

Oracle9i Reports

Building Reports

Release 9.0

March 2002

Part No. A92101-01

ORACLE[®]

Oracle9i Reports Building Reports, Release 9.0

Part No. A92101-01

Copyright © 2002, Oracle Corporation. All rights reserved.

Primary Author: Vanessa Wang

Contributors: Christian Bauwens, Darren McBurney, Jim Safcik, Philipp Weckerle

The Programs (which include both the software and documentation) contain proprietary information of Oracle Corporation; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent and other intellectual and industrial property laws. Reverse engineering, disassembly or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. Oracle Corporation does not warrant that this document is error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Oracle Corporation.

If the Programs are delivered to the U.S. Government or anyone licensing or using the programs on behalf of the U.S. Government, the following notice is applicable:

Restricted Rights Notice Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and Oracle Corporation disclaims liability for any damages caused by such use of the Programs.

Oracle is a registered trademark, and Oracle9i and PL/SQL are trademarks or registered trademarks of Oracle Corporation. Other names may be trademarks of their respective owners.

Contents

Send Us Your Comments	vii
Preface.....	ix
1 Building a Report with a Barcode	
1.1 Prerequisites for this example	1-2
1.1.1 Example files.....	1-2
1.1.2 Access to the sample schema.....	1-4
1.1.3 Update the REPORTS_CLASSPATH environment variable	1-4
1.2 Create a barcode report for paper	1-5
1.2.1 Import the Java classes into Reports Builder	1-5
1.2.2 Create a package to store your information.....	1-6
1.2.3 Create a Before Report trigger.....	1-7
1.2.4 Create a data model with a formula column	1-8
1.2.5 Create a layout for your report	1-12
1.3 Create a barcode report for the Web	1-14
1.3.1 Create a query in an existing HTML file	1-15
1.3.2 Create three formula columns in your data model.....	1-17
1.3.3 Initialize the barcode JavaBean and set its properties	1-20
1.3.4 Run your report to the Web.....	1-24
1.4 Summary	1-25
2 Bursting and Distributing a Report	
2.1 Prerequisites for this example	2-2

2.1.1	Example files	2-2
2.1.2	Access to the sample schema	2-3
2.2	Set up an existing report for bursting	2-3
2.3	Edit the distribution XML definition	2-5
2.4	Run the report	2-7
2.5	Summary	2-8

3 Building a Paper Report with Ref Cursors

3.1	Prerequisites for this example	3-3
3.1.1	Access to the sample Human Resources schema	3-4
3.2	Defining a ref cursor type	3-4
3.3	Creating a ref cursor query	3-6
3.4	Refining the data model	3-9
3.5	Creating links between ref cursor queries	3-11
3.6	Adding summary columns	3-13
3.7	Creating a layout	3-15
3.8	Moving the SELECT statement into a package	3-18
3.9	Moving the packages into a library	3-21
3.10	Summary	3-22

4 Building a Report with an XML Pluggable Data Source

4.1	Prerequisites for this example	4-2
4.1.1	Example files	4-2
4.1.2	Access to the sample schema	4-4
4.2	Create a report manually with SQL and XML queries	4-4
4.2.1	Create a SQL query for your new report	4-4
4.2.2	Create an XML query to access your XML data source	4-6
4.2.3	Create a data link between two queries	4-7
4.2.4	Create a layout for your report using the Report Wizard	4-9
4.2.5	Apply alternating row colors to your report	4-10
4.2.6	Filter your XML data using groups	4-13
4.3	Run your report to paper	4-16
4.4	Summary	4-17

5 Building a Report with a Text Pluggable Data Source

5.1	Prerequisites for this example	5-2
5.2	Set up the textpds.conf file	5-3
5.3	Use the Report Wizard to create a report	5-4
5.4	Summary	5-6

6 Building a Report Using Express Data

6.1	Prerequisites for this example	6-2
6.1.1	Example files	6-2
6.1.2	Access to an Express Server	6-3
6.2	Creating an Express report with the Report Wizard	6-3
6.3	Refining the Express query	6-7
6.4	Adding summary columns and custom measures to the data model	6-9
6.4.1	Renaming data objects	6-11
6.4.2	Creating summary columns	6-11
6.4.3	Creating a custom measure	6-12
6.5	Enhancing the report layout	6-15
6.5.1	Inserting summary fields in the report	6-15
6.5.2	Inserting the custom measure field into the report	6-17
6.5.3	Sorting dimension values	6-19
6.5.4	Making format changes in the Paper Design view	6-20
6.6	Summary	6-21

Glossary

Index

Send Us Your Comments

Oracle9i Reports Building Reports, Release 9.0

Part No. A92101-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this document. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most?

If you find any errors or have any other suggestions for improvement, please indicate the document title and part number, and the chapter, section, and page number (if available). You can send comments to us to this address:

Oracle Corporation
Oracle Reports Documentation
500 Oracle Parkway, 2op8
Redwood Shores, CA 94065
USA

If you would like a reply, please give your name, address, telephone number, and (optionally) electronic mail address.

If you have problems with the software, please contact your local Oracle Support Services.

Preface

The examples in this book are intended to help you learn about Oracle9i Reports and its features.

Intended audience

This book is intended for anyone who uses Oracle9i Reports to build reports. The needs of both novice and advanced users are addressed. Each chapter contains step-by-step instructions to build a report from start to finish. Each report that you build will demonstrate how to use many of the powerful features in Reports Builder.

Structure

This book contains the following chapters:

Table 0-1 Summary of Manual Contents

Chapter	Description
Chapter 1, "Building a Report with a Barcode"	Describes how to build a report that uses the barcode Java bean. Also describes how to use the Java Importer to import a Java class for a paper report, and use a Java bean with a JSP-based Web report.
Chapter 2, "Bursting and Distributing a Report"	Describes how to set up an existing paper report to burst on a section, then distribute to multiple e-mail addresses.

Table 0–1 Summary of Manual Contents

Chapter	Description
Chapter 3, "Building a Paper Report with Ref Cursors"	Describes how to define a ref cursor query, which uses PL/SQL to fetch data.
Chapter 4, "Building a Report with an XML Pluggable Data Source"	Describes how to build a report by combining an XML data source and a SQL-based data source in a single query.
Chapter 5, "Building a Report with a Text Pluggable Data Source"	Describes how to build a paper report using a text pluggable data source.
Chapter 6, "Building a Report Using Express Data"	Describes how to build a paper report using Oracle Express as a data source

We are continually building new examples of how to use Oracle9i Reports. For the latest examples, documentation, and demonstrations, visit the Oracle Technology Network (<http://otn.oracle.com/products/reports/>), then click **Getting Started with Oracle9i Reports**.

Other resources

You can find more information about using Oracle9i Reports on the Oracle Technology Network, the *Reports Builder online help*, and *Publishing Reports to the Web with Oracle9iAS Reports Services*. The latest information is available on the Oracle Technology Network (<http://otn.oracle.com/products/reports/>), under **Getting Started with Oracle9i Reports**.

The Oracle Technology Network

The Oracle Technology Network is located at <http://otn.oracle.com>. You can access general information about the product, as well as delve into more detail by navigating to **Getting Started with Oracle9i Reports**. The Getting Started Web site provides you with up-to-date examples, as well as the latest white papers and demonstrations. Here, you can access the latest version of the *Building Reports with Oracle9i Reports* and *Publishing Reports to the Web with Oracle9iAS Reports Services* manuals.

Oracle9i Reports Tutorial

The tutorial describes how to build a simple JSP-based Web report with a graph, and how to generate a paper report from the same data model. You can access this manual via the **Getting Started with Oracle9i Reports** Web site. Simply go to the Web site, click **Index**, and navigate to the *Oracle9i Reports Tutorial*.

Reports Builder online help

You can access the *Reports Builder online help* by clicking a **Help** button on any dialog box, as well as choosing **Help > Help Contents**.

Publishing Reports to the Web with Oracle9iAS Reports Services

You can access this manual on the Oracle9iAS product CD, as well as on the Oracle Technology Network. Simply go to the Web site, click **Index**, and navigate to the *Publishing Reports to the Web with Oracle9iAS Reports Services*. This book describes how to set up Oracle9iAS Reports Services to perform the tasks you require. If you are setting up your Reports server for distribution, for example, you should refer to this manual for further information.

Building a Report with a Barcode

Reports Builder enables you to create any type of report that displays barcodes. By using the Oracle9i Reports barcode JavaBean, you can build reports for the Web or for paper that display a barcode to make tasks like tracking shipping orders and employee identification numbers easier. In Reports 6i, you had to use a barcode font to generate the barcode. In Oracle9i Reports Builder, the JavaBean automatically generates the barcode for you.

To learn more about the barcode JavaBean, visit the Oracle Technology Network (<http://otn.oracle.com/products/reports/>), then click **Getting Started with Oracle9i Reports** and click **PL/SQL-Java Bridge** in the navigation bar.

Example Scenario

You will build two reports in this section, one for paper and one for the Web. The paper report shows an invoice for a single customer who has ordered multiple items from a company. The barcode indicates the tracking information for the order.

To build either of these reports, you must first refer to Section 1.1, "Prerequisites for this example".

Table 1–1 Features Demonstrated in the Barcode Bean Example

Feature	Location
Use the Java importer to add the barcode JavaBean for a paper report.	Section 1.2.1, "Import the Java classes into Reports Builder"
Use the Program Unit editor to create a PL/SQL package for a paper report.	Section 1.2.2, "Create a package to store your information"

Table 1–1 Features Demonstrated in the Barcode Bean Example

Feature	Location
Create a Before Report trigger to set up your barcode JavaBean for a paper report.	Section 1.2.3, "Create a Before Report trigger"
Use the Data Model view and toolbar to create a data model with a formula column for a paper report.	Section 1.2.4, "Create a data model with a formula column"
Create a simple JSP-based Web report.	Section 1.3.1, "Create a query in an existing HTML file"
Create formula columns to call the barcode data for your Web report.	Section 1.3.2, "Create three formula columns in your data model"
Edit the JSP code in the Web source view.	Section 1.3.3, "Initialize the barcode JavaBean and set its properties"
View your JSP-based Web report in a browser.	Section 1.3.4, "Run your report to the Web"

1.1 Prerequisites for this example

To build the examples in this manual, you must have the example files we've provided, as well as access to the sample schema that is shipped with the Oracle9i database.

1.1.1 Example files

If you haven't already done so, you can download the files you'll need to complete this example from the Oracle Technology network and install them on your machine.

To download and install the example files:

1. Go to the Oracle Technology Network Web site (<http://otn.oracle.com/product/reports/>).
2. Click **Getting Started with Oracle9i Reports**.
3. Click **Index**, then find the "Building a Paper Report with a Barcode JavaBean" example and "Building a Web Report with a Barcode JavaBean" example. To

complete this chapter, you need both sets of files. Note that this chapter covers how to build both a paper report and a Web report with a barcode bean.

4. Download the files BarcodePaper.zip and BarCodeWeb.zip into a temporary directory on your machine (e.g., "d:\temp").
5. Unzip the contents of the file, maintaining the directory structure, into an examples directory on your machine (e.g., d:\orawin90\examples).

This zip file contains the following files:

Table 1–2 Files necessary for building the barcode JavaBean sample reports

File	Description
<i>Examples\BarCodeBeanPaper\result\ShippingManifest.pdf</i>	The final PDF version of the paper report, containing the barcode.
<i>Examples\BarCodeBeanPaper\scripts\oraclebarcode.jar</i>	The barcode JavaBean.
<i>Examples\BarCodeBeanPaper\source\ShippingManifest.rdf</i>	The source file for the sample paper report. Running this RDF in Reports Builder will display the final result of your paper report in the Paper Design view.
<i>Examples\BarCodeBeanWeb\result\ShippingManifestWeb.jsp</i>	The final JSP version of the Web report, containing the barcode.
<i>Examples\BarCodeBeanWeb\result\ShippingManifestWeb.rdf</i>	The final RDF version of the Web report, containing the barcode.
<i>Examples\BarCodeBeanWeb\result\assets</i>	The images that Oracle9i Reports generated when the JSP was run.
<i>Examples\BarCodeBeanWeb\scripts\SQL.txt</i>	The SQL for the query you need to enter.
<i>Examples\BarCodeBeanWeb\source\ShippingLabel.html</i>	The HTML page that you will use as a basis for the Web report.
<i>Examples\BarCodeBeanWeb\source\ShippingManifestWeb.rdf</i>	The source file for the sample Web report. Running this RDF in Reports Builder to the Web will display the final result of your Web report in your browser.
<i>Examples\BarCodeBeanWeb\source\assets*</i>	The images and other files that your JSP-based Web report will require to display properly on the Web.

Table 1–2 Files necessary for building the barcode JavaBean sample reports

File	Description
<i>Examples</i> \BarCodeBeanWeb\source\assets\BLAFbeige_logo.gif	The image you will use in your JSP-based Web report.

1.1.2 Access to the sample schema

If you don't know if you have access to the sample schema provided with the Oracle9i database, contact your database administrator. You should have access to the "Order Entry" portion of the schema to complete this example.

1.1.3 Update the REPORTS_CLASSPATH environment variable

Before you use a Reports JavaBean (for paper or the Web), you need to perform several steps. You first need to set up your environment to use the correct classpath for the bean. For a paper report, you must then use the Java Importer to import the JavaBean into Reports Builder. For a Web report, you must call the JavaBean from your JSP-based (JavaServer Page) report.

In this section, you will update the Reports class path with the location of the JavaBean. When you launch Reports Builder, it will use this new class path to recognize the location of the barcode bean.

Note: You must follow the steps in this section before you can even run the finished report we've provided, called `ShippingManifestPaper.rdf` and `ShippingManifestWeb.rdf`.

1. Find the class path:
 - In Windows, open the registry using regedit (you may want to export a backup before you modify your registry) and update the REPORTS_CLASSPATH environment variable.
 - On UNIX, update the REPORTS_CLASSPATH environment variable.
2. Modify the class path to reflect the following:

```
ORACLE_HOME/Examples/BarCodeBeanPaper/Scripts/oraclebarcode.jar;
```

Note: ORACLE_HOME is where Reports Builder is installed (ex. d:\orawin90\) and Examples is where your examples are installed.

3. Save the new environmental variable.

You are now ready to begin building your report.

1.2 Create a barcode report for paper

In this section, you will create a paper-based report that shows the invoice for a particular customer. This invoice will display the address of the customer, his order, and a barcode that represents the tracking number for the order. The company can scan this barcode to find out the status of the order.

Next, you will import the JavaBean, then create a barcode report for paper (Acrobat PDF). If you want to learn how to create a barcode JSP-based report for the Web, skip to Section 1.3.

1.2.1 Import the Java classes into Reports Builder

To create a paper report using the barcode JavaBean, you must first import two Java classes into Reports Builder. When you import these Java classes, Reports Builder automatically creates the packages you need to build the report.

Note: You do not need to perform this task if you're creating a Web report, as you will write JSP that calls the JavaBean.

To import the Java classes:

1. Launch Reports Builder.

Note: You must launch Reports Builder now so that the new classpath is used.

2. Close the Welcome dialog box by clicking **Cancel**.
3. Choose **Program > Import Java Classes...**

4. In the **Import Classes** field of the Import Java Classes dialog box, type `oracle.apps.barcode.util.BarCodeConstants`, then click **Import**.
5. Once the packages have been created, import the second JavaBean: `oracle.apps.barcode.BarCodeMaker`.
6. Click **Close**.
7. In the Object Navigator, look under the report called "MODULE 1." Click the **Program Units** node. You'll notice that Reports Builder created two package specs and two package bodies named **BARCODECONSTANTS** and **BARCODEMAKER**.

1.2.2 Create a package to store your information

In this report, you want to create a package where the information will be stored.

To create a package for storing your information:

1. In the Object Navigator, under your new report, click the **Program Units** node.
2. Click the **Create** icon to add a program unit.
3. In the New Program Unit dialog box, type `globals`.
4. Select the **Package Spec** radio button, then click **OK**.
5. Type the following code in the editor:

```
PACKAGE globals IS
    bcobj ora_java.jobject;
    barcode_to_use varchar2(256);
    tempdir varchar2(100);
    directory_sep varchar2(2);
END
```

You can also copy and paste the code from the complete example (`Examples\BarCodeBeanPaper\source\ShippingManifest.rdf`). Open the RDF in the Object Navigator. Under the example name, open the Program Units node. You should see the "Globals" package body listed. Open this package, then copy and paste the code from this package into your new program unit.

6. Click **Compile** to make sure there are no errors in your code.

Note: If your code does not compile, make sure you've typed in exactly the code we've provided.

7. Once the code is compiled, click **Close**.
8. In the Object Navigator, select your report (e.g., "MODULE 1").
9. Choose **File > Save**.
10. Name the file `shippingmanifest_<your initials>.rdf` (e.g., `shippingmanifest_vw`) and make sure you save it to a new directory (e.g., "My Examples"), where your new files are stored.
11. Click **Save**.

You have created a package that will contain the global information for your report.

1.2.3 Create a Before Report trigger

You can use the Before Report trigger to initialize specific tasks that will run before the report runs. Here, you will define the type of barcode you want to use in your report, as well as the temporary directory where your barcode images will be stored.

1. In the Object Navigator, under `SHIPPINGMANIFEST_<your initials>`, click the **Report Triggers** node.
2. Double-click **BEFORE REPORT**.
3. Type the following code in the editor, between the existing text (indicated in bold):

```
function BeforeReport return boolean is  
  begin  
    globals.barcode_to_use := BarCodeConstants.BAR_CODE_128;  
    globals.bcobj := barcodemaker.new();  
return (TRUE);  
end;
```

You can change the value `BarCodeConstants.BAR_CODE_128` to any other valid value. To determine which values are valid, check the contents of the package by opening the `BarCodeConstants` package spec in the Object Navigator, under the Program Units node.

Note: You can also copy and paste the code from the complete example (`Examples\BarCodeBeanPaper\source\ShippingManifest.rdf`). Open the RDF in the Object Navigator. Under the example name, open the Report Triggers node. You should see the "BEFORE REPORT TRIGGER" listed. Open this trigger, then copy and paste the code from the editor into your new function.

4. Click **Compile** to make sure there aren't any errors.

Note: If you have errors, make sure you've imported the necessary Java classes and that your code matches the code above. If you change the code, be sure to compile it again

5. When the code is compiled, click **Close**.
6. Save your report.

You have created a trigger that will set up the barcode type for you when you run the report.

1.2.4 Create a data model with a formula column

In this section, you will manually create the query that the report will use to retrieve data from the sample schema. You will also create a formula column that will communicate with the JavaBean to create the barcode, then return the file name of the generated image.

1.2.4.1 Create the query

1. In the Object Navigator, under `SHIPPINGMANIFEST_<your initials>`, double-click **Data Model** to display the Data Model view for your report.
2. In the tool palette, click the **SQL Query** icon, then click in the main area (canvas region) of the Data Model view.
3. In the **SQL Query Statement** field, type (or paste) the following code:

```
SELECT ALL CUSTOMERS_A1.CUST_FIRST_NAME,  
        CUSTOMERS_A1.CUSTOMER_ID, CUSTOMERS_A1.CUST_LAST_NAME,  
        CUSTOMERS_A1.CUST_ADDRESS.STREET_ADDRESS,
```

```

CUSTOMERS_A1.CUST_ADDRESS.POSTAL_CODE,
CUSTOMERS_A1.CUST_ADDRESS.CITY,
CUSTOMERS_A1.CUST_ADDRESS.STATE_PROVINCE,
CUSTOMERS_A1.CUST_ADDRESS.COUNTRY_ID, ORDERS.ORDER_ID,
ORDERS.ORDER_DATE,
ORDERS.ORDER_TOTAL, ORDER_ITEMS.LINE_ITEM_ID,
PRODUCTS.PRODUCT_NAME,
ORDER_ITEMS.UNIT_PRICE, ORDER_ITEMS.QUANTITY,
COUNTRIES.COUNTRY_NAME
FROM CUSTOMERS CUSTOMERS_A1, ORDER_ITEMS, ORDERS,
PRODUCTS, HR.COUNTRIES
WHERE ((ORDER_ITEMS.ORDER_ID = ORDERS.ORDER_ID)
AND (ORDERS.CUSTOMER_ID = CUSTOMERS_A1.CUSTOMER_ID)
AND (ORDER_ITEMS.PRODUCT_ID = PRODUCTS.PRODUCT_ID)
AND (CUSTOMERS_A1.CUST_ADDRESS.COUNTRY_ID =
HR.COUNTRIES.COUNTRY_ID))
AND ORDERS.ORDER_ID = :P_ORDER_ID
ORDER BY order_ID, line_item_ID

```

You can also copy and paste the code from the complete example (Examples\BarCodeBeanPaper\source\ShippingManifest.rdf). Open the RDF in the Object Navigator. Under the example name, double-click **Data Model** to display the Data Model view. Double-click the query (entitled Q_1) to display the SQL Query Statement field.

4. Click **OK**.

If you are not connected to a database that contains the sample schema we've provided, you must log in now. If you're not sure what your connection string is, contact your database administrator.

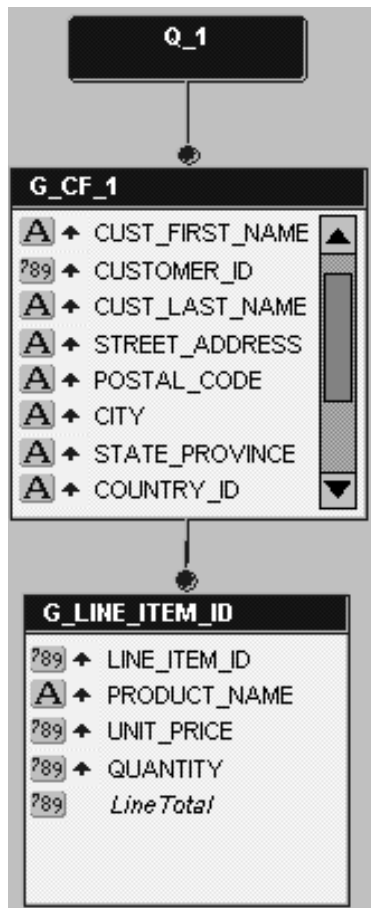
5. When a message displays indicating that the bind parameter `p_order_id` was created, click **OK**.

6. In the data model you just created, select all of the following columns at once by using your mouse and the Shift key, then drag them below the current query into a detail group:

- LINE_ITEM_ID
- PRODUCT_NAME
- UNIT_PRICE
- QUANTITY

The resulting data model should look like this:

Figure 1–1 Data Model for the query



1.2.4.2 Create the formula column

To create the formula column:

1. In the data model for your report, click the Formula Column icon to create a formula column.
2. Click in the master group (the main group that still contains most of the column names).

The Property Inspector for the formula column (CF_1) displays.

3. Under Column, next to the **Datatype** property, select **Character** from the drop-down list.
4. Click the **PL/SQL Formula** property to display the editor for the formula column.
5. Type the following code in the editor (existing text is in bold):

```
function CF_1Formula return Char is  
    myFilename varchar2(20);  
    result varchar2(20);  
    barcodeData VarChar2(50) := :customer_ID ||  
:order_ID;  
begin  
    myFileName := srw.create_temporary_filename;  
    barcodemaker.setBarWidthInch(globals.bcobj, 0.005);  
    barcodemaker.setBaseCodeData(globals.bcobj,barcodeData);  
    barcodemaker.setBarCodeType(globals.bcobj,globals.barcode_to_use);  
    barcodemaker.setFullPath(globals.bcobj, myFileName);  
    barcodemaker.renderBarCode(globals.bcobj);  
    return(myfilename);  
end;
```

You can also copy and paste the code from the complete example (*Examples\BarCodeBeanPaper\source\ShippingManifest.rdf*). Open the RDF in the Object Navigator. In the Data Model view, open the Property Inspector for the formula column called "CF_1." Click the **PL/SQL Formula** property to display the code for the formula column.

6. Click **Compile** to make sure there aren't any errors.

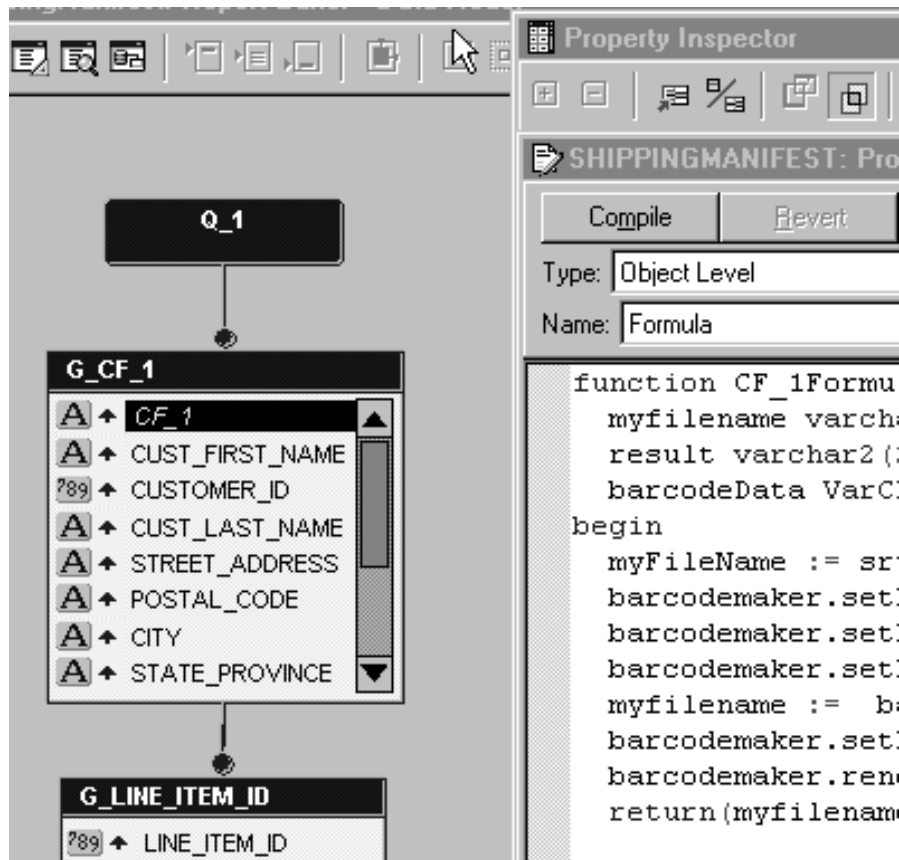
Note: If you have errors, make sure you've imported the necessary Java classes and that your code matches the code above. If you change the code, be sure to compile it again.

7. When the code is compiled, click **Close**.
8. In the Property Inspector, find the **Read from File** property and set it to Yes.
9. Set the **File Format** property to Image.
10. Save the report.

You have created the data model for your barcode report, which contains a formula column that retrieves the barcode information and displays the barcode image on your report.

Your data model and the PL/SQL for the formula column should look similar to this:

Figure 1–2 Data Model and Program Unit Editor for the Formula Column CF_1



1.2.5 Create a layout for your report

Before you can run your report, you must create a layout.

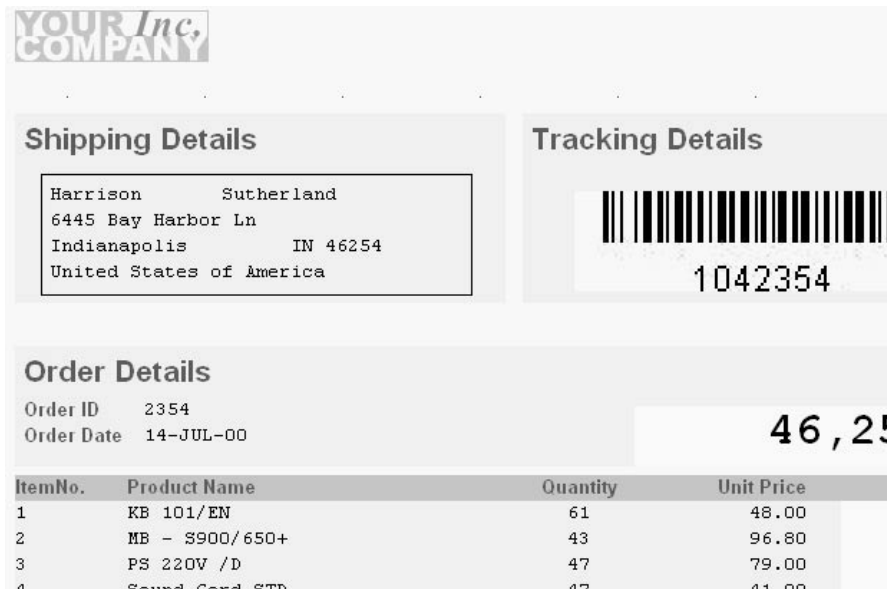
To create a paper layout:

1. Under your report's node in the Object Navigator, right-click **Paper Layout**, then choose **Report Wizard** from the pop-up menu.
2. On the Report Type page, click **Next** to create both a Web and paper layout.
3. On the Style page, select the **Group Above** radio button, then click **Next**.
4. On the SQL page, click **Next**.
5. On the Data page, click **Next**.
6. On the Groups page, make sure the following fields display in the Group Fields column:
 - ORDER_ID
 - ORDER_DATE
 - CUSTOMER_ID
 - CUST_FIRST_NAME
 - CUST_LAST_NAME
 - STREET_ADDRESSS
 - POSTAL_CODE
 - COUNTRY_NAME
 - CITY
 - STATE_PROVINCE
 - COUNTRY_ID
 - ORDER_TOTAL
 - CF_1
7. On the Fields page, click the double arrows to display all fields.
8. On the Totals page, make sure no totals display, then click **Finish**.

You have completed the layout for your paper report. When the report runs, the Runtime Parameter Form displays. Next to P_ORDER_ID, type 2354.

Your report displays in the Paper Design view, and should look something like the following:

Figure 1–3 Paper Design view of the Barcode Paper Report



Note: If you aren't sure whether you produced the desired results, you can always open the file we provided, called `ShippingManifest.pdf` in Acrobat Reader. Or, you can run `ShippingManifest.rdf` to paper and the report will display in the Paper Design view.

1.3 Create a barcode report for the Web

The steps in this section show you how to build a Web report using JavaServer Pages (JSPs), that uses the barcode JavaBean you imported in Section 1.2.1, "Import the Java classes into Reports Builder". If you want to build a paper report with a barcode, see Section 1.2, "Create a barcode report for paper".

If you are not familiar with creating a JSP-based Web report and would like an introduction to building them, refer to the *Oracle9i Reports Tutorial*, located in the **Getting Started with Oracle9i Reports** Web site on the Oracle Technology Network (<http://otn.oracle.com/reports/>).

The report you will create in this section is the same as the one you created for paper. You will create a report that displays the invoice for a particular customer.

This invoice will display the address of the customer, his order, and a barcode that represents the tracking number for the order. The company can scan this barcode to find out the status of the order.

You can run the final version of the JSP report we've provided to see what you'll build in these steps, but please note that you will need to update the location of the images in the source code (see Section 1.3.3.1, "Update the paths to the assets") before you can run the report to the Web.

If you're not familiar with creating a JSP-based Web report, refer to the *Oracle9i Reports Tutorial* to learn how to create a simple JSP-based Web report.

Note: Before you begin this section, make sure you have all the necessary files, and that you've imported the Java classes, and set up the class path. See Section 1.1, "Prerequisites for this example" and Section 1.2.1, "Import the Java classes into Reports Builder".

1.3.1 Create a query in an existing HTML file

When you create a JSP-based Web report, you can use an existing HTML file as a template.

To create a query in an existing HTML file:

1. In Reports Builder, open the file
Examples\BarCodeBeanWeb\source\ShippingLabel.html.
If you don't know how to open the file, follow these steps:
 - a. In Reports Builder, choose **File > Open**.
 - b. Navigate to your examples directory. This is the directory where you've downloaded the examples files we've provided.
 - c. Find and open the file called *ShippingLabel.html*.
2. In the Object Navigator, under SHIPPINGLABEL, double-click the **Data Model** node to display the Data Model view for the report.
3. In the tool palette, click the **SQL Query** icon, then click in the main area (canvas region) of the Data Model view.
4. In the **SQL Query Statement** field, type (or paste) the following code:

```
SELECT ALL CUSTOMERS_A1.CUST_FIRST_NAME,
```

```
CUSTOMERS_A1.CUSTOMER_ID, CUSTOMERS_A1.CUST_LAST_NAME,
CUSTOMERS_A1.CUST_ADDRESS.STREET_ADDRESS,
CUSTOMERS_A1.CUST_ADDRESS.POSTAL_CODE,
CUSTOMERS_A1.CUST_ADDRESS.CITY,
CUSTOMERS_A1.CUST_ADDRESS.STATE_PROVINCE,
CUSTOMERS_A1.CUST_ADDRESS.COUNTRY_ID,
ORDERS.ORDER_ID, ORDERS.ORDER_DATE, ORDERS.ORDER_TOTAL,
COUNTRIES.COUNTRY_NAME FROM CUSTOMERS CUSTOMERS_A1, ORDERS,
HR.COUNTRIES
WHERE ((ORDERS.CUSTOMER_ID = CUSTOMERS_A1.CUSTOMER_ID)
AND (CUSTOMERS_A1.CUST_ADDRESS.COUNTRY_ID = HR.COUNTRIES.COUNTRY_ID))
AND ORDERS.ORDER_ID = :P_ORDER_ID ORDER BY order_ID
```

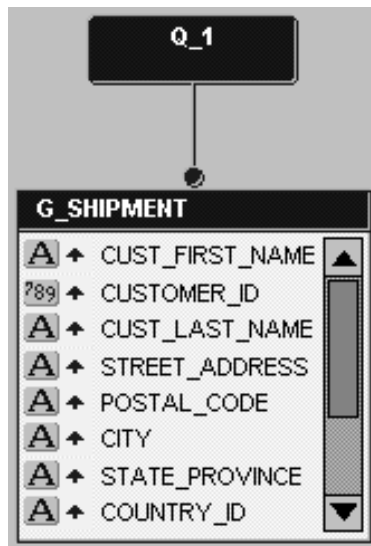
Note: You can also copy and paste the code from the file we've provided: *Examples\BarCodeBeanWeb\scripts\SQL.txt*.

5. Click **OK**.

If you are not connected to a database that contains the sample schema we've provided, you must log in now. If you're not sure what your connection string is, contact your database administrator.

6. When a message displays indicating that the bind parameter was created, click **OK**.
7. In the Data Model view, right-click the group and choose **Property Inspector** to display the properties for the group.
8. In the **Name** field, delete the existing text and type `G_SHIPMENT`, then press **Enter**.
Doing so renames the group "G_SHIPMENT."
9. Close the Property Inspector.
10. Your data model should look something like this:

Figure 1–4 Data Model for the JSP-based Web Report Query



11. Save your report as `ShippingLabel_<your initials>.jsp`.

Doing so creates the JSP-based Web source for this report.

You have now created the query that will pull in the data for your report.

1.3.2 Create three formula columns in your data model

You will need to create three formula columns in your report to retrieve the tracking number for the order, the origin of the order, and the destination for the order.

1.3.2.1 Create the TrackingNumber formula column

To create the TrackingNumber formula column:

1. In the Data Model view, click the **Formula Column** icon in the tool palette.
2. Click in the group to create your first formula column, temporarily called **CF_1**.
3. In the Property Inspector for **CF_1**, rename the column **TrackingNumber**.
4. Next to the Datatype property, choose **Character** from the list.
5. Next to the **PL/SQL Formula** property, click the field to display the code editor.

6. Type the following code in the editor (existing text is in bold):

```
function TrackingNumberFormula return char is
begin
    return(:Customer_id||:Order_ID||:country_ID);
end;
```

You can also copy and paste the code from the complete example (*Examples\BarCodeBeanWeb\source\ShippingManifestWeb.rdf*). Open the RDF in the Object Navigator. Double-click the Data Model node to display the data model. In the group, scroll down to find the "TrackingNumber" formula column, then open the Property Inspector for the column. Click the field next to "PL/SQL Formula" to display the code.

7. Click **Compile**.

Note: If your code does not compile, make sure you've typed in exactly the code we've provided.

8. When the code is compiled, click **Close**.

The new formula column, *TrackingNumber*, displays in the data model.

1.3.2.2 Create the OriginScan formula column

To create the OriginScan formula column:

1. Create another formula column, temporarily called **CF_1**, in the G_SHIPMENT group.
2. In the Property Inspector for **CF_1**, rename the column OriginScan.
3. Choose the **Character** datatype.
4. Next to the **PL/SQL Formula** property, click the field to display the code editor.
5. Type the following code in the editor (existing text is in bold):

```
function OriginScanFormula return char is
begin
    return('34324-OH-US');
end;
```

You can also copy and paste the code from the complete example

(*Examples\BarCodeBeanWeb\source\ShippingManifestWeb.rdf*). Open the RDF in the Object Navigator. Double-click the Data Model node to display the data model. In the group, scroll down to find the "OriginScan" formula column, then open the Property Inspector for the column. Click the field next to "PL/SQL Formula" to display the code.

6. Click **Compile**.

Note: If your code does not compile, make sure you've typed in exactly the code we've provided.

7. When the code is compiled, click **Close**.

The new formula column, *OriginScan*, displays in the data model.

1.3.2.3 Create the DestinationScan formula column

To create the DestinationScan formula column:

1. Create another formula column, temporarily called **CF_1**, in the G_SHIPMENT group.
2. In the Property Inspector for **CF_1**, rename the column *DestinationScan*.
3. Choose the **Character** datatype.
4. Next to the **PL/SQL Formula** property, click the field to display the code editor.
5. Type the following code in the editor (existing text is in bold):

```
function DestinationScanFormula return char is
begin
    return(:postal_code||'-'||:state_province||'-'||:country_ID);
end;
```

You can also copy and paste the code from the complete example (*Examples\BarCodeBeanWeb\source\ShippingManifestWeb.rdf*). Open the RDF in the Object Navigator. Double-click the Data Model node to display the data model. In the group, scroll down to find the "DestinationScan" formula column, then open the Property Inspector for the column. Click the field next to "PL/SQL Formula" to display the code.

6. Click **Compile**.

Note: If your code does not compile, make sure you've typed in exactly the code we've provided.

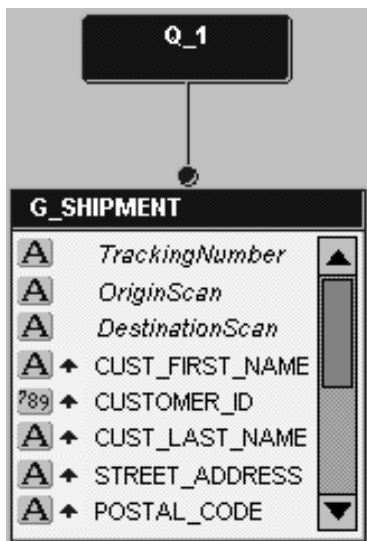
- When the code is compiled, click **Close**.

The new formula column, *DestinationScan*, displays in the data model.

- Save your report as a JSP.

You have created the three formula columns that will hold the values for the tracking number, the origin, and the destination of the order. Your data model should look something like this:

Figure 1–5 Data Model with the Three Formula Columns



1.3.3 Initialize the barcode JavaBean and set its properties

To tell your JSP-based Web report to communicate with the JavaBean, you need to initialize it in the JSP. Unlike using the JavaBean with a paper report, you do *not* need to use the Java importer to import the Java classes.

In this section, you will also learn how to set the properties for the bean so that the correct data is being used to produce the barcode.

If you don't want to bother with typing the code in yourself, you can always open the source file (*Examples/BarCodeWeb/source/ShippingManifestWeb.rdf*) and copy the appropriate pieces of the Web source into your report.

1.3.3.1 Update the paths to the assets

To ensure that the JavaBean references the correct barcode images, you must update your report with the correct path.

To update the paths:

1. Open the Web source for your JSP-based Web report, called `ShippingLabel_<your initials>`.

In the Data Model view of your report, click the **Web Source** icon.

2. In the Web source, look for the text: `Define Path` information for your barcode images.
3. Under this text, update the paths to make sure they point to the directories where your images will be generated (e.g., `d://temp//docroot//images//` and `images//`). Be sure to maintain the integrity of the paths we provide.

Note: The directory where these files are located must be accessible by the Web server. This directory is typically located somewhere below the directory where you keep your JSPs.

1.3.3.2 Initialize the JavaBean

To initialize the JavaBean:

1. In the Web source, look for the text: `Initialize the JavaBeans`.
2. Under this text, type the following code:

```
<jsp:useBean id="BC" scope="page"
class="oracle.apps.barcode.util.BarCodeConstants" />
<jsp:useBean id="BM" scope="page" class="oracle.apps.barcode.BarCodeMaker"
/>
```

1.3.3.3 Set the barcode JavaBean properties

To set the JavaBean properties:

1. In the Web source, look for the text: `Setting the barcode's properties`.

2. Under this text, type the following code:

```
<jsp:setProperty name="BM" property="BarCodeType" value="<%= BC.BAR_CODE_
128 %>" />
<jsp:setProperty name="BM" property="BarWidthInch" value="0.01"/>
<jsp:setProperty name="BM" property="Directory" value="<%=
BarcodePhysicalPath %>"/>
```

1.3.3.4 Define the barcode variables

To define the barcode variables:

1. In the Web source, look for the text: Define variables to hold the data for the three barcodes.
2. Under the text, type the following code:

```
<%! private String BarCodeData1 = "12345-XX-XX"; %>
<%! private String BarCodeData2 = "12345-XX-XX"; %>
<%! private String BarCodeData3 = "12345-XX-XX"; %>
```

1.3.3.5 Create a For Each loop

To create a For Each loop:

1. In the Web source, look for the text: Replace this with your RW:FOREACH open tag.
 2. Replace the line with the following code:
- ```
<rw:foreach id="R_G_SHIPMENT" src="G_SHIPMENT">
```
3. Look for the text: Replace this with your RW:FOREACH close tag.
  4. Replace the line with the following code:

```
</rw:foreach>
```

### 1.3.3.6 Code the formula columns to render the barcodes

1. In the Web source, look for the text: **\*\*BARCODETrackingNumber\*\***.
2. Under the text, type the following code:

```
<!-- Get the value of the TrackingNumber and assign it to the variable -->
<rw:getValue id="BarCodeData1" src="TrackingNumber"/>
<!-- Set the data for the barcode and the filename -->
```

```

<jsp:setProperty name="BM" property="BaseCodeData" value="<%= BarCodeData1
%>"/>
<jsp:setProperty name="BM" property="FileName" value="<%= BarCodeData1 %>"/>
<!-- Render the barcode -->
<% BM.renderBarCode(); %>
<!-- View the image in the page -->


```

The line beginning with "`<!--`" are comments. If you don't want comments in your code, you don't have to add these lines.

3. In the Web source, look for the text: `**BARCODEOriginScan**`.
4. Under the text, type the following code:

```

<!-- Get the value of the OriginScan and assign it to the variable -->
<rw:getValue id="BarCodeData2" src="OriginScan"/>
<!-- Set the data for the barcode and the filename -->
<jsp:setProperty name="BM" property="BaseCodeData" value="<%= BarCodeData2
%>"/>
<jsp:setProperty name="BM" property="FileName" value="<%= BarCodeData2 %>"/>
<!-- Render the barcode -->
<% BM.renderBarCode(); %>
<!-- View the image in the page -->


```

The line beginning with "`<!--`" are comments. If you don't want comments in your code, you don't have to add these lines.

5. In the Web source, look for the text: `**BARCODEDestinationScan**`.
6. Under the text, type the following code:

```

<!-- Get the value of the DestinationScan and assign it to the variable -->
<rw:getValue id="BarCodeData3" src="DestinationScan"/>
<!-- Set the data for the barcode and the filename -->
<jsp:setProperty name="BM" property="BaseCodeData" value="<%= BarCodeData3
%>"/>
<jsp:setProperty name="BM" property="FileName" value="<%= BarCodeData3 %>"/>
<!-- Render the barcode -->
<% BM.renderBarCode(); %>
<!-- View the image in the page -->


```

**Note:** the line beginning with "<!--" are comments. If you don't want comments in your code, you don't have to add these lines.

7. Save your report as a JSP. If you're not sure whether you copied the code in correctly, check the *Examples/result/ShippingManifestWeb.rdf* we've provided.

### 1.3.4 Run your report to the Web

Since you've created a Web report, you must run this report to the Web to see your results.

1. Make sure your report, `ShippingManifestWeb_<your initials>`, is selected in the Object Navigator.
2. In the toolbar, click the **Run Web Layout** icon to run your report to a browser.

Your report displays in your Web browser, and should look something like this:

*Figure 1–6 Snapshot of the Final JSP-based Web Report with Barcode*



---

---

**Note:** If you aren't sure whether you produced the desired results, you can always open the file we provided in the *results* directory, called `ShippingManifestWeb.html` in your Web browser. Or, you can run `ShippingManifestWeb.jsp` to the Web, and the report will display in the your browser.

---

---

## 1.4 Summary

Congratulations! You have created a paper report and a JSP-based Web report that use the barcode JavaBean to generate barcode images.

You now know how to:

- Use the Java importer to add Java classes to a paper report
- Use JSP to call a JavaBean from within a report
- Create a PL/SQL package
- Use a Before Report trigger to tell Reports Builder what type of barcode image to use
- Manually build a data model with a SQL query and formula columns
- Edit the Web source for a JSP-based Web report
- Set up a JavaBean in JSP

For more information on JSPs (JavaServer Pages), refer to the *Oracle9i Reports Tutorial or the Reports Builder online help*.



---

## Bursting and Distributing a Report

Oracle9i Reports enables you to deliver a single report to multiple destinations simultaneously. By taking advantage of this feature, you can create a single report, then send it in any format (e.g., PDF or HTML) to multiple destinations.

In this example, you will modify a simple report we've provided to burst each section to a separate report. You will then modify a sample distribution XML file to send an e-mail to each destination with an attachment based on the separate reports. You will also send multiple e-mails to the same e-mail address with a single attachment (the entire report).

### About bursting and distribution

Oracle9i Reports enables you to deliver a single report to multiple destinations simultaneously. Using the new, enhanced distribution feature, you can set up your report to be distributed to an e-mail destination, a portal, a printer, or anywhere else when the report is run. This feature also enables you to improve performance, since you only fetch the data once for many different formats and destinations. Using distribution also reduces your maintenance overhead because you only need one job request to publish the report to multiple destinations. You can refine this further by sending the header section to some recipients, the main to others, and the entire report to an entirely different recipient list.

For more information on bursting and distribution, see *Publishing Reports to the Web with Oracle9iAS Services* manual located on the Oracle Technology Network (<http://otn.oracle.com>) and on the documentation CD provided with Oracle9iAS.

### Example Scenario

Suppose you are the report developer for a manufacturing company who needs to deliver monthly information to its warehouses. In this example, you will modify an existing report to burst on sections, based on each warehouse ID, creating separate

PDF reports for each warehouse. You will then edit the distribution XML file we've provided to e-mail each section as an attachment to individual warehouses.

**Table 2–1 Features demonstrated in the Bursting and Distribution example**

| Feature                                                                                                               | Location                                              |
|-----------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------|
| Modify an existing report (or one that you created with the Report Wizard) for bursting on a section.                 | Section 2.2, "Set up an existing report for bursting" |
| Modify the XML definition file for distribution.                                                                      | Section 2.3, "Edit the distribution XML definition"   |
| Run the report to PDF format so that the report is e-mailed to the destinations defined in the distribution.xml file. | Section 2.4, "Run the report"                         |

## 2.1 Prerequisites for this example

To build the examples in this manual, you must have the example files we've provided, as well as access to the sample schema that is shipped with the Oracle9i database.

### 2.1.1 Example files

If you haven't already done so, you can download the files you'll need to complete this example from the Oracle Technology network (<http://otn.oracle.com/products/reports>) and install them on your machine.

#### To download and install the example files:

1. Go to the Oracle Technology Network Web site (<http://otn.oracle.com/product/reports>).
2. Click **Getting Started with Oracle9i Reports**.
3. Click **Index**, then find the "Bursting and Distributing a Report" example.
4. Download the file `Distribution.zip` into a temporary directory on your machine (e.g., "d:\temp").



5. Unzip the contents of the file, maintaining the directory structure, into an examples directory on your machine (e.g., d:\orawin90\examples).

This zip file contains the following files:

**Table 2–2 Files necessary for building the sample report for bursting and distribution**

| File                                                           | Description                                                             |
|----------------------------------------------------------------|-------------------------------------------------------------------------|
| <i>Examples</i> \Distribution\source\distribution.xml          | The XML file that controls the distribution properties for your report. |
| <i>Examples</i> \Distribution\source\inventory_report_dist.rdf | The report you will distribute.                                         |
| <i>Examples</i> \Distribution\result\inventory_report_dist.rdf | The report you will burst.                                              |
| <i>Examples</i> \Distribution\result\REP_*.pdf                 | The PDFs that are generated when you distribute and burst your report.  |

---



---

**Note:** You can save the `distribution.xml` file we provide and reuse it later, so you don't have to create another one from scratch.

---



---

## 2.1.2 Access to the sample schema

If you don't know if you have access to the sample schema provided with the Oracle9i database, contact your database administrator. You should have access to the "Order Entry" portion of the schema to complete this example.

## 2.2 Set up an existing report for bursting

For the purposes of this chapter, we've provided you with an RDF file you can use for bursting. When you open this report, you should also connect to the sample schema in your database by choosing **File > Connect**. If you don't know the connection information for the "Order Entry" portion of the schema, contact your database administrator.

In this section, you will set up a repeating frame so that the data bursts on each warehouse ID. This way, you can later send the data resulting from each section as a report to each individual warehouse.

### To set up a repeating frame for bursting:

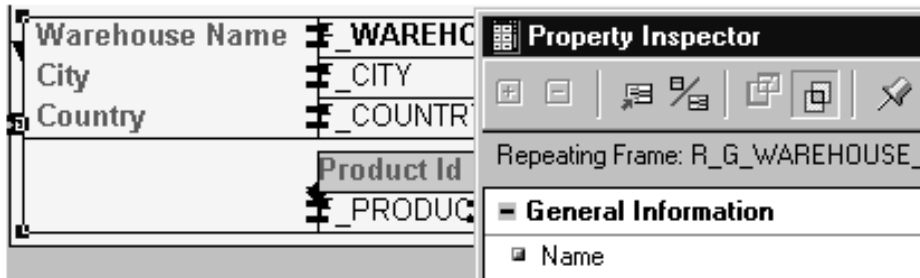
1. In Reports Builder, choose **File > Open**.

2. Navigate to the directory where your examples source files are located, and open the file `inventory_report_dist.rdf`.

The report displays in the Object Navigator.

3. In the Object Navigator, under the report name, double-click the **Paper Layout** node to display the Paper Layout view.
4. In the Paper Layout view, select the outermost repeating frame (in the Property Inspector, the name is `R_G_WAREHOUSE_ID`), then delete it.

*Figure 2–1 Deleting the Repeating Frame*



---

**Note:** You must delete the existing repeating frame because you will modify the report to burst on each section. Later, you will distribute each section to each warehouse e-mail ID

---

**To delete the repeating frame:**

- You can select the frame in the Object Navigator:
- Under the report name, double-click the **Body** node to expand the list of individual layout elements.
- Under the Body node, double-click the **R\_G\_WAREHOUSE\_ID** node.
- The repeating frame is selected in the Paper Layout view.
- In the Paper Layout view, delete the repeating frame.

**Tip:** Do not delete the repeating from in the Object Navigator, as you will delete all the objects within the frame, as well.

For more information on frames and repeating frames, refer to the *Reports Builder online help*.

5. In the Object Navigator, right-click the **Main Section** node, then choose **Property Inspector**.

The Property Inspector for the Main Section displays.

6. Under Section, find the **Repeat On** property and set it to **G\_WAREHOUSE\_ID**.

You can set the Repeat On property by clicking the drop-down list, then selecting the item.

7. Save the report as `inventoryreport_dist_<your initials>.rdf`.

You have set up your report to burst based on the warehouse ID.

## 2.3 Edit the distribution XML definition

The Oracle9i Reports distribution XML file enables you to specify the details of your distribution. For example, if you're distributing via e-mail, you can specify such details as the addressee, the reply to address, and the subject.

In this section, we'll show you how to modify a distribution XML file. We've indicated locations where you need to enter your own information to make the distribution work. Note that you can save this `distribution.xml` file to use later, so that you don't have to manually create another file every time you want to use distribution. Our sample file also contains comments that might come in handy later.

When you want to distribute a report, you need to either:

- Make sure your source report (e.g., `inventoryreport_dist.rdf`) and your distribution XML file (e.g., `distribution.xml`) are in the same directory.
- OR
- When you run the report from the Reports Server, set the destination to the path of the XML file. (We'll explain this in Section 2.4, "Run the report".)

We've provided both these files in a single directory:

*Examples/Distribution/source/*.

For more information on distribution, see the *Publishing Reports to the Web with Oracle9iAS Reports Services* manual available on the Oracle Technology Network (<http://otn.oracle.com>).

**To edit the distribution XML file:**

1. In a text editor, such as Notepad, open the file we've provided called `distribution.xml`.
2. Find the placeholder text we've provided: `<YourFilePath>`, and replace it with the location of where your resulting PDFs will be stored.

**Example:** Replacing the placeholder text with:

```
d:\temp\
changes the path to:
d:\temp\Rep_&<city&>.pdf
```

Using this complete path would place the resulting PDF files in your `d:\temp` directory.

3. Find the placeholder text we've provided: `<OriginEmailAddress>`, and replace it with the sender's e-mail address.
4. Perform step 3 for all instances of the placeholder text: `<OriginEmailAddress>`.
5. Find the placeholder text we've provided: `<DestinationEmailAddress>`, and replace it with the first recipient's e-mail address.
6. Perform step 5 for all instances of the placeholder text: `<DestinationEmailAddress>`.

---

---

**Note:** For this example, we do not show you how to send multiple e-mails at once, as we do not supply built-in e-mail addresses with Reports Builder. However, if you wanted to send the report to various e-mail destinations, you would need to create a recipient field in your data model. Then, in the "ex2" section of the `distribution.xml`, replace the placeholder text "`<DestinationEmailAddress>`" with "`&&lt;recipient&>`". You can then delete the first section of the `distribution.xml` file (marked "ex1").

---

---

7. Save the XML file to the same directory where you've saved `inventoryreport_dist_<your initials>.rdf`.

---

---

**Note:** It is not required that you save the XML file to the same directory where your RDF is located, as you can specify the destination of the XML file at runtime.

---

---

You have finished customizing the distribution XML file to send a single e-mail to corporate headquarters with all of the individual warehouse reports, and multiple e-mails with a single attachment each to the individual warehouses.

## 2.4 Run the report

Since this example is a Web report, you can distribute it from either the command line or using your Web browser. For both methods, the following parameters apply:

<YourPath> is where your RDF file is located and where your distribution.xml file is located. Your login ID and Reports server name is your login information for the sample schema you've used with the sample RDF.

### From the command line

Type the following text in the command line:

```
RWRUN REPORT=<YourPath>/inventoryreport_dist_<YourInitials> USERID=<Your Login ID> SERVER=<Your Server Name> DISTRIBUTE=YES DESTINATION=<YourPath>/distribution.xml
```

### Using a URL

Type the following text in the **Location** field of your browser:

```
../RWSRVLET?REPORT=<YourPath>/inventoryreport_dist_<YourInitials>&USERID=<Your Login ID>&SERVER=<Your Server Name>&DISTRIBUTE=YES&DESTINATION=<YourPath>/distribution.xml
```

Running the report creates a file for each warehouse based on the warehouse ID in the specified directory. When the report is distributed, a single e-mail is sent to one address with all of these files attached to the e-mail.

If you set up the distribution.xml file for multiple e-mail addresses, each warehouse (or each e-mail address) would be sent a single e-mail with a single attachment file that includes the report for that warehouse.

## 2.5 Summary

Congratulations! You have distributed a report. You now know how to:

- Modify the layout of an existing report to burst on a section in your layout
- Distribute a report using e-mail by modifying the distribution.xml file

---

---

## Building a Paper Report with Ref Cursors

Reports Builder enables you to easily manage your queries by use of ref cursors. By using a ref cursor, which is a PL/SQL cursor datatype, you can reference a cursor from within a PL/SQL query. For example, if you already have numerous queries built and you want to reuse those queries in your reports, you can simply use a ref cursor in your report data model to access those queries.

In this chapter, you will learn how to use Reports Builder's features for using ref cursors. To build this paper report, you will use the Data Model view to create a multi-query data model, and then use the Report Wizard to create the report layout. You will make fairly extensive manual refinements in the Data Model view.

### About ref cursor queries

A ref cursor is a PL/SQL datatype that you can use in a query to fetch data. Each ref cursor query is associated with a PL/SQL function that returns a strongly typed ref cursor. The PL/SQL function must ensure that the ref cursor is opened and associated with a SELECT statement that has a SELECT list that matches the ref cursor type. You base a query on a ref cursor when you want to:

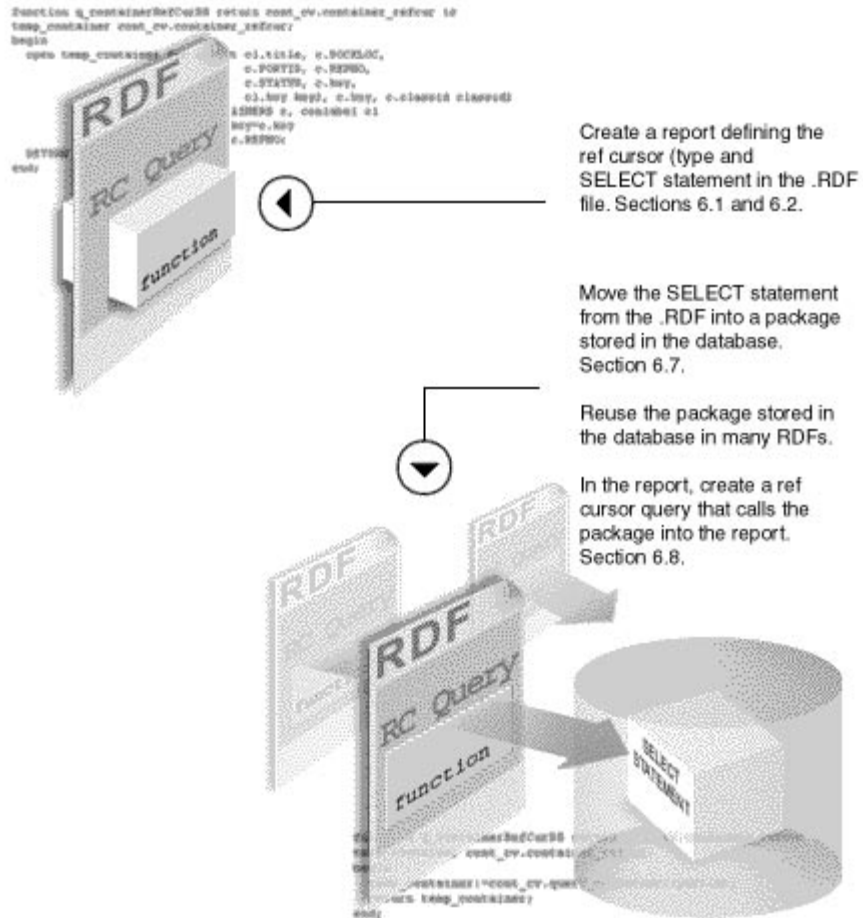
- more easily administer SQL
- avoid the use of lexical parameters in your reports
- share data sources with other applications
- increase control and security
- encapsulate logic within a subprogram

Furthermore, if you use a stored program unit to implement ref cursors, you receive the added benefits that go along with storing program units in the Oracle database.

The following figure shows that you create a report with the SELECT statement in the ref cursor query of the report. It also shows that you can store the SELECT

statement in a package in the database. Then, from the report, you can call the package from the database allowing you to reuse the package in many reports.

**Figure 3-1 Overview of the Ref Cursor Example**



### Example Scenario

In this example, you will create a detailed report showing information about employees and the job position they hold in each department.



**Table 3–1 Features demonstrated in this Ref Cursor Example**

| <b>Feature</b>                                                                                                                | <b>Location</b>                                           |
|-------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------|
| Create package specs that define ref cursors.                                                                                 | Section 3.2, "Defining a ref cursor type"                 |
| Create ref cursor queries that will use the ref cursors.                                                                      | Section 3.3, "Creating a ref cursor query"                |
| Rename objects in the data model so that they have more meaningful names.                                                     | Section 3.4, "Refining the data model"                    |
| Create group-to-group data links between ref cursor queries to create relationships between them.                             | Section 3.5, "Creating links between ref cursor queries"  |
| Create summaries that better describe the data.                                                                               | Section 3.6, "Adding summary columns"                     |
| Use the Report Wizard to create a report layout.                                                                              | Section 3.7, "Creating a layout"                          |
| Move the SELECT statements used by the ref cursor queries from the report and into packages that define the ref cursor types. | Section 3.8, "Moving the SELECT statement into a package" |
| Move the packages into a PL/SQL library so that other reports can share the code.                                             | Section 3.9, "Moving the packages into a library"         |

## 3.1 Prerequisites for this example

To build the examples in this manual, you must have the example files we've provided. If you haven't already done so, you can download the files you'll need to complete this example from the Oracle Technology network and install them on your machine.

### To download and install the example files:

1. Go to the Oracle Technology Network Web site (<http://otn.oracle.com/product/reports/>).
2. Click **Getting Started with Oracle9i Reports**.
3. Click **Index**, then find the "Building a Paper Report with Ref Cursors" example.
4. Download the file `refcursor.zip` into a temporary directory on your machine (e.g., "d:\temp").
5. Unzip the contents of the file, maintaining the directory structure, into an examples directory on your machine (e.g., `d:\orawin90\examples`).

This zip file contains the following files:

**Table 3–2 Files necessary for building this sample report using ref cursors**

| File                                                 | Description                                                                                                                                                                                         |
|------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Examples\RefCursor\result\ref_emp*.rdf</i>        | The different stages of the RDF. You can refer to these files as you complete each section of this chapter. The file <i>ref_emp68.rdf</i> is the final report.                                      |
| <i>Examples\RefCursor\scripts\refcursor_code.txt</i> | The PL/SQL code you will write in this chapter. You can use this file as a reference point to make sure your code is accurate, or you can simply cut and paste from this file into Reports Builder. |

### 3.1.1 Access to the sample Human Resources schema

If you don't know if you have access to the sample Human Resources schema provided with the Oracle9i database, contact your database administrator.

## 3.2 Defining a ref cursor type

To create a ref cursor query, you first create a package spec that defines the ref cursor. Then you create a query that uses the ref cursor. The steps in this section will help you create package specs that define ref cursors.

### To define a ref cursor type:

1. Open Reports Builder. If the Welcome dialog box appears, click **Build a new report manually** and click **OK**. If not, choose **File > New > Report**. Click **Build a new report manually**, and click **OK**.
2. In the Object Navigator, click the **Program Units** node under your UNTITLED report node.
3. Click the **Create** icon in the Object Navigator toolbar to add a program unit.
4. In the New Program Unit dialog box, type `concl_cv` as the name of the program unit.
5. Click **Package Spec**, then click **OK**.
6. Type the following package spec definition in the editor (existing text is in bold):

```
PACKAGE concl_cv IS
 type conclass_rec is RECORD
```

```

 (EMPLOYEE_ID NUMBER(6),
 FIRST_NAME VARCHAR2(20),
 LAST_NAME VARCHAR2(25),
 EMAIL VARCHAR2(25),
 PHONE_NUMBER (VARCHAR2(20),
 HIRE_DATE DATE,
 JOB_ID VARCHAR2(10),
 SALARY NUMBER(8,2)
 DEPARTMENT_ID NUMBER(4));
type conclass_refcur is REF CURSOR return conclass_rec;
END;

```

This package spec does two things:

- Defines a record (`conclass_rec`) that describes the data you want to select from the database.
- Defines a ref cursor that returns the data in the format described by the record.

---



---

**Note:** You can open the file *Examples/RefCursor/scripts/refcursor\_code.txt* to copy and paste the code into Reports Builder.

---



---

7. Click **Compile**.
8. If any compilation errors occur, check the code for syntax errors and recompile as needed.
9. Click **Close**.
10. Repeat steps 2 through 8 to create two more package specs with the following characteristics (existing text is in bold):

- **Package Spec Name: `cont_cv`**

```

PACKAGE cont_cv IS
 type container_rec is RECORD
 (EMPLOYEE_ID NUMBER(6),
 START_DATE DATE,
 END_DATE DATE,
 JOB_ID VARCHAR2(10),
 DEPARTMENT_ID NUMBER(4));
 type container_refcur is REF CURSOR return container_rec;

```

```
END;
```

- **Package Spec Name: port\_cv**

```
PACKAGE port_cv IS
 type portdesc_rec is RECORD
 (DEPARTMENT_ID NUMBER(4),
 DEPARTMENT_NAME VARCHAR2(30));
 type portdesc_refcur is REF CURSOR return portdesc_rec;
END;
```

---

---

**Note:** You can open the file *Examples/RefCursor/scripts/refcursor\_code.txt* to copy and paste the code into Reports Builder.

---

---

11. Choose **File > Save As**. Save the report in the directory of your choice, and name the report `ref61_<your initials>.rdf`.

---

---

**Note:** It is good practice when you are designing your report to save it frequently under a different file name. If you generate an error or if you don't like some of the changes you made, you easily can go back to the previously saved file and make revisions from that point.

---

---

### 3.3 Creating a ref cursor query

After creating package specs that define the ref cursors, you are ready to define the queries, as described in this section.

**To create a ref cursor query:**

1. In the Object Navigator, double-click the **Data Model** node to go to the Data Model view.
2. Click the **PL/SQL** icon.
3. Click in the main area (canvas region) of the Data Model view.
4. Drag your mouse in the canvas region to display the Program Unit Editor.
5. In the Program Unit Editor, type in the bold code that follows to define the function. New code is displayed in bold:

```
function q_portdescRefCurDS return port_cv.portdesc_refcur is
temp_portdesc port_cv.portdesc_refcur;
begin
 open temp_portdesc for select department_id, department_name from depart-
ments;
 return temp_portdesc;
end;
```

---



---

**Note:** You can open the file *Examples/RefCursor/scripts/refcursor\_code.txt* to copy and paste the code into Reports Builder.

---



---

6. Click **Compile**.
7. If any compilation errors occur, check the code for syntax errors and recompile as needed.
8. Click **Close**. The data objects display in the Data Model view.
9. In the Data Model view, click the ref cursor query object (QR\_1), then choose **Tools > Property Inspector**.
10. Under the **General Information** node, change the **Name** property to `q_portdesc`.

**Tip:** It is usually a good idea to give objects meaningful names, particularly when building a report with many objects. Later when building the layout, it is helpful to have queries and groups with meaningful names.

11. Press ENTER or RETURN, or click any other field in the Property Inspector to accept the change.
12. Close the Property Inspector.
13. Repeat steps 2 through 11 to create two more queries with the following characteristics. Be sure to rename the queries using the Property Inspector after creating them. New code is displayed in bold:
  - **Query name: q\_container**

```
function q_containerRefCurDS return cont_cv.container_refcur is
temp_container cont_cv.container_refcur;
```

```
begin
 open temp_container for
 select employee_id,
 start_date,
 end_date,
 job_id,
 department_id
 from job_history;
 return temp_container;
end;
```

■ **Query name: q\_conclass**

```
function q_conclassRefCurDS return concl_cv.conclass_refcur is
temp_concl concl_cv.conclass_refcur;
begin
 open temp_concl for
 select employee_id,
 first_name,
 last_name,
 email,
 phone_number,
 hire_date,
 job_id,
 salary,
 department_id
 from employees;
 return temp_concl;
end;
```

---

---

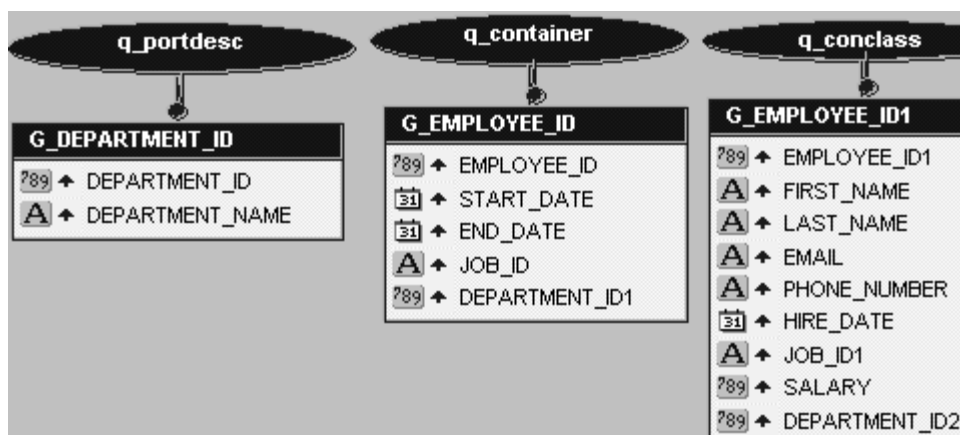
**Note:** You can open the file *Examples/RefCursor/scripts/refcursor\_code.txt* to copy and paste the code into Reports Builder.

---

---

The Data Model should look similar to the following figure:

Figure 3–2 Data Model with Three Queries



14. Save the report as `ref_62_<your initials>.rdf`.

---

**Note:** You can open the file `Examples/RefCursor/result/ref_emp62.rdf` and display the Data Model to compare your results.

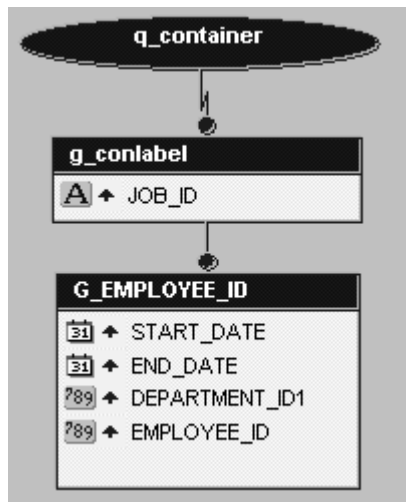
---

## 3.4 Refining the data model

In this section, you will rename some of the objects in the data model so that they have more meaningful names. You will also create a break group.

### To refine the data model:

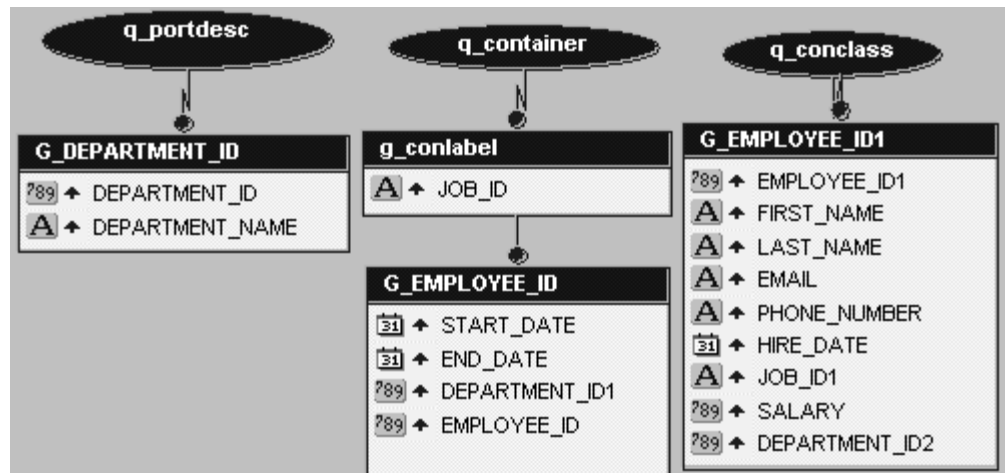
1. In the Data Model view, drag the title bar of the group `G_EMPLOYEE_ID` down a few inches to move the entire group.
2. Click and drag the column named `JOB_ID` out of and above `G_EMPLOYEE_ID` to create a new break group, as shown in the following figure:

**Figure 3–3 Query with Group**

3. Click the title bar of the new group that contains JOB\_ID, and choose **Tools > Property Inspector**.
4. Under the **General Information** node, change the Name property to G\_conlabel.
5. Press ENTER or RETURN, or click any other field in the Property Inspector to accept the change.
6. Close the Property Inspector.
7. In the Data Model view, your data model should look similar to the following figure:



Figure 3–4 Data Model with Group




---

**Note:** You can open the file *Examples/RefCursor/result/ref\_emp63.rdf* and display the Data Model to compare your results.

---

8. Save the report as `ref_63_<yourinitials>.rdf`.

### 3.5 Creating links between ref cursor queries

Currently, the queries that you have created are unrelated. To create relationships between them, you need to create group-to-group data links. The steps in this section will help you create the links.

#### To create links between ref cursor queries:

1. In the Data Model view, click the **Data Link** icon.
2. Click the title bar of `G_DEPARTMENT_ID`, and drag to the title bar of `G_EMPLOYEE_ID`.
3. Double-click `q_container`. The Program Unit Editor displays.

4. Now, you will append code to the WHERE clause of the SELECT statement to specify which columns are being used as primary and foreign keys.

After from job\_history, add the following code:

```
where :department_id = department_id;
```

Be sure that the semicolon ( ; ) now follows the WHERE clause.

Note that :department\_id is a bind variable referring to the DEPARTMENT\_ID in G\_DEPARTMENT\_ID.

5. Click **Compile**.
6. If any compilation errors occur, check the code for syntax errors and recompile as needed.
7. Click **Close**.
8. Click the **Data Link** icon.
9. Click the title bar of G\_EMPLOYEE\_ID and drag to the title bar of G\_EMPLOYEE\_ID1.
10. Double-click **q\_conclass**.
11. Now you will add a WHERE clause to the SELECT statement. Insert your cursor between FROM EMPLOYEES and the semicolon (;), and press ENTER or RETURN to create a new line.
12. Add the following code:

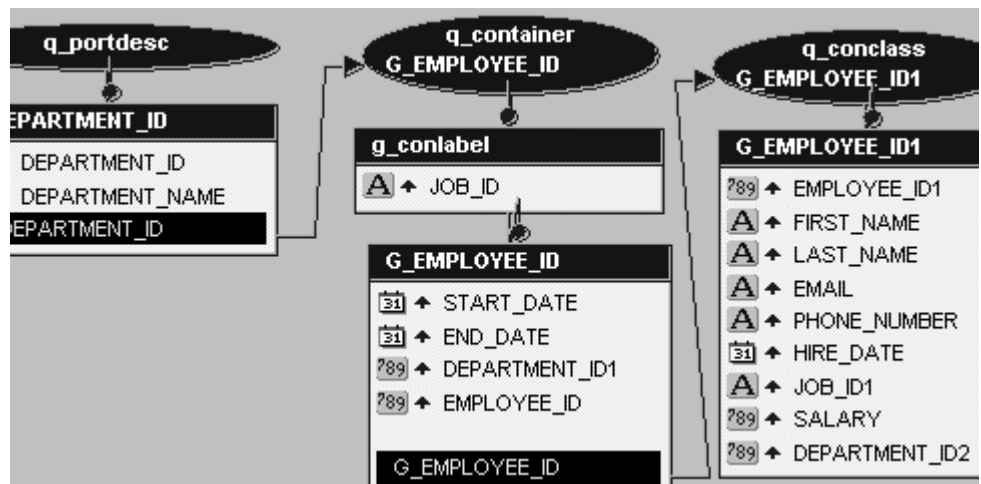
```
where :employee_id = employee_id;
```

Be sure that the semicolon ( ; ) now follows the WHERE clause.

Note that :employee\_id is a bind variable referring to the EMPLOYEE\_ID column in G\_employee\_id.

13. Click **Compile**.
14. If any compilation errors occur, check the code for syntax errors and recompile as needed.
15. Click **Close**.
16. Your data model should look similar to the following figure

Figure 3–5 Data Model with Links




---

**Note:** You can the file *Examples/RefCursor/result/ref\_emp64.rdf* and display the Data Model to compare your results.

---

17. Save the report as `ref_64_<your initials>.rdf`.

## 3.6 Adding summary columns

Now that your queries are complete and linked, the steps in this section will help you to create columns to summarize the data.

### To add summary columns:

1. In the Data Model view, click the **Summary Column** icon.
2. Click inside the `G_EMPLOYEE_ID` group. This creates a new column, `CS_1`.
3. Double-click the newly created column to open the Property Inspector.
4. Under the **General Information** node, change the Name property to `CS_classcount`.
5. Under the **Summary** node, change the following settings:

**Table 3–3 Summary Column Settings**

| Property | Setting         |
|----------|-----------------|
| Function | Count           |
| Source   | employee_id     |
| Reset At | G_department_id |

6. Click any other field in the Property Inspector to accept the changes.
7. Close the Property Inspector.

You have now created a summary that counts up the number of employees. You will not use the summary in this report’s layout, but you will use it as the source for other, more interesting summaries later.

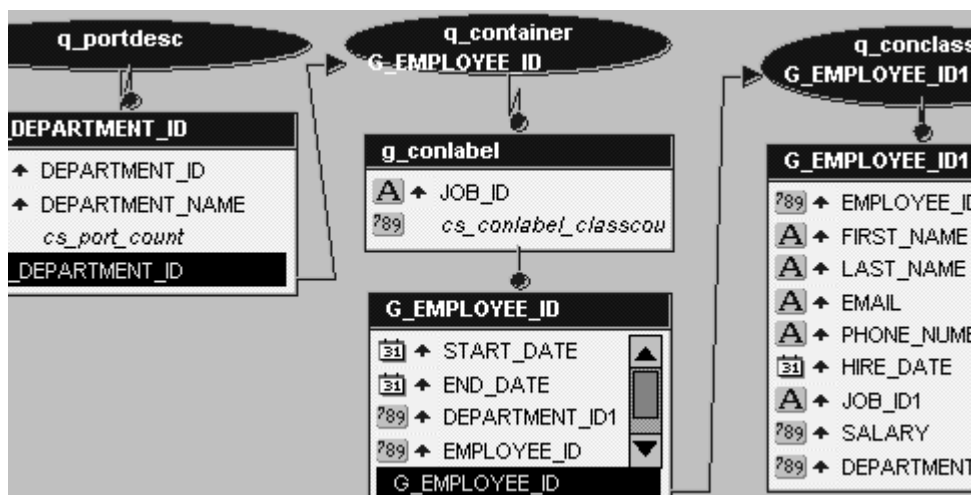
8. Repeat steps 1 through 5 to create summaries with the following characteristics:

| Create in Group | Name                   | Function | Source                 | Reset At        |
|-----------------|------------------------|----------|------------------------|-----------------|
| G_conlabel      | CS_conlabel_classcount | Sum      | CS_classcount          | G_conlabel      |
| G_department    | CS_port_count          | Sum      | CS_conlabel_classcount | G_DEPARTMENT_ID |

You may not understand these summaries now. Their purpose will become clearer when you create the report layout and preview the live data.

Your data model should look similar to the following figure:

Figure 3–6 Data model with Summary Columns




---

**Note:** You can also compare your results to the file we've provided, called `ref_emp65.rdf`.

---

9. Save the report as `ref_65_<your initials>.rdf`.

## 3.7 Creating a layout

Now that you have a working data model, the steps in this section will help you to create a layout.

### To create a paper layout:

1. In the Object Navigator, right-click the report name and choose **Report Wizard**.
2. On the Report Type page, select the **Create Paper Layout Only** radio button, then click **Next**.
3. On the Style page, type `My Employees` as the Title.
4. Click **Group Above** as the report style.
5. Click **Next**.
6. On the Groups page, click `G_conlabel` and click **Down**.

7. Repeat step 5 for:
  - G\_DEPARTMENT\_ID
  - G\_EMPLOYEE\_ID
8. Click **Next**.
9. On the Fields page, click DEPARTMENT\_NAME, and click the right arrow.
10. Repeat step 8 for:
  - DEPARTMENT\_NAME
  - EMPLOYEE\_ID
  - START\_DATE
  - END\_DATE
  - JOB\_ID
  - CS\_port\_count
11. Click **Next**.

12. On the Labels page, type in the labels and widths as shown in the following table:

| <b>Column</b>   | <b>Label</b> | <b>Width</b> |
|-----------------|--------------|--------------|
| DEPARTMENT_NAME | Department   | 10           |
| EMPLOYEE_ID     | Employee ID  | 8            |
| START_DATE      | Start Date   | 9            |
| END_DATE        | End Date     | 9            |
| JOB_ID          | Job ID       | 10           |
| CS_port_count   | Total:       | 12           |

13. Click **Next**.
14. On the Template page, click **Predefined template**, and choose **Cyan Grid Landscape**.
15. Click **Finish**. The report automatically displays in the Paper Design view.

**Figure 3–7 Paper Design View of the Ref Cursor Report**

| YOUR Inc.<br>COMPANY      |            | Employment History |           |
|---------------------------|------------|--------------------|-----------|
| Department Administration |            |                    |           |
| Total:                    |            |                    |           |
| Department Marketing      |            |                    |           |
| Total:                    |            |                    |           |
| Department Purchasing     |            |                    |           |
| Job #                     | PU_CLERK   | Total:             | 7         |
| Start Date                | 08-MAR-97  | End Date           | 01-JUL-99 |
| Employee #                | First Name | Last Name          |           |
| 14                        | Den        | Raphaely           |           |
| Start Date                | 01-OCT-96  | End Date           | 11-AUG-98 |
| Employee #                | First Name | Last Name          |           |
| 45                        | John       | Russell            |           |
| Start Date                | 19-AUG-93  | End Date           | 17-MAY-96 |
| Employee #                | First Name | Last Name          |           |
| 03                        | Alexander  | Hunold             |           |

---

**Note:** You can open the file *Examples/RefCursor/result/ref\_emp66.rdf* and display the Paper Design view to compare your results.

---

16. Save the report as `ref_66_<your initials>.rdf`.

### 3.8 Moving the SELECT statement into a package

In your current report configuration, the SELECT statements used by the ref cursor queries reside within the report itself. In many cases, it is advantageous to have SELECT statements reside in the packages that define the ref cursor types. Then, you can simply reference the packages, rather than typing the same SELECT statement directly into every report that uses it. If you need to change the SELECT



statement (for example, to modify or add clauses), you simply update it once in the package, rather than in every report that uses it.

The steps in this section will help you to move the SELECT statements to the packages that define the ref cursor types.

**To move the SELECT statement into a package:**

1. In the Object Navigator, click the **Program Units** node for your report.
2. Click the **Create** icon to add a program unit.
3. In the New Program Unit dialog box, type `cont_cv` as the name of the program unit.
4. Click **Package Body**, and click **OK**.
5. Type the following code in the editor.

```
PACKAGE BODY cont_cv IS
 function query_container (p_department_id number) return container_
 refcur is tempcv_container cont_cv.container_refcur;
begin
 open tempcv_container for
 select employee_id,
 start_date,
 end_date,
 ob_id,
 department_id
 from job_history
 where :department_id=department_id;
 return tempcv_container;
end;
END;
```

---

---

**Note:** You can open the file *Examples/RefCursor/scripts/refcursor\_code.txt* to copy and paste the code into Reports Builder.

---

---

6. Click **Compile**.
7. If any compilation errors occur, check the code for syntax errors and recompile as needed.
8. Click **Close**.

9. Now that the function is defined, you must add it to the package spec so that it can be referenced. Other program units will know about the function in the package body only if it is described in the package spec.

In the Object Navigator, double-click the **CONT\_CV(Package Spec)** object.

10. In the Program Unit editor, type the following line above the `END;` statement:

```
function query_container (p_department_id number) return container_refcur;
```

11. Click **Close**.

12. Choose **Program > Compile > All**.

13. Click **OK** when done.

14. In the Object Navigator, double-click the **Q\_CONTAINERREFCURDS** object under the **Program Units** object.

15. Edit the code to look as follows:

```
function Q_containerRefCurDS return cont_cv.container_refcur is
 temp_container cont_cv.container_refcur;
begin
 temp_container:=cont_cv.query_container (:department_id);
 return temp_container;
end;
```

When you are done, all of the query's logic will reside in the function named `query_container`. From now on, when you change `query_container`, you will change this and any other queries that reference it.

---

---

**Note:** You can open the file `Examples/RefCursor/scripts/refcursor_code.txt` to copy and paste the code into Reports Builder.

---

---

16. Click **Compile**.
17. If any compilation errors occur, check the code for syntax errors and recompile as needed.
18. Click **Close**.
19. Click the **Paper Design** icon to view the report in the Paper Design view.
20. Save the report as `ref_67_<your initials>.rdf`.

**Optional Exercise:**

Repeat steps 1 through 19 for the other two queries in the report.

## 3.9 Moving the packages into a library

If you have many reports that use these same ref cursor types and SELECT statements, you can move the program units that you created into a PL/SQL library stored in a file or the database, so that other reports can easily share the code. The steps in this section will help you to move the program units to a PL/SQL library.

**To move the packages into a library:**

1. In the Object Navigator, click the **PL/SQL Libraries** object.
2. Click the **Create** icon to add a new library.
3. Choose **File > Save As**.
4. Type `DEPT_CONTAINER` as the Library.
5. Click **File System**.
6. Click **OK**.
7. Drag and drop the following program units from your report to the **Program Units** node under the newly created `DEPT_CONTAINER` library:
  - `CONCL_CV(Package Spec)`
  - `CONT_CV(Package Spec)`
  - `CONT_CV(Package Body)`
  - `PORT_CV(Package Spec)`
8. Save `DEPT_CONTAINER`.
9. If the Paper Design view is open, close it.
10. In the Object Navigator, under the **Program Units** node of your report, delete `CONCL_CV(Package Spec)`, `CONT_CV(Package Spec)`, `CONT_CV(Package Body)`, and `PORT_CV(Package Spec)`.

---

---

**Note:** If the Paper Design view is open when you delete the packages from the report, you may get some errors.

---

---

11. Click the **Attached Libraries** node for your report.
12. Click the **Create** icon to add a new attached library.
13. In the Attach Library dialog box, click **File System**.
14. Click **Browse** to find the DEPT\_CONTAINER library. It will have a .PLL file extension. After you have found and selected DEPT\_CONTAINER, click **Open**.
15. Click **Attach**.
16. Choose **Program > Compile > All**.
17. Click **OK** to close the Compile window.
18. In the toolbar, click the **Paper Design** icon to view the report.

---

---

**Note:** If you get an error when you attempt to view the report, repeat steps 16 through 18.

---

---

19. Save the report as `ref_68_<your initials>.rdf`.

**Optional Exercise:**

Store the PL/SQL library in the database rather than in a file. Note that you will need “create” privileges on the database to complete this optional exercise.

## 3.10 Summary

Congratulations! You have finished the Ref Cursor Query sample report. You now know how to:

- Create package specs that define ref cursors.
- Create ref cursor queries.
- Create data links between ref cursor queries.
- Create summaries to describe data.
- Create a report layout.
- Move SELECT statements into packages.
- Move packages into a PL/SQL library.

For more information about using ref cursors, see the *Reports Builder online help*.

---

# Building a Report with an XML Pluggable Data Source

Reports Builder enables you to use any data source you wish. In this chapter, you will learn how to use the XML pluggable data source that is provided with Oracle9i Reports.

## About Pluggable Data Sources

The information you must publish is often derived from data in various corporate data sources. These data sources may be SQL-based (relational databases) or non-SQL-based, such as XML, OLAP, and the like. Often, you must combine data from one or more of these data sources to publish meaningful information. For example, you may need to combine data that exists in a relational database with data from a multi-dimensional database to compare trends and performance.

Oracle9i Reports enables you to leverage capabilities, such as aggregation, summarization, formatting, and scheduling, on data from any data source. You can leverage the PDS (pluggable data source) architecture to connect to your own data source, as well as to the data sources available with Oracle9i Reports (XML, JDBC, text, and Express).

For more information on pluggable data sources, refer to the *Reports Builder online help* and the Javadoc documentation for the PDS APIs.

## Example Scenario

Suppose you have an international business with warehouses in the United States and overseas. These warehouses are running a de-centralized management system that stores the operational data locally at each site. The inventory of the warehouses are managed by the local managers. However, for planning purposes, a team at corporate headquarters needs to access the inventory data (in SQL), including the

most recent data, of every warehouse. The warehouse data is only available as an XML stream. In this example, you will learn how to combine data from a local database (i.e., the warehouse data) and data from an XML feed to create a Web report.

In this example, you will use static XML files that we've provided for you. The report will access the XML feed online using the business-to-business interface of your order entry system.

**Table 4-1 Features Demonstrated in the XML PDS Example**

| <b>Feature</b>                                                      | <b>Location</b>                                                          |
|---------------------------------------------------------------------|--------------------------------------------------------------------------|
| Manually create a SQL query.                                        | Section 4.2.1, "Create a SQL query for your new report"                  |
| Use the Data Wizard to create an XML query.                         | Section 4.2.2, "Create an XML query to access your XML data source"      |
| Create a data link between a SQL query and an XML query.            | Section 4.2.3, "Create a data link between two queries"                  |
| Use the Report Wizard to create a layout for your report.           | Section 4.2.4, "Create a layout for your report using the Report Wizard" |
| Use format triggers and procedures to apply alternating row colors. | Section 4.2.5, "Apply alternating row colors to your report"             |
| Use a group filter to sort your XML data.                           | Section 4.2.6, "Filter your XML data using groups"                       |

## 4.1 Prerequisites for this example

To build the examples in this manual, you must have the example files we've provided and access to the sample schema that comes with the Oracle9i database.

### 4.1.1 Example files

If you haven't already done so, you can download the files you'll need to complete this example from the Oracle Technology network (<http://otn.oracle.com/products/reports/>) and install them on your machine.

**To download and install the example files:**

1. Go to the Oracle Technology Network Web site (<http://otn.oracle.com/product/reports/>).
2. Click **Getting Started with Oracle9i Reports**.
3. Click **Index**, then find the "Building a Report using an XML Pluggable Data Source" example.
4. Download the file XML\_PDS.zip into a temporary directory on your machine (e.g., "d:\temp").
5. Unzip the contents of the file, maintaining the directory structure, into an examples directory on your machine (e.g., d:\orawin90\examples).

This zip file contains the following files:

**Table 4–2 Files necessary for building this sample report using XML PDS**

| File                                                    | Description                                                                                                               |
|---------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|
| <i>Examples\XML_PDS\result\inventory_report.pdf</i>     | The final PDF version of the paper report.                                                                                |
| <i>Examples\XML_PDS\result\inventory_report.rdf</i>     | The final RDF version of the paper report.                                                                                |
| <i>Examples\XML_PDS\result\warehouse_inventory.rdf</i>  | A version of the final report which we've enhanced and formatted.                                                         |
| <i>Examples\XML_PDS\source\inventory_report_NB.rdf</i>  | The source file for the report. Running this RDF in Reports Builder will display a paper report in the Paper Design view. |
| <i>Examples\XML_PDS\scripts\XMLPDS_SQL.txt</i>          | The various SQL statements you will use in this report.                                                                   |
| <i>Examples\XML_PDS\scripts\warehouse_inventory.xml</i> | The XML data source for the query in your report.                                                                         |
| <i>Examples\XML_PDS\scripts\warehouse_inventory.xsd</i> | The XML data stream for your report.                                                                                      |

---

**Note:** The index.html file and assets directory are used as part of the **Getting Started with Oracle9i Reports** Web site. Please do not delete or move these files

---

## 4.1.2 Access to the sample schema

If you don't know if you have access to the sample schema provided with the Oracle9i database, contact your database administrator. You should have access to the "Order Entry" portion of the schema to complete this example. Typically, you can log into this schema by using the user ID and password "oe/oe", then enter the name of the database.

## 4.2 Create a report manually with SQL and XML queries

When you create a report, you can either use the Report Wizard to assist you or create the report yourself. To build this report, you'll need to create two queries: a SQL query and an XML query.

### 4.2.1 Create a SQL query for your new report

When creating the SQL query, you'll need access to the Order Entry part of the sample schema provided with the Oracle9i database. If you don't have access, contact your database administrator. Typically, you can log in using the connection string "oe/oe@<database name>".

#### To create a SQL query:

1. In Reports Builder, choose **File > New > Report**.
2. In the New Report dialog box, select the **Build a new report manually** radio button, then click **OK**.

Your new report displays in the Object Navigator as something like "MODULE 2." You will also see the Data Model view of your new report.

3. In the Data Model view, click the **SQL Query** icon in the tool palette.
4. Draw an area on the canvas to create a query.
5. In the **SQL Statement** field, type the following code:

```
select W.WAREHOUSE_ID,
 W.WAREHOUSE_NAME,
 L.CITY,
 L.STATE_PROVINCE,
 C.COUNTRY_NAME
from WAREHOUSES W
 HR.LOCATIONS L
 HR.COUNTRIES C
```



```

where (W.LOCATION_ID = L.LOCATION_ID(+))
and (L.COUNTRY_ID = C.COUNTRY_ID(+))
order by C.COUNTRY_NAME, W.WAREHOUSE_NAME

```

---

**Note:** You can also copy and paste the code from the text file we've provided, `xmlpds_sql.txt`. Open the file in a text editor, then copy the List of Warehouse query) into the SQL Statement text box.

---

6. Click **OK**.

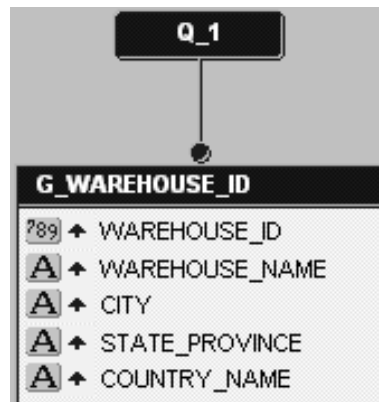
---

**Note:** If the Connect dialog box displays, enter the user ID, password, and name of the database that contains the sample schema.

---

The data model displays in the Data Model view, and should look something like this

**Figure 4–1** Data Model for the XML PDS Example SQL Query



7. Save your report as `inventoryreport_xml_<your initials>.rdf`. You have created a SQL query to retrieve the data for your report.

## 4.2.2 Create an XML query to access your XML data source

In this section, you will create a query to access the XML data source. You can view the resulting report we've provided to make sure your query is correct. Please note, though, that you must update the paths to the Data Definition files with the location of the example files we provided to you.

### To create an XML query:

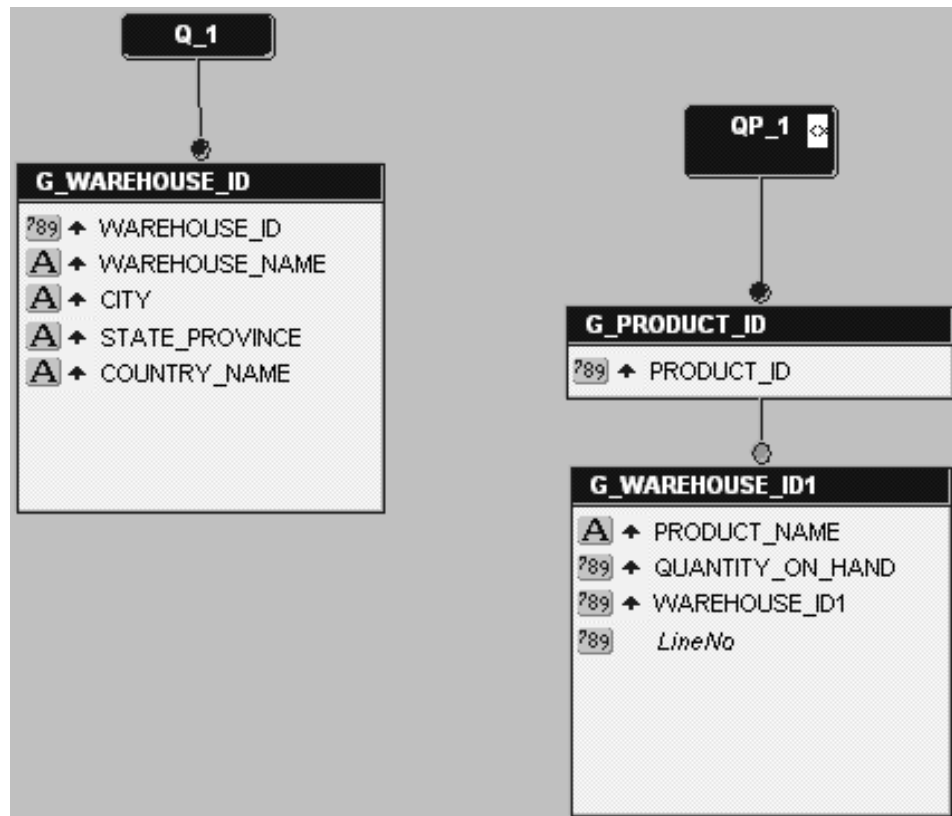
1. In the Data Model view, create another query by choosing **Insert > Query**.
2. When the Data Wizard displays, click **Next** on the Welcome page.
3. On the Query Name page, click **Next**.
4. On the next page, click **XML Query**, then click **Next**.
5. On the Data Source Definition page, click **Query Definition**.
6. In the Define XML Query dialog box, under Data Definition, click **Browse** to locate the XSD file we've provided, `warehouse_inventory.xsd`.
7. When you've located the file, choose it, then click **Open**.
8. In the Define XML Query dialog box, under Data Source, click **Browse** to locate the XML file we've provided that contains your data, `warehouse_inventory.xml`.

If you want to compare your data definition to the one we provided, make sure that you replace the data definition locations with the locations of your files.

9. When you've located the file, choose it, then click **Open**.
10. In the Define XML Query dialog box, click **OK**.
11. In the Data Wizard, click **Next**.
12. On the Groups page, click the **PRODUCT\_ID** group, then click the right arrow button.
13. Click **Finish**.

The Data Model view displays your two queries, and should look something like this:

**Figure 4–2 Data Model for the XML PDS Example with XML and SQL Queries**



14. Save your report as `inventoryreport_xml_<your initials>.rdf`.  
You have created an XML query to access the XML data source we've provided.

### 4.2.3 Create a data link between two queries

You will now need to link the SQL query and the XML query so that you can access your corporate data as well as the data for each of the local warehouses.

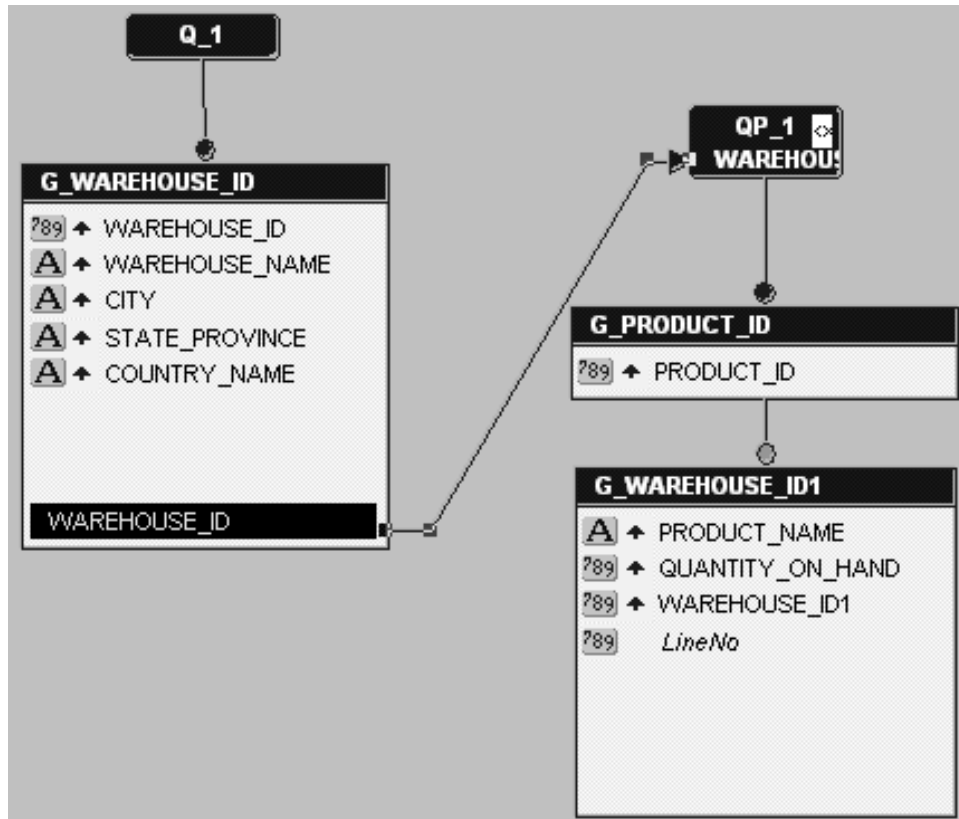
#### To create a data link:

1. In the Data Model view for your report, click the **Data Link** icon in the tool palette.

2. Click the **WAREHOUSE\_ID** column in your first query (called Q\_1).
3. Drag your cursor until your cursor is over the **WAREHOUSE\_ID1** column in the second query (called Q\_2).

Your data model should now look something like this:

**Figure 4-3** Data Model with a Data Link between a SQL Query and an XML Query



You'll notice that the **WAREHOUSE\_ID** column is now highlighted at the bottom of Q\_1, with a line pointing to the **WAREHOUSE\_ID1** column. You can close the Property Inspector for the link.

4. Save your report as `inventoryreport_xml_<your initials>.rdf`.

You have created a data link between the **WAREHOUSE\_ID** columns in the two queries.

## 4.2.4 Create a layout for your report using the Report Wizard


Before you can run any report, you must define a layout. The easiest way to do this is to use the Report Wizard.

### To create a paper layout:

1. In the Data Model view for your report, right-click the canvas, then choose **Report Wizard** from the menu.
2. On the Report Type page, select the **Create Paper Layout only** radio button, then click **Next**.
3. On the Style page, select the **Group Above** radio button.
4. On the Groups page, make sure the **G\_WAREHOUSE\_ID** and **G\_WAREHOUSE\_ID1** groups are listed in the Displayed Groups list as going Down.
5. On the Fields page, click the double arrows to display all the fields.
6. On the Templates page, click **Blue**, then click **Finish**.

Your report displays in the Paper Design view, and should look something like this:

**Figure 4–4 Paper Design View of your XML PDS Example Report**



| Warehouse Id | Warehouse Name | Sydney          |
|--------------|----------------|-----------------|
| State        | Province       | New South Wales |
|              | Country        | Name            |
|              | Austral        |                 |
| 6            | 1820           | 69              |
| 6            | 2414           | 59              |
| 6            | 2417           | 29              |
| 6            | 2395           | 56              |
| 6            | 2396           | 57              |
| 6            | 2371           | 71              |
| 6            | 2492           | 41              |
| 6            | 2270           | 64              |
| 6            | 1742           | 31              |
| 6            | 2596           | 51              |
| 6            | 2430           | 65              |
| 6            | 2319           | 44              |

7. Save your report as `inventoryreport_xml_<your initials>.rdf`.

You have created the layout for your paper report.

## 4.2.5 Apply alternating row colors to your report

Now that you've created the report, you can make it more user-friendly by using a summary column to apply alternating row colors.

### 4.2.5.1 Create a summary column to count the rows

**To create a summary column:**

1. In the Data Model view of your report, click the **Summary Column** icon in the tool palette.

If you are still in the Paper Design view, you can click the **Data Model** icon in the tool palette to display the Data Model view.

2. Click in the XML query group (G\_WAREHOUSE\_ID1) to create a summary column.
3. In the Property Inspector, name your new summary column `LineNo`.
4. Make sure the Column Type property is set to **Summary**.
5. From the Datatype property drop-down box, choose **Number**.
6. Under Summary, from the Function property drop-down box, choose **Count**.
7. From the Source property drop-down box, choose **PRODUCT\_NAME**.
8. Close the Property Inspector for the new summary column.

#### 4.2.5.2 Create a procedure that changes the line colors

##### To create a procedure:

1. In the Object Navigator, under your report name, click the **Program Units** node, then click the **Create** icon.
2. In the **Name** field, type `linecolors`.
3. Under Types, make sure the **Procedure** radio button is selected, then click **OK**.
4. Type the following code in the editor (existing text is in bold);

```
PROCEDURE LineColors IS
BEGIN
 if (:LineNo mod 2 = 0)
 then
 srw.set_text_color('TextColor');
 else srw.set_text_color('black');
 end if;
END;
```

---

---

**Note:** You can copy and paste this code from the procedure provided in the `xmlpds_sql.txt` file. Just copy the text under Line Colors Procedure.

---

---

5. Click **Compile** to compile the procedure.

If any errors display, make sure the code is correct, and that you created the summary column in Section 4.2.5.1, "Create a summary column to count the rows".

6. Click **Close**.

### 4.2.5.3 Create a format trigger for each field that calls the procedure

#### To create a format trigger:

1. In the Object Navigator, under your report name, expand the **Paper Layout** node and navigate to: Body/M\_G\_WAREHOUSE\_ID\_GRPFR/R\_G\_WAREHOUSE\_ID/M\_G\_WAREHOUSE\_ID1\_GRPFR/R\_G\_WAREHOUSEID1.
2. Select the first field, **F\_PRODUCT\_ID**, and open the Property Inspector.

---

---

**Note:** If you can't find a particular field, use the **Find** field at the top of the Object Navigator.

---

---

3. Under Advanced Layout, click the field next to the **Format Trigger** property.
4. In the resulting code editor, type the following code (existing text is in bold):

```
function F_PRODUCT_IDformatTrigger return Boolean is
begin
 LineColors;
 return (TRUE);
end;
```

5. Perform steps 3 through 5 for the three remaining fields:
    - **F\_PRODUCT\_NAME**
    - **F\_QUANTITY\_ON\_HAND**
    - **F\_WAREHOUSE\_ID1**
  6. Save your report.
  7. Run your report to paper by clicking the **Run Paper Layout** icon in the toolbar.
- You have now applied alternating row colors to your report.



## 4.2.6 Filter your XML data using groups

If you have a lot of data in your XML file, you might want to consider sorting and filtering it. You can do so by creating a group filter and a hierarchy.

### 4.2.6.1 Create a group filter on the repeating frame

**To create a group filter:**

1. In the Object Navigator, double-click the **Paper Layout** node.
2. Under the Paper Layout node, double-click the **Main Section** node.
3. Under the Main Section node, double-click the **Body** node.
4. Right-click the **R\_G\_WAREHOUSE\_ID** group and choose **Property Inspector** from the pop-up menu.

---



---

**Note:** If you can't find the repeating frame, use the **Find** field at the top of the Object Navigator.

---



---

5. In the Property Inspector, under Group, choose **PL/SQL** from the Filter Type property drop-down list.

The **PL/SQL Filter** property displays.

6. Next to the PL/SQL filter property, click the field to display the code editor.
7. Type the following code in the editor that displays (the existing text is in bold):

```
function G_WAREHOUSE_IDGroupFilter return boolean is
begin
 if (:LineNo mod 2 = 0)
 then
 srw.set_background_fill_color('TableCell');
 end if;
 if (:LineNo mod 2 = 1)
 then srw.set_background_fill_color('Totals');
 end if;
return (TRUE);
end;
```

8. Click the **Compile** button to compile the code.
9. If any errors display, check to make sure you copied the text exactly.

---

---

**Note:** If you are not familiar with compiling PL/SQL, refer to a PL/SQL reference manual.

---

---

10. When the code is compiled, click **Close**.
11. Save your report.

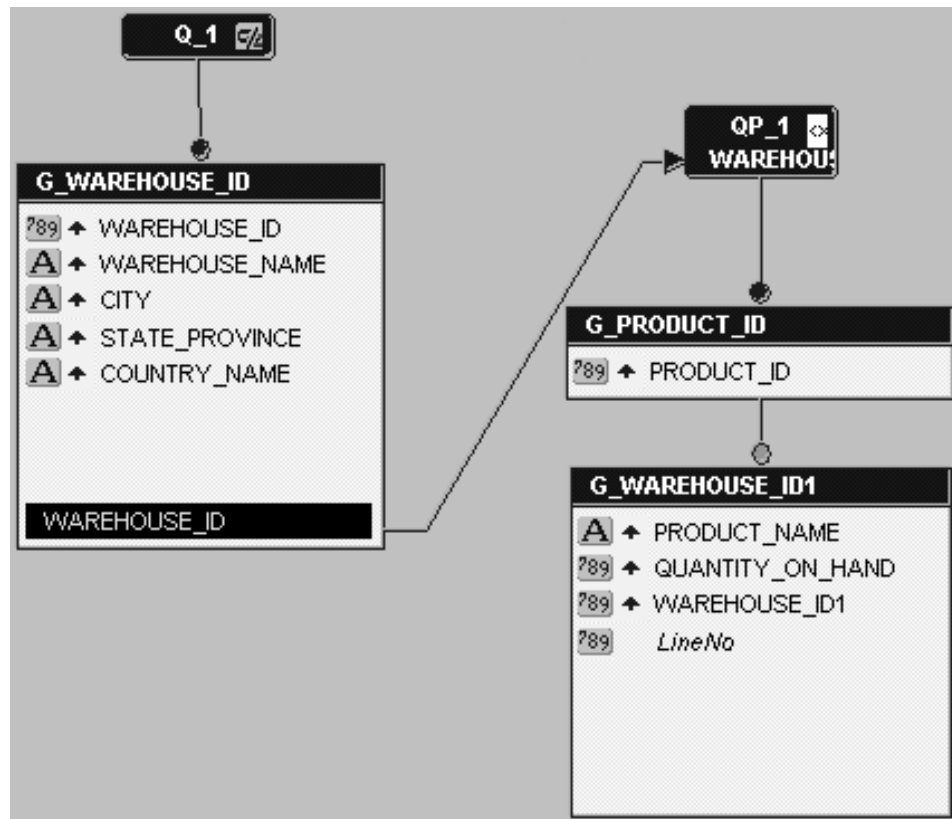
#### 4.2.6.2 Create a hierarchy for the XML query

**To create a hierarchy:**

1. In the Data Model view for your report, find the XML query.
2. In the XML query, click the **PRODUCT\_ID** column, and drag it between the query name and the group.

Your new data model should look something like this:

Figure 4–5 Data Model with Hierarchy




---

**Note:** By dragging the PRODUCT\_ID column above the rest of the query, you've created a hierarchy for the XML query.

---

3. Run your report to paper.

You can also run the RDF we've provided (*Examples\XML\_PDS\source\inventoryreport.rdf*) to view the results in Reports Builder.

4. Save your report as `inventoryreport_xml_<your initials>.rdf`.

You have now created a group filter that sorts your XML data.

## 4.3 Run your report to paper

### To run your paper report:

1. In the Object Navigator, make sure your report (inventoryreport\_xml\_<your initials>.rdf) is selected.
2. Click the **Run Paper Layout** icon to run your report to paper.
3. Your report displays in the Paper Design view, and should look something like this:

*Figure 4-6 Final Paper Design View of the XML PDS Example Report*



| Warehouse Name | Sydney            |                |
|----------------|-------------------|----------------|
| City           | Sydney            | State Province |
| Country        | Australia         |                |
| Product Id     | Product Name      |                |
| 1733           | PS 220V /UK       |                |
| 1734           | Cable RS232 10/A  |                |
| 1737           | Cable SCSI 10/FW  |                |
| 1738           | PS 110V /US       |                |
| 1739           | SDRAM - 128 MB    |                |
| 1740           | TD 12GB/DAT       |                |
| 1742           | CD-ROM 500/16x    |                |
| 1745           | Cable SCSI 20/WVD |                |
| 1748           | PS 220V /EUR      |                |
| 1749           | DIMM - 256MB      |                |
| 1750           | DIMM - 2GB        |                |
| 1755           | 32MB Cache /NM    |                |

## 4.4 Summary

Congratulations! You have successfully used an XML data source for a paper report. You now know how to:

- Create a SQL query from scratch
- Use the Data Wizard to create an XML query
- Create a data link between a SQL query and an XML query
- Create a layout for your report using the Report Wizard
- Apply alternating row colors to your report using format triggers and procedures
- Filter your XML data using a group filter and hierarchy

For more information on using XML as a data source, refer to *Getting Started with Oracle9i Reports* or the *Reports Builder online help*.



---

---

# Building a Report with a Text Pluggable Data Source

Reports Builder enables you to use any data source you wish. In this chapter, you will learn how to use character-delimited text as a data source.

## About Pluggable Data Sources

The information you must publish is often derived from data in various corporate data sources. These data sources may be SQL-based (relational databases) or non-SQL-based, such as XML, OLAP, and the like. Often, you must combine data from one or more of these data sources to publish meaningful information. For example, you may need to combine data that exists in a relational database with data from a multi-dimensional database to compare trends and performance.

Oracle9i Reports enables you to leverage capabilities, such as aggregation, summarization, formatting, and scheduling, on data from any data source. You can leverage the PDS (pluggable data source) architecture to connect to your own data source, as well as to the data sources available with Oracle9i Reports (XML, JDBC, text, and Express).

For more information on pluggable data sources, refer to the *Reports Builder online help* and the Javadoc documentation for the PDS APIs.

## Example Scenario

Suppose you downloaded the US Census Bureau data in CSV (comma-separated values) format and want to generate a report. In this example, you will create a paper report that queries this data.

**Table 5–1 Features demonstrated in this Text PDS example**

| Feature                                                                           | Location                                                |
|-----------------------------------------------------------------------------------|---------------------------------------------------------|
| Configure Reports Builder to recognize your text file as a pluggable data source. | Section 5.2, "Set up the textpds.conf file"             |
| Use the Report Wizard to create an paper report based on the text data source.    | Section 5.3, "Use the Report Wizard to create a report" |

## 5.1 Prerequisites for this example

To build the examples in this manual, you must have the example files we've provided. If you haven't already done so, you can download the files you'll need to complete this example from the Oracle Technology network (<http://otn.oracle.com/products/reports/>) and install them on your machine.

### To download and install the example files:

1. Go to the Oracle Technology Network Web site (<http://otn.oracle.com/product/reports/>).
2. Click **Getting Started with Oracle9i Reports**.
3. Click **Index**, then find the "Building a Report Using a Text Pluggable Data Source" example.
4. Download the file textpds.zip into a temporary directory on your machine (e.g., "d:\temp").
5. Unzip the contents of the file, maintaining the directory structure, into an examples directory on your machine (e.g., d:\orawin90\examples).

This zip file contains the following files:

**Table 5–2 Files necessary for building this Text PDS example**

| File                                            | Description                                                                 |
|-------------------------------------------------|-----------------------------------------------------------------------------|
| <i>Examples\TextPDS\result\censusreport.rdf</i> | The finished RDF for your report.                                           |
| <i>Examples\TextPDS\result\censusreport.pdf</i> | A PDF version of your finished report.                                      |
| <i>Examples\TextPDS\scripts\census_csv.txt</i>  | The character-delimited data downloaded from the US Census Bureau Web site. |
| <i>Examples\TextPDS\scripts\config.txt</i>      | Code for the TextPDS.conf file.                                             |



## 5.2 Set up the textpds.conf file

Before you can use a text file as your pluggable data source, you must set up the text PDS configuration file (textpds.conf) with the definition of the values in the text file. When you add this format information to the configuration file, Reports Builder can then recognize your entries as a valid format.

---



---

**Note:** You must edit your configuration file *before* you launch Reports Builder in order for the changes to take effect.

---



---

### To set up your textpds.conf:

1. In a text editor, such as UltraEdit, open the file `textpds.conf`, located in the `Reports_Home/reports/conf` directory.

---



---

**Note:** `Reports_Home` is where Reports Builder is installed on your computer.

---



---

2. In another text editor window, open the `config.txt` file we've provided in the `Examples/TextPDS/Scripts` directory.
3. Cut the text we've provided in `config.txt` and paste it into `textpds.conf` before the `</textPDS>` entry, so that the resulting `textpds.conf` file contains an entry like this:

**Figure 5–1** Snapshot of the `textpds.conf` Entry

```
<!--Data definition for Text PDS example -->
 <fileFormat name="CensusCSV" comment="#" deli
 <columnInfo>
 <column name="Category"
 <column name="Subject" type
 <column name="value" type
 <column name="percentage"
 </columnInfo>
 </fileFormat>

</textPDS>
```

---

---

**Note:** This entry enables Reports Builder to recognize a text file as a PDS. When you choose your PDS in the Report Wizard, the text file displays as an option. Here, you also define the properties of each column in the file

---

---

4. Save the `textpds.conf` file.

You have set up your `textpds.conf` file for the character-delimited text data source.

### 5.3 Use the Report Wizard to create a report

When you create a report, you can either use the Report Wizard to assist you or create the report yourself. To build the simple report in this example, you can use the Report Wizard.

Before you use a text pluggable data source, you might want to examine the text file first to see what it looks like. You can open the `census_csv.conf` file in a text editor, like UltraEdit or WordPad, to see the data we'll be using in this example.

#### To create a simple report:

1. Now that you've updated the `textpds.conf` file, launch **Reports Builder**.

**Tip:** If Reports Builder was already open when you modified the `textpds.conf` file, you should shut down Reports Builder and relaunch it.

2. In the New Report dialog box, select the **Use the Report Wizard** radio button, then click **OK**.
3. If the **Welcome** page displays, click **Next**.
4. On the Report Type page, select the **Create Paper Layout Only** radio button, then click **Next**.
5. On the Title page, type a title for your report, then select the **Group Above** radio button.
6. Click **Next**.
7. On the Data Source page, click **Text Query**, then click **Next**.
8. On the Data Source Definition page, click **Query Definition**.

If you get an error message and cannot display the Query Definition dialog box, check your configuration file (textpds.conf) to confirm the code you added.

9. Under Data Definition, choose the **CensusCSV** format from the drop-down list.

---

---

**Note:** The CensusCSV format displays in this list because you added it to the textpds.conf file.

---

---

10. Under Data Source, click **Browse** to find the `census_csv.txt` file we provided in the *Examples/TextPDS/Scripts* directory.

If you do not see the file listed in your directory, make sure you've selected **TXT** from the drop-down list.

11. Select the file, then click **Open**.
12. When the `census_csv.txt` file displays in the **Location** field, click **OK**.  
The data source definition displays in the text box.
13. Click **Next**.
14. On the Groups page, in the left-hand list, click **CATEGORY**, then click the right arrow to move the group to the Group Fields list.
15. Click **Next**.
16. On the Fields page, click the double right arrows to display all available fields.
17. Click **Next** in the Report Wizard until you see the **Templates** page.
18. On the Templates page, choose **Beige**, then click **Finish**.

The report displays in the Paper Design view, and should look something like this:

**Figure 5-2 Paper Design View of the Final Text PDS Example Report**

Category HISPANIC OR LATINO AND RACE		
Subject	Value	Percentage
Total population	33871648	100
Hispanic or Latino (of any race)	10966556	32.4
Mexican	8455926	25
Puerto Rican	140570	.4
Cuban	72286	.2
Other Hispanic or Latino	2297774	6.8
Not Hispanic or Latino	22905092	67.6
White alone	15816790	46.7
Category HOUSEHOLDS BY TYPE		
Subject	Value	Percentage
Total households	11502870	100
Family households (families)	7920049	68.9
With own children under 18 years	4117036	35.8
Married-couple family	5877084	51.1
With own children under 18 years	2989974	26

19. Save the report as `censusreport_<your initials>.rdf`.

## 5.4 Summary

Congratulations! You have successfully used a text pluggable data source for a paper report. You now know how to:

- Set up your textpds.conf file
- Use the Report Wizard to create a paper report based on the text data source

For more information on using text as a data source, refer to *Getting Started with Oracle9i Reports* on the Oracle Technology Network (<http://otn.oracle.com/products/reports/>), or the *Reports Builder online help*, which you can access by choosing **Help > Help Contents** in Reports Builder.



---

## Building a Report Using Express Data

The report that is described in this chapter is designed to help you learn more about the Reports Builder features for Express data. You will build an Express report that summarizes the yearly projected and actual sales for each region and sales channel in a product division.

To build this report, you will use the Report Wizard to create the initial data model and report layout. You will make refinements to the data model and to the Express query. Finally, you will enhance the look of the report in the Paper Layout view and in the Paper Design view.

### About Express

Express delivers on-line analytical processing (OLAP) using a multidimensional data model. This model is optimized for the analysis of trends or patterns of intersecting corporate data — such as sales, marketing, or financial variables.

### Example Scenario

In this example, you will build a Sales report. Think of the data that you want to extract as being contained in the volume of a cube. Each side of the cube is a list of variable data that is contained in a category (such as Product). This category and its list of values together is called a dimension. You will select portions of each dimension and analyze them for their interaction with other dimensions. This analysis is called a measure.

An example measure for a sales analysis might select data from dimensions for time, product, geographic division, and channel. With Express, you can create a query to report on information that is as broad (for example., yearly direct and indirect sales for products sold everywhere) or as narrow (for example, monthly direct sales for all televisions sold in California) as you like.

describes the steps that you will take to create this report.

**Table 6–1 Features demonstrated in this Express example**

Feature	Location
Use the Report Wizard to define the Express query and create a first draft of the report.	Section 6.2, "Creating an Express report with the Report Wizard"
Streamline the Express query by specifying dimension values.	Section 6.3, "Refining the Express query"
Add summary and calculated totals using the Data Model view.	Section 6.4, "Adding summary columns and custom measures to the data model"
Add summary and calculated totals to the report layout. Enhance the look of the report.	Section 6.5, "Enhancing the report layout"

**Tips for working with Express data** Before you start building a report, be sure that you have reviewed the tips for working with Express data.

---

---

**Note:** For more information on tips for Express data, refer to "About working with Express data" in the *Reports Builder online help*.

---

---

## 6.1 Prerequisites for this example

To build the examples in this manual, you must have the example files we've provided, as well as access to an Oracle Express data source.

### 6.1.1 Example files

If you haven't already done so, you can download the files you'll need to complete this example from the Oracle Technology network (<http://otn.oracle.com/products/reports>) and install them on your machine.

**To download and install the example files:**

1. Go to the Oracle Technology Network Web site (<http://otn.oracle.com/product/reports/>).
2. Click **Getting Started with Oracle9i Reports**.



3. Click **Index**, then find the "Building a Report Using an Express Data Source" example.
4. Download the file `Express.zip` into a temporary directory on your machine (e.g., "d:\temp").
5. Unzip the contents of the file, maintaining the directory structure, into an examples directory on your machine (e.g., d:\orawin90\examples).

This zip file contains the following file:

**Table 6–2** *File(s) necessary for building the Express sample report*

File	Description
<code>Examples\Express\result\xprs.rdf</code>	The final report you will have created when you finish this chapter.

## 6.1.2 Access to an Express Server

Before you start building this Express report, you must have already configured Reports Builder to run with Express Server.

---

**Note:** For more information on configuring for Express data, refer to "About configuring the Express data source" in the *Reports Builder online help*.

---

## 6.2 Creating an Express report with the Report Wizard

You can use the Report Wizard as a great way to start building a report. The Report Wizard alone may give you an Express report that satisfies your requirements. If it does not, then you can use the Data Model view, the Paper Design view, and the Paper Layout view to further refine the report. For this report, you will start with the Report Wizard. The steps in this section will help you to create the initial report.

---

**Note:** For more information, refer to "Building a standard report" in the *Reports Builder online help*.

---

The report that you create in this exercise will present the monthly regional and channel projected and actual sales for each product division. The Express query will

have two measures, and each measure will be dimensioned by product, time, geographic area, and channel.

1. If the Welcome page of the Report Wizard appears, then choose **Use the Report Wizard**, and click **OK**.
2. On the Layout page, select **Create Paper Layout only**, then click **Next**.
3. On the Style page, type `Sales Report` as the Title, and choose **Matrix with Group** as the report style.
4. Click **Next**.
5. On the Data Source Type page, choose **Express Server Query**, and click **Next**.
6. On the Data page, click **Query Definition**.

---

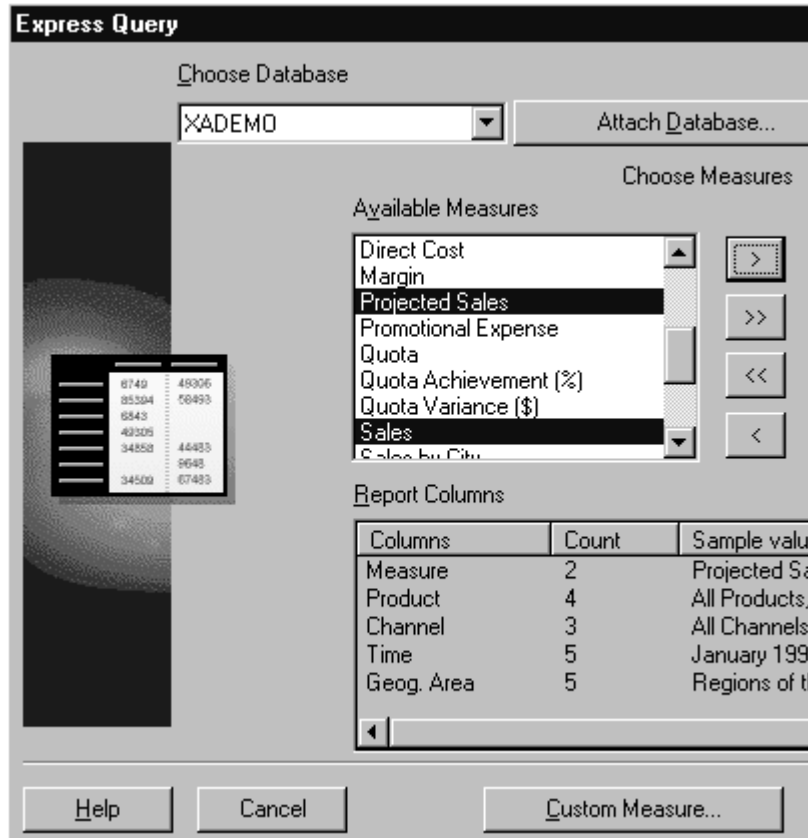
---

**Note:** If you have not already connected to Express Server, then the Connect dialog box appears. Choose the Express Server instance that you want to access. Choose **OK**.

---

---

7. In the Express Query dialog box, choose **Attach Database** to choose the path and name of the database that you want to attach to during this session.
8. In the Attach Database dialog box, select the directory with a label such as `/oec632/`. Select `xademo.db`. This is the sample database that is provided with Express Server.
9. Click **Open** to attach the database to the session.
10. In the Express Query dialog box, CTRL-click to select Sales and Projected Sales from the Available Measures list.
11. Click the right arrow to move Sales and Projected Sales to the Selected Measures list box. The Express Query dialog box looks similar to the following figure:



12. Click **OK** to accept the Express query selections. You will return to the dialog box in a later step to refine the dimension values that are associated with the Sales and Projected Sales measures.
13. On the Data page, click **Next**.
14. On the Groups page, select **PRODUCT** in the Available Fields list box and click the right arrow. to move this field to the Matrix Group Fields list box.

15. Select Level1, then select **TIME** and click the right arrow so that the Matrix Group Fields box appears as follows:



16. Click **Next**.
17. On the Rows page, click **GEOG\_AREA**.
18. Click the right arrow.
19. Click **Next**.
20. On the Columns page, choose **CHANNEL** in the Available Fields list box.
21. Click the right arrow.
22. Click **Next**.
23. On the Cells page, select **PROJECTED SALES** in the Available Fields list box and choose the Add One button to move this field to the Matrix Cell Fields list box.
24. Repeat this step for **SALES**.
25. Click **Next**.
26. On the Totals page, click **Next**.

You will add summary totals in a later step.

27. On the Labels page, change the following labels and widths:

Field	Label	Width
SALES	Actual Sales	7
PROJECTED_SALES	Projected Sales	7
GEOG_AREA	Region	10
PRODUCT	Product:	10
TIME	Time:	10
CHANNEL	Channel	7

You should change the width of labels at this point, because in a later step you will add a new layout column. This will cause columns to wrap to the next page at their current default width.

28. Click **Next**.
29. On the Template page, choose **Predefined template** if it is not already selected, and choose **Gray** in the list box.
30. Click **Finish**. The report output automatically displays in the Paper Design view and should look similar to the following figure.

Region	Projected Sales		Projected Sales	
<b>Regions of the World</b>	9262779	8769594	4068757.5	4068757.5
<b>Areas in the Americas</b>	2768417.25	2620113.5	1206539.625	1206539.625
<b>Australia</b>	782333.875	740387	340566.09375	340566.09375
<b>Europe</b>	3982743.75	3773573	1779844	1779844
<b>Asia</b>	1729284.5	1635521.25	741807.6875	741807.6875

31. Choose **File > Save As**. Save the report in the directory of your choice, and name the report `xprs_910.rdf`.

---

**Note:** It is good practice when you are designing a report to save it frequently under a different file name. If you generate an error or if you do not like some of the changes that you made, then you easily can go back to the previously saved file and make revisions from that point.

---

## 6.3 Refining the Express query

The steps in this section will help you refine the Express query. So far you have developed a useful report that shows the monthly projected and actual sales for

each region and channel in a product category. But you are really interested in the yearly projected and actual sales results for each channel and region in a product division. You can achieve this by restricting the dimension values that you want to view.

---

---

**Note:** For more information, refer to "Selecting data" in the *Reports Builder online help*.

---

---

In this exercise, you will specify the following dimension values in the Express Query dialog box:

- projected and actual sales for 1997
  - geographic regions, such as Asia and the Americas
  - product divisions, such as the Accessory and Audio division
1. In the Paper Design view, choose **Tools > Report Wizard**.
  2. On the Data page, choose **Query Definition**.
  3. In the Edit Query dialog box, choose **Selector**.
  4. In the Selector dialog box, choose **Time Period** from the Dimensions option.
  5. Click the **List** button to select the List tool from the toolbar.
  6. In the List dialog box, choose **1997** from the Available Time Periods list box.
  7. Click **Select**. Notice that "1997" replaces the previous selections.
  8. Click **OK**.
  9. In the Selector dialog box, choose **Geographical Area** from the Dimensions option.
  10. Click the **Level** button to select the Level tool from the toolbar.
  11. In the Select by Level dialog box, choose **Continents/Regions** in the At level(s) list box.
  12. Click **OK**.
  13. In the Selector dialog box, choose **Product** from the Dimensions option.
  14. Click the **Level** button.
  15. In the Select by Level dialog box, choose **Divisions** in the At level(s) list box.

16. Click **OK**.
17. In the Selector dialog box, click **OK**.
18. In the Express Query dialog box, click **OK**.
19. On the Groups page, choose **TIME** in the Matrix Group Fields list box. Note that using **TIME** as a break group is no longer necessary since the Express query will retrieve only aggregate data for 1997.
20. Click the **Remove One** button. **PRODUCT** should be the only dimension that is listed in the Matrix Group Fields list box.
21. On the Style page, change the title to 1997 Sales Report.
22. Choose **Finish**. Your report should look similar to the following figure:

Product: Audio Division		1997 Sales Report			
Region	Channel	All Channels	Direct		
		Projected Sales	Actual Sales	Projected Sales	
Areas in the Americas		5499198	7905061	3627530	3
Australia		1539263	2225799	1005164.6875	1
Europe		7315498.5	10423398	4897673	4
Asia		3088618.5	4461758	2020369.25	2

23. Save the report as `xprs_920.rdf`.
24. Optionally, you can compare this report with the one that you previously saved as `xprs_910.rdf`.

Notice the projected and actual sales. In the new report, each cell represents the yearly sales for a region and channel in a product division for 1997, while the previous report displays sales data for a region and channel in a product division for each month.

## 6.4 Adding summary columns and custom measures to the data model

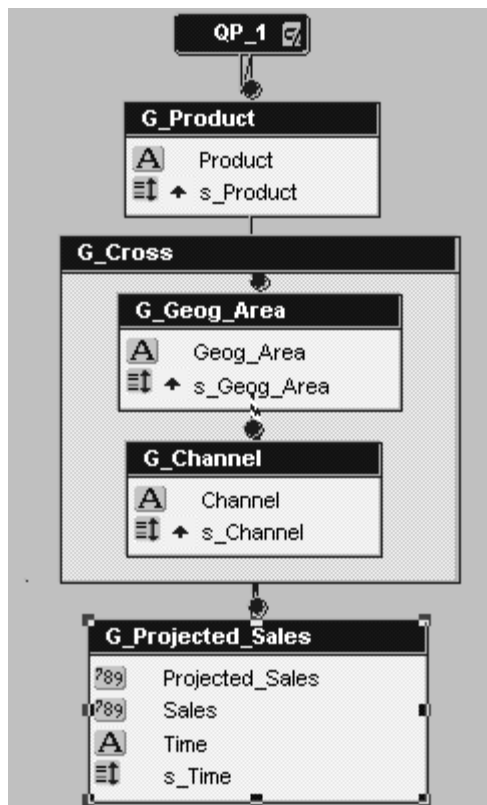
The steps in this section will help you refine the data model to include summary totals for each channel in a product division. Additionally, you are curious about

how accurately you predicted the actual sales. You can determine this by creating a *custom measure* that calculates the percent of sales above projected sales.

First, you will create the summary column using the Summary tool in the Data Model view.

Next, you will create the custom measure using the Custom Measure tool in the Express Query dialog box.

Before you begin, examine the data model:



In the Data Model view you may notice additional columns, such as S\_GEOG\_AREA, or S\_CHANNEL. These are *dimension sorting* columns. They are visible only in the data model and are the index used to sort dimensions by logical order, as opposed to alpha-numeric order. If you move a column to a new group, then you must also move the associated sort column into that group as well.



In a later step, you will sort dimension values using the Sort tool in the Edit Query dialog box.

### 6.4.1 Renaming data objects

1. In the Object Navigator, double-click the Data Model button under your report's node if you are not already viewing the Data Model view.
2. Select QP\_1.
3. Choose **Tools > Property Inspector**.  
If you want to modify the Express query, then choose the Express Query property under the **Query** node.
4. Under the **General Information** node, change the Name property to QP\_SALES.
5. Press ENTER or click outside of the property to accept the value. Close the Property Inspector.
6. Repeat steps 2 through 5 and change the Name property of the **G\_PROJECTED SALES** group to G\_SALES\_DATA.
7. Save the report as `xprs_931.rdf`.

### 6.4.2 Creating summary columns

In this exercise, you will add two summary columns to the G\_Cross group. Each summary column will calculate the projected and actual sales totals for each channel (all channels, direct, and indirect) in a product division.

1. In the Data Model view, click the **Summary Column** icon, then click the G\_Cross group.
2. Choose **Tools > Property Inspector**.
3. Set the following properties for projected sales:

Node	Property	Value
General Information	Name	CS_PjSalesPerChannel
Column	Product Order	G_CHANNEL
Summary	Source	PROJECTED_SALES
Summary	Reset At	G_CHANNEL

4. Press ENTER or click outside of the property to accept the value.
5. Close the Property Inspector.
6. Repeat steps 1 through 4 to create a summary column for actual sales. Set the following properties:

Node	Property	Value
General Information	Name	CS_SalesPerChannel
Column	Product Order	G_CHANNEL
Summary	Source	SALES
Summary	Reset At	G_CHANNEL

7. Save the report as `xprs_932.rdf`.

### 6.4.3 Creating a custom measure

In this exercise, you will create a custom measure that will calculate the percent of actual sales above projected sales for each region and in each product division. To do this, you will use the Custom Measure tool within the Express Query dialog box to build the new measure called Increase.

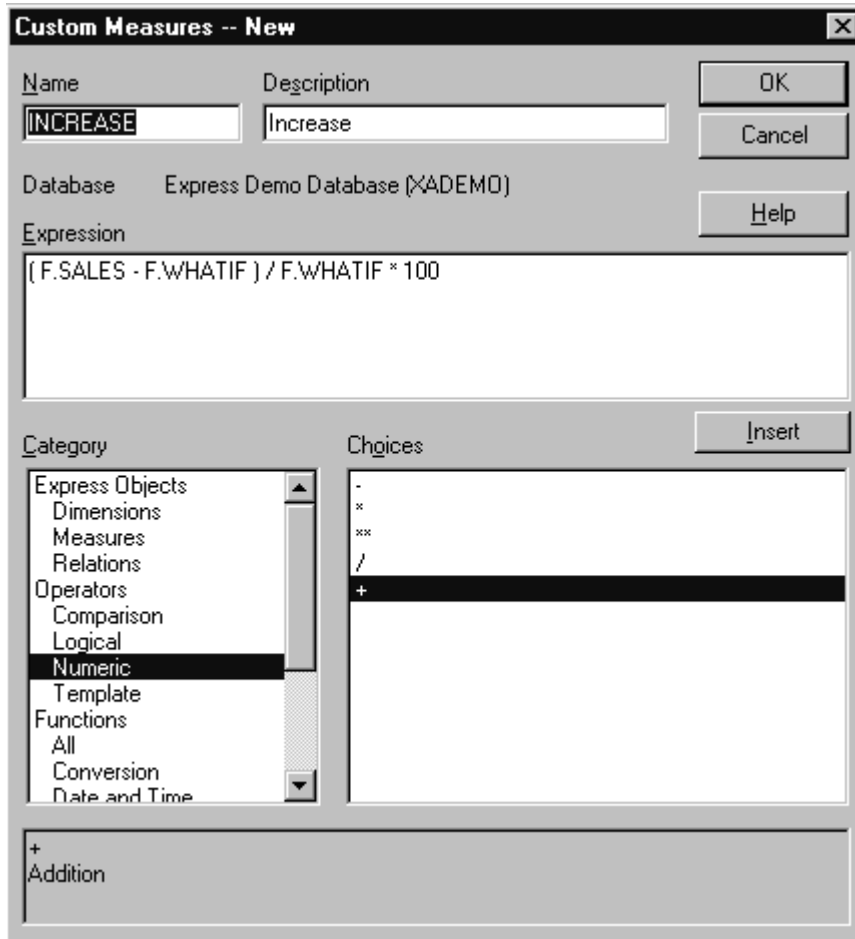
1. In the Data Model view, double-click the **QP\_Sales** query object to open the Express Query dialog box.
2. Choose **Custom Measure** at the bottom of the Express Query dialog box.
3. Choose **New** to open the Custom Measure — New dialog box.
4. In the **Name** box, type `INCREASE`.
5. In the **Description** box, type `Increase`.
6. Choose **Template** under Operators in the Category box. Notice a list of templates appears under Choices.
7. Select the left parenthesis and choose **Insert**. A left parenthesis appears in the **Expression** box.
8. Choose **Measures** under **Express Objects** in the Category box.
9. Select **F.SALES**, and choose **Insert**.

10. Use the following table to build the expression:

Category	Sub-category	Choose	or Type:
Operators	Numeric	Minus Sign	-
Express Objects	Measures	F.WHATIF	F.WHATIF
Operators	Template	Right parenthesis	)
Operators	Numeric	Forward slash	/
Express Objects	Measures	F.WHATIF	F.WHATIF
Operators	Numeric	asterisk	*

11. Following the asterisk, type 100 in the **Expression** box.

12. When you are finished, the expression should look similar to the one in the following figure:



13. Click **OK**. Note that "Increase" is listed in the **Custom Measures** text box in the Custom Measures dialog box.
14. Click **Close**.
15. In the Express Query dialog box, scroll through the **Available Measures** box. "Increase" now appears alphabetically. Choose **Increase** and choose the Add One button. "Increase" appears in the Selected Measures box, below Projected Sales and Sales.
16. Click **OK** to return to the Data Model.

The group G\_SALES\_DATA now includes the custom measure that you just created, INCREASE.

17. Click the **Run Paper Layout** button to view the report in the Paper Design view. Note that neither the summary columns nor the custom measure are available in the report. This occurred because you have not yet added them as fields to the report layout. You will do this in the next few exercises.
18. Save the report as `xprs_933.rdf`.

## 6.5 Enhancing the report layout

The steps in this section show you how to re-arrange the report layout, add the summary and custom measure columns that you created in Section 6.3, "Refining the Express query", and format objects to further enhance the look of the report. You make these changes using the Paper Layout view and the Paper Design view.

### 6.5.1 Inserting summary fields in the report

1. In the Object Navigator, double-click the **Paper Layout** icon under the report's node to display the Paper Layout view.
2. Arrange the workspace to display the Object Navigator and the Paper Layout view side-by-side. Expand the Paper Layout, Main Section, Body, and the remaining nested nodes, such as the M\_G\_PRODUCT\_GRPFR and R\_G\_PRODUCT nodes.
3. In the Object Navigator, type `M_G_CROSS_GRPFR` in the Find field to locate this object. In the Paper Layout view, the master cross-matrix frame is selected.
4. Extend the selected frame down about 1/4 inch.
5. In the Object Navigator, choose **F\_CHANNEL**.
6. Click the **Select Parent Frame** button to select the parent frame, R\_G\_CHANNEL.

---

---

**Note:** You may need to resize the Paper Layout window to see the Select Parent Frame button, as it is located on the far right of the toolbar.

---

---

7. Extend the frame down about 1/4 inch.
8. Click the **Field** button.

9. Click and drag a rectangle in the area directly under the `F_PROJECTED_SALES` field to insert a field object.
10. Choose **Tools > Property Inspector**.
11. In the Property Inspector, set the Source property under the Field node to the value `CS_PjSalesPerChannel`.
12. Select the object in the Object Navigator and rename it to `F_PjSalesPerChannel`.
13. Arrange this field and change the format as follows:
  - Click the **Fill Color** button to change fill color to light yellow.
  - Click the **Text Color** button to change the text color to dark brown.
  - Click the **Line Color** button to surround the field with dark brown border lines.

---

---

**Note:** You can turn Snap to Grid on or off as desired to help you arrange objects in the layout. Select **View > Snap to Grid**. A check mark indicates that the option is on.

---

---

14. Repeat steps 8 and 9 except place the new object directly under `F_SALES`.
15. In the Property Inspector, set the Source property under the Field node to the value `CS_SalesPerChannel`.
16. Select the object in the Object Navigator and rename it to `F_SalesPerChannel`.

---

---

**Note:** The fill and text colors, as well as the border lines, match the field that you just created, `F_PjSalesPerChannel`.

---

---

17. Click the **Text** button.
18. Click and drag a rectangle to fill the space directly under `F_GEO_AREA`.
19. In the rectangle, type `Totals:`.
20. Align the text object to center by clicking the **Align Center** button and make format changes to match the summary fields that you created.

21. Click the **Paper Design** icon to view the changes in the Paper Design view.
22. Save the report as `xprs_941.rdf`.

## 6.5.2 Inserting the custom measure field into the report

In this section, you will add a column to display the custom measure that you created in Section 6.4.3, "Creating a custom measure" by inserting a field object in the report layout.

To do this, you will add a new column to the layout of the report and insert the field object into the column.

**Tip:** The new field object also must have the same frequency as `F_PROJECTED_SALES` and `F_SALES`. If the field object is not at the same frequency, then the report will fail to run.

1. In the Paper Design view, click the **Paper Layout** icon to display the Paper Layout view. Ensure that the Paper Layout view and Object Navigator are placed side-by-side.
2. In the Object Navigator, Ctrl-click `M_G_PRODUCT_GRPFR` and `R_G_PRODUCT`.

**Tip:** `M_G_PRODUCT_GRPFR` is the underlying master group. It is hidden directly under `R_G_PRODUCT`. In the Paper Layout view, it may look like only one group is selected when, in fact, both frames are selected.

3. In the Paper Layout view, expand the width of the selected frames to about 4 3/4 inches.

---

---

**Note:** Click the **Flex On** button to turn Flex mode on, or click the **Flex Off** button to turn Flex mode off if you are unable to resize or move an object.

---

---

4. In the Object Navigator, click `M_G_CROSS_GRPFR`.
5. In the Paper Layout view, expand the width of the selected frame to about 4 3/4 inches.

6. Choose the **F\_GEOG\_AREA** object, then click the **Select Parent Frame** button to select the parent frame, **R\_G\_GEOG\_AREA**. Expand the width of the selected frame to about 4 3/4 inches.
7. Choose **F\_CHANNEL** and click the **Select Parent Frame** button to select the parent frame, **R\_G\_CHANNEL**.
8. Expand the width of the selected frame to about 4 3/4 inches.
9. Choose **F\_CHANNEL** again and expand the width of the object to about 4 3/4 inches.
10. Click the **Field** button.
11. Click and drag a box to the right of the **F\_SALES** object.
12. Choose **Tools > Property Inspector**.
13. Set the following properties:

Node	Property	Value
General Information	Name	F_Increase
Field	Source	INCREASE

14. Click the **Run Paper Layout** icon to run the report. You should see an error message that indicates that **F\_Increase** references **INCREASE** at a frequency below its group. You are unable to run the report.

To understand why this error occurred, look for **F\_INCREASE** in the Object Navigator. It is probably placed at a higher level (and lower frequency) than **R\_G\_PROJECTED\_SALES**. Recall that the column **INCREASE** calculates the percent of actual sales above projected sales. In order to run this report, **F\_INCREASE** must have the same frequency as **F\_PROJECTED\_SALES** and **F\_SALES** to reference the data that it needs to calculate the value.

15. Click **OK** to close the error message.
16. Click the **Paper Layout** icon to display the Paper Layout view.
17. Select the **F\_INCREASE** field and delete it.
18. Choose **F\_SALES**, then choose the **Select Parent Frame** button to select the parent frame, which is called **R\_G\_PROJECTED\_SALES**.
19. Expand the width of the selected frame to about 4 3/4 inches.
20. Repeat steps 10 through 13 to create the field object.



21. With the **F\_Increase** object selected, locate **F\_INCREASE** in the Object Navigator to ensure that it has the same frequency as **F\_PROJECTED\_SALES** and **F\_SALES**.
22. Change the format of the **F\_Increase** field as follows:
  - Click the **Fill Color** button to change fill color to light yellow.
  - Click the **Text Color** button to change the text color to dark brown.
  - Click the **Line Color** button to surround the field with dark brown border lines.
  - Click the **Bold** button to make the text darker and more noticeable.
23. Click the **Text** button.
24. Click and drag a rectangle above **F\_Increase** to add the column title.
25. In the rectangle, type **Increase**.
26. Arrange the text object in the column and change the format to match the field to its left, **Actual Sales**.

---

---

**Note:** You may want to turn off **Snap to Grid** on the View menu in order to extend the text object to cover the entire field. Ensure that the text object is selected when you apply formatting, or it will not take effect.

---

---

27. Click the **Run Paper Layout** icon to view your report.
28. Save the report as `xprs_942.rdf`.

### 6.5.3 Sorting dimension values

Suppose you want to change the sorting order of the distribution channels in the report. In this exercise, you will change the sorting criteria for the Channel dimension by using the Selector in the Express Query dialog box. Instead of listing the order by the default channel hierarchy (top to bottom), you will display data from the lowest to the highest channel in the hierarchy. Note that the hierarchy is predefined in the database to place "All Channels" first, with "Indirect" placed last.

1. In the Data Model view, double-click the query object, **QE\_SALES**.

2. Choose **Selector** in the Express Query dialog box.
3. In the **Dimensions** list, select **Distribution Channel** and choose the **Sort** button.
4. In the Sort Selection dialog box, choose the following values:

<b>Criteria</b>	<b>Selection</b>
based on	hierarchy
in order	bottom to top
in hierarchy	Standard

5. Click **OK** in the Sort Selection dialog box.
6. Click **OK** in the Selector dialog box.
7. Click **OK** in the Express Query dialog box.
8. Click the **Run Paper Layout** icon to view the report.
9. Save the report as `xprs_943.rdf`.

#### 6.5.4 Making format changes in the Paper Design view

1. In the Paper Design view, SHIFT-click the columns under Projected Sales and Actual Sales, and the Projected Sales total and the Sales total fields.
2. Click the **Currency** button to change the format mask to currency.
3. Click the **Align Right** button to right justify the values.
4. Click the **Add Decimal Place** button twice to insert two decimal places.
5. Under **Increase**, click the column.
6. Click the **Percent** button to change the format mask to percentage.
7. Click the **Align Center** button to center the values.

The report should now look similar to the following figure:



## 1997 Sales Report

Product: Audio Division			
	Channel	Indirect	
Region	Projected Sales	Actual Sales	Increase
Areas in the Americas	\$1871668.13	\$4277531.00	129%
Australia	\$534098.31	\$1220634.25	129%
Europe	\$2417825.50	\$5525725.00	129%
Asia	\$1068249.13	\$2441388.50	129%
Totals:	\$5891841.06	\$13465278.75	

8. Save the report as `xprs_944.rdf`.

## 6.6 Summary

Congratulations! You have finished the Express sample report. You now know how to:

- Use the Report Wizard to define a data model and layout.
- Make changes to the Express query by restricting the dimension values.
- Use the Data Model view to add summary and custom measures columns to the report.
- Use the Paper Layout view to insert fields and re-arrange the layout.
- Use the Paper Design view to enhance the look of the report.



---

---

# Glossary

## **checkbox**

A interface element, appearing as a small square, that a user can toggle on or off.

## **column**

1. A vertical space in a database table that represents a particular domain of data. A column has a column name (e.g., ENAME) and a specific datatype (e.g., CHAR). For example, in a table of employee information, all of the employees' names would constitute one column. A record group column represents a database column.

2. A data model object created automatically for each column expression in a query's SELECT list, or created manually to perform summaries, formulas, or act as a placeholder.

## **data model**

A relational model that defines what data should be fetched from the database, what values should be computed, and how data should be ordered in a report. Reports Builder objects that define the data model are queries, groups, columns, parameters, and links.

## **Data Model view**

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.

## **database**

1. A set of dictionary tables and user tables that are treated as a unit.

2. (Oracle Express) A single file (possibly accompanied by extension files) that contains objects that organize, store, and manipulate data. In Express, examples of such objects are variables, dimensions, formulas, models, and programs.

**data source**

A source for data returned by a query, including database objects such as tables, views, synonyms, snapshots, and queries stored as views.

**dialog box**

A partial screen or window that prompts you to enter information necessary to complete an operation.

**editor**

A work area in which you perform a specific set of tasks, such as creating a program unit or designing an application.

**field**

1. An interface element in which you enter, edit, or delete data. 2. A layout object that defines how the data for a specific query column appears.

**HTML (Hypertext Markup Language)**

Acronym for Hypertext Markup Language. A tag-based ASCII language used to specify the content and links to other documents on Web servers on the Internet. End users with Web browsers view HTML documents and follow links to display other documents.

**icon**

A graphic representation of a window or tool.

**image**

A bitmapped object that can be stored and loaded into an application. The client cannot modify an imported image.

**intranet**

An internal TCP/IP network, access to which is restricted (via a firewall) to individuals inside the company or organization. An intranet provides similar services within an organization to those provided by the Internet, but is not necessarily connected to the Internet. A common example of an intranet is when a

company sets up one or more Web servers on an internal network for distribution of information or applications within the company.

## **Java**

A computer language that supports programming for the Internet in the form of platform-independent "applets".

## **JSP (JavaServer Page)**

An extension to the servlet functionality that enables a simple programmatic interface to Web pages. JSPs are HTML pages with special tags and embedded Java code that is executed on the Web or application server providing dynamic functionality to HTML pages. JSPs are actually compiled into servlets when first requested and run in the server's JVM.

## **layout**

The area of an editor in which you can create, modify, position, or delete objects.

## **object**

1. An item that can be placed on the layout. The following are examples of objects: rectangle, line, ellipse, arc, polygon, polyline, rounded rectangle, freehand, graph, text, symbol, and text field. 2. In Oracle8, an instance of an object type. An object can be a row in an object table, or the portion of a row contained in a column object in a relational table.

## **Oracle9i Application Server**

The Oracle9i Application Server (Oracle9iAS) is a strategic platform for network application deployment. By moving application logic to application servers and deploying network clients, organizations can realize substantial savings through reduced complexity, better manageability, and simplified development and deployment. The Oracle9iAS provides the only business-critical platform that offers easy database web publishing and complete legacy integration while transition from traditional client-server to network application architectures.

## **Oracle9i Developer Suite**

Oracle9i Developer Suite (Oracle9iDS) combines leading Oracle application development and business intelligence tools into a single, integrated product. Built on Internet standards such as Java and XML, the suite provides a complete and highly productive development environment for building applications for Oracle9i Application Server and the Oracle9i Database.

### **<Oracle\_Home>**

An alternate name for the top directory in the Oracle directory hierarchy on some directory-based operating systems. An environment variable that indicates the root directory of Oracle products.

### **Oracle9iAS Portal**

Oracle9iAS Portal is an HTML-based development tool for building scalable, secure, extensible HTML applications and Web sites. Oracle9iAS Reports Services uses Oracle9iAS Portal to control end user access to reports published on the Web by storing information about report requests, the secured server, and any Oracle9iAS Reports Services printer used to print report output.

### **Oracle9iDS Reports Builder (rwbuilder)**

Creates, develops, and maintains report definitions.

### **Oracle9iAS Reports Services**

The runtime environment for Reports Developer applications. Oracle9iAS Reports Services executes, distributes, and publishes your reports for enterprise wide reporting. Using Reports Services to deploy your reports results in gains of flexibility, time savings, and processing capacity.

### **Oracle9iAS Reports Servlet**

An interface between a Java-based Web server and Oracle9iAS Reports Runtime, enabling you to run report dynamically from your Web browser.

### **Paper Design view**

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 Layout view**

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.

### **PDF (Portable Document Format)**

A file format (native for Adobe Acrobat) for representing documents in a manner that is independent of the original application software, hardware, and operating system used to create the documents. A PDF file can describe documents containing



any combination of text, graphics, and images in a device-independent and resolution independent format.

### **PL/SQL**

Oracle's proprietary extension to the SQL language. Adds procedural and other constructs to SQL that make it suitable for writing applications.

### **query**

A SQL SELECT statement that specifies the data you wish to retrieve from one or more tables or views of a database.

### **RDF file**

A file that contains a single report definition in binary format. .RDF files are used to both run and edit reports.

### **SELECT statement**

A SQL statement that specifies which rows and columns to fetch from one or more tables or views.

### **SQL**

A standard interface for storing and retrieving information in a relational database. SQL is an acronym for Structured Query Language.

### **SQL file**

A file that contains a query stored in text (e.g., ASCII or EBCDIC) format.

### **SQL script**

A file containing SQL statements that you can run to perform database administration quickly and easily. Several SQL scripts are shipped with Oracle products.

### **SQL statement**

A SQL instruction to Oracle. A SELECT statement is one type of SQL statement.

### **style sheet**

HTML extensions that provide powerful formatting flexibility in HTML documents. To view an HTML document that takes advantage of style sheets, display it in a browser that supports style sheets.

**syntax**

The orderly system by which commands, qualifiers, and parameters are combine to form valid command strings.

**table**

A named collection of related information, stored in a relational database or server, in a two-dimensional grid that is made up of rows and columns.

**tabular**

A default layout displaying labels at the top of the page and rows of data underneath the labels.

**template**

A skeleton definition containing common style and standards, and may include graphics. A template provides a standard format to enable quick and easy development of professional standard look-and-feel reports.

**toolbar**

A collection of iconic buttons that perform product commands. Usually aligned horizontally along the top, or vertically down the side of a window.

**tool palette**

A collection of tools.

**URL (Uniform Resource Locator)**

A URL, a form of URI, is a compact string representation of the location for a resource that is available through the Internet. It is also the text string format clients use to encode requests to Oracle9iAS.

**Web browser**

A program that end users utilize to read HTML documents and programs stored on a computer (serviced by a Web server).

**Web server**

A server process (HTTP daemon) running at a Web site which sends out Web pages in response to HTTP requests from remote Web browsers.

**Web source view**

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. Experienced Java developers can edit the Web source directly in this view.

**window**

A rectangular area of the desktop that contains an application. Each window has an area where you can interact with the application. Windows can be opened, resized, moved, reduced to an icon, or enlarged to fill the entire desktop.

**wizards**

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

- Report Wizard: guides you through the steps to create a basic paper or Web report. Each page of the wizard asks you for information to help you create your initial report.
- Data Wizard: helps you quickly define or modify a query for a multiquery data models.
- Graph Wizard: Adds variety of charts and graphs, including true 3-dimensional graphs. Implemented in Reports Builder with the Oracle BI graph bean.
- Report Block Wizard: enables you to add data to a static HTML page.

**XML**

Extensible Markup Language -- a metalanguage using SGML to define and structure data. Reports Builder supports XML output to enable Web publishing as well as electronic data exchange with third-party applications. You can also use XML to build report definitions that can be merged with other report definitions at runtime or run separately.



---

---

# Index

## A

---

- adding
  - summary columns, 3-13
- alternating row colors, applying, 4-10

## B

---

- barcode
  - barcode JavaBean, about, 1-1
  - barcode JavaBean, initializing, 1-20
  - barcode, using for a Web report, 1-14
  - barcode, using in a paper report, 1-5
- Before Report trigger, creating, 1-7
- break group, creating, 3-9
- bursting, 2-3

## C

---

- columns, summary, 3-13
- creating
  - break group, 3-9
  - layout, 3-15
  - links between ref cursor queries, 3-11
  - package body, 3-19
  - ref cursor query, 3-6

## D

---

- data link, creating, 4-7
- data source, about XML, 4-1
- data, filtering, 4-13
- defining
  - package spec, 3-4

- ref cursor types, 3-4
- distribution XML definition, editing, 2-5
- distribution, editing the XML definition, 2-5
- distribution, using, 2-1

## F

---

- format, applying alternating row colors, 4-10
- formula column, adding to a data model, 1-8
- formula column, creating, 1-8, 1-17

## H

---

- HTML file, adding a query, 1-15

## J

---

- Java classes, importing, 1-5
- Java importer, using, 1-5
- JavaBean, importing, 1-5
- JavaBean, using for a Web report, 1-14
- JavaBean, using in a paper report, 1-5

## L

---

- layout, creating, 3-15
- layout, creating for paper, 1-12, 4-9
- links, creating between ref cursor queries, 3-11

## M

---

- moving packages into a library, 3-21
- multiple destinations, distributing a report, 2-1

## P

---

package  
  body, creating, 3-19  
  moving a SELECT statement into, 3-18  
  moving into a library, 3-21  
  spec, defining, 3-4  
package, creating, 1-6  
paper layout, creating, 1-12, 4-9  
paper report, creating a layout, 4-9  
paper report, using a barcode JavaBean, 1-5  
pluggable data source, using text, 5-1  
pluggable data source, using XML, 4-1  
Program Unit Editor, using, 3-6

## Q

---

query, building, 1-15  
query, building a SQL query manually, 4-4  
query, building an XML query, 4-6  
query, using XML and SQL, 4-4

## R

---

ref cursor  
  queries, creating links between, 3-11  
  query, creating, 3-6  
  type, defining, 3-4  
Report Wizard  
  using, 6-3  
report, bursting, 2-3  
report, creating manually, 4-4  
report, distributing to multiple destinations, 2-1  
report, running to the Web, 1-24, 2-7  
reports  
  Express, 6-1  
REPORTS\_CLASSPATH, updating, 1-4  
row colors, applying alternating row colors, 4-10

## S

---

SELECT statement, moving into a package, 3-18  
SQL query, building manually, 4-4  
SQL, building a query manually, 4-4  
SQL, building a query with SQL and XML, 4-4  
summary

columns, adding, 3-13

## T

---

text data source, building a report, 5-4  
text data source, configuring, 5-3  
text data source, using, 5-1  
text pluggable data source, building a report, 5-4  
text pluggable data source, configuring, 5-3  
text pluggable data source, using, 5-1  
text, building a report using a text data source, 5-4  
text, configuring the textpds.conf file, 5-3  
text, using as a data source, 5-1  
textpds.conf, editing, 5-3

## U

---

using  
  Report Wizard, 6-3

## W

---

Web report, running, 1-24, 2-7  
Web report, using a barcode JavaBean, 1-14

## X

---

XML data source, about, 4-1  
XML data source, building an XML query, 4-6  
XML query, building, 4-6  
XML, building a query with SQL and XML, 4-4