

REVIEW:
By John J Dyer, CCP, CDP
PL/SQL Contract Developer
COMSYS
John.dyer@comsys.com

Oracle PL/SQL Developer's Workbook
Steven Feuerstein with Andrew Odewahn

May 2000
ISBN 1-56592-674-9
588 pages

O'Reilly Item #674-9, \$36.95

In my current assignment, I am working in Oracle PL/SQL on a Unix machine. When I saw the opportunity to review Feuerstein's *Developer's Workbook*, I jumped at it. I recommend this book to anyone who wants to further his or her skills in writing good Oracle PL/SQL code.

Steven Feuerstein is considered one of the world's leading experts on the Oracle PL/SQL language. Here is a list of his other Oracle books:

[Oracle PL/SQL Programming, 2nd Edition](#)
[Oracle Built-in Packages](#)
[Oracle PL/SQL Built-ins Pocket Reference](#)
[Oracle PL/SQL Language Pocket Reference](#)
[Oracle PL/SQL Programming Guide to Oracle8i Features](#)
[Oracle PL/SQL Best Practices](#)
[Advanced Oracle PL/SQL Programming with Packages](#)
[Mastering Oracle Power Objects](#)
[Oracle Distributed Systems](#)
[Oracle Security](#)

Feuerstein says that this workbook is designed to engage us actively, to provoke a response, to get us solving problems immediately with the techniques at hand. I am studying the book to deepen my understanding of, and facility with, the PL/SQL language. I could not pass up the opportunity to expand and enhance my personal toolkit by becoming more familiar with Procedural Language/SQL. The primary criticism I have of the book is that it does not stand-alone. To effectively use the Workbook, you need access to one of Feuerstein's other books, *Oracle PL/SQL Programming, 2nd Edition*, or an equivalent book such as the Osborne book entitled *Oracle 8: PL/SQL Programming*.

The *Workbook* is divided into two parts: problems and solutions. Feuerstein has separated the two so we could concentrate on the problems and come up with our own solutions. This separation also makes the book more useful in a classroom environment. I have skimmed through the entire book and am now going back and working all of the *Beginner* exercises. When I finish the *Beginner* exercises, I will complete the *Intermediate* and *Expert* exercises. If you wish to really understand Oracle PL/SQL, you should plan on spending a good bit of time with the workbook.

REVIEW:
By John J Dyer, CCP, CDP
PL/SQL Contract Developer
COMSYS
John.dyer@comsys.com

However excellent they are, most computer books are inherently passive--we must simply take in text without having any opportunity to react to it. *The Oracle PL/SQL Developer's Workbook* is very different! It's designed to involve us actively, to get us solving programming problems immediately, and to help us apply what we've learned about PL/SQL--and in the process deepen our knowledge of the language. By completing the exercises in this workbook, we will find ourselves moving more rapidly along the learning curve to join the growing ranks of PL/SQL experts.

The *Workbook* presents a carefully constructed set of problems and solutions that will test our language skills and help us become better developers. The three levels of exercises - *beginner*, *intermediate*, and *expert* - cover the full set of language features. These include variables, loops, exception handling, data structures, object technology, cursors, built-in functions and packages, PL/SQL tuning, and the new Oracle8i features (including Java and the Web).

Within each chapter, exercises are organized by level of expertise as follows:

Beginner: Problems in this section assume very little prior experience with the PL/SQL language and can be used to increase our familiarity with the basic functionality of the language.

Intermediate: You have been working with PL/SQL for a while and feel comfortable with finding your way around the built-in functions, cursor FOR loops, etc. The problems in this category will stretch our comfort level, confirm our knowledge, and take us in new directions.

Expert: The problems in this category often require that we apply our knowledge in new and creative ways. Some will take a fair amount of effort, but Feuerstein hopes we will then be able to apply the resulting code to our development environment.

Chapters are grouped into categories as follows:

Chapters 1 through 5 contain exercises testing your mastery of *language fundamentals*:

1. **Declaring Variables and Naming Elements:** In this chapter, our ability to work these most basic PL/SQL elements is tested.
2. **Loops:** The four basic looping (or iterative) structures are covered: the simple loop, the FOR loop, the cursor-based FOR loop, and the WHILE loop.
3. **Conditional and Sequential Logic:** In this chapter, our ability to use conditional and sequential structures to direct our program's flow is tested.

REVIEW:
By John J Dyer, CCP, CDP
PL/SQL Contract Developer
COMSYS
John.dyer@comsys.com

4. **Exception Handling:** Our author outlines the two steps to handling an exception: To define the conditions that represent exceptions and to create an exception handling section in your code.

5. **Records:** A record is a construct structurally and conceptually similar to a row in a database table. Here we learn about PL/SQL records.

Chapters 6 through 9 contain exercises in the area of *data structures*:

6. **Index-by Tables:** This structure, formerly known as a PL/SQL table, is PL/SQL's answer to an array. An index-by table consists of homogeneous elements (of the same data type) and indexed by an integer. These tables are one-dimensional, sparse (meaning that there can be "gaps" between elements), and unconstrained (meaning that the number of elements can grow).

7. **Nested Tables:** Here you determine whether you can create a nested table type, store the array in a "store table", add and delete elements, and traverse each element.

8. **Variable Arrays:** These arrays contain a fixed number of elements. Here you learn to define a VARRAY, store a VARRAY in a database table, and add new elements to a VARRAY.

9. **Object Technology:** Here your ability to apply object-oriented principles when designing a data structure is exercised.

Chapters 10 through 13 cover *database interaction*:

10. **Cursors:** In one form or another, every SQL statement in our PL/SQL programs is associated with either an implicit or explicit cursor. We can think of a *cursor* as a pointer into the result set of a SQL statement whose job is to allow us to access and manipulate the data inside the set.

11. **DML and Transaction Management:** Any of the Data Manipulation Language statements (INSERT, UPDATE, DELETE, and SELECT) can be embedded inside our PL/SQL programs. These statements can also be combined into an "all or nothing" unit of work called a *transaction*. The basic idea is that once you begin an operation, you must complete the entire set of operations before any changes are committed, or saved, to the database.

12. **Cursor Variables:** Cursor variables are reference pointers to a cursor's result set. The main advantage of a cursor variable is that it allows us to easily pass the results of a query among different PL/SQL blocks, such as procedures or functions.

REVIEW:
By John J Dyer, CCP, CDP
PL/SQL Contract Developer
COMSYS
John.dyer@comsys.com

13. **Native Dynamic SQL:** At runtime, we can construct the query, the DELETE or CREATE TABLE statement, or even the PL/SQL block as a string and then execute it. Web-based applications have a frequent requirement to use dynamic SQL. Native dynamic SQL (NDS) is available only in Oracle 8i.

Chapters 14 through 17 test your expertise in *program construction*:

14. **Procedures, Functions, and Blocks:** A procedure is a named group of instructions - a block - that perform a specific task. A function is similar in structure to a procedure, but it returns a value to the block that called it.

15. **Packages:** A package is a named collection of procedures, functions, and data structures. It has two parts: a specification, which lists its publicly available elements, and a body, which contains the actual implementations for the procedures and functions listed in the specifications.

16. **Triggers:** A trigger is a special PL/SQL procedure that fires, or executes, in response to a specific triggering event.

17. **Calling Functions in SQL:** This chapter describes the features that allow PL/SQL developers to embed calls to our own functions inside SQL statement.

Chapters 18 through 25 focus on *built-in functionality*:

18. **Character Functions:** These functions allow us to manipulate VARCHAR2, CHAR, and RAW variables.

19. **Date Functions:** These functions allow us to manipulate dates in their various formats.

20. **Conversion, Numeric, and Miscellaneous Functions:** Here are described an assortment of functions to compute mathematical equations, convert data from one format to another, and perform miscellaneous operations.

21. **DBMS_SQL Built-in Package:** Here is described how you construct and execute SQL and PL/SQL statements and commands on the fly.

22. **DBMS_PIPE Built-in Package:** A pipe is a named communication channel that allows different database sessions to communicate.

23. **DBMS_OUTPUT Built-in Package:** This is the package that allows us to display messages to our session's standard output device.

REVIEW:
By John J Dyer, CCP, CDP
PL/SQL Contract Developer
COMSYS
John.dyer@comsys.com

24. **UTL_FILE Built-in Package:** This package allows PL/SQL programs to read and write operating-system files. We should probably be very careful in using these functions.

25. **DBMS_JOB Built-in Package:** This package allows us to schedule PL/SQL routines, or jobs, to run periodically based on intervals we define.

Chapters 26 through 30 include the following *miscellaneous topics*:

26. **Using Java with PL/SQL:** Here we see one of Oracle8i's major new features, the ability to store and execute Java programs inside the database. This gives us another server-side development language (the first being PL/SQL).

27. **External Procedures:** Here we learn how to make calls to external libraries written in other languages, such as C.

28. **PL/SQL Web Development:** Although the Oracle database does not currently support web access directly, there are a number of Oracle tools, such as Oracle Application Server (OAS) and WebDB, that allow us to build web applications using PL/SQL.

29. **Tuning PL/SQL:** Good performance is a natural consequence of clean, well-designed programs. This chapter shows how to more effectively use algorithms, SQL, and data structures.

30. **PL/SQL for DBAs:** Here our ability to use the base PL/SQL language, as well as an assortment of built-in packages, to simplify and automate a variety of database maintenance tasks.