

*Oracle JDeveloper for Database  
Developers and DBAs*

*An Oracle White Paper  
January 2005*

# *Oracle JDeveloper for Database Developers and DBAs*

Introduction .....	3
Online Database Development .....	3
Managing Connections .....	3
Database Browsing .....	4
Schema Object Creation .....	4
Executing Statements in SQL Worksheet .....	6
Registering Third-Party Drivers .....	6
Offline Schema Modeling .....	7
Generating SQL Scripts .....	8
PL/SQL Development .....	8
Running PL/SQL Procedures, Functions, and Packages .....	8
Editing PL/SQL in the Code Editor .....	9
Getting More Detail from the Navigator and Structure Windows .....	10
PL/SQL Debugging .....	11
PL/SQL Debugging Features .....	11
Additional PL/SQL Debugging Features .....	12
Setup Requirements for PL/SQL Debugging .....	13
Remotely Debugging PL/SQL .....	13
Conclusion .....	14

## INTRODUCTION

Oracle JDeveloper 10g is an Integrated Development Environment providing end-to-end support for building, testing, and deploying J2EE applications, Web services, and PL/SQL. JDeveloper 10g is written entirely in Java, and, as such, is a multi-platform development environment. Currently, JDeveloper is tested and supported on Windows, Linux, Mac OS/X, Solaris, and HP/UX.

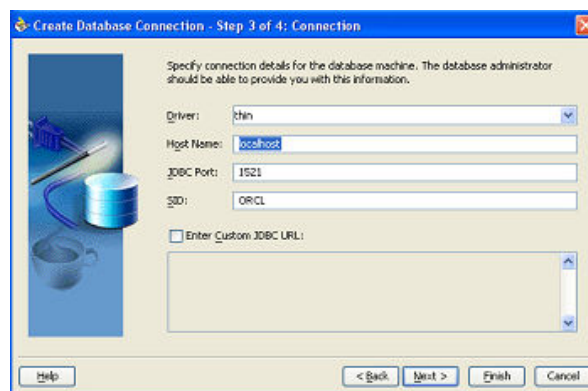
The two main areas this paper describes are the aspects of direct connection to a database schema, often referred to in this context as “working online”, and modeling schemas and database structures, in this context referred to as “working offline”.

This paper describes the features for PL/SQL development, working online, and general database development both online and offline, available in Oracle JDeveloper 10g. The many JDeveloper features for building applications that access the database (for example, Oracle ADF Business Components, publishing PL/SQL as a Web service, etc) are not in the scope of this document.

## ONLINE DATABASE DEVELOPMENT

### Managing Connections

JDeveloper allows you to create stored database connections using a simple wizard interface. Use these connections to browse the database, create schema objects, execute and tune ad-hoc SQL statements, identify connections in other wizards, and deploy server-side code.

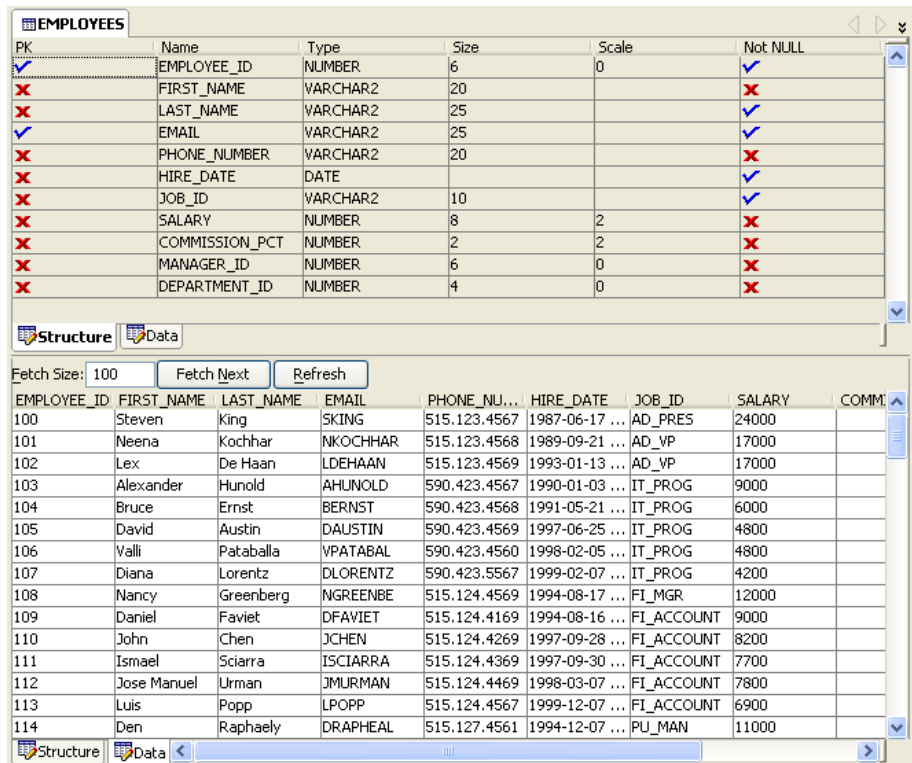


**Figure 1. Managing Connections**

## Database Browsing

Using a database connection created in the Wizard, you can browse through the objects in a database. The database browser is implemented with JDBC and thus allows you to browse any database with a JDBC driver. Browsing an Oracle database yields access to additional database objects not available with other databases.

One of the key aspects of browsing the database is the Table Viewer. In the Table Viewer, you can see at a glance details about columns and constraints for a table, as well as quickly retrieve the data from the table for further examination. The query is done in a scalable manner where (by default) just the first 100 rows are fetched, and you can manually fetch subsequent rows if needed.



The screenshot shows the Table Viewer for the EMPLOYEES table. The top section displays the table structure with columns: PK, Name, Type, Size, Scale, and Not NULL. The bottom section displays the data for the first 100 rows, with columns: EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, EMAIL, PHONE\_NU..., HIRE\_DATE, JOB\_ID, SALARY, and COMM.

PK	Name	Type	Size	Scale	Not NULL
✓	EMPLOYEE_ID	NUMBER	6	0	✓
✗	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	✗
✗	COMMISSION_PCT	NUMBER	2	2	✗
✗	MANAGER_ID	NUMBER	6	0	✗
✗	DEPARTMENT_ID	NUMBER	4	0	✗

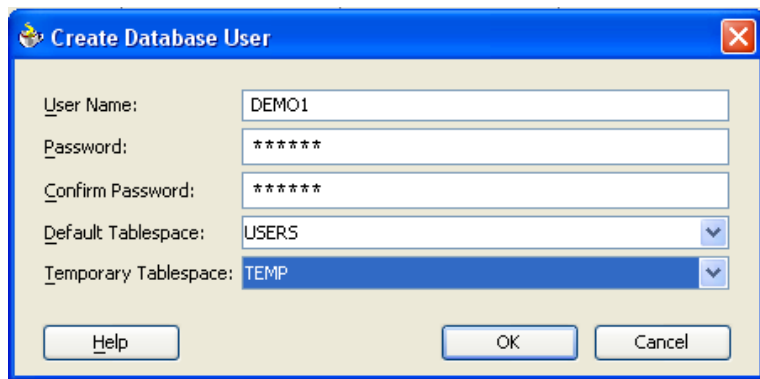
  

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NU...	HIRE_DATE	JOB_ID	SALARY	COMM.
100	Steven	King	SKING	515.123.4567	1987-06-17 ...	AD_PRES	24000	
101	Neena	Kochhar	NKOCHHAR	515.123.4568	1989-09-21 ...	AD_VP	17000	
102	Lex	De Haan	LDEHAAN	515.123.4569	1993-01-13 ...	AD_VP	17000	
103	Alexander	Hunold	AHUNOLD	590.423.4567	1990-01-03 ...	IT_PROG	9000	
104	Bruce	Ernst	BERNST	590.423.4568	1991-05-21 ...	IT_PROG	6000	
105	David	Austin	DAUSTIN	590.423.4569	1997-06-25 ...	IT_PROG	4800	
106	Valli	Pataballa	VPATABAL	590.423.4560	1998-02-05 ...	IT_PROG	4800	
107	Diana	Lorentz	DLORENTZ	590.423.5567	1999-02-07 ...	IT_PROG	4200	
108	Nancy	Greenberg	NGREENBE	515.124.4569	1994-08-17 ...	FI_MGR	12000	
109	Daniel	Faviet	DFAVIET	515.124.4169	1994-08-16 ...	FI_ACCOUNT	9000	
110	John	Chen	JCHEN	515.124.4269	1997-09-28 ...	FI_ACCOUNT	8200	
111	Ismael	Sciarra	ISCIARRA	515.124.4369	1997-09-30 ...	FI_ACCOUNT	7700	
112	Jose Manuel	Urman	JMURMAN	515.124.4469	1998-03-07 ...	FI_ACCOUNT	7800	
113	Luis	Popp	LPOPP	515.124.4567	1999-12-07 ...	FI_ACCOUNT	6900	
114	Den	Raphaely	DRAPHEAL	515.127.4561	1994-12-07 ...	PJ_MAN	11000	

Figure 2. Browsing the database with the Table Viewer

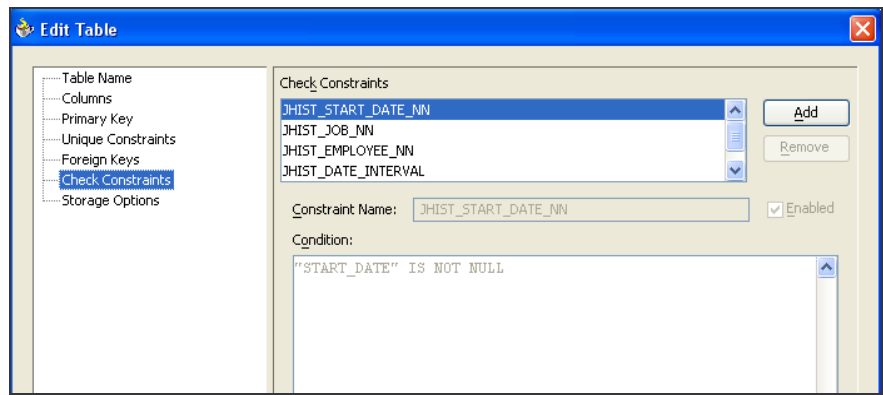
## Schema Object Creation

JDeveloper supports the creation of any schema object by executing a SQL statement in the SQL Worksheet. Assistant tools are provided for a subset of schema object types. JDeveloper provides assistant tools for creating database users and PL/SQL programs (procedures, functions, packages) in addition to tables, views, and triggers. View support in JDeveloper 10g production allows you to enter a SQL query. In the next release of JDeveloper, there is also support for declarative view creation.



**Figure 3. Support for Database User Creation**

Object	Description
User	Creates a new user. You can specify user details, such as user name, password, default tablespace and temporary tablespace.
PL/SQL Procedure	Creates a new procedure. You can edit and debug the procedure in the Code Editor.
PL/SQL Function	Creates a new function. You can edit and debug the function in the Code Editor.
PL/SQL Package and Package Body	Creates a new package or package body. You can edit and debug these in the Code Editor.
Table	Creates a new table. You can specify columns, constraints, and storage parameters.
View	Creates a new view. You can specify SQL statements for view, alias clause, and other view options.
Trigger	Creates a new trigger. You can edit the trigger in the Code Editor.

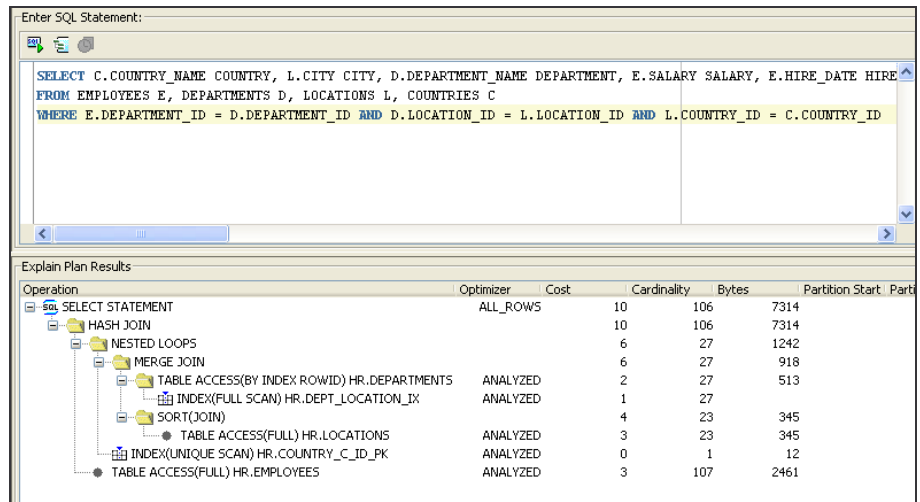


**Figure 4. Editing an Online Table Definition**

The next release of JDeveloper also supports declarative view creation. In addition, you will be able to create objects such as synonyms and sequences.

### Executing Statements in SQL Worksheet

JDeveloper 10g supplies a SQL Worksheet for executing and tuning SQL statements. It provides quick access to commands to either execute or get the explain plan for the selected statement. Previously executed statements can be recalled using the History button.



**Figure 5. Viewing the Explain Plan of a Statement in the SQL Worksheet.**

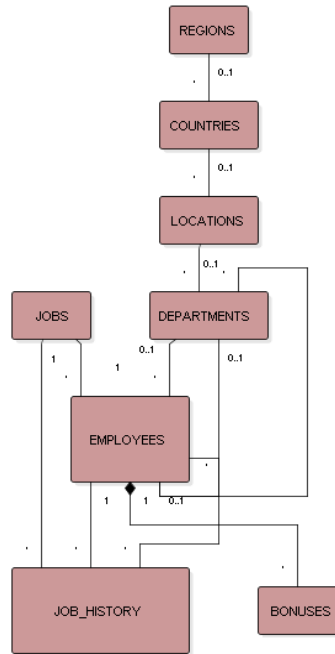
### Registering Third-Party Drivers

Accessing a data source using third-party drivers is simplified by registering drivers. Once registered these drivers are easily reused. You can register and use these drivers from the database connections wizard. Once created you can also manage these drivers from a central location (**Tools > Preferences, Database Connections**).

Note that this is intended mostly for Type IV (all-Java) drivers. You may wish to register a Type II driver in the same way to take advantage of the library creation and reusable connection name. Read more in the JDeveloper help on how to set these up.

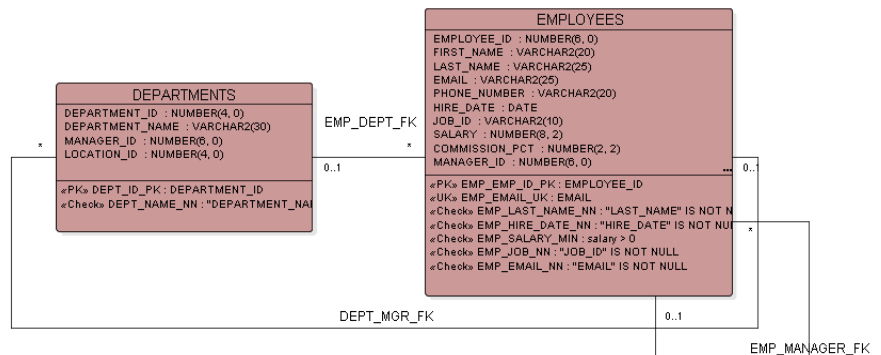
## OFFLINE SCHEMA MODELING

In JDeveloper 10g, you can quickly and easily visualize your database schema by creating a database diagram in a project and dragging a selection of tables from your database connection to the diagram.



**Figure 6. A Visual Representation of a Selection of Tables**

By setting display properties you can add more or less detail to the diagram. In addition you can fine-tune the layout according to your needs.



**Figure 7. Display Table Detail on a Schema Diagram**

## Generating SQL Scripts

Once you have your schema in an offline project, you can easily edit and add to it using the wizards provided or by using elements in the database component palette. The DDL you can generate can either be to create new database objects, for a new schema, or for updating an existing schema. In this latter case, you need to supply a connection so that the DDL Generator can read the database schema to provide ALTER table scripts.

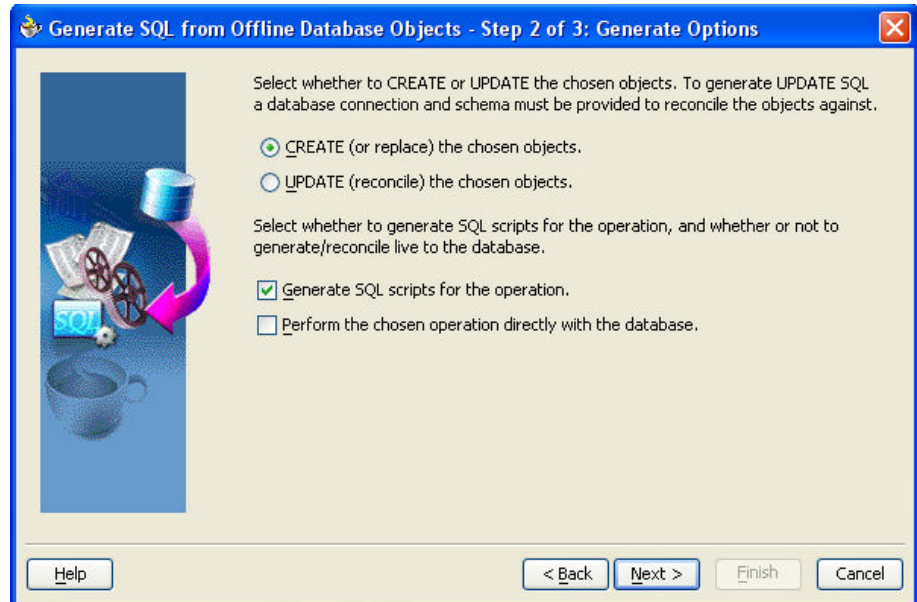
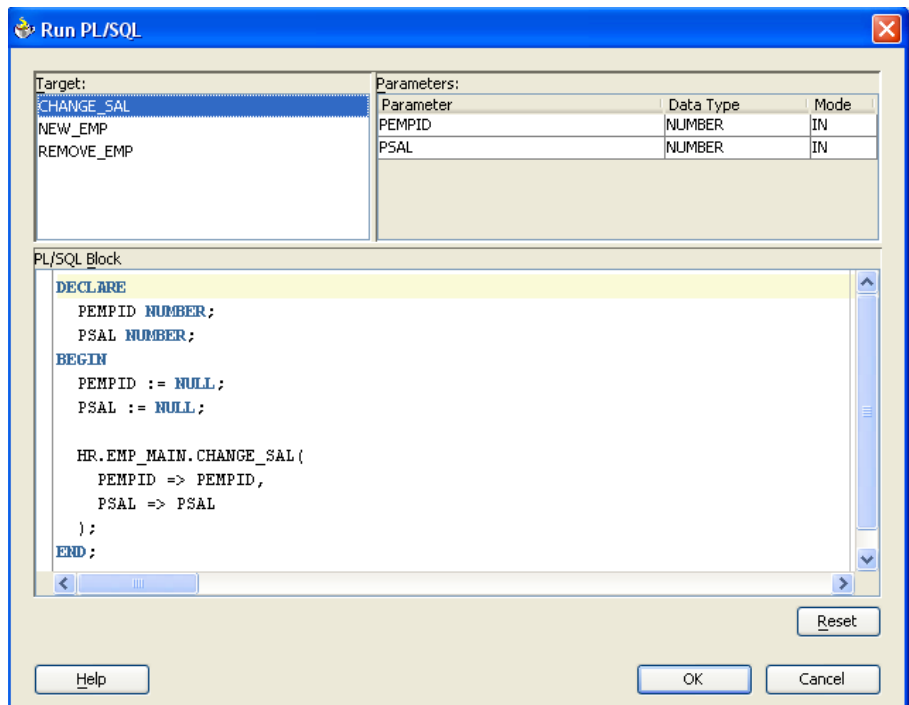


Figure 8. DDL Generate Options Available

## PL/SQL DEVELOPMENT

### Running PL/SQL Procedures, Functions, and Packages

You can run PL/SQL procedures, functions, and packages by simply right clicking on the object in the Navigator and choosing **Run <plsql\_object\_name>**. You will be presented with a dialog showing details about the arguments and, for functions, return values for the selected object. If the selected object is a package, the dialog will display a list of the procedures and functions defined in the package spec. You can select one of these procedures or functions as the target you want to run.



**Figure 9. Running a Function in a Package**

When you invoke the Run PL/SQL dialog, code is automatically generated to call the target PL/SQL program unit. You can modify this code directly in the dialog to initialize and pass parameters. Additionally you can save the modified code for reuse in subsequent runs of that program. (Note, available in the production version.)

When you run a PL/SQL program that makes calls to `DBMS_OUTPUT`, the results will be displayed in the Log window. Likewise, return values from functions, and values of OUT parameters are also displayed in the Log window.

### Editing PL/SQL in the Code Editor

JDeveloper includes a full-featured editor for PL/SQL program units, including customizable PL/SQL syntax highlighting in addition to common editor functions such as Bookmarks, Macros, Code Templates, Search and Replace, etc. PL/SQL Code Insight is also available from the Editor. For example, if you type `DBMS_OUTPUT`. followed by **Ctrl+Space**, you can select from a list of members of that package. Similarly, when typing a SQL statement, you can type `EMP`. followed by **Ctrl+Space** to invoke a list of columns in that table. Note that by default, Code Insight will be invoked automatically (without pressing **Ctrl+Space**) if you pause after typing a period (".") for more than one second.

When using the Code Editor to edit PL/SQL code, you can Make, Rebuild, or Save (all of these actions have the same result for PL/SQL) your work. This sends the source code to the database for recompilation and storage of the program. Any syntax errors encountered during compilation will be displayed in the Log window.

You can navigate to the source of a syntax error by double-clicking the error message in the Log window.

```

SELECT l.state_province, l.country_id, d.department_name, e.last_name,
       j.job_title, e.salary, e.commission_pct
FROM locations l, departments d, employees e, jobs j
WHERE l.location_id = d.location_id
AND d.department_id = e.department_id
AND e.job_id = j.job_id;
emp_record emp_cursor%ROWTYPE;
TYPE emp_tab_type IS TABLE OF emp_cursor%ROWTYPE INDEX BY BINARY_INTEGER;
emp_tab emp_tab_type;
i NUMBER := 1;
BEGIN
OPEN emp_cursor;
FETCH emp_cursor INTO emp_record;
emp_tab(i) := emp_record;
DBMS_OUTPUT.PUT_LINE(' ');
WHILE emp_cursor%FOUND LOOP
  i := i + 1;
  FETCH emp_cursor INTO emp_record;
  -- Fetch department information
  emp_tab(i).deptrec := emp_record;
  GET_LINE INTO deptrec.department_id, deptrec.department_name, deptrec.job_id;
  GET_LINES INTO emp_tab(i).last_name, emp_tab(i).job_title;
  NEW_LINE;
  PUT;
  PUT_LINE(emp_tab(i).last_name);
END LOOP;
END;

```

Figure 10. Editing PL/SQL in the Code Editor

### Getting More Detail from the Navigator and Structure Windows

The Navigator and Structure windows in JDeveloper provide additional information that can be useful for PL/SQL developers. For example, invalid PL/SQL objects are shown in the Navigator with a red "X" overlay icon.

For some database object types, the Structure window displays additional information about the currently selected object. The following table identifies the additional details provided by the Structure window for various object types.

Navigator Node/Object Type	Details in Structure Window
Connection Node	Shows the connection type, driver name, user, and URL for the selected connection.
Table, View, or Synonym for a Table or View	Shows the columns and indexes for the selected table.
PL/SQL Procedure or Function	Shows the arguments for the procedure or functions, any embedded procedures and

functions, and variables defined in the program unit. Note that double-clicking on an element in the Structure window will navigate you to the appropriate location in the source code.

If you pause while working on a PL/SQL program unit, JDeveloper's PL/SQL parser will attempt to parse the file. Any syntax errors the parser detects will be displayed in the Structure window in a folder named Errors.

PL/SQL Package or Package Body Shows the procedures, functions, and variables defined in the selected program unit. Arguments of the procedures and functions are also displayed. Note that double-clicking on an element in the Structure window will navigate you to the appropriate location in the source code.

If you pause while working on a PL/SQL program unit, JDeveloper's PL/SQL parser will attempt to parse the file. Any syntax errors the parser detects will be displayed in the Structure window in a folder named Errors.

Java Stored Procedure (or other deployed Java Class) Show the details of the Java class, including package name, imports, methods, and members of the class. Note that double-clicking on an element in the Structure window will navigate you to the appropriate location in the source code.

## PL/SQL DEBUGGING

JDeveloper 10g provides full support for PL/SQL debugging with Oracle8i, Oracle9i and Oracle10g databases.

### PL/SQL Debugging Features

Highlights of the PL/SQL debugging features include:

- **Control program execution:** The PL/SQL debugger in JDeveloper supplies many commands to control program execution including **Step Into**, **Step Over**, **Step Out**, **Run to Cursor**, **Pause**, **Resume**, and **Terminate**.
- **View and modify variables:** While the debugger is paused, you can examine and modify the values of variables from the Smart Data, Data,

Watches or Inspector windows. For PL/SQL collections, you can adjust the range of visible values in the debugger.

- **Customizable breakpoints:** JDeveloper breakpoints are highly configurable. For PL/SQL debugging, you can use source breakpoints (associated with a particular line of executable code in a particular program unit) and exception breakpoints (associated with any unhandled exception, or a specific Oracle exception). You can define conditional breakpoints for PL/SQL and customize the action of breakpoints, for example, pause the debugger (default), beep, log occurrence to a text file, or enable or disable other breakpoints.

PL/SQL debugging Information is available from several windows in JDeveloper. The following list provides examples of the kind of information available during debugging.

- **Code Editor:** Shows the execution point. Flyover tool tips display the name and value of the variable under the pointer.
- **Breakpoints window:** Lists the defined breakpoints. You can use this window to add new breakpoints, or customize the behavior of existing breakpoints.
- **Data window:** Displays all variables that are currently in scope, including package variables, package body variables, variables declared in the current procedure or function, and local variables (such as those declared in a nested block or implicitly declared).
- **Smart Data window:** Displays only the variables referenced in the line of code about to be executed and in the previous two locations. Note that this is customizable. For example, you might want the Smart Data window to show the variables used in the line of code about to be executed and the next two lines, and to retain variables used in the previous four locations.
- **Watches window:** Displays expressions or variables you've added to the Watches window by either selecting Add Watch from the context menu of the window and entering the expression, or by dragging a variable from one of the other windows (such as the Data or Smart Data windows) to the watches window.
- **Inspector windows:** Display expressions or variables you've added to the Inspector window. This is similar to the Watches window, except that Inspector windows float by default and only one expression is displayed in each Inspector window.
- **Stack window:** Shows the execution stack. Note that you can use the Stack window to change the context in the debugger.
- **Classes window:** Shows a list of PL/SQL programs and Java classes that have been loaded in this database session.

### Additional PL/SQL Debugging Features

PL/SQL debugging in JDeveloper is implemented in two distinct ways. For debugging against database versions prior to Oracle9i Release 2 (Oracle 9.2), JDeveloper uses the DBMS\_DEBUG API provided by the server. Starting with Oracle9i Release 2, JDeveloper utilizes the new JDWP (Java Debugging Wire Protocol) implementation provided by the server. JDeveloper automatically detects

which version of the database you are using for debugging and uses the appropriate method for that version.

Because of the variations in the two debugging implementations, you will see some differences when debugging with specific database versions. JDeveloper is able to take advantage of several features that are provided only with the JDWP implementation in Oracle9i Release 2 and beyond:

- **Debugging Java stored procedures:** You can debug Java stored procedures and PL/SQL programs seamlessly. For example, if you have a PL/SQL procedure that calls a Java stored procedure, you can step into the Java stored procedure call from the PL/SQL procedure. Or, you can set a breakpoint in the Java stored procedure and then debug the PL/SQL procedure -- the debugger will pause when the breakpoint in the Java stored procedure is reached. Debugging Java stored procedures in database versions prior to Oracle9i Release 2 is not possible with JDeveloper.
- **PL/SQL collections:** The DBMS\_DEBUG API has limited support for PL/SQL collections, such as tables, records, and cursors. Using JDeveloper to debug PL/SQL in an Oracle9i Release 2 database, you have complete access to composite PL/SQL structures. For example, if your PL/SQL program uses a PL/SQL table of records, you can expand the table object in the Data window to see the records, and then expand a record to see the fields, then select a field and modify its value on the fly. It is possible to access composite PL/SQL structures in database versions prior to Oracle9i Release 2, but you must manually enter the fully qualified name of the element in the Watches window, for example, for a table named "tab" containing records with a field named "field", tab(4).field.
- **Debugging remotely:** With the JDWP implementation, JDeveloper is able to leverage its remote debugging capabilities with server-side code. Remotely debugging PL/SQL involves starting the JDeveloper debugger listener, then attaching to that listener via the database session you want to debug. (See the section Remotely Debugging PL/SQL below.) Remotely debugging PL/SQL in database versions prior to Oracle9i Release 2 is not possible with JDeveloper.

### **Setup Requirements for PL/SQL Debugging**

To enable PL/SQL debugging, several conditions must be met. See the JDeveloper help for more detail.

### **Remotely Debugging PL/SQL**

Using JDeveloper to locally debug PL/SQL entails setting a breakpoint where you want the debugger to pause, then selecting a PL/SQL procedure, function, or package in the Navigator and pressing the Debug button. In this case, JDeveloper starts the debugging session, connects to it, and pauses when the breakpoint is reached, all without your intervention. Another way to think of it is that with local debugging, JDeveloper is the client that initiates debugging.

Using JDeveloper and Oracle9i Release 2 and beyond, you can also remotely debug PL/SQL. Remotely debugging PL/SQL means that you initiate the debug action from a client external to JDeveloper, for example, a PL/SQL web application, an

OCI program, a SQL\*Plus session, etc. In this case, you must manually perform some steps that JDeveloper would otherwise do for you.

## **CONCLUSION**

Oracle JDeveloper 10g is an Integrated Development Environment providing end-to-end support for building, testing, and deploying J2EE applications, Web services, and PL/SQL. JDeveloper 10g is written entirely in Java, and, as such, is a multi-platform development environment. As such JDeveloper is often not considered by application developers as a tool for database work. In this paper we have tried to illustrate a few of the database features provide by JDeveloper 10g to support both online and offline development. PL/SQL support includes creating, editing and debugging code. JDeveloper's offline database support gives users the ability to create schema diagrams, modify schema objects and update the database by executing SQL scripts against existing schemas.

Future releases of JDeveloper see ongoing additions to the list of objects that users can create online and offline. In the first release beyond JDeveloper 10g a significant addition to the database support is the ability to declaratively create database views offline diagrammatically or through wizards and dialogs. For further information on the new functionality coming to JDeveloper see <http://www.oracle.com/technology/products/jdev/index.html>



White Paper Title  
January 2005  
Author: Sue Harper  
Contributing Authors: Brian Fry

Oracle Corporation  
World Headquarters  
500 Oracle Parkway  
Redwood Shores, CA 94065  
U.S.A.

Worldwide Inquiries:  
Phone: +1.650.506.7000  
Fax: +1.650.506.7200  
[www.oracle.com](http://www.oracle.com)

Copyright © 2003, Oracle. All rights reserved.

This document is provided for information purposes only  
and the contents hereof are subject to change without notice.

This document is not warranted to be error-free, nor subject to  
any other warranties or conditions, whether expressed orally  
or implied in law, including implied warranties and conditions of  
merchantability or fitness for a particular purpose. We specifically  
disclaim any liability with respect to this document and no  
contractual obligations are formed either directly or indirectly  
by this document. This document may not be reproduced or  
transmitted in any form or by any means, electronic or mechanical,  
for any purpose, without our prior written permission.

Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective owners.