

**Oracle Database:Oracle  
Developer:Reports & BI publisher-  
IBM Program**

**Student Guide – Volume 1**

D81924GC10  
Edition 1.0  
June 2013

**ORACLE®**

**Copyright © 2013, Oracle and/or its affiliates. All rights reserved.**

**Disclaimer**

This document contains proprietary information and is protected by copyright and other intellectual property laws. You may copy and print this document solely for your own use in an Oracle training course. The document may not be modified or altered in any way. Except where your use constitutes "fair use" under copyright law, you may not use, share, download, upload, copy, print, display, perform, reproduce, publish, license, post, transmit, or distribute this document in whole or in part without the express authorization of Oracle.

The information contained in this document is subject to change without notice. If you find any problems in the document, please report them in writing to: Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. This document is not warranted to be error-free.

**Restricted Rights Notice**

If this documentation is delivered to the United States Government or anyone using the documentation on behalf of the United States Government, the following notice is applicable:

**U.S. GOVERNMENT RIGHTS**

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

**Trademark Notice**

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

## **Contents (*Oracle Developer: Reports Accelerated - IBM Graduate Program*)**

### **1 Introduction to Oracle Reports Developer**

- Objectives 1-2
- Business Intelligence 1-3
- Enterprise Reporting 1-5
- Web Publishing 1-9
- Paper Publishing 1-10
- Oracle Reports Developer 1-11
- Benefits 1-12
- Oracle 10g Products 1-13
- Oracle Database 10g 1-14
- Oracle Developer Suite 10g 1-15
- Oracle Application Server 10g 1-18
- Oracle Reports Developer 1-20
- Summary 1-22

### **2 Designing and Running Reports**

- Objectives 2-2
- Understanding User Requirements 2-3
- Designing Reports 2-5
- Tabular 2-7
- Master-Detail 2-8
- Master with Two Details 2-9
- Matrix 2-10
- Retrieving and Sharing Data 2-11
- Running a Report 2-13
- Previewing Reports 2-15
- Print Preview 2-16
- Supported File Types 2-19
- Summary 2-21
- Practice 2 Overview 2-22

### **3 Working in Oracle Reports Developer**

- Objectives 3-2
- Reports Developer Executables 3-3
- Invoking Reports Builder 3-5
- Reports Builder Modules 3-6

Report Data and Layout	3-7
Reports Builder Components	3-8
Main Menu Structure	3-13
Wizards	3-18
Report Editor	3-19
PL/SQL Development Environment: Syntax Palette	3-21
Object Navigator	3-22
Report-Level Objects	3-23
Data Model Objects	3-24
Paper Layout Objects	3-25
Paper Parameter Form Objects	3-26
Object Interrelationship	3-27
Customizing Your Oracle Reports Developer Session	3-28
Saving Preferences	3-29
Oracle Reports Environment Variables	3-30
Using the Online Help System	3-32
Summary	3-34
Practice 3 Overview	3-35

#### **4 Creating a Paper Report**

Objectives	4-2
Report Module Components	4-3
Building a Paper Report	4-4
Invoking the Report Wizard	4-5
Choosing the Layout Type	4-6
Creating a Tabular Report	4-7
Selecting the Data Source Type	4-9
Using Query Builder	4-10
Building a Query	4-11
Query Builder Functions	4-13
Selecting Displayed Fields	4-15
Totals and Labels	4-16
Selecting a Report Template	4-17
Viewing the Paper Report Output	4-18
Saving the Report Definition	4-19
Reentering the Wizard	4-20
Creating a New Report	4-22
Creating Break Reports	4-23
Break Report Labels	4-24
Creating Mailing Labels and Letters	4-25
Creating a Matrix Report	4-27

Previewing a Paper Report in a Browser	4-29
Summary	4-31
Practice 4 Overview	4-32

## **5 Enhancing a Basic Paper Report**

Objectives	5-2
What Is the Paper Design?	5-3
The Paper Design Window	5-4
Modifying a Report	5-5
Aligning Columns	5-6
Setting a Format Mask	5-7
Manipulating Objects	5-9
Inserting Page Numbers, Dates, and Times	5-10
Customizing Dates	5-12
Summary	5-13

## **6 Enhancing Reports Using the Data Model: Queries and Groups**

Objectives	6-2
The Data Model Objects	6-3
Modifying Properties of a Query	6-5
More Properties	6-8
Applying Changes	6-10
Changing the Group Structure	6-12
Group Hierarchy	6-14
Ordering Data in a Group	6-16
Query Modifications	6-18
Filtering Data in a Group	6-20
Using a Packaged Filter	6-21
Summary	6-22
Practice 6: Overview	6-23

## **7 Enhancing Reports Using the Data Model: Data Sources**

Objectives	7-2
Data Source Types	7-3
Summary	7-5

## **8 Enhancing Reports Using the Data Model: Creating Columns**

Objectives	8-2
Data Model Columns	8-3
Maintaining Data Source Columns	8-5
Producing File Content Output	8-7

Creating a Column	8-9
Creating Summary Columns	8-11
Displaying Subtotals	8-13
Displaying Percentages	8-15
Resetting Summary Values	8-17
Creating a Formula Column	8-18
Creating a Placeholder Column	8-20
Populating a Placeholder Column	8-22
Summary	8-24
Practice 8 Overview	8-25

## **9 Enhancing Reports Using the Paper Layout**

Objectives	9-2
Viewing the Paper Layout	9-3
Viewing the Section Areas	9-7
Designing Multipanel Reports	9-8
Printing Multipanel Reports	9-10
Different Objects in the Paper Layout	9-11
The Paper Layout Layers	9-13
Avoiding Layout Errors	9-15
Report Processing	9-17
Creating Layout Objects	9-19
Paper Layout Tools	9-21
Creating Variable Length Lines	9-22
Summary	9-24
Practice 9 Overview	9-25

## **10 Controlling the Paper Layout: Common Properties**

Objectives	10-2
Modifying Paper Layout Object Properties	10-3
Comparing Properties	10-5
Common Layout Properties	10-6
Sizing Objects	10-7
Anchors	10-9
Layout Object Relationships	10-11
Pagination Icons in the Paper Layout	10-14
Using Page Break Before	10-16
Using Page Break After	10-21
Using Page Protect	10-23
Controlling Print Frequency	10-26
Using Format Triggers	10-28

Summary 10-29  
Practice 10 Overview 10-30

## **11 Controlling the Paper Layout: Specific Properties**

Objectives 11-2  
Properties of a Repeating Frame 11-3  
Specifying Print Direction 11-4  
Controlling the Number of Records per Page 11-6  
Controlling Spacing Between Records 11-7  
Minimum Widow Records 11-8  
Column Mode 11-10  
Properties of a Field 11-12  
System Variables 11-14  
Page Numbering 11-15  
Valid Source Columns 11-17  
When Are the Contents Updated? 11-19  
Summary 11-20  
Practice 11 Overview 11-21

## **12 Creating and Using Report Parameters**

Objectives 12-2  
Creating User Parameters 12-3  
Referencing Parameters in a Report Query 12-6  
Using Bind References 12-8  
Using Lexical References 12-10  
Hints and Tips When Referencing Parameters 12-12  
Creating a List of Values 12-14  
Referencing System Parameters 12-18  
Building a Paper Parameter Form 12-20  
Customizing a Paper Parameter Form 12-22  
Using Parameter Form HTML Extensions 12-24  
Parameter Form Header and Footer 12-26  
Summary 12-27  
Practice 12 Overview 12-28

## **13 Coding PL/SQL Triggers**

Objectives 13-2  
Types of Triggers in Reports 13-3  
Trigger Code 13-5  
Using Report Triggers 13-6  
Using Data Model Triggers: PL/SQL Group Filter 13-10

Using Data Model Triggers: Parameter Validation	13-12
Using Layout Triggers	13-14
Using a Format Trigger on a Frame	13-16
Using a Format Trigger on a Repeating Frame	13-18
Using a Format Trigger on a Field	13-20
Using a Format Trigger in a Web Layout	13-21
Using a Format Trigger on a Boilerplate Object	13-22
Writing Common Code	13-24
Event-Based Reporting	13-26
Event-Driven Publishing API	13-27
Invoking a Report from a Database Event	13-29
Summary	13-31
Practice 13 Overview	13-33

## **14 Extending Functionality Using the SRW Package**

Objectives	14-2
Contents of the SRW Package	14-3
Outputting Messages	14-5
Executing a Nested Report	14-7
Restricting Data	14-11
Initializing Fields	14-13
Creating a Table of Contents	14-15
Performing DDL Statements	14-17
Setting Format Attributes	14-19
Using Format Attributes in a Web Layout	14-22
Summary	14-24
Practice 14 Overview	14-25

## **15 Building Reports: Efficiency Guidelines**

Objectives	15-2
Tuning Reports	15-3
Performance Measurement	15-5
Investigating the Data Model	15-6
Investigating the Paper Layout	15-10
Running the Report	15-12
Setting NLS Language Environment Variables	15-13
Translating an Oracle Reports Application	15-15
Summary	15-17
Practice 15 Overview	15-18

## **Appendix A: CASE LITE**

# 1

## Introduction to Oracle Reports Developer

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Objectives

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

- Describe Oracle Reports and its evolution
- Describe the key features of Oracle Reports Developer
- Describe the architecture of Oracle Application Server
- Compare various Reporting Tools.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

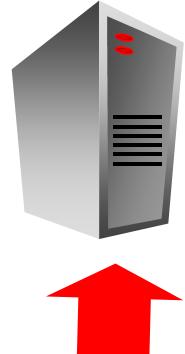
## Overview

Oracle Reports Developer is a powerful enterprise reporting tool that allows developers to rapidly develop and deploy sophisticated high quality reports from any data source, in any format, to any destination.

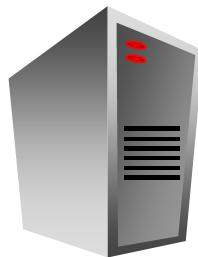
This lesson identifies the key features and benefits of Oracle Reports Developer and its relationship with Oracle's end-to-end business intelligence solution.

# Business Intelligence

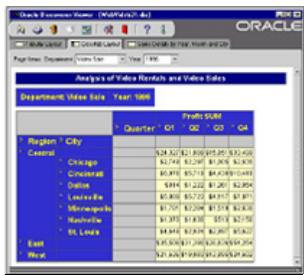
Discoverer Server



Reports Server



XML



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## What Is Business Intelligence?

Reporting is the delivery of information to information consumers. These consumers must often further investigate that information. Historically, different tools have performed different tasks. There is however a growing need to integrate data consolidation, data analysis, and enterprise reporting tools. This provides a seamless environment that allows users to move from the role of pure information consumer to information investigator.

The combination of providing information and enabling additional investigation of that information is commonly referred to as business intelligence (BI).

Oracle offers an integrated business intelligence solution that provides the user with a complete picture across the entire organization. The Oracle BI solution is designed to easily and quickly put data into the database, find information from the database, share this information, and exploit BI to learn more about a business and its customers.

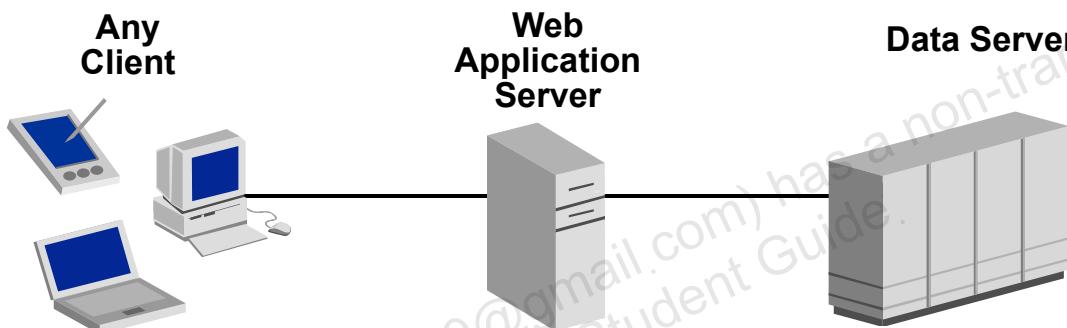
## **What Is Business Intelligence? (continued)**

Oracle's set of integrated BI tools allow you to seamlessly move between the roles of information consumer and information investigator:

- Oracle Warehouse Builder (OWB) is designed to consolidate disparate data sources, performs any required data transformations, manage warehouse lifecycle, and integrate with analysis tools.
- OracleAS Discoverer, the powerful ad hoc query and analysis tool, can be unleashed to reveal potential opportunities and risks associated with your products, customers, and marketplace.
- Oracle Reports, the high-fidelity enterprise reporting tool, enables businesses to give immediate access to information to all levels within and outside of the organization in a scalable and secure environment. Oracle Reports is the solution for Web and paper publishing, enabling you to publish any data, in any format, anywhere. Oracle Reports delivers high-quality information to users with only an Internet browser in open Internet document standards. Authentication is handled through single sign-on, and scalability is guaranteed through OracleAS Reports Services.

# Enterprise Reporting

- Provides access for more users to vital customized information
- Lowers cost of ownership
- Uses integrated business intelligence



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Enterprise Reporting

For as long as there has been electronic data storage, there has been reporting. In the mainframe computing era a “one-report-fits-all” metaphor was used. Reports were expensive to develop and were typically generated in overnight batch jobs. With the advent of the personal computer and its rich graphical user interface and fourth generation languages, reports were easier to produce. Report developers could now write reports tailored to the needs of particular information consumers. Most companies however shifted from a “one-fits-all” model to a paradigm of “one-report-fits-one”. Reports were stored on an individual PC, or at best shared across small work groups. Enterprise reporting grew out of a business need for better, faster, and more flexible delivery of individually tailored information to a very large number of users. No environment is better suited to fulfill this than the Internet.

## **Enterprise Reporting (continued)**

Enterprise reporting offers a number of important benefits. These include:

- **Widening the reach of information access:**

The ability to provide information electronically to a large and often geographically distributed user base, in a timely manner, picking up live data on-demand, means that everybody is kept informed at all times.

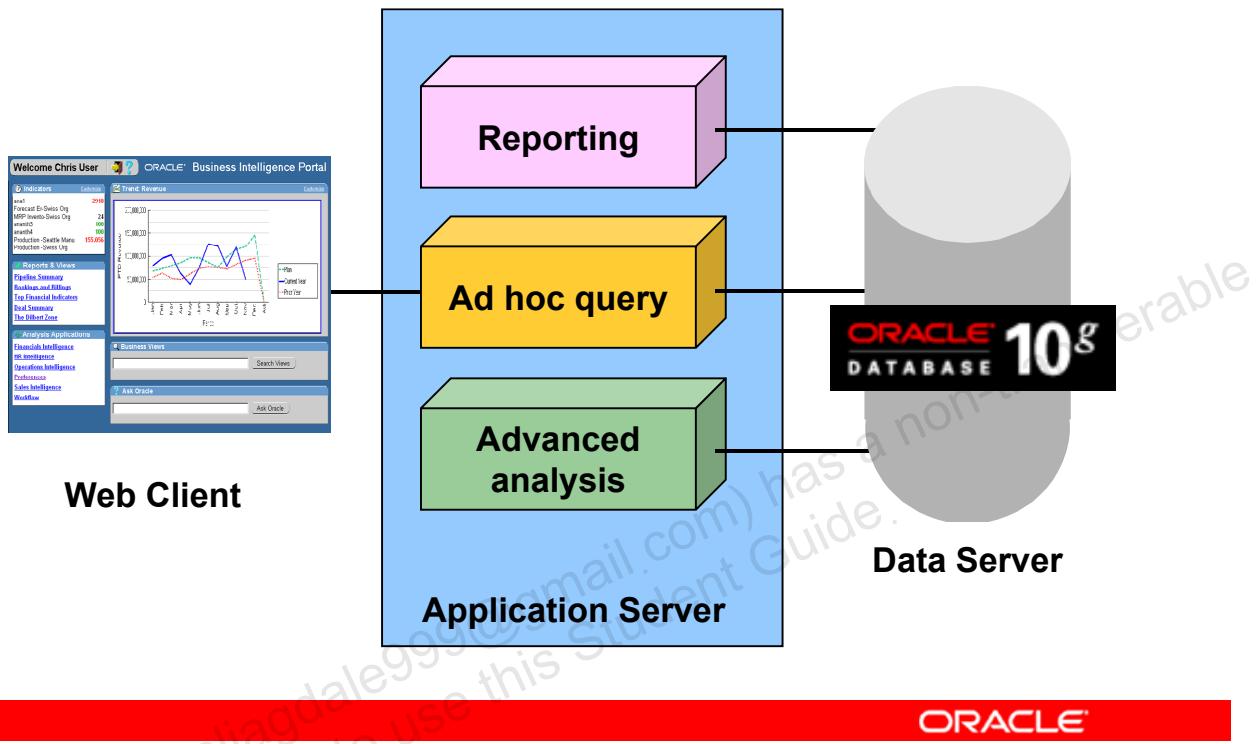
- **Lowering the cost of computing:**

Costs are minimized through a centrally managed architecture. This approach takes the labor out of the computing infrastructure, and lowers the maintenance needs and associated costs.

- **Allowing for extensible business intelligence:**

Reporting is the delivery of information. Delivery alone however is not enough. If, for example, a possible trend is identified, such as a drop or rise in sales, you must find out why and be able to move seamlessly from mere consumer to investigator.

# Enterprise Reporting



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Enterprise Reporting (continued)

Enterprise reporting provides an organization with significant benefits. In order to obtain those benefits however, a number of challenges need to be met, including:

- **High performance and scalability:**

Timely information delivery is essential. The powerhouse of Oracle's reporting architecture is Oracle Application Server. OracleAS Reports Services, the report publishing component of Oracle Application Server, provides out-of-the-box optimized performance and scalability to run Oracle Reports applications in Web and non Web environments.

- **High quality reports:**

Enterprise reporting uses the Internet as its conduit for information delivery. Oracle Reports adheres to Internet document standards and supports Hypertext Markup Language (HTML), HTML Cascading Style Sheets (HTMLCSS), Extensible Markup Language (XML), Rich Text Format (RTF), Postscript, and Portable Document Format (PDF) formats. This enables you to generate and deliver information to a browser offering a rich variety of fonts, page layouts, and graphical widgets.

## **Enterprise Reporting (continued)**

- **Reduced time to market:**

In Internet-based environments, information consumers have a low tolerance for delays in information delivery. The key to reducing development time is to take the hard labor out of report development. Oracle Reports does not require the report developer to write numerous lines of code or manually format complex layout structures. The entire development environment is wizard driven and Oracle Reports Developer generates the code automatically, making this an extremely productive development environment.

# Web Publishing



GIF89a

[hyperlinks](#)

JavaScript

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Web Publishing and Paper Publishing

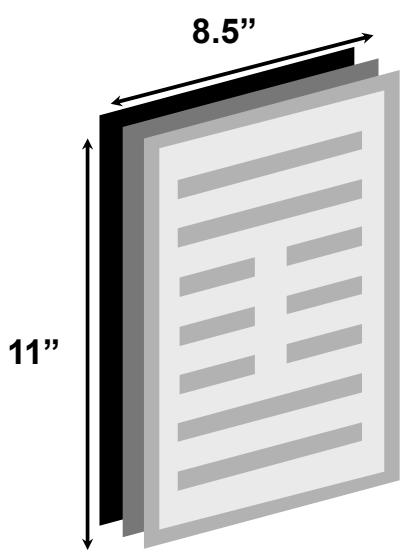
Today, many people turn to the Web first to find information. However, publishing enterprise data using a browser has not lowered the demand for high fidelity paper publishing. HTML pages that look great in a browser often do not look professional when printed from the browser. There are few tools today that recognize this and help developers publish data with high fidelity to both the Web and paper.

### Web Publishing

A Web page is very fluid. A Web page does not have the limitation of page size; a page can contain as much or as little data as you wish. If there is more content than will fit in the browser window, scroll bars are displayed to allow users to navigate through the content. Web specific features, such as bookmarks and hyperlinks, can also aid the user in navigation.

A Web page comes alive with rich images, color, JavaScript, and animation. Users can interactively drill down to see details or related data. Style sheets are used to universally control appearance.

# Paper Publishing



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Web Publishing and Paper Publishing (continued)

### Paper Publishing

A paper report has characteristics that include rigid geometry restrictions, headers and footers repeated on each page, and higher resolution allowing more details to be presented to the end user at once, giving the end user the option of printing the output and the ability to “study” the data.

Oracle Reports has historically done a very good job of publishing high fidelity paper reports. The tools understand the concept of a paper page. It manages geometry and includes features to control behavior when a page break occurs.

### Effective Publishing

The key to effective publishing is understanding the medium, utilizing the functionality the medium provides, and generating visually attractive content for that medium. For the Web, this means the ability to incorporate Java, JavaScript, and animated Graphic Interchange Formats (GIFs). For paper, it requires the understanding of paper layouts and complicated geometry management.

Recognizing the differences between Web publishing and paper publishing, Oracle Reports Developer provides a declarative environment with the power to generate high quality output for the Web and e-business requirements, as well as high fidelity paper reports.

# Oracle Reports Developer

Publish data from any source, in any format, to any destination with high fidelity.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

**ORACLE®**

## Oracle Reports Developer

Oracle Reports Developer provides an enterprise-wide publishing solution that delivers high fidelity, dynamic Web pages to corporate (intranet) and Internet users without compromising scalability and performance. Using Oracle Reports, you can publish information from any data source, in any format (PDF, HTML, printed, XML, etc.), to any destination (Web, Portal, e-mail, file, etc.) in a scalable, efficient manner.

The goal of Oracle Reports is to be the universal publishing solution of choice for any publishing need within an organization.

## Benefits

- Publish data from any source, in any format, with high fidelity
- Develop one time and deploy anywhere
- Open, standards-based, modular architecture



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Benefits

- Oracle Reports enables you to publish data from any source in any format with high fidelity. In addition to SQL, PL/SQL, and Express, you can publish data from Java Database Connectivity (JDBC) sources, XML, text files, or your own data sources that you have defined. Report output can go anywhere you wish: the Web, e-mail, the printer, wireless devices, and so forth.
- Develop your report once, deploy it anywhere.
  - Output formats include HTML, HTMLCSS, XML, PDF, Postscript, RTF, delimited, character, etc.
  - The Web layout is optimized for HTML.
  - The paper layout is designed for PDF and Postscript.
  - Share a single data model among all output formats.
  - Customize the appearance and content of your output at runtime by applying XML customization files.
- Oracle Reports uses a standards-based, modular architecture.
  - Use the Reports Java APIs to define your own plug-ins for data sources, output destinations, security infrastructure, cache management, engines, and so on.
  - Take advantage of the industry standards used in Oracle Reports: JavaServer Pages (JSPs), servlets, JavaBeans, CORBA, and IIOP information protocol.
  - Flexible design fits any middle-tier configuration.

## Oracle 10g Products

Oracle 10g provides the complete solution:

- Oracle Database 10g
- Oracle Developer Suite 10g
- Oracle Application Server 10g



ORACLE®

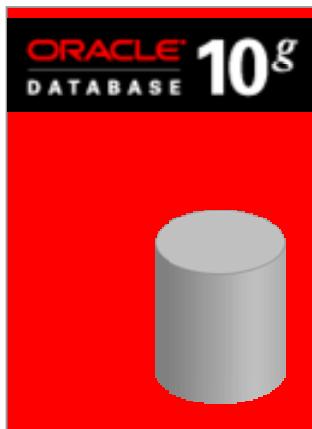
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Oracle 10g Products

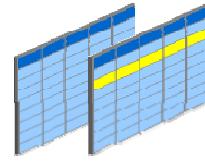
Oracle 10g provides the complete Internet infrastructure that makes it easy for companies interested in e-commerce to create and deploy scalable, Web-based applications. Oracle markets three products to help you achieve this: Oracle Database 10g, Oracle Application Server 10g and Oracle Developer Suite 10g.

# Oracle Database 10g

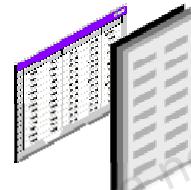
Oracle Database 10g manages all of your data:



**Relational Data**



**Documents**



**Multimedia**



**ORACLE**

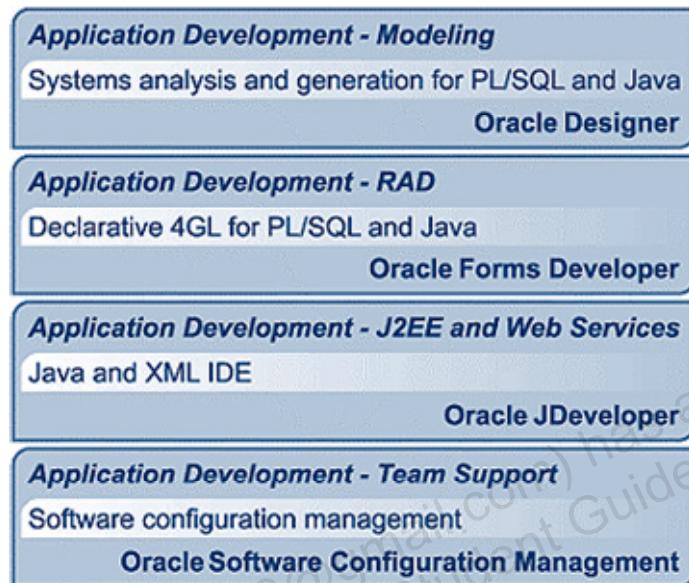
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Oracle Database 10g

The Oracle Database manages all of your structured and unstructured data, including Word documents, Excel spreadsheets, XML, images, and more. It provides a scalable, secure, and reliable architecture.

# Oracle Developer Suite 10g

## Application Development



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Oracle Developer Suite 10g

The Oracle Developer Suite offers a complete set of integrated development tools, empowering you to easily and quickly create Internet applications for personalized Web portals and hosted software services. Oracle Developer Suite 10g combines leading Oracle application development and business intelligence tools into a single integrated product. Built on Internet standards such as Java and XML, Oracle Developer Suite provides a complete development environment.

The components of Oracle Developer Suite 10g include Oracle Designer, Oracle Forms Developer, Oracle JDeveloper, Oracle Software Configuration Manager, Oracle Discoverer, Oracle Reports Developer, Oracle Warehouse Builder, and Oracle Business Intelligence Beans.

## Oracle Developer Suite 10g (continued)

For application development, Oracle Developer Suite 10g provides capabilities in:

- **Modeling:** Oracle Designer 10g delivers dramatic increases in productivity for database application developers. Oracle Designer provides a complete toolset to model, design, generate, and capture the enterprise application requirements.
- **Rapid Application Development (RAD):** RAD capabilities in Oracle Developer Suite feature integrated builders, reentrant wizards, live previewers, and property inspectors. Oracle JDeveloper is an integrated development environment with end-to-end support for modeling, developing, debugging, optimizing, and deploying Java applications and Web services. Oracle JDeveloper 10g introduces a new approach to J2EE development with features that enable visual and declarative development. The innovative Oracle Application Development Framework simplifies J2EE development. Oracle JDeveloper's "productivity with choice" offers a choice of: development approach, technology scope, and deployment platform.
- **J2EE and Web Services:** Oracle Developer Suite supports the latest J2EE application programming interfaces (APIs) including Enterprise Java Beans (EJB), JavaServer Pages (JSP), and servlets. Web services support Simple Object Access Protocol (SOAP), Web Service Definition Language (WSDL), and Universal Description, Discovery, and Integration (UDDI).
- **Team Support:** Oracle Software Configuration Management provides versioning, dependency management, and impact analysis for all objects and file types.

# Oracle Developer Suite 10g

## Business Intelligence



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Oracle Developer Suite 10g (continued)

For business intelligence, Oracle Developer Suite provides the capabilities for:

- **Extract, Transform, and Load:** Oracle Warehouse Builder provides an easy to use, graphical environment for rapidly designing, deploying, and managing business intelligence systems. It also provides an extensible framework for integrating a diverse set of data sources with BI tools.
- **End User Query and Analysis:** With Oracle Discoverer, you can create, modify, and execute ad hoc queries and reports. More casual users can view and navigate through predefined reports and graphs. Discoverer provides a business view to hide the complexity of the underlying data structure. It enables you to focus on solving business problems and brings insight to your data.
- **Enterprise Reporting:** Oracle Reports Developer enables you to access any data, publish it in any format, and send it to any destination.

# Oracle Application Server 10g



**ORACLE®**

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Oracle Application Server 10g

Oracle Application Server 10g is a comprehensive and integrated application server that runs any Web site, portal, or Internet application. Oracle Application Server consists of a set of services and utilities that can be used to implement applications in a distributed environment for scalability and reliability. These are:

- **Communication Services:** These services handle incoming requests received by Oracle Application Server. Some of these requests are processed by the Oracle HTTP Server and some requests are routed to other areas of Oracle Application Server for processing.
- **Presentation Services:** The presentation services of Oracle Application Server generally output some kind of graphical representation, often in the form of HTML. Oracle Application Server supports a variety of ways to generate presentation presentations that can be delivered to the client ranging from low level programming using Perl scripts and Java servlets, to high level frameworks using Oracle portal services.

## Oracle Application Server 10g (continued)

- **Business Logic Services:** Oracle Application Server provides several ways to develop business logic, utilizing both Java development approaches and high level, model-driven techniques. These approaches include Java 2 Platform Enterprise Edition (J2EE), Enterprise JavaBeans (EJB), and Oracle Business Components for Java (BC4J), as well as rich GUI oriented approaches such as Oracle Forms Developer and Oracle Reports Developer.
- **Data Management Services:** To reduce the load on the database instance and to avoid network roundtrips for read-only data, Oracle Application Server includes Oracle Application Server Web Cache.
- **System Services:** To provide system management and security services, Oracle Application Server includes Oracle Enterprise Manager and Oracle Advanced Security. These system services provide a comprehensive management framework for your entire Oracle environment and network security using Secure Sockets Layer (SSL)-based encryption and authentication facilities.

### Technical Note

For more information on Oracle Application Server, refer to the Oracle Technology Network:  
<http://otn.oracle.com>.

# Oracle Reports Developer

- User-friendly wizards
- Pluggable data sources
- Customizable report templates
- WYSIWYG live editor for paper reports
- Dynamic Web publishing using JSP/HTML
- Run-time customization
- Dynamic SQL execution
- Portal integration
- Event-based reporting



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## What Is Oracle Reports Developer?

Oracle Reports Developer is a component of the Oracle Developer Suite. Oracle Reports Developer is a collection of programs that allows you to centralize report processing and better manage reporting efforts. Reports Builder is one of the program components included with Oracle Reports Developer. Features include:

- Wizards that guide you through the report design process
- The ability to access data from any data source
- A query builder with a graphical representation of the SQL statement to obtain report data
- Default report templates and layout styles that can be customized if needed
- A live editor that allows you to modify paper report layouts in WYSIWYG (“what you see is what you get”) mode
- The ability to add dynamic report output to an HTML page by embedding custom JavaServer Page (JSP) tags within an HTML document
- An integrated chart builder to graphically represent report data
- The ability to generate code to customize how reports will run

## **What Is Oracle Reports Developer? (continued)**

- Web publishing tools that dynamically generate Web pages based on your data
- Standard report output formats such as HTML, HTMLCSS, XML, PDF, PCL (Printer Control Language), Postscript, and ASCII
- The ability to apply run-time customization
- The ability to execute dynamic SQL statements within PL/SQL procedures
- Support for Oracle objects
- Seamless integration of Oracle Reports with OracleAS Portal for administering report security
- The ability to publish report output to portlets
- Report execution based on database events

## Summary

In this lesson, you should have learned how to:

- Describe integrated business intelligence
- List the benefits of enterprise reporting
- Describe the challenges of publishing for different media
- List the Oracle 10g products
- List the key features and benefits of Oracle Reports Developer
- Compare various Reporting Tools



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Summary

- A business intelligence solution provides a set of integrated tools that enable you to move seamlessly between the roles of information consumer and information investigator.
- Enterprise reporting grew out of a business need for better, faster, and more flexible delivery of individually tailored information to a very large number of users. The conduit of choice is the Internet.
- Effective publishing requires an understanding of the features and limitations of the different medium. Reports need to be designed for the specific output medium. Web publishing and paper publishing differ.
- Oracle Reports Developer enables you to create and deploy reports using any data, in any format, for any medium.
- The Oracle 10g products provide the infrastructure to easily develop, deploy, and manage Internet applications and Web sites. Oracle Reports Developer is a component of the Oracle Developer Suite. Reports created with Oracle Reports are deployed by Oracle Application Server, specifically OracleAS Reports Services.

## Designing and Running Reports



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Objectives

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

- Identify user requirements
- Name the common report styles
- Describe the structure of each style
- Run prebuilt reports as an end user
- Identify supported report file types



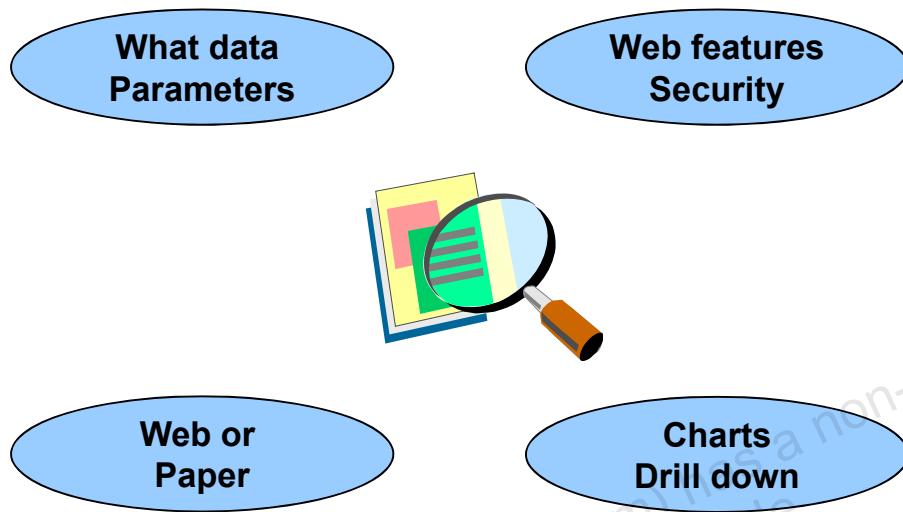
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Overview

Your reports need to satisfy the requirements of your business. To help you begin the process of translating report requirements into Reports Developer solutions, you need to understand the users' needs, the potential range of report styles, the distribution and output requirements, and gain an appreciation of the underlying report structure. This helps you to make the right decision about which style to use for the report.

This lesson discusses understanding the user requirements, standard report design styles, and describes options to execute reports.

# Understanding User Requirements



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Understanding User Requirements

In the first stage of the development, you determine what the user needs and expects. While it may be tempting to skip this stage and start building right away, it is not a good idea to do so. Without a clear understanding of the users and their reporting needs, it is virtually impossible to create effective enterprise reports.

To define user requirements:

- Gather relevant policies, business rules, and existing documentation
- Observe users and their daily job activities
- Interview a wide variety of users

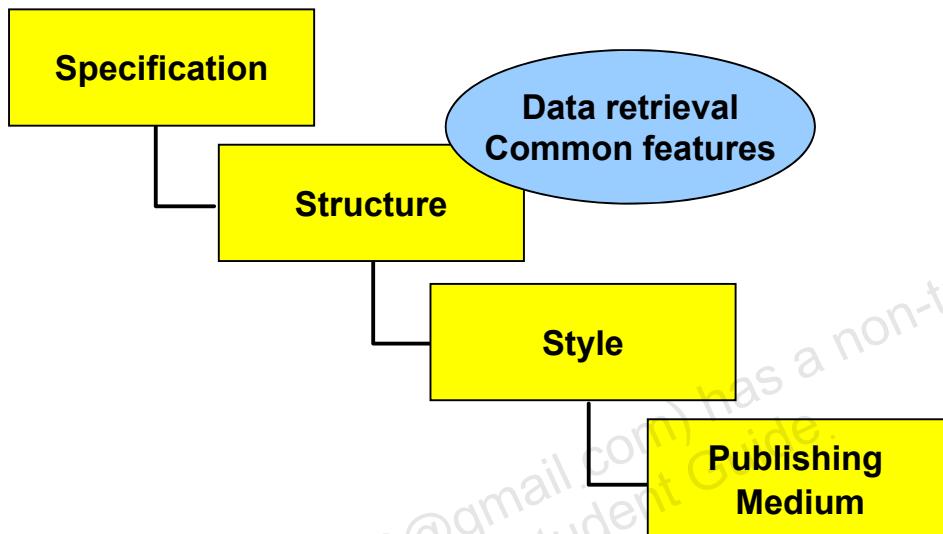
## **Understanding User Requirements (continued)**

Helpful questions to help you determine the user requirements:

- What data will people want, and in what priority?
- How is the data stored?
- Is there a corporate standard that must be met? If so, define standard templates.
- Will users want Web reports, paper reports, or both?
- For Web reports, will the reports be static or dynamic?
- Will users want charts in the report? If so, what data will be used in the graph?
- Will users want to drill down on data? If so, plan on using hyperlinks.
- Will users want to specify input parameters? If so, you need to create the necessary parameters and establish the validation rules.
- Will users want a report to be embedded in a form? If so, you will have to call the report from a form and have the form pass the data to the report.
- Will the same report serve different types of users? If so, you need to think about report sectioning and report distribution.
- Are there any administration or security issues? If so, you need to set up the necessary specifications in the OracleAS Portal Reports Security option.
- Will the users want run-time customizations? If so, plan on using XML files.

# Designing Reports

Before you start development, consider:



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Designing Reports

Before you create a report using any report-writing software, you must first consider the type of report that you are being asked to produce. You will have a specification of the needs, required output, and the expected publishing medium, but you also need to know the underlying structure that supports the requirement and the most efficient way to retrieve data.

Also, consider whether this is a one-time requirement, or whether this specification shares common features with other reports, especially where multiple reports are required in the same application.

There are a few standard styles of reports that form the majority of all reporting requirements. This section teaches you to recognize the common styles in order to enable you to interpret your report requirements and choose the correct style when developing report definitions.

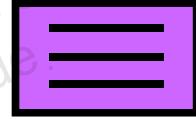
## **Designing Reports (continued)**

The majority of report requirements fall into the following categories:

- Tabular
- Master-detail
- Master and multiple details
- Matrix

# Tabular

List of Products		
Product Number	Description	Price



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

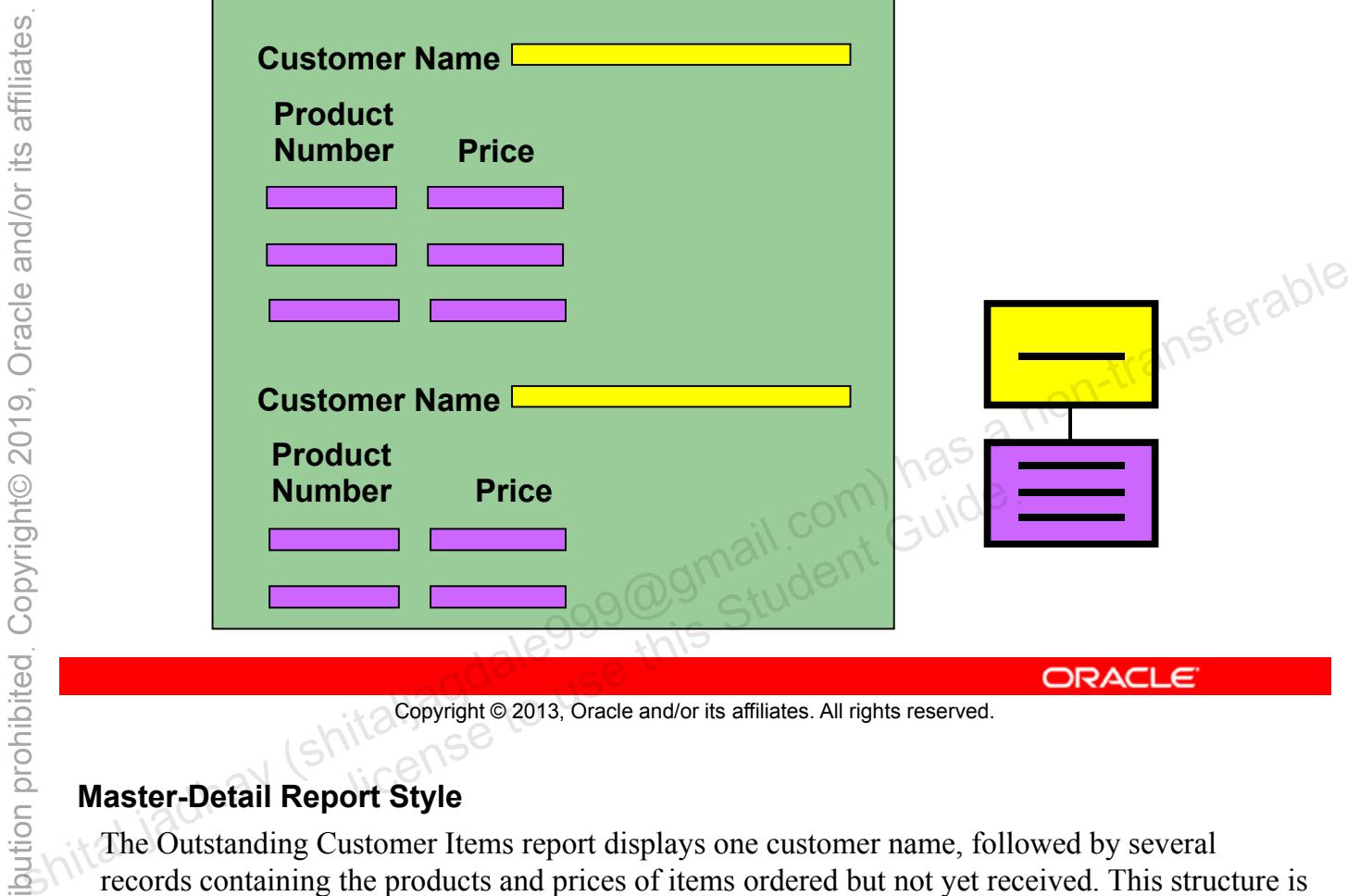
## Tabular Report Style

You define the report structure by identifying the number of times each piece of data is printed. Separate the data into groups based on this frequency. You may also need to identify any relationships that exist between groups, so that you can create a *hierarchy* in the internal report structure.

In the List of Products report displayed above, all fields for each product number, product description, and price are printed with the same frequency. They each repeat a value for every product record.

This report structure contains a single group.

In tabular reports, the headings or labels appear once above each field.



### Master-Detail Report Style

The Outstanding Customer Items report displays one customer name, followed by several records containing the products and prices of items ordered but not yet received. This structure is repeated for every customer who has unfulfilled orders. The customer name does not repeat for every outstanding product; the frequency is different.

This report structure contains two groups.

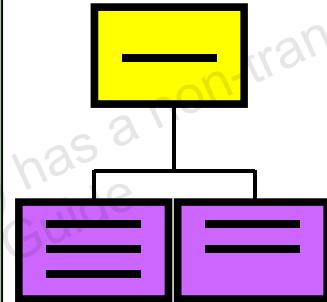
Each list of products is directly related to the preceding customer name, which means that there is a master-detail hierarchy. Customer information is in one group, called the *master*, and the product information is in a second group, called the *detail*.

In this Master-Detail report, the headings or labels appear as follows:

Group	Label Placement
Master	Labels appear to the left of fields.
Detail	Labels appear above fields (as for a tabular report).

## Master with Two Details

Customer Statistics					
Customer					
Outstanding Items		Orders in Last Six Months			
Product Number	Price	Order No.	Date	Total Value	



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Master with Two Details Report Style

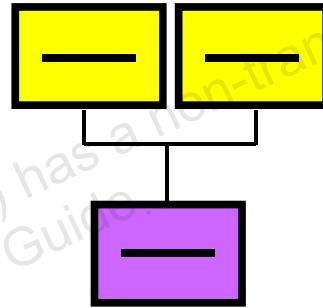
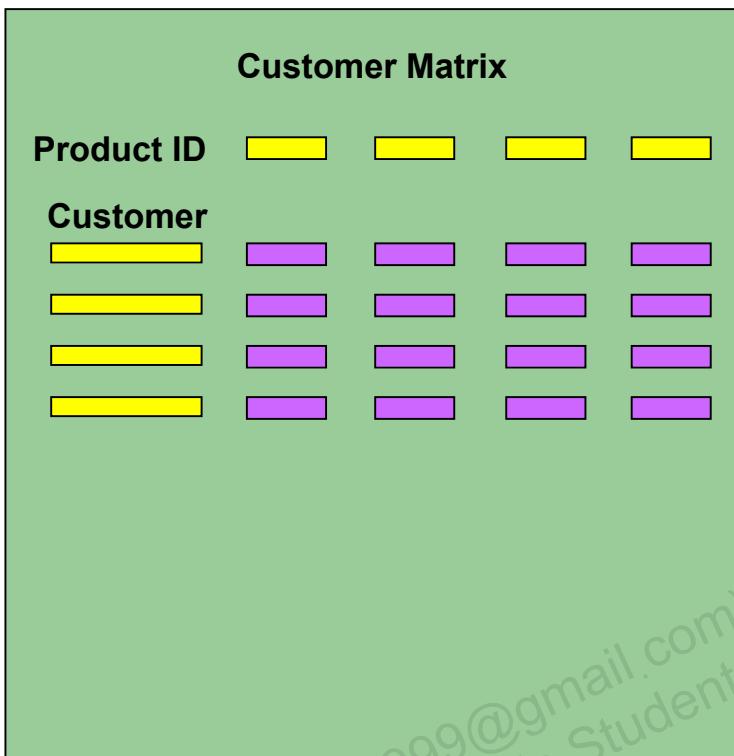
More than one group may appear at the same level in a hierarchy.

Consider the Customer Statistics report. This report has the same information as the Outstanding Customer Items report, with the addition of another group displaying orders placed in the last six months.

Both groups, Outstanding Items and Orders in Last Six Months, relate to each customer, but not directly to each other.

This report has three groups, one master and two detail groups. The two detail groups are related to the master at the same lower level.

# Matrix



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Matrix Report Style

The Customer Matrix report displays a group of master customer records down the page, a group of master product records across the page, and a group of detail summary calculations.

The detail group is related to both of the master groups.

In this report structure, the master groups are on the same level, with the detail group below.

This is an example of a simple matrix. Later you will see that matrix reports can have multiple levels of nesting.

The matrix style is the only style in which a detail group is related to two different master groups at the same level.

## Retrieving and Sharing Data

- Keep database access to a minimum
- Consider report structure and number of queries
- Make effective use of common code and objects



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Retrieving and Sharing Data

One of the primary considerations of any report is to ensure that it runs efficiently. The following points provide guidance for future reference.

#### SQL and Database Access

In Oracle Reports, data for your report can come from any data source. This section provides considerations for data stored in relational databases.

For data that is retrieved from a database, you use SQL SELECT statements. Aim to keep database access to a minimum. The fewer queries your report contains, the faster it retrieves data.

With hierarchical reports that contain more than one group, you can either use one query and create additional groups, or use many queries and manually link the groups. The one-query approach is usually preferable.

## **Retrieving and Sharing Data (continued)**

### **Report Structures**

The report style also affects the number of queries required. A master with multiple detail groups requires an additional query for each extra “ sibling” group. Matrix reports can contain one or several queries.

The following table shows the number of queries you can use to create the reports in this lesson.

<b>Report Name</b>	<b>Number of Queries</b>
List of Products	1
Outstanding Customer Items	1 or 2
Customer Statistics	2 or 3
Customer Matrix	1, 2, or 3

### **Using Common Code**

In Oracle Reports, you can create queries, PL/SQL libraries, and stored program units that are accessible to more than one report. Consider which code is common and ensure that all developers have access to it.

### **Using Common Objects**

Many reports have a standard layout for features such as company logo image, font size, and style. Paper reports often have a standard page size and margin widths. For paper reports, consider creating one template containing these common features that can be the basis of all reports. You can also enhance individual reports with specific changes. For Web reports, consider using an HTML template for your reports. Good template design standardizes the appearance of your reports and helps to ensure that, as a developer, you maintain documented standards.

## Running a Report

- Your browser
- OracleAS Portal application
- Command line (Start > Run)
- OracleAS Reports Queue Manager
- Java application
- Database trigger
- Menu integrated with a Forms application
- Button in a Forms application



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Running a Report

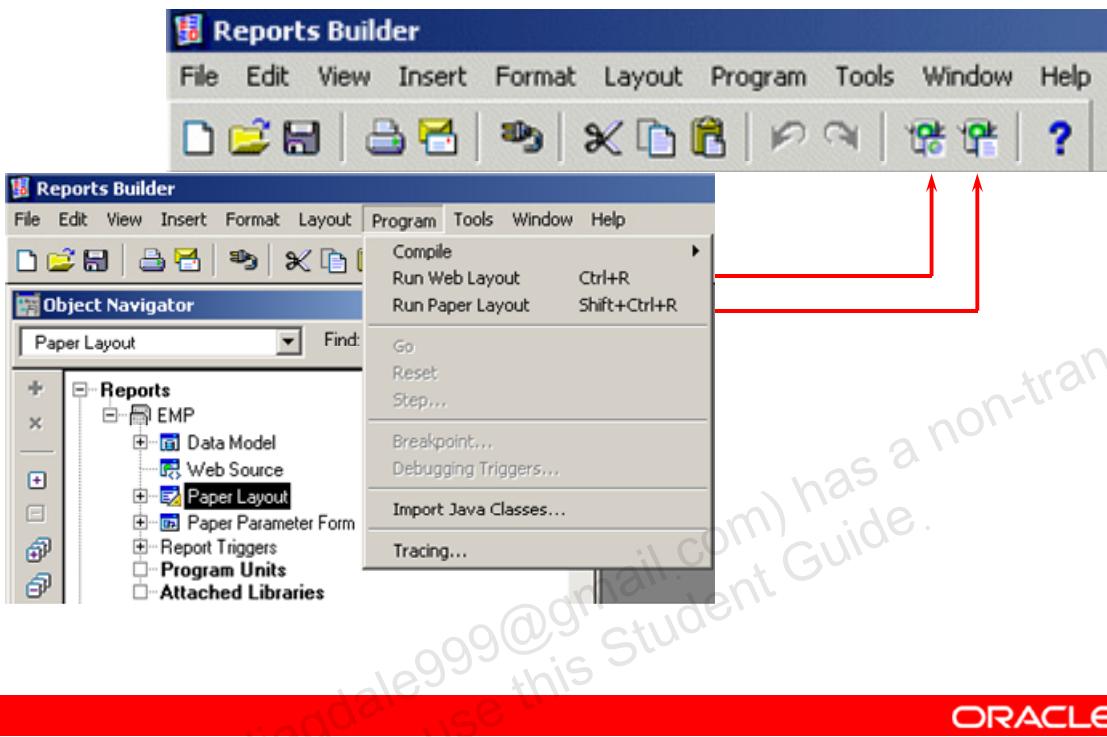
There are many ways of running a report, depending on the application design. You can call a report from:

- A Web browser
- An OracleAS Portal application
- The command line, using the Start > Run option in Windows
- The OracleAS Reports Queue Manager

## **Running a Report (continued)**

- A Java application
- A database trigger
- A customized menu in a Forms application
- A button in a Forms application

# Previewing Reports



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Previewing Reports

In Reports Builder, there are a number of ways in which you can preview your report.

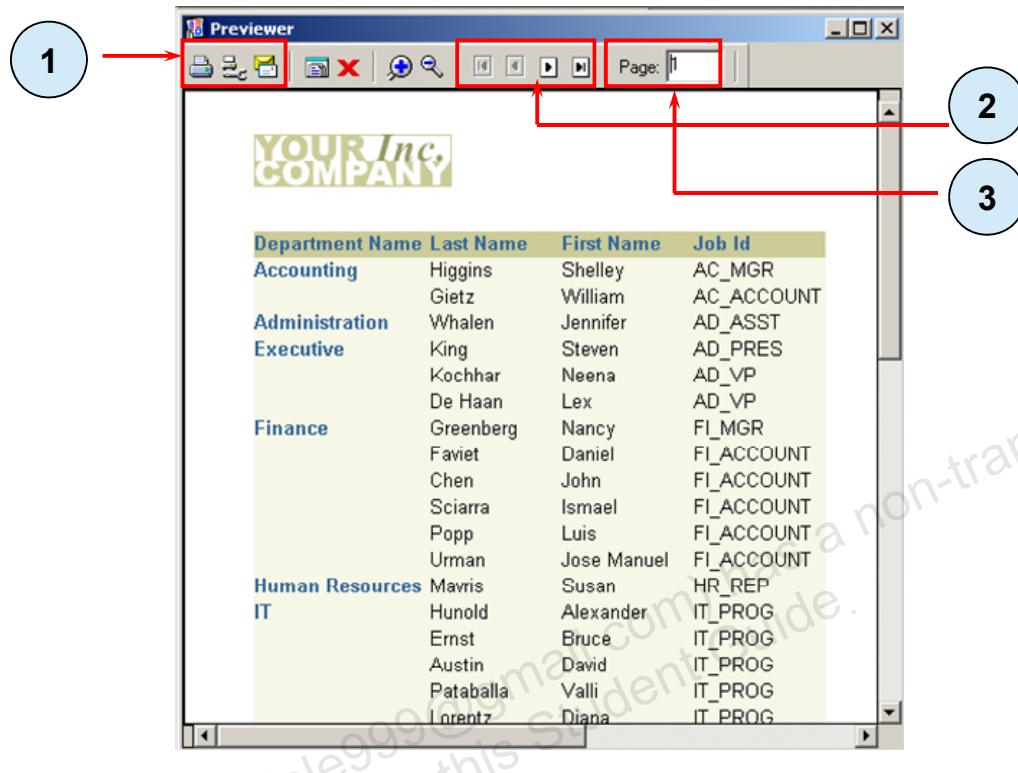
### Run Web Layout

Reports Builder provides the option of previewing your Web report in a browser. Select Program > Run Web Layout from the menu, or click the Run Web Layout iconic button on the horizontal toolbar.

### Run Paper Layout

To preview your paper report, select Program > Run Paper Layout from the menu, or click the Run Paper Layout iconic button on the horizontal toolbar. Your output displays in the Report Editor, which you will learn more about later in this course.

# Print Preview



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Previewing Reports (continued)

### Print Preview

Select File > Print Preview to display your formatted paper report. Your output displays in the Previewer window.

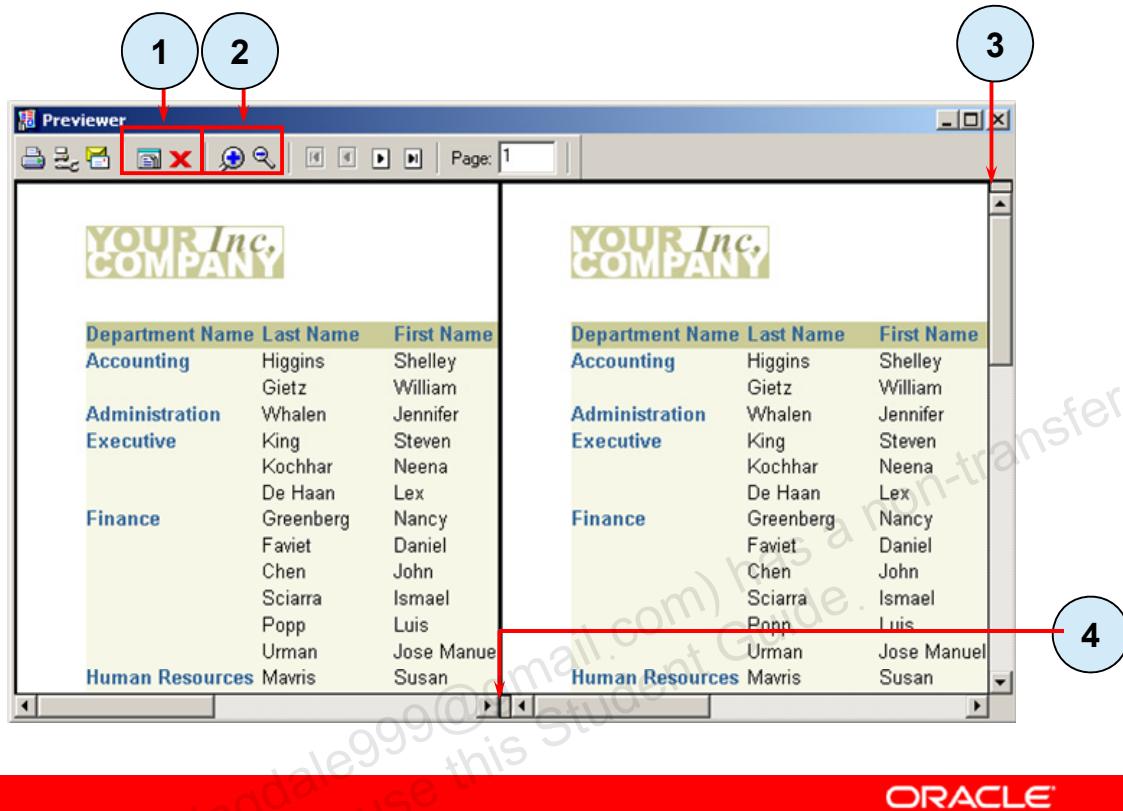
You can navigate through the Previewer using the scrollbars and the iconic buttons in the toolbar at the top of the Previewer window. The toolbar includes buttons to perform the following actions:

1	Print, Page Setup, and Mail options
2	Page options: First, Previous, Next, Last
3	Go to specific page number

Button	Description
First Page	Go to first page
Previous Page	Go to previous page
Next Page	Go to next page
Last Page	Go to last page

Button	Description
Page	Go to the page number that you enter in the field
Print	Print the report
Mail	Send report to a SMTP-compliant mail system

# Print Preview



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

**ORACLE**

## Previewing Reports (continued)

To view the contents of more than one page at the same time, click New Previewer to open another Previewer window. Close each Previewer window individually.

When viewing large report pages, you can split the Previewer window either horizontally or vertically to show the extreme left and right or top and bottom portions of a page.

- **Horizontal split:** Click and drag the gray rectangle above the vertical scrollbar.
- **Vertical split:** Click and drag the gray rectangle to the left of the horizontal scrollbar.

Release the mouse button at the position at which you want to split the viewing region.

**Note:** To remove the split, drag the rectangle back to its original position.

1	New Previewer, Close Previewer options
2	Zoom in, zoom out tools
3	Drag this rectangle to split Previewer horizontally
4	Drag this rectangle to split Previewer vertically

## **Previewing Reports (continued)**

Zoom in and zoom out buttons are available on the toolbar. Select the large plus sign to zoom in and the small minus sign to zoom out.

## Supported File Types

- RDF
- REP
- JSP
- HTML
- XML



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Supported File Types

As you will learn in later lessons, report definitions created with Oracle Reports can be saved in a variety of formats and then deployed with Oracle Application Server.

File Type	Description
RDF	Report Definition File: binary file containing source code
REP	Report: binary file without source code
JSP	JavaServer Page format
HTML	HyperText Markup Language
XML	Extensible Markup Language

## **Supported File Types (continued)**

Reports definitions RDF and REP are “owned” by Oracle Reports. In other words, you will need to use Oracle Reports to modify the definition of these reports. Report definitions JSP, HTML, and XML are saved and can be retrieved in a textual format. This gives you the flexibility to use third party text editors and HTML tools to modify the module definition.

### **Technical Note**

Another file type REX is supported in Oracle Reports for backward compatibility. A REX file contains a report definition stored in text (e.g., ASCII or EBCDIC) format. A REX file is not executable and cannot be directly modified in Reports Builder. You can convert a .rex file to a .rdf by selecting Tools > File Conversion from the menu.

# Summary

In this lesson, you should have learned how to:

- Identify key questions to help understand user requirements
- List the simple report styles and describe their underlying structures
- Identify key design considerations:
  - Report style
  - Database access
  - Common code
  - Common objects
- List the various options for running a report



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Summary

### Designing Reports

Make sure you fully understand and document the user requirements, and then pick a report style and structure that maps to the need.

By discussing the report styles and structures, you should now be able to:

- Appreciate the principle of groups and frequencies of data
- Describe simple report styles and their underlying structures
- Realize the importance of database access and report performance, and consider this for every report you design
- Consider usage of common code and objects at the very early stages of development

### Executing Reports

You have learned that there are a number of different ways to run a report, including a Web browser, an OracleAS Portal application, and the command line. You can preview a report in Reports Builder using the Run Web Layout, Run Paper Layout, and Print Preview options.

## Practice 2 Overview

Executing existing reports



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Practice Session Overview: Lesson 2

This practice session contains:

- Previewing a report in a browser
- Executing a number of different reports. For each report, answer some questions

This practice session consists of a combination of practical and paper-based questions.

## Practice Session: Lesson 2

1. Invoke the Reports Builder executable.
3. Open report p2q3.rdf and run the paper layout.
  - a. In the Paper Design view, move to the next page. Notice that this report has a header page with text.
  - b. What style of report is this?
  - c. Close the Paper Design view. From the File menu, select Print Preview. In the Previewer window, experiment with the horizontal and vertical split screen features.
  - d. Close the Previewer window.
4. Open report p2q4 . rdf and run the Web layout .
  - a. What style of report is this?
  - b. How many groups of data are in this report?
  - c. How many pages are there in this report?
  - d. Close the browser.
5. Open report p2q6 . rdf and run the paper layout .
  - a. Notice the Parameter Form. You are requested to enter a customer ID. The report will display order information based on your input.
  - b. Specify a valid customer ID and run the report. Valid values include 101, 118, 148 and 170; pick one of these and then select Run Report.
  - c. What do you think would be a better method of providing valid values to a user?
  - d. Close the Paper Design view.
  - e. Run the paper layout again. This time do not specify a customer ID.
    - i. What is the result?
    - ii.In a production situation, what should happen in this instance?
  - f. Close the Paper Design view.

## **Practice Session: Lesson 2 (continued)**

- g. Run the paper layout again. This time enter Customer ID 999.
    - i. What is the result?
    - ii. In a production situation, what should happen in this instance?
  - h. Close the Paper Design view.
7. Open report p2q7.rdf and run the paper layout.
- a. In the Parameter Form, notice the list of values for the customer information. The report displays order information based on your input.
  - b. Select a valid customer name and run the report.
  - c. Close the Paper Design view.

## Working in Oracle Reports Developer

3

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Objectives

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

- Describe the main Oracle Reports executables
- Describe the main components of Reports Builder
- Describe the main objects in a report



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Overview

This lesson describes the Oracle Reports executables and gives an overview of Reports Builder, including a high-level description of its components and object hierarchy.

# Reports Developer Executables

## Developer

**Reports Builder**  
**rwbuilder**

**Reports Runtime**  
**rwrn**

**Reports Converter**  
**rwcvt**

## Reports Services

**Reports Server**  
**rwsrvr**

**Reports Client**  
**rwlnt**

**Queue Manager**  
**rwrqm**

**Reports Servlet**  
**rwsrvlt**

**Reports CGI**  
**rwcgi**

**ORACLE®**

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Reports Developer Executables

Reports Developer executable filenames are lower case.

In Windows, all Oracle Reports executables follow a similar pattern: `rw<>.exe`.

In UNIX, the executables follow a pattern of `rw<>`.

## Reports Developer Executables (continued)

The main Reports Developer executables are:

NT Filename	Executable Title	Description
rwbuilder	Reports Builder	Create, develop, and maintain report definitions.
rwrn	Reports Runtime	Run-time environment for testing.
rwconvert er	Reports Converter	Converts a report definition to alternate storage formats.
rwserver	Reports Server	Install/invoke a Reports server.
rwclient	Reports Client	Parses and transfers a command line to the specified (or default) Reports Server.
Rwrgm	Reports Queue Manager	View/schedule reports on a remote server.
Rwservlet	Reports Servlet	Runs a report as a servlet, translating and delivering information between HTTP and the Reports Server.
Rwcgi	Reports CGI	Provides a connection between a Web server and Reports Services, enabling you to run reports dynamically from your browser. Reports CGI is maintained only for backward compatibility.

### Technical Note

A *servlet* is a Java application that runs in a Web server or application server and provides server-side processing, typically to access a database or perform e-commerce processing. Servlets provide an alternative to CGI (Common Gateway Interface) scripts. Because they are written in Java, servlets are portable between servers and operating systems. Servlets are also more efficient than CGI scripts as they can remain running inside the servlet engine of the Web listener, waiting for new requests, instead of being shut down once a request is processed, and then restarted when a new request is issued.

# Invoking Reports Builder



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Invoking Reports Builder

In this lesson, you invoke Reports Builder, the `rwbuilder` executable, and look at the definitions of some reports to describe the components and objects in the builder.

When you invoke Reports Builder, the initial Welcome dialog box contains a check box “Display at startup” that you can clear if you want to suppress this dialog box.

If you clear this option, and then later want to see the Welcome dialog box, you must change the relevant option in the Preferences dialog box.

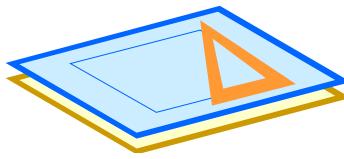
To display the Welcome dialog box:

1. Select Edit > Preferences.
2. Select the Wizards tab.
3. Select the Welcome Dialog check box.

# Reports Builder Modules



**Report**



**Template**



**PL/SQL Library**

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

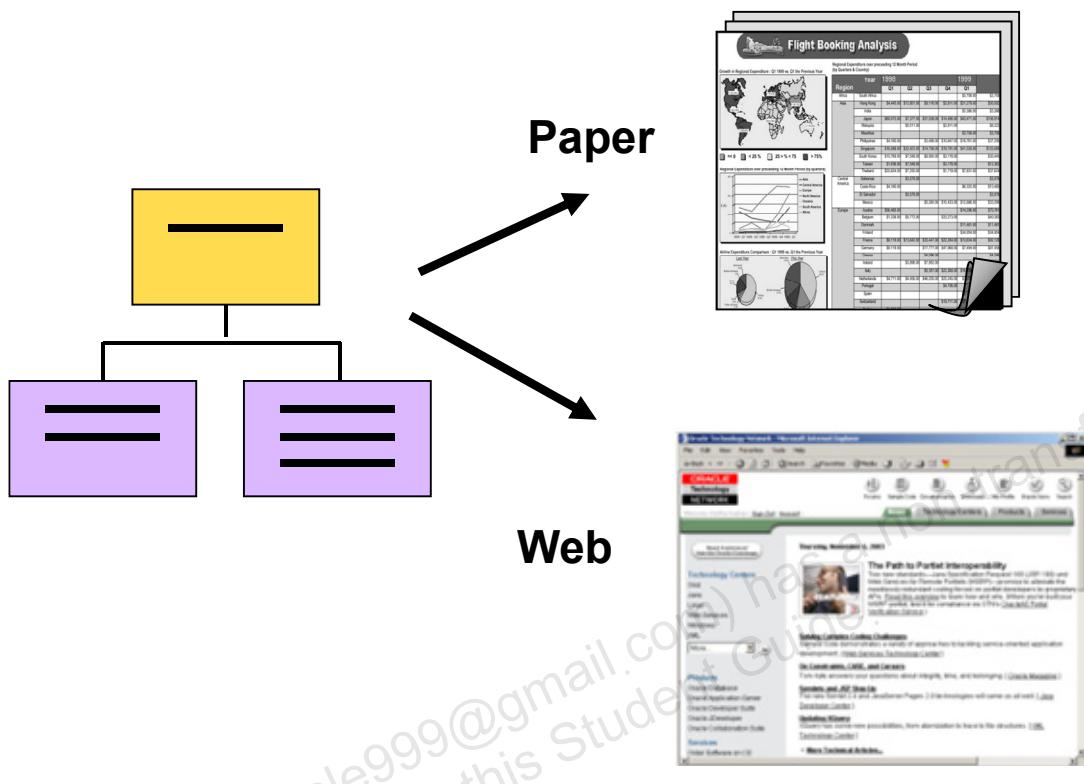
## Reports Builder

The Reports Builder interface enables you to create a number of different types of modules, and it provides a Report Editor in which you can view the structure and objects in a report module.

The Reports Builder module types are:

Module Type	Description
Report	A report definition
Template	A skeleton definition containing common style and standards; can include graphics; provides a standard format to enable quick and easy development of professional standard look-and-feel reports
PL/SQL Library	A stand-alone library containing PL/SQL program units—procedures, functions, packages—that can be called from multiple reports

# Report Data and Layout



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

ORACLE

## Report Data and Layout

A report definition defines two main parts of a report and brings them together in the output.

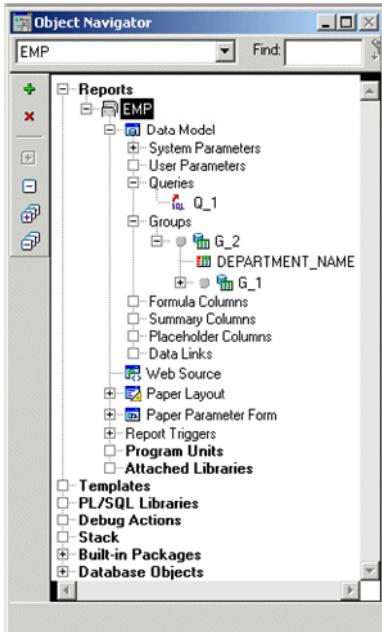
- Data: Data structure and data to be displayed
- Layout: Formatting information about how the data appears in the output

Each report module can have a data model, a paper layout, and a Web layout. The data model, as well as program units, can be shared by the paper and Web layouts.

A report can consist of:

- A data model and a paper layout.
- A data model and a Web layout.
- A data model, a paper layout, and a Web layout.

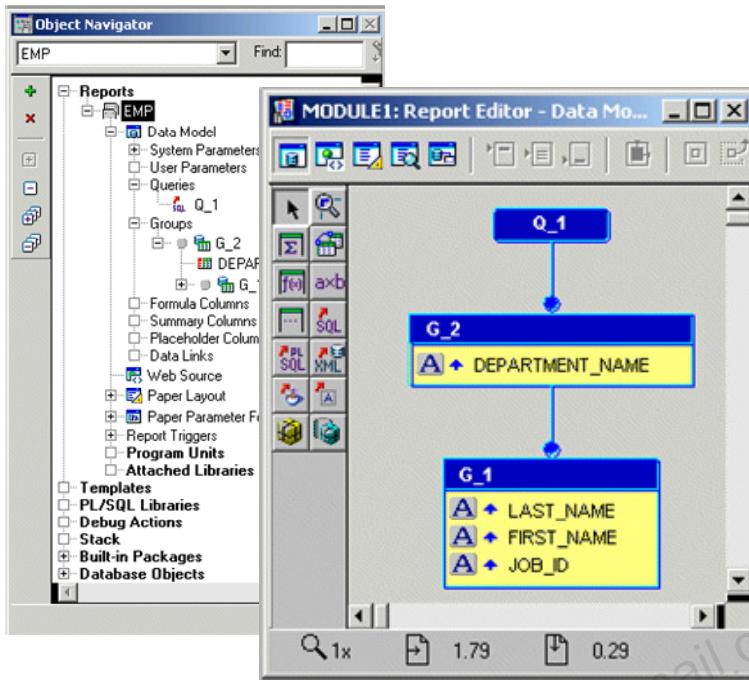
# Reports Builder Components



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

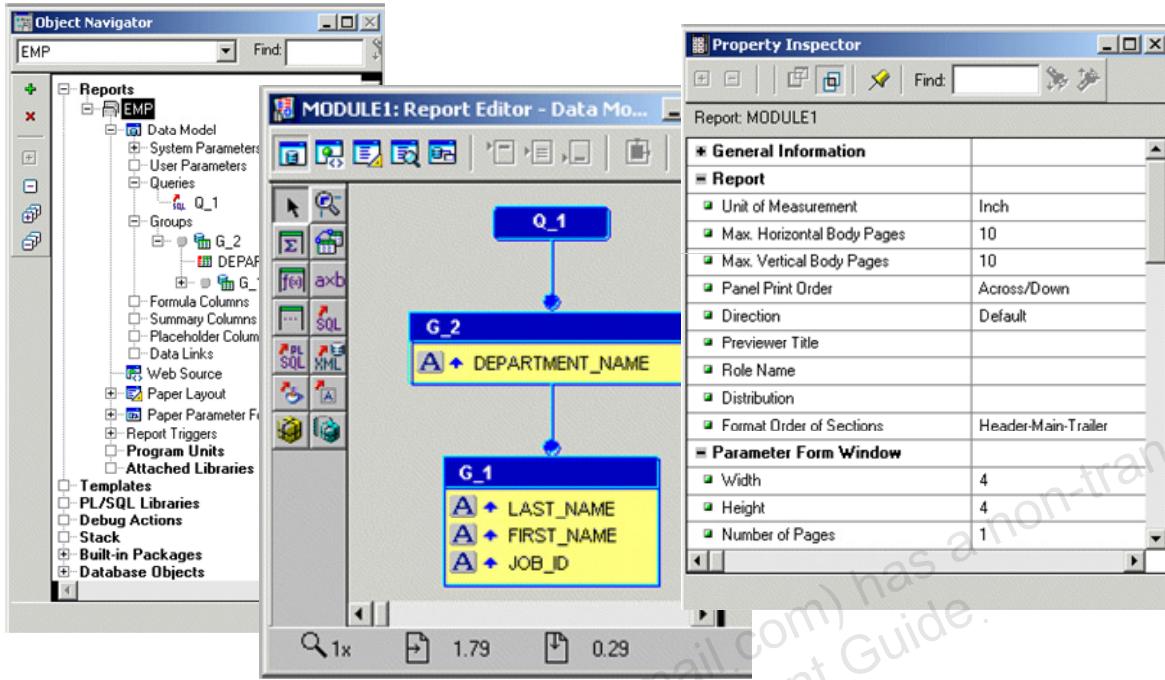
# Reports Builder Components



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

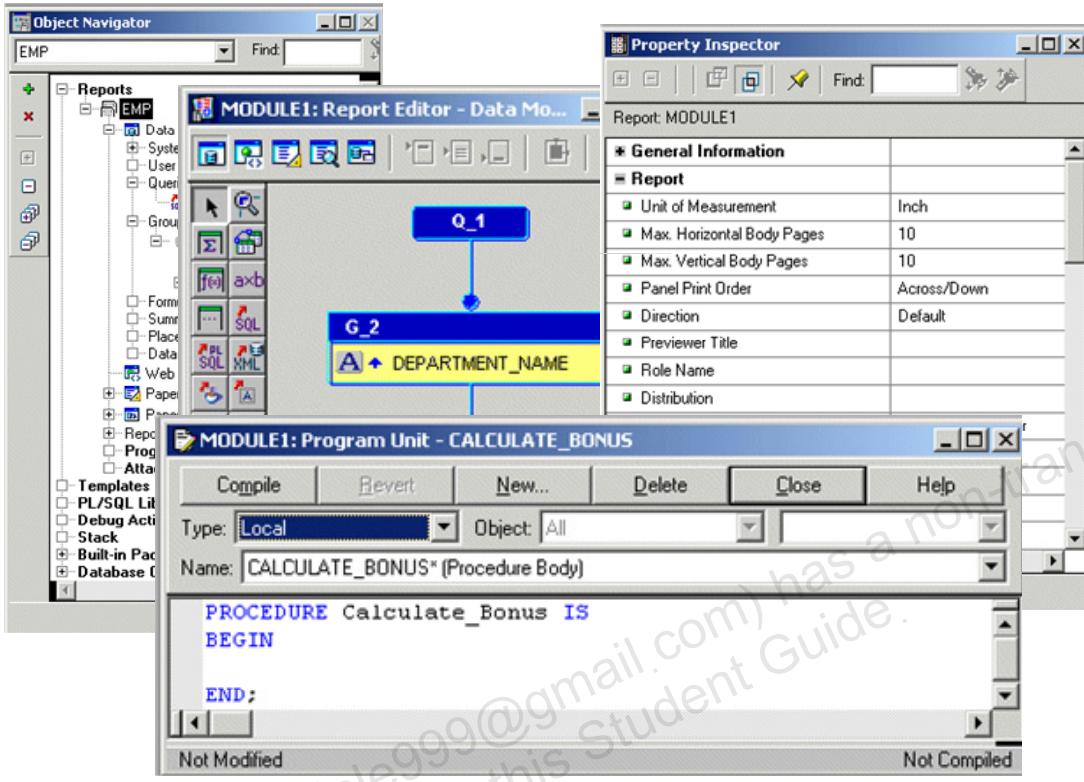
# Reports Builder Components



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Reports Builder Components



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Reports Builder Components

### Object Navigator

The Object Navigator is a hierarchical browsing and editing interface that enables you to locate and manipulate application objects quickly and easily. Features include:

- A hierarchy represented by indentation and expandable nodes (Top-level nodes show module types, database objects, and built-in packages.)
- A find field and icons, enabling forward and backward searches for any level of node or for an individual item in a node
- Icons in the horizontal toolbar replicating common File menu functions

### Report Editor

The Report Editor contains different views to help you handle the data objects and layout objects for Web and paper reports. These views will be introduced later in this lesson.

## **Reports Builder Components (continued)**

### **Property Inspector**

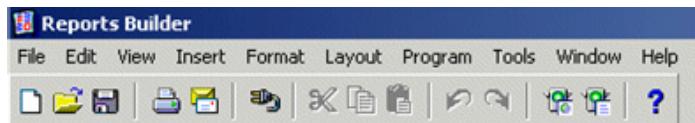
All objects in a module, including the module itself, have properties that you can see and modify in the Property Inspector. Features include:

- Expandable and collapsible nodes
- In-place property editing
- Search features
- Multiple selection of objects
- Complex property dialogs
- Ability to invoke multiple instances of the Property Inspector

### **PL/SQL Editor**

The PL/SQL Editor enables you to create and compile program units such as procedures, functions, and packages within the current report.

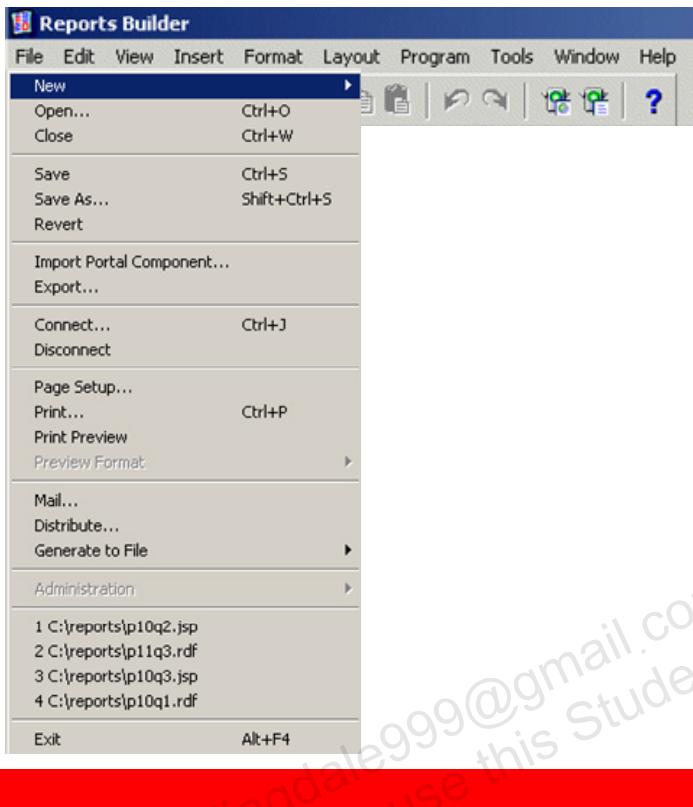
## Main Menu Structure



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

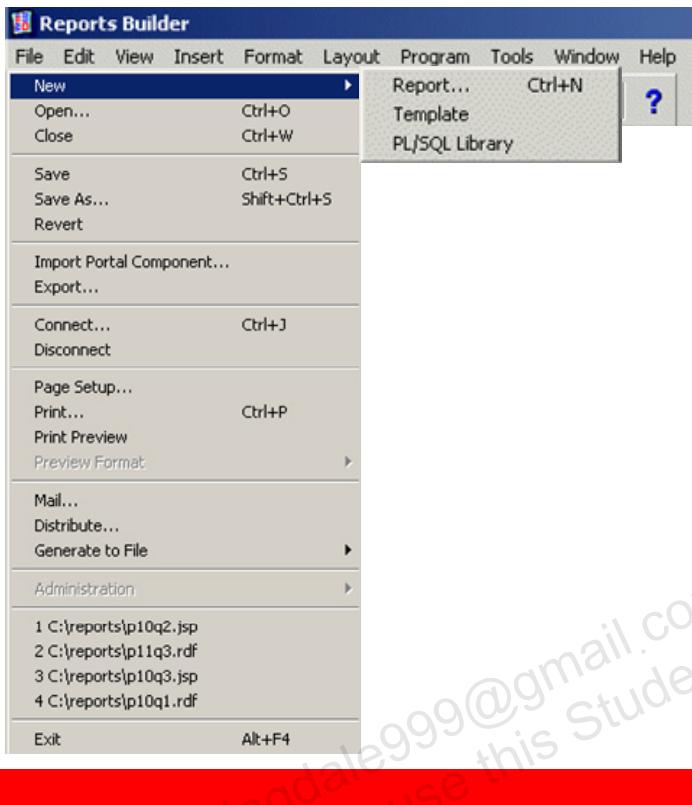
## Main Menu Structure



ORACLE

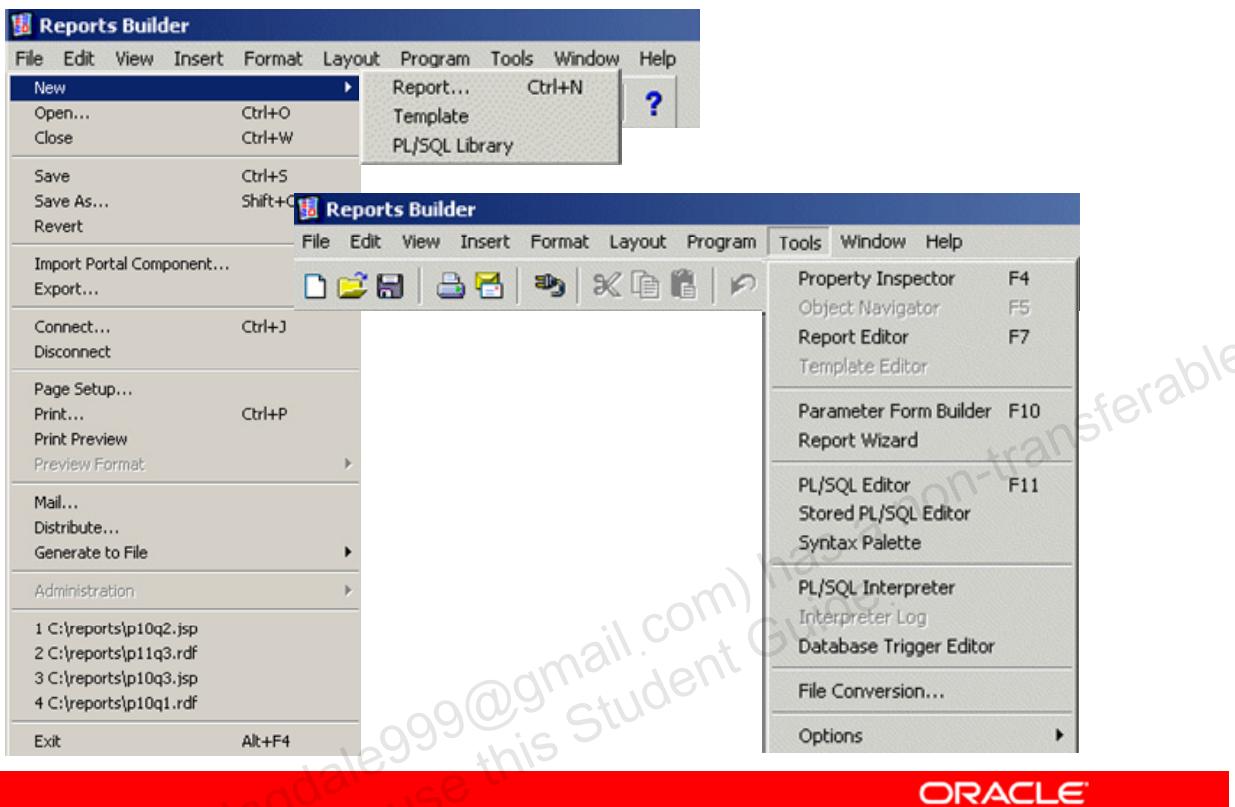
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Main Menu Structure



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Main Menu Structure



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Navigating Around the Oracle Reports Main Menu

The main menu contains options to enable you to create, modify, and manage your Reports Builder modules.

The following table describes some common features in GUI menus.

Feature	Description
Underline	Shortcut key: [Alt]+letter
Ellipsis (...)	Additional input, usually by dialog box
>	Menu option has a submenu
Windows menu	List of open windows; choose any window to make it active
Help	List of Help facilities, such as online Help text and Quick Tour

## Navigating Around the Oracle Reports Main Menu (continued)

The main menu options in Reports Builder are:

Menu Item	Description
File	Common file utilities, such as open, save, connect, administration
Edit	Cut, copy, paste, other editing functions; session preferences
View	Switch view in current window; options vary greatly depending on context
Insert	Add dynamic data to a static HTML page; add fields and layout objects to paper reports
Format	Change the style and appearance of objects in the Paper Layout view
Layout	Arrange and reshape objects in the Paper Layout view
Program	Includes compilation, run options for Web and paper, and the Java Importer
Tools	Includes wizards and access to PL/SQL editors

**Note:** Some menu items are selectable depending on the current context. For example, the items in the Layout menu are selectable only when the context is the Paper Layout view.

# Wizards

- Report Wizard
- Data Wizard
- Graph Wizard
- Report Block Wizard

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Wizards in Reports Builder

Wizards provide an easy step-by-step interface for commonly performed tasks. The wizards in Reports Builder are:

- **Report Wizard:** The Report Wizard guides you through the steps to create a basic paper report. Each page of the wizard asks you for information to help you create your initial report.
- **Data Wizard:** This wizard helps you quickly define or modify a query for a multiquery data models.
- **Graph Wizard:** You can add a variety of charts and graphs, including true 3-dimensional graphs, to a report using the Graph Wizard. Charting is implemented in Reports Builder with the Oracle BI graph bean.
- **Report Block Wizard:** This wizard enables you to quickly create a JSP report by embedding report data into a Web page using Reports custom JSP tags.

# Report Editor

- Data Model
- Web Source
- Paper Layout
- Paper Design
- Paper Parameter Form



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## The Report Editor

The Report Editor contains views to handle the data objects and layout objects separately:

View	Description
Data Model	Displays a structural representation of the data in a report. The objects do not appear in the report output, but the structure determines the layout style, and the data objects provide the values that appear in the layout objects.
Web Source	Displays the HTML / JSP source for a report. You can use this view to add dynamic content to a Web page using the Report Block Wizard and the Graph Wizard. You can also edit the Web source directly in this view.
Paper Layout	Displays the layout objects in a paper report and allows you to make many modifications to any layout object. All layout objects have properties that you can modify using the Property Inspector. The hierarchy of the layout objects is determined by the Data Model.

## The Report Editor (continued)

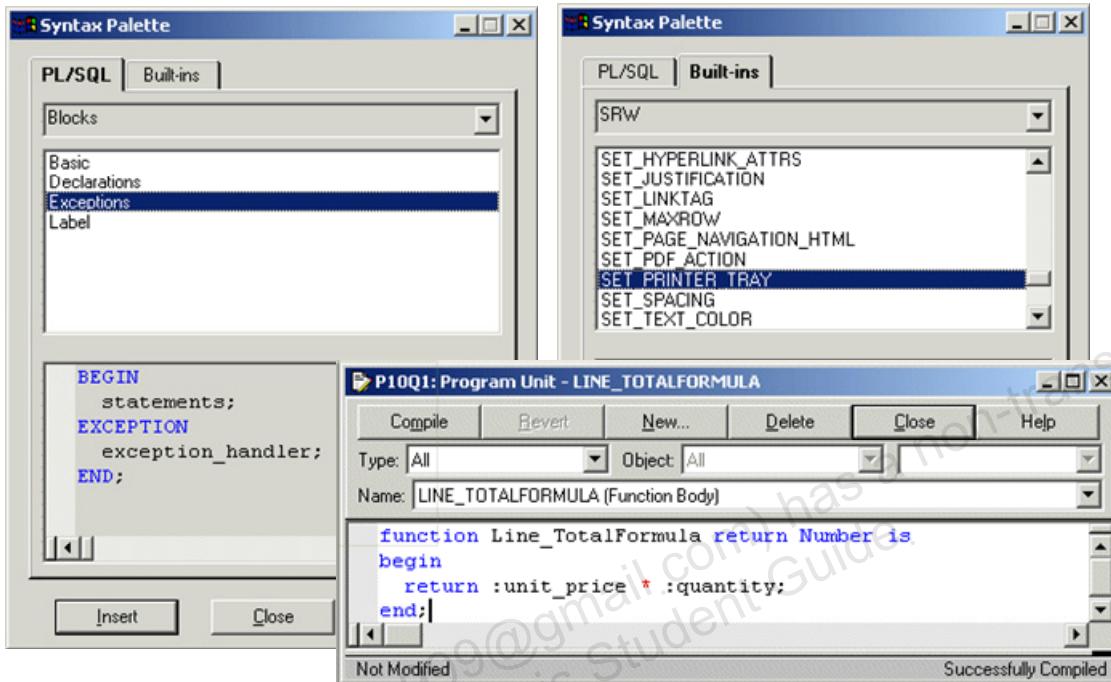
View	Description
Paper Design	Displays output for paper reports and allows you to make many commonly required, simple modifications to the layout, such as spacing, formatting fields, color, and editing text, without having to open the Paper Layout view.
Paper Parameter Form	Displays the layout of the Paper Parameter Form that, at run time, allows user input of parameter values.

You can create many fully functional paper reports simply by using the Wizard and modifying the report in the Paper Design view. However, this course also teaches you in later lessons how to use the Data Model, Paper Layout, and Paper Parameter Form so that you can create more complex paper reports. You will also learn how to use the Web Source view to add dynamic content to HTML pages to create reports for Web publishing.

### Technical Note

Oracle Reports is integrated with Oracle JDeveloper to enable Java developers to leverage the powerful publishing capabilities of Oracle Reports within their Java applications. You can create a new JSP-based Web report or a Pluggable Destination from within Oracle JDeveloper. You can also debug a Reports JSP from within JDeveloper. For more information, see the Oracle Technology Network (<http://otn.oracle.com>).

# PL/SQL Development Environment: Syntax Palette



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

ORACLE

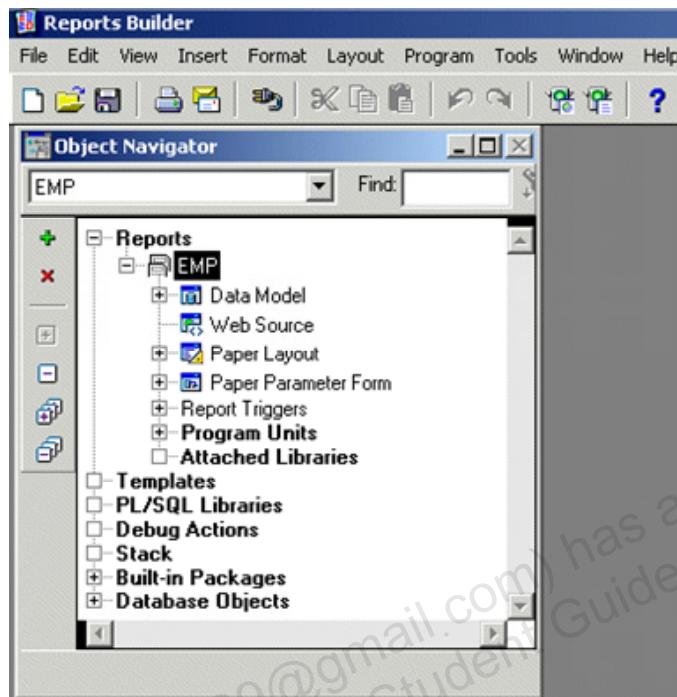
## PL/SQL Development Environment

The PL/SQL development environment is the integrated functionality of Procedure Builder with Reports Builder. It provides:

- Development of server-side database triggers and stored procedures, functions, and packages
- Development of libraries to hold PL/SQL program units
- Statement-level debugging of PL/SQL at run time

The Syntax Palette is a programming tool that enables you to display and copy the constructs of PL/SQL language elements and built-in packages into the PL/SQL editor.

# Object Navigator



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Object Categories

The Report Module consists of many objects that fall into the following four categories:

- Report level
- Data Model
- Paper Layout
- Paper Parameter Form

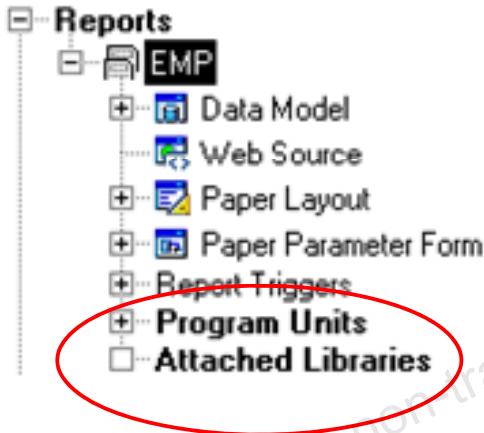
The hierarchy of object categories can be viewed in the Object Navigator.

This section gives an overview of the objects in each category.

**Note:** All the objects mentioned in this section are discussed in greater detail in later lessons.

# Report-Level Objects

- Properties
- Triggers
- PL/SQL Program Units
- Attached Libraries



**ORACLE**

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Object Categories (continued)

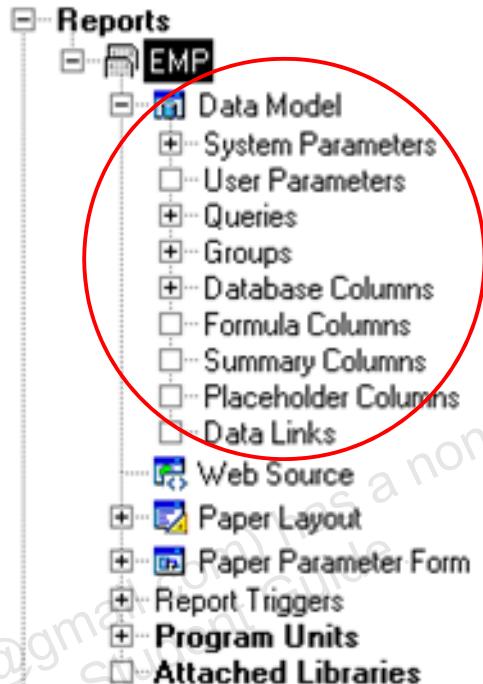
### Report-Level Objects

The report objects define the measurements, dimensions, triggers, and PL/SQL program units of a report. The report object itself consists of the following:

Object	Description
Properties	Define page dimensions and Previewer settings, for example
Triggers	Allow PL/SQL to be executed at different stages of the report execution
PL/SQL Program Units	Contain functions and procedures that can be called from report-level objects in the same report
Attached Libraries	External PL/SQL library file that contains sets of PL/SQL program units that are independent of a report definition

# Data Model Objects

- Parameters
- Queries
- Groups
- Columns
- Data Links



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Object Categories (continued)

### Data Model Objects

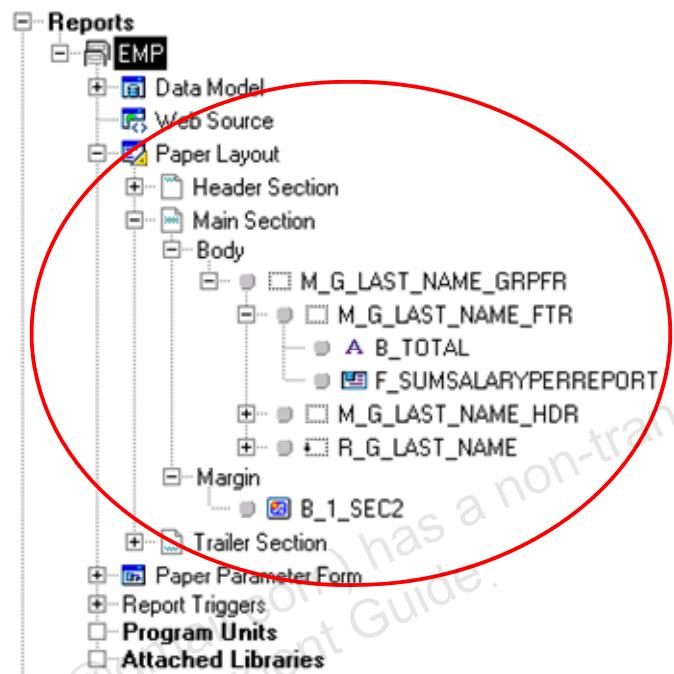
The Data Model objects for a report define the data used in the report and the data structure. Data Model objects appear in the Data Model view of the Report Editor and the Object Navigator. Data Model objects can be of the following types:

Object	Description
Parameters	Provide for run-time defaults or user input; system parameters exist by default; you can also create user parameters
Queries	Select the data for your report
Groups	Organize the data to form the required hierarchical structure
Columns	Contain individual data values; database columns exist by default and contain data from the database columns or expressions defined in the query; you can also create Formula, Summary, and Placeholder column types
Data Links	Join queries for complex data relationships

With the exception of parameters, you create all objects in the relevant editor, not in the Object Navigator. Parameters do not appear in the editor. You create parameters in the Object Navigator and modify them in the Property Inspector.

# Paper Layout Objects

- Frames
- Repeating frames
- Fields
- Boilerplate



**ORACLE®**

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Object Categories (continued)

### Paper Layout Objects

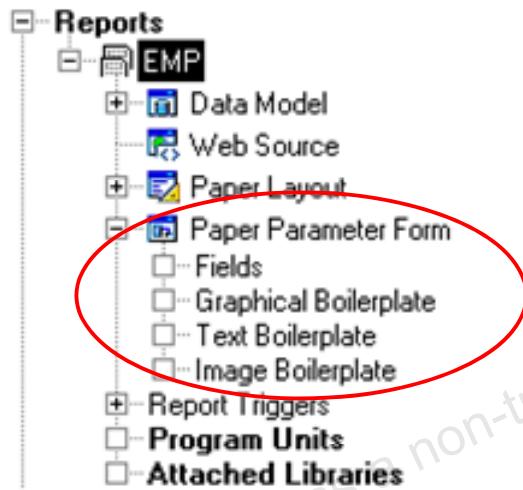
The Paper Layout objects define the format of the report including the positioning and appearance of data, text, and graphics. The main layout objects fall into the following categories:

Object	Description
Repeating frame	Contains other objects and prints once for each record of the associated group
Frame	Contains other objects and prints only once
Field	Contains data and other variable values and their formats
Boilerplate	Contains text or graphics that may appear anywhere in the report

These and other layout objects are discussed later in the course.

# Paper Parameter Form Objects

- Fields
- Boilerplate



**ORACLE®**

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Object Categories (continued)

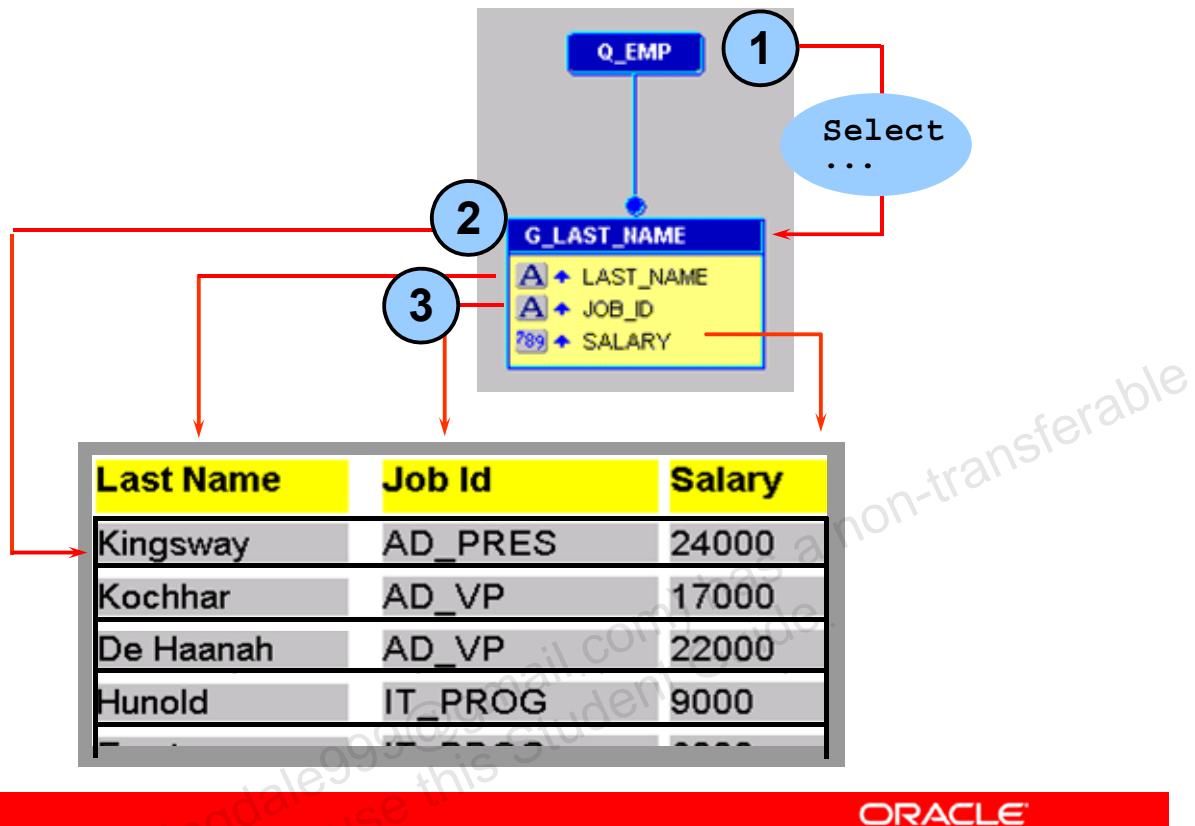
### Paper Parameter Form Objects

The Paper Parameter Form objects define the appearance of the run-time parameter form. You create and modify run-time paper parameter form objects.

Object	Description
Field	Contains parameter values
Boilerplate	Contains constant text or graphics that appear on the run-time paper parameter form

**Note:** The Paper Parameter Form controls the layout of the run-time paper parameter form. The objects are similar to paper layout objects. The source of a parameter field comes from a parameter that is a Data Model object. Parameters appear in the Object Navigator, not in the Data Model view.

# Object Interrelationship



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Object Interrelationship

Now that you know the different categories of objects, it is also important to understand the relationship between these objects. The diagram above shows the relationships between some of the most common objects, explained in terms of a simple tabular report.

### Data Model Objects

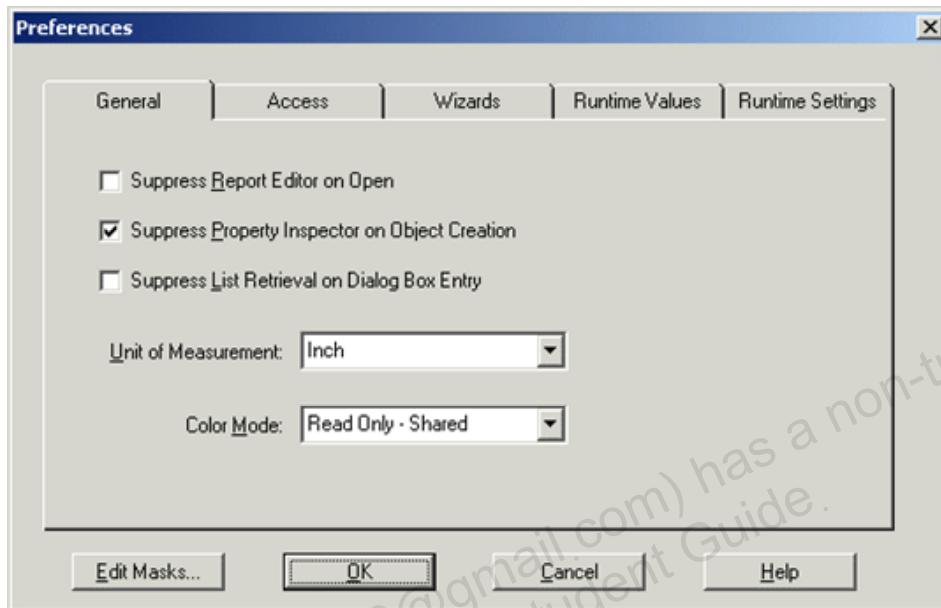
- A *query* fetches records from the data source.
- Each fetched record becomes an instance of the related *group*.
- Each data source value is fetched into the related *column*.

### Paper Layout Objects

- A *column* provides the value that is displayed in one or more layout *fields*.
- A *field* must display all instances of its related column value; therefore, each record instance of a group is represented by a *repeating frame*.

1	Each query fetches data records and structures them in the group hierarchy.
2	Each group is the source of a repeating frame.
3	Each column is a source of a field.

# Customizing Your Oracle Reports Developer Session



ORACLE®

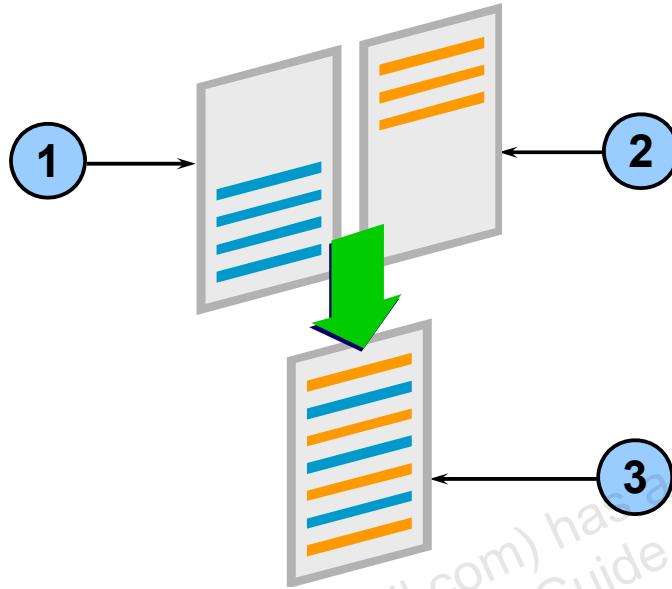
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Customizing Your Oracle Reports Session

Reports Builder Preferences allow you to customize some aspects of your Reports Builder session.

To access the Reports Preferences dialog box, select Edit > Preferences from the menu.

# Saving Preferences



**ORACLE®**

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Saving Preferences

There are four tab pages in the Reports Preferences dialog box. Press the Help key ([F1] in Windows) in the Preferences dialog box to see a description of each preference.

As well as session preferences, Reports Builder preferences enable you to set run-time options when running your report within the builder. The following table describes a few example preferences. Others are introduced when applicable throughout the course.

Tab	Preference Name	Description
General	Suppress Report Editor on Open	Do not display the Report Editor when opening a report. This saves you time when opening several reports to make changes in the Object Navigator.
	Unit of Measurement	Set the unit of measurement that you want to use for new reports that you create. Altering this setting does not affect existing report definitions.
Wizards	Welcome Dialog	Check box to suppress or display the first Welcome dialog box. There are several similar check boxes.

Your preferences are maintained in the file `cauprefs.ora`, located in the `<oracle_home>` directory.

# Oracle Reports Environment Variables

- REPORTS\_PATH
- REPORTS\_TMP
- REPORTS\_RESOURCE
- ORACLE\_PATH
- REPORTS\_CLASSPATH

**Windows: Modify in Registry**



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Oracle Reports Environment Variables

Oracle Reports Developer uses many environment variables. All necessary environment variables used to run reports are initialized for you by the installer. You can modify these variables in your own environment and for different applications.

### Setting Pathnames

Oracle Reports uses some environment variables to search for files at run time. This enables you to build applications that are portable across platforms and directory structures by avoiding hard-coded paths in file references in a report.

Variable	Description
REPORTS_PATH	A path that Reports searches for files at run time
REPORTS_TMP	A path that will be used to create temporary files
REPORTS_RESOURCE	A path that contains the location of the Reports resource files, such as icon files
REPORTS_CLASSPATH	A path that Reports searches to locate Java objects

## **Oracle Reports Environment Variables (continued)**

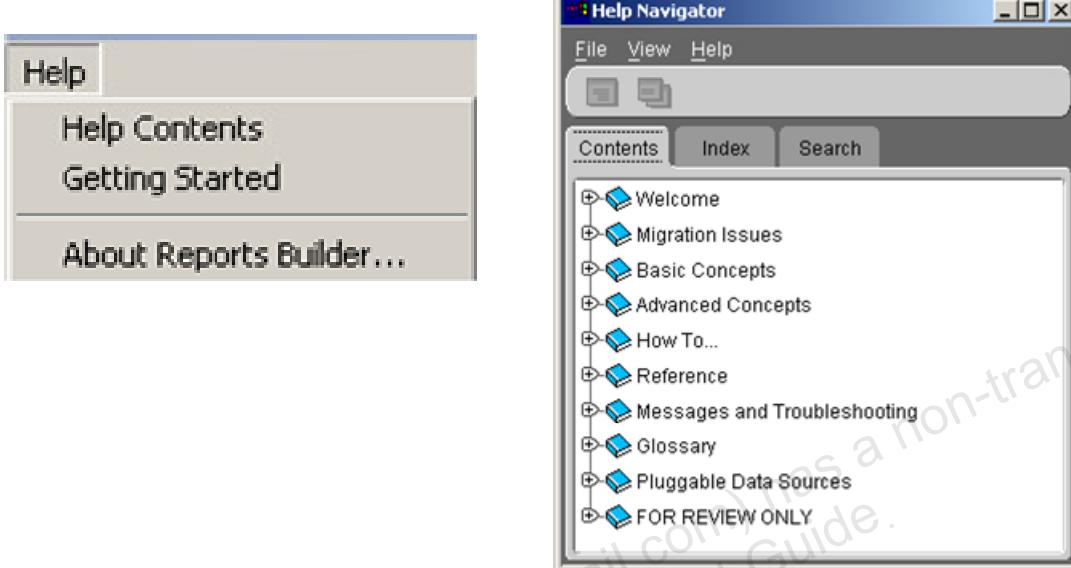
### **Generic Oracle Path**

ORACLE\_PATH is an additional path that all Oracle Developer Suite components search if they cannot find a file in their own specific path.

### **Modifying Environment Variables**

In a Windows 32-bit environment, use the Windows Registry to modify these paths. Registry path: HKEY\_LOCAL\_MACHINE/SOFTWARE/ORACLE.

# Using the Online Help System



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Using the Online Help System

The table describes the Help menu options in Reports Builder.

Help Menu Option	Description
Help Topics	This is the contents page for comprehensive online Help; includes Index and Find tabs. The Help key ([F1] for Windows) displays context-sensitive online Help at any place in the builder.
Getting Started	This option will navigate you to the Oracle Reports home page on the Oracle Technology Network ( <a href="http://otn.oracle.com/products/reports/content.html">http://otn.oracle.com/products/reports/content.html</a> ). From here you can access <i>Getting Started with Oracle Reports</i> , a self-paced tutorial on this release.
About Reports Builder	Appears as two panels: The upper panel shows a server-side connection when you are connected to a database server. If there is no connection, this panel is blank. The lower panel shows client-side components and their version numbers.

## **Technical Note**

In Oracle Reports Builder Developer 10g, the ‘Quick Tour’ option in the Help menu that was available in earlier releases such as Oracle Reports Builder 9.0.2.0.1, has been changed to ‘Getting Started’.

The Oracle Reports online Help system is also accessible through the Oracle Technology Network (<http://otn.oracle.com>).

# Summary

In this lesson, you should have learned how to:

- Describe the Oracle Reports executables
- List the types of modules you can create in Reports Builder
- Describe the views of the Report Editor
- Describe the main object categories in a report module



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Summary

Oracle Reports Developer provides a number of executables for developing and deploying reports, including Reports Builder.

The Reports Builder interface allows you to develop three types of modules, including report definitions.

Reports Builder provides a Report Editor in which you can view and modify the objects that the wizard creates. You can also create your own objects to enhance your report structure and layout.

The Report Editor enables you to switch views, depending on the objects that you want to modify. For Web reports, use the Web Source view.

Objects in a Report module fall into four categories: Report, Data Model, Paper Layout, and Paper Parameter Form.

## Practice 3 Overview

- Invoking Reports Builder
- Opening an existing report
- Switching views in the Report Editor
- Accessing the Help system



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Practice Session Overview: Lesson 3

This practice session contains:

- Invoking Reports Builder
- Opening an existing report
- Switching views in the Report Editor
- Accessing the Help system

The questions in the practice session provide an introduction to the Reports Builder interface. You open and navigate through an existing report definition and use the Help system to obtain information about some of the Reports executables.

## Practice Session: Lesson 3

1. Start Reports Builder and open the existing report: p3q1.rdf. Run the paper layout.
2. Using the same report, display the Data Model view of the Report Editor.

In the Object Navigator, select Q\_1.

**Hint:** Move the Data Model window to the right so that you can also see the Object Navigator. Use the Find field at the top of the Object Navigator to locate Q\_1.

Notice the object that is selected in the Data Model view.

3. Using the same report, switch to the Paper Layout view of the Report Editor.

In the Object Navigator, select F\_CUSTOMER\_ID.

**Hint:** Use the Find field at the top of the Object Navigator.

Notice the object that is selected in the Paper Layout view.

Fully expand the Paper Layout node and select R\_G\_ORD\_ID.

Notice the object that is selected in the Paper Layout view.

4. Using the same report, run the Web layout.

5. Use Help Contents to answer the following questions:

- a. What is RWSERVLET?
- b. What is RWCLIENT?
- c. What is the Web Source view?

## Creating a Paper Report



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Objectives

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

- Create a simple tabular paper report using the Report Wizard
- Describe the methods of building the report query
- Summarize report values
- Modify the style and content of a report
- Create other report styles available in the Report Wizard
- Preview a paper report on the Web



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

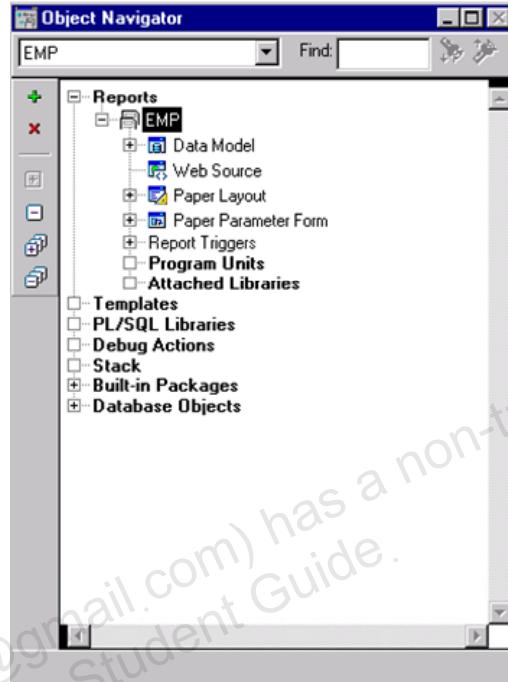
## Overview

In this lesson, you learn how to create a simple tabular paper report and modify the report by adding data and altering the report style. This lesson covers how to create other styles of reports that are available from the Report Wizard. You will also learn how to preview a paper report on the Web.

Oracle Reports enables you to easily model, design, and publish high fidelity Web reports. You will learn about this in a later lesson.

# Report Module Components

- Data Model
- Web Source
- Paper Layout
- Paper Parameter Form
- Program Units



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Report Module Components

Each report definition consists of a data model, Web source, paper layout, program units, and paper parameter form, regardless of the storage format for the report. The same data model and program unit objects can be shared by a paper based layout and a Web based layout. In essence, you only have to define the actual report once because the same data model and business logic used for paper publishing can also be used for Web publishing. You will develop Web reports later in the course.

This lesson focuses on defining a simple data model and creating a paper layout.

# Building a Paper Report

You have two options:

- Use Reports Builder
  - Wizards
  - Paper Layout
  - Paper Design
- Define the report in XML



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Building a Paper Report

In Oracle Reports, you have two options for building a paper report. The simplest method is to use the wizards and editors in Reports Builder. Designed for the paper or Web layout, the Report Wizard guides you through the steps to create a basic report.

As an alternative, you can define the data model and/or layout for your paper report in XML.

In this course, you will build paper reports using the Report Wizard.

# Invoking the Report Wizard



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Invoking Reports Builder and the Report Wizard

When you invoke Reports Builder, the Welcome dialog box gives you the option of using the Wizard to build a new report. The Report Wizard provides an easy step-by-step interface to create a new report.

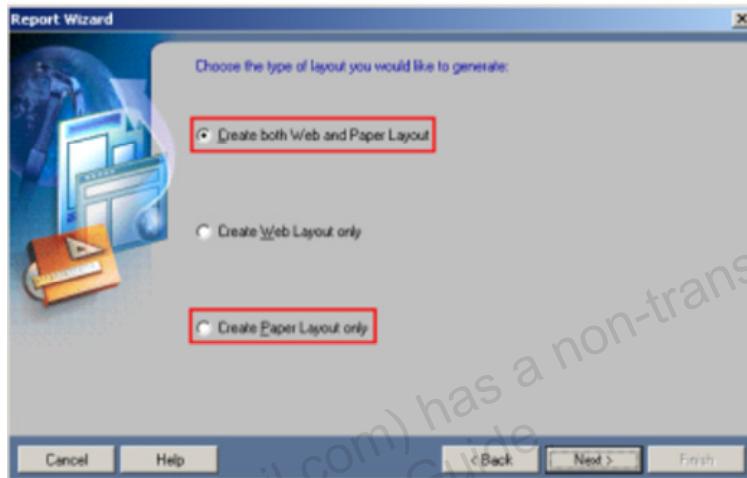
The Report Wizard opens with a Welcome page. To suppress this page, clear the “Display at startup” check box. You can reinstate this page in the same way as the Welcome dialog box in Reports Builder; select the Wizard tab in the Preferences dialog box and then select Report Wizard Welcome Page.

Each page of the Report Wizard asks you for information to help you create your initial report. Step through the wizard pages, selecting Next and Back, until you are satisfied with the initial information that you have entered. On the last page, select Finish.

# Choosing the Layout Type

## Wizard Pages

- Report Style
- Data Source Type
- Data Source Definition
- Fields
- Totals
- Labels
- Template



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Creating a Tabular Report Using the Wizard

You can use the Report Wizard to build eight styles of reports. In this section, you step through the wizard pages to create a tabular report.

### Report Layout

On the first page you specify the type of layout you want the Wizard to generate. Your options are:

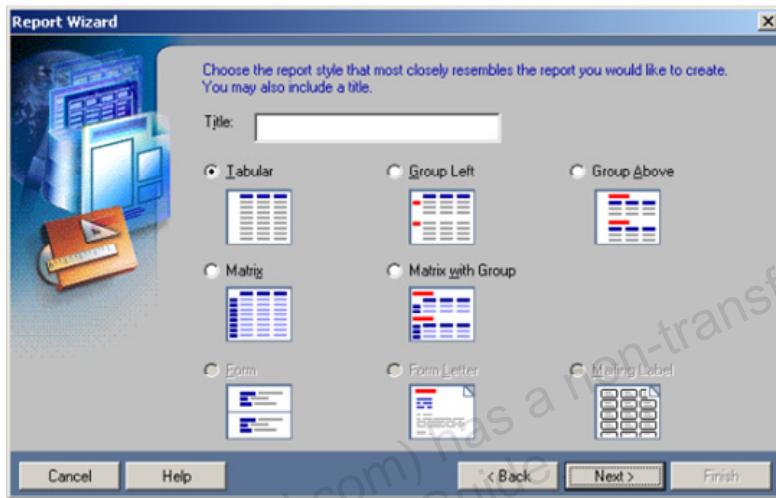
- Web and Paper Layout
- Web Layout only
- Paper Layout only

In this lesson, you will create a paper report. You can select either Create both Web and Paper Layout or Create Paper Layout only and then click Next.

# Creating a Tabular Report

## Wizard Pages

- Report Style
- Data Source Type
- Data Source Definition
- Fields
- Totals
- Labels
- Template



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Creating a Tabular Report Using the Wizard (continued)

### Report Styles

The second page of the Report Wizard shows the various styles of reports. Select Tabular and then click Next.

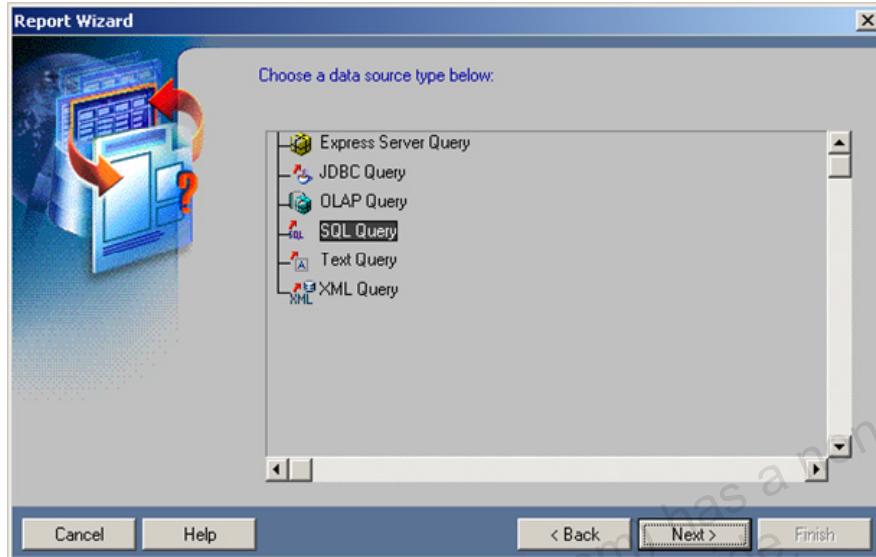
## **Creating a Tabular Report Using the Wizard (continued)**

### **Wizard Pages**

When you choose the Tabular style of report, the Wizard takes you through the following pages.

<b>Page Name</b>	<b>Description</b>
Data Source Type	Select the data source type on which you want to base your report.
Data Source Definition	Define the data you want to retrieve for your report.
Fields	Select the fields that you want to display in the output.
Totals	Select the fields that you want to summarize.
Labels	Alter the labels that appear for each field and the width of each field.
Template	Select the template that you want to use for this paper report. A template contains formatting information and can also contain standard information such as company logo, date, and so on.

# Selecting the Data Source Type



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

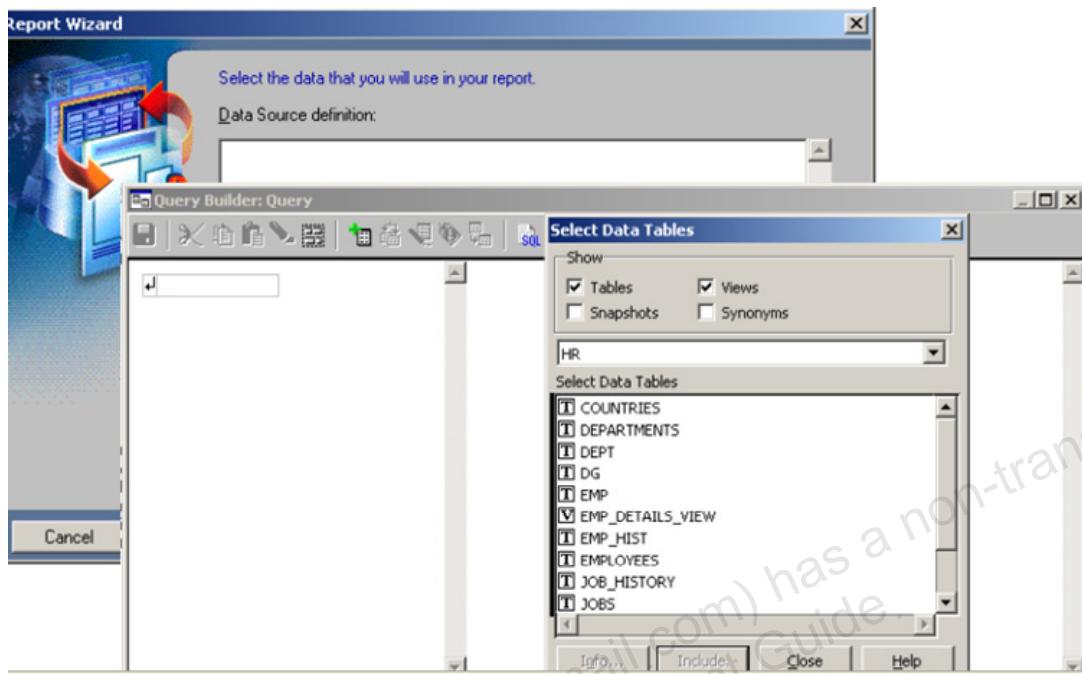
## Selecting the Data Source Type

Next, you define the data source type for your report. Through the implementation of the Pluggable Data Source (PDS) feature in Oracle Reports, the data for your report can come from any source you choose. Reports Builder provides interface definitions that act as a translator between Reports Builder and a PDS by redefining Reports Builder's requests in terms your data source uses.

Oracle Express Server, OLAP, JDBC, Text and XML pluggable data sources are shipped with Oracle Reports. You can also define your own data source.

This lesson will use the default data source, SQL query.

# Using Query Builder



ORACLE®

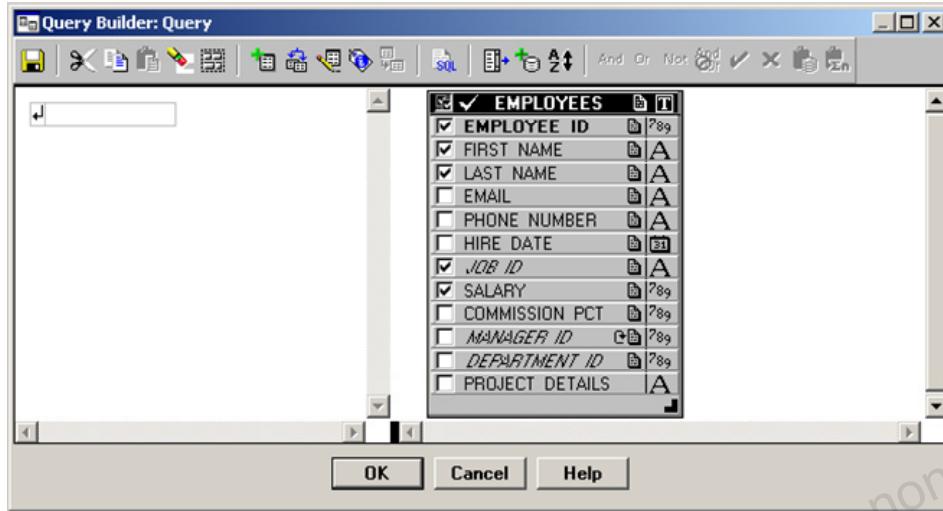
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Using Query Builder

Building your query with the Query Builder GUI saves you time and increases the ease of use for developers not familiar with building SQL statements or with the application tables.

# Building a Query

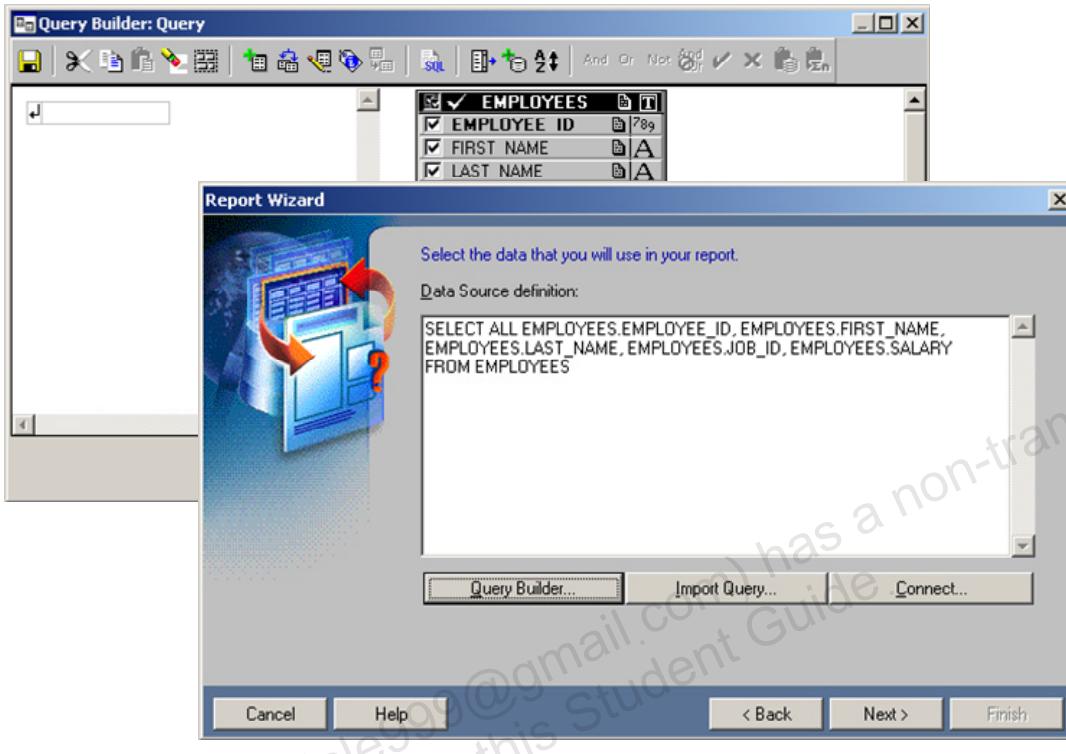
Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Building a Query



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

ORACLE

## Building a Query

To build a query using Query Builder:

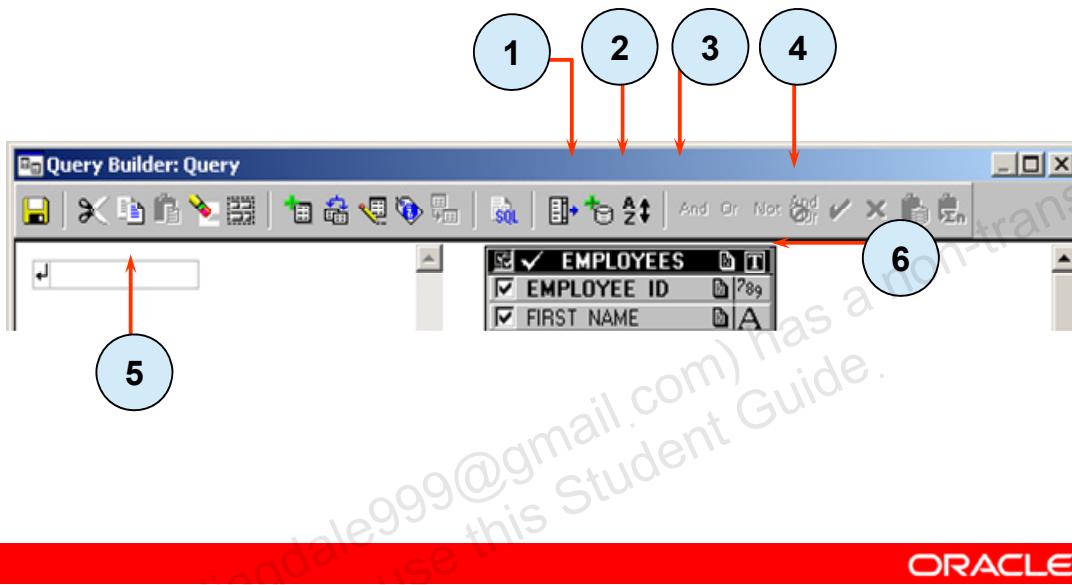
1. Select Query Builder from the Query page in the Report Wizard.
2. Enter your username, password, and alias in the Connect dialog box that appears if you have not already connected to the database.
3. Select the data tables to build the query.
4. Click Include. The tables appear in the selection area.
5. Click Close to close the Select Data Tables window.
6. In each table, double-click the column names that you want in the query, or use the check boxes. To select all columns, double-click the Table title bar.
7. Click OK.

Query Builder copies the query syntax into the Report Wizard. You can modify the query by reentering Query Builder or by modifying the SQL query statement text.

**Note:** If you prefer to write your own SQL statement, enter the syntax directly in the SQL query statement area of the Query page. Alternatively, you can import the contents of a file by clicking Import SQL Query.

# Query Builder Functions

User-friendly interface: Alternative to writing SQL syntax



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Query Builder Functions

This course does not teach the detailed functionality of Query Builder. You can find comprehensive help in the Reports Builder online Help.

You can use Query Builder to build almost any query that you can write as a SQL SELECT statement.

1	Column sequence
2	Define column
3	Sort
4	Logical operators for conditions
5	Conditions box for WHERE and HAVING clauses
6	Object type (T=Table, V= View, S=Synonym, A=Alias)

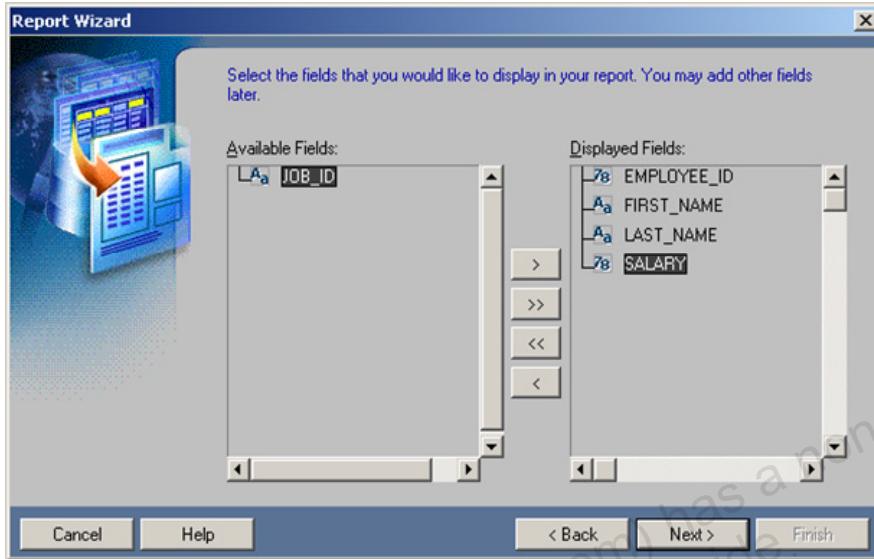
## Query Builder Functions (continued)

You can select from a table, view, or synonym. The letter T, V, or S appears in the object title bar to indicate which it is. If you define the same table more than once, Query Builder creates an alias (A).

The following is a brief description of some Query Builder toolbar buttons and functions.

Function	Description
Column sequence	Defines the sequence of column names in the SELECT clause
Define column	Defines a new, derived, or calculated column to be included in the SELECT clause; use this definition to calculate and retrieve derived values from the server
Sort	Defines the ORDER BY clause; select the columns that you want to be sorted, and choose ascending or descending sorting order
WHERE and HAVING clauses	Place the cursor in the Conditions box on the left side of the Query Builder window. Type a condition. Use the AND, OR, and NOT buttons to create compound conditions.

# Selecting Displayed Fields



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Selecting Displayed Fields

In the Field page, select each field from the Available Fields list and click >. The selected fields move to the Displayed Fields list.

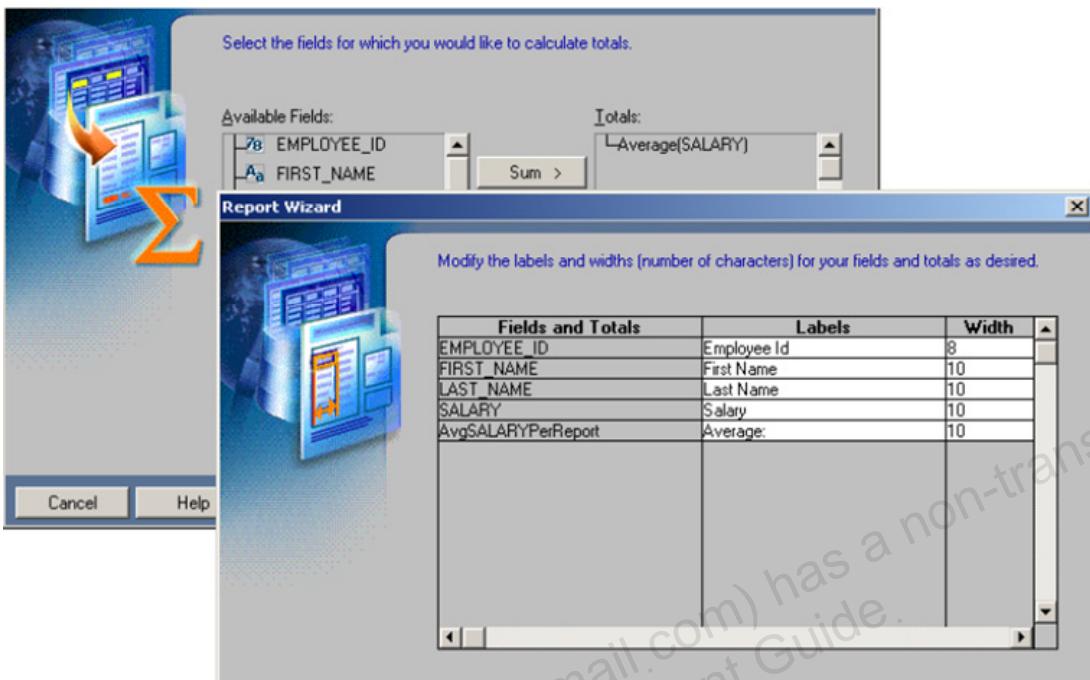
To display all fields, click >>.

You can alter the sequence of displayed fields by dragging one field above or below another in the list. The sequence of fields in this list determines how the fields appear in the report output. In a tabular report, the fields appear in sequence from left to right across the page.

Fields that remain in the Available Fields list are available for you to reference in your report definition as hidden fields or in PL/SQL trigger code.

In the report output, the user sees only those fields that you transfer to the Displayed Fields list.

# Totals and Labels



**ORACLE**

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Totals and Labels

In the next two pages of the Report Wizard, you can create totals based on any of the displayed fields and modify the labels and width of the displayed fields.

**Totals:** Standard SQL aggregate functions are provided for creating totals in your report. If you choose a total, the Wizard creates the total at each level of the report; that is, at report level and also at each break level, if your report contains break (master/detail) groups.

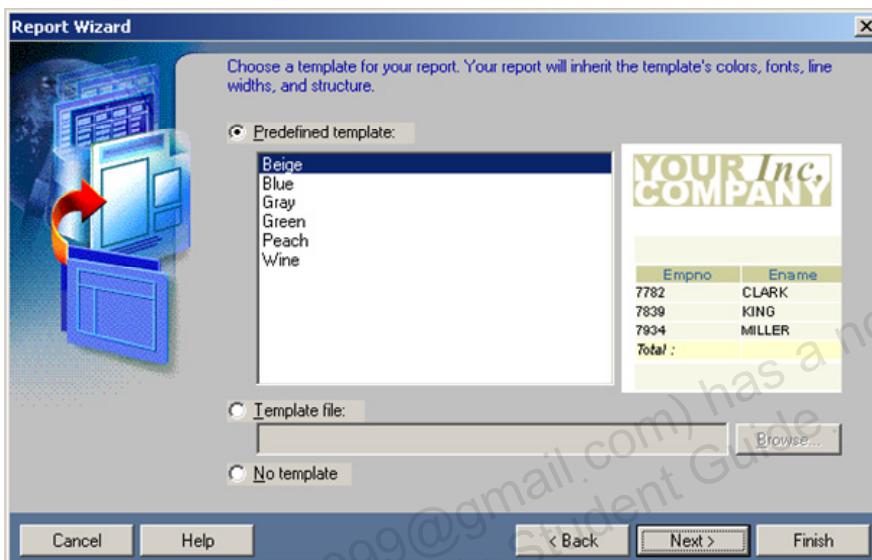
You can clear some of these totals later (by reentering the Wizard) if, for example, you want a report-level total only.

**Labels:** The field label is displayed on one or more lines in the report output. In a tabular report, the labels appear above the field values.

- If the initial label is wider than the field, Reports Builder allows enough space for the label, or displays it on multiple lines.
- If you increase the number of characters in the label text in the reentrant Wizard, the label can appear truncated in the report output.

## Selecting a Report Template

- Enforce corporate standards
- Create professional-looking paper reports easily



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Selecting a Report Template

Select a template from the list of predefined template names.

In a template, the fonts, styles, and colors are already selected for designated objects.

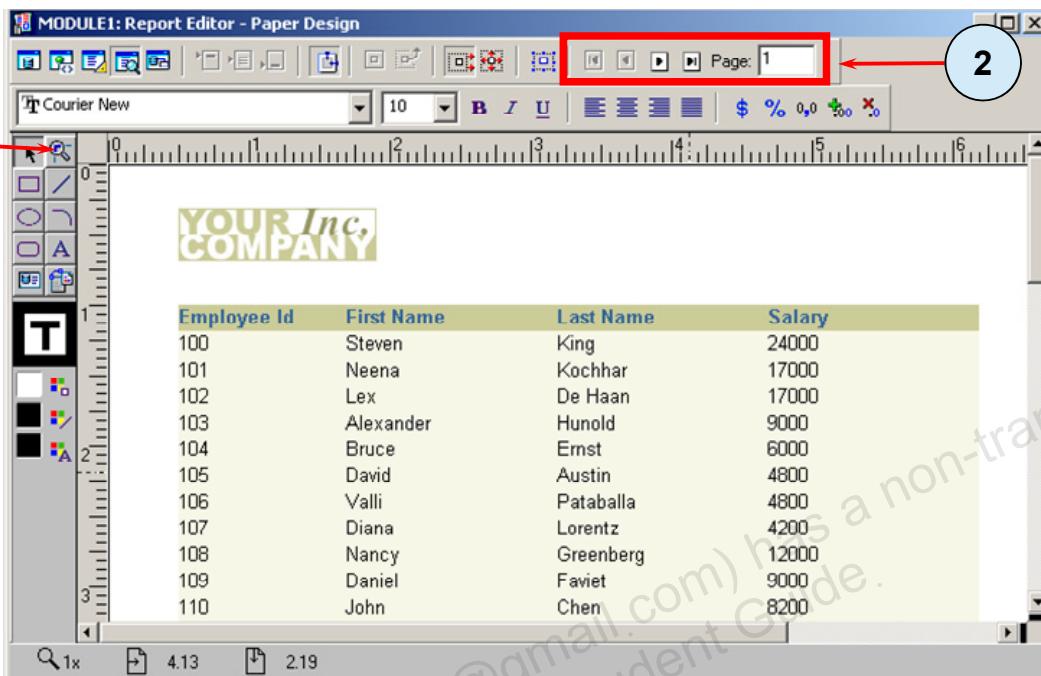
A variety of templates are available with the standard Reports installation.

To select a predefined template:

1. Select the Predefined Template option button, if it is not already selected.
2. Select a template from the Template list.
3. Click Finish.

**Note:** How to modify and use your own user-defined templates is covered later in the course.

# Viewing the Paper Report Output



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Viewing the Paper Report Output

When you finish creating your report in the Report Wizard, the output appears in the Paper Design view of the Report Editor.

### Magnifying the Output

The Paper Design view contains a Magnify tool in the vertical toolbar. This provides a view of the area of layout you want to see.

- To increase the view size, select the Magnify tool and click in the layout area.
- To reduce the view size, select the Magnify tool, hold down the Shift key, and click in the layout area.

You can also use the View menu to magnify or reduce the size of the output. Select View > Zoom to see your options.

### Viewing Different Pages

The Paper Design toolbar contains four buttons, and the specific page option, with which you can scroll through the pages of your report.

1	Magnify tool
2	Page buttons

# Saving the Report Definition

Save changes frequently.



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Saving the Report Definition

Remember to save the report frequently by selecting Save in the toolbar, or by using the File > Save menu option.

The recommended format for storing paper reports is with an .rdf extension.

If you want to make a copy of the report definition in a different filename, use the menu option File > Save As.

There is no toolbar button for the Save As option.

## Reentering the Wizard

- Select Tools > Report Wizard.
- Tabs are different for each report style.
- Wizard preserves all previous settings.



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Creating Other Report Styles

In this section, you discover the other report styles available in the Wizard and the pages that the Wizard displays. There are two methods of creating additional report definitions with the Report Wizard:

- Modifying an existing report definition by reentering the Report Wizard
- Creating a new report definition by restarting the step-by-step Wizard

### Modifying a Report by Reentering the Wizard

The reentrant Report Wizard preserves your current settings and query. You can make changes to these or other settings and click Finish when you are ready to reapply all the wizard settings to your report.

To reenter the Wizard in an existing report definition, follow one of these steps:

1. Select Tools > Report Wizard.
2. In the Object Navigator, select Report Wizard from the right-mouse-button menu.

## **Creating Other Report Styles (continued)**

When you reenter the Report Wizard, you see a tab for each page. You can navigate directly to the page you want to modify instead of clicking Next to move through each page in turn.

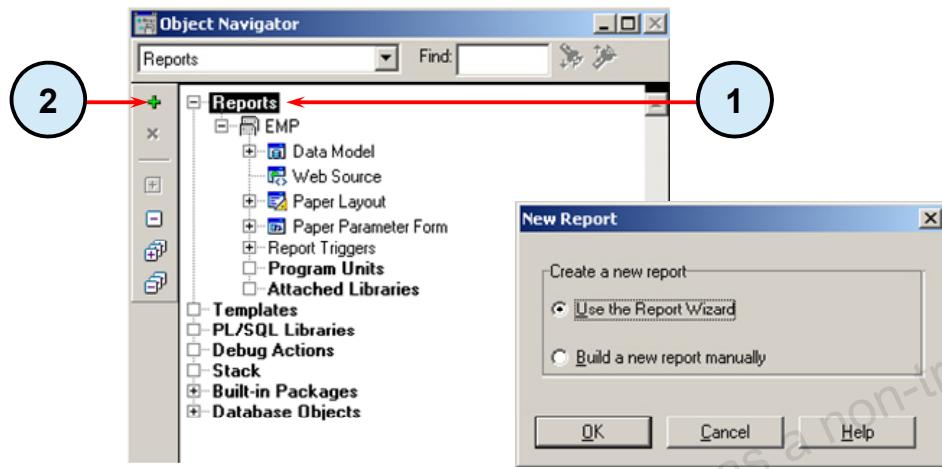
Click Finish at any time to apply the changes.

### **Creating a Form Report**

There are three significant features in the Form style that differ from the Tabular style.

- Labels appear to the left of each field.
- Each field appears to the right of the previous field, across the page.
- Each record appears on a new page.

# Creating a New Report



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Creating a New Report

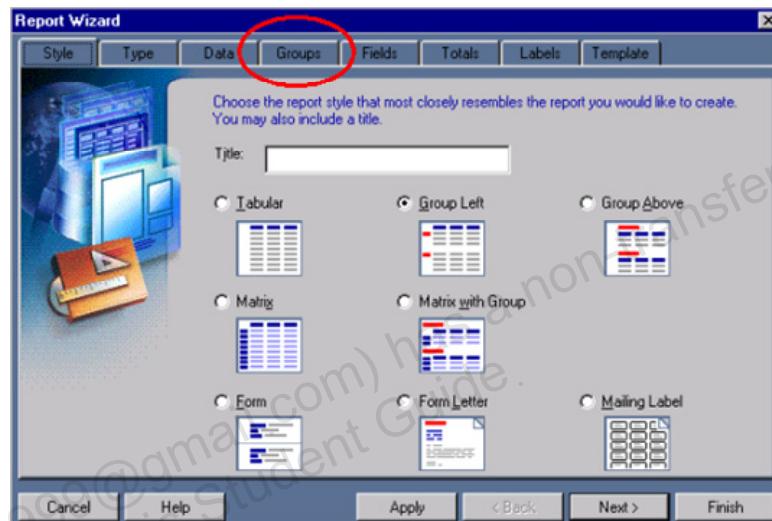
To create a new report with the Wizard when you are already in Reports Builder:

1. Select the Reports node in the Object Navigator.
2. Select the green plus sign in the vertical toolbar.
3. Select the Use the Report Wizard check box.

1	Reports node
2	Create object button

# Creating Break Reports

- Break report styles:  
Group Left, Group Above
- Additional  
wizard page:  
Groups



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Creating Break Reports

When you select one of the break (or master-detail) styles—Group Left or Group Above—the Wizard displays an extra page, called the Groups page, in which you choose the columns that should be in each break group of the report. You can choose multiple levels of breaks.

- **Group Left:** The output is displayed as columns across the page, with the groups next to each other and details to the right.
- **Group Above:** The output is displayed with the groups below each other nested within the parent group. The labels for all master groups appear to the side of the fields and details below.

**Note:** Do not select columns for the lowest (detail) group.

# Break Report Labels

## Group Left

<u>Location Id</u>	<u>Department Name</u>	<u>Last Name</u>	<u>Job Id</u>
xxxx	xxxxxx	xxxx	xxxx
		xxxxxx	xxxx

## Group Above

<u>Location Id</u>	xxxx
<u>Department Name</u>	xxxxxx
<u>Last Name</u>	<u>Job Id</u>
xxxx	xxxx
xxxxxx	xxxx

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Break Report Labels

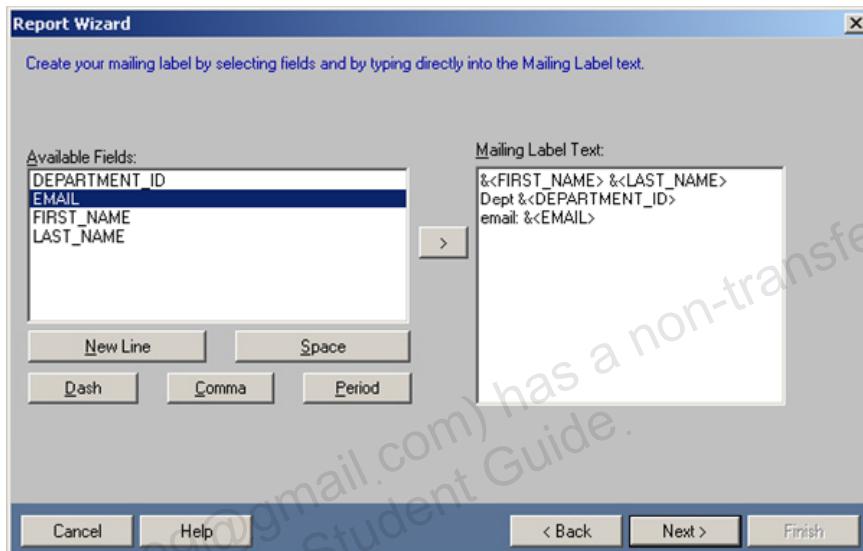
In a Group Left report, all the labels appear above the fields.

In a Group Above report, labels for the bottom detail groups appear above the fields, as in a tabular report.

# Creating Mailing Labels and Letters

Wizard pages:

- Style
- Data
- Text
- Template



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Creating Mailing Labels and Letters

When you choose the Mailing Label or Form Letter report style, the Wizard displays only four tab pages: Style, Data, and Template pages as before, and the Text page.

To create text:

1. Select a field from the Available Field list.
2. Click > to display in the text area.
3. Type new lines and punctuation as required, or use the punctuation buttons supplied: New Line, Space, Dash, Comma, and Period.
4. Select additional fields from the Available Fields list.

The selected field name appears in the text area, enclosed in optional angled brackets (<>) and prefixed by an ampersand (&). This indicates that the field name is a variable. Each variable is replaced by a specific value at run time.

## **Creating Mailing Labels and Letters (continued)**

You can type directly into the text area instead of selecting from the Available Fields list. However, remember to prefix each field name (variable) with an ampersand.

Any word that you type without an ampersand appears as a text string in your mailing label output. For example:

<b>Text Area</b>	<b>Output</b>
LAST_NAME &<LAST_NAME>	LAST_NAME Kingsway LAST_NAME Kochhar

The optional angled brackets allow you to display two variables side by side with no separating space.

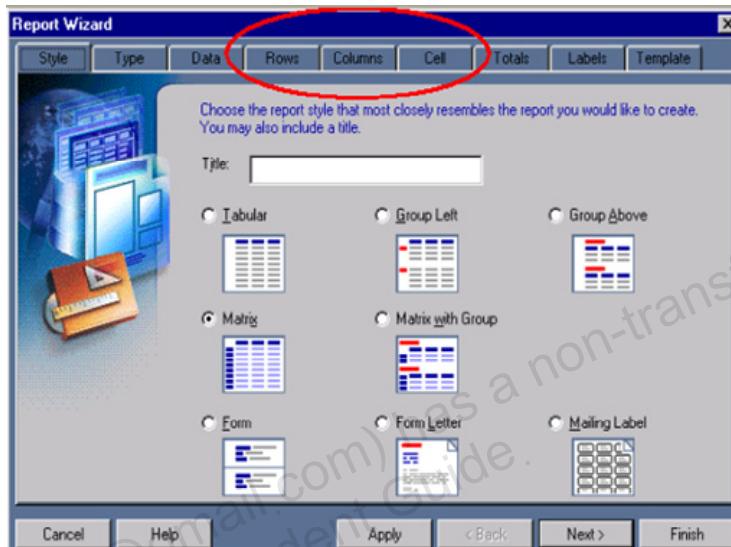
### **What Is the Difference Between Mailing Labels and Form Letters?**

<b>Mailing Label</b>	<b>Form Letter</b>
Multiple records on one page	One record on each page

# Creating a Matrix Report

Three additional wizard pages:

- Matrix rows
- Matrix columns
- Matrix cells



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Creating a Matrix Report

When you choose the Matrix report style, the Wizard displays three new tab pages.

Tab Page	Description
Rows	The field to be displayed vertically down the left side of the matrix; you can choose multiple levels of rows to create a vertically nested matrix
Columns	The field values to be displayed horizontally across the top of the matrix; you can choose multiple levels of columns to create a horizontally nested matrix
Cell	The field value that becomes the cell, or cross-product, of the matrix

## Creating Matrix Totals

When you select a total in the Totals page, the Wizard creates three totals in the matrix.

Summary	Description	Position in Output
Row	One value for each row	Right side of matrix, at end of row
Column	One value for each column	Bottom of matrix, below column
Report	One value for the report	Bottom right corner of matrix

## **Creating a Matrix Report (continued)**

### **Creating a Matrix for Each Group Record**

Select the Matrix with Group report style. This provides a similar group structure to the Group Above report.

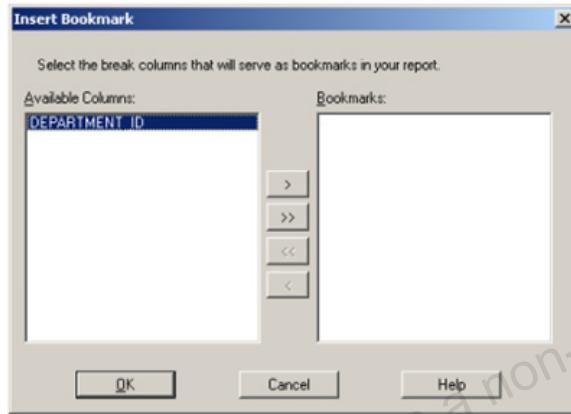
In your Regional report, you can select CITY in the Group page so that the Wizard structures the Department/Job\_ID matrix for each record.

The differences from a nested matrix are:

- Only the relevant Departments and Job\_IDs are displayed for each city.
- If you create summaries for a Matrix with Group style, the Wizard calculates and displays summaries for each group as well as the report total. The report total is displayed at the end of the report, in the bottom left corner.

## Previewing a Paper Report in a Browser

- Use Insert > Bookmark to create a bookmark for your break report



- Use File > Generate to File

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Previewing a Paper Report in a Web Browser

As you will learn in a later lesson, Oracle Reports enables you to easily create Web reports. However, you can deploy your paper reports on the Web with Oracle Reports. If you know that your paper report will be deployed on the Web, you can preview your report in your Web browser. You need to generate HTML or PDF output of your report in order to display it in your browser.

### Using Bookmarks

If you have used a break style for your paper report, you have the option of creating an outline for navigation, or bookmark, within your paginated HTML or PDF report, using the break column.

1. Select Insert > Bookmark.

**Note:** This menu option is enabled for the Paper Design and Paper Layout views only.

2. Select a column from the Available Columns list.
3. Click > to display in the Bookmark area.

# Previewing a Paper Report in a Browser

The screenshot shows a Microsoft Internet Explorer window displaying a paper report. The report is organized into sections based on department:

- Department Name Accounting**

Employee Id	First Name	Last Name	Salary
205	Shelley	Higgins	12000
206	William	Gietz	8300
- Department Name Administration**

Employee Id	First Name	Last Name	Salary
200	Jennifer	Whalen	4400
- Department Name Executive**

Employee Id	First Name	Last Name	Salary
100	Steven	King	24000
101	Neena	Kochhar	17000
102	Lex	De Haan	17000
- Department Name Finance**

Employee Id	First Name	Last Name	Salary
108	Nancy	Greenberg	12000
109	Daniel	Faviet	9000
110	John	Chen	8200
111	Ismael	Sciarra	7700
113	Luis	Popp	6900
112	Jose Manuel	Urman	7800

The report also features a sidebar with a navigation menu and a company logo for "YOUR INC COMPANY".

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Previewing a Paper Report in a Web Browser (continued)

To preview your paper report output in a Web browser:

1. Select or open the report.
2. Select File > Generate to File to specify the output format:  
Use Paginated HTML for HTML output.  
Use Paginated HTMLCSS for HTML Style Sheets.  
Use PDF for PDF output.
3. In the Save dialog box, you can modify the name and location of your HTML or PDF file.
4. Open your browser.
5. If you are using Internet Explorer, select File > Open and specify the location and name of your file. If you are using Netscape Navigator, select File > Open Page and specify the location and name of your file.

# Summary

In this lesson, you should have learned how to:

- Create paper reports with the Report Wizard
- Build queries using the Query Builder
- Apply templates to paper reports
- Modify reports by reentering the wizard
- Create new reports of different styles
- Preview a paper report on the Web



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Summary

### Report Styles

There are eight common styles of report that you can create by using the Report Wizard. The Wizard steps you through a variety of pages, depending on your chosen report style.

### Query Builder

The built-in Query Builder provides a user-friendly graphical interface for you to build the SQL queries to retrieve your report data. Alternatively, you can write the SQL statement yourself, or import the contents of a file.

### Predefined Templates

Predefined templates offer standard fonts, colors, formats, and images that enable you to create many reports with a professional appearance and a standard look and feel very quickly and easily.

### Reentrant Wizard

The Report Wizard is reenterable; it retains all your previous settings, and enables you to modify any settings by selecting the relevant tabbed page.

### Paper Reports on the Web

While Oracle Reports does offer the solution for creating true Web reports, you are able to publish a paper report on the Web.

## Practice 4 Overview

- Creating a Tabular report
- Modifying the report to create a Group Above break report
- Creating a Form Letter
- Creating a Matrix report
- Creating a Matrix with Group report



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Practice Session Overview: Lesson 4

This practice session contains:

- Creating and saving a Tabular report
- Modifying the report to create and save a Group Above break report
- Creating and saving a Form Letter
- Creating and saving a Matrix report
- Creating and saving a Matrix with Group report

The questions in the practice session are intended to ensure that you have a good understanding of how to create various styles of reports using the Report Wizard.

**Note:** When you are completing the practice sessions, use the file naming conventions that the questions specify. You may need these files in later practices.

In addition, note the field headings and widths. With many reports, you need to change these to achieve the desired result.

Many of the practice sessions ask you to rename files. Be sure to use Save As, not Save, so that you do not overwrite the existing filename.

## Practice Session: Lesson 4

1. Create a tabular report, paper layout only, containing the following data. Use Query Builder to build the query.

DEPARTMENTS: DEPARTMENT\_NAME

EMPLOYEES: FIRST\_NAME, LAST\_NAME, JOB\_ID, SALARY

2. Using the reentrant Report Wizard, make the following modifications.

- a. Modify the query to sort the data in descending order of salary.
- b. Add a summary to show the total salary value.
- c. Ensure the width of SALARY and TOTAL is 6.
- d. Select the Green template.
- e. Save the report to a file named p4q2.rdf.

3. Using the reentrant Wizard, modify the report to create a break report.

- a. Select the report style Group Above.
- b. Modify the query and remove the join clause so that the report contains all employees and not just managers.
- c. Group the report by the department name.
- d. Save the report as p4q3.rdf.

4. Create a new report as follows:

- a. Using the Report Wizard, create a paper layout and select the Mailing Label style.
- b. Enter the query in the data source definition window:

```
select c.cust_first_name || c.cust_last_name Name,  
c.cust_address  
from customers c
```

This query text is available in the file p4q4.sql.

- c. In the text area, include each of the following fields on a new line: NAME, STREET\_ADDRESS, CITY, and STATE\_PROVINCE. Include COUNTRY\_ID, and POSTAL\_CODE on the same line.
- d. Select No template and click Finish.
- e. Save the report to a file named p4q4.rdf.

## Practice Session: Lesson 4 (continued)

5. Using the reentrant Wizard, modify the report to create a Form Letter style.
  - a. Modify the query to include order information for each customer:

```
select c.cust_first_name || c.cust_last_name Name,
c.cust_address, o.order_id, o.order_total
from customers c, orders o
where c.customer_id = o.customer_id
```

This query text is available in the file p4q5.sql.
  - b. In the text area, enter some free-flowing text for each order. For example:  
Order No. &<ORDER\_ID> has been shipped to &<NAME> in &<C\_CITY>. The order has a total value of &<ORDER\_TOTAL> dollars.  
Thank you for your business.
  - c. Select a different template and click Finish.
  - d. Save the report as p4q5.rdf.
6. Create a new report.
  - a. Create both a Web and paper layout. Select the matrix report style.
  - b. For the query, import the contents of p4q6.sql.
  - c. Display customer names down the left side of the page.
  - d. Display product numbers across the top of the page.
  - e. Display the sum of the total values in the cells.
  - f. Create a summary to give the total of the sum(total\_value) values.
  - g. Change the width of all four summaries to 4. Change the PRODUCT\_ID label to Product. Remove the label for SumTOTAL\_VALUE.
  - h. Select any template and click Finish to preview your report.
  - i. Save the report to a file named p4q6.jsp.
7. Modify the matrix report to create a Matrix with Group.
  - a. Select Month as the group and check that all totals have a width of 7.
  - b. Click Finish to preview your report and save the report as p4q7.jsp.

**Note:** The query for the matrix reports above has been restricted to display only products beginning with “1,” so that you can see and understand the complete matrix more easily.

## **Practice Session: Lesson 4 (continued)**

8. Web-enable a paper report.
  - a. Open report p2q9.rdf and run the paper layout.
  - b. Add a bookmark to the report, letting the department names serve as bookmarks.
  - c. Generate HTML Style Sheet output. Save the file as p4q8.htm..
  - d. Open the report in a browser.
  - e. What happens when you click on the bookmarks?
  - f. Close the browser and in Reports Builder, save the report as p4q8.rdf.
  - g. Save and close all reports.

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

Shital jadhav (shitaljagdale99@gmail.com) has a non-transferable  
license to use this Student Guide.

## Enhancing a Basic Paper Report



**ORACLE®**

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Objectives

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

- Describe the Paper Design view
- Modify the display of report data in the Paper Design view
- Modify the positioning of report data
- Highlight data using conditional formatting
- Add page numbering and the current date



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Overview

In this lesson, you learn to enhance your paper reports with the most commonly required changes, while viewing the live run-time output in the Paper Design view in Reports Builder. You will learn how to enhance your Web reports is a later lesson.

## What Is the Paper Design?

The Paper Design is a view of report output that allows live editing of text and paper layout attributes:

- True WYSIWYG report editing
- Easy editing: See it. Click it. Change it.
- Cached report data



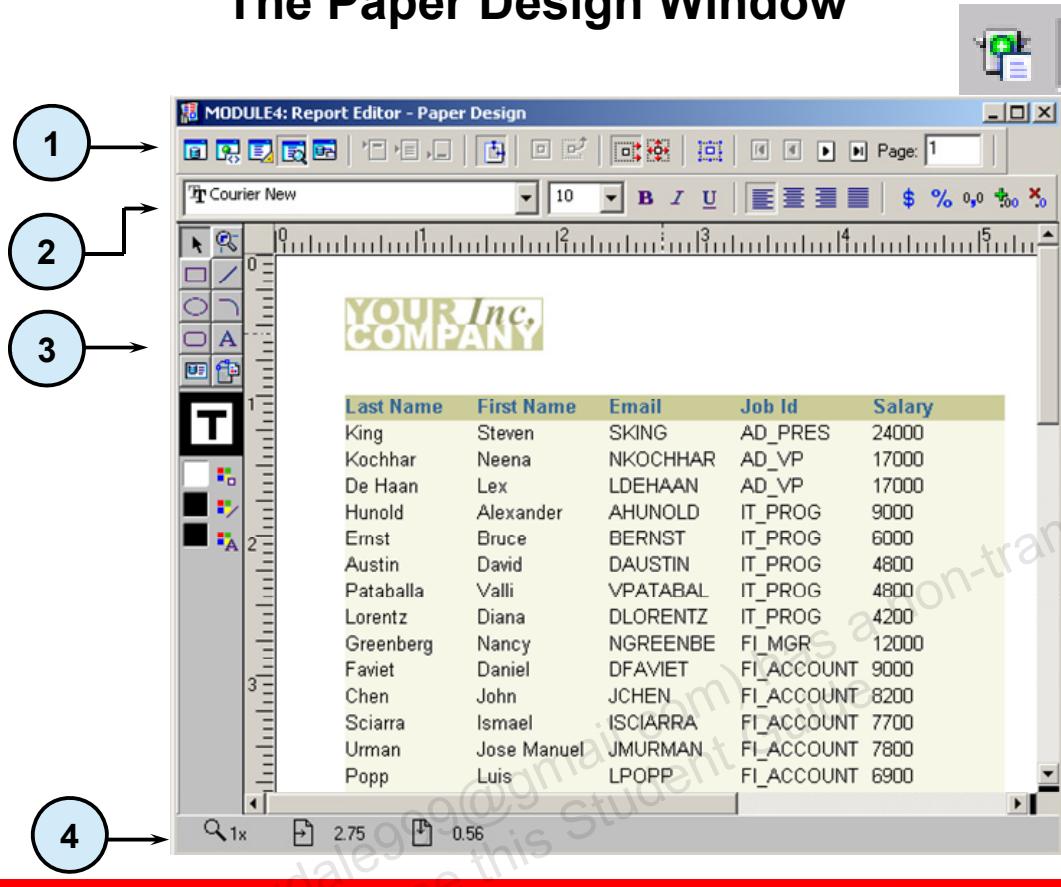
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### What Is the Paper Design View?

The report Paper Design view is a WYSIWYG editor. All edits that you perform in the Paper Design view are on live data.

Oracle Reports Developer caches the data the first time that you run the report, and then reuses the cached data each time you run the report during the session, unless you modify the report in any way that requires refreshed data, for example, changing the group structure or adding a summary column.

# The Paper Design Window



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## The Paper Design Window

The Paper Design window has a horizontal toolbar and stylebar that contain common functions, also available from the pull-down menu.

The vertical tool palette provides tools that enable you to create simple boilerplate objects and text to enhance your report, as well as color palettes to modify the color fill and borders of objects.

You can suppress the tool palette and status bar from the View pull-down menu.

Save your report definition often, using the Save icon on the toolbar or the Save or Save As options on the File menu, to preserve the changes that you make in the Paper Design view.

1	Toolbar
2	Stylebar
3	Tool palette
4	Status bar

# Modifying a Report

Common modifications:

- Align columns
- Set format masks
- Manipulate objects
- Edit text
- Insert page numbers and current date



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Modifying a Report in the Paper Design View

In this section, you learn about some of the most commonly required changes that you need to make to reports after creating the initial definition with the wizard:

- Aligning columns
- Setting format masks
- Manipulating objects
- Editing text
- Modifying visual attributes
- Highlighting data using conditional formatting
- Inserting page numbering
- Inserting current date and time

# Aligning Columns

The diagram illustrates the process of aligning column values. It shows two tables side-by-side. The left table displays salary values aligned to the left. An arrow points from the right table to the left table, indicating the transformation. Below the tables are two labels: "Point and select" next to the left table, and "Select 'Align Right'" next to the right table.

Last Name	Salary
Kingsway	\$24000
Kochhar	\$17000
De Haanah	\$22000
Hunold	\$9000
Ernst	\$6000
Austin	\$4800
Pataballa	\$4800
Lorentz	\$4200
Greenberg	\$12000
Faviet	\$9000

Last Name	Salary
Kingsway	24000
Kochhar	17000
De Haanah	22000
Hunold	9000
Ernst	6000
Austin	4800
Pataballa	4800
Lorentz	4200
Greenberg	12000
Faviet	9000

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Aligning Columns

In the sample report output, in the left panel, the salary values are aligned to the left. You often need to align numbers to the right.

1. Select the column that you want to align.
2. Select the “align right” icon on the stylebar.

You can align each object separately, so that you can center the labels if you want. However, if you want to align the values and the labels to the right, use [Shift]-click to select more than one object at a time.

# Setting a Format Mask

Change format masks from the stylebar.

Last Name	Salary
Kingsway	\$24,000.00
Kochhar	\$17,000.00
De Haanah	\$22,000.00
Hunold	\$9,000.00
Ernst	\$6,000.00
Austin	\$4,800.00
Pataballa	\$4,800.00
Lorentz	\$4,200.00
Greenberg	\$12,000.00
Faviet	\$9,000.00

The stylebar shows five numeric format symbols: 1. Currency symbol (\$), 2. Percentage symbol (%), 3. Thousand separator (,), 4. Add decimal places (.0+), and 5. Remove decimal places (.0+). Arrows point from each symbol to a corresponding numbered circle below it.

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Setting a Format Mask

The most commonly used numeric format symbols are available on the stylebar:

- Currency
- Percentage
- Thousand separator
- Decimal places: add and remove

1	Currency symbol
2	Percentage symbol
3	Thousand separator
4	Add decimal places
5	Remove decimal places

## Setting a Format Mask (continued)

To alter a numeric format:

1. Select the numeric field to which you want to apply a format mask.
  2. Select the corresponding format mask button on the stylebar.
- The format mask is applied.

**Note:** Reports Developer applies the format mask only to fields whose datatype is NUMBER. Also, the currency and percentage format masks are mutually exclusive: only one or the other can be applied to a numeric field, never both.

### NLS Support for Format Masks

The stylebar always shows “\$” for the currency button, “,0” for thousands, and “.0” for decimal places, but the output currency symbols are language-specific and can differ at run time.

If you use National Language Support (NLS) the symbols that appear in the report output depend on the value of the territory portion of the NLS\_LANG parameter.

# Manipulating Objects

- Clear fields
- Move fields
- Resize fields

Last Name	Salary
Kingsway	\$24,000.00
Kochhar	\$17,000.00
De Haanah	\$22,000.00
Hunold	\$9,000.00
Ernst	\$6,000.00
Austin	\$4,800.00
Pataballa	\$4,800.00
Lorentz	\$4,200.00
Greenberg	\$12,000.00
Faviet	\$9,000.00

Flex Mode adjusts layout during changes.



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Manipulating Objects

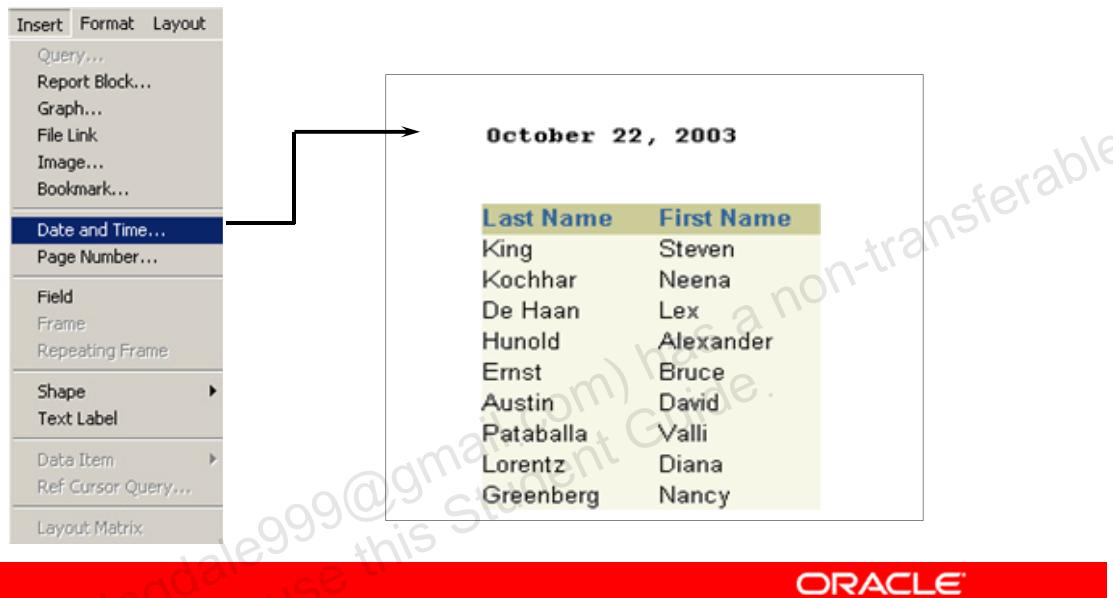
You can alter the position of objects, such as fields and labels, by selecting the object you want to move and dragging it to a new position.

Alter the size of an object by selecting one of the square handles and dragging to the required size, either smaller or larger.

**Flex Mode:** When you move or resize an object, the rest of the report adjusts as necessary. This is controlled by the Flex Mode option, which is a button in the toolbar. Flex Mode is enabled by default, so that all report objects *flex*, or adjust, to make room for your modifications.

# Inserting Page Numbers, Dates, and Times

- Inserted easily
- Customizable extensions



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Inserting Page Numbers, Dates, and Times

To add page numbers and the current date to your report in the Paper Design view, select the appropriate items from the Insert menu.

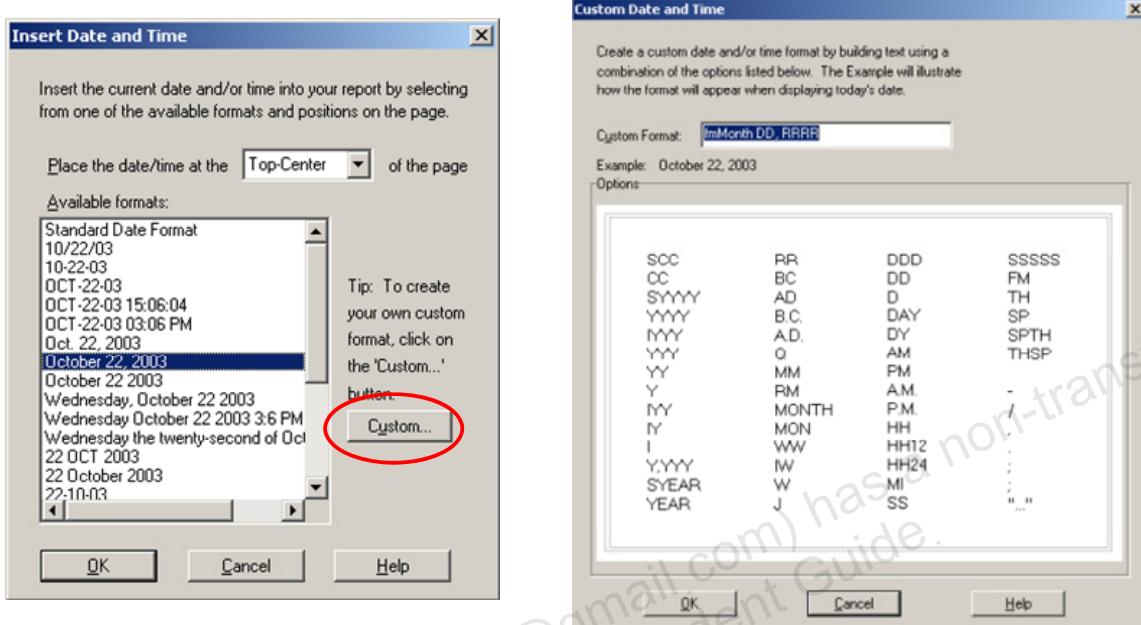
Insert Menu Item	Description
Page Number	Use this item to add a page number to the margin of a report.
Date and Time	Use this item to add the current date and time to the margin of a report. You can define your own date and time format masks.

## **Inserting Page Numbers, Dates, and Times (continued)**

The Insert Date and Time dialog box displays the current date in many different formats. You simply choose the style that you want. The underlying format mask is composed of tokens representing each element.

The list of formats depends on the entries in your preferences file. To modify the preferences list, select Edit > Preferences > Edit Masks.

# Customizing Dates



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Customizing Dates

If the mask that you want does not exist, and you do not want to add it to your preferences, select Custom. This dialog box provides a read-only list of tokens available for you to build your own style, if you understand the Oracle format tokens.

**Note:** The Date and Time Format Mask Syntax topic in the Help system fully describes the format mask syntax.

# Summary

In this lesson, you should have learned how to:

- Enhance report output using live data:
  - Move, resize, delete objects
  - Edit text
  - Apply format masks
  - Add page numbering and current date
- Save report to preserve changes



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Summary

### Features of the Paper Design View

The Paper Design view is a WYSIWYG editor that enables you to enhance your report output using the live data. You can manipulate objects and edit text content. You can also alter the font, colors, and patterns of individual objects.

The format mask buttons provide quick and easy currency formatting. The Insert menu enables you to add page numbering and the current date, using predefined formats or customizing your own.

### Saving Changes

Remember to save your report definition often during editing in order to preserve the changes you make in the Paper Design view.

When you save the report, all changes become part of your report definition and always appear at run time.

## Practice Session: Lesson 5

1. Open report p4q2.rdf. Make the following changes in the Paper Design view:
  - a. Make the SALARY column right-justified. Do not forget to justify the column header accordingly.
  - b. Add a comma and a currency symbol to SALARY. Add two decimal places.
  - c. Make the total at the end of the report right-justified and add commas, a currency symbol, and two decimal places, as in the column SALARY.
  - d. Change the label of the total to italic font.
  - e. Make whatever other changes you like.
  - f. Save the report to a file named p5q1.rdf and close it.
2. Open report p4q3.rdf. Make the following changes in the Paper Design:
  - a. Add a border with a hairline width around the total for each department.
  - b. Add commas and two decimal places to the SALARY field and the total and make them right-justified. Do not forget to justify the column header for the SALARY field accordingly.
  - c. Make the same changes to the total at the end of the report. Move the grand total so that it aligns with the SALARY field (you must do this manually; use the ruler guides to help you).
  - d. Make whatever other changes you like.
  - e. Save the report to a file named p5q2.rdf and close it.
3. Open report p5q3\_a.rdf. Make the following changes in the Paper Design:
  - a. Add a date at the top center of the page. Give it any format you want.
  - b. Make whatever other changes you like.
  - c. Save the report to a file named p5q3.rdf and close it.
4. Open report p4q6.jsp. Make the following changes in the Paper Design:
  - a. Right-justify the Product ID field.
  - b. Add commas and two decimal places to the cells and summaries and make them right-justified.
  - c. Why are some of the cell and summary values displayed with asterisks? What can you do to correct this?
  - d. Save the report to a file named p5q4.jsp and close it.

## Enhancing Reports Using the Data Model: Queries and Groups



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Objectives

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

- Describe the Data Model objects
- Modify query properties
- Modify the report hierarchy
- Change the order of data in a group
- Eliminate data from a report



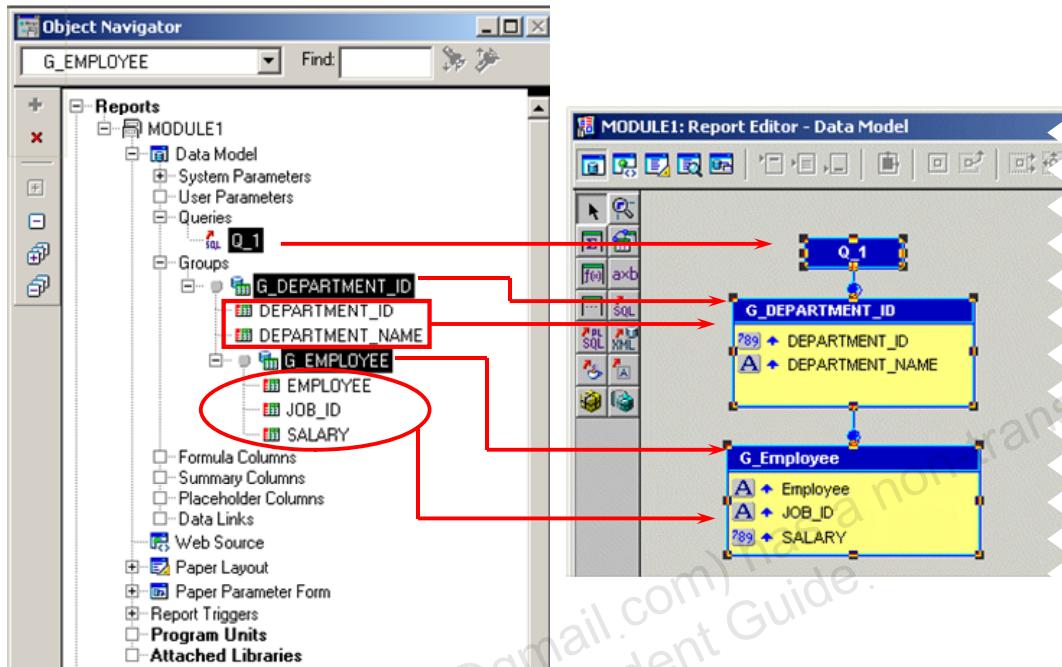
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Overview

You have learned to create simple, single-query paper reports using Reports Builder.

In this lesson, you learn about the Data Model structure and objects, how to modify SQL queries created with the Report Wizard, and how to enhance reports by creating additional groups to achieve more complex report structures.

# The Data Model Objects



**ORACLE**

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## The Data Model Objects

The Data Model defines the report data and its hierarchy—that is, the report structure. The Data Model does not define any formatting attributes for the report output. This section introduces you to the objects in the Data Model and their appearance in the Data Model editor.

The Data Model consists of these objects:

- **Query:** You can create a report with one or more queries. The Report Wizard creates a single-query report. Use the query tool or the Data Wizard to create additional queries.
- **Group:** Each group is owned by a query. By default, Oracle Reports creates one group for each query.
- **Column:** Columns are owned by the group to which they belong. By default, the group contains one column for each select expression in the Query.
- **Link:** Links join a parent group to a child group. You can create links to form a relationship between groups from different queries. Links are *never* created by default.
- **Parameter:** Parameters are owned by the report. You can create parameters that allow users to enter value restrictions at run time. Oracle9i Reports also provides a number of system parameters by default.

## The Data Model Objects (continued)

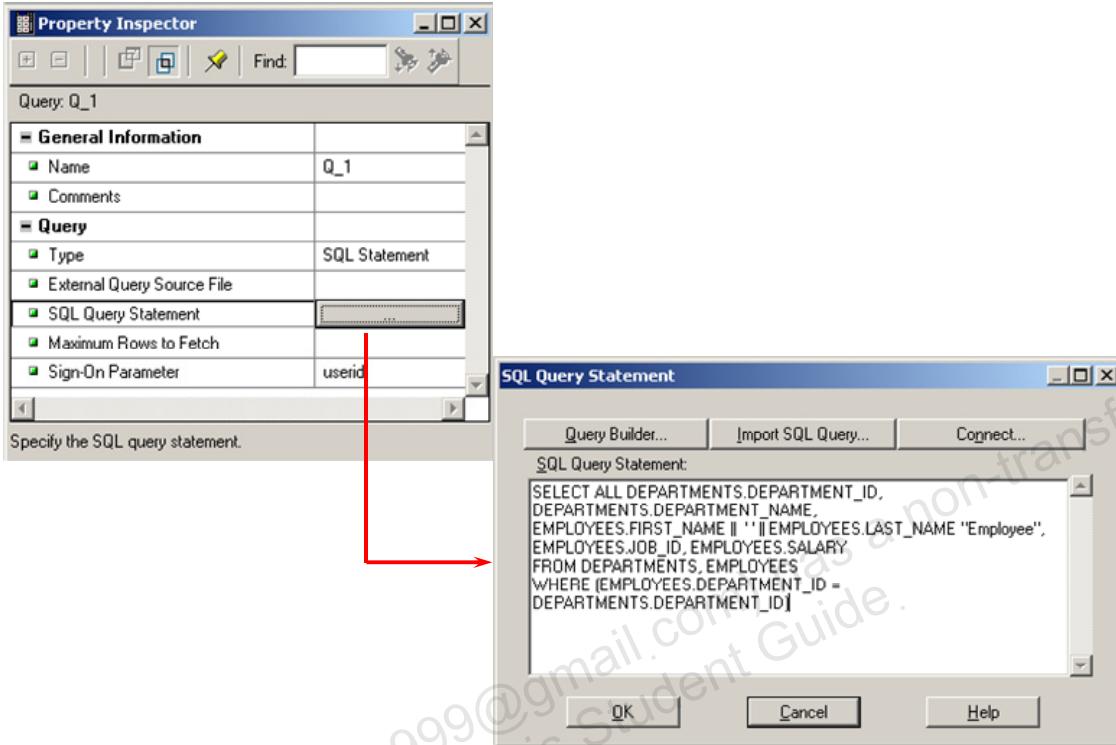
The objects that the Report Wizard creates by default depend on the style of report that you choose.

- Tabular style: One query, one group
- Group Above, Group Left: One query, two or more groups

The number of group objects depends on the number of break groups you define in the Report Wizard.

The Report Wizard creates only one query.

## Modifying Properties of a Query



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Modifying Properties of a Query

You can modify a query, including the SQL statement, by changing properties in the query Property Inspector.

To edit the query statement, open the Property Inspector, choose the SQL Query Statement property, and click the left button.

The SQL Query Statement dialog box appears.

You can also use the Data Wizard to make changes to the query. The Data Wizard is covered in a later lesson.

### Modifying Columns

If you add to, rename, or delete a column or expression in the query statement, Oracle Reports automatically creates, revises, or deletes the corresponding column in the Data Model.

## Modifying Properties of a Query

- Modify SQL query statement:
  - Add, rename, or delete columns
  - Use column and table aliases
  - Remove or modify schema name
- Syntax error checks occur when:
  - Exiting SQL query statement
  - Compiling or executing a report



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Modifying Properties of a Query (continued)

#### Using Column and Table Aliases

Remember that you can use aliases against the database columns and tables in the query. Report Builder uses your column aliases to create the relevant report column names. Table aliases help your query access the database and return data as quickly as possible. Aliases therefore provide:

- Fast database access and return of data
- Shorter, more meaningful names for query expressions, such as Employee and Annual\_Salary as in this example:

```
SELECT      d.department_id, d.department_name, e.job_id,
            e.first_name || ' ' || e.last_name Employee,
            e.salary*12 Annual_Salary
FROM        employees e, departments d
WHERE       e.department_id = d.department_id
```

## **Modifying Properties of a Query (continued)**

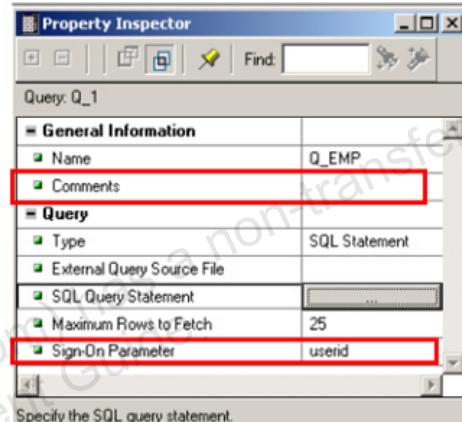
Oracle Reports checks the syntax when you try to leave the SQL Query Statement dialog box and whenever you generate or execute the report.

If Oracle Reports detects an error, it displays the error in an alert.

# More Properties

Aid maintenance and testing:

- Rename queries in complex reports
- Add comments to describe queries:
  - Use the Comment property
  - -- comment for single lines
  - /\*comment \*/ for multiple lines
- Set Maximum Rows to Fetch to restrict data



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## More Properties

### Renaming Queries

It is recommended that you rename queries in a complex report to make the Data Model easy to understand.

Renaming a query does not automatically rename the group below it.

### Commenting Your Queries

Use the Comment property to enter meaningful text that describes the query.

You can also insert comment lines directly in the SQL Query Statement area, using the Oracle standard comment delimiters:

- --comment for a single line
- /\*comment \*/ for multiple lines

**Note:** For ease of maintenance, do not enter comments in different places for the same query.

## **More Properties (continued)**

### **Restricting Rows**

Set the Maximum Rows property to restrict the rows returned from a query. This is useful for testing your report against a large data source.

## Applying Changes

Update the paper layout and Web source to reflect changes in the Data Model.

- For paper reports:
  - Select Report Wizard.
  - Alter the necessary tabbed pages.

The Wizard destroys previous layout and creates new objects.

- For Web reports:
  - Select the Report Wizard and navigate through each of the tabbed pages, reselecting the desired options.  
or
  - Edit the Web source manually.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Applying Changes

When you alter the Data Model, you must update your layout to see the changes.

For paper reports, select Tools > Report Wizard from the menu, and alter the tabs to create a new layout with additional groups and columns.

## Applying Changes (continued)

For Web reports, reentering the Report Wizard will delete the layout that was previously created. A new Web layout will be created based upon the selections you make on each of the tab pages in the Wizard. You can also manually edit the Web layout using the Web Source view of the Report Editor. You will learn more about this in a later lesson.

**Note:** If you edit a query statement to alter any columns that are part of an existing layout, and run the report without re-creating the layout, the Source property becomes null, and the field in the Previewer appears with a large cross.

# Changing the Group Structure

Groups determine hierarchy and frequency.

- Wizard creates default groups.
  - Default naming conventions
  - You can change query name.
  - You can change group name.
- Developer-created groups for:
  - Control break reports
  - Complex matrix reports



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Changing the Group Structure

Groups organize your data into sets of records and, in a hierarchy, determine the frequency at which records in that group appear in the output. Each group contains columns that Reports Builder assigns to the group by default, or that you specifically assign.

### Changing the Group Name

If you change a query name after the group has been created, the group name *does not* change automatically.

There are two ways you can change the group name:

- Open the group Property Inspector and change the Name property.
- In the Object Navigator, select the node representing the group and then select the group name again. This removes the highlight and changes the cursor to an I-beam. Edit the group name. The next time you activate the Data Model editor, the new group name is visible.

**Note:** If you create a Web layout and change the name of a group, the Web Source will not reflect the change. You will get an error when you select Run Web Layout: “Cannot create cursor for group <original\_group\_name>”. To update the Web Source, edit the Web Source code, locate the rw:foreach tag for that group, and modify the corresponding source (`src`) attribute.

## Changing the Group Structure (continued)

**Example:** You create a Web layout using the Report Wizard and the default group name is G\_LAST\_NAME. You change the name to G\_EMP in the Property Inspector. Now do the following:

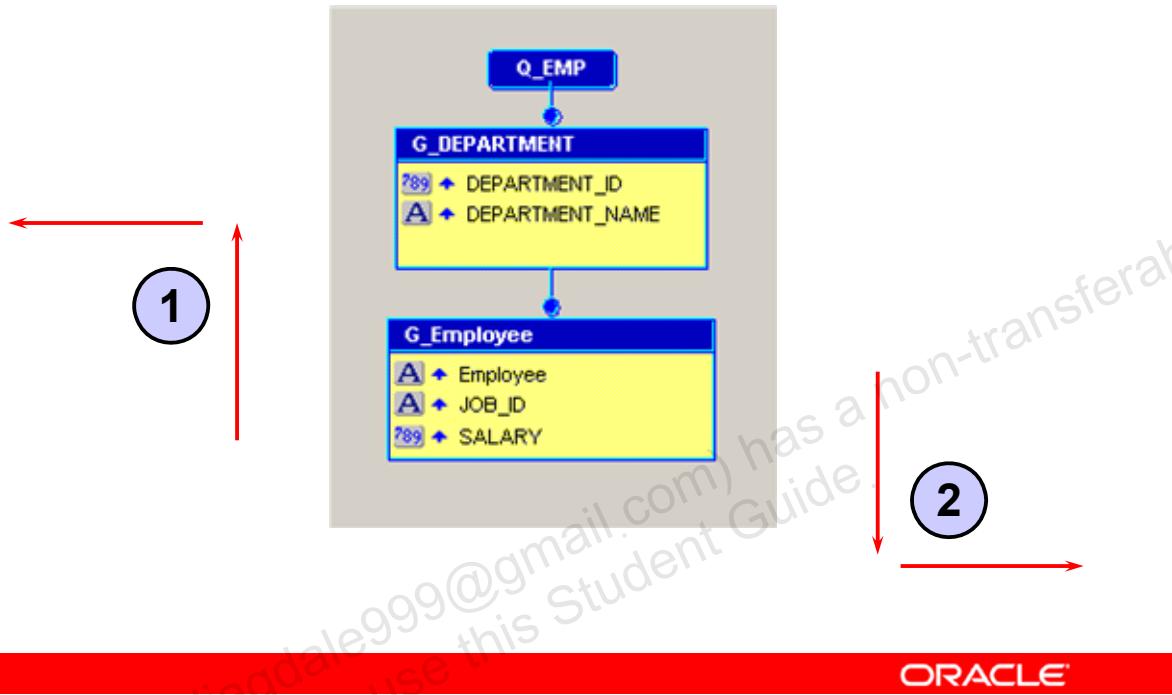
1. Display the Web Source view of the Report Editor.
2. Locate `<rw:foreach id="R_G_last_name_1" src="G_last_name">`.
3. Change the value of src to G\_emp:  
`<rw:foreach id="R_G_last_name_1" src="G_emp">`

## Developer-Created Groups

Sometimes you do not want to modify existing Data Model objects in the Report Wizard; for example, during later maintenance. In this case, you might need to create your own groups in the Data Model to perform the following actions:

- Produce control break (nested) reports
- Produce complex matrix reports

## Group Hierarchy



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Group Hierarchy

The groups in Oracle Reports exist in a hierarchy. You can alter the group hierarchy by creating new groups.

As you have already seen, a single query produces one default group. You can produce a control break report structure by creating one or more additional groups above the default. The following example of a single query, called Q\_EMP, is the basis for the hierarchical report.

```
SELECT e.first_name, e.last_name, e.job_id,
       d.department_id, d.department_name
  FROM employees e, departments d
 WHERE e.department_id = 14
```

1	Drag columns up or left to create a parent group.
2	Drag columns down or right to create a child group.

## Group Hierarchy (continued)

### Creating a New Group

Select DEPARTMENT\_ID and drag it out of and above the default group to create a break group. Change the group name to make it more meaningful.

You can drag other columns, such as DEPARTMENT\_NAME, into the new group. Alternatively, move more than one column into a new group by using [Shift]-click to select the columns before dragging them all together.

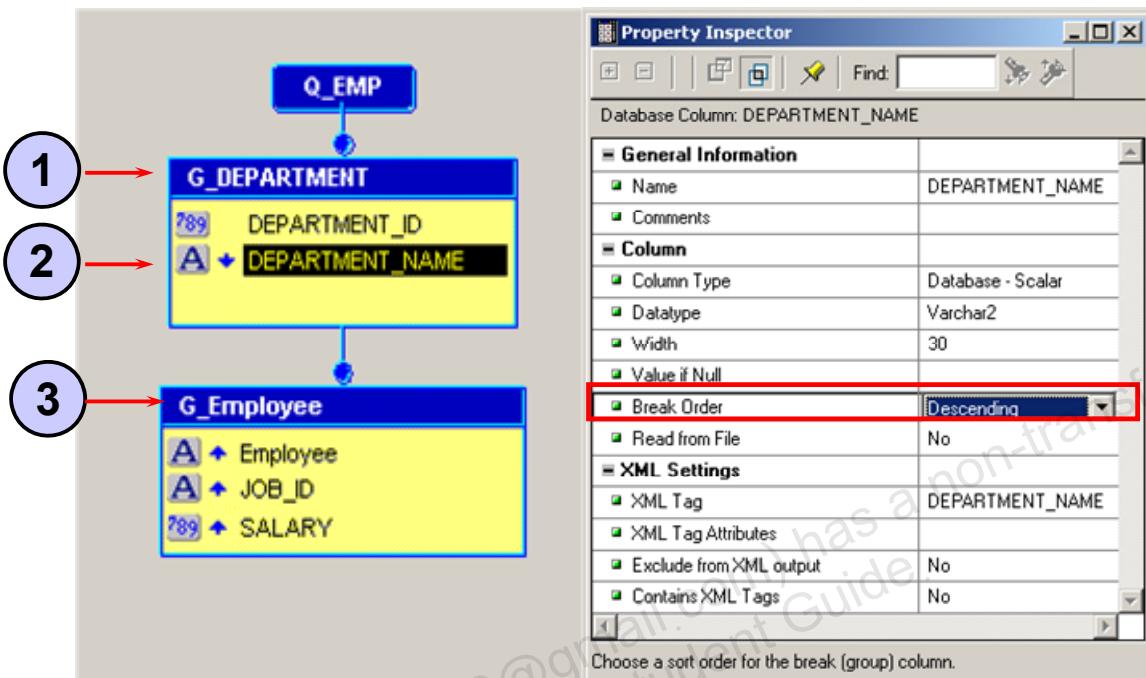
**Note:** You can also drag columns *down* a hierarchy, from a higher group into a new or existing lower-level group. The order in which the groups exist in the Data Model has an effect on the output. Always make sure that your parent (master) group is first and the child (detail) group is second.

When you drag a column to create a new group, the new group is only one level above or below the column's original group in the hierarchy.

To create a group more than one level from the original group, follow these steps:

1. Move the column to the level immediately above or below the required position of the new group.
2. Drag the column out to create the new group at the next level.

# Ordering Data in a Group



**ORACLE®**

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Ordering Data in a Group

Break groups that you create in a hierarchical report display data in a default order. You can modify this default.

*Break order* controls the break structure and the order in which to display the column's values. It is denoted by a solid arrowhead to the left of the column. For column values in user-created groups, you must use the Break Order property to specify how to order the break column's values. The order of precedence depends on the order of columns, from the top down, in the group.

To reverse the break order, double-click the column in the group to open up the column Property Inspector. Change the break order from ascending to descending using the pop-up list. In the Data Model, the arrowhead is displayed in reverse.

**Note:** All break groups must have at least one column with Break Order set.

1	Break group controls primary sort; use the Break Order property
2	Order by DEPARTMENT_NAME column: Break Order = Descending
3	Child group controls final sort; uses ORDER BY clause in the query

## Ordering Data in a Group (continued)

### Break Order in Break Groups

Remove unnecessary break columns in each break group to improve the performance of your report. A break group is any group other than the lowest group of each query. If several columns are all unique, such as the DEPARTMENT\_ID and DEPARTMENT\_NAME columns, having break order set on both columns can sometimes cause unnecessary sorting.

1. Open the DEPARTMENT\_NAME column Property Inspector.
2. Alter Break Order to None.
3. Repeat for each column that is not part of the sort.

### Break Order in Lower Groups

The Break Order property has no effect on columns in the lowest group of each query. Modify the ORDER BY clause in the SQL query statement to control this group.

Removing the Break Order property from columns in this lowest group is optional and has no effect on the output.

## Query Modifications

In a break report, data order is determined by:

- Break order columns in the break groups
- Columns that you specify in the ORDER BY clause

```
SELECT d.department_id, d.department_name,
       e.last_name...
  FROM employees e, departments d
 WHERE e.department_id = d.department_id
 ORDER BY 2, e.salary
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Query Modifications

The following example shows how the ORDER BY clause is modified.

For each database column with the Break Order property set, Oracle Reports appends an ORDER BY clause to the query or modifies the existing ORDER BY clause.

The break columns always take precedence over the lowest group columns. For example, suppose that your report contains the following query:

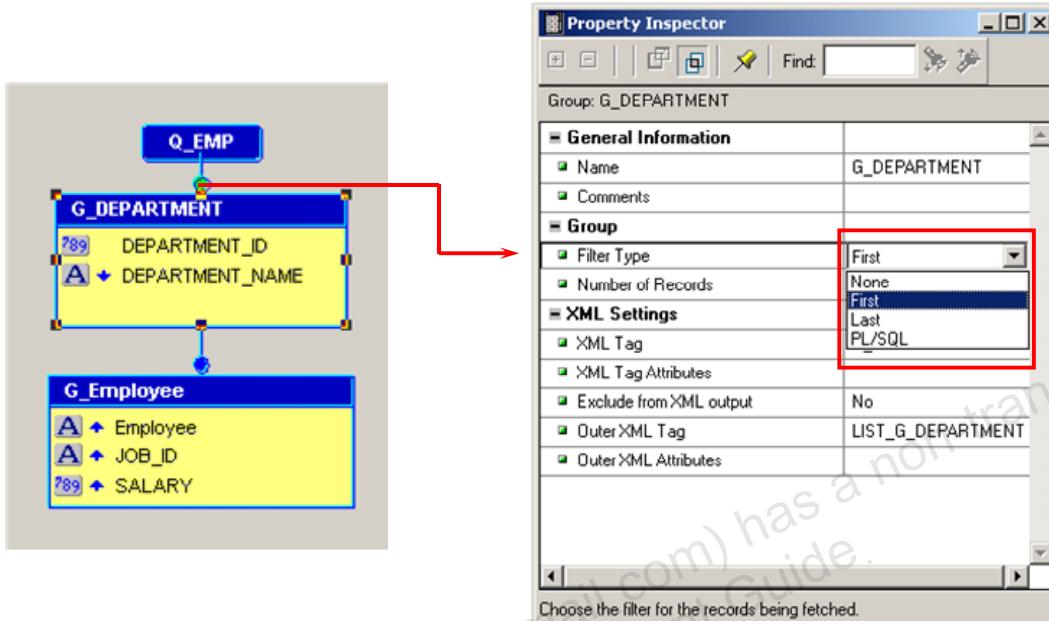
```
SELECT d.department_id, d.department_name, e.last_name,
       e.salary
  FROM employees e, departments d
 WHERE e.department_id = d.department_id
 ORDER BY e.salary
```

## Query Modifications (continued)

If you create a break group containing DEPARTMENT\_ID and DEPARTMENT\_NAME, but with DEPARTMENT\_NAME as the only break column, then at run time your query becomes:

```
SELECT d.department_id, d.department_name, e.last_name,  
      e.salary  
  FROM employees e, departments d  
 WHERE e.department_id = d.department_id  
 ORDER BY 2, e.salary
```

# Filtering Data in a Group



**ORACLE®**

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Filtering Data in a Group

A *group filter* allows you to control the processing of records in each group. This section briefly describes the two types of filters available and how to use a packaged filter to restrict the number of records returned.

There are two types of group filters:

- A packaged filter allows you to restrict processing to a predetermined number of records.
- A developer-created filter allows you to control processing dependent on conditional PL/SQL code.

**Note:** Developer-created filters are covered in more detail later in the course.

In the Data Model editor, the circle above the group object changes to green when a filter is active.

# Using a Packaged Filter

Reports Developer provides two packaged filters:

- First: Retrieves the first <n> records for the group
- Last: Retrieves the last <n> records for the group



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Using a Packaged Filter

A *packaged filter* allows you to specify the number of records that you wish to retrieve in a group. You can use this to restrict the data while developing your reports.

There are two packaged filters, *First* and *Last*.

- The First filter retrieves the first <n> records for the group.
- The Last filter retrieves the last <n> records for the group.

To apply a packaged filter:

1. Double-click the group title bar, or its icon in the Object Navigator, to display the Property Inspector.
2. Change Filter Type to First or Last.  
The Number of Records property appears.
3. Enter the number of records you require.

How is data fetched in a packaged filter?

- First filter: Oracle Reports retrieves a multiple of the array size, sufficient to satisfy the filter, based on a parameter value at run time.
- Last filter: Oracle Reports must retrieve all records to establish which are last.

# Summary

In this lesson, you should have learned how to:

- Identify Data Model objects
- Modify query properties
- Modify the report hierarchy by creating additional groups
- Filter data in a group



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Summary

### Queries

A report performs well only if the SQL code that accesses the database is efficient. Write well-constructed SELECT statements.

You can restrict the rows returned by a query using the Maximum Rows property. This is useful for testing purposes.

### Groups

Organize data into sets of records, to establish hierarchical reports to any depth and width.

## Practice 6: Overview

- Modifying report hierarchy
- Restricting records in a query
- Restricting records with a packaged filter



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Practice Session Overview: Lesson 6

This practice session contains:

- Modifying report hierarchy by creating additional groups
- Restricting records in a query
- Restricting records with a packaged filter

In this practice session, you modify an existing tabular report and create a break report. Using two different techniques, you restrict the records in a report.

## Practice Session: Lesson 6

1. Open report p8q1\_a.jsp, a tabular report showing customer information.
  - a. Run the paper layout.
  - b. In the Data Model view, create a break group using NLS\_TERRITORY as the break column. Rename the group G\_COUNTRY.
  - c. Reenter the Report Wizard to update the layout. Select the Group Left style. Ensure that all fields are displayed.
  - d. Save the report as p8q1.jsp. Do not close the report.
2. Modify the previous report to restrict the number of customer records processed by the group.
  - a. Place a filter on the G\_CUSTOMER\_ID group to display only the first ten customers. Run to test.
  - b. Save the report as p8q2.jsp and close it.
3. Modify p8q1.jsp to restrict the report to customers who have a credit limit greater than 1500. You need to modify the query properties.
  - a. Include CREDIT\_LIMIT in the query statement, although you do not need to display it. Restrict the query statement to show those customers who have a credit limit greater than 1500.
  - b. Save the report as p8q3.jsp and run the Web layout.

## Enhancing Reports Using the Data Model: Data Sources



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Objectives

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

- Describe the various data source types



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Overview

To keep pace with today's demand for information, you need to be able to access all available data. You often need to access data from your corporate internal data sources as well as data sources outside your corporation.

In this lesson, you learn about the different data source types that can be used for a report. You will learn how to access data using the Pluggable Data Source (PDS) feature in Oracle Reports and how to combine data from multiple sources to publish meaningful information.

# Data Source Types

Access data from a variety of sources:

- SQL-based
- XML
- Oracle OLAP
- JDBC
- Express
- Text



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Data Source Types

With Oracle Reports, you can publish data from any data source, and even combine data from one or more of these data sources in a single report.

For example, today e-businesses use XML as a means of data interchange. Imagine a business-to-business scenario where a wholesaler's inventory is available to a supplier through XML. The supplier wants to see what products to ship to his customer, but obviously, he can only ship those products he has in stock himself. This data is stored in his internal relational database. Using Oracle Reports, you can create a report merging the XML data and the relational data in real-time and display the results in a Web page. The merge, or join, returns only the items the wholesaler needs and the supplier has in stock.

In another scenario, you may need to combine data from a relational database with data from a multidimensional database to compare trends and performance.

With Oracle Reports, you can access, combine, and publish any data.

## Data Source Types (continued)

The information you need to publish is often derived from various data sources. These data sources may be:

- SQL-based: relational databases like Oracle10g
- Non SQL-based: data, such as XML, that has been generated from outside the corporation
- Oracle OLAP (Online Analytical Processing ): an integrated part of Oracle Database that provides support for multidimensional calculations and predictive functions. Oracle OLAP supports both the Oracle relational tables and multidimensional data types.
- Java Database Connectivity (JDBC): other relational data sources such as SQL Server, Sybase, or another Oracle database
- Express: multidimensional database that stores decision support data. The Express data source is provided for backward compatibility
- Data in flat files

## Summary

In this lesson, you should have learned :

- The various data source types

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

Shital jadhav (shitaljagdale99@gmail.com) has a non-transferable  
license to use this Student Guide.

# Enhancing Reports Using the Data Model: Creating Columns

8

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Objectives

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

- Describe various Data Model columns
- Display the contents of a file
- Identify characteristics of user-defined columns
- Create report summaries and subtotals
- Derive data using a formula column
- Create and populate a placeholder



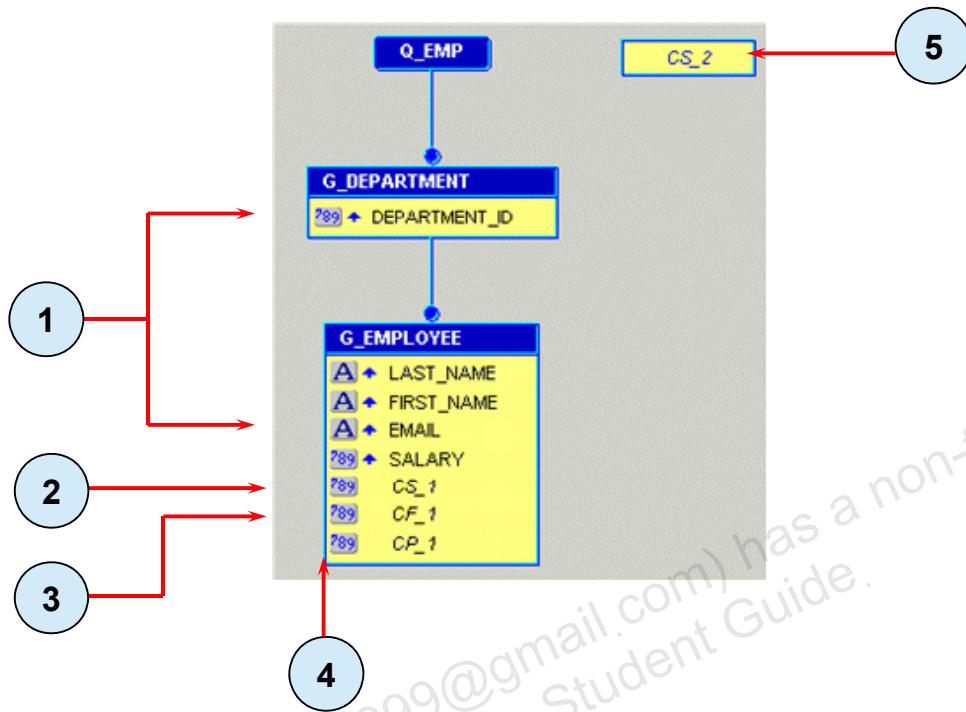
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Overview

When you define a query to extract the data from the database or from another data source and modify the group structure, you might want to modify the Data Model further to perform complex calculations not included in your query.

In this lesson, you learn more about the Data Model objects and how to enhance reports by creating user-defined columns for summaries and calculations.

# Data Model Columns



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Data Model Columns

A Data Model column is a data container. A column defines the type of data, but does not include any formatting information relating to the report output.

There are two main types of columns: those that Reports Builder creates from your data source definition and those that you create yourself.

- **Data source columns:** Reports Builder creates one column for each item in the data source query or definition. Data source columns are directly associated with a column or expression retrieved from the data source as specified by the data source definition. You cannot modify most properties of a data source column.
- **Developer-created columns:** You can create columns, either in a specific group or at report level.

1	Data source columns
2	Summary column
3	Formula column
4	Placeholder column
5	Summary column at report level

## Data Model Columns (continued)

### Developer-Created Columns

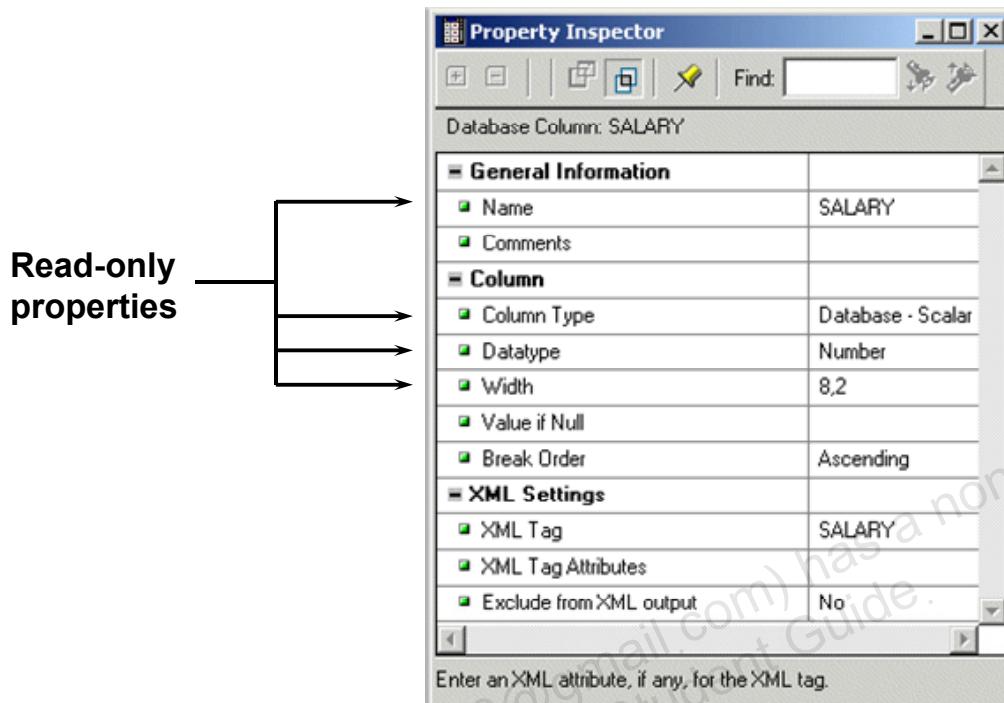
A developer-created column can be one of three types:

- **Summary column:** Summarizes another column and can recalculate for each record in a specified group
- **Formula column:** Uses a formula you have written in PL/SQL to calculate a value from one or more other columns
- **Placeholder column:** Has its value set from another object at run time

Each column that you create in the Data Model has an initial default name, which you should change to a descriptive name.

Column Type	Default Name
Summary	CS_1
Formula	CF_1
Placeholder	CP_1

# Maintaining Data Source Columns



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Maintaining Data Source Columns

Because Reports Builder creates data source columns from the contents of the query statement, there are some restrictions on how you can modify these columns.

To display a column's Property Inspector, as for other report objects, you can use one of several methods:

- Double-click the Column icon in the Object Navigator.
- Double-click the column in the Data Model.
- Select the column and use the right-mouse-button menu.
- Select the column and use the Tools menu.

## Maintaining Data Source Columns (continued)

For data source columns, some of the property settings are not editable in the Property Inspector. You can see the values, but you cannot modify them. These values are dependent on the column name and type in the data source definition.

The read-only properties are:

- Name
- Column Type
- Data type
- Width

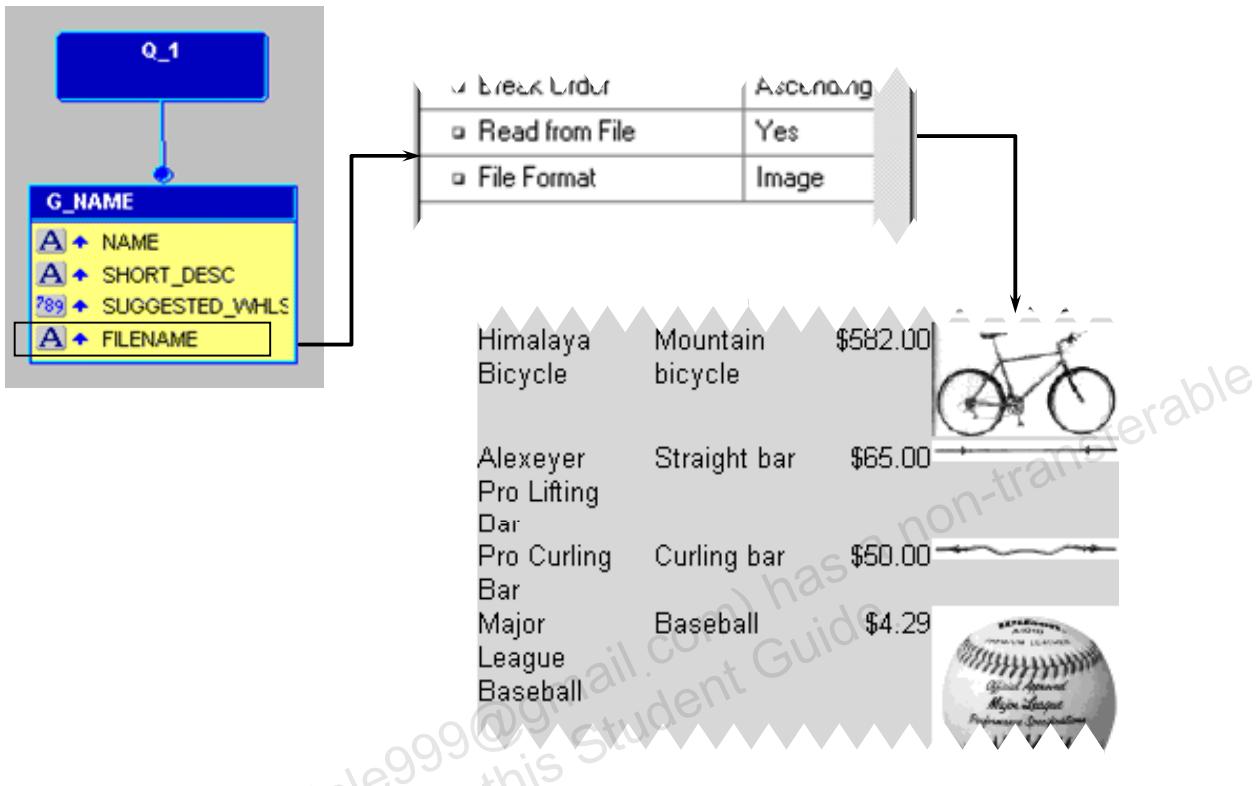
All editable properties, except Value if Null, provide a list of values.

Some properties may or may not appear in the Property Inspector, depending on the column datatype or on the value of another property.

**Note:** The column types *Database - Object* and *Database - Ref* refer to object columns and reference columns in Oracle 10g. Their properties should not be changed.

You cannot delete a data source column object directly from the group. To delete a column, you must remove the corresponding expression from the data source definition. For database columns, you must edit the SELECT statement in the query.

## Producing File Content Output



**ORACLE®**

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Producing File Content Output

You can include in your report the contents of a file for which the filename varies for each record retrieved. The file can contain an image—for example, the picture of each employee or of each product, as shown above.

A *file column* is a data source column that contains the name of an operating system file or a URL. The file can be text, image, or multimedia.

The Read from File property does not exist in the Property Inspector unless the column is of the Database - Scalar type and has a data type of Character.

Reports Builder provides a list of values for the file type: Text, Image, CGM, and Image URL.

**Note:** Text is the only valid format in character mode reports. Video and sound formats are no longer supported in Oracle Reports. Refer to the *Oracle Reports Statement of Direction* on the Oracle Technology Network (<http://otn.oracle.com>) for more information.

## Producing File Content Output (continued)

To create a file column:

1. Open the column Property Inspector.
2. Set Read from File to Yes.
3. Select the appropriate file format from the list of valid types.

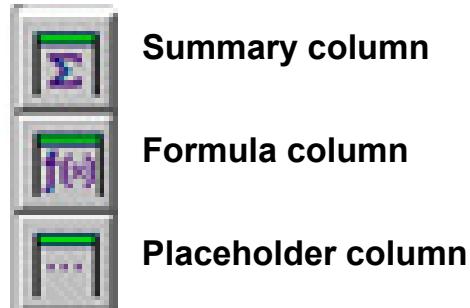
The Report Wizard creates a field for a file column and provides a default size for the displayed field. However, you can resize it either in the Paper Design or the Paper Layout.

For portability, do not prefix the full path to the filename in the column. If you do not prefix a path, Reports Builder can use its file path search order to find the file.

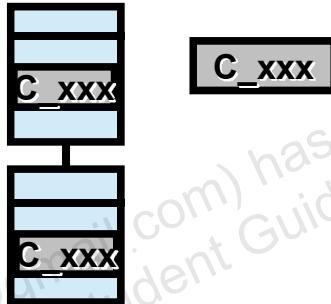
Include the correct paths in the environment variable REPORTS\_PATH.

## Creating a Column

- What type of value?  
Choose the correct column tool



- What frequency?  
Create in a group or at report level



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Creating a Column

There are two important questions to ask yourself before you create a column:

- What type of column do I require? The answer determines which one of the three tools to select.
- How often do I want this column to be displayed? The frequency at which the column value appears in the output determines the group in which you create the column.

## **Creating a Column (continued)**

To create a column:

1. Select the appropriate tool from the toolbar in the Data Model editor. Click inside a group to create the column at the same frequency as other columns in that group.
- Note:** To display the column once for the report, click in an open area of the Data Model editor.
2. Double-click the column to display its Property Inspector.
3. Replace the default column number with a brief but descriptive name. Consider standard naming conventions for different types of columns.
4. Fill in the rest of the settings as appropriate for the type of column you require.

## Creating Summary Columns

- Specific properties:
  - Function
  - Source
  - Reset At
  - Compute At
- Data type depends on Source data type
- Page summaries: Not supported in the wizard



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Creating Summary Columns

A summary column performs a computation on another column's data.

The following properties apply specifically to summary columns:

- Function: The calculation to be performed on the values of the column specified in Source.
- Source: The name of the column whose values are to be summarized. Source is a list of values containing those columns that are valid for the specified Function.
- Reset At: The group at which the summary column value resets to zero.
- Compute At: The group for which a % of Total summary column is computed. Compute At is used only for columns with a function of % of Total.

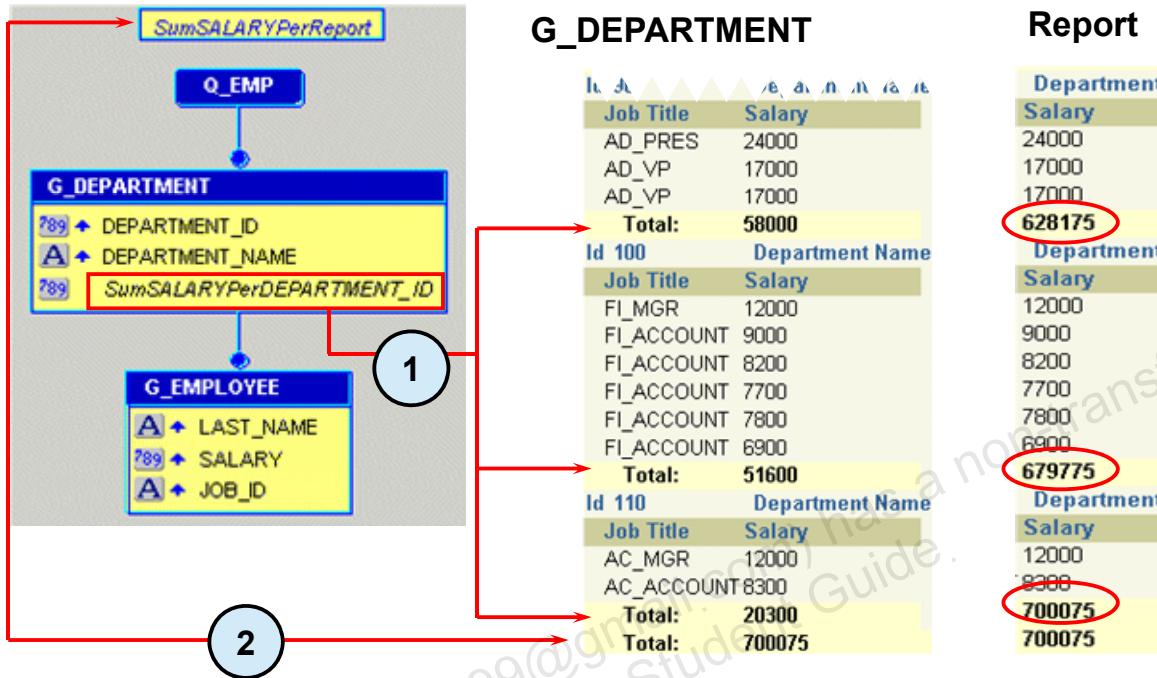
## **Creating Summary Columns (continued)**

The datatype of a summary column depends on the data type of the source of the summary. If you change the data type of the source column, the datatype of the summary also changes.

The Report Wizard does not support page summaries. If you select a page summary in the Field tab of the Report Wizard, an error message appears. Clear the page summary from the Field tabbed page and create the field manually in the Paper Layout. Creating fields in the Paper Layout is described later in the course.

# Displaying Subtotals

Reset At:



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Displaying Subtotals

The control break report output shows two summary columns created by the Report Wizard. One summary shows the total salary for each department, and the other shows the total salary for the whole report.

The wizard creates these summaries at one or more levels above the salary source column.

The Reset At property defines the group at which a summary resets to zero to enable you to calculate a number of subtotals.

The wizard chooses a default reset value for these summaries.

- The department total in G\_DEPARTMENT is reset at G\_DEPARTMENT—that is, values start from zero for each new department record.
- The report total at report level is reset at Report.

To create an accumulating total for the department total, change the Reset At property to Report.

1	Department totals
2	Report total

## Displaying Subtotals (continued)

The Reset At property displays only options that are valid for each summary.

There are two options, Report and Page, that are valid for all summaries. Depending on the level of the summary, some groups are also available in the valid list.

The frequency at which the summary value is displayed depends on the group within which you create the column.

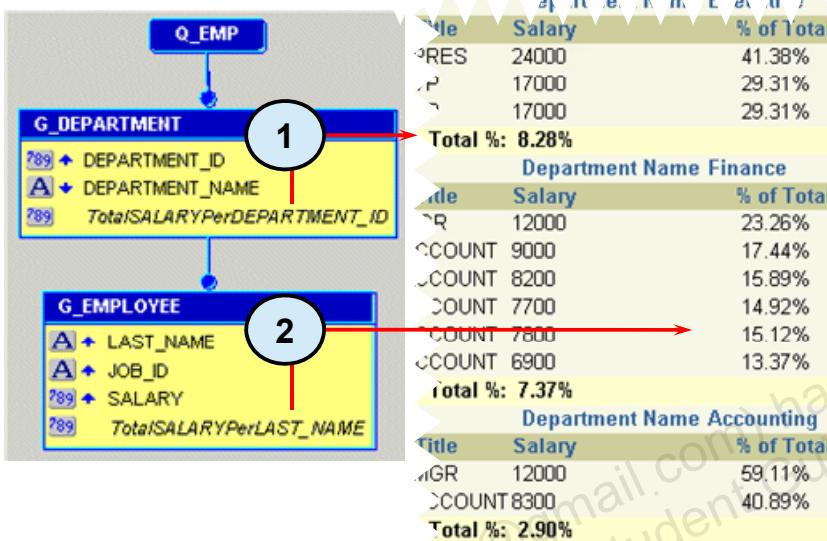
To create your own summaries, use the following rule:

- Discrete Total: Reset At value is the same as the summary group.
- Accumulating Total: Reset At value is a higher group than the summary.

# Displaying Percentages

Reset At: **G\_EMPLOYEE**  
 Compute At: **G\_DEPARTMENT**

**G\_EMPLOYEE**  
**Report**



Department	Salary	% of Total
FINANCE	12000	3.43%
ACCOUNTING	9000	2.43%
SALES	8300	2.43%
RESEARCH	12000	2.28%
WAREHOUSE	7700	1.71%
STOCKISTA	8200	1.29%
CLERK	9000	1.17%
ANALYST	12000	1.10%
PROGRAMMER	8300	1.11%
MANAGER	12000	0.99%
TOTAL		7.37%

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Displaying Percentages

The control break report shows the following two percent of totals created by the Report Wizard:

- Employee's salary as an individual percentage of the department total  
 Notice that this is a column in the G\_EMPLOYEE group. This is the only summary function for which the wizard creates a column in the same group as its source.
- Department total as a percentage of the report total

1	Department percent of total
2	Employee percent of total

## Displaying Percentages (continued)

The Reset At property works the same way that the Sum function does. The default, Reset At G\_EMPLOYEE, results in a discrete value for each employee.

For both summaries in the example on the previous page, the Reset At property is G\_EMPLOYEE. Both percentages reset to zero for each employee record. If you want to create a running percentage that eventually accumulates to 100%, change the Reset At property to reset at a higher level than its source column. For example, to create an accumulating percentage that accumulates for each department and then resets to zero, change Reset At to G\_DEPARTMENT.

The Compute At setting defines the total value that a % of Total summary uses in the percentage calculation. This property is only applicable to % of Total summaries.

The wizard sets the Compute At property to one group above the source column. In the left example opposite, the default is G\_DEPARTMENT. To display the salary as a percentage of the whole report value, change Compute At to Report.

# Resetting Summary Values

Data Model Group	Reset At		
	REPORT	G_DEPARTMENT	G_EMPLOYEE
REPORT	Grand Total	XXXX	XXXX
G_DEPARTMENT	Running Total	Sub Total	XXXX
G_EMPLOYEE	Running Total	Running Total	Record Total



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Resetting Summary Values

The Data Model group defines how often a value is calculated. Therefore, Reports Builder does not allow you to reset a calculated value at a lower level of the hierarchy; that is, you cannot reset a value more often than you calculate it.

The table above identifies which combinations of group and reset options are valid. The XXXX denotes invalid combinations.

## Creating a Formula Column

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

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



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Creating a Formula Column

A formula column performs a user-defined computation on the data of one or more other columns.

The PL/SQL Formula property button opens the Program Unit editor where you enter and edit your code.

The formula is a PL/SQL function that returns a single value. It must return a value of the same type as specified in the data type property. The following code is a simple example that calls another function, passing the :salary value as an argument, and returns the result in the SALCOMM formula column.

```
function salcomm return number is
begin
    return(my_function (:salary));
end;
```

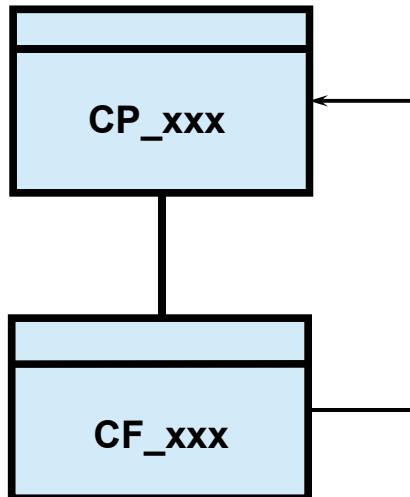
Any columns referenced by the formula column are calculated first.

## **Creating a Formula Column (continued)**

**Note:** The code displayed is a simple example to illustrate the syntax. The MY\_FUNCTION function might be a local program unit at report level, a program unit in an attached PL/SQL library, or a stored program unit.

Consider using a formula column only if you cannot include your calculation in the query statement.

## Creating a Placeholder Column



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

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Creating a Placeholder Column

A placeholder column is an empty container at design time. The placeholder can hold a value at run time that has been calculated and placed into it by PL/SQL code from another object.

You can set the value of a placeholder column in the following places:

- A Before Report trigger
- A report-level formula column (if the placeholder column is at report level)
- A formula column in the placeholder group or a group below it

## **Creating a Placeholder Column (continued)**

Using placeholder columns, you can:

- Populate multiple columns from one piece of code. You can calculate several values in one block of PL/SQL code in a formula column and assign each value to a different placeholder column. Thus, you create and maintain only one program unit instead of many.
- Store a temporary value for future reference. For example, store the current maximum salary as records are retrieved.

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

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

# Populating a Placeholder Column

The diagram illustrates the Oracle Report structure and the corresponding database schema. The report displays maximum salaries and earners for three departments (10, 20, 30). A placeholder column 'CP\_MAX\_EARNER' is populated from the 'G\_EMPLOYEE' table using a formula.

**Report Data:**

- Dept: 10 Name: Administration**

Last Name	Title	Salary
Whalen	AD_ASST	\$4,400
<b>Maximum:</b>		<b>\$4,400</b>
- Dept: 20 Name: Marketing**

Last Name	Title	Salary
Hartstein	MK_MAN	\$13,000
Fay	MK_REP	\$6,000
<b>Maximum:</b>		<b>\$13,000</b>
- Dept: 30 Name: Purchasing**

Last Name	Title	Salary
Raphaely	PU_MAN	\$11,000
Khoo	PU_CLERK	\$3,100
Baida	PU_CLERK	\$2,900
Colmenares	PU_CLERK	\$2,500
Himuro	PU_CLERK	\$2,600
Tobias	PU_CLERK	\$2,800
<b>Maximum:</b>		<b>\$11,000</b>

**Placeholder Column:** CP\_MAX\_EARNER

**Database Schema:**

```

    graph TD
        Q_EMP --> G_DEPARTMENT
        Q_EMP --> G_EMPLOYEE
        G_DEPARTMENT --> G_EMPLOYEE
        G_EMPLOYEE --> CS_MAX_SALARY
        G_EMPLOYEE --> CP_MAX_EARNER
    
```

**Tables:**

- G\_DEPARTMENT**
  - DEPARTMENT\_ID
  - DEPARTMENT\_NAME
  - MaxSALARYPerDEPARTMENT\_ID
- G\_EMPLOYEE**
  - LAST\_NAME
  - JOB\_ID
  - SALARY
  - CP\_CALC\_MAX (highlighted with a red circle)

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Populating a Placeholder Column

The sample report shown is a report of all employees. The aim of the report is to:

- Calculate and temporarily store the name of the employee who earns the highest salary in the company
- Display the highest earner and the maximum salary once at the beginning of the report

For this report, you need to create the following columns:

- A summary to show the maximum salary for the company
- A placeholder to contain the highest earner's name at run time
- A formula to:
  - Compare each employee salary with the maximum salary
  - Populate the placeholder with the employee name if salary equals maximum salary

## Populating a Placeholder Column (continued)

To create the columns:

1. Create a summary column CS\_MAX\_SALARY at report level (outside all groups). Data type: Number; Width: 10; Function: Maximum; Source: Salary; Reset At: Report
2. Create a placeholder column CP\_MAX\_EARNER at report level (outside all groups). Data type: Character; Width: 80
3. Create a formula column CF\_CALC\_MAX in the G\_EMP group. Data type: Number; Width: 10 (these are the default values)
4. Open the Program Unit editor from the PL/SQL Formula property in CF\_CALC\_MAX.

Create a function:

```
function CF_CALC_MAXFormula return Number is
begin
    if :salary = :CS_MAX_SALARY then
        :CP_MAX_EARNER := (:LAST_NAME || ' in
                            Department ' || :DEPARTMENT_ID || '--'
                            || :DEPARTMENT_NAME);
    end if;
    return(0);
end;
```

# Summary

In this lesson, you should have learned how to:

- Identify column types and their uses
- Create Summary columns
- Define valid summary levels in a report
- Specify column properties based on required output:
  - Reset At: Resets to zero
  - Compute At: % of Total only
- Create Formula columns to return values
- Create Placeholder columns to hold values



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Summary

### Column Types

- A data source column exists for each column or expression in the query or data definition.
- Create summary columns for computations and summaries.
- Create formula columns to perform a calculation or to populate a placeholder column.
- Create a placeholder column to provide an empty container that is populated from another object at run time.

### Summary Levels

- Report
- Page (not supported by the Report Wizard)
- Groups within the report

### Reset At

The level at which to reset the summary back to zero

### Compute At

Valid only for summary columns with a function of % of Total

## Practice 8 Overview

- Creating a new report with summaries
- Adding summary calculations to an existing report
- Creating a new report with ranking summary columns
- Adding placeholders for highest and lowest values



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Practice Session Overview: Lesson 8

This practice session contains:

- Creating a new report with summaries
- Opening an existing report to add summary calculations to the columns
- Creating a new report with ranking summary columns
- Opening an existing report to add placeholders for the highest and lowest values in the report

In your reports, you might need to include additional columns with summary functions—or even add summaries, such as break totals and report totals. This practice provides the opportunity for you to create summaries and additional columns with functionality that cannot be achieved using SQL alone.

## Practice Session: Lesson 8

1. Create a new report using the Report Wizard. Create both a Web and paper layout and select a Group Above report style.
  - a. Select the following columns in the Query Builder:

Table	Columns
ORDERS	order_id, order_date
ORDER_ITEMS	line_item_id, product_id, unit_price, quantity

**Hint:** Manually edit the query created by Query Builder to modify the display of order\_date using the TO\_CHAR function and specify an alias:

TO\_CHAR(order\_date, 'dd-Mon-yyyy') Order\_Date

- b. Modify the query to include line\_total as a calculation of unit\_price\*quantity.
- c. Group the data on Order\_ID and Order\_Date and display all the fields.
- d. Create the following summaries. Can you use the Report Wizard to create the summaries?

Column	Summary
line_total	sum
line_total	% Total

- e. Specify the following labels and widths.

Column	Label	Width
order_id	Ord. No.	4
order_date	Order Date	10
line_item_id	No.	2
product_id	Product No.	5
unit_price	Price	4
quantity	Quantity	4
line_total	Total	4

- f. Use the Gray template file and finish the Wizard.
- g. In the Data Model, alter the position of columns in the G\_ORDER\_ID group so that ORDER\_DATE is above ORDER\_ID, and change the break order on ORDER\_DATE so that it is sorted in descending order.
- h. Run both the paper and Web layouts to test the report (do not reenter the Wizard).
- i. Save the report as p10q1.rdf and close it.

## Practice Session: Lesson 8 (continued)

2. Open the p9q3.jsp report.
  - a. Create columns to show the percentage of the total per sales representative for PROD\_TOTAL and CUST\_TOTAL.
  - b. Update the layout. Make all the new summaries four characters wide.
  - c. Save the report as p10q2.jsp.
  - d. Run the Web layout.
3. Create a new report to list employees by department ranked by their salaries. Create both a Web and paper layout and select the Group Left report style.
  - a. Import the query from p10q3.sql. Select the group field as DEPARTMENT\_NAME and display the following data.

Table	Column	Label	Width
departments	department_name	Department	10
employees	employee_id	ID	2
	first_name	First Name	6
	last_name	Last Name	6
	salary	Salary	8

Include a summary of the salary column in the report.

- b. Use the Beige template and finish the Wizard.
- c. Modify the report to list employees by salary, beginning with the highest paid in each department.
- d. To display the ranking, add a column call RANK in the group G\_EMPLOYEE\_ID.
- e. Use the Report Wizard to display the rank, giving it a width of 2.
- f. Run the paper layout to test.
- g. Save the report as p10q3.rdf.

## Practice Session: Lesson 8 (continued)

4. Open the p10q1.rdf report to display the highest and lowest orders at the end of the report.
  - a. In the Data Model, add the columns necessary to list the maximum order total and the corresponding ORDER\_ID, as well as the minimum order total and the corresponding ORDER\_ID.

**Hint:** You need one formula column and two summaries and two placeholders. You can copy the code for the formula column from the file p10q4.txt.
  - b. Use the Report Wizard to add the new columns to the layout. Display the maximum order total, the maximum ORDER\_ID, the minimum order total, and the minimum ORDER\_ID.
  - c. Run the paper layout to test. Scroll down to the last page to see the new columns.
  - d. Save the report as p10q4.rdf and close it.

# Enhancing Reports Using the Paper Layout

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Objectives

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

- Identify the paper report sections
- Design multipanel reports
- Mail a report
- Distribute a report to various destinations
- Describe the layout objects and relationships
- Modify an existing paper report layout
- Create variable length lines



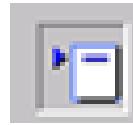
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Overview

You can fulfill many of your paper report requirements simply by making modifications in the Paper Design view. However, for more complex paper reports you need to modify or create objects in the Paper Layout editor.

The layout can be a complex area to view and modify. This lesson aims to give you an appreciation of the layout sections, as well as the objects that you see and create using object tools in the toolbar.

# Viewing the Paper Layout



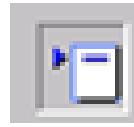
Header  
section

Compensation  
Report

ORACLE®

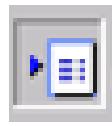
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Viewing the Paper Layout



**Header  
section**

Compensation  
Report



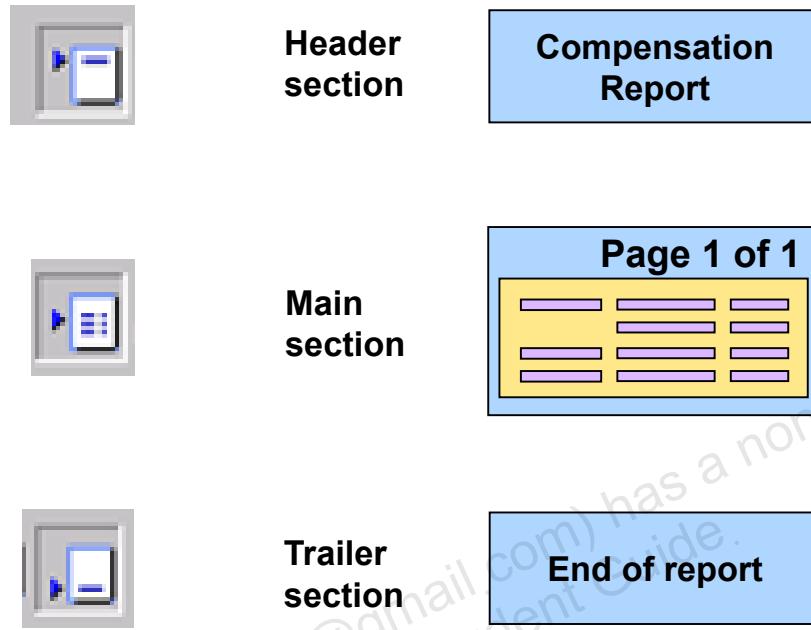
**Main  
section**

Page 1 of 1

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Viewing the Paper Layout



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Viewing the Paper Layout

The Report Editor Paper Layout view enables you to define and modify the layout objects for a paper report. Layout objects define the report's appearance and are displayed in the Paper Layout view.

### The Paper Layout Sections

The Paper Layout editor allows you to modify the report layout in three sections: header, main, and trailer. Each section has its own body and margin area. Report sectioning allows you to define multiple layouts for the same data model, producing output in a number of styles. For example, a single report can include an executive summary in the header section, and a detailed breakdown in the main section.

When you enter the Paper Layout editor, you see the body region of the main section of your report. To change the section, choose the Main Section, Header Section, or Trailer Section tool, or choose View > Layout Section > Header, Main, or Trailer.

## **Viewing the Paper Layout (continued)**

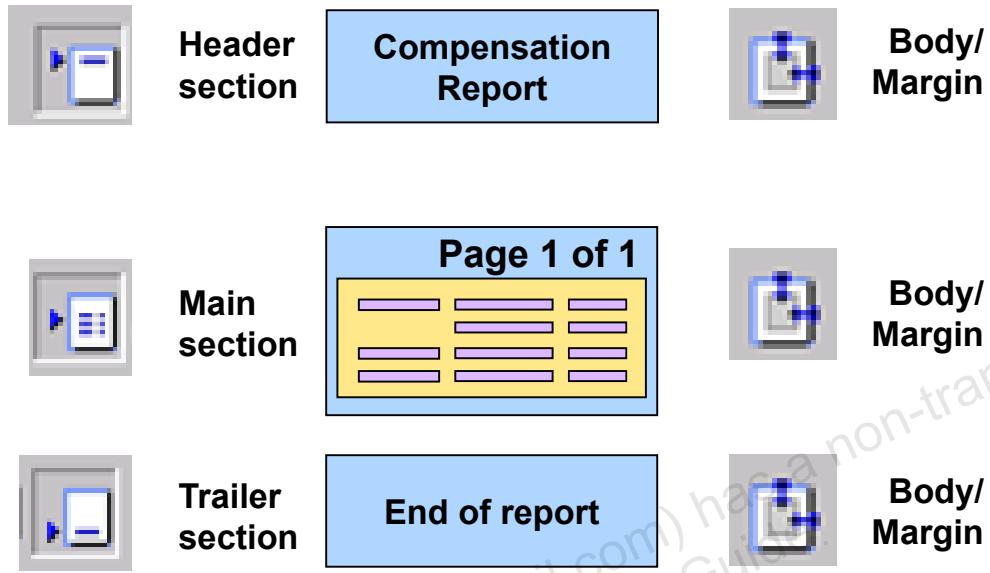
**Header Section:** The header section consists of one or more pages that appear once, on a separate set of pages, as the first part of the report.

**Main Section:** By default, the Report Wizard creates the paper layout in the Main Section of a report, positioned between the header and trailer pages.

**Trailer Section:** The trailer section consists of one or more pages that appear once, on a separate set of pages, as the last part of the report.

**Note:** In the Object Navigator window, the three report sections are listed under the Paper Layout node.

# Viewing the Section Areas



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Viewing the Section Areas

The Paper Layout editor allows you to modify the body and margin report areas in one of three sections at any time. This enables you to separate the formatting of margins from the body of your report pages in any section. Each physical page consists of a body and margin area.

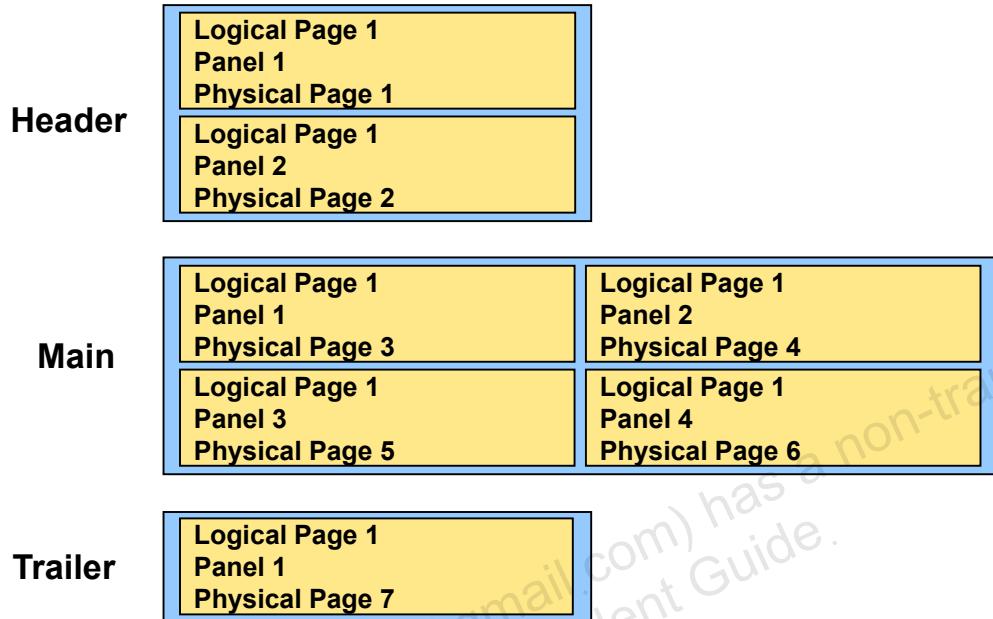
- **Body:** The body area of a section can contain text, graphics, data, computations, and summaries.
- **Margin:** The margin area can contain text, graphics, page numbers, page totals, and grand totals.

When you choose the margin view, the body also remains visible, but read-only, in the editor. The body appears within a black rectangle that defines the size of margin and body. You cannot modify objects in the body while working in the margin view.

To adjust the size of the margin, choose the margin view, select the black margin rectangle, click on a select handle, and drag to resize.

Modify margins in your template definition to apply the same dimensions to many reports, or to apply the same dimensions to several sections of the same report.

# Designing Multipanel Reports



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Designing Multipanel Reports

Reports Builder enables you to design large, or multipanel, report layouts by maintaining two different definitions of a page.

- A *physical page* represents the actual paper page size.
- A *logical page* represents a conceptual page on which you can design the layout of your report.

Reports Builder enables you to design multipanel reports where the logical design of the report output is wider than the physical (printer) page. A report might need a layout width of 16 inches that can be printed on two physical printer pages, each 8 inches wide, and then placed alongside each other.

The unit of measurement is a property of the report object itself.

## Designing Multipanel Reports (continued)

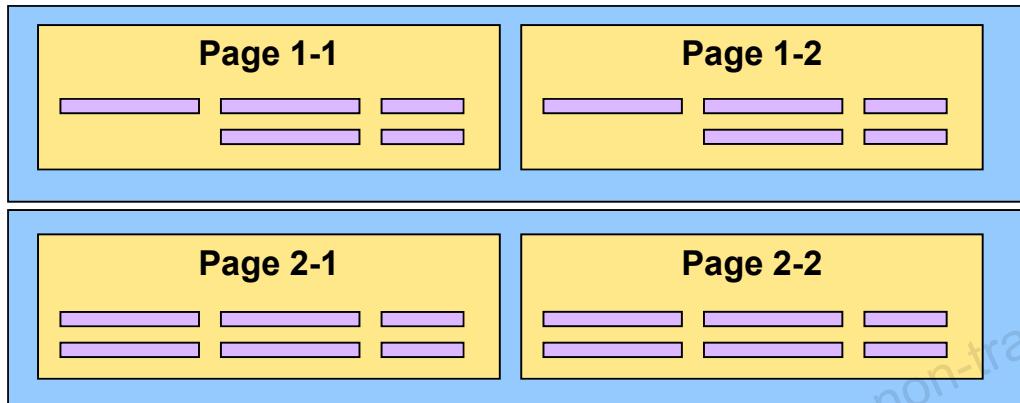
To define dimensions for a section of the report:

1. Select the Header Section, Main Section, or Trailer Section node in the Object Navigator.
2. Open the Property Inspector.
3. Specify physical page size: Width and Height properties.
4. Specify logical page size: Horizontal Panels per Page/Vertical Panels per Page properties.  
Define the logical dimensions in multiples of the physical page width and height.
5. Define the page orientation as required.

**Example:** In the diagram opposite, the layout is twice the width of the physical page. For example, to output a logical design of 16 inches across 8-inch pages, the settings are:

- Unit of Measurement: Inch
- Physical Page Height: 8
- Horizontal Panels per Page: 2
- Vertical Panels per Page: 1

# Printing Multipanel Reports



**Logical horizontal panels = 2**

ORACLE

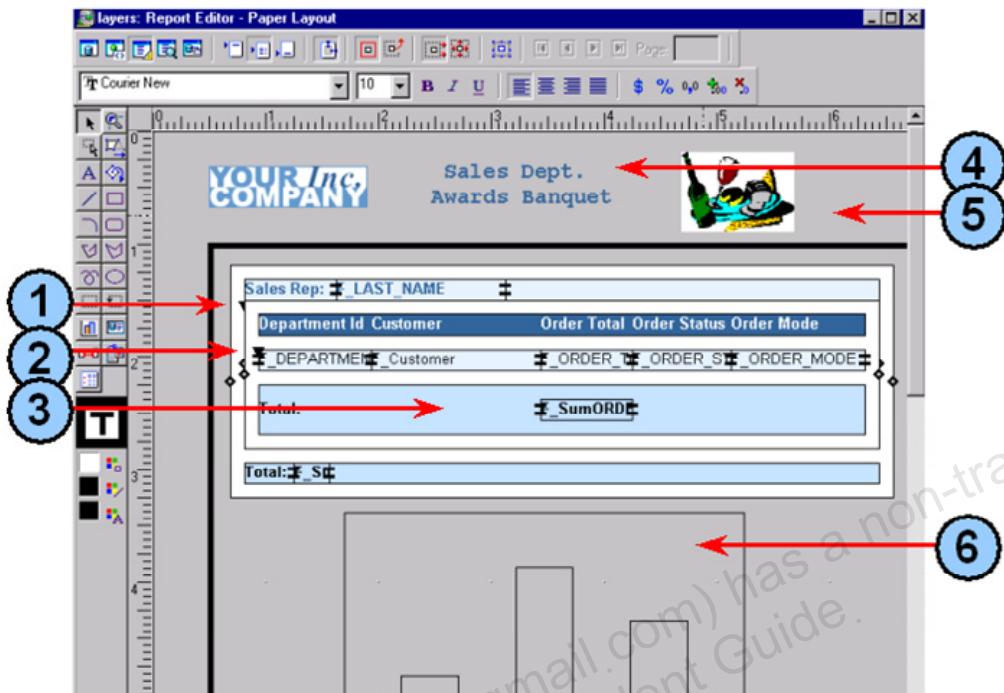
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Printing Multipanel Reports

The default order in which panels are printed is across/down. That means that horizontal panels are printed before vertical panels. To change this order, modify the Panel Print Order property of the report object.

1. Select the report node in the Object Navigator.
2. Open the Property Inspector.
3. Specify the Panel Print Order property.

# Different Objects in the Paper Layout



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Different Objects in the Paper Layout

The Report Editor Paper Layout view supports a large number of different objects. The Paper Layout objects and their properties are represented symbolically to distinguish their types and relationships.

- **Frame:** This object encloses other layout objects and controls formatting, frequency, and positioning of several objects simultaneously. Created by default or by the developer
- **Repeating Frame:** This object displays rows of data that are retrieved for a group. Created by default or by the developer

1	Frame
2	Repeating frame
3	Field
4	Boilerplate
5	Imported boilerplate image
6	Graph

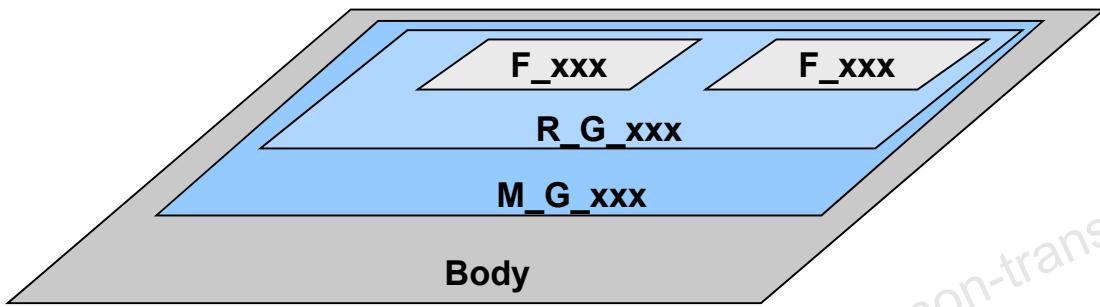
## Different Objects in the Paper Layout (continued)

- **Field:** A field defines the appearance of a column's data. It may contain character, number, date, graphics, image, or sound data. Created by default or by the developer
- **Boilerplate:** Hard-coded text or graphics that appears in a report every time it is run. Created by default or by the developer
- **External Boilerplate:** Text or graphics that appears in a report every time it is run, but read from a file. Created by the developer
- **Graph:** An object that allows data to be represented in a graphical way

Every object has a property sheet. You can invoke the Property Inspector in a number of ways:

- Double-click the object
- Highlight the object and use the right mouse button menu to choose Property Inspector
- Highlight the object and choose Tools > Property Inspector

# The Paper Layout Layers



**Layers of a tabular report**

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## The Paper Layout Layers

All layout objects exist in a hierarchy. Each object is contained inside and on a layer above another object. The hierarchy is very important in the processing of report objects. If you destroy the correct layering, you receive error or warning messages, and the report might not run.

The bottom layer of a paper report is effectively the page itself.

### Tabular Report

To explain the concept of layers, the exploded diagram shows the objects in a simple tabular report. From bottom up, the layers are:

- Body
- M\_G\_xxx: Group frame encloses other objects and controls the format, frequency, and position of several objects simultaneously
- R\_G\_xxx: Repeating frame is displayed for each row of data that is retrieved for a group
- F\_xxx: Fields define the appearance of columns; can contain character, number, and date for each column in the data source

In addition, the tabular style creates a boilerplate text object, B\_xxx, for each field label; these objects occur outside the repeating frame, not once for each record.

## **The Paper Layout Layers (continued)**

### **Group Reports**

Group Left and Group Above reports create two repeating frames; the detail group frame is nested inside the master repeating frame to produce a detail tabular listing for each master record.

# Avoiding Layout Errors

- Confine Mode



- Flex Mode



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

ORACLE

## Avoiding Layout Errors

To avoid common hierarchy errors in your report layout, always work in the Paper Layout using the Confine and Flex modes.

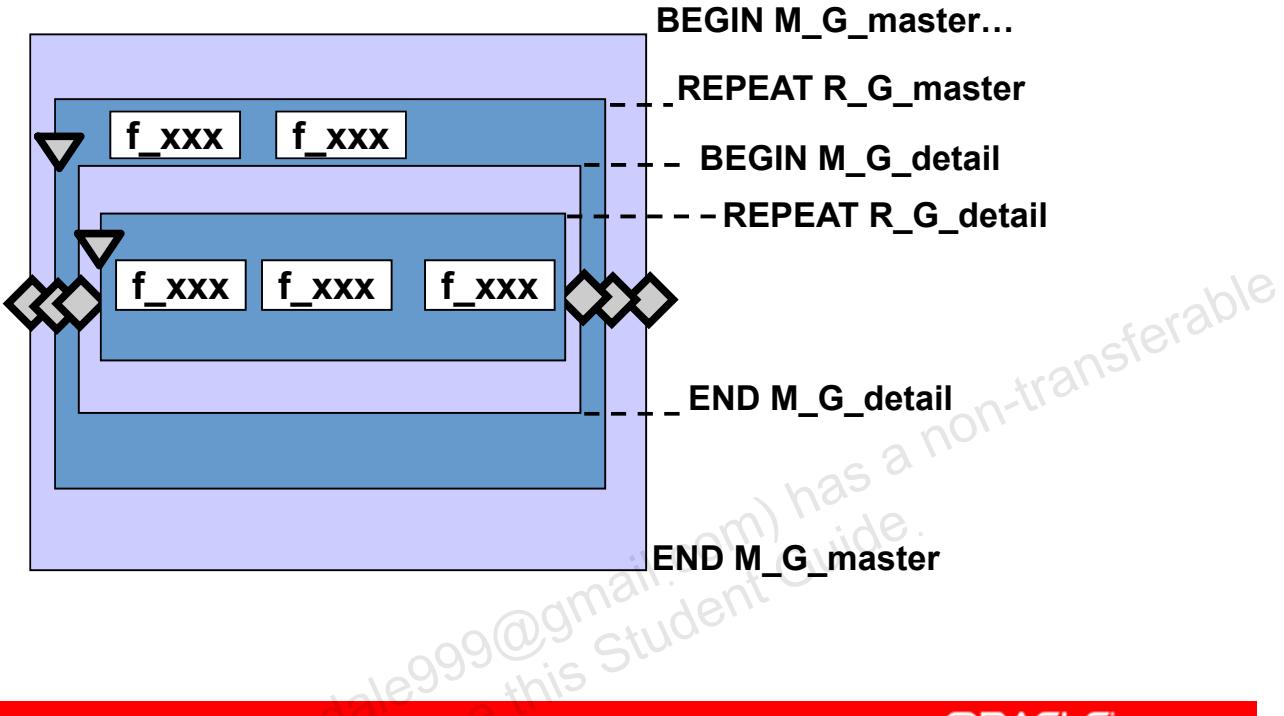
- Confine Mode

- Work with Confine Mode enabled (on) to avoid accidentally moving one object outside or below its correct enclosing object.
- Work with Confine Mode disabled (off) when you want to move one object outside its enclosing object.

## Avoiding Layout Errors (continued)

- Flex Mode
  - Work with Flex Mode enabled (on) to adjust all affected enclosing objects when you move or resize one object; the whole layout flexes to accommodate your changes.
  - Work with Flex Mode disabled (off) when you want to move or resize an individual object without moving or resizing other objects.

# Report Processing



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Report Processing

To understand how the Paper Layout maps to the output that you see in the Paper Design view, consider the layout as a 3GL program, where the group frame is a WHILE loop and the repeating frame is a block.

In a two-level group report, the report processes a master record, including all details that relate to that master record, and then processes the next master.

```
BEGIN M_G_master
  while M_G_master has records to process
    REPEAT R_G_master
      BEGIN M_G_detail
        while M_G_detail has records to process
          BEGIN R_G_detail
            end R_G_detail;
          END M_G_detail;
        END R_G_master;
      END M_G_master;
```

## **Report Processing (continued)**

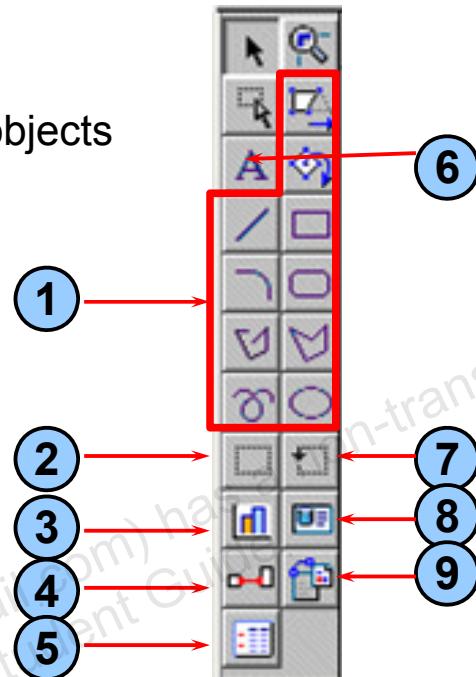
Layout objects often show symbols and icons that indicate their characteristics. The diagram opposite shows two of these symbols:

- Black arrowhead: Indicates a repeating frame object
- Diamond: Indicates that the object is a variable size; it can increase or decrease in size depending on the contents

# Creating Layout Objects

The tool palette contains:

- Standard GUI drawing tools
- Frame, repeating frame, field objects
- Other layout objects



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Creating Layout Objects

When you create or modify your paper report using the Report Wizard, Reports Builder automatically creates many objects, such as the frame, repeating frame, fields, and labels for a tabular report style.

The tool palette provides tools for you to create these and other types of objects. The tools available depend on the view currently in the Paper Layout.

1	Drawing objects	6	Text
2	Frame	7	Repeating frame
3	Graph	8	Field
4	Anchor	9	File Link
5	Report Block		

## **Creating Layout Objects (continued)**

When you create a layout object, the default name is an alphabetic character that indicates the object type, followed by a number. Rename your layout objects meaningfully to help you in developing and maintaining the report.

<b>Object Type</b>	<b>Default Name</b>	<b>Description</b>
Graph	CT_	A display object that represents data as a graph, such as a pie or bar chart
Boilerplate	B_	Constant text or an image that appears in a report every time it is run, and that is not dynamic; example: the Report Wizard creates column labels and template objects
File Link	B_	Constant text or an image that appears in a report every time it is run, but is read from a file on disk
Anchor	None	A layout object that anchors the position of one object relative to another; create an anchor to override the default position

### **Technical Note**

The ability to embed an OLE2 object in a report is obsolete in Oracle Reports 10g. While existing applications using OLE2 objects should continue to run without modification, this functionality can be mimicked by using mime types with associated plug-ins and hyperlinks.

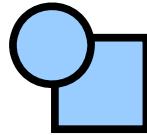
# Paper Layout Tools



- Pin a tool: Double-click



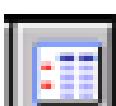
- Shift key for constrained objects



- Magnify



- Frame Select



- Report Block

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

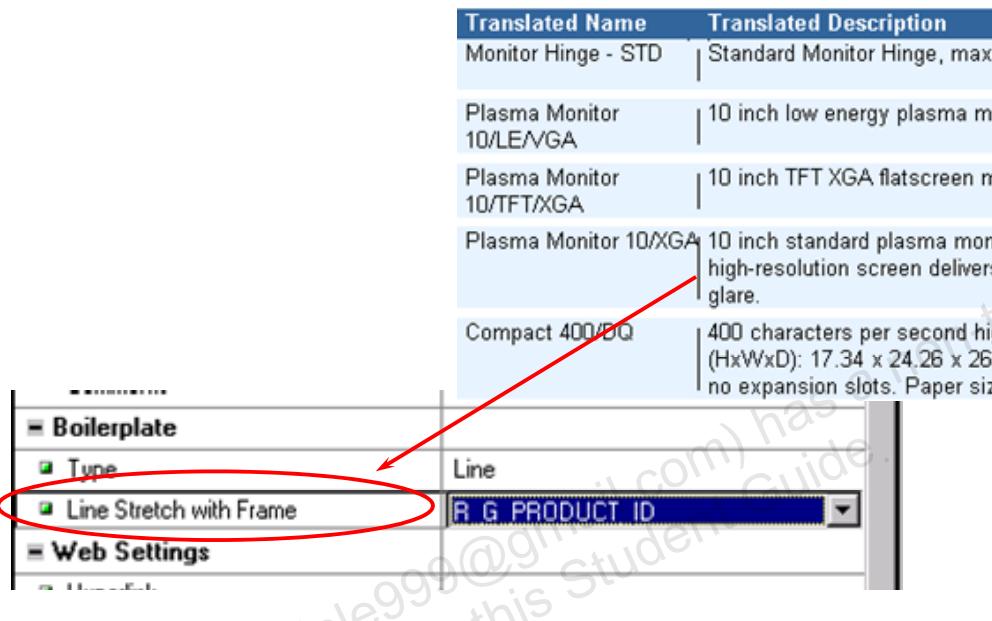
## The Paper Layout Tools

In addition to the object tools in the tool palette, the Paper Layout provides some specific tools to help you modify layout objects.

- **Pinning a tool:** If you want to create several objects of the same type, double-click the drawing tool. A small red pin appears on the tool icon, and the tool remains active until you click the tool again or choose a different tool.
- **Drawing constrained objects:** To create a constrained object, such as a perfect circle or square, hold down the constrain [Shift] key when drawing the object.  
To resize an object to scale, hold down the constrain key when resizing the object.
- **Magnifying the layout:** Select the magnify tool when you want to focus on one part of a large report. The Magnify tool centers the layout area on the point that you choose and enlarges objects to twice their size.  
Hold down the constrain key when using the magnify tool to reduce objects by half. There are also zoom in and zoom out buttons in the toolbar.
- **Selecting all objects in a frame:** Choose the Frame Select tool and then click a frame or repeating frame in the layout. This selects the frame and all objects inside it.
- **Report Block:** The Report Block tool invokes the Report Block Wizard to enable you to create additional layouts in your report without deleting existing objects, and to let you mix multiple report styles in the same section of a report.

# Creating Variable Length Lines

Flex lines adjust to fit variable frames and repeating frames.



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Creating Variable Length Lines

The line drawing tool enables you to draw straight vertical lines between columns of a report. However, these lines are a fixed length, whereas the data in your report might be variable. Reports Builder enables you to specify that a line varies in length or width to fill a chosen frame.

### Separating Vertical Columns in a Tabular Report

In the report above, the translated description has a variable number of characters; sometimes the characters require a single line field, sometimes more. The field expands vertically if necessary at run time. The repeating frame also expands to contain the field.

If you draw a vertical line inside the repeating frame, the line is not long enough to span the extra line of characters at run time.

- To make the line stretch to fit the repeating frame, open the Property Inspector for the line object. Select Line Stretch with Frame and select the repeating frame you require, such as the repeating frame, R\_xxx.

## **Creating Variable Length Lines (continued)**

- To make the line stretch to fit the group frame, draw the line from the top of the column labels through the repeating frame. Change the property Line Stretch with Frame to the group frame, M\_xxx\_GRPFR.

## **Paper Design Versus Paper Layout**

You can create and modify a line in the Paper Design or in the Paper Layout. There are advantages and disadvantages to both:

- Paper Design: It is easy to create a line and change the property. You can easily see the line stretching with the data, but you cannot move the line outside its enclosing object, because you cannot disable Confine Mode.
- Paper Layout: You can disable Confine Mode to change the position of the line, but you cannot see the effect of Line Stretch with Frame until you view the report in the Paper Design.

# Summary

In this lesson, you should have learned how to modify the Paper Layout using:

- Report sections
- Layout objects and tools
- Variable length lines



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Summary

In this lesson, you learned to modify a paper report layout in the Paper Layout.

### Modifying the Paper Layout

- There are three report sections, Header, Main and Trailer. Each of them has two areas, Body and Margin.
- You can use the tool palette to create standard drawing objects and report objects. The tool palette also provides some useful editing tools, such as the Frame Select tool.
- You can create many additional layouts in the same report by using the Report Block tool.

### Example Reports

This lesson discussed solutions to the following report requirements:

- A report with sections
- Distributing reports such that each user only receives information that is relevant and appropriate
- Creating variable length lines

## Practice 9 Overview

- Using report sections
- Adding a repeating frame to the header page
- Adding flexible lines
- Distributing a report



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Practice Session Overview: Lesson 9

This practice session contains:

- Opening an existing report to add header and trailer pages
- Distributing a report to various destinations
- Modifying an existing report to add a repeating frame to the header section displaying order information
- Opening an existing report and adding flexible lines

In this practice session, you use reports created in earlier practice sessions. The aim is to familiarize you with the different layout sections and regions of a report. You also add flexible lines to a frame.

## Practice Session: Lesson 9

1. Open report p10q1.rdf. Modify the report as follows:
  - a. Create a header section.  
Display the monitor.jpg file as fixed boilerplate in the Body region.  
Add a rectangle boilerplate object. Give it fill color.  
Place the image on top of the rectangle. Center the two objects.  
**Hint:** Use the Layout option from the menu.  
Include today's date in the top right portion of the margin.
  - b. Add a report title in the body region.  
Use Arial bold as the font and 20 as the size.
  - c. Add a trailer section that displays today's date with the text "End of Report" on one line, and "Report run on:" on the second line in the body region. Select a larger font.  
**Hint:** Copy and paste the date from the Margin Area of the header section of the report.
  - d. Save the report as p11q1.rdf.
  - e. Run the Paper Layout. Run the Web layout. What is different?
2. Open report p10q1.rdf.
  - a. Modify the report to add a header section that lists all order numbers, order dates, and order totals in the body region.
  - b. Add "Executive Summary" as boilerplate text in the margin region of the header section.
  - c. Set the distribution for the header section to an HTMLCSS file.
  - d. Set the distribution for the main section to a PDF file.
  - e. Run and test the report using the Distribute facility in Reports Builder.
  - f. Look at the output generated for both file formats.
  - g. Save the report as p11q2.rdf and close it.

## Practice Session: Lesson 9 (continued)

- c. Define the distribution for the report. Generate a PDF file for each warehouse, using the warehouse name as a unique identifier for each PDF file, such as `rptBeijing.pdf`.
  - d. Activate the distribution.
  - e. Save the report.
  - f. Look at the output generated for several of the warehouses.
4. Open report `p11q4.rdf`.
    - a. Use the Report Wizard to increase the width of `PRODUCT_DESCRIPTION` to 30.
    - b. Add a flexible line vertically between the `PRODUCT_NAME` and `PRODUCT_DESCRIPTION` fields.
    - c. Run the Paper Layout to test.
    - d. Save the report as `p11q4.rdf`.
  5. Open and run report `p2q10.rdf`.
    - a. Using the Paper Design view, compare the output with `s11q5a.rdf`. What is different? Change the report so that it looks like `s11q5a.rdf`.
    - b. Fully expand the Paper Layout node for `p2q10.rdf`.
    - c. Open the Paper Layout and move the window to the right. Make sure it doesn't obscure the Object Navigator window.
    - d. Move the `F_EMPLOYEE` field down 2 inches. It should not be enclosed by any frame.
    - e. Resize the `R_G_EMPLOYEE` frame so that it only encloses the `F_SALARY` and `F_JOB_ID` fields.
    - f. Resize the `M_G_EMPLOYEE_GRPFR` frame so that it encloses the `R_G_EMPLOYEE` repeating frame.
    - g. Move the `F_DEPARTMENT_NAME` and `F_DEPARTMENT_ID` fields to the right.
    - h. Create a new repeating frame in the free space. Set the Line color attribute to No line.
    - i. Link the repeating frame to the `G_EMPLOYEE` group.
    - j. Move the `F_EMPLOYEE` field inside the repeating frame `R_1`.
    - k. Save the report as `p11q5a.rdf` and run the paper layout. What happens?
    - l. Fix the layout hierarchy error.
    - m. Rearrange the column headers
    - n. Run the paper layout.
    - o. Close the report. Save as `p11q5a.rdf`.

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

Shital jadhav (shitaljagdale99@gmail.com) has a non-transferable  
license to use this Student Guide.

# 10

## Controlling the Paper Layout: Common Properties

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Objectives

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

- Identify common paper layout properties
- Modify common paper layout properties
- Explain the use of format triggers
- Modify Web Settings properties



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Overview

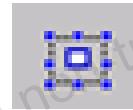
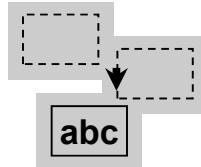
All Reports Builder objects have properties that provide default settings, which in many cases are suitable for your report. However, you can modify these property values; for example, to change the pagination of objects in the report output.

In this lesson, you learn about properties that are common to all paper layout objects. These are properties that enable you to control the size, placement, and pagination of objects in the paper report output, as well as properties for Web support of paper reports.

# Modifying Paper Layout Object Properties

Accessing a Property Inspector:

- Object Navigator:
  - Select object
  - Double-click node icon
- Paper Design or Paper Layout view:
  - Use Select Parent Frame tool
  - Select Tools > Property Inspector



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Modifying Paper Layout Object Properties

You have already seen that you can access the Property Inspector for any object in several different ways. For layout objects, you can access the same properties by selecting the object in one of the following windows:

- Object Navigator
- Paper Layout
- Paper Design

### Selecting an Object in Lower Layers

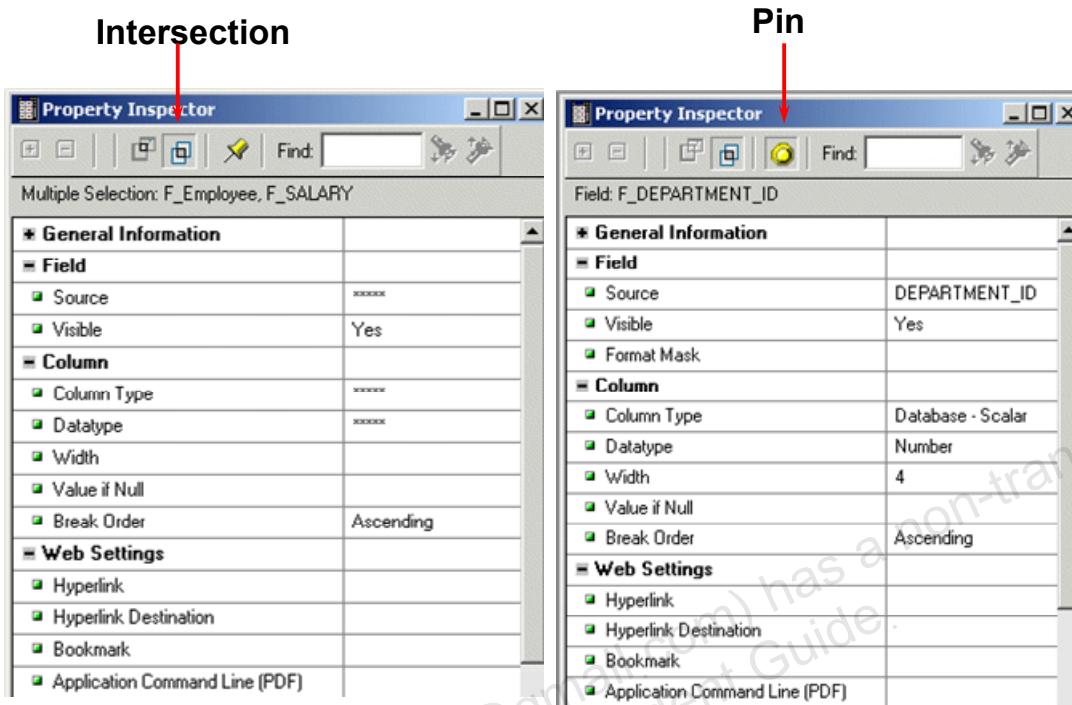
Because there are several layers in even a simple report, it is sometimes difficult to select the correct object in the Paper Layout or Paper Design view when other objects overlay the object you want.

## **Modifying Paper Layout Object Properties (continued)**

- Using the Object Navigator:
  - Select the object directly in the Object Navigator, where you can easily see the name and type of object.
  - To open the Property Inspector, click the node icon to the left of the object name, or use the right-mouse-button menu.
- Using the Select Parent Frame tool:
  - Select the top object in the Paper Layout or Paper Design. Choose Select Parent Frame to select the enclosing object on the layer below. For example, select a field, choose Select Parent Frame to select the repeating frame, and choose Select Parent Frame again to select the group frame.
  - To open the Property Inspector, select Tools > Property Inspector or use the Object Navigator.

Do not double-click the object in the Paper Design, because you will probably select the wrong object on the top layer.

# Comparing Properties



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Comparing Properties

To compare the properties of two objects:

1. Select one or multiple objects and open the Property Inspector.  
**Note:** Selecting multiple objects results in an *intersection* of properties.
2. Select the Pin tool to “freeze” the Property Inspector.
3. Select the second object and open the Property Inspector.
4. Reposition the Property Inspectors to view them side by side.

# Common Layout Properties

Four objects with common properties:

- Frames
- Repeating Frames
- Fields
- Boilerplate Objects

Some common properties affect:

- Sizing
- Pagination
- Frequency of display



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Common Layout Properties

Most layout objects share a number of common properties. This section explains how to use these common properties.

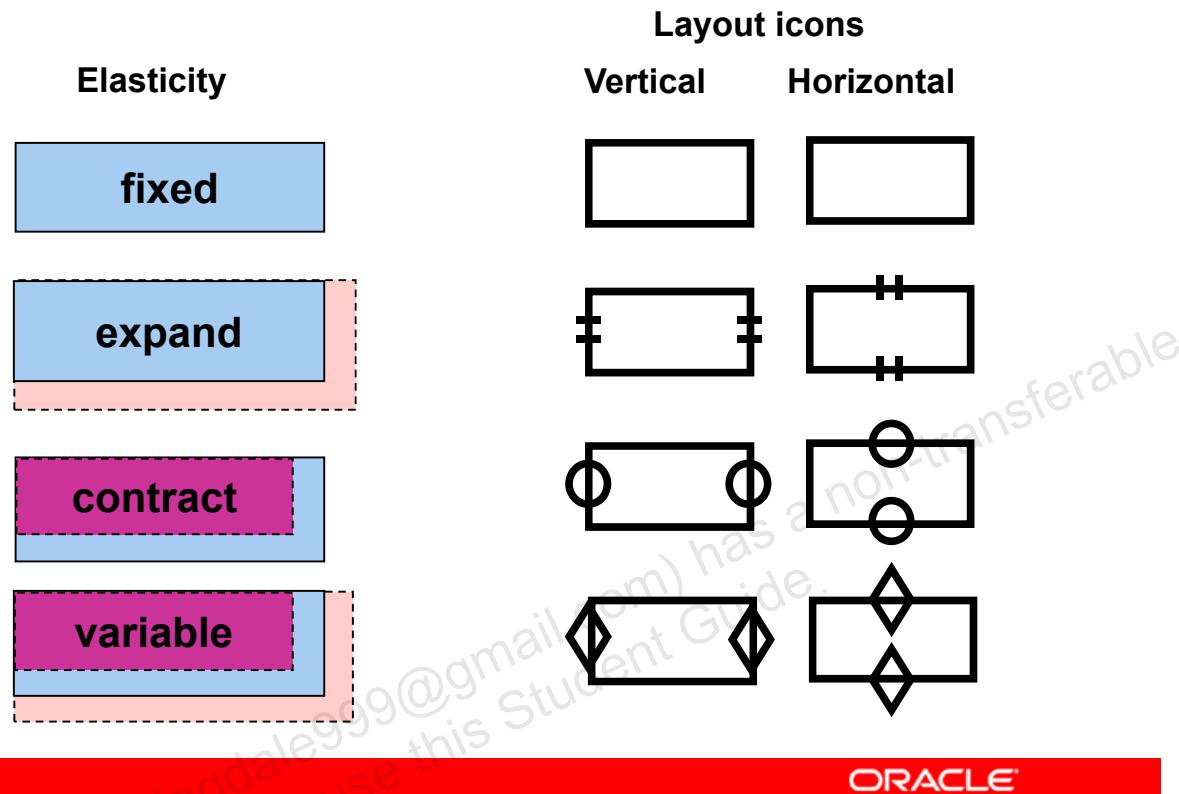
The four layout objects that have common properties are:

- Frames
- Repeating Frames
- Fields
- Boilerplate Objects

These common properties include:

- Sizing: Horizontal and Vertical Elasticity
- Pagination: Page Break Before, Page Break After, Page Protect, and Keep with Anchoring Object
- Frequency of Display: Print Object On and Base Printing On

# Sizing Objects



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Sizing Objects

You can create objects of any size in the Paper Layout. The elasticity properties enable you to specify whether the object can change size at run time.

The four types of object elasticity and the output object size are:

- **Fixed:** identical to the layout object size
- **Expand:** can be larger than the layout object, but not smaller
- **Contract:** can be smaller than the layout object, but not larger
- **Variable:** can be larger or smaller than the layout object

## Sizing Objects (continued)

### How Wizard Report Styles Affect Elasticity

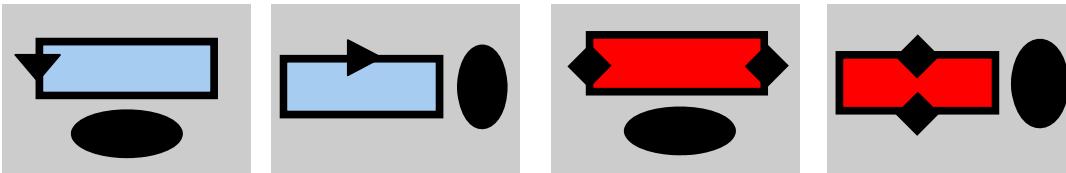
Frames: The Report Wizard sets the elasticity of enclosing objects, such as a group frame, to allow for multiple occurrences of a repeating frame.

Fields: When you reduce the size of a character field in the Labels tab, the Report Wizard, depending on the report style, sometimes alters the vertical elasticity of the field to allow for additional characters.

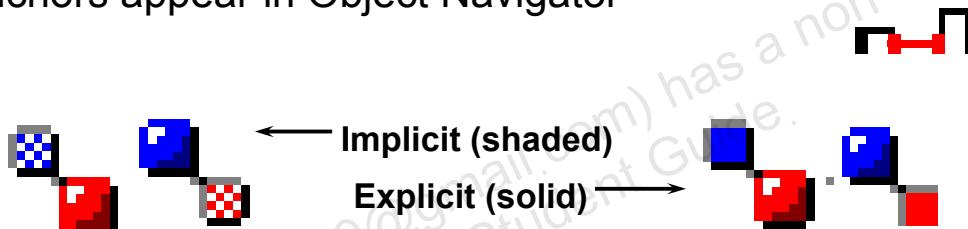
Layout Style	Horizontal Elasticity	Vertical Elasticity
Tabular	Fixed	Expand
Group Above	Fixed	Expand
Form	Fixed	Expand
Form Letter	N/A	N/A
Mailing Label	N/A	N/A

# Anchors

- Objects in the *push path* have implicit anchors (not visible in Paper Layout)



- Explicit anchors override implicit anchors
- All anchors appear in Object Navigator



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Anchors

Anchors are used to determine the vertical and horizontal positioning of a child object relative to its parent. An anchor defines the relative position of a child object to the parent object to which it is anchored. An object is called an internal child object when it is enclosed by the parent object. An external child object is not enclosed by its parent object.

The absolute position of each layout object is, by default, related to the position of its enclosing object at run time, unless one of the following is true:

- Reports Builder determines that the position causes a conflict with other objects, so Reports Builder creates an *implicit* anchor.
- You decide to override the default position of an object, so you create an *explicit* anchor.

Reports Builder creates an implicit anchor for any object that it considers to be in the *push path* of another object. For example:

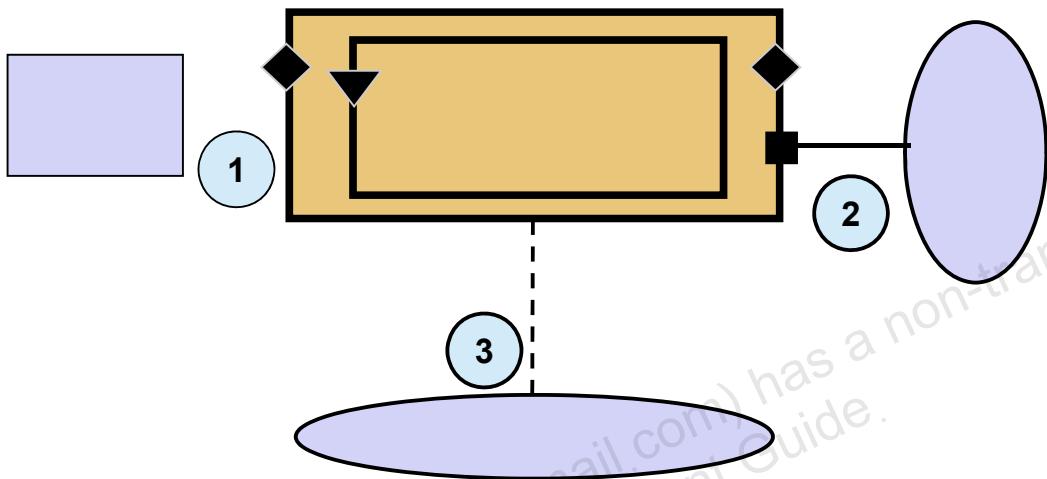
- An object occurring below a vertical repeating frame
- An object occurring to the right of a horizontal repeating frame
- An object occurring below an object that has a variable or expandable vertical size
- An object occurring to the right of an object that has a variable or expandable horizontal size

## Anchors (continued)

The chart below compares implicit and explicit anchors.

Implicit	Explicit
Not visible in the Paper Layout	Visible in the Paper Layout
Created and maintained automatically by Reports Builder	Created by the developer using the anchor tool in the tool palette
Visible in the Object Navigator if Anchoring Information is enabled (Navigator option)	

# Layout Object Relationships



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Layout Object Relationships

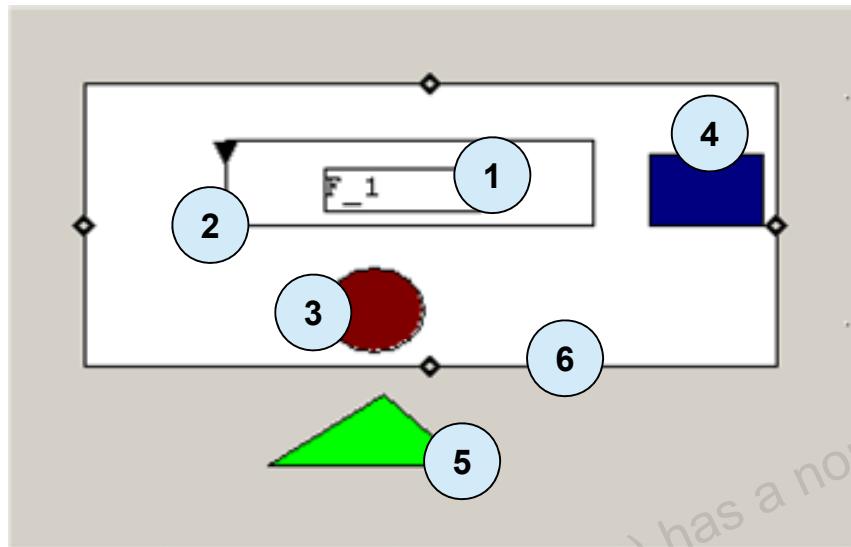
To understand how the pagination properties work, you must remember that all objects in the Paper Layout are related in a hierarchical structure.

You learned that the relationship depends on anchors, either explicit or implicit. Each object is anchored either to the logical page or to another object.

You can use this hierarchical relationship to control the display of related objects, for example, if you want to force a page break between related objects, or keep related objects together on a page.

1	No relationship, no anchor
2	Explicit anchor to enforce relationship
3	Implicit anchor because object is in the <i>push path</i> of a frame

## Layout Object Relationships



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Layout Object Relationships (continued)

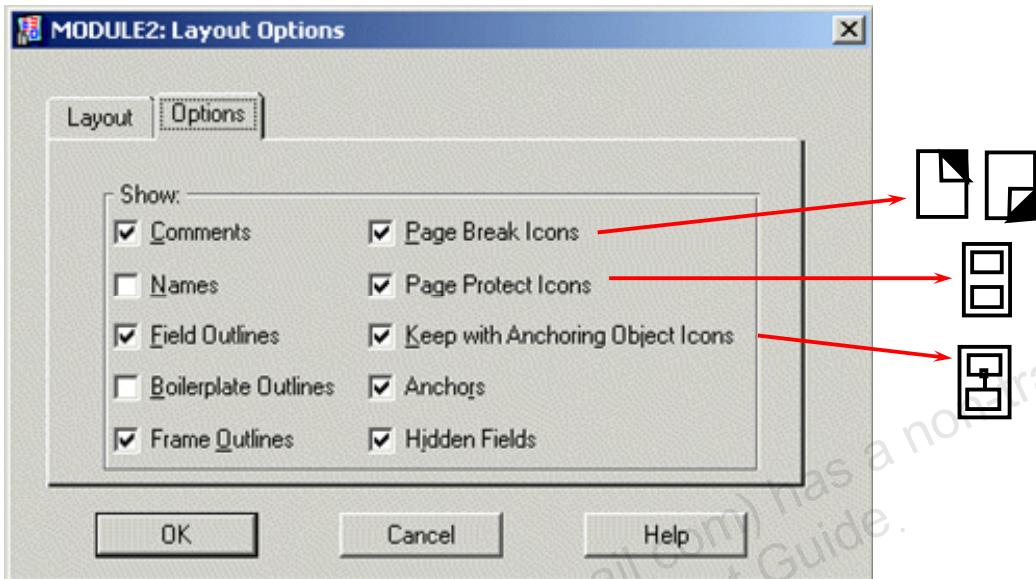
	<b>Layout Object</b>	<b>Relationship</b>
1	Field	Repeating frame, enclosing object
2	Repeating frame	Group frame, enclosing object
3	Object 1	Repeating frame, implicit anchor
4	Object 2	Group frame, enclosing object
5	Object 3	Group frame, implicit anchor
6	Group frame	

## **Layout Object Relationships (continued) More About the Push Path**

By default, Reports Builder implicitly anchors each object to its immediate enclosing object. However, sometimes Reports Builder must alter the implicit anchor to ensure that objects do not overlap at run time when their size is not fixed.

Reports Builder uses an implicit anchoring algorithm to determine whether an object lies in the *push path* of another object.

## Pagination Icons in the Paper Layout



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Pagination Icons in the Paper Layout

The four common properties that affect whether and where additional page breaks occur in your paper report are Page Break Before, Page Break After, Page Protect, and Keep with Anchoring Object (Advanced Layout).

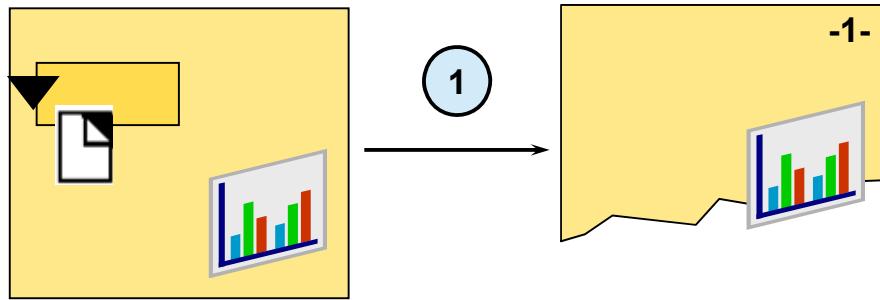
- **Page Break Before:** Force the object to be formatted on the next logical page after the one on which it would initially print. All external child objects also print on the new page. Unrelated objects may print on the original page.
- **Page Break After:** Force all external child objects to print on the next logical page. That is, every child object that is attached by an anchor (explicit or implicit) reacts as if it has Page Break Before set on itself.
- **Page Protect:** Cause the entire object and its enclosed objects to be kept together on the same logical page. This is a useful way of keeping the entire contents of a frame or repeating frame on the same logical page, without using anchors.
- **Keep with Anchoring Object:** Cause the current object and the object to which it is anchored to be kept together on the same logical page.

## Pagination Icons in the Paper Layout (continued)

The Paper Layout displays icons representing each of the properties when they are set to Yes (this is the default). These icons are very useful, especially for maintenance and support. However, you can suppress these and other icons in the Paper Layout options.

1. Select Tools > Options > Paper Layout.
2. Click the Options tab.
3. Select or clear the options as required.

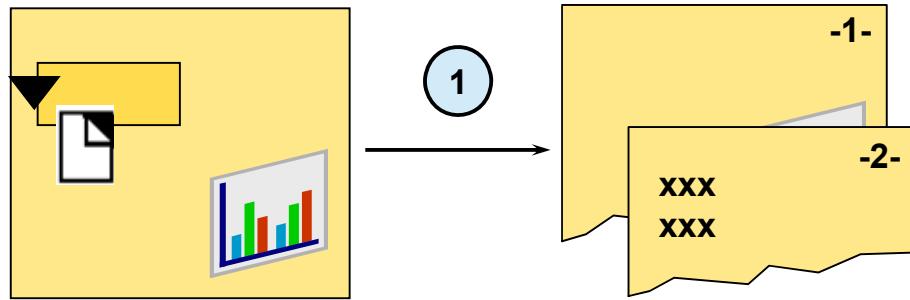
## Using Page Break Before



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

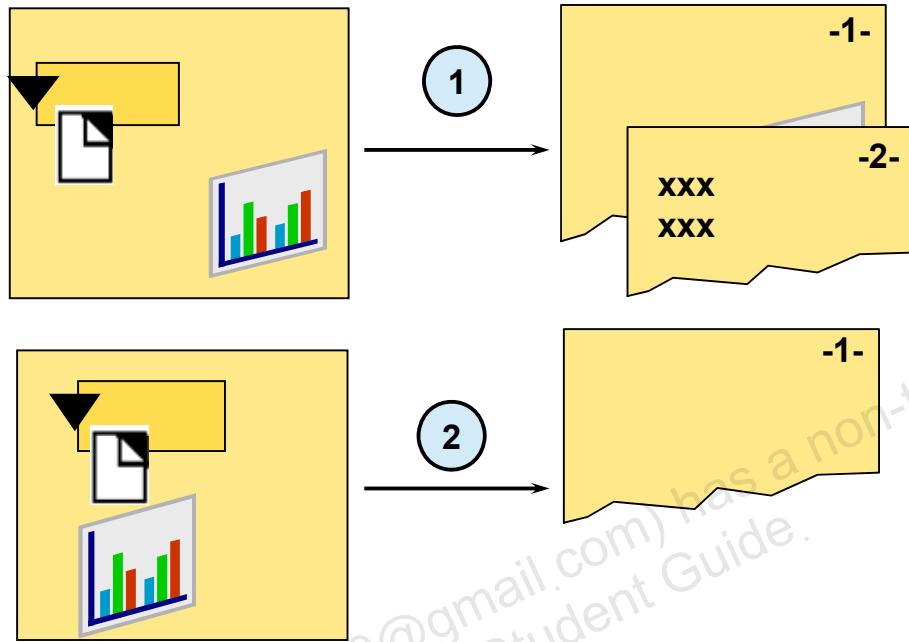
## Using Page Break Before



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

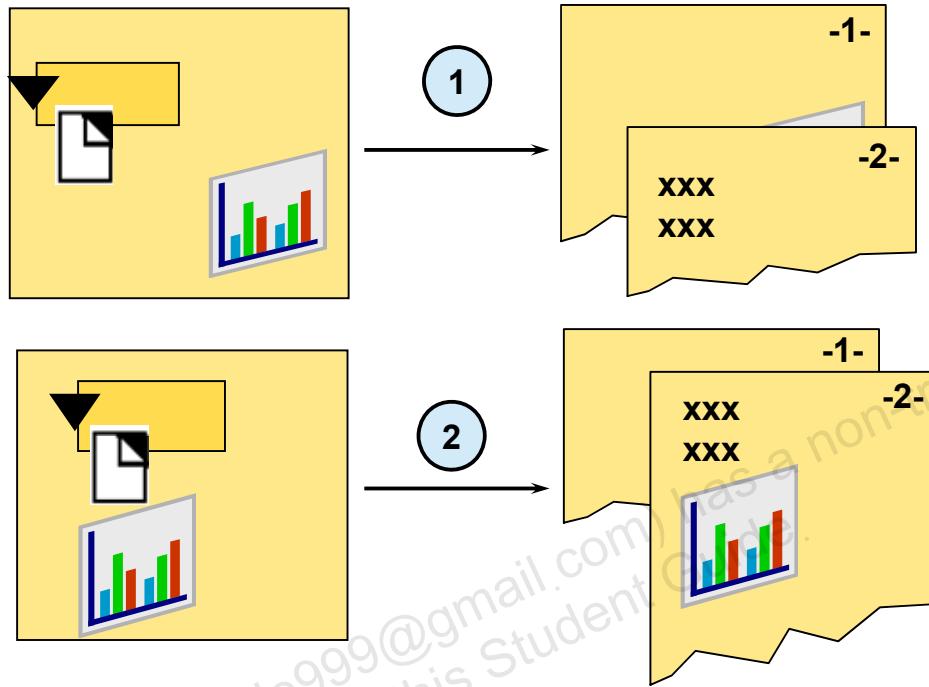
## Using Page Break Before



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Using Page Break Before



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Using Page Break Before

Page Break Before delays the formatting of the current object and its anchored child objects until the next page. Objects that are not implicitly or explicitly anchored to the current object do not move to the next page, even if they appear below the current object in the layout.

**Note:** Remember to use the Navigator options to view implicit and explicit anchor information in the Object Navigator. This helps you to understand the relationship between layout objects, and which objects are affected by setting a page break.

1	Image is not related to repeating frame
2	Image is in the <i>push path</i> of and implicitly related to the repeating frame

## Using Page Break Before (continued)

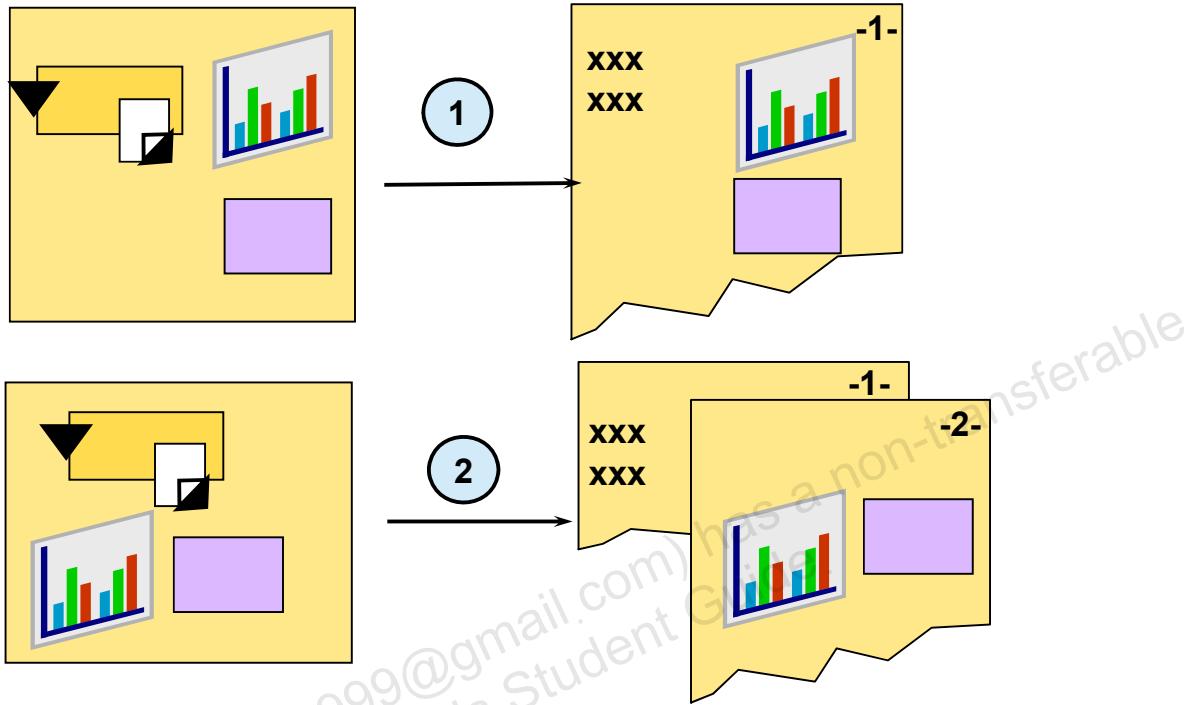
### Examples

Example 1: The repeating frame and image are not related. Therefore, when you select the Page Break Before check box for the repeating frame, the image does not move to the next page.

Example 2: The image is below, and in the *push path* of, the repeating frame. Therefore, there is an implicit anchor between the repeating frame (parent) and the image (child). The image formats after the repeating frame on the same page as the repeating frame.

**Note:** Setting Page Break Before on a repeating frame gives a page break before the first occurrence of the repeating frame only. It does not provide a page break between each record.

## Using Page Break After



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Using Page Break After

Page Break After delays only the formatting of those objects that are anchored either implicitly or explicitly to the current object.

Use Page Break After when you want to move multiple related objects to a new page. This is quicker and easier to maintain than setting Page Break Before on each of the individual child objects.

To see a warning message when Page Break After has no effect, in the Preferences dialog, go to Runtime Settings and make sure that Run Debug is enabled

**Note:** Remember to use the Object Navigator to view implicit and explicit anchor information.

1	Image and rectangle are not related to repeating frame; Page Break After has no effect
2	Image and rectangle are in the <i>push path</i> of and are implicitly related to the repeating frame

## Using Page Break After (continued)

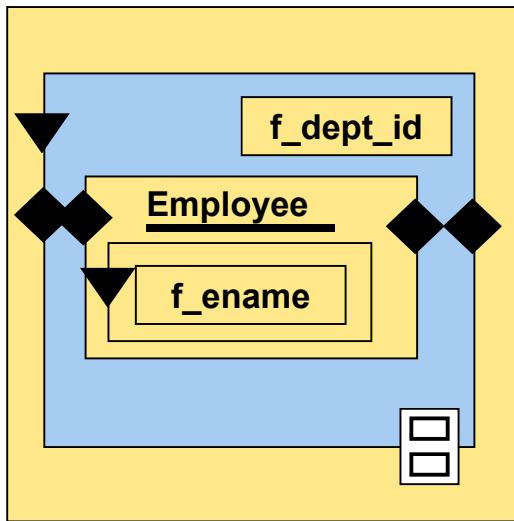
### Examples

Example 1: The repeating frame and image are not related. Therefore, when you set Page Break After to Yes for the repeating frame, the image does not move to the next page.

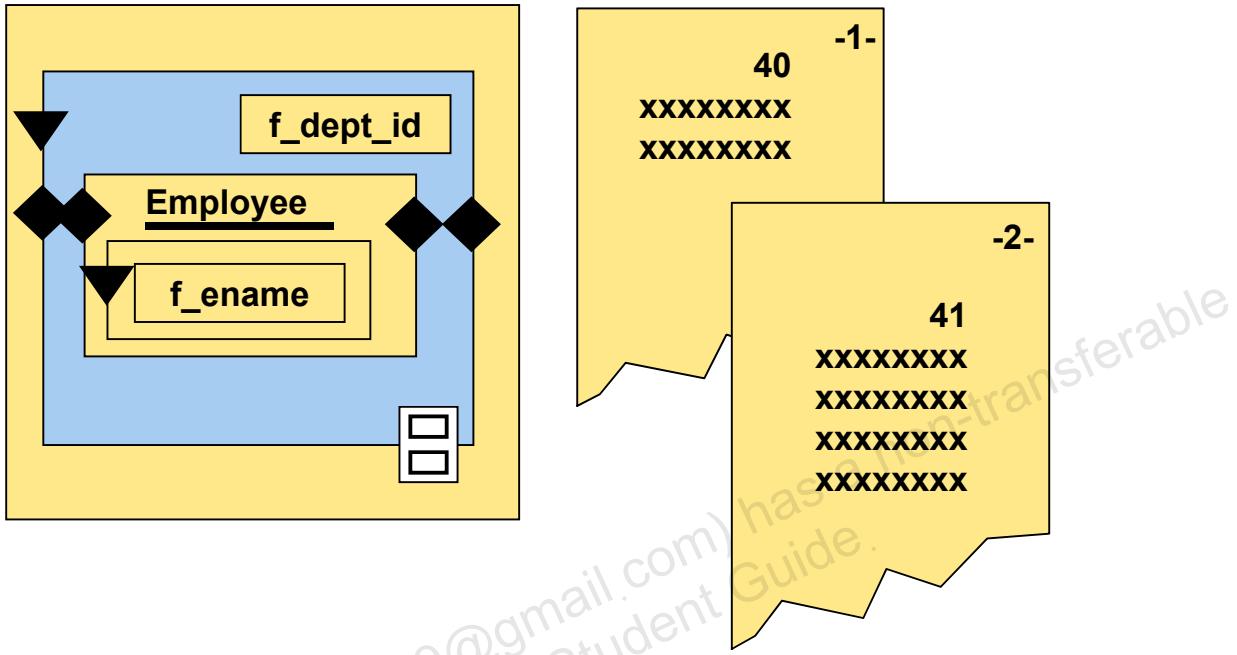
Example 2: The image and rectangle are below, and in the *push path* of, the repeating frame. Therefore, there is an implicit anchor between the repeating frame (parent) and the image and rectangle (children). The page break occurs immediately after the last instance of the repeating frame, and the image and rectangle appear on the new page.

**Note:** Setting Page Break After on a repeating frame gives a page break after the last occurrence of the repeating frame only. It does not provide a page break between each record.

# Using Page Protect



# Using Page Protect



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Using Page Protect

Page Protect indicates whether to keep the entire object and its internal child (enclosed) objects on the same logical page.

Page Protect applies only to the first logical page on which the object normally formats; Reports Builder ignores Page Protect on subsequent pages to avoid an infinite loop.

**Example:** A particular instance of a master repeating frame might contain so many detail records at run time that it is not possible to display them all on the same page. If so, you can force a page break before the master instance begins.

- To start the master on a new page, set Page Protect to Yes on the master repeating frame.
- If there is no suitable frame to protect several objects, create your own frame, enclose the objects in the frame, and set Page Protect on the frame.

**Hint:** If you create your own frame to protect several objects, give the frame a solid fill so that you can ensure that you place it at the correct layer of the layout. The frame must be behind all the objects it encloses.

Remember to remove the fill when you are satisfied with the layout.

Ensure that the frame's elasticity properties are set to Variable, Expand, or Contract, depending on the sizing of the objects it encloses.

## Using Page Protect (continued)

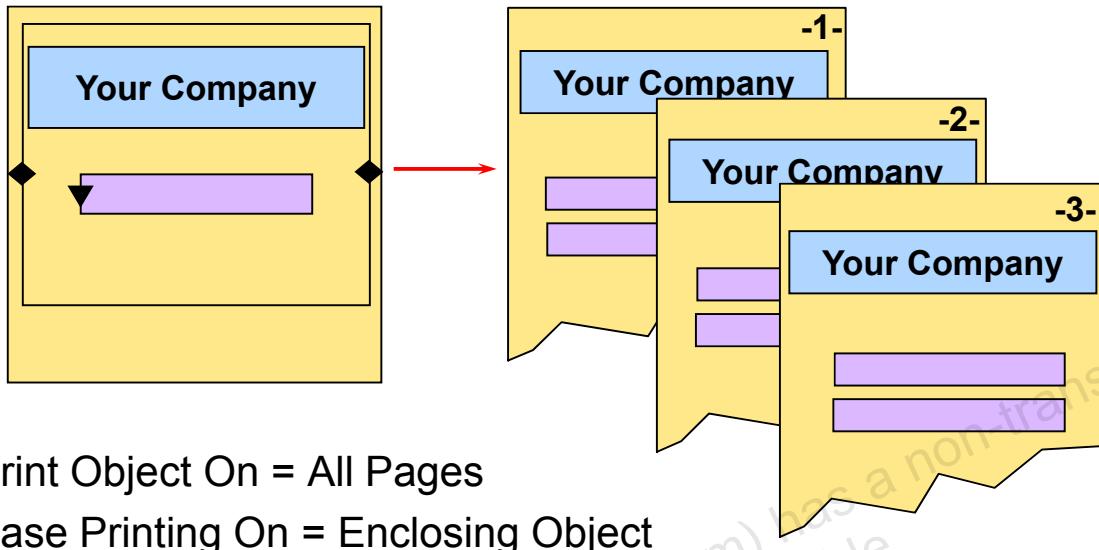
### Using Keep with Anchoring Object

The Keep with Anchoring Object property is similar to Page Protect, except that it affects anchored objects instead of enclosing objects.

Keep with Anchoring Object is useful when you want to keep two objects together on the same page.

If there is not enough room to display the child object on the same page as its parent, Keep with Anchoring Object moves both parent and child objects to a new page.

# Controlling Print Frequency



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Controlling Print Frequency

Two related properties, Print Object On and Base Printing On, control how often to print the object in relation to another object.

### Which Page?

Print Object On determines on which of the parent's logical pages the object is printed.

Print Object on	Object Prints on
All Pages	All logical pages of the parent
All But First Page	All logical pages except the first or last page
All But Last Page	
First Page/Last Page	The first or last logical page only
Default	The option that Reports Builder chooses; Reports Builder resets the property, using an internal algorithm; use the default setting if you have made a mistake and want to return to the report default

## Controlling Print Frequency (continued)

### Which Parent?

Base Printing On determines whether the parent object is to be the Enclosing Object or the Anchoring Object.

### Example

The report on the previous page shows a text title inside the group frame. The output for this frame shows there are several records that span three pages.

When you create a piece of text, by default it is printed on the First Page only of its enclosing object. In this case, the enclosing object is the group frame. You can modify the properties to select the pages on which the text is printed.

To print the text on all pages of the group frame, set Print Object On to All Pages; set Base Printing On to Enclosing Object (default).

**Note:** If the text is outside all group frames, its parent is the body page itself. Therefore, you can use these two properties to display “Continued...” on all but the last page, or “...continued” on all but the first page.

There are several restrictions that apply to these settings. If you receive an “invalid setting error,” look at the explanation and restrictions in the Reports Builder Help Topics.

# Using Format Triggers

The screenshot shows the Oracle Reports Designer interface. In the top window, titled "MODULE3: Program Unit - F\_SALARYFORMATTRIGGER", there is a code editor containing a PL/SQL function named F\_SALARYFormatTrigger. The function checks if the salary is greater than or equal to 12000 and changes the font color to red, font face to Arial, and font size to 10. It returns TRUE. Below the code editor is a status bar showing "Not Modified" and "Successfully Compiled". To the right of the code editor is a preview window displaying a report with columns "Name" and "Salary". The report data is as follows:

Name	Salary
Mark	\$24,000.00
Char	\$17,000.00
Laan	\$17,000.00
Hold	\$9,000.00
T	\$6,000.00
In	\$4,800.00
Balla	\$4,800.00
Mitz	\$4,200.00
Greenberg	\$12,000.00
Daniel	\$9,000.00
Faviet	\$8,200.00
John	\$8,200.00
Chen	\$7,700.00
Ismael	\$7,700.00
Scierra	\$7,800.00
Jose Manuel	\$7,800.00
Urman	\$6,900.00
Luis	\$6,900.00
Popp	\$16,500.00
Den	\$16,500.00
Raphaely	

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Using Format Triggers

A *format trigger* is a user-written PL/SQL function that executes each time before the object containing the trigger is formatted.

All the main layout objects, frames, repeating frames, fields and boilerplate objects, can contain a format trigger.

You can use a format trigger to dynamically change the formatting attributes of an object. The PL/SQL function must return a Boolean value (TRUE or FALSE). This dictates whether the current instance of the object is included or excluded from the report output.

# Summary

In this lesson, you should have learned how to:

- Identify common properties for most layout objects:
  - Size
  - Pagination
  - Print frequency
  - Format trigger
  - Web settings
- Modify common properties for special reporting needs



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Summary

Report objects provide many properties that you can modify in the Property Inspector to alter the appearance and position of objects in your report output.

- Common properties exist for most layout objects.
- You do not have to make many alterations for most standard reporting needs.

This lesson explained how to use common object properties and gave examples of some common uses, such as:

- Forcing pagination by setting Page Break Before or Page Break After
- Format triggers
- Web settings

## Practice 10 Overview

- Creating a break report with continuation pages displaying limited information
- Modifying properties for object Web support



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Practice Session Overview: Lesson 10

This practice session contains:

- Creating a break report with continuation pages displaying limited information
- Modifying properties for object Web support

Sometimes detail records get separated from their master header. You ensure that this does not happen.

All paper layout objects support Web features. You will create a report that takes you to a PDF document with a simple click using a hyperlink.

## Practice Session: Lesson 10

1. Create a new break report showing each customer and the relevant orders.
  - a. Create a group above master detail report. Use the query p12\_1.sql to create the report.
  - b. Select Name as the group field and display all the customer data in the break group. Create a sum for the quantity.
  - c. Modify the width for the fields listed below:

Column	Label	Width
name	Name:	20
street_address	Address:	10
city	City:	10
state_province	State / Province:	10
postal_code	Zip:	5
country_id	Country:	2
credit_limit	Credit Limit:	6

- d. Run the paper layout. What do you see?
  - e. Ensure that no customer data displays on a page without details.
  - f. Save the report as p12q1.rdf.
  - g. Run the paper layout again to test.
2. Create a new tabular report.
  - a. Define the SQL query as:

```
select last_name, first_name, hire_date
from employees
```

Display all fields. Choose the Wine template.
  - b. In the margin region of the report's Main Section, delete the company logo. Resize the margin to 1.75 inches. Insert the image dinner1.bmp. Create a rectangle around it and give it a solid fill. Move the rectangle behind the image to create the effect of a colored border around the image
  - c. When you click on the image in HTML output you want to open a local .pdf file named mousse.pdf .
  - d. Save the report as p12q2.rdf and generate a file in paginated HTMLCSS format. Open the file in your browser and test the hyperlink.

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

Shital jadhav (shitaljagdale99@gmail.com) has a non-transferable  
license to use this Student Guide.

# 11

## Controlling the Paper Layout: Specific Properties

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Objectives

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

- Identify specific layout properties
- Modify specific layout properties
- Explain the use of specific properties
- Specify the format order for the report sections
- Enable cataloguing and searching of PDF output



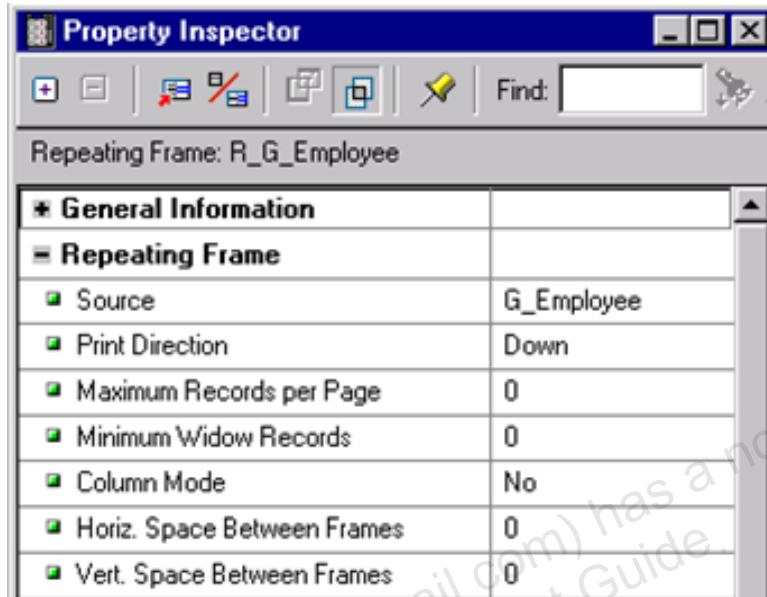
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Overview

In the previous lesson, you learned about common object properties. There are however a number of properties that are unique to the reports and specific types of layout objects.

In this lesson, you learn about specific properties for reports, repeating frames, fields, and boilerplate. You also learn how to reference the contents of a file at run time. In addition, you learn to specify the format order for the report sections. Finally, you learn to specify taxonomy properties for PDF documents.

# Properties of a Repeating Frame



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

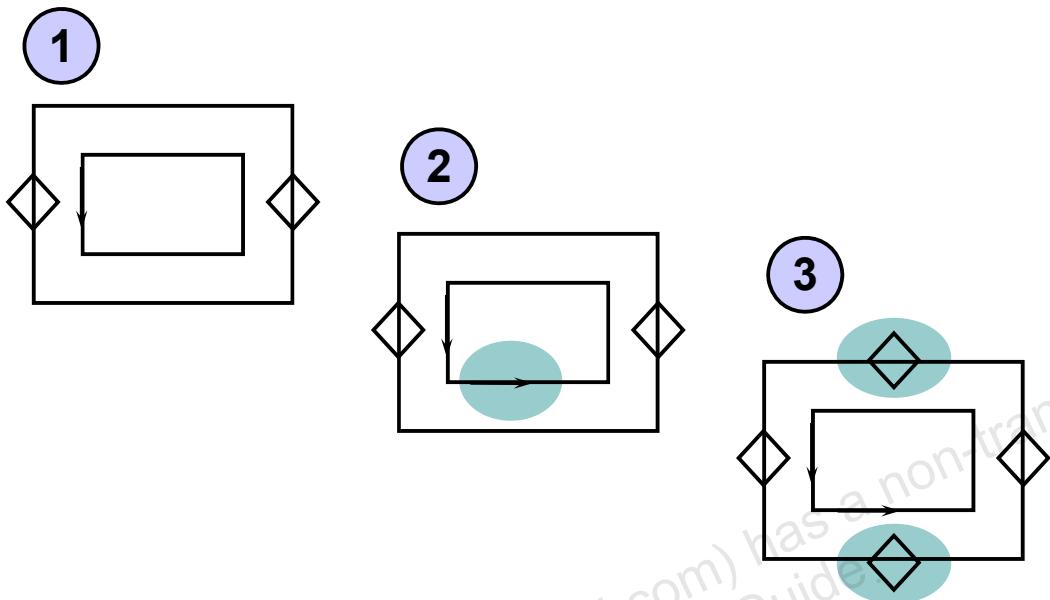
## Modifying Properties of a Repeating Frame

A repeating frame is record-related, and its properties allow you to modify specific attributes associated to the display of the records.

- **Source:** Name of the Data Model group that provides the source data for this repeating frame; you must source every repeating frame to a valid group
- **Print Direction:** Direction in which records are printed: Down or Across, Across/Down, or Down/Across
- **Maximum Records per Page:** Maximum number of records that can be displayed on a single page
- **Minimum Widow Records:** Minimum number of records from a group that can be displayed at the bottom of a page
- **Column Mode:** Whether to maintain the column for each record across multiple pages
- **Horiz. Space Between Frames, Vert. Space Between Frames:** Spacing between each record, horizontally and vertically

The most commonly used properties are discussed in more detail on the following pages.

# Specifying Print Direction



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Specifying Print Direction

Print direction specifies the direction in which to print the repeating frame's records, and may be any one of these values:

- **Down:** Prints sequential records down the page. At the end of the page, records overflow onto the next page.
- **Down/Across:** Prints sequential records down the page. At the end of the page, records move across to form a new column if there is room; otherwise they overflow onto a new page.
- **Across:** Prints sequential records across the page. When records reach the right edge of the page, they overflow onto a new page.
- **Across/Down:** Prints sequential records across the page. When records reach the right edge of the page, they move down below the previous set of records if there is room; otherwise they overflow onto a new page.

When you alter the print direction of a repeating frame, you must consider the properties of any enclosing objects. Enclosing objects may include other repeating frames and group frames.

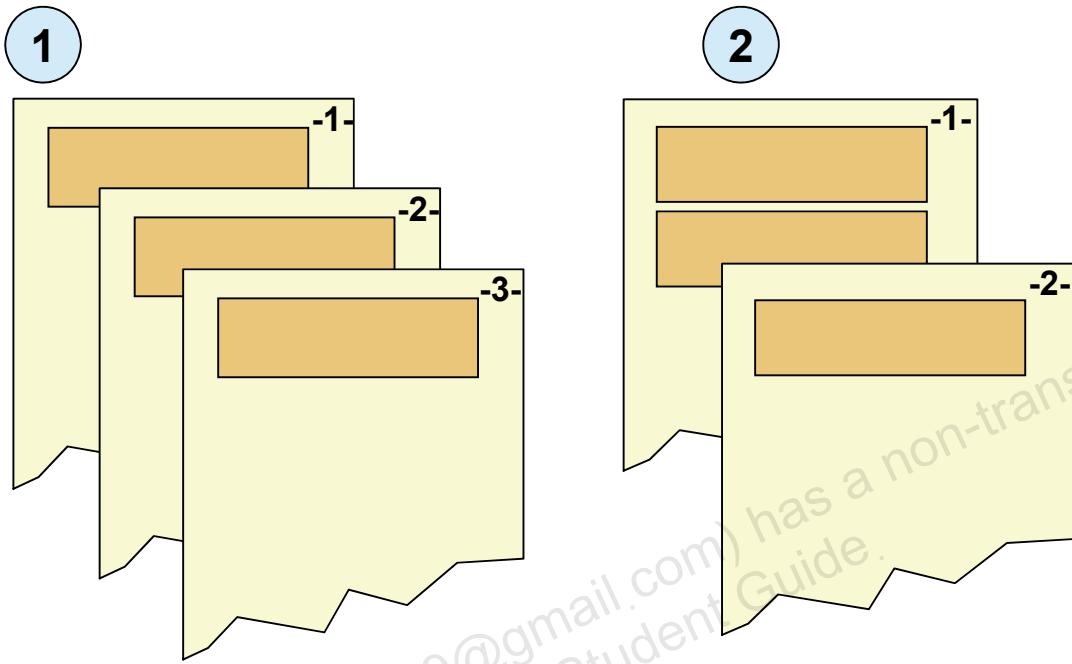
## **Specifying Print Direction (continued)**

### **Example**

In diagram 1 opposite, the default print direction of the records is Down. Therefore, the vertical elasticity property of the enclosing group frame is set to Variable to allow for a variable number of records, but the horizontal size is fixed.

If you decide to alter the print direction to Down/Across as in diagram 2, you may need to modify the properties of the enclosing group frame to be variable (diagram 3). If not, the records still overflow to the next page.

## Controlling the Number of Records per Page



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Controlling the Number of Records per Page

You can specify the maximum number of records that you want to display on each logical page.

In the repeating frame Property Inspector, choose the Maximum Records per Page property. The value you enter for this property can be any positive whole number or blank.

If this setting is 0 (zero), Reports Builder formats as many records as possible on the logical page.

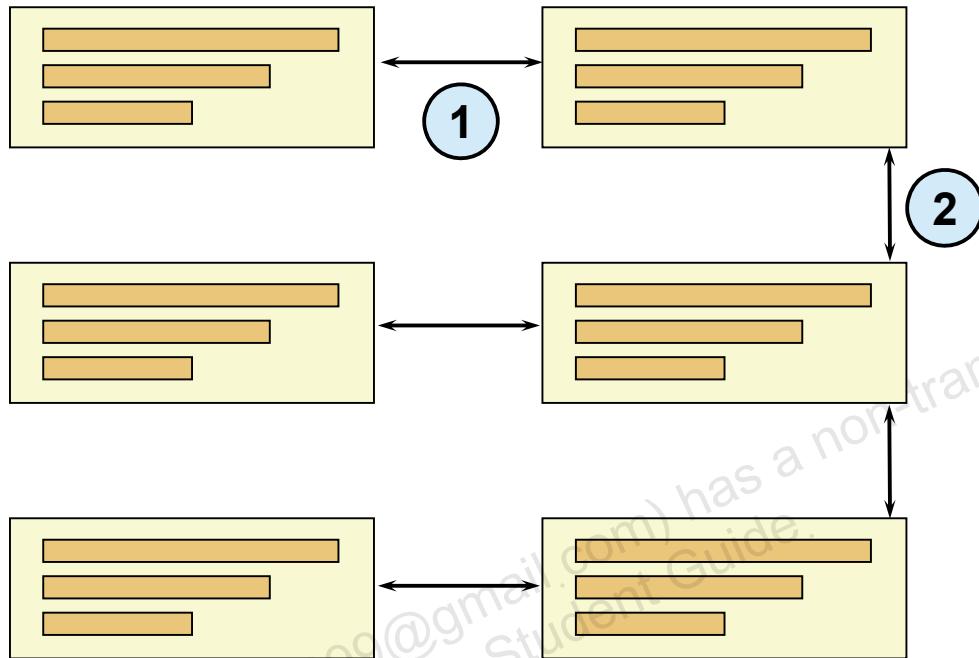
#### Examples:

Recall the Form and Form Letter report styles. Reports Builder automatically sets Maximum Records per Page to 1. If you require more than one record on each page, you can modify this property.

In the Group Above report style, Reports Builder formats as many records as possible on a page. (Maximum Records per Page is null.) Modify this property for the master repeating frame to display one master record per page.

# Controlling Spacing Between Records

## Mailing Labels



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Controlling Spacing Between Records

The properties Horiz. Space Between Frames and Vert. Space Between Frames control the amount of space between occurrences of a repeating frame; that is, record spacing.

**Horizontal Space Between Frames:** Define the amount of space you want between records horizontally across the page by entering zero or any positive number for this setting.

This setting has no effect if the repeating frame has a print direction of Down.

**Vertical Space Between Frames:** Define the distance you want between records vertically down the page by entering zero or any positive number for this setting.

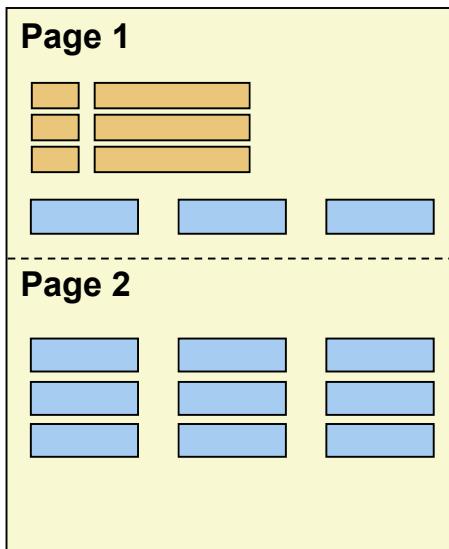
This setting has no effect if the repeating frame has a print direction of Across.

**Note:** In bitmapped reports, the distance is a whole or part of the unit of measurement, for example,.25 (of an inch).

In character mode reports, the distance is the number of blank characters (horizontal) or lines (vertical). Therefore, you must enter a whole number, for example, 10 (characters).

## Minimum Widow Records

Without widow control

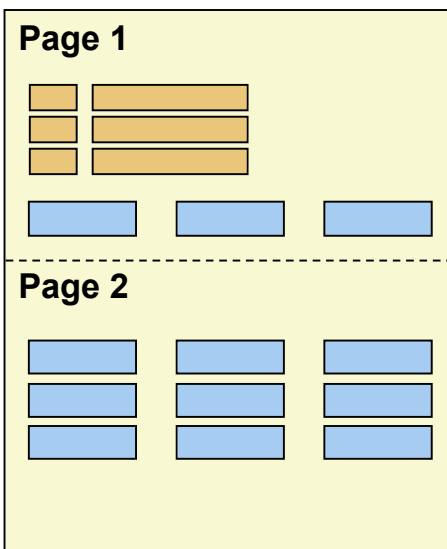


ORACLE

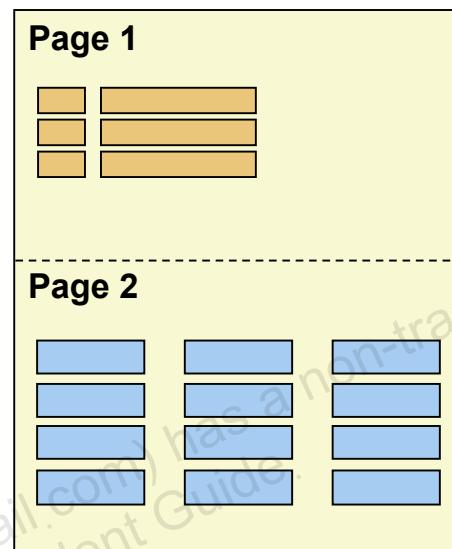
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Minimum Widow Records

**Without widow control**



**With widow control  
(Min Widow Records = 3)**



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Minimum Widow Records

You can specify the minimum number of records that must display on a page. The default is zero, which means a single record may display alone on a page.

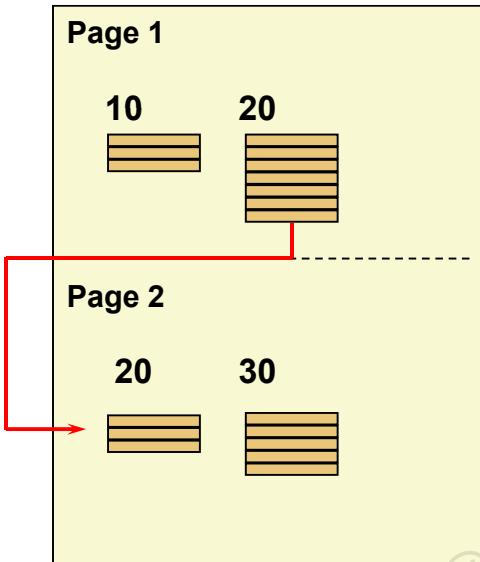
#### Example

In the diagram above, there are two separate repeating frames printing down the page. The second repeating frame is near the end of the page, and there is only room for one record.

If you want to force at least three records on a page, enter 3 in the Minimum Widow Records property so that the repeating frame starts to format on the next page.

# Column Mode

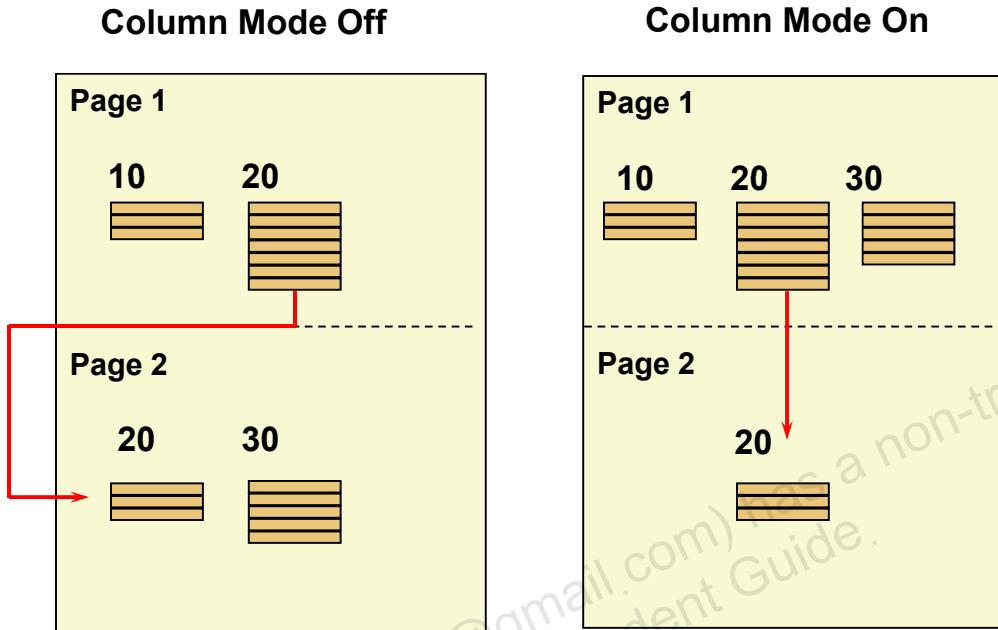
## Column Mode Off



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Column Mode



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Column Mode

You can set Column Mode to Yes to allow the next instance of a repeating frame to begin formatting before the previous instance completes. Column Mode is used mainly for master repeating frames or repeating frames that contain fields that may expand vertically or horizontally (for example, elasticity is Variable or Expand).

### Example

The example above shows the use of Column Mode for the department master repeating frame given three records, 10, 20, and 30. The department repeating frame's print direction is Across, and the employee repeating frame's print direction is Down.

Use Column Mode to start formatting department 30 on page 1, and also to align the overflow for department 20 on page 2.

**Note:** Column Mode does not make sense for repeating frames that have a print direction of Across/Down or Down/Across.

## Properties of a Field

- Fields define formatting attributes and logic
- Wizard creates a field for each column
- You create additional fields in the Paper Layout:
  - Source: Column, parameter, or system variable
  - Format mask: Standard Oracle format symbols

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Properties of a Field

A field is a layout container for a column or a parameter.

A field defines formatting attributes or logic for the data of its related column or parameter.

The Report Wizard creates one field for each column and places these fields inside a repeating frame or at report level.

You can create a field in the Paper Layout and relate it to a Data Model column or parameter.

## **Properties of a Field (continued)**

The Source property provides a list of columns, of which there are three categories:

- Columns in the Data Model
- Parameters (always listed in uppercase)
- System variables, such as today's date and current page number

The datatype of a field depends on its source.

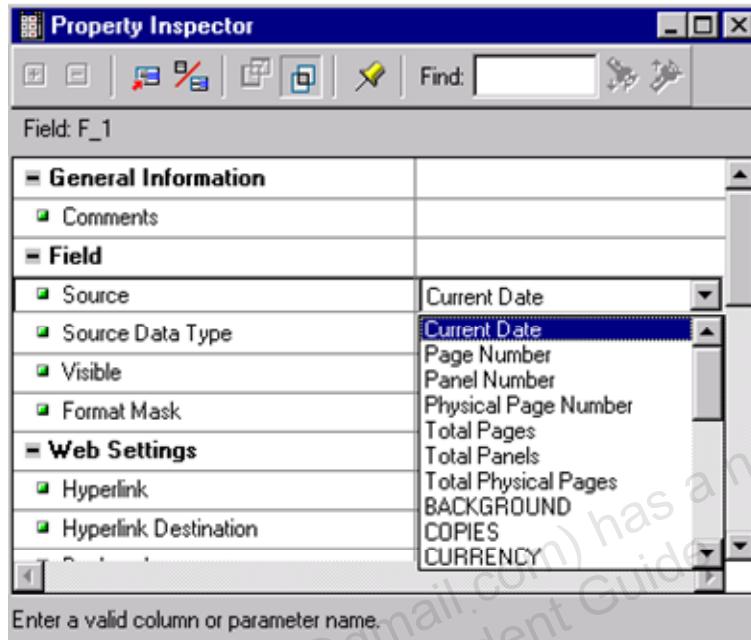
## **Defining a Format Mask**

When you insert a date object (Insert > Date and Time) in the Paper Design or the Paper Layout, Reports Builder provides sample dates to enable you to pick a format easily.

The format mask property appears in the standard Oracle date format symbols, such as MM, which displays the number of the month, or Month, which spells out the name of the month in full.

The list provides all the format masks that exist in your preferences. Modify your preferences to include the common masks that you use, or enter a different mask in this property.

# System Variables



**ORACLE**

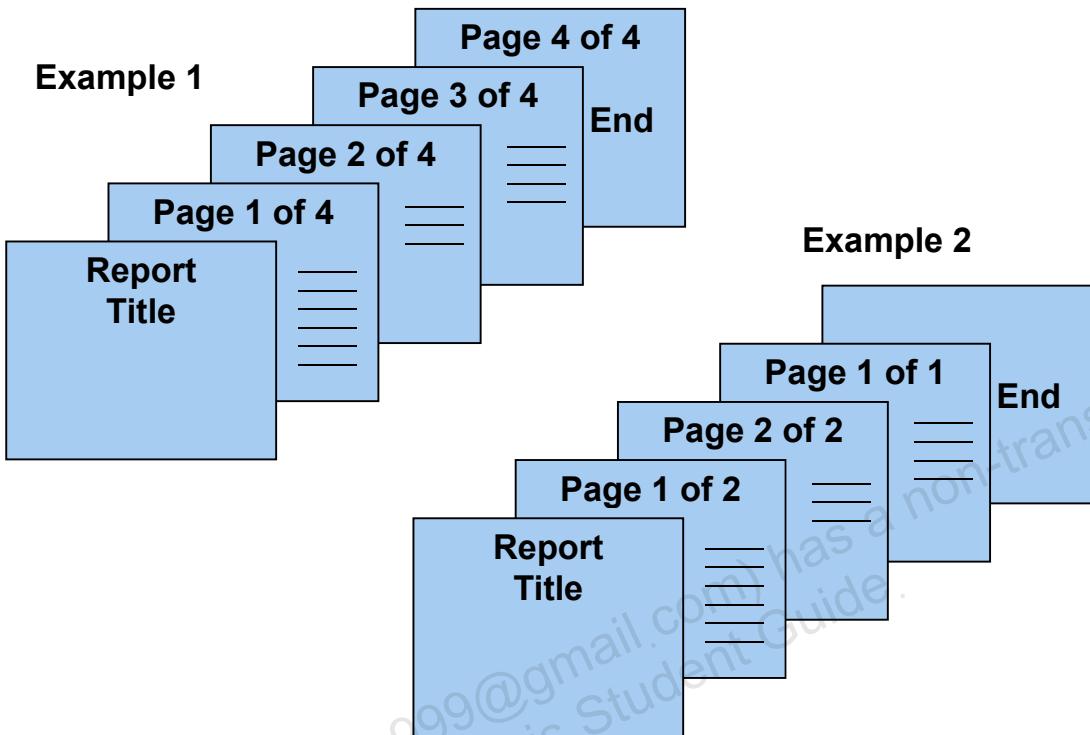
Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## System Variables as the Source of a Field

The following table lists the variables and describes their display values.

Variable Name	Description
Current Date	The operating system date on the mid-tier when the report runs, after the Runtime Parameter Form has been displayed
Page Number	The current page number based upon numbering the output by logical pages
Panel Number	The current panel number in the current logical page
Physical Page Number	The current page number based upon numbering the output by physical pages
Total Pages	The total number of logical pages in the current report run
Total Panels	The total number of panels in a logical page
Total Physical Pages	The total number of pages based upon numbering the output by physical pages

# Page Numbering



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Page Numbering

When the source of a field is one of the system variables for page or panel numbers, you can define where and how your page numbers are calculated by choosing the Page Numbering property.

### Example 1

To number all the pages in the main section of a report and include trailer section pages but exclude header section pages, you would specify:

Page Numbering Property	Value
Include	Main Section, Trailer Section
Start At	1
Increment By	1
Reset At	Report

## Page Numbering (continued)

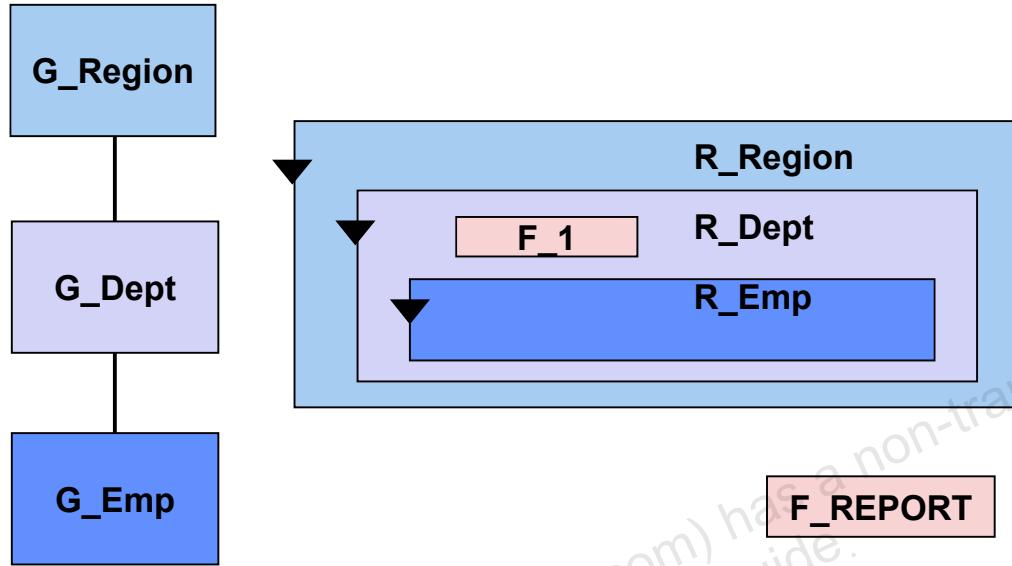
### Example 2

To number all pages in the main section of a report, excluding header and trailer pages, and to restart the numbering at 1 for each new department record in the repeating frame R\_Dept, you would specify:

Page Numbering Property	Value
Include	Main Section
Start At	1
Increment By	1
Reset At	R_Dept

**Note:** When resetting page numbers for each record in a repeating frame, make sure that each record starts on a new page. Use the Maximum Records per Page property; this property of the repeating frame was discussed earlier in this lesson.

## Valid Source Columns



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Valid Source Columns

When creating new fields, make sure that they exist at the correct level within the report; otherwise frequency errors occur.

If you create a field in a repeating frame, the column you choose as the source must be in the relevant group for the repeating frame or in an ancestor of that group.

In the diagram above, the Data Model shows a three-level hierarchy:

- G\_Region is the parent of G\_Dept.
- G\_Dept is the parent G\_Emp.

The layout shows the three nested repeating frames, R\_Report, R\_Dept, and R\_Emp.

If you create another field in the R\_Dept repeating frame, the source column must come from either G\_Dept or G\_Region; a column in G\_Emp would be an invalid source, because values in the G\_Emp group occur more often than R\_Dept would be printed.

## **Valid Source Columns (continued)**

### **Report-Level Fields**

If you create a field outside any repeating frame, that is, at report level, the source column must be a report-level column—outside all groups in the Data Model.

System columns, such as current date or page numbers, are report-level columns; you can use these in report-level fields, or at any lower level of your report. For example, you can display the current date at report level or within a repeating frame.

You often use the Header section or margin region of each section to display report-level information.

## When Are the Contents Updated?

Reports Builder reads the file contents when you:

- Modify the Property Inspector
- Open a report definition
- Run a report



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### When Are the Contents Updated?

Oracle Reports picks up the contents of a file link at run time. Therefore, the file contents can change dynamically for each report run.

The file link object automatically redisplays the current contents of the file when you:

- Modify the source filename in the Property Inspector
- Open the report (in Reports Builder)
- Run the report

**Note:** The file must exist at run time; otherwise the report fails with an error message.

# Summary

In this lesson, you should have learned how to:

- Identify object-specific properties
- Modify these properties when necessary
- Modify the default format order for report sections
- Specify taxonomy properties for PDF documents



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Summary

Report objects provide many properties that you can modify in the Property Inspector.

You do not have to make many alterations for most standard reporting needs.

- Common properties exist for most layout objects.
- Specific properties exist for individual object types, such as fields and repeating frames.

You also learned about some report-specific properties. In this lesson, you learned how to specify the order in which the Header, Main, and Trailer sections of a report are formatted. You also learned how to specify metadata for your PDF reports.

## Practice 11 Overview

- Modifying the printing of a mailing label report
- Controlling the number of records on a page
- Ensuring all details of a master print on the same page



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Practice Session Overview: Lesson 11

This practice session contains:

- Modifying the printing of a mailing label report
- Controlling the number of records on a page
- Ensuring that all details of a master are printed on the same page

Modify a mailing label report to ensure you fill the printed page with label records. You then restrict the number of labels that are printed on each page.

## Practice Session: Lesson 11

1. Open report p4q4.rdf.
  - a. Modify the layout so that the mailing labels are 2 inches wide and 1 inch high. Use the Layout option on the menu. Ensure that the customer names are printed in alphabetical order across the page.
  - b. Give each label a solid fill.
  - c. Make the boilerplate object 1.8 inches wide and center it horizontally and vertically in the repeating frame.
  - d. Add spacing between the labels: .25 inch between each column of labels across the page; .5 inch between each row of labels down the page.
  - e. Run the paper layout to test. Save the report as p13q1.rdf.
2. Continue with report p13q1.rdf.
  - a. Modify the report so that only six labels are printed on each page.
  - b. Run the paper layout to test. Save the report as p13q2.rdf and close it.
3. Continue with report p13q3.rdf.
  - a. Ensure that all items of one order are printed on the same page.
  - b. Save the report as p13q4.rdf and close it.

# Creating and Using Report Parameters

12

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Objectives

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

- Create and reference a parameter
- Create a list of values for parameter input
- Use and modify a system parameter
- Build a Paper Parameter Form layout
- Customize a Paper Parameter Form layout
- Use HTML parameter form extensions



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Overview

Parameters enable you to develop dynamic report documents that produce variable report output depending on the parameter value that you enter at run time. This lesson shows you how to create and reference user and system parameters and how to customize a Paper Parameter Form to display parameters at run time.

## Creating User Parameters

- Restrict values in a WHERE clause

```
SELECT CUST_LAST_NAME, ACCOUNT_MGR_ID  
FROM CUSTOMERS  
WHERE ID = <a value>
```

- Substitute any part of a SELECT statement

```
SELECT CUST_LAST_NAME, ACCOUNT_MGR_ID  
FROM CUSTOMERS  
<a where clause>
```

- Substitute a single column or expression

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



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Creating User Parameters

You can create your own parameters and use them to change the SELECT statement of your query at run time. Parameters can also be used with pluggable data sources: XML, JDBC, Text, and Express Server queries.

#### What Is a User Parameter?

A *user parameter* is a Data Model object that you create to hold a value that users can change at run time.

You can reference a parameter anywhere in a query. For example:

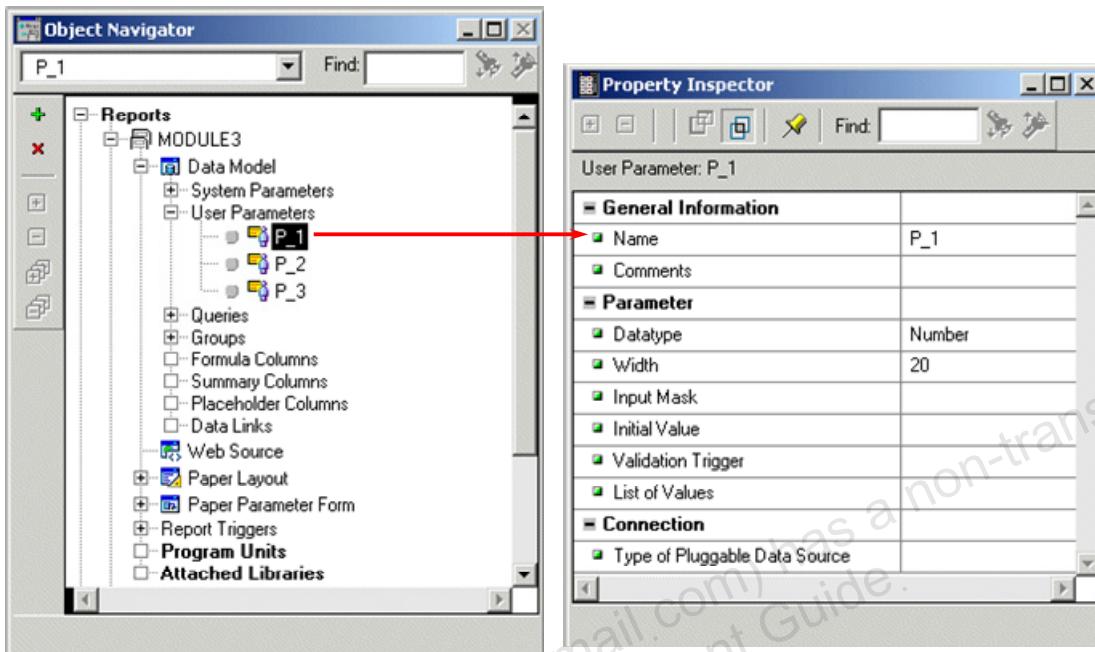
- Restrict values in the WHERE clause of the SELECT statement
- Substitute any part of the SELECT statement, including the entire statement
- Substitute a single column or expression in the SELECT list

You can display the contents of a parameter in your report by creating a paper layout field and entering the parameter name in the Field Source property.

#### Technical Note

For more information on using parameters with pluggable data sources, see the eClass *Oracle9i Reports: Integrate Pluggable Data Sources*.

# Creating User Parameters



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Creating User Parameters (continued)

To create a user parameter in the Object Navigator:

1. In the Object Navigator, click the User Parameter node, and then select the Create tool.  
**Note:** If this is the first parameter, you can create it by double-clicking the User Parameter node.
2. Rename the parameter and open the Property Inspector.
3. Verify the data type and width. Enter an initial value if required.

## Creating User Parameters (continued)

### Parameter Properties

Property	Description
Datatype	Specify whether parameter value is Character, Number, or Date; default is Number when created in the Object Navigator.
Width	Specify maximum allowable width of parameter value; maximum is 64 KB.
Input Mask	Allow users to enter a numeric or date value using a specific format.
Initial Value	Specify the value to use. You can override this value at run time from the command line or parameter form.
Validation Trigger	Validate the parameter value. Enter a PL/SQL function.
List of Values	Create a list of values from which users select a valid value at run time.
Type of Pluggable Data Source	Used by the PDS for acquiring single sign-on information. It identifies that parameter as containing sign-on information for that PDS connection.

## Referencing Parameters in a Report Query

- Bind reference replaces a value:  
`:parameter_name`
- Lexical reference replaces a clause:  
`&parameter_name`



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Referencing Parameters in a Report Query

There are two ways to reference parameters in a query:

- Use a bind reference
- Use a lexical reference

#### What Is a Bind Reference?

A *bind reference* is used to replace a single value or expression in SQL or PL/SQL.

To create a bind reference in a query, prefix the parameter name with a colon (:).

If the parameter object does not exist, Reports Builder automatically creates it for you and displays a message. In this case, the parameter default datatype is CHARACTER, not NUMBER.

## Referencing Parameters in a Report Query (continued)

### What Is a Lexical Reference?

A *lexical reference* is a placeholder that is used to replace any part of a SELECT statement, such as column names, the FROM clause, the WHERE clause, or the ORDER BY clause.

To create a lexical reference in a query, prefix the parameter name with an ampersand (&).

A lexical reference for a column or table must be created explicitly in the Object Navigator before you can use it in a query. For other clauses in the SELECT statement, if the parameter object does not exist, Reports Builder automatically creates it for you and displays a message. In this case, the parameter default datatype is CHARACTER, not NUMBER.

### Comparing Bind and Lexical Reference Types

Type	Prefix	Use to Replace	Parameter Created by Default?
Bind	:	Single value or expression in the following clauses: WHERE, GROUP BY, ORDER BY, HAVING, CONNECT BY, START WITH	Yes, if it does not already exist. Reports Builder displays a warning message and adds the parameter to User Parameters in the Object Navigator.
Lexical	&	Any part of a SELECT statement	Provided it is not used for a column or table, yes: if it does not already exist. Reports Builder displays a warning message and adds the parameter to User Parameters in the Object Navigator. Datatype must always be Character.

# Using Bind References

- Restrict values in a WHERE clause

```
SELECT CUST_LAST_NAME, ACCOUNT_MGR_ID  
FROM CUSTOMERS  
WHERE CUSTOMER_ID > :P_CUST
```

- Substitute a single value or expression in the select statement

```
SELECT CUST_LAST_NAME, ACCOUNT_MGR_ID  
FROM CUSTOMERS  
ORDER BY DECODE(:SORT, 1, CUST_LAST_NAME,  
                  2, NLS_TERRITORY,  
                  ACCOUNT_MGR_ID)
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Using Bind References

Use a bind reference anywhere in a query where you can use a single literal value, such as a character string, number, or date.

### Examples

```
SELECT CUST_LAST_NAME, ACCOUNT_MGR_ID  
FROM CUSTOMERS  
WHERE CUSTOMER_ID = :P_CUST
```

This statement enables you to enter a specific customer number at run time. For example, if you enter 102, the WHERE clause uses the value 102 to restrict data fetched and to fetch the one customer that has ID 102.

```
SELECT CUST_LAST_NAME, ACCOUNT_MGR_ID  
FROM CUSTOMERS  
ORDER BY DECODE(:SORT, 1, CUST_LAST_NAME,  
                  2, NLS_TERRITORY,  
                  ACCOUNT_MGR_ID)
```

## Using Bind References (continued)

```
SELECT CUST_LAST_NAME, ACCOUNT_MGR_ID  
FROM CUSTOMERS  
ORDER BY DECODE (:SORT, 1, CUST_LAST_NAME,  
2, NLS_TERRITORY,  
ACCOUNT_MGR_ID)
```

This statement enables you to output different versions of the report:

- Enter 1 for the parameter SORT to display the output ordered by customer name.
- Enter 2 to display the output ordered by NLS\_territory.

## Where Can You Not Use Bind References?

You cannot use a bind reference to:

- Replace a column name in the SELECT clause, although you can reference a value, such as the contents of a parameter:

```
SELECT LAST_NAME, SALARY * :P_RATE  
FROM EMPLOYEES
```

- Replace any part of a FROM clause
- Replace reserved words or clauses

Note: In particular to running report in Oracle Apps, we need to create atleast one bind variable P\_CONC\_REQUEST\_ID with the datatype Number. The concurrent manager passes the concurrent request ID to your report using this parameter.

# Using Lexical References

- Use to substitute any part of the query.

```
SELECT CUST_LAST_NAME, ACCOUNT_MGR_ID  
FROM CUSTOMERS  
&P_WHERE_CLAUSE  
&P_ORD_CLAUSE  
  
SELECT CUST_LAST_NAME, ACCOUNT_MGR_ID  
FROM CUSTOMERS  
&P_WHERE_ORD_CLAUSE  
  
SELECT &P_CUSTNAME CUST, &P_ACCTMGR MGR  
FROM &P_TABLE
```

- Make sure that the number and data types match at run time.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Using Lexical References

Use a lexical reference to replace any clause in a SELECT statement, or even to replace the entire statement.

### Examples

The following statements use lexical references to substitute parts of the query at run time:

- To specify a WHERE clause, ORDER BY clause, or both at run time (as two separate parameters):

```
SELECT CUST_LAST_NAME, ACCOUNT_MGR_ID  
FROM CUSTOMERS  
&P_WHERE_CLAUSE  
&P_ORD_CLAUSE
```

## Using Lexical References (continued)

- To specify a WHERE clause, ORDER BY clause, or both at run time (as one parameter):

```
SELECT CUST_LAST_NAME, ACCOUNT_MGR_ID
FROM CUSTOMERS
&P_WHERE_ORD_CLAUSE
```
- To specify two column names and the table names at run time:

```
SELECT &P_CUSTNAME CUST, &P_ACCTMGR MGR
FROM &P_TABLE
```

In this example, you must create the parameters first and provide each with an initial value to ensure that the SELECT statement is syntactically correct when parsed.

**Note:** When you use lexical references in the SELECT list, you must, at run time, specify the same number of items of the correct data type, as defined in the Data Model.

## Hints and Tips When Referencing Parameters

Always do the following:

- Specify column aliases when substituting column names
- Create lexical parameters explicitly in the Object Navigator
- Enter an initial value for parameters that affect query validation when NULL



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Hints and Tips for Referencing Parameters

If you substitute the name of a column or expression in the SELECT list with a lexical reference in the SELECT list, always add a column alias after the reference.

The alias serves as a placeholder in the Data Model and can help to prevent confusion that may arise when the column name in the SELECT list does not match the report column object.

Remember that Reports Builder automatically creates a parameter when you make a reference to a nonexistent parameter.

When you create a lexical parameter, you may need to include an initial value before referencing the parameter in the query, because depending on your use of the parameter, a NULL value can cause a syntax failure.

## Hints and Tips for Referencing Parameters (continued)

### Example:

The following statement does not cause a syntax error when P\_ORDER\_CLAUSE is NULL.

```
SELECT ... FROM ...
&P_ORDER_CLAUSE
```

The following statement does cause a syntax error when P\_ORDER\_CLAUSE is NULL.

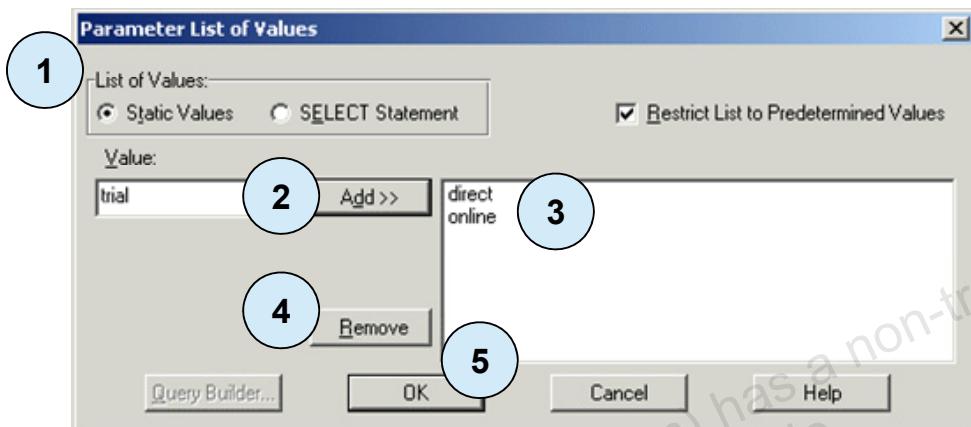
```
SELECT ...
FROM ...
ORDER BY &P_ORDER_CLAUSE
```

Remember to define an initial value for lexical parameters that affect the validity of the statement, and to validate those that you allow a user to enter at run time. Initial values are required for lexical parameters used for column names, table names, and incomplete clauses as shown in the example above.

**Note:** Always use column aliases when substituting column names with lexical references.

# Creating a List of Values

## Static list of values



**ORACLE®**

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Creating a List of Values

You can create a list of values from which users select a valid value at run time. You can restrict users to only those values in the list, or allow them to enter a different value.

For bind parameters, the list can be a static list of values or a dynamic list that selects values from the database at run time.

For lexical parameters, you can enter a static list of values.

1	Choose Static Values.
2	Enter a value and click Add.
3	Repeat for each value to build the list of values.
4	To remove a value; select a value and click Remove.
5	Select OK to accept the list and exit.

## **Creating a List of Values (continued)**

To create a static list of values:

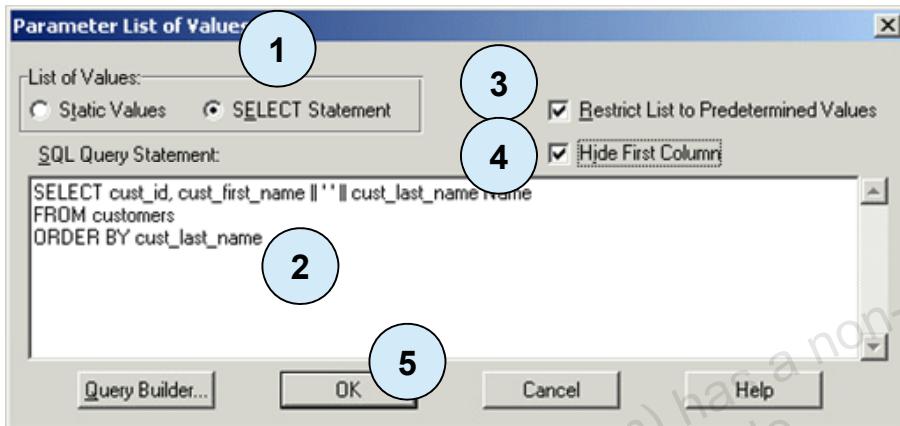
1. In the Parameter Property Inspector, Select the List of Values property. The Static Values option button is selected by default.
2. Enter a value in the Value field and click Add.
3. Repeat for each value you want in the list.

**Note:** To remove a value, select the value in the list and click Remove.

Set the Restrict List to Predetermined Values property to determine whether to prevent users from entering any value not included in your list. By default, this property is Yes. If you set the property to No, users can still select a value from the list, but they can also enter other values or leave the value blank.

# Creating a List of Values

## Dynamic list of values



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Creating a List of Values (continued)

To create a dynamic list of values:

1. In the Parameter Property Inspector, select the List of Values property.
2. Choose SELECT Statement.
3. The SQL Query Statement field is displayed.

Enter a query to populate the list of values. You can include more than one column. The parameter takes its value from the first column in the list and the column values appear concatenated in the list at run time.

4. Set the Restrict List to Predetermined Values property, as required.

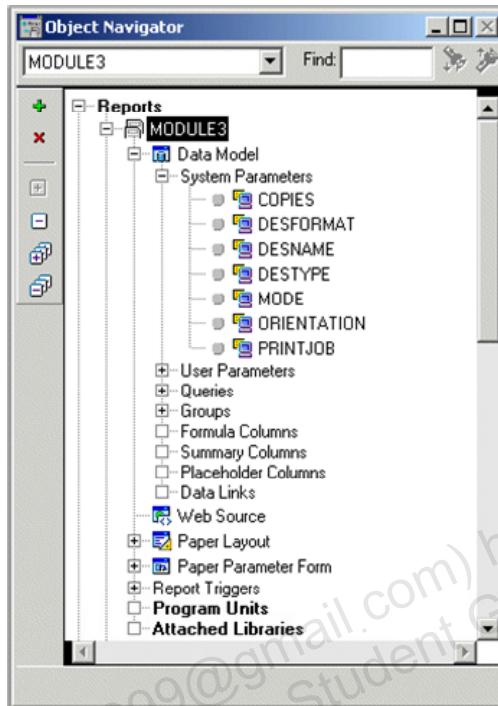
1	Choose SELECT Statement.
2	Enter a valid query.
3	Select or clear the Restrict List to Predetermined Values check box.
4	Hide the first column from the list.
5	Click OK to accept the query and exit.

## **Creating a List of Values (continued)**

To see a more meaningful list of descriptive values, instead of the primary or foreign key column that you reference in the query, select Hide First Column.

Make sure that the primary or foreign key column is first in the SQL query statement, because this is still the value that the parameter object contains at run time.

# Referencing System Parameters



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Referencing System Parameters

Reports Builder provides *system parameters* to allow you to modify standard run-time settings for each report definition. Each parameter has a default value that you can modify.

There are currently seven system parameters. You cannot delete them.

### Technical Note

The following system parameters have been deprecated in Oracle Reports:

BACKGROUND, CURRENCY, DECIMAL and THOUSANDS.

## Referencing System Parameters (continued)

The system parameters are:

Parameter	Use to specify	Values	Default
COPIES	Number of copies that should be made if the report is printed	Any integer	1
DESFORMAT	Either the output format for the report, or the printer definition to use when formatting the report when DESTYPE=FILE and DESNAME= <i>filename</i>	Examples: PDF, HTML, HTMLCSS, RTF, XML	dflt
DESNAME	Name of output device, such as <i>filename</i> , printer name, mail userid		<reportname>.lis
DESTYPE	Type of device that will receive the report output	Cache, localfile, file, printer, sysout, mail, oracleportal, ftp, webdav, screen	Cache
MODE	Whether report executes in bitmapped or character mode	Default, bitmap, Character	Default
ORIENTATION	The print direction of printer output	Default, portrait, Landscape	Default
PRINTJOB	Whether print job dialog box is displayed at run time, if destype=file or printer	Yes, No	Yes

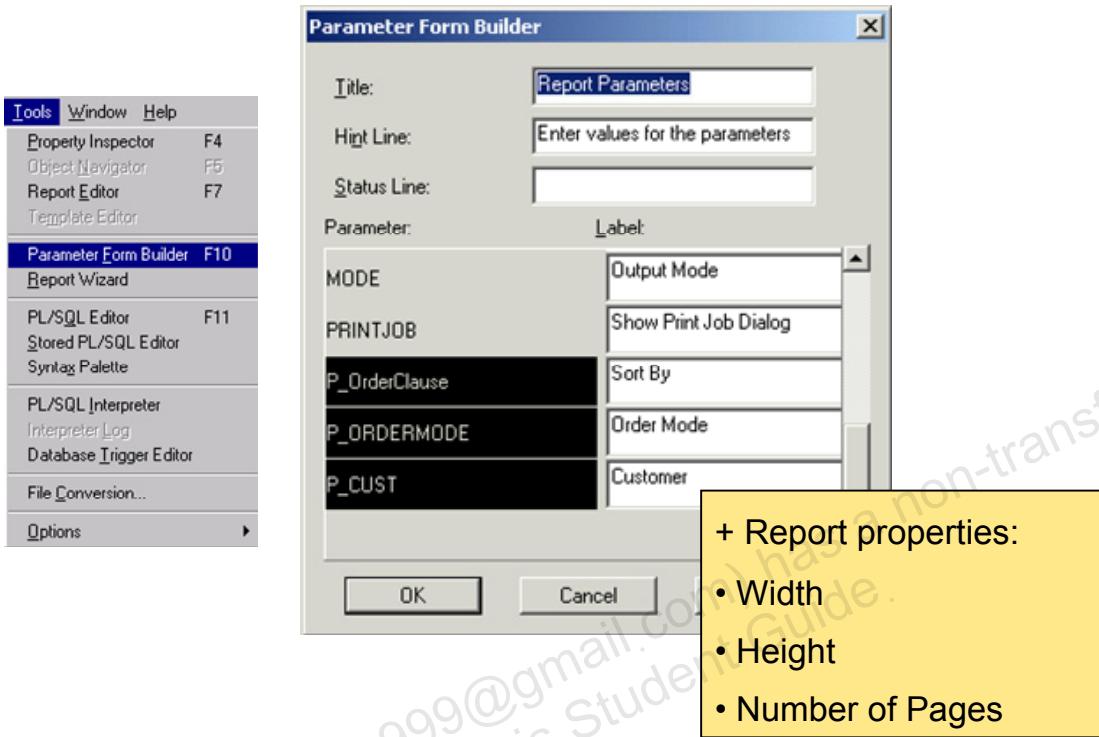
In Oracle Reports, valid before version 11g.

In Reports Builder (rwbuilder), you can still set the DESTYPE system parameter to SCREEN or PREVIEW to format a report to display screen fonts or printer (preview) fonts in the Reports Builder user interface.

The following command line options have been deprecated in Oracle Reports:

Command Line Option	Option to use instead
CURRENCY	Set the NLS_CURRENCY environment variable.
DECIMAL	Set the NLS_NUMERIC_CHARACTERS environment variable.
THOUSANDS	Set the NLS_NUMERIC_CHARACTERS environment variable.

# Building a Paper Parameter Form



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Building a Paper Parameter Form

The paper reports that you created in previous lessons ran successfully without a parameter form. However, when you created parameters, a default run-time parameter form appeared to enable you to change the values at run time.

You can build your own parameter form, select the parameters that you want to display, and then customize the appearance of the parameter form in the Report editor.

### How to Build a Parameter Form

1. Select menu item Tools > Parameter Form Builder.
2. The Parameter Form Builder dialog box appears.

If desired, modify the title, hint, and status line text. These lines are displayed at the top of the parameter form at run time.

3. Select or deselect parameters in the parameter section as necessary.

**Note:** Selected parameters are displayed in the dialog box as highlighted on a black background. Deselected parameters do not appear in the parameter form.

4. Modify label text of the selected parameters, if you wish.
5. Click OK to create the parameter form.

## **Building a Paper Parameter Form (continued)**

If you create many parameters in your report, the run-time parameter form can spread over more than one page. You can view subsequent pages at run time by clicking Next.

### **Setting Report Properties for the Parameter Form**

There are three properties in the Report Property Inspector, under the Parameter Form Window node, that specifically apply to the parameter form.

<b>Report Property</b>	<b>Use to</b>
Width	Define the width of the parameter form that the user sees at run time
Height	Define the height of the parameter form that the user sees at run time
Number of Pages	Define the number of physical pages that make up the parameter form

## Customizing a Paper Parameter Form



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Customizing a Paper Parameter Form

You can customize the parameter form layout in a similar way to customizing the paper report layout. You can modify colors and fill patterns (except in fields), move or delete existing objects, create additional objects, import file contents such as a company logo, and so on.

#### Creating Parameter Fields

A *parameter field* is a placeholder for a parameter value on the run-time parameter form, in the same way that a field is a placeholder for a column value in the layout.

You cannot modify the color or fill patterns of a parameter field. However, the Property Inspector enables you to modify all parameter properties.

Reports Builder creates one field for each parameter that you select in the Parameter Form Builder. If a parameter in your Data Model does not have an associated field, it does not appear in the run-time parameter form because it does not have any display attributes.

You can create additional fields by using the field tool in the parameter form toolbar and sourcing the field to an existing parameter.

## Customizing a Paper Parameter Form (continued)

### Creating Parameter Boilerplate Objects

*Parameter boilerplate* is text or graphics that either you create or the Parameter Form Builder creates by default.

The Parameter Form Builder creates parameter boilerplate objects for the labels of each parameter field as well as for the title, hint, and status lines.

You can create additional boilerplate objects to customize your parameter form, for example, create report heading pages, modify default labeling, insert images, and so on.

**Note:** You cannot create a file link in the parameter form; however, you can insert file contents, as you can in the layout.

## Using Parameter Form HTML Extensions

- Boilerplate text with HTML tags
- Parameter fields with JavaScript
- Parameter Form header
- Parameter Form footer



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Using Parameter Form HTML Extensions

HTML Parameter Form extensions enable you to enhance your run-time paper parameter form with HTML tags and JavaScript. Using HTML, you can create boilerplate text. Incorporating JavaScript extends your parameter form to support client-side validation eliminating network roundtrips. You can do the following:

Extension	Purpose
Create boilerplate text with HTML tags	Add hyperlinks or HTML tagged text
Insert parameter fields with Javascript	Define input or select events such as validation or raising errors
Create a Paper Parameter Form header	Place a logo or standard links in the header of the HTML Parameter Form
Create a Paper Parameter Form footer	Place a logo or standard links in the footer of the HTML Parameter Form

## Using Parameter Form HTML Extensions (continued)

To create a boilerplate text object for HTML tags:

1. Create a boilerplate text object using the Text tool.
2. Enter or import the desired HTML code.
3. Open the Property Inspector of the text object and set the Contains HTML Tags property to Yes.

**Note:** The text only shows for HTML output formats.

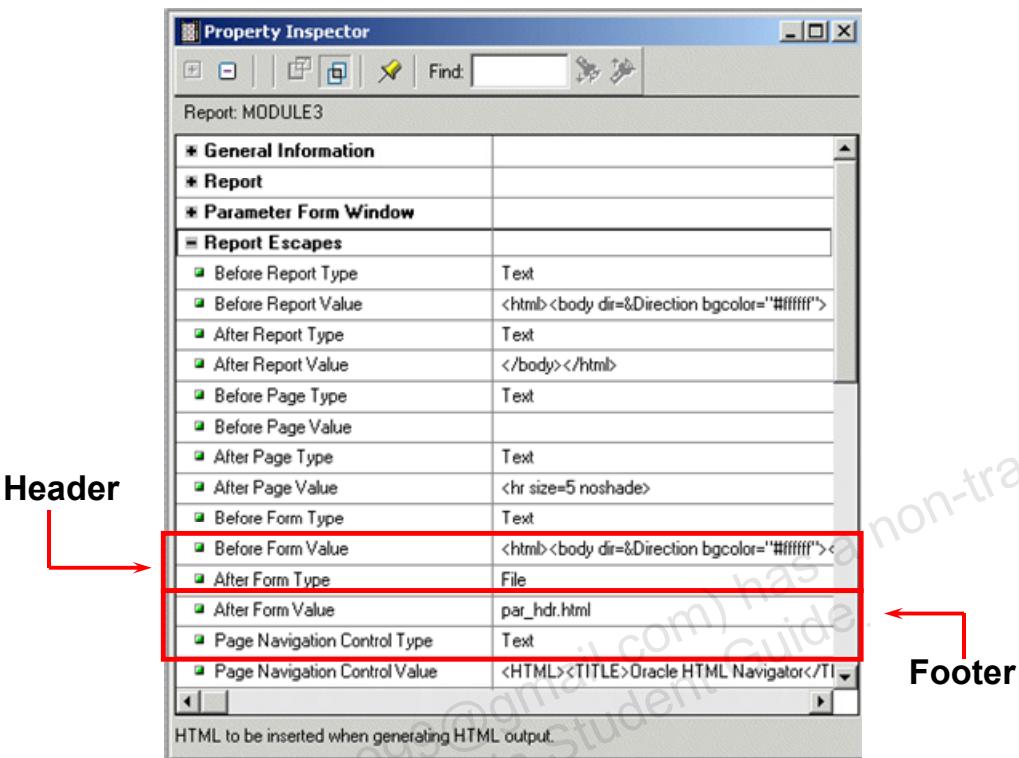
To create HTML parameter form fields with input or select events:

1. Create or edit a Parameter Form field.
2. Open the Property Inspector of the Parameter Form field and set the Additional Attributes(HTML) property to a valid JavaScript event handler.

**Note:** In some cases, such as raising messages, the JavaScript code may have to be entered in the Before Form report properties. To insert the JavaScript code in the Before Form report properties:

- Open the report Property Inspector.
- Set the Before Form Type property to Text if you enter the JavaScript code, or to File if you will import the code from a file.
- Set the Before Form Value property by clicking the ... button to either enter the JavaScript, or select the HTML file with the JavaScript.

## Parameter Form Header and Footer



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Parameter Form Header and Footer

To create an HTML parameter form header:

1. Open the Property Inspector of the report.
2. Set the Before Form Type property to Text if you will enter the header, or to File if you will import the header from a file.
3. Set the Before Form Value property by clicking the ... button to either enter the HTML code or select an HTML file to import.

To create an HTML parameter form footer:

1. Open the Property Inspector of the report.
2. Set the After Form Type property to Text if you will enter the footer, or to File if you will import the footer from a file.
3. Set the After Form Value property by clicking the ... button to either enter the HTML code, or select an HTML file to import.

# Summary

In this lesson, you should have learned how to:

- Define and use bind and lexical parameters
- Create a list of values
- Reference system parameters
- Modify parameter values at run time, using:
  - Command line arguments
  - Run-time parameter form
- Create a run-time paper parameter form:
  - Build it using the Parameter Form Builder
  - Customize it using the Report Editor, HTML



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Summary

- Parameter types:
  - User parameters:  
References: bind or lexical  
List of values: dynamic (bind only) or static
  - System parameters
- Modifying parameter values at run time:
  - Pass the parameter on the RWRUN command line
  - Allow users to change values in the run-time parameter form
- Creating a run-time parameter form:
  - Build a parameter form with the Parameter Form Builder
  - Customize the paper parameter form layout in the Report editor
  - Evaluate using HTML Parameter Form extensions

## Practice 12 Overview

- Modifying an existing report to use a bind parameter
- Adding a dynamic list of values
- Adding a lexical parameter
- Adding a static list of values
- Creating a customized parameter form



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Practice Session Overview: Lesson 12

This practice session contains:

- Modifying an existing report to use a bind parameter
- Adding a dynamic list of values
- Adding a lexical parameter
- Adding a static list of values
- Creating a customized parameter form

In this practice session, you create bind and lexical parameters to restrict records in the report. You add lists of values to select valid values at run time.

You then create your own default parameter form and customize it to include extra text and graphics.

## Practice Session: Lesson 12

1. Open report p13q3.rdf.
  - a. Modify the data model and add the following columns from the CUSTOMERS table.  
Create a new group for the customer information above the G\_ORDER\_ID group.

Column	Label	Width
cust_first_name    ' '    cust_last_name	Name	30
cust_email	Email	20

Create a reference to a bind parameter P\_CUSTOMER in the query to restrict orders to the parameter value you input at run time.

- Note how Reports creates the parameter automatically.
- b. Run and test the report with valid customer IDs.
  - c. Save the report as p16q1.rdf. Do not close it
  2. Continue with report p16q1.rdf.
    - a. Add a list of values to select any valid CUSTOMER\_ID in the CUSTOMERS table that has an order in the ORDERS table. Do not allow users to enter any other value. The list should display customer names, not CUSTOMER\_ID numbers. Ensure that each customer name appear only once in the list, even if the customer has several orders.
    - b. Save the report as p16q2.rdf. Do not close it for the moment.
    - c. Run the paper layout to test the report. Run the Web layout as well.
  3. Open report p5q3.rdf.
    - a. Create a lexical parameter to enable users to order the data by different columns (or not at all) at run time. Include an ORDER BY clause in the query with a lexical reference to the parameter. Enter an initial value for the parameter to order by the employee's last name.
    - b. Add a static list of values for the parameter P\_ORDER. Code two or three alternative ORDER BY clauses. Allow any clause to be entered at run time.
    - c. Run the paper layout to test.  
Select an ORDER BY clause from the list.  
Enter your own ORDER BY clause.  
Delete the initial parameter value.
    - d. Save the report as p16q3.rdf and close it.

## **Practice Session: Lesson 12 (continued)**

4. Continue with report p16q2.rdf.
  - a. Create a default parameter form.  
Include a Title message but not a Hint message.  
Display the P\_CUSTOMER parameter.
  - b. Use your imagination to customize the form in the Parameter Form editor.  
Make sure that the parameter P\_CUSTOMER is placed in a prominent position for user entry.  
Add some graphics; modify colors and fill patterns.
  - c. Add a format mask to the field displaying line\_total.
  - d. Run the paper layout to test. Select a customer from the list.
  - e. Save the report as p16q4.rdf and close it.

# 13

## Coding PL/SQL Triggers

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Objectives

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

- Describe the different types of triggers
- Describe sample uses of triggers
- Write and reference common code
- Create a PL/SQL library
- Publish a report as a result of a database event



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Overview

This lesson identifies different trigger types and creates conditional trigger code to control report output. It also explains how to improve productivity and maintenance by sharing and reusing commonly referenced code. Finally, you will learn about publishing reports based on database events.

# Types of Triggers in Reports

- Report:
  - Five triggers
  - Report Triggers node in Object Navigator
- Data Model:
  - Formula (column)
  - Filter (group)
  - Parameter validation
- Layout: Format trigger on most objects



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Types of Triggers in Reports

You can code PL/SQL in a variety of triggers in a report to provide conditional or additional processing at run time. This lesson discusses examples of each type of PL/ SQL trigger.

There are three trigger types in Oracle Reports: Report, Data Model, and Layout.

- Report:

- A set of five report-level triggers. Each trigger fires at a different stage of the report execution.
- Access report triggers from the Report Triggers node in the Object Navigator.

## Types of Triggers in Reports (continued)

- **Data Model:**

- Column formula fires each time the column is processed.
- Group filter fires for each record in the group.
- Parameter validation fires when the run-time parameter form is displayed and when the user leaves the parameter field.
- Access Data Model triggers in the Property Inspector of a Data Model object—column, group, or parameter—or the corresponding layout object—field, repeating frame, or parameter field.

- **Layout:**

- Format triggers on most paper layout objects (excluding anchors). Each trigger fires as the layout object is processed.
- Access format triggers in the Property Inspector of a paper layout object.
- For Web layouts, use the formatTrigger attribute in the `rw:field` tag.

**Note:** Reports Builder does not allow data manipulation language (DML) commands—`INSERT`, `DELETE`, `UPDATE`—in layout format triggers.

# Trigger Code

The screenshot shows two Oracle Developer windows. The top window, titled 'L13FIELD: Program Unit - P\_1VALIDTRIGGER', has 'Object' set to 'Parameter' and contains the following PL/SQL code:

```
function P_1ValidTrigger return boolean is
begin
    return (TRUE);
end;
```

A blue oval to the right of the code contains the text: **Boolean:  
true  
false**. A red box highlights the word 'boolean' in the code.

The bottom window, titled 'L13FIELD: Program Unit - PU\_021', has 'Object' set to 'Column' and contains the following PL/SQL code:

```
function CF_1Formula return Number is
begin
    -
end;
```

A blue oval to the right of the code contains the text: **Character  
Number  
Date**. A red box highlights the word 'Number' in the code.

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

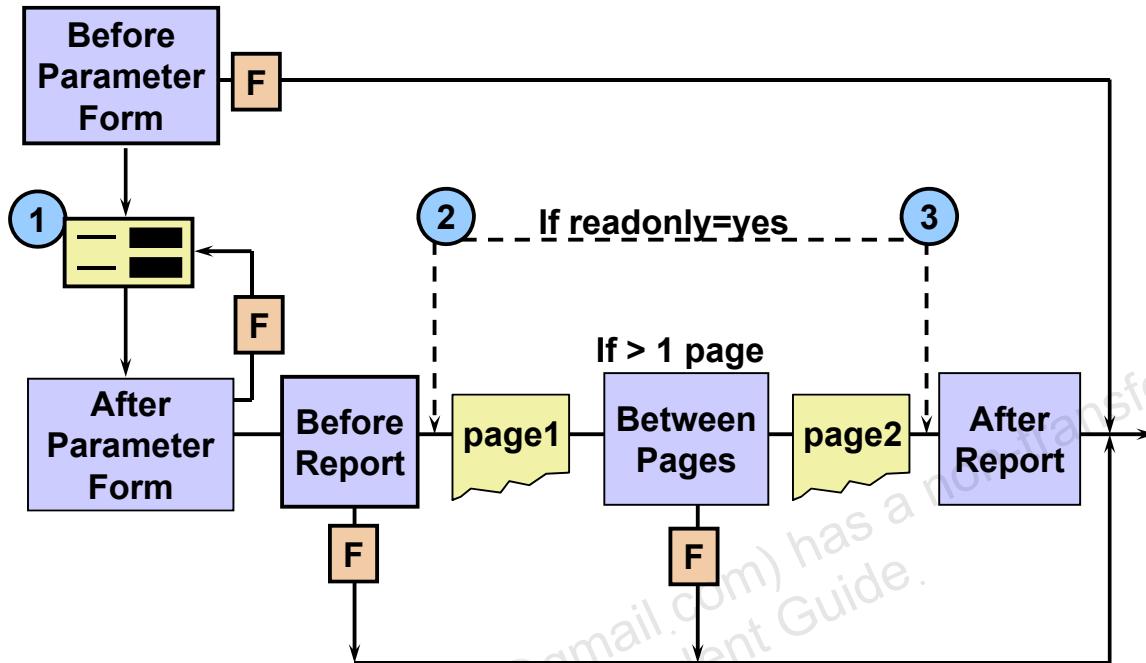
## Trigger Code

When you create a trigger such as a report trigger or a PL/SQL formula, the Program Unit editor supplies a template for the function block, giving the basic syntax that you require.

You can change the name of the function.

The value that you return must match the return data type in the function declaration.

# Using Report Triggers



**ORACLE®**

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Using Report Triggers

Report triggers enable you to execute PL/SQL functions at specific times during the execution and formatting of your report. Using these triggers, you can:

- Customize the formatting of your report
- Perform initialization tasks
- Access the database

Report triggers must return a Boolean value—true or false.

1	If After Parameter Form returns FALSE, the cursor returns to the run-time parameter form. If the run-time parameter form is not displayed, control returns to the calling program.
2	If READONLY=Yes, Reports Builder implicitly sets the readonly transaction (SET TRANSACTION READONLY) after the Before Report Trigger fires.
3	If READONLY=Yes, Reports Builder implicitly ends the read-only transaction (COMMIT) after the final page and before the After Report Trigger fires.

## Using Report Triggers (continued)

You can access the report triggers from the Object Navigator. Open the Report Triggers node and double-click the object icon of the required trigger to open the program unit.

Trigger	Fires	Use to
Before Parameter Form	Before run-time parameter form is displayed; fires even if parameter form is suppressed	Access and change parameter values (initial or command line), global variables; create temporary tables; insert into tables to be queried
After Parameter Form	After run-time parameter form is displayed; fires even if parameter form is suppressed	Access and change parameter values (initial, command line, or user-entered); create temporary tables; insert into tables to be queried
Before Report	Before a report executes; after queries are parsed, but before records are fetched	Carry out initialization procedures
Between Pages	Before each page is formatted, except the first page; in Previewer, fires only once for each page, even if you revisit the page	Carry out customized page formatting; insert or update values in tables
After Report	After report output is sent to its destination; fires on success only	Clean up initial processing, such as deleting temporary tables

In particular to Oracle Apps we must call FND SRWINIT from the Before Report Trigger as follows:

```
SRW.USER_EXIT('FND_SRWINIT');
```

This user exit sets up information for use by profile options and other AOL features.

You always call FND SRWEXIT from the After Report Trigger as follows:

```
SRW.USER_EXIT('FND_SRWEXIT');
```

This user exit frees all the memory allocation done in other AOL exits.

# Using Report Triggers

- After Parameter Form

Example: Build dynamic WHERE clause

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

- Query syntax

```
SELECT CUSTOMER_ID, CUSTOMER_NAME
FROM CUSTOMERS
&p_where_clause
ORDER BY CUSTOMER_NAME
```

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Using Report Triggers: Creating a Dynamic WHERE Clause

Use the After Parameter Form report trigger to build a dynamic WHERE clause depending on the value of a bind parameter that the user enters at run time.

## **Using Report Triggers: Creating a Dynamic WHERE Clause (continued)**

There are two possibilities at run time:

- The bind parameter is NOT NULL.

The WHERE clause restricts the query; therefore the Oracle server makes use of an existing index when retrieving data.

- The bind parameter is NULL.

The WHERE clause is also NULL; therefore the Oracle server uses a full table scan to retrieve all records.

## Using Data Model Triggers: PL/SQL Group Filter

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

```
FUNCTION G_empGroupFilter RETURN BOOLEAN IS
BEGIN
  IF :department_name = 'Operations' AND
    :salary > 5000 THEN
    RETURN(my_function);
  ELSE
    RETURN(TRUE);
  END IF;
END;
```

- PL/SQL filters result in all records being fetched



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Using Data Model Triggers: PL/SQL Group Filter

Use a group filter when you want to:

- Restrict records in a specific group, depending on other processing in the report; that is, when a WHERE clause in a query is not applicable
- Perform some PL/SQL processing for each record in a group

A group filter is useful when you want to restrict group values of a lower group in the group hierarchy.

The function must return a boolean value:

- True: The record is included.
- False: The record is excluded.

Reports Builder fetches all records and applies the group filter to each record. Therefore, avoid group filters as a method of restricting many records in a group.

**Note:** You cannot add a filter to a matrix cross product group.

## Using Data Model Triggers: PL/SQL Group Filter (continued)

### Example

Use a group filter that calls a function for some records in the G\_emp group, depending on the department in the master group.

1. Open the Property Inspector for the G\_emp group.
2. Change Group Filter to PL/SQL.  
The PL/SQL Filter property appears.
3. Select the PL/SQL Filter property to open the PL/SQL editor.
4. Enter the code. This filter code must return a boolean value of true or false.

**Note:** It is more efficient to use a WHERE clause in a query when possible. You cannot use a WHERE clause in this case, because you want to process all records and call the function for some.

## Using Data Model Triggers: Parameter Validation

- Example: Do not allow report output to be sent directly to a printer.

```
FUNCTION DESTYPEValidTrigger RETURN BOOLEAN IS
BEGIN
  IF :DESTYPE = 'Printer' THEN
    RETURN(FALSE);
  ELSE
    RETURN(TRUE);
  END IF;
END;
```

- You cannot reassign values to parameters or columns in this trigger.



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Using Data Model Triggers: Parameter Validation

Both system parameters and user parameters have a Validation Trigger property in which you can write a PL/SQL function to validate the value of the parameter.

The trigger code must return a boolean value—true or false—depending on whether the validation is successful or not.

If the returned value is false, the cursor remains in the parameter and the user has the following two options:

- Enter a different value
- Cancel the report

You can test the value of a parameter, but you cannot change the value. If you want to assign a new value to the parameter, use the After Parameter Form report trigger.

## **Using Data Model Triggers: Parameter Validation (continued)**

**Example:** To prevent report output from being sent directly to a printer, test the value of the DESTYPE system parameter.

1. Open the Property Inspector for the DESTYPE parameter.
2. Select Validation Trigger to open the PL/SQL editor.
3. Enter the code and make sure that the value Printer has an initial capital.

# Using Layout Triggers

Format triggers:

- Exist on most layout objects
- Can suppress an entire layout section (master group frame): No records fetched
- Can suppress the display of individual records (repeating frame): All records fetched



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Using Layout Triggers

Most paper layout objects can contain a format trigger; it is one of the common properties of a paper layout object.

Format triggers enable you to modify the display of objects dynamically at run time or to suppress display altogether.

The trigger must return a boolean value—true or false.

The following pages discuss some examples of format triggers.

### Suppressing the Entire Paper Layout for a Query

Because Reports is driven by the layout, you can use a format trigger on a group frame to conditionally suppress the entire layout associated with a query. In this case, the query does not retrieve any data.

## Using Layout Triggers (continued)

### Suppressing Individual Records

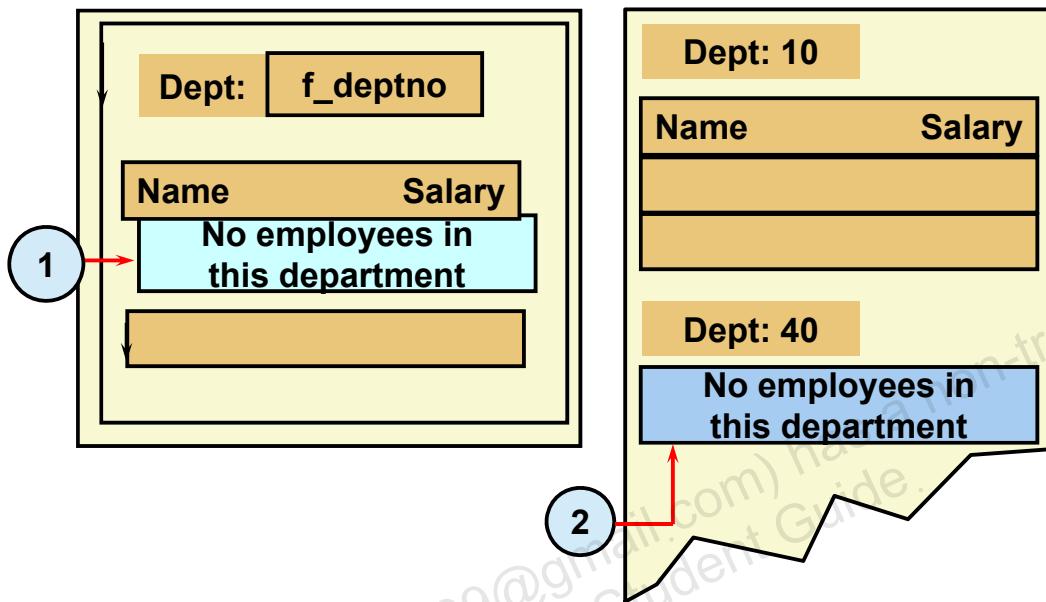
Do not use format triggers to suppress individual records that you can restrict on the Data Model. For example, a format trigger on a repeating frame does not prevent data from being retrieved, it simply determines whether or not you see it in the output.

Therefore, it is more efficient to restrict individual records using the Maximum Records property in a query or a First type of group filter when possible.

**Note:** Summaries compute against *all* data fetched by the Data Model. If you suppress records in the layout only, summaries give misleading results.

# Using a Format Trigger on a Frame

Displaying a Text String in Place of Column Headings



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Using a Format Trigger on a Frame

**Requirement: Display a Text String in Place of Column Headings**

Display a master-detail report for departments and employees. If a department has no employees, suppress the detail column heading frame and display a piece of text instead.

1	Boilerplate text: Position behind column heading frame, or position below and create collapsible anchor to column heading
2	Column heading frame does not display; suppressed by format trigger

## Coding a Format Trigger on a Frame (continued)

### Example:

1. Create a summary column, :count\_emp, in the master group that counts the number of employee records (reset = master group).
2. Create a piece of boilerplate text, "No employees in this department".
3. Position the text behind the column heading frame.
4. Code a format trigger on the column heading:

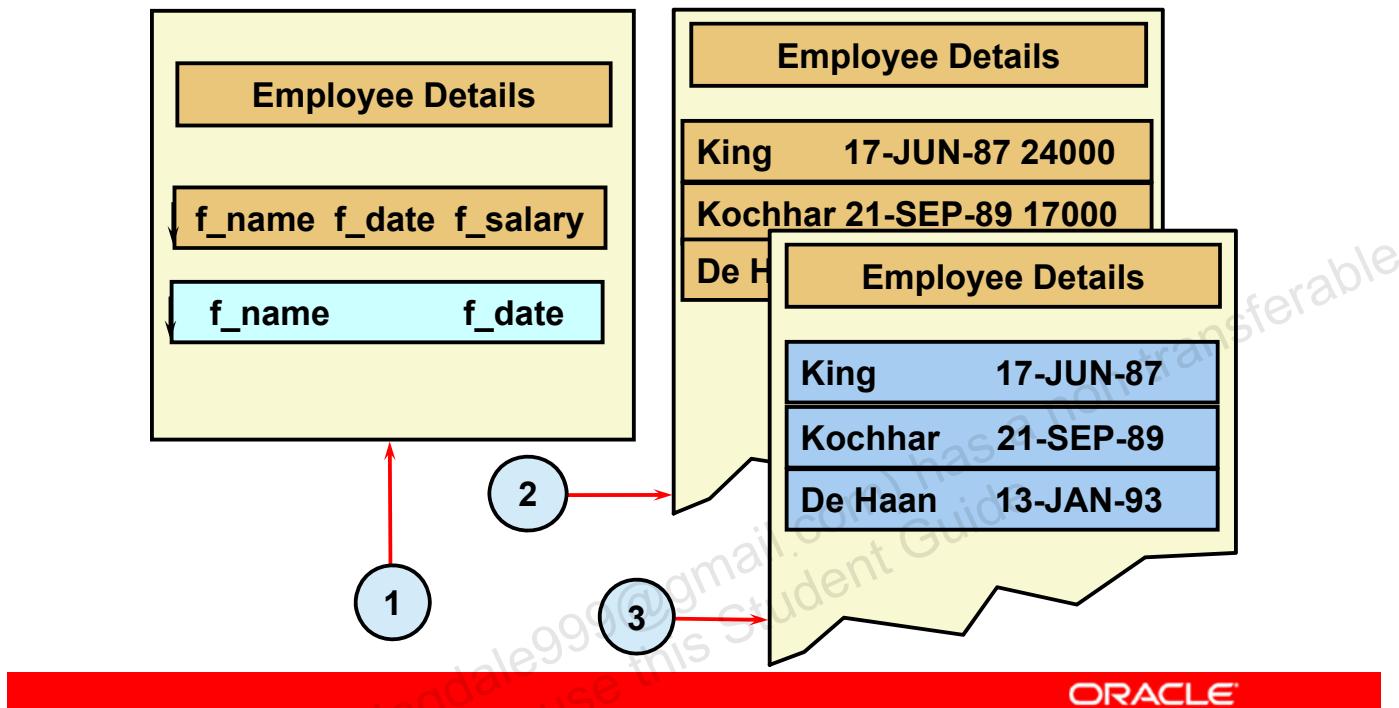
```
function hide_headings return boolean is
begin
    if :count_emp = 0 then
        return(false);
    else
        return(true);
    end if;
end;
```

5. Either code the reverse logic in the format trigger of the boilerplate text item created in step 2 or ensure that the column heading frame has a solid fill.

**Note:** You should code format triggers at the highest possible level of the paper layout hierarchy. The format trigger on the column headings frame suppresses all objects enclosed by the frame. Do not duplicate code unnecessarily by coding the same trigger on all individual objects in the frame; this can cause increased processing and maintenance overheads.

# Using a Format Trigger on a Repeating Frame

Dynamically Altering the Display of Records



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Using a Format Trigger on a Repeating Frame

**Requirement: Alter the Display of Records Dynamically:** In a report of employee details, display salaries if the user is a payment clerk, but hide salaries *and* alter the appearance of remaining fields when any other employee runs the report.

1	One repeating frame is positioned behind the other or positioned below with collapsing anchor
2	Report output run by payment clerks
3	Report output run by other employees

## Using a Format Trigger on a Repeating Frame (continued)

### Example:

1. In the layout, create two repeating frames sourced by the same group. One repeating frame must contain a salary field, the other must not.

**Note:** You can create two separate default layouts and modify the layout as required.

2. Place one repeating frame behind the other.
3. Add a format trigger to suppress the top repeating frame (the one that contains the salary field), or alternatively use conditional formatting:

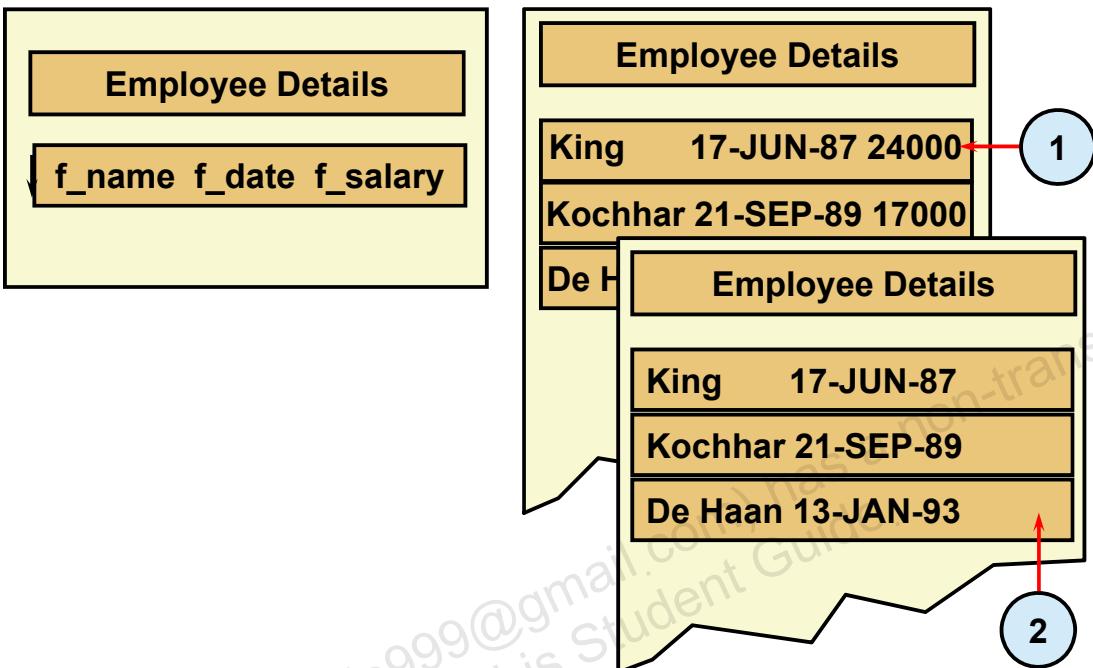
```
function pay_clerks_see_this return boolean is
begin
    if :user_type = 'PAY_CLERK' then
        return(true);
    else
        return(false);
    end if;
end;
```

**Note:** User\_type can be a column or parameter that is initialized at run time.

4. Either code the reverse logic on the other repeating frame or ensure that the top repeating frame (containing salary) has a solid fill.

# Using a Format Trigger on a Field

## Dynamically Hiding Fields



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Using a Format Trigger on a Field

**Requirement: Dynamically Hide Fields:** Given the same employee details report as before, hide the salary field without rearranging the other fields in the repeating frame. The report contains a gap in the layout when the salary field is not displayed.

**Example:** Place the format trigger on the salary field to display it only when payment clerks run the report. The code is the same as before:

```
function pay_clerks_see_this return boolean is
begin
    if :user_type = 'PAY_CLERK' then
        return(true);
    else
        return(false);
    end if;
end;
```

In this example, you do not need to create any additional layout object or code.

1	Report for payment clerks shows all fields
2	Report for other employees shows gap where salary is hidden

# Using a Format Trigger in a Web Layout

```
<tbody> <rw:foreach id="R_G_DEPARTMENT_ID_1"
                     src="G_DEPARTMENT_ID">
  <tr>
    ...
  <td>
    <rw:headers id="HFSALARY" src="HBSALARY"/>
    class="OraCellNumber">
    <rw:field id="F_SALARY" src="SALARY"
              nullValue=" "*
              formatMask="$999,999.00" formatTrigger="mystyles">
      F_SALARY
    </rw:field>
  </td>
  ...
</tr>
</rw:foreach> </tbody>
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Using a Format Trigger on a Field in a Web Layout

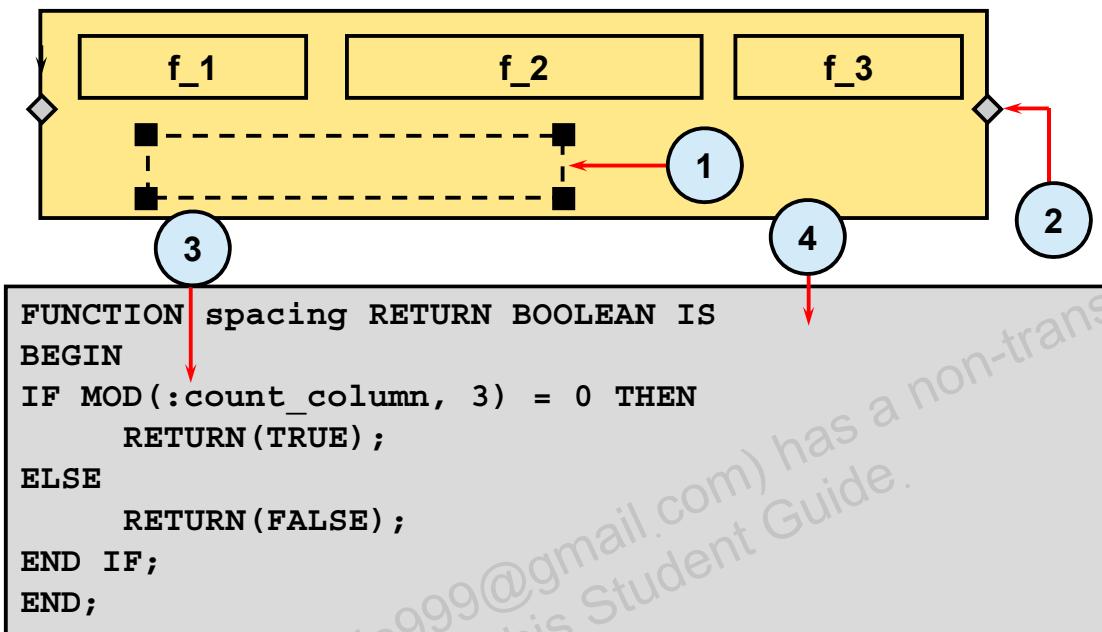
In an earlier lesson, you learned that you can customize your Reports JSPs by editing the Web source and including attributes for the Reports custom JSP tags.

Recall that the rw:field tag has an attribute called formatTrigger. You set the formatTrigger attribute to the name of a format trigger function. If the format trigger returns a Boolean value of true, the field is displayed according to the attributes specified in the trigger. The formatTrigger attribute is used to modify the display characteristics of the field and can only use specific SRW packaged functions to set the field's attributes.

You will learn about the SRW package in the next lesson.

# Using a Format Trigger on a Boilerplate Object

Insert Spacing Between Groups of Records



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Using a Format Trigger on a Boilerplate Object

**Requirement: Insert Spacing Between Groups of Records:** Display additional spacing between a defined number of records; for example, group every three records together.

1	Transparent rectangle
2	Repeating frame, Vertical Elasticity = Variable
3	Summary column name (created in Data Model)
4	Format trigger on transparent rectangle

## Using a Format Trigger on a Boilerplate Object (continued)

### Example

1. Create a transparent rectangle inside the repeating frame to increase spacing conditionally at run time.
2. In the repeating frame Property Inspector, set Vertical Elasticity to Variable.
3. In the Data Model, create a summary column named COUNT\_COLUMN that counts the number of records retrieved (Reset=Report).
4. On the rectangle, create a format trigger that references the summary column, as shown on the previous page.

**Note:** You can vary the number of records in each group dynamically at run time by creating a parameter.

# Writing Common Code

At the Report level:

- Object Navigator, Program Units
- Menu: Program > PL/SQL Editor

In a library:

- Object Navigator, PL/SQL Library
- File > New: Create new library
- File > Open: Add to existing library
- Attach library to report



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Writing Common Code

If you use the same program unit in several places in a report, or in multiple reports, consider writing the code once as a report-level program unit, or in a PL/SQL library, and then call the program unit when you require it in your PL/SQL trigger code; this enables you to maintain the code in one place.

If you have to write code that is not simple, then define the code at the report level. When you have pieces of logic that are complex and/or potentially reusable, create a procedure, or split the logic into several procedures, and then call the procedure(s) from your main code. This is a common development practice. It makes the main logic more readable and easier to understand and enables you to break up a complex problem into smaller, more manageable pieces.

Remember that you can also code stored procedures and functions in the database; you must decide whether database-side or middle-tier processing is more efficient in each case.

## Two Ways to Create Report-Level Program Units

- In the Object Navigator, select the Program Units node and create a new program unit.
- From the menu, select Program > PL/SQL Editor. Click New to create a new program unit.

## Writing Common Code (continued)

### Writing Code as Report-Level Program Units

You can create a report-level program unit that contains a function, procedure, or package, and that you call from any object in the same report. You cannot reference a report-level program unit from a different report.

### Three Ways to Create Library Program Units

- In the Object Navigator, select the PL/SQL Libraries node and create a new library.
- From the menu, select File > New > PL/SQL Library.
- From the menu, select File > Open to open an existing library and create additional program units.

### Referencing Code from a PL/SQL Library

You can reference library code in a report trigger by attaching one or more libraries to one or more report definitions, which enables you to reference the same code in multiple reports.

To attach a library to a report, open the report definition, select the Attached Libraries node in the Object Navigator, and add each library that you need in the report.

## Event-Based Reporting

- Implemented through PL/SQL stored procedures
- Uses include:
  - Running a report
  - Displaying report status
  - Canceling a report
  - Managing parameter lists

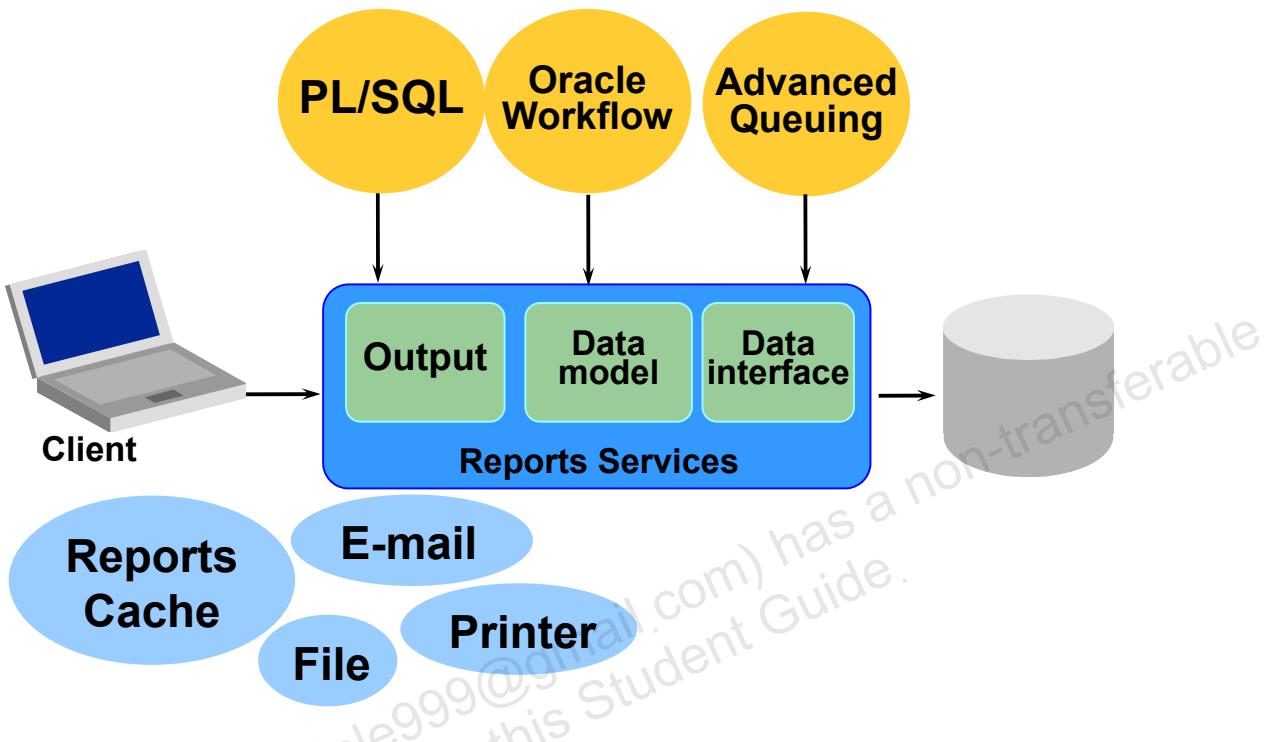


Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Event-Based Reporting

Today, it is often desirable to blend automation into business processes through the invocation of behind-the-scenes procedures and functions. These behind-the-scenes tasks can include the spontaneous production of output such as an invoice that prints automatically when an order is processed, a Web site that is automatically updated with fresh data, or an automatic e-mail alerting an employee's manager that the employee has submitted an expense report for approval. Consider this last scenario. When an employee submits an expense report, new data is inserted into the database. When this database event occurs, you want to alert the employee's manager through an e-mail or a notification in his portal page that a submittal is awaiting his approval. With the Oracle Reports event-driven publishing API, this is possible.

# Event-Driven Publishing API



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Event-Driven Publishing API

The event-driven publishing API is a PL/SQL package that provides the basic functions required for the development of procedures that respond to events in the database. The procedures are called from a database trigger to run a report when an event occurs. The trigger invokes the Reports server and the job is submitted to the server.

In addition to database events, the API can be integrated with Advanced Queuing (AQ) and Oracle Workflow.

## Event-Driven Publishing API (continued)

The API consists of several key elements:

- The *SRW Package* contains all relevant procedures and functions for submitting, checking the status, and canceling jobs, as well as manipulating parameter lists.
- The *ParamList Type* defines a parameter list. A parameter list is the main vehicle for passing values when submitting a job and is required for each job submittal.
- The *ParamList Object* is required for such features as Advanced Queueing, where a parameter list must be stored in the database so that it can be passed along with a message.

### Technical Note

Advanced Queueing is the message queuing functionality of the Oracle database. Oracle Workflow is a tool that manages complex user-based business processes. For more information refer to the Oracle Technology Network, <http://otn.oracle.com>.

See *OracleAS Reports Services Publishing Reports to the Web* manual for more information on the Event-Driven Publishing.

## Invoking a Report from a Database Event

- Create a database trigger
- Include a parameter list with the required entries:
  - GATEWAY
  - SERVER
  - REPORT
  - USERID



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Invoking a Report from a Database Event

Database triggers are the primary mechanism for invoking reports using the event-driven publishing API. The Oracle database allows you to define various scopes of triggers that fire in response to various events. One way to use event-based reporting is to create automated processes that respond to certain types of changes to data in a table.

## Invoking a Report from a Database Event (continued)

**Example:** Send a report when an employee has submitted an expense report. The code for the trigger is below.

```
CREATE TRIGGER EXP_REP_TRG AFTER INSERT OR UPDATE on
    EXP_REP FOR EACH ROW
    myPlist SRW_PARAMLIST;
    myIdent SRW.Job_Ident;
BEGIN
    IF (:new.ExpStat = 'DONE') THEN
        myPlist := SRW_PARAMLIST(SRW_PARAMETER('',''));
        srw.add_parameter(myPlist,'GATEWAY','http://...');
        srw.add_parameter(myPlist,'SERVER','repSVR');
        srw.add_parameter(myPlist,'REPORT','alert.RDF');
        srw.add_parameter(myPlist,'USERID','scott/tiger');
        srw.add_parameter(myPlist,'ExpenseID',:new.ExpID);
        myIdent := srw.run_report(myPlist);
    END IF;
END;
```

This trigger will fire after each insert or update on the EXP\_REP table. When the status of an expense report changes to DONE, the parameter list is constructed and a report is invoked. The parameter list, myPlist, contains the necessary parameters for submitting the job:

- GATEWAY provides the URL to the Reports Servlet that will be used to process the request.
- SERVER identifies the name of the Reports Server to be used in conjunction with the servlet.
- REPORT identifies the report file to be run.
- USERID identifies the user ID and password of the person running the report.

The API method RUN\_REPORT takes the parameter list containing all vital information as input, creates and submits the request, and returns the job identification record. The information in the record can be used to check the status of the job.

# Summary

In this lesson, you should have learned how to:

- Select the appropriate trigger type for your requirement: Report, Data Model, or Layout
- Identify the trigger and code needed for:
  - Building a dynamic WHERE clause
  - Validating a parameter value
  - Dynamically altering record display
  - Suppressing null fields
- Use Report-level program units and create PL/SQL libraries for common code
- Invoke a report from a database event



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Summary

### Using PL/SQL Triggers

Select the most applicable type of trigger for your requirement.

Report-level triggers fire once for each of the five events during the report execution.

Data Model triggers can affect which records are processed and how they are processed.

Layout triggers affect the way in which objects are formatted.

## **Summary (continued)**

### **Objects That Must Return a Value**

The following table is a reminder of the Reports Builder triggers that must always return a value, and the values that are acceptable.

<b>Trigger Type</b>	<b>Valid Values</b>	<b>Result if False</b>
Report trigger	True, False	Abort execution
Group filter	True, False	Do not process record
Validation trigger	True, False	Return to parameter form
Format trigger	True, False	Do not format object
PL/SQL Formula (column)	Computation: Character, Number, Date	N/A

### **Writing Common Code**

Use report-level program units or PL/SQL libraries for common code requirements to improve productivity and maintenance.

### **Event-Driven Publishing**

Use the PL/SQL procedures and functions in the event-driven publishing API to automatically submit jobs to OracleAS Reports Services when an event occurs in the database.

## Practice 13 Overview

- Creating a format trigger to display different paper layouts conditionally
- Creating and using a report-level PL/SQL function
- Creating and using an external PL/SQL library
- Creating a PL/SQL group filter



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Practice Session Overview: Lesson 13

This practice session contains:

- Creating a format trigger to conditionally display different layouts
- Creating and use a report-level PL/SQL function
- Creating and use an external PL/SQL library
- Creating a PL/SQL group filter

In this practice session, you create a format trigger to conditionally flag one of two paper layouts for the same data. You also conditionally flag some records in a report by creating and referencing a report-level (local) PL/SQL function. You then move the code into a new PL/SQL library and alter the reference to use the library function instead.

If you have time, restrict the number of records processed in a report by creating your own group filter.

## Practice Session: Lesson 13

1. Create a new report, using p19q1.sql.
  - a. Display the records in two different layouts, showing different columns in each.
  - b. Create a Group Above layout using the following columns from the query:

Group	Column
G_CUSTOMER_ID	customer_id
G_ORDER_ID	order_id
	order_date
	order_mode
	order_total

- c. Create a Tabular layout using the following columns from the query

Column
order_id
order_date
order_total

- d. Create a user parameter, user\_type, that has two possible values: Sales Representative and Sales Clerk.
- e. Ensure that only the Group Above layout is displayed when the user type is Sales Representative, and that only the Tabular layout is displayed when the user type is Sales Clerk.

**Hint:** As an alternative to coding format triggers, you can also use conditional formatting.

To test the result, display the USER\_TYPE parameter in the margin of the report. Make sure that you can enter a parameter value at run time.

- f. Save the report as p19q1.rdf.

## Practice Session: Lesson 13 (continued)

2. Continue with the previous report.
  - a. Modify the report so that the Tabular (Sales Clerk) layout displays an extra column that indicates, with an asterisk, those orders that have an order status of 5.
  - b. Remember to change user type to Sales Clerk during testing.
  - c. Move the function code into a new PL/SQL library named p19lib.pll
  - d. Run the report.
  - e. Save the report to p19q2.rdf.
3. Continue with the previous report.
  - a. Modify the same layout to display additional spacing between every two records.
  - b. Save the report as p19q3.rdf and close it.
4. Open report p19q1.rdf.
  - a. Write your own group filter to restrict the number of customers displayed, depending on the value of a parameter that you enter at runtime.  
**Hint:** You need to create two parameters, to keep track of how many records have been processed, compared to the cutoff number of records required. Make sure that you can enter the cutoff parameter at run time.
  - b. Test the report several times by changing the cutoff parameter at run time.
  - c. Save the report as p19q4.rdf.

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

Shital jadhav (shitaljagdale99@gmail.com) has a non-transferable  
license to use this Student Guide.

# Extending Functionality Using the SRW Package

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Objectives

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

- Describe the package contents
- Output messages at run time
- Execute a drill-down report
- Create a simple table of contents
- Create and populate temporary tables
- Modify visual attributes dynamically



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Overview

This lesson provides examples of the contents and use of the Reports Builder built-in package. The package contains program units that you can reference in your PL/SQL code.

## Contents of the SRW Package



ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Contents of the Reports Builder Built-in Package

The Reports Builder built-in package, known as the SRW package, contains procedures, functions, and exceptions that can help you to do the following:

- Control report execution
- Output messages at run time
- Initialize layout fields
- Create a table of contents
- Perform DDL statements (create or drop temporary tables)
- Dynamically set format attributes, such as font style and fill patterns

### Referencing the Contents of the Package

You can reference any part of the SRW package in a PL/SQL statement in any program unit of a report, such as a paper layout format trigger or a report-level trigger.

## **Contents of the Reports Builder Built-in Package (continued)**

### **Rules**

- You can only call the contents of the SRW package from within Reports Builder. Other tools, such as Forms Builder, do not recognize this package.
- You must always reference an SRW procedure, function, or exception by preceding it with the package name, SRW; for example, SRW.MESSAGE or SRW.DO\_SQL.

# Outputting Messages

- Warning:

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

- Error:

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

- Exceptions:

- SRW.INTEGER\_ERROR
- SRW.NULL\_ARGUMENTS



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Outputting Messages

The SRW.MESSAGE packaged procedure displays a message with a code and text string, which you specify. The message is displayed in the following format:

MSG-code : text

The message is displayed in a small dialog box. The user must acknowledge the message before continuing.

### Message Code and Text

You can enter a code number from zero up to ten digits. If you enter a number of fewer than five digits, the code is displayed with leading zeros.

You can enter a text string to a maximum of 190 characters, excluding the code number. You can embed extra spaces to display your message neatly in the message dialog box. Reports does not suppress extra spaces in the message text.

## Outputting Messages (continued)

### Is It a Warning or an Error?

SRW.MESSAGE does not implicitly terminate the report execution. You can choose to issue a warning message that enables the report to continue execution after the user accepts the message.

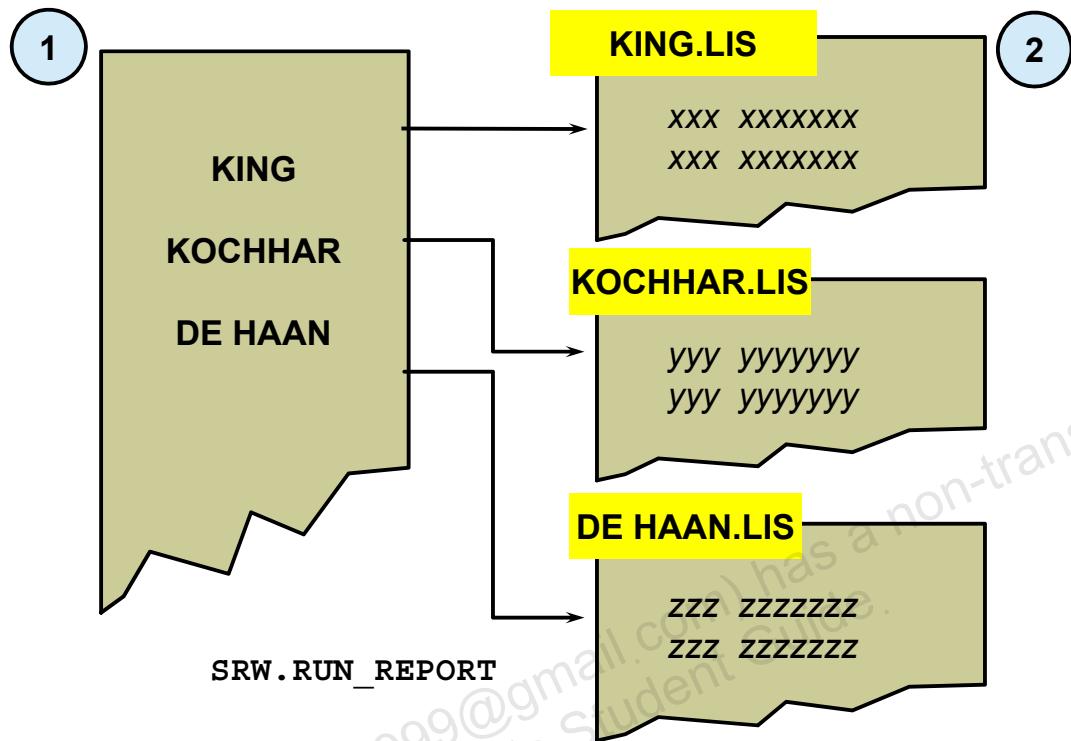
To force the report to terminate after outputting the message, raise the exception SRW.PROGRAM\_ABORT.

**Note:** You cannot suppress or replace the default message that Reports Builder gives if you abort a report. Your own error message augments the existing messages.

### Causes of Exceptions

Exception Name	Cause
SRW.INTEGER_ERROR	You entered a code that is not a numeric integer.
SRW.NULL_ARGUMENTS	You omitted the code number or text message or both.

## Executing a Nested Report



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Executing a Nested Report

SRW.RUN\_REPORT invokes RWRUN with the string that you specify. Use SRW.RUN\_REPORT to execute a second report from within a report process. For example:

- Output parts of a report to different destinations
- Divide a large report into several smaller reports that can be run conditionally

Both reports use the same process. SRW.RUN\_REPORT starts only one run-time process; it does not start a new process for each report execution.

1	Output from master output
2	Output from detailed report

## **Executing a Nested Report (continued)**

### **Example**

Run a report called MANAGERS that retrieves the employee record for each manager. Call the EMPS report (for example, from a group filter). EMPS retrieves the employee records of all employees managed by the current manager only.

Sample code to implement this example appears on the following page.

## Executing a Nested Report

- Example:

```
SRW.RUN_REPORT
  ('Report=EMPS
  DESTYPE=FILE
  DESNAME=' || :LAST_NAME || '.LIS
  BATCH=YES
  MGRNO=' || TO_CHAR (:MANAGER_ID)) ;
```

- Exceptions:

- SRW.RUN\_REPORT\_FAILURE
- SRW.RUN\_REPORT\_BATCHNO

- Function: SRW.GETERR\_RUN



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Executing a Nested Report (continued)

### Example Function Code

```
FUNCTION MGR REP RETURN BOOLEAN IS
BEGIN
  SRW.RUN_REPORT
    ('REPORT=EMPS
    DESTYPE=FILE
    DESNAME=' || :LAST_NAME || '.LIS
    BATCH=YES
    MGRNO=' || TO_CHAR (:MANAGER_ID)) ;
  RETURN (TRUE) ;
EXCEPTION
  WHEN SRW.RUN_REPORT_FAILURE THEN
    SRW.MESSAGE(100, 'Error executing EMPS report') ;
    RAISE SRW.PROGRAM_ABORT;
END ;
```

**Note:** Use LAST\_NAME to create the output filename for each manager; use MANAGER\_ID to restrict the EMPS query.

## Executing a Nested Report (continued)

Reports Builder raises a specific exception that applies to the SRW.RUN\_REPORT procedure in two cases:

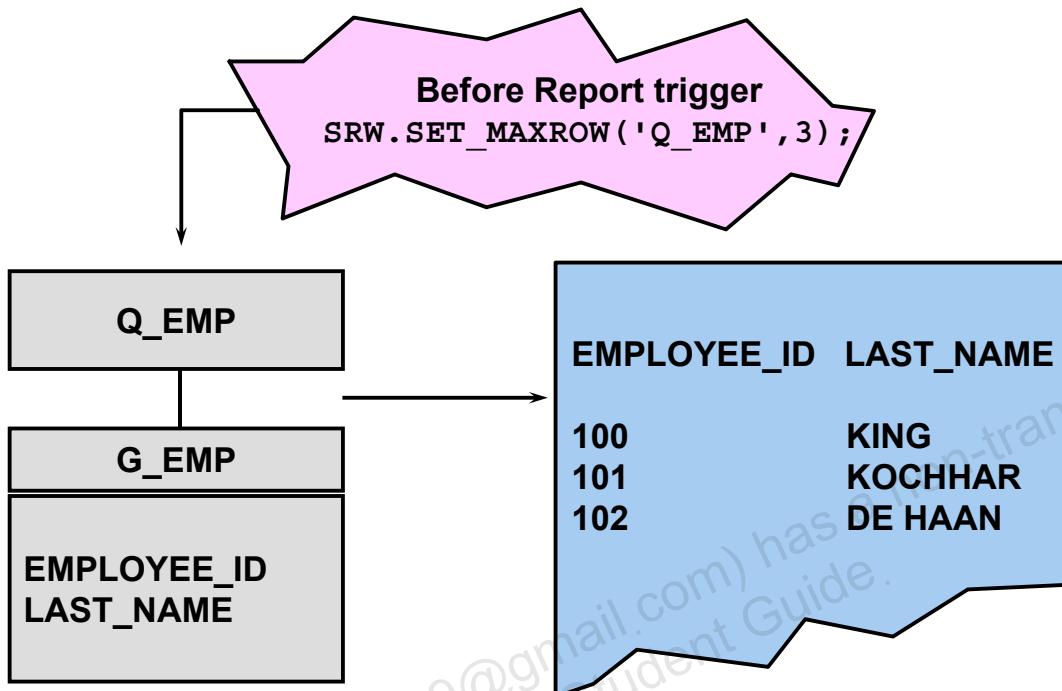
Exception Name	Causes
SRW.RUN_REPORT_FAILURE	Detail report does not exist.
	Detail report failed during execution.
SRW.RUN_REPORT_BATCHNO	BATCH parameter is used inconsistently: master report running with BATCH=YES, called a detail report with BATCH=NO
Displaying Report Failure Message	

If the detail report fails during execution, you can display information about the cause by using the SRW.GETERR\_RUN function.

Assign the function to a local PL/SQL character variable, which you display using the SRW.MESSAGE procedure.

```
...
msg char(150);
...
EXCEPTION
WHEN SRW.RUN_REPORT_FAILURE THEN
    msg := SRW.GETERR_RUN;
    SRW.MESSAGE (100, msg);
...
```

# Restricting Data



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Restricting Data

Use SRW.SET\_MAXROW to restrict the maximum number of records you want to retrieve for a specified query. Reports Builder uses only the retrieved rows of the query in subsequent processing, such as computations and summaries.

## Suppressing a Query

At run time, you can choose not to display any data from a query by setting the maximum number of records to zero. This prevents Reports Builder from retrieving any records from the database.

## Restricting Data (continued)

### Example

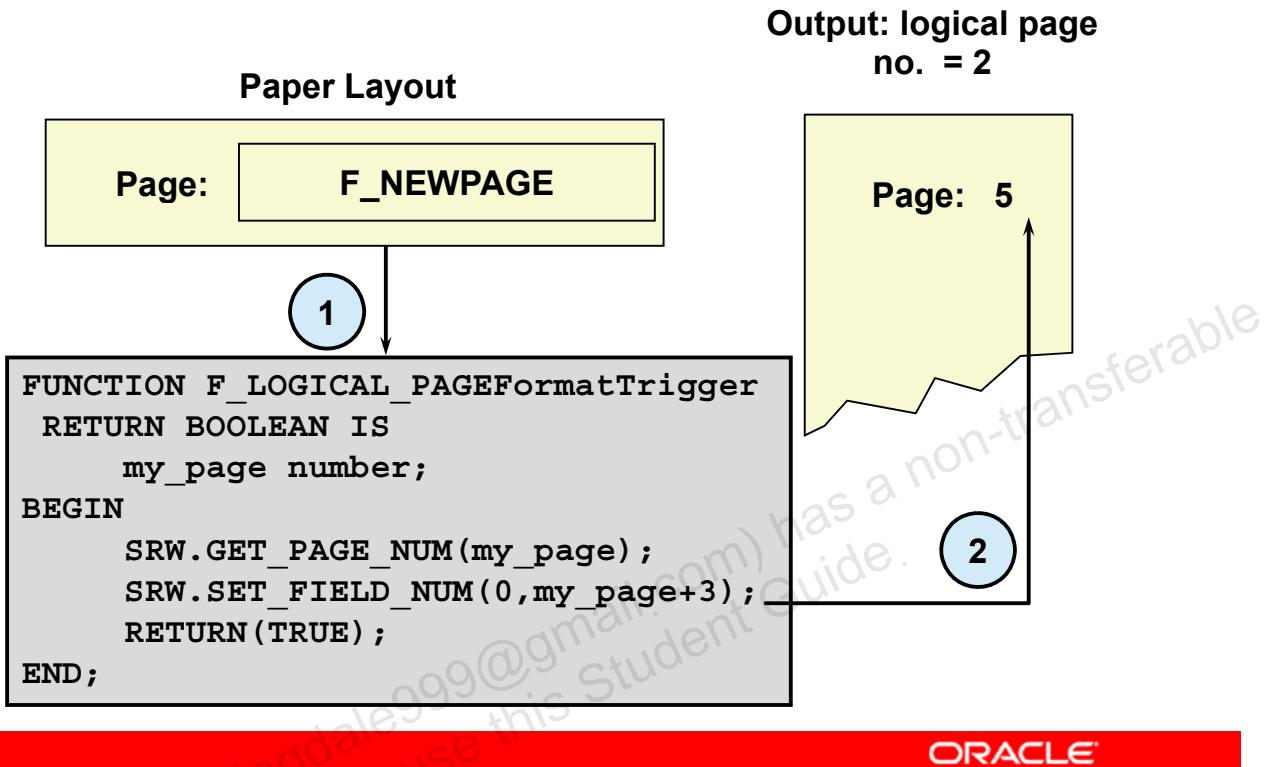
Below is a function to display data for the Q\_emp query only if a value of one or more is entered in a run-time parameter called HOW\_MANY:

```
FUNCTION FETCHIT RETURN BOOLEAN IS
BEGIN
    IF :HOW_MANY >= 1 THEN
        SRW.SET_MAXROW ('Q_emp', :how_many);
    ELSE
        SRW.SET_MAXROW ('Q_emp', 0);
    END IF;
    RETURN (TRUE);
EXCEPTION
    WHEN SRW.MAXROW_UNSET THEN
        SRW.MESSAGE (100,
                     'Data was fetched before SRW.MAXROW was called');
        RAISE SRW.PROGRAM_ABORT;
    WHEN SRW.MAXROW_INERR THEN
        SRW.MESSAGE (200, 'Contact someone clever!');
        RAISE SRW.PROGRAM_ABORT;
END;
```

You must set SRW.SET\_MAXROW in the Before Report trigger; that is, after the query has been parsed but before it has been executed. If you call it after the Before Report trigger, Reports Builder raises the SRW.MAXROW\_UNSET exception.

If you handle SRW.MAXROW\_INERR, always raise SRW.PROGRAM\_ABORT, because your report has an internal problem and you cannot guarantee the outcome.

# Initializing Fields



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Initializing Fields

You can dynamically initialize the value in a layout field by using the relevant SET\_FIELD packaged procedure; for example:

- SRW.SET\_FIELD\_CHAR (0, 'a text string')
- SRW.SET\_FIELD\_NUM (0, 1234)
- SRW.SET\_FIELD\_DATE (0, '01-JAN-2002')

The first argument is always 0 (zero); you can set the value of the current object.

These procedures are relevant only in the format trigger of a field of the correct type. If you use a procedure that conflicts with the field type, it has no effect in the report; Reports Builder ignores the code and does not raise an error.

1	Create format trigger on F_NEWPAGE field
2	Set field value to my_page+3

## Initializing Fields (continued)

You can use SRW.SET\_FIELD to retrieve the current logical page number and recalculate the value before displaying it.

**Example:** Create a function to display the logical page number plus three in a field called F\_NEWPAGE.

Because the logical page number variable is available only in the report layout, not in the Data Model, you cannot calculate the value in a data column. Therefore, you use a paper layout field to get the current page number and output a new number by initializing the same field.

Make use of a packaged procedure called SRW.GET\_PAGE\_NUM to retrieve the current logical page number into a local PL/SQL variable, as in the format trigger code on the previous page.

# Creating a Table of Contents

- Use SRW.SET\_FORMAT\_ORDER in an After Parameter Form trigger to format the Main section of the report first
- Use SRW.GET\_PAGE\_NUM in a format trigger to fetch the page number for each topic in the TOC

Topic	Pages
Argentina	1-7
Australia	7-26
Brazil	26-45
Denmark	45-54
France	54-147
Germany	147-345
India	345-362
Ireland	362-411
Japan	411-426
Malaysia	426-440
New Zealand	440-446
Poland	446-462



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Creating a Table of Contents

Earlier in the course, you learned that you could use the Format Order of Sections property to modify the order in which Reports Builder formats the sections of the report to enable you to create a table of contents (TOC).

You can also use the SRW.SET\_FORMAT\_ORDER built-in procedure to format the Main section of the report first and use report triggers to build a table containing the TOC entries. When the first element for the TOC is formatted, a trigger fires and creates a row in the TOC table containing the TOC entry and the page number. After the Main section has completed formatting, the format order setting can define that the Header section is formatted next. The Header section can contain a report block based on the TOC table. After formatting, you can output your report with a TOC (the Header section), followed by the report body (the Main section), followed by the Trailer section.

## Creating a Table of Contents (continued)

**Example:** You want to generate a report of all e-mail addresses of your customers. Your customer database contains thousands of records. You want to group the customers by country and then generate a table of contents so you can easily find the e-mail address for the desired customer.

To create a simple table of contents:

1. Create a table in the database to hold the TOC information. You need two columns, one for the topic and the other for the page number.
2. Use the Report Wizard to create a Group Above report. This report will display the country name, then the customers and customer e-mail addresses under each country name. The table of contents you create will then be based on the country name in this report.
3. Create an After Parameter Form trigger to modify the default order of the report execution, so that the Main section is formatted first.

```
FUNCTION AfterPForm RETURN BOOLEAN IS
BEGIN
    SRW.SET_FORMAT_ORDER(SRW.MAIN_SECTION,
                         SRW.HEADER_SECTION, SRW.TRAILER_SECTION);
    RETURN (TRUE);
END;
```

4. Create a format trigger on the field that displays the country name. The format trigger will fetch the page number for each country name, so that the TOC will enable the user to navigate to various parts of the report based on the name of the country.

```
FUNCTION F_COUNTRY_NAMEFormatTrigger RETURN BOOLEAN IS
    pageNum NUMBER;
BEGIN
    -- get current page number
    SRW.GET_PAGE_NUM(pageNum);
    -- insert row into database table
    INSERT INTO TOC_TABLE (TOPIC, PAGE_NUMBER)
        VALUES (:country_name, pageNum);
    RETURN (TRUE);
END;
```

5. Use the Data Model to create a second query with a formula column that calculates the page range for the data under each country name. This query will fetch the information from the database table you created in step 1.
6. Create a tabular report block in the Header section of your report to display the headings in the table of contents (i.e., the country name), and the page range where the information can be found in the report.

## Performing DDL Statements

- Example:

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

- Exception: SRW.DO\_SQL\_FAILURE



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Performing DDL Statements

The SRW.DO\_SQL packaged procedure executes any specified SQL statement from within Reports Builder.

Use SRW.DO\_SQL to perform DDL statements dynamically during run time. For example, create a temporary table and insert rows into it during execution.

You can also perform DML statements using this procedure, but DML statements run faster if you code them directly within a PL/SQL block. Code DML statements in the SRW.DO\_SQL procedure only if necessary. For example:

- To insert records into a permanent table in the database, code DML in a PL/SQL block.
- To insert records into a temporary table that does not exist until run time, code DML in SRW.DO\_SQL.

## **Performing DDL Statements (continued)**

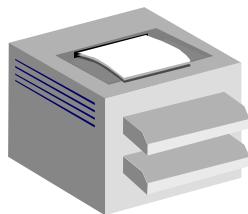
### **Querying from Temporary Tables**

If you want to create a temporary table and reference it in a query in the same report, the table must exist before the Before Report trigger fires. That is when Reports Builder parses its queries. Therefore, you must create this table in the Before Parameter Form or After Parameter Form report trigger.

# Setting Format Attributes



- Borderwidth
- Fill pattern
- Format mask



- Printer tray control



- Define a bookmark

ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Setting Format Attributes

The SRW.SET\_<attributes> packaged procedures apply specified format attributes to the current layout object: frame, repeating frame, field, or boilerplate object.

Some of the reasons that you would want to use format attributes are:

- To change the visual appearance of layout objects conditionally, depending on values in a repeating frame, or on run-time parameter values
- To switch to different printer trays as your report formats
- To place a logo or standard links in an HTML document
- To define a bookmark in an HTML or PDF document

You can code the packaged procedure in a function as a local or report-level program unit, or in a PL/SQL library. This is useful if you want a common format attribute in more than one object or more than one report. You can then call the function whenever required from the format trigger of a specific object.

As an alternative to a PL/SQL library, in the Object Navigator, you can drag and drop report-level program units between different reports. However, note that this creates a copy of the program unit in each report, rather than a reference to one piece of code.

## Setting Format Attributes

- Visual attributes:

```
SRW.SET_FILL_PATTERN('solid') ;  
SRW.SET_BORDER_WIDTH(250) ;  
SRW.SET_FORMAT_MASK('Day, Month yyyy') ;
```

- Printer tray control:

```
SRW.SET_PRINTER_TRAY('letterhead') ;
```

- Defining a bookmark:

```
SRW.SET_BOOKMARK('Car Policy') ;
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Setting Format Attributes (continued)

#### Example

Conditionally set the following attributes if salary is greater than \$12000:

Attribute	Value
Background fill color	Blue
Fill pattern	Sandpaper
Font style	Italic
Font face	Helvetica
Font weight	Extra bold
Text color	White

## Setting Format Attributes (continued)

### Example (continued)

```
FUNCTION SAL_MASK RETURN BOOLEAN IS
BEGIN
  IF :SALARY > 12000 THEN
    SRW.SET_BACKGROUND_FILL_COLOR('blue');
    SRW.SET_FILL_PATTERN('sandpaper');
    SRW.SET_FONT_STYLE(SRW.ITALIC_STYLE);
    SRW.SET_FONT_FACE('helvetica');
    SRW.SET_FONT_WEIGHT(SRW.EXTRABOLD_WEIGHT);
    SRW.SET_TEXT_COLOR('white');
  END IF;
  RETURN (TRUE);
END;
```

Remember that you can use the Conditional Formatting dialog box to specify output formatting attributes (font and/or color) for a selected layout object based on conditions that exist. However, if you need to apply other attributes such as fill color, for example, then you need to use the SRW.SET\_<attribute> procedures.

## Using Format Attributes in a Web Layout

```
...
<rw:field id="F_SALARY" src="SALARY" . . .
    formatTrigger="mystyles">
    F_SALARY
</rw:field>
...
...
```

```
FUNCTION mystyles RETURN BOOLEAN IS
BEGIN
    SRW.SET_JUSTIFICATION(SRW.RIGHT_HJUST);
    SRW.SET_TEXT_COLOR('Red');
    SRW.SET_FONT_WEIGHT(SRW.BOLD_WEIGHT);
    SRW.SET_FONT_STYLE(SRW.ITALIC_STYLE);
    return (TRUE);
END;
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Using Format Attributes in a Web Layout

Recall that the formatTrigger attribute of the rw:field tag is used to modify the display characteristics of the field. You can only use specific SRW packaged procedures to set the field's attributes. These are:

- SET\_FORMAT\_MASK
- SET\_HYPERLINK
- SET\_LINKAGE
- SET\_TEXT\_COLOR
- SET\_JUSTIFICATION
- SET\_FONT\_FACE
- SET\_FONT\_STYLE
- SET\_FONT\_SIZE
- SET\_FONT\_WEIGHT
- SET\_DISPLAY\_NAME
- SET\_FIELD\_CHAR
- SET\_FIELD\_DATE
- SET\_FIELD\_NUM

## Using Format Attributes in a Web Layout (continued)

### Example

In the example on the previous page, the formatTrigger attribute contains the name of the function mystyles. When executed, mystyles will set the following attributes on the salary field:

Attribute	Value
Justification	Right
Text color	Red
Font weight	Bold
Font style	Italic

## Using Format Attributes in Character Mode Reports

Some attributes are applicable to character environments only; some are for bitmapped environments only.

If you include a bitmapped attribute and then run the report in character mode, or the reverse, Reports Builder ignores the code and does not cause an error. This enables you to develop a report for use in a different environment.

# Summary

In this lesson, you should have learned how to:

- Describe the SRW package contents:
  - Procedures
  - Functions
  - Exceptions
- Use SRW.PROGRAM\_ABORT to stop report processing
- Use the SRW package to:
  - Execute a detail report
  - Create a table of contents
  - Dynamically execute DDL statements
  - Provide conditional layout formatting and display



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Summary

- The SRW package contains procedures, functions, and exceptions.  
Remember that you can raise the exception SRW.PROGRAM\_ABORT in your PL/SQL code to stop report processing at any time.
- The SRW package provides a number of features, including:
  - Executing a second report from within a report process, using SRW.RUN\_REPORT
  - Modifying the format order for a report and then getting the resulting page numbers for the Main section so that you can create a table of contents in the Header section
  - Support for DDL statements, using SRW.DO\_SQL, to enable you to create or drop temporary tables from PL/SQL code in a report
  - Conditional formatting and display using SRW.SET\_<attributes>

## Practice 14 Overview

- Building a report containing conditional highlighting
- Modifying a Web report with conditional highlighting
- Executing a detail report from a master report
- Writing to a temporary table from a report



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Practice Session Overview: Lesson 14

This practice session contains:

- Building a report containing conditional highlighting
- Modifying a Web report with conditional highlighting
- Executing a detail report from a master report
- Writing to a temporary table from a report

In this practice session, you highlight values based on conditional PL/SQL code.

You build a master report that enables you to call a detail report and generate a separate PDF file for each detail report.

If you have time, you add report-level triggers to an existing report to write messages to a temporary table at various stages of report execution.

- Write PL/SQL functions as local PL/SQL (report-level), and reference the code in a condition within the relevant object trigger in the report.

## Practice Session: Lesson 14

(Hint - copy the s9q3.jsp from sol folder and rename it as p9q3.jsp)

1. Open report p9q3.jsp.
  - a. Display the column CUST\_TOTAL in the customer group as red, bold italics if its value is 10,000 or more.

**Hint:** Check the Help system for the SRW.SET <attributes> or use conditional formatting.

Mask Attribute	Suggested Value
SRW.SET_TEXT_COLOR	Red
SRW.SET_FONT_WEIGHT	SRW.BOLD_WEIGHT
SRW.SET_FONT_STYLE	SRW.ITALIC_STYLE

You can copy the code from the file p20q1a.txt.

- b. Alter the pattern and color of those records where the customer total is 5,000 or less.

**Hint:** Check the Help system for the SRW.SET <attributes> or use conditional formatting.

Attribute	Suggested Values
Fill pattern	Crisscross, sandpaper
Foreground fill color	Yellow
Background fill color	Green

You can import the file p20q1b.txt.

- c. Write the necessary format triggers.
  - d. Save the report as p20q1.jsp.
  - e. Run the paper layout.
2. Continue with the same report.
    - a. Run the Web layout for the report. Is there formatting on the CUST\_TOTAL column?
    - b. Modify the Web source so that the column CUST\_TOTAL in the customer group as displays as red, bold italics if its value is 10,000 or more.
    - c. Save the report as p20q2.jsp.
    - d. Run the Web layout to test.

## Practice Session: Lesson 14 (continued)

3. Execute a detail report from a master report.
  - a. Create a simple tabular report called MGR\_MASTER. The report should query last names and employee IDs for those employees who are managers. Display all fields. You can use the query:

```
SELECT distinct manager.last_name Manager,
manager.employee_id
FROM   employees worker, employees manager
WHERE  worker.manager_id = manager.employee_id
```
  - b. Save the report as mgr\_master.rdf.
  - c. Create another tabular report, the detail report, and name it EMP\_RPT. The report should return department number, employee name, and salary for a specific manager. The manager\_id will be passed to EMP\_RPT by MGR\_MASTER.
  - d. Save the report as emp\_rpt.rdf and close it.
  - e. Write the code to execute the detail report. For each manager returned, MGR\_MASTER should invoke the detail report, EMP\_RPT, pass it the manager's employee\_id, and produce a report in PDF format containing information about the employees working for that manager. Use the manager's employee\_id to create an output filename for each manager:  
MGR\_<employee\_id>.PDF.  
**Hint:** You can write the code in the group filter.
  - f. Compile the PL/SQL code and close the window.
  - g. Run the paper layout for MGR\_MASTER to test. Access the file system and locate the output for the detail reports. View two or three of the output files.
  - h. Save mgr\_master.rdf and close it.

### If You Have Time

4. Open report p10q1.rdf

In this report, you use report triggers to create a temporary table at the start of report execution, and you insert rows into the table at various stages of the execution.

- a. Write a trigger that creates a temporary table to hold the date and userid each time the report is run. The table should contain three columns: one date column, one character column of width 10, and one character column of width 80.

```
CREATE TABLE RUNREPORT
(DATE_RUN DATE, USER_RUN VARCHAR2(10), COMMENTS
VARCHAR2(80))
```

## Practice Session: Lesson 14 (continued)

- b. The same trigger should also insert the current date, userid, and “Starting Report” into this table.

```
INSERT INTO RUNREPORT  
  (DATE_RUN, USER_RUN, COMMENTS)  
VALUES (SYSDATE, USER, 'Starting Report')
```

**Hint:** Think about how you code the single quotes.

- c. Compile the PL/SQL code and close the window.
- d. Save the report as p20q4.jsp.
- e. Run the paper layout.
- f. Using SQL\*Plus, verify that your table exists and was populated successfully.
- g. Log on to SQL\*Plus (in Microsoft Windows, select the SQL\*Plus icon in the Oracle group) using the same userid as you used for Reports.
- h. Display all records in the RUNREPORT table.

```
SQL> col comments format a40  
SQL> SELECT * FROM RUNREPORT;
```

- i. Try running the report more than once. When you run the report a second time, the table already exists and Reports Builder raises an exception. Make the necessary change to your code so that the report runs even when the table exists.
- j. Write a trigger that inserts a record with the comment “Printing another page” whenever it begins a new page.
- k. Write a trigger that inserts a record with the comment “Report completed” when the report finishes.
- l. Save the report as s20q41.jsp. Generate the output to a PDF file. Verify the results in SQL\*Plus.

15

## Building Reports: Efficiency Guidelines

ORACLE®

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

# Objectives

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

- Describe performance areas
- Use performance measuring tools
- Build reports to run in other languages



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Overview

A report does not run in a vacuum. It is dependent on the underlying hardware, software, and network connection. Because of this complex interrelationship, performance is always an issue. The second part of this lesson focuses on performance in the run-time environment and gives you a number of performance tips. You should also realize that Report applications are portable across multiple platforms and multiple languages. You can reduce cross-platform development effort by planning for platform-specific features or restrictions. In this lesson, you also learn how to build reports to run on various platforms, and in different environments and languages.

# Tuning Reports

- No absolute rules
- Investigate specific areas:
  - Data Model
  - Paper Layout
  - Web Layout and JSP report definitions
  - Run-time arguments



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Tuning Oracle Reports

Tuning a report is not a matter of applying a few small tweaks that will result in dramatic improvements in performance. It is more a case of investigating specific areas. The areas to investigate are the Data Model, the Paper Layout, the Web Layout and JSP reports definitions, and run-time arguments. Investigating some of these areas can result in significant performance improvements, minor performance improvements, or may have no effect on the actual performance at all, but rather impact the perceived performance. Be aware that there are no absolute standards.

## The Reports Measurement Tools

The first step to take when attempting to tune a report is to determine where the report is spending the majority of its time in execution. In other words, does the report spend a large portion of its time in the database retrieving the actual data, in the formatting process once the data has been retrieved, or waiting for runtime resources and distribution?

## Tuning Oracle Reports (continued)

### The Reports Trace Option

The Reports Trace option produces a file that describes the series of steps completed during the execution of the report. You have the option to log all events or only a subset of events such as SQL execution or breakpoints. The trace file can provide very useful information for performance evaluation, and also tells you what executed when.

Tracing can be set for various methods of report execution, for both RDF and JSP report definitions containing SQL and non SQL data sources, such as the XML and Text pluggable data sources.

To set the Trace option, use the TRACEFILE=<filename> command line argument when running the report, or from the main menu select Program > Tracing.

Typical profile output is shown below:

```
Reports Profiler Statistics
TOTAL ELAPSED Time: 29.00 seconds
Reports Time: 24.00 seconds (82.75% of TOTAL)
ORACLE Time: 5.00 seconds (17.24% of TOTAL)
UPI: 1.00 seconds
SQL: 4.00 seconds
```

In this profile output, you see the execution time (total elapsed time) for the report, the amount of time spent formatting the retrieved data (Reports time), and the amount of time spent waiting for the data to be retrieved (Oracle time), comprised of time spent establishing the database connection, parsing and executing the SQL statement (UPI time), and the time spent fetching the data and executing SRW.DO\_SQL() statements, EXEC\_SQL statements, PL/SQL cursors, and so forth (SQL time).

# Performance Measurement

Measurement tool: Trace option

Server and network measurements:

- SQL Trace
- Traffic and bandwidth



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Server and Network Specific Measurements

Oracle Reports uses Structured Query Language (SQL) to retrieve data from a relational database. It is essential for anyone tuning reports to have a good working knowledge of SQL and to understand how the database is going to execute these statements. For those users less proficient in SQL, Oracle Reports provides wizards to help build queries. However, wizards cannot prevent inefficient SQL from being created (e.g., SQL that does not use indexes).

Database servers and network systems often provide measurement tools to obtain performance information. An invaluable aid for tuning SQL is the SQL Trace facility provided by the database. SQL trace allows you to determine the SQL statement sent to the database as well as the time taken to parse, execute, and fetch the data. Once the trace file has been generated, use the TKPROF utility to generate the execution plan as well as vital time statistics. For more information, refer to the *Oracle Database SQL Reference Manual*.

You may also want to consult with the database and network administrators. They may be able to suggest ways to improve performance for their respective areas. An obvious approach is to consider upgrading to a later, more efficient product release, or to increase the capacities and speeds of the hardware systems and network connections.

# Investigating the Data Model

- Schema design
- Indexes
- Efficient SQL
- Efficient PL/SQL
- Calculations
- Redundant queries
- Break groups
- Group filters
- Linking queries



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Investigating the Data Model

If the report spends a large amount of time in the database, then it is often wise to review the data structure and how it is being used. The following areas should be examined:

- **Schema design:** The specific nature of your application will determine the most efficient Data Model, but bad schema design can have a dramatic effect. Generally speaking, query-driven applications can benefit from denormalized tables, whereas normalized tables are usually best for applications that do many updates and inserts. Oracle Reports is a typical nonprocedural tool geared for set-based data retrieval. Hence, an overly normalized Data Model can result in many avoidable joins or queries.
- **Use of indexes:** A lack of appropriate indexes can result in many full table scans and Cartesian joins. Indexes should be placed on columns used in the query WHERE clause, and on the tables in the detail queries. Indexes on columns in the master queries of a report have limited impact as these queries only access the database once.

## Investigating the Data Model (continued)

- **Efficient use of SQL:** Inefficient SQL can severely impact performance. As mentioned earlier, an invaluable aid to tuning SQL in a report is the SQL Trace facility provided by the database. For reports that have large queries, it is advisable to either activate the cost based optimizer (by running ANALYZE on the tables or by setting the appropriate parameter in the init.ora file), or optimize all SQL statements following the rules laid out for the rule based optimizer.
- **Efficient use of PL/SQL:** PL/SQL that performs database operations will give better performance if it is used in stored database procedures. Stored procedures perform database operations faster than local PL/SQL program units. The opposite is true if PL/SQL does not involve any database operations. In this case, PL/SQL should be coded locally using the Program Units node in the Report Object Navigator. You may also want to consider using PL/SQL libraries, but the performance overhead is outweighed only when the benefits of code sharing can be used.

# Investigating the Data Model

- Schema design
- Indexes
- Efficient SQL
- Efficient PL/SQL
- Calculations
- Redundant queries
- Break groups
- Group filters
- Linking queries



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Investigating the Data Model (continued)

- **Performing calculations:** For calculations within a report (either through summary or formula columns), the rule of thumb is that the more calculations that can be performed within SQL, the better. If calculations are included in SQL, they are performed before the data is retrieved rather than performed on the retrieved data by Oracle Reports. PL/SQL code that performs a significant amount of database operations will perform better if it is implemented as a stored database procedure. SRW.DO\_SQL() should be used as sparingly as possible because each call to SRW.DO\_SQL() necessitates parsing and binding of the command and the opening a new cursor (just as with a normal query). Unlike the query, however, this operation will occur each time the object owning the SRW.DO\_SQL() fires. For example, if a block of PL/SQL in a formula column calls SRW.DO\_SQL() and the data model group where the formula resides returns 100 records, then the parse/bind/create cursor operation will occur 100 times. It is therefore advisable to only use SRW.DO\_SQL() for operations that cannot be performed within normal SQL (for example to create temporary tables or any other form of DDL).

## Investigating the Data Model (continued)

- **Redundant queries:** Ideally, a report should have no queries that return data that is not required in the report. In general, the fewer queries you have, the faster your report will run. Single query Data Models tend to execute more quickly than multiquery Data Models. However, situations can arise where a report not only needs to produce a different format for different users, but also needs to utilize different query statements depending on user input. Clearly, this could be achieved by producing two different reports.
- **Break groups:** If the report requires the use of break groups, care should be taken to ensure that the break order property is set for as few columns in the break group as possible as Oracle Reports appends an extra column to the ORDER BY clause of the query for every break column. The creation of a break group may also make the ORDER BY clause redundant and in this case should be removed. To give better performance, break order columns should be as small as possible and should be database columns wherever this is feasible.
- **Group filters:** It is usually more efficient to incorporate a group filter into the query WHERE clause. This restricts the number of rows returned from the database and avoids local filtering by Oracle Reports itself.
- **Linking queries:** There are a number of ways to create Data Models that include more than one table. You can create a single query with a join condition or you can create two separate queries and create a link between the two. In general, it is preferable to minimize the actual number of queries as each time a query is issued, Oracle Reports needs to parse, bind, and execute a cursor. A single query report is therefore able to return all the required data in a single cursor rather than many. Also, with master detail queries, the detail query will be reparsed, rebound, and reexecuted for each master record retrieved. In this case, it is often more efficient to use a single query report with break groups.

## Investigating the Paper Layout

- Frames
- Object size
- Format triggers
- Fetching ahead
- Bursting and distribution



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Investigating the Paper Layout

Once the data has been retrieved, Oracle Reports needs to format the output following the Paper Layout the user has created. The time it takes to generate the layout depends on a number of factors, such as:

- **Frames:** When generating a default layout, Oracle Reports puts a frame around virtually every object to protect the object from being overwritten by another object. In some situations, such as boilerplate text column headings, there is clearly no risk of the objects being overwritten, hence the immediately surrounding frame can be removed.
- **Object size:** An object that is defined as variable, expanding, or contracting in either or both the horizontal and vertical directions requires extra processing. In this case, Oracle Reports must determine the size for the object's instance before actually formatting the object and any objects around it. If the sizing can be set to fixed, this can be avoided as the positional relationship between objects is already known.

## Investigating the Paper Layout (continued)

- **Format triggers:** Care should be taken when using format triggers as the trigger fires every time the object is formatted at runtime. Format triggers should also be placed at the highest level possible in the object/frame hierarchy so that the triggers fires at the lowest possible frequency.
- **Fetching ahead:** Oracle Reports provides the ability to display data such as the total number of pages or grand totals in the report margin or header pages. This requires that the entire report is “fetched ahead” before the first page can be output. The usual execution model is to format pages on an as-needed basis. Using the “read ahead” functionality does not affect the overall time it takes to generate the report, but rather affects the amount of temporary storage required and the time it takes before the first page can be viewed. To enhance perceived performance, reading ahead should be avoided when a report is going to be run to the screen in a production environment.
- **Bursting and distribution:** With the introduction of report bursting, a report layout can be made up of three distinct sections: header, body, and trailer. A report can comprise all three sections or it can be viewed as three separate reports within one report. Oracle Reports allows you to control bursting at the group record level, offering a further level of granularity in report bursting. The use of report bursting in conjunction with distribution offers considerable performance gains. Each section of a report can have multiple different formats and can be sent to multiple destinations in one single run rather than running the same report multiple times.

## Web Layout and JSP Report Definition

You learned that Oracle Reports takes full advantage of JSP technology to deliver high quality Web publishing. Recall that you can use your favorite Web authoring tool to design the static portion of your Web page and then use Oracle Reports to add the dynamic portion (data) into the appropriate sections of the page. A poorly designed Web page has an impact on the perceived performance.

Because it is possible to include any Java code in the JSP, it is easy to get carried away and mix business and data access Java code with the presentation logic. This should be avoided as it increases the footprint of the JSP and limits the efficient use and management of system resources.

Customized formatting of a Web page is always an expensive operation. Any type of formatting that you cannot natively achieve through Oracle Reports (e.g., changing the foreground color of a report block) should be done using Java in the most direct way. The use of PL/SQL wrappers for formatting purposes is discouraged.

A .jsp report definition can contain both a paper layout and a Web layout. Since the Web layout section of a JSP report could contain an <rw:include> tag referencing a paper layout object, Oracle Reports always formats the paper layout when executing the report. If your JSP report does not reference any paper layout objects, it is recommended that you use the SUPPRESSLAYOUT command option to prevent executing the paper layout.

## Running the Report

- RUNDEBUG=NO
- Array processing
- LONGCHUNK
- PARAMFORM
- Batch processing



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Running the Report

Having designed a report to run efficiently, it is possible to further affect the overall performance of a report by setting specific run-time arguments.

By default, Oracle Reports automatically runs error checks on the layout and bind variables within a report. This is useful at the design stage but is generally superfluous in a production environment. Setting the runtime parameter RUNDEBUG= No turns off this extra checking at run-time.

Oracle Reports is able to take advantage of array processing capabilities. This allows records to be fetched in batches instead of one at a time and results in fewer calls to the database. The downside of array processing is that more space is required on the execution platform for storing the arrays of records returned. If the load on the network becomes a major bottleneck in the production environment, the ARRAYSIZE argument (defined in kilobytes) can be set to as large a value as possible for the execution environment to reduce the number of network trips made to fetch the data for the report. The default value for ARRAYSIZE is 10 kilobytes.

## Setting NLS Language Environment Variables

- `NLS_LANG`
- `DEVELOPER_NLS_LANG`, `USER_NLS_LANG`

```
NLS_LANG=French_France.WE8DEC
```

Unicode:

```
NLS_LANG=<lang>_<territory>.UTF8
```



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Building Reports to Run in Different Languages

This section gives a brief overview of the features available on the server and in Oracle Reports Developer that provide support for building multilingual applications.

- Switching languages using Oracle's National Language Support (NLS)
- Translating an application using Oracle's TranslationHub

### Setting NLS Language Environment Variables

Three language environment variables are available:

- `NLS_LANG`
- `DEVELOPER_NLS_LANG`
- `USER_NLS_LANG`

`DEVELOPER_NLS_LANG` and `USER_NLS_LANG` take the same parameter as `NLS_LANG`. Use them as an alternative to `NLS_LANG` when you need to use two sets of resource and message files at the same time; for example, if you prefer to develop using English but need to build an application in another language.

## **Building Reports to Run in Different Languages (continued)**

### **Syntax**

`NLS_LANG = <language>_<territory>.character_set`

- Language: Language conventions for displaying messages and day and month names
- Territory: Conventions for default date format, and for displaying currency, decimal, and thousands symbols
- Character\_set: Character set in which data is displayed

### **Example**

`NLS_LANG=French_France.WE8DEC`

### **Using Unicode in Oracle Reports Developer**

Unicode is a global character set that allows multilingual text to be displayed in a single application. Unicode is a fixed-length two-byte character set that represents up to 65,536 characters.

Using Unicode in Oracle Reports enables you to display multiple languages in one application without switching character sets.

To enable Unicode support, set the `NLS_LANG` environment variable as follows:

`NLS_LANG = <language>_<territory>.UTF8`

The data must also be stored in Unicode.

## Translating an Oracle Reports Application

Use TranslationHub to translate messages and text defined as part of the application.

Consider:

- Format masks
- Hard-coded strings
- Multiple character sets



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Translating an Oracle Reports Application

In any Oracle Reports application, you see many types of messages, including:

- Error messages from the database
- Runtime error messages produced by OracleAS Reports Services
- Messages and boilerplate text defined as part of the application

If the NLS environment variable is set correctly and the appropriate message files are available, then translation of messages for the first two bullets is done for you. To translate messages and boilerplate text defined as part of the application, you can use the Oracle translation tool, TranslationHub.

For more information about using TranslationHub, see the Oracle Technology Network (<http://otn.oracle.com>).

## **Translating an Oracle Reports Application (continued)**

### **Considerations When Designing Multilingual Applications**

- Format masks: Make sure to provide extra space for translation of date and numeric format masks.
- Text strings: Avoid hard-coding a string containing language-specific words such as a month name.
- Character sets: If using an application that will run with multiple character sets, determine the one that is most frequently used and generate the application files with the relevant NLS language settings.

# Summary

In this lesson, you should have learned how to:

- Use performance measuring tools
- Describe the facilities for running reports in other languages



Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

## Summary

- Analyze the performance of your reports using the Trace option in Oracle Reports, as well as the SQL trace facility in the database. Tune reports by optimizing SQL, minimizing calls to the database, and minimizing the amount of unnecessary format processing required for the layout of returned data.
- Using Oracle's translation tool, TranslationHub, the NLS parameters, and various features in Oracle Reports, you can build multilingual report applications.

# Practice 15

## Overview

- Identifying areas to investigate for tuning reports
- Using the trace facility
- Interpreting time statistics
- Improving performance



ORACLE

Copyright © 2013, Oracle and/or its affiliates. All rights reserved.

### Practice Session Overview: Lesson 15

This practice session contains:

- Generating and interpreting report trace information
- Modifying a report to make it more efficient

Report performance is always an issue. Reports Builder offers a few tools that allow you to measure performance. In this practice you will use them and interpret the results.

## Practice Session: Lesson 15

1. Name four main areas to investigate when tuning reports.
2. What is the name of the font mapping file?
3. Generate and interpret Reports Runtime Trace information.
  - a. Open report p19q4.rdf. From the menu select Program > Tracing.
  - b. In the Runtime Trace Settings dialog box specify p19q4.txt as the trace file and select the All check box under Trace Settings.
  - c. Select OK to close the dialog box.
  - d. Run the paper layout. Paginate through the report in the Paper Design view.
  - e. Open p19q4.txt and look at the results. Where does this report spend time?
  - f. Repeat the same steps for report p22q2b.rdf. Specify p22q2.txt as the trace file. Compare the result. How do you explain this?
4. Make a suggestion to improve performance.  
( Hint – Copy s12q1.jsp from sol folder into labs, rename it as p12q1.jsp)
  - a. Open the report p12q1.rdf and generate Reports Trace information. Look at the time statistics.
  - b. Examine the Data Model. What could you possibly do to make the report run faster?
  - c. Implement your change and regenerate the time statistics. Did things improve?

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

Shital jadhav (shitaljagdale99@gmail.com) has a non-transferable  
license to use this Student Guide.

# CASE LITE

---

**Reports:** ..... **Duration 0.5 Day**

## **Business Scenario:**

Vision Corporation Inc is a US based company, is in the field of manufacturing computer systems. It has sales offices across the globe and its sells to its customers through following options.

- Direct sales - For key customers with cumulative annual sales over 1 million USD per annum. They are treated as privilege customers, their early discount structure is decided by the top management and also the key customers have the first access to sample products for feedback before market launch.
- Distribution channels – For small account key customers, there are country specific re-seller. In this case the discount structure is fixed based on volume. The discount structure will be two folds
  - Reseller discount.
  - End customer discount – based on volume.

## **Booking & Sales process:**

Vision Corporation receives Purchase order from end customer in case of direct sales and from its own distributor.

Vision Corporation is expected to maintain a sales system where customer purchase orders and sales invoices for customer are generated at Vision corporation central office. Once goods are ready for delivery invoices are generated.

Vision Corporation has its system assembly plants in Europe, US & Asia. The component sourcing operation for assembly of its product happens across the globe through centralized sourcing team based at US. The vision corporation also

maintains ware houses in US, Europe, Asia & Africa to store finished products so that it can cater to global market needs at shortest possible time.

The operation manager of Vision Corporation wants the following reports at the start of the week (assume today to be start of the week) towards operation planning.

1. Pending Orders List.
2. Order Details report for a customer.

### **Problem Statement**

#### **1. Pending Orders List.**

The Head of the sales department needs to know as of date, the pending order which is to be delivered, so that he can inform production team about the requirement. The sales head had asked you to prepare a report in the following format and it should also display the grand totals of the Order amounts.

Order ID	Type	Order Date	Status	Payment Terms	Order Amount	Disc Percent	Tax	Requested Delivery Date

#### **Bonus Points:**

- a. High value orders (say Order Amount > 5000) should display differently in a colour of your choice.

## **Problem Statement**

### **2. Order Details Report**

The sales manager wants a detailed report of sales orders by customer to build necessary business intelligence & promotion. He has requested you to create a report as per the format shown below, the report should take customer id as input. Use a summary column to calculate the grand total and use formula column to calculate amount

Customer No		Name			Phone	
		Order ID			Order Date	
					Delivery Date	
Line ID	Product Name	Description	Quantity	Price	Amount	
		Order ID			Order Date	
					Delivery Date	
Line ID	Product Name	Description	Quantity	Price	Amount	

### **Bonus Points:**

1. Display the highest amount for order using a combination of summary column and placeholder column. This needs to be displayed at the end of the report.
2. Display a Message using a report trigger. Also customer name needs to be provided as parameter. The parameter is converted into customer id in Report Trigger after parameter and used towards querying the transactions for building the report.
3. Control the sort order of the output using a lexical parameter.

**Note :** Towards this case lite solution, please execute the SQL script

SQL\_CL\_Sol\_All.sql and PLSQL script PLSQL\_CL\_Sol\_All.sql. This will create associated tables and populate relevant transactions, which can be used for report & BI case build.

Unauthorized reproduction or distribution prohibited. Copyright© 2019, Oracle and/or its affiliates.

Shital jadhav (shitaljagdale99@gmail.com) has a non-transferable  
license to use this Student Guide.