ORACLE®

White Paper

Procedure Builder: Unified Client/Server Development and Application Partitioning

Part Number A32502

Procedure Builder: Unified Client/Server Development and Application Partitioning

Introduction

Oracle offers the only unified client/server development environment on the market, Procedure Builder. Version 1.5 of Procedure Builder is a highly productive client/server development tool, providing a powerful GUI environment for developing both client-side *and* server-side application logic. Developers can visualize the structure of their entire application. They can view and edit all objects, whether on the client or server, using a unified set of editors. It is trivial to reconfigure an application using Procedure Builder's drag and drop client/server partitioning.

Oracle supports a single programming language on both sides of the client/server equation, *PL/SQL*. PL/SQL is the standards-based programming language[1] used in the Oracle RDBMS for writing database logic. PL/SQL is also the same programming language used for writing application logic in Oracle's Developer/2000 product suite. The primary advantages of using a unified client/server language are:

- Learn a single programming language for all client/server development
- Easily reuse code, whether it originates on the client or server side
- Easily partition applications between client and server
- Reduce network traffic

Procedure Builder includes facilities for browsing, editing, compiling, executing, debugging, and partitioning PL/SQL. This paper discusses the use of Procedure Builder as a client/server development environment and as a powerful tool for application partitioning.

PL/SQL Architecture Overview

PL/SQL logic consists of a collection of *program units*. As shown in Figure 1 below, program units are classified into subprograms (functions and procedures), packages, and blocks.

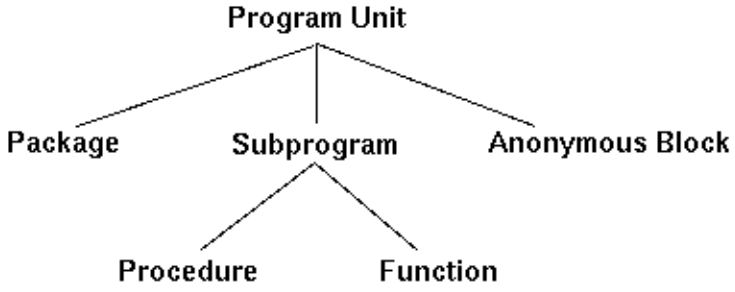

Figure 1

[1] "PL/SQL" stands for **P**rocedural **L**anguage extensions to standard **SQL** (and not some hybrid of the PL/I programming language). These language extensions are modeled on the ANSI Ada standard.

Subprograms can be executed via an explicit reference from a calling program unit. For example, the block below explicitly calls the subprogram named "sample", passing it two arguments:

```
begin
  sample(1, 'two');
end;
```

Some blocks and subprograms, known as *triggers*, are invoked implicitly by the application environment. A trigger executes, or "fires", automatically in response to a specific event, such as a mouse click in the client application or the insertion of a row on the server side.

Distributed PL/SQL application logic consists of both client-side and server-side program units. Client-side program units reside and execute in an application object such as a form, report, or chart designed using Developer/2000. Server-side, or *stored*, program units (including database triggers) reside and execute in an Oracle7 Server.

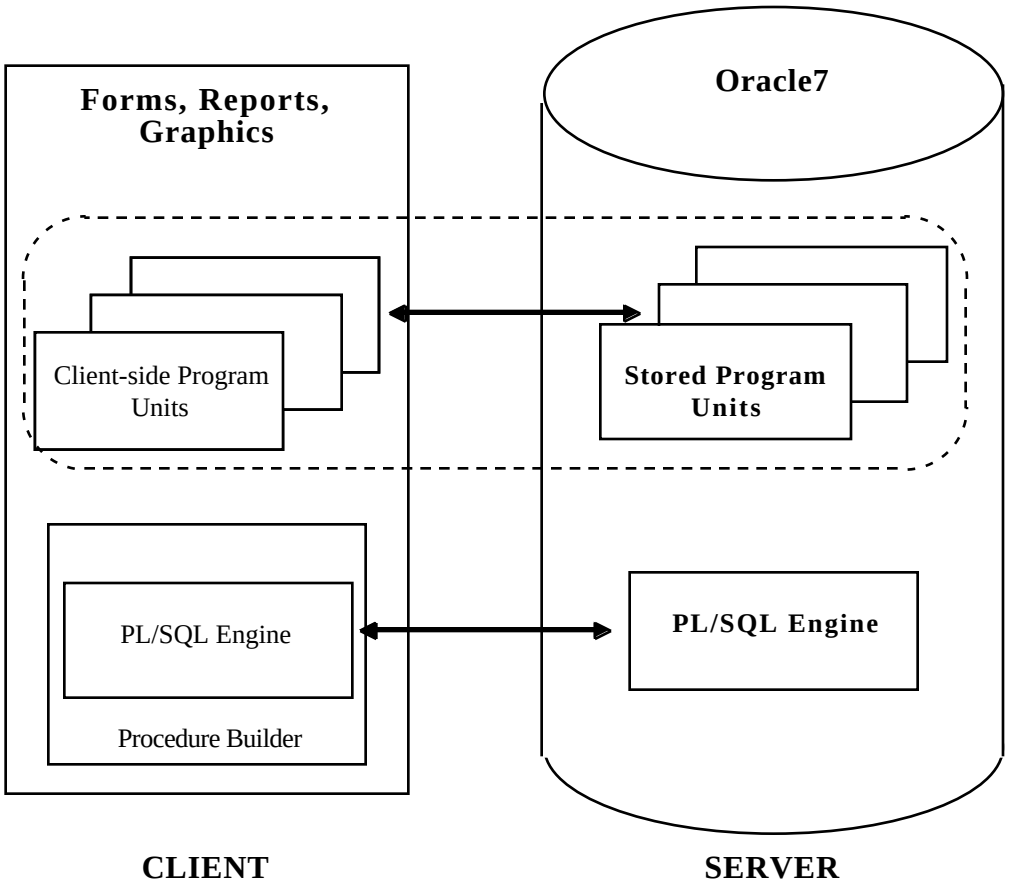

Figure 2

As shown in Figure 2, a PL/SQL compilation and runtime engine resides in both the client and the server. Procedure Builder sits on top of the client-side PL/SQL engine, either by itself or embedded within the client-side Developer/2000 Tools, such as Forms, Reports, and Graphics.

This architecture simplifies distributed application development by allowing you to use the same language at both ends of the database connection. It also enables the seamless *partitioning* of application logic — you can relocate program units without re-coding to the server from the client, or vice versa, with a simple drag and drop operation. For example, a program unit prototyped and debugged on the client side using Procedure Builder, can be dragged and dropped onto the database server where a stored procedure is automatically created with the same logic. Or conversely, a stored procedure can be dragged and dropped into the application. You can apply the same easy drag and drop manipulation into PL/SQL libraries for code reuse.

Integrated Tool for Incremental Development

Procedure Builder supports an *incremental* approach to distributed database application development. The incremental approach breaks the traditionally monolithic, multi-tool edit-compile-link-debug cycle into small interleaved steps performed within a single environment. Individual program units can be incrementally edited, compiled, linked, and debugged within the same tool and without disturbing the rest of the application. This approach makes application development more efficient and facilitates iterative development as well as rapid prototyping.

Browsing

Procedure Builder consolidates the browsing of application objects into a single, integrated window known as the *Object Navigator*. The Object Navigator presents a hierarchical, or outline, view of the objects and subobjects in an application, as well as the relationships between them (as shown in Figure 3).

Objects and relationships displayed in the Navigator include:

- Program units within an application object, a library, or database
- The subprograms of a package, along with their parameters
- Tables and views
- Triggers attached to tables
- Cross reference relationships between program units, tables, and application objects

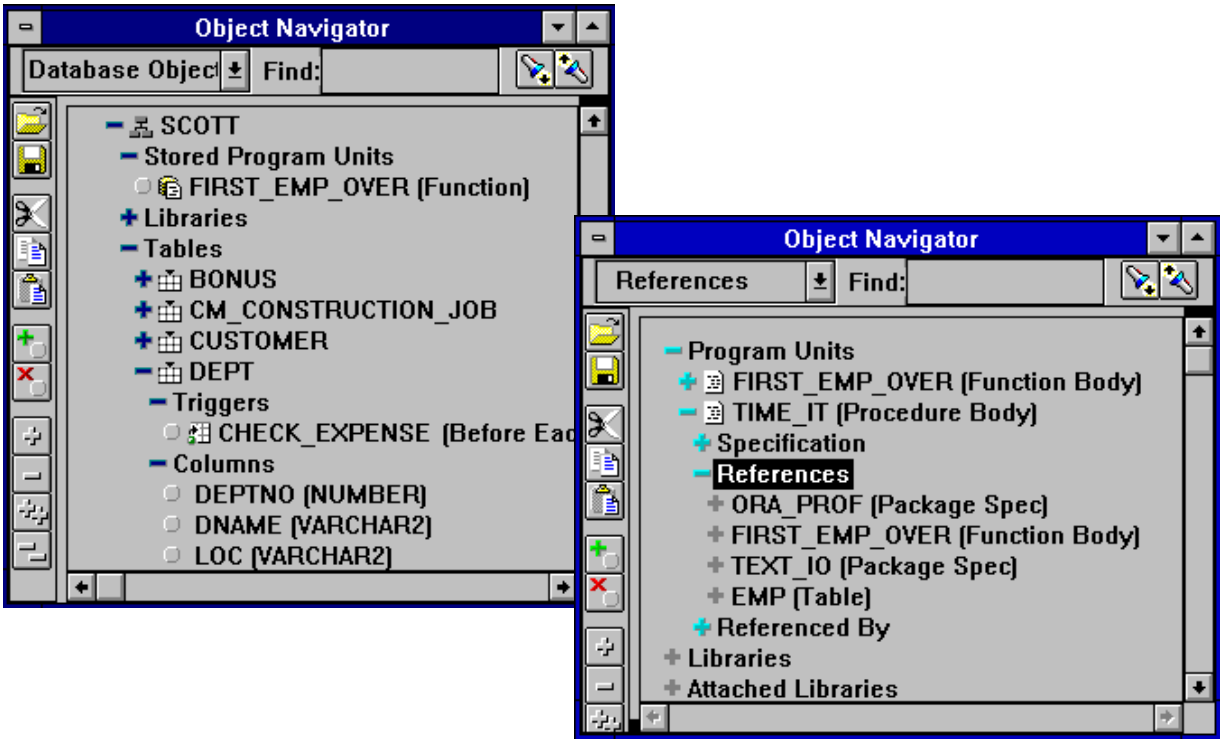

Figure 3

The Object Navigator is interactive. You can click on entries in the outline to show or hide (zoom in or zoom out on) successive levels of detail. You can also drag a subobject and drop it into a new container object. For example, you may wish to move a program unit from a form to a PL/SQL library where it can be more easily shared by others.

Editing and Compiling Client and Server Code

Double-clicking on a program unit in the Navigator launches one of three PL/SQL editors:

1. the *Program Unit Editor* for client-side program units
2. the *Stored Program Unit Editor* for server-side packages and subprograms
3. the *Database Trigger Editor* for server-side triggers

The Program Unit Editor (shown in Figure 4) supports editing source text, client-side incremental compilation, error message browsing, search and replace (both local and global), and quick navigation to other program units in the same scope.

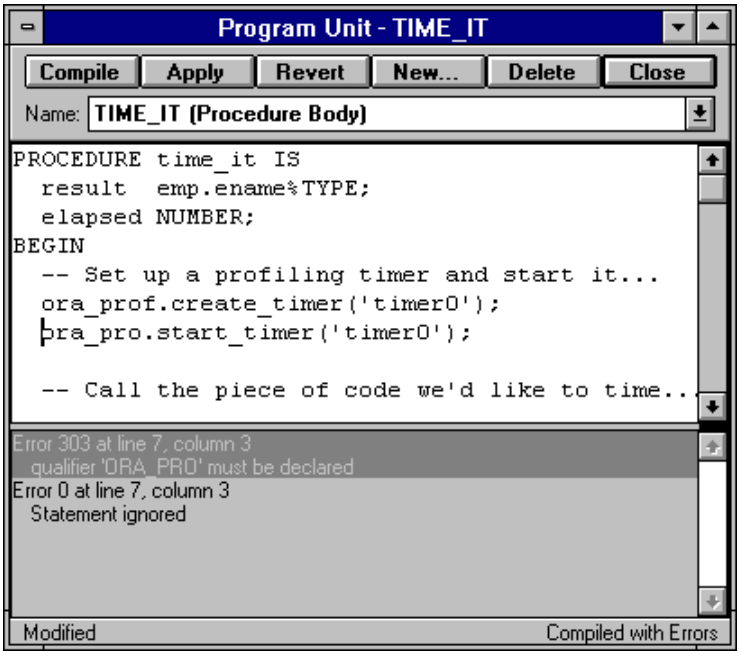

Figure 4

Incremental compilation allows you to change the *body* of a program unit without having to recompile any other program units that reference it. This permits very short edit-compile-debug cycles for individual program units. If you change the parameters of a stand-alone subprogram or any part of a package specification, Procedure Builder automatically invalidates all program units that referenced the modified program unit. You can then visit the invalidated program units (marked in the Object Navigator), make the necessary source changes, and recompile them. If no source changes are necessary (as is the case when a new subprogram is simply added to a package specification), you need not visit and manually recompile each invalidated program unit. Instead, Procedure Builder automatically recompiles the invalid program units on demand.

The Stored Program Unit Editor (shown in Figure 5) is a client-side GUI environment for editing server-side packages or subprograms. Visually and functionally, it is very similar to the client-side Program Unit Editor. However, the semantics of each operation are quite different — each maps to an appropriate SQL statement. For example, the compile operation submits the source text to the server-side PL/SQL engine via the appropriate CREATE OR REPLACE SQL statement.

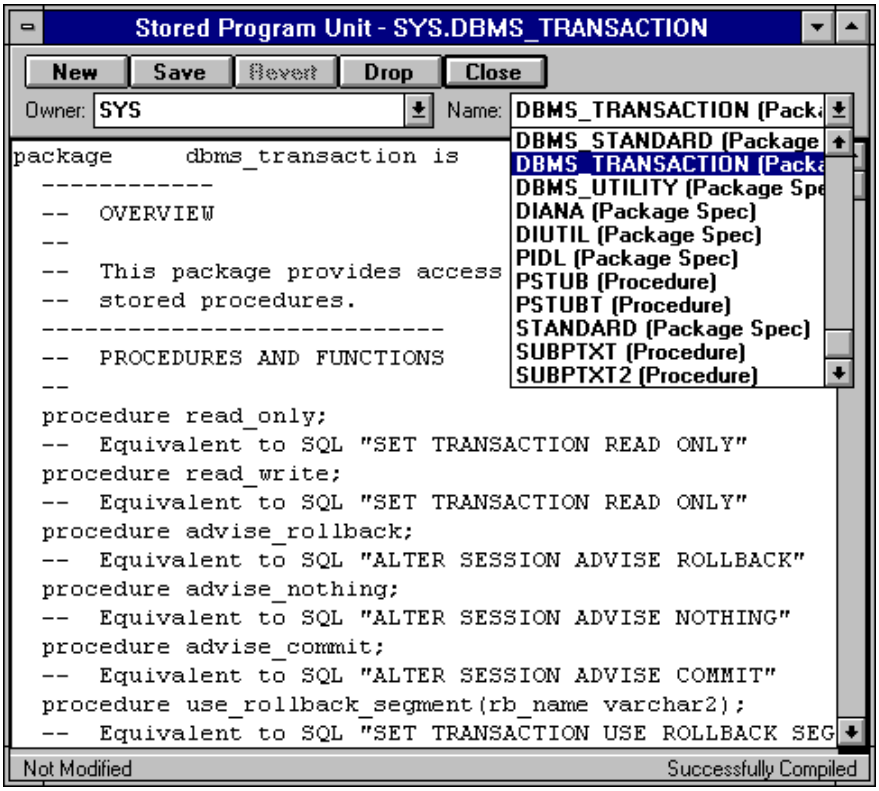

Figure 5

The Database Trigger Editor (shown in Figure 6) provides a client-side GUI to a server-side trigger. It presents the various trigger options via intuitively organized GUI controls and allows quick navigation between triggers associated with other users and tables.

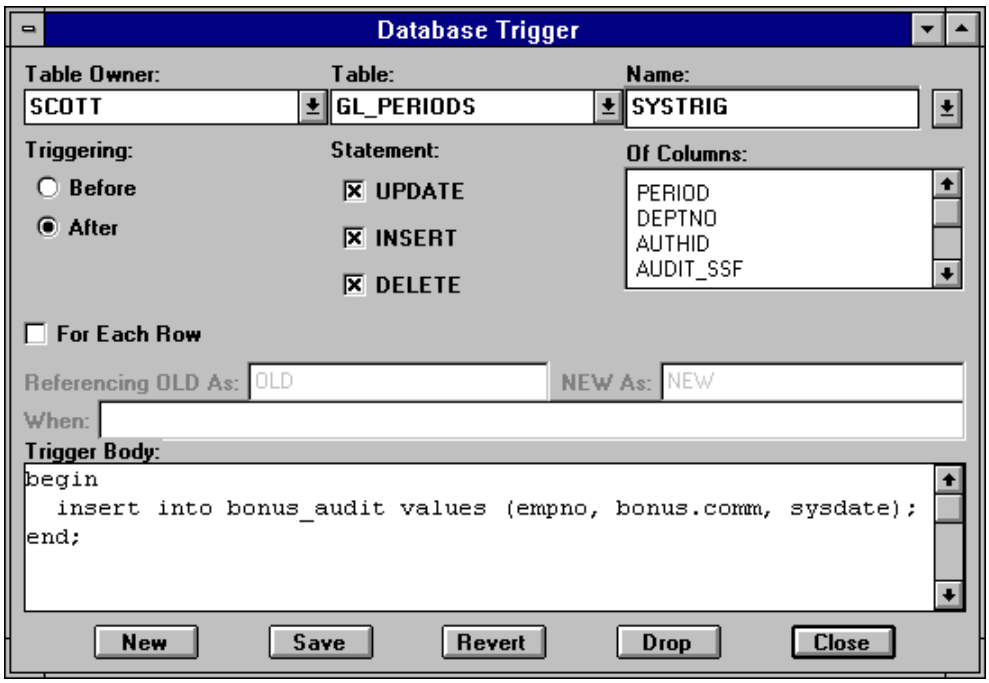

Figure 6

Source-Level Debugging

The Interpreter (shown in Figure 7) serves as the focal point for client-side, source-level debugging and real-time prototyping.

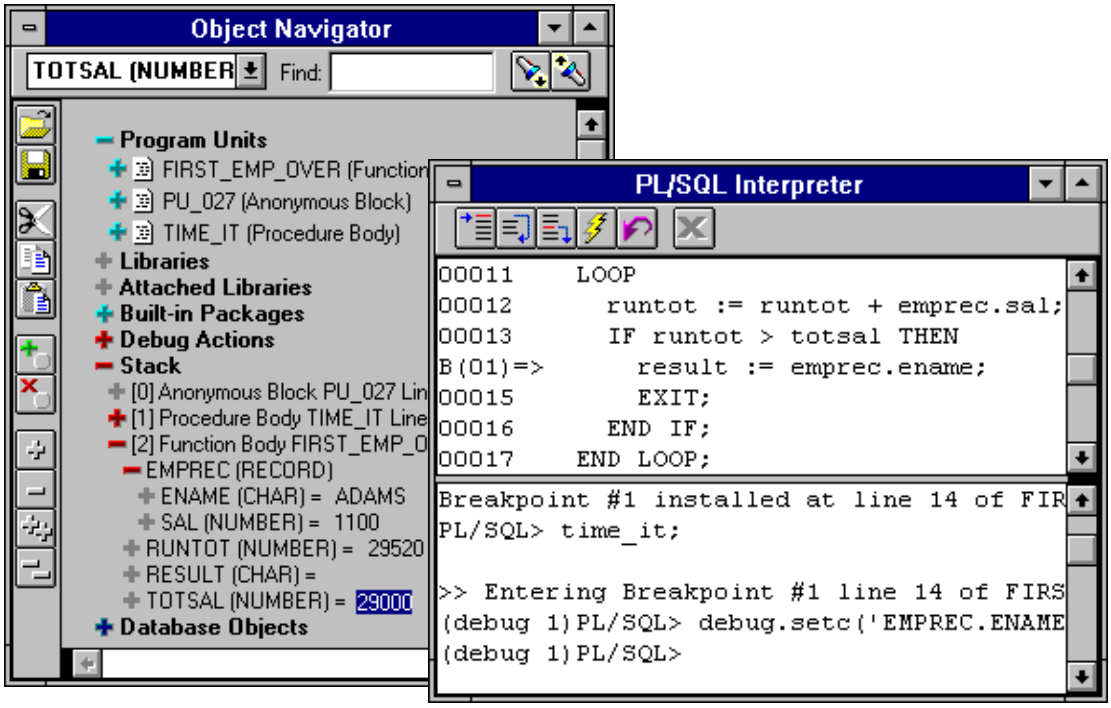

Figure 7

Between debugging sessions, the Source Pane of the Interpreter is synchronized with the Object Navigator and displays the source text of the currently selected client-side program unit node. Breakpoints can be added and removed by double-clicking on line numbers displayed in the margin. While debugging, the Source Pane tracks the current source location as you encounter breakpoints and step through your code. Simultaneously, the Object Navigator tracks the current call stack. Each frame of the call stack appears as an entry in the outline. You can click on an individual stack frame entry, expanding it to display the values of its associated parameters and local variables. You can edit these values directly in the Object Navigator.

The Interpreter Pane appears at the bottom of the Interpreter. It supports the interactive evaluation of PL/SQL constructs, SQL commands, and Procedure Builder commands. The Interpreter Pane is particularly handy for quick prototyping and testing of application logic. The following transcript of a simple Interpreter session illustrates some of its capabilities:

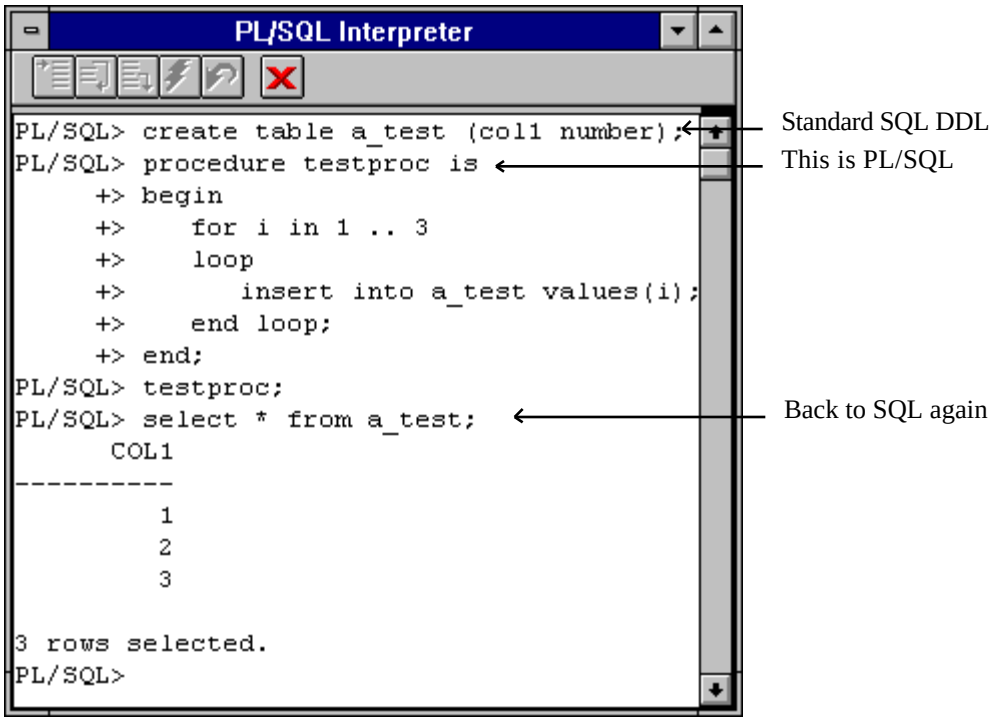

Figure 8

Note that Procedure Builder allows you to instantly transition between SQL, PL/SQL, and Procedure Builder commands[2] as suits your needs.

Application Partitioning

As mentioned above, Oracle's client/server architecture supports the same programming language on both sides of the database connection. Developers can take advantage of Procedure Builder's drag and drop functionality to transport PL/SQL program units between client and server, to effortlessly reconfigure distributed applications for scalable performance. . The transfer of application functionality (application partitioning) is a flexible and simple scheme to adequately address distributed application performance. (When Developer/2000 is used in conjunction with Designer/2000, distributed partitioned applications can be generated automatically.)

The use of stored procedures, which encapsulate multiple SQL operations into a single parameterized procedure stored directly in the database, can reap tremendous network performance gains for an application. But why would one ever want to move a program unit in the other direction — off of the server and onto the client? There are some reasons why one might wish to do this (at least temporarily):

- *Leverage client-side debugging* — To use the powerful source-level debugging features of the current version of Procedure Builder, the program unit must be on the client side. You can transport a stored program unit to the client side. In many cases, you can transport a stored program unit to the client side without disturbing its semantics (references to external stored subprograms will rebind on the client). Once it is local, you can debug and test it, and transport the corrected version back to the server.

- *Prototype against a non-Oracle database via ODBC* — Using Oracle's Open Client Adapter for ODBC, you can perform "disconnected" application prototyping against a light-weight, non-Oracle

[2] Whether entered interactively or read from stored scripts.

data source. To set up the prototype environment, you might download stored program units from an Oracle7 Server representing the production environment. In the prototype environment, all PL/SQL would be located on the client side. When you reconnect to the production environment, you can upload the prototyped stored program units back into the Oracle7 Server.

- *Prototype or debug without disturbing deployed stored program units* — Even when server-side debugging becomes available in Procedure Builder, you still may not want to debug or refine a stored program unit in place if it is part of a production system. In such cases, you may choose to copy the program unit over to the client side and debug it locally instead.

- *Prototype or debug without privileges* — You may not have the database privileges necessary to create or modify the stored program units that you are interested in. In this case, you can copy the program unit onto the client side and experiment locally.

Whatever your reasons, Procedure Builder makes it easy to move program units between the client and the server. In the Object Navigator, migration is performed interactively with a simple drag and drop operation. If program units are migrating regularly and systematically (as in the disconnected prototyping case mentioned above), you can automate the relocation process using scripts of Procedure Builder commands.

Summary

Version 1.5 of Procedure Builder is unique in offering:

- A unified client/server development environment
  - Partition logic between client and server with drag and drop
  - Code, test, and debug stored procedures or application code in the same environment
  - Quickly create and test code in the design-time interpreter
  - Debug code with a powerful debugger supporting breakpoints, call stack viewing, and the dynamic modification of parameter and local variable values
- The Object Navigator
  - Use the same interface to browse database schemas and application logic
  - Enjoy the easy-to-use outline metaphor that expands or collapses with a click of the mouse
  - View cross reference relationships between objects
  - Drag and drop objects for reuse or reorganization
- Incremental development
  - Edit, compile, link, and debug all within the same tool
  - Rapidly prototype program units without disturbing the rest of the application
  - Be more productive with incremental compilation that automatically determines the minimum number of units to recompile

Procedure Builder is a fully integrated tool for the incremental development of distributed database applications. It is the only true unified client/server development environment on the market. By exploiting its rich set of features, you can dramatically reduce your application development and maintenance costs.

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
USA

Worldwide Inquiries:
Phone: +1.415.506.7000
Fax +1.415.506.7200
http://www.oracle.com

To offer our customers the most complete and effective information management solutions, Oracle Corporation offers its products along with support, education, and consulting, in more than 90 countries.

Part #: A32502

Printed on recycled paper
with soy-based inks