

*A Guide to Oracle's
PL/SQL Language Fundamentals*

3rd Edition
Covers Oracle Database 10g



Oracle PL/SQL Language

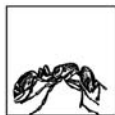
Pocket Reference



O'REILLY®

*Steven Feuerstein,
Bill Pribyl & Chip Dawes*

Oracle PL/SQL Language Pocket Reference



The third edition of this popular pocket guide provides quick-reference information that will help you use Oracle's PL/SQL language, including the newest Oracle Database 10g features. It's a companion to Steven Feuerstein and Bill Pribyl's bestselling *Oracle PL/SQL Programming*.

This concise guide boils down the most vital PL/SQL information into an accessible summary of fundamental language elements (e.g., block structure, identifiers, variables, datatypes, and declarations); statements for program control, cursor management, and exception handling; the basics of records, procedures, functions, triggers, and packages; and the calling of PL/SQL functions in SQL. It also includes Oracle's object-oriented features, collections, external procedures, and Java integration. The third edition describes such Oracle Database 10g elements as regular expressions, compile-time warnings, more implicit conversions, FORALL support for nonconsecutive indexes, additional nested table functionality, user-defined quote characters, new datatypes (BINARY FLOAT and BINARY DOUBLE), and enhancements to PL/SQL native compilation.

Steven Feuerstein has adopted PL/SQL as a member of his immediate family. Pay them both a visit at <http://www.minmaxplsql.com>.

Bill Pribyl is thankful if his writing and lectures about PL/SQL make things a little bit easier for Oracle users. **Chip Dawes** is a DBA and PL/SQL developer consultant in the Chicago area.

ISBN 0-596-00680-2

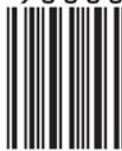
US \$ 9.95

CAN \$14.95

90000



9 780596 006808



Visit O'Reilly on
the Web at
www.oreilly.com



6 36920 00680 0

THIRD EDITION

Oracle PL/SQL Language

Pocket Reference

*Steven Feuerstein, Bill Pribyl,
and Chip Dawes*

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Paris • Sebastopol • Taipei • Tokyo

Oracle PL/SQL Language Pocket Reference, Third Edition

by Steven Feuerstein, Bill Pribyl, and Chip Dawes

Copyright © 2004, 2003, 1999 O'Reilly Media, Inc. All rights reserved.
Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North,
Sebastopol, CA 95472.

O'Reilly Media, Inc. books may be purchased for educational, business, or
sales promotional use. Online editions are also available for most titles
(*safari.oreilly.com*). For more information, contact our corporate/
institutional sales department: (800) 998-9938 or *corporate@oreilly.com*.

Editor:	Deborah Russell
Production Editor:	Mary Brady
Cover Designer:	Edie Freedman
Interior Designer:	David Futato

Printing History:

April 1999:	First Edition.
February 2003:	Second Edition.
April 2004:	Third Edition.

Nutshell Handbook, the Nutshell Handbook logo, and the O'Reilly logo are registered trademarks of O'Reilly Media, Inc. The *Pocket Reference* series designations, *Oracle PL/SQL Language Pocket Reference*, Third Edition, the image of ants, and related trade dress are trademarks of O'Reilly Media, Inc. Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and O'Reilly Media, Inc. was aware of a trademark claim, the designations have been printed in caps or initial caps. Oracle® and all Oracle-based trademarks and logos are trademarks or registered trademarks of Oracle Corporation, Inc. in the United States and other countries. O'Reilly Media, Inc. is independent of Oracle Corporation. Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. O'Reilly Media, Inc. is independent of Sun Microsystems, Inc.

While every precaution has been taken in the preparation of this book, the publisher and authors assume no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

Contents

Introduction	1
Acknowledgments	2
Conventions	2
PL/SQL Language Fundamentals	2
PL/SQL Character Set	2
Identifiers	3
Boolean, Numeric, and String Literals	4
Numeric Literals	5
Datetime Interval Literals	5
Delimiters	6
Comments	7
Pragmas	8
Statements	9
Block Structure	9
Variables and Program Data	11
Scalar Datatypes	11
NLS Character Datatypes	17
LOB Datatypes	17
Implicit Datatype Conversions	18
NULLs in PL/SQL	18
Declaring Variables	18
Anchored Declarations	21
Programmer-Defined Subtypes	22

Conditional and Sequential Control	22
Conditional Control Statements	22
Sequential Control Statements	26
Loops	27
Simple Loop	27
Numeric FOR Loop	28
Cursor FOR Loop	29
WHILE Loop	29
REPEAT UNTIL Loop Emulation	30
EXIT Statement	30
Loop Labels	30
Database Interaction	31
Transaction Management	31
Autonomous Transactions	34
Cursors in PL/SQL	35
Explicit Cursors	35
Implicit Cursors	39
Dynamic Cursors	42
Cursor Variables	43
Cursor Expressions	44
Exception Handling	46
Declaring Exceptions	46
Raising Exceptions	48
Scope	49
Propagation	49
Records in PL/SQL	52
Declaring Records	52
Referencing Fields of Records	53
Assigning Records	53
DML and Records	54
Nested Records	55

Named Program Units	56
Procedures	56
Functions	57
Parameters	58
Triggers	72
Creating Triggers	73
Trigger Predicates	76
DML Events	76
DDL Events	77
Database Events	77
Packages	77
Package Structure	78
Referencing Package Elements	80
Package Data	80
SERIALLY_REUSABLE Pragma	80
Package Initialization	81
Calling PL/SQL Functions in SQL	82
Calling a Function	83
Requirements and Restrictions	84
Calling Packaged Functions in SQL	84
Column/Function Name Precedence	85
Oracle's Object-Oriented Features	85
Object Types	86
Type Inheritance	87
Methods	88
Methods in Subtypes	91
Manipulating Objects in PL/SQL and SQL	93
Upcasting and Downcasting	94
Changing Object Types	97

Collections	99
Declaring a Collection	101
Initializing a Collection	102
Adding and Removing Elements	103
Nested Table Functions	104
Collection Methods	107
Collections and Privileges	110
Nested Collections	110
Bulk Binds	110
External Procedures	114
Creating an External Procedure	114
Parameters	120
Java Language Integration	124
Example	125
Publishing Java to PL/SQL	126
Data Dictionary	128
Regular Expressions (Oracle Database 10g)	128
Metacharacters	128
REGEXP_LIKE	129
REGEXP_INSTR	130
REGEXP_SUBSTR	131
REGEXP_REPLACE	132
Match Modifiers	133
Reserved Words	133
Index	135

Oracle PL/SQL Language Pocket Reference

Introduction

The *Oracle PL/SQL Language Pocket Reference* is a quick reference guide to the PL/SQL programming language, which provides procedural extensions to the SQL relational database language and a range of Oracle development tools. Where a package, program, or function is supported only for a particular version of Oracle (e.g., Oracle Database 10g), we indicate this in the text.

The purpose of this pocket reference is to help PL/SQL users find the syntax of specific language elements. It is not a self-contained user guide; basic knowledge of the PL/SQL programming language is required. For more information, see the following O'Reilly books:

Oracle PL/SQL Programming, Third Edition, by Steven Feuerstein with Bill Pribyl

Learning Oracle PL/SQL, by Bill Pribyl with Steven Feuerstein

Oracle in a Nutshell, by Rick Greenwald and David C. Kreines

Oracle PL/SQL Best Practices, by Steven Feuerstein

Acknowledgments

Many thanks to all those who helped in the preparation of this book. In particular, thanks to first-edition reviewers Eric J. Givler and Stephen Nelson, to second-edition reviewer Jonathan Gennick, and to both Bryn Llewellyn and Jonathan Gennick for their input on this latest revision. In addition, we appreciate all the good work by the O'Reilly crew in editing and producing this book.

Conventions

UPPERCASE indicates PL/SQL keywords.

lowercase indicates user-defined items such as parameters.

Italic indicates filenames and parameters within text, as well as the first use of a term.

Constant width is used for code examples and output.

Constant width bold indicates user input in examples showing an interaction.

[] enclose optional items in syntax descriptions.

{ } enclose a list of items in syntax descriptions; you must choose one item from the list.

| separates bracketed list items in syntax descriptions.

PL/SQL Language Fundamentals

This section summarizes the fundamental components of the PL/SQL language: characters, identifiers, literals, delimiters, use of comments and pragmas, and construction of statements and blocks.

PL/SQL Character Set

The PL/SQL language is constructed from letters, digits, symbols, and whitespace, as defined in the following table:

Type	Characters
Letters	A–Z, a–z
Digits	0–9
Symbols	~!@#\$\$%^&*()_-=+ [] { } ; : " ' < > , . ? / ^
Whitespace	space, tab, newline, carriage return

Characters are grouped together into four lexical units: identifiers, literals, delimiters, and comments.

Identifiers

Identifiers are names for PL/SQL objects such as constants, variables, exceptions, procedures, cursors, and reserved words. Identifiers have the following characteristics:

- Can be up to 30 characters in length
- Cannot include whitespace (space, tab, carriage return)
- Must start with a letter
- Can include a dollar sign (\$), an underscore (_), and a pound sign (#)
- Are not case-sensitive

In addition, you must not use PL/SQL’s reserved words as identifiers. For a list of those words, see the “Reserved Words” section at the end of the book.

If you enclose an identifier within double quotes, all but the first of these rules are ignored. For example, the following declaration is valid:

```
DECLARE
    "1 ^abc" VARCHAR2(100);
BEGIN
    IF "1 ^abc" IS NULL THEN ...
END;
```

Boolean, Numeric, and String Literals

Literals are specific values not represented by identifiers. For example, TRUE, 3.14159, 6.63E-34, 'Moby Dick', and NULL are all literals of type Boolean, number, or string. There are no complex datatype literals as their values are internal representations; complex types receive values through direct assignment or via constructors. Unlike the rest of PL/SQL, literals are case-sensitive. To embed single quotes within a string literal, place two single quotes next to each other.

Oracle Database 10g allows you to define your own quoting mechanism for string literals in both your SQL and PL/SQL statements. Use the characters q' (q followed by a straight single quote) to designate the programmer-defined delimiter for your string literal. Terminate the literal string with the programmer-defined delimiter followed by a trailing single quote—for example, q'!my string!'. NCHAR and NVARCHAR delimiters are preceded by the letters nq, as in nq'^nchar string^'. This technique can simplify your code when consecutive single quotes appear within a string, such as the literals in a SQL statement. If you define your delimiter with one of the four bracketing characters ([{ < , you must use the righthand version of the bracketing character as the closing delimiter. For example, q'[must be closed with]'.

See the following table for examples:

Literal	Actual value
'That''s Entertainment!'	That's Entertainment!
q'#That's Entertainment!#'	That's Entertainment!
'"The Raven"'	"The Raven"
'TZ='CDT6CST'''	TZ='CDT6CST'
q'\$TZ='CDT6CST'\$'	TZ='CDT6CST'
q'[TZ='CDT6CST']'	TZ='CDT6CST'
''''	,
'''hello world'''	'hello world'
q'!'hello world!'	'hello world'

Literal	Actual value
<code>''</code>	<code>''</code>
<code>q['']</code>	<code>''</code>
<code>nq'<Price='£10'></code>	<code>Price='£10'</code>
<code>nq'-WHERE name LIKE '%ñ%' -'</code>	<code>WHERE name LIKE '%ñ%'</code>

Numeric Literals

You may achieve improvements in runtime performance by making explicit the datatype of numeric literals. You can do so by including or excluding a decimal point, or by using a trailing `f` or `d`, as shown in the following table:

Literal	Datatype
<code>3.14159</code>	NUMBER
<code>42</code>	INTEGER
<code>0.0</code>	NUMBER
<code>3.14159f</code>	BINARY_FLOAT (Oracle Database 10g)
<code>3.14159d</code>	BINARY_DOUBLE (Oracle Database 10g)

Oracle Database 10g introduced several special named constants:

BINARY_FLOAT_NAN (Not a Number)
 BINARY_FLOAT_INFINITY
 BINARY_FLOAT_MAX_NORMAL
 BINARY_FLOAT_MIN_NORMAL
 BINARY_FLOAT_MAX_SUBNORMAL
 BINARY_FLOAT_MIN_SUBNORMAL

as well as the `BINARY_DOUBLE_` versions of these constants.

Datetime Interval Literals

The datetime interval datatypes, introduced in Oracle9i, represent a chronological interval expressed in terms of either years and months or days, hours, minutes, seconds, and

fractional seconds. Literals of these datatypes require the keyword `INTERVAL` followed by the literal and format string(s). The interval must go from a larger field to a smaller one, so `YEAR TO MONTH` is valid, but `MONTH TO YEAR` is not. See the following table for examples:

Literal	Actual value
<code>INTERVAL '1-3' YEAR TO MONTH</code>	1 year and 3 months later
<code>INTERVAL '125-11' YEAR(3) TO MONTH</code>	125 years and 11 months later
<code>INTERVAL '-18' MONTH</code>	18 months earlier
<code>INTERVAL '-48' HOUR</code>	48 hours earlier
<code>INTERVAL '7 23:15' DAY TO MINUTE</code>	7 days, 23 hours, 15 minutes later
<code>INTERVAL '1 12:30:10.2' DAY TO SECOND</code>	1 day, 12 hours, 30 minutes, 10.2 seconds later
<code>INTERVAL '12:30:10.2' HOUR TO SECOND</code>	12 hours, 30 minutes, 10.2 seconds later

Delimiters

Delimiters are symbols with special meaning, such as `:=` (assignment operator), `||` (concatenation operator), and `;` (statement delimiter). The following table lists the PL/SQL delimiters:

Delimiter	Description
<code>;</code>	Terminator (for statements and declarations)
<code>+</code>	Addition operator
<code>-</code>	Subtraction operator
<code>*</code>	Multiplication operator
<code>/</code>	Division operator
<code>**</code>	Exponentiation operator
<code> </code>	Concatenation operator
<code>:=</code>	Assignment operator
<code>=</code>	Equality operator
<code><></code> and <code>!=</code>	Inequality operators

Delimiter	Description
\neq and \approx	Inequality operators
<	"Less-than" operator
<=	"Less-than or equal to" operator
>	"Greater-than" operator
>=	"Greater-than or equal to" operator
(and)	Expression or list delimiters
<< and >>	Label delimiters
,	(Comma) Item separator
'	(Single quote) Literal delimiter
q' and '	Programmer-defined string literal delimiter
nq' and 'n	Programmer-defined NCHAR string literal delimiter
"	(Double quote) Quoted literal delimiter
:	Host variable indicator
%	Attribute indicator
.	(Period) Component indicator (as in <i>record.field</i> or <i>package.element</i>)
@	Remote database indicator (database link)
=>	Association operator (named notation)
..	(Two periods) Range operator (used in the FOR loop)
--	Single-line comment indicator
/* and */	Multiline comment delimiters

Comments

Comments are sections of code that exist to aid readability. The compiler ignores them.

A single-line comment begins with a double hyphen (--) and ends with a new line. The compiler ignores all characters between the -- and the new line.

A multiline comment begins with slash asterisk (/*) and ends with asterisk slash (*). The /* */ comment delimiters also can

be used for a single-line comment. The following block demonstrates both kinds of comments:

```
DECLARE
  -- Two dashes comment out only the physical line.
  /* Everything is a comment until the compiler
     encounters the following symbol */
```

You cannot embed multiline comments within a multiline comment, so be careful during development if you comment out portions of code that include comments. The following code demonstrates this issue:

```
DECLARE
  /* Everything is a comment until the compiler
     /* This comment inside another WON'T work!*/
     encounters the following symbol. */

  /* Everything is a comment until the compiler
     -- This comment inside another WILL work!
     encounters the following symbol. */
```

Pragmas

The `PRAGMA` keyword is used to give instructions to the compiler. There are four types of pragmas in PL/SQL:

EXCEPTION_INIT

Tells the compiler to associate the specified error number with an identifier that has been declared an `EXCEPTION` in your current program or an accessible package. See the “Exception Handling” section for more information on this pragma.

RESTRICT_REFERENCES

Tells the compiler the purity level of a packaged program. The purity level is the degree to which a program does not read/write database tables and/or package variables. See the “Calling PL/SQL Functions in SQL” section for more information on this pragma.