Matthew Bennett

Programming with Oracle Developer

SAMS

A Division of Pearson Technology Group
201 West 103rd Street, Indianapolis, Indiana 46290

Programming with Oracle Developer

Copyright © 2002 by Sams

International Standard Book Number: 0672321106

Library of Congress Catalog Card Number: 00-110296

Printed in the United States of America

First Printing: August 2001

04 03 02 01 4 3 2 1

Trademarks

Warning and Disclaimer

Executive Editor
Rosemarie Graham

Acquisitions Editor
Angela Kozlowski

Development Editor
Kevin Howard

Managing Editor
Charlotte Clapp

Project Editor
Heather McNeill

Production Editor
Michael Henry

Indexer
Aamir Burki

Proofreader
Teresa Stephens

Technical Editor
Bruce Kostival
Jeff Rowan

Team Coordinator
Lynne Williams

Interior Designer
Anne Jones

Cover Designer
Alan Clements

Page Layout
Lizbeth Patterson

ESSENTIALS

- Query Builder is linked into Oracle Developer tools, but also can stand alone to help build complex SQL queries.
- Query Builder allows you to create simple or inner joins as well as outer joins. An outer join allows all rows for the specified table to be displayed regardless of whether there are corresponding rows in the lookup table.
- Query Builder allows you to save your query in three different formats. The first two formats are used to specify extra information used by Query Builder. The last method is to save a SQL SELECT statement that can is easily imported into other applications.
- Query Builder allows you to see the results of your query as you build it. Simply click the Execute Query button found on the toolbar. If too many rows are returned, you can qualify your query using conditional logic and rerun the query.
- Schema Builder is a tool used to create database objects such as tables, views, and indexes. It is geared toward small projects. Larger projects should utilize the enhanced features found in Oracle Designer.
- Specifying relationships between tables by using Schema Builder will help you create queries across tables later as you develop your application.

Oracle Developer 6i includes several other tools to help you create database applications. These tools include Query Builder, Schema Builder, Procedure Builder, and Project Builder. Procedure Builder is discussed in Chapter 6, "PL/SQL Workshop," and Project Builder is covered in Chapter 14, "Project Management and Source Code." This chapter covers the Query Builder running outside Report Builder and Schema Builder.

More About the Query Builder

Query Builder was used to help create the SQL query used for the report in Chapter 3, "Oracle Forms Developer." Although it can be used with Report Builder, it can also be used as a standalone application. Doing so allows you to create complex queries and save them for later use.

Query Builder can be used as a simple report-creation tool. If you get a one-time request for a complex but short report, it is often easier to create it using Query Builder than Report Builder. You can save the query, should the report be requested again.

In this section, you are shown how to use Query Builder to create short reports. Instead of creating the same master-detail request used in Chapter 4, "Oracle Reports Developer and Graphics," you create a lookup query. This requires you to use existing database table relationships as well as to create undefined ones.

Running Query Builder

Start Query Builder using the Start menu on Windows, or by running the obe60 program on Unix. If you have any problems running obe60, you may need to contact your system administrator to help correctly configure your environment. You are asked to log in to the database before you are allowed to do anything else. Use the scott/tiger account. If the Query Builder tables were not created in the database when it was installed, you get an informational message telling you such. Unfortunately, that informational message looks like an error message and might lead you to believe there is a problem. The message simply means that you can't store queries in the database and must store them in files. I suggest that you install the tables to take advantage of being able to create views from queries with the Schema Builder.

QUERY BUILDER TABLES

If you want to install the Query Builder tables, you can do so by rerunning the Oracle installer and selecting Create database objects. You can also create the tables by running the brwin60.sql script found in $HOME_ORACLE/browser60/admin/sql on Unix or $HOME_ORACLE\BROWSE60\SQL on Windows.

On login, Query Builder asks whether you want to create a new query, load one out of the database, or load one from the file system. If you have not installed the required tables in the database, the option to load a query from the database will be disabled. Create a new query by selecting Create New Query and clicking OK.

You are now asked to specify the tables to use for the query. Notice the icon on the left that indicates the database object is a table. A different icon is used to indicate views. For the first example, select the EMP table, click Include, and then click Close. The results in the Query window should look similar to Figure 5.1.

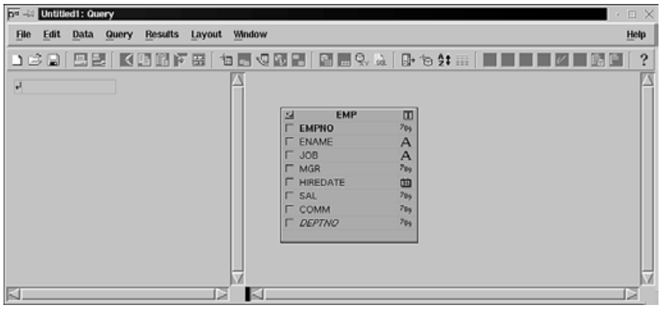


FIGURE 5.1
The Query window of Query Builder after selecting the EMP table to use for the query.

FIGURES MAY DIFFER

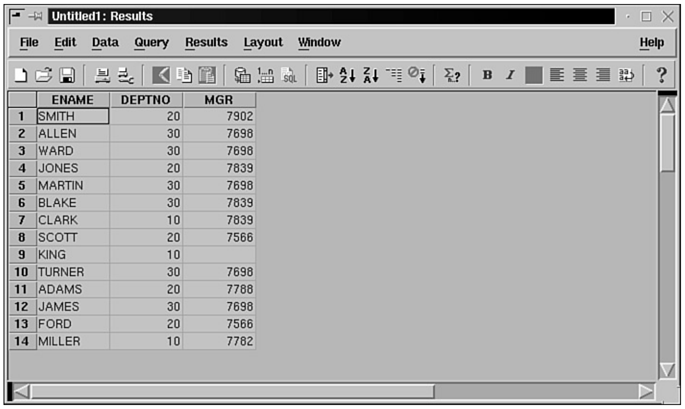
The Windows version of Query Builder uses a multiple document interface (MDI) to display both the Query and Results windows. On Unix, there will be two separate windows. This should not cause any problems. Just be aware that the toolbars in the figures might be slightly different than what is on your screen, depending on the platform you use.

You should be very familiar with all the values for the various columns in the EMP table by now. Therefore, select only the ENAME, DEPTNO, and MGR columns to display with this query. You select a column for display by checking the box next to the column name. As you select columns, they will appear in the results window. Execute the query using the Query, Execute menu item. The results are shown in Figure 5.2.

Creating Lookup Queries

The results from this query don't provide an easy way for the user to see department names and managers. This section shows how to replace cryptic values with easily readable ones by using lookup values.

The first value to change is DEPTNO. It should be replaced with DNAME from the DEPT table. The DEPT table is added to the Query screen by selecting the Data, Select Data Tables menu option. Again you are presented with the Select Data Tables dialog. Highlight the DEPT table, click on Include, and then click the Close button.



	ENAME	DEPTNO	MGR
1	SMITH	20	7902
2	ALLEN	30	7698
3	WARD	30	7698
4	JONES	20	7839
5	MARTIN	30	7698
6	BLAKE	30	7839
7	CLARK	10	7839
8	SCOTT	20	7566
9	KING	10	
10	TURNER	30	7698
11	ADAMS	20	7788
12	JAMES	30	7698
13	FORD	20	7566
14	MILLER	10	7782

FIGURE 5.2
The Results window showing the rows from the query.

The Query window now contains both the EMP and DEPT tables, with the relationship between the two symbolized by a line. Deselect the DEPTNO column (so it is unchecked) and select the DNAME column instead. Execute the query and the DEPTNO column is replaced with the department names for each employee.

The MGR column in the report refers to the employee number of the manager for each employee. Notice that the employee named KING does not have a manager. Instead, KING's MGR column is left NULL or blank. It is important to make sure that KING appears in the final report, and that takes some modifications to the standard query lookup performed earlier.

Changing MGR from an employee number reference to the actual name of the manager requires adding the EMP table to the report for a second time. This is done in the same way the DEPT table was added. Notice that the second instance of the EMP table carries an A1 suffix in the table heading. Query Builder uses this accompanying suffix to differentiate between different instances of the same table. If you choose, you can change the assigned default name to something more descriptive.

Also notice that there is no relationship defined between the two EMP tables. Create one by choosing the Data, Set Table Relationship menu option. Doing so brings up the definition form shown in Figure 5.3.

The A-> prompt is used to create the relationship in the main EMP table. EMP.MGR is the value to type in . This refers to the manager's employee number in the table used to drive the query. Next, specify the join condition. As in most cases, the = (equal sign) is used. The B-> prompt is used to set up the lookup value and should be EMP_A1.ENAME. As you can see, the _A1 differentiates the second EMP table.

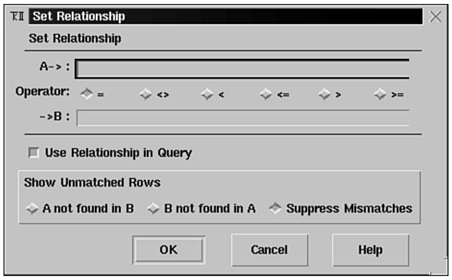


FIGURE 5.3
The dialog used to define table relationships.

Creating Table Relationships

The next value in the Set Relationship dialog is whether or not to use the relationship in the query. You can generally accept the default, which is to include the relationship. Should there be a reason not to include the relationship, you can exclude it at will.

Lastly, you have the option of including rows with no corresponding lookup values. Accepting the default of Suppress Mismatches would eliminate KING from the query results. Therefore, you should choose A not found in B to make sure that KING is included. This creates an outer join, as indicated by the + (plus sign) in the relationship line shown in Figure 5.4.

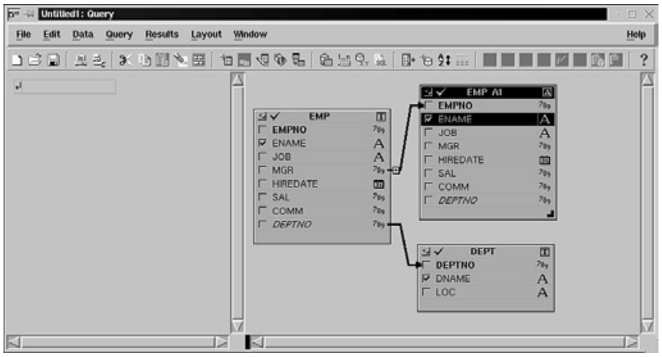
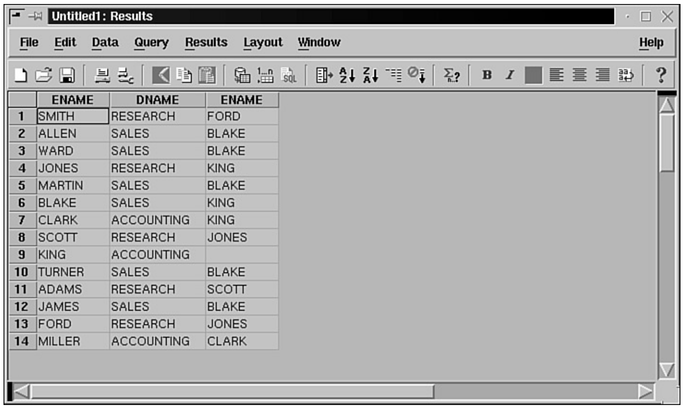


FIGURE 5.4
The two EMP tables shown joined together in the Query window.

Ordinary, or inner joins, discard all rows that don't have matching values in the other table being joined. An outer join makes sure that rows with no corresponding lookup values are included in the result. If the B not found in A option had been chosen, the plus sign, or outer join indicator, would appear next to the EMP_A1 table.

The query no longer needs to include the MGR column, but instead should include the ENAME column from the EMP_A1 table. Deselect MGR in EMP and select ENAME in EMP_A1, as is also shown in Figure 5.4. When that is done, execute the query; the results should be similar to those in Figure 5.5.



	ENAME	DNAME	ENAME
1	SMITH	RESEARCH	FORD
2	ALLEN	SALES	BLAKE
3	WARD	SALES	BLAKE
4	JONES	RESEARCH	KING
5	MARTIN	SALES	BLAKE
6	BLAKE	SALES	KING
7	CLARK	ACCOUNTING	KING
8	SCOTT	RESEARCH	JONES
9	KING	ACCOUNTING	
10	TURNER	SALES	BLAKE
11	ADAMS	RESEARCH	SCOTT
12	JAMES	SALES	BLAKE
13	FORD	RESEARCH	JONES
14	MILLER	ACCOUNTING	CLARK

FIGURE 5.5
The results of the sample query with employee names replacing the employee number references for manager, and department names replacing the department numbers.

Saving the Query

Save the query so that it can be used for later exercises. Selecting the File, Save menu option calls the Save dialog, which asks for the format in which to save the query. If you installed the Query Builder tables in the database, you have the option of saving there as well as two different formats in the file system. The first file system saves the query in a data format, whereas the second option, QFX, stores it as text. The text save uses name-value pairs to describe the query. Try saving it in both formats using names such as ch5q1 for the file system save and ch5q2 for the QFX save.

You also have the option of saving the query in SQL. This is done by choosing the File, Save SQL menu choice. Although you can view the SQL in the Query window with the Query, Show SQL menu choice, saving the query as a SQL statement allows you to import it into other applications easily.

The Query Window Toolbar

The wide toolbar hints at a large amount of functionality available in the Query Builder. Some buttons invoke the tools used in the previous examples, whereas other help narrow down query