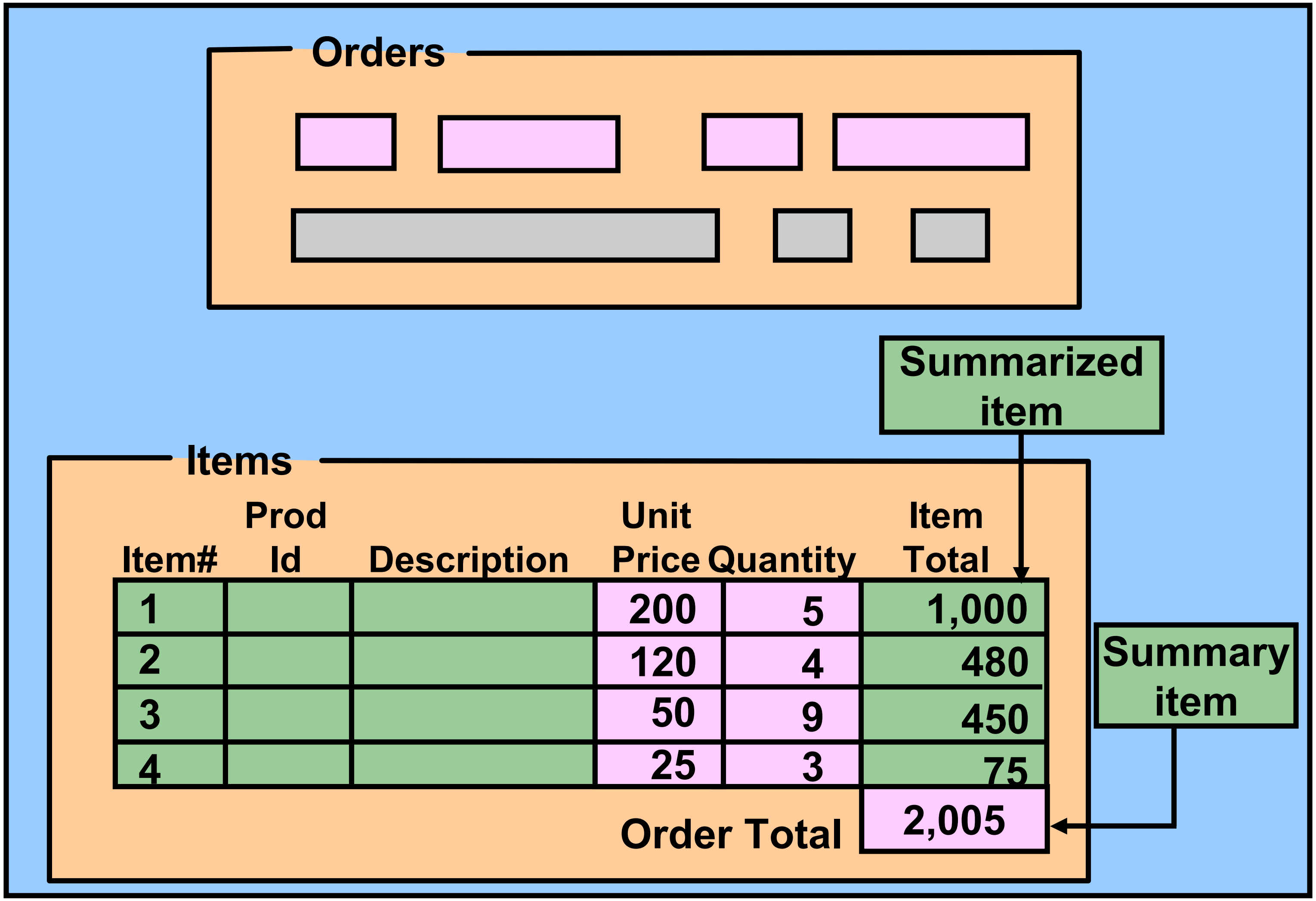
Item#	Prod Id	Description	Unit Price	Quantity	Item Total
1			200	5	1,000
2			120	4	480
3			50	9	450
4			25	3	75

Rules for Calculated Item Formulas

Create calculated item formulas according to the following rules:

- A formula item must not invoke restricted built-ins.
- A formula item cannot execute any DML statements.
- Do not terminate a PL/SQL expression with a semicolon.
- Do not enter a complete PL/SQL statement in assignment expressions.

Calculated Item Based on a Summary

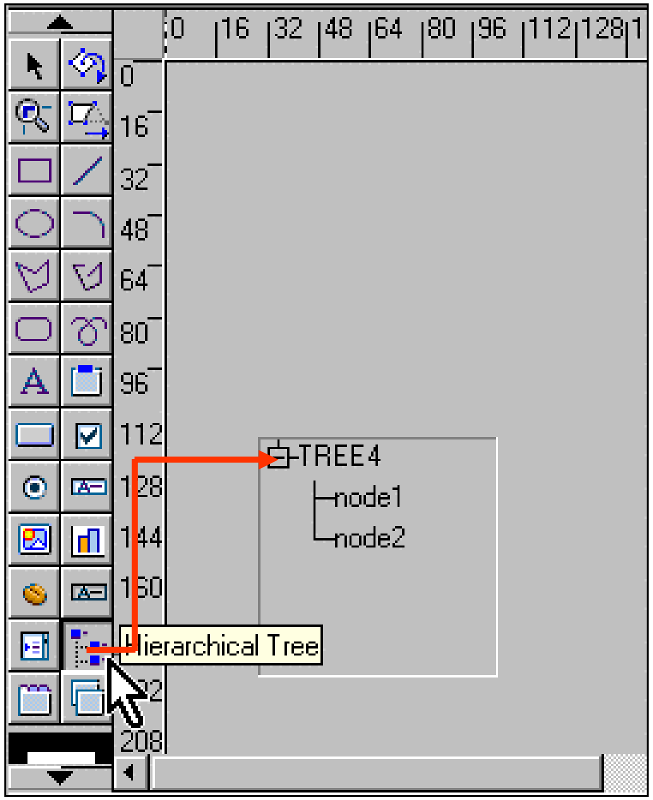


Item#	Prod Id	Description	Unit Price	Quantity	Item Total
1			200	5	1,000
2			120	4	480
3			50	9	450
4			25	3	75
				Order Total	2,005

Rules for Summary Items

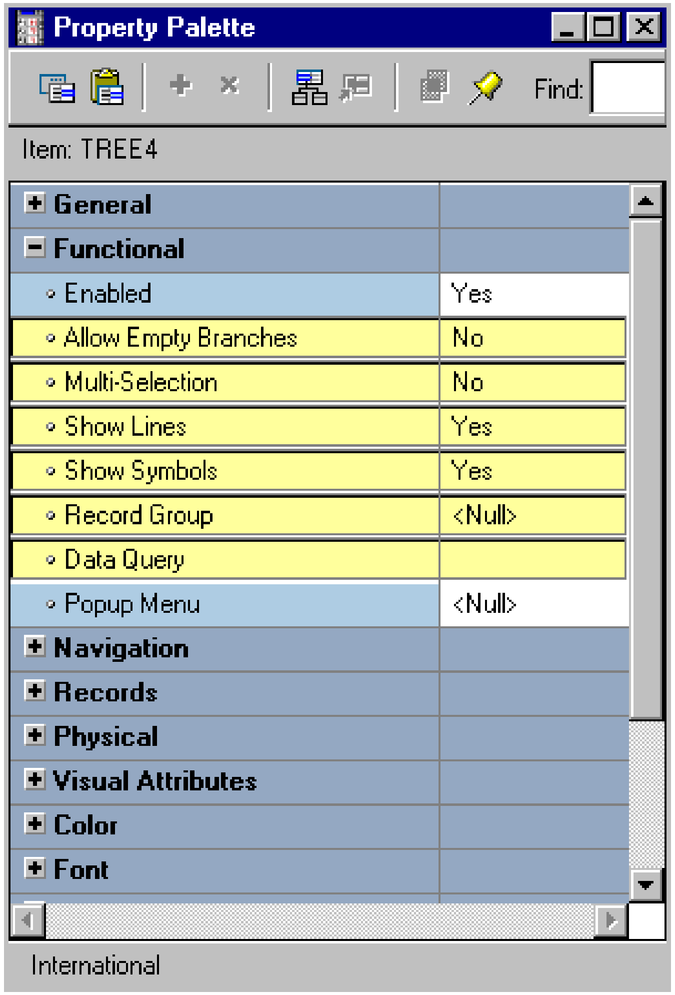
- Summary item must reside in:
  - The same block as the summarized item
  - A control block with Single Record property set to Yes
- Summarized item must reside in:
  - A data block with Query All Records property or Precompute Summaries property set to Yes
  - A control block
- Datatype of summary item must be Number, unless using MAX or MIN

Creating a Hierarchical Tree Item



Setting Hierarchical Tree Item Properties

- Allow empty branches
- Multi selection
- Show lines
- Show symbols
- Record group
- Data query



Item: TREE4

+ General	
− Functional	
Enabled	Yes
Allow Empty Branches	No
Multi-Selection	No
Show Lines	Yes
Show Symbols	Yes
Record Group	<Null>
Data Query	
Popup Menu	<Null>
+ Navigation	
+ Records	
+ Physical	
+ Visual Attributes	
+ Color	
+ Font	

International

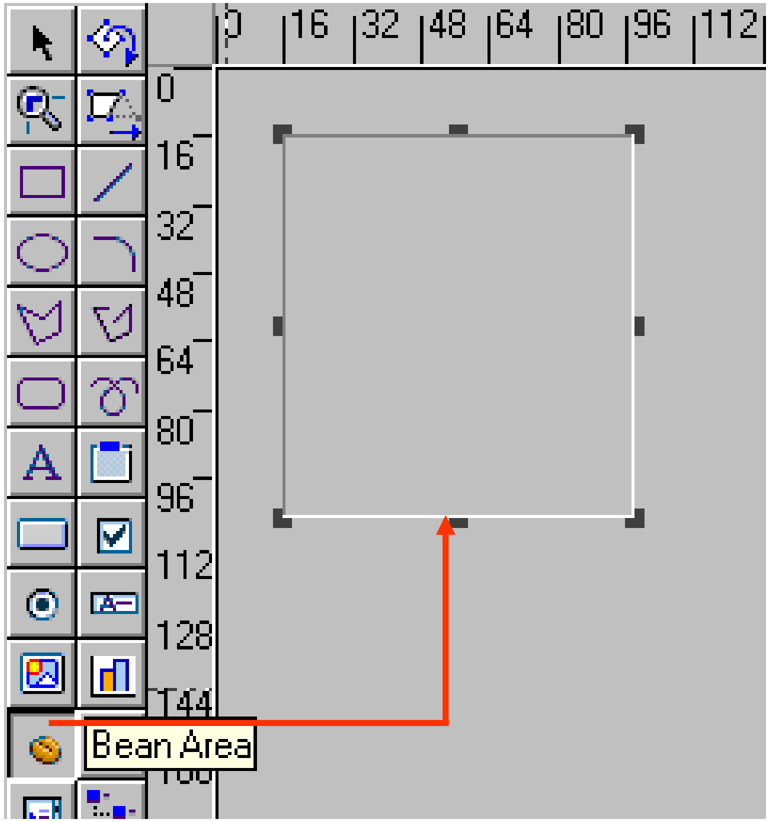
Bean Area Items

The Bean Area item enables you to:

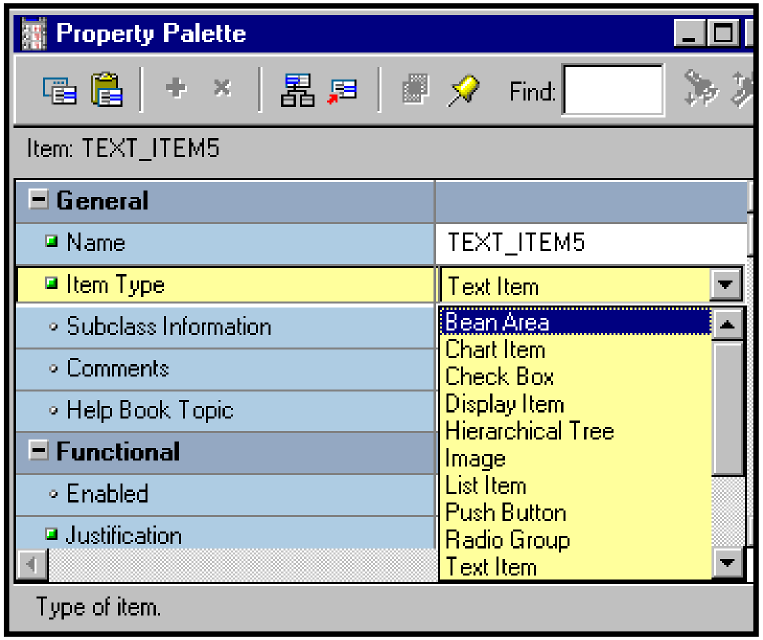
- Add a JavaBean to a form
- Extend Forms functionality
- Interact with client machine
- Reduce network traffic

Creating a Bean Area Item

Create bean area in Layout Editor



Convert existing item to bean area

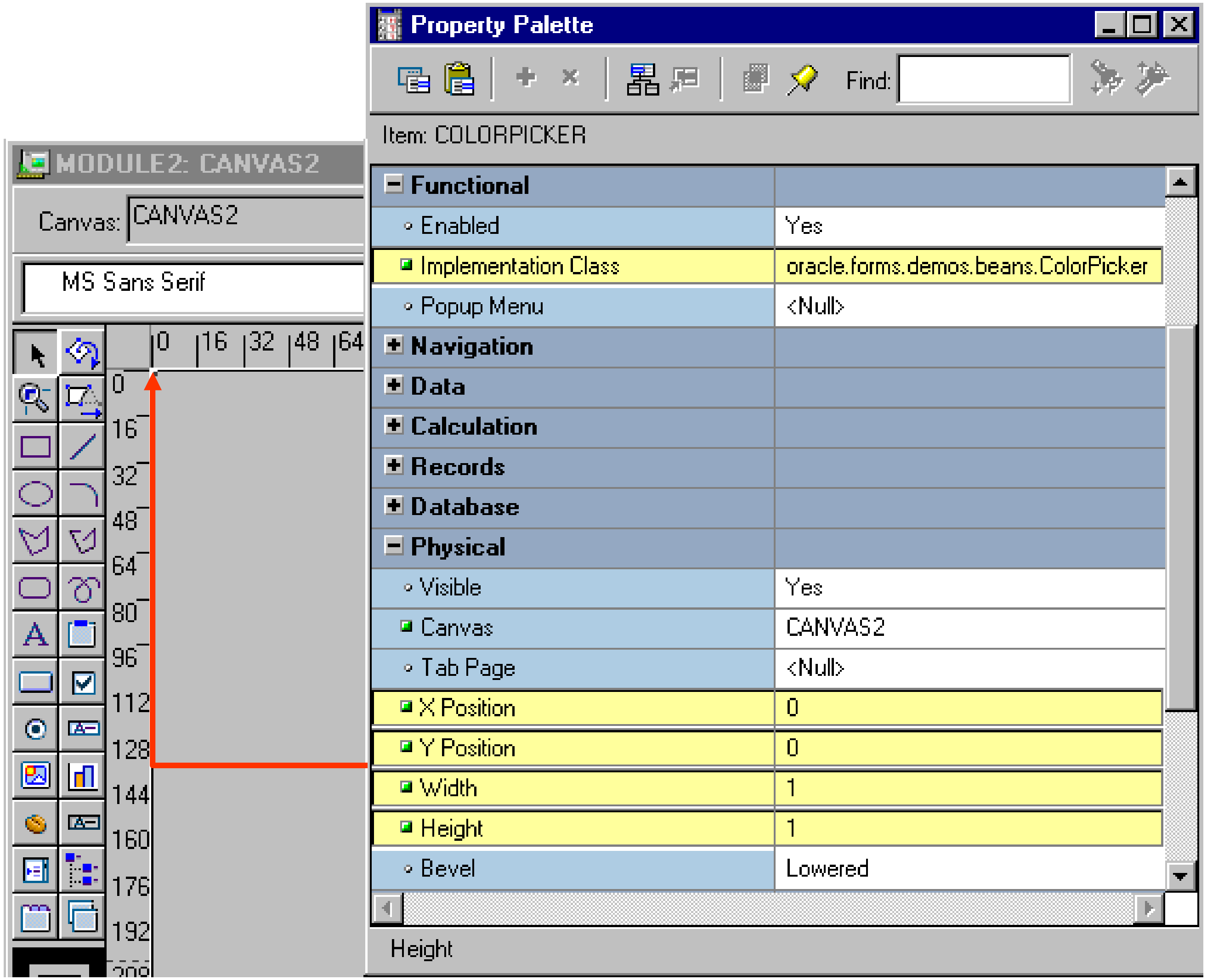


Item: TEXT_ITEM5

General	
Name	TEXT_ITEM5
Item Type	Text Item
Subclass Information	Bean Area Chart Item Check Box Display Item Hierarchical Tree Image List Item Push Button Radio Group Text Item
Comments	
Help Book Topic	
Functional	
Enabled	
Justification	

Type of item.

Setting Bean Area Item Properties

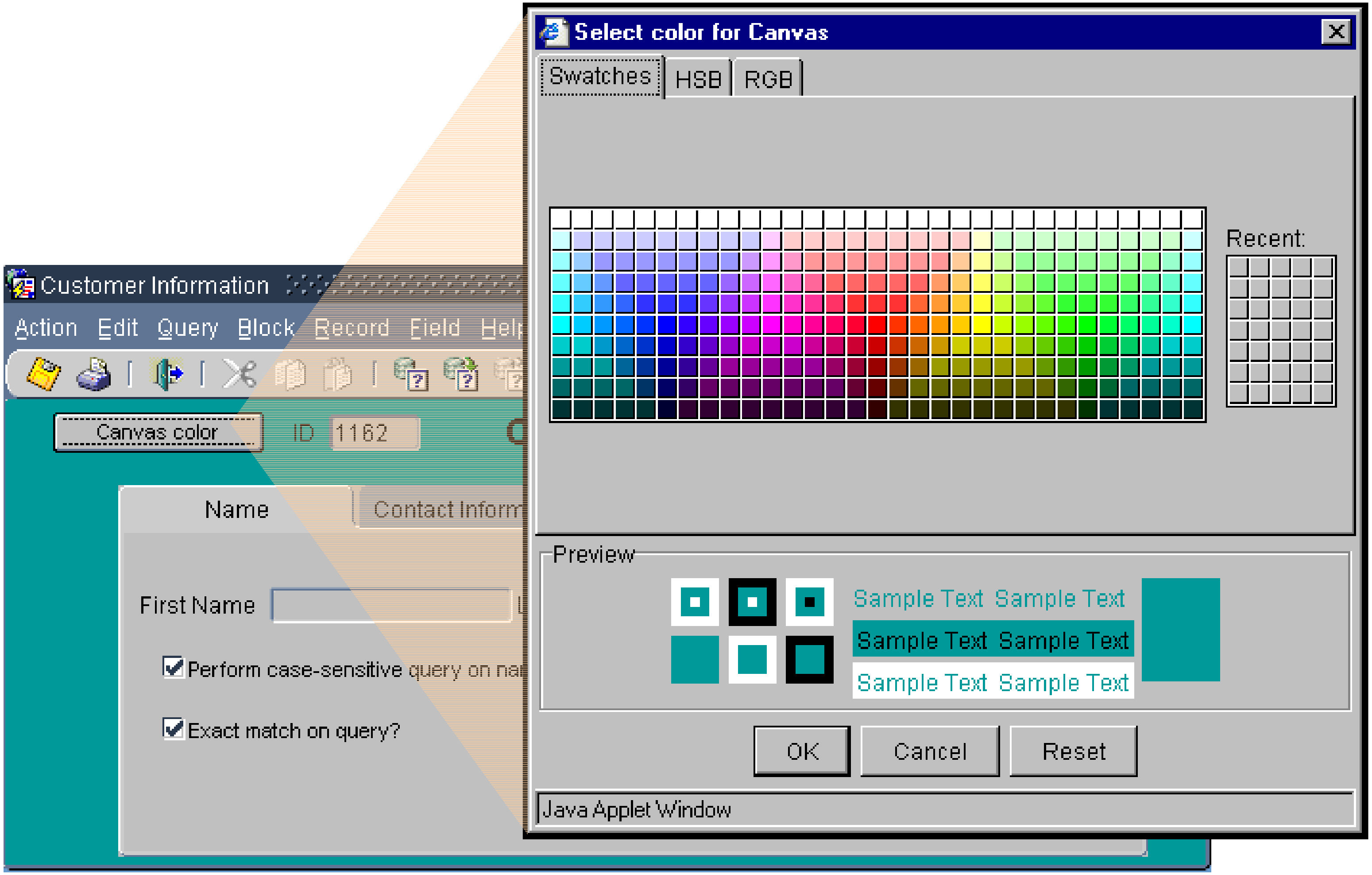


Item: COLORPICKER

⊟ Functional	
∘ Enabled	Yes
▪ Implementation Class	oracle.forms.demos.beans.ColorPicker
∘ Popup Menu	<Null>
⊞ Navigation	
⊞ Data	
⊞ Calculation	
⊞ Records	
⊞ Database	
⊟ Physical	
∘ Visible	Yes
▪ Canvas	CANVAS2
∘ Tab Page	<Null>
▪ X Position	0
▪ Y Position	0
▪ Width	1
▪ Height	1
∘ Bevel	Lowered

Height

The JavaBean at Run Time



Summary

In this lesson, you should have learned that:

- The following item types do not allow input:
  - Display items
  - Image items
  - Push buttons
  - Calculated items
  - Hierarchical tree items
  - Bean area items
- You create noninput items by:
  - Changing the type of an existing item and setting certain properties
  - Using the appropriate tool in the Layout Editor

Summary

- You can use:
  - A display item to show nonbase table information
  - An image item to display an image
  - A push button to initiate action
  - A calculated item to display the results of a formula or a summary function of another item
  - A hierarchical tree item to display related data in a hierarchical fashion
  - A bean area item to execute client-side Java code

Practice 10 Overview

This practice covers the following topics:

- Creating display items
- Creating an image item
- Creating iconic buttons
- Creating calculated items:
  - Formula
  - Summary
- Creating a bean area item

11

Creating Windows and Content Canvases

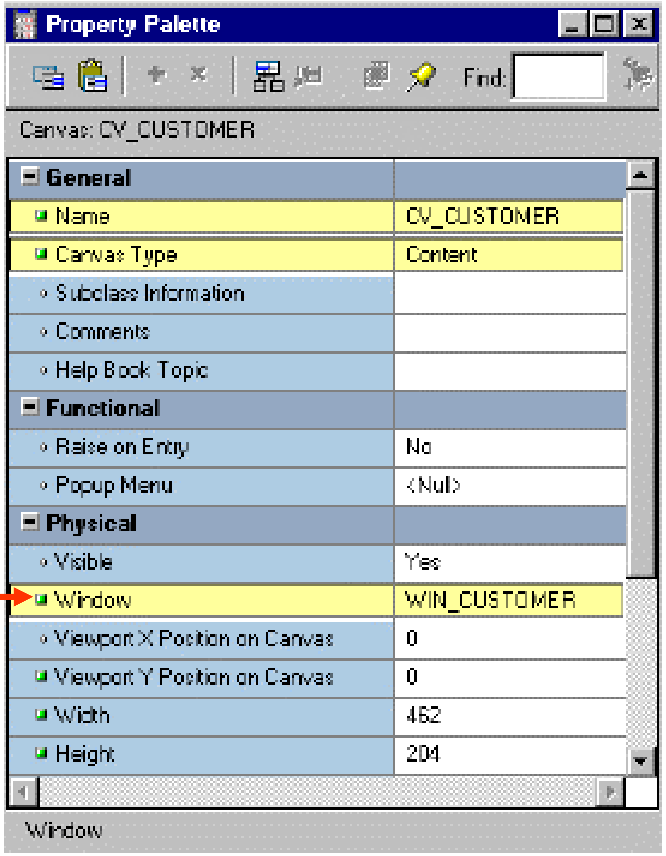
Objectives

After completing this lesson, you should be able to do the following:

- Describe the relationship between windows and content canvases
- Create windows and content canvases
- Display a form module in multiple windows
- Display a form module on multiple layouts

Windows and Canvases

- Window: Container for Forms Builder visual objects
- Canvas: Surface on which you “paint” visual objects
- To see a canvas and its objects, display the canvas
  in a window.

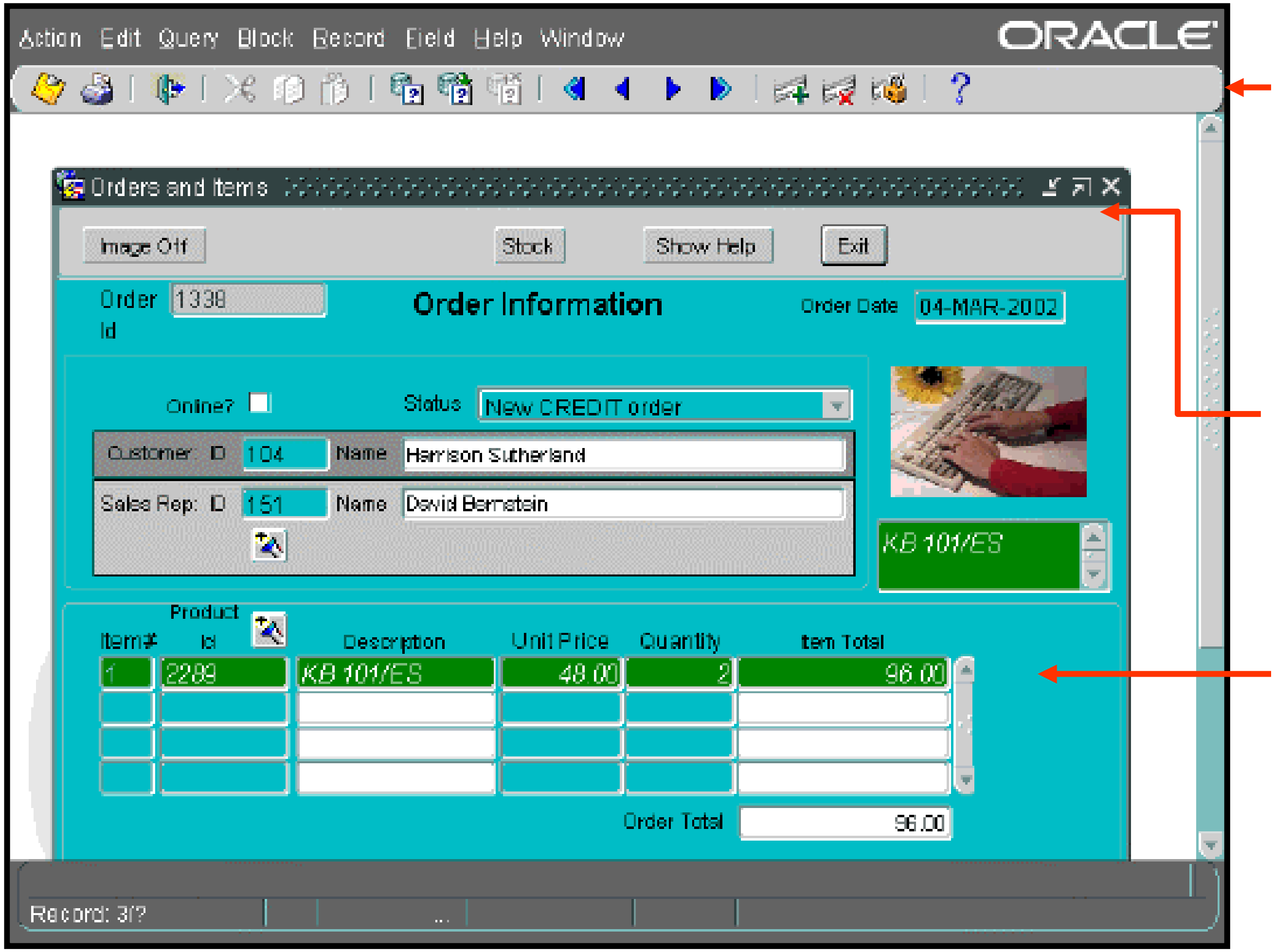


Canvas: CV_CUSTOMER

General	
Name	CV_CUSTOMER
Canvas Type	Content
Subclass Information	
Comments	
Help Book Topic	
Functional	
Raise on Entry	No
Popup Menu	<Null>
Physical	
Visible	Yes
Window	WIN_CUSTOMER
Viewport X Position on Canvas	0
Viewport Y Position on Canvas	0
Width	462
Height	204

Window

Window, Canvas, and Viewport



Order Id: 1338

Order Date: 04-MAR-2002

Online? □

Status: New CREDIT order

Customer ID: 104 Name: Harrison Sutherland

Sales Rep: ID: 151 Name: David Bernstein

KB 101/ES

Item#	Product Id	Description	Unit Price	Quantity	Item Total
1	2289	KB 101/ES	48.00	2	96.00

Order Total: 96.00

Record: 3/?

MDI parent window

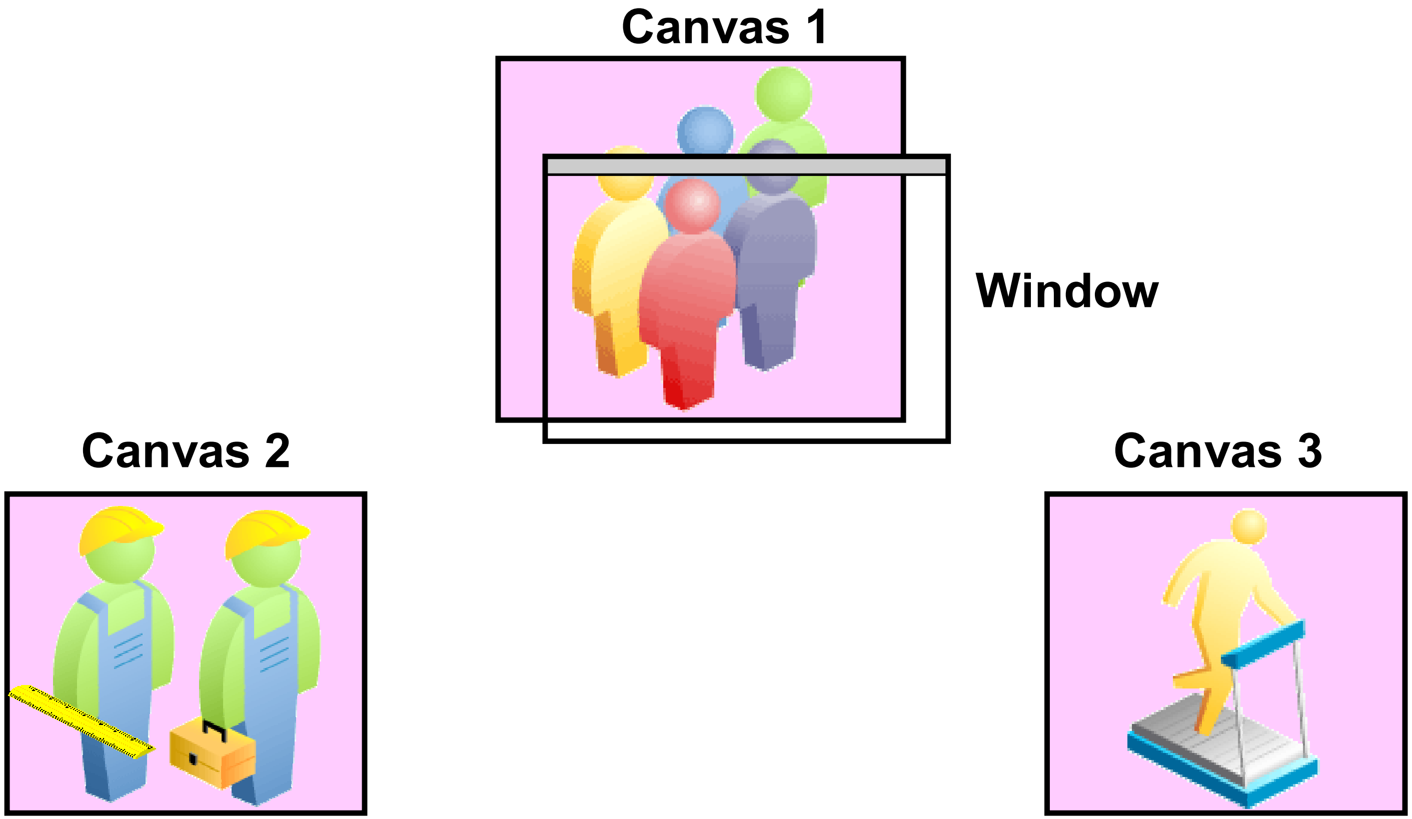
Document window

Canvas

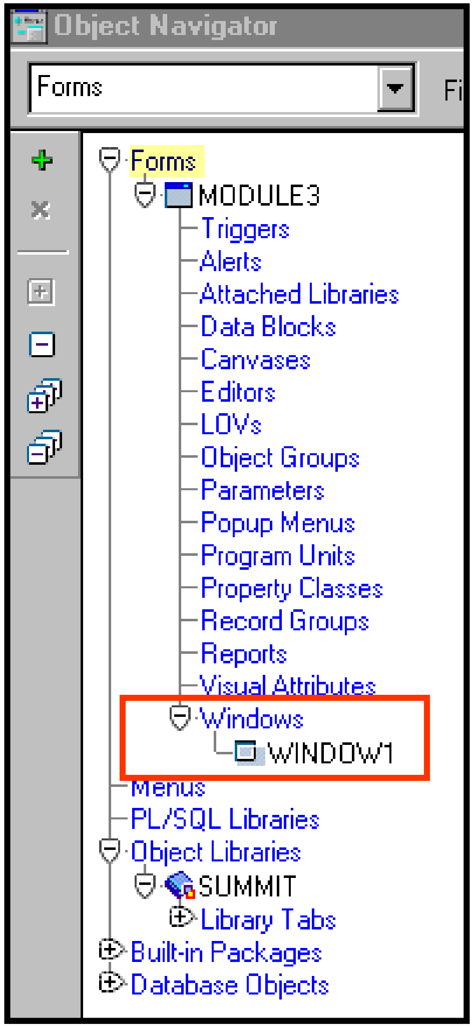
The Content Canvas

- “Base” canvas
- View occupies entire window
- Default canvas type
- Each window should have at least one content canvas

Relationship Between Windows and Content Canvases



The Default Window

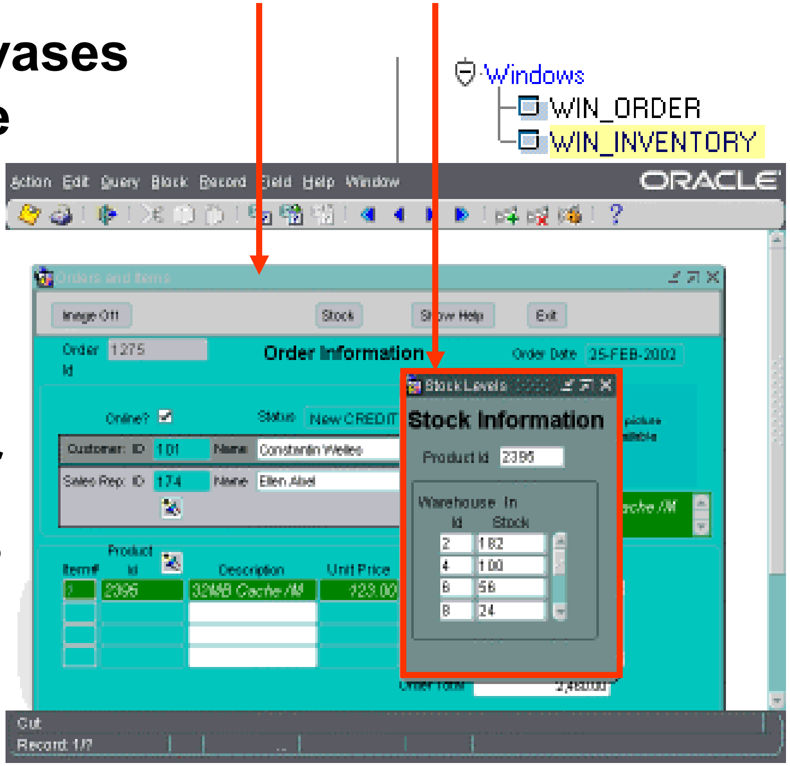


WINDOW1:

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

Displaying a Form Module in Multiple Windows

- Use additional windows to:
  - Display two or more content canvases at once
  - Switch between canvases without replacing the initial one
  - Modularize form contents
  - Take advantage of the window manager
- Two types of windows
  - Modal
  - Modeless



Order Id 1275

Order Date 25-FEB-2002

Online? ☑

Status New CREDIT

Customer ID 101 Name Constantin Welles

Sales Rep ID 174 Name Ellen Abel

Item#	Product Id	Description	Unit Price
1	2395	32MB Cache /M	123.00

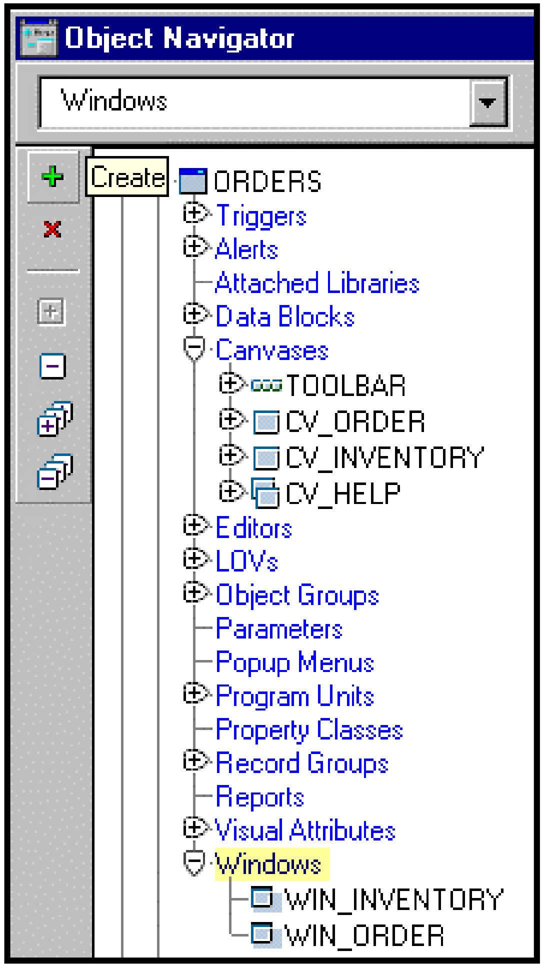
Product Id 2395

Warehouse Id	In Stock
2	182
4	100
6	56
8	24

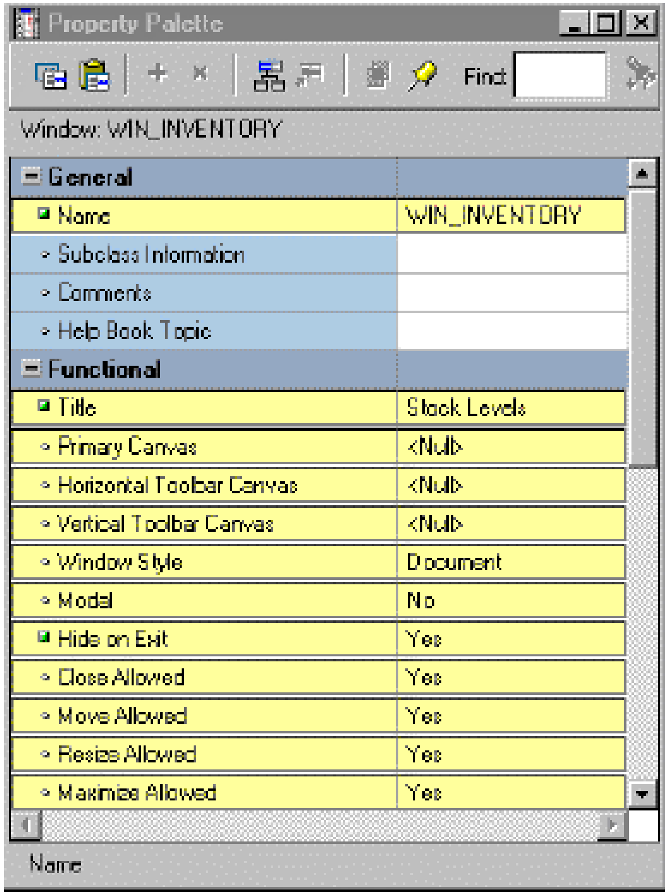
Order Total 2,480.00

Creating a New Window

Object Navigator: Click Create with Windows node selected



Property Palette: Set properties

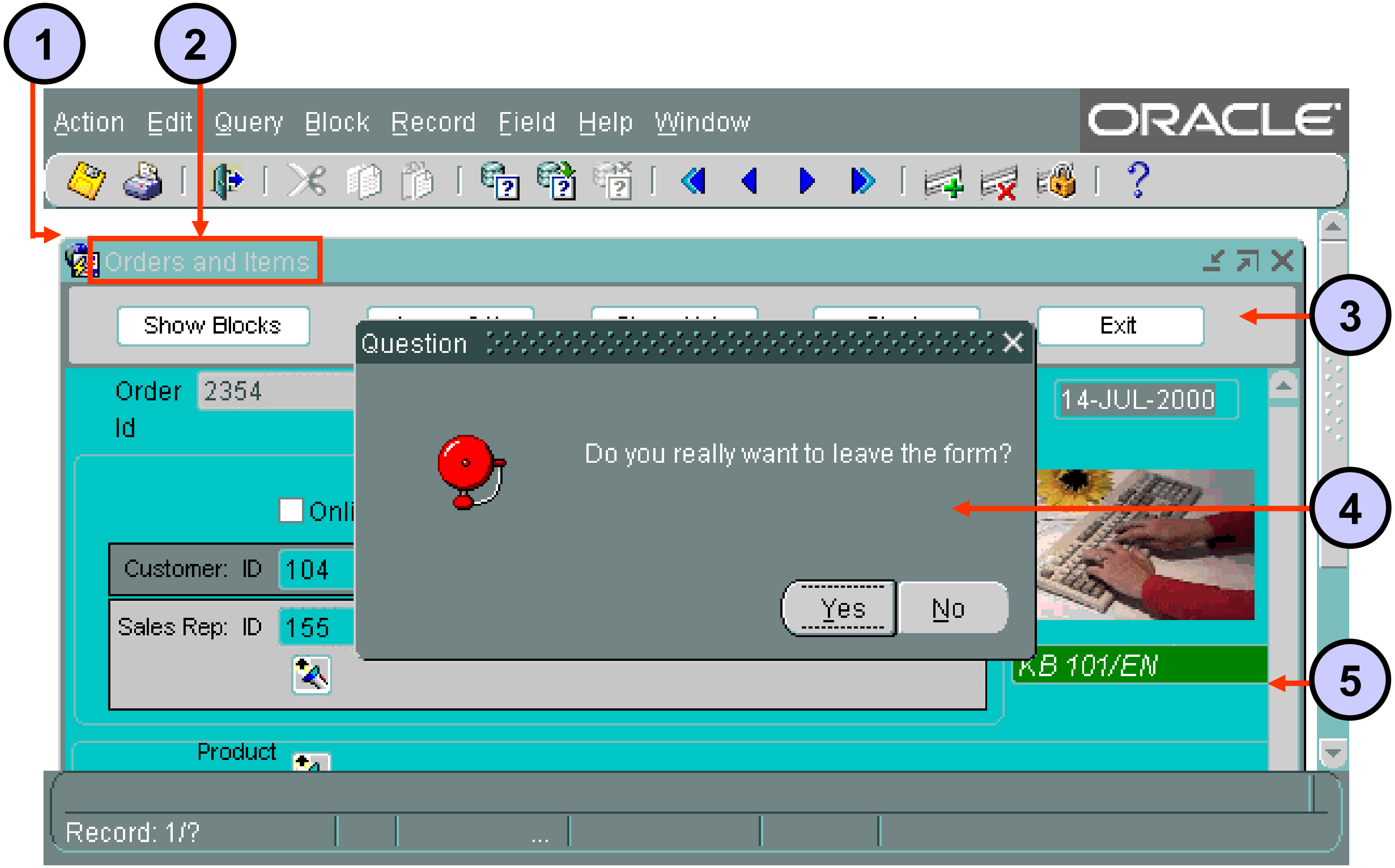


Window: WIN_INVENTORY

General	
Name	WIN_INVENTORY
Subclass Information	
Comments	
Help Book Topic	
Functional	
Title	Stock Levels
Primary Canvas	<Null>
Horizontal Toolbar Canvas	<Null>
Vertical Toolbar Canvas	<Null>
Window Style	Document
Modal	No
Hide on Exit	Yes
Close Allowed	Yes
Move Allowed	Yes
Resize Allowed	Yes
Maximize Allowed	Yes

Name

Setting Window Properties



Order Id 2354

14-JUL-2000

Do you really want to leave the form?

☐ Onl

Customer: ID 104

Sales Rep: ID 155

KB 101/EN

Product

GUI Hints

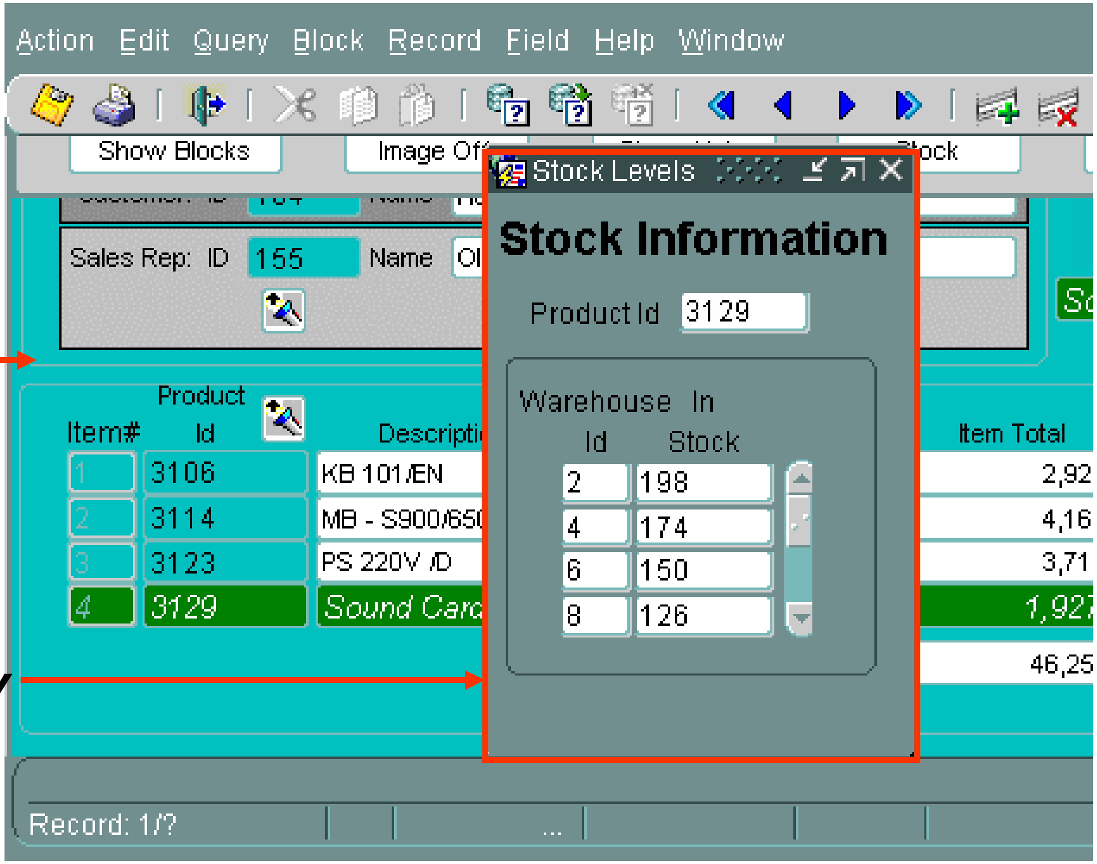
- GUI hints are recommendations to the window manager about window appearance and functionality.
- If the window manager supports a specific GUI Hint and its property is set to Yes, it will be used.
- Functional properties for GUI Hints:
  - Close Allowed
  - Move Allowed
  - Resize Allowed
  - Maximize Allowed
  - Minimize Allowed
  - Inherit Menu

Displaying a Form Module on Multiple Layouts

PROPERTIES:

Canvas CV_ORDER
Window: WIN_ORDERS

Canvas CV_INVENTORY
Window: WIN_INVENTORY



Sales Rep: ID 155 Name Ol

Item#	Product Id	Descripti	Item Total
1	3106	KB 101/EN	2,92
2	3114	MB - S900/650	4,16
3	3123	PS 220V /D	3,71
4	3129	Sound Card	1,92
			46,25

Stock Information

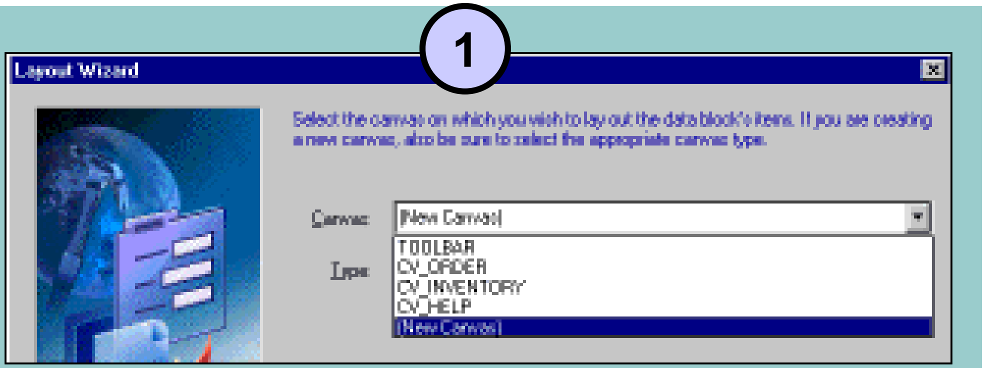
Product Id 3129

Warehouse Id	In Stock
2	198
4	174
6	150
8	126

Record: 1/? ...

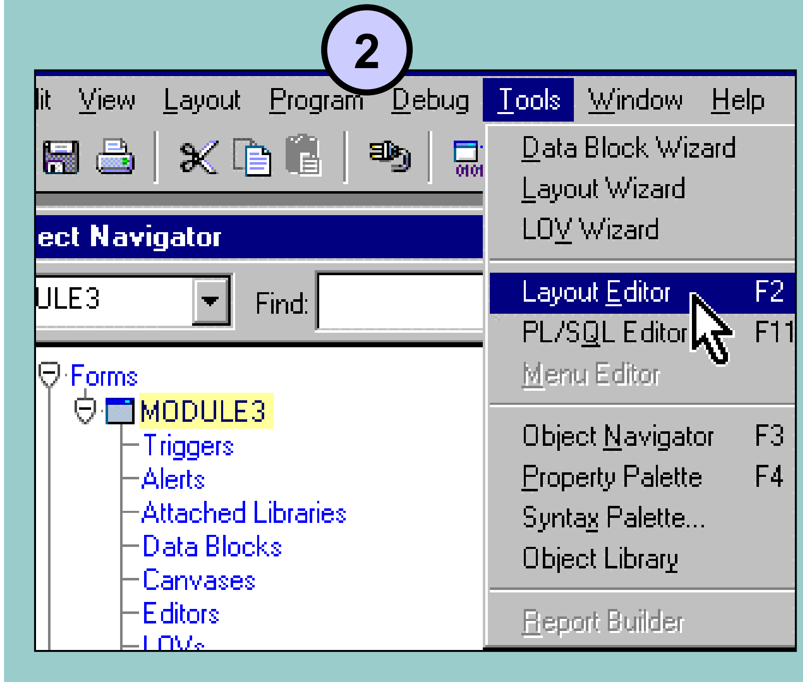
Creating a New Content Canvas

- **Implicitly:**

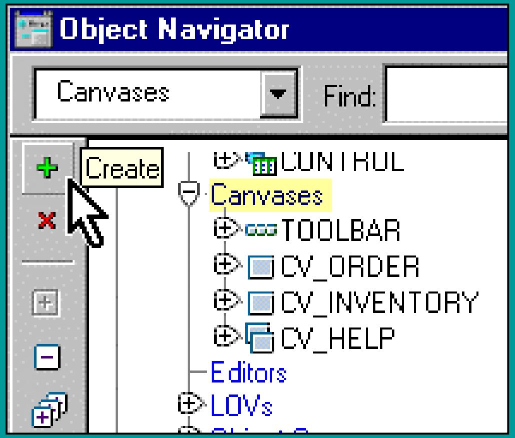


Select the canvas on which you wish to lay out the data block's items. If you are creating a new canvas, also be sure to select the appropriate canvas type.

Canvas: [New Canvas]

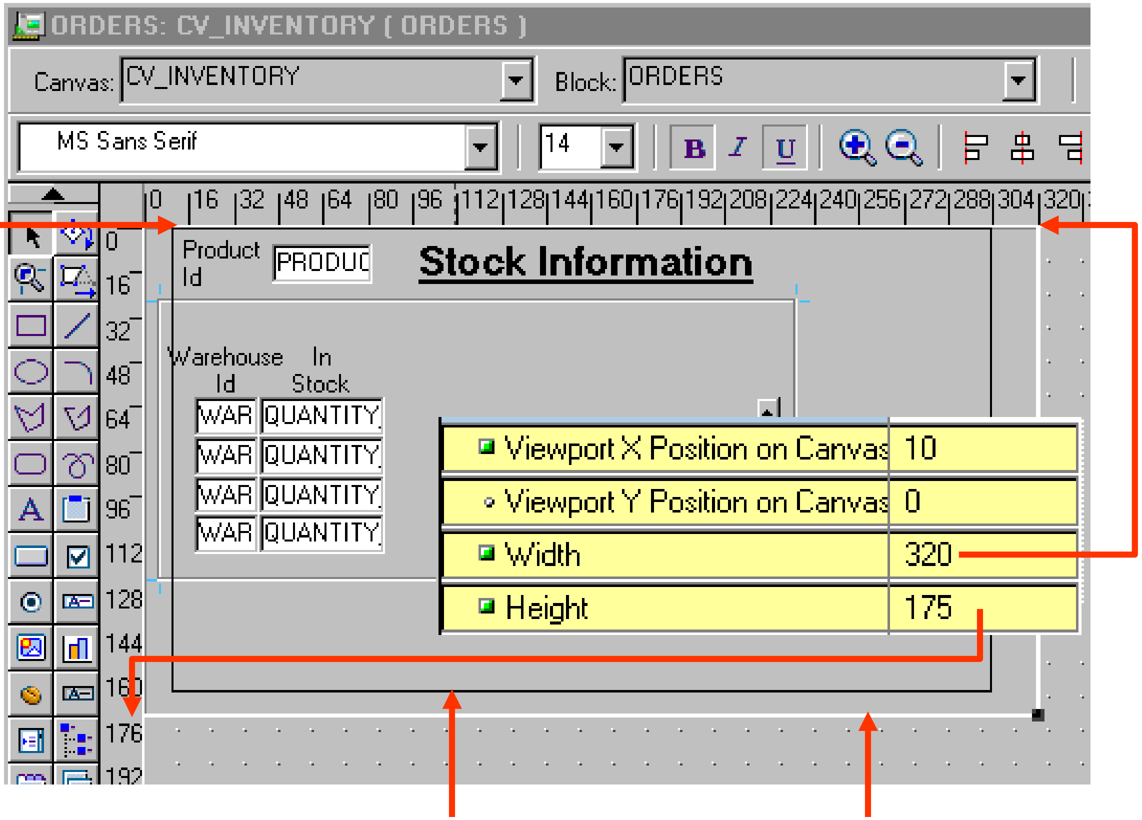


- **Explicitly:**



Setting Content Canvas Properties

Viewport X/Y Position on Canvas



Viewport X Position on Canvas	10
Viewport Y Position on Canvas	0
Width	320
Height	175

Viewport

Canvas

Summary

In this lesson, you should have learned that:

- Windows can display multiple content canvases, but can display only one canvas at a time
- Content canvases are displayed only in the window to which they are assigned
- You must assign at least one content canvas to each window in your application
- You create windows in the Object Navigator; one is created by default with each new module
- You create canvases in the Object Navigator, by using the Layout Wizard, or by invoking the Layout Editor in a module without a canvas
- You can display a multiple layouts by assigning canvases to different windows.

Practice 11 Overview

This practice covers the following topics:

- Changing a window size, position, name, and title
- Creating a new window
- Displaying data block contents in the new window

12

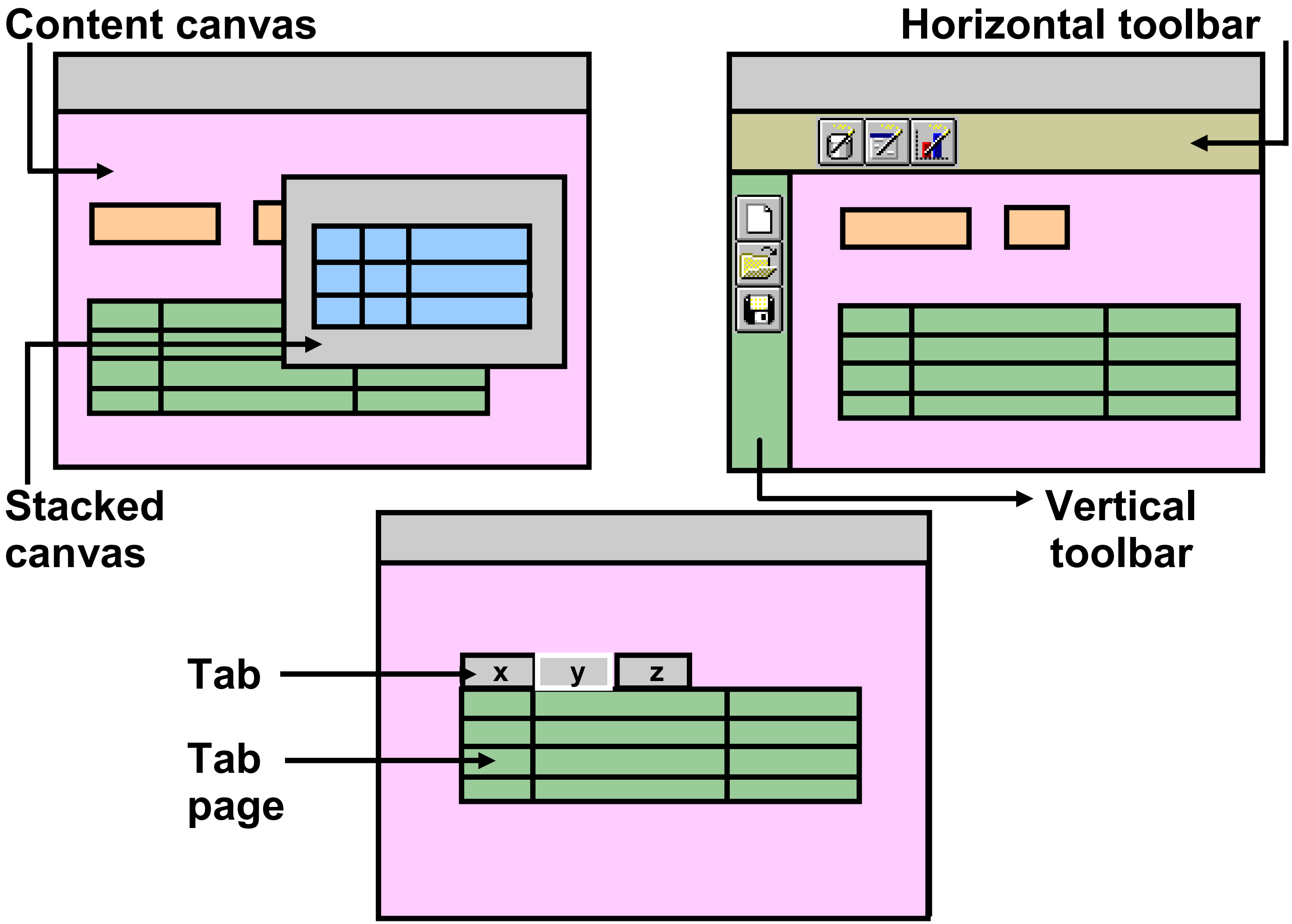
Working with Other Canvas Types

Objectives

After completing this lesson, you should be able to do the following:

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

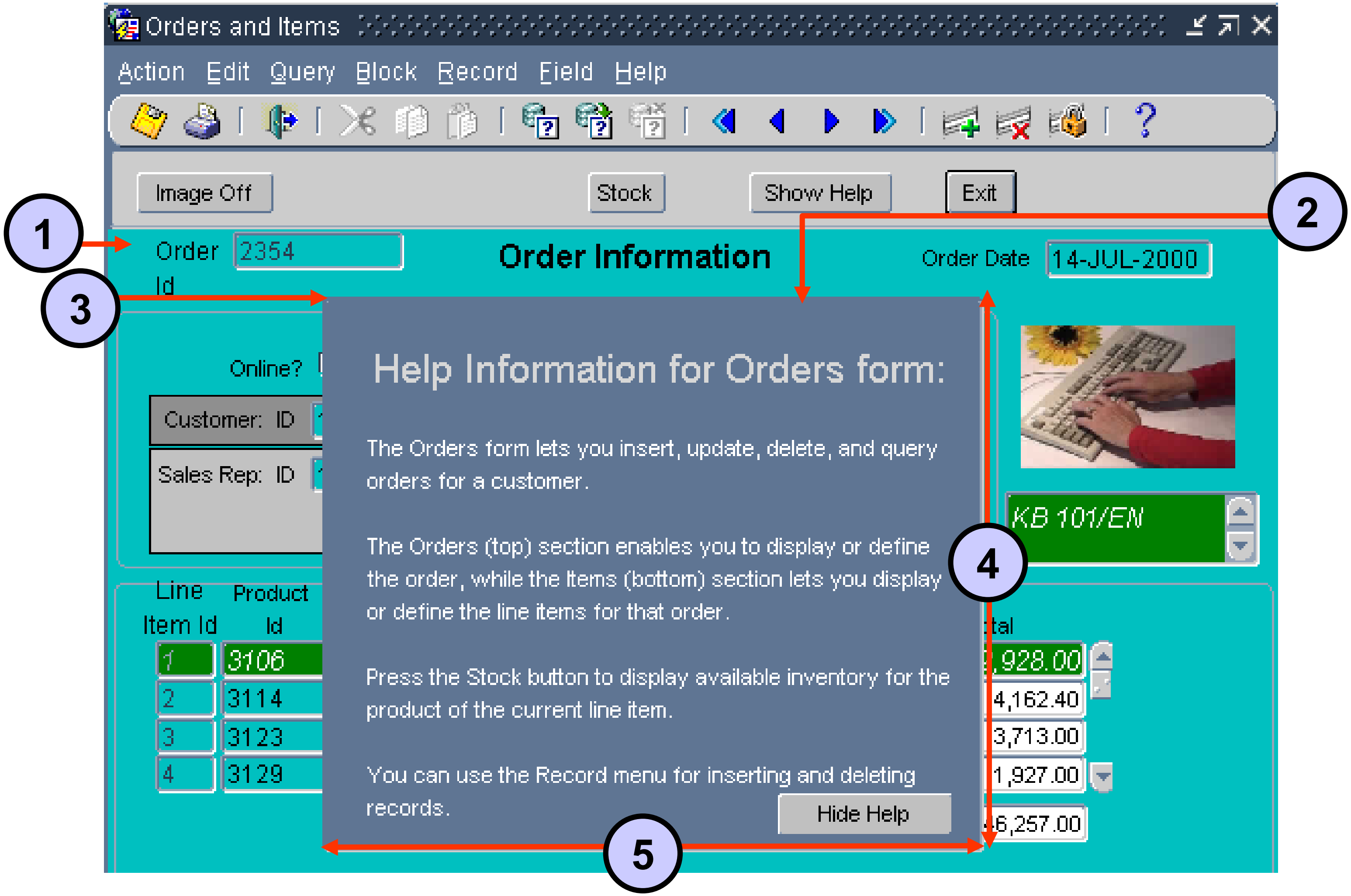
Overview of Canvas Types



The Stacked Canvas

- Displayed on top of a content canvas
- Shares a window with a content canvas
- Size:
  - Usually smaller than the content canvas in the same window
  - Determined by viewport size
- Created in:
  - Layout Editor
  - Object Navigator

The Stacked Canvas



Order Id 2354

Order Date 14-JUL-2000

Online?

Customer: ID

Sales Rep: ID

Line Item Id	Product Id	tal
1	3106	,928.00
2	3114	4,162.40
3	3123	3,713.00
4	3129	1,927.00
		6,257.00

Help Information for Orders form:

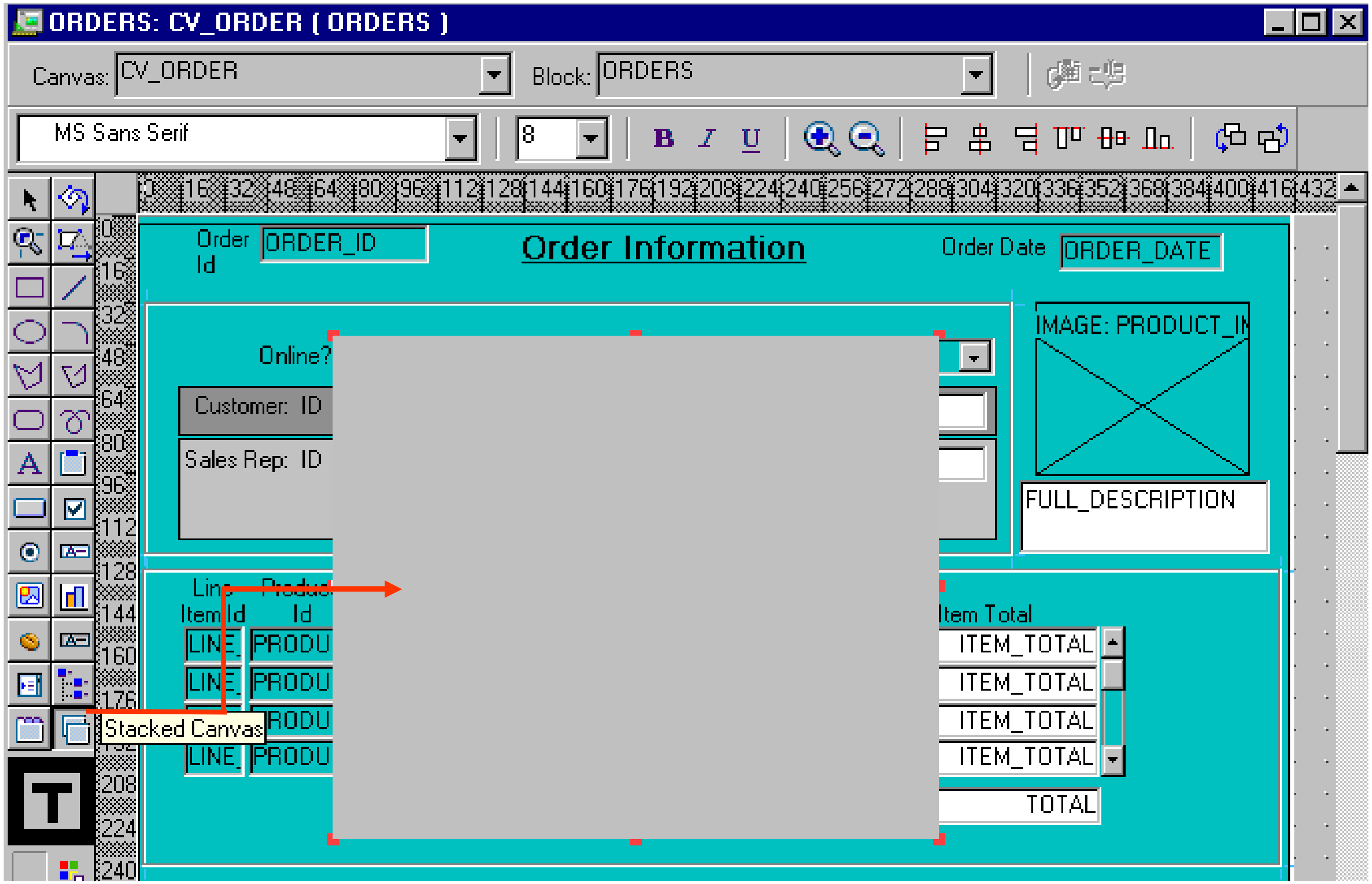
The Orders form lets you insert, update, delete, and query orders for a customer.

The Orders (top) section enables you to display or define the order, while the Items (bottom) section lets you display or define the line items for that order.

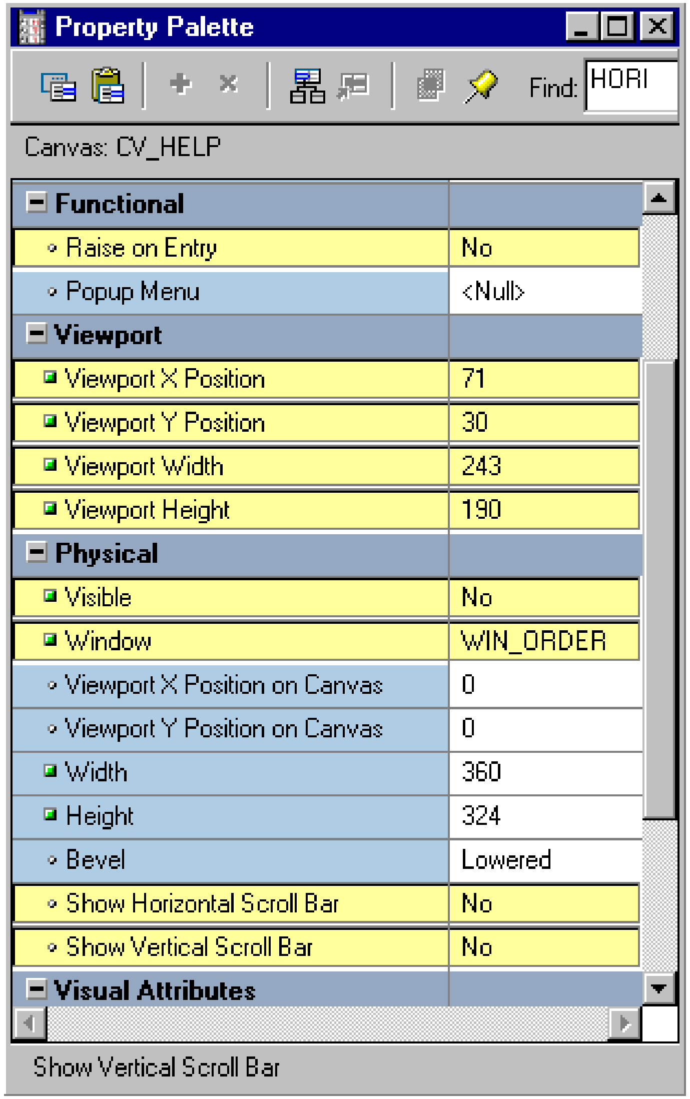
Press the Stock button to display available inventory for the product of the current line item.

You can use the Record menu for inserting and deleting records.

Creating a Stacked Canvas



Setting Stacked Canvas Properties



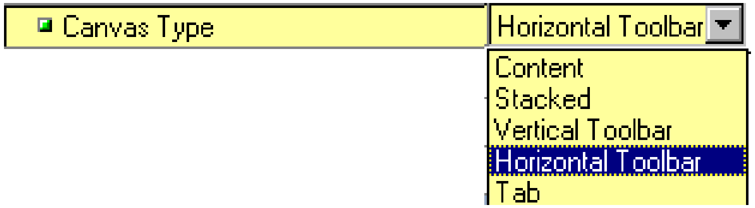
Canvas: CV_HELP

Functional	
Raise on Entry	No
Popup Menu	<Null>
Viewport	
Viewport X Position	71
Viewport Y Position	30
Viewport Width	243
Viewport Height	190
Physical	
Visible	No
Window	WIN_ORDER
Viewport X Position on Canvas	0
Viewport Y Position on Canvas	0
Width	360
Height	324
Bevel	Lowered
Show Horizontal Scroll Bar	No
Show Vertical Scroll Bar	No
Visual Attributes	

Show Vertical Scroll Bar

The Toolbar Canvas

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



Canvas Type: Horizontal Toolbar

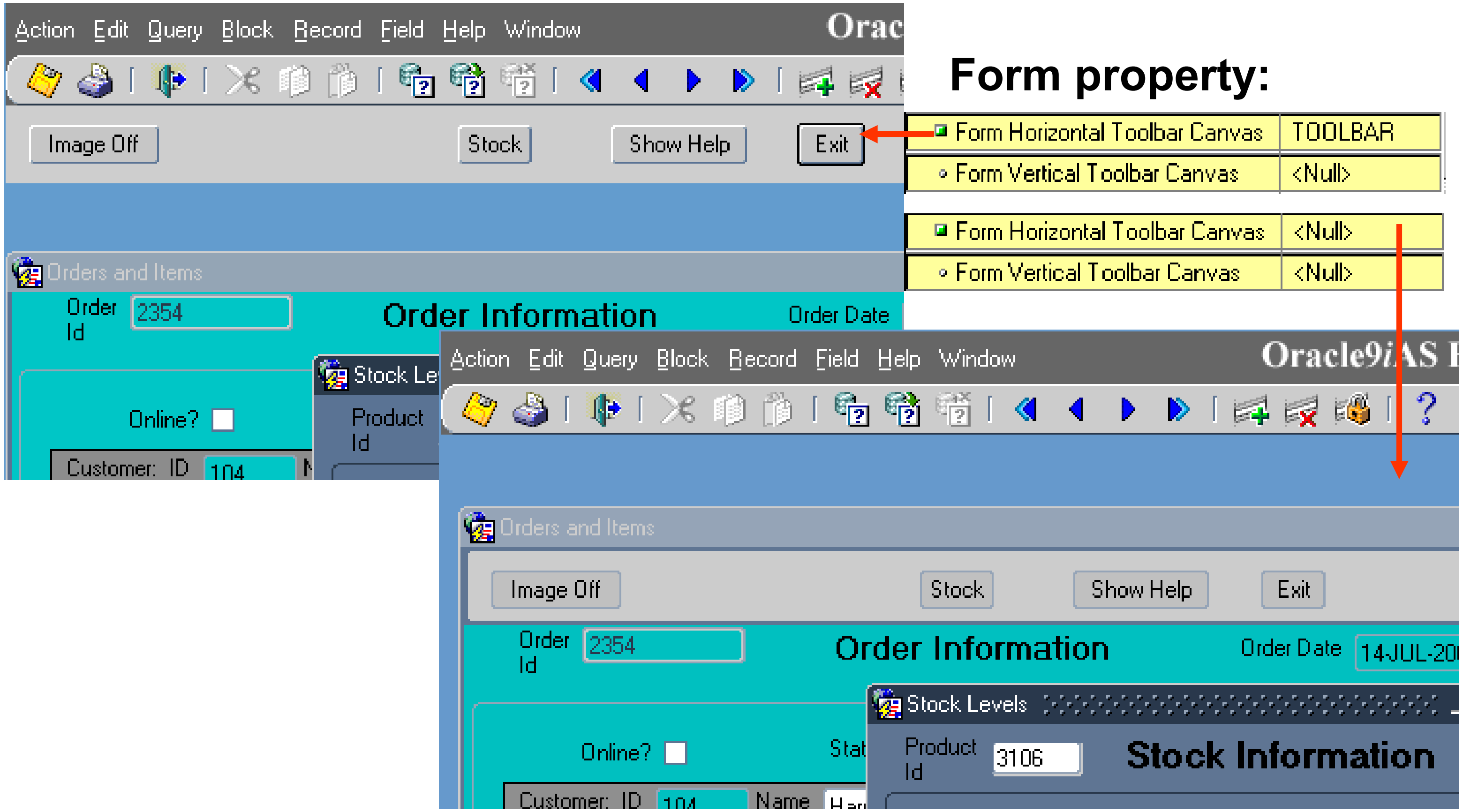
The MDI Toolbar

Runtime parameter:

`otherparams=useSDI=no`

Window property:

■ Horizontal Toolbar Canvas	TOOLBAR
∘ Vertical Toolbar Canvas	<Null>



Form property:

■ Form Horizontal Toolbar Canvas	TOOLBAR
∘ Form Vertical Toolbar Canvas	<Null>

■ Form Horizontal Toolbar Canvas	<Null>
∘ Form Vertical Toolbar Canvas	<Null>

Order Id: 2354

Order Date: 14-JUL-20

Online? □

Customer: ID 104

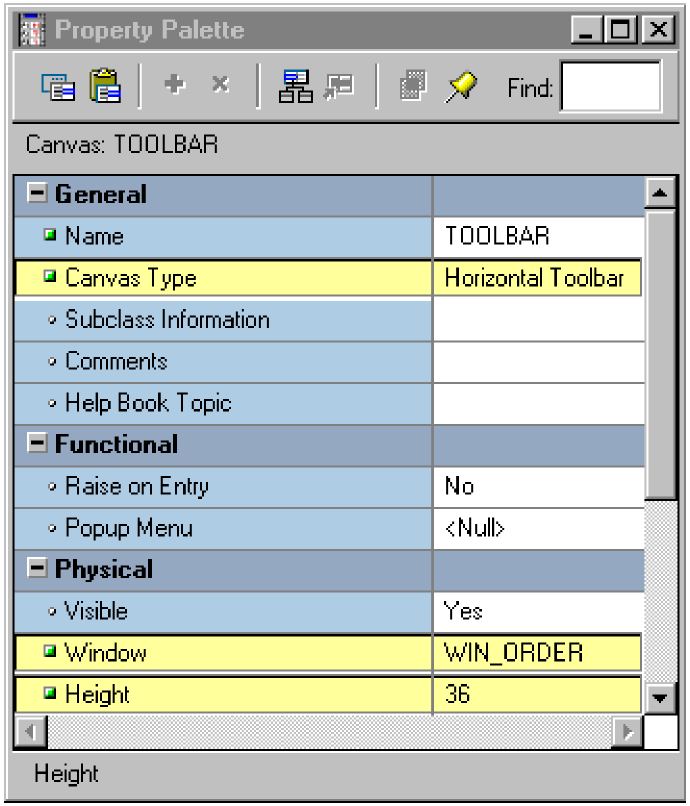
Product Id: 3106

Creating a Toolbar Canvas

1. Create:
   - Click Create in Object Navigator
   - Change Canvas Type
   - Set other properties as required
2. Add functionality
3. Resize the canvas (not the view)
4. Assign to window and/or form

Setting Toolbar Properties

- Canvas properties:
  - Canvas Type
  - Window
  - Width or Height
- Window properties:
  - Horizontal Toolbar Canvas
  - Vertical Toolbar Canvas
- Form Module properties:
  - Form Horizontal Toolbar Canvas
  - Form Vertical Toolbar Canvas

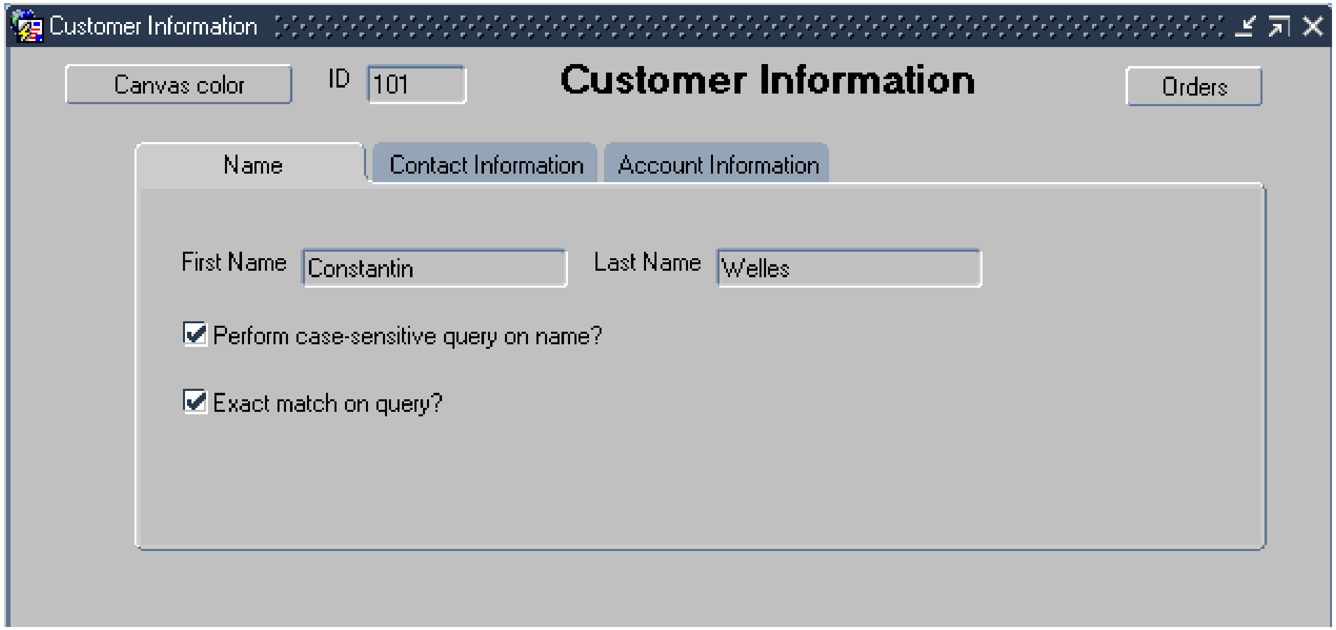


Canvas: TOOLBAR

General	
Name	TOOLBAR
Canvas Type	Horizontal Toolbar
Subclass Information	
Comments	
Help Book Topic	
Functional	
Raise on Entry	No
Popup Menu	<Null>
Physical	
Visible	Yes
Window	WIN_ORDER
Height	36

Height

The Tab Canvas



Customer Information

ID 101

First Name Constantin

Last Name Welles

☑ Perform case-sensitive query on name?

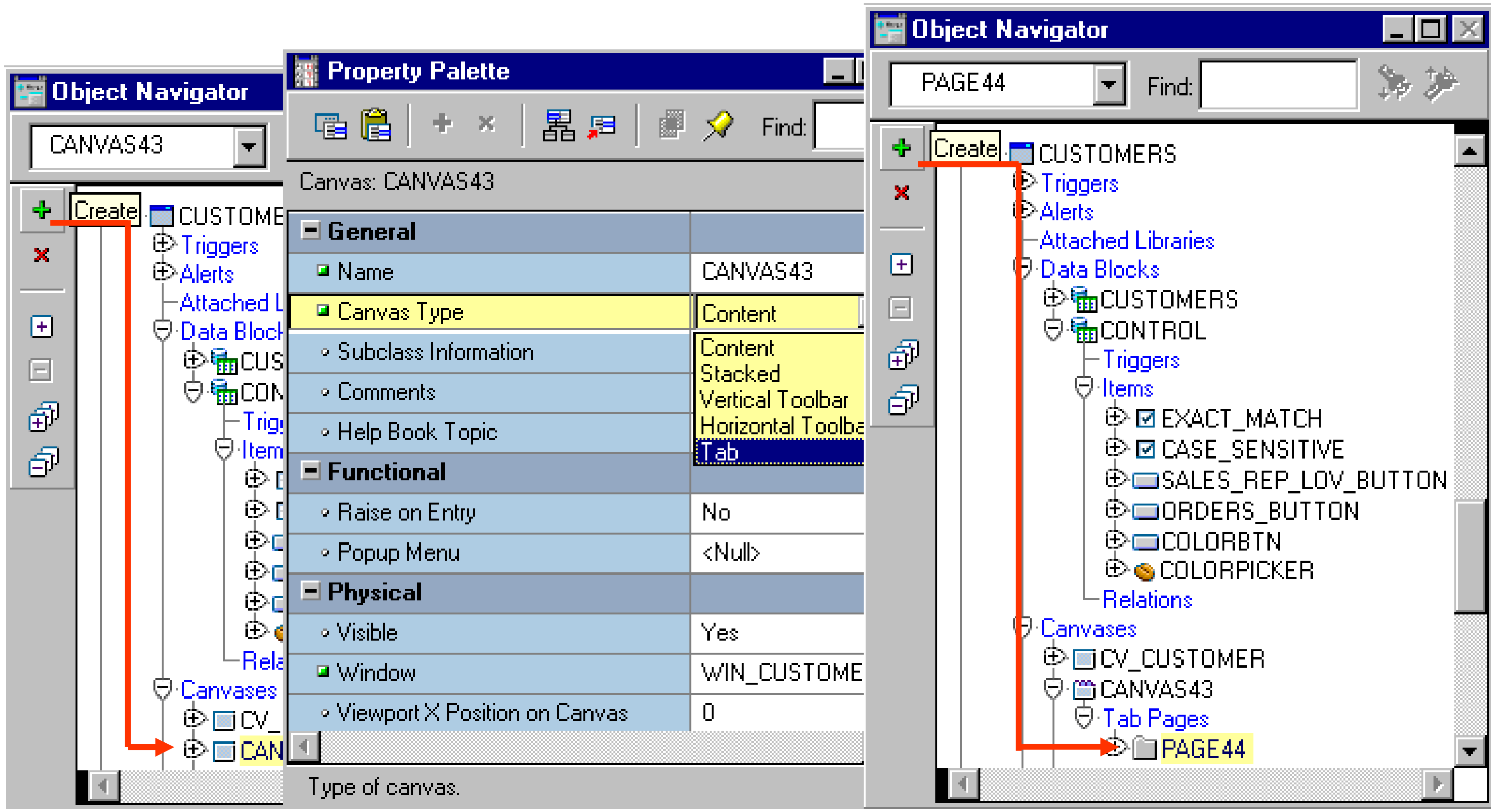
☑ Exact match on query?

- Enables you to organize and display related information on separate tabs
- Consists of one or more tab pages
- Provides easy access to data

Creating a Tab Canvas

- Create in:
  - Object Navigator
  - Layout Editor
- Define tab pages
- Place items on tab pages

Creating a Tab Canvas in the Object Navigator



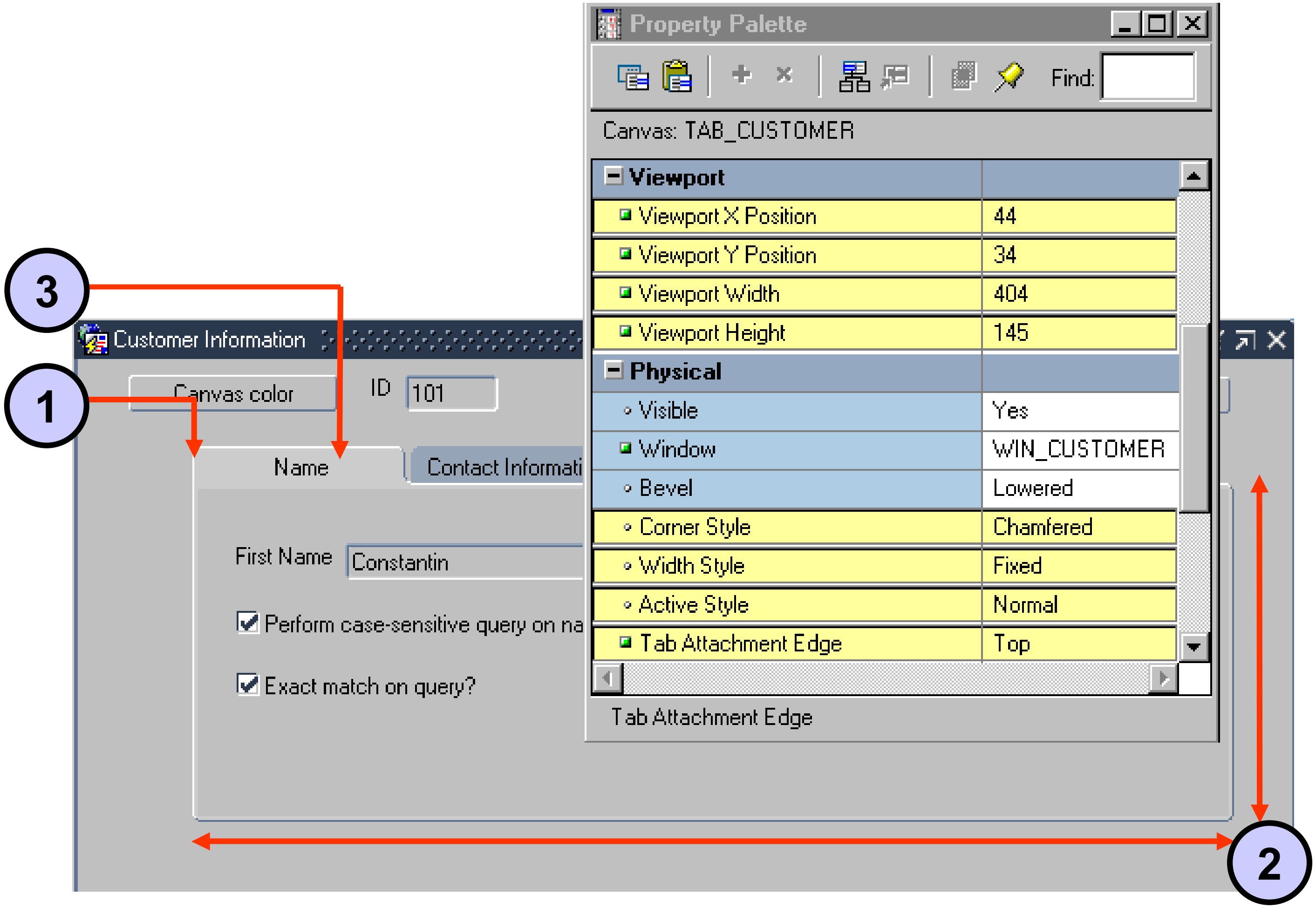
Canvas: CANVAS43

General	
Name	CANVAS43
Canvas Type	Content
Subclass Information	Content Stacked Vertical Toolbar Horizontal Toolba Tab
Comments	
Help Book Topic	
Functional	
Raise on Entry	No
Popup Menu	<Null>
Physical	
Visible	Yes
Window	WIN_CUSTOME
Viewport X Position on Canvas	0

Type of canvas.

Create new Canvas **Set Canvas Type** **Create Tab Pages**

Setting Tab Canvas, Tab Page, and Item Properties



Property Palette

Find:

Canvas: TAB_CUSTOMER

⊟ **Viewport**	
Viewport X Position	44
Viewport Y Position	34
Viewport Width	404
Viewport Height	145
⊟ **Physical**	
Visible	Yes
Window	WIN_CUSTOMER
Bevel	Lowered
Corner Style	Chamfered
Width Style	Fixed
Active Style	Normal
Tab Attachment Edge	Top

Tab Attachment Edge

Customer Information

Canvas color

ID 101

Name | Contact Informati

First Name Constantin

☑ Perform case-sensitive query on na

☑ Exact match on query?

Placing Items on a Tab Canvas

- Place items on each tab page for user interaction.
- Set the item properties:
  - Canvas
  - Tab Page

Summary

In this lesson, you should have learned:

- Canvas types other than content canvases:
  - Stacked: Overlays and shares window with content canvas; use to create cascading or revealing effect within a single window, display additional information, display or hide information conditionally, or display context-sensitive help
  - Toolbar: Area that displays at the top or to the left of a content canvas; use to to hold buttons and other frequently used GUI elements with a standard look and feel across canvases displayed in the same window
  - Tab: Has multiple pages where you navigate using tabs; use to organize and display related information on different tabs

Summary

- You can create these in Object Navigator and change the canvas type, then set properties.
- You can create stacked or tab canvases with the appropriate tool in the Layout Editor.
- You can attach a Toolbar canvas to single window, or to entire form if using MDI.
- After creating a tab canvas, create tab pages and place related items on them.

Practice 12 Overview

This practice covers the following topics:

- Creating a toolbar canvas
- Creating a stacked canvas
- Creating a tab canvas
- Adding tab pages to the tab canvas

13

Introduction to Triggers

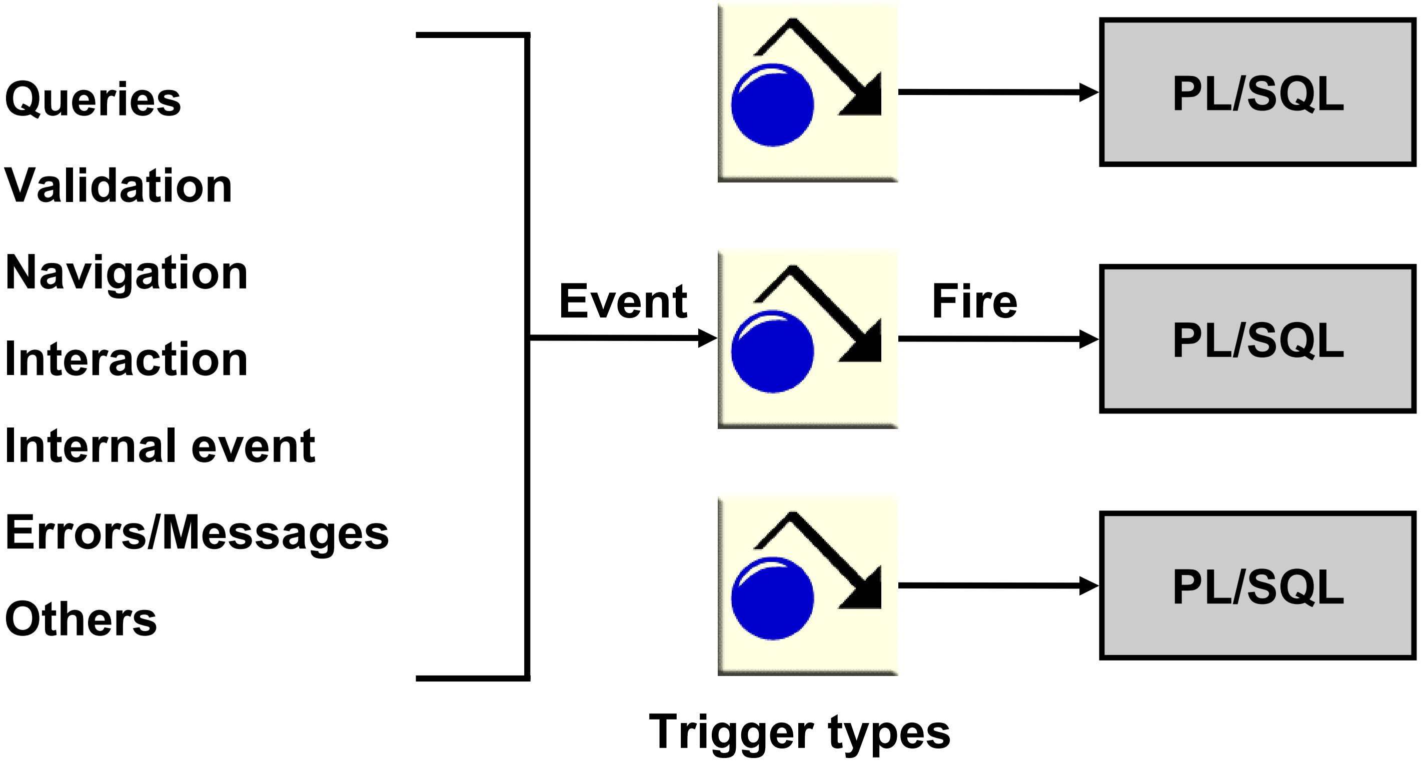
Objectives

After completing this lesson, you should be able to do the following:

- Define triggers
- Identify the different trigger categories
- Plan the type and scope of triggers in a form
- Describe the properties that affect the behavior of a trigger

ORACLE

Trigger Overview



Which trigger would you use to perform complex calculations after a user enters data into an item?

Grouping Triggers into Categories

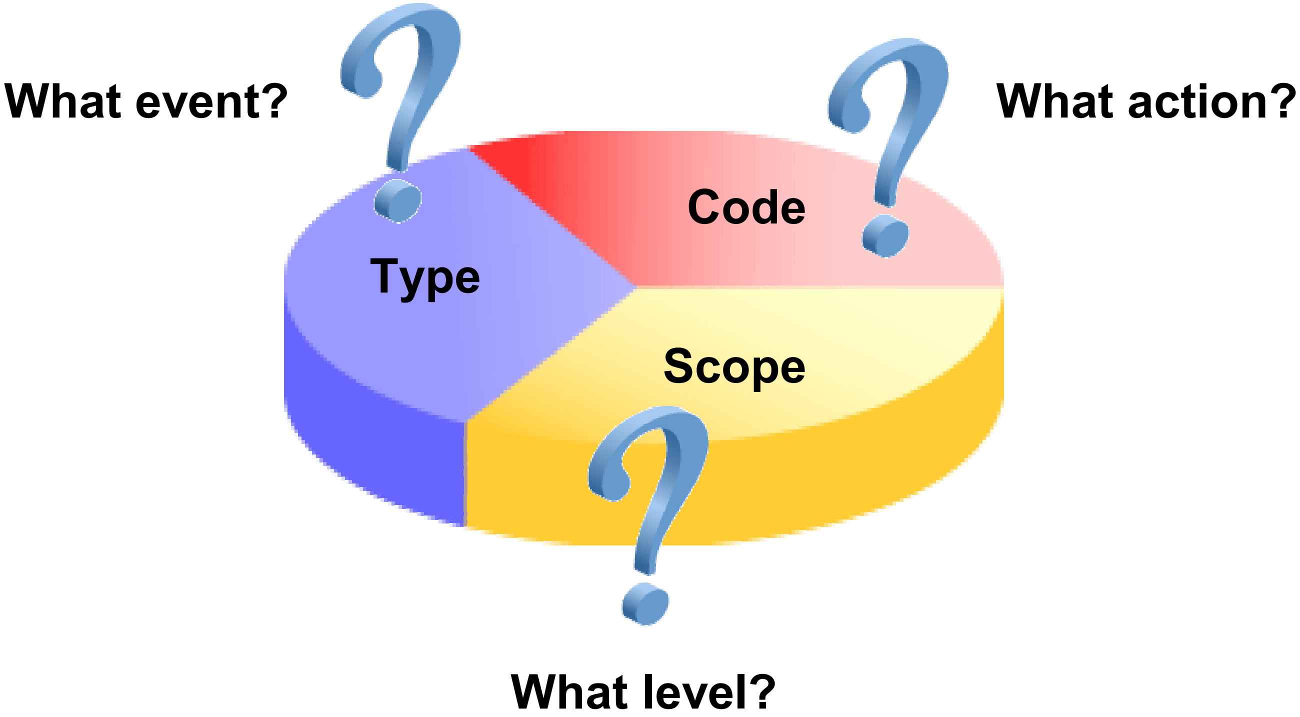
Triggers may be grouped into functional categories:

- Block processing triggers
- Interface event triggers
- Master-detail triggers
- Message handling triggers
- Navigational triggers
- Query-time triggers
- Transactional triggers
- Validation triggers

Triggers may be grouped into categories based on name:

- When-Event triggers
- On-Event triggers
- Pre-Event triggers
- Post-Event triggers
- Key triggers

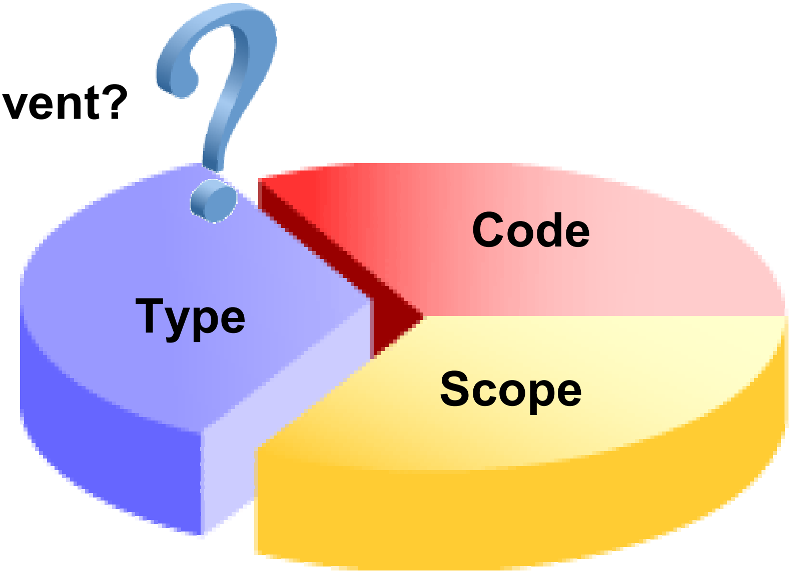
Defining Trigger Components



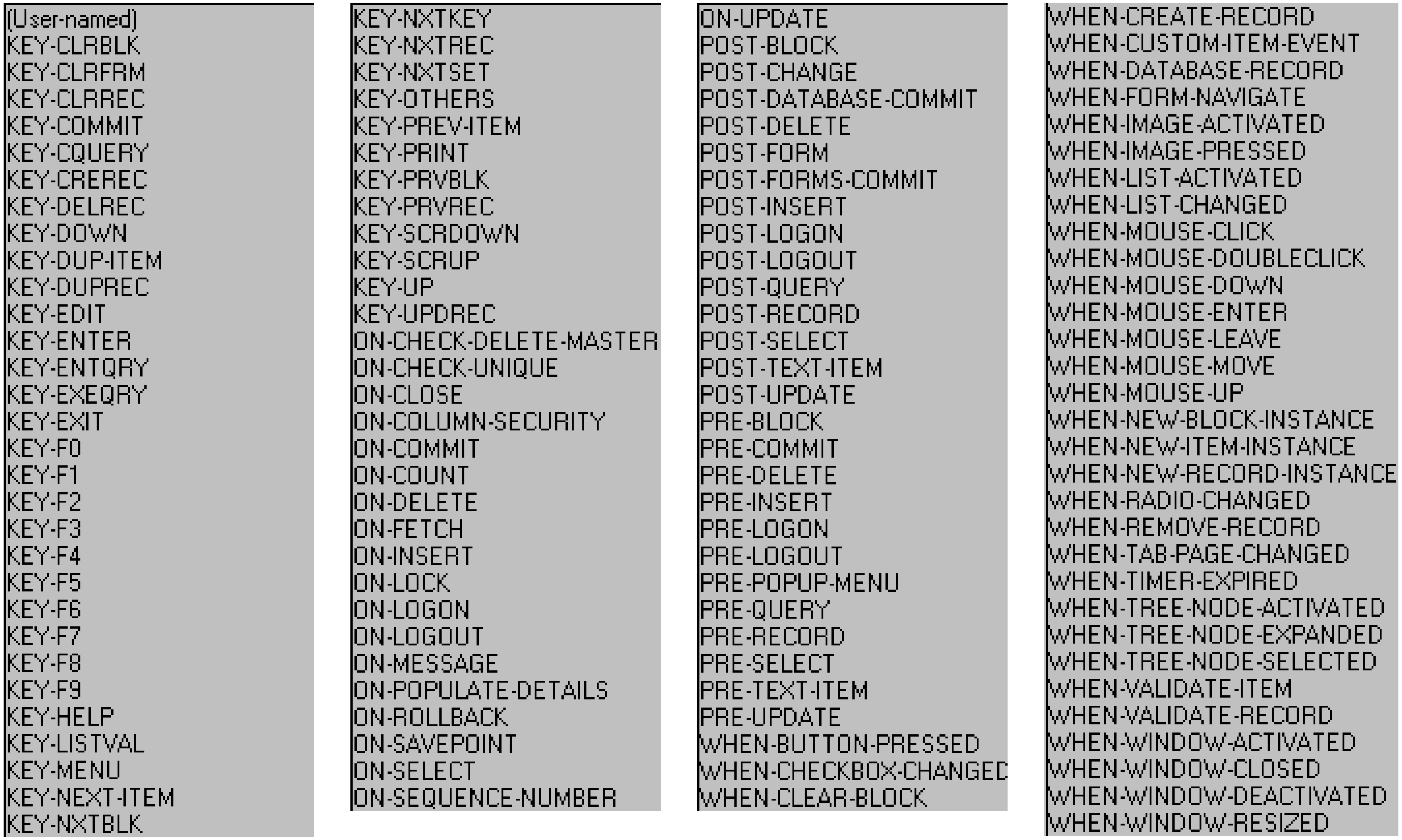
Trigger Type

- Pre-
- Post-
- When-
- On-
- Key-
- User-named

What event?



Trigger Type



(User-named)
KEY-CLRBLK
KEY-CLRFRM
KEY-CLRREC
KEY-COMMIT
KEY-CQUERY
KEY-CREREC
KEY-DELREC
KEY-DOWN
KEY-DUP-ITEM
KEY-DUPREC
KEY-EDIT
KEY-ENTER
KEY-ENTQRY
KEY-EXEQRY
KEY-EXIT
KEY-F0
KEY-F1
KEY-F2
KEY-F3
KEY-F4
KEY-F5
KEY-F6
KEY-F7
KEY-F8
KEY-F9
KEY-HELP
KEY-LISTVAL
KEY-MENU
KEY-NEXT-ITEM
KEY-NXTBLK

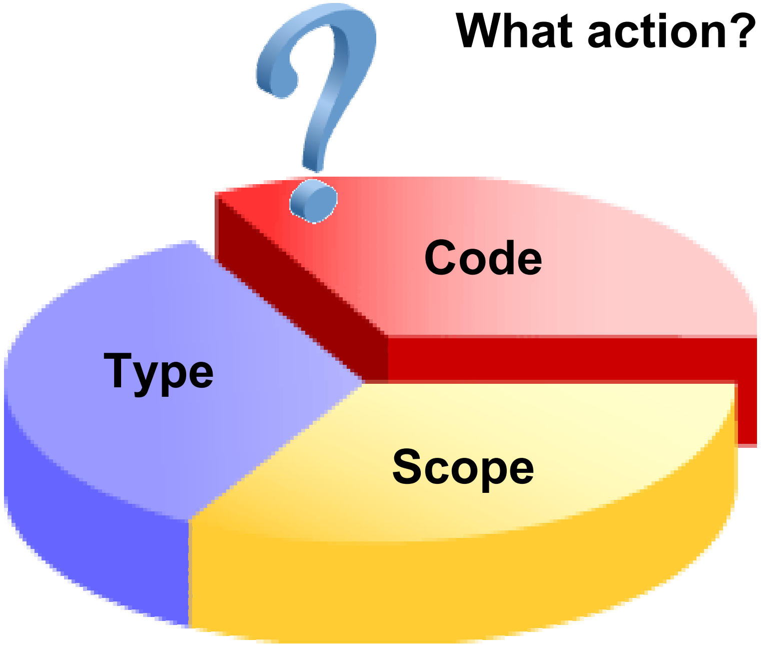
KEY-NXTKEY
KEY-NXTREC
KEY-NXTSET
KEY-OTHERS
KEY-PREV-ITEM
KEY-PRINT
KEY-PRVBLK
KEY-PRVREC
KEY-SCRDOWN
KEY-SCRUP
KEY-UP
KEY-UPDREC
ON-CHECK-DELETE-MASTER
ON-CHECK-UNIQUE
ON-CLOSE
ON-COLUMN-SECURITY
ON-COMMIT
ON-COUNT
ON-DELETE
ON-FETCH
ON-INSERT
ON-LOCK
ON-LOGON
ON-LOGOUT
ON-MESSAGE
ON-POPULATE-DETAILS
ON-ROLLBACK
ON-SAVEPOINT
ON-SELECT
ON-SEQUENCE-NUMBER

ON-UPDATE
POST-BLOCK
POST-CHANGE
POST-DATABASE-COMMIT
POST-DELETE
POST-FORM
POST-FORMS-COMMIT
POST-INSERT
POST-LOGON
POST-LOGOUT
POST-QUERY
POST-RECORD
POST-SELECT
POST-TEXT-ITEM
POST-UPDATE
PRE-BLOCK
PRE-COMMIT
PRE-DELETE
PRE-INSERT
PRE-LOGON
PRE-LOGOUT
PRE-POPUP-MENU
PRE-QUERY
PRE-RECORD
PRE-SELECT
PRE-TEXT-ITEM
PRE-UPDATE
WHEN-BUTTON-PRESSED
WHEN-CHECKBOX-CHANGED
WHEN-CLEAR-BLOCK

WHEN-CREATE-RECORD
WHEN-CUSTOM-ITEM-EVENT
WHEN-DATABASE-RECORD
WHEN-FORM-NAVIGATE
WHEN-IMAGE-ACTIVATED
WHEN-IMAGE-PRESSED
WHEN-LIST-ACTIVATED
WHEN-LIST-CHANGED
WHEN-MOUSE-CLICK
WHEN-MOUSE-DOUBLECLICK
WHEN-MOUSE-DOWN
WHEN-MOUSE-ENTER
WHEN-MOUSE-LEAVE
WHEN-MOUSE-MOVE
WHEN-MOUSE-UP
WHEN-NEW-BLOCK-INSTANCE
WHEN-NEW-ITEM-INSTANCE
WHEN-NEW-RECORD-INSTANCE
WHEN-RADIO-CHANGED
WHEN-REMOVE-RECORD
WHEN-TAB-PAGE-CHANGED
WHEN-TIMER-EXPIRED
WHEN-TREE-NODE-ACTIVATED
WHEN-TREE-NODE-EXPANDED
WHEN-TREE-NODE-SELECTED
WHEN-VALIDATE-ITEM
WHEN-VALIDATE-RECORD
WHEN-WINDOW-ACTIVATED
WHEN-WINDOW-CLOSED
WHEN-WINDOW-DEACTIVATED
WHEN-WINDOW-RESIZED

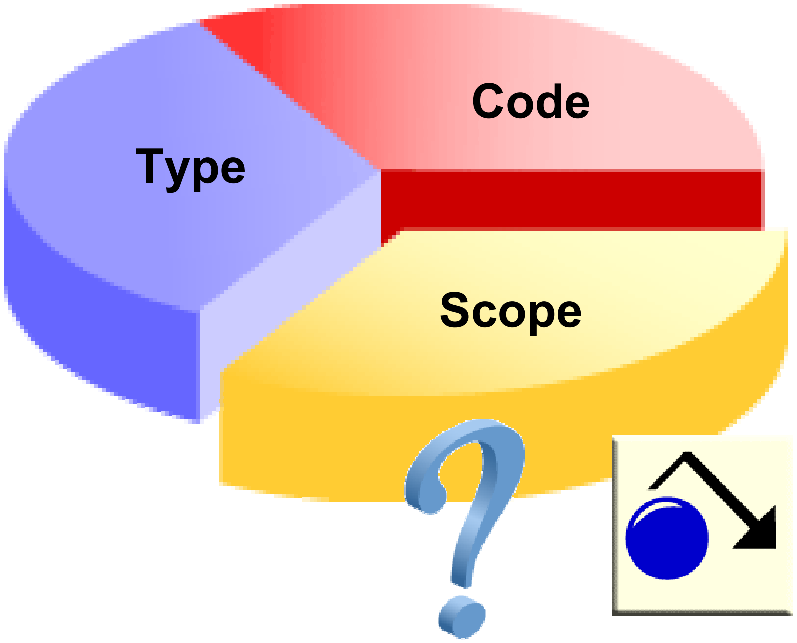
Forms Builder Trigger Types

Trigger Code



- Statements
- PL/SQL
- User subprograms
- Built-in subprograms

Trigger Scope

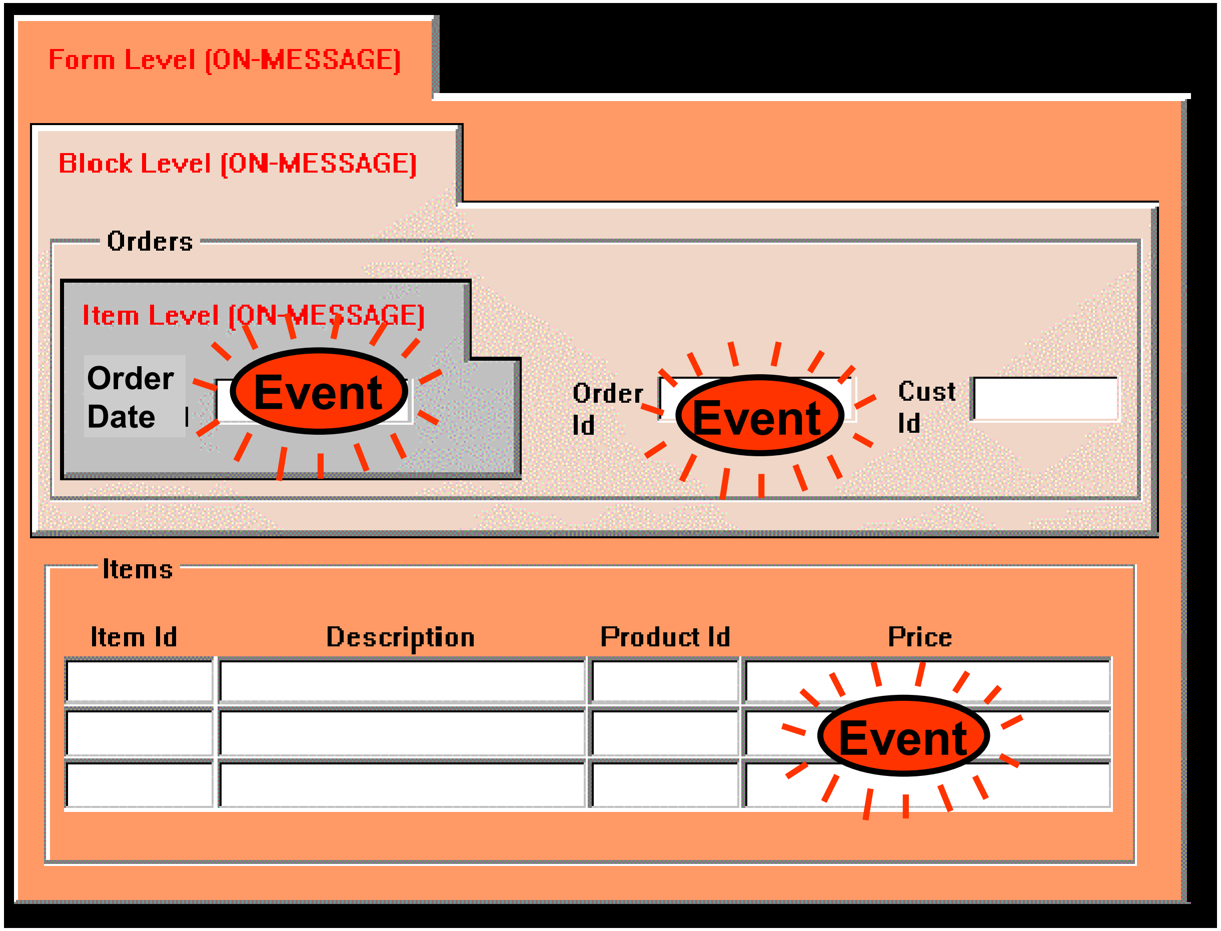


What level?

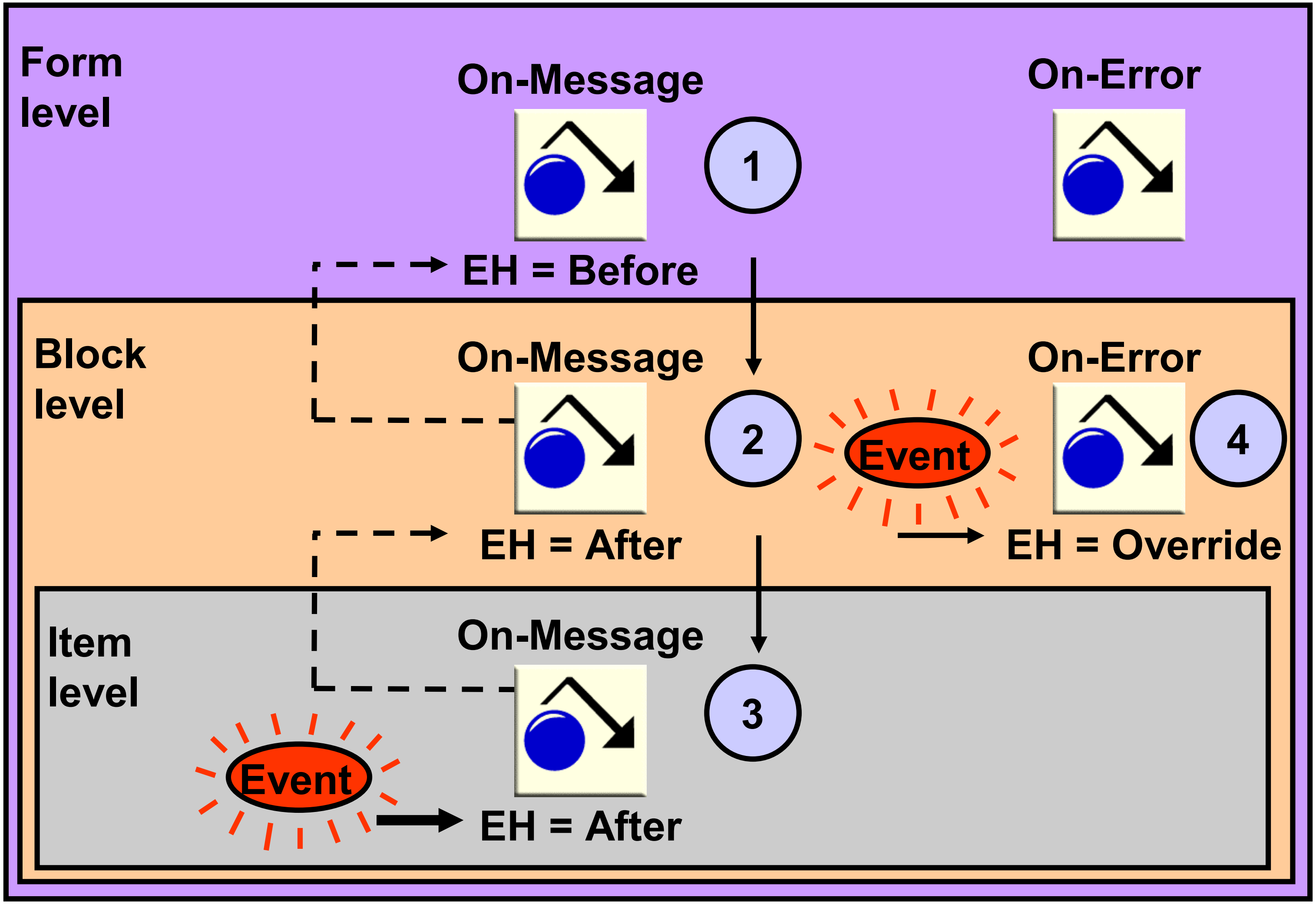
Levels

- Form
- Block
- Item

Trigger Scope



Specifying Execution Hierarchy



Summary

In this lesson, you should have learned that:

- Triggers are event-activated program units
- You can categorize triggers based on function or name to help you understand how they work
- Trigger components are:
  - Type: Defines the event that fires the trigger
  - Code: The actions a trigger performs
  - Scope: Specifies the level (form, block, or item) at which the trigger is defined
- The Execution Hierarchy trigger property alters the firing sequence of a trigger

14

Producing Triggers

Objectives

After completing this lesson, you should be able to do the following:

- Write trigger code
- Explain the use of built-in subprograms in Forms applications
- Describe the When-Button-Pressed trigger
- Describe the When-Window-Closed trigger

Creating Triggers in Forms Builder

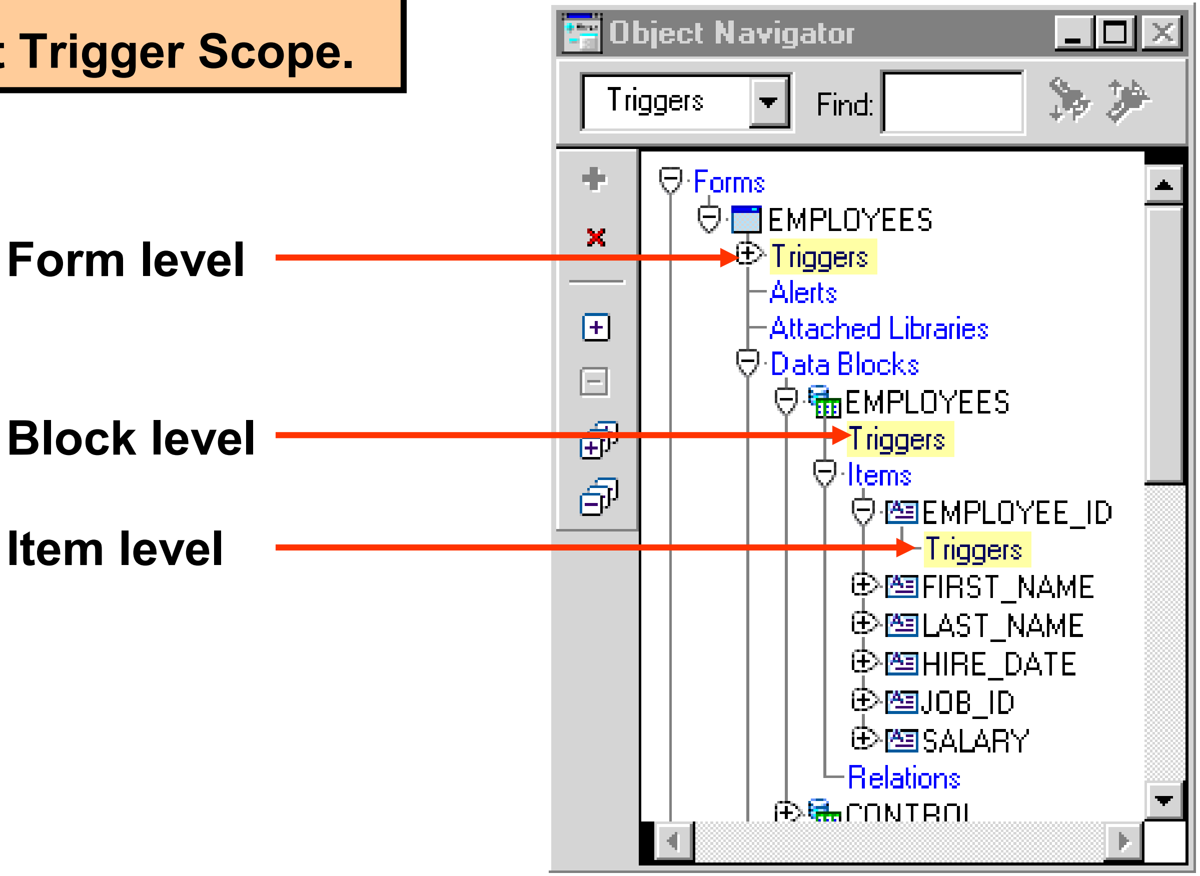
To produce a trigger:

1. Select a scope in the Object Navigator.
2. Create a trigger and select a name from the Trigger LOV, or use the SmartTriggers menu option.
3. Define code in the PL/SQL Editor.
4. Compile.

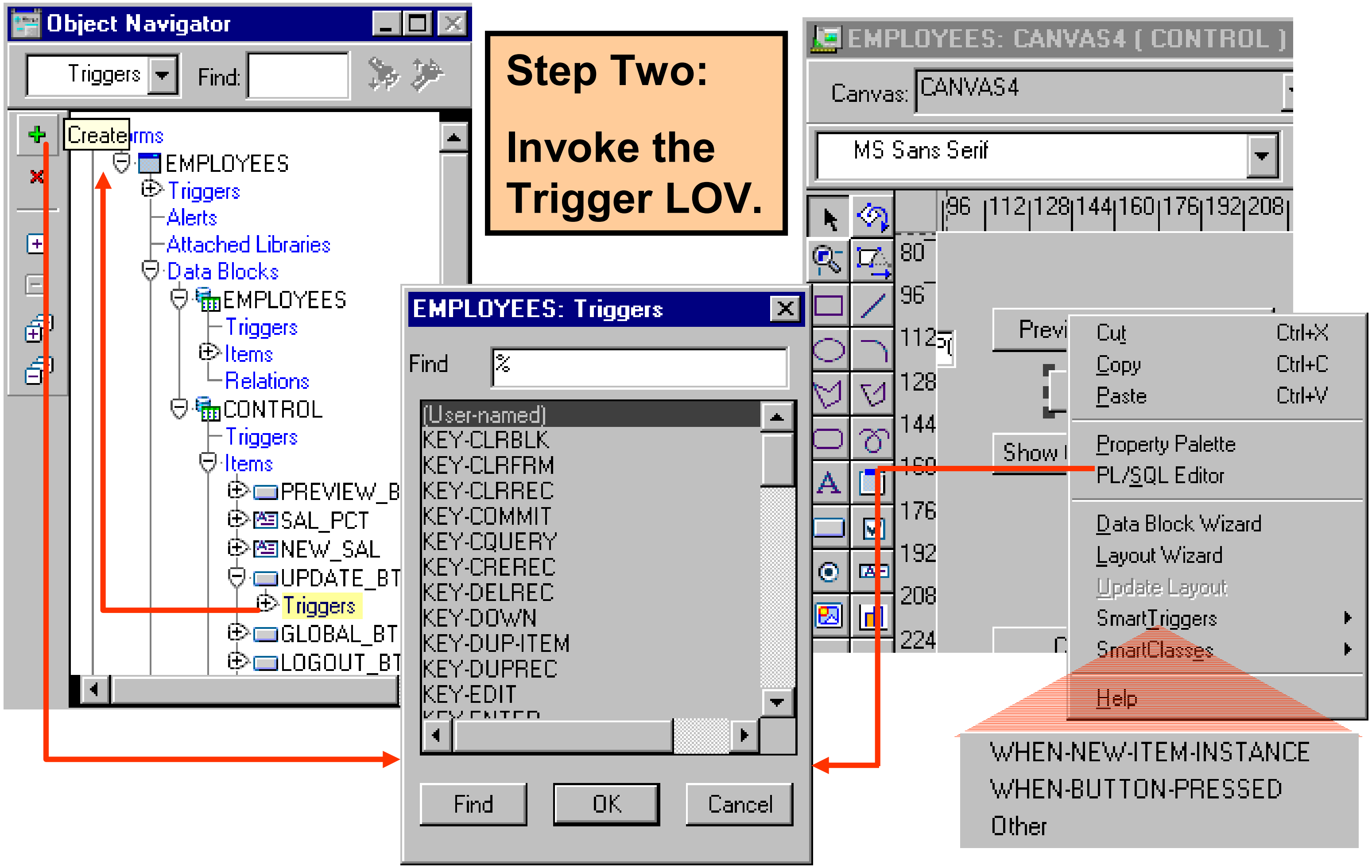
Creating a Trigger

Step One:

Select Trigger Scope.



Creating a Trigger



Step Two: Invoke the Trigger LOV.

EMPLOYEES: Triggers

Find %

- (User-named)
- KEY-CLRBLK
- KEY-CLRFRM
- KEY-CLRREC
- KEY-COMMIT
- KEY-CQUERY
- KEY-CREREC
- KEY-DELREC
- KEY-DOWN
- KEY-DUP-ITEM
- KEY-DUPREC
- KEY-EDIT
- KEY-ENTER

- WHEN-NEW-ITEM-INSTANCE
- WHEN-BUTTON-PRESSED
- Other

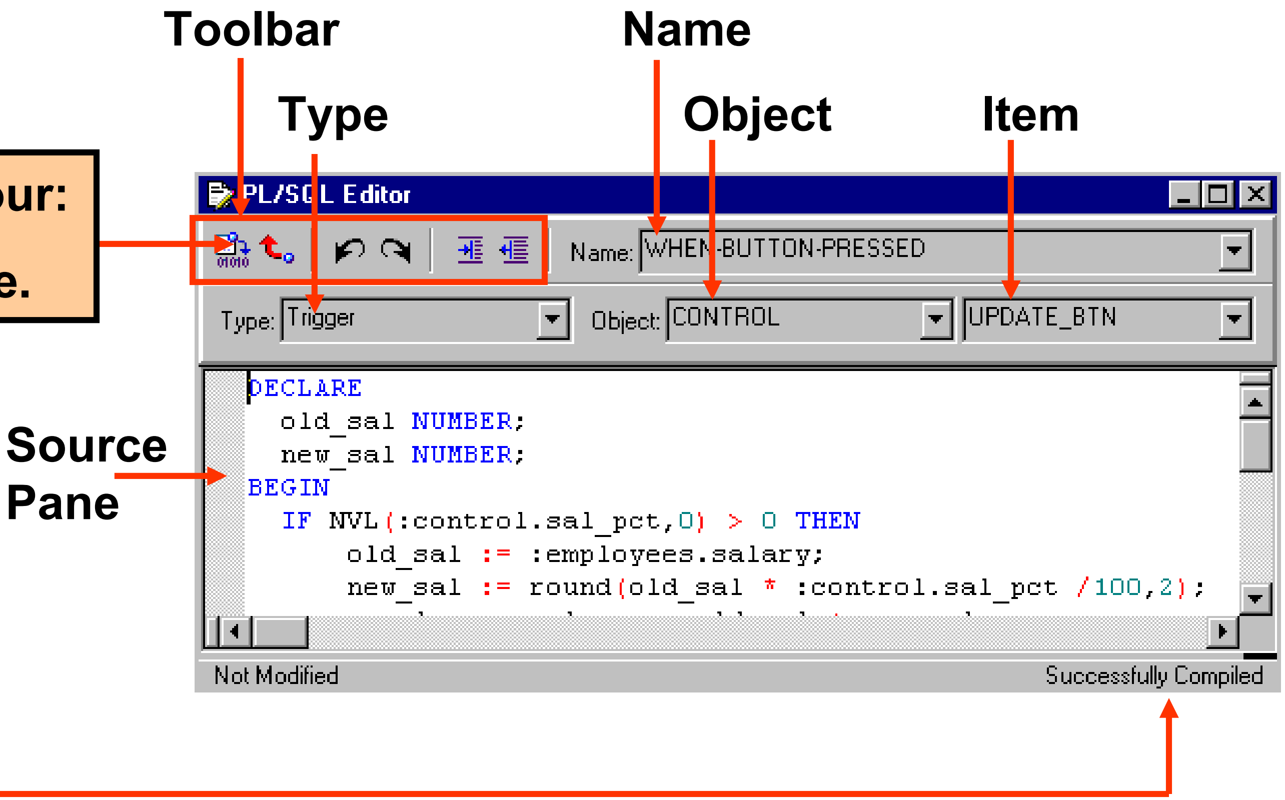
Creating a Trigger

Step Three:

Use the PL/SQL Editor to define the trigger code.

Step Four:

Compile.



Toolbar

Name

Type

Object

Item

Source Pane

Name: WHEN-BUTTON-PRESSED

Type: Trigger

Object: CONTROL

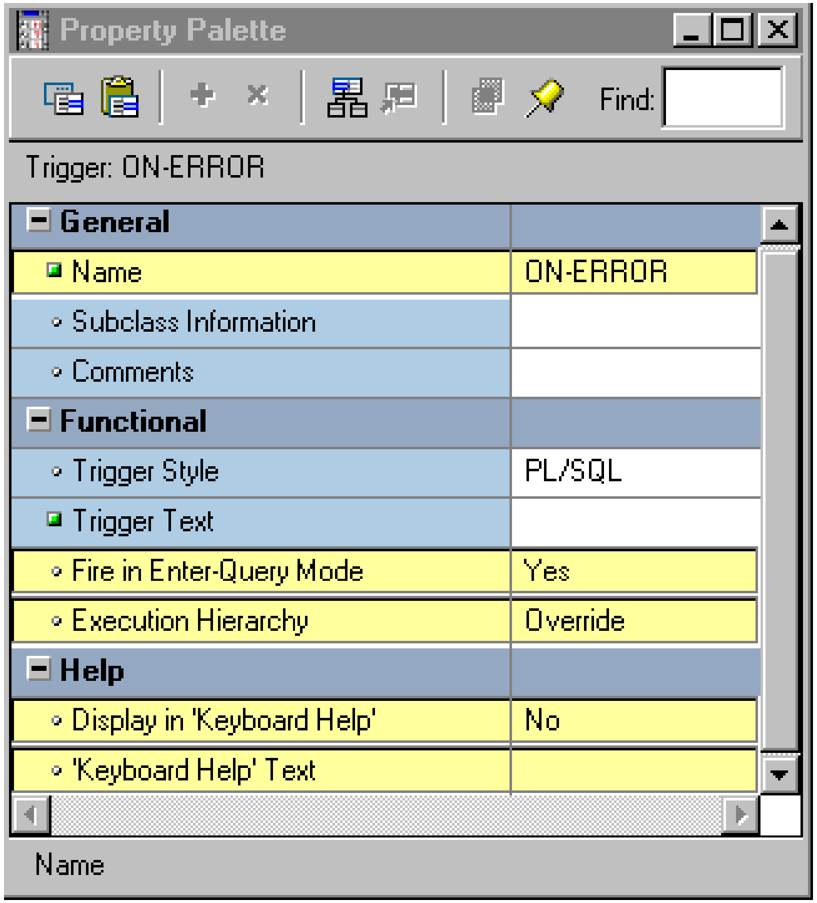
UPDATE_BTN

```
DECLARE
  old_sal NUMBER;
  new_sal NUMBER;
BEGIN
  IF NVL(:control.sal_pct,0) > 0 THEN
      old_sal := :employees.salary;
      new_sal := round(old_sal * :control.sal_pct /100,2);
```

Not Modified

Successfully Compiled

Setting Trigger Properties

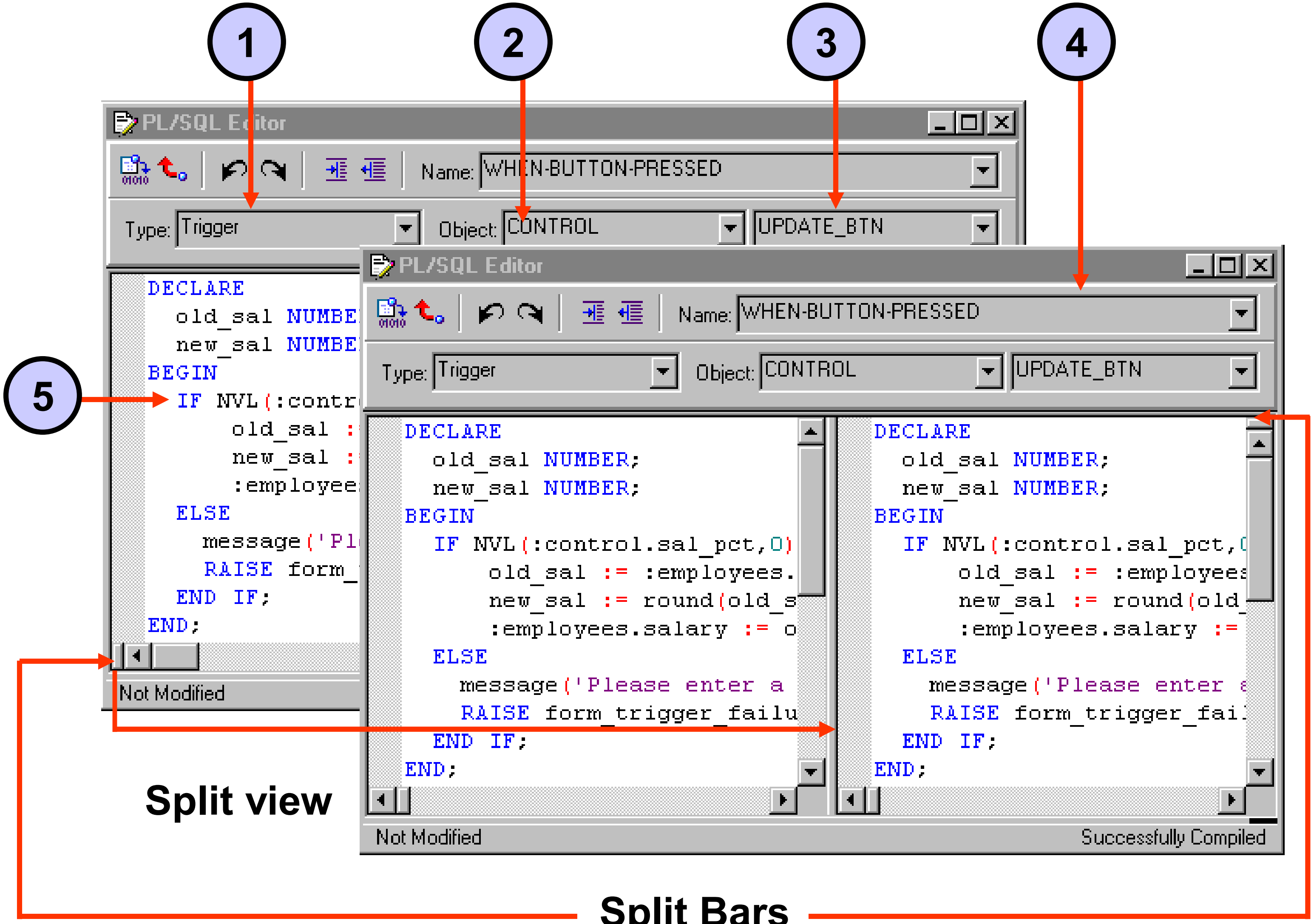


Trigger: ON-ERROR

General	
Name	ON-ERROR
Subclass Information	
Comments	
Functional	
Trigger Style	PL/SQL
Trigger Text	
Fire in Enter-Query Mode	Yes
Execution Hierarchy	Override
Help	
Display in 'Keyboard Help'	No
'Keyboard Help' Text	

Name

PL/SQL Editor Features



Name: WHEN-BUTTON-PRESSED

Type: Trigger Object: CONTROL UPDATE_BTN

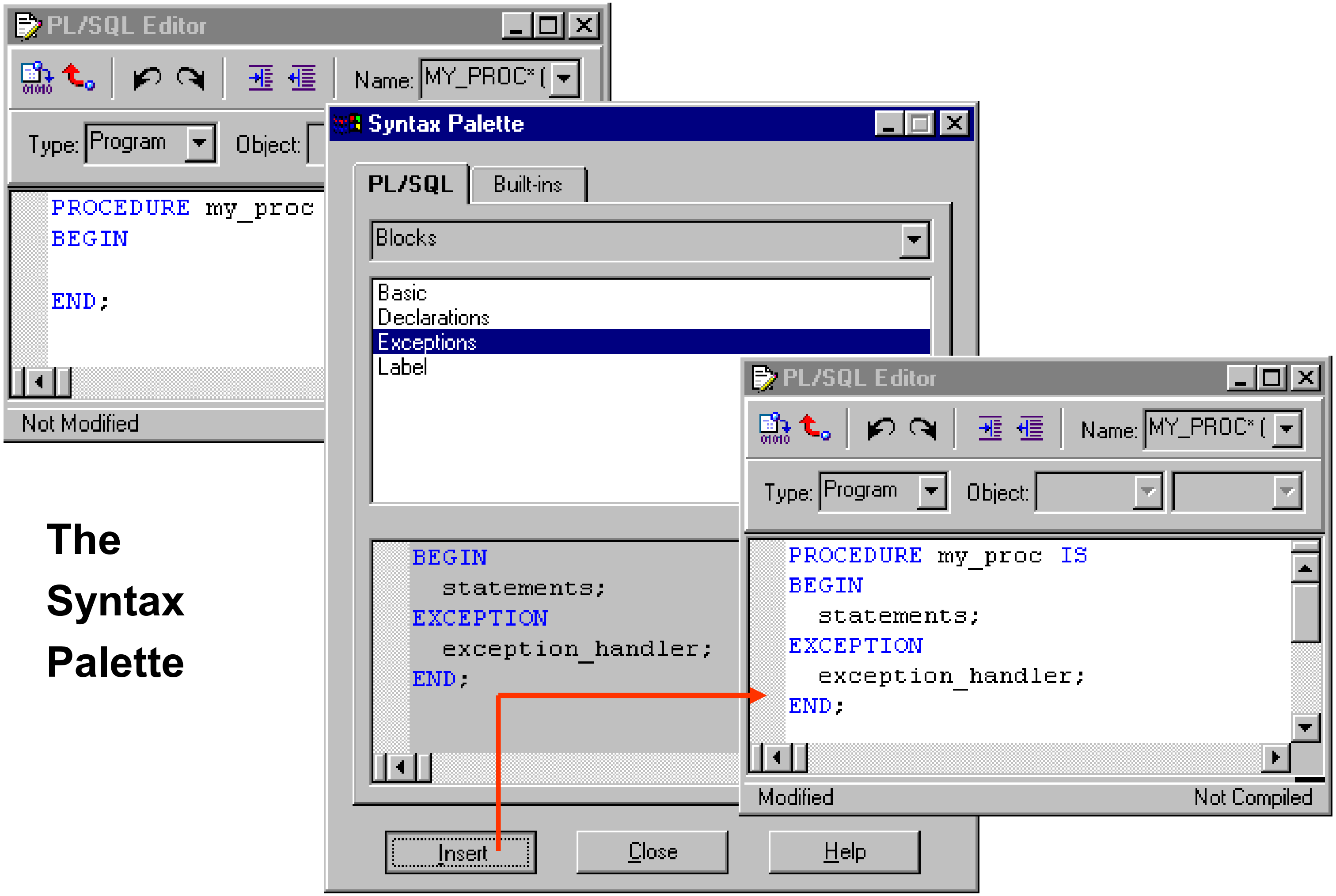
```
DECLARE
  old_sal NUMBER;
  new_sal NUMBER;
BEGIN
  IF NVL(:control.sal_pct,0)
      old_sal := :employees.
      new_sal := round(old_s
      :employees.salary := o
  ELSE
    message('Please enter a
    RAISE form_trigger_failu
  END IF;
END;
```

Not Modified Successfully Compiled

Split view

Split Bars

PL/SQL Editor Features



```
PROCEDURE my_proc
BEGIN

END;
```

```
BEGIN
  statements;
EXCEPTION
  exception_handler;
END;
```

```
PROCEDURE my_proc IS
BEGIN
  statements;
EXCEPTION
  exception_handler;
END;
```

The Syntax Palette

The Database Trigger Editor

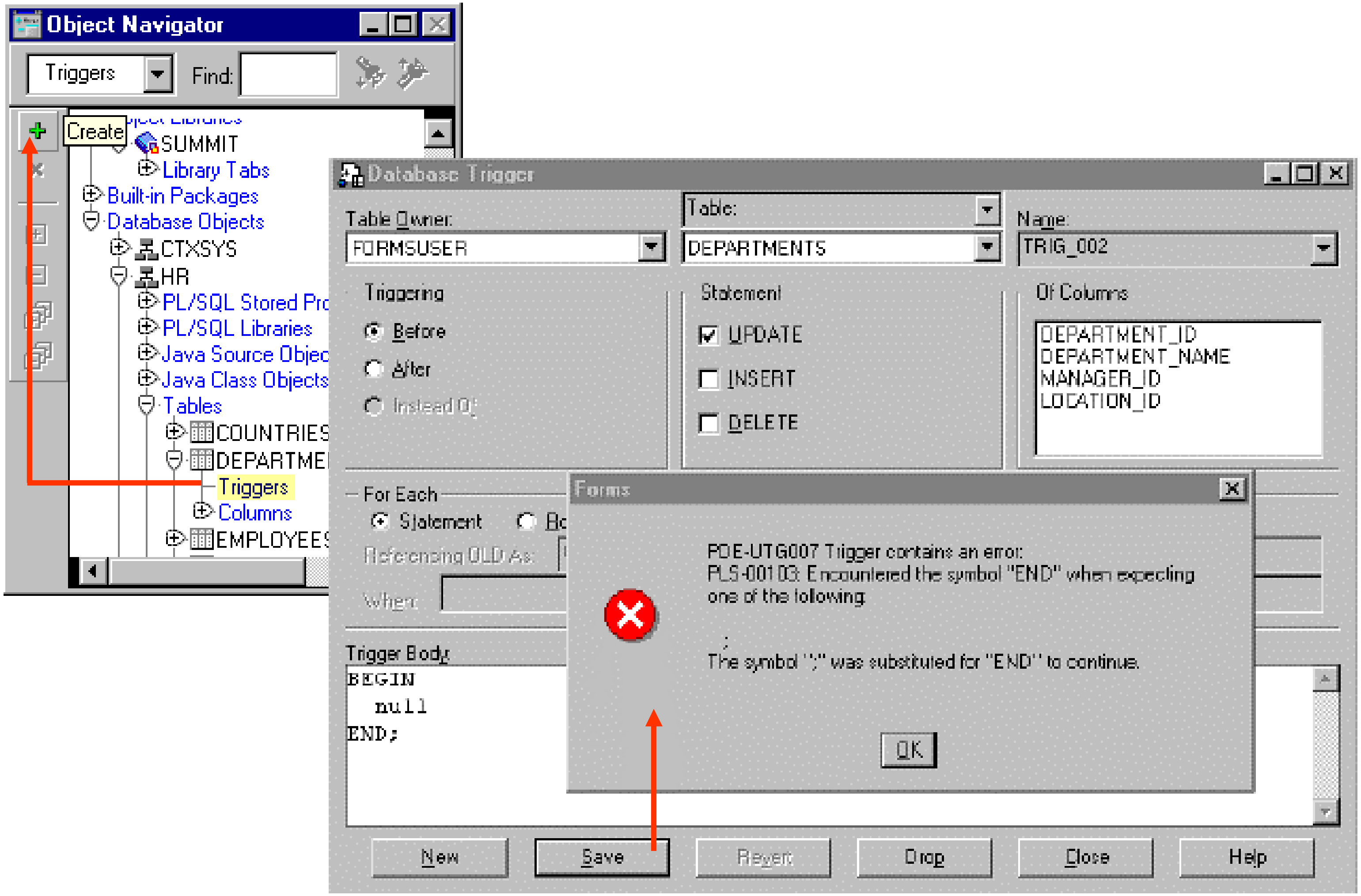


Table Owner: FORMSUSER

Table: DEPARTMENTS

Name: TRIG_002

Triggering
- ● Before
- ○ After
- ○ Instead Of

Statement
- ☑ UPDATE
- □ INSERT
- □ DELETE

Of Columns
- DEPARTMENT_ID
- DEPARTMENT_NAME
- MANAGER_ID
- LOCATION_ID

For Each
- ● Statement
- ○ Ro

Referencing OLD As:

When:

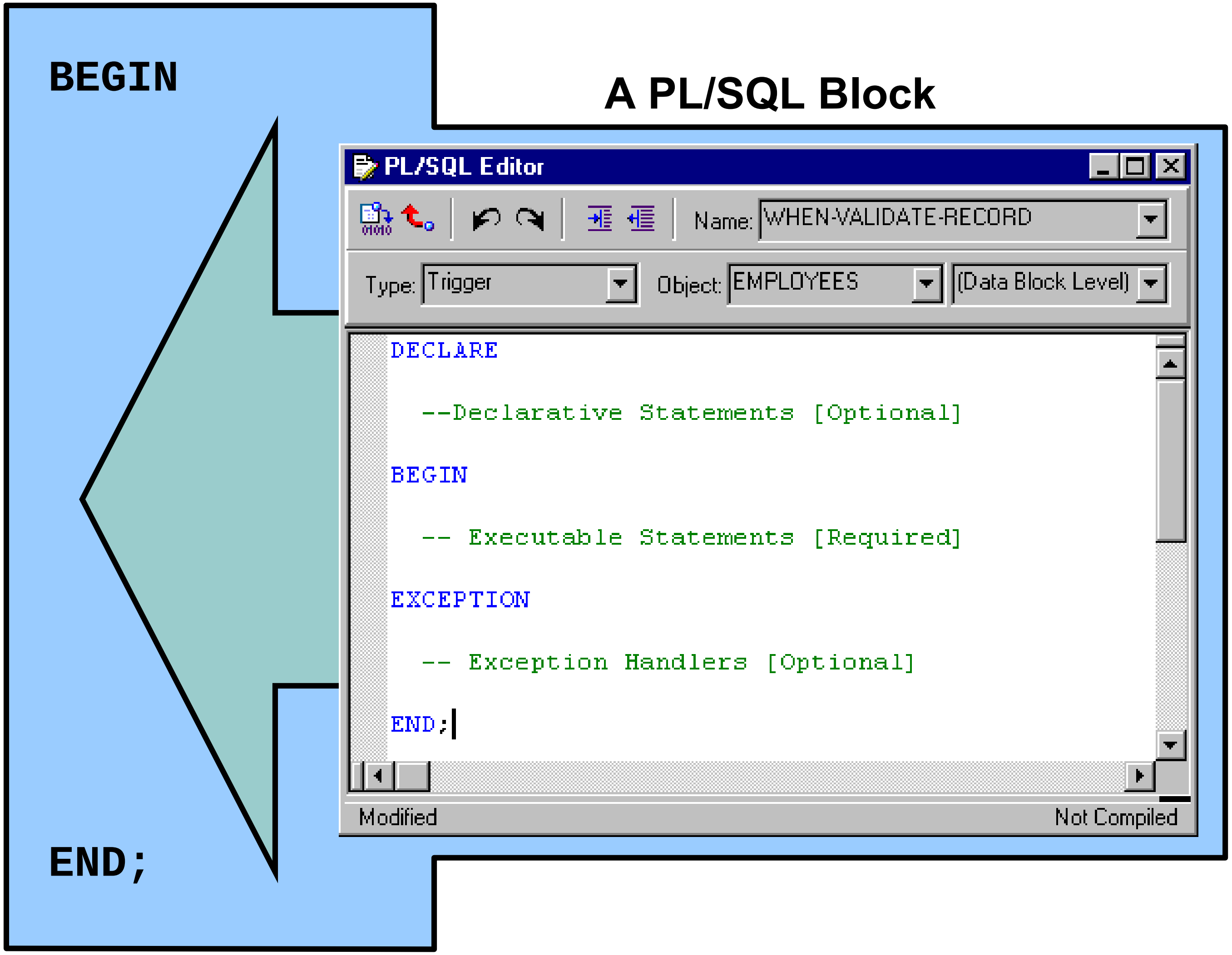
Trigger Body

```
BEGIN
  null
END;
```

FDE-UTG007 Trigger contains an error:
PLS-00103: Encountered the symbol "END" when expecting one of the following:

;
The symbol ";" was substituted for "END" to continue.

Writing Trigger Code



BEGIN

A PL/SQL Block

Name: WHEN-VALIDATE-RECORD

Type: Trigger Object: EMPLOYEES (Data Block Level)

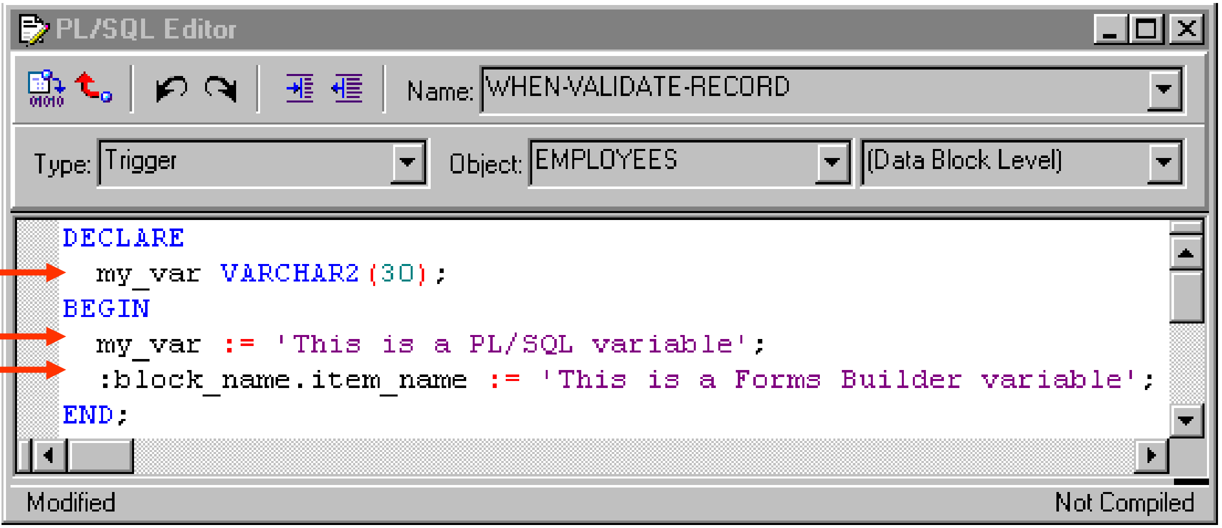
```
DECLARE

  --Declarative Statements [Optional]

BEGIN

  -- Executable Statements [Required]

EXCEPTION

  -- Exception Handlers [Optional]

END;
```

Modified Not Compiled

END;

Using Variables in Triggers

- PL/SQL variables must be declared in a trigger or defined in a package



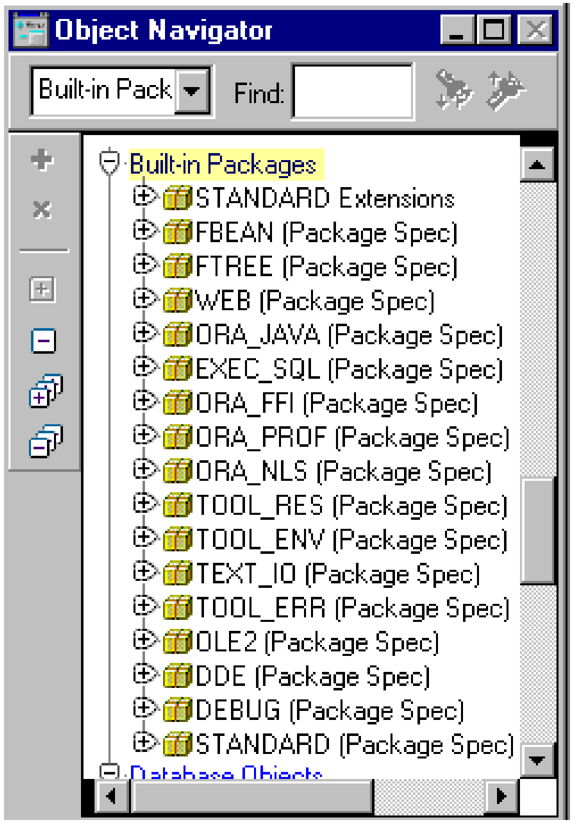
```
DECLARE
  my_var VARCHAR2(30);
BEGIN
  my_var := 'This is a PL/SQL variable';
  :block_name.item_name := 'This is a Forms Builder variable';
END;
```

- Forms Builder variables
  - Are not formally declared in PL/SQL
  - Need a colon (:) prefix in reference

Forms Builder Variables

Variable Type	Purpose	Syntax
Items	Presentation and user interaction	:block_name.item_name
Global variable	Session-wide character variable	:GLOBAL.variable_name
System variables	Form status and control	:SYSTEM.variable_name
Parameters	Passing values in and out of module	:PARAMETER.name

Adding Functionality with Built-In Subprograms



Built-in Packages
STANDARD Extensions
FBEAN (Package Spec)
FTREE (Package Spec)
WEB (Package Spec)
ORA_JAVA (Package Spec)
EXEC_SQL (Package Spec)
ORA_FFI (Package Spec)
ORA_PROF (Package Spec)
ORA_NLS (Package Spec)
TOOL_RES (Package Spec)
TOOL_ENV (Package Spec)
TEXT_IO (Package Spec)
TOOL_ERR (Package Spec)
OLE2 (Package Spec)
DDE (Package Spec)
DEBUG (Package Spec)
STANDARD (Package Spec)
Database Objects

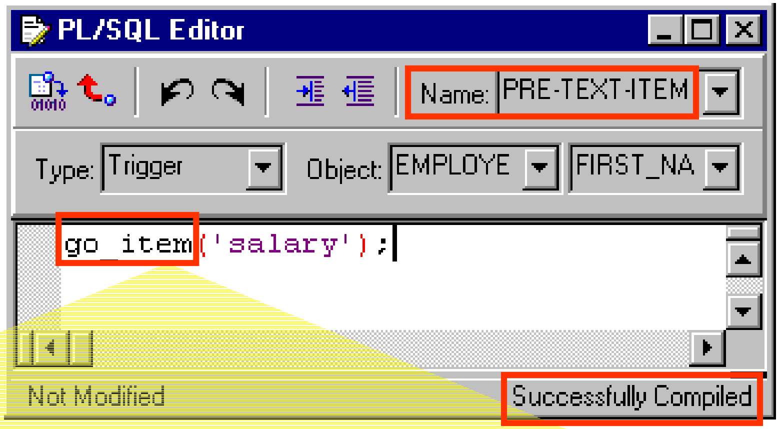
Built-ins belong to either:

- **The Standard Extensions package where no prefix is required**
- **Another Forms Builder package where a prefix is required**

Limits of Use

- Unrestricted built-ins are allowed in any trigger or subprogram.
- Restricted built-ins are allowed only in certain triggers and subprograms called from such triggers.
- Consult the Help system.

Compiles:



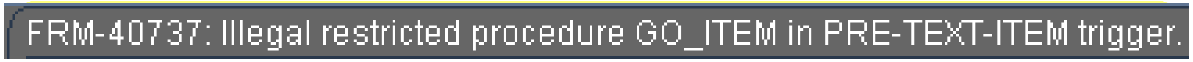
Name: PRE-TEXT-ITEM

Type: Trigger Object: EMPLOYE FIRST_NA

```
go_item('salary');
```

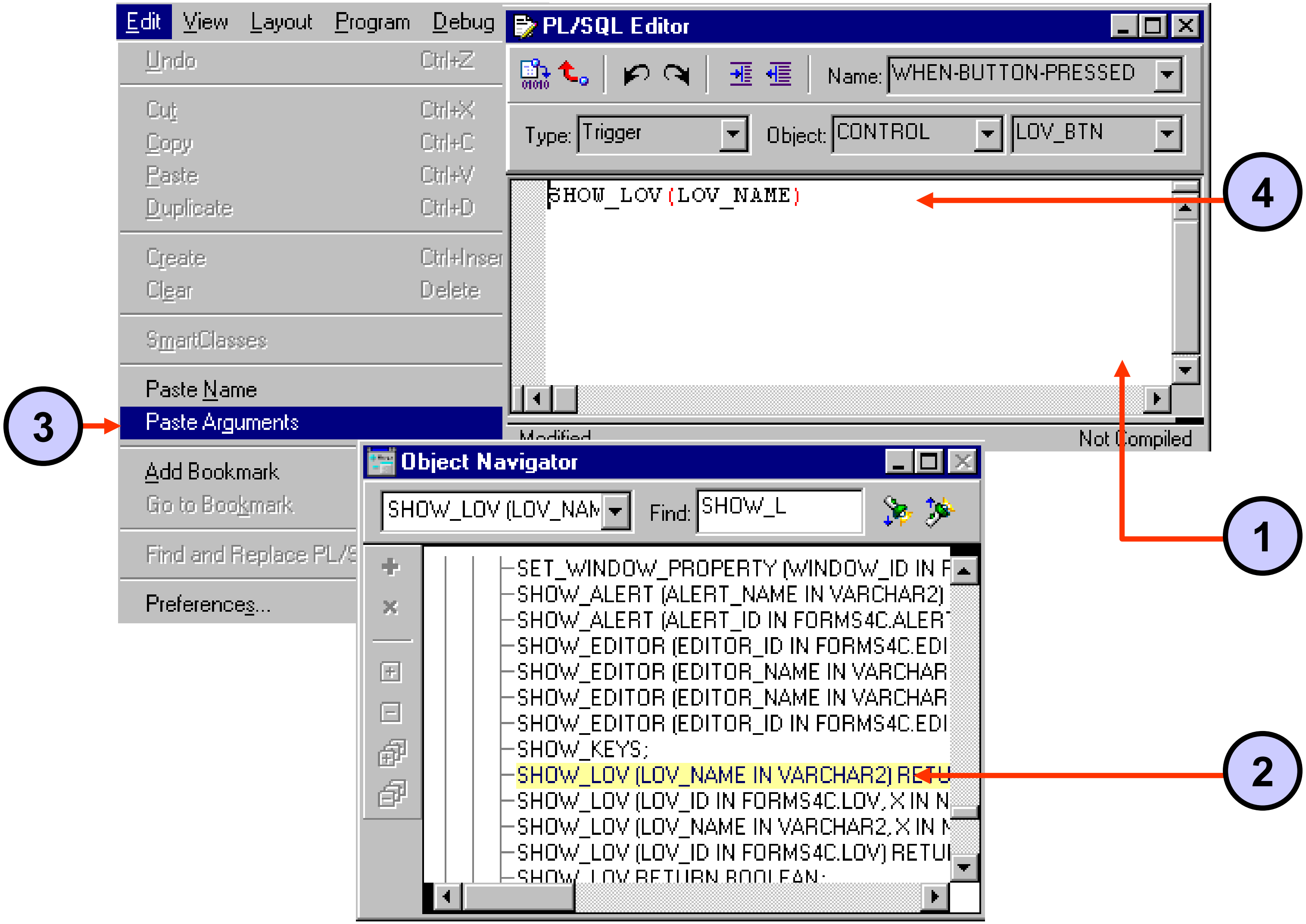
Not Modified Successfully Compiled

Run-time error when trigger fires:



FRM-40737: Illegal restricted procedure GO_ITEM in PRE-TEXT-ITEM trigger.

Using Built-In Definitions



Name: WHEN-BUTTON-PRESSED

Type: Trigger

Object: CONTROL LOV_BTN

```
SHOW_LOV(LOV_NAME)
```

SHOW_LOV (LOV_NAM

Find: SHOW_L

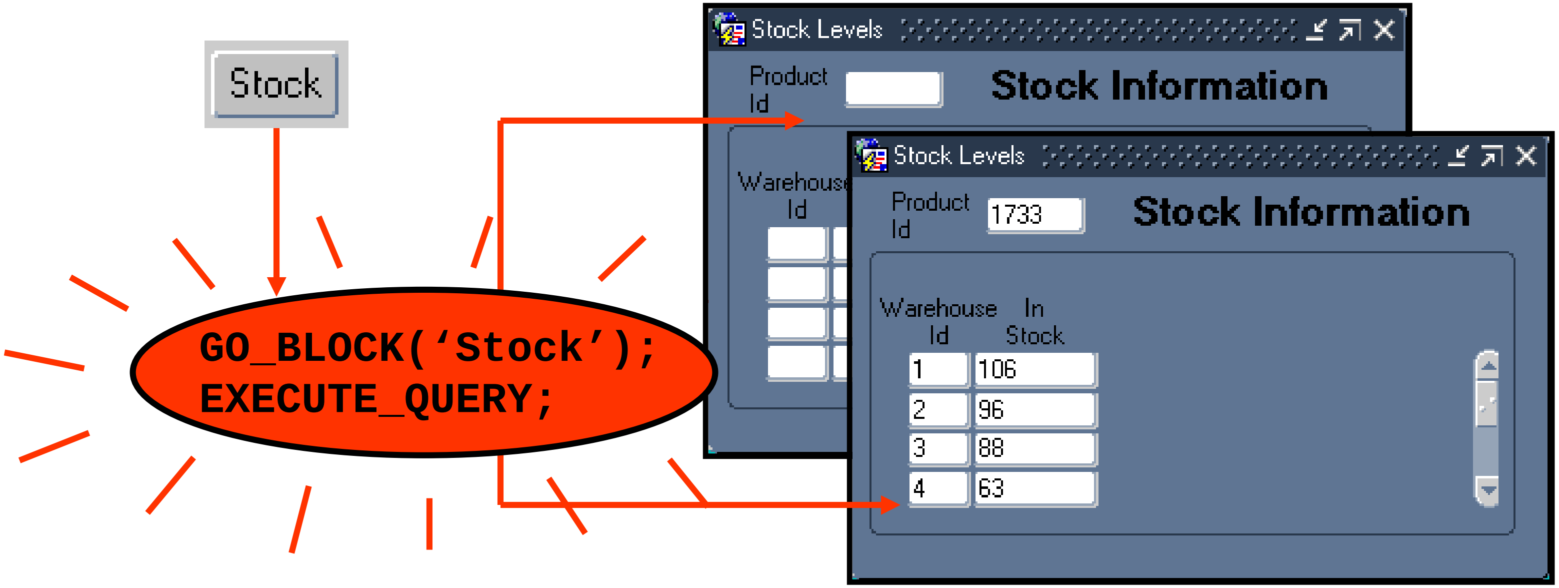
```
SET_WINDOW_PROPERTY (WINDOW_ID IN F
SHOW_ALERT (ALERT_NAME IN VARCHAR2)
SHOW_ALERT (ALERT_ID IN FORMS4C.ALER
SHOW_EDITOR (EDITOR_ID IN FORMS4C.EDI
SHOW_EDITOR (EDITOR_NAME IN VARCHAR
SHOW_EDITOR (EDITOR_NAME IN VARCHAR
SHOW_EDITOR (EDITOR_ID IN FORMS4C.EDI
SHOW_KEYS;
SHOW_LOV (LOV_NAME IN VARCHAR2) RETU
SHOW_LOV (LOV_ID IN FORMS4C.LOV, X IN N
SHOW_LOV (LOV_NAME IN VARCHAR2, X IN N
SHOW_LOV (LOV_ID IN FORMS4C.LOV) RETU
SHOW_LOV RETURN BOOLEAN;
```

Useful Built-Ins

- EDIT_TEXTITEM
- ENTER_QUERY, EXECUTE_QUERY
- EXIT_FORM
- GET_ITEM_PROPERTY, SET_ITEM_PROPERTY
- GO_BLOCK, GO_ITEM
- MESSAGE
- SHOW_ALERT, SHOW_EDITOR, SHOW_LOV
- SHOW_VIEW, HIDE_VIEW

Using Triggers:
`When-Button-Pressed` Trigger

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



```
GO_BLOCK('Stock');
EXECUTE_QUERY;
```

Stock Levels

Stock Information

Product Id

Warehouse Id

Stock Levels

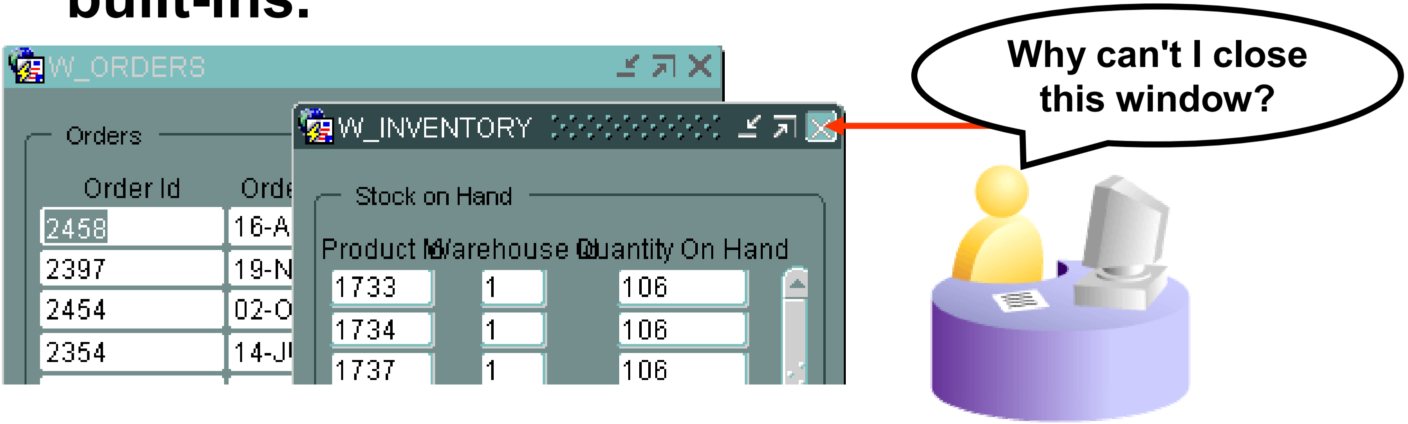
Stock Information

Product Id 1733

Warehouse Id	In Stock
1	106
2	96
3	88
4	63

Using Triggers:
`When-Window-Closed` Trigger

- Fires when the operator closes a window by using a window manager-specific close command.
- Accepts restricted and unrestricted built-ins.
- Used to programmatically close a window when the operator issues a window manager-specific close command. You can close a window by using built-ins.



W_ORDERS

Orders

Order Id	Orde
2458	16-A
2397	19-N
2454	02-O
2354	14-J

W_INVENTORY

Stock on Hand

Product Id	Warehouse Id	Quantity On Hand
1733	1	106
1734	1	106
1737	1	106

Summary

In this lesson, you should have learned that:

- You can use the PL/SQL Editor to write trigger code
- Trigger code has three sections:
  - Declaration section (optional)
  - Executable statements section (required)
  - Exception handlers section (optional)
- You can add functionality by calling built-in subprograms from triggers
- Restricted built-ins are not allowed in triggers that fire while navigation is occurring

Summary

- The `When-Button-Pressed` trigger fires when the user presses a button
- The `When-Window-Closed` trigger fires when the user closes a window

Practice 14 Overview

This practice covers the following topics:

- Using built-ins to display LOVs
- Using the `When-Button-Pressed` and `When-Window-Closed` triggers to add functionality to applications
- Using built-ins to display and hide the Help stack canvas

15

Debugging Triggers

Objectives

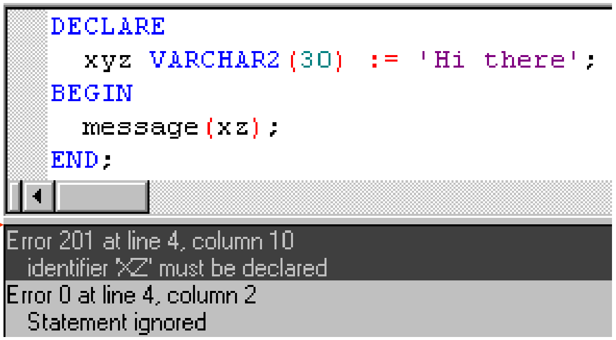
After completing this lesson, you should be able to do the following:

- Describe the components of the Debug Console
- Use the Run Form Debug button to run a form module in debug mode
- Debug PL/SQL code

The Debugging Process

Monitor and debug triggers by:

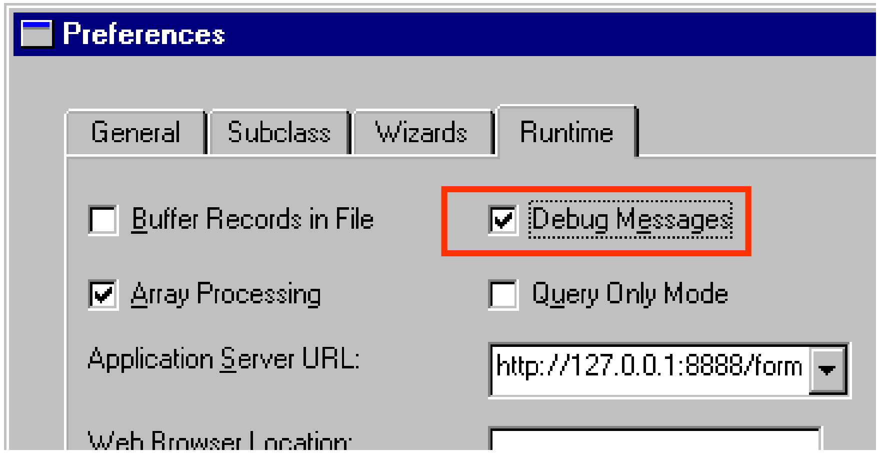
- Compiling and correcting errors in the PL/SQL Editor



```
DECLARE
  xyz VARCHAR2(30) := 'Hi there';
BEGIN
  message(xz);
END;
```

```
Error 201 at line 4, column 10
  identifier 'XZ' must be declared
Error 0 at line 4, column 2
  Statement ignored
```

- Displaying debug messages at run time



☐ Buffer Records in File

☑ Debug Messages

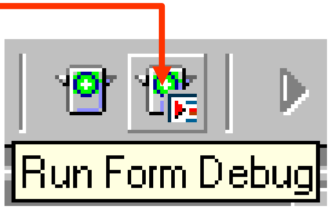
☑ Array Processing

☐ Query Only Mode

Application Server URL: http://127.0.0.1:8888/form

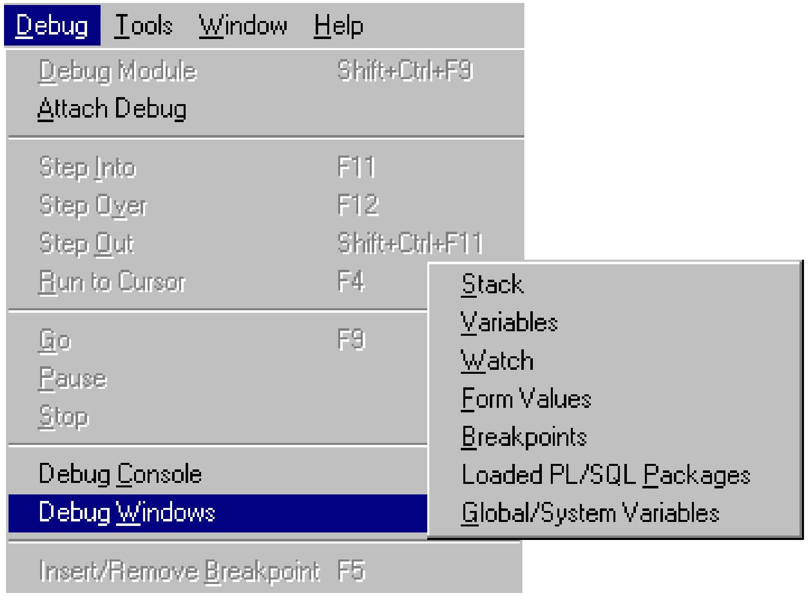
Web Browser Location:

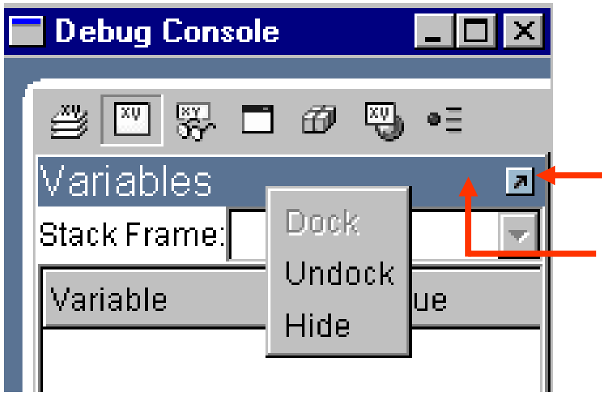
- Invoking the PL/SQL Debugger



The Debug Console

- Stack
- Variables
- Watch
- Form Values
- PL/SQL Packages
- Global and System Variables
- Breakpoints



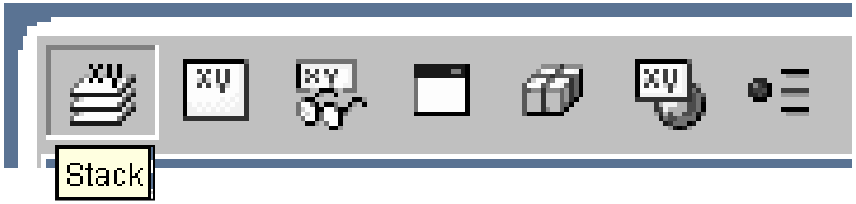


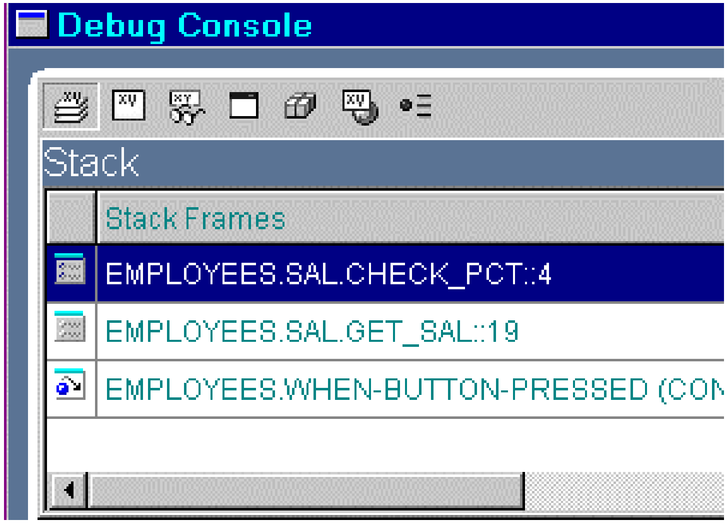
Dock/ Undock

Click bar for Pop-up Menu

The Debug Console: Stack Panel

- **Stack**
- Variables
- Watch
- Form Values
- PL/SQL Packages
- Global and System Variables
- Breakpoints

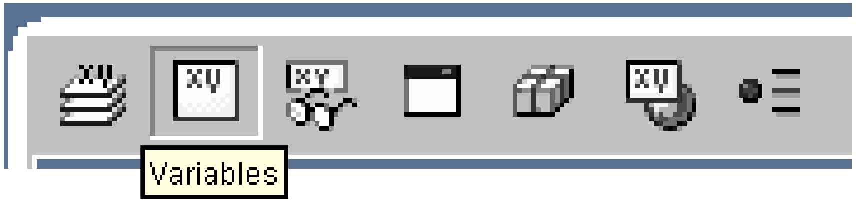


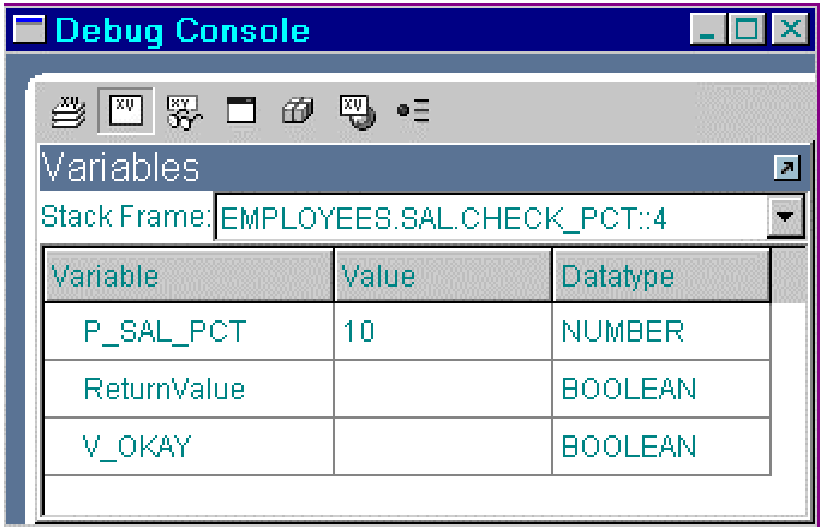


	Stack Frames
	EMPLOYEES.SAL.CHECK_PCT::4
	EMPLOYEES.SAL.GET_SAL::19
	EMPLOYEES.WHEN-BUTTON-PRESSED (CON

The Debug Console: Variables Panel

- Stack
- **Variables**
- Watch
- Form Values
- PL/SQL Packages
- Global and System Variables
- Breakpoints



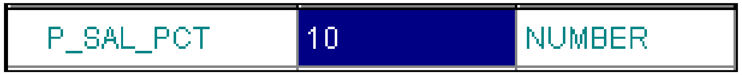


Variables

Stack Frame: EMPLOYEES.SAL.CHECK_PCT::4

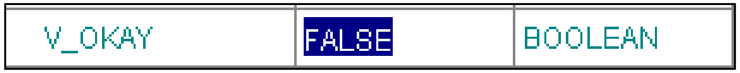
Variable	Value	Datatype
P_SAL_PCT	10	NUMBER
ReturnValue		BOOLEAN
V_OKAY		BOOLEAN

Read-only:



P_SAL_PCT	10	NUMBER

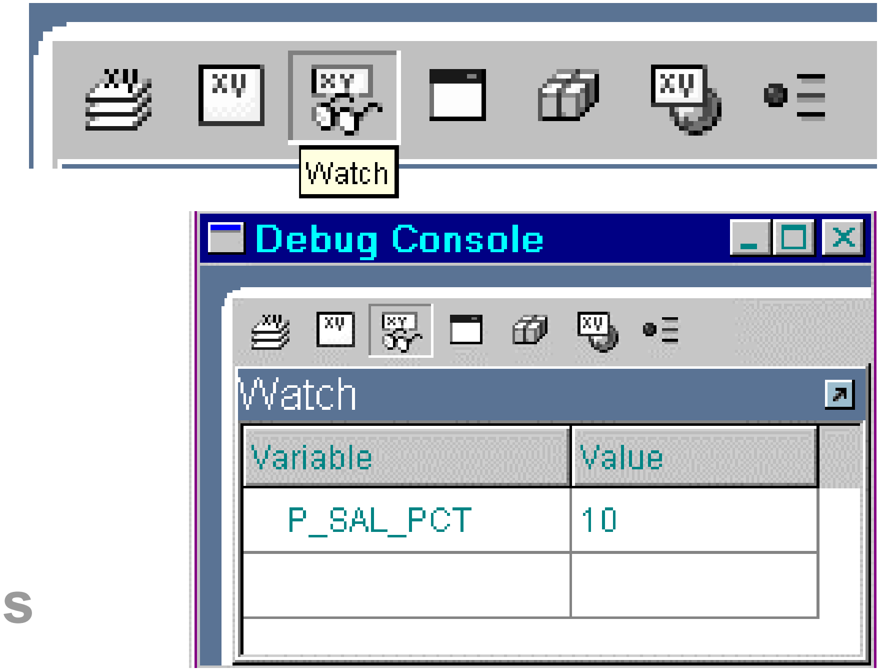
Modifiable:



V_OKAY	FALSE	BOOLEAN

The Debug Console: Watch Panel

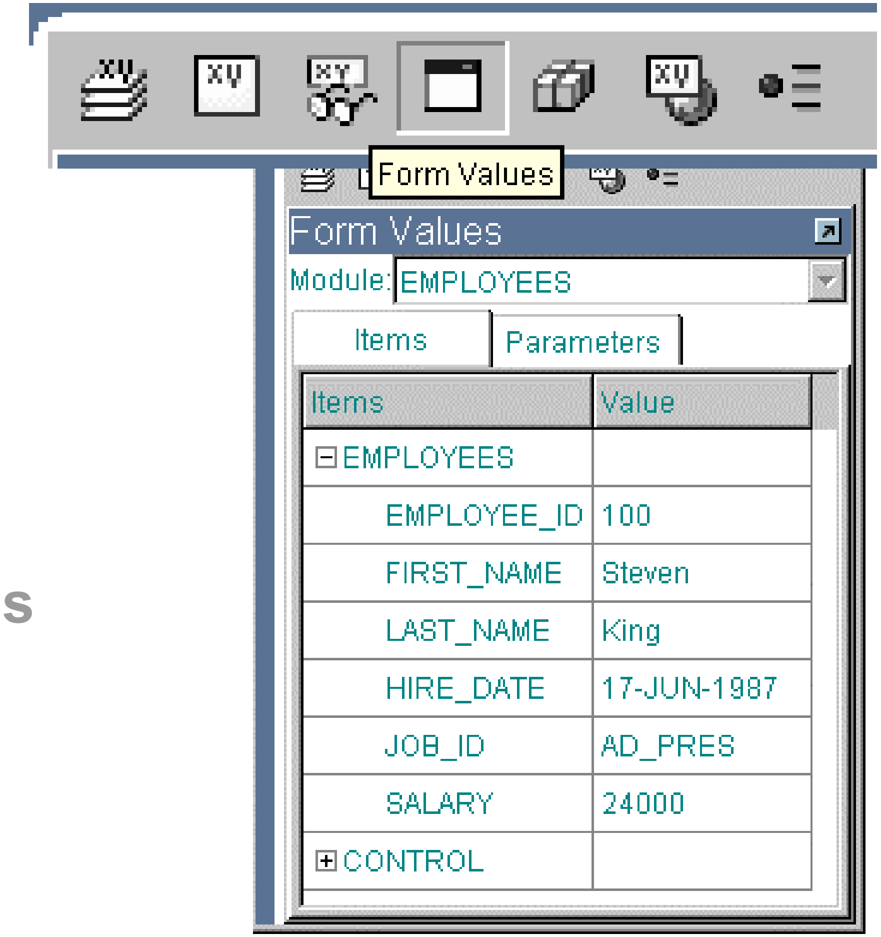
- Stack
- Variables
- **Watch**
- Form Values
- PL/SQL Packages
- Global and System Variables
- Breakpoints



Variable	Value
P_SAL_PCT	10

The Debug Console: Form Values Panel

- Stack
- Variables
- Watch
- **Form Values**
- PL/SQL Packages
- Global and System Variables
- Breakpoints

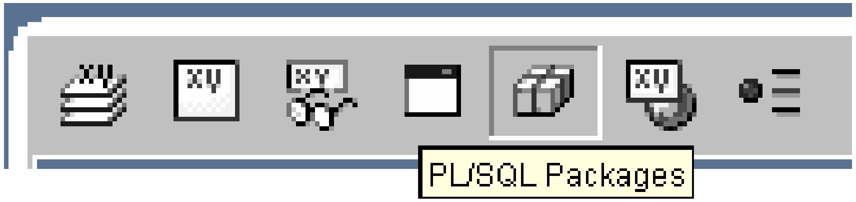


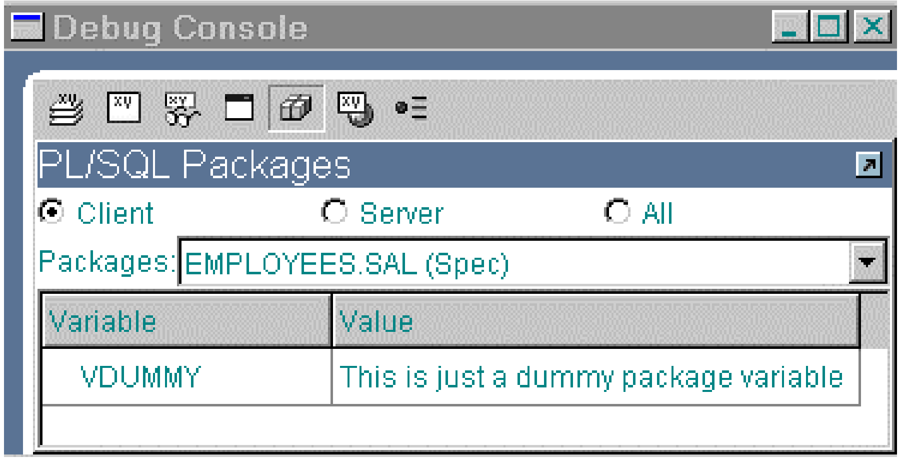
Module: EMPLOYEES

Items	Value
⊟ EMPLOYEES	
EMPLOYEE_ID	100
FIRST_NAME	Steven
LAST_NAME	King
HIRE_DATE	17-JUN-1987
JOB_ID	AD_PRES
SALARY	24000
⊞ CONTROL	

The Debug Console: PL/SQL Packages Panel

- Stack
- Variables
- Watch
- Form Values
- **PL/SQL Packages**
- Global and System Variables
- Breakpoints





PL/SQL Packages

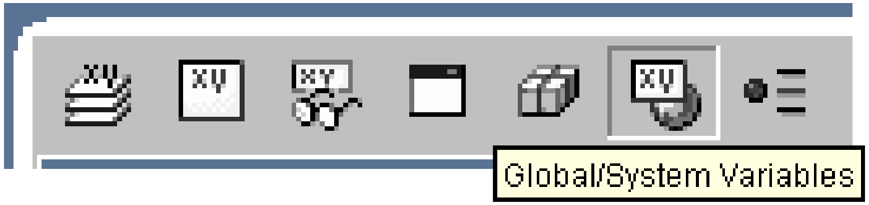
● Client ○ Server ○ All

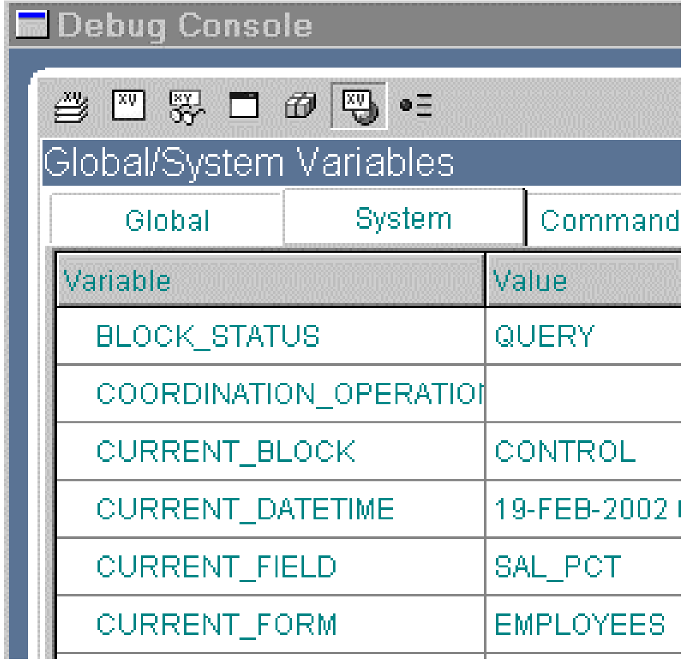
Packages: EMPLOYEES.SAL (Spec)

Variable	Value
VDUMMY	This is just a dummy package variable

The Debug Console: Global/System Variables Panel

- Stack
- Variables
- Watch
- Form Values
- Loaded PL/SQL Packages
- **Global and System Variables**
- Breakpoints

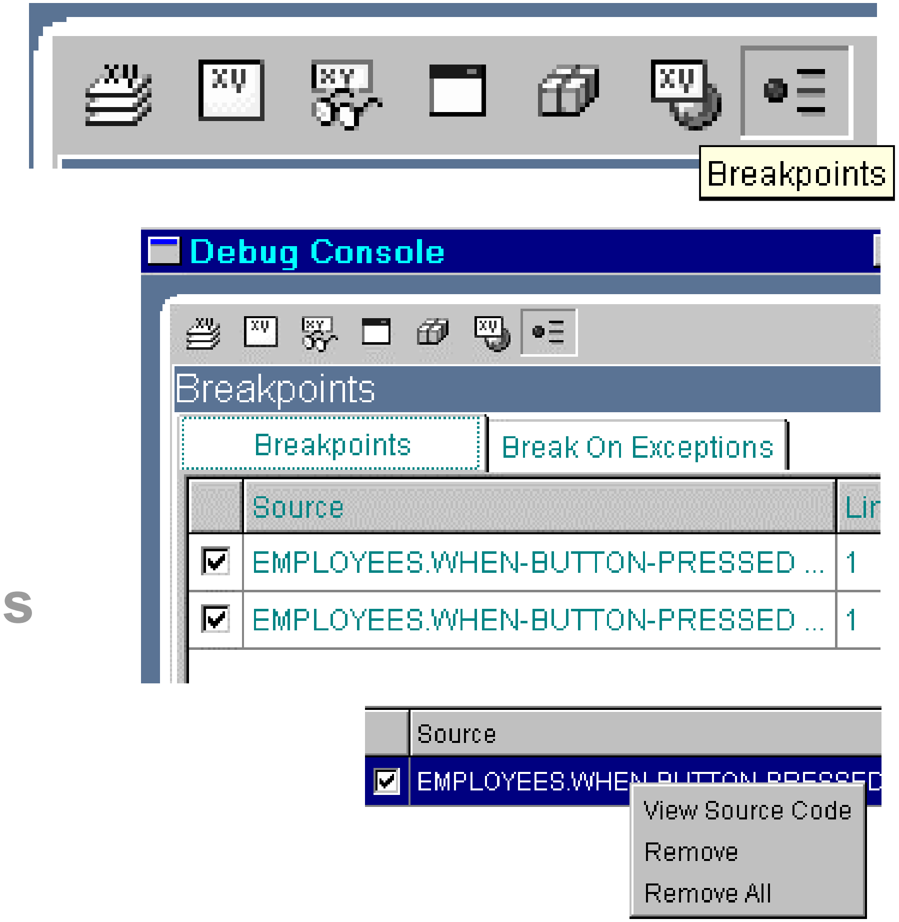




Variable	Value
BLOCK_STATUS	QUERY
COORDINATION_OPERATIO	
CURRENT_BLOCK	CONTROL
CURRENT_DATETIME	19-FEB-2002
CURRENT_FIELD	SAL_PCT
CURRENT_FORM	EMPLOYEES

The Debug Console: Breakpoints Panel

- Stack
- Variables
- Watch
- Form Values
- Loaded PL/SQL Packages
- Global and System Variables
- **Breakpoints**

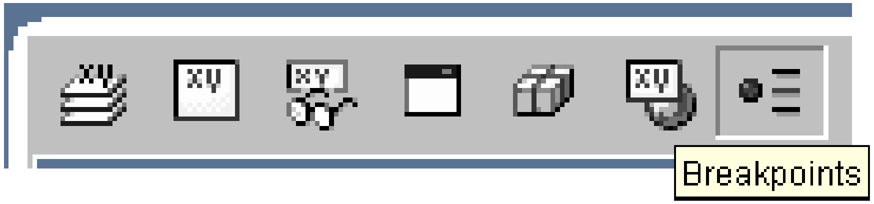


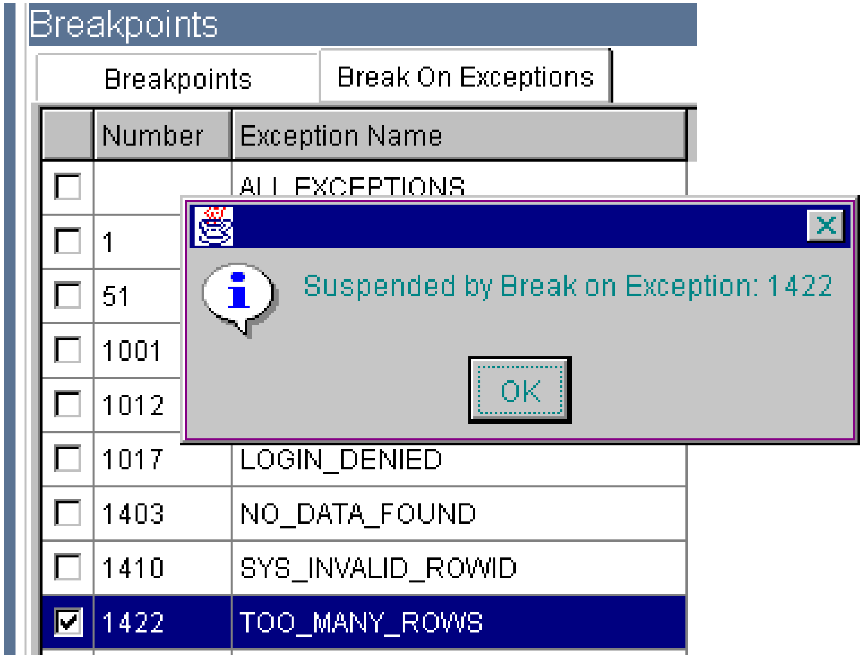
	Source	Lir
☑	EMPLOYEES.WHEN-BUTTON-PRESSED ...	1
☑	EMPLOYEES.WHEN-BUTTON-PRESSED ...	1

	Source
☑	EMPLOYEES.WHEN-BUTTON-PRESSED

The Debug Console

- Stack
- Variables
- Watch
- Form Values
- Loaded PL/SQL Packages
- Global and System Variables
- **Breakpoints**





	Number	Exception Name
☐		ALL_EXCEPTIONS
☐	1	
☐	51	
☐	1001	
☐	1012	
☐	1017	LOGIN_DENIED
☐	1403	NO_DATA_FOUND
☐	1410	SYS_INVALID_ROWID
☑	1422	TOO_MANY_ROWS

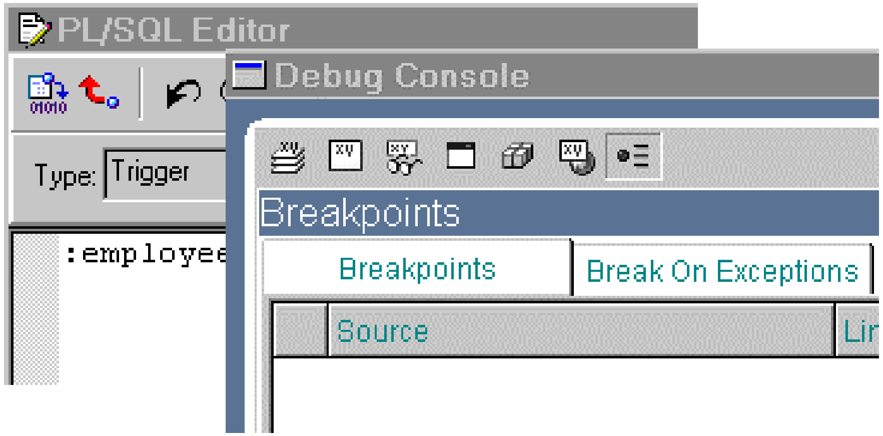
Suspended by Break on Exception: 1422

Setting Breakpoints in Client Code

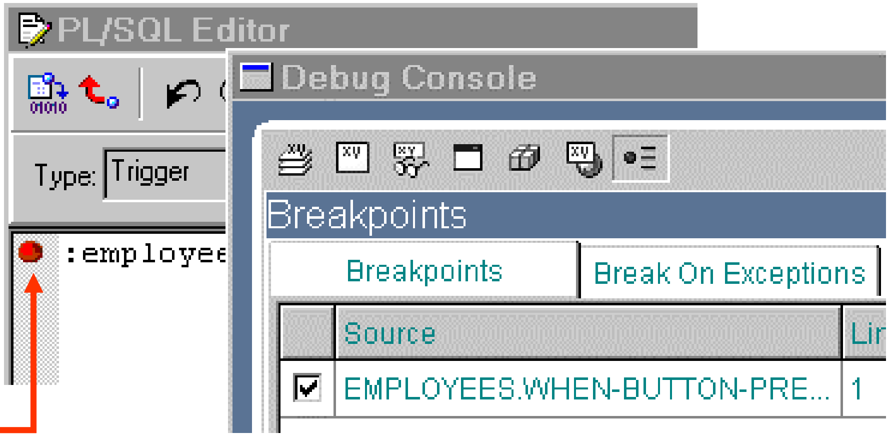
Breakpoints:

- Suspend form execution
- Return control to the debugger
- Remain in effect for the Forms Builder session
- May be enabled and disabled
- Are set in the PL/SQL Editor on executable lines of code

Before setting breakpoint:



After setting breakpoint:



	Source	Lir
☑	EMPLOYEES.WHEN-BUTTON-PRE...	1

Setting Breakpoints in Stored Code

- Can set on stored program units:
  - Expand Database Objects node
  - Expand <schema> node
  - Expand PL/SQL Stored Program Units node
  - Double-click program unit
  - Set breakpoint in PL/SQL Editor
- Cannot set on database triggers or stored PL/SQL libraries
- Compile with debug information

Debugging Tips

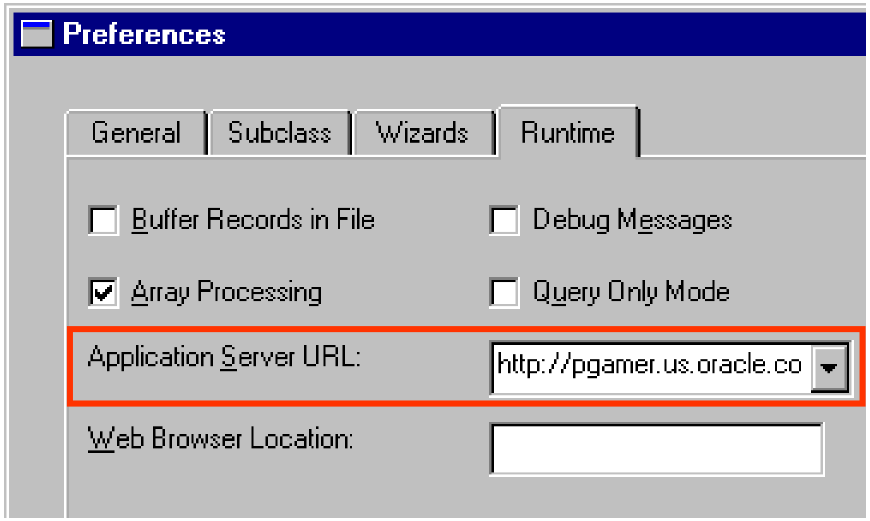
- Connect to the database for SQL compilation.
- The line that fails is not always responsible.
- Watch for missing semicolons and quotation marks.
- Define triggers at the correct level.
- Place triggers where the event will happen.

Running a Form in Debug Mode

Run Form Debug (Compiles automatically)

Contains source code and executable run file .FMX (Runs automatically)

Runs Form in Debug Mode on Server specified in Runtime Preferences



Preferences

□ Buffer Records in File

□ Debug Messages

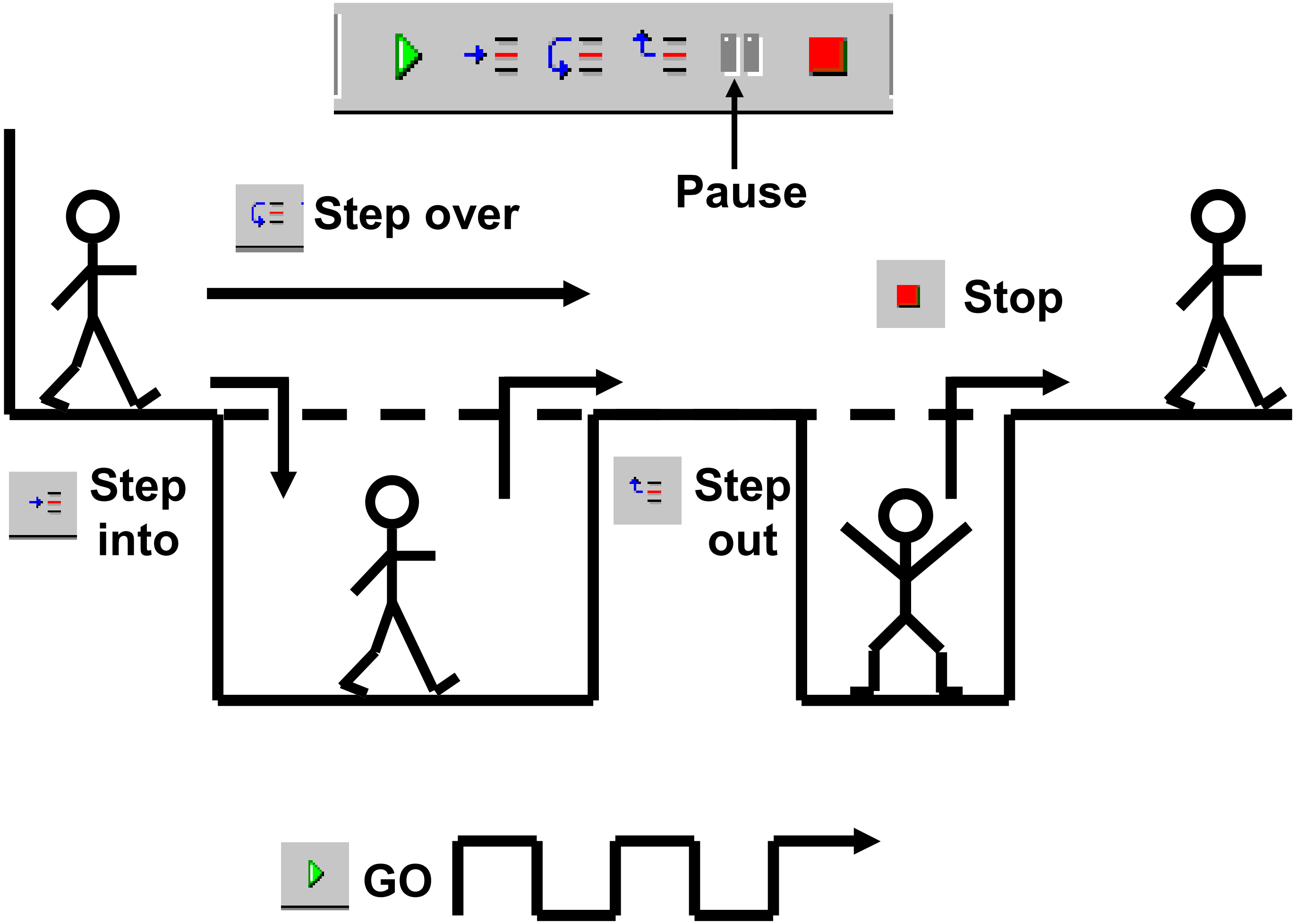
☑ Array Processing

□ Query Only Mode

Application Server URL: http://pgamer.us.oracle.co

Web Browser Location:

Stepping Through Code



Step over

Pause

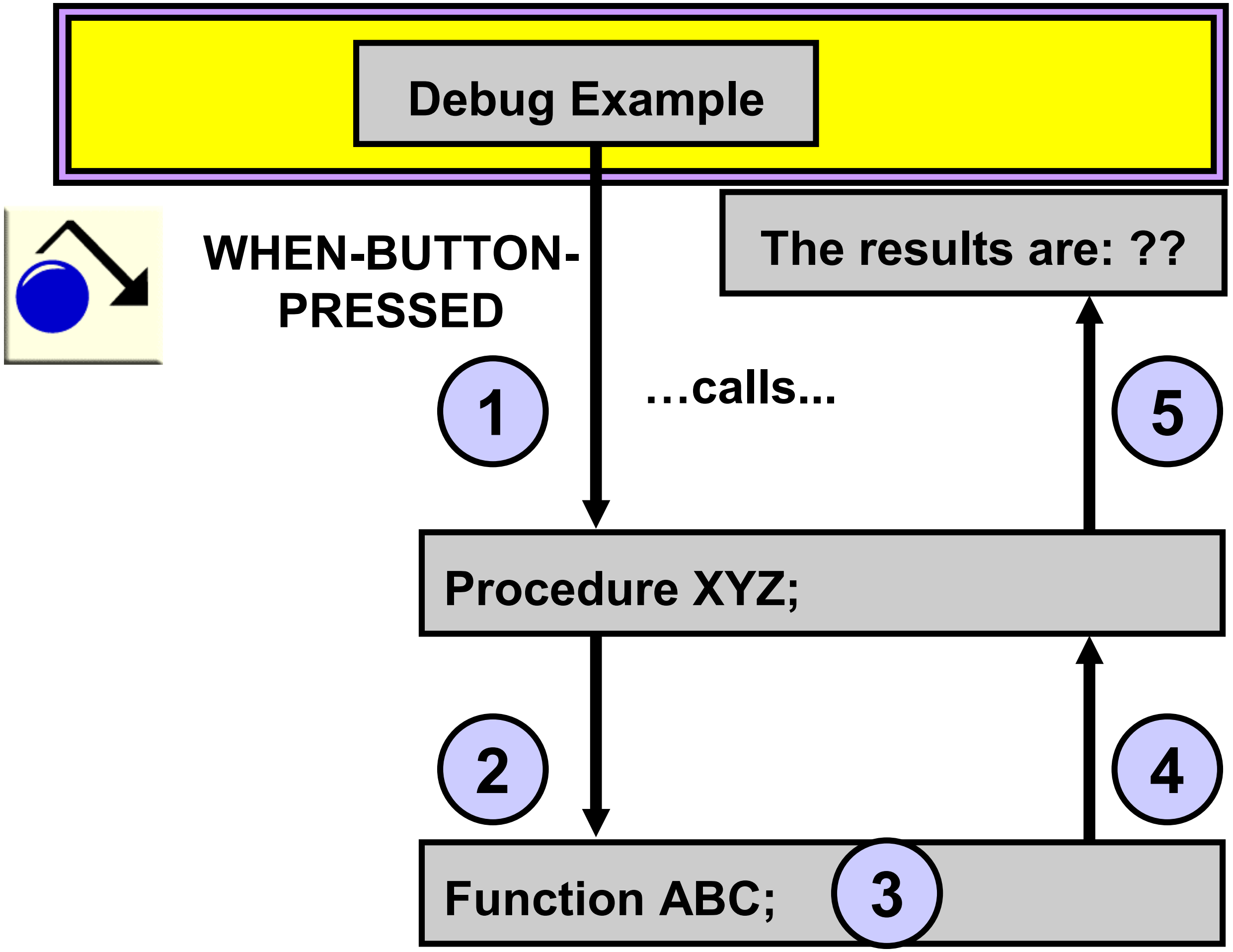
Stop

Step into

Step out

GO

Debug Example



Debug Example

WHEN-BUTTON-PRESSED

…calls…

The results are: ??

Procedure XYZ;

Function ABC;

Summary

In this lesson, you should have learned that:

- The Debug Console consists of panes to view the call stack, program variables, a user-defined watch list, Form values, loaded PL/SQL packages, global and system variables, and breakpoints
- You use the Run Debug button to run a form module in debug mode within Forms Builder
- You can set breakpoints in the PL/SQL Editor by double-clicking to the left of an executable line of code
- The debug buttons in the Forms Builder toolbar enable you to step through code in various ways

Practice 15 Overview

This practice covers the following topics:

- Running a form in debug mode from Forms Builder
- Setting breakpoints
- Stepping through code
- Viewing variable values while form is running

16

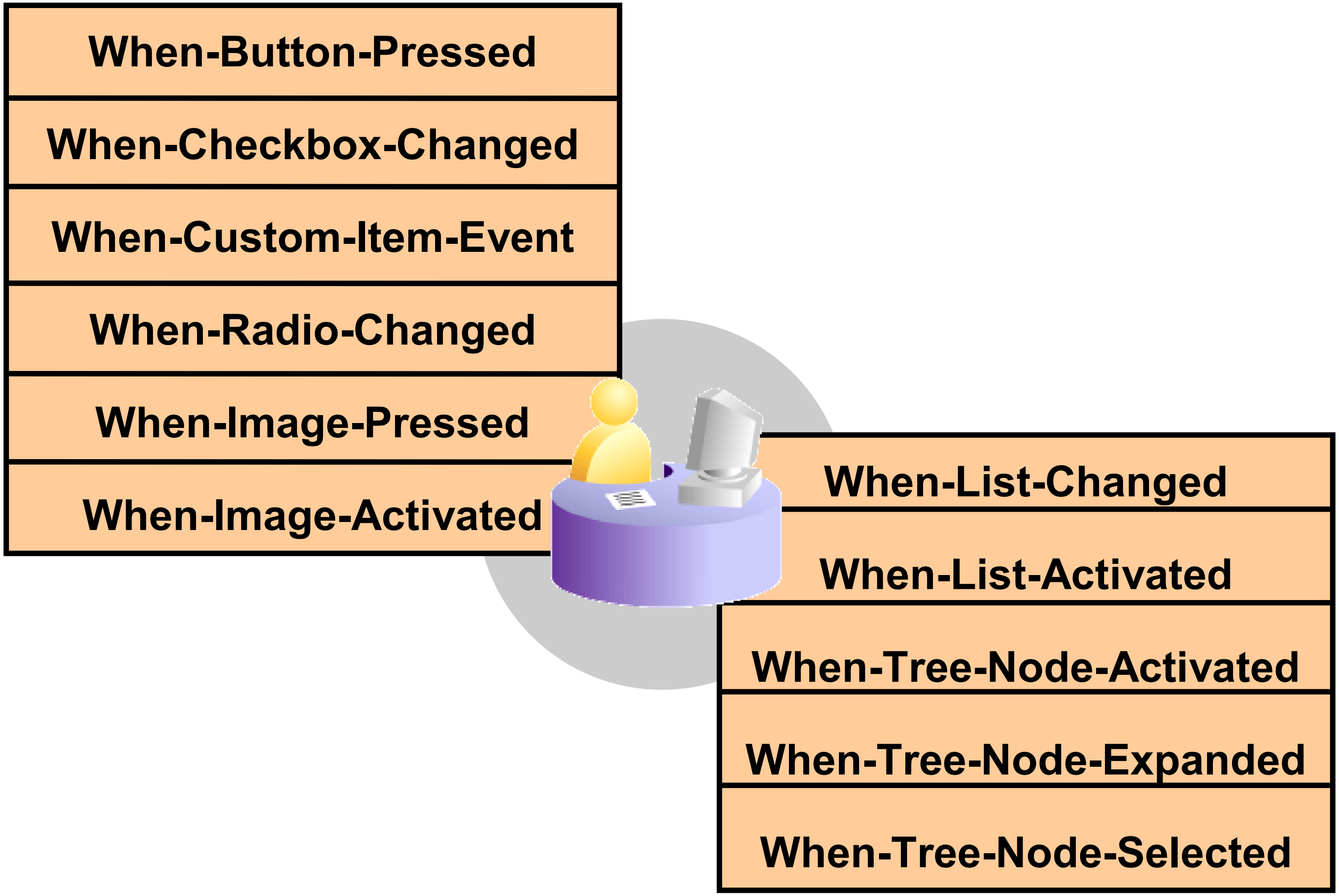
Adding Functionality to Items

Objectives

After completing this lesson, you should be able to do the following:

- Supplement the functionality of input items by using triggers and built-ins
- Supplement the functionality of noninput items by using triggers and built-ins

Item Interaction Triggers



When-Button-Pressed

When-Checkbox-Changed

When-Custom-Item-Event

When-Radio-Changed

When-Image-Pressed

When-Image-Activated

When-List-Changed

When-List-Activated

When-Tree-Node-Activated

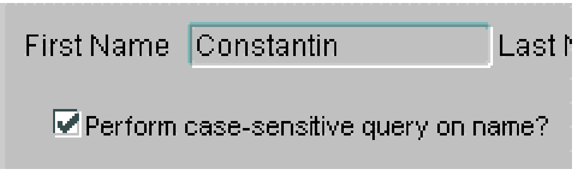
When-Tree-Node-Expanded

When-Tree-Node-Selected

Coding Item Interaction Triggers

- Valid commands:
  - SELECT statements
  - Standard PL/SQL constructs
  - All built-in subprograms
- Do not fire during:
  - Navigation
  - Validation (use When-Validate-“object” to code actions to take place during validation)

Interacting with Check Boxes



First Name Constantin

Last

☑ Perform case-sensitive query on name?

When-Checkbox-Changed

```
IF CHECKBOX_CHECKED('CONTROL.case_sensitive') THEN
  SET_ITEM_PROPERTY('CUSTOMERS.cust_first_name',
    CASE_INSENSITIVE_QUERY, PROPERTY_FALSE);
  SET_ITEM_PROPERTY('CUSTOMERS.cust_last_name',
    CASE_INSENSITIVE_QUERY, PROPERTY_FALSE);
ELSE
  SET_ITEM_PROPERTY('CUSTOMERS.cust_first_name',
    CASE_INSENSITIVE_QUERY, PROPERTY_TRUE);
SET_ITEM_PROPERTY('CUSTOMERS.cust_last_name',
    CASE_INSENSITIVE_QUERY, PROPERTY_TRUE);
END IF;
```

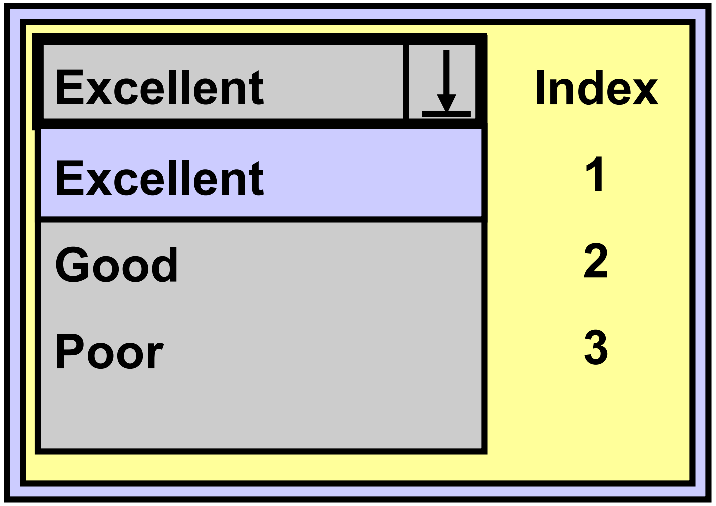
Changing List Items at Run Time

Triggers:

- When-List-Changed
- When-List-Activated

Built-ins:

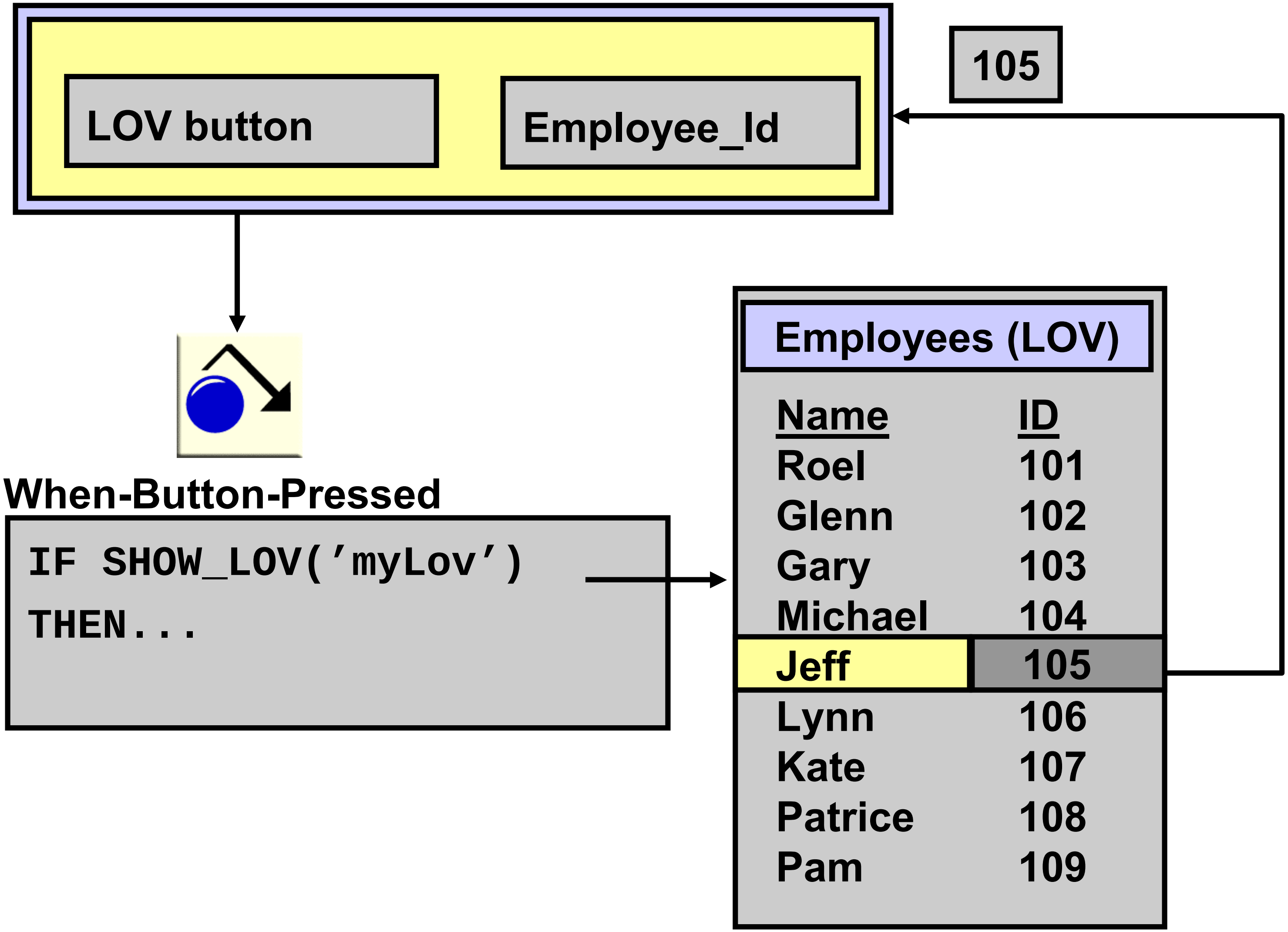
- `ADD_LIST_ELEMENT`
- `DELETE_LIST_ELEMENT`



Displaying LOVs from Buttons

- Uses:
  - Convenient alternative for accessing LOVs
  - Can display independently of text items
- Needs:
  - When-Button-Pressed trigger
  - `LIST_VALUES` or `SHOW_LOV` built-in

LOVs and Buttons



LOV button

Employee_Id

105

When-Button-Pressed

```
IF SHOW_LOV('myLov')
THEN...
```

Employees (LOV)

Name	ID
Roel	101
Glenn	102
Gary	103
Michael	104
Jeff	105
Lynn	106
Kate	107
Patrice	108
Pam	109

Populating Image Items

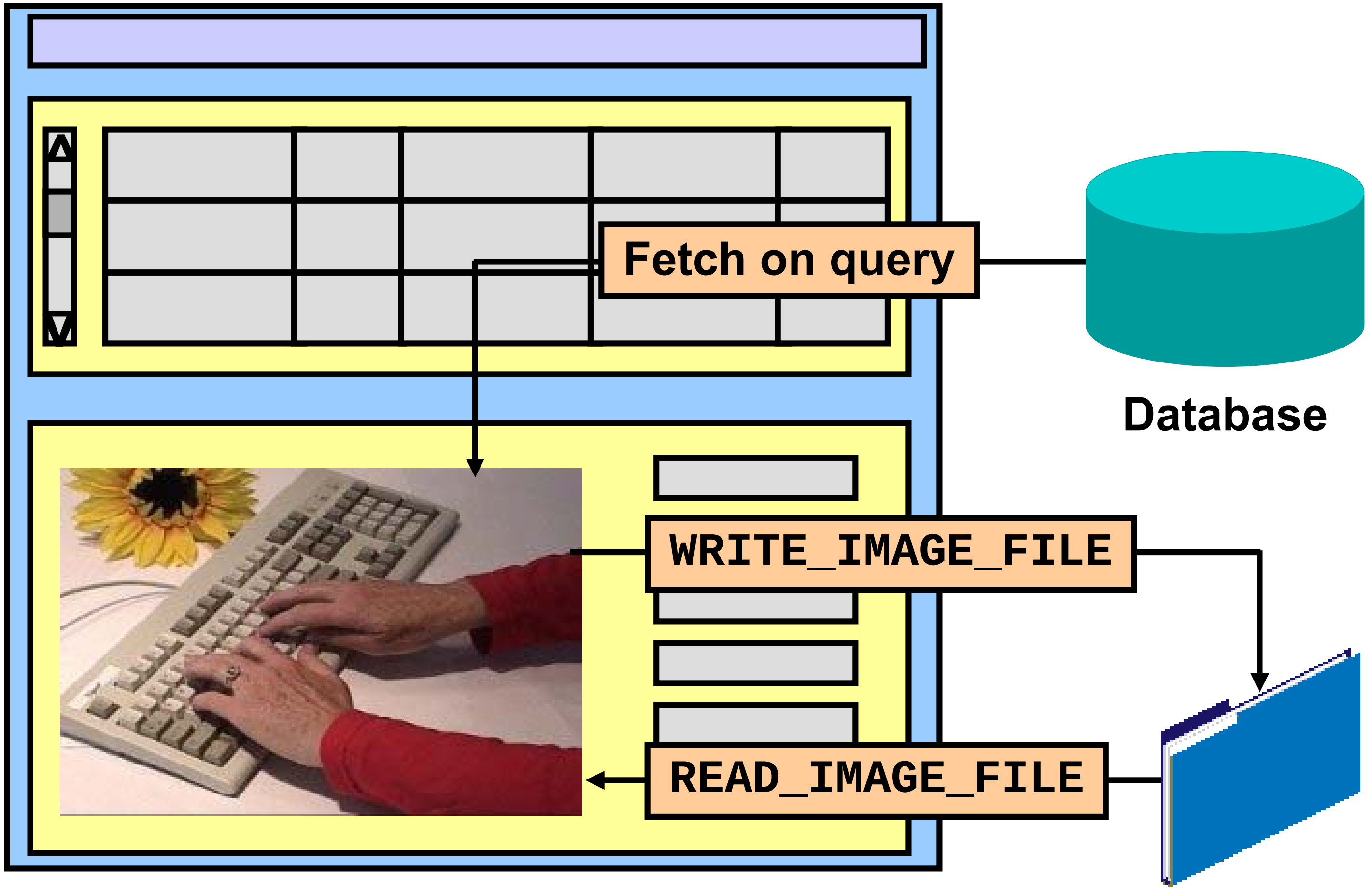


Image file
(in the application server file system)

Loading the Right Image

```
READ_IMAGE_FILE
   (TO_CHAR(:ORDER_ITEMS.product_id)||’.JPG’,
   ’JPEG’,’ORDER_ITEMS.product_image’ );
```

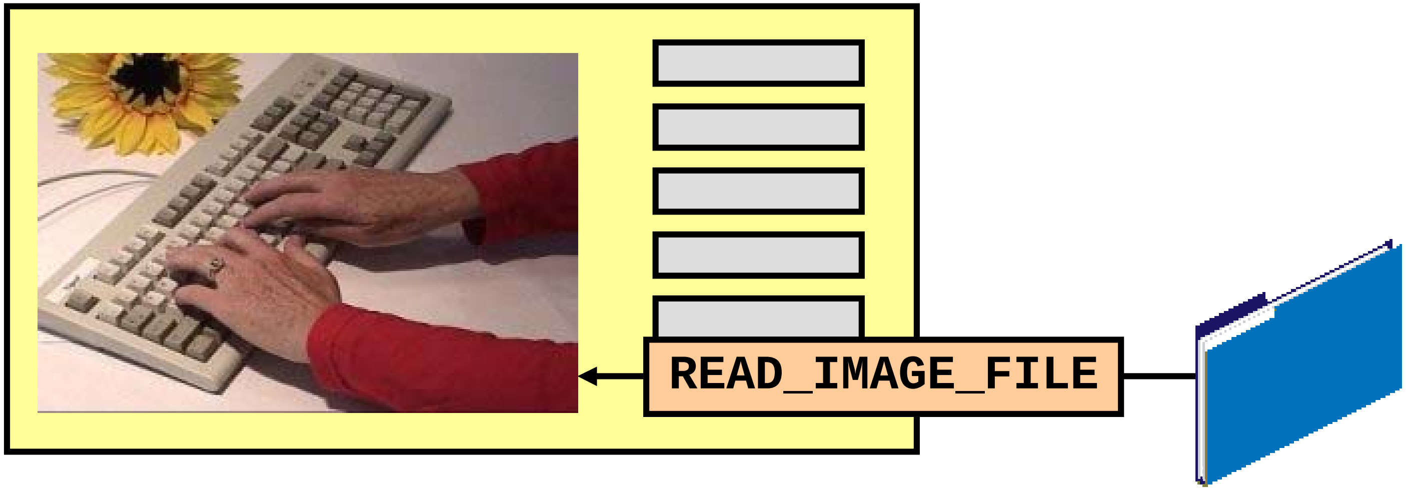
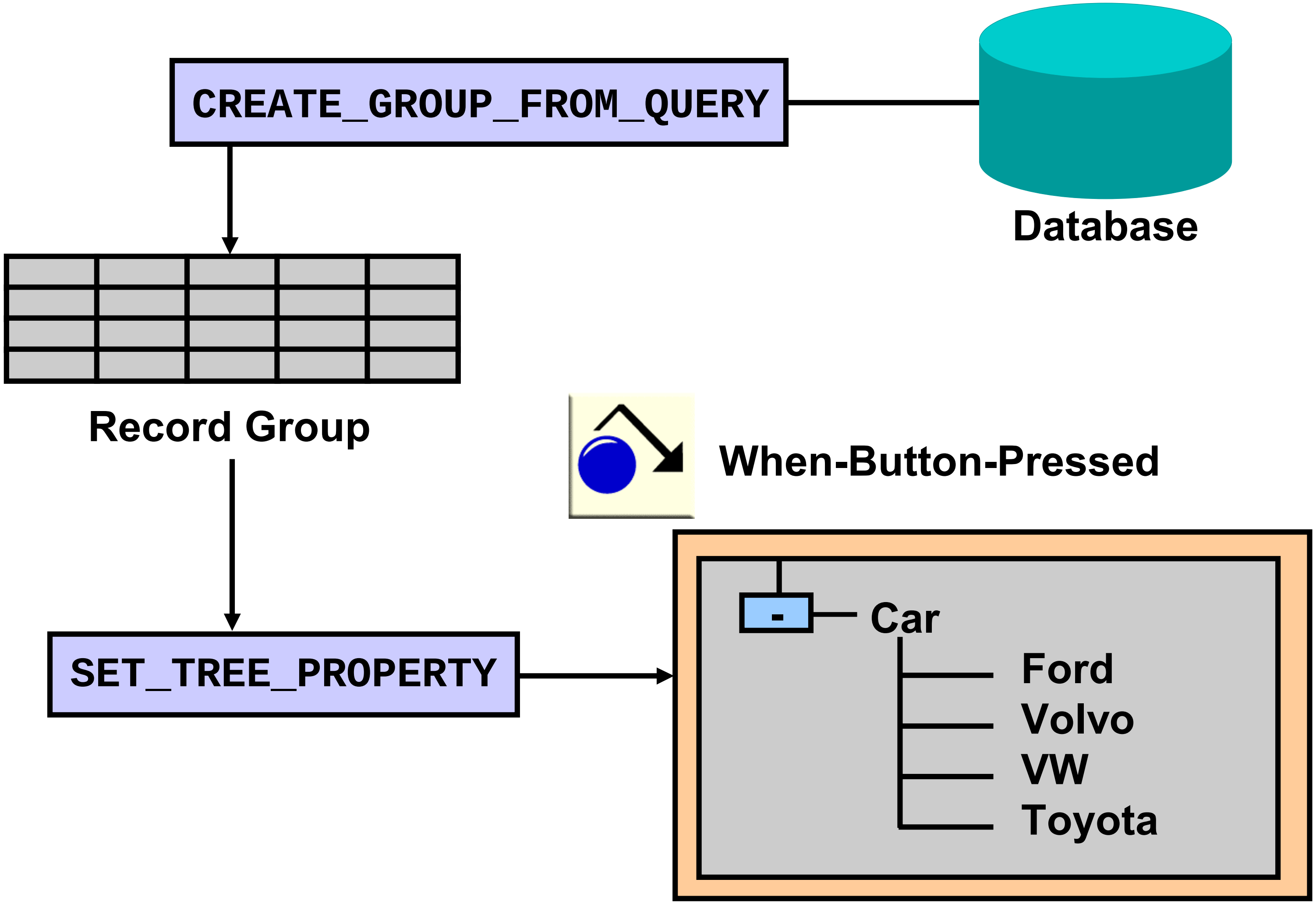


Image file
in the application server file system

Populating Hierarchical Trees



CREATE_GROUP_FROM_QUERY

Database

Record Group

When-Button-Pressed

SET_TREE_PROPERTY

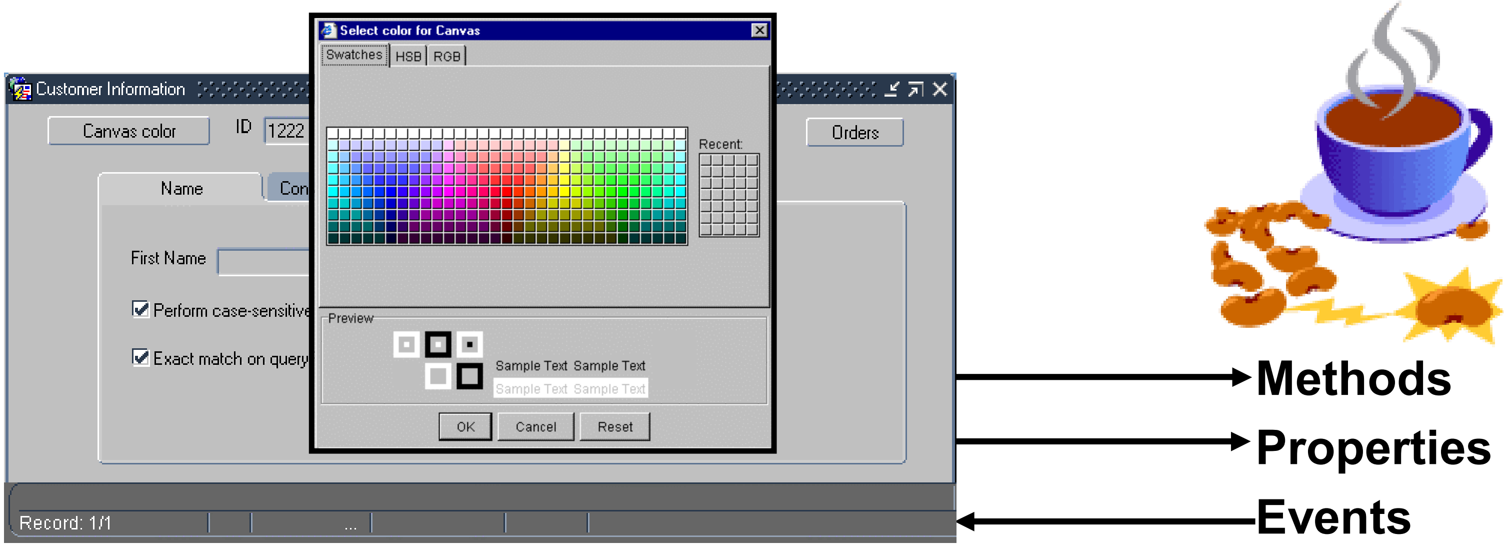
Displaying Hierarchical Trees

When-Button-Pressed

```
rg_emps :=  create_group_from_query('rg_emps',
  'select 1, level, last_name, NULL,
  to_char(employee_id) ' ||
  'from employees ' ||
  'connect by prior employee_id = manager_id '||
  'start with job_id = ''AD_PRES''');

v_ignore := populate_group(rg_emps);

ftree.set_tree_property('block4.tree5',
            ftree.record_group, rg_emps);
```

Interacting with JavaBeans

- Tell Forms about the bean: Register
- Communication from Forms to JavaBean:
  - Invoke Methods
  - Get/Set Properties
- Communication from JavaBean to Forms: Events



ID 1222

☑ Perform case-sensitive

☑ Exact match on query

Record: 1/1

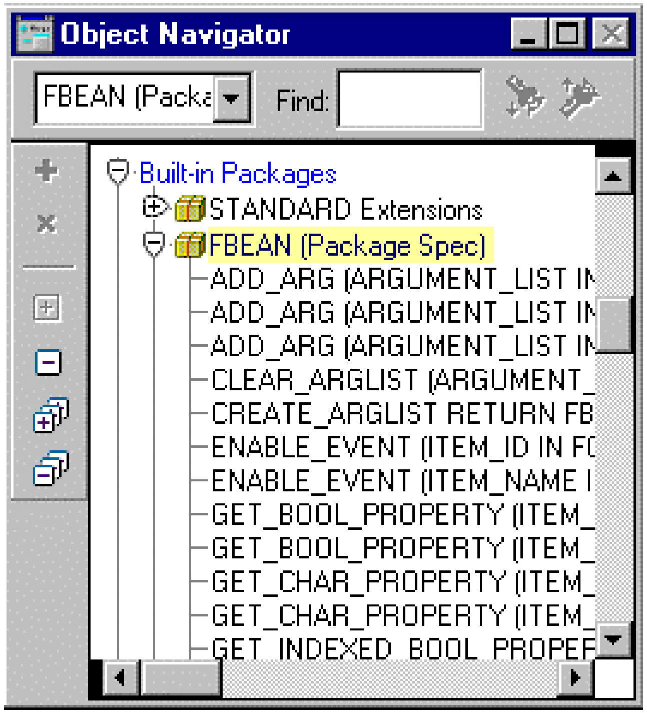
Methods

Properties

Events

Interacting with JavaBeans

The FBEAN package provides built-ins to:



ADD_ARG (ARGUMENT_LIST IN
ADD_ARG (ARGUMENT_LIST IN
ADD_ARG (ARGUMENT_LIST IN
CLEAR_ARGLIST (ARGUMENT_
CREATE_ARGLIST RETURN FB
ENABLE_EVENT (ITEM_ID IN FC
ENABLE_EVENT (ITEM_NAME I
GET_BOOL_PROPERTY (ITEM_
GET_BOOL_PROPERTY (ITEM_
GET_CHAR_PROPERTY (ITEM_
GET_CHAR_PROPERTY (ITEM_
GET_INDEXED_BOOL_PROPEF

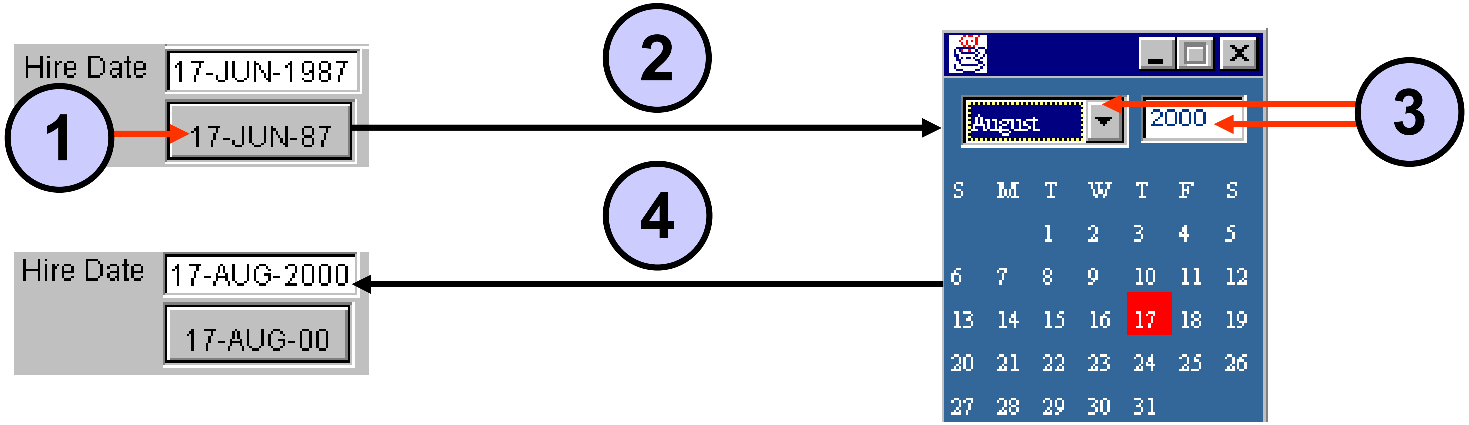
- Register the bean
- Invoke methods of the bean
- Get and set properties on the bean
- Subscribe to bean events

Interacting with JavaBeans

- Register a listener for the event:
  ```
  FBEAN.ENABLE_EVENT('MyBeanArea',1,'mouseListener'
  , true);
  ```
- When an event occurs on the bean:
  - The When-Custom-Item-Event trigger fires.
  - The name and information are sent to Forms in:

    `:SYSTEM.CUSTOM_ITEM_EVENT`

    `:SYSTEM.CUSTOM_ITEM_EVENT_PARAMETERS`



Hire Date 17-JUN-1987

17-JUN-87

August 2000

S	M	T	W	T	F	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

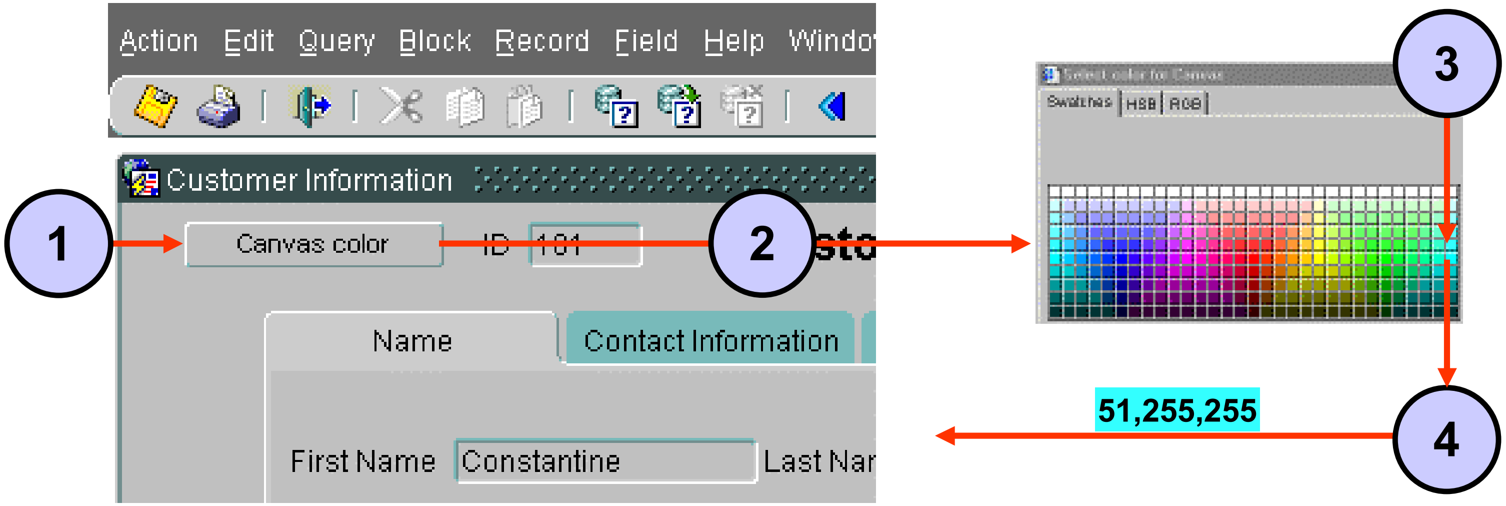
Hire Date 17-AUG-2000

17-AUG-00

Interacting with JavaBeans

The JavaBean may:

- Not have a visible component
- Not communicate via events
- Return a value to the form when invoked (use like a function)



ID 101

First Name Constantine

Last Nar

Summary

In this lesson, you should have learned that:

- You can use triggers to supplement the functionality of:
  - Input items:
    When-[Checkbox | Radio]-Changed
    When-List-[Changed | Activated]
  - Noninput items:
    When-Button-Pressed
    When-Image-[Pressed | Activated]
    When-Tree-Node-[Activated | Expanded | Selected]
    When-Custom-Item-Event

Summary

- You can call useful built-ins from triggers:
  - `CHECKBOX_CHECKED`
  - `[ADD | DELETE]_LIST_ELEMENT`
  - `SHOW_LOV`
  - `[READ | WRITE]_IMAGE_FILE`
  - `FTREE: POPULATE_TREE, ADD_TREE_DATA, [GET | SET]_TREE_PROPERTY`
  - `FBEAN: [GET | SET]_PROPERTY, INVOKE, REGISTER_BEAN, ENABLE_EVENT`

Practice 16 Overview

This practice covers the following topics:

- Writing a trigger to check whether the customer's credit limit has been exceeded
- Creating a toolbar button to display and hide product images
- Coding a button to enable users to choose a canvas color for a form

17

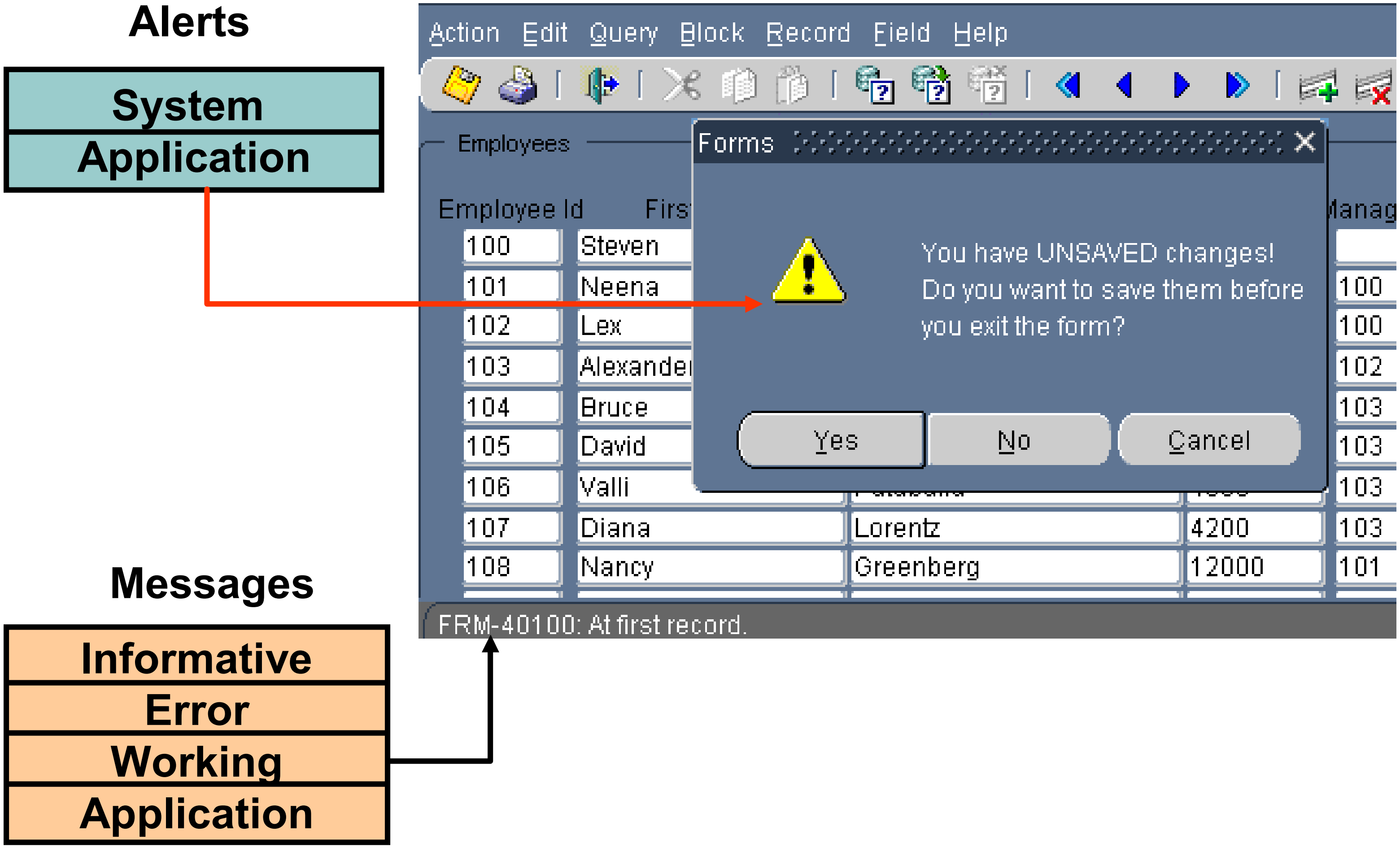
Run Time Messages and Alerts

Objectives

After completing this lesson, you should be able to do the following:

- Describe the default messaging behavior of a form
- Handle run-time failure of built-in subprograms
- Identify the different types of Forms messages
- Control system messages
- Create and control alerts
- Handle database server errors

Run-Time Messages and Alerts Overview



Employee Id	Firs			Manag
100	Steven			
101	Neena			100
102	Lex			100
103	Alexander			102
104	Bruce			103
105	David			103
106	Valli			103
107	Diana	Lorentz	4200	103
108	Nancy	Greenberg	12000	101

You have UNSAVED changes!
Do you want to save them before
you exit the form?

FRM-40100: At first record.

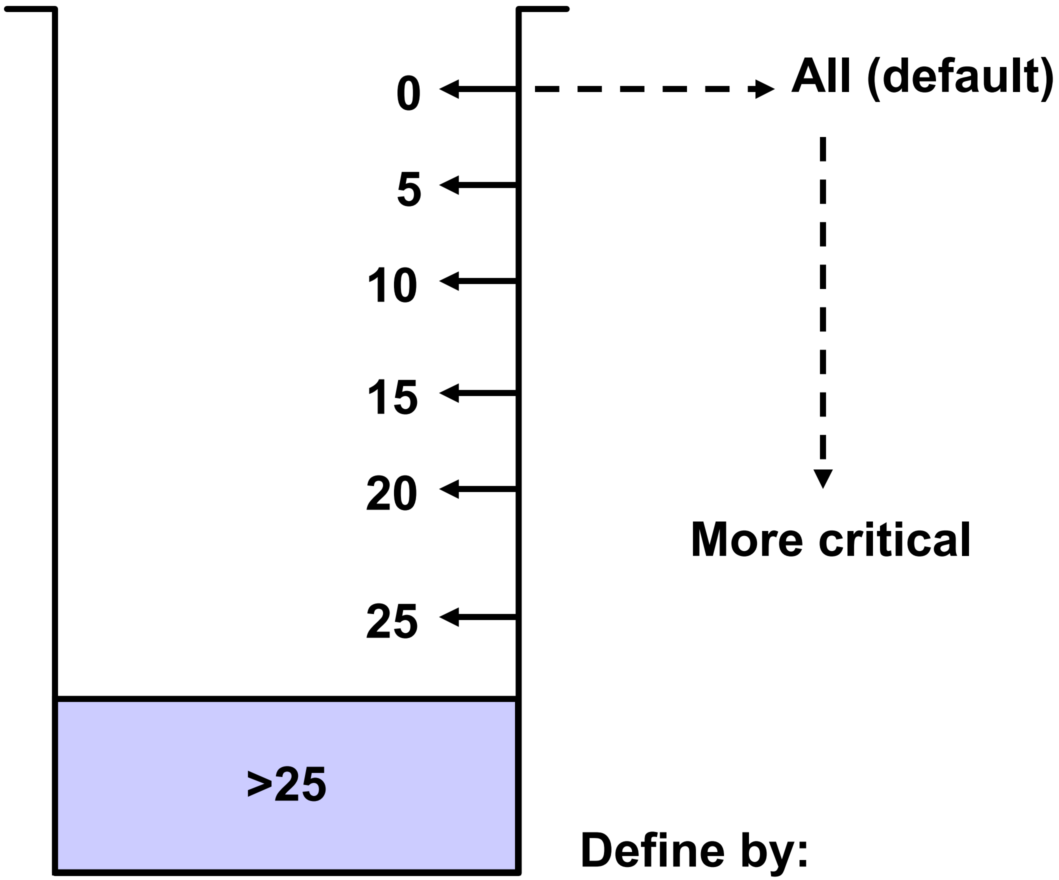
Detecting Run-Time Errors

- `FORM_SUCCESS`
  - TRUE: Action successful
  - FALSE: Error/Fatal error occurred
- `FORM_FAILURE`
  - TRUE: A nonfatal error occurred
  - FALSE: Action successful or a fatal error occurred
- `FORM_FATAL`
  - TRUE: A fatal error occurred
  - FALSE: Action successful or a nonfatal error occurred

Errors and Built-Ins

- Built-In failure does not cause an exception.
- Test built-in success with FORM_SUCCESS function.
  `IF FORM_SUCCESS THEN . . .`
  OR `IF NOT FORM_SUCCESS THEN . . .`
- What went wrong?
  - ERROR_CODE, ERROR_TEXT, ERROR_TYPE
  - MESSAGE_CODE, MESSAGE_TEXT, MESSAGE_TYPE

Message Severity Levels



Define by:

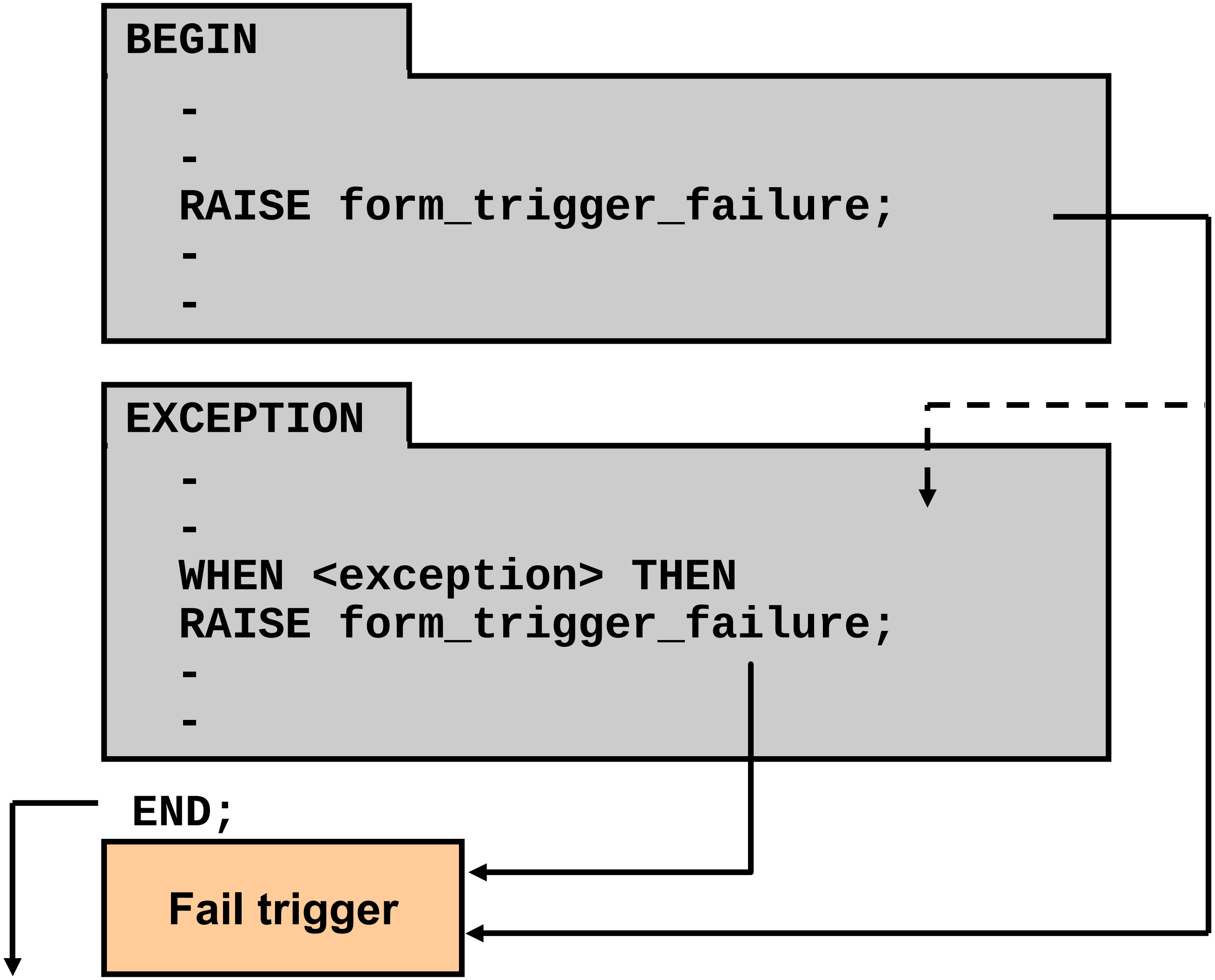
`:SYSTEM.MESSAGE_LEVEL`

Suppressing Messages

```
:SYSTEM.MESSAGE_LEVEL := ’5’;
UP;
IF NOT FORM_SUCCESS THEN
  MESSAGE(’Already at the first Order’);
END IF;
:SYSTEM.MESSAGE_LEVEL := ’0’;
```

```
:SYSTEM.SUPPRESS_WORKING := ’TRUE’;
```

The FORM_TRIGGER_FAILURE Exception



```
BEGIN
   -
   -
   RAISE form_trigger_failure;
   -
   -
EXCEPTION
   -
   -
   WHEN <exception> THEN
   RAISE form_trigger_failure;
   -
   -
 END;
```

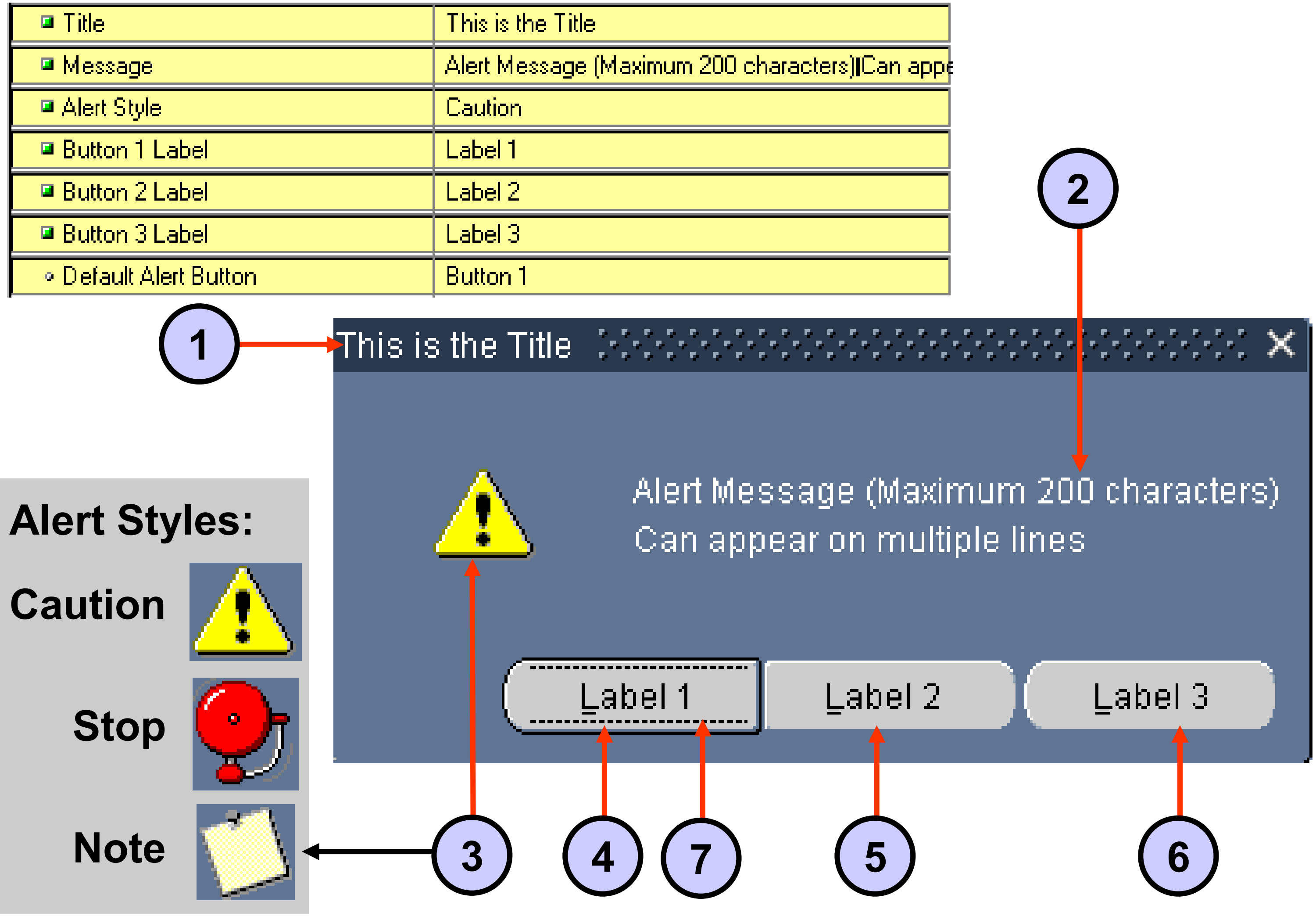
Triggers for Intercepting System Messages

- On-Error:
  - Fires when a system error message is issued
  - Is used to trap Forms and Oracle Server errors, and to customize error messages
- On-Message:
  - Fires when an informative system message is issued
  - Is used to suppress or customize specific messages

Handling Informative Messages

- On-Message trigger
- Built-in functions:
  - `MESSAGE_CODE`
  - `MESSAGE_TEXT`
  - `MESSAGE_TYPE`

Setting Alert Properties



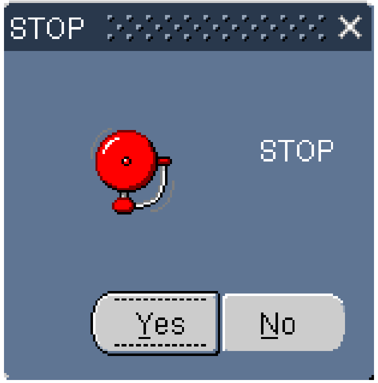
■ Title	This is the Title
■ Message	Alert Message (Maximum 200 characters)|Can appe
■ Alert Style	Caution
■ Button 1 Label	Label 1
■ Button 2 Label	Label 2
■ Button 3 Label	Label 3
◦ Default Alert Button	Button 1

This is the Title

Alert Message (Maximum 200 characters)
Can appear on multiple lines

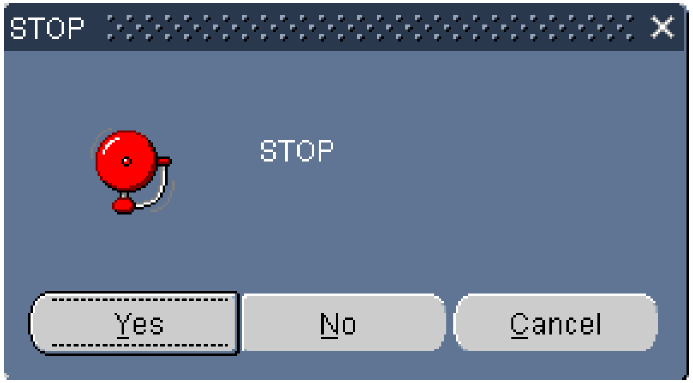
Planning Alerts

Yes/No questions

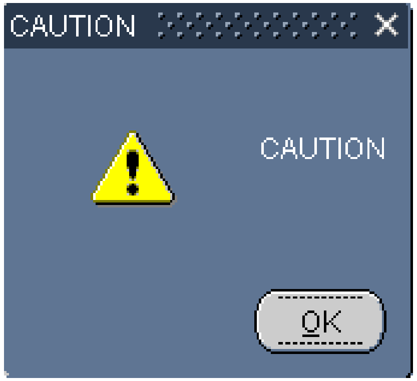


STOP

Yes/No/Cancel questions

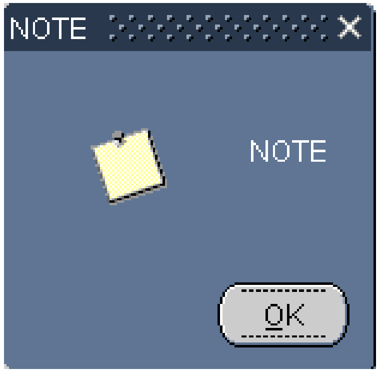


STOP



CAUTION

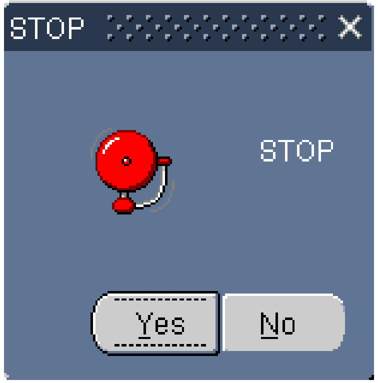
Caution messages

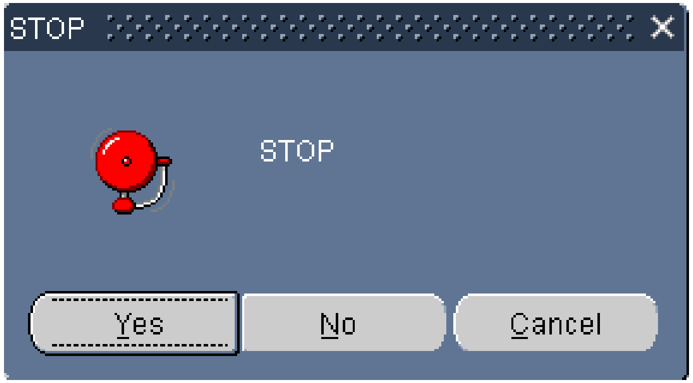


NOTE

Informative messages

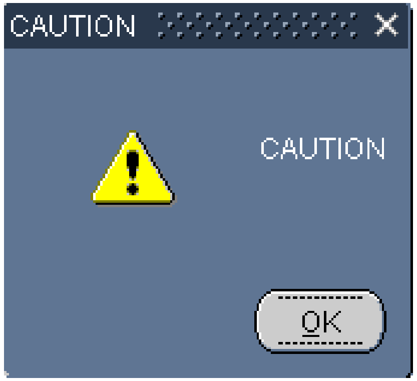
Controlling Alerts

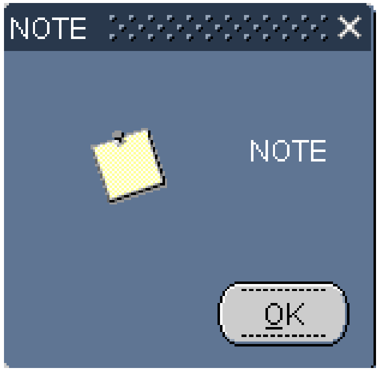




SET_ALERT_PROPERTY

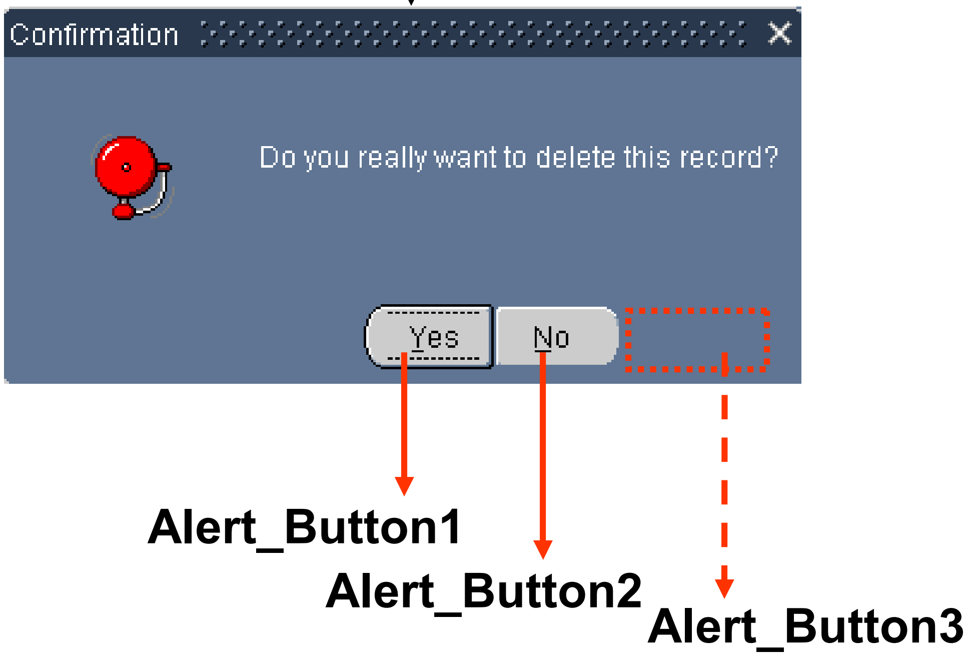
SET_ALERT_BUTTON_PROPERTY





SHOW_ALERT Function

```
IF SHOW_ALERT('del_Check')=ALERT_BUTTON1 THEN
. . .
```



Confirmation

Do you really want to delete this record?

Alert_Button1

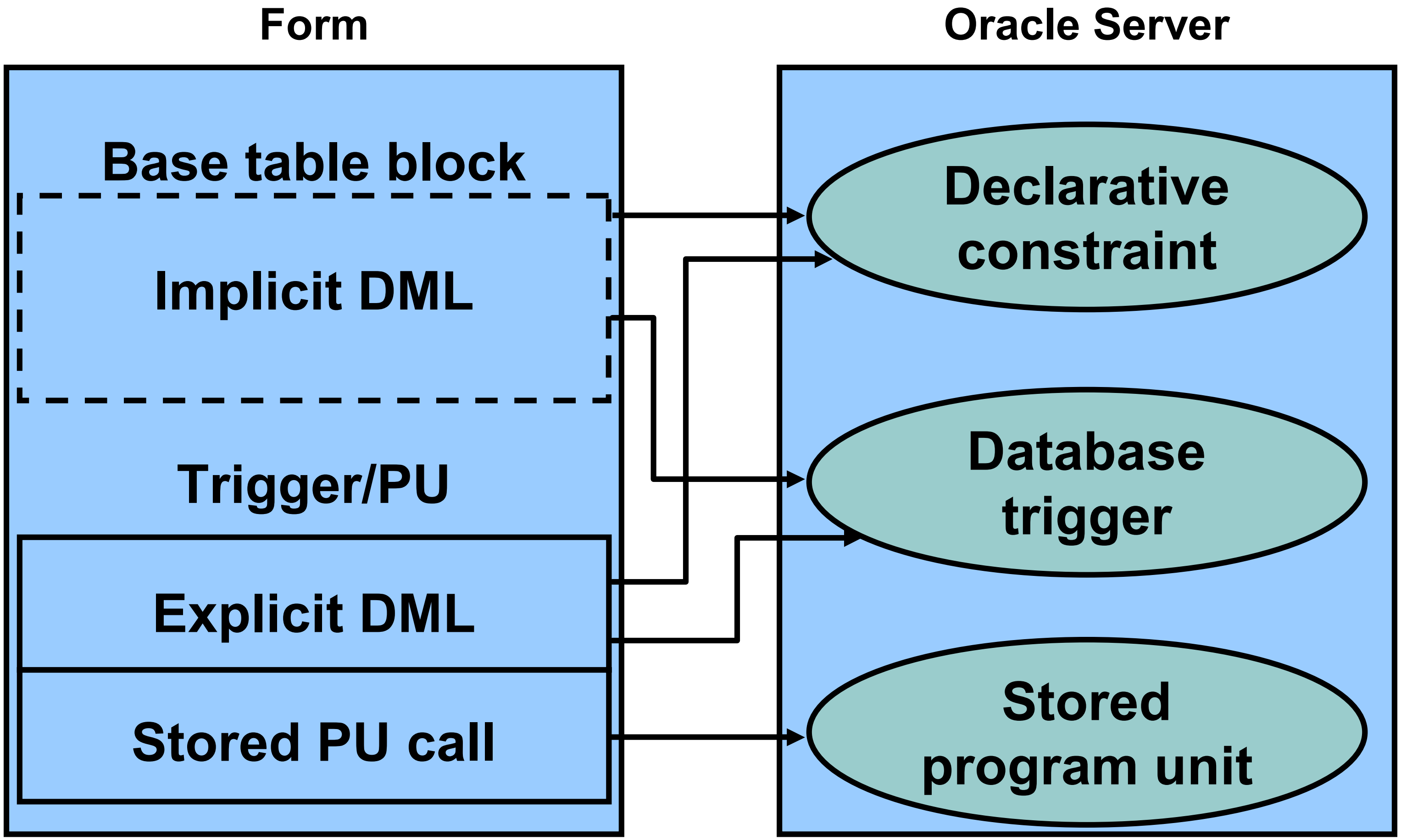
Alert_Button2

Alert_Button3

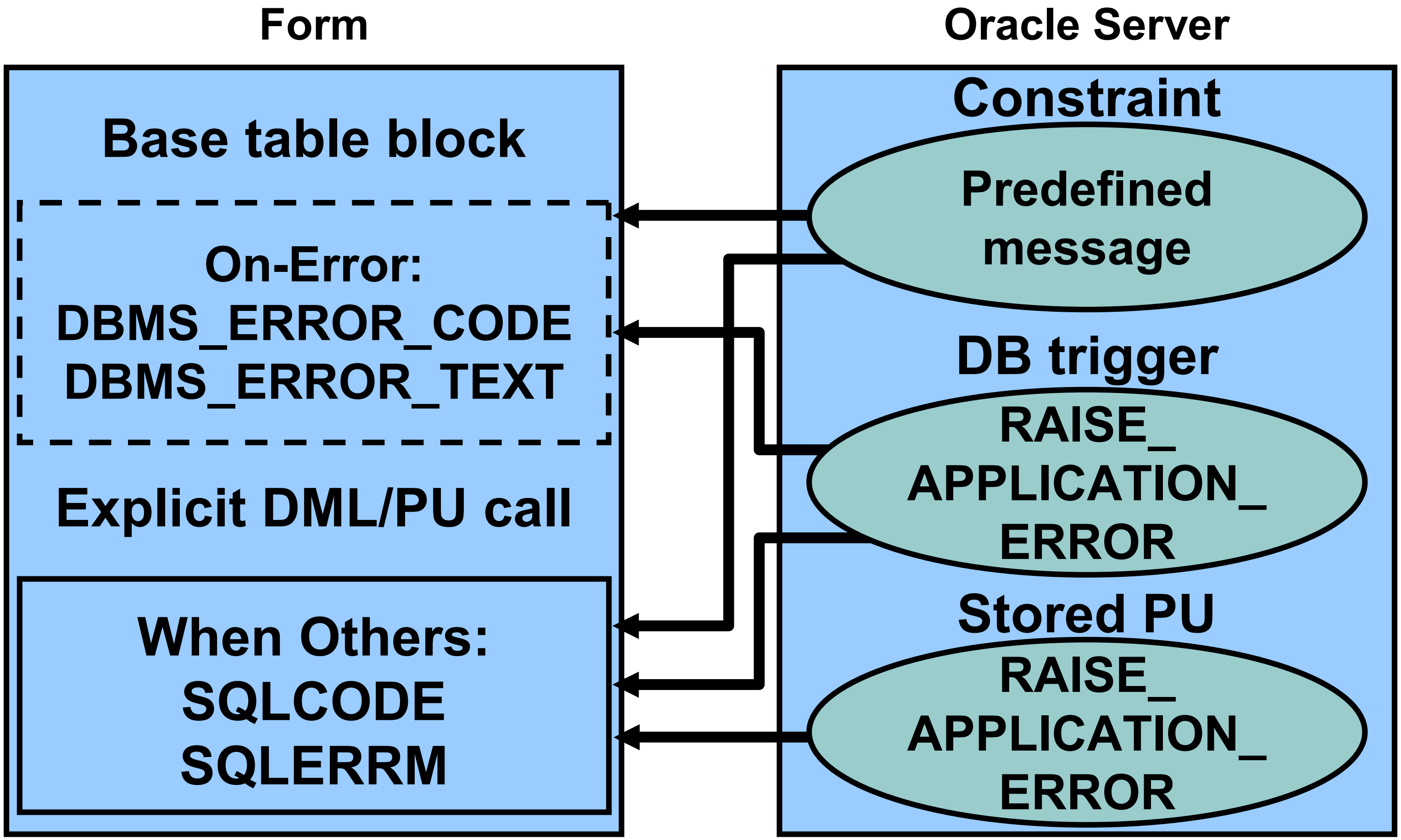
Directing Errors to an Alert

```
PROCEDURE Alert_On_Failure IS
  n NUMBER;
BEGIN
  SET_ALERT_PROPERTY('error_alert',
    ALERT_MESSAGE_TEXT,ERROR_TYPE||
    '-'||TO_CHAR(ERROR_CODE)||
     ': '||ERROR_TEXT);
  n := SHOW_ALERT('error_alert');
END;
```

Causes of Oracle Server Errors



Trapping Server Errors



Summary

In this lesson, you should have learned that:

- Forms displays messages at run time to inform the operator of events that occur in the session.
- You can use `FORM_SUCCESS` to test for run-time failure of built-ins.
- There are four types of Forms messages:
  - Informative
  - Error
  - Working
  - Application

Summary

- You can control system messages with built-ins and triggers:
  - `MESSAGE_LEVEL`
  - `SUPPRESS_WORKING`
  - On-[Error | Message] triggers
  - `[ERROR | MESSAGE]_[CODE | TEXT | TYPE]`
- Types of alerts: Stop, Caution, Note
- Alert built-ins:
  - `SHOW_ALERT`
  - `SET_ALERT_PROPERTY`
  - `SET_ALERT_BUTTON_PROPERTY`

Summary

- Handle database server errors:
  - Implicit DML: Use `DBMS_ERROR_CODE` and `DBMS_ERROR_TEXT` in On-Error trigger
  - Explicit DML: Use `SQLCODE` and `SQLERRM` in `WHEN OTHERS` exception handler

Practice 17 Overview

This practice covers the following topics:

- Using an alert to inform the operator that the customer’s credit limit has been exceeded
- Using a generic alert to ask the operator to confirm that the form should terminate

18

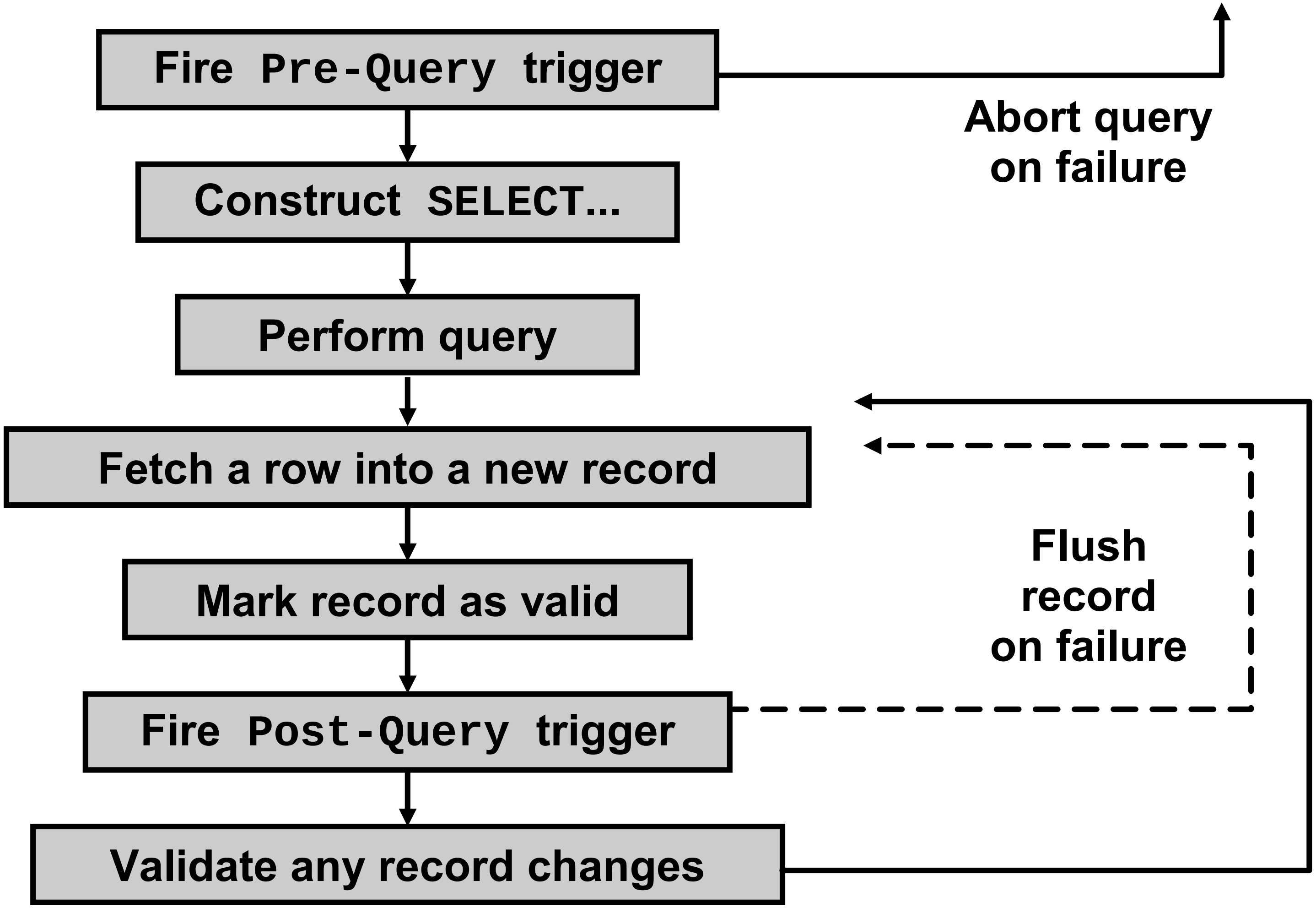
Query Triggers

Objectives

After completing this lesson, you should be able to do the following:

- Explain the processes involved in querying a data block
- Describe query triggers and their scope
- Write triggers to screen query conditions
- Write triggers to supplement query results
- Control trigger action based on the form’s query status

Query Processing Overview



SELECT Statements Issued During Query Processing

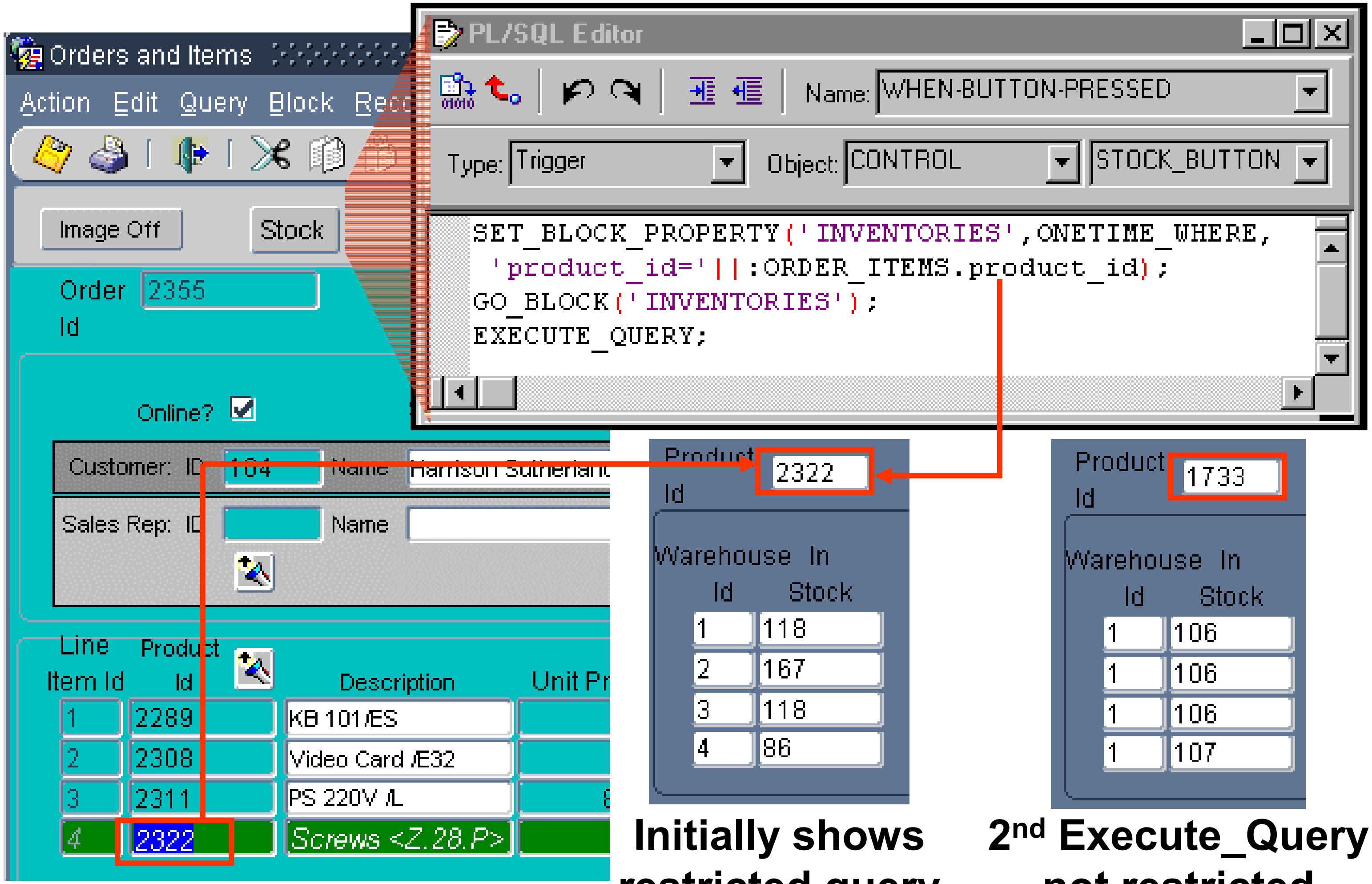
```
SELECT     base_column, ..., ROWID
INTO       :base_item, ..., :ROWID
FROM       base_table
WHERE      (default_where_clause OR
           onetime_where_clause)
 AND       (example_record_conditions)
 AND       (query_where_conditions)
ORDER BY   default_order_by_clause |
           query_where_order_by
```

Slightly different for COUNT

WHERE Clause

- Four sources for the WHERE clause:
  - WHERE Clause block property
  - ONETIME_WHERE block property
  - Example Record
  - Query/Where dialog box
- WHERE clauses are combined by the AND operator, except that WHERE and ONETIME_WHERE are combined with the OR operator.

ONETIME_WHERE Property



Name: WHEN-BUTTON-PRESSED

Type: Trigger Object: CONTROL STOCK_BUTTON

```
SET_BLOCK_PROPERTY('INVENTORIES',ONETIME_WHERE,
 'product_id='||:ORDER_ITEMS.product_id);
GO_BLOCK('INVENTORIES');
EXECUTE_QUERY;
```

Order Id 2355

Online? ☑

Customer: ID 104 Name Harrison Sutherland

Sales Rep: ID Name

Line Item Id	Product Id	Description	Unit Pr
1	2289	KB 101/ES	
2	2308	Video Card /E32	
3	2311	PS 220V /L	8
4	2322	Screws <Z.28.P>	

Product Id 2322

Warehouse Id	In Stock
1	118
2	167
3	118
4	86

Product Id 1733

Warehouse Id	In Stock
1	106
1	106
1	106
1	107

Initially shows restricted query

2nd Execute_Query not restricted

ORDER BY Clause

- Two sources for the ORDER BY clause:
  - ORDER BY Clause block property
  - Query/Where dialog box
- Second source for ORDER BY clause overrides the first one

Writing Query Triggers: Pre-Query Trigger

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

```
IF   TO_CHAR(:ORDERS.ORDER_ID)||
     TO_CHAR(:ORDERS.CUSTOMER_ID)
IS NULL THEN
     MESSAGE(’You must query by
     Order ID or Customer ID’);
     RAISE form_trigger_failure;
END IF;
```

Writing Query Triggers: Post-Query Trigger

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

```
SELECT   COUNT(order_id)
INTO     :ORDERS.lineitem_count
FROM     ORDER_ITEMS
WHERE    order_id = :ORDERS.order_id;
```

Writing Query Triggers: Using SELECT Statements in Triggers

- Forms Builder variables are preceded by a colon.
- The query must return one row for success.
- Code exception handlers.
- The `INTO` clause is mandatory, with a variable for each selected column or expression.
- `ORDER BY` is not relevant.

Query Array Processing

- Reduces network traffic
- Enables Query Array processing:
  - Enable Array Processing option
  - Set Query Array Size property
- Query Array Size property
- Query All Records property

Coding Triggers for Enter-Query Mode

- Some triggers may fire in Enter-Query mode.
- Set the Fire in Enter-Query Mode property.
- Test mode during execution with `:SYSTEM.MODE`
  - `NORMAL`
  - `ENTER-QUERY`
  - `QUERY`

Coding Triggers for Enter-Query Mode

- Example

```
IF :SYSTEM.MODE = ’NORMAL’
THEN ENTER_QUERY;
ELSE EXECUTE_QUERY;
END IF;
```

- Some built-ins are illegal.
- Consult online Help.
- You cannot navigate to another record in the current form.

Overriding Default Query Processing

Additional Transactional Triggers for Query Processing

Trigger	Do-the-Right-Thing Built-in
On-Close	
On-Count	COUNT_QUERY
On-Fetch	FETCH_RECORDS
Pre-Select	
On-Select	SELECT_RECORDS
Post-Select	

Overriding Default Query Processing

- On-Fetch continues to fire until:
  - It fires without executing `CREATE_QUERIED_RECORD`.
  - The query is closed by the user or by `ABORT_QUERY`.
  - It raises `FORM_TRIGGER_FAILURE`.
- On-Select replaces open cursor, parse, and execute phases.

Obtaining Query Information at Run Time

- SYSTEM.MODE
- SYSTEM.LAST_QUERY
  - Contains bind variables (ORD_ID = :1) before SELECT_RECORDS
  - Contains actual values (ORD_ID = 102) after SELECT_RECORDS

Obtaining Query Information at Run Time

- `GET_BLOCK_PROPERTY`
  `SET_BLOCK_PROPERTY`
  - Get and set:
    ```
    DEFAULT_WHERE
    ONETIME_WHERE
    ORDER_BY
    QUERY_ALLOWED
    QUERY_HITS
    ```
  - Get only:
    ```
    QUERY_OPTIONS
    RECORDS_TO_FETCH
    ```

Obtaining Query Information at Run Time

- `GET_ITEM_PROPERTY`
- `SET_ITEM_PROPERTY`
  - Get and set:
    `CASE_INSENSITIVE_QUERY`
    `QUERYABLE`
    `QUERY_ONLY`
  - Get only:
    `QUERY_LENGTH`

Summary

In this lesson, you should have learned that:

- Query processing includes the following steps:
  1. Pre-Query trigger fires
  2. SELECT statement constructed
  3. Query performed
  4. Record fetched into block
  5. Record marked Valid
  6. Post-Query trigger fires
  7. Item and record validation if the record has changed (due to a trigger)
  8. Steps 4 through 7 repeat till all fetched

Summary

- The query triggers, which must be defined at block or form level, are:
  - Pre-Query: Use to screen query conditions (set `ONETIME_WHERE` or `DEFAULT_WHERE` properties, or assign values to use as query criteria)
  - Post-Query: Use to supplement query results (populate nonbase table items, perform calculations)
- You can use transactional triggers to override default query processing.
- You can control trigger action based on the form's query status by checking `SYSTEM.MODE` values: `NORMAL`, `ENTER-QUERY`, or `QUERY`

Practice 18 Overview

This practice covers the following topics:

- Populating customer names and sales representative names for each row of the ORDERS block
- Populating descriptions for each row of the ORDER_ITEMS block
- Restricting the query on the INVENTORIES block for only the first query on that block
- Disabling the effects of the Exit button and changing a radio group in Enter-Query mode
- Adding two check boxes to enable case-sensitive and exact match query

19

Validation

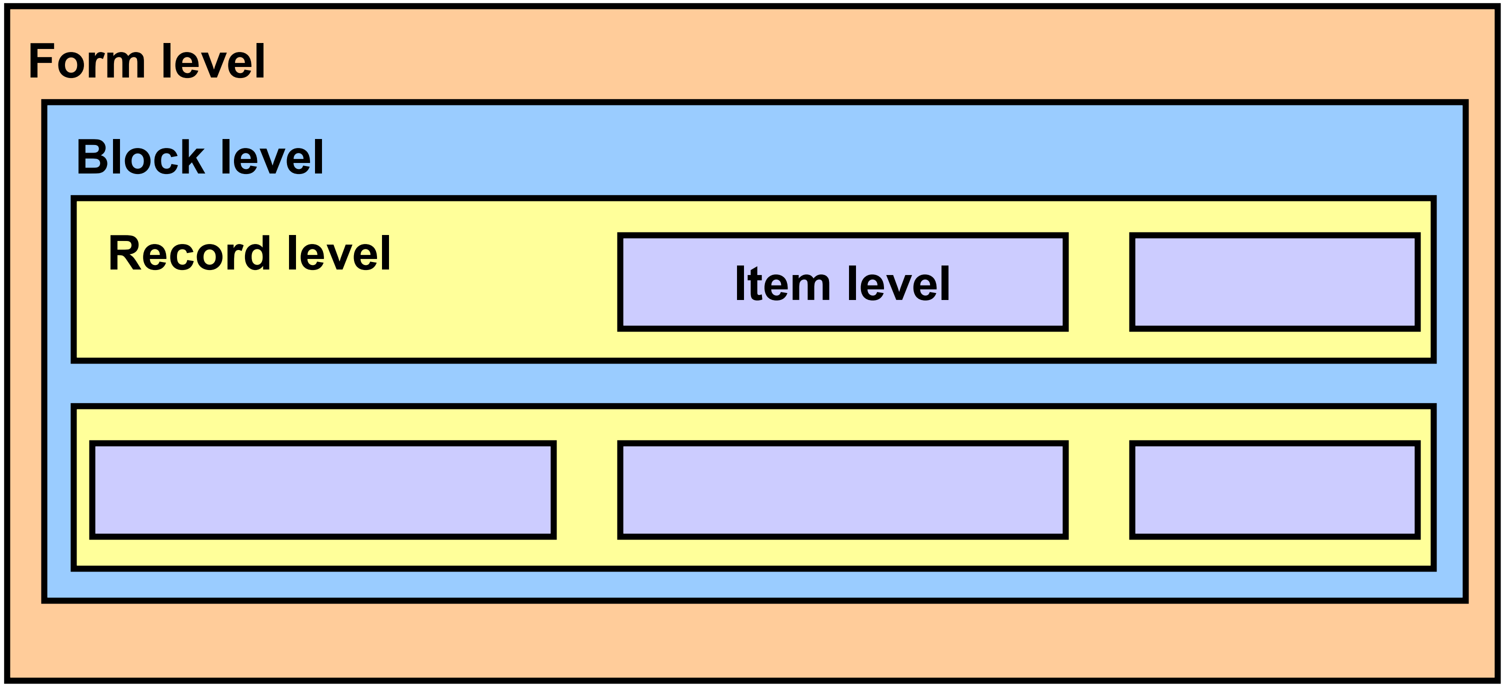
Objectives

After completing this lesson, you should be able to do the following:

- Explain the effects of the validation unit upon a form
- Control validation:
  - Using object properties
  - Using triggers
  - Using Pluggable Java Components
- Describe how Forms tracks validation status
- Control when validation occurs

The Validation Process

Forms validates at the following levels:

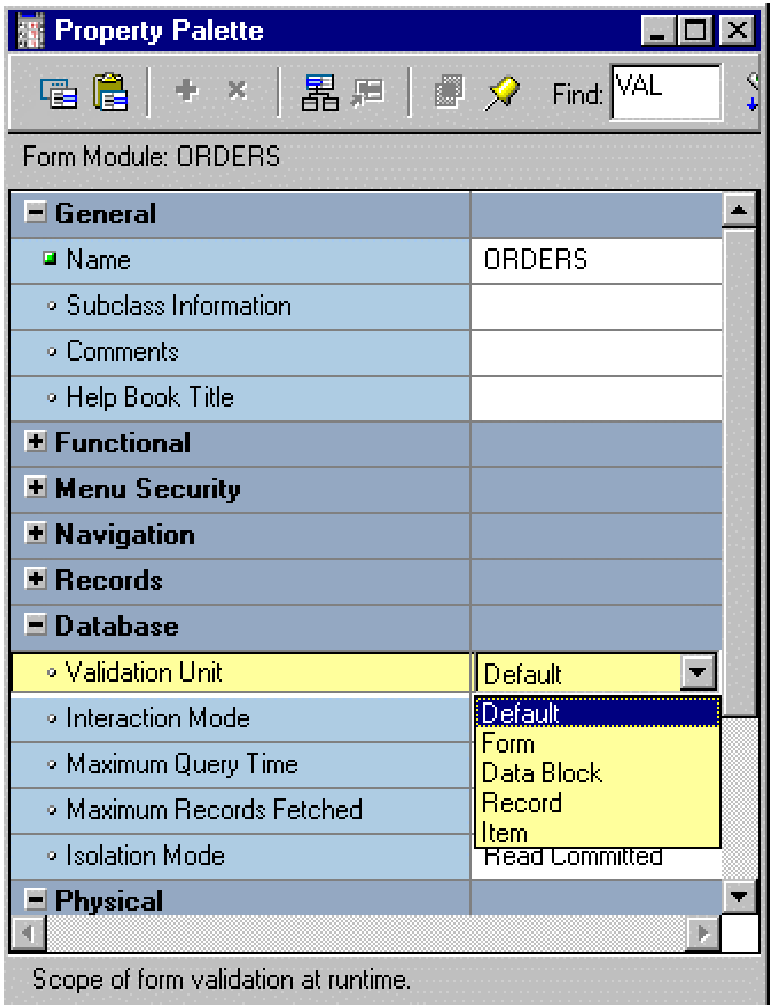


The Validation Process

Validation occurs when:

- `[Enter]` key or ENTER Built-in is obeyed
- Operator or trigger leaves the validation unit (includes a Commit)

Controlling Validation Using Properties: Validation Unit



VAL

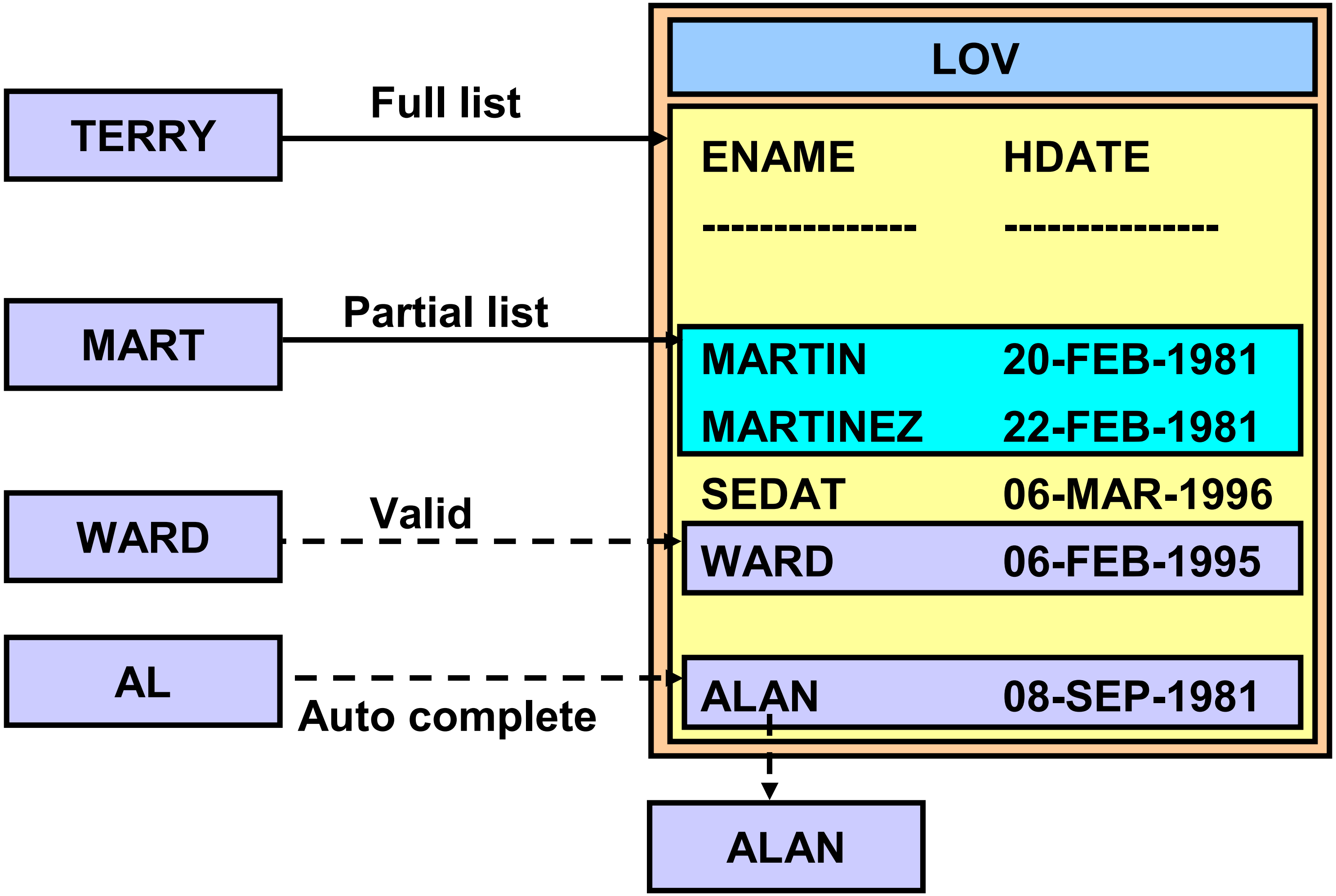
Form Module: ORDERS

General	
Name	ORDERS
Subclass Information	
Comments	
Help Book Title	
Functional	
Menu Security	
Navigation	
Records	
Database	
Validation Unit	Default
Interaction Mode	
Maximum Query Time	
Maximum Records Fetched	
Isolation Mode	Read Committed
Physical	

Default
Form
Data Block
Record
Item

Scope of form validation at runtime.

Controlling Validation Using Properties: Validate from List



TERRY

Full list

MART

Partial list

WARD

Valid

AL

Auto complete

LOV

ENAME	HDATE
---------------	---------------
MARTIN	20-FEB-1981
MARTINEZ	22-FEB-1981
SEDAT	06-MAR-1996
WARD	06-FEB-1995
ALAN	08-SEP-1981

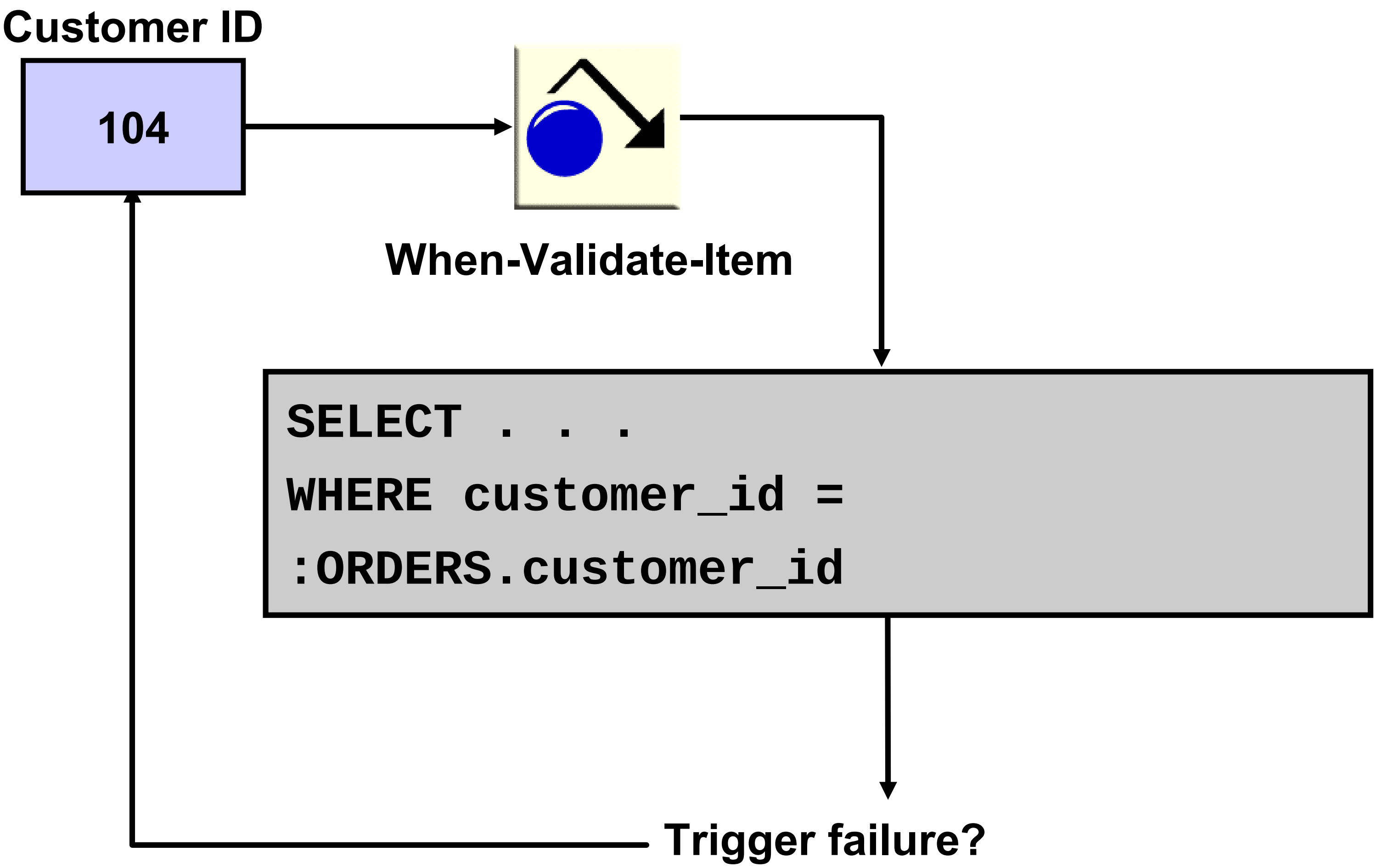
ALAN

Controlling Validation Using Triggers

- Item level:
  When-Validate-Item
- Block level:
  When-Validate-Record

```
IF :ORDERS.order_date > SYSDATE THEN
  MESSAGE(Order Date is later than today!’);
  RAISE form_trigger_failure;
END IF;
```

Example: Validating User Input



Customer ID

104

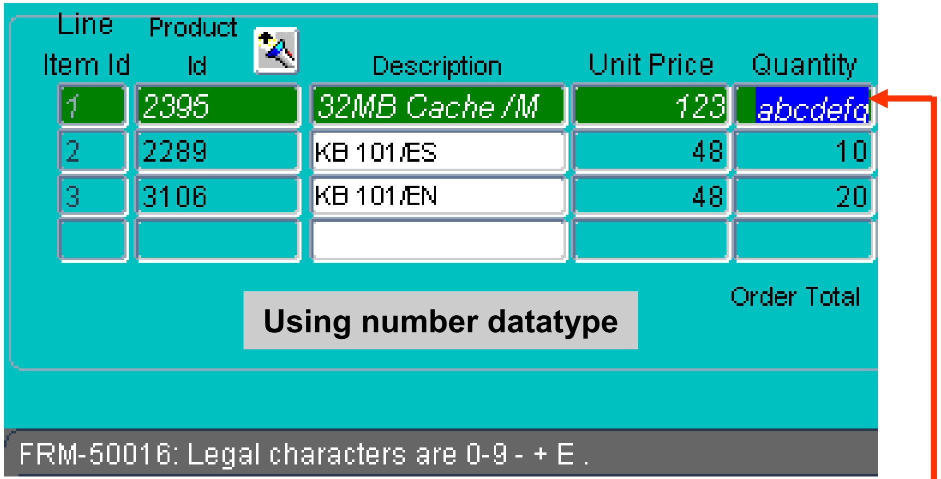
When-Validate-Item

```
SELECT . . .
WHERE customer_id =
:ORDERS.customer_id
```

Trigger failure?

Using Client-Side Validation

- **Forms validation:**
  - **Occurs on middle tier**
  - **Involves network traffic**
- **Client-side validation:**
  - **Improves performance**
  - **Implemented with PJC**

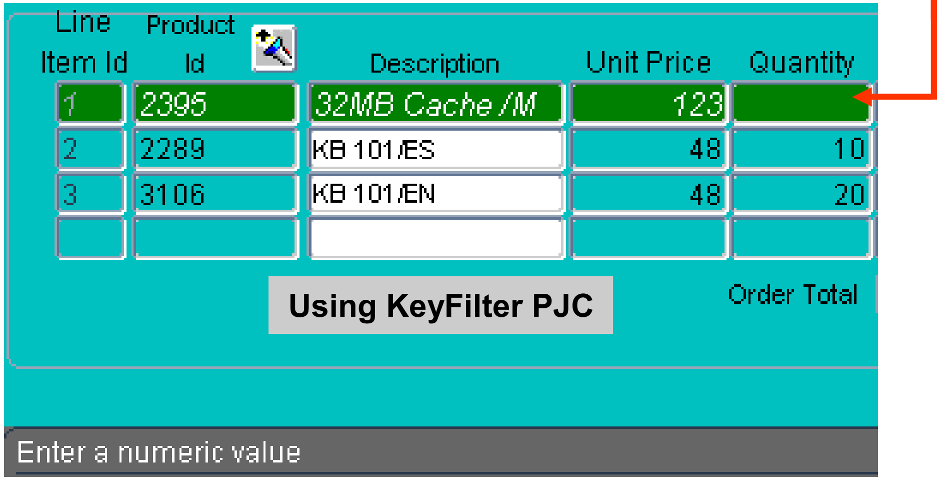


Line Item Id	Product Id	Description	Unit Price	Quantity
1	2395	32MB Cache /M	123	abcdefg
2	2289	KB 101/ES	48	10
3	3106	KB 101/EN	48	20

Using number datatype

FRM-50016: Legal characters are 0-9 - + E .

Attempt to enter alphabetic characters



Line Item Id	Product Id	Description	Unit Price	Quantity
1	2395	32MB Cache /M	123	
2	2289	KB 101/ES	48	10
3	3106	KB 101/EN	48	20

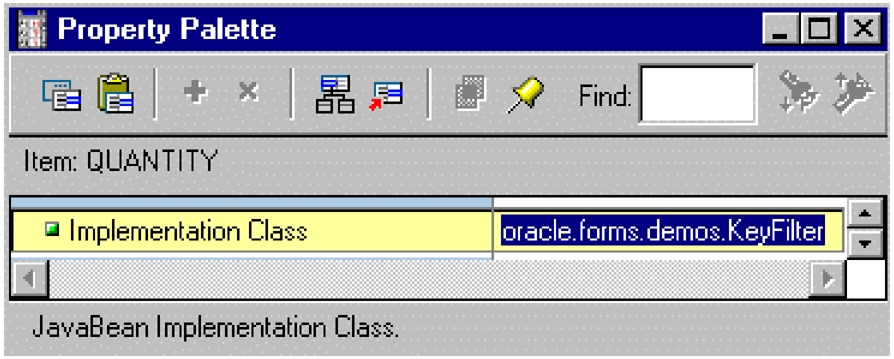
Using KeyFilter PJC

Enter a numeric value

Using Client-Side Validation

To use a PJC:

1. Set the item's Implementation Class property



Item: QUANTITY

Implementation Class: oracle.forms.demos.KeyFilter

JavaBean Implementation Class.

2. Set properties for the PJC

```
SET_CUSTOM_PROPERTY('order_items.quantity',
                1,'FILTER_TYPE','NUMERIC');
```

Tracking Validation Status

- NEW
  - When a record is created
  - Also for Copy Value from Item or Initial Value
- CHANGED
  - When changed by user or trigger
  - When any item in new record is changed
- VALID
  - When validation has been successful
  - After records are fetched from database
  - After a successful post or commit
  - Duplicated record inherits status of source

Controlling When Validation Occurs with Built-Ins

- CLEAR_BLOCK, CLEAR_FORM, EXIT_FORM
- ENTER
- SET_FORM_PROPERTY
  - (..., VALIDATION)
  - (..., VALIDATION_UNIT)
- ITEM_IS_VALID item property
- VALIDATE (scope)

Summary

In this lesson, you should have learned that:

- The validation unit specifies how much data is entered before validation occurs.
- You can control validation using:
  - Object properties: Validation Unit (form); Validate from List (item)
  - Triggers: When-Validate-Item (item level); When-Validate-Record (block level)
  - Pluggable Java Components for client-side validation

Summary

- Forms tracks validation status of items and records, which are either `NEW`, `CHANGED`, or `VALID`.
- You can use built-ins to control when validation occurs:
  - `CLEAR_BLOCK`
  - `CLEAR_FORM`
  - `EXIT_FORM`
  - `ENTER`
  - `ITEM_IS_VALID`
  - `VALIDATE`

Practice 19 Overview

This practice covers the following topics:

- Validating the Sales Representative item value by using an LOV
- Writing a validation trigger to check that online orders are CREDIT orders
- Populating customer names, sales representative names, and IDs when a customer ID is changed
- Writing a validation trigger to populate the name and the price of the product when the product ID is changed
- Restricting user input to numeric characters using a Pluggable Java Component

20

Navigation

Objectives

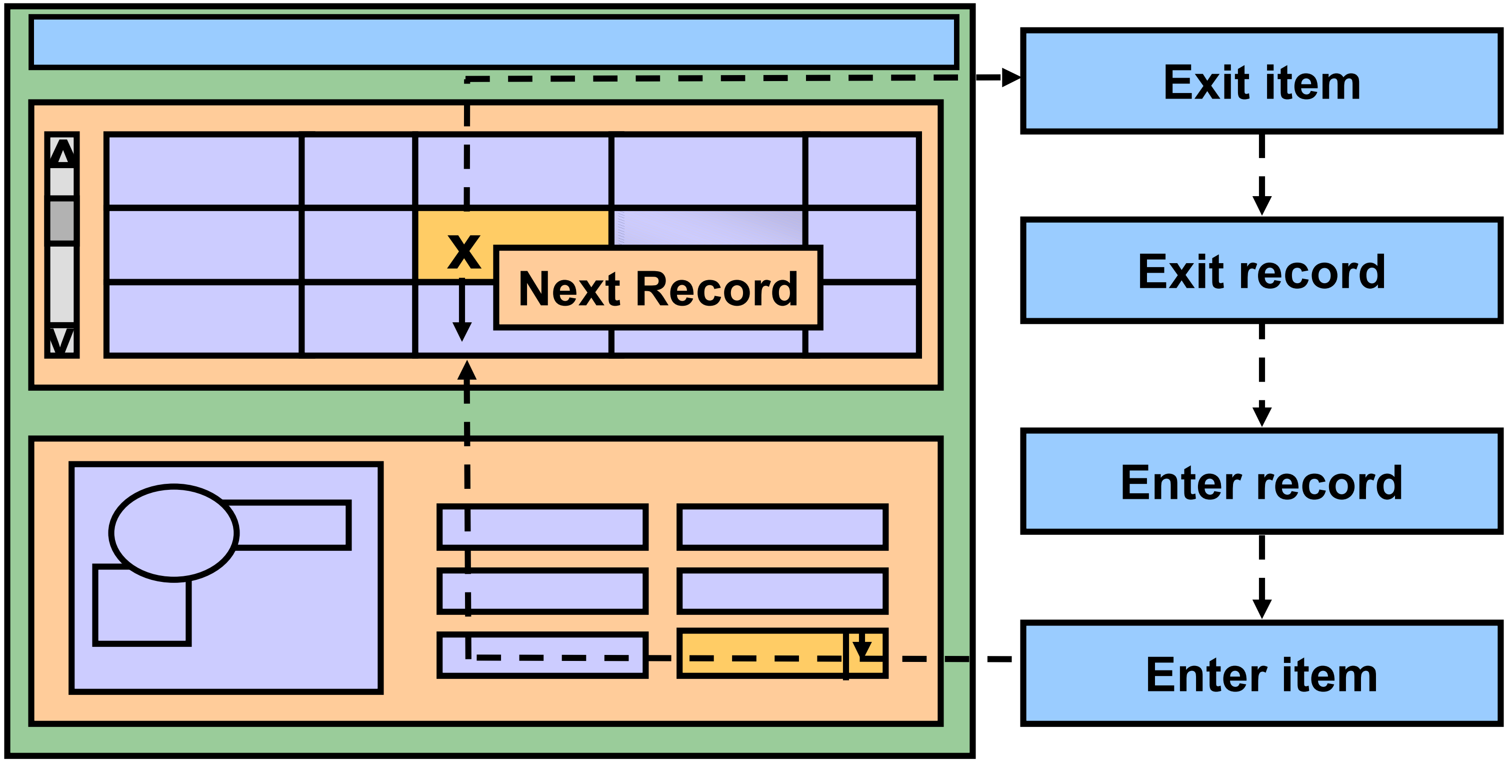
After completing this lesson, you should be able to do the following:

- Distinguish between internal and external navigation
- Control navigation with properties
- Describe and use navigation triggers to control navigation
- Use navigation built-ins in triggers

Navigation Overview

- What is the navigational unit?
  - Outside the form
  - Form
  - Block
  - Record
  - Item
- Entering and leaving objects
- What happens if navigation fails?

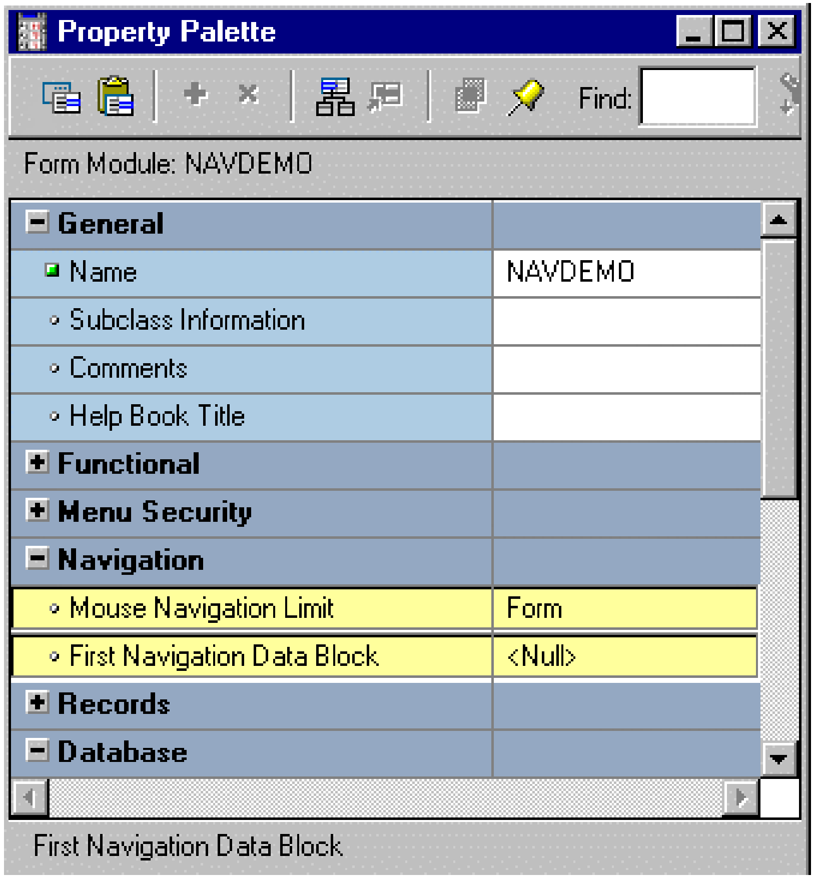
Understanding Internal Navigation



Using Object Properties to Control Navigation

- Block
  - Navigation Style
  - Previous Navigation Data Block
  - Next Navigation Data Block
- Item
  - Enabled
  - Keyboard Navigable
  - Mouse Navigate
  - Previous Navigation Item
  - Next Navigation Item

Using Object Properties to Control Navigation



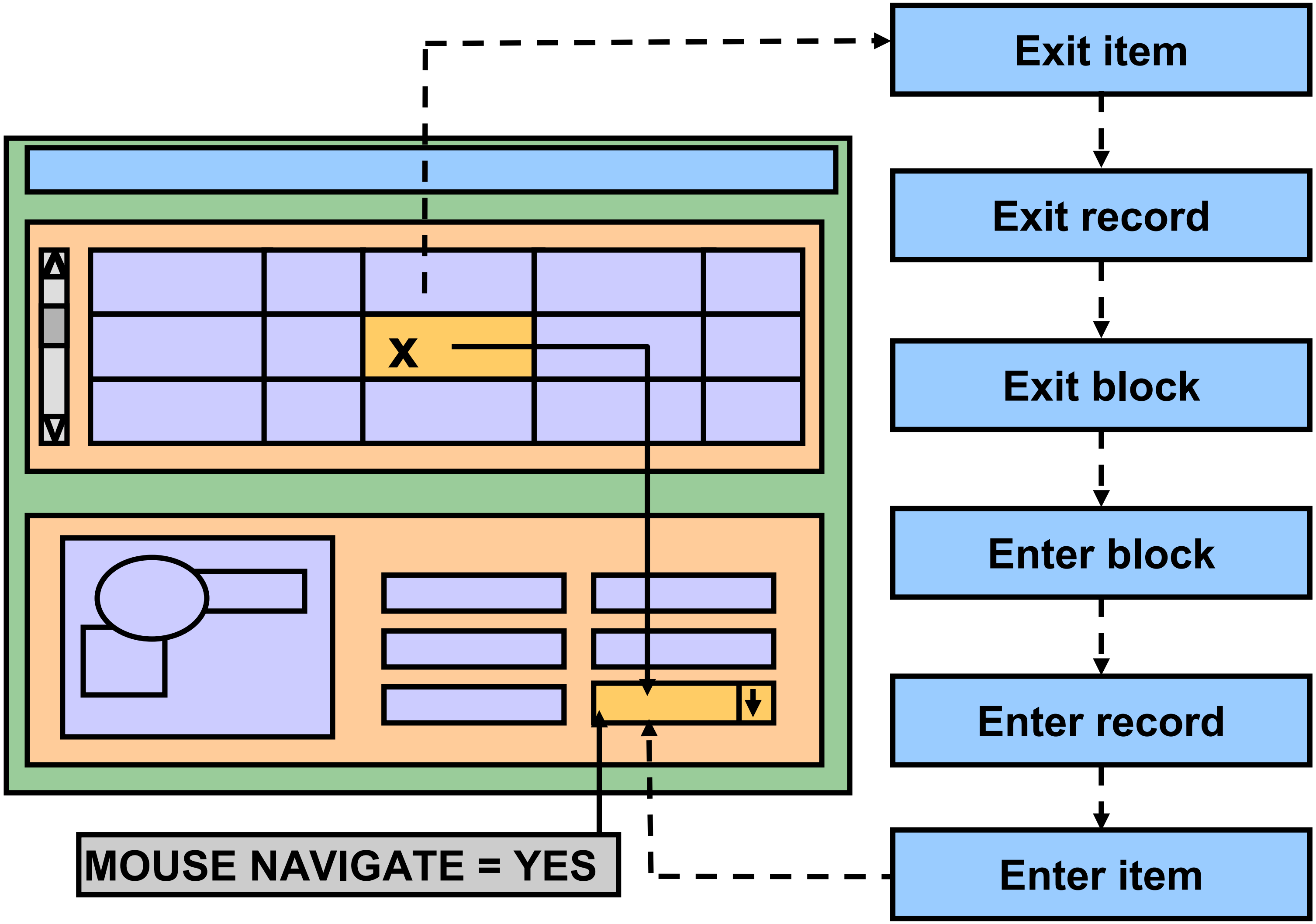
Form Module: NAVDEMO

⊟ General	
Name	NAVDEMO
Subclass Information	
Comments	
Help Book Title	
⊞ Functional	
⊞ Menu Security	
⊟ Navigation	
Mouse Navigation Limit	Form
First Navigation Data Block	<Null>
⊞ Records	
⊟ Database	

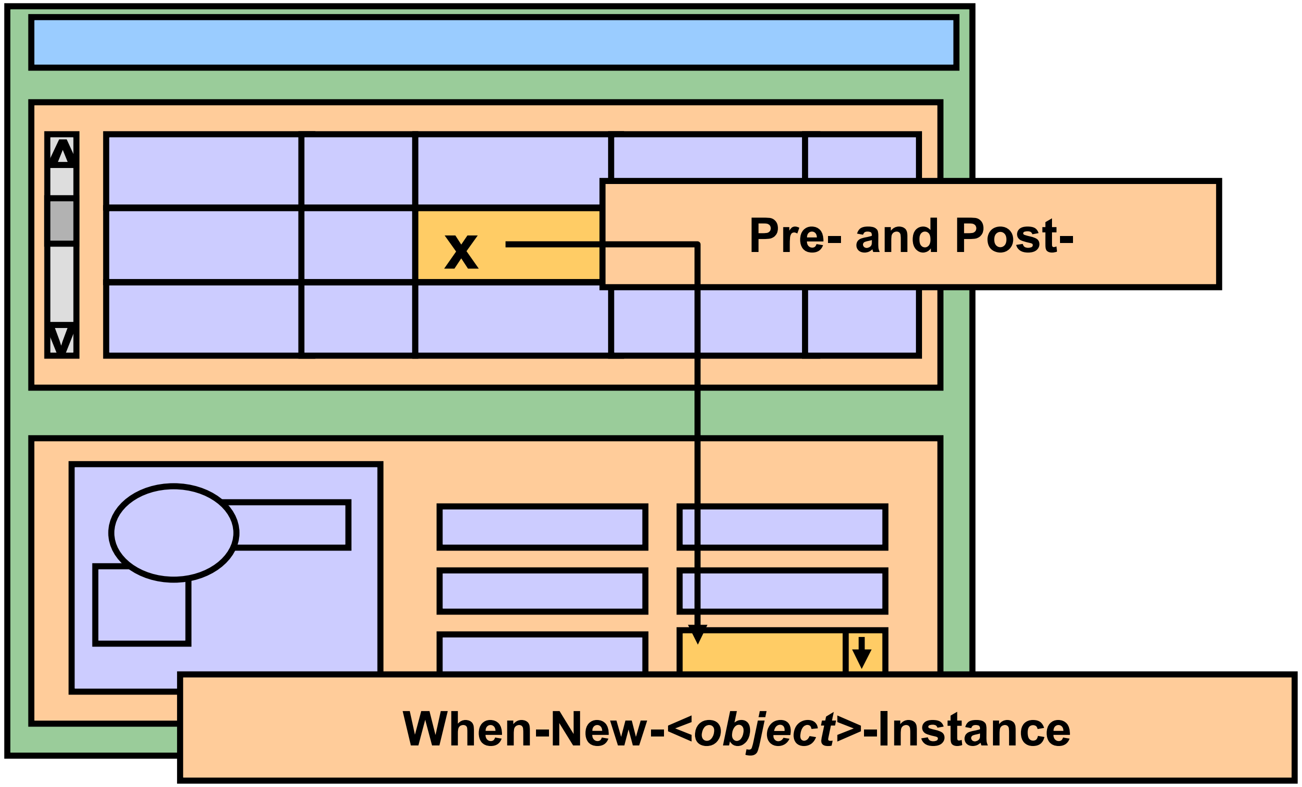
First Navigation Data Block

- Form module
  - Mouse Navigation Limit
  - First Navigation Data Block

Mouse Navigate Property



Writing Navigation Triggers



Pre- and Post-

When-New-<*object*>-Instance

Navigation Triggers

Pre- and Post-	When-New-<*object*>-Instance
Fire during navigation	Fire after navigation
Do not fire if validation unit is higher than trigger object	Fire even when validation unit is higher than the trigger object
Allow unrestricted built-ins	Allow restricted and unrestricted built-ins
Handle failure by returning to initial object	Are not affected by failure

When-New-<*object*>-Instance Triggers

- When-New-Form-Instance
- When-New-Block-Instance
- When-New-Record-Instance
- When-New-Item-Instance

SET_<*object*>_PROPERTY Examples

```
SET_FORM_PROPERTY(FIRST_NAVIGATION_BLOCK,
'ORDER_ITEMS');
```

```
SET_BLOCK_PROPERTY('ORDERS', ORDER_BY,
'CUSTOMER_ID');
```

```
SET_RECORD_PROPERTY(3, 'ORDER_ITEMS', STATUS,
QUERY_STATUS);
```

```
SET_ITEM_PROPERTY('CONTROL.stock_button',
ICON_NAME, 'stock');
```

The Pre- and Post-Triggers

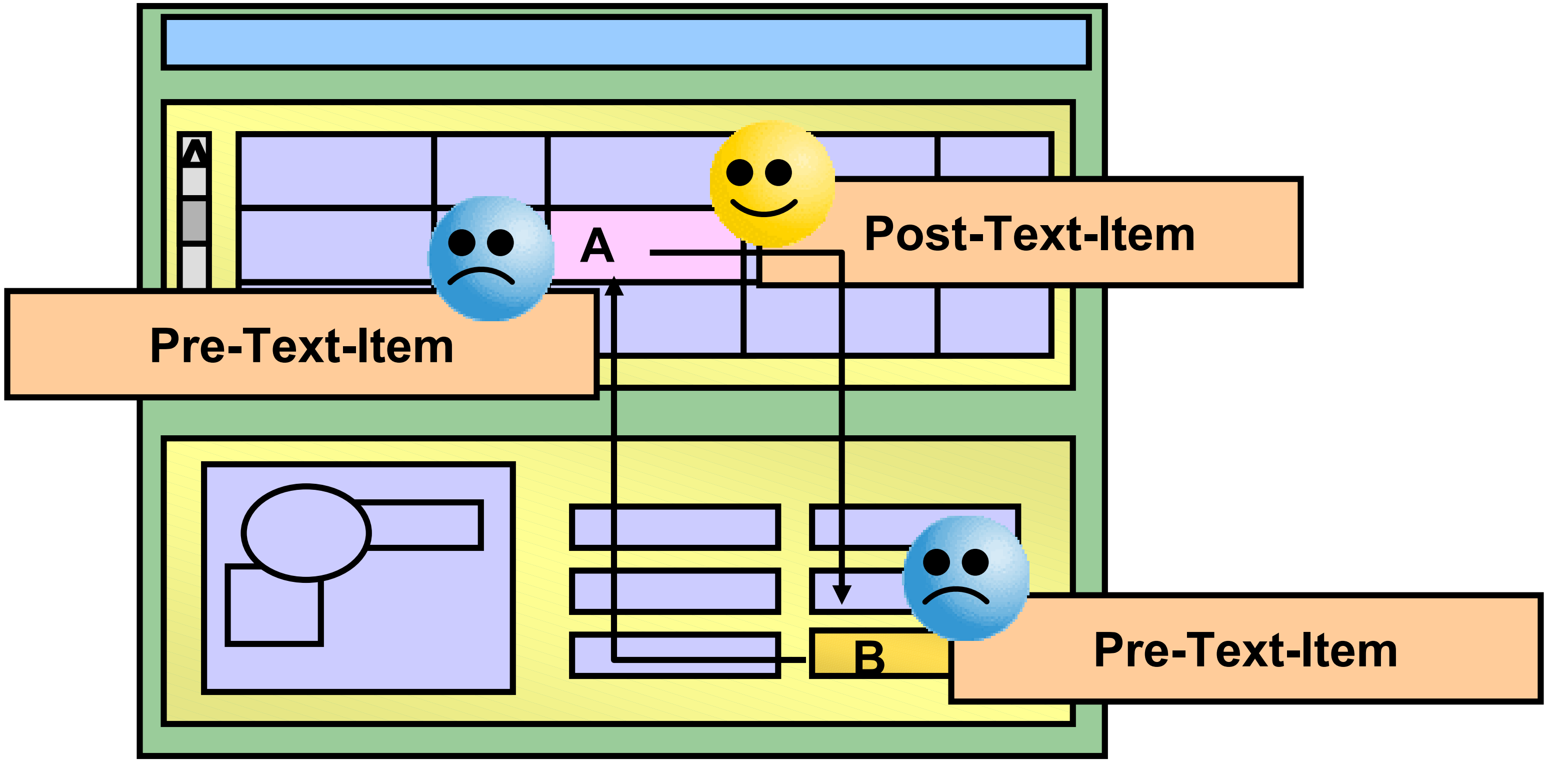
- Pre/Post-Form
- Pre/Post-Block
- Pre/Post-Record
- Pre/Post-Text-Item

Post-Block Trigger Example

Disabling Stock button when leaving the `ORDER_ITEMS` block:

```
SET_ITEM_PROPERTY(’CONTROL.stock_button’,
enabled, property_false);
```

The Navigation Trap



Using Navigation Built-Ins in Triggers

```
GO_FORM
GO_BLOCK
GO_ITEM
GO_RECORD
NEXT_BLOCK
NEXT_ITEM
NEXT_KEY
NEXT_RECORD
```

```
NEXT_SET
UP
DOWN
PREVIOUS_BLOCK
PREVIOUS_ITEM
PREVIOUS_RECORD
SCROLL_UP
SCROLL_DOWN
```

Using Navigation Built-Ins in Triggers

- **When-New-Item-Instance**

```
IF CHECKBOX_CHECKED('ORDERS.order_mode') --Online
  THEN                                   -- order
     ORDERS.order_status := 4; --Credit order
     GO_ITEM('ORDERS.order_status');
END IF;
```

☺

- **Pre-Text-Item**

```
IF CHECKBOX_CHECKED('ORDERS.order_mode') --Online
  THEN                                   -- order
     ORDERS.order_status := 4; --Credit order
     GO_ITEM('ORDERS.order_status');
END IF;
```

☹

Summary

In this lesson, you should have learned that:

- External navigation is visible to the user, while internal navigation occurs behind the scenes.
- You can control navigation with properties of the form, block, or item:
  - Set in Navigation category of the Property Palette

  OR
  - Use `SET_[FORM | BLOCK | ITEM]_PROPERTY`

Summary

- Navigation triggers:
  - Those that fire during navigation (watch out for the navigation trap):
    [Pre | Post] - [Form | Block | Item]
  - Those that fire after navigation:
    When-New- [Form | Block | Record | Item] -Instance
- You can use navigation built-ins in triggers (except for triggers that fire during navigation):
  - `GO_[FORM | BLOCK | RECORD | ITEM]`
  - `NEXT_[BLOCK | RECORD | ITEM | KEY | SET]`
  - `UP`
  - `DOWN`
  - `PREVIOUS_[BLOCK | RECORD | ITEM]`
  - `SCROLL_[UP | DOWN]`

Practice 20 Overview

This practice covers the following topics:

- Registering the bean area's JavaBean at form startup
- Setting properties on a Pluggable Java Component at form startup
- Executing a query at form startup
- Populating product images when cursor arrives on each record of the `ORDER_ITEMS` block

21

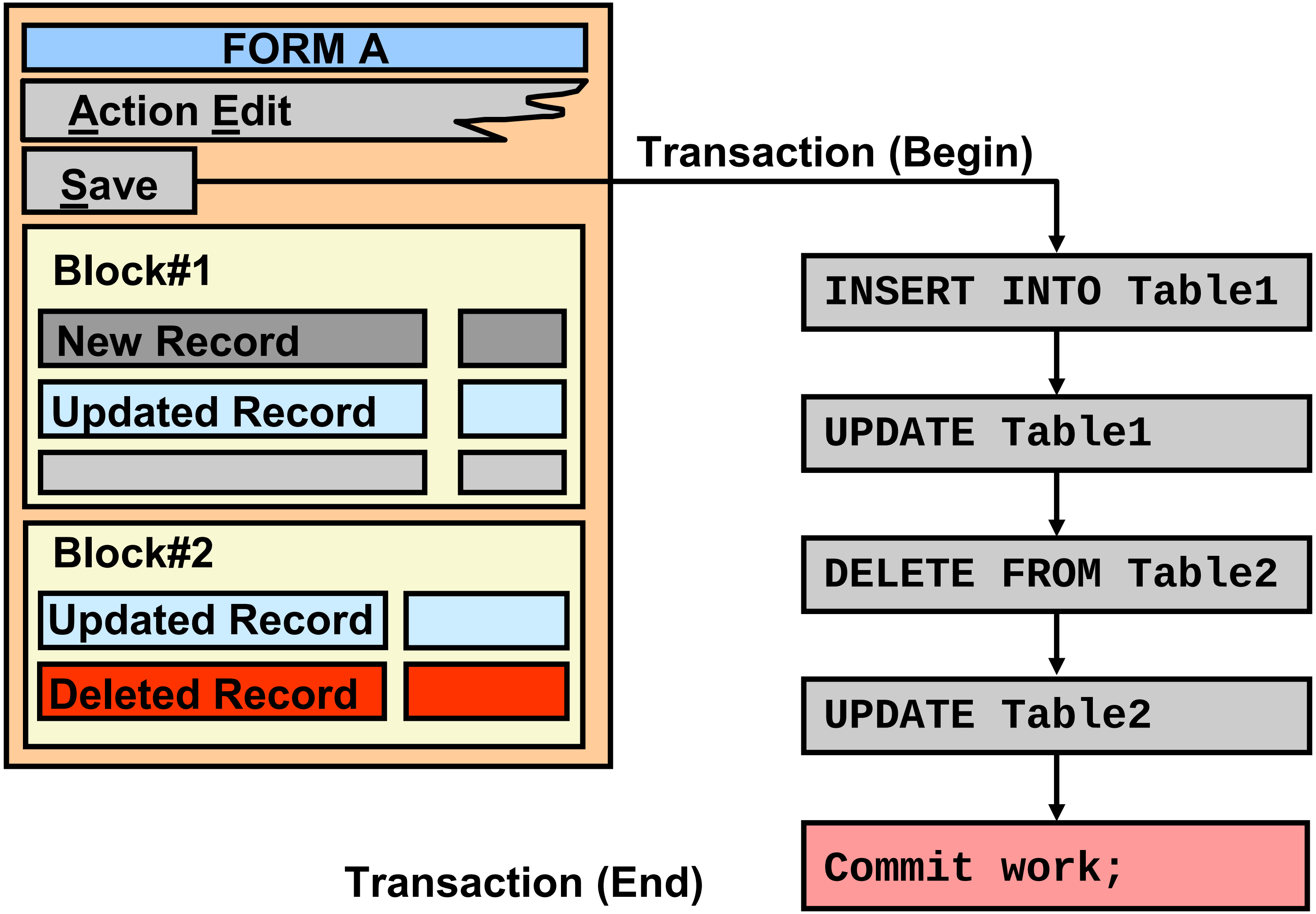
Transaction Processing

Objectives

After completing this lesson, you should be able to do the following:

- Explain the process used by Forms to apply changes to the database
- Describe the commit sequence of events
- Supplement transaction processing
- Allocate sequence numbers to records as they are applied to tables
- Implement array DML

Transaction Processing Overview



FORM A

Action Edit

Save

Block#1

New Record

Updated Record

Block#2

Updated Record

Deleted Record

Transaction (Begin)

```
INSERT INTO Table1
UPDATE Table1
DELETE FROM Table2
UPDATE Table2
Commit work;
```

Transaction (End)

Transaction Processing Overview

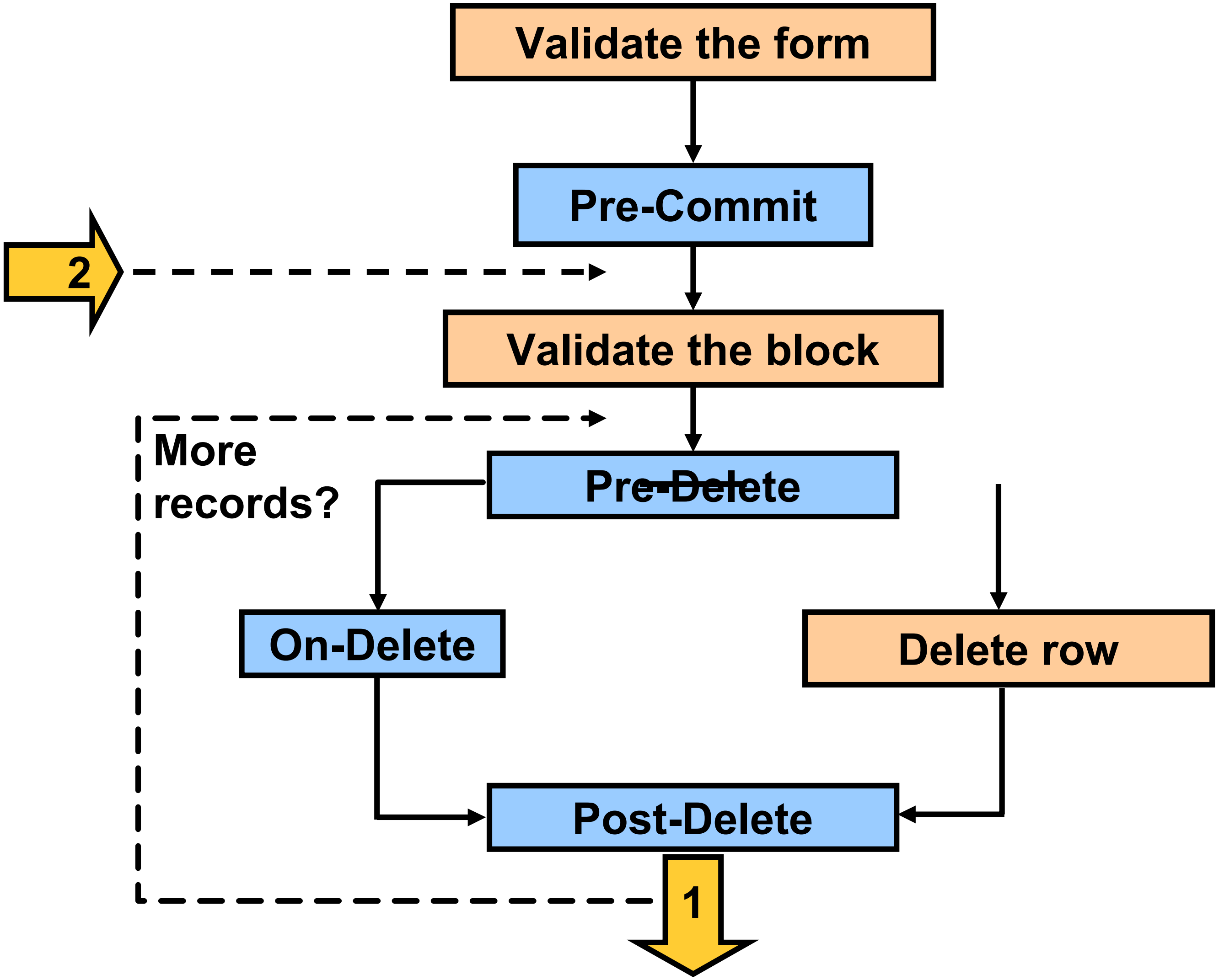
Transaction processing includes two phases:

- Post:
  - Writes record changes to base tables
  - Fires transactional triggers
- Commit: Performs database commit

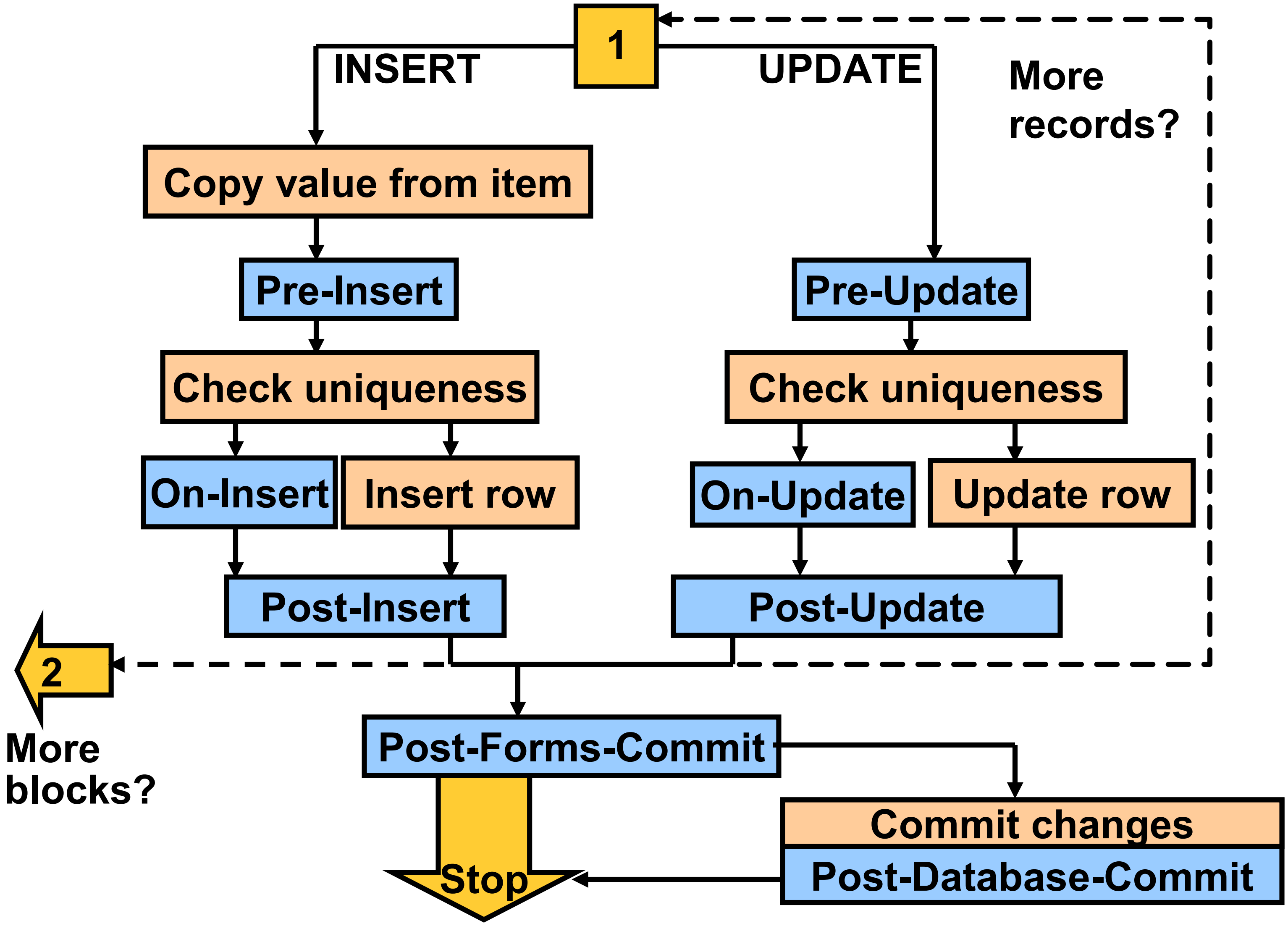
Errors result in:

- Rollback of the database changes
- Error message

The Commit Sequence of Events



The Commit Sequence of Events



Characteristics of Commit Triggers

- Pre-Commit: Fires once if form changes are made or uncommitted changes are posted
- Pre- and Post-DML
- On-DML: Fires per record, replacing default DML on row
  Use DELETE_RECORD, INSERT_RECORD, UPDATE_RECORD built-ins

Characteristics of Commit Triggers

- **Post-Forms-Commit: Fires once even if no changes are made**
- **Post-Database-Commit: Fires once even if no changes are made**

Note: A commit-trigger failure causes a rollback to the savepoint.

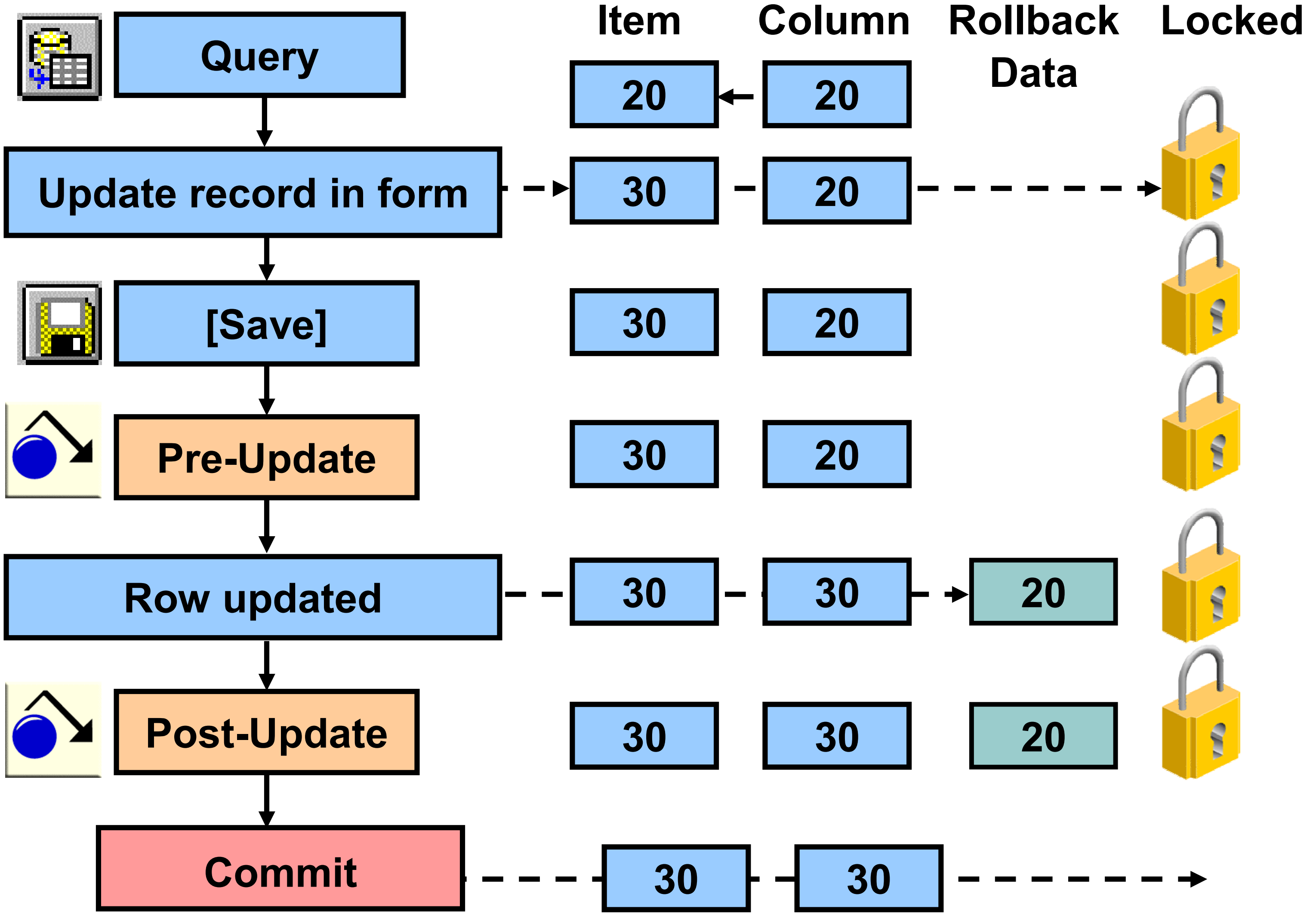
Common Uses for Commit Triggers

Pre-Commit	Check user authorization; set up special locking
Pre-Delete	Journaling; implement foreign-key delete rule
Pre-Insert	Generate sequence numbers; journaling; automatically generated columns; check constraints
Pre-Update	Journaling; implement foreign-key update rule; auto-generated columns; check constraints

Common Uses for Commit Triggers

On-Insert/Update/Delete	Replace default block DML statements
Post-Forms-Commit	Check complex multirow constraints
Post-Database-Commit	Test commit success; test uncommitted posts

Life of an Update



	Item	Column	Rollback Data	Locked
Query	20	20		
Update record in form	30	20		🔒
[Save]	30	20		🔒
Pre-Update	30	20		🔒
Row updated	30	30	20	🔒
Post-Update	30	30	20	🔒
Commit	30	30		

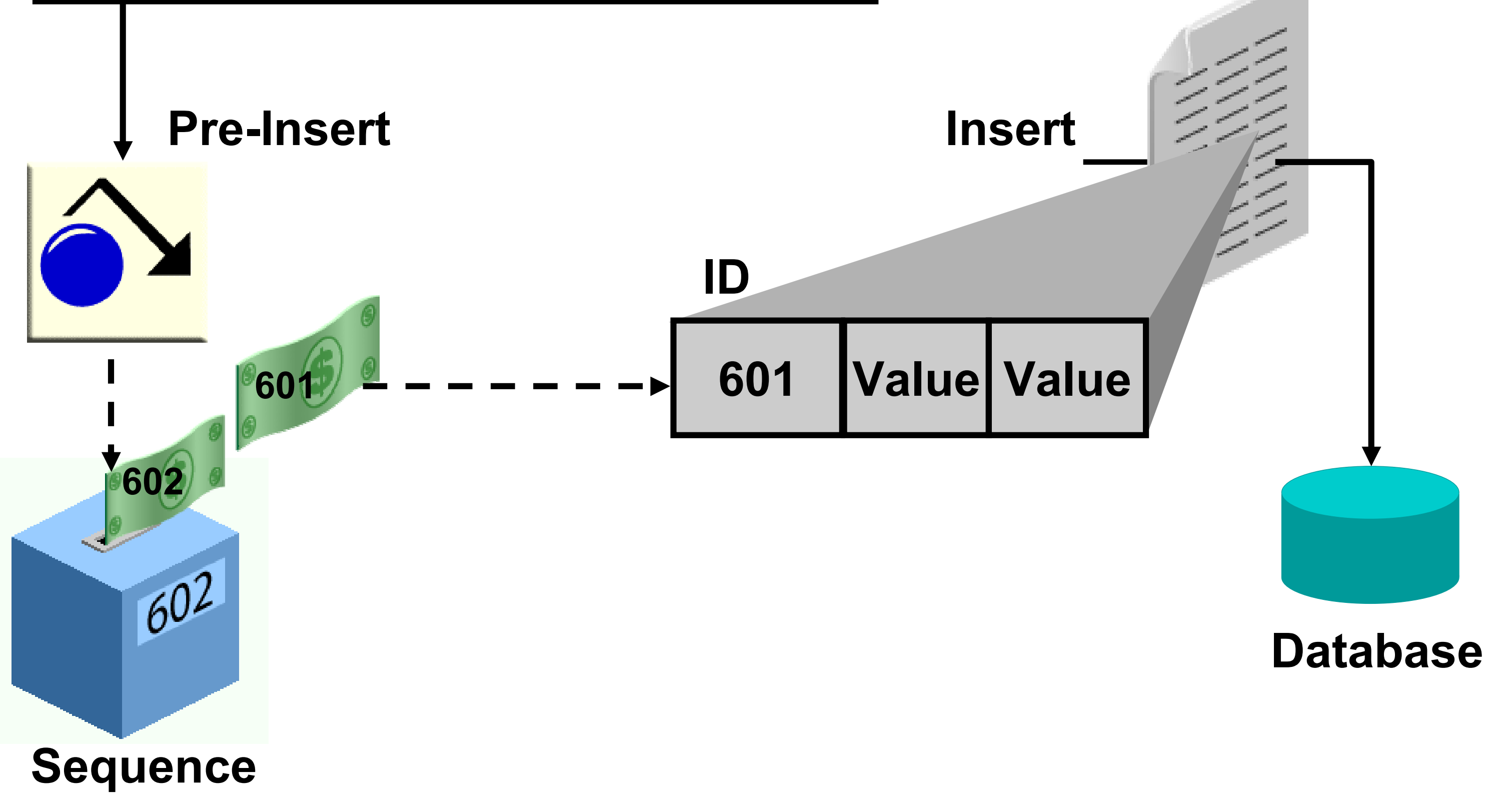
Delete Validation

- Pre-Delete trigger
- Final checks before row deletion

```
DECLARE
      CURSOR C1 IS
      SELECT ’anything’ FROM ORDERS
      WHERE customer_id = :CUSTOMERS.customer_id;
BEGIN
      OPEN C1;
      FETCH C1 INTO :GLOBAL.dummy;
      IF C1%FOUND THEN
           CLOSE C1;
           MESSAGE(’There are orders for this
customer!’);
           RAISE form_trigger_failure;
      ELSE
           CLOSE C1;
      END IF;
END;
```

Assigning Sequence Numbers

```
SELECT   ORDERS_SEQ.nextval
INTO     :ORDERS.order_id
FROM     SYS.dual;
```



Pre-Insert

Insert

ID

601	Value	Value

Database

Sequence

Keeping an Audit Trail

- Write changes to nonbase tables.
- Gather statistics on applied changes.

Post-Insert example:

```
:GLOBAL.insert_tot :=
  TO_CHAR(TO_NUMBER(:GLOBAL.insert_tot)+1);
```

Testing the Results of Trigger DML

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

```
UPDATE ORDERS
 SET order_date = SYSDATE
 WHERE order_id = :ORDERS.order_id;
IF SQL%NOTFOUND THEN
  MESSAGE(’Record not found in database’);
  RAISE form_trigger_failure;
END IF;
```

Testing the Results of Trigger DML

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

```
UPDATE S_ORD
 SET date_shipped = SYSDATE
 WHERE id = :S_ORD.id;
IF SQL%NOTFOUND THEN
  MESSAGE(’Record not found in database’);
  RAISE form_trigger_failure;
END IF;
```

DML Statements Issued During Commit Processing

```
INSERT INTO base_table   (base_column, base_column,...)
VALUES                   (:base_item, :base_item, ...)
```

```
UPDATE    base_table
SET       base_column = :base_item, base_column =
                :base_item, ...
WHERE     ROWID = :ROWID
```

```
DELETE    FROM base_table
WHERE     ROWID = :ROWID
```

DML Statements Issued During Commit Processing

Rules:

- DML statements may fire database triggers.
- Forms uses and retrieves `ROWID`.
- The Update Changed Columns Only and Enforce Column Security properties affect `UPDATE` statements.
- Locking statements are not issued.

Overriding Default Transaction Processing

Additional transactional triggers:

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

Note: These triggers are meant to be used when connecting to data sources other than Oracle.

Overriding Default Transaction Processing

Transactional triggers for logging on and off:

Trigger	Do-the-Right-Thing Built-in
Pre-Logon	-
Pre-Logout	-
On-Logon	LOGON
On-Logout	LOGOUT
Post-Logon	-
Post-Logout	-

Running Against Data Sources Other than Oracle

- Two ways to run against data sources other than Oracle:
  - Oracle Transparent Gateways
  - Write appropriate transactional triggers

Running Against Data Sources Other than Oracle

- Connecting with Open Gateway:
  - Cursor and Savepoint mode form module properties
  - Key mode and Locking mode block properties
- Using transactional triggers:
  - Call 3GL programs
  - Database data block property

Getting and Setting the Commit Status

- Commit status: Determines how record will be processed
- `SYSTEM.RECORD_STATUS`:
  - NEW
  - INSERT (also caused by control items)
  - QUERY
  - CHANGED
- `SYSTEM.BLOCK_STATUS`:
  - NEW (may contain records with status INSERT)
  - QUERY (also possible for control block)
  - CHANGED (block will be committed)
- `SYSTEM.FORM_STATUS`: NEW, QUERY, CHANGED

Getting and Setting the Commit Status

- System variables versus built-ins for commit status
- Built-ins for getting and setting commit status:
  - `GET_BLOCK_PROPERTY`
  - `GET_RECORD_PROPERTY`
  - `SET_RECORD_PROPERTY`

Getting and Setting the Commit Status

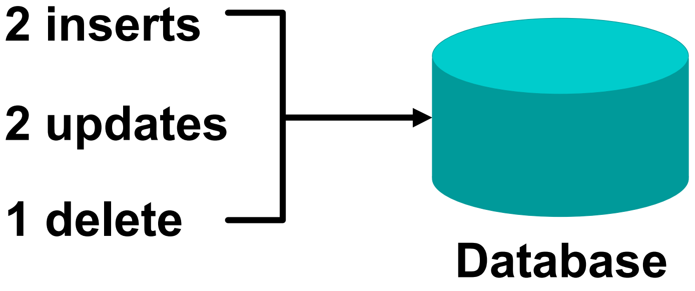
- Example: If the third record of block `ORDERS` is a changed database record, set the status back to QUERY.
- Warnings:
  - Do not confuse commit status with validation status.
  - The commit status is updated during validation.

Array DML

- Performs array inserts, updates, and deletes
- Vastly reduces network traffic

Empno	Ename	Job	Hiredate
1234	Jones	Clerk	01-Jan-1995
1235	Smith	Clerk	01-Jan-1995
1236	Adams	Clerk	01-Jan-1995
1237	Clark	Clerk	01-Jan-1995

Fewer round trips (exact number depends on array size)



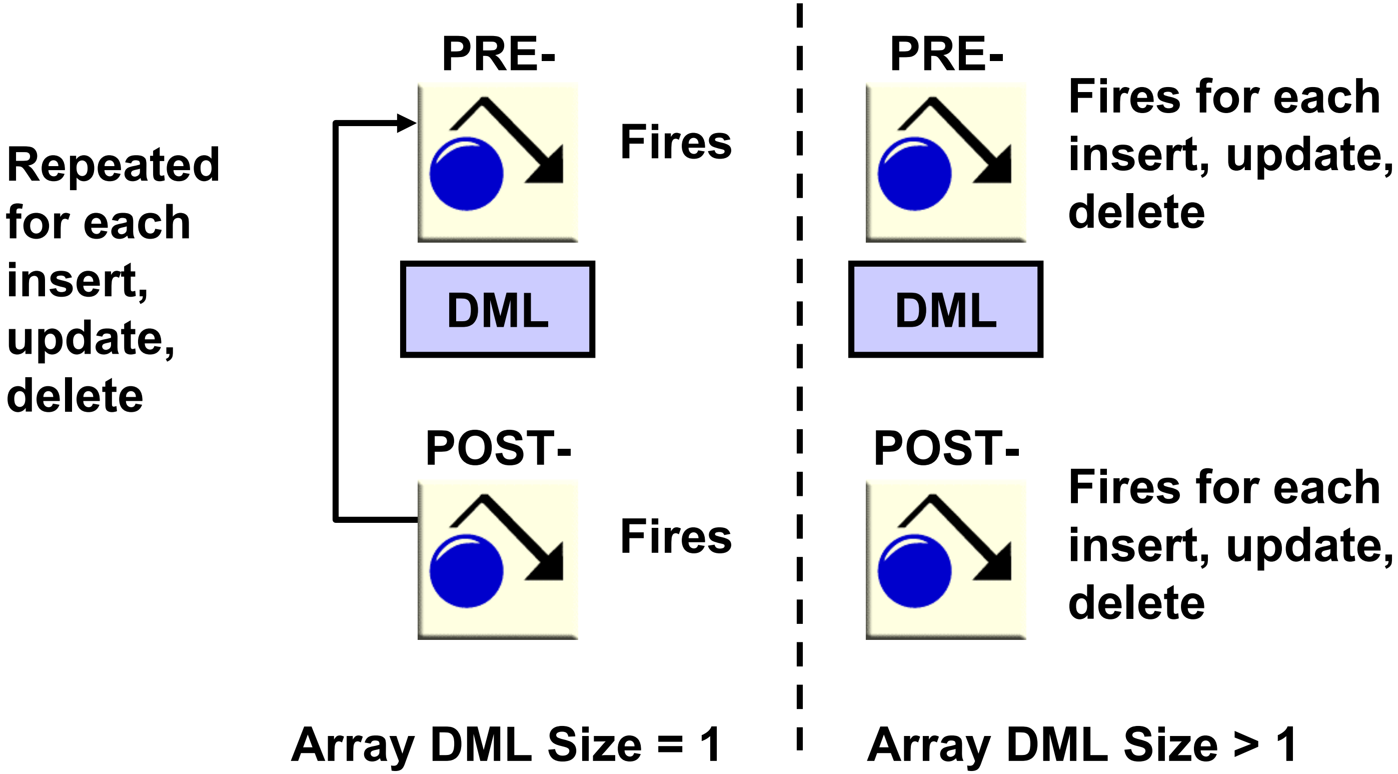
2 inserts

2 updates

1 delete

Database

Effect of Array DML on Transactional Triggers



Repeated for each insert, update, delete
PRE-
Fires
DML
POST-
Fires
Array DML Size = 1
PRE-
Fires for each insert, update, delete
DML
POST-
Fires for each insert, update, delete
Array DML Size > 1

Implementing Array DML

1. Enable the Array Processing option.
2. Specify a DML Array Size of greater than 1.
3. Specify block primary keys.

Summary

In this lesson, you should have learned that:

- To apply changes to the database, Forms issues post and commit.
- The commit sequence of events:
  1. Validate the form.
  2. Process savepoint.
  3. Fire Pre-Commit.
  4. Validate the block (performed for all blocks in sequential order).

Summary

5. Perform the DML:

   Delete records: Fire Pre-Delete, delete row or fire On-Delete, fire Post-Delete trigger

   Insert records: Copy Value From Item, fire Pre-Insert, check record uniqueness, insert row or fire On-Insert, fire Post-Insert

   Update records: Fire Pre-Update, check record uniqueness, update row or fire On-Update, fire Post-Update
6. Fire Post-Forms-Commit trigger.

If the current operation is COMMIT, then:

7. Issue an SQL-COMMIT statement.
8. Fire the Post-Database-Commit trigger.

Summary

- You can supplement transaction processing with triggers:
  - Pre-Commit: Fires once if form changes are made or uncommitted changes are posted
  - [Pre | Post] – [Update | Insert | Delete]
  - On- [Update | Insert | Delete]:
    Fires per record, replacing default DML on row
    Perform default functions with built-ins:
    `[UPDATE|INSERT|DELETE]_RECORD`

Summary

- Use the Pre-Insert trigger to allocate sequence numbers to records as they are applied to tables.
- Check or change commit status:
  - `GET_BLOCK_PROPERTY, [GET | SET]_RECORD_STATUS`
  - `:SYSTEM.[FORM | BLOCK | RECORD]_STATUS`
- Use transactional triggers to override or augment default commit processing.
- Reduce network roundtrips by setting DML Array Size block property to implement Array DML.

Practice 21 Overview

This practice covers the following topics:

- Automatically populating order IDs by using a sequence
- Automatically populating item IDs by adding the current highest order ID
- Customizing the commit messages in the `CUSTOMERS` form
- Customizing the login screen in the `CUSTOMERS` form

22

Writing Flexible Code

Objectives

After completing this lesson, you should be able to do the following:

- Describe flexible code
- State the advantages of using system variables
- Identify built-in subprograms that assist flexible coding
- Write code to reference objects:
  - By internal ID
  - Indirectly

What Is Flexible Code?

Flexible code:

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

Using System Variables for Current Context

- Input focus:
  - SYSTEM.CURSOR_BLOCK
  - SYSTEM.CURSOR_RECORD
  - SYSTEM.CURSOR_ITEM
  - SYSTEM.CURSOR_VALUE

```
IF :SYSTEM.CURSOR_BLOCK = ’ORDERS’ THEN
  GO_BLOCK(’ORDER_ITEMS’);
ELSIF :SYSTEM.CURSOR_BLOCK = ’ORDER_ITEMS’ THEN
  GO_BLOCK(’INVENTORIES’);
ELSIF :SYSTEM.CURSOR_BLOCK = ’INVENTORIES’ THEN
  GO_BLOCK(’ORDERS’);
END IF;
```

Using System Variables for Current Context

- Trigger focus:
  - `SYSTEM.TRIGGER_BLOCK`
  - `SYSTEM.TRIGGER_RECORD`
  - `SYSTEM.TRIGGER_ITEM`

System Status Variables

When-Button-Pressed

```
ENTER;
IF :SYSTEM.BLOCK_STATUS = ’CHANGED’ THEN
  COMMIT_FORM;
END IF;
CLEAR_BLOCK;
```

GET_<*object*>_PROPERTY Built-Ins

- `GET_APPLICATION_PROPERTY`
- `GET_FORM_PROPERTY`
- `GET_BLOCK_PROPERTY`
- `GET_RELATION_PROPERTY`
- `GET_RECORD_PROPERTY`
- `GET_ITEM_PROPERTY`
- `GET_ITEM_INSTANCE_PROPERTY`

GET_<*object*>_PROPERTY Built-Ins

- GET_LOV_PROPERTY
- GET_RADIO_BUTTON_PROPERTY
- GET_MENU_ITEM_PROPERTY
- GET_CANVAS_PROPERTY
- GET_TAB_PAGE_PROPERTY
- GET_VIEW_PROPERTY
- GET_WINDOW_PROPERTY

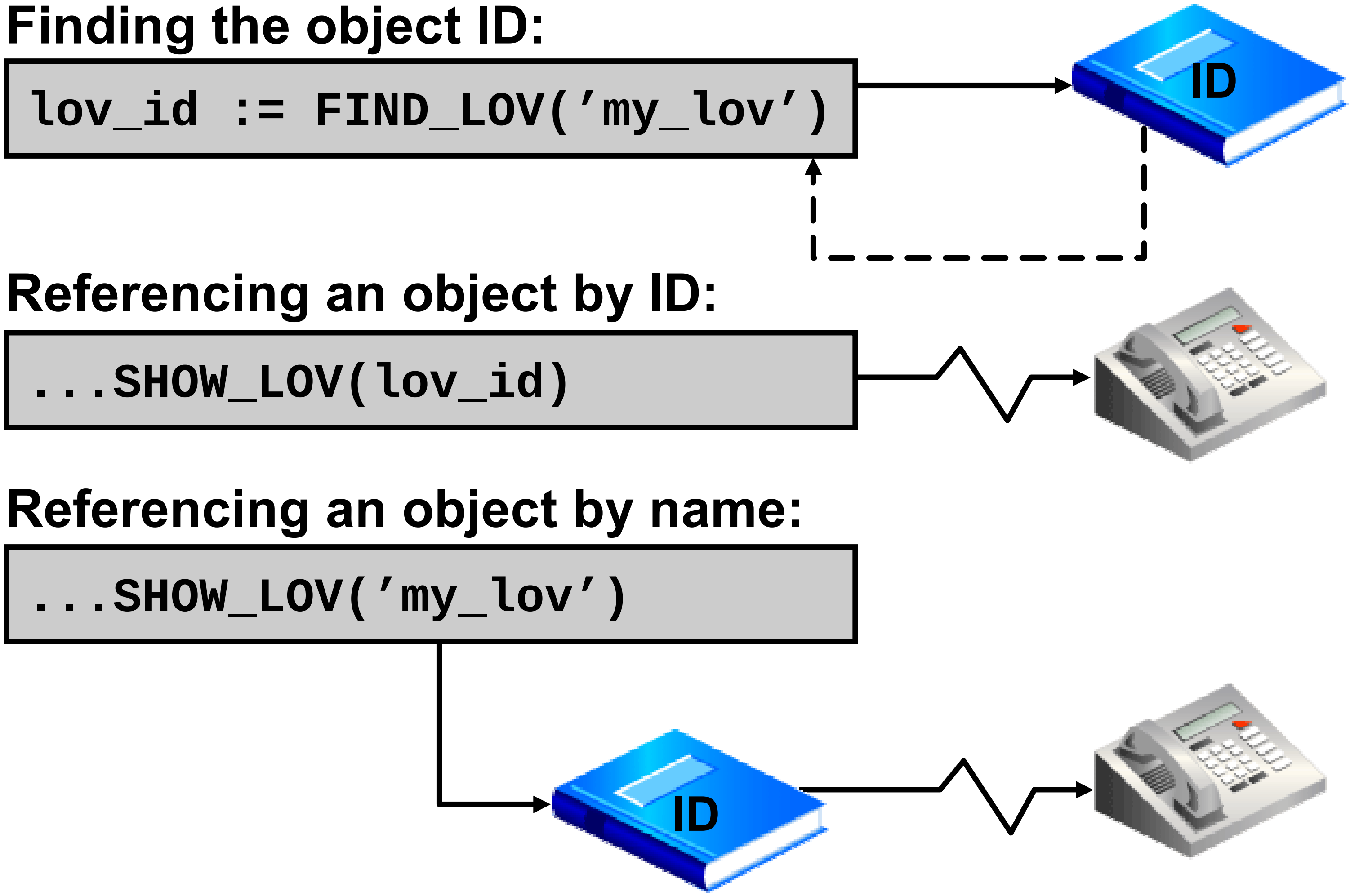
SET_<*object*>_PROPERTY Built-Ins

- `SET_APPLICATION_PROPERTY`
- `SET_FORM_PROPERTY`
- `SET_BLOCK_PROPERTY`
- `SET_RELATION_PROPERTY`
- `SET_RECORD_PROPERTY`
- `SET_ITEM_PROPERTY`
- `SET_ITEM_INSTANCE_PROPERTY`

SET_<*object*>_PROPERTY Built-Ins

- `SET_LOV_PROPERTY`
- `SET_RADIO_BUTTON_PROPERTY`
- `SET_MENU_ITEM_PROPERTY`
- `SET_CANVAS_PROPERTY`
- `SET_TAB_PAGE_PROPERTY`
- `SET_VIEW_PROPERTY`
- `SET_WINDOW_PROPERTY`

Referencing Objects by Internal ID



Finding the object ID:

```
lov_id := FIND_LOV('my_lov')
```

Referencing an object by ID:

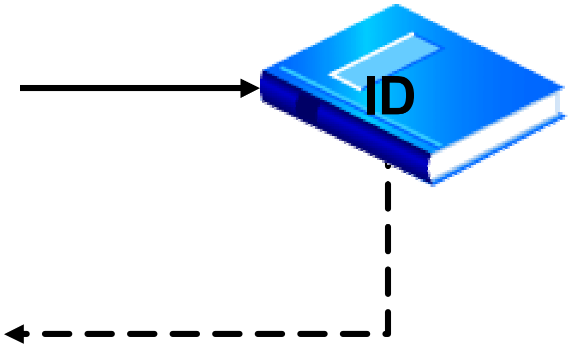
```
...SHOW_LOV(lov_id)
```

Referencing an object by name:

```
...SHOW_LOV('my_lov')
```

FIND_ Built-Ins

- FIND_ALERT
- FIND_BLOCK
- FIND_CANVAS
- FIND_EDITOR
- FIND_FORM
- FIND_ITEM
- FIND_LOV
- FIND_RELATION
- FIND_VIEW
- FIND_WINDOW



Using Object IDs

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

Using Object IDs

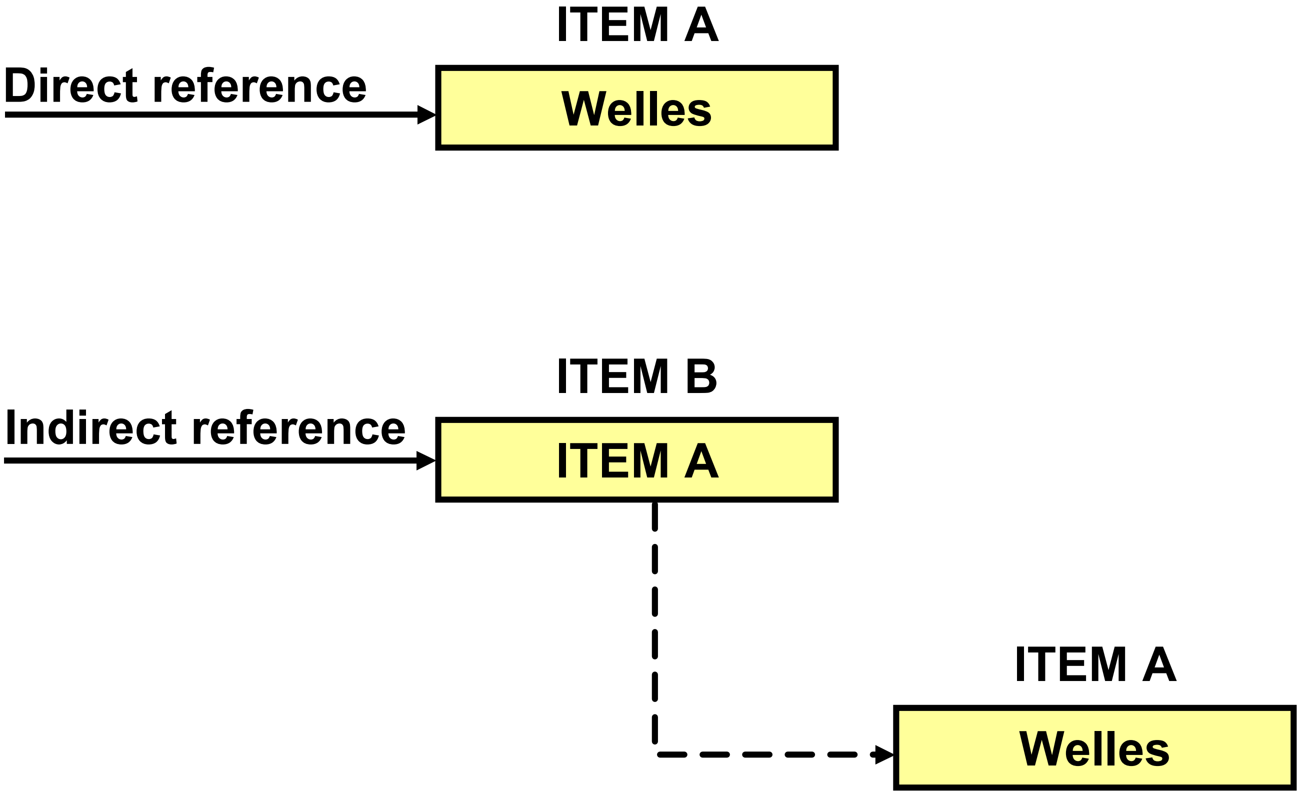
Example:

```
DECLARE
   item_var item;
BEGIN
   item_var := FIND_ITEM(:SYSTEM.CURSOR_ITEM);
   SET_ITEM_PROPERTY(item_var,position,30,55);
   SET_ITEM_PROPERTY(item_var,prompt_text,’Cur
rent’);
END;
```

Increasing the Scope of Object IDs

- A PL/SQL variable has limited scope.
- An `.id` extension:
  - Broadens the scope
  - Converts to a numeric format
  - Enables assignment to a global variable
  - Converts back to the object data type

Referencing Objects Indirectly



Referencing Objects Indirectly

The `NAME_IN` function:

- Returns:
  - The contents of variable
  - Character string
- Use conversion functions for NUMBER and DATE

Referencing Objects Indirectly

The COPY procedure allows:

- Direct copy:

```
COPY('Welles','CUSTOMERS.cust_last_name');
```

- Indirect copy:

```
COPY('Welles',NAME_IN('global.customer_name_item'));
```

Summary

In this lesson, you should have learned that:

- Flexible code is reusable, generic code that you can use in any form module in an application.
- With system variables you can:
  - Perform actions conditionally based on current location (`SYSTEM.CURSOR_[RECORD | ITEM | BLOCK]`)
  - Use the value of an item without knowing its name (`SYSTEM.CURSOR_VALUE`)
  - Navigate to the initial location after a trigger completes: (`SYSTEM.TRIGGER_[RECORD | ITEM | BLOCK]`)
  - Perform actions conditionally based on commit status: `SYSTEM.[RECORD | BLOCK | FORM]_STATUS`

Summary

- The [`GET` | `SET`]_<*object*>_`PROPERTY` built-ins are useful in flexible coding.
- Code that references objects is more efficient and generic:
  - By internal ID: Use `FIND_`<object> built-ins
  - Indirectly: Use `COPY` and `NAME_IN` built-ins

Practice 22 Overview

This practices covers the following topics:

- Populating product images only when the image item is displayed.
- Modifying the `When-Button-Pressed` trigger of the Image_Button in order to use object IDs instead of object names.
- Write generic code to print out the names of the blocks in a form.

23

Sharing Objects and Code

Objectives

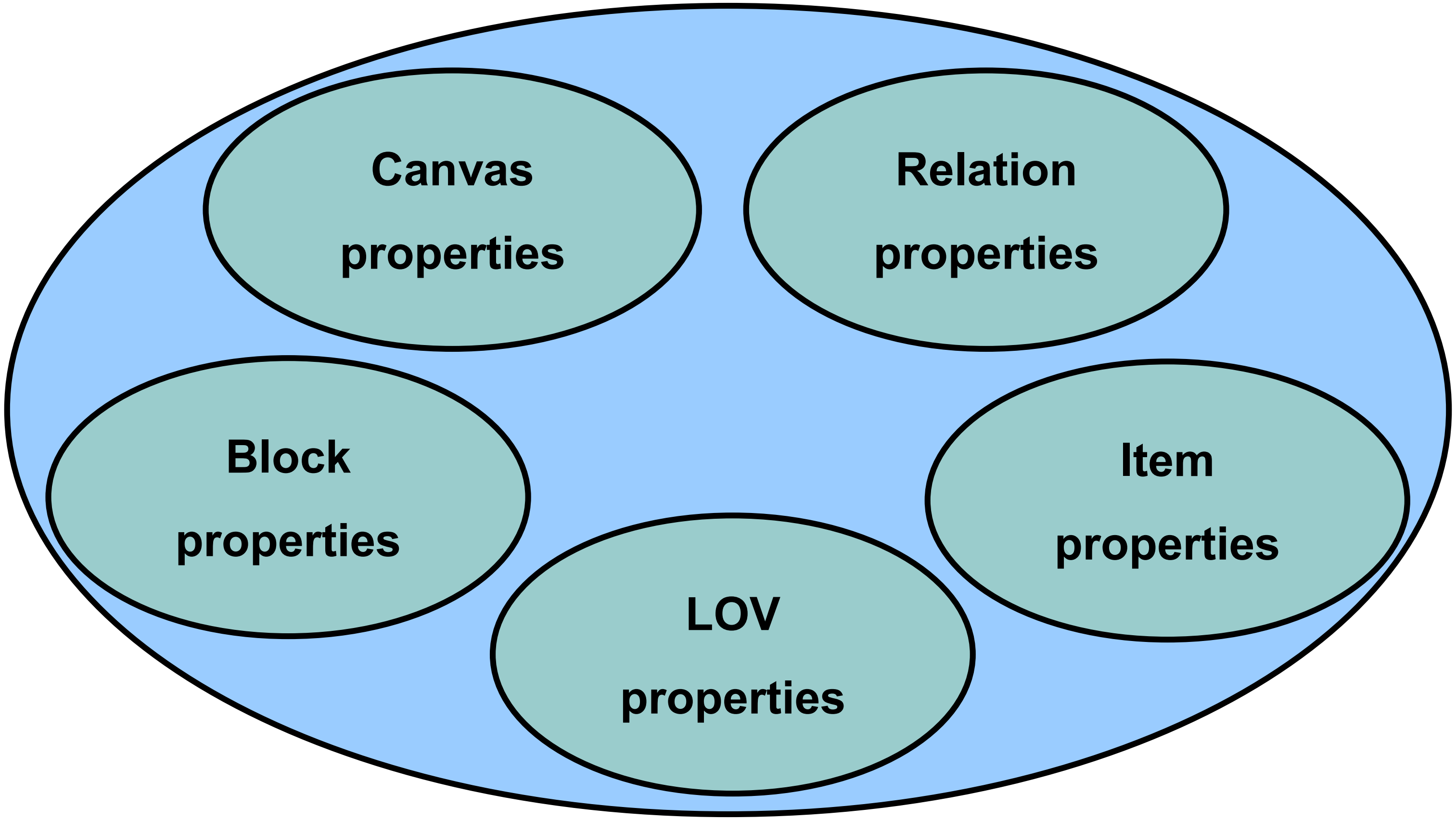
After completing this lesson, you should be able to do the following:

- Describe the various methods for reusing objects and code
- Inherit properties from property classes
- Group related objects for reuse
- Explain the inheritance symbols in the Property Palette
- Reuse objects from an object library
- Reuse PL/SQL code

Benefits of Reusing Objects and Code

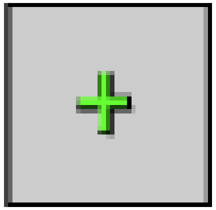
- Increases productivity
- Decreases maintenance
- Increases modularity
- Maintains standards
- Improves application performance

What Are Property Classes?

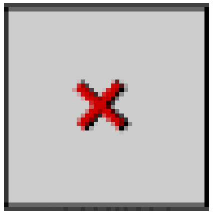


Creating a Property Class

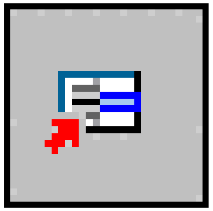
Add Property



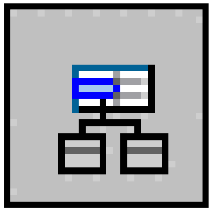
Delete Property



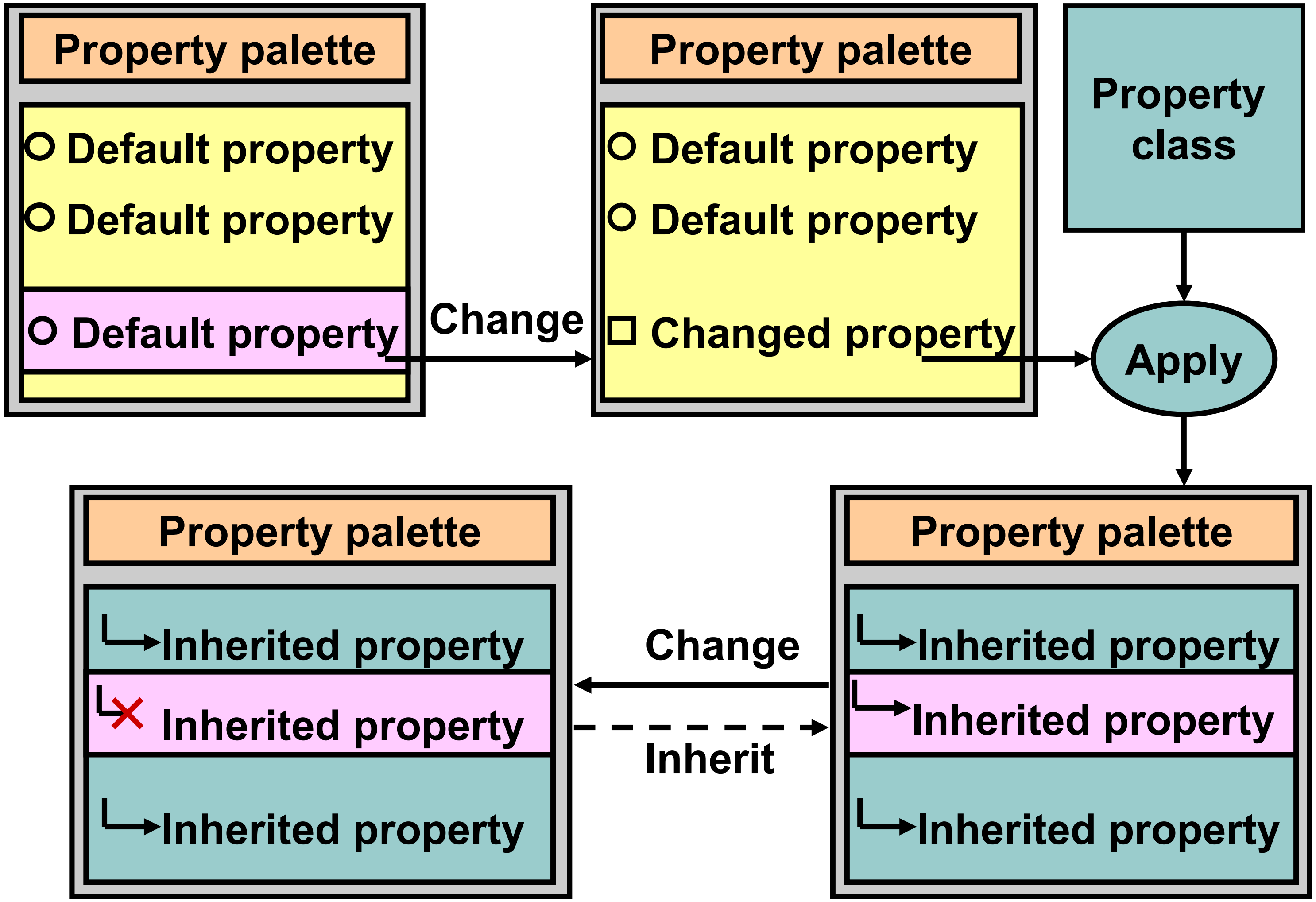
Inherit Property



Property Class

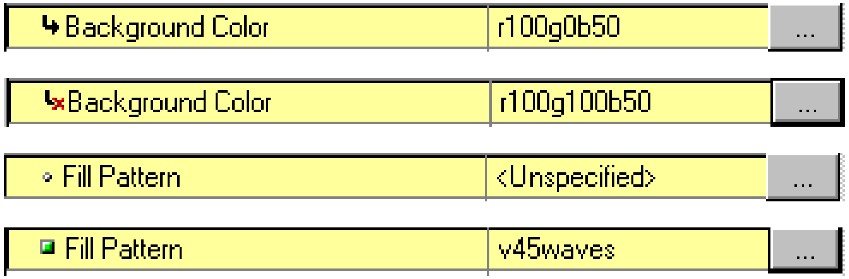


Inheriting from a Property Class



Inheriting from a Property Class

- Set the Subclass Information property.
- Convert an inherited property to a variant property.
- Convert a variant property to an inherited property.
- Convert a changed property to a default property.



Inherited Property	Background Color	r100g0b50
Variant Property	Background Color	r100g100b50
Default Property	Fill Pattern	<Unspecified>
Changed Property	Fill Pattern	v45waves

What Are Object Groups?

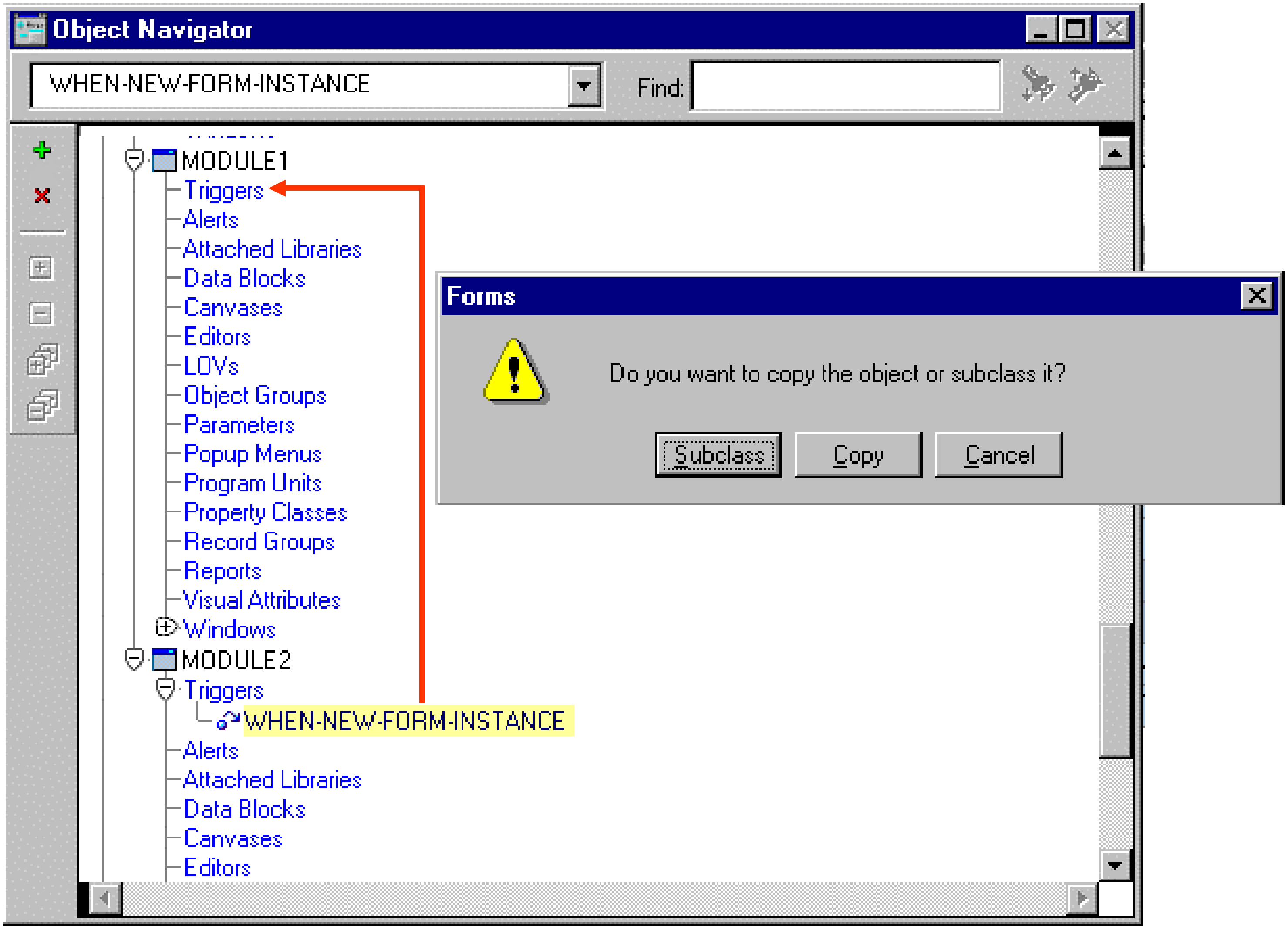
Object groups:

- Are logical containers
- Enable you to:
  - Group related objects
  - Copy multiple objects in one operation

Creating and Using Object Groups

- Blocks include:
  - Items
  - Item-level triggers
  - Block-level triggers
  - Relations
- Object groups cannot include other object groups
- Deleting an object group does not affect the objects
- Deleting an object affects the object group

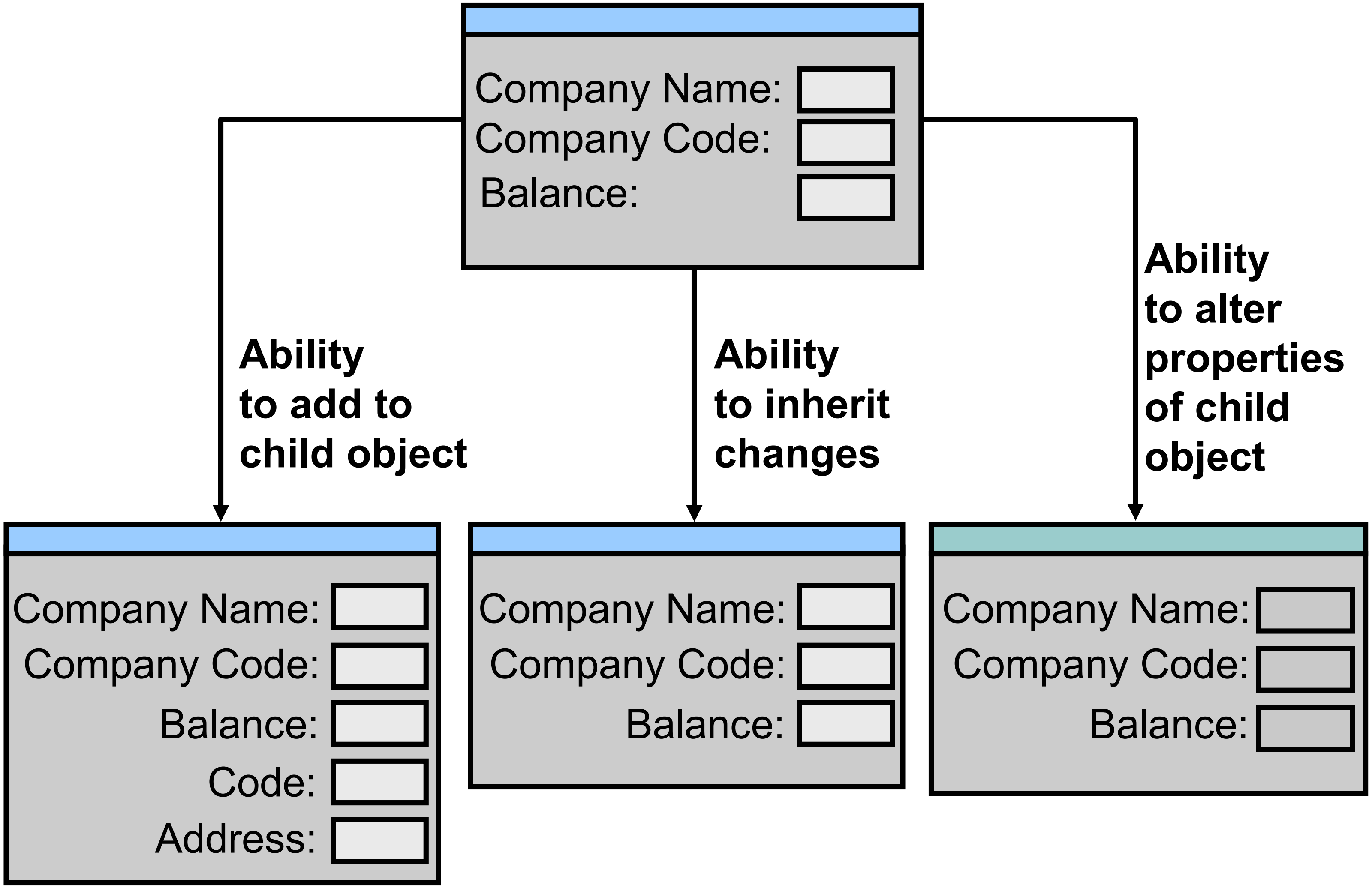
Copying and Subclassing Objects and Code



WHEN-NEW-FORM-INSTANCE

Do you want to copy the object or subclass it?

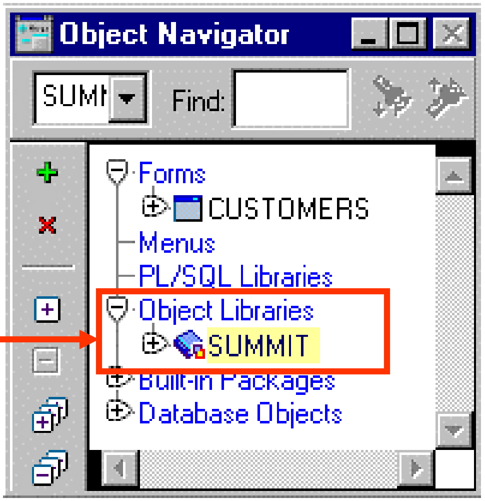
Subclassing



What Are Object Libraries?

An Object Library:

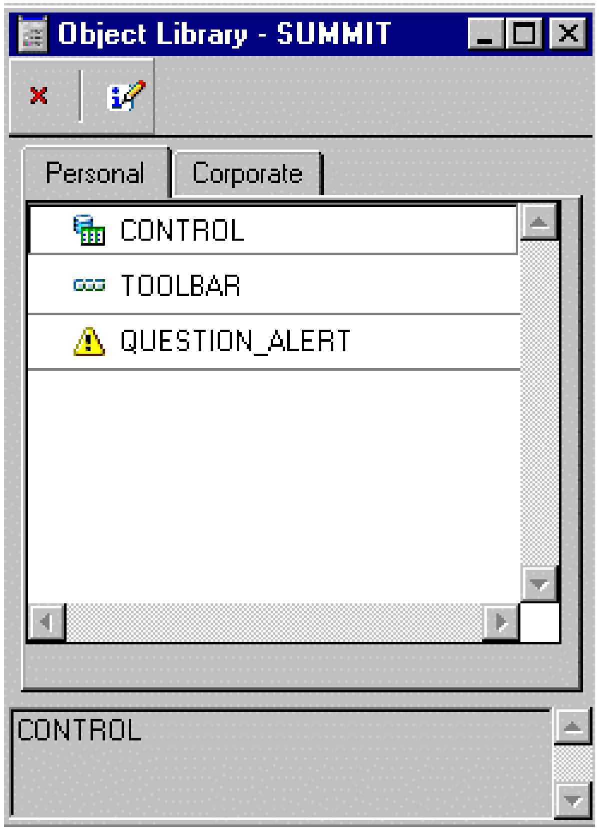
- Is a convenient container of objects for reuse
- Simplifies reuse in complex environments
- Supports corporate, project, and personal standards
- Simplifies the sharing of reusable components
- Is separate from the form module



Benefits of the Object Library

- Simplifies the sharing and reuse of objects
- Provides control and enforcement of standards
- Promotes increased network performance
- Eliminates the need to maintain multiple referenced forms

Working with Object Libraries

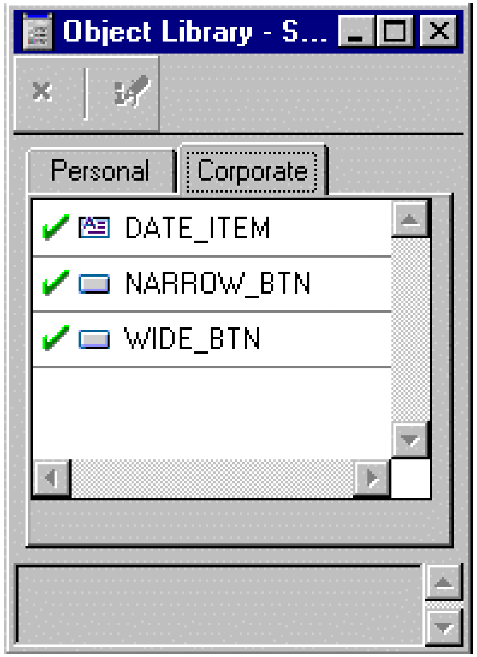


Object Libraries:

- Appear in the Navigator if they are open
- Are used with a simple tabbed interface
- Are populated by dragging Form objects to tab page
- Are saved to `.olb` file

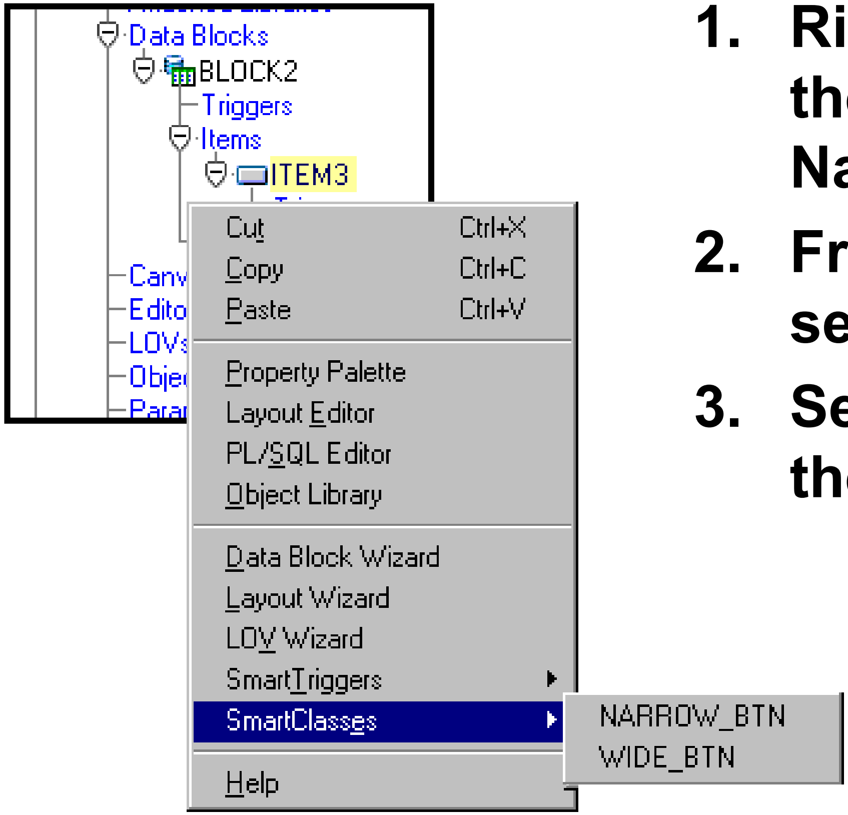
What Is a SmartClass?

- A SmartClass:
  - Is an object in an object library that is frequently used as a class
  - Can be applied easily and rapidly to existing objects
  - Can be defined in many object libraries
  - Is the preferred method to promote similarity among objects for performance
- You can have many SmartClasses of a given object type.



Check indicates a SmartClass

Working with SmartClasses

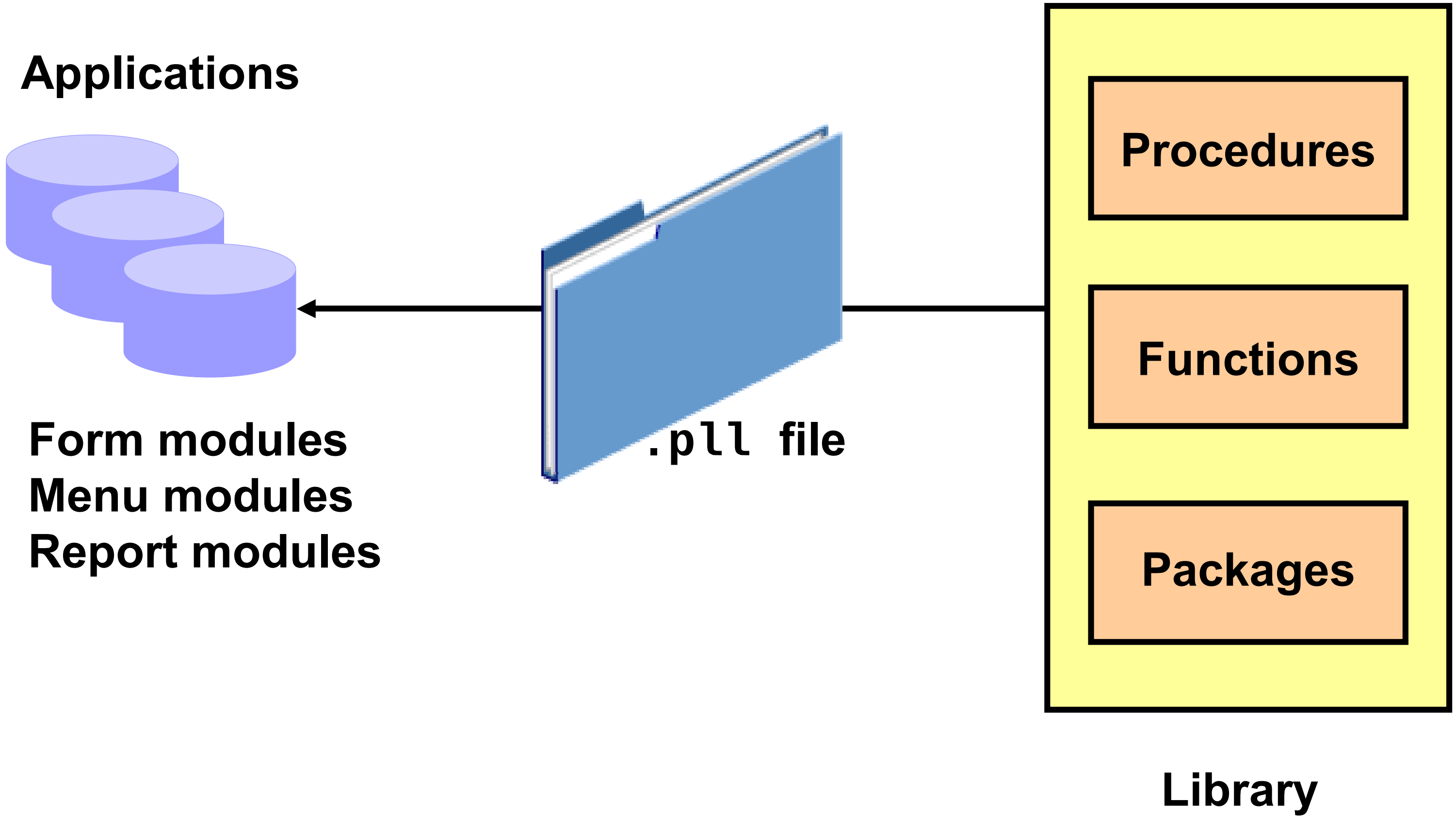


1. Right-click an object in the Layout Editor or Navigator.
2. From the pop-up menu, select SmartClasses.
3. Select a class from the list.

Reusing PL/SQL

- Triggers:
  - Copy and paste text
  - Copy and paste within a module
  - Copy to or subclass from another module
  - Move to an object library
- PL/SQL program units:
  - Copy and paste text
  - Copy and paste within a module
  - Copy to or subclass in another module
  - Create a library module
  - Move to an object library

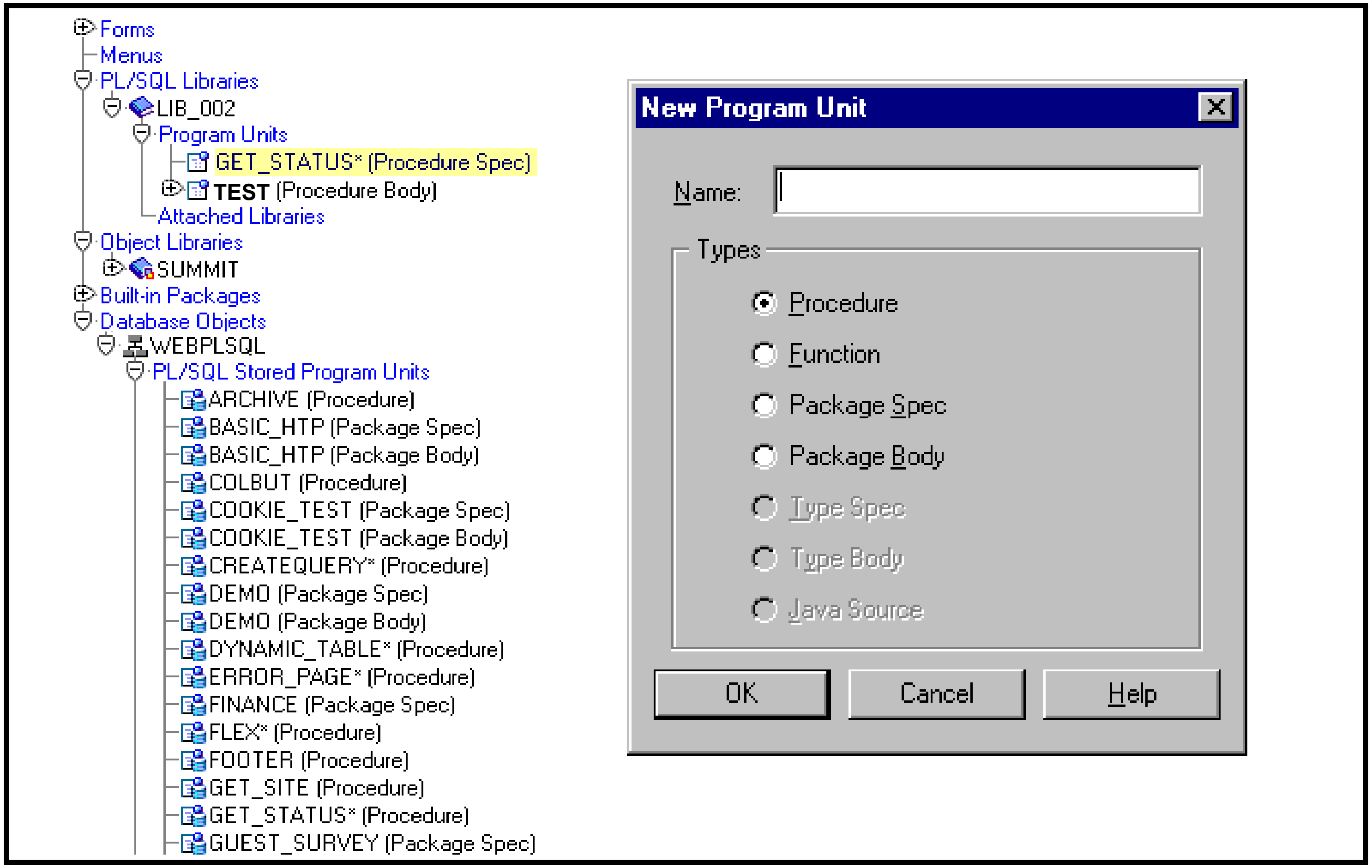
What Are PL/SQL Libraries?



Writing Code for Libraries

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

Creating Library Program Units



- Forms
- Menus
- PL/SQL Libraries
  - LIB_002
    - Program Units
      - GET_STATUS* (Procedure Spec)
      - **TEST** (Procedure Body)
    - Attached Libraries
- Object Libraries
  - SUMMIT
- Built-in Packages
- Database Objects
  - WEBPLSQL
    - PL/SQL Stored Program Units
      - ARCHIVE (Procedure)
      - BASIC_HTP (Package Spec)
      - BASIC_HTP (Package Body)
      - COLBUT (Procedure)
      - COOKIE_TEST (Package Spec)
      - COOKIE_TEST (Package Body)
      - CREATEQUERY* (Procedure)
      - DEMO (Package Spec)
      - DEMO (Package Body)
      - DYNAMIC_TABLE* (Procedure)
      - ERROR_PAGE* (Procedure)
      - FINANCE (Package Spec)
      - FLEX* (Procedure)
      - FOOTER (Procedure)
      - GET_SITE (Procedure)
      - GET_STATUS* (Procedure)
      - GUEST_SURVEY (Package Spec)

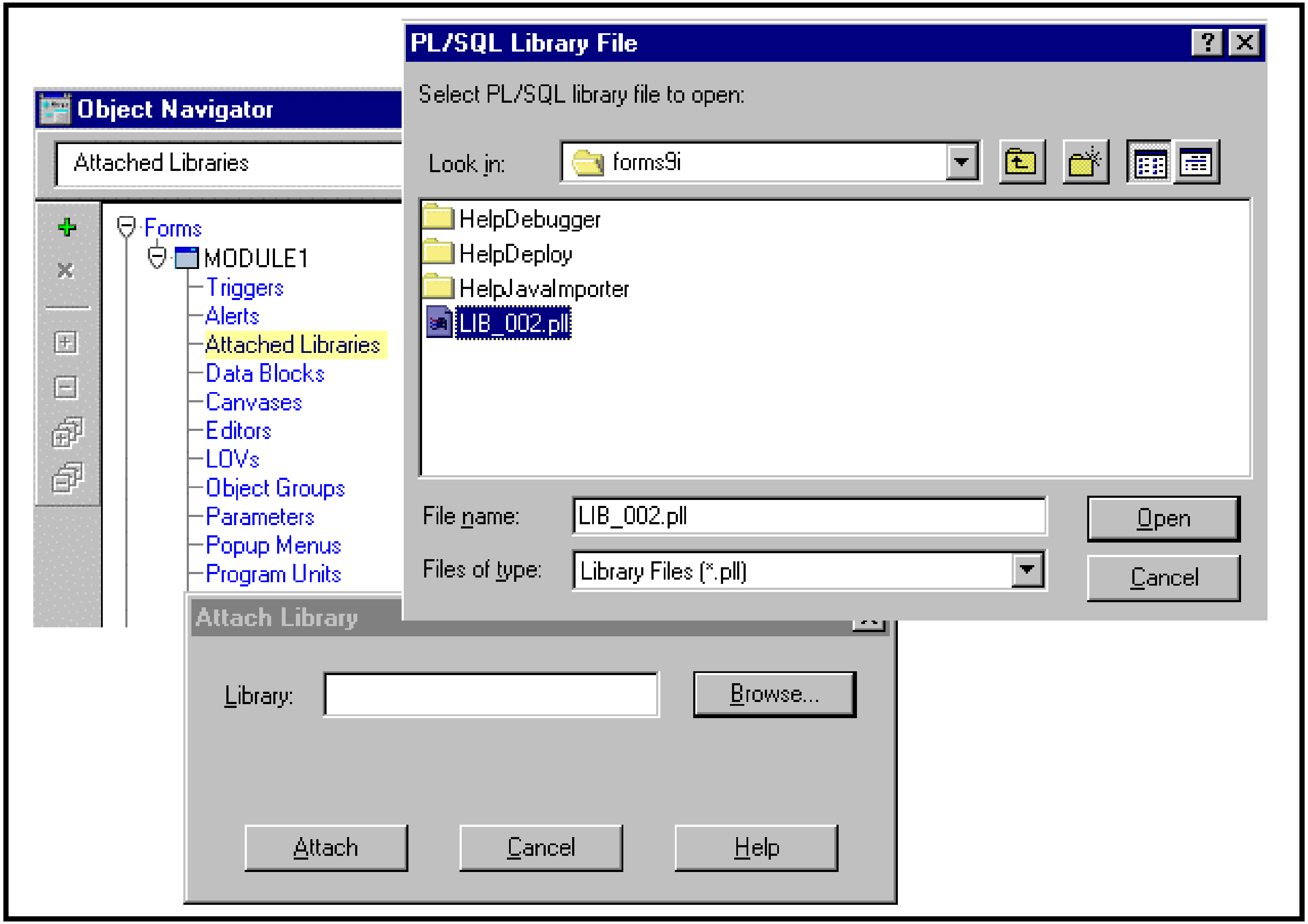
New Program Unit

Name:

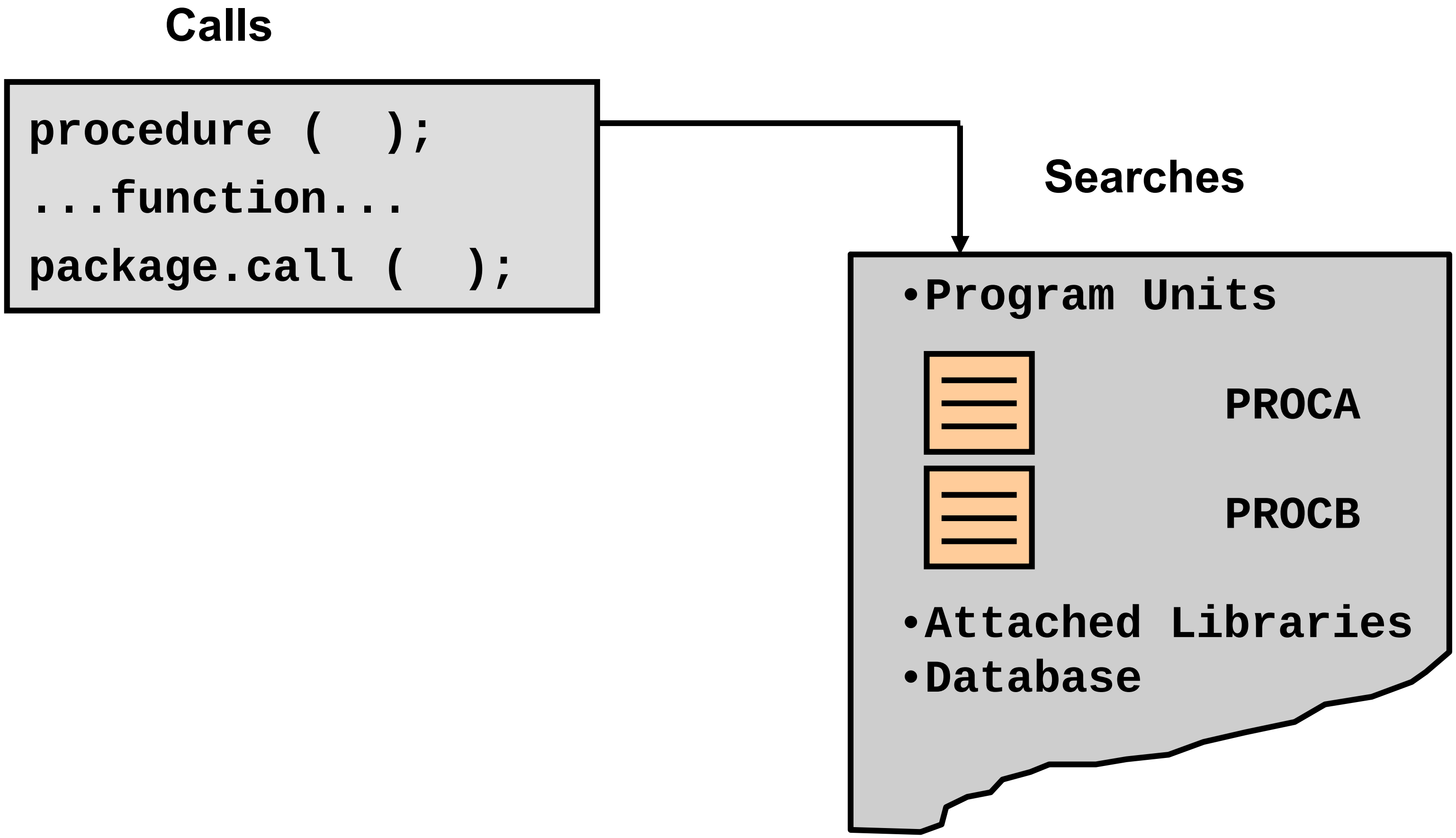
Types

● Procedure
○ Function
○ Package Spec
○ Package Body
○ Type Spec
○ Type Body
○ Java Source

Attach Library Dialog Box



Calls and Searches



Calls

```
procedure (   );
...function...
package.call (   );
```

Searches

- Program Units

PROCA

PROCB

- Attached Libraries
- Database

Summary

In this lesson, you should have learned that:

- You can reuse objects or code in the following ways:
  - Property Classes
  - Object Groups
  - Copying and subclassing
  - Object Libraries and SmartClasses
- To inherit properties from a property class, set an item's Subclass Information property.
- You can create an object group in one module to make it easy to reuse related objects in other modules.

Summary

- Inheritance symbols in the Property Palette show whether the value is changed, inherited, overridden, or the default.
- You can drag objects from an object library or mark them as SmartClasses for even easier reuse.
- You can reuse PL/SQL code by:
  - Copying and pasting in the PL/SQL Editor
  - Copying or subclassing
  - Defining program units to call the same code at multiple places within a module
  - Creating PL/SQL library to call the same code from multiple forms

Practice 23 Overview

This practice covers the following topics:

- Creating an object group and using this object group in a new form module
- Using property classes
- Creating an object library and using this object library in a new form module
- Modifying an object in the object library and observing the effect on subclassed objects
- Setting and using SmartClasses
- Creating a PL/SQL program unit to be called from multiple triggers

24

Using WebUtil to Interact with the Client

Objectives

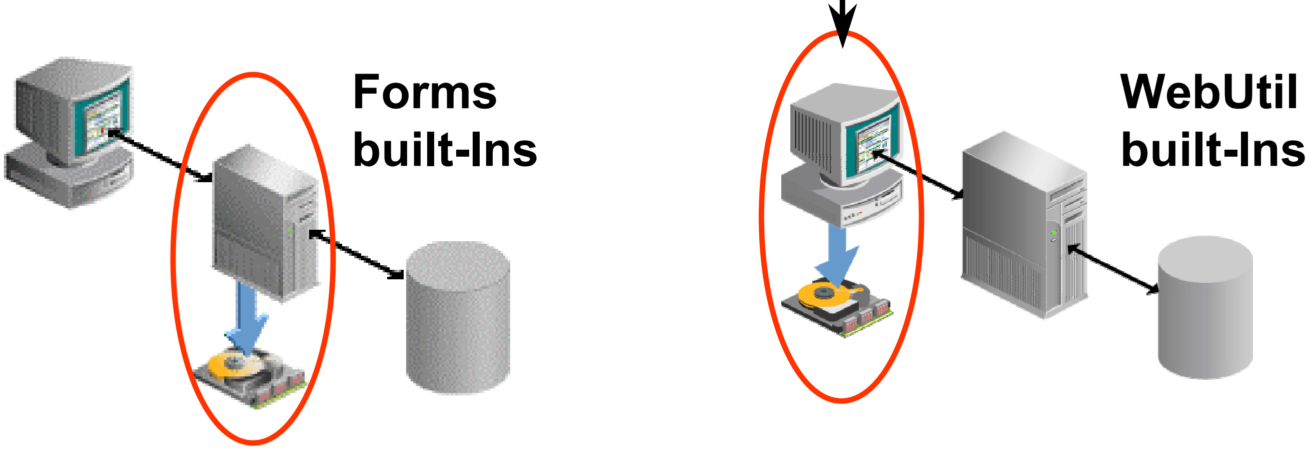
After completing this lesson, you should be able to do the following:

- Describe the benefits of the WebUtil utility
- Integrate WebUtil into a form
- Use WebUtil to interact with a client machine

WebUtil Overview

WebUtil is a utility that:

- Enables you to provide client-side functionality on Win32 clients



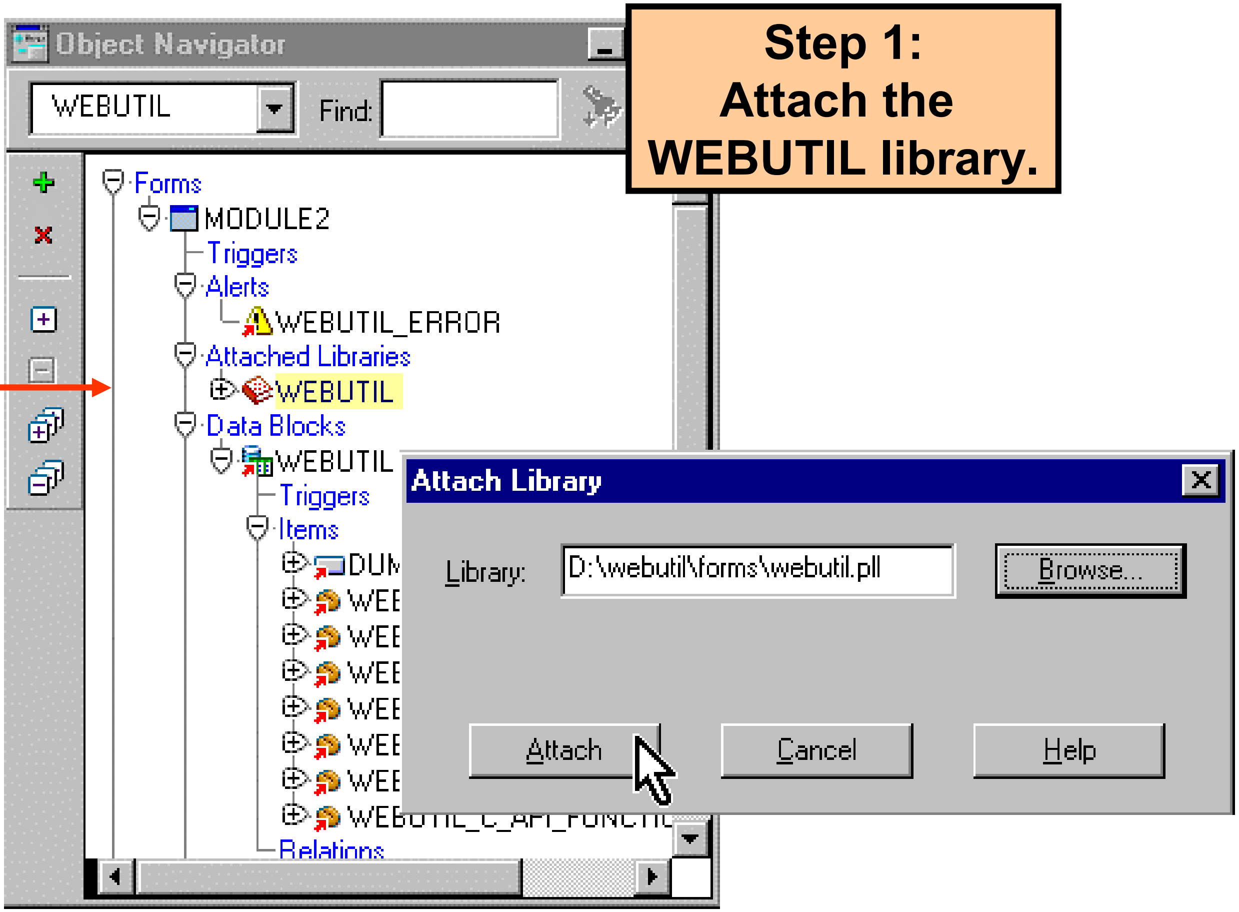
- Consists of:
  - Java classes
  - Forms objects
  - PL/SQL library

Benefits of the WebUtil Utility

Why use WebUtil?

- Developer has only to code in PL/SQL (no Java knowledge required)
- Free download (part of Forms 10*g* in a patch set)
- Easy to integrate into a Forms application
- Extensible
- WebUtil provides:
  - Client-server parity APIs
  - Client-server added value functions
  - Public functions
  - Utility functions
  - Internal functions

Integrating WebUtil into a Form



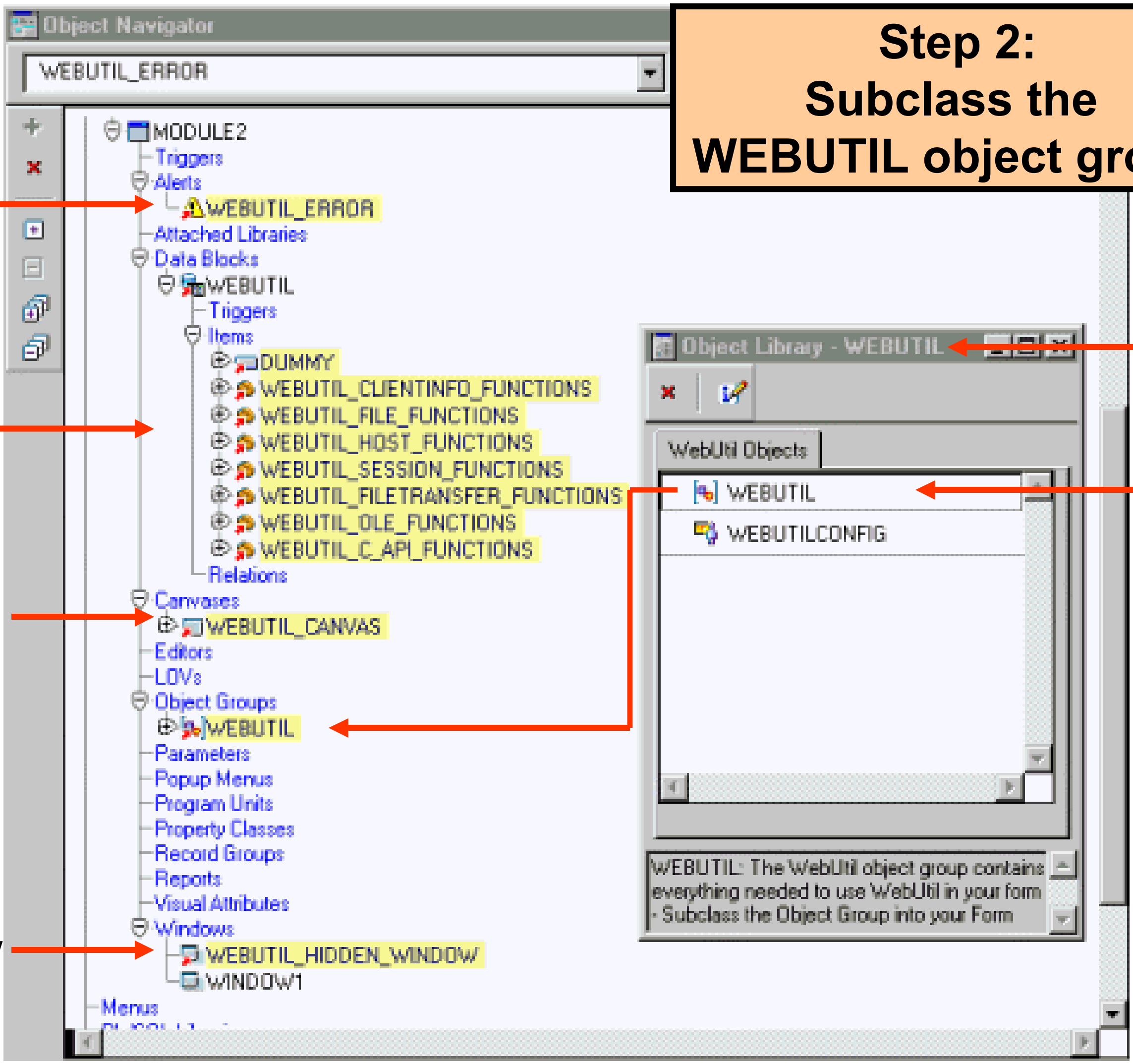
Library: D:\webutil\forms\webutil.pll

Step 1: Attach the WEBUTIL library.

Library

Integrating WebUtil into a Form

Step 2:
Subclass the WEBUTIL object group.



Object Navigator

WEBUTIL_ERROR

```
MODULE2
  Triggers
  Alerts
    WEBUTIL_ERROR
  Attached Libraries
  Data Blocks
    WEBUTIL
      Triggers
      Items
        DUMMY
        WEBUTIL_CLIENTINFO_FUNCTIONS
        WEBUTIL_FILE_FUNCTIONS
        WEBUTIL_HOST_FUNCTIONS
        WEBUTIL_SESSION_FUNCTIONS
        WEBUTIL_FILETRANSFER_FUNCTIONS
        WEBUTIL_OLE_FUNCTIONS
        WEBUTIL_C_API_FUNCTIONS
      Relations
  Canvases
    WEBUTIL_CANVAS
  Editors
  LOVs
  Object Groups
    WEBUTIL
  Parameters
  Popup Menus
  Program Units
  Property Classes
  Record Groups
  Reports
  Visual Attributes
  Windows
    WEBUTIL_HIDDEN_WINDOW
    WINDOW1
Menus
```

Object Library - WEBUTIL

WebUtil Objects

- WEBUTIL
- WEBUTILCONFIG

WEBUTIL: The WebUtil object group contains everything needed to use WebUtil in your form - Subclass the Object Group into your Form

Alert

Items

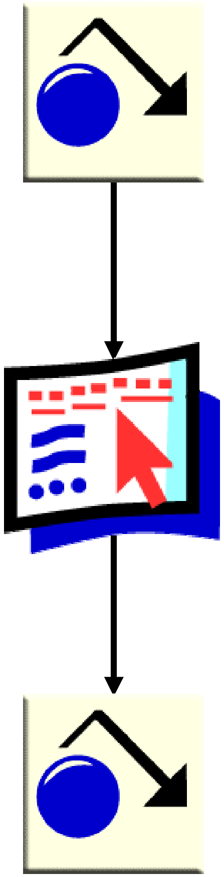
Canvas

Window

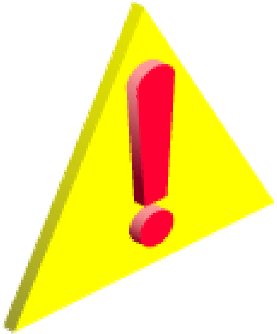
Object Library

Object group

When to Use WebUtil Functionality



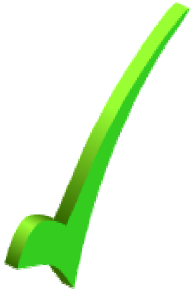
`Pre-Form`
`When-New-Form-Instance`
`When-New-Block-Instance`
(first block)



Form starts

JavaBeans are instantiated

Any trigger after form starts and while form is running

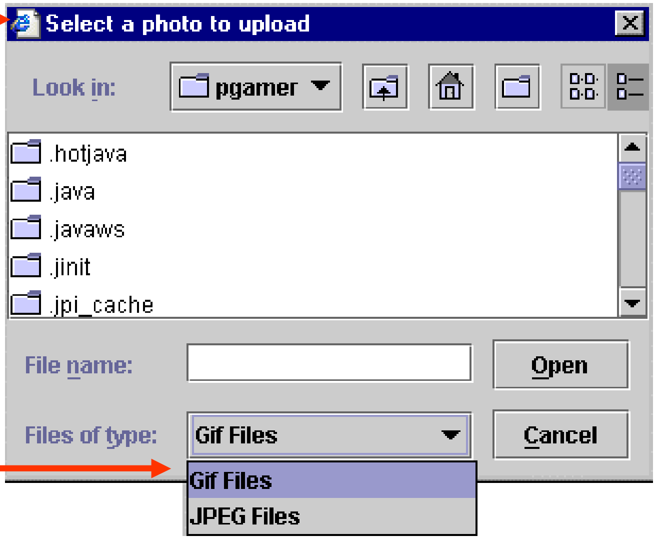


Interacting with the Client

Forms Built-Ins / Packages	WebUtil Equivalents
HOST	CLIENT_HOST
GET_FILE_NAME	CLIENT_GET_FILE_NAME
READ_IMAGE_FILE WRITE_IMAGE_FILE	CLIENT_IMAGE.READ (WRITE)_IMAGE_FILE
OLE2	CLIENT_OLE2
TEXT_IO	CLIENT_TEXT_IO
TOOL_ENV	CLIENT_TOOL_ENV

Example: Opening a File Dialog on the Client

```
DECLARE
  v_file VARCHAR2(250):= CLIENT_GET_FILE_NAME('','',
    'Gif Files|*.gif|JPEG Files|*.jpg|',
    'Select a photo to upload',open_file,TRUE);
```



Select a photo to upload

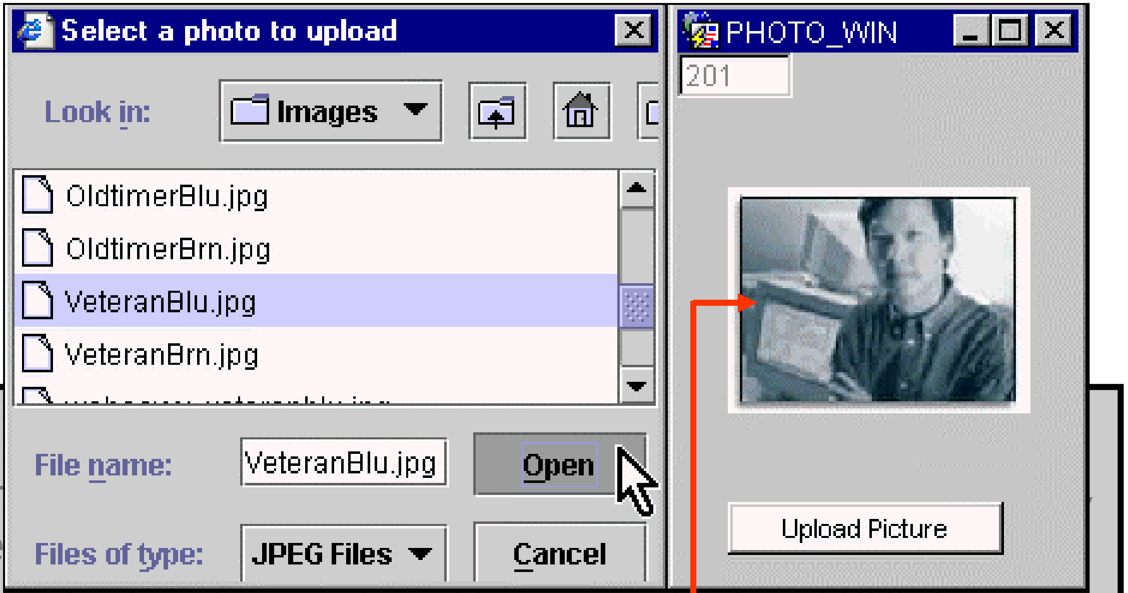
Look in: pgamer

.hotjava
.java
.javaws
.jinit
.jpi_cache

File name:

Files of type: Gif Files

Example: Reading an Image File into Forms from the Client



Select a photo to upload

Look in: Images

OldtimerBlu.jpg
OldtimerBrn.jpg
VeteranBlu.jpg
VeteranBrn.jpg

File name: VeteranBlu.jpg

Files of type: JPEG Files

PHOTO_WIN

201

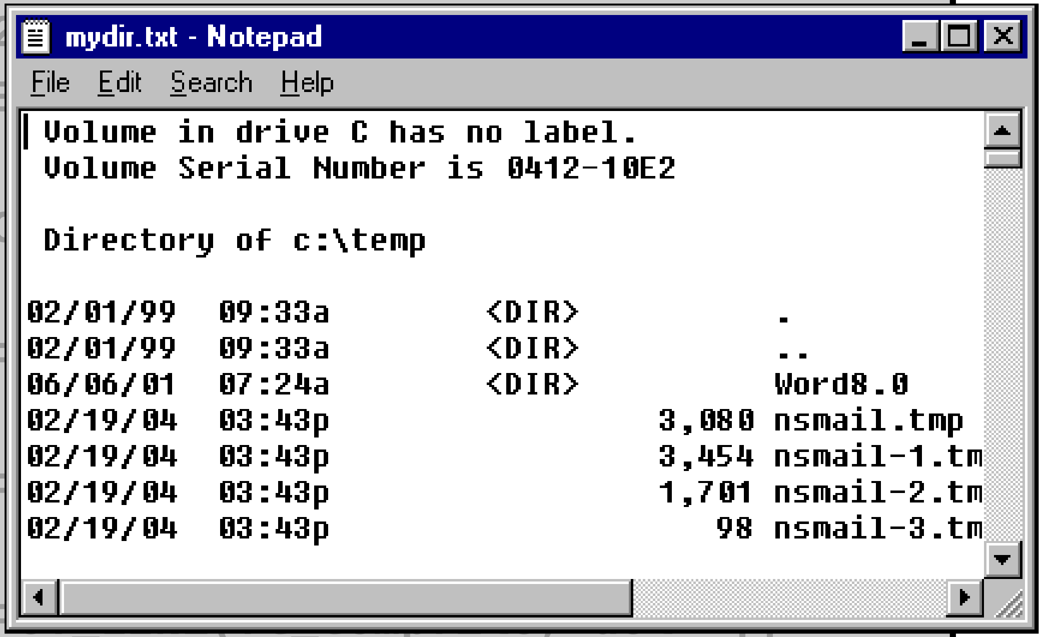
```
DECLARE
   v_file VARCH
      'Gif File
      'Select a photo to upload',open_file,TRUE);
   it_image_id ITEM := FIND_ITEM
      ('employee_photos.photo');
BEGIN
  CLIENT_IMAGE.READ_IMAGE_FILE(v_file,'',it_image_id);
END;
```

Example: Writing Text Files on the Client

```
DECLARE
  v_dir VARCHAR2(250) := 'c:\temp';
  ft_tempfile CLIENT_TEXT_IO.FILE_TYPE;                   1
begin
  ft_tempfile := CLIENT_TEXT_IO.FOPEN(v_dir ||            2
     '\tempdir.bat','w');
  CLIENT_TEXT_IO.PUT_LINE(ft_tempfile,'dir ' ||
     v_dir || '> '|| v_dir || '\mydir.txt');
  CLIENT_TEXT_IO.PUT_LINE(ft_tempfile,                    3
     'notepad ' || v_dir || '\mydir.txt');
  CLIENT_TEXT_IO.PUT_LINE(ft_tempfile,'del '||
     v_dir || '\mydir.*');
  CLIENT_TEXT_IO.FCLOSE(ft_tempfile);                     4
  CLIENT_HOST('cmd /c ' || v_dir || '\tempdir');
END;
```

Example: Executing Operating System Commands on the Client

```
DECLARE
  v_dir VARCHAR2(2
  ft_tempfile CLIE
begin
  ft_tempfile := C
    '\tempdir.bat
  CLIENT_TEXT_IO.F
    v_dir || '> '
  CLIENT_TEXT_IO.F
    'notepad ' ||
  CLIENT_TEXT_IO.F
    v_dir || '\mydir.*');
  CLIENT_TEXT_IO.FCLOSE(ft_tempfile);
  CLIENT_HOST('cmd /c ' || v_dir || '\tempdir');
END;
```

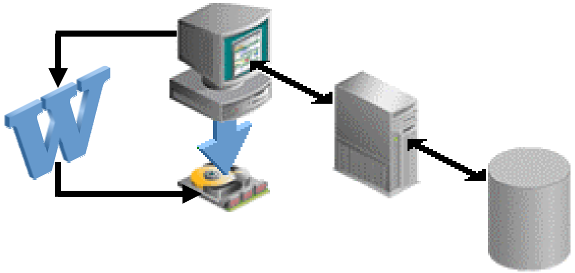


```
 Volume in drive C has no label.
 Volume Serial Number is 0412-10E2

 Directory of c:\temp

02/01/99  09:33a        <DIR>          .
02/01/99  09:33a        <DIR>          ..
06/06/01  07:24a        <DIR>          Word8.0
02/19/04  03:43p                 3,080 nsmail.tmp
02/19/04  03:43p                 3,454 nsmail-1.tm
02/19/04  03:43p                 1,701 nsmail-2.tm
02/19/04  03:43p                    98 nsmail-3.tm
```

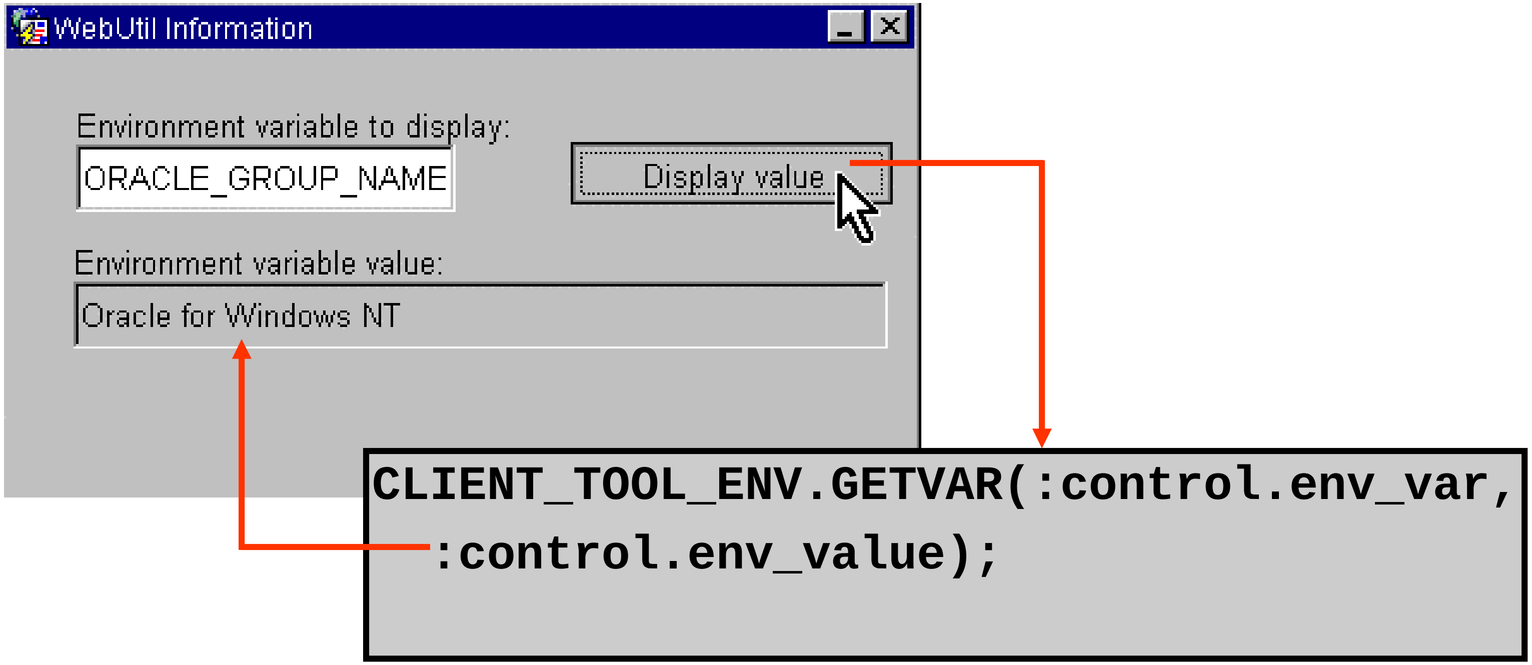
Example: Performing OLE Automation on the Client

You can use the following for OLE automation:

- CLIENT_OLE2.OBJ_TYPE
- CLIENT_OLE2.LIST_TYPE
- CLIENT_OLE2.CREATE_OBJ
- CLIENT_OLE2.SET _PROPERTY
- CLIENT_OLE2.GET_OBJ _PROPERTY
- CLIENT_OLE2.INVOKE_OBJ
- CLIENT_OLE2.CREATE _ARGLIST
- CLIENT_OLE2.ADD_ARG
- CLIENT_OLE2.INVOKE
- CLIENT_OLE2.DESTROY _ARGLIST
- CLIENT_OLE2.RELEASE_OBJ



Example: Obtaining Environment Information about the Client



Environment variable to display: ORACLE_GROUP_NAME

Environment variable value: Oracle for Windows NT

```
CLIENT_TOOL_ENV.GETVAR(:control.env_var,
  :control.env_value);
```

Summary

In this lesson, you should have learned that:

- WebUtil is a free extensible utility that enables you to interact with the client machine
- Although WebUtil uses Java classes, you code in PL/SQL
- You integrate WebUtil into a form by attaching its PL/SQL library and using an object group from its object library; then you can use its functions after the form has started and while it is running
- With WebUtil, you can do the following on the client machine: open a file dialog box, read and write image or text files, execute operating system commands, perform OLE automation, and obtain information about the client machine

Practice 24 Overview

This practice covers the following topics:

- Integrating WebUtil with a form
- Using WebUtil functions to:
  - Open a file dialog box on the client
  - Read an image file from the client into the form
  - Obtain the value of a client environment variable
  - Create a file on the client
  - Open the file on the client with Notepad
  - Use OLE automation to create a form letter on the client

25

Introducing Multiple Form Applications

Objectives

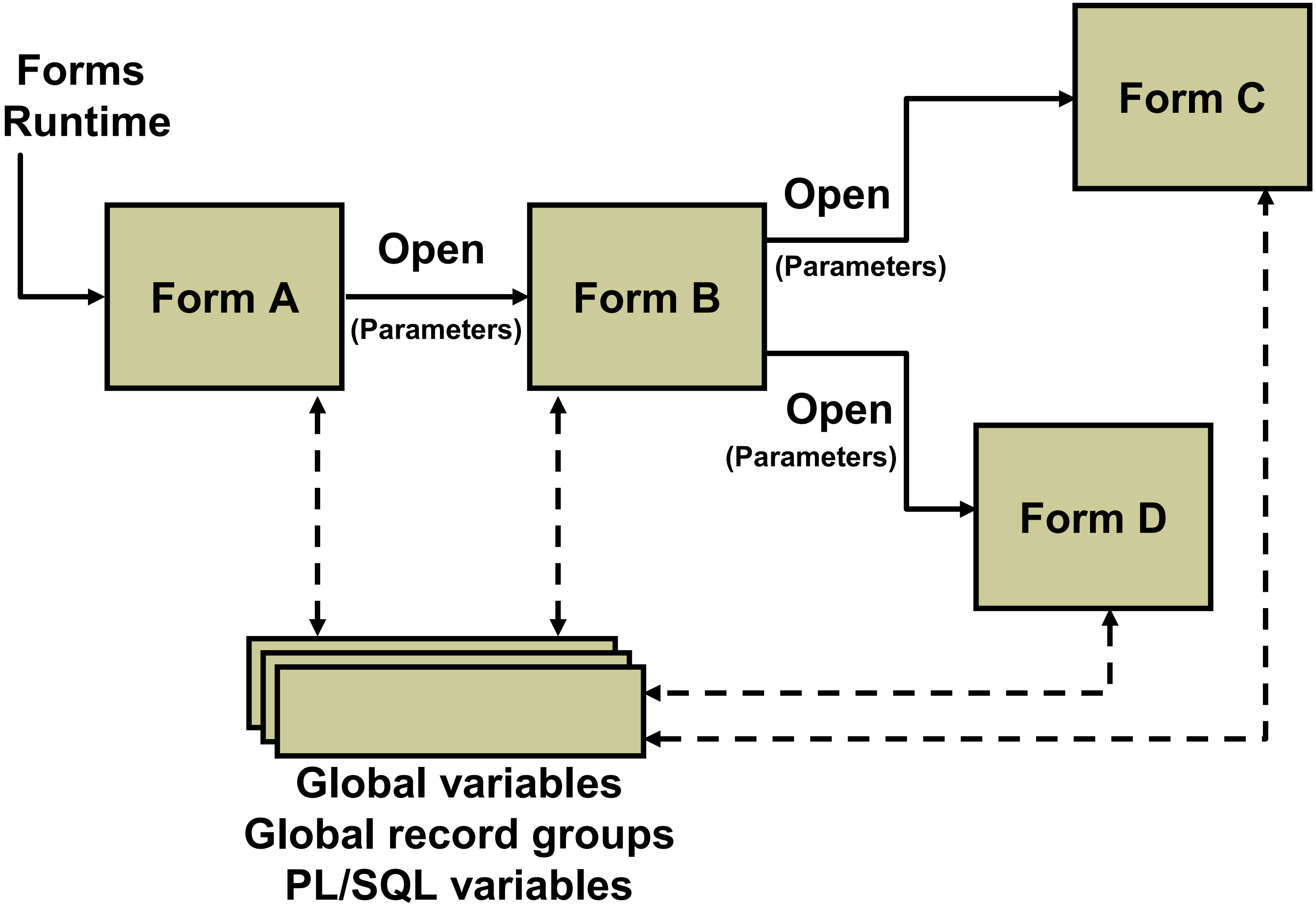
After completing this lesson, you should be able to do the following:

- Call one form from another form module
- Define multiple form functionality
- Share data among open forms

Multiple Form Applications Overview

- Behavior:
  - Flexible navigation between windows
  - Single or multiple database connections
  - Transactions may span forms, if required
  - Commits in order of opening forms, starting with current form
- Links:
  - Data is exchanged by global variables, parameter lists, global record groups, or PL/SQL variables in shared libraries
  - Code is shared as required, through libraries and the database

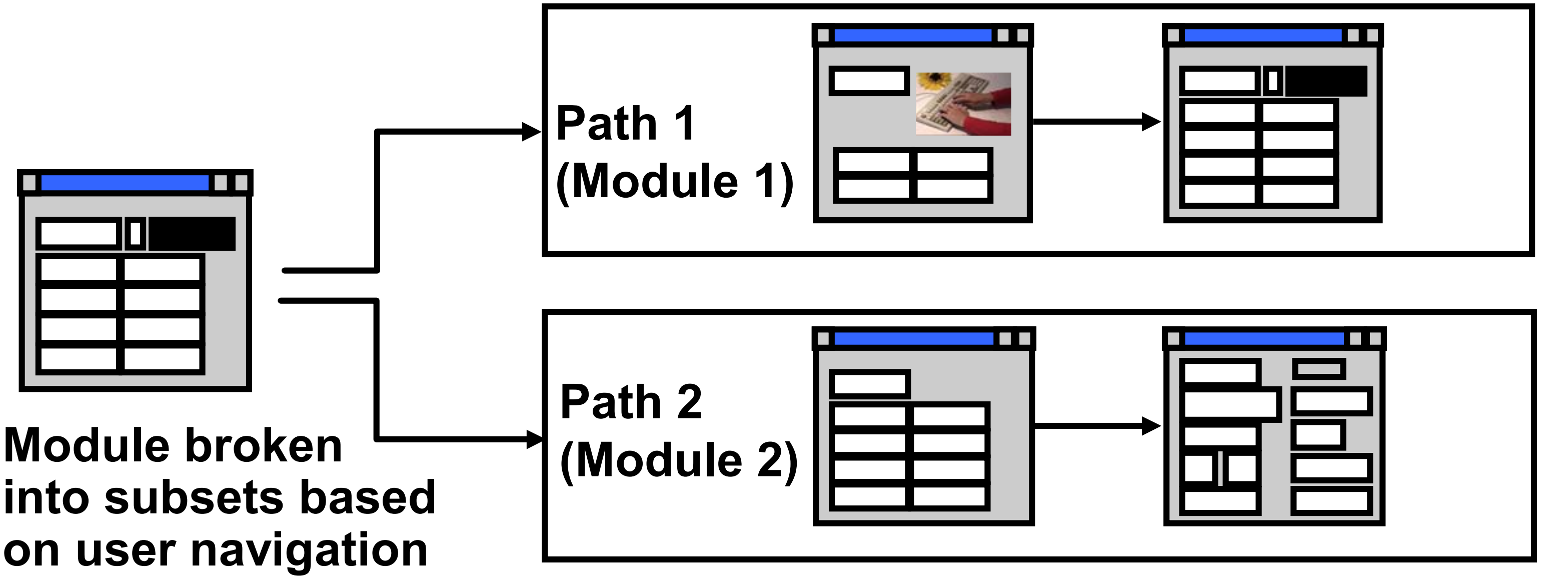
Multiple Form Session



Benefits of Multiple Form Applications

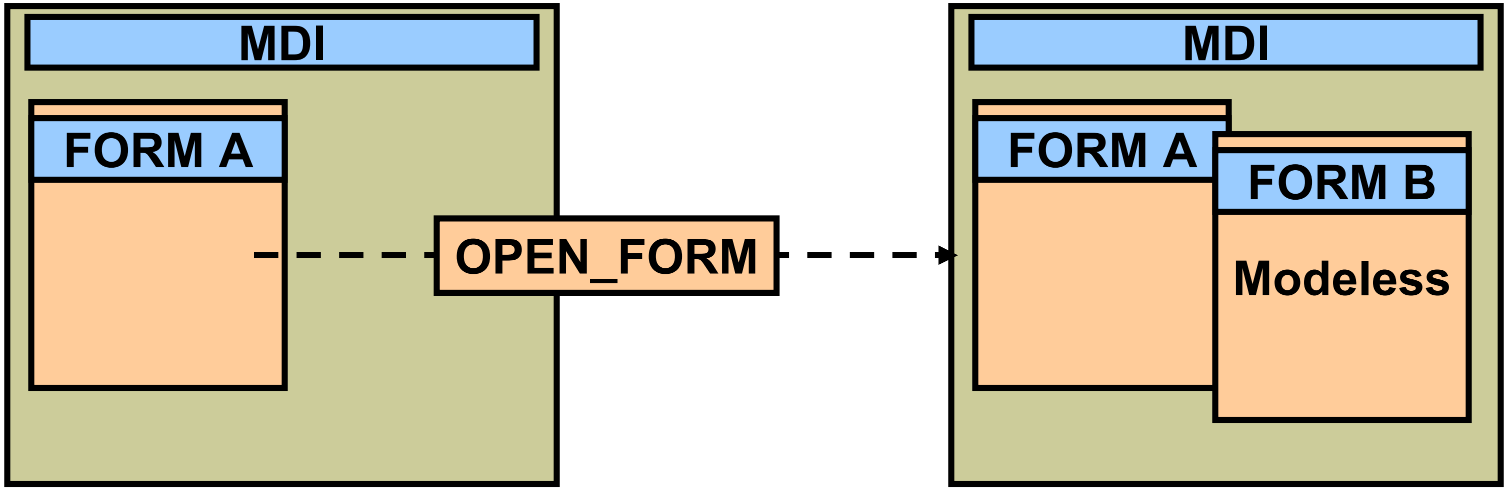
Breaking your application into multiple forms offers the following advantages:

- Easier debugging
- Modularity
- Performance and scalability



Starting Another Form Module

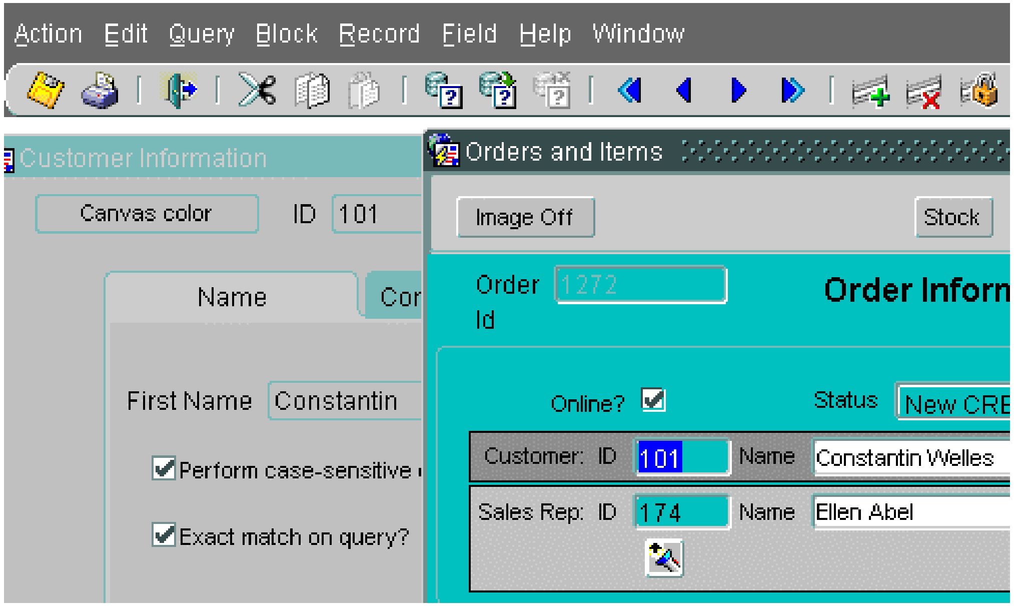
`OPEN_FORM`



Defining Multiple Form Functionality

Summit application scenario:

- Run the CUSTOMERS and ORDERS forms in the same session, navigating freely between them.
- You can make changes in the same transaction across forms.
- All forms are visible together.



ID 101

First Name Constantin

☑ Perform case-sensitive

☑ Exact match on query?

Order Id 1272

Online? ☑

Status New CRE

Customer: ID 101 Name Constantin Welles

Sales Rep: ID 174 Name Ellen Abel

Defining Multiple Form Functionality

Actions:

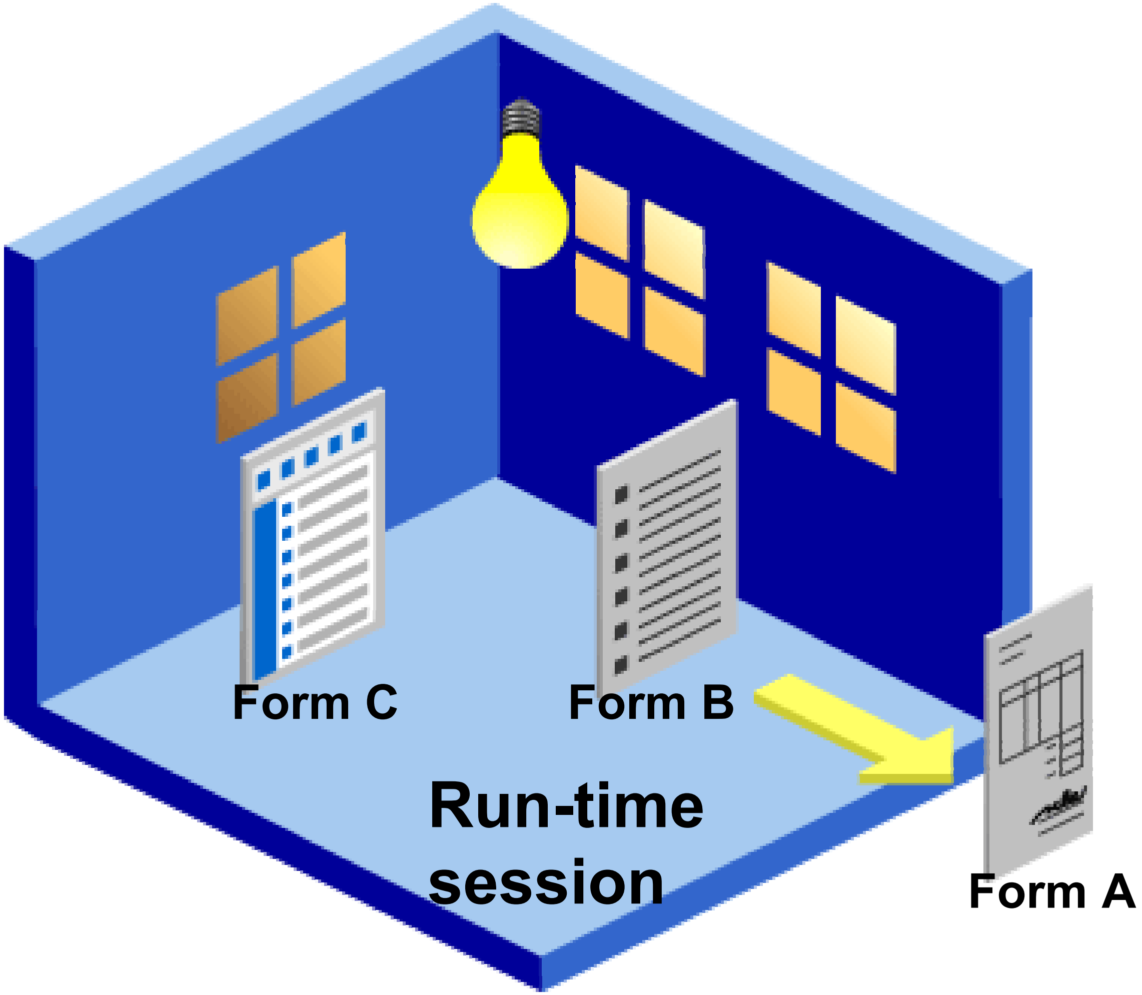
1. Define windows and positions for each form.
2. Plan shared data, such as global variables and their names.
3. Implement triggers to:
   - Open other forms
   - Initialize shared data from calling forms
   - Use shared data in opened forms

Conditional Opening

Example

```
IF  ID_NULL(FIND_FORM(’ORDERS’)) THEN
     OPEN_FORM(’ORDERS’);
ELSE
     GO_FORM(’ORDERS’);
END IF;
```

Closing the Session



"Will the last one out please turn off the lights?"

Closing a Form with EXIT_FORM

- **The default functionality is the same as for the Exit key.**
- **The Commit_Mode argument defines action on uncommitted changes.**

```
ENTER;
IF  :SYSTEM.FORM_STATUS = ’CHANGED’ THEN
    EXIT_FORM( DO_COMMIT );
ELSE
    EXIT_FORM( NO_COMMIT );
END IF;
```

Other Useful Triggers

Maintain referential links and synchronize data between forms:

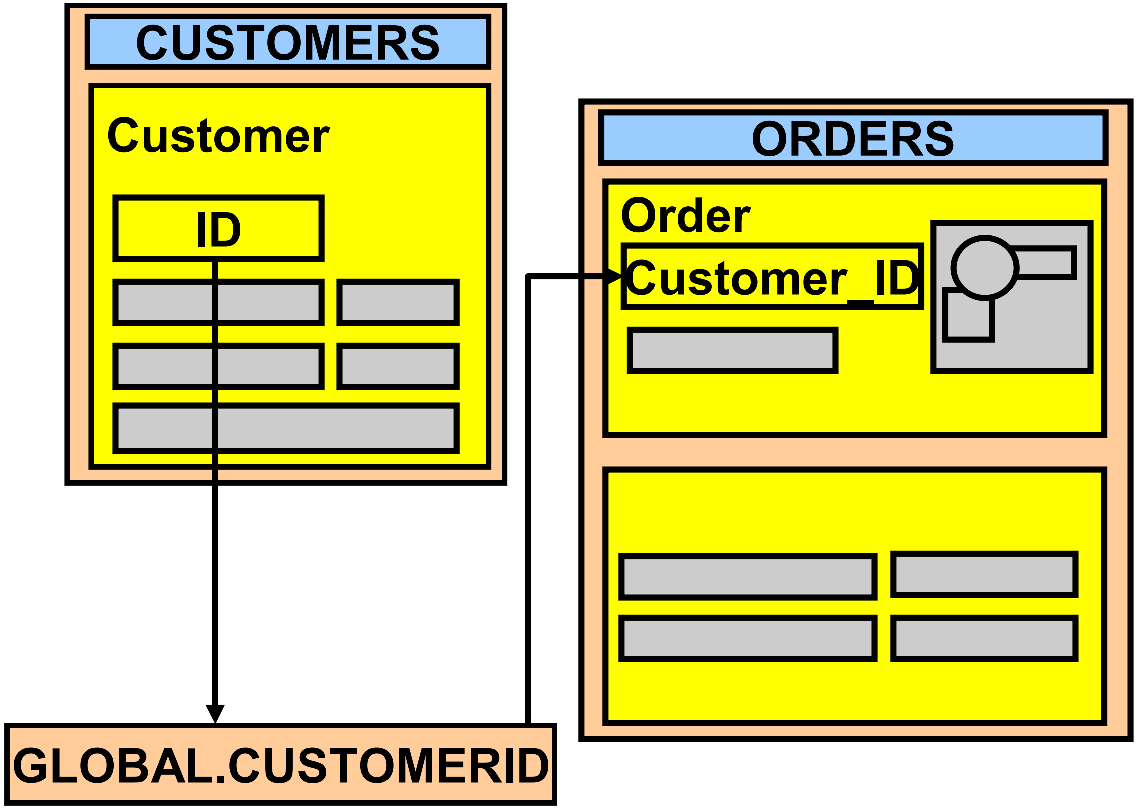
- **In the parent form:**
  - `When-Validate-Item`
  - `When-New-Record-Instance`
- **In opened forms:** `When-Create-Record`
- **In any form:** `When-Form-Navigate`

Sharing Data Among Modules

You can pass data between modules using:

- Global variables
- Parameter lists
- Global record groups
- PL/SQL package variables in shared libraries

Linking by Global Variables



Global Variables: Opening Another Form

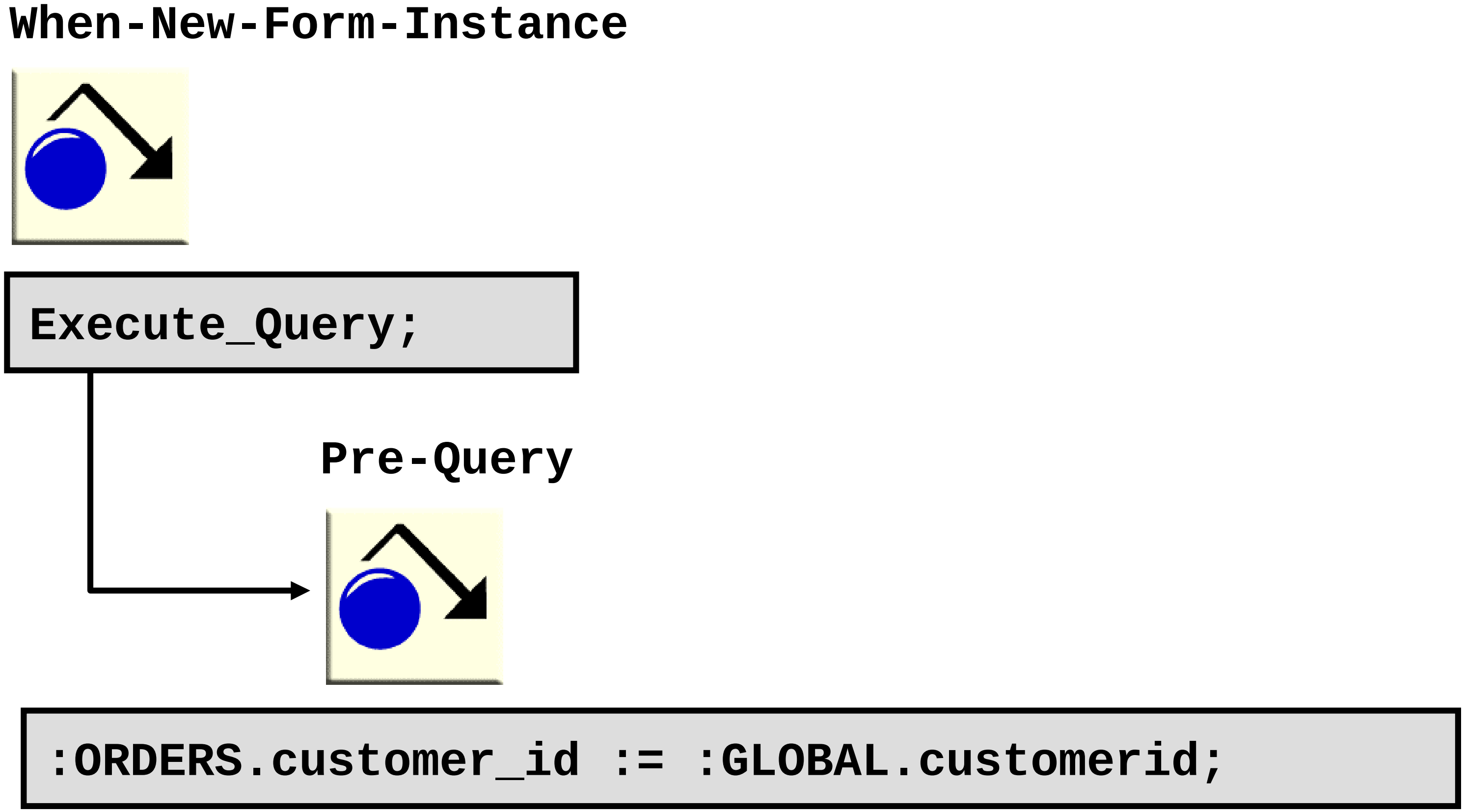
Example

```
:GLOBAL.customerid := :CUSTOMERS.customer_id;
OPEN_FORM('ORDERS');
```

Notes

- Control passes immediately to the `ORDERS` form—no statements after `OPEN_FORM` are processed.
- If the Activate_Mode argument is set to `NO_ACTIVATE`, you retain control in the current form.
- The transaction continues unless it was explicitly committed before.

Global Variables: Restricted Query at Startup



When-New-Form-Instance

```
Execute_Query;
```

Pre-Query

```
:ORDERS.customer_id := :GLOBAL.customerid;
```

Assigning Global Variables in the Opened Form

- `DEFAULT_VALUE` ensures the existence of globals.
- You can use globals to communicate that the form is running.

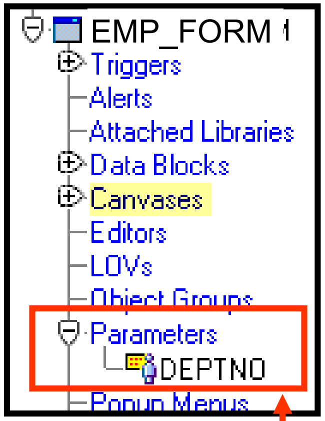
Pre-Form example:

```
DEFAULT_VALUE(’’, ’GLOBAL.customerid’);
```

Linking by Parameter Lists

Parameters:

- Are form module objects
- Properties:
  - Name
  - Parameter Data Type
  - Maximum Length
  - Parameter Initial Value
- Can optionally receive a new value:



```
http://myhost:8889/forms90/f90servlet
?form=emp.fmx&otherparams=deptno=140
```

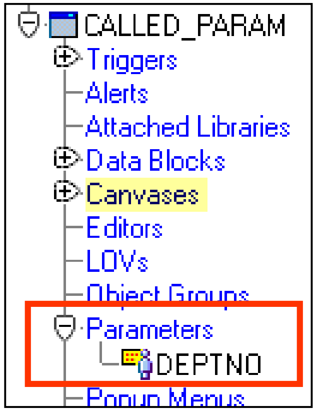
Linking by Parameter Lists

Example:

```
DECLARE
    pl_id   ParamList;
    pl_name VARCHAR2(10) := 'tempdata';
BEGIN
    pl_id := GET_PARAMETER_LIST(pl_name);
    IF ID_NULL(pl_id) THEN
1    pl_id := CREATE_PARAMETER_LIST(pl_name);
    ELSE
     DELETE_PARAMETER(pl_id,'deptno');
    END IF;
2   ADD_PARAMETER(pl_id,'deptno',TEXT_PARAMETER,
       to_char(:departments.department_id));
3   OPEN_FORM('called_param',ACTIVATE,NO_SESSION,pl_id);
END;
```

Linking by Parameter Lists

Example: Called form



Create parameter in the form

When-New-Form-Instance Trigger

```
IF :parameter.deptno IS NOT NULL THEN
   SET_BLOCK_PROPERTY('employees',
      DEFAULT_WHERE,'department_id =
      '||:parameter.deptno);
   SET_WINDOW_PROPERTY('window1',
      TITLE,'Employees in Department  ’
      ||:parameter.deptno);
END IF;
GO_BLOCK('employees');
EXECUTE_QUERY;
```

Use parameter name preceded by `:parameter`

Linking by Global Record Groups

1. Create record group with global scope:

```
DECLARE
  rg_name    VARCHAR2(40) := 'LIST';
  rg_id      RecordGroup;
  Error_Flag NUMBER;
BEGIN
  rg_id := FIND_GROUP(rg_name);
  IF ID_NULL(rg_id) THEN
    rg_id := CREATE_GROUP_FROM_QUERY('LIST',
      'Select last_name, to_char(employee_id)
      from employees',GLOBAL_SCOPE);
  END IF;
```

2. Populate record group:

```
Error_Flag := POPULATE_GROUP(rg_id);
```

3. Use record group in any form.

Linking by Shared PL/SQL Variables

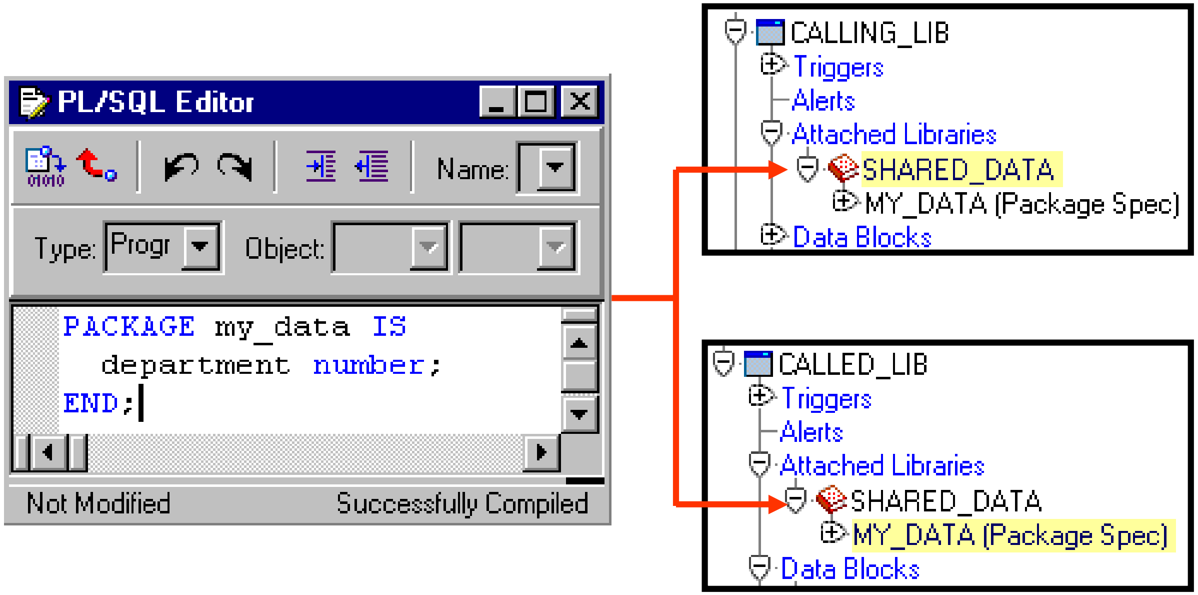
Advantages:

- Use less memory than global variables
- Can be of any data type

To use:

1. Create a PL/SQL library.
2. Create a package specification with variables.
3. Attach the library to multiple forms.
4. Set variable values in calling form.
5. `OPEN_FORM` with `SHARE_LIBRARY_DATA` option.
6. Use variables in opened form.

Linking by Shared PL/SQL Variables



Type: Progr

```
PACKAGE my_data IS
  department number;
END;
```

Not Modified   Successfully Compiled

```
OPEN_FORM('called_lib',ACTIVATE,
  NO_SESSION,SHARE_LIBRARY_DATA);
```

Summary

In this lesson, you should have learned that:

- `OPEN_FORM` is the primary method to call one form from another form module
- You define multiple form functionality such as:
  - Whether all forms run in the same session
  - Where the windows appear
  - Whether multiple forms should be open at once
  - Whether users should be able to navigate among open forms
  - How data will be shared among forms

Summary

- You can share data among open forms with:
  - Global variables, which span sessions
  - Parameter lists, for passing values between specific forms
  - Record groups created in one form with global scope
  - PL/SQL variables in shared libraries

Practice 25 Overview

This practice covers the following topics:

- Using a global variable to link ORDERS and CUSTOMERS forms
- Using built-ins to check whether the ORDERS form is running
- Using global variables to restrict a query in the ORDERS form

C

Introduction to Query Builder

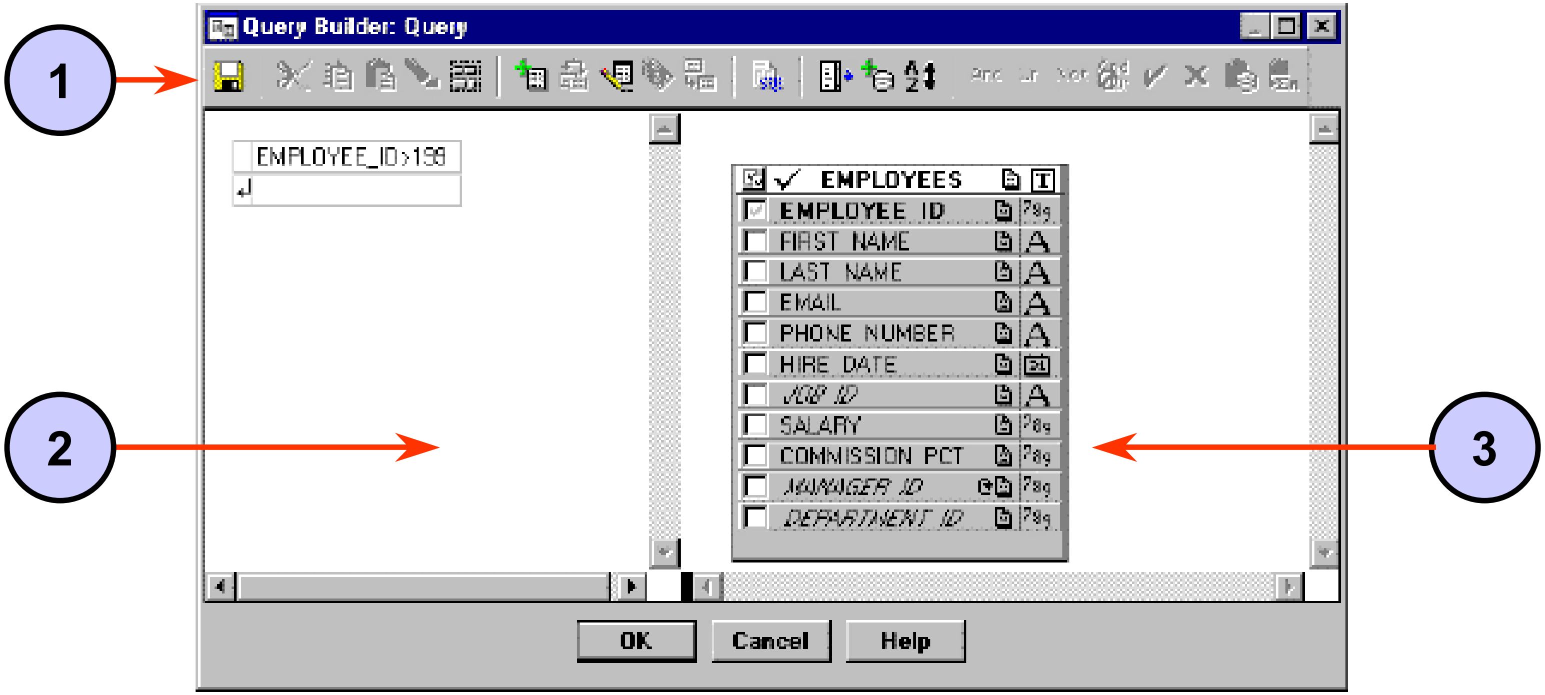
Query Builder Features

- Easy-to-use data access tool
- Point-and-click graphical user interface
- Distributed data access
- Powerful query building

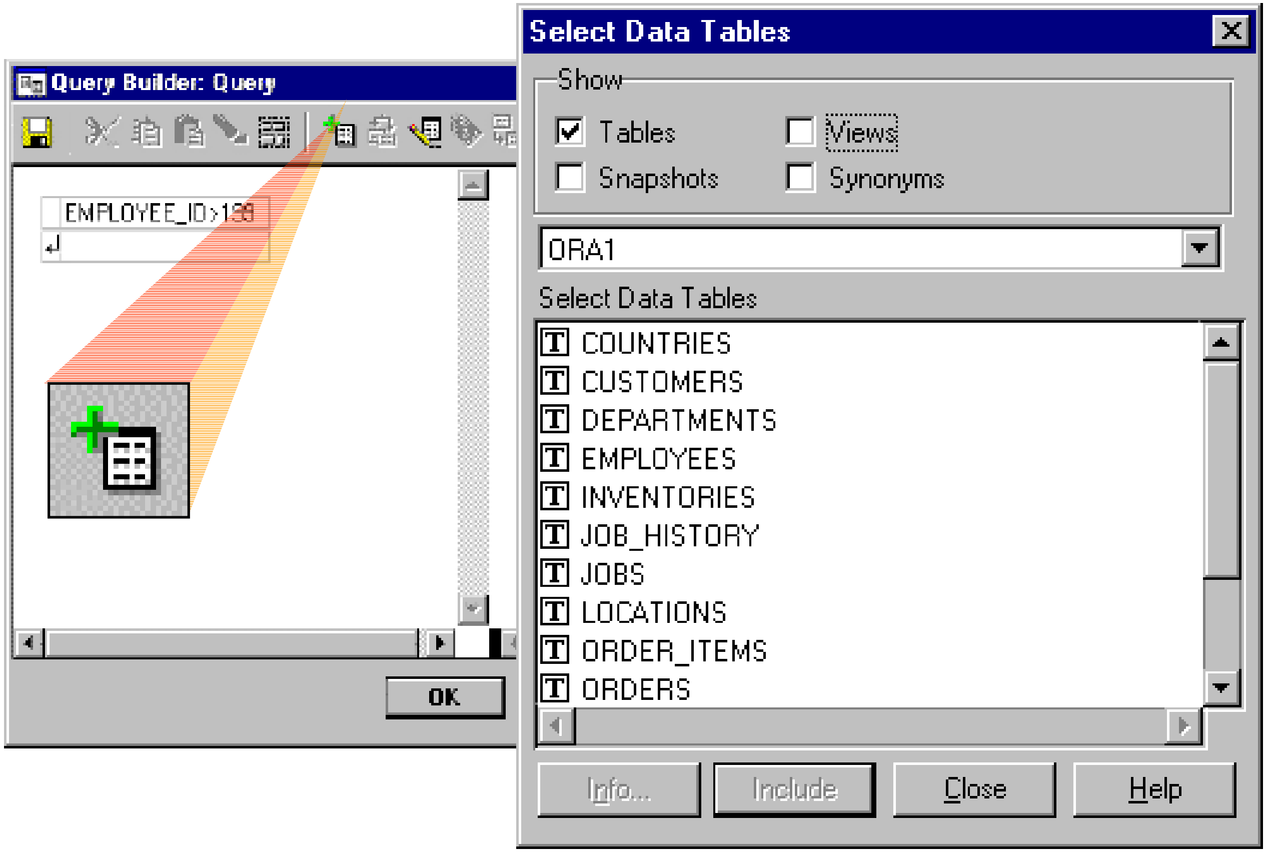
Query Builder Features

- Easy-to-use data access tool
- Point-and-click graphical user interface
- Distributed data access
- Powerful query building

Query Builder Window



Building a New Query



Query Builder: Query

EMPLOYEE_ID>138

Select Data Tables

Show

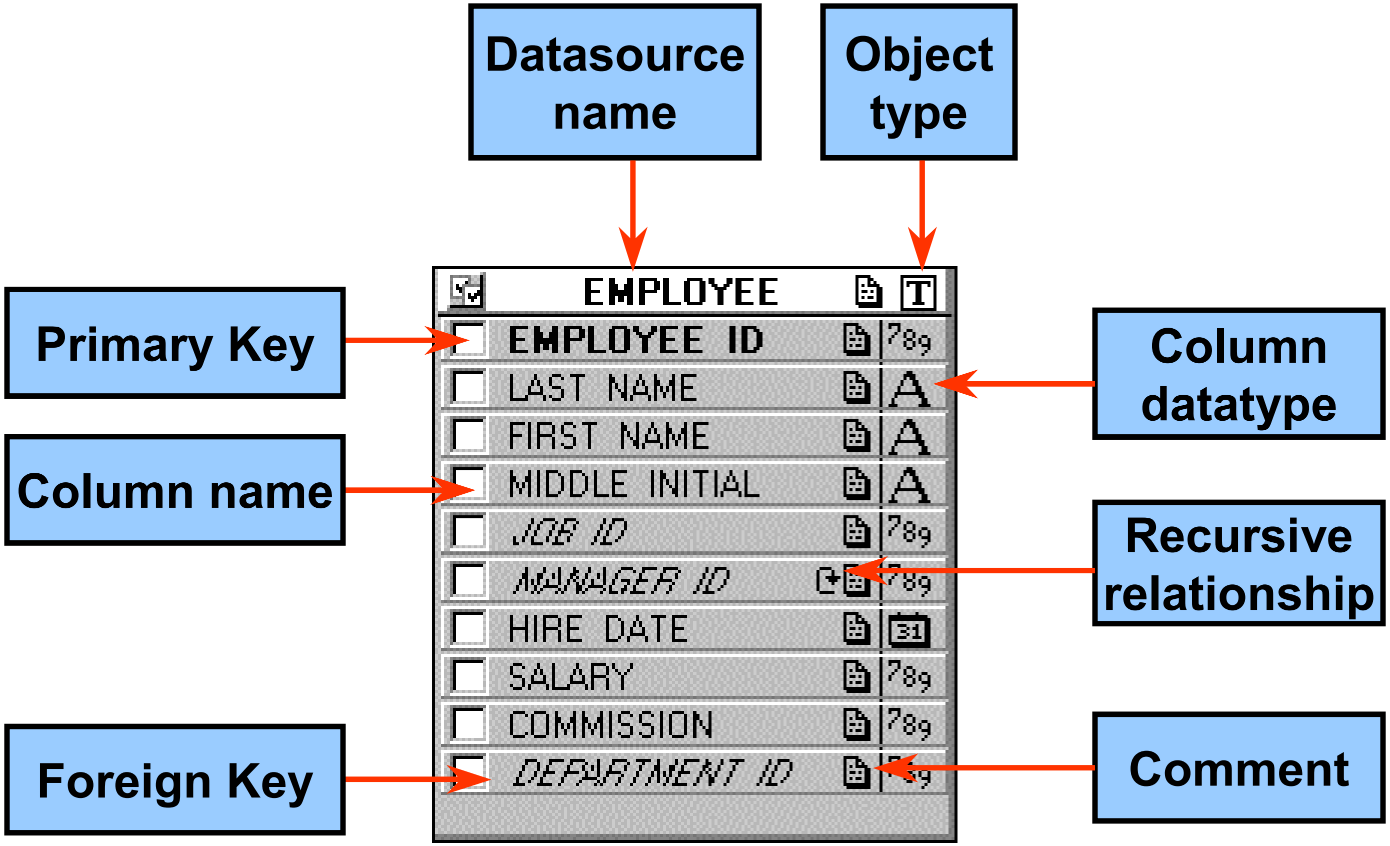
☑ Tables
□ Views
□ Snapshots
□ Synonyms

ORA1

Select Data Tables

- COUNTRIES
- CUSTOMERS
- DEPARTMENTS
- EMPLOYEES
- INVENTORIES
- JOB_HISTORY
- JOBS
- LOCATIONS
- ORDER_ITEMS
- ORDERS

Datasource Components



Datasource name

Object type

Primary Key

Column name

Foreign Key

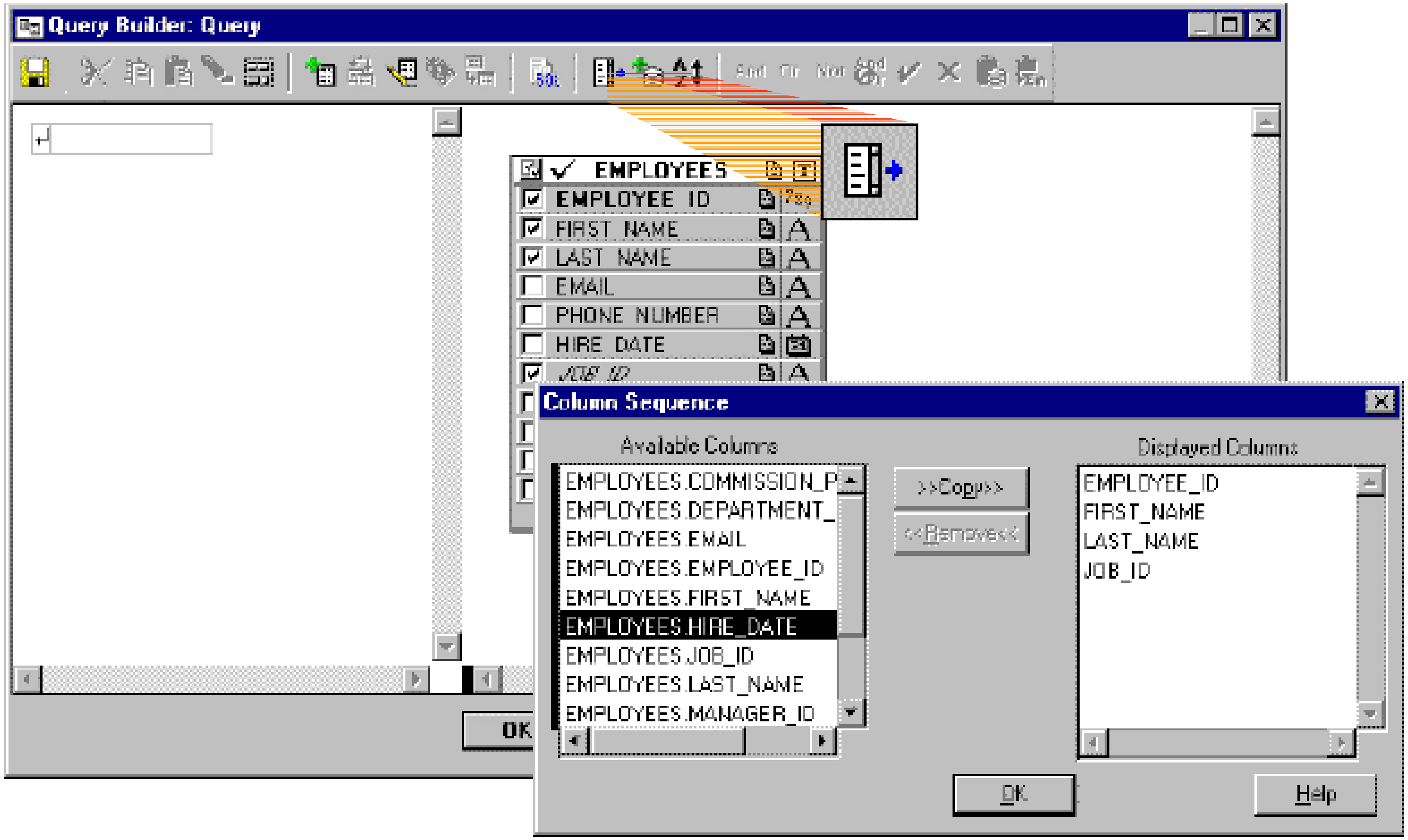
Column datatype

Recursive relationship

Comment

EMPLOYEE
- EMPLOYEE ID
- LAST NAME
- FIRST NAME
- MIDDLE INITIAL
- *JOB ID*
- *MANAGER ID*
- HIRE DATE
- SALARY
- COMMISSION
- *DEPARTMENT ID*

Refining a Query



EMPLOYEES

- ☑ EMPLOYEE_ID
- ☑ FIRST_NAME
- ☑ LAST_NAME
- □ EMAIL
- □ PHONE_NUMBER
- □ HIRE_DATE
- ☑ JOB_ID

Column Sequence

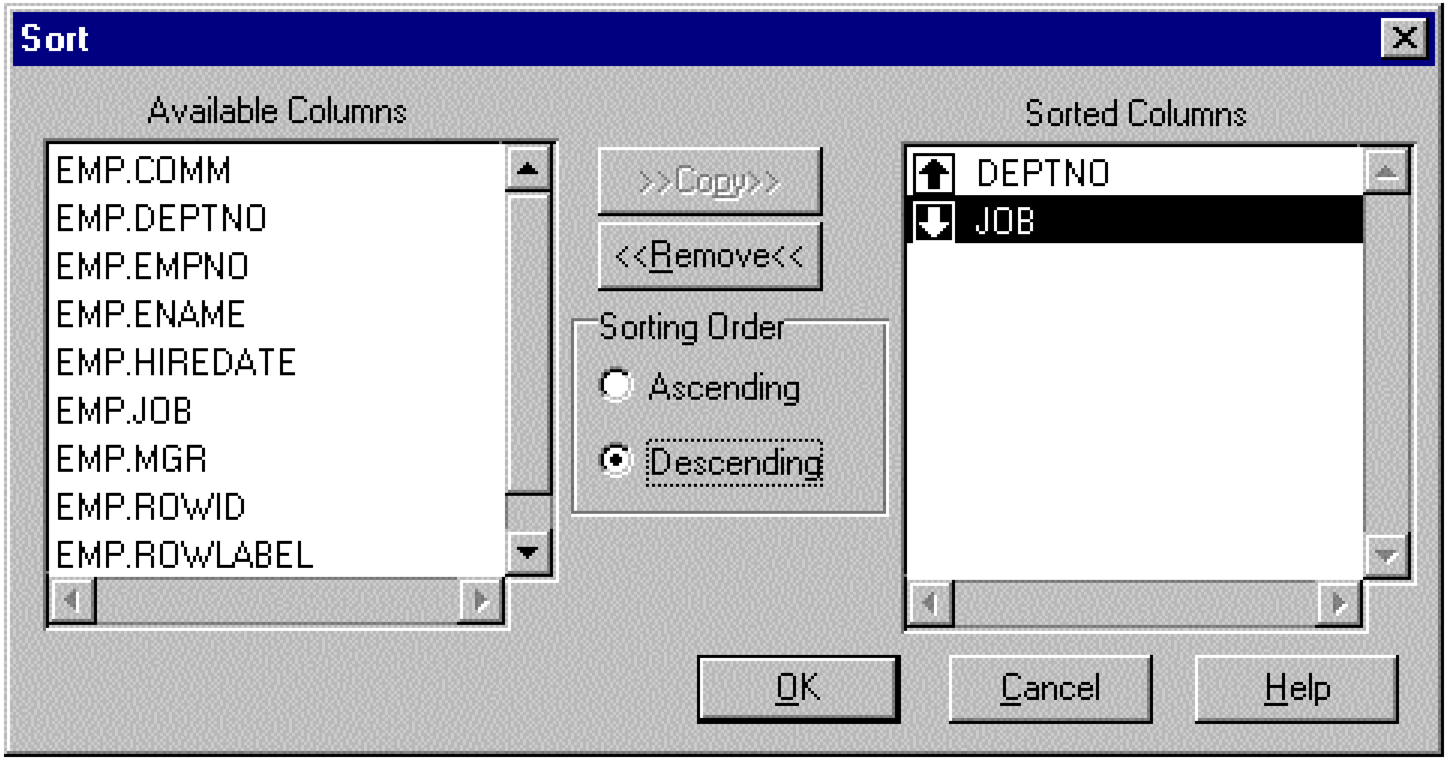
Available Columns

- EMPLOYEES.COMMISSION_P
- EMPLOYEES.DEPARTMENT_
- EMPLOYEES.EMAIL
- EMPLOYEES.EMPLOYEE_ID
- EMPLOYEES.FIRST_NAME
- EMPLOYEES.HIRE_DATE
- EMPLOYEES.JOB_ID
- EMPLOYEES.LAST_NAME
- EMPLOYEES.MANAGER_ID

Displayed Columns

- EMPLOYEE_ID
- FIRST_NAME
- LAST_NAME
- JOB_ID

Sorting Data



Sort

Available Columns

EMP.COMM
EMP.DEPTNO
EMP.EMPNO
EMP.ENAME
EMP.HIREDATE
EMP.JOB
EMP.MGR
EMP.ROWID
EMP.ROWLABEL

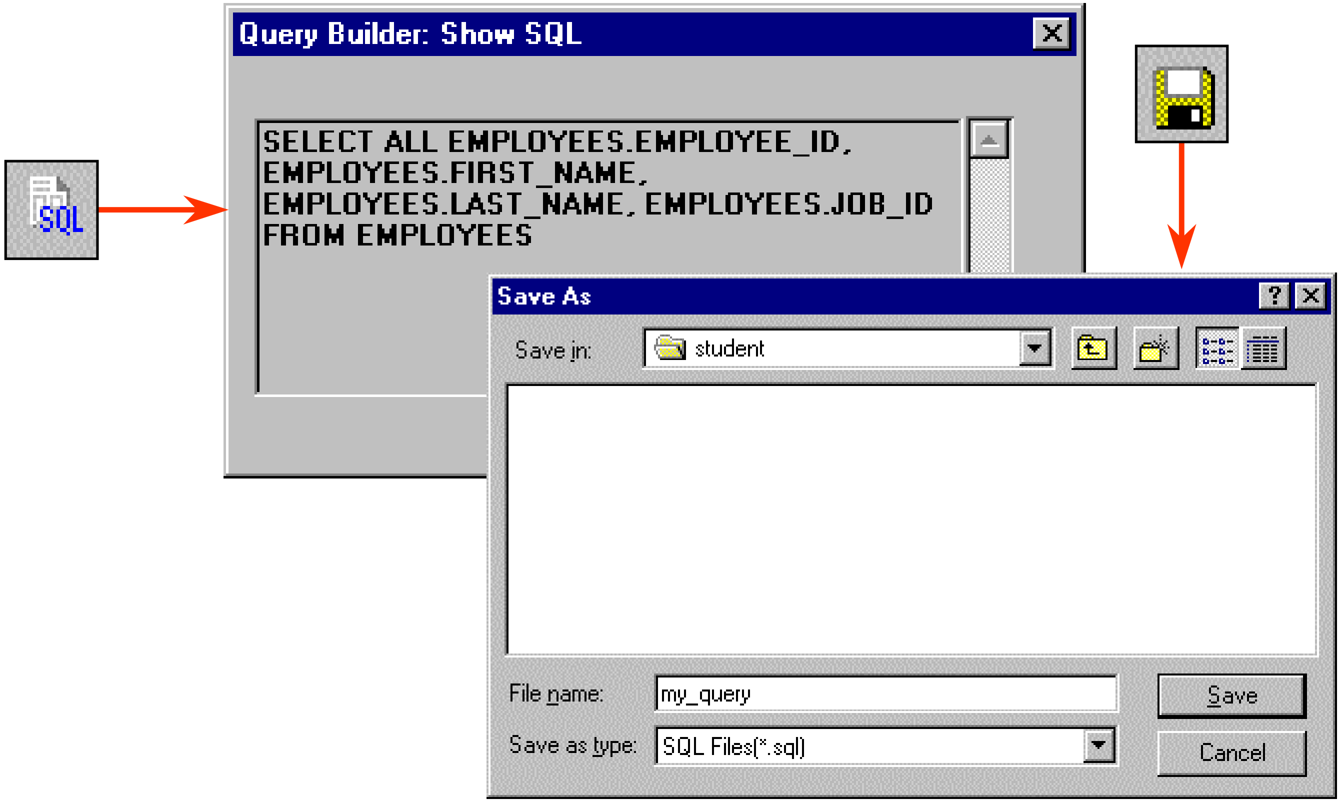
Sorted Columns

⬆ DEPTNO
⬇ JOB

Sorting Order

○ Ascending
● Descending

Viewing and Saving Queries

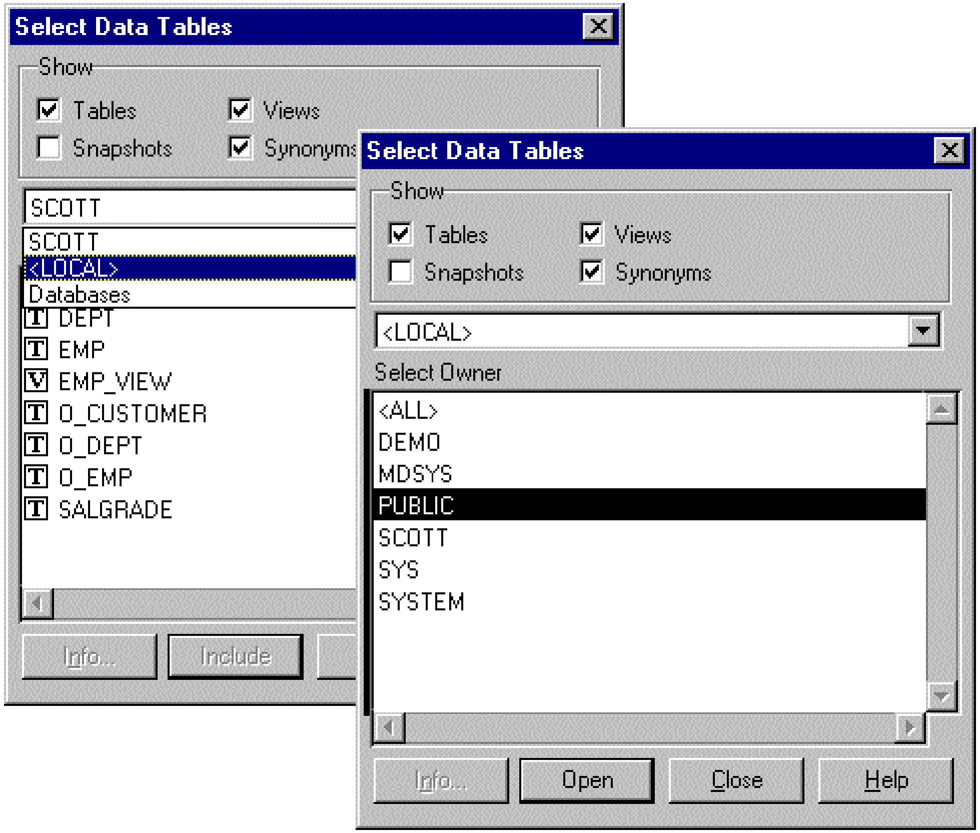


```
SELECT ALL EMPLOYEES.EMPLOYEE_ID,
EMPLOYEES.FIRST_NAME,
EMPLOYEES.LAST_NAME, EMPLOYEES.JOB_ID
FROM EMPLOYEES
```

File name: my_query

Save as type: SQL Files(*.sql)

Including Additional Tables



Select Data Tables

Show

☑ Tables ☑ Views

□ Snapshots ☑ Synonyms

SCOTT

SCOTT

<LOCAL>

Databases

T DEPT

T EMP

V EMP_VIEW

T O_CUSTOMER

T O_DEPT

T O_EMP

T SALGRADE

Select Data Tables

Show

☑ Tables ☑ Views

□ Snapshots ☑ Synonyms

<LOCAL>

Select Owner

<ALL>

DEMO

MDSYS

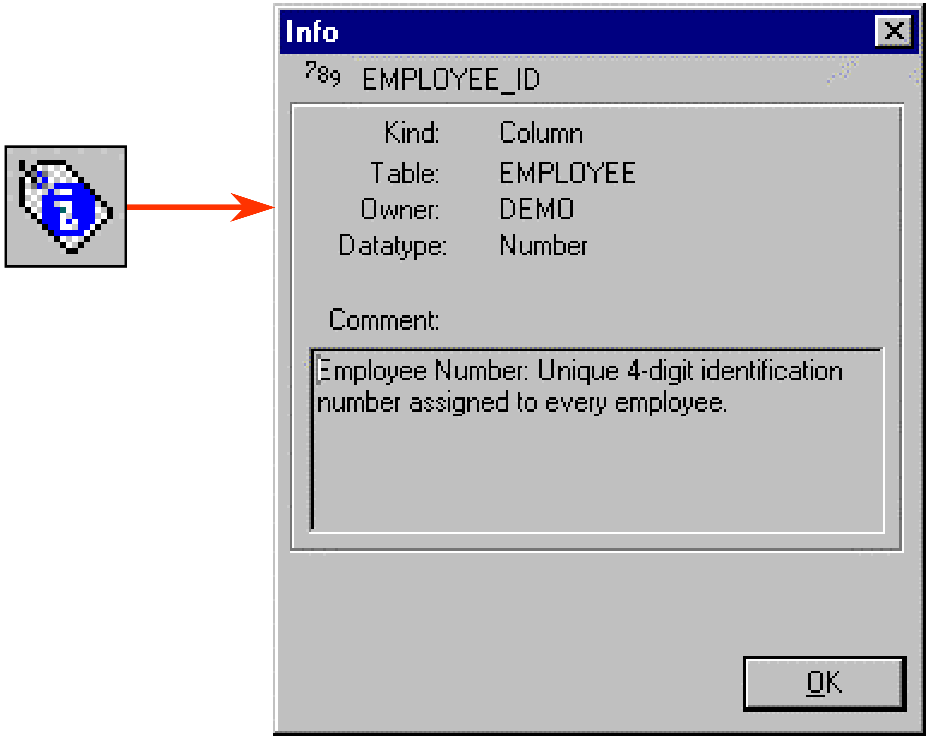
PUBLIC

SCOTT

SYS

SYSTEM

Viewing Comments

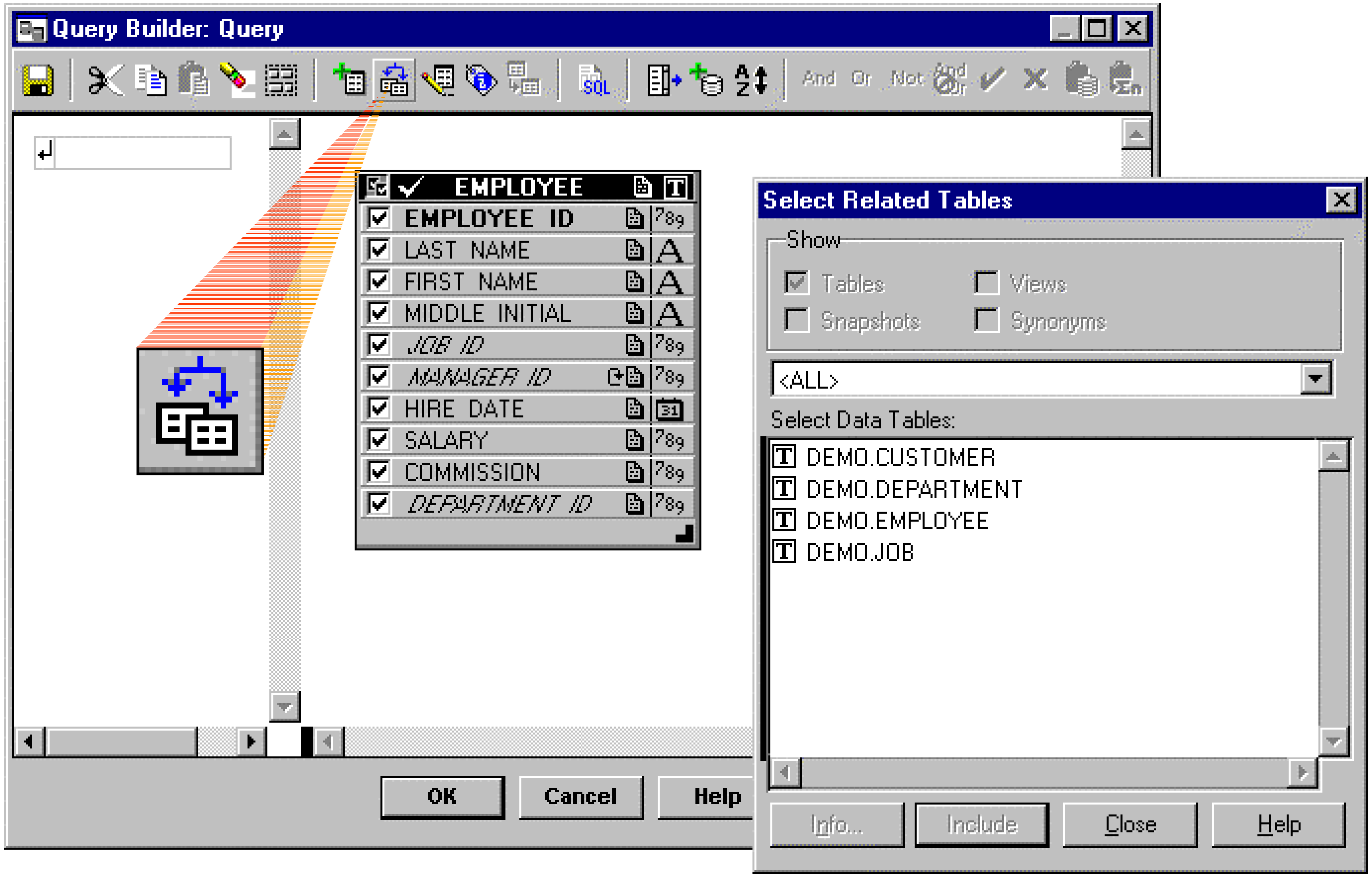


Kind: Column
Table: EMPLOYEE
Owner: DEMO
Datatype: Number

Comment:

Employee Number: Unique 4-digit identification number assigned to every employee.

Including Related Tables



Query Builder: Query

	EMPLOYEE
☑	EMPLOYEE ID
☑	LAST NAME
☑	FIRST NAME
☑	MIDDLE INITIAL
☑	JOB ID
☑	MANAGER ID
☑	HIRE DATE
☑	SALARY
☑	COMMISSION
☑	DEPARTMENT ID

Select Related Tables

Show

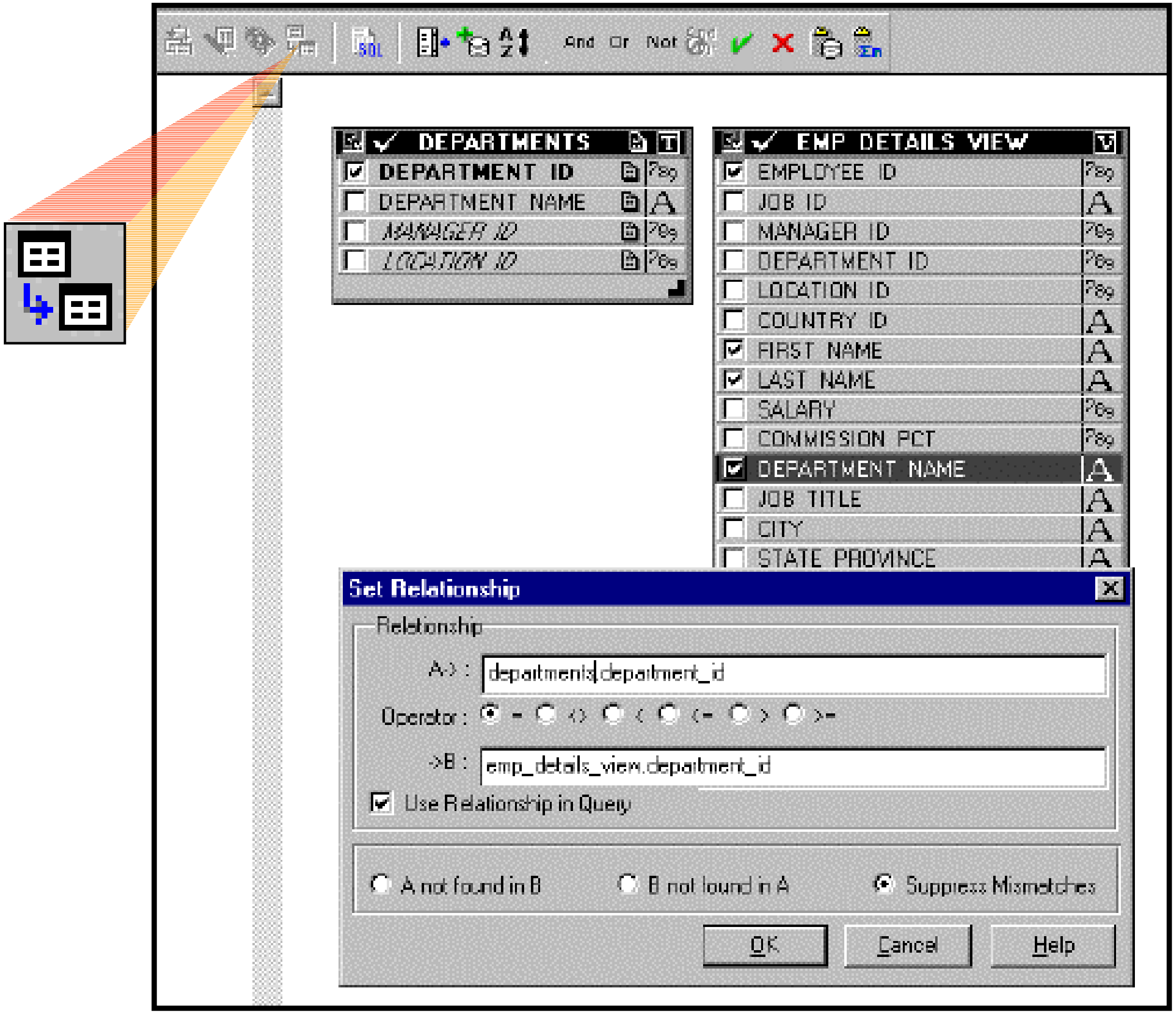
☑ Tables ☐ Views
☐ Snapshots ☐ Synonyms

<ALL>

Select Data Tables:

- DEMO.CUSTOMER
- DEMO.DEPARTMENT
- DEMO.EMPLOYEE
- DEMO.JOB

Creating a User-Defined Relationship



DEPARTMENTS

- ☑ DEPARTMENT ID
- □ DEPARTMENT NAME
- □ *MANAGER ID*
- □ *LOCATION ID*

EMP DETAILS VIEW

- ☑ EMPLOYEE ID
- □ JOB ID
- □ MANAGER ID
- □ DEPARTMENT ID
- □ LOCATION ID
- □ COUNTRY ID
- ☑ FIRST NAME
- ☑ LAST NAME
- □ SALARY
- □ COMMISSION PCT
- ☑ DEPARTMENT NAME
- □ JOB TITLE
- □ CITY
- □ STATE PROVINCE

Set Relationship

Relationship

A-> : departments.department_id

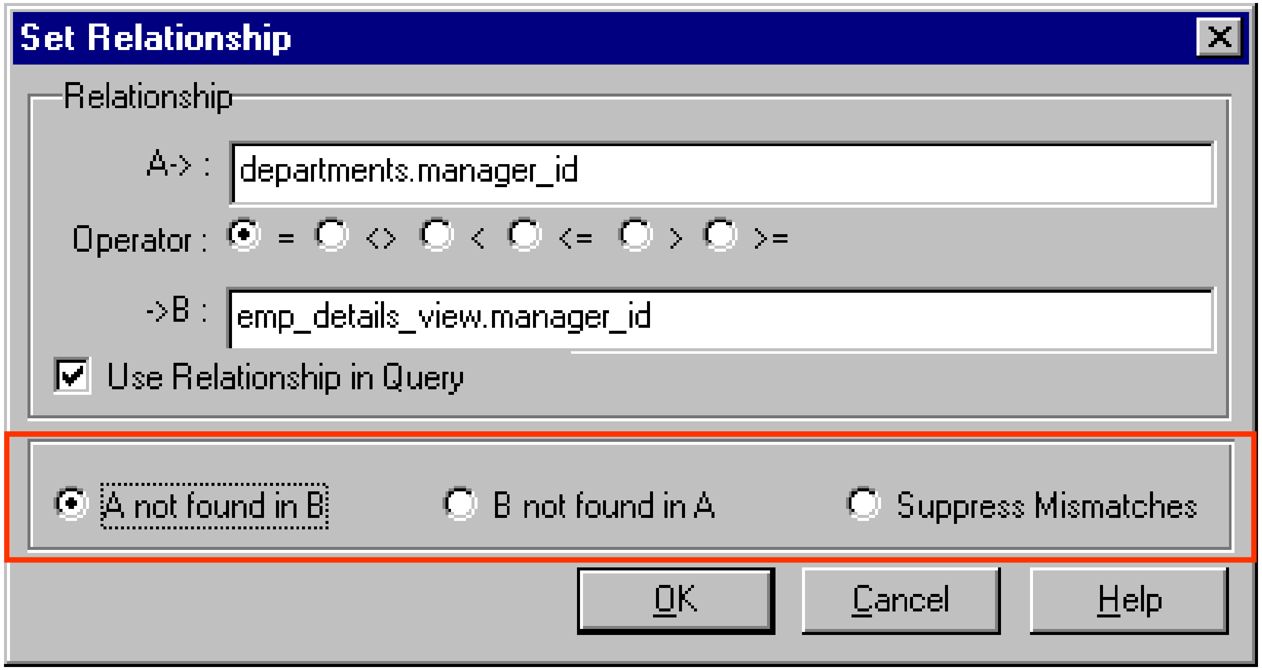
Operator : ● = ○ <> ○ < ○ <= ○ > ○ >=

->B : emp_details_view.department_id

☑ Use Relationship in Query

○ A not found in B ○ B not found in A ● Suppress Mismatches

Unmatched Rows



Set Relationship

Relationship

A-> : departments.manager_id

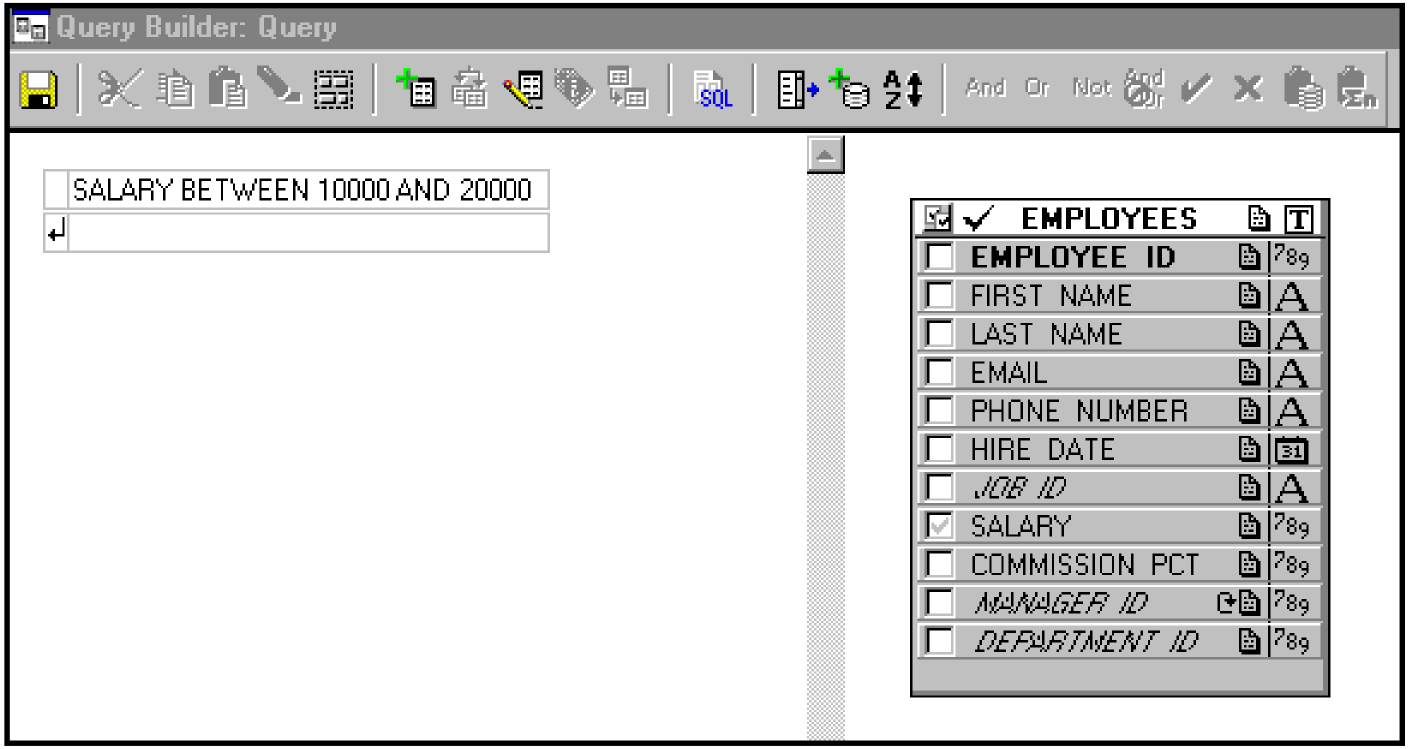
Operator : ● = ○ <> ○ < ○ <= ○ > ○ >=

->B : emp_details_view.manager_id

☑ Use Relationship in Query

● A not found in B ○ B not found in A ○ Suppress Mismatches

Conditions



SALARY BETWEEN 10000 AND 20000

✓ EMPLOYEES
□ EMPLOYEE_ID
□ FIRST_NAME
□ LAST_NAME
□ EMAIL
□ PHONE_NUMBER
□ HIRE_DATE
□ JOB_ID
☑ SALARY
□ COMMISSION_PCT
□ MANAGER_ID
□ DEPARTMENT_ID

Operators

Arithmetic

- Perform calculations on numeric and date columns
- Examples: +, -, x, /

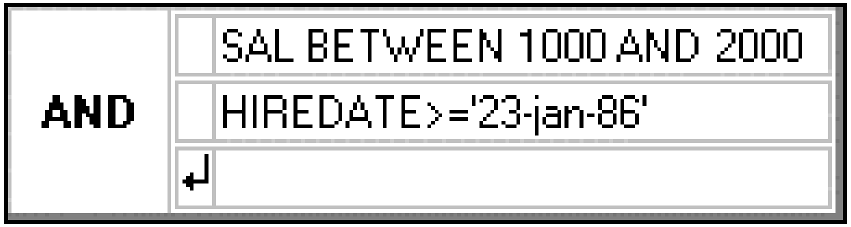
Logical

- Combine conditions
- Examples: AND, OR, NOT

Comparison

- Compare one expression with another
- Examples: =, <>, <, IN, IS NULL, BETWEEN ... AND

Multiple Conditions

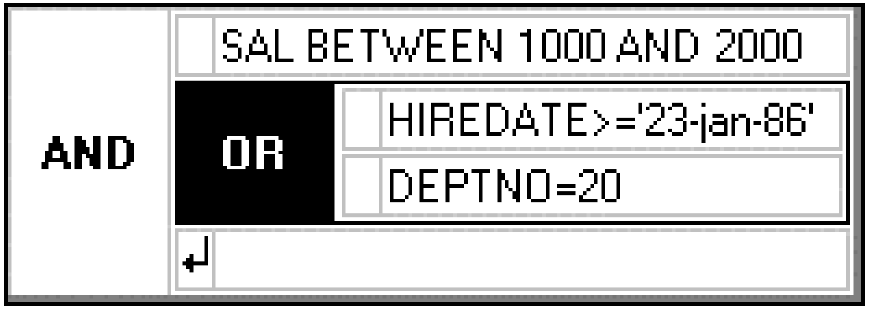


AND

SAL BETWEEN 1000 AND 2000

HIREDATE>='23-jan-86'

↵



AND

SAL BETWEEN 1000 AND 2000

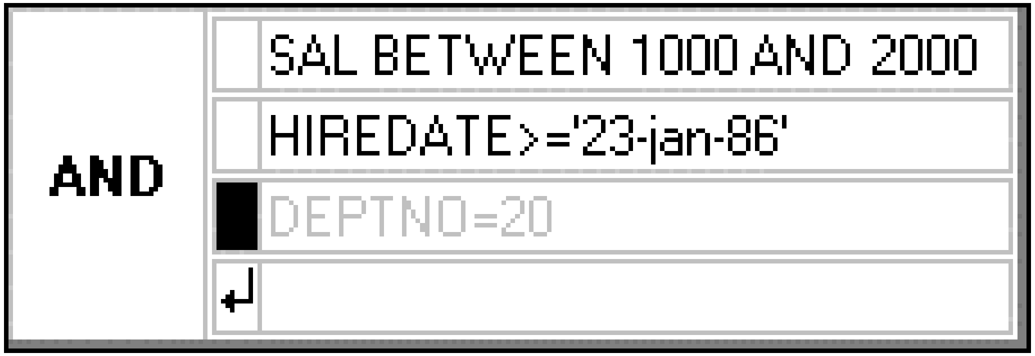
OR

HIREDATE>='23-jan-86'

DEPTNO=20

↵

Deactivating a Condition



AND

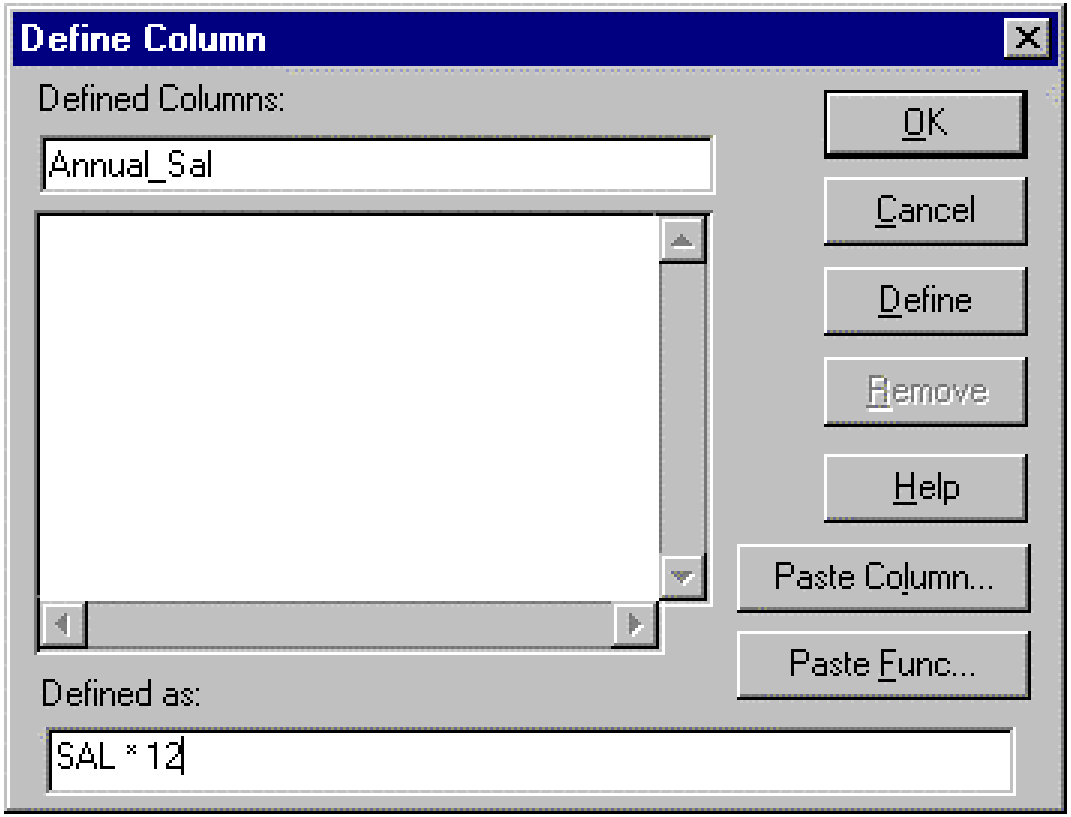
SAL BETWEEN 1000 AND 2000

HIREDATE>='23-jan-86'

DEPTNO=20

↵

Defining Columns Using an Expression

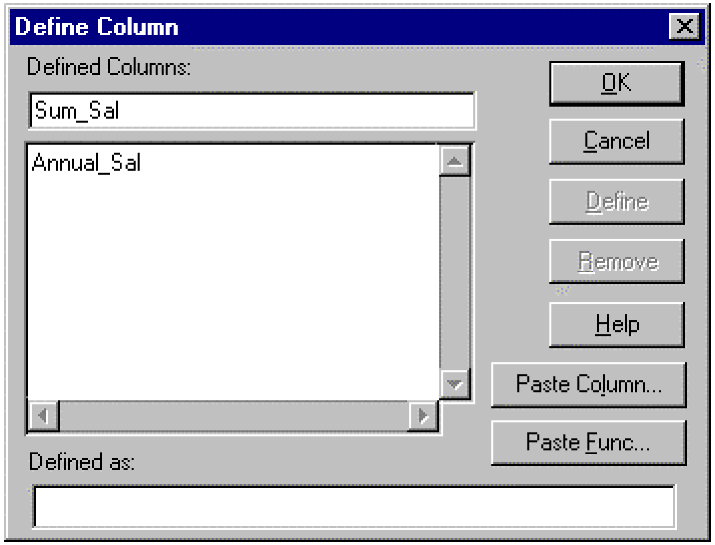


Define Column

Defined Columns: Annual_Sal

Defined as: SAL * 12

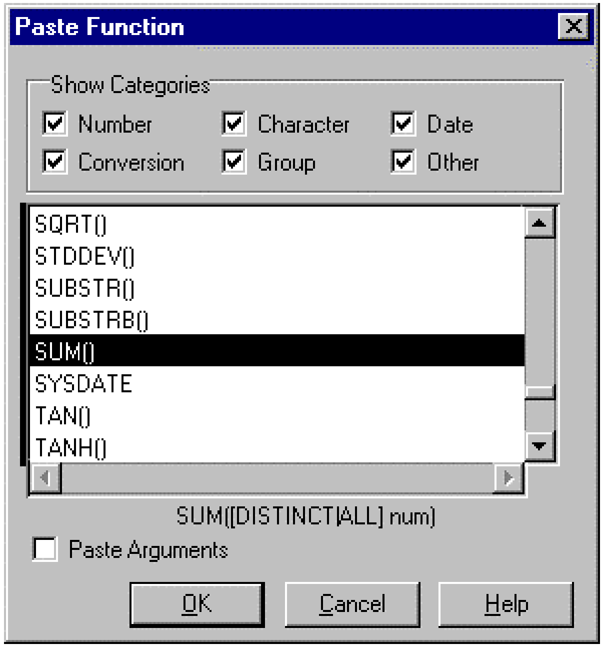
Defining Columns Using a Function



Defined Columns: Sum_Sal

Annual_Sal

Defined as:



Show Categories

☑ Number ☑ Character ☑ Date

☑ Conversion ☑ Group ☑ Other

SQRT()
STDDEV()
SUBSTR()
SUBSTRB()
SUM()
SYSDATE
TAN()
TANH()

SUM([DISTINCT|ALL] num)

□ Paste Arguments

D

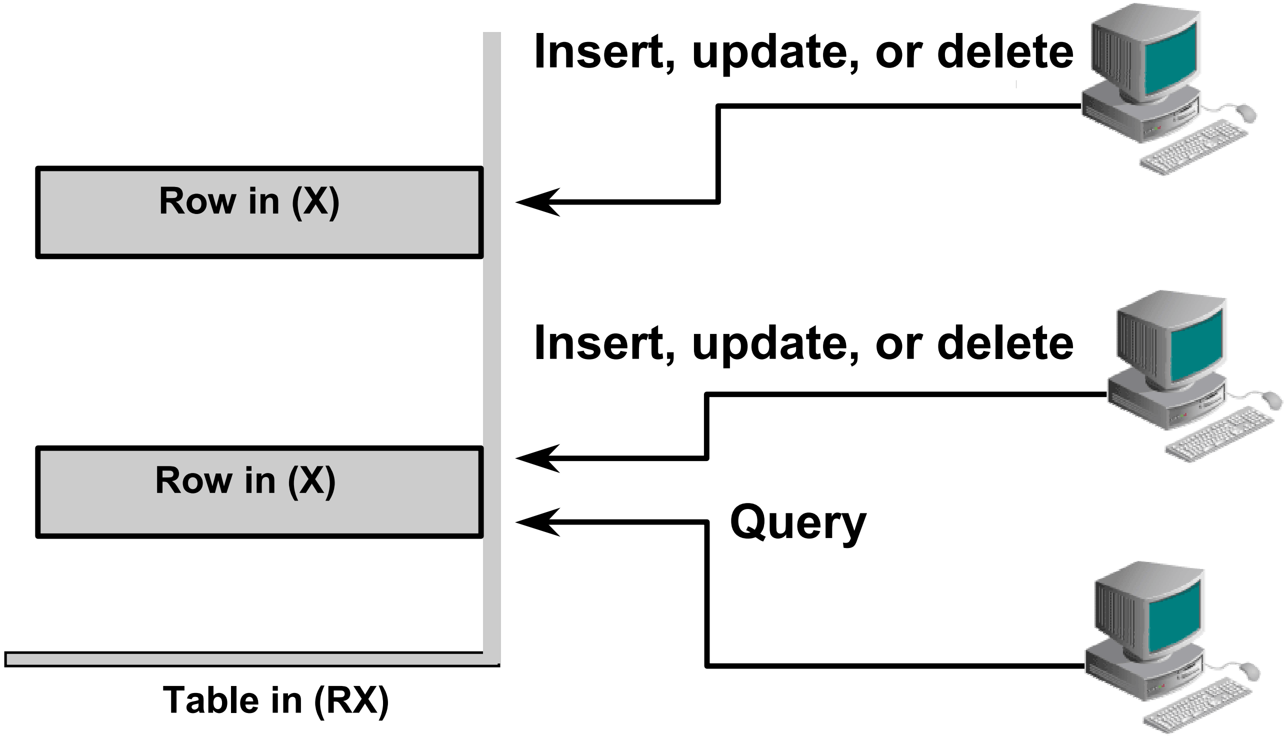
Locking in Forms

Objectives

After completing this lesson, you should be able to do the following:

- Identify the locking mechanisms in Forms
- Write triggers to invoke or intercept the locking process
- Plan trigger code to minimize overheads on locking

Locking



Insert, update, or delete

Row in (X)

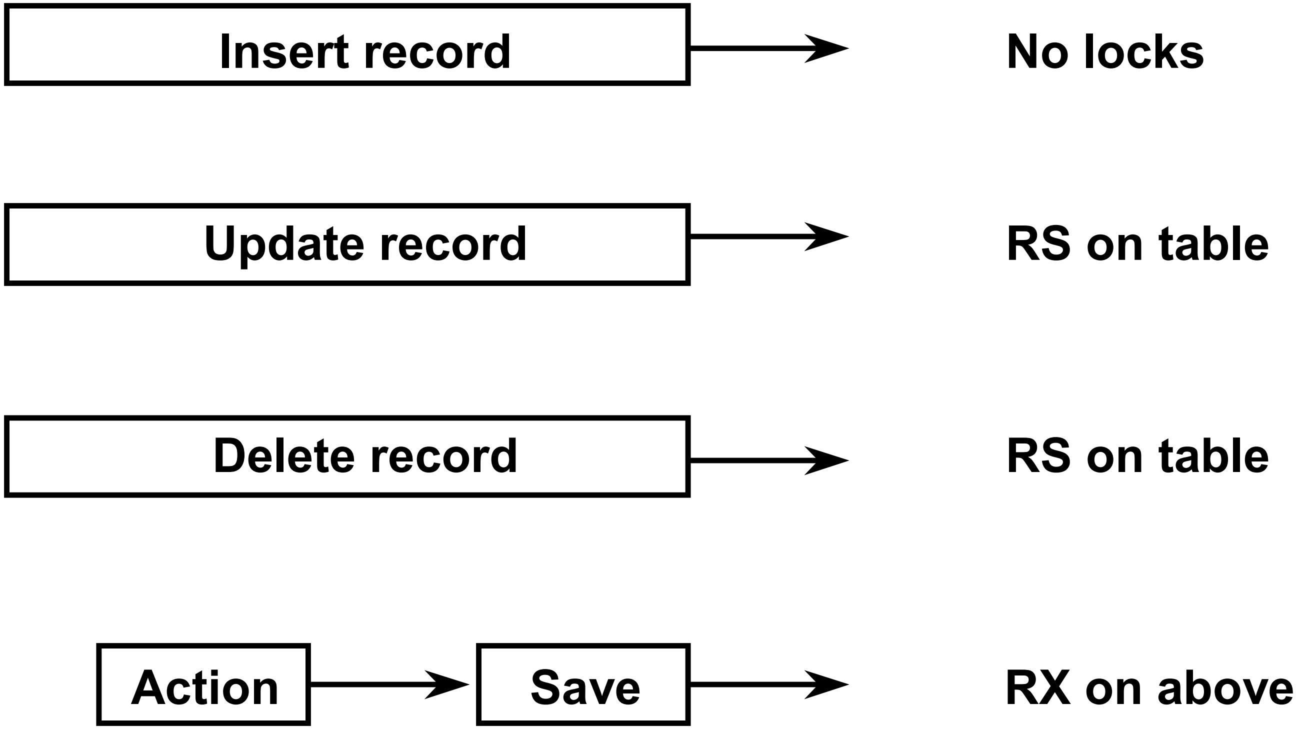
Insert, update, or delete

Row in (X)

Query

Table in (RX)

Default Locking in Forms



Insert record → No locks

Update record → RS on table

Delete record → RS on table

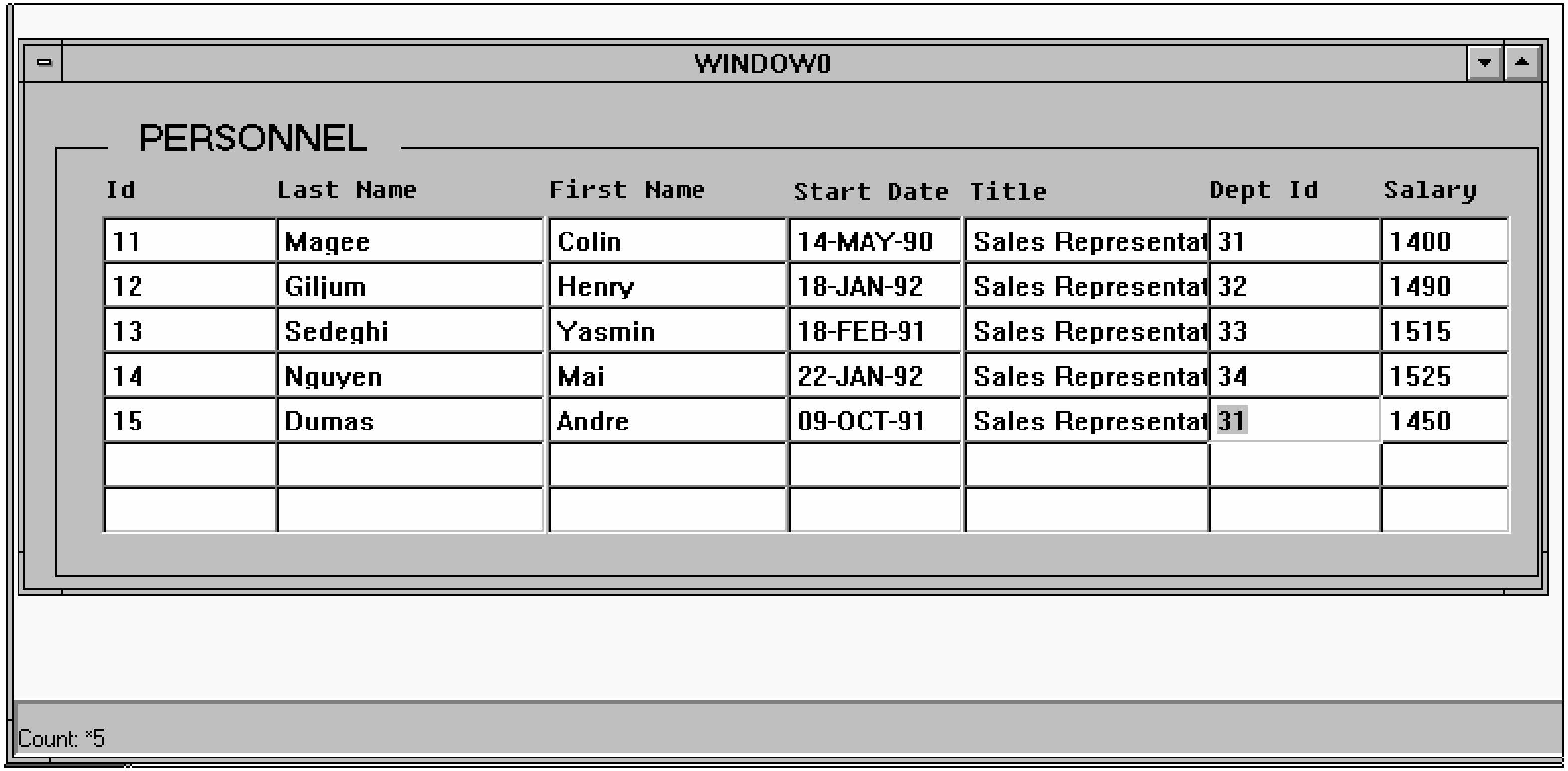
Action → Save → RX on above

ORACLE

Concurrent Updates and Deletes

- When users compete for the same record, normal locking protection applies.
- Forms tells the operator if another user has already locked the record.

User A: Step 1

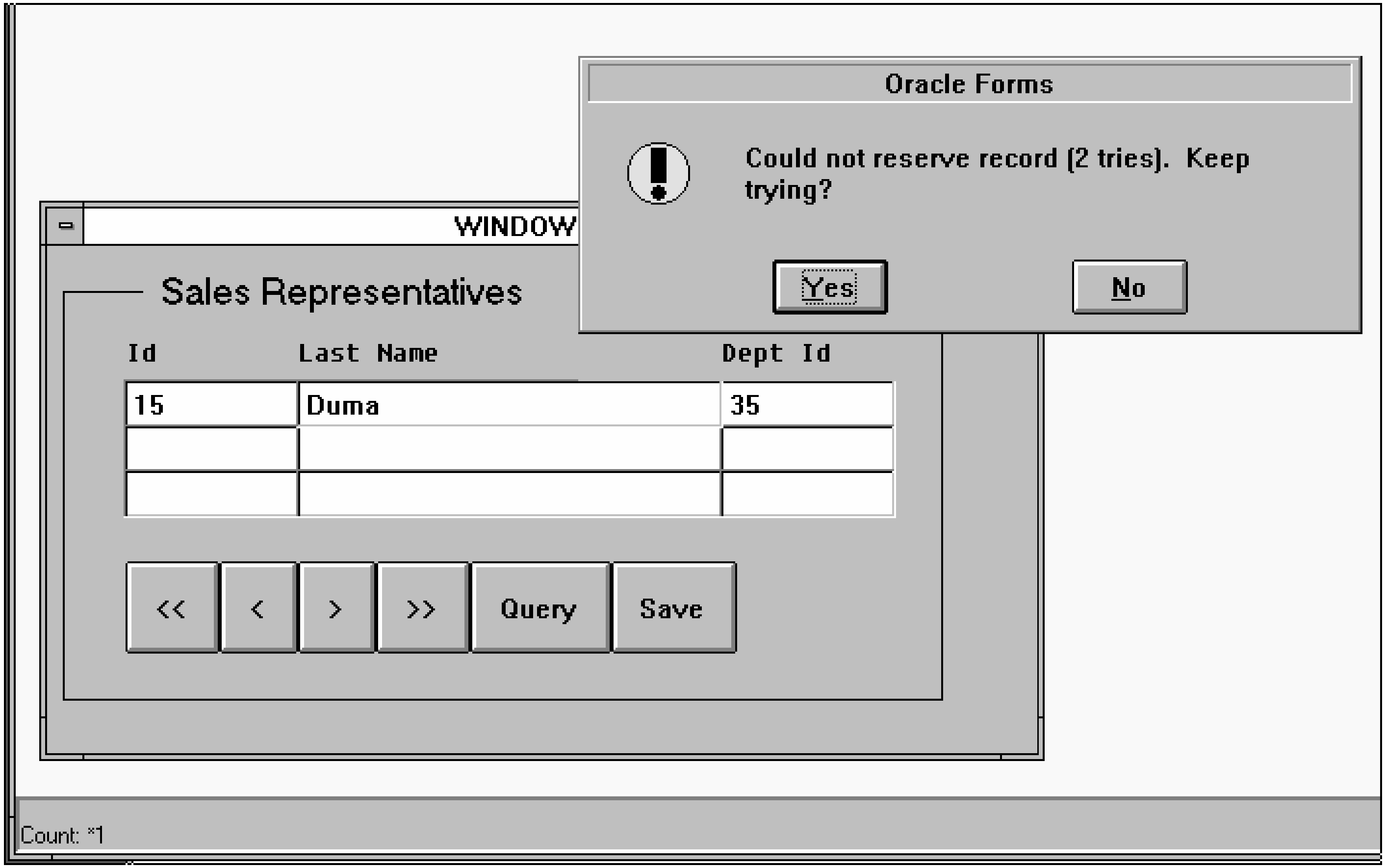


PERSONNEL

Id	Last Name	First Name	Start Date	Title	Dept Id	Salary
11	Magee	Colin	14-MAY-90	Sales Representat	31	1400
12	Giljum	Henry	18-JAN-92	Sales Representat	32	1490
13	Sedeghi	Yasmin	18-FEB-91	Sales Representat	33	1515
14	Nguyen	Mai	22-JAN-92	Sales Representat	34	1525
15	Dumas	Andre	09-OCT-91	Sales Representat	31	1450

Count: *5

User B: Step 2

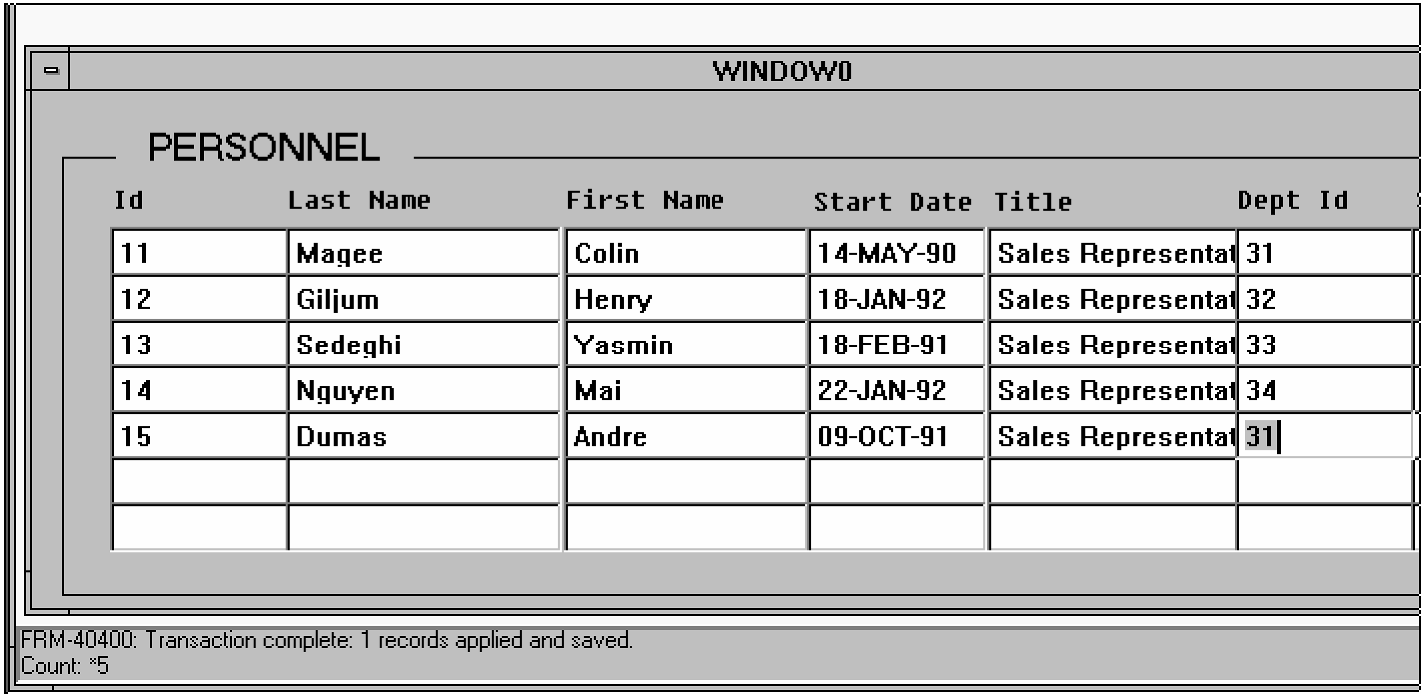


Oracle Forms

Could not reserve record (2 tries). Keep trying?

Id	Last Name	Dept Id
15	Duma	35

User A: Step 3

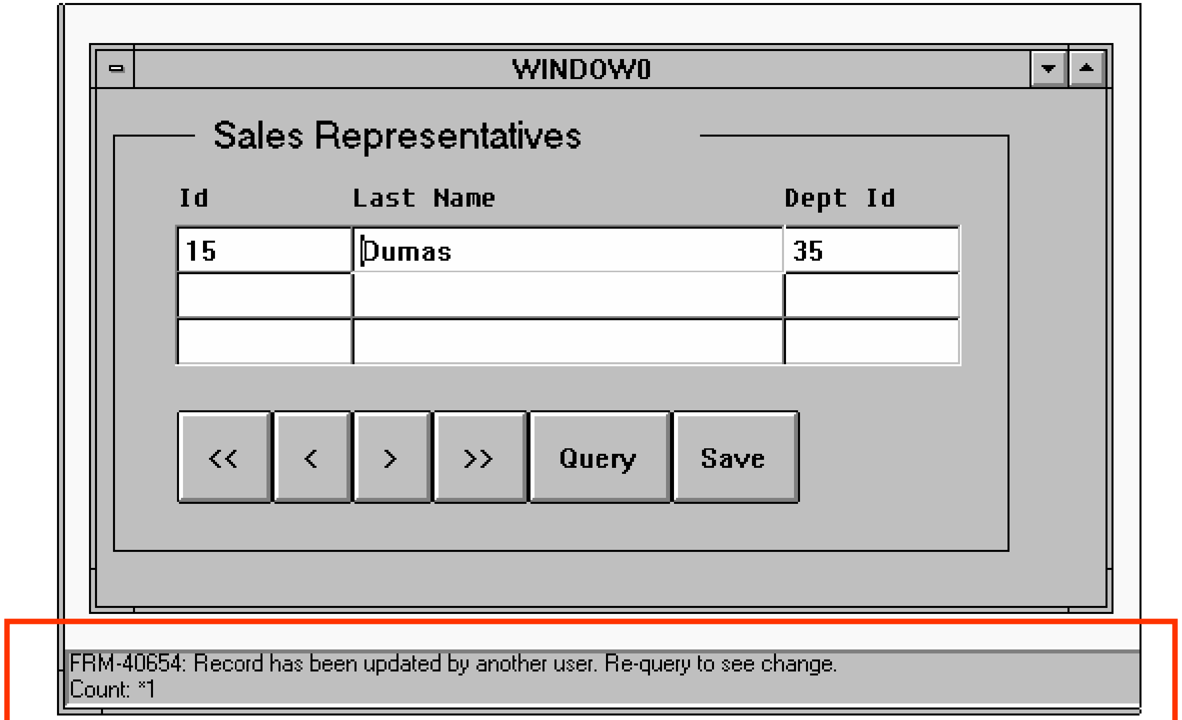


PERSONNEL

Id	Last Name	First Name	Start Date	Title	Dept Id
11	Magee	Colin	14-MAY-90	Sales Representat	31
12	Giljum	Henry	18-JAN-92	Sales Representat	32
13	Sedeghi	Yasmin	18-FEB-91	Sales Representat	33
14	Nguyen	Mai	22-JAN-92	Sales Representat	34
15	Dumas	Andre	09-OCT-91	Sales Representat	31

```
FRM-40400: Transaction complete: 1 records applied and saved.
Count: *5
```

User B: Step 4



Id	Last Name	Dept Id
15	Dumas	35

FRM-40654: Record has been updated by another user. Re-query to see change.
Count: *1

Locking in Triggers

Achieved by:

- SQL data manipulation language
- SQL explicit locking statements
- Built-in subprograms
- DML statements

Locking with Built-Ins

- ENTER_QUERY (FOR_UPDATE)
- EXECUTE_QUERY (FOR_UPDATE)

On-Lock Trigger

Example

```
IF USER = 'MANAGER' THEN
  LOCK_RECORD;
ELSE
  MESSAGE('You are not authorized to change
  records here');
  RAISE form_trigger_failure;
END IF;
```

Summary

- Default locking
  - Locks rows during update and delete
  - Informs user of concurrent update and delete
- Locking in triggers
  - Use SQL and certain built-ins
  - On-Lock trigger: `LOCK_RECORD` built-in available

E

Oracle Object Features

Objectives

After completing this lesson, you should be able to do the following:

- Describe the Oracle scalar datatypes
- Describe object types and objects
- Describe object tables, object columns, and object views
- Describe the INSTEAD-OF triggers
- Describe object REFs
- Identify the display of objects in the Object Navigator

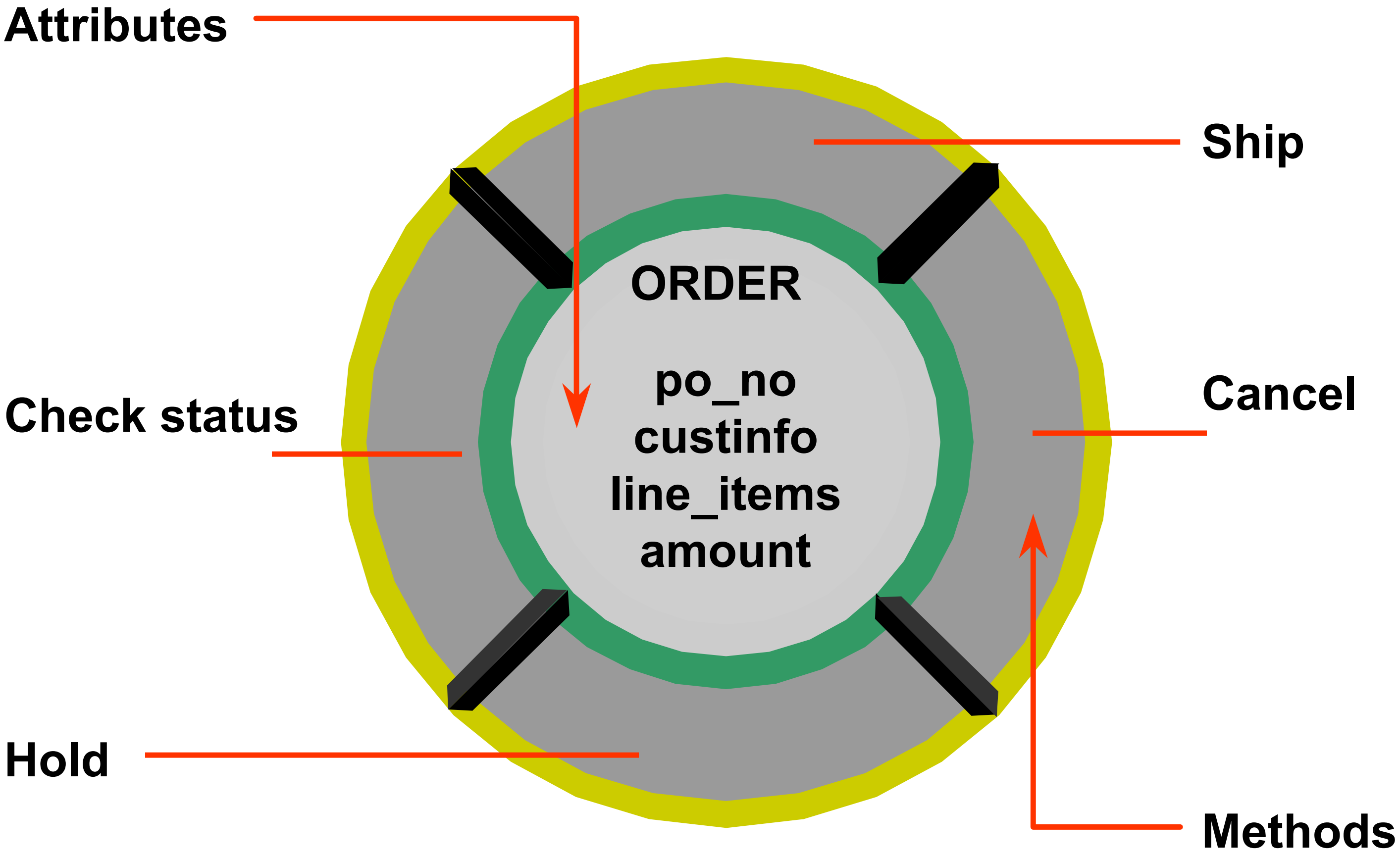
Oracle Scalar Datatypes

- Automatically converted:
  - FLOAT
  - NLS types

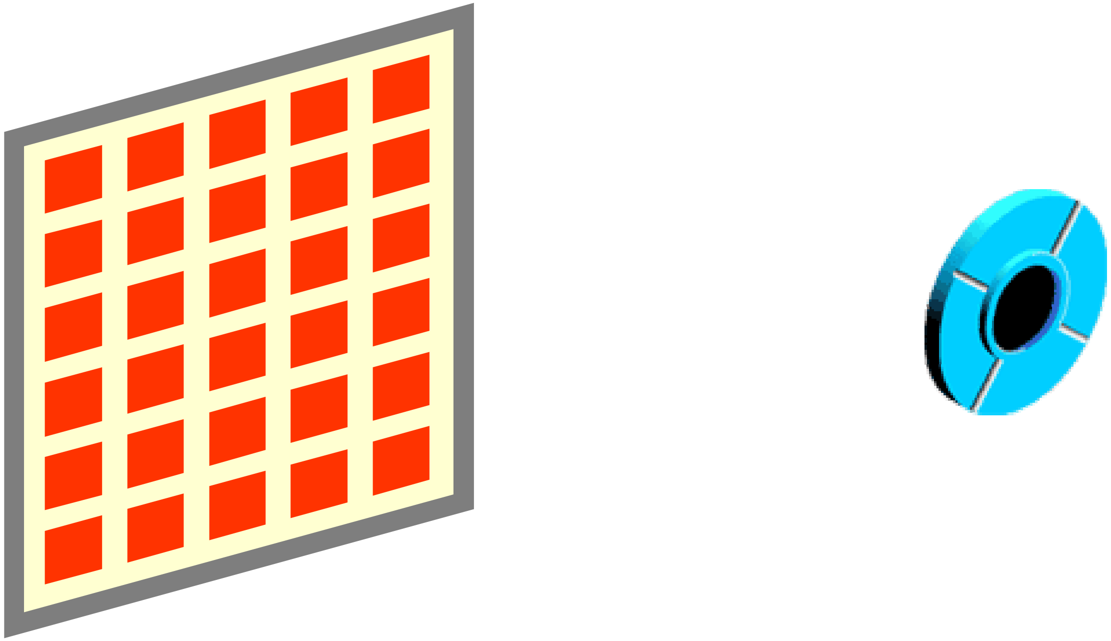
    NCHAR

    NVARCHAR2
- Unsupported:
  - Timestamp
  - Interval

Object Types

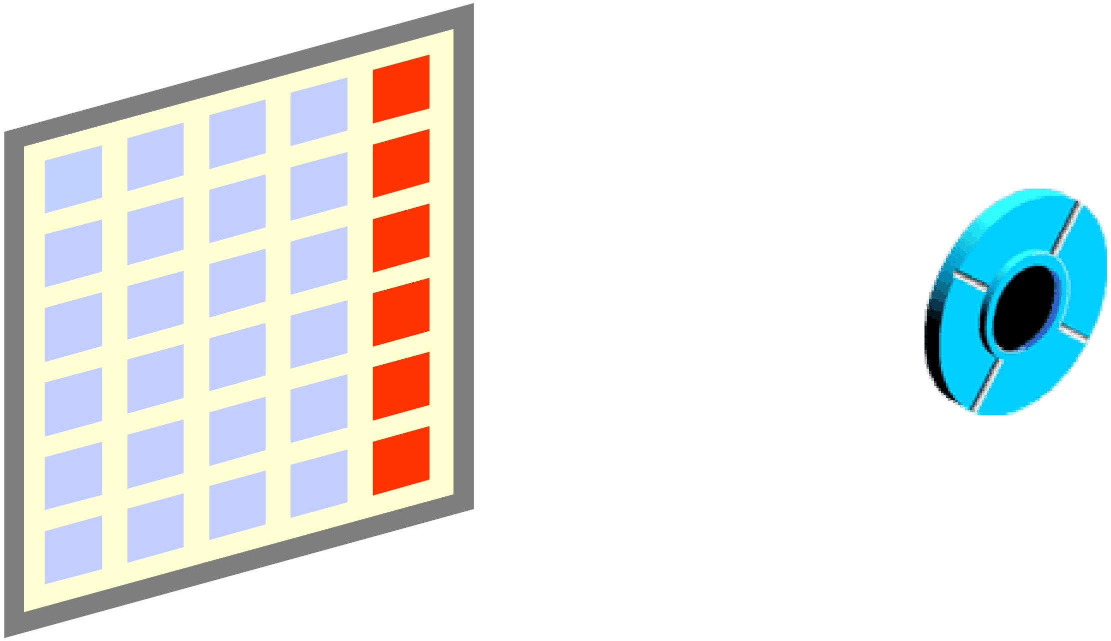


Object Tables



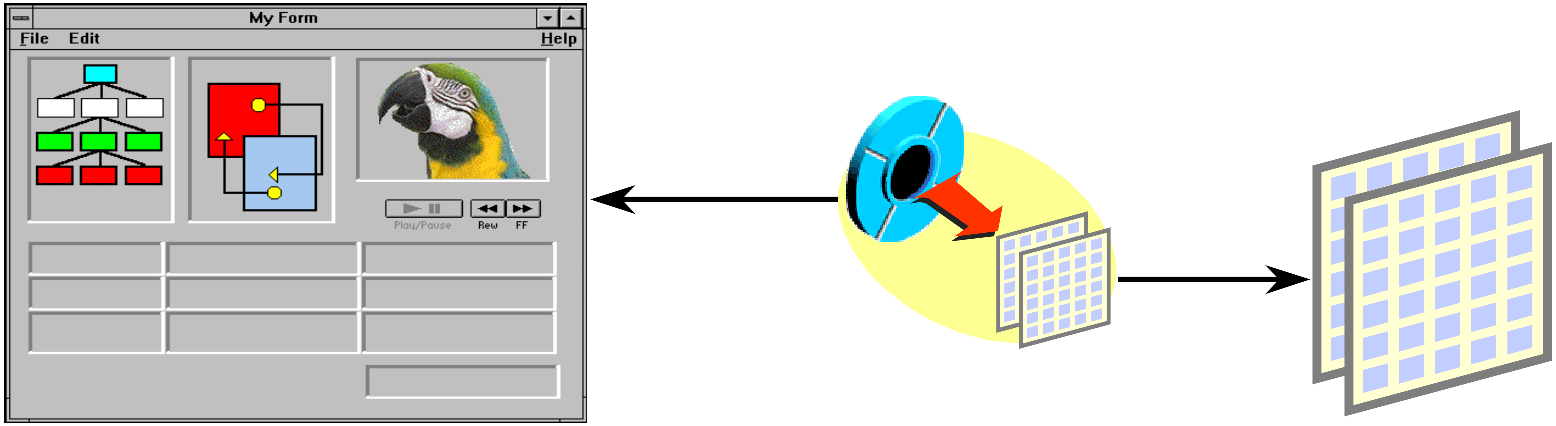
Object table based on object type

Object Columns



Object column based on object type

Object Views



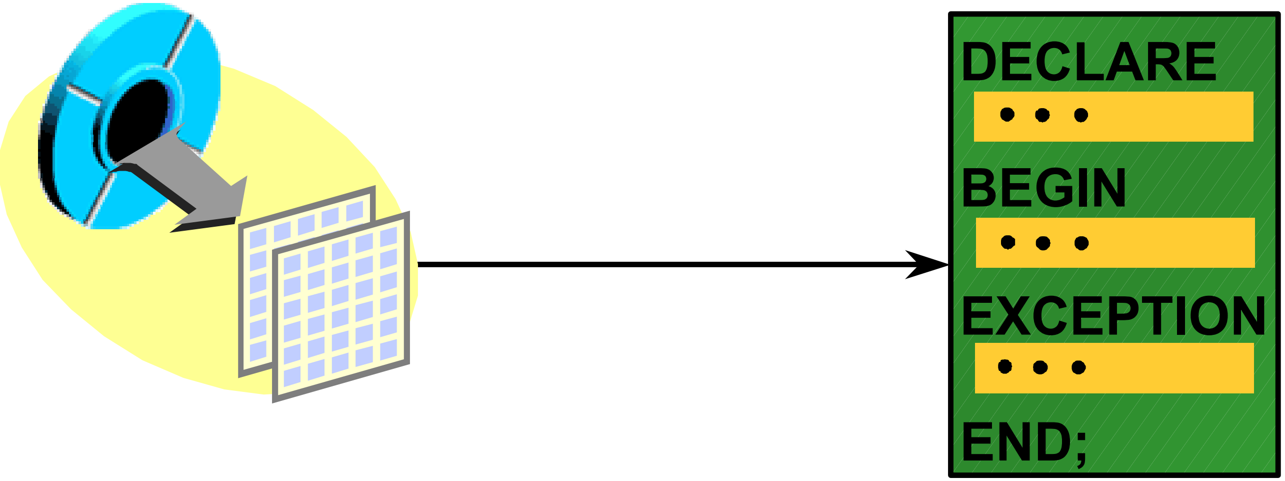
Object-oriented application

Object view

Relational table

Object views based on object types

INSTEAD-OF Triggers



DECLARE

• • •

BEGIN

• • •

EXCEPTION

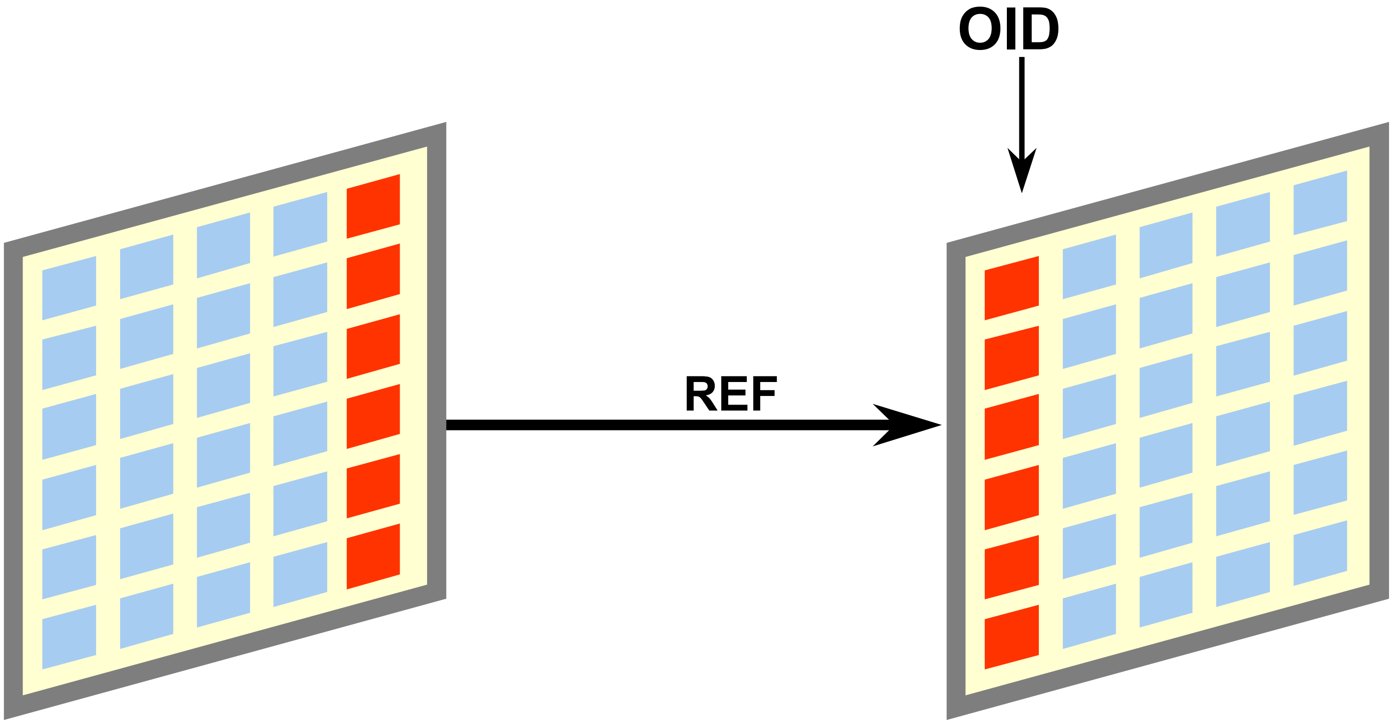
• • •

END;

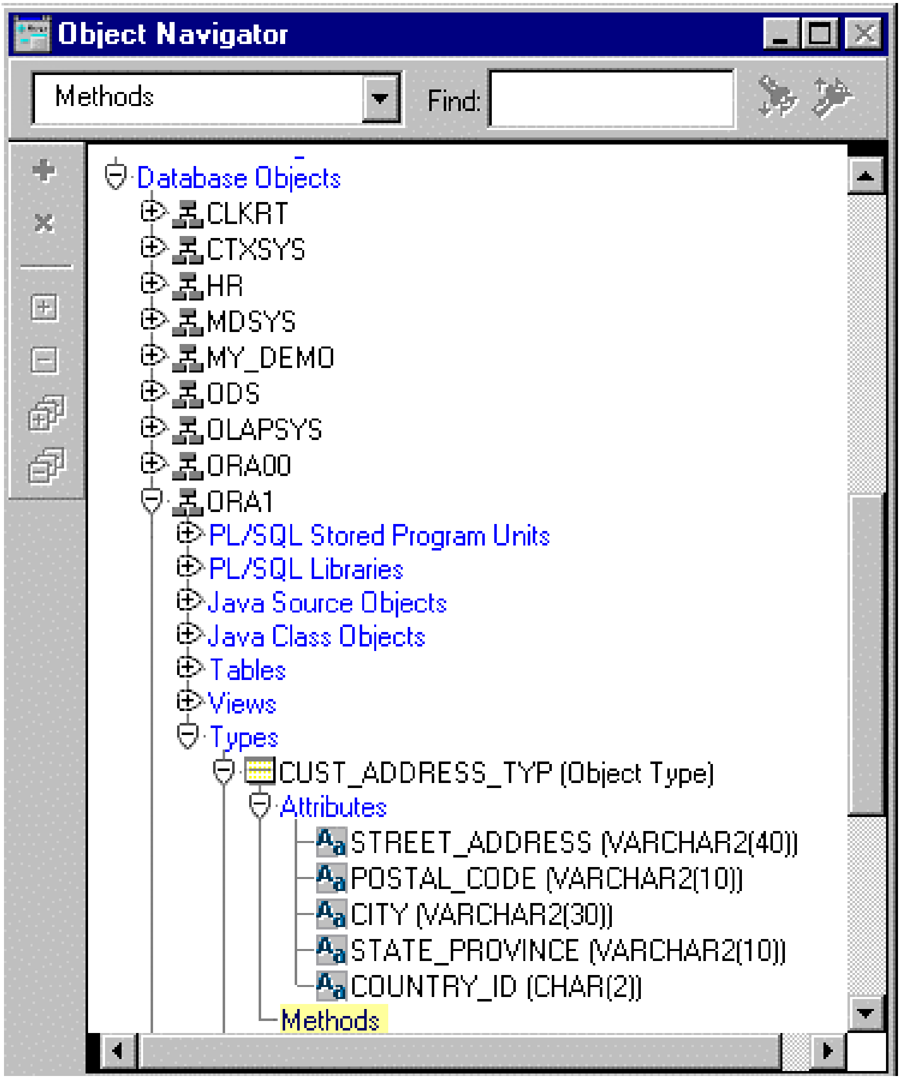
Nonupdatable
view

INSTEAD-OF
Trigger

References to Objects

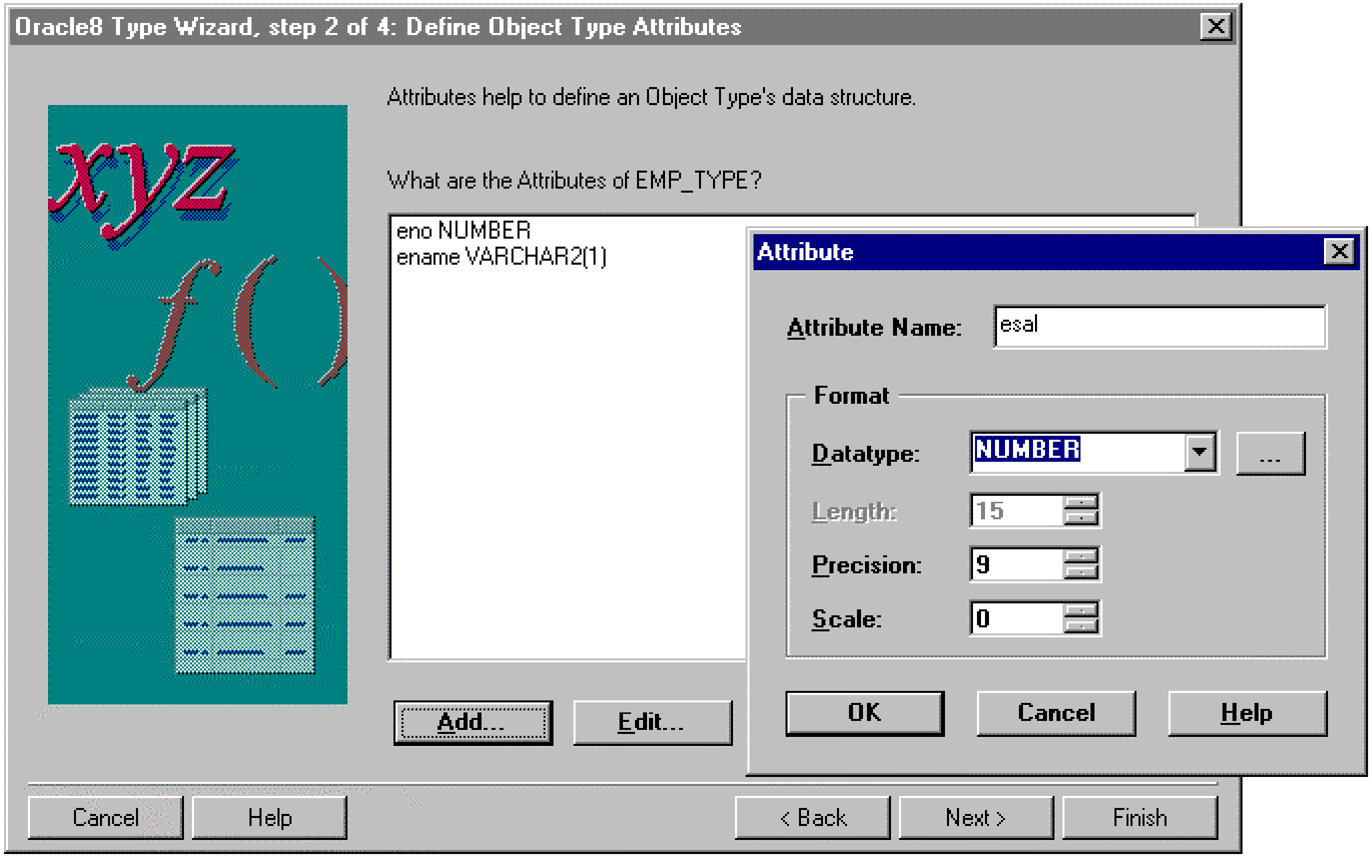


Object Types in Object Navigator



- Database Objects
  - CLKRT
  - CTXSYS
  - HR
  - MDSYS
  - MY_DEMO
  - ODS
  - OLAPSYS
  - ORA00
  - ORA1
    - PL/SQL Stored Program Units
    - PL/SQL Libraries
    - Java Source Objects
    - Java Class Objects
    - Tables
    - Views
    - Types
      - CUST_ADDRESS_TYP (Object Type)
        - Attributes
          - STREET_ADDRESS (VARCHAR2(40))
          - POSTAL_CODE (VARCHAR2(10))
          - CITY (VARCHAR2(30))
          - STATE_PROVINCE (VARCHAR2(10))
          - COUNTRY_ID (CHAR(2))
        - Methods

Object Type Wizard



Oracle8 Type Wizard, step 2 of 4: Define Object Type Attributes

Attributes help to define an Object Type's data structure.

What are the Attributes of EMP_TYPE?

```
eno NUMBER
ename VARCHAR2(1)
```

Attribute

Attribute Name: esal

Format

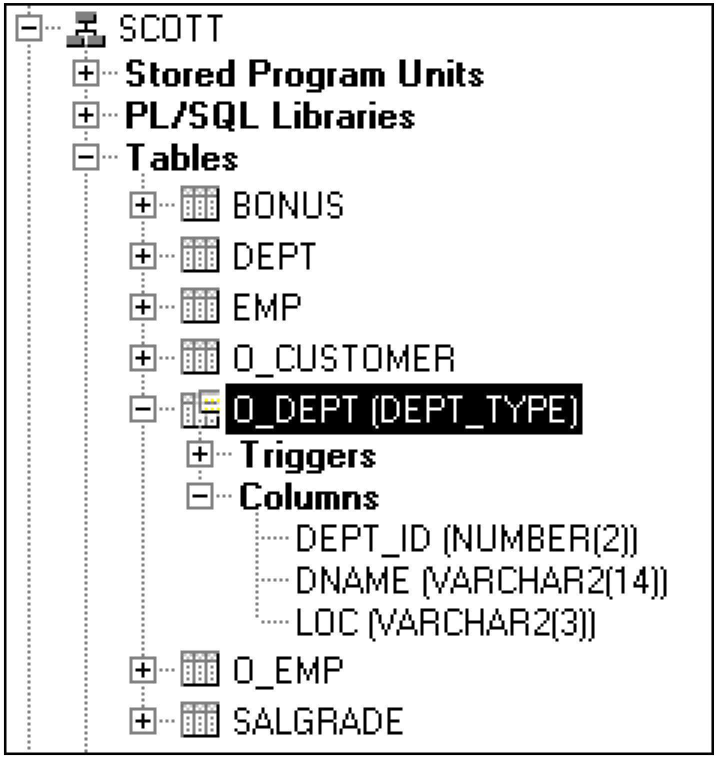
Datatype: NUMBER

Length: 15

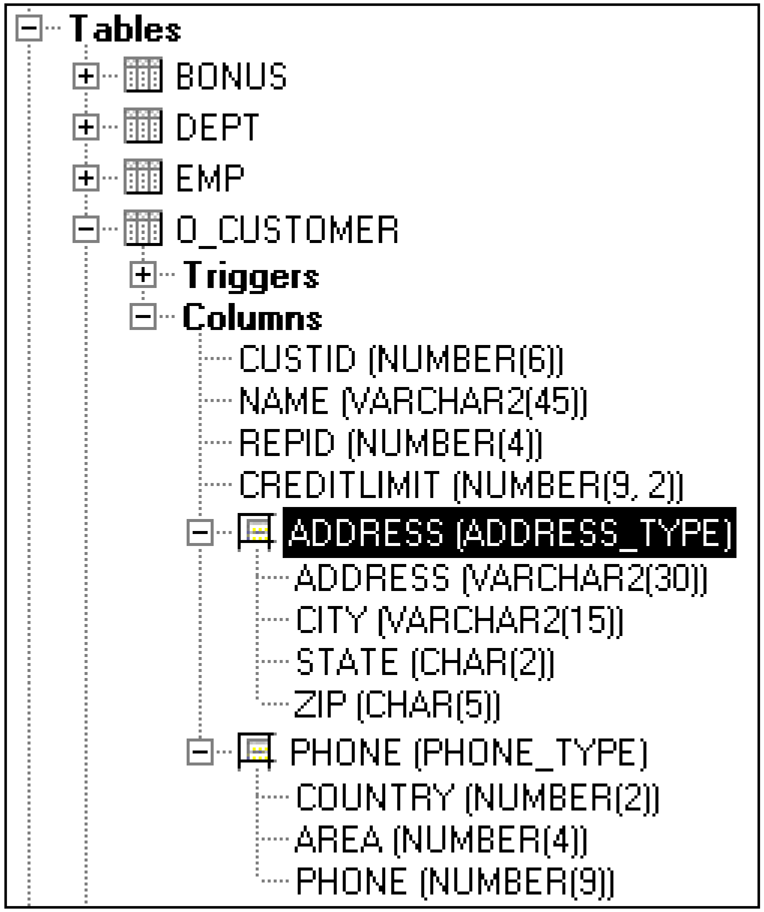
Precision: 9

Scale: 0

Object Tables and Columns in Object Navigator

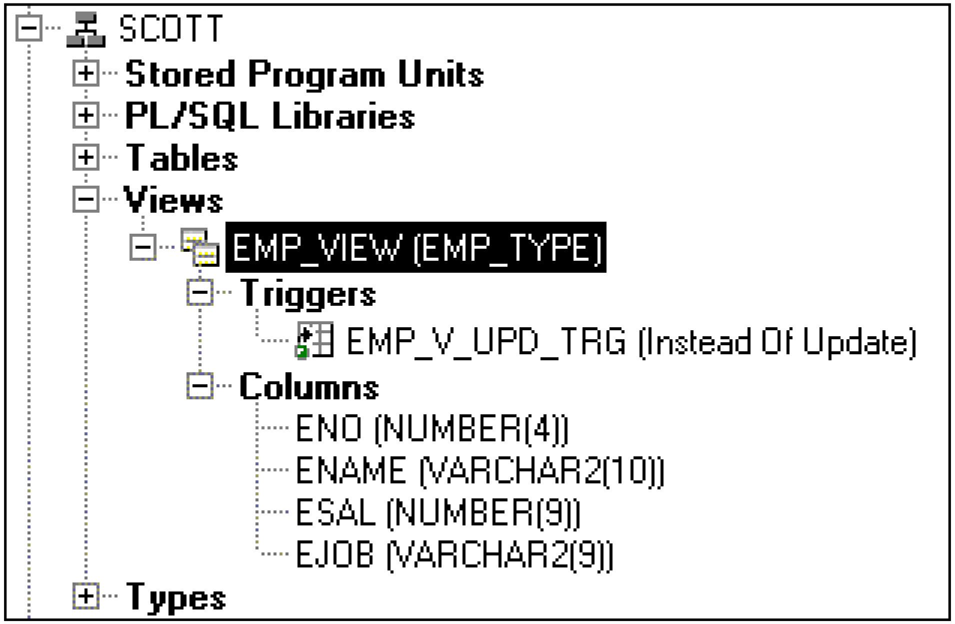


```
SCOTT
  Stored Program Units
  PL/SQL Libraries
  Tables
    BONUS
    DEPT
    EMP
    O_CUSTOMER
    O_DEPT (DEPT_TYPE)
      Triggers
      Columns
        DEPT_ID (NUMBER(2))
        DNAME (VARCHAR2(14))
        LOC (VARCHAR2(3))
    O_EMP
    SALGRADE
```



```
Tables
  BONUS
  DEPT
  EMP
  O_CUSTOMER
    Triggers
    Columns
      CUSTID (NUMBER(6))
      NAME (VARCHAR2(45))
      REPID (NUMBER(4))
      CREDITLIMIT (NUMBER(9, 2))
      ADDRESS (ADDRESS_TYPE)
        ADDRESS (VARCHAR2(30))
        CITY (VARCHAR2(15))
        STATE (CHAR(2))
        ZIP (CHAR(5))
      PHONE (PHONE_TYPE)
        COUNTRY (NUMBER(2))
        AREA (NUMBER(4))
        PHONE (NUMBER(9))
```

Object Views in Object Navigator



```
SCOTT
  Stored Program Units
  PL/SQL Libraries
  Tables
  Views
    EMP_VIEW (EMP_TYPE)
      Triggers
        EMP_V_UPD_TRG (Instead Of Update)
      Columns
        ENO (NUMBER(4))
        ENAME (VARCHAR2(10))
        ESAL (NUMBER(9))
        EJOB (VARCHAR2(9))
  Types
```

INSTEAD-OF Trigger Dialog Box

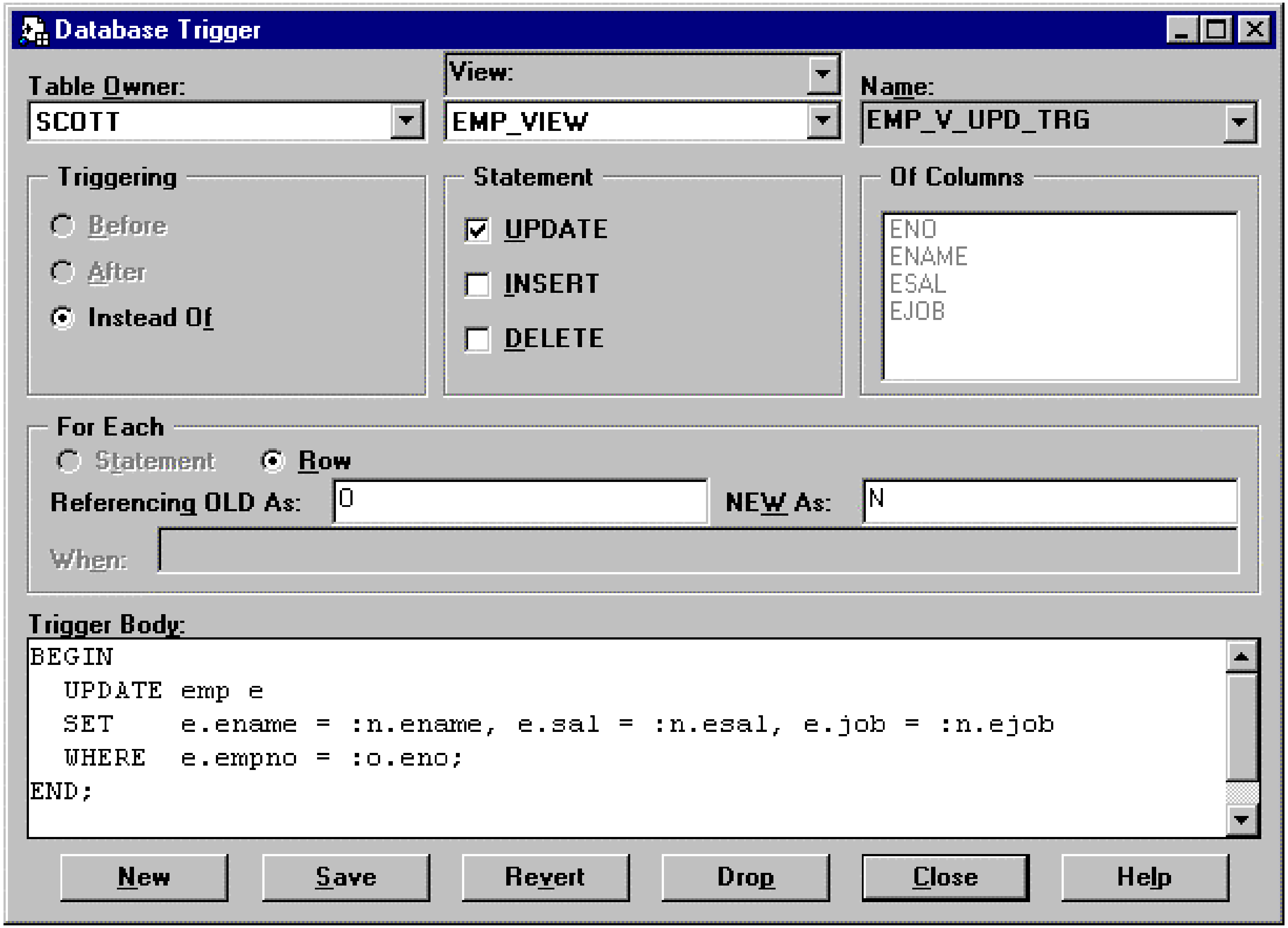


Table Owner: SCOTT

View: EMP_VIEW

Name: EMP_V_UPD_TRG

Triggering
○ Before
○ After
● Instead Of

Statement
☑ UPDATE
□ INSERT
□ DELETE

Of Columns
ENO
ENAME
ESAL
EJOB

For Each
○ Statement ● Row

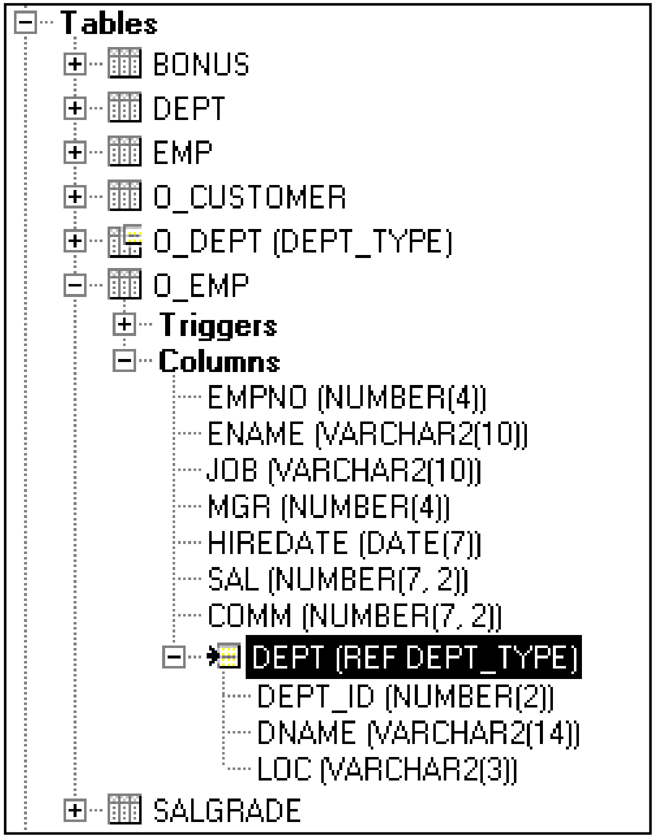
Referencing OLD As: O NEW As: N

When:

Trigger Body:

```
BEGIN
  UPDATE emp e
  SET    e.ename = :n.ename, e.sal = :n.esal, e.job = :n.ejob
  WHERE  e.empno = :o.eno;
END;
```

Object REFs in Object Navigator



```
Tables
    BONUS
    DEPT
    EMP
    O_CUSTOMER
    O_DEPT (DEPT_TYPE)
    O_EMP
        Triggers
        Columns
            EMPNO (NUMBER(4))
            ENAME (VARCHAR2(10))
            JOB (VARCHAR2(10))
            MGR (NUMBER(4))
            HIREDATE (DATE(7))
            SAL (NUMBER(7, 2))
            COMM (NUMBER(7, 2))
            DEPT (REF DEPT_TYPE)
                DEPT_ID (NUMBER(2))
                DNAME (VARCHAR2(14))
                LOC (VARCHAR2(3))
    SALGRADE
```

Summary

- Oracle8 introduced three scalar datatypes.
- Objects and object types allow representation of complex data.
- Three kinds of objects are object tables, object columns, and object views.

Summary

- INSTEAD-OF triggers allow DML on object views.
- Object REFs store the object identifier of certain types of objects.
- The Object Navigator can display certain types of objects.

F

Using the Layout Editor

ORACLE

Objectives

After completing this lesson, you should be able to do the following:

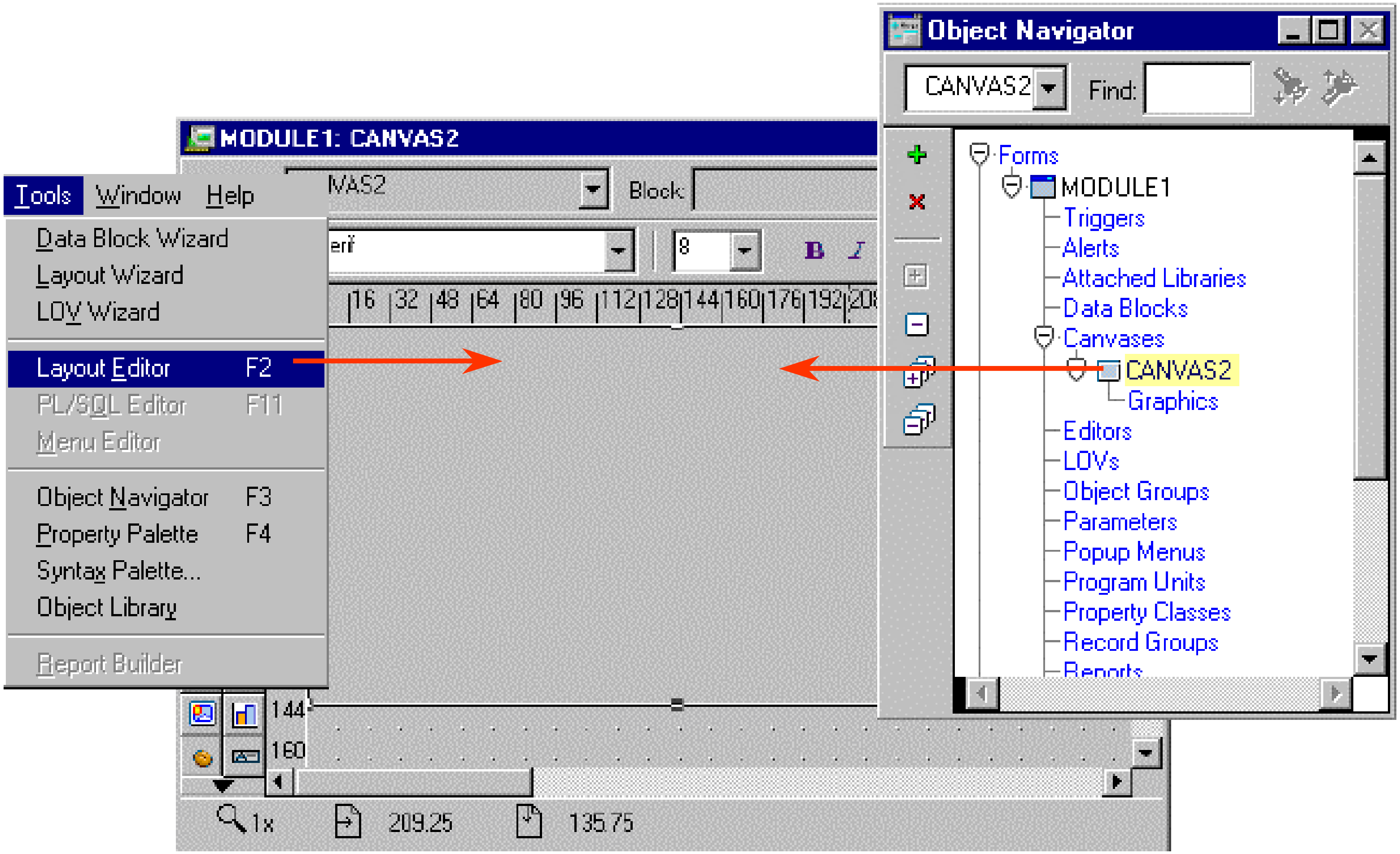
- Control the position and size of objects in a layout
- Add lines and geometric shapes
- Define the colors and fonts used for text
- Color the body and boundaries of objects
- Import images onto the layout

Using the Layout Editor

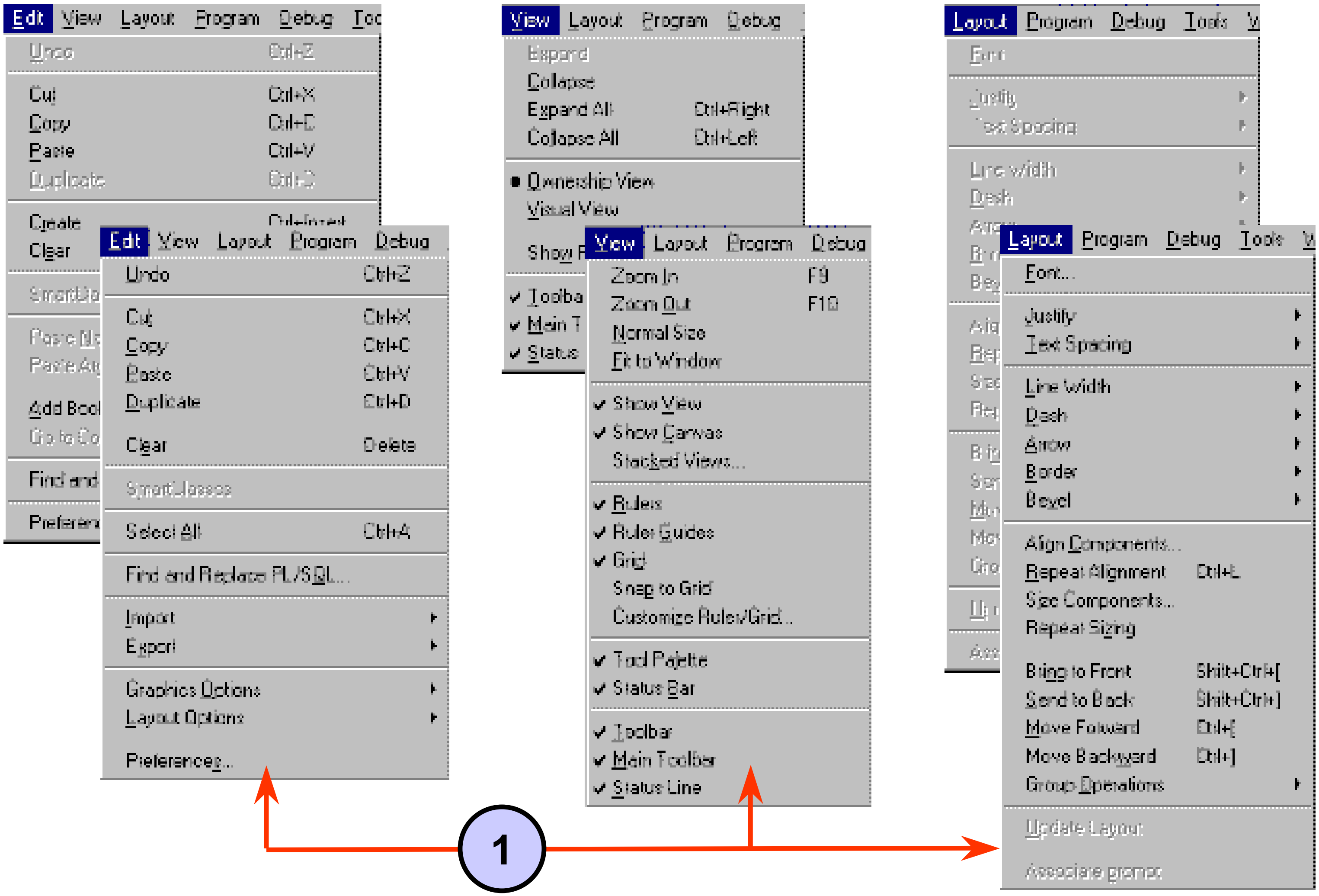
Common features:

- Moving and resizing objects and text
- Defining colors and fonts
- Importing and manipulating images and drawings
- Creating geometric lines and shapes
- Layout surface: Forms canvas view

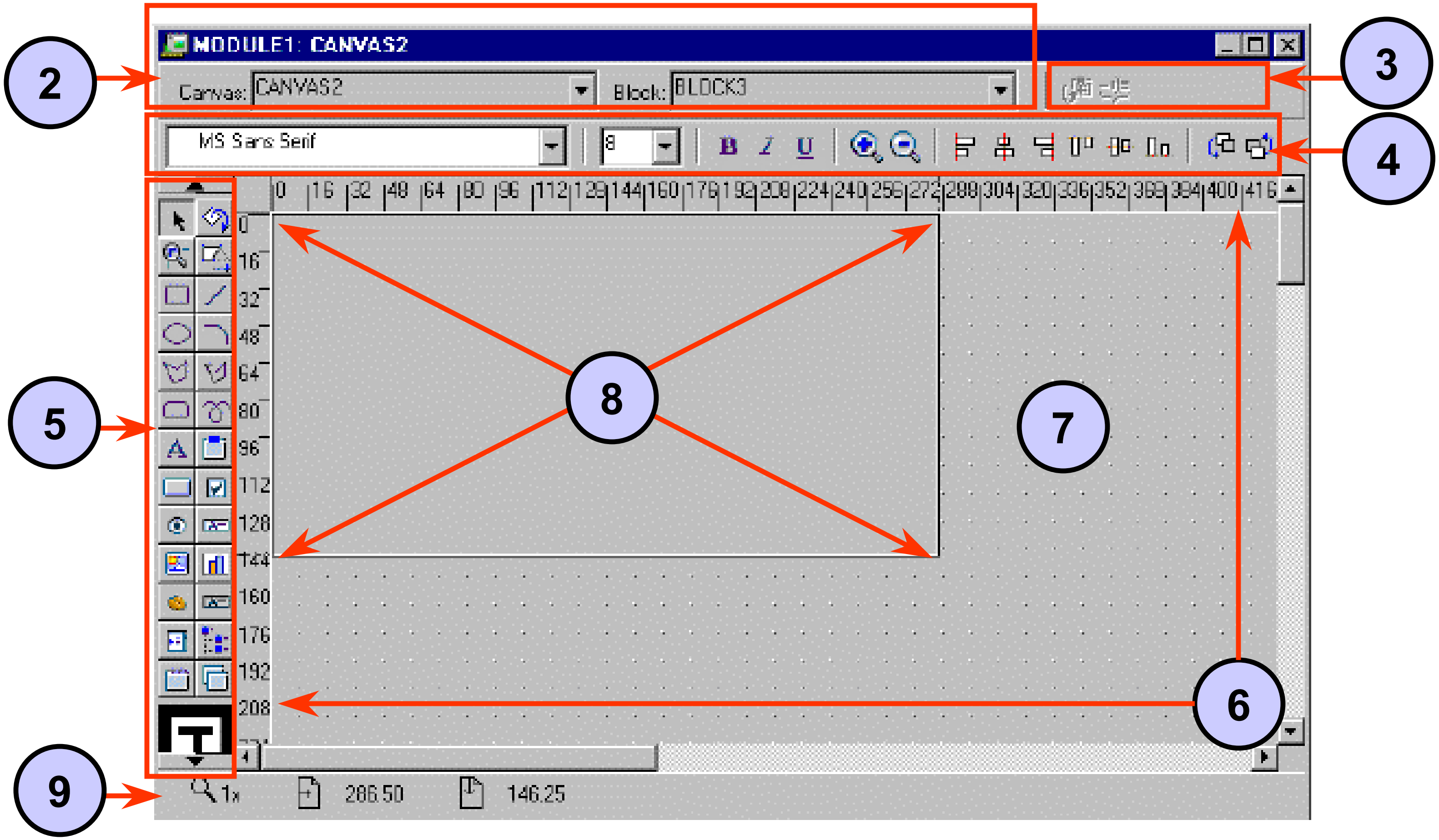
Invoking the Layout Editor



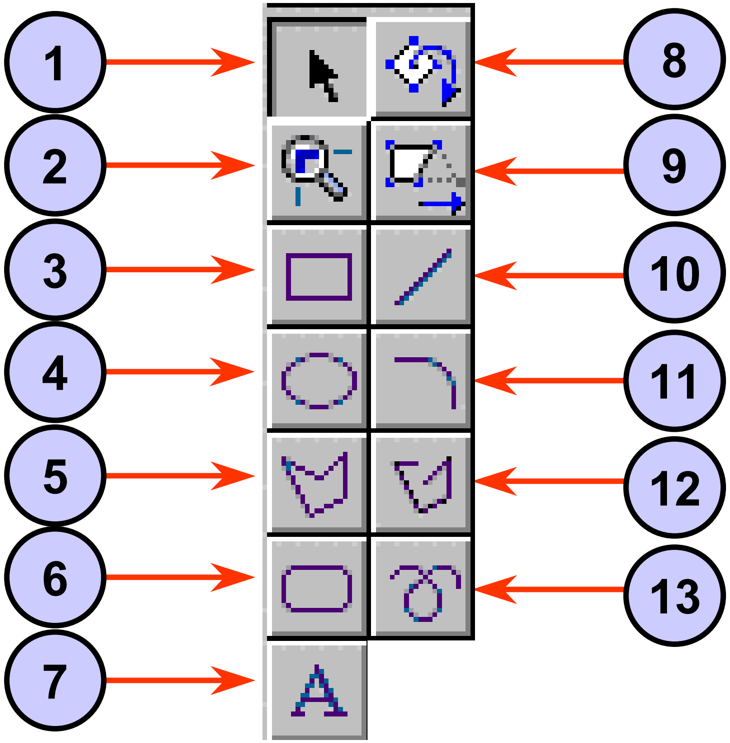
Layout Editor: Components



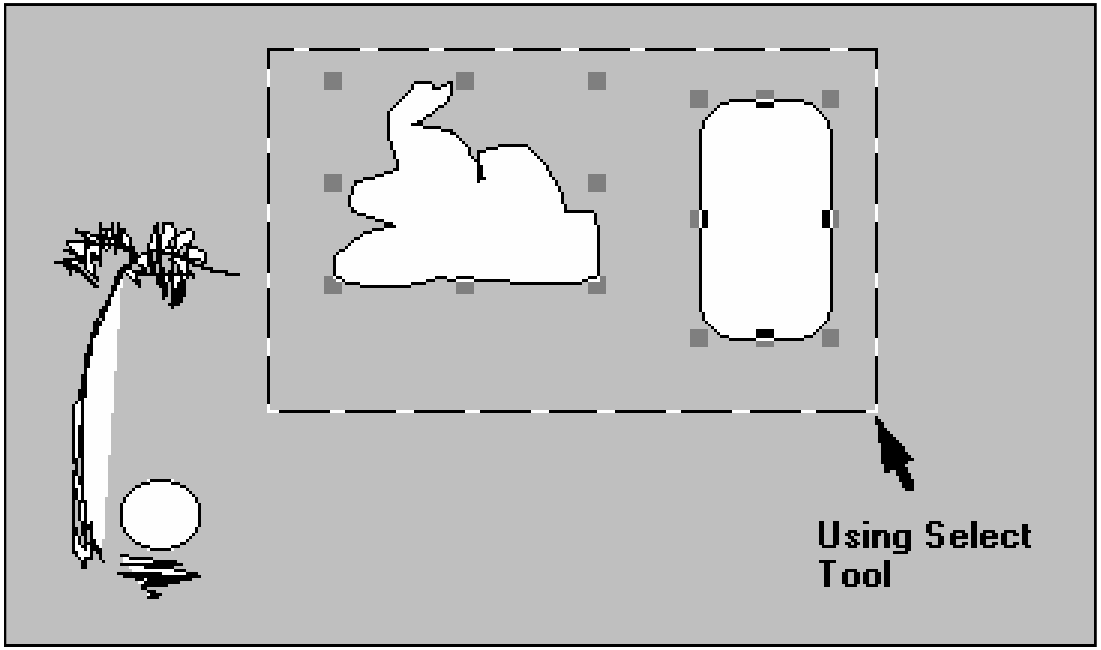
Layout Editor: Components



Tool Palette

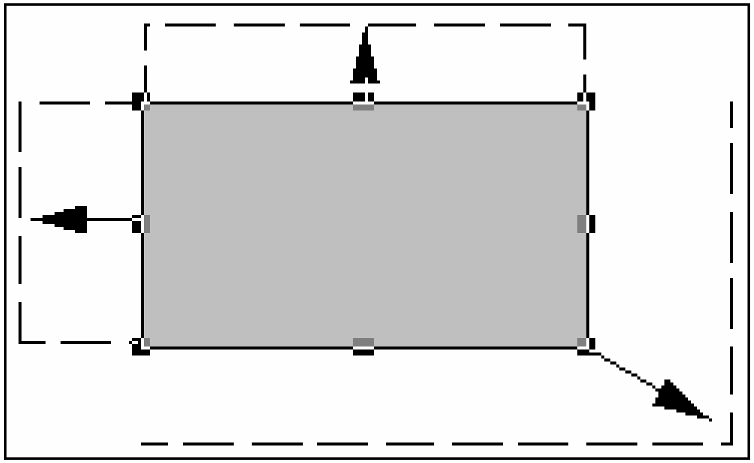


Selecting Objects



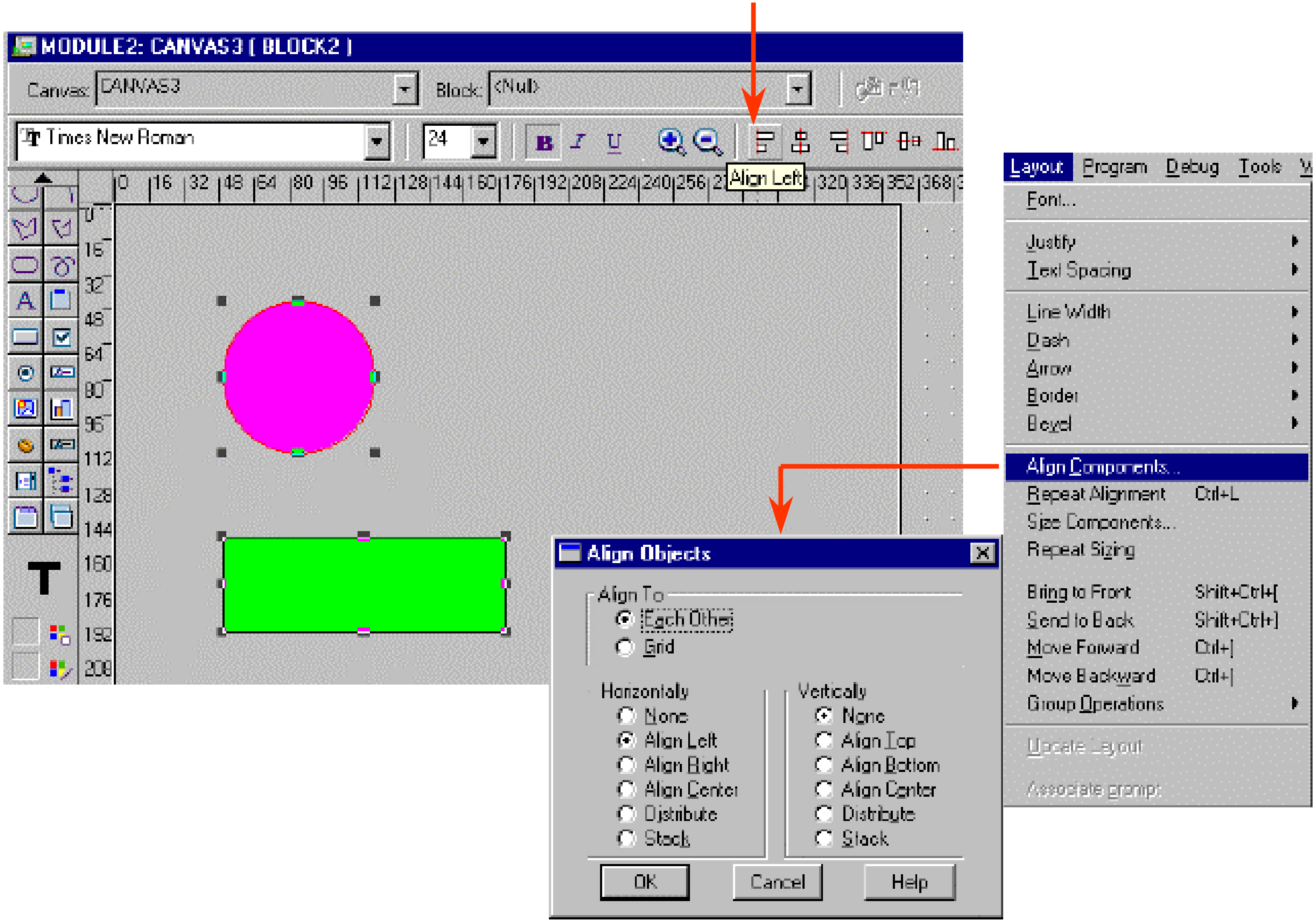
Manipulating Objects

Expand/contract
in one direction



Expand/contract
diagonally

Moving, Aligning, and Overlapping



Align Objects

Align To
- ● Each Other
- ○ Grid

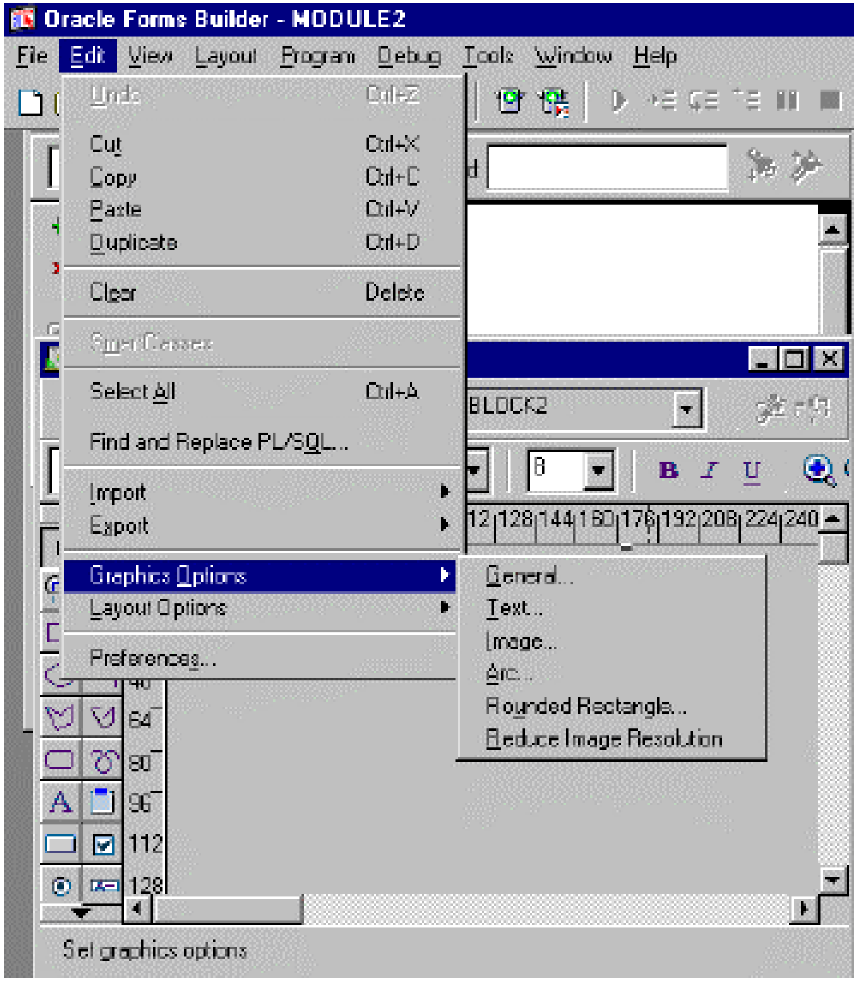
Horizontally
- ○ None
- ● Align Left
- ○ Align Right
- ○ Align Center
- ○ Distribute
- ○ Stack

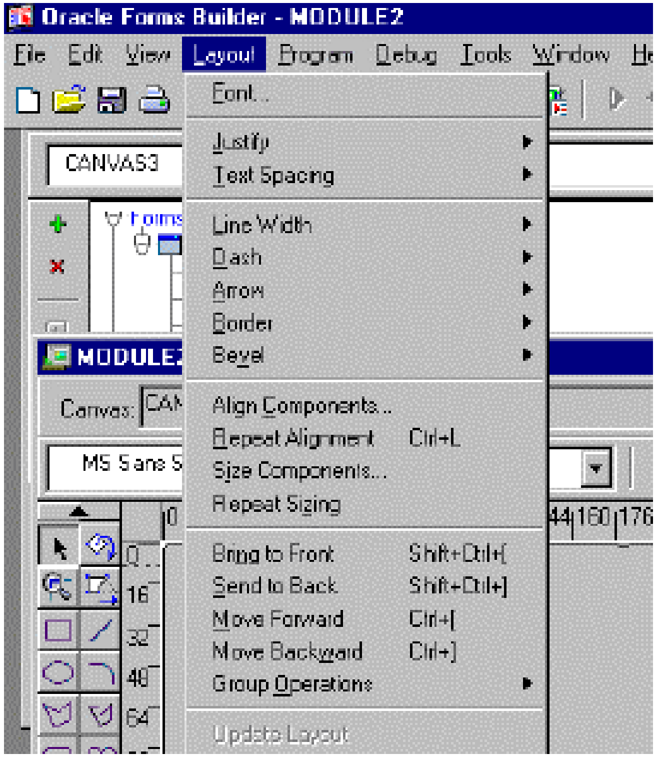
Vertically
- ● None
- ○ Align Top
- ○ Align Bottom
- ○ Align Center
- ○ Distribute
- ○ Stack

Groups in the Layout

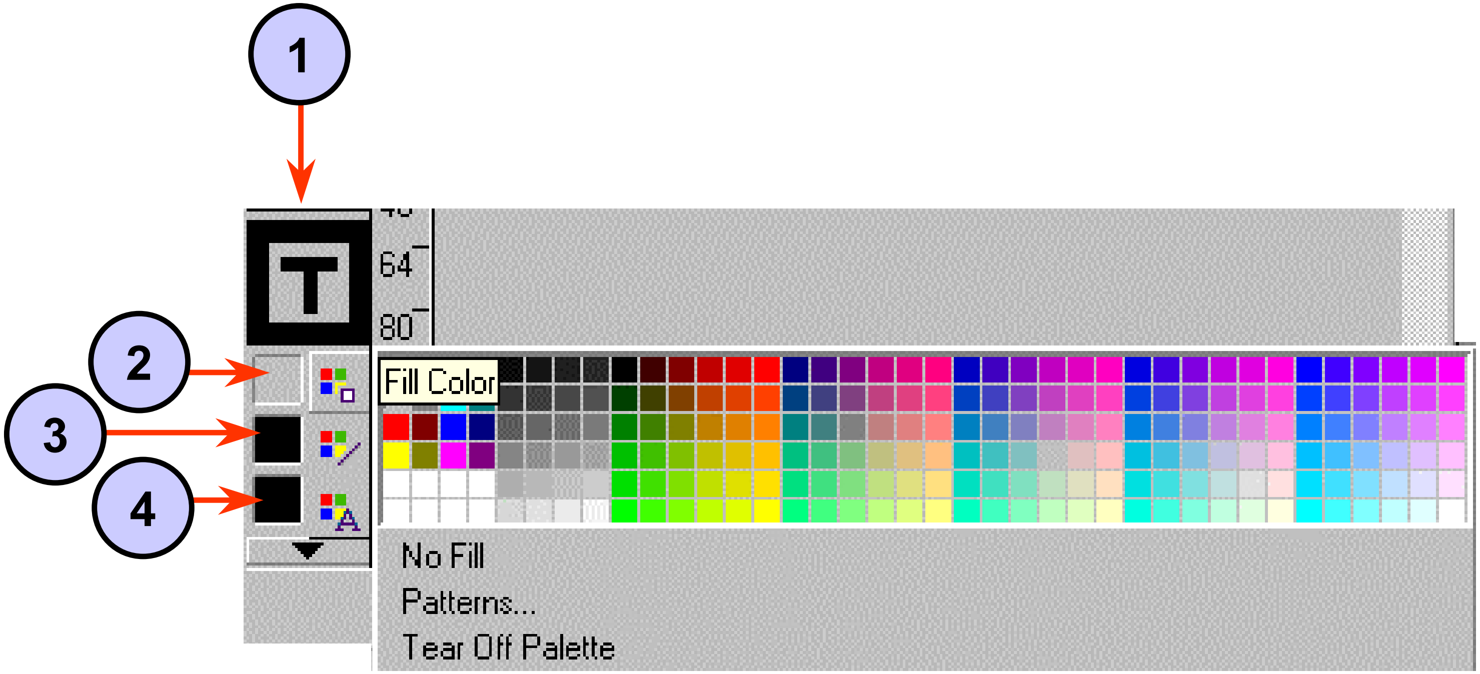
- Groups allow several objects to be repeatedly treated as one.
- Groups can be colored, moved, or resized.
- Tool-specific operations exist for groups.
- Groups have a single set of selection handles.
- Members can be added or removed.

Edit and Layout Menus





Color and Pattern Tools

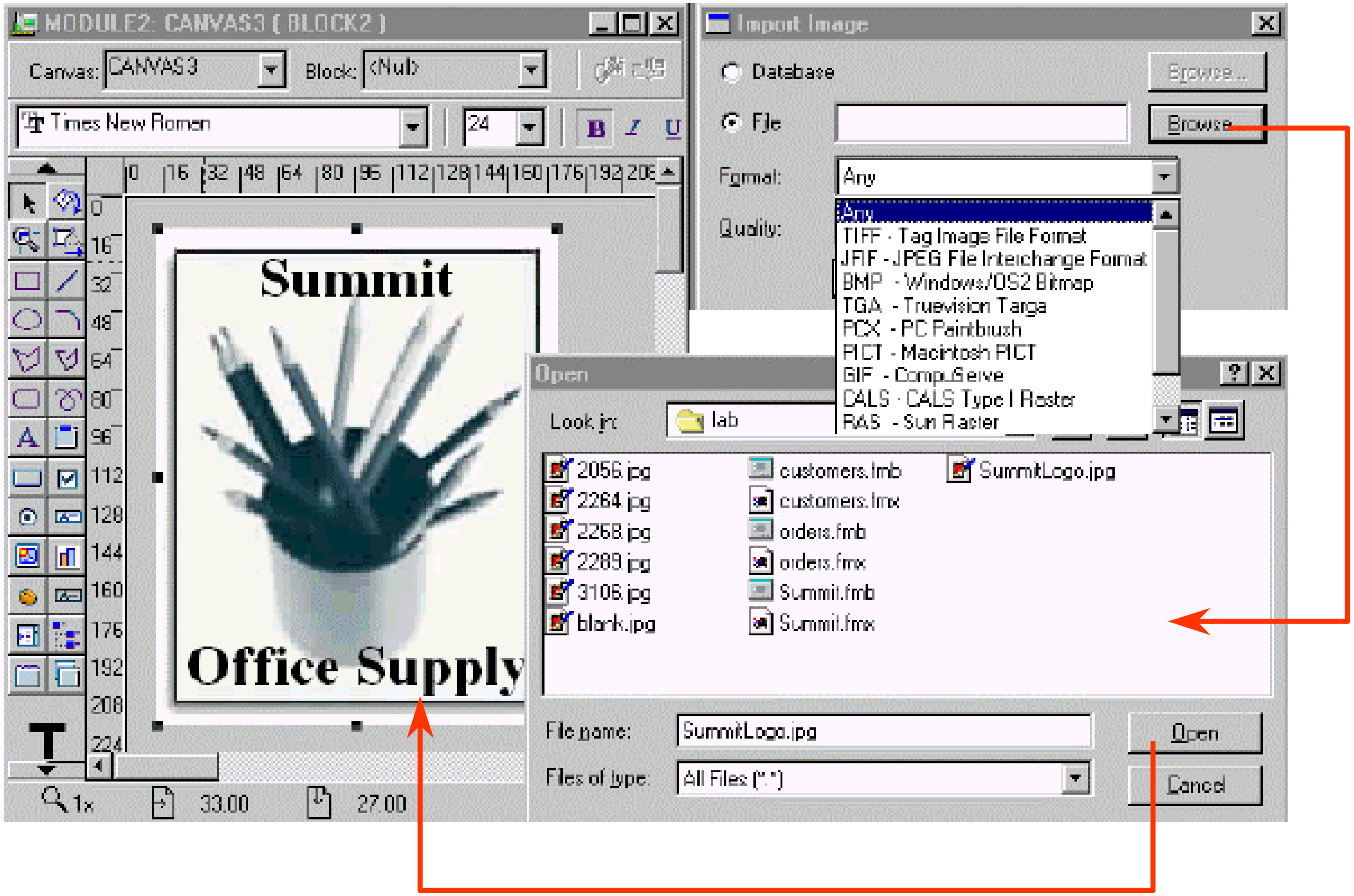


No Fill

Patterns...

Tear Off Palette

Importing Images



Canvas: CANVAS3

Block: <Null>

Times New Roman 24

○ Database

● File

Format: Any

Quality:

- Any
- TIFF - Tag Image File Format
- JFIF - JPEG File Interchange Format
- BMP - Windows/OS2 Bitmap
- TGA - Truevision Targa
- PCX - PC Paintbrush
- PICT - Macintosh PICT
- GIF - CompuServe
- CALS - CALS Type I Raster
- RAS - Sun Raster

Look in: lab

2056.jpg	customers.fmb	SummitLogo.jpg
2264.jpg	customers.fmx	
2268.jpg	orders.fmb	
2289.jpg	orders.fmx	
3106.jpg	Summit.fmb	
blank.jpg	Summit.fmx	

File name: SummitLogo.jpg

Files of type: All Files (*.*)

1x 33.00 27.00

Summary

- You can create objects by:
  - Choosing a palette tool
  - Clicking and dragging on a layout region
- There are color palette tools for fill area, lines, and text.
- View, Edit, and Layout menus display additional options for layout.
- Objects can be grouped for operations.
- You can import images by using Edit > Import.