

# Regular Expression Metacharacters and Function Parameters

This appendix describes the various regular expression metacharacters available starting with Oracle Database 10g. It also provides a summary of the syntax of the REGEXP\_ functions. For more details on Oracle's regular expression support, see [Chapter 8](#).

## Metacharacters

The “Initial release” column in [Table A-1](#) through [Table A-3](#) indicates which metacharacters were introduced in Oracle Database 10g Release 1 and which in Release 2.

*Table A-1. Character-matching metacharacters*

Syntax	Initial release	Description
.	10gR1	Matches any single character except for newline. Will match newline when the <i>n</i> flag is set. On Windows, Linux, and Unix platforms, chr(10) is recognized as the newline.
[ ... ]	10gR1	Defines a <i>matching list</i> that matches any character listed between the brackets. You may specify ranges of characters, as in a–z. These ranges are interpreted based on the NLS_SORT setting. A dash (-) is a literal when it occurs first or last in the list (e.g., [abc-]). A closing bracket (]) is a literal when it occurs first in the list (e.g., [)abc]). A caret (^) in the first position makes the list a <i>nonmatching list</i> (see the next entry).
[^ ... ]	10gR1	Matches any character not listed between the brackets. Referred to as a <i>nonmatching list</i> .
[ :class: ]	10gR1	Matches any character that belongs to the specified character class. May only be used within a matching list: [[:class:]]abc is a valid expression, but [:class:]abc is not. <a href="#">Table A-5</a> lists the valid character class names.
[.coll.]	10gR1	Matches the specified collation element, which may be one or more characters. May only be used within a matching list. For example, the expression [[.ch.]] matches the Spanish letter <i>ch</i> . <a href="#">Table A-4</a> lists the valid collation elements.

Syntax	Initial release	Description
[ <i>=char=</i> ]	10gR1	Matches all characters that share the same base character as <i>char</i> . May be used only within a matching list. For example, [ <i>=e=</i> ] matches any of: “éèëÊËËË”.
\d	10gR2	Matches any digit. Equivalent to [[:digit:]].
\D	10gR2	Matches any nondigit. Equivalent to [^[:digit:]].
\w	10gR2	Matches any <i>word character</i> . Word characters are defined to be alphabetic characters, numeric characters, and the underscore.
\W	10gR2	Matches any nonword character.
\s	10gR2	Matches any whitespace character. Equivalent to [[:space:]].
\S	10gR2	Matches nonwhitespace characters. Equivalent to [^[:space:]].

Table A-2. Quantifiers

Syntax	Initial release	Description
?	10gR1	Zero or one.
*	10gR1	Zero or more.
+	10gR1	One or more.
{ <i>m</i> }	10gR1	Exactly <i>m</i> occurrences.
{ <i>m</i> ,}	10gR1	At least <i>m</i> occurrences.
{ <i>m</i> , <i>n</i> }	10gR1	At least <i>m</i> , and at most <i>n</i> occurrences.
+?	10gR2	One or more, but nongreedy.
??	10gR2	Zero or one, but nongreedy.
{ <i>m</i> }?	10gR2	The same as { <i>m</i> }.
{ <i>m</i> ,}?	10gR2	At least <i>m</i> occurrences, but nongreedy and stops as soon as <i>m</i> occurrences are reached.
{ <i>m</i> , <i>n</i> }?	10gR2	At least <i>m</i> , and at most <i>n</i> occurrences, but nongreedy; when possible, <i>m</i> occurrences are matched.

Table A-3. Other metacharacters

Syntax	Initial release	Description
	10gR1	Specifies an alternation. An alternation within a subexpression doesn’t extend beyond the subexpression.
( ... )	10gR1	Defines a subexpression.
\n	10gR1	References the text matched by the <i>n</i> th subexpression. Backreferences may range from \1 through \9.
\	10gR1	When not followed by a digit, the \ is an escape character. For example, use the pattern \\1 to look for a single backslash followed by the digit 1; use \( to look for an opening parenthesis (rather than begin a subexpression), etc.
^	10gR1	Anchors an expression to the beginning of the string (in multiline mode, to the beginning of a line).
\$	10gR1	Anchors an expression to the end of the string (in multiline mode, to the end of a line).
\A	10gR2	Anchors an expression to the beginning of the string regardless of whether multiline mode is specified.

Syntax	Initial release	Description
\Z	10gR2	Anchors an expression to the end of the string, or a newline that happens to be ending a string, regardless of whether multiline mode is specified.
\z	10gR2	Anchors an expression to the end of the string regardless of whether multiline mode is specified.

Table A-4. Collation elements

NLS_SORT	Multicharacter collation elements		
XCROATIAN	d_	D_	D_
	lj	LJ	Lj
	nj	Nj	NJ
XCZECH	Ch	CH	Ch
XCZECH_PUNCTUATION	Ch	CH	Ch
XDANISH	aa	AA	Aa
	oe	OE	Oe
XHUNGARIAN	cs	CS	Cs
	gy	GY	Gy
	ly	LY	Ly
	ny	NY	Ny
	sz	SZ	Sz
	ty	TY	Ty
	zs	ZS	Zs
XSLOVAK	dz	DZ	Dz
	d_	D_	D_
	ch	CH	Ch
XSPANISH	ch	CH	Ch
	ll	LL	Ll

Table A-5. Supported character classes

Class	Description
[alnum:]	Alphanumeric characters (same as [:alpha:] + [:digit:])
[alpha:]	Alphabetic characters only
[blank:]	Blank space characters, such as space and tab
[cntrl:]	Nonprinting (control) characters
[digit:]	Numeric digits
[graph:]	Graphical characters (same as [:punct:] + [:upper:] + [:lower:] + [:digit:])
[lower:]	Lowercase letters
[print:]	Printable characters
[punct:]	Punctuation characters
[space:]	Whitespace characters such as space, formfeed, newline, carriage return, horizontal tab, and vertical tab

Class	Description
[upper:]	Uppercase letters
[xdigit:]	Hexadecimal characters

## Functions and Parameters

The following subsection shows the syntax of Oracle’s regular expression functions. The meaning of the parameters is shown in [“Regular Expression Parameters” on page 1279](#).

### Regular Expression Functions

The syntax for each regular expression function is shown next.

#### REGEXP\_COUNT (Oracle Database 11g and later)

Returns a tally of occurrences of an expression in a target string. The syntax is:

```
REGEXP_COUNT(source_string, expression
             [, position
             [, match_parameter]])
```

#### REGEXP\_INSTR

Returns the character position at which text can be found matching a regular expression in a target string. The syntax is:

```
REGEXP_INSTR(source_string, expression
             [, position [, occurrence
             [, return_option
             [, match_parameter
             [, subexpression]]]]])
```

#### REGEXP\_LIKE

Determines whether a given string contains text matching an expression. This is a Boolean function, returning TRUE, FALSE, or NULL. The syntax is:

```
REGEXP_LIKE (source_string, expression
             [, match_parameter])
```

#### REGEXP\_REPLACE

Performs a regular expression search-and-replace operation (see [Chapter 8](#) for details). The syntax is:

```
REGEXP_REPLACE(source_string, expression
               [, replace_string
               [, position [, occurrence
               [, match_parameter]]]])
```

## REGEXP\_SUBSTR

Extracts text matching a regular expression from a string. The syntax is:

```
REGEXP_SUBSTR(source_string, expression  
              [, position [, occurrence  
              [, match_parameter  
              [, subexpression]]]])
```

## Regular Expression Parameters

These are the parameters that may be included in the regular expression functions described in the preceding subsection:

*source\_string*

Is a string to be searched.

*expression*

Is a regular expression describing the pattern of text that you seek.

*replace\_string*

Is a string generating the replacement text to be used in a search-and-replace operation.

*position*

Is the character position within *source\_string* at which to begin a search. This defaults to 1.

*occurrence*

Is the occurrence of the pattern you want to locate. This defaults to 1, giving you the first possible match.

*return\_option*

Is valid only for REGEXP\_INSTR, and determines whether the beginning or ending character position is returned for text matching a pattern. The default is 0, for the beginning. Use 1 to return the ending position.

*match\_parameter*

Is a text string through which you may specify options to vary the behavior of the regular expression matching engine:

*'c'*

Requests a case-sensitive search. (By default, your NLS\_SORT setting determines whether a search is case-sensitive.)

*'i'*

Requests a case insensitive search.

**'n'**

Allows the period to match newline characters. By default, the period does not match newlines.

**'m'**

Changes the definition of *line* with respect to the ^ and \$ metacharacters. By default, line means *the entire target string*. Using the `m` option, however, causes the definition of line to change from *the entire target string* to *any line within that string*, where lines are delimited by newline characters.

*subexpression (Oracle Database 11g and later)*

Is a number (0–9) identifying which subexpression to match on. The default is 0 and signifies that subexpressions will not be used.

You can specify multiple match parameters in any order. For example, 'in' means the same as 'ni'. If you specify conflicting options (such as 'ic'), the last option ('c', in this case) is the one that takes precedence.

## Number Format Models

Number formats are used with both the `TO_CHAR` function and the `TO_NUMBER` function. You use number formats in calls to `TO_CHAR` to specify exactly how a numeric value should be translated into a `VARCHAR2` string. You can specify the punctuation to use, the location of the positive or negative sign, and other useful items. Conversely, you use number formats in calls to `TO_NUMBER` to specify how a string representing a numeric value should be interpreted.

A number format mask can comprise one or more elements from [Table B-1](#). The resulting character string (or the converted numeric value) reflects the combination of the format model elements you use. You will find examples of different applications of the format models in the descriptions of `TO_CHAR` and `TO_NUMBER`.

Format elements with a description starting with “Prefix:” can be used only at the beginning of a format mask; when a description starts with “Suffix:” the element can be used only at the end of a format mask. Most format elements are described in terms of their effect on a conversion of a number to its character string representation. Bear in mind that the majority of such elements may also be used in the converse manner—to specify the format of a character string to be converted into a number.

*Table B-1. Number format model elements*

Format element	Description
\$	Prefix: puts a dollar sign in front of a number (for the currency symbol, see the C format element).
, (comma)	Places a comma into the return value. This comma is used as a group separator (see the G format element).
. (period)	Places a period into the return value. This period is used as a decimal point (see the D format element).
0	Each zero represents a significant digit to be returned. Leading zeros in a number are displayed as zeros.
9	Each 9 represents a significant digit to be returned. Leading zeros in a number are displayed as blanks.
B	Prefix: returns a zero value as blanks, even if the 0 format element is used to show leading zeros.

Format element	Description
C	Specifies the location of the ISO currency symbol in the returned value. The NLS_ISO_CURRENCY parameter specifies the ISO currency symbol.
D	Specifies the location of the decimal point in the returned value. All format elements to the left of the D format the integer component of the value. All format elements to the right of the D format the fractional part of the value. The character used for the decimal point is determined by the NLS_NUMERIC_CHARACTERS database parameter.
EEEE	Suffix: specifies that the value be returned in scientific notation.
FM	Prefix: removes any leading or trailing blanks from the return value.
G	Specifies the location of the group separator (for example, a comma or period to separate thousands, as in 6,754 or 6.754) in the returned value. The character used for the group separator is determined by the database parameter NLS_NUMERIC_CHARACTERS.
L	Specifies the location of the local currency symbol (such as \$ or €) in the return value. The NLS_CURRENCY parameter specifies the local currency symbol.
MI	Suffix: places a minus sign (–) after the number if it is negative. If the number is positive, a trailing space is placed after the number.
PR	Suffix: places angle brackets (< and >) around a negative value. Positive values are given a leading and a trailing space.
RN or rn	Specifies that the return value be converted to upper- or lowercase Roman numerals. The range of valid numbers for conversion to Roman numerals is between 1 and 3999. The value must be an integer. RN returns uppercase Roman numerals, while rn returns lowercase Roman numerals.
S	Prefix: places a plus sign (+) in front of a positive number and a minus sign (–) in front of a negative number.
TM	Prefix: returns a number using the minimum number of characters. TM stands for <i>text minimum</i> . Follow TM with one 9 if you want regular decimal notation (the default). Follow TM with one E if you want scientific notation.
U	Places the dual currency symbol (often €) at the specified location. The NLS_DUAL_CURRENCY parameter controls the character returned by this format element.
V	Multiplies the number to the left of the V in the format model by 10 raised to the <i>n</i> th power, where <i>n</i> is the number of 9s found after the V in the format model.
X	Returns a number in hexadecimal form. You can precede this element with 0s to return leading zeros or with FM to trim leading and trailing blanks. X cannot be used in combination with any other format elements.

Notice that sometimes two elements can specify the same thing, or seemingly the same thing. For example, you can use the dollar sign (\$), comma (,), and period (.), or you can use the L, G, and D elements, respectively. The letter elements respect your current NLS settings and return the proper characters for whatever language you are using. For example, some European languages use a comma rather than a period to represent the decimal point. The dollar sign, comma, and period format elements are US-centric and always return those three characters. We recommend that you use the NLS-sensitive format model elements (such as L, G, and D) unless you have a specific reason to do otherwise.



# Denoting Monetary Units

**Table B-1** shows four format elements you can use to denote currency symbols. These elements are \$, L, C, and U, and you may be wondering about the differences among them:

*The \$ format element*

Is US-centric and always returns a dollar sign (\$).

*The L format element*

Respects your current NLS\_CURRENCY setting, which specifies your local currency indicator. If, for example, you set your NLS\_TERRITORY to indicate that you're in the United Kingdom, NLS\_CURRENCY will default to £, and the L format element will result in £ being used as the currency indicator.

*The C format element*

Is similar to the L element but results in the ISO currency indicator, as specified by your current NLS\_ISO\_CURRENCY setting. For the United Kingdom, you'll get GBP (for Great Britain pounds), while for the United States, you'll get USD (for US dollars), and so forth.

*The U format element*

Was added to support the euro and uses the currency indicator specified by NLS\_DUAL\_CURRENCY. For countries that support the euro, the NLS\_DUAL\_CURRENCY setting defaults to the euro symbol (€).

To view your current NLS\_CURRENCY and NLS\_ISO\_CURRENCY settings, you can query the NLS\_SESSION\_PARAMETERS or V\$NLS\_PARAMETERS system views.



---

## Date Format Models

**Table C-1** lists the date format model elements that you can use with the conversion functions `TO_CHAR`, `TO_DATE`, `TO_TIMESTAMP`, and `TO_TIMESTAMP_TZ`. Some of the model elements in **Table C-1** are also used with `ROUND` and `TRUNC`.

You have the option of specifying default date and timestamp formats at the session level, a capability that can come in handy if your particular needs differ from those of the majority of database users. Use the `ALTER SESSION` command to specify session-level default date and timestamp formats. The following example works in Oracle8i Database or higher, and sets the default date format to `MM/DD/YYYY`:

```
BEGIN
  EXECUTE IMMEDIATE 'ALTER SESSION SET NLS_DATE_FORMAT='''MM/DD/YYYY''';
END;
```

To check the default date format in effect for your session at any given time, issue the following query against the `NLS_SESSION_PARAMETERS` data dictionary view:

```
SELECT value
FROM nls_session_parameters
WHERE parameter='NLS_DATE_FORMAT';
```

To set or check default timestamp formats, use `NLS_TIMESTAMP_FORMAT` and `NLS_TIMESTAMP_TZ_FORMAT`.

Some elements in **Table C-1** apply only when translating datetime values from Oracle's internal format into character strings, and not vice versa. Such elements can't be used in a default date model (e.g., with `NLS_DATE_FORMAT`) because the default date model applies to conversions in both directions. These elements are noted as "Output only" in the table.

*Table C-1. Date format model elements*

Element	Description
Other text	Any punctuation, such as a comma (,), slash (/), or hyphen (-), will be reproduced in the formatted output of the conversion. You can also include text within double quotes (" ") and the text will be represented as entered in the converted value.
A.M. or P.M.	The meridian indicator (morning or evening) with periods.
AM or PM	The meridian indicator (morning or evening) without periods.
B.C. or A.D.	The B.C. or A.D. indicator, with periods.
BC or AD	The B.C. or A.D. indicator, without periods.
CC and SCC	The century. If the SCC format is used, any B.C. dates are prefaced with a minus sign (-). Output only.
D	The day of the week, from 1 through 7. The day of the week that is decreed the first day is specified implicitly by the NLS_TERRITORY initialization parameter for the database instance.
DAY, Day, or day	The name of the day in uppercase, mixed case, or lowercase format.
DD	The day of the month, from 1 through 31.
DDD	The day of the year, from 1 through 366.
DL	Long date format. Depends on the current values of NLS_TERRITORY and NLS_LANGUAGE. May be used alone or with TS, but not with any other elements.
DS	Short date format. Depends on the current values of NLS_TERRITORY and NLS_LANGUAGE. May be used alone or with TS, but not with any other elements.
DY, Dy, or dy	The abbreviated name of the day, as in TUE for Tuesday.
E	The abbreviated era name. Valid only for the following calendars: Japanese Imperial, ROC Official, and Thai Buddha.
EE	The full era name.
FF	The fractional seconds. Only valid when used with TIMESTAMP values. The number of digits returned will correspond to the precision of the datetime being converted. Always use FF (two Fs) regardless of the number of decimal digits you wish to see or use. Any other number of Fs is invalid.
FF1..FF9	Same as FF, but the digit (1..9) controls the number of decimal digits used for fractional seconds. Use FF1 to see one digit past the decimal point, FF2 to see two digits past, and so forth.
FM	Element that toggles suppression of blanks in output from conversion. (FM stands for Fill Mode.)
FX	Element that requires exact pattern matching between data and format model. (FX stands for Format eXact.)
HH or HH12	The hour of the day, from 1 through 12. Output only.
HH24	The hour of the day, from 0 through 23.
IW	The week of the year, from 1 through 52 or 1 through 53, based on the ISO standard. Output only.
IYY or IY or I	The last three, two, or one digits of the ISO standard year. Output only.
IYYY	The four-digit ISO standard year. Output only.
J	The Julian day format of the date (counted as the number of days since January 1, 4712 B.C., the earliest date supported by the Oracle database).
MI	The minutes component of the datetime value, from 0 through 59.

Element	Description
MM	The number of the month in the year, from 01 through 12. January is month number 01, September is 09, etc.
MON, Mon, or mon	The abbreviated name of the month, as in JAN for January. This also may be in upper-, mixed-, or lowercase format.
MONTH, Month, or month	The name of the month, in upper-, mixed-, or lowercase format.
Q	The quarter of the year, from 1 through 4. January through March are in the first quarter, April through June in the second quarter, and so on. Output only.
RM	The Roman numeral representation of the month number, from I through XII. January is I, September is IX, and so on. Output only.
RR	The last two digits of the year. This format displays years in centuries other than our own.
RRRR	Same as RR when used for output; accepts four-digit years when used for input.
SCC or CC	The century. If the SCC format is used, any B.C. dates are prefaced with a minus sign (–). Output only.
SP	Suffix that converts a number to its spelled format. This element can appear at the end of any element that results in a number. For example, a model such as “DDth-Mon-Yyyysp” results in output such as “15th-Nov-One Thousand Nine Hundred Sixty-One”. The return value is always in English, regardless of the date language. (Note that Yyyy resulted in mixed-case words.)
SPTH or THSP	Suffix that converts a number to its spelled and ordinal format; for example, 4 becomes FOURTH and 1 becomes FIRST. This element can appear at the end of any element that results in a number. For example, a model such as “Ddspth Mon, Yyyysp” results in output such as “Fifteenth Nov, One Thousand Nine Hundred Sixty-One”. The return value is always in English, regardless of the date language.
SS	The seconds component of the datetime value, from 0 through 59.
SSSSS	The number of seconds since midnight of the time component. Values range from 0 through 86399, with each hour comprising 3,600 seconds.
SYEAR, YEAR, SYear, Year, syear, or year	The year spelled out in words (e.g., “two thousand two”). The S prefix places a negative sign in front of B.C. dates. The format may be uppercase, mixed-case, or lowercase. Output only.
SYYYY or YYYY	The four-digit year. If the SYYYY format is used, any B.C. dates are prefaced with a minus sign (–).
TH	Suffix that converts a number to its ordinal format; for example, 4 becomes 4th and 1 becomes 1st. This element can appear at the end of any element that results in a number. For example, “DDth-Mon-YYYY” results in output such as “15th-Nov-1961”. The return value is always in English, regardless of the date language.
TS	Short time format. Depends on the current values of NLS_TERRITORY and NLS_LANGUAGE. May be used alone or with either DL or DS, but not with any other elements.
TZD	The abbreviated time zone name—for example, EST, PST. This is an input-only format, which may seem odd at first.
TZH	The time zone hour displacement. For example, –5 indicates a time zone five hours earlier than UTC.
TZM	The time zone minute displacement. For example, –5:30 indicates a time zone that is five hours, thirty minutes earlier than UTC. A few such time zones do exist.
TZR	The time zone region. For example, “US/Eastern” is the region in which EST (Eastern Standard Time) and EDT (Eastern Daylight Time) are valid.

Element	Description
W	The week of the month, from 1 through 5. Week 1 starts on the first day of the month and ends on the seventh. Output only.
WW	The week of the year, from 1 through 53. Output only.
X	The local radix character. In American English, this is a period (.). This element can be placed in front of FF so that fractional seconds are properly interpreted and represented.
Y,YYY	The four-digit year with a comma.
YYY or YY or Y	The last three, two, or one digits of the year. The current century is the default when you're using these elements to convert a character string value into a date.

Whenever a date format returns a spelled value—words rather than numbers, as with MONTH, MON, DAY, DY, AM, and PM—the language used to spell these words is determined by the Globalization Support (formerly National Language Support) parameters NLS\_DATE\_LANGUAGE and NLS\_LANGUAGE, or by the optional date language argument you can pass to both TO\_CHAR and TO\_DATE.

## ISO Dates

The IYY and IW elements represent the ISO (International Standards Organization) year and week. The ISO calendar is a good example of “design by committee.” The first day of the ISO year is always a Monday and is determined by the following rules:

- When January 1 falls on a Monday, the ISO year begins on the same day.
- When January 1 falls on a Tuesday through Thursday, the ISO year begins on the preceding Monday.
- When January 1 falls on a Friday through Sunday, the ISO year begins on the following Monday.

These rules lead to some strange situations. For example, 31-Dec-2008 is considered to be the first day of ISO year 2009, and if you display that date using the IYYY format, 31-Dec-2009 is exactly what you'll get.

ISO weeks always begin on Mondays and are numbered from the first Monday of the ISO year.

Here are some examples of date format models composed of the preceding format elements:

```
'Month DD, YYYY'
'MM/DD/YY Day A.M.'
'Year Month Day HH24:MI:SS'
'J'
```

```
'SSSSS-YYYY-MM-DD'
```

```
'"A beautiful summer morning on the" DDth" day of "Month'
```

You can use the format elements in any combination, in any order. Older releases of Oracle allowed you to specify the same date element twice. For example, the model “Mon (MM) DD, YYYY” specifies the month twice. However, you can specify an element only once in a format model. For example, you can specify only one of MONTH, MON, and MM because all three refer to the month.

See the description of the TO\_CHAR and TO\_DATE functions in [Chapter 10](#) for more examples of the use and resulting values of date format models.





## Symbols

- ! (exclamation mark)
  - != (not equal) operator, 66, 270
  - testing nested tables, 408
- " " (quotation marks, double)
  - data structure names enclosed in, 174
  - inside string literals, 73
  - surrounding identifiers, 68
- \$ (dollar sign)
  - \$\$identifier syntax for inquiry directives, 1065
  - end-of-line matching in regular expressions, 218
  - in identifiers, 67
- \$\$PLSQL\_UNIT\_OWNER directive, 14
- \$\$PLSQL\_UNIT\_TYPE directive, 14
- % (percent sign)
  - attribute indicator and wildcard character, 66
  - in cursor attribute names, 488
- & (ampersand)
  - causing problems in SQL\*Plus when executing PL/SQL code, 204
  - referring to SQL\*Plus variables, 32
- ' ' (quotation marks, single)
  - embedding inside literal strings, 73
  - embedding within string constants, 203
  - enclosing string constants, 203
  - representing empty string, 206
  - representing null strings, 72
- () (parentheses)
  - escaping in regular expressions, 221
  - functions without parameters, 604
  - grouping in regular expressions, 218
  - procedures without parameters, 596
- \* (asterisk)
  - \*\* (exponentiation) operator, 66, 270
  - multiplication operator, 270
  - quantifier in regular expressions, 217
- + (plus sign)
  - addition operator, 270
  - identity operator, 270
  - quantifier in regular expressions, 218
- (hyphen), -- (double hyphen) delimiting single-line comments, 66, 75
- (minus sign)
  - negation operator, 270
- (underscore)
  - single-character wildcard in LIKE condition, 66
- . (dot notation)
  - accessing fields within records, 334
  - referencing package elements, 654, 667
  - referring to object attributes or methods, 1150
- . (dot)
  - .. (double dot) range operator, 66

*We'd like to hear your suggestions for improving our indexes. Send email to [index@oreilly.com](mailto:index@oreilly.com).*

- / (slash)
  - /\* \*/ multiline comment block delimiters, [66](#), [76](#)
  - division operator, [270](#)
  - trailing slash after PL/SQL statement in SQL\*Plus, [28](#)
- : (colon)
  - := (assignment) operator, [66](#), [181](#)
    - assigning values to a collection, [374](#)
    - changing value of field in a record, [335](#)
    - specifying default values for fields in a record, [329](#)
    - specifying default values for parameters, [618](#)
  - host variable indicator, [66](#)
- ; (semicolon)
  - in IF statement combinations, [89](#)
  - in NULL statement, [101](#)
  - terminating declarations and statements, [66](#)
  - terminating statements and declarations, [75](#)
- < > (angle brackets)
  - < (less than) operator, [270](#)
  - <%= %>, embedding PL/SQL into HTML page, [51](#)
  - << and >> label delimiters, [66](#)
  - <= (less than or equal to) operator, [66](#), [270](#)
  - <> (not equal) operator, [66](#), [270](#)
  - > (greater than) operator, [270](#)
  - >= (greater than or equal to) operator, [66](#), [270](#)
- = (equals sign)
  - => (association) operator, [66](#), [613](#), [614](#)
  - equality operator, [270](#)
    - testing nested table equality, [408](#)
- ? (question mark)
  - nongreedy matching in regular expressions, [226](#)
  - quantifier in regular expressions, [221](#)
- @ (at sign)
  - remote location indicator, [66](#)
  - SQL\*Plus @ command, [29](#)
- [ ] (square brackets)
  - defining character sets in regular expressions, [217](#)
- \ (backslash)
  - escaping regular expression metacharacters, [221](#)

- ^ (caret)
  - beginning-of-line matching in regular expressions, [218](#)
  - ^= (not equal) operator, [66](#), [270](#)
- { } (curly braces)
  - quantifiers in regular expressions, [221](#)
- | (vertical bar)
  - || (concatenation) operator, [66](#), [205](#), [206](#)
- ~ (tilde), ~= (not equal) operator, [66](#), [270](#)
- π (pi), [272](#)

## A

- ABORT procedure, DBMS\_RESUMABLE package, [740](#)
- ABS function, [272](#)
- abstract datatypes, [188](#), [1142](#)
- accent-insensitive sorting of strings, [209](#)
- access control lists (ACLs)
  - creating for network access, [452](#)
  - setting up for email, [946](#)
- ACCESSIBLE BY clause, [13](#), [816](#)
- ACID principle (transactions), [461](#)
- actual parameters, [608](#)
  - explicit association with formal parameters, [613–617](#)
  - NOCOPY parameter mode qualifier and, [918](#)
- Ada, [xxv](#), [1040](#), [1046](#)
  - PL/SQL called from, [52](#)
- addition operator (+), [270](#)
- ADD\_MONTHS function, [312](#)
- Advanced Encryption Standard (AES) algorithm, [976](#), [979](#)
- Advanced Queuing (AQ), [184](#), [668](#)
- Advanced Security Option (Oracle), [976](#)
- AFTER SUSPEND triggers, [736–743](#)
  - deciding whether to fix problems, [743](#)
  - examining the actual trigger, [738](#)
  - ORA\_SPACE\_ERROR\_INFO function, [739](#)
  - setting up, [736](#)
  - trapped multiple times, [742](#)
- AFTER triggers, [689](#)
  - AFTER EACH ROW trigger, [708](#)
  - AFTER GRANT DDL triggers, [717](#)
  - AFTER INSERT OR UPDATE trigger, [692](#)
  - AFTER STATEMENT trigger, [708](#)
- AGENT clause, CREATE LIBRARY statement, [1255](#)
- AGENT IN clause, [1258](#)

- agents, external procedure, 1246
  - multithreaded, 1248, 1252
    - changes required to use two agents, 1254
    - nondefault agents, 1269–1272
- aggregate assignments, 375
- agctl utility, 1253
- AGTCTL\_ADMIN environment variable, 1252
- alerts, 968
- algorithms, encryption, 973
  - Advanced Encryption Standard 128-bit, 979
  - commonly used with Oracle, 976
  - DBMS\_CRYPTO algorithm constants, 978
- aliases
  - column aliases in explicit cursors, 507
  - cursor variable, 529
- ALL\_\* views, 751
- ALL\_DIRECTORIES view, 932
- ALL\_SOURCE view, 1197
- ALTER JAVA statement, 1206
- ALTER TRIGGER command, 743
- ALTER TYPE command, 1162
  - changing nested table or VARRAY collection characteristics, 369
  - INVALIDATE and CASCADE options, 369
- ALTER...COMPILE recompilation, 774
- ANALYZE utility of DBMS\_HPROF, 831
- anchored declarations
  - anchoring to cursors and tables, 185
  - anchoring to NOT NULL datatypes, 188
  - benefits of, 186
  - subtypes, 189
- AND operator, 270
  - short-circuit evaluation in IF statements, 91
- anonymous blocks, 53, 55
  - general syntax, 55
  - nested, 57
  - uses of, 56
  - using labels with, 78
- anonymous exceptions, 131
  - system and programmer-defined, scope of, 139
- ANY partitioning, 902
- Any types, 453–456, 1171–1175
  - AnyData type, 453
  - ANYDATA type
    - dealing with, 1172–1174
    - what it's not, 1172
  - AnyData.ConvertObject method, 455
  - AnyDataSet type, 453
  - AnyType type, 453
    - creating a transient type, 1174
    - resources for more information, 456
- AnyData type, 180
- AnyDataSet type, 180
- AnyType type, 180
- API (application programming interface), 658
- application contexts, 972, 1019–1028
  - as predicates in RLS, 1022–1026
  - identifying nondatabase users, 1026–1028
  - security in, 1022
- Application Developers Guide – Large Objects manual, 444
- application security, 971–1037
  - application contexts, 1019–1028
  - encryption, 973–999
  - fine-grained auditing (FGA), 1028–1037
  - overview, 971
  - RLS (row-level security), 999–1019
    - debugging RLS, 1015–1019
    - reasons for learning about RLS, 1002
    - simple RLS example, 1003–1007
    - static versus dynamic policies, 1007–1012
    - using column-sensitive RLS, 1012–1015
- AQ (Advanced Queuing), 184, 668
- architecture of PL/SQL, 1039–1096
  - conditional compilation, 1064–1076
  - default packages, 1045–1048
  - DIANA, 1040
  - execution authority models, 1049–1064
    - BEQUEATH CURRENT\_USER for views, 1061
  - combining rights models, 1056
  - constraining invoker rights privileges, 1063
  - definer rights model, 1049–1053
  - granting roles to program units, 1057–1060
  - invoker rights model, 1054–1056
  - how Oracle database executes PL/SQL code, 1040–1045
    - compiler limits, 1045
    - example, 1041–1044
  - native compilation, 1094
  - PL/SQL and database instance memory, 1076–1093
  - summary of important points, 1095

- argument modes, 551
    - (see also IN mode; IN OUT mode; OUT mode)
    - bind arguments, 551
  - arguments
    - analyzing in USER\_ARGUMENTS view, 758
  - arrays
    - collections as, 343
    - Java, passing collection into, 1241
    - multidimensional, emulation with unnamed multilevel collections, 391
    - nonsequential, driving FORALL with, 884
    - retrieval of web page into consecutive elements of associative array, 956
    - tuning pipelined functions with array fetches, 893
    - variable-sized (see VARRAYs)
  - AS LANGUAGE clause, 1257
  - AS LANGUAGE JAVA syntax, 1228
  - ASCII function, 206, 231
  - ASCIISTR function, 231, 1106
  - assignment
    - aggregate assignment of entire collection to another collection, 375
    - initializing collection variables implicitly during direct assignment, 371
    - rows from relational table to a collection, 376
  - assignment operator (:=), 66, 181
    - assigning values to a collection, 374
    - specifying default values for fields in a record, 329
    - specifying default values for parameters, 618
  - ASSM (Automatic Segment Space Management), 437
  - association operator (=>), 66, 613, 614
  - associative arrays, 329
    - assigning data to, index values, 375
    - collection type, 343, 345
      - calling built-in methods, 357
      - comparison to VARRAYs and nested tables, 354
      - declaring, 365
      - example of, 346
    - initialization, 370
  - asymmetric encryption algorithms, 975
  - atoms of PL/SQL, 66
  - attribute indicator (%), 66
  - attributes
    - application context, 1020
      - setting or obtaining current value, 1021
    - DDL trigger, 713
      - working with, 715–719
    - file, retrieving, 943
    - object
      - attribute-level comparisons, 1181
      - object, retrieving, 1158
  - audit trail, 1028, 1034
    - checking, 1033
  - auditing, 1028
    - (see also fine-grained auditing)
  - Aurora JVM, 1206
  - authentication
    - HTTP, 959
    - Microsoft NTLM-based, 967
  - AUTHID clause, 6
    - examining in USER\_PROCEDURES view, 757
    - in package specification, 659
    - in procedures, 594
  - AUTHID CURRENT\_USER clause, 562, 1054
  - AUTHID DEFINER clause, 587, 1055
  - automatic runtime compilation, 773
  - autonomous transactions, 462, 477–484
    - audit trails and, 1037
    - building autonomous logging mechanism, 482
    - defining, 478
    - rules and restrictions on, 479
    - transaction visibility, 480
    - when to use, 481
  - AUTONOMOUS\_TRANSACTION pragma, 77
  - autostartup scripts (SQL\*Plus), 35
  - AUTO\_LEXER, 1122
- ## B
- back references in regular expressions, 225
  - backtracing error, 145–148
  - base type, creating, 1144
  - basic authentication, 959
  - BasicFiles, 436, 983
  - BEFORE triggers, 689
    - BEFORE DDL trigger, 716
    - BEFORE EACH ROW trigger, 707
    - BEFORE INSERT trigger, 692
    - BEFORE STATEMENT trigger, 707

- multiple row-level BEFORE INSERT triggers, 702
- BEQUEATH CURRENT\_USER clause for views, 13, 1061
- BETWEEN operator, 270
- BFILE type, 179, 420
  - accessing BFILES, 433
  - creating a BFILE locator, 432
  - difference from internal LOBs, 431
  - sending email attachment as, 953
  - using BFILES to load LOB columns, 434
- BFILENAME function, 433
- binary data
  - case-insensitive sorts, 208
  - datatypes, 179
- binary files (see BFILE type)
- binary float (32-bit) or binary double (64-bit)
  - floating-point literals, 73
- binary large object (see BLOB type)
- binary sort, 1116
- BINARY\_DOUBLE and BINARY\_FLOAT
  - types, 177, 242, 251–256
    - Oracle implementation versus IEEE-754 standard, 255
  - performance, BINARY\_DOUBLE versus NUMBER type, 253
  - using for floating-point arithmetic, 922
- BINARY\_INTEGER type, 242, 249
- bind operations (cursor), 488
- bind values, 1030
- bind variables
  - reducing memory use with, 1081–1084
  - supplying bind arguments with USING clause of EXECUTE IMMEDIATE, 539
  - supplying bind arguments with USING clause of OPEN FOR, 547
  - use in statement sharing, 1079
  - using FGA with, 1035
  - using in NDS to minimize dangers of code injection, 567
- binding variables, 550–557
  - argument modes, 551
  - duplicate placeholders, 553
  - passing NULL values as bind arguments, 554
- binding, using instead of concatenation in NDS, 564
- BITAND function, 272
- blank-padding comparisons, 230
- BLOB type, 179, 420
  - BasicFiles, 983
  - loading from BFILES, 434
  - reading from, 430
  - retrieving web page into a BLOB, 958
  - selecting BLOB column into a BLOB PL/SQL variable, 423
  - writing into BLOBs, 429
- block-structured language, 57
- blocks, 53–65
  - anonymous, 55
  - building dynamic PL/SQL blocks, 558
  - labeling, 77
  - named, 57
  - nested, 57
  - qualifying references to variables and columns in SQL statements, 59
  - replacing repetitive code with dynamic blocks, 560
  - scope, 58
  - sections, 53
  - visibility of variables, 62
- body
  - of a cursor, 503
  - of a function, 605
  - of a loop, 107
  - of a package, 652, 654, 656
    - rules for building, 660
  - of a procedure, 597
- Booch diagram of private and public package elements, 657
- books on PL/SQL, 15
- Boolean literals, 74
- BOOLEAN type, 178, 415
- Booleans
  - binding PL/SQL-specific Boolean with EXECUTE IMMEDIATE, 541
  - displaying Boolean values with DBMS\_OUTPUT, 927
  - using Boolean variables as flags for expression evaluation, 87
- bottlenecks in PL/SQL code, identifying, 827–832
- boundary (loop), 107
- bounded collections, 344
- browsers
  - Unicode character encodings, 1103
- buffer
  - reading contents of, 928

- writing lines to, 926
- bugs, avoiding through qualifiers, 61
- BULK COLLECT clause, 870–877
  - array fetches with, using to tune pipelined functions, 893
  - bulk fetching of multiple columns, 874
  - FETCH FIRST clause and, 14
  - high-speed querying with, 870
  - LIMIT clause with, reducing memory usage, 1088
  - limiting rows retrieved with, 873
  - populating collections with data, 377
  - rules and restrictions, 871
  - using in FORALL statement with RETURNING clause, 468, 875
- bulk processing, 488, 869–888
  - BULK COLLECT INTO clause in SELECT statements, 494
  - high-speed DML with FORALL, 877–888
- %BULK\_EXCEPTIONS attribute, 492
- %BULK\_ROWCOUNT attribute, 492, 881
- BY REFERENCE, 917, 1263, 1265
- BY VALUE, 917, 1263, 1265

## C

- C language
  - calling PL/SQL from, 46
  - external procedure written in, call spec for, 1256–1266
  - extprocsh function, 1244
  - raising exception from C program called in PL/SQL, 1266–1269
  - shared libraries callable from, 1244
- caching, 844–868
  - collection caching technique, 377
  - with deterministic functions, 850–852
  - function result cache, 852–868
    - details of its behavior, 861
    - enabling, 853
    - example, caching a collection, 858
    - example, deterministic function, 855
    - example, querying data from a table, 856
    - fine-grained dependencies (11.2 and higher), 863
    - managing, 861
    - RELIES\_ON clause, 854
    - when not to use, 860
    - when to use, 859
  - package-based, 845–850
    - caching table contents, 848
    - example of, 846
    - just-in-time caching of table data, 850
    - when to use, 845
  - session cursor cache, 1079
  - static session data, 663, 684
  - summary of recommendations for, 868
- calculated columns, 507
- calendars, NLS\_CALENDAR setting, 1130
- call specs, 1228, 1244
  - rules for, 1229
  - writing for external procedure, 1256–1266
    - overall syntax of call spec, 1257
    - parameter mapping, 1258–1262
    - PARAMETERS clause, 1262–1266
- CALL statement, 41
- cardinality for pipelined table functions, 904–909
- carriage return character, 206
- case
  - case-insensitive indexes, 446
  - case-insensitivity in PL SQL, 66, 174
  - case-sensitivity in Java, 1215
  - handling in strings, 207
    - capitalizing each word, 209
    - case-insensitivity and indexes, 209
    - forcing string to all upper- or lowercase, 208
    - making comparisons case insensitive, 208
  - returning strings in upper- or lower-case for national languages, 235
  - setting first letter of each word to uppercase with INITCAP, 233
  - setting first letter of word to uppercase with NLS\_INITCAP, 235
- CASE statements and expressions, 93–100
  - CASE expressions, 98
  - CASE statement, 8
    - nested CASE statements, 97
    - searched CASE statements, 95
    - simple CASE statements, 93
    - using NULL statement in, 102
- CASE\_NOT\_FOUND error, 94
- CAST function, 194, 267
  - casting DATES to TIMESTAMPS, 316

- converting `TIMESTAMP WITH TIME ZONE` to `TIMESTAMP WITH LOCAL TIME ZONE`, 1129
  - supported conversions between built-in datatypes, 194
  - using with date and time values, 283
  - using with datetime values, 308
- CAST pseudofunction, 399
- CAST/MULTISET clause, 1167
- casting
  - narrowing or downcasting, using `TREAT` function, 1160
  - NULL to different types, 72
  - widening or upcasting, 1148
- CBC (Cipher Block Chaining), 977, 979
- CBO (cost-based optimizer), pipelined functions and, 903–909
  - cardinality heuristics for pipelined table functions, 904
  - extensible optimizer and pipelined function cardinality, 906
  - providing cardinality statistics to optimizer, 906
  - using optimizer dynamic for pipelined functions, 905
- CEIL function, 271, 272
- century, RR format element and, 299
- certificates, 960
- CGA (call global area), 1041, 1085
- chaining, 977
  - Cipher Block Chaining (CBC), 979
  - DBMS\_CRYPTO chaining constants, 978
  - method, 1155
- CHAIN\_CBC package, 977
- CHAIN\_CFB package, 977, 977
- CHAIN\_ECB package, 977
- CHAR type, 176
  - assigning zero-length string to, 72
  - conversion to ROWID, 193
  - declaring, 201
  - empty strings and, 228
  - maximum length, 202
  - mixing with VARCHAR2 values in strings, 229
  - TO\_CHAR function, 238, 261–267
  - use in argument definitions, 758
- character data, types, 176
- character encodings
  - defined, 1100
  - Unicode, 1102
- character functions and CHAR arguments, 231
- character large object type (see CLOB type)
- character semantics, 1111–1115
- character sets
  - AL32UTF8, required by ENCRYPT function, DBMS\_CRYPTO, 981
  - choosing, 1101
  - converting string from database character set to target character set, 232
  - defined, 1100
  - Oracle database, 1101
  - PL/SQL character set, 65
- CHARSETID and CHARSETFORM properties, PARAMETERS clause, 1265
- child blocks, 58
- child rows, 135
- CHR function, 205, 232
- Cipher Block Chaining (CBC), 977
- Cipher Feedback (CHAIN\_CFB), 977
- CLASSPATH environment variable, 1208
- CLOB type, 176, 421
  - applying UPPER function to, 445
  - BasicFiles, 983
  - empty versus NULL CLOBs, 426
  - loading from BFILES, 434
  - reading from, 430
  - using interchangeably with VARCHAR2, 442
- CLOSE cursor statement, 508
  - for packaged cursors, 509
- closing cursors, 489, 1078
- CLUSTER streaming, 902
- COBOL
  - PL/SQL embedded in, 52
  - Pro\*Cobol precompiler, 47
- code
  - managing (see managing code)
- code blocks (see blocks)
- code examples
  - conventions used in, xxxi
  - downloading, xxxii
- code injection
  - minimizing dangers of in NDS, 566
  - preventing with use of DBMS\_ASSERT, 1084
  - privilege escalation and, using definer rights model, 1053

- code points
  - defined, 1100
  - multiple ways to represent same thing, 1106
  - syntax of, 1102
- code, creating and running, 21–52
  - calling PL/SQL from other languages, 45–52
  - creating a stored program, 37–41
  - creating and editing source code, 22
  - database navigation, 22
  - dropping a stored program, 43
  - editing environments for PL/SQL, 45
  - executing a stored program, 41
  - hiding source code of stored programs, 44
  - managing grants and synonyms for stored programs, 42
  - showing stored programs, 42
  - using SQL\*Plus, 23–37
- COLLECT pseudofunction, 400
- collections, 341–413
  - accessing data in, 379
  - based on DBMS\_SQL.DESC\_TAB collection type, 569
  - caching, using function result cache, 858
  - choosing a type, 354
  - of complex datatypes, 385–389
    - collections of records, 386
    - objects and other complex types, 389
  - concepts and terminology, 343
    - collection or collection instance, 343
  - conversions between type, using CAST, 195
  - declaring and initializing collection variables, 369–374
    - initializing implicitly during direct assignment, 371
    - initializing implicitly via FETCH, 372
    - VARRAY integration, 373
  - declaring collection types, 365–369
    - associative array, 365
    - nested table or VARRAY, 368
  - examples of, 345–349
    - using a nested table, 347
    - using a VARRAY, 348
    - using an associative array, 346
  - improving views with, 1185
  - large, memory use and, 1085
  - memory consumption by, 827
  - methods (built-in), 356–365
    - COUNT method, 357
    - DELETE method, 358
    - EXISTS method, 359
    - EXTEND method, 360
    - FIRST and LAST methods, 361
    - LIMIT method, 362
    - PRIOR and NEXT methods, 362
    - TRIM method, 363
- multilevel, 389–398
  - exploring multidim API, 393
  - extending string\_tracker with, 396
  - number of levels of nesting, 398
  - unnamed, emulation of multidimensional arrays, 391
- nested table multiset operations, 406–412
  - handling duplicates, 411
  - high-level set operations, 409
  - testing equality and membership, 408
- NULL, 1190
- object view with collection attribute, 1188–1191
- passing into an array in Java, 1241
- populating with data, 374–379
  - aggregate assignments, 375
  - assigning rows from relational table, 376
  - index values, 375
  - nonsequential population, advantages of, 377
  - using the assignment operator, 374
- schema-level, maintaining, 412
- string-indexed, 380–385
  - emulating primary keys and unique indexes, 383
  - other examples of, 385
  - performance of, 384
  - simplifying algorithmic logic with string indexes, 381
- types of, 345
- uses of, 341
- where to use, 349–354
  - as attributes of an object type, 354
  - as columns in database table, 352
  - as components of a record, 350
  - as datatype of function's return value, 351
  - as program parameters, 350
- working with in SQL, 398–406
  - CAST pseudofunction, 399
  - COLLECT pseudofunction, 400
  - MULTISET pseudofunction, 400
  - sorting contents, 405
  - TABLE pseudofunction, 402



- working with, using NDS, 555–557
- column aliases, 507
- column objects, 1179
- column-sensitive RLS, 1012–1015
- columns
  - anchoring a variable to, 184
  - calculated or virtual columns, 507
  - collections as columns in database table, 352
  - determining which column was altered in
    - ALTER TABLE, 716
  - invisible (Oracle Database 12c), 339
  - pseudo-columns, 419
- command-line interpreter for SQL and PL/SQL, 23
- COMMENT keyword (in COMMIT statement), 474
- comments, 75
  - (double dash), single-line comment indicator, 66
  - /\* \*/ multiline comment block delimiters, 66
  - lines beginning with REM in SQL\*Plus, 30
- COMMIT statements, 474
  - autonomous transactions and, 480
  - DML triggers and, 690
  - releasing locks with, 517
- comparison methods, 1151
- comparisons, string
  - making comparisons case insensitive, 208
  - mixing CHAR and VARCHAR2 types, 229
- compilation
  - conditional (see conditional compilation)
  - native, 1094
  - of PL/SQL programs, modifications of SQL statements, 1044
  - recompiling invalid program units, 773–777
  - setting compilation environment parameters, 1068
- compile-time warnings, 777–788
  - categories of, 778
  - determining if stored programs have disabled warnings, 757
  - enabling, 778
  - example, 777
  - some handy warnings, 780–788
- compiler directives, 1065–1076
  - new conditional compilation directives, 14
- compiler optimization levels, using most aggressive level, 826
- compilers
  - Ada, DIANA, 1040
  - GCC (GNU C compiler), 1245
  - getting and setting options with
    - DBMS\_JAVA programs, 1224
  - Java (jacac.exe), 1215
  - limits of PL/SQL compiler, 1045
  - optimizing compiler, 838–843
  - SQL, shared by PL/SQL and database, 1043
- complex object retrieval (COR), 1169
- COMPOSE function, 232, 1106
- composite data, 173
- compound triggers, 706–710
- compression
  - LOBs, 437
  - zip/compression functionality in PL/SQL, 1240
- CONCAT function, 206, 232
- concatenation
  - binding versus, for dynamic SQL strings, 564
  - using for NDS strings, 550
- concatenation operator (||), 66, 205, 206
- conditional compilation, 1064–1076
  - \$ERROR directive, 1072
  - \$IF directive, 1070
  - examples of, 1065
  - inquiry directives, 1066–1070
  - new directives in Oracle Database 12c, 14
  - program-specific settings with inquiry directives, 1073
  - synchronizing code with packaged constants, 1072
  - working with postprocessed code, 1074
- conditional control statements, 8, 83
  - CASE statements and expressions, 93–100
    - CASE expressions, 98
    - nested CASE statements, 97
    - searched CASE statements, 95
    - simple CASE statements, 93
  - IF statements, 83–92
    - avoiding syntax gotchas, 89
    - IF-THEN, 84
    - IF-THEN-ELSE, 86
    - IF-THEN-ELSIF, 87
    - nested, 90
    - short-circuit evaluation, 91
- CONNECT command (SQL\*Plus), 25, 33
- connect identifiers (Oracle Net), 25

- connection pools, 1026
- console program, SQL\*Plus, 23
- constants
  - building package of named constants, 681
  - declaring, 182
  - in program data, 173
  - naming, 174
  - specifying string constants, 203
- constrained and unconstrained declarations, 608
- constrained subtypes, 188
- constructor methods, 1147–1150
- containers, 173
- CONTAINS predicate, 446
- CONTEXT index (Oracle Text), 1121
- CONTEXT keyword, 1262
- context switches, 869
  - one switch with FORALL statement, 870
  - reducing with UDF pragma, 922
  - reduction with BULK COLLECT, 870
- context-sensitive policies, 1010, 1025
  - shared, 1012
- CONTINUE statements, 121–123
  - CONTINUE WHEN, 121
  - GOTO versus, 122
- control statements, 8
- conversion between datatypes, 189
- CONVERT function, 196, 232
- cookies, disabling or making persistent, 965
- Coordinated Universal Time (see UTC)
- cost-based optimizer (see CBO, pipelined functions and)
- COUNT method, 357
- CPAN (Comprehensive Perl Archive Network), 48
- CREATE CONTEXT statement, 1020
- CREATE JAVA statement, 1206, 1219
- CREATE LIBRARY privilege, 1245
- CREATE LIBRARY statement, 1255
- CREATE OR REPLACE TYPE BODY statement, 1163
- CREATE PROCEDURE privilege, 38
- CREATE statements
  - appending SHOW ERRORS after statements
    - building stored programs, 40
  - conversion to plain text and hex, 44
  - CREATE FUNCTION, 37
  - CREATE OR REPLACE FUNCTION, 38
- CREATE TABLE statement, OBJECT IDENTIFIER IS PRIMARY KEY clause, 1156
- CREATE\_WRAPPED program of DBMS\_DDL package, 819
- creating and running PL/SQL code (see code, creating and running)
- creative or radical approach, taking, 19
- cryptographic hashing, 991
- cryptography, 972
  - (see also encryption)
  - DMBS\_CRYPTO package, 977
- CTXCAT, CTXRULE, and CTXXPATH indexes (Oracle Text), 1121
- currency
  - conversions, 1131–1132
  - NLS\_CURRENCY settings, 1104
  - number format elements specifying symbols, 1282
- current directory (SQL\*Plus), 30, 37
- current row, 487
- CURRENT\_DATE function, 282
- CURRENT\_TIMESTAMP function, 282, 283
- cursor attributes
  - %BULK\_EXCEPTIONS, 492
  - %BULK\_ROWCOUNT, 492
  - %FOUND, 490
  - %ISOPEN, 491
  - %ROWCOUNT, 491
  - defined, 488
  - for DML operations, 466
  - explicit cursor attributes, 510
  - for FORALL statements, 881
  - implicit SQL cursor attributes, 498
  - inability to use in SQL statements, 492
  - in PL/SQL, summary of, 489
  - referencing for cursor variables, 521
- cursor expressions, 533–536
  - restrictions on, 536
  - syntax, 533
  - using to implement a streaming function, 535
  - using to retrieve subquery as a column, 534
- cursor FOR loops, 106, 117–119
  - declaring a record implicitly, 327
  - example of, 118
  - example of implicit use of %ROWTYPE declaration, 186
  - moving multiple rows into a collection, 376
  - obtaining information about execution, 126

- retrieving and displaying data for dynamic cursor, 577
- runtime optimization of, 843
- termination issues, 125
- CURSOR operator, 533
- cursor variables, 179, 519–533
  - declaring, 522
  - declaring REF CURSOR types, 521
  - defined, 488
  - fetching from, 524
    - handling ROWTYPE\_MISMATCH exception, 525
  - opening, 523
  - passing as arguments, 530
    - identifying REF CURSOR type, 530
    - setting parameter mode, 531
  - passing table function results with, 640
  - reasons to use, 520
  - referencing across two programs, 520
  - restrictions on, 532
  - rules for, 527
    - compile-time rowtype matching, 527
    - cursor variable aliases, 529
    - runtime rowtype matching, 528
    - scope of cursor object, 529
  - similarities to static cursors, 521
  - support for dynamic SQL, 543
- cursor-based records, 326
- cursors, 485–493
  - anchoring to, 186
  - converting cursor number to weakly typed cursor variable, 580
  - converting REF CURSOR variable to SQL cursor number, 582
  - cursor variable versus cursor object, 523
  - data retrieval terminology, 487
  - declaring and using, 486
  - defining structure in dynamic SQL method 4 with DBMS\_SQL, 575
  - enhanced security for DBMS\_SQL package, 584
    - denial of access when bad cursor number is used, 584
    - denial of DBMS\_SQL operation with change of effective user, 586
    - unpredictable cursor numbers, 584
  - explicit and implicit, choosing between, 493
  - explicit cursors, working with, 500–515
  - implicit cursors, working with, 494–500

- memory usage, 1077
- minimizing parsing of dynamic cursors with DBMS\_SQL, 578
- packaged, 659, 660, 669–674
  - declaring, 670
  - working with, 672
- referencing PL/SQL variables in, 492
- remote invocation model and, 772
- SELECT FOR UPDATE statement in, 515–519
- typical query operations, 488
- CURSOR\_ALREADY\_OPEN exception, 138
- CURSOR\_SHARING initialization parameter, 1084
- custom-named directives, 1066

## D

- \d (digit), in regular expressions, 221
- dangerous characters in dynamic text, 568
- dangling REFs, 1171
- Data Definition Language (see DDL)
- data dictionary entries for object types, 1197
- data dictionary views, 751
  - for collections, 413
  - dependency analysis with, 763–767
  - for external procedures, 1272
  - Java class information in, 1222
- Data Encryption Standard (DES) algorithm, 976
- Data Manipulation Language (see DML)
- data retrieval terminology, 487
- data structures declared at package level, 503
- Database Error Messages document, 39
- database event triggers, 688, 720
  - creating, 721
  - LOGOFF, 723
  - LOGON, 723
  - SERVERERROR, 724–728
    - central error handler, 727
    - examples of use, 725
  - SHUTDOWN, 723
  - STARTUP, 722
- database pipes, 111, 967
- database sessions, 1041
- database triggers, 687–746
  - AFTER SUSPEND, 736–743
    - DBMS\_RESUMABLE package, 740
    - deciding whether to fix problems, 743
    - examining the actual trigger, 738

- ORA\_SPACE\_ERROR\_INFO function, 739
- setting up, 736
- trapped multiple times, 742
- analyzing and modifying state through views, 757
- anonymous blocks in, 56
- checking validity of, 746
- database event triggers, 720–728
- DDL, 710–720
  - attribute functions for trigger events, 713
  - available events for which triggers can be coded, 713
  - creating, 710
  - INSTEAD OF CREATE trigger, 719
  - preventing DROP operations, 719
  - working with events and attributes, 715–719
- defined, 593
- disabled, creating, 744
- disabling, enabling, and dropping, 743
- DML, 688–710
  - compound triggers, 706–710
  - concepts and terminology, 689
  - creating, 691
  - determining DML action in a trigger, 695
  - example (No Cheating Allowed), 696–702
  - examples of usage, 692
  - flow of control, 689
  - multiple triggers of same type, 702
  - mutating table errors, 705
  - order of firing, 703
  - WHEN clause, 692
  - working with NEW and OLD pseudo-records, 694
- INSTEAD OF, 728–736
  - creating, 728
  - for updating object views, 1193
  - INSTEAD OF DELETE trigger, 733
  - INSTEAD OF INSERT trigger, 730
  - INSTEAD OF UPDATE trigger, 732
  - on nested tables, 734
  - populating the tables, 733
- pseudorecords, 338
- types of events having trigger code attached, 687
- uses of, 687
- using with RLS, 999
- viewing, 745

datatypes, 175–181, 415–456

- anchored, 183
- benefits of, 186
- to cursors or tables, 185
- Any types, 180
- binary data, 179
- binding in NDS statements, 540
- binding PL/SQL-specific Boolean with EXECUTE IMMEDIATE, 541
- BOOLEAN, 415
- Booleans, 178
- character data, 176
- collection as datatype of function's return value, 351
- collection type, 343
- collection, declaring, 365–369
- conversions between, 189–197
  - explicit conversions, 192–197
  - implicit conversion, 190
- date and time, 278–288, 278
  - conversions, 289–302
- dates, timestamps, and intervals, 178
- definitions in STANDARD package, 1048
- functions called from SQL, 632
- Internet, 180
- LOB types, 420–447
- LONG and LONG RAW, 421
- mapping Java types to SQL types, 1230
- mappings of PL/SQL and C types, 1260
- more PL/SQL-only datatypes crossing PL/SQL-to-SQL interface, 12
- naming, 175
- national character set, 1102
- NOT NULL, anchoring to, 188
- numbers, 177
- numeric, 241–257
- of variables, 181
- performance improvement using right type, 921
- predefined object types, 447–456
- programmer-defined record type, declaring, 328
- programmer-defined subtypes, 188
- RAW, 417
- REF CURSOR, 179
- RETURN datatype in functions, 601
- ROWID, 417
- ROWIDs, 179

- string, 199–203
  - mixing CHAR and VARCHAR2 values, 229
  - type evolution, 1162
  - UROWID, 417
  - user-defined, 181
- date format model, 289
  - FF format element, 293
  - FM (fill mode) format modifier, 299, 302
  - format elements for TIMESTAMP versus DATE type, 294
  - format\_mask parameter, datetime conversion functions, 290
  - FX modifier in format masks, 298
  - RR element, interpreting two-digit years, 299
  - rules for, 291
  - using with TO\_CHAR function, 292
- date literals, 302
- DATE type, 278
  - computing interval between DATE values, 314
  - considerations when choosing, 281
  - mixing with TIMESTAMPs in datetime arithmetic, 316
- dates and time, 277–322
  - calculating elapsed time for programs, 833
  - datetime arithmetic
    - computing interval between datetimes, 313
  - datatypes in PL/SQL, 178
  - date and timestamp literals, 71, 302
  - date format models, 1285–1289
  - date/time function quick reference, 319
  - datetime arithmetic, 311–319
    - adding and subtracting intervals, 317
    - date arithmetic with DATE datatypes, 312
    - mixing DATES and TIMESTAMPs, 316
    - multiplying and dividing intervals, 317
    - using unconstrained INTERVAL types, 318
    - with intervals and datetimes, 311
  - datetime conversions, 289–302
    - easing up on exact format mask matches, 299
    - from datetimes to strings, 292
    - from strings to datetimes, 289
    - interpreting two-digit years, 299
    - padding output with fill mode (FM), 302
    - requiring format mask to match exactly, 298
    - time zones, 295
    - time zones to character strings, 301
    - with CAST function, 194
  - datetime datatypes, 278–282
    - choosing a datatype, 281
    - declaring datetime variables, 280
  - getting current date and time, 282
  - globalization and localization, 1126–1131
    - date/time formatting, 1127–1131
    - TIMESTAMP datatypes, 1126
  - interval conversions, 304–306
  - interval datatypes, 284–288
  - interval literals, 307
  - using CAST function, 308
  - using EXTRACT function, 310
- daylight saving time, 296
- DBA\_\* views, 751
- DBA\_JAVA\_POLICY view, 1211
- DBA\_POLICIES data dictionary view, 1005
- DBA\_RESUMABLE view, 737
- DBA\_SOURCE view, 1197
- DBD::Oracle driver (Perl), 48
- DBI (Database Interface), Perl, 48
- dbmsany.sql script, 453
- dbmshptab.sql script, 831
- DBMS\_ALERT package, 968
- DBMS\_APPLICATION\_INFO package, 796
  - using for tracing, 801
- DBMS\_ASSERT package
  - using to avoid code injection, 1084
  - using to validate inputs, 569
- DBMS\_CRYPTO package, 977–979, 996
  - algorithm constants, 978
  - DECRYPT function, 983
  - ENCRYPT function, 979
  - HASH function, 992
  - MAC function, 993
  - overloaded procedure version of ENCRYPT, 982
  - padding and chaining constants, 978
  - RANDOMBYTES function, 985
- DBMS\_DB\_VERSION package, 1067
- DBMS\_DESCRIBE package, 759
- DBMS\_HPROF (hierarchical profiler), 828, 830
  - calling ANALYZE utility, 831
  - components of, 830

- creating profiler tables and necessary database objects, 831
- obtaining profile information, 831
- steps in using, 830
- DBMS\_JAVA package, 1206, 1223–1228
  - EXPORT\_SOURCE, EXPORT\_RESOURCE, AND EXPORT\_CLASS, 1226
  - getting and setting compiler options, 1224
  - GRANT\_PERMISSION procedure, 1210
  - LONGNAME function, 1223
  - SET\_OUTPUT procedure, 1225
- DBMS\_LOB package, 427
  - predefined exceptions in, 137
  - reading past end of a BFILE, 936
- DBMS\_OBFUSCATION\_TOOLKIT, 996
- DBMS\_OUTPUT package, 925
  - CASE expression used with, 99
  - displaying information with PUT or PUT\_LINE, 926
  - enabling, 926
  - flushing the buffer with NEW\_LINE, 926
  - PUT\_LINE function, 41
  - reading buffer contents with GET\_LINE or GET\_LINES, 928
  - writing lines to the buffer with PUT\_LINE or PUT, 926
- DBMS\_PIPE package, 668, 967
- DBMS\_PREPROCESSOR package, 1074
- DBMS\_PROFILER package, 827
  - creating PLSQL\_PROFILER tables, 828
  - running queries against data in PLSQL\_PROFILER tables, 829
- DBMS\_RESUMABLE package
  - ABORT procedure, 740
  - getting and setting timeout values, 741
- DBMS\_RLS package
  - ADD\_POLICY function, 1004
    - policy\_type parameter, 1009
    - statement\_types parameter, 1005
    - static\_policy parameter, 1009
    - update\_check parameter, 1007
  - DROP\_POLICY function, 1006
- DBMS\_SCHEDULER package, 52
- DBMS\_SESSION package
  - clearing out memory used by packaged collections, 1087
  - SET\_CONTEXT, 1021, 1022
- DBMS\_SHARED\_POOL.SIZES procedure, 1091
- DBMS\_SQL package, 551
  - dynamic SQL, 538
  - for dynamic SQL method 4, 548
  - PARSE, 1056
  - using to gain explicit control over low-level cursor behavior, 1079
  - when to use, 569–587
    - enhanced security for DBMS\_SQL, 584
    - meeting method 4 dynamic SQL requirements, 571–578
    - minimizing parsing of dynamic cursors, 578
    - obtaining information about query columns, 569
    - Oracle Database 11g new dynamic SQL features, 580–584
- DBMS\_STANDARD package, 1045
  - DDL trigger event and attribute functions, 713
  - RAISE\_APPLICATION\_ERROR procedure, 141
  - server error functions, 724
- DBMS\_TRACE package, 796, 804
  - constants controlling elements to be traced, 805
  - constants controlling tracing process, 805
  - controlling trace file contents, 806
  - format of collected data, 807
  - installing, 805
  - pausing and resuming trace process, 807
  - subprograms in the package, 805
- DBMS\_TYPES package, 1172
- DBMS\_UTILITY package
  - FORMAT\_CALL\_STACK function, 145, 796, 1069
  - FORMAT\_ERROR\_BACKTRACE function, 145, 1069
  - FORMAT\_ERROR\_STACK function, 145
  - GET\_CPU\_TIME, 1011
  - GET\_CPU\_TIME function, 833
  - GET\_TIME function, 833
    - encapsulation in sf\_timer package, 834
  - RECOMPILE\_SCHEMA procedure, 776
- DBMS\_WARNING package, 780
- DBTIMEZONE function, 283
- DBUri-REFs, 180
- DBUriType, 451
- DB\_BLOCK\_SIZE initialization parameter, 904
- DB\_SECUREFILE initialization parameter, 437

- DDL (Data Definition Language), 461
  - executing statements with dynamic SQL, 539, 548
  - invalid operations in system triggers, 738
  - triggers, 710–720
    - attribute functions for, 713
    - available events for triggers, 713
    - creating, 710
    - INSTEAD OF CREATE, 719
    - preventing DROP operations, 719
    - working with events and attributes, 715–719
- debugging, 808–815
  - DEBUG privilege for object types and object views, 1200
  - external procedures, 1270
  - factors making it difficult, 808
  - RLS (row-level security), 1015–1019
  - tips and strategies for, 811
    - analyzing instead of trying, 813
    - changing and testing one area of code at a time, 815
    - gathering data, 811
    - remaining logical, 812
    - taking breaks and asking for help, 814
    - using source code debugger, 811
  - tracing versus, 795
  - wrong ways to debug, 809
- decimal numbers
  - binary types and, 252
  - fixed-point, representing, 247
- declarations
  - constrained and unconstrained, 608
  - declaration section in code blocks, 54
    - anonymous block, 56
    - nested programs in, 64
  - forward, 630
  - in procedures, 594
  - termination with semicolon, 75
- DECOMPOSE function, 232, 1107
- DECRYPT function (DBMS\_CRYPTO), 983
- DEFAULT operator
  - setting default values for variables, 181
  - specifying default values for parameters, 618
  - supplying default value for constants, 182
  - supplying default values for individual fields in a record, 329
- DEFINE command (SQL\*Plus), 31
- definer rights model, 462, 1049–1053
  - advantages of, 1050
  - AUTHID clause, 6
  - combining with invoker rights model, 1056
  - disadvantages of, 1051
    - dynamic SQL and definer rights, 1052
    - maintaining code, 1052
    - privilege escalation and SQL injection, 1053
- DELETE method, 358
- DELETE statements, 465
  - using FORALL with, 878
  - WHERE CURRENT OF clause, 518
- DELETE\_FAILED exception, 169
- DELETING function, 695
- deliberate exceptions, 160
  - guidelines for dealing with, 162
- delimiters for strings, specifying your own, 203
- dense collections, 344
- dependencies
  - analyzing with data dictionary views, 763–767
  - checking before dropping libraries, 1272
  - defined, 762
  - fine-grained dependencies in 11.2 and higher, function result cache and, 863
  - fine-grained dependency tracking, 763, 767
  - remote, 769
- DEPTREE view, 766
- DEREF function, 1166, 1170
- DES (Data Encryption Standard) algorithm, 976
- DES3 (Triple DES) algorithm, 976
- DESCRIBE command (SQL\*Plus), 42
- deterministic functions, 647
  - caching using function result cache, 855
  - caching with, 850–852, 868
- developers, resources for, 14
- DIANA (Distributed Intermediate Annotated Notation for Ada), 1040
- direct path operations, RLS policies and, 1017
- directives, 1065–1076
- directories
  - current directory for SQL\*Plus, 30, 37
  - obtaining contents using Java class, 1239
  - setting up with UTL\_FILE, 930
  - UTL\_FILE\_DIR parameter, 929
  - working with Oracle directories, 931
- disabled triggers, creating, 744
- DISCONNECT command (SQL\*Plus), 33



- disorganized debugging, 809
- displaying information (see I/O)
- DISTRIBUTED\_LOCK\_TIMEOUT initialization parameter, 516
- division operator ( / ), 270
- DML (Data Manipulation Language), 461–473
  - cursor attributes for DML operations, 466
  - DELETE statement, 465
  - DML privileges for object types, 1200
  - DML statements in SQL, 463
  - and exception handling, 468
  - high-speed DML with FORALL, 877–888
  - INSERT statement, 463
  - MERGE statement, 466
  - performing on object relational tables, 1159
  - and records, 470
  - RETURNING clause in statements, 875
  - returning information from DML statements, 468
  - set-based versus row-based, performance of, 892
  - statements available in SQL, 463
  - techniques for encapsulating DML on object views, 1194
  - triggers, 688–710
    - compound triggers, 706–710
    - concepts and terminology, 689
    - creating, 691
    - determining DML action in a trigger, 695
    - example (No Cheating Allowed), 696–702
    - examples of usage, 692
    - flow of control, 689
    - multiple triggers of same type, 702
    - mutating table errors, 705
    - order of firing, 703
    - scripts for, 690
    - transaction participation, 690
    - WHEN clause, 692
    - working with NEW and OLD pseudo-records, 694
  - UPDATE statement, 464
- downcasting, 1160
  - REFs, 1170
- DROP JAVA statement, 1206
- DROP LIBRARY statement, 1272
- DROP operations, DDL trigger preventing, 719
- DROP statements
  - DROP SYNONYM, 43
  - dropping stored programs, 43
- DROP TRIGGER command, 744
- DROP TYPE statement, 1164
- dropjava command, 1206
  - using, 1221
- DSINTERVAL\_UNCONSTRAINED type, 319
- duplicates
  - deduplication of LOBs with SecureFiles, 437
  - handling in nested tables, 411
- dynamic method dispatch, 1148
- dynamic polymorphism, 625
- dynamic sampling, 905
- dynamic SQL and dynamic PL/SQL, 537–587
  - binding variables, 550–555
  - definer rights and dynamic SQL, 1052
  - dynamic PL/SQL, 557–561
    - building dynamic blocks, 558
    - replacing repetitive code with dynamic blocks, 560
  - dynamic SQL, 486, 487
  - four methods of dynamic SQL, 548
  - invoker rights for dynamic SQL, 1055
  - NDS statements, 538–550
  - recommendations for NDS, 561–569
  - when to use DBMS\_SQL, 569–587
  - working with objects and collections, 555–557
- dynamically typed programming languages, 175

## E

- Eastern Time versus Eastern Standard Time, 296
- EDIT command (SQL\*Plus), 33
- editing environments for PL/SQL, 45
- edition-based redefinition, 822
- editors
  - default external editors assumed by Oracle, 34
  - for source code, 22
  - using SQL\*Plus built-in line editor, 34
- elapsed time, calculating for programs, 833
- Electronic Code Book format (CHAIN\_ECB), 977
- elements (collection), 343
- ELSE keyword, 86
  - explicit ELSE clause that does nothing, 101
  - in searched CASE statements, 96
  - in simple CASE statements, 94
- ELSIF keyword, 86, 88



- email, sending, 944–955
  - attaching file of arbitrary size, 953
  - configuring network security, 946
  - including friendly names in addresses, 948
  - message with short attachment, 951
  - Oracle prerequisites for, 945
  - plain-text message of arbitrary length, 950
  - short, plain-text message, 947
  - small file as attachment, 953
- embedded languages, 21
- empty LOBs, 425
- empty strings, 206
  - working with, 227
- EMPTY\_CLOB function, 426
- encapsulation, 483
  - data encapsulation in packages, 677–680
  - single-row queries, 493
- enclosed blocks, 58
- enclosing block, 58
- ENCRYPT function (DBMS\_CRYPTO), 979
- ENCRYPT procedure (DBMS\_CRYPTO), 982
- encryption, 973–999
  - algorithms, 975
  - basic components, 973
  - DBMS\_CRYPTO package, 977–979
  - decrypting data, 983
  - encrypting data, 979
  - encrypting LOBs, 982
  - Exadata and, 999
  - key generation, 985
  - key management, 985–991
  - LOBs, with SecureFiles, 438
  - padding and chaining, 977
  - summary of, 996
  - using cryptographic hashing instead of, 991
  - using message authentication codes (MACs), 993
  - using TDE (transparent data encryption), 995
  - using TTE (transparent tablespace encryption), 997
  - wrap utility and, 44
- END IF keyword
  - and formatting of IF statements, 85
  - closing executable IF statements, 89
- END keyword
  - terminating CASE expressions, 99
- END label
  - for functions, 602
  - for packages, 659
  - for procedures, 597
- END LOOP statement, 106
- end-of-file, reaching, 936
- entry point for loops, 124
- environment variables
  - CLASSPATH, for Java compilation, 1208
  - PATH, for Java compiler, 1215
  - setting up for listeners for external procedures, 1250
- environment, Unicode and, 1103
- equality operator (see = (equals sign), under Symbols)
- equality, testing for nested tables, 406, 408
- error codes
  - application-specific, organizing use of, 162
  - associating exception names with, 133
  - date conversions, 291
  - documentation for Oracle error numbers, 142
  - integer values, constraints on, 134
  - lacking, for UTL\_FILE package exceptions, 168
  - Oracle error numbers returned by
    - SQLCODE function, 137
    - values returned by SQLCODE function, 144
  - 20, NNN error codes passed to RAISE\_APPLICATION\_ERROR, 136
- error directives, 923, 1065
  - \$ERROR directive, 1072
  - inquiry directives for use in, 1069
- error management architecture, building, 157–169
  - challenges, 157
  - creating standard templates for error handling, 167
  - deciding on error strategy, 158
  - making most of PL/SQL error management, 169
  - organizing use of application-specific error codes, 162
  - standardizing handling of different types of exceptions, 159
  - using standardized error management programs, 163
- error messages
  - datetime conversions, date format not recognized, 294

- formatting for different locales with  
UTL\_LMS, 1136
- in different languages, 142
- returned by SQLERRM function, 144
- errors
  - anticipating and handling with dynamic  
SQL, 562–564
  - continuing past, in SQL, 128
  - external procedure agent, inability to connect, 1254
  - handling in SQL\*Plus, 36, 39
  - handling with implicit cursors, 496
  - Java, 1235
  - logging with autonomous transactions, 482
  - mutating table errors, 705
  - raising and handling, 9
  - RLS (row-level security), 1016
  - SERVERERROR triggers, 724–728
  - SQL\*Plus, 29
- errpkg.pkg file, 165
- event-driven model for error processing, 130
- events
  - database event triggers, 720–728
  - DDL trigger events, 713
    - working with, 715–719
  - setting to view SQL statements in RLS debugging, 1019
- Exadata, 999
- EXCEPTION datatype
  - limitations of Oracle's implementation, 157
  - working with your own exception objects to overcome its limitations, 165
- exception handling, 129–170
  - building effective error management architecture, 157–169
  - concepts and terminology, 129
  - defining exceptions, 132–140
  - DML and, 468
  - handling exceptions, 143–157
    - built-in error functions, 144
    - combining multiple exceptions in single handler, 149
    - continuing past exceptions, 153, 153
    - exception handlers for individual exceptions, 143
    - propagation of unhandled exceptions, 150–153
    - unhandled exceptions, 149
    - WHEN OTHERS handling code, 155
  - Java, 1232–1235
  - making most of PL/SQL error management, 169
  - raising exceptions, 140–143
- EXCEPTION keyword, 132, 143
- exception section, 131
  - exception handlers in, 143
  - RAISE statement in, 133
- exceptions
  - %BULK\_EXCEPTIONS cursor attribute, 492
  - categories of, 159
    - deliberate exceptions, 160
    - guidelines for dealing with, 162
    - unfortunate and unexpected exceptions, 161
  - continuing past, with SAVE EXCEPTIONS clause in FORALL, 883
  - defined, 129
  - defining, 132–140
    - associating exception names with error codes, 133
    - declaring named exceptions, 132
    - named system exceptions, 136
    - scope, 139
  - exception section in code blocks, 54
    - anonymous block, 56
  - from package initialization failure, 665
  - in dynamic PL/SQL blocks, 560
  - raising from called C program, 1266–1269
  - remote, 773
  - RLS (row-level security), 1016
  - scope, 58
  - some predefined exceptions in PL/SQL, 137
  - STANDARD versus user-defined, 1047
  - types of, 130
  - unhandled
    - function result cache and, 861
    - NOCOPY parameter mode qualifier and, 920
- EXCEPTION\_INIT pragma, 77, 131
  - recommended uses of, 135
  - using to associate exception name with error code, 134
- executable statements in procedures, 594, 594
- EXECUTE authority and access to program units, 816

- EXECUTE command (SQL\*Plus), 28, 41
  - translation of argument into anonymous blocks, 56
- EXECUTE IMMEDIATE statements, 538–542
  - binding PL/SQL-specific Boolean with, 541
  - DML statements, records and, 473
  - examples of use, 540
  - using in dynamic SQL, 548
- execute operations (cursor), 489
- EXECUTE privileges
  - for collections, 412
  - for object types, 1199
  - managing for stored programs, 42
  - on a library, 1255
- execution authority models, 1049–1064
  - BEQUEATH CURRENT\_USER for views, 1061
  - combining rights models, 1056
  - constraining invoker rights privileges, 1063
  - definer rights model, 1049–1053
  - functions returning invoking user, 1060
  - granting roles to PL/SQL program units, 1057–1060
  - invoker rights model, 1054–1056
- execution authority, improvements in, 6
- execution section, code blocks, 54
- execution, tracing (see tracing PL/SQL execution)
- EXISTS method, 359
- EXIT command (SQL\*Plus), 33
- EXIT statements
  - terminating simple loops, 109
  - using properly with loops, 124
- EXIT WHEN statements
  - terminating simple loops, 109
  - using properly with loops, 124
- explicit cursors, 500–515
  - attributes, 510
  - choosing between implicit cursors and, 493
  - closing, 508
  - column aliases in, 507
  - declaring, 501
    - in packages, 502
    - naming your cursor, 502
  - defined, 487
  - example, 500
  - fetching from, 506
    - examples, 506
    - fetching past the last row, 506
  - opening, 504
  - parameters, 512–515
- explicit datatype conversions, 192–197
  - built-in conversion functions, 192
  - CAST function, 194
  - CHARTOROWID function, 193
  - CONVERT function, 196
  - HEXTORAW function, 196
  - RAWIDTOCHAR function, 197
  - RAWTOHEX function, 196
- exponentiation operator (\*\*), 66, 270
- EXTEND method, 360, 380
- extents, 742
- external datatype identifier, 1263
- external LOBs, 421
  - BFILES, 431
- external procedures, 1243–1274
  - architecture of, 1246
  - creating an Oracle library, 1255
  - example, invoking operating system command, 1244
  - maintaining, 1272
    - data dictionary, 1272
    - dropping libraries, 1272
  - nondefault agents, 1269–1272
  - Oracle Net configuration, 1248–1252
  - raising exception from called C program, 1266–1269
  - rules for and warnings about, 1273
  - setting up multithreaded mode, 1252–1254
  - writing call spec for, 1256–1266
    - overall syntax, 1257
    - parameter mapping, 1258–1262
    - PARAMETERS clause, 1262–1266
- external references, invoker rights resolution of, 1054, 1056
- EXTRACT function
  - conversions of intervals to character strings, 306
  - using with datetime values, 310
- extracting substrings from strings, 211
  - using regular expressions, 220

## F

- FALSE values, 74, 178
- FETCH command, 489
- FETCH FIRST clause, 14
- fetch operations (cursor), 489

- FETCH statements, 506
  - fetching from cursor variables, 521, 524
    - handling ROWTYPE\_MISMATCH exception, 525
  - fetching into variables or records, 546
  - initializing collection variables implicitly via, 372
  - using BULK COLLECT with LIMIT clause, 873
- FF date format element, 293
- FGA (see fine-grained auditing)
- fields
  - field-level operations with nested records, 335
  - field-level operations with package-based records, 336
  - field-level operations within records, 334
- fileIO.pkg, 169
- FILESYSTEM\_LIKE\_LOGGING, 437
- FINAL keyword, 1145
- fine-grained access control (FGAC) (see row-level security)
- fine-grained auditing (FGA), 688, 972, 1028
  - checking audit trail, 1033
  - reasons for learning about, 1029
  - simple example, 1030
  - specifying combination of columns as audit condition, 1032
  - summary of, 1034
  - using bind variables, 1035
  - using handler modules, 1036
- FIRST method, 361, 379
- FIRST ROWS clause, limiting rows fetched with BULK COLLECT, 874
- fixed-point numbers, declaring, 244
- FLOAT type, 189
- floating-point literals, 71, 73
- floating-point numbers
  - BINARY\_FLOAT and BINARY\_DOUBLE types, 177, 251–256
  - converting BINARY\_FLOAT and BINARY\_DOUBLE to NUMBER, 257
  - mixing floating-point types in comparisons, 252
  - resulting from NUMBER declaration, 242
  - SIMPLE\_FLOAT and SIMPLE\_DOUBLE types, 177, 256
  - subtypes, 257
  - using BINARY\_FLOAT or BINARY\_DOUBLE for floating-point arithmetic, 922
- FLOOR function, 271
- FM (fill mode) format modifier, 299
  - using when converting from datetime to character string, 302
- FOLLOWS clause
  - guaranteeing DML trigger firing order with, 704
  - use with compound DML triggers, 709
- FOR loops, 106
  - cursor FOR loop, 117–119, 186
  - exiting, 124
  - numeric FOR loop, 114–116
    - examples of, 115
    - nontrivial increments, 116
    - rules for, 114
  - obtaining information about execution, 126
- FOR UPDATE clause, SELECT statement, 515–517
- FORALL statements, 869
  - high-speed DML with, 877–888
    - continuing past exceptions with SAVE EXCEPTIONS, 883
  - cursor attributes for FORALL, 881
  - driving FORALL with nonsequential arrays, 884
  - examples of use, 880
  - FORALL syntax, 878
  - ROLLBACK behavior with FORALL, 882
  - rules for using FORALL, 878
  - one context switch with, 870
- RETURNING clause with BULK COLLECT, 468, 875
- SAVE EXCEPTIONS clause, 128
- foreign key, REF-based foreign key constraint, 1165
- formal parameters, 608
  - explicit association with actual parameters, 613–617
  - NOCOPY parameter mode qualifier and, 918
  - with OUT or IN OUT modes, 613
- FORMAT\_CALL\_STACK function, 145
- FORMAT\_ERROR\_BACKTRACE function, 145–148
- FORMAT\_ERROR\_STACK function, 145
- FORTRAN, PL/SQL embedded in, 52
- forward declarations, 630

- %FOUND attribute, 490, 510
- fractional seconds
  - FF format element, 294, 303
  - precision, 286
  - XFF format mask, 291
- fractional values in date arithmetic, 313
- “Freedom, Order, and PL/SQL Compilation”
  - whitepaper, 841
- FREMOVE program, 169
- FROM clause, calling a table function in, 638
- FTP server, retrieving data from, 966
- function result cache, 852–868
  - enabling, 853
  - example, caching a collection, 858
  - example, deterministic function, 855
  - example, querying data from a table, 856
  - fine-grained dependencies in 11.2 and higher, 863
  - managing, 861
  - recommendations for, 868
  - RELIES\_ON clause, 854
  - useful details of its behavior, 861
  - virtual private database (VPD) and, 865
  - when not to use, 860
  - when to use, 859
- functions, 597–607
  - body, 605
  - calling, 603
  - calling from within SQL, 631
    - defining PL/SQL subprograms in SQL statements, 634
  - read consistency and user-defined functions, 633
  - requirements for, 632
  - restrictions on user-defined functions, 632
  - collection as datatype of return value, 351
  - conversion functions, built-in, 192
  - creating stored function, 37
  - date/time
    - returning current date and time, 282
  - date/time, quick reference, 319
  - datetime conversion, 290
  - defined, 593
  - deterministic, 647
  - dropping stored functions, 43
  - encapsulating single-row queries behind, 493
  - END label, 602

- error functions, built-in, 144–149
- header, 604
- invoking PL/SQL functions within SQL statements, 41
- Java method call spec defined as standalone function, 1228
- LOB conversion functions, 447
- optimizing execution in SQL, 14
- optimizing performance in SQL (12.1 and higher), 922
- parameters (see parameters)
- passing string constants to built-in functions, 204
- RETURN datatype, 601
- RETURN statement, 605
- streaming or transformative, 535
- string, 231–240
- structure of, 598
- total\_sales function (example), 600
- Unicode, 1105–1111
- using NULL statement as placeholder, 102
- viewing in USER\_PROCESURES view, 757
- without parameters, 604
- FX modifier in format masks, 298

## G

- g11n (see globalization and localization)
- g11n schema
  - ISO currency values, 1132
  - USERS table and LOCALE table, 1127
- GCC (GNU C compiler), 1244, 1245
  - g option, 1270
- GDB (GNU debugger), 1270
- GDK (Globalization Development Kit), 1133–1139
  - implementation options, 1137
  - UTL\_I18N package, 1133–1136
  - UTL\_LMS error-handling package, 1136
- general invocation feature (SELF AS supertype), 1153
- GET method, submitting data to web page via, 962
- GET\_LOCAL\_LANGUAGES function
  - (UTL\_I18N), 1134
- global data, 668
  - package data structures acting as globals, 1090
- globalization, 982
  - defined, 1099

- globalization and localization, 1097–1139
  - accent-insensitive sorting of strings, 209
  - character semantics, 1111–1115
  - currency conversion, 1131–1132
  - dates and time, 1126–1131
  - Globalization Development Kit (GDK), 1133–1139
  - globalization strategy, 1098
  - multilingual information retrieval, 1120–1125
  - overview and terminology, 1099
  - string constants represented in national character set, 204
  - string sort order, 1115–1120
  - Unicode primer, 1100–1111
- Globalization Development Kit (see GDK)
- Globalization Support (NLS) parameters, 1104
- glogin.sql script, 35
- glyphs, 1100, 1103
- GNU C compiler (GCC), 1244, 1245, 1270
- GNU debugger (GDB), 1270
- GOTO statements, 8, 100
  - CONTINUE statement versus, 122
  - labels serving as targets of, 78
  - using NULL after a label, 102
- GRANT statements, granting privileges for stored programs, 42
- GREATEST function, 233
- greediness (in regular expressions), 226
- GROUP BY clause, cursor SELECT statement, 493

## H

- handler modules, 1030
  - using in FGA, 1036
- handlers (exception), 131
  - writing for individual exceptions, 143
- handling exceptions, 131, 143
  - (see also exception handling)
- hard-closed cursors, 1079
- HASH partitioning, 902
- hashing, cryptographic, 991
- HAVING clause, cursor SELECT statement, 493
- headers
  - cursor, separating from the body, 503
  - function, 604
    - RESULT\_CACHE clause, 853
  - header section in code blocks, 54
  - named blocks, 57

- procedure, 596
- help
  - asking for, 18
  - in SQL\*Plus, 37
- HEXTORAW function, 196
- hierarchical profiler (see DBMS\_HPROF)
- high-concurrency environments, sharing code
  - at application level, 1081
- homogeneous elements (collections), 343
- host string (pseudo-GUI version of SQL\*Plus), 26
- “hot patching” of PL/SQL-based applications, 822
- HTML forms, 962
- HTML pages, embedding PL/SQL in, 51
- HTTP authentication, 959
- HTTP data (see web-based data, working with)
- HTTP redirects, 963
- HTTPS, retrieving SSL-encrypted web page via, 960
- HTTPURITYPE, 180, 451, 958
  - support for HTTP authentication, 959

## I

- I/O (input/output), 925–968
  - displaying information, 926–929
    - enabling DBMS\_OUTPUT, 926
    - writing lines to the buffer, 926
  - enabling output from Java, 1225
  - extending file I/O capabilities using a Java class, 1236–1240
  - other types available in PL/SQL, 967
    - database pipes, queues, and alerts, 967
    - TCP sockets, 968
  - output from operating system command, 1246
  - reading and writing files, 929–944
    - checking if file is already open, 934
    - closing files, 934
    - copying files, 941
    - deleting files, 942
    - opening files, 932
    - reading from files, 935
    - renaming and moving files, 943
    - retrieving file attributes, 943
    - working with Oracle directories, 931
    - writing to files, 938
  - sending email, 944–955
    - attaching file of arbitrary size, 953

- configuring network security, 946
- including friendly names in addresses, 948
- Oracle prerequisites for, 945
- plain-text message of arbitrary length, 950
- sending message with short attachment, 951
- short, plain-text message, 947
- small file as attachment, 953
- working with web-based data (HTTP), 956–967
  - disabling cookies or making them persistent, 965
  - HTTP authentication, 959
  - retrieving data from FTP server, 966
  - retrieving SSL-encrypted web page via, 960
  - retrieving web page in pieces, 956
  - retrieving web page into a LOB, 958
  - submitting data to web page via GET or POST, 961–965
  - using a proxy server, 966
- i18n (see internationalization)
- identifiers
  - analyzing usage with Oracle 11g's PL/Scope, 759–762
  - defined, 67–70
  - from STANDARD package, 69
  - in inquiry directives, 1066
  - in labels, 77
  - naming, rules for, 67
  - qualified, 62
  - qualified versus unqualified references to, 1047
  - qualifying with module names, 63
  - reserved words, 68
    - avoiding use of, 69
  - scope of, 58
  - visible, 62
  - whitespace and keywords, 70
- identity operator (+), 270
- IDEPTREE view, 766
- IDEs (integrated development environments), 22
  - for Java, 1207
  - popular IDEs for PL/SQL, 45
- IEEE 754 floating-point standard, 922
- IEEE-754 floating-point standard, 251–256
  - floating-point literals, 251
  - Oracle implementation versus, 255
  - reasons to use binary types, 253
- \$IF directive, 1065, 1070
  - inquiry directives for use in, 1069
  - using application package constants in, 1065
- IF statements, 8, 83–92
  - avoiding syntax gotchas, 89
  - comparing two VARCHAR2 values, 228
  - IF-THEN, 84
  - IF-THEN-ELSE, 86
    - implications of NULL Boolean variables, 416
  - IF-THEN-ELSIF, 87
  - nested, 90
  - short-circuit evaluation, 91
  - using NULL statement in, 101
- implicit conversions, 268
  - dangers of, 269
  - DATES into TIMESTAMPS, 316
  - datetime variables and, 289
  - drawbacks of, 191
  - limitations of, 191
  - performed by PL/SQL, summary of, 190
- implicit cursor results, 649
- implicit cursors, 494–500
  - choosing between explicit cursors and, 493
  - defined, 487
  - error handling with, 496
  - examples of use, 495
  - in SELECT statement, characteristics of, 494
  - SQL cursor attributes, 498
- implicit statement results, 13
- implicitly defined directives, 1067
- IN mode
  - bind arguments, 551
  - cursor parameters, 514
  - cursor variable arguments, 531
  - parameter passing by reference, 917
  - parameters, 609
    - default values for, 618
- IN OUT mode
  - bind arguments, 551
  - cursor parameters and, 514
  - cursor variable arguments, 531
  - cursor variable parameters, 525
  - NOCOPY parameter mode qualifier and, 918



- parameter passing by value, 917
- parameters, 609, 612
- INDEX BY clause, 345, 369
  - associative arrays, index values, 375
  - datatypes used with, 366
- index increments (nontrivial), handling in numeric FOR loops, 116
- index value (collections), 343
- indexed by integers (collections), 344
- indexed by strings (collections), 344
- indexes
  - case-insensitivity and, 209
  - creating with NDS EXECUTE IMMEDIATE statement, 540
  - Oracle Text, 1121
  - restricted access to rows when creating indexes, in RLS policies, 1005
- INDICATOR property, PARAMETERS clause, 1263
- INDICES OF clause, FORALL statement, 879, 885
  - example of use, 885
- indirect referencing, 560
- infinite loops
  - avoiding, 836
  - terminating, 111
  - using intentionally, 111
- infinity
  - BINARY\_FLOAT\_INFINITY and BINARY\_DOUBLE\_INFINITY, 252
  - IS INFINITE and IS NOT INFINITE predicates, 252
- information hiding, 655
- information retrieval (IR), multilingual, 1120–1125
- informational compile-time warnings, 778
- INHERIT PRIVILEGES privilege, 1063
- INITCAP function, 209, 233
- initialization of packages, 656, 662–666
  - avoiding side effects, 664
  - caching static session information, 663
  - executing complex initialization logic, 663
  - failure of, 664
  - initialization section, 656, 660, 841
    - example of, 662
- initjvm.sql script, 1207
- inner table, 345
- inquiry directives, 1065, 1066–1070
  - DBMS\_DB\_VERSION package constants, 1067
  - program-specific settings with, 1073
  - referencing unit name and line number, 1069
  - setting compilation environment parameters, 1068
  - using in \$ERROR directive, 1072
  - using PLSQL\_CCFLAGS parameter, 1070
- INSERT statements, 463
  - creating XML documents in a table, 448
  - record-level inserts, 335
  - restrictions on record-based inserts, 472
  - using FORALL with, 878
  - using records in, 470
- INSERTING function, 695
- inserts
  - record-level, 335
  - row-based, replacing with pipelined function-based loads, 889–896
- instantiable types, 1145
- instantiating objects, 1152
- instants, 284
- INSTEAD OF triggers, 688, 728–736, 999, 1185
  - creating, 728
  - for updating object views, 1193
  - INSTEAD OF CREATE, 719
  - INSTEAD OF CREATE TABLE, 710
  - INSTEAD OF DELETE, 733
  - INSTEAD OF INSERT, 730
  - INSTEAD OF UPDATE, 732
  - on nested tables, 734
  - populating the tables, 733
- INSTR function, 210, 233
  - negative string positioning, 213
- INSTR functions, 1107
- instrumentation, 795
- intab (in table) procedure (example), 571–578
- integer types, 178
- integers, 73
  - integer values represented by NUMBER, 246
- integrated development environments (see IDEs)
- internal LOBs, 421
- internationalization (i18n), 982
  - defined, 1099
- Internet datatypes, 180
- Internet resources for PL/SQL developers, 16



- INTERSECT operator, 402
  - MULTISET INTERSECT, 409
- INTERVAL datatypes, 284–288
  - declaring INTERVAL variables, 286
  - INTERVAL DAY TO SECOND type, 285
    - converting numbers to, 305
    - converting strings to, 305
  - INTERVAL type, 178
    - intervals as literals, 71
  - INTERVAL YEAR TO MONTH type, 285
    - converting numbers to, 304
    - converting strings to, 305
  - reasons for two type, 285
  - using unconstrained INTERVAL types, 318
  - when to use, 287
    - designating periods of time, 288
    - finding difference between two datetime values, 287
- interval literals, 307
- intervals
  - adding and subtracting, 317
  - computing between datetimes, 313
  - conversions, 304–306
    - formatting intervals for display, 306
    - from numbers to intervals, 304
    - from strings to intervals, 305
    - interval element names, 304
  - date arithmetic with, 311
  - defined, 285
  - multiplying and dividing, 317
- INTO clause, 514
  - BULK COLLECT INTO, 495
  - EXECUTE IMMEDIATE statement, 539
  - FETCH statement, 524, 528
  - SELECT statement, 494
- introspection, 453
- introspection functions, determining type of data held by AnyData or AnyDataSet variable, 454
- invalid program units, 765
  - avoiding invalidations, 777
  - recompiling, 773–777
- INVALID\_OPERATION exception
  - in UTL\_FILE package, 169
- invisible columns, 14, 339
- invoker rights model, 462, 1054–1056
  - AUTHID CURENT\_USER clause, 6
  - combining with definer rights model, 1056
  - constraining invoker rights privileges, 1063
  - for dynamic SQL, 1055
  - rules and restrictions, 1055
  - syntax, 1054
  - using for shared dynamic SQL programs, 561
- invoker rights units, 13
- invoking user, functions for, 1060
- irrational debugging, 810
- IS A SET operator, 411
- IS NAN and IS NOT NAN predicates, floating-point literals, 252
- IS NOT NULL operator, 85
- IS NULL operator, 85, 270
- IS OF operator, 1161
- IS [NOT] A SET operator, 411
- ISO (International Standards Organization)
  - currency indicator, 261, 1283
  - dates, 1288
  - SQL standard, CAST function, 268
  - SQL, SQL/DS and DB2 datatypes, 256
- isolation levels (transactions), 476
  - controlling transaction visibility, 480
- %ISOPEN attribute, 491
  - checking i cursor is open, 505
  - values before and after cursor operations, 510
  - values returned by, 510
- ISO\_CURRENCY\_FUNC function, 1132
- iSQL\*Plus, 1103
- iterative control structures (see loops)

## J

- Java, 1205–1242
  - calling PL/SQL from, using JDBC, 47
  - differences between PL/SQL and, 1214
  - dropping Java files from database using dropjava, 1221
  - example, deleting a file from within PL/SQL, 1212–1218
    - building a custom Java class, 1213
    - building PL/SQL wrapper, 1217
    - compiling and loading into Oracle, 1215
    - deleting a file, 1217
    - finding the Java functionality, 1212
  - getting Oracle ready to use, 1207–1211
    - building and compiling Java code, 1208
    - installing Java, 1207
    - setting permissions, 1209

- loading Java files into the database with
  - loadjava, 1218
- managing in the database, 1221–1223
  - examining loaded Java elements, 1221
  - Java namespace in Oracle, 1221
- Oracle databases and, 1205
  - Oracle components and commands for Java, 1206
- publishing and using in PL/SQL, 1228–1242
  - call specs, 1228
  - calling a Java method in SQL, 1232
  - exception handling with Java, 1232–1235
  - extending file I/O capabilities, 1236–1240
  - mapping datatypes, 1230
  - other examples, 1240
- using DBMS\_JAVA package, 1223–1228
  - converting Java long names, 1223
  - enabling output from Java, 1225
  - exporting source, resources, and classes, 1226
  - getting and setting compiler options, 1224
- JAVASYSPRIV role, 1209
- JAVAUSERPRIV role, 1209
- JDBC (Java Database Connectivity), 7, 47
- JDeveloper, 1207
- JDK (Java Development Kit), 1207
- JRE (Java Runtime Engine), 1208
- JSPs (Java stored procedures), 1205
  - accessing from Oracle database, 1206
- Julian date, 290
- just-in-time caching of table data, 850
- JVM (Java virtual machine), 1205
  - Aurora JVM, 1206
  - Security Manager, 1209

## K

- keys, encryption, 973
  - length of, 974
- MAC (message authentication code), 993
- managing, 985–991
  - combined approach, 988
  - single key for each row, 987
  - single key for the database, 985
- performing key generation, 985
- public and private, 975
- kill command, 111

## L

- l10n (see localization; globalization and localization)
- label delimiters (<< and >>), 66
- labels, 77
  - as targets for GOTO statements, 100
  - loop, 119, 121
- LANGUAGE JAVA clause, 1228
- large objects (see LOB types)
- Large Objects Developer's Guide, 421
- LAST method, 361, 379
- LEAST function, 234
- LENGTH function
  - passing string constants to, 204
  - returning number of characters in a string, 234
- LENGTH functions, 1108
  - LENGTH and LENGTHC, 1109
  - LENGTHB, 1109, 1112
- LENGTH property, PARAMETERS clause, 1264
- LEXERS, 1121
- lexical units, 66
- libraries
  - creating an Oracle library, 1255
  - dropping, 1272
  - shared, 1244
- LIMIT clause for BULK COLLECT, 873, 1088
- LIMIT method, 362, 380
- line numbers (compilation unit), 1069
- linefeeds, 205
- listener.ora file, 1249
- listeners for external procedures
  - configuring, 1248–1252
  - setting up multithreaded mode, 1252
- literals, 70–74
  - avoiding hardcoding of, 681
  - Boolean, 74
  - embedding single quotes inside literal strings, 73
  - NULLs, 72
  - numeric, 73
- loadjava command, 1206
  - common options, 1219
  - loading JDelete class (example), 1216
  - using, 1218
- LOB types, 420–447
  - BFILE, 431
  - conversion functions, 447
  - empty versus NULL LOBs, 425

- encrypting, 982
  - getting length of, 431
  - native LOB operations, 442
    - SQL semantics, 443–447
  - Oracle documentation for, 425
  - reading from a LOB, 430
  - retrieving web page into a LOB, 958
  - SecureFiles, 983
  - SecureFiles versus BasicFiles, 436
  - temporary LOBs, 439
    - checking if LOB is temporary, 441
    - creating, 440
    - freeing, 441
    - managing, 442
  - understanding LOB locators, 423
  - working with, 422
  - writing into a LOB, 427
  - local modules, 619–624
    - benefits of, 620
      - improving readability, 621
      - reducing code volume, 620
    - forward declaration of, 630
    - scope of, 623
    - sprucing up code with nested subprograms, 623
  - locale buttons (GDK), 1138
  - LOCALE table in g11n schema, 1127
  - localization, 1097
    - (see also globalization and localization)
    - defined, 1099
  - LOCALTIMESTAMP function, 282
  - LOCK TABLE statements, 476
  - locks
    - COMMIT statement and, 474
    - releasing with COMMIT, 517
  - LOG ERRORS clause, 128
  - Log4PLSQL open source tracing framework, 796
  - logging
    - building autonomous logging mechanism, 482
    - levels for LOBs, 437
    - NOLOGGING option for LOBs, 425
  - login.sql script, 35, 37
  - LOGOFF triggers, 723
  - LOGON triggers, 723, 1020
  - LONG and LONG RAW types, 421
    - conversion to CLOB and BLOB, 447
  - LONG RAW type, 179
  - loop body, 107
  - loop boundary, 107
  - loop invariants, 840
  - Loop Killer package, 836
  - loops, 8, 105–128
    - avoiding infinite loops, 836
    - CONTINUE statements, 121–123
    - cursor FOR loop, 117–119
    - examples of, 106
      - FOR loop, 106
      - simple loop, 106
      - WHILE loop, 107
    - labels, 119
    - numeric FOR loop, 114–116
    - simple loop, 108–112
    - structure of, 107
    - tips on writing, 123–128
      - exiting loops, 124
      - obtaining information about FOR loop execution, 126
      - SQL statements as loops, 126
      - understandable names for loop indexes, 124
    - types of, 105
      - WHILE loop, 112
  - Lovelace, Ada, 1046
  - LOWER function, 208, 234
  - LPAD function, 213, 234
  - LTRIM function, 215, 234, 267
- ## M
- MACs (message authentication codes), 993, 995
  - magic values, 681
  - main transaction, 477
  - managing code, 749–824
    - code management issues, 22
    - compile-time warnings, 777–788
    - debugging PL/SQL programs, 808–815
    - dependencies and recompiling code, 762
      - analyzing dependencies with data dictionary views, 763–767
      - fine-grained dependency, 767
      - limitations of Oracle's remote invocation model, 772
      - recompiling invalid program units, 773–777
      - remote dependencies, 769
    - edition-based redefinition, 822

- in the database, 750–762
  - analyzing and modifying triggers through views, 757
  - analyzing argument information, 758
  - analyzing identifier usage, 759–762
  - data dictionary views, 751
  - displaying and searching source code, 754
  - displaying information about stored objects, 753
  - obtaining properties of stored code, 756
  - pinning requirements, determining from program size, 755
  - protecting stored code, 818
  - testing PL/SQL programs, 788–795
  - tracing PL/SQL execution, 795–808
    - using DBMS\_TRACE package, 804–808
    - using DBMS\_UTILITY.FOR-MAT\_CALL\_STACK, 796
    - using opp\_trace utility, 803
    - using UTL\_CALL\_STACK, 798–801
  - using whitelisting to control access to program units, 816
- MAP method, 1151
  - additional recommendations for, 1184
  - comparing objects, 1181
- MAXLEN property, PARAMETERS clause, 1265
- MAX\_SQL\_STRING\_SIZE initialization parameter, 201
- member method syntax, collection methods, 356
- member methods, 1147
- MEMBER operator, 409
- memory
  - analyzing memory usage, 827
  - database instance memory and PL/SQL, 1076–1093
    - running out of memory, 1091–1093
    - SGA, PGA, and UGA, 1076
    - tips on reducing memory usage, 1079–1090
  - database instance memory and PLSQL cursors and, 1077
  - for serialized packages, 676
  - performance optimization and, 923
  - PGA (program global area)
    - BULK COLLECT and, 873
    - pipelined table functions and, 888
  - PGA (program global area), data caching and, 844
  - reducing consumption using PLS\_INTE-GER, 922
  - SGA (system global area), 844
    - function result cache, 861
    - use by packages, 674
  - merge operations, tuning with pipelined functions, 896–898
  - row-based PL/SQL merge processing, 896
  - set-based MERGE with pipelined functions, 897
  - MERGE statements, 466
    - using FORALL with, 878
  - message authentication codes (MACs), 993, 995
  - Message Digest (MD4 and MD5) algorithms
    - use in hashing, 992
    - use in message authentication codes, 994
  - message IDs (AQ), 185
  - metacharacters in regular expressions, 221, 1275–1278
  - method chaining, 1155
  - methods, 1144, 1147–1152
    - collection, 356–365
    - comparison, 1151
    - constructor, 1147
    - Java
      - call specs for, 1228
      - calling in SQL, 1232
    - member, 1147
    - object type method, call spec defined as, 1229
    - static, 1151
    - supertype, invoking from subtype, 1152
  - Microsoft
    - NTLM-based authentication, 967
    - Open Database Connectivity (ODBC), 45
  - MIME (Multipurpose Internet Mail Extensions), 951
    - MIME content types, IANA, 952
  - MinGW (Minimal GNU for Windows), 1245
  - MINUS operator, 402
  - modular code, 592
    - calling PL/SQL functions from SQL, 631–637
    - deterministic functions, 647
    - forward declarations, 630
    - functions, 597–607
    - implicit cursor results, 649

- local or nested modules, 619–624
- parameters, 607–619
- procedures, 594–597
- subprogram overloading, 624–630
- table functions, 637–647
- modularization, 592
- modules
  - qualifying identifier names with module names, 63
  - scope, 58
- monetary values
  - binary types, not recommended to represent, 252
- monolingual sort, 1117
- MONTHS\_BETWEEN function, 314
- multibyte character sets
  - CHAR length qualifier and, 200
  - translating single-byte characters to, 238
- multibyte characters, 1101
  - defined, 1100
  - environment's support for, 1103
  - support for, 1111–1115
- multidim package, 392
  - exploring the multidim API, 393
  - multidim2, 395
- multidimensional arrays
  - collections of collections, 344
  - emulation of, unnamed multilevel collections, 391
- multilevel collections, 389–398
  - extending string\_tracker package with, 396
  - number of levels of nesting, 398
  - unnamed, emulation of multidimensional arrays, 391
- multiline comments, syntax, 76
- multilingual information retrieval, 1120–1125
- multilingual sort, 1119
- multiplication operator (\*), 270
- Multipurpose Internet Mail Extensions (MIME), 951
  - IANA content types, 952
- MULTISET EXCEPT operator, 347
- MULTISET operators, 406
  - MULTISET EXCEPT, 409
  - MULTISET INTERSECT, 409
  - MULTISET UNION, 409
- MULTISET pseudofunction, 400
- multisets, 345

- multithreading, 1248
  - setting up multithreaded mode, 1252–1254
- MULTI\_LEXER, 1121
- mutating table errors, 705

## N

- n-prefixed strings, 204
- name resolution in SQL, 64
- named blocks, 53, 57
- named exceptions, 132
  - associating exception names with error codes, 133
  - declaring, 132
  - system and programmer-defined, scope of, 139
  - system exceptions, 136
- named notation, associating actual and formal parameters, 613, 614, 618
  - benefits of, 615
- names
  - converting Java long names to maximum SQL identifier length, 1223
  - explicit cursor, 502
  - of exceptions, 133
- namespace (Java), in Oracle, 1221
- naming conventions
  - for data structures, 174
  - violations of, analyzing code for, 760
- NaN (Not a Number), 74
  - BINARY\_FLOAT\_NAN and BINARY\_DOUBLE\_NAN, 251
  - IS NAN and IS NOT NAN predicates, floating-point literals, 252
  - support by BINARY\_FLOAT and BINARY\_DOUBLE, 177
- narrowing or downcasting, 1160
  - REFs, 1170
- National Language Support (see NLS; NLS settings)
- National Language Support CLOB type (see NCLOB type)
- national languages, 199
  - (see also globalization and localization)
  - string constants represented in national character set, 204
- native compilation, 1094
- native dynamic SQL (see NDS)
- NATURAL subtype, 257
- navigation, database, 22

- NCHAR type, 176, 200, 1101, 1102
  - TO\_NCHAR function, 238
- NCHR function, 235
- NCLOB type, 176, 421, 422, 1102
- NDS (native dynamic SQL), 538
  - EXECUTE IMMEDIATE statement, 538–542
  - implementing dynamic SQL methods with NDS statements, 548
  - OPEN FOR statement, 543–548
  - recommendations for, 561–569
    - anticipating and handling dynamic errors, 562–564
    - minimizing dangers of code injection, 566
    - using binding rather than concatenation, 564
    - using invoker rights for shared programs, 561
  - rules for working with dynamic PL/SQL blocks and NDS, 558
  - working with objects and collections, 555–557
- negation operator (-), 270
- negative scale values (NUMBER), 245
- negative string positioning, 213
- nested blocks, 57
- nested CASE statements, 97
- nested collections, 389
  - (see also multilevel collections)
- nested IF statements, 90
- nested loops, using labels with, 120
- nested modules (see local modules)
- nested programs, 64
- nested records, 328
- nested tables
  - collection type, 343, 345
    - changing nested table characteristics, 369
    - comparison to associative arrays and VARRAYs, 354
    - declaring, 368
    - example of, 347
  - declaring and initializing collection variables of type, 370
- INSTEAD OF triggers on, 734
- multiset operations on, 406–412
  - checking membership of element in nested table, 409
  - handling duplicates, 411
  - high-level set operations, 409
    - testing equality and membership, 408
  - VARRAYs versus, as column datatypes, 353
- NEW and OLD pseudo-records, 338
  - referencing fields in WHEN clause of DML trigger, 693
  - using to fine-tune trigger execution, 701
  - working with, in DML triggers, 694
- newline character, 206
- NEXT method, 362
- next value from a sequence, getting, 464
- NLS (National Language Support)
  - character set in Oracle databases, 1101
  - defined, 1100
- NLS settings, 1104
  - for datetimes, 309
  - NLS\_CALENDAR, 1130
  - NLS\_CHARACTERSET, 1101
  - NLS\_COMP, 208
  - NLS\_CURRENCY, 1131
  - NLS\_DATE\_FORMAT, 289, 290, 300, 1128
  - NLS\_DATE\_LANGUAGE, 291, 1129
  - NLS\_ISO\_CURRENCY symbol, 1132
  - NLS\_LENGTH\_SEMANTICS, 1114
  - NLS\_NCHAR\_CHARACTERSET, 1101
  - NLS\_SORT, 208, 1117
    - monolingual values, 1118
    - multilingual values, 1119
  - NLS\_TERRITORY, 291
  - passing to TO\_CHAR function, 267
  - passing to TO\_NUMBER function, 260
- NLSORT function, 236, 1118
  - multilingual sorts, 1119
- NLS\_COMP function, 208
- NLS\_INITCAP function, 235
- NLS\_LENGTH\_SEMANTICS initialization parameter, 201
- NLS\_LOWER function, 235
- NLS\_SESSION\_PARAMETERS view, 1104
- NLS\_SORT function, 208
- NLS\_UPPER function, 235
- NOCOPY parameter mode qualifier, 611, 617, 917
  - downside of, 920
  - performance benefits of, 919
  - restrictions on, 918
- non-blank-padding comparisons, 230
- nongreedy quantifiers for regular expressions, 226

- nonprintable characters, 205
- nonsequential population of a collection, 377
  - advantages of, 377
- normal range of values, BINARY FLOAT, 252
- normalization of local variables, drawbacks of
  - anchored declarations, 187
- Not a Number (see NaN)
- not equal operators, 66, 270
- NOT FINAL keyword, 1145
- NOT INSTANTIABLE keyword, 1145
- NOT NULL clause, 183
- NOT NULL datatypes, anchoring to, 188
- NOT operator, 270
- %NOTFOUND attribute, 118, 491, 510
- NOWAIT keyword, 477
  - appending to FOR UPDATE clause, 516
- NO\_DATA\_FOUND exception
  - as unfortunate exception, 161
  - handling, 137
  - handling with implicit cursors, 496
  - raised by UTL\_FILE.GET\_LINE, 936
  - STANDARD versus user-defined, 1047
- NTLM-based authentication, 967
- NULL statement, 101
  - using after GOTO statement label, 102
  - using to avoid ambiguity in IF and CASE statements, 101
- NULLs, 72, 178
  - atomically null objects, 1151
  - Boolean expressions returning, 84
  - Boolean variables as, 416
  - detecting Oracle NULL in C program, 1259
  - empty strings treated as, 227
  - equality check for nested tables, 408
  - in record comparisons, 338
  - NULL LOBs, 425
  - null versus empty collections, 1190
  - operators and functions to detect and deal with, 85
  - passing as bind arguments, 554
  - returned by CASE expressions, 100
- NULLS
  - casting to different types, 72
- number format models, 1281
  - using TO\_CHAR function with, 262
  - using TO\_NUMBER function with, 259
  - V format element, 263
- NUMBER type, 177, 242
  - comparison to other floating-point types, 251
  - converting strings and IEEE-754 floating point types to, 258–261
  - declaring
    - constraining precision and scale, 243
    - scale as optional and defaulting to zero, 246
    - simplest declaration, 242
  - effect of negative scale, 245
  - interval between two DATE values, 317
  - performance, BINARY\_DOUBLE versus, 253
  - PLS\_INTEGER type and, 248
  - rounding of values, 244
  - used for fixed-point values, range of, 246
  - using to represent monetary values, 252
- numbers, 241–276
  - conversions, 257–270
    - implicit conversions, 268
    - using CAST function, 267
    - using TO\_CHAR function, 261–267
    - using TO\_NUMBER function, 258–261
  - converting to intervals, 304
- numeric datatypes, 177, 241–257
  - BINARY\_FLOAT and BINARY\_DOUBLE types, 251–256
  - BINARY\_INTEGER, 249
  - core types, 241
  - NUMBER, 242
  - overloading with, 629
  - PLS\_INTEGER, 247
  - SIMPLE\_FLOAT and SIMPLE\_DOUBLE, 256
  - SIMPLE\_INTEGER type, 249
  - subtypes, 256
- numeric FOR loops, 106, 114–116
  - examples of, 115
  - nontrivial increments, 116
  - rules for, 114
- numeric functions, 271–276
  - quick reference, 272–276
  - rounding and truncation, 271
  - trigonometric, 272
- numeric literals, 73
- numeric operators, 270
- numeric overflow or underflow exception, 243
- NUMTOYMINTERVAL function, 304



NVARCHAR2 type, 176, 200, 1101, 1102  
NVL function, 87  
NVL2 function, 85, 206

## O

obfuscation of code, 818  
OBJECT IDENTIFIER IS PRIMARY KEY  
    clause, 1156  
object identifiers (see OIDs)  
object tables  
    building, 1154  
    differences between object views and, 1196  
object types, 447–456, 593, 1144–1184  
    Any types, 453–456  
    comparing objects, 1179–1184  
        attribute-level comparisons, 1181  
        MAP method, 1181  
        ORDER method, 1182  
        recommendations, additional, 1184  
        ways to compare objects, 1180  
    creating base type, 1144  
    creating self-sufficient type, 1176–1179  
    creating subtype, 1146  
    data dictionary entries for, 1197  
    evolution and creation, 1162  
    example, modeling trivial library catalog,  
        1144  
    generic data, ANY types, 1171–1175  
    HTTPURITYPE, 958  
    invoking supertype methods (Oracle 11g  
        and later), 1152  
    memory consumption by, 827  
    methods, 1147–1152  
    object references (REFs), 1164–1171  
    packages and, 685  
    privileges for, 1199  
    storing, retrieving, and using persistent ob-  
        jects, 1154–1162  
        object identity, 1156  
        TREAT function, 1160  
        VALUE function, 1157  
    substitutable object types, using with pipe-  
        lined functions, 910  
    URI types, 451  
    XMLType, 448–451  
object views, 1184–1197  
    data dictionary entries for object types, 1197  
    DEBUG privilege, 1200  
    differences between object tables and, 1196

object subview, 1191  
    privilege to create a subview, 1200  
    sample relational system, 1186  
    with a collection attribute, 1188–1191  
    with inverse relationship, 1192  
object-oriented programming (OOP), 1141–  
    1203  
    maintaining object types and object views,  
        1197–1200  
    object types example, 1144–1184  
    object views, 1184–1197  
    Oracle's object features, 1142  
    when best to use Oracle's object features,  
        1202  
objects  
    collections as attributes of, 354  
    collections of, 389  
    displaying information about stored objects,  
        753  
    working with, using NDS, 555–557  
OCI (Oracle Call Interface)  
    and full transaction support by external pro-  
        cedures, 1248  
    anonymous blocks, use of, 56  
    C language interface, 46  
    complex object retrieval, 1169  
    OCI8 PHP functions, 50  
    routines for writing external procedures,  
        1268  
ODBC (Open Database Connectivity), 7, 45  
OIDs (object identifiers), 1155  
    in object views versus object tables, 1196  
    REFs to nonunique OIDs, 1197  
    virtual OID based on primary key, 1189  
OBJECT\_VALUE pseudocolumn, 1180  
OLD pseudo-records (see NEW and OLD  
    pseudo-records)  
one-dimensional or single-dimensional (collec-  
    tions), 343  
online transaction processing (OLTP), suspend-  
    ed statement pauses and, 743  
ONLY keyword, 1161  
OPEN FOR statements, 524, 532, 543–548  
    FETCH into variables or records, 546  
    program to display table column for rows  
        indicated by WHERE clause, 544  
    USING clause, 547  
OPEN statements  
    opening cursor variables, 523



- opening explicit cursors, 504
  - opening a cursor, 489
  - operating systems
    - default editors assumed by Oracle, 34
    - external procedure invoking operating system command, 1244
    - launching SQL\*Plus from command prompt, current directory, 30
    - setting up Oracle wallet on, 960
    - starting SQL\*Plus, 24
    - Unicode support, 1103
  - opp\_trace package, 796
  - opp\_trace utility
    - tracing with, 803
  - optimization level settings, 838
  - OPTIMIZER\_MODE parameter, 488
  - optimizing PL/SQL performance (see performance optimization)
  - OR operator, 270
    - combining multiple exceptions in single handler, 149
    - short-circuit evaluation in IF statements, 92
  - Oracle Advanced Queuing (AQ), 184, 668
  - Oracle Application Express, 1103
  - Oracle Call Interface (see OCI)
  - Oracle Database 12c, xxvii
    - highlights, 11
    - new PL/SQL features, 12
  - Oracle Database Globalization Support Guide, 209
  - Oracle databases
    - error messages, more detail on, 39
    - execution of PL/SQL code, 1040–1045
    - getting ready to use Java in, 1207–1211
      - building and compiling Java code, 1208
      - installing Java, 1207
      - setting permissions, 1209
    - and Java, 1205
      - Oracle components and commands for Java, 1206
    - location of shared library files, 1245
    - native compilation and database release, 1094
    - navigating, 22
    - object features, 1142
    - security, 971
    - versions, xxvii
      - and corresponding PL/SQL versions, 11
    - with PHP, 50
  - Oracle Net communications layer, 1248
    - connect identifier, 25
    - security characteristics of listener configuration, 1251
    - specifying listener configuration, 1248
  - Oracle Text, 429, 1121–1125
    - and SQL semantics, 446
    - formatting search strings for, 1123
    - indexes, 1121
  - Oracle Wallet Manager (owm), 960
  - orakill.exe command, 111
  - ORA\_INVOKING\_USER function, 1061, 1062
  - ORA\_INVOKING\_USERID function, 1061
  - ORA\_INVOKING\_USER\_ID function, 1062
  - ORA\_IS\_ALTER\_COLUMN function, 716
  - ORA\_SPACE\_ERROR\_INFO function, 739
  - ORDER method, 1151
    - additional recommendations for, 1184
    - comparing objects, 1182
  - order of execution, 841
  - ORDER streaming, 902
  - OUT mode
    - bind arguments, 551
    - cursor parameters and, 514
    - cursor variable arguments, 531
    - NOCOPY parameter mode qualifier and, 918
    - parameter passing by value, 917
    - parameters, 609, 611
  - outer table, 344
  - Output Feedback (CHAIN\_OFB), 977
  - overloading
    - subprogram, 624–630
      - benefits of, 625
      - restrictions on, 628
      - with numeric types, 629
- ## P
- package-level data, 658
  - packages, 593, 651–686
    - benefits of using, 651
    - caching based on, 845–850, 868
      - caching table contents, 848
      - example of, 846
      - just-in-time caching of table data, 850
      - when to use, 845
    - compound triggers and, 707
    - concepts related to, 655
    - declaring explicit cursors in, 502

- default packages of PL/SQL, 1045–1048
- demonstration of power of, 652
- diagram of public and private elements, 657
- dropping stored package or package body, 43
- granting roles to, 13
- I/O in PL/SQL, 925
- and object types, 685
- package-based collections, example of, 350
- preservation of state of package data structures, 1089
- qualifying identifier names with package names, 63
- rules for building, 658–666
  - initialization, 662–666
  - package body, 660
  - package specification, 658
- rules for calling elements, 666
- synchronizing code with packaged constants, 1072
- using application package constants in \$IF directive, 1065
- using to improve memory use and performance, 1085
- when to use, 677–685
  - avoiding hardcoding of literals, 681
  - caching static session data, 684
  - encapsulating data access, 677–680
  - grouping together logically related functionality, 683
  - improving usability of built-in features, 683
- working with package data, 668–676
  - cursors, 669–674
  - global public data, 669
  - global within single Oracle session, 668
  - serializable packages, 674–676
- working with XML data, 451

padding

- encryption, 977, 980, 984
  - DBMS\_CRYPTO padding constants, 978
- Public Key Cryptography System #5 (PKCS#5), 979
- string, 213
  - LPAD function, 234
  - RPAD function, 236
  - string comparisons with mixed CHAR and VARCHAR2 values, 230

parallel execution of table functions

- asynchronous data unloading with parallel pipelined functions, 898–902
- enabling a function for parallel execution, 646
- enabling parallel pipelined function execution, 895
- exploiting parallel pipelined functions for ultimate performance, 894
- performance and partitioning and streaming clauses in pipelined functions, 902

PARALLEL\_UNLOAD\_BUFFERED function, 901

parameters, 607–619

- actual and formal, 608
- collections as program parameters, 350
- cursor, 512–515
- default values, 618
- defining, 608
- explicit association of actual and formal parameters in PL/SQL, 613–617
- functions called from SQL, 632
- Globalization Support (NLS) parameters, 1104
- modes, 609
  - IN mode, 610
  - IN OUT mode, 612
  - OUT mode, 611
- NOCOPY parameter mode qualifier, 617, 917–921
- regular expression functions, 1279

PARAMETERS clause, 1262–1266

- CHARSETID and CHARSETFORM properties, 1265
- INDICATOR property, 1263
- LENGTH property, 1264
- MAXLEN property, 1265

parent block, 58

parsing operations (cursor), 488

partitioning clauses in parallel pipelined functions, 902

partitioning options, allocating data to parallel processes, 902

PATH environment variable, 1215

performance

- compile-time warnings about, 778
- impact of using SQL semantics, 446
- improving with fine-grained auditing (FGA), 1030

- LOBs, improvement with SecureFiles, 437
- string-indexed collections, 384
- performance optimization, 825–923
  - analyzing memory usage, 827
  - avoiding infinite loops, 836
  - big picture on performance, 923
  - bulk processing for repeated SQL statement execution, 869–888
    - high-speed DML with FORALL, 877–888
    - high-speed querying with BULK COLLECT, 870–877
  - calculating elapsed time, 833
  - choosing the fastest program, 835
  - data caching techniques, 844–868
    - deterministic function caching, 850–852
    - function result cache, 852–868
    - package-based caching, 845–850
  - identifying bottlenecks in PL/SQL code, 827–832
    - using DBMS\_HPROF (hierarchical profiler), 830
    - using DBMS\_PROFILER, 828
  - optimizing compiler, 838–843
    - how the optimizer works, 840
    - runtime optimization of fetch loops, 843
  - optimizing context in which PL/SQL code runs, 825
  - specialized optimization techniques, 917–922
    - optimizing function performance in SQL, 922
    - using NOCOPY parameter mode hint, 917–921
    - using right datatype, 921
  - steps in optimizing PL/SQL code, 826
  - using pipelined table functions, 888–916
    - asynchronous data unloading with parallel pipelined functions, 898–902
    - partitioning and streaming clauses in parallel pipelined functions, 902
    - pipelined functions and cost-based optimizer, 903–909
    - replacing row-based inserts, 889–896
    - summary of use for performance tuning, 916
    - tuning complex data loads, 909–916
    - tuning merge operations, 896–898
  - warnings related to performance, 837
- periods, 285, 285
  - 24-hour periods between two DATE values, 314
  - designating periods of time, 288
- Perl, calling PL/SQL from, 48
- permissions for Java development and execution, 1209, 1217
- persistence, 657
- PGA (program global area), 1041
  - data caching in, 844
  - defined, 1077
  - finding how much current session uses, 1087
  - performance improvement versus memory usage, 826
- PHP, calling PL/SQL from, 50
- pipelined table functions, 638, 888–916
  - asynchronous data uploading with parallel pipelined functions, 898–902
  - parallel-enabled pipelined function unloader, 899
  - typical data-extract program, 898
- cost-based optimizer and, 903–909
  - cardinality heuristics for pipelined table functions, 904
  - extensible optimizer and pipelined function cardinality, 906
  - providing cardinality statistics to optimizer, 906
  - using optimizer dynamic sampling, 905
- creating, 644
- parallel, performance implications of partitioning and streaming clauses, 902
- replacing row-based inserts, 889–896
  - enabling parallel pipelined function execution, 895
  - exploiting parallel pipelined functions for ultimate performance, 894
  - implementation example, 890
  - loading from pipelined function, 891
  - tuning pipelined functions with array fetches, 893
- summary of use for performance tuning, 916
- tuning complex data loads, 909–916
  - loading multiple tables from multitype pipelined function, 914
  - multitype method, alternative, 915
  - multitype pipelined function, 911
  - one source, two targets, 909

- pipinging multiple record types from pipelined functions, [910](#)
  - querying multitype pipelined function, [912](#)
  - using object-relational features, [910](#), [910](#)
- tuning merge operations with, [896–898](#)
- PKCS#5 (Public Key Cryptography Standard #5), [977](#), [979](#)
- \*.pkg files, [683](#)
- PL/Scope, analyzing identifier usage, [759–762](#)
- PL/SQL
  - advanced elements of procedural languages in, [xxvi](#)
  - advice on working more effectively in, [17](#)
  - block structure, [53–65](#)
  - calling from other languages, [45–52](#)
  - character set, [65](#)
  - comments, [75](#)
  - control and conditional logic, [8](#)
  - defining characteristics, [3](#)
  - ease of learning, [7](#)
  - error-handling mechanisms, [9](#)
  - floating-point literals supported, [251](#)
  - identifiers, [67–70](#)
  - integration with SQL, [xxvi](#), [7](#)
  - labels, [77](#)
  - literals, [70–74](#)
  - more places and ways to use it, [51](#)
  - new features in Oracle Database 12c, [12](#)
  - optimizing function execution in SQL, [14](#)
  - origins and brief history, [4](#)
  - performing essential tasks, [37–44](#)
    - creating a stored program, [37–41](#)
    - creating stored program, [37](#)
    - dropping a stored program, [43](#)
    - executing a stored program, [41](#)
    - grants and synonyms for stored programs, [42](#)
    - hiding source code of stored programs, [44](#)
    - showing stored programs, [42](#)
  - PRAGMA keyword, [76](#)
  - resources for developers, [14](#)
  - semicolon delimiter, [75](#)
  - versions, [xxxii](#), [11](#)
- PL/SQL Developer IDE, [45](#)
- PL/SQL runtime engine, [1041](#)
- PL/SQL Server Pages (PSP), [51](#)
- PL/SQL virtual machine (PVM), [1041](#)
  - (see also PL/SQL runtime engine)
- planning before writing code, [17](#)
- platforms, [xxxii](#)
- PLSCOPE\_SETTINGS compilation parameter, [760](#)
- plshprof command-line utility, [831](#)
- PLSQL\_CCFLAGS parameter, [1066](#), [1070](#), [1073](#)
- PLSQL\_OPTIMIZE\_LEVEL initialization parameter, [838](#)
- PLSQL\_PROFILER tables, [828](#)
- PLS\_INTEGER type, [178](#), [242](#), [247](#)
  - NATURAL and POSITIVE subtypes, [257](#)
  - use for intensive integer computations, [922](#)
- pluggable databases, [719](#)
- pointers, [1164](#)
- policies (FGA), [1030](#)
- policies (RLS), [1001](#)
  - context-sensitive, [1010](#), [1025](#)
  - creating, [1004](#)
  - direct path operations and, [1017](#)
  - dropping, [1006](#)
  - shared context-sensitive policies, [1012](#)
  - shared static policies, [1010](#)
  - static versus dynamic, [1007](#)
  - upgrade strategy for Oracle Database 10g/11g policy types, [1012](#)
- policy function, [1001](#)
  - defining so restrictive predicate is not applied to schema owner, [1008](#)
  - errors caused by, [1016](#)
  - using application contexts, [1023](#)
  - writing, [1003](#)
- polymorphism, [625](#)
- portability of PL/SQL applications, [5](#)
- Portable Operating System Interface (POSIX)
  - regular expression standard, [216](#)
- positional notation, associating actual and formal parameters, [613](#)
- positive scale values (NUMBER), [245](#)
- POSITIVE type, [188](#), [257](#)
- POST method, submitting data to web page via, [962](#)
- postprocessed code, working with, [1074](#)
- PR format element, [266](#)
- PRAGMA AUTONOMOUS TRANSACTION statement, [706](#)
- PRAGMA keyword, [76](#)

- pragmas
  - autonomous transaction, 478
  - types of pragmas in PL/SQL, 77
- precision
  - datetime variables, 280
  - for NUMBER type
    - constraining, 243
    - range of values for precision, 246
    - scale exceeding precision, 244
  - INTERVAL variables, 286
- precompiler, Oracle's Pro\*C, 46
- predicates, 1001, 1003
  - application contexts as, 1022–1026
  - errors in, 1016
- primary keys
  - emulating in collections with string indexes, 383
  - object tables, 1155
  - OIDs based on, 1156, 1189
- PRINT command (SQL\*Plus), 32
- PRIOR method, 362
- private code, 656
- private key encryption, 975
- privileges
  - constraining invoker rights privileges, 1063
  - DDL trigger detailing effects of grant operations, 717
  - for object types, 1199
  - for stored programs, 42
  - privilege escalation and SQL injection, 1053
  - restriction on user schemas, 567
  - role-based, for program units, 1057–1060
  - for schema-level collections, 412
  - viewing granted privileges for users and roles, 42
- Pro\*C precompiler, 46
  - anonymous blocks, use of, 56
- Procedural Language extensions to the Structured Query Language (see PL/SQL)
- procedures, 8, 594–597
  - apply\_discount procedure (example), 595
  - body, 597
  - calling, 596
  - defined, 593
  - END label, 597
  - external (see external procedures)
  - general format of, 594
  - Java method call spec defined as procedure, 1229
  - parameters (see parameters)
  - procedure header, 596
  - RETURN statement, 597
  - using NULL statement as placeholder, 102
  - viewing in USER\_PROCEDES view, 757
- profilers, 827
  - DBMS\_PROFILER, 828
- program data, 173–197
  - conversions between datatypes (see datatypes, conversions between)
  - declaring, 181–188
    - anchored declarations, 183
    - anchoring to cursors or tables, 185
    - anchoring to NOT NULL datatypes, 188
    - benefits of anchored declarations, 186
    - constants, 182
    - NOT NULL clause, 183
    - variables, 181
  - granting roles to program units, 1057–1061
  - naming, 174
  - PL/SQL datatypes, 175–181
  - program-specific settings with inquiry directives, 1073
  - programmer-defined subtypes, 188
- programmer-defined exceptions, 131
  - declarations, 132
  - scope of, 139
- programmer-defined records, 326, 327–330
  - declaring record TYPES, 328
  - declaring the record, 329
  - examples of declarations, 329
- programs
  - size limit for PL/SQL compiler, 1045
- propagation of exceptions, 131, 150–153
  - examples of, 151
  - losing exception information, 150
- proxy server, using for web traffic, 966
- pseudo-columns, 419
- Pseudo-Random Number Generator (PRNG), 985
- pseudo-records (see NEW and OLD pseudo-records)
- pseudocolumn OBJECT\_VALUE, 1180
- pseudofunctions, 398–405
  - CAST, 399
  - COLLECT, 400
  - MULTISET, 400
  - TABLE, 402
- PSP (PL/SQL Server Pages), 51

- public code, 656
- Public Key Cryptography Standard #5 (PKCS#5), 977, 979
- public key encryption, 975
- PUT\_LINE function, CASE expression used with, 99

## Q

- q-prefixed strings, 203
- qualifying references, 59
  - labels as aid in, 78
  - reasons for, 60
- quantifiers in regular expressions, 217
  - nongreedy, 226
- query operations, typical, 488
- Quest Error Manager (QEM), 163
  - table of error definitions and error instances, 166
- queues, 967

## R

- RAISE statement, 131
  - forms of, 140
  - in exception section, example of, 133
- RAISE\_APPLICATION\_ERROR procedure, 131
  - errors raised by, use of error codes for, 162
  - using to raise exceptions, 141
- raising exceptions, 131
- random numbers, 985
- range operator (..), 66
- RANGE partitioning, 902
- ranges, numeric FOR loops, 115
- RAW type, 179, 417
  - converting VARCHAR2 type to, 980
  - HEXTORAW function, 196
  - returned by DECRYPT function, DBMS\_CRYPTO, 983
  - small file as email attachment, 953
- RAWTOHEX function, 196
- READ COMMITTED isolation level, 476
- read-only transactions, 476
- read-write transactions, 476
- readability
  - improving by labeling blocks, 78
  - improving with qualifiers, 60
- real numbers, 73
- record anchoring, 184

- records, 323–340
  - benefits of using, 324
  - collections as components of, 350
  - collections of, 386
  - comparing, 337
  - declaring, 326
  - DML and, 470
  - fetching from explicit cursors, 506
  - fetching into, using NDS, 546
  - field-level operations, 334
    - with nested records, 335
    - with package-based records, 336
  - in PL/SQL, 323
  - programmer-defined, 327–330
  - pseudo-records (see NEW and OLD pseudo-records)
  - record-level operations, 331
  - restrictions on record-based inserts and updates, 472
  - triggering pseudorecords, 338
  - using in UPDATE statements, 471
  - using with RETURNING clause, 472
- records equal generator, 338
- recursion, 630
- redirects (HTTP), 963
- REF CURSORS, 179
  - compatible REF CURSOR rowtype and select list, 524
  - declaring type, 521
  - FETCH INTO statements, compatible rowtype, 525
  - identifying REF CURSOR type of cursor
    - variable parameter, 530
  - never closed implicitly, 508
  - passing as parameters to parallel pipelined functions, 896
  - passing as pipelined function parameter, 890
  - rowtype matching at compile-time, 527
  - rowtype matching at runtime, 528
  - using with NDS OPEN FOR statement, 543
- references
  - invoker rights resolution of external references, 1056
  - object (see REFs)
- REFERENCING clause
  - using to change names of pseudo-records in DML triggers, 694
- referencing, indirect, 560

- REFs (object references), 1164–1171
  - and type hierarchies, 1170
  - dangling REFs, 1171
  - physical versus virtual, storeability of, 1196
  - system-generated OIDs, 1157
  - to nonunique OIDs, 1197
  - using, 1165–1168
  - using virtual REFs in object views, 1190
  - UTL\_REF package, 1168
- REGEXP\_COUNT function, 223, 1278
- REGEXP\_INSTR function, 218, 1278
- REGEXP\_LIKE function, 217, 1278
- REGEXP\_REPLACE function, 223, 1278
- REGEXP\_SUBSTR function, 220, 1279
- regular expressions, 216–227
  - counting matches, 223
  - detecting a pattern in strings, 216
  - extracting text matching a pattern, 220
  - functions for, 236, 1278
    - parameters, 1279
  - greedy matching, 226
  - locating a pattern within a string, 218
  - metacharacters, 1275–1278
  - replacing text in strings, 223
  - resources for further learning, 227
- RELIES\_ON clause, 854
  - as RESULT\_CACHE subclause, 857
- REM (remark or comment) lines in SQL\*Plus, 30
- remote dependencies, 769
- remote invocation model (Oracle), limitations of, 772
- REMOTE\_DEPENDENCIES\_MODE parameter, 770
- REPEAT UNTIL loop, emulating, 110
- REPLACE function, 212, 236
- replacing existing objects (CREATE OR REPLACE statement), 38
- replacing text in strings, 223
- reserved words, 68
  - how to avoid using, 69
- RESTRICT\_REFERENCES pragma, 77
- result sets, 487
- RESULT\_CACHE clause, 853
  - RELIES\_ON subclause, 857
- resumable statements, 737
- RETURN clause
  - cursors with, 501
  - datatype structures in, 503
  - reasons for using RETURN, 503
  - in functions, 601
    - RETURN statement versus, 606
  - packaged cursors with, 671
- RETURN keyword, 1262
- RETURN statements
  - in functions, 605
    - as last executable statement, 606
    - multiple RETURNS, 606
    - returning any valid expression, 606
  - procedures, 597
- RETURNING clause
  - bind argument modes in dynamic queries, 552
  - returning information from DML statements, 468
  - using records with, 472
  - using with bulk operations, 875
- REUSE SETTINGS clause, 779, 1073
- REVERSE keyword, using in numeric FOR loops, 115
- revoking privileges on stored programs, 42
- RLS (see row-level security)
- roles
  - granting EXECUTE privilege to, 42
  - granting in program units, 13
  - granting to PL/SQL program units, 1057–1060
  - viewing list of privileges granted to, 43
- ROLLBACK statements, 474
  - autonomous transactions and, 480
  - DML triggers and, 690
  - locks released after, 517
- rollbacks
  - in programs performing DML, 469
  - SET TRANSACTION USE ROLLBACK SEGMENT statement, 476
  - uncommitted changes in SQL\*Plus, prior to exiting, 36
  - with FORALL statements, 882
- Roman numerals, 260
- ROUND function, 271, 275
- rounding numbers
  - functions for, 271
  - NUMBER values, 244
    - effect of negative scale, 246
    - when scale exceeds precision, 245
  - TIMESTAMP values with TO\_CHAR function, 294



- when converting numbers to character strings, 264
- row number, 343
- row-level security (RLS), 865, 972, 999–1019
  - application contexts as predicates, 1022–1026
  - application users and, 1026
  - debugging, 1015–1019
  - diagram of RLS infrastructure, 1001
  - reasons for learning about, 1002
  - simple example, 1003–1007
  - static versus dynamic policies, 1007
    - context-sensitive policies, 1010
    - shared static policies, 1010
  - summary of, 1018
  - using column-sensitive RLS, 1012–1015
- row-level triggers, 689
- %ROWCOUNT attribute, 491
  - values before and after cursor operations, 510
  - values returned by, 510
- ROWID type, 179, 417
  - conversion of CHAR or VARCHAR2 to, 193
  - conversions of UROWID to, using CAST, 194
  - getting ROWIDs, 418
  - using, 419
- ROWIDTOCHAR function, 197
- rows, 343
- %ROWTYPE attribute, 326
  - anchoring to cursors or tables, 185
  - and invisible columns, 339
  - general form of %ROWTYPE declaration, 327
  - records defined with, fetching into with cursors, 506
  - use in record anchoring, 184
  - using with invisible columns, 14
- ROWTYPE\_MISMATCH exception, 525
- RPAD function, 213, 236
- RR format element, interpreting two-digit years, 299
- RTRIM function, 215, 236, 267

## S

- SAVE EXCEPTIONS clause, FORALL statement, 883
- SAVEPOINT statements, 475

- savepoints, 469
  - autonomous transactions and, 480
  - COMMIT statement and, 474
  - ROLLBACK statement and, 475
- scalar anchoring, 183
- scalars, 173
- scale, for NUMBER type
  - constraining, 243
  - effect of negative scale, 245
  - effect of scale exceeding precision, 244
  - range of scale values, 246
  - scale as optional and defaulting to zero, 246
- schema-level collections
  - maintaining, 412
    - collections and the data dictionary, 413
    - privileges, 412
- schema-level recompilation, 775
- schema-level type, defining with CREATE TYPE, 350
- schemas
  - g11n schema, USERS and LOCALE tables, 1127
  - preparation for edition-based redefinition, 822
  - query showing all Java-related objects in, 1222
  - user, restriction of privileges on, 567
- scope, 58
  - of cursor object, 529
  - of cursor parameters, 514
  - of exceptions, 131, 139
  - of local modules, 623
  - qualifying all references to variables and columns in SQL statements, 59
- scripts
  - anonymous blocks running from, 56
  - echoing contents of, 38
  - running from command-line environments, 22
  - running from SQL\*Plus, 29
- search strings, formatting for Oracle Text, 1123
- searched CASE statements, 95
- searching
  - case-insensitive, for strings, 209
  - traditional searching of strings, 210
- Secure Hash Algorithm (SHA-1 and SHA-2), 992, 994
- SecureFiles, 421, 436, 983
  - compression option, 437



- deduplication option, 437
- encryption option, 438
- SecureFiles and Large Objects Developer's Guide manual, 444
- security, 971
  - (see also application security)
  - configuring network security to send email, 946
  - enhanced security for DBMS\_SQL in Oracle Database 11g, 584
  - for external procedures, 1251
    - vulnerability of Oracle, 1248
  - Java security for Oracle from 8.1.6, 1210
  - Java security for Oracle through 8.1.5, 1209
  - Oracle's security alerts page, 1251
- Segment Space Management settings, 437
- SELECT DISTINCT operation, 1193
- SELECT INTO statements
  - collection variable initialization via, 372
  - populating single row of data in a collection, 376
- select list, 492
- SELECT statements, 7
  - as loops, 126
  - associating with a cursor, 486
  - compatible REF CURSOR rowtype and select list, 524
  - constructing for dynamic SQL method 4 with DBMS\_SQL, 573
  - as cursor body, 503
  - emulating triggers on, 688
  - implicit cursor in, 494
  - in dynamic SQL, 548
  - referencing PL/SQL variables in cursor's SELECT, 492
  - SELECT COUNT(\*), replacement of, 540
  - SELECT FOR UPDATE, 488, 515–519
    - explicit cursor in, 501
    - locking of rows for the query, 505
  - select list, 492
  - WHERE clause, 506
- selection directives, 1065
  - referencing packages, 1072
  - using application package constants in \$IF directive, 1065
- SELF AS supertype, 1153
- SELF keyword, 1150, 1262
- sequences
  - native PL/SQL support for, in Oracle Database 11g, 464
- SERIALIZABLE isolation level, 476, 480
- serializable packages, 674–676
- SERIALLY\_REUSABLE pragma, 77, 674, 1089
- server-based errors, getting more information on, 39
- SERVERERROR triggers, 724–728
  - central error handler, 727
  - examples of use, 725
- SERVEROUTPUT command (SQL\*Plus), 27
- servers
  - Oracle's built-in web server, 968
  - using proxy server for web traffic, 966
- service names (Oracle Net), 25
- session persistence, 657
- SESSIONTIMEZONE function, 283, 295
- SET clause, UPDATE statement, using record variables, 472
- SET command (SQL\*Plus), 31
  - SET EDITFILE, 34
- SET DEFINE OFF command, 205
- SET ECHO ON command (SQL\*Plus), 29, 38
- set operations, 402
  - performing on nested tables, 406, 409
- SET operator, 411
- SET TRANSACTION statements, 476
  - controlling transaction visibility, 480
- set-based inserts, performance improvement with, 892
- SET\_PLSQL\_TRACE procedure, 807
- severe compile-time warnings, 778
- sf\_timer package, 834
- SGA (system global area), 485
  - data caching in, 844
  - defined, 1076
  - tuning access to code and data in, 825
- SHA-1 and SHA-2 (Secure Hash Algorithm), 992, 994
- shared context-sensitive policies, 1012
- shared libraries, 1244
  - creating Oracle file for, 1255
  - dynamic or static linking, 1247
- shared policies, 1010
- shared static policies, 1010
- short-circuit evaluation, 91

- SHOW ERRORS command (SQL\*Plus)
  - appending after CREATE statements that build stored programs, 40
- SHOW ERRORS command, SQL\*Plus, 39
- show\_all\_arguments.sp file, 758
- SHUTDOWN triggers, 723
- signature
  - for referenced remote procedures, 769
- simple loops, 106, 108–112
  - emulating a REPEAT UNTIL loop, 110
  - intentionally infinite loop, 111
  - terminating with EXIT and EXIT WHEN, 109
- SIMPLE\_DOUBLE type, 177, 242, 256
- SIMPLE\_FLOAT type, 177, 242, 256
- SIMPLE\_INTEGER type, 178, 242, 249
  - performance gains versus using other numeric types, 251
- single-byte characters, translating to multibyte, 238
- single-line comments, syntax, 75
- skewed data, partitioning with, 903
- SMTP (Simple Mail Transfer Protocol), 945
  - conversation between PL/SQL mail client and SMTP server, 947
  - UTL\_SMTP package, 945
- SMTP\_OUT\_SERVER initialization parameter, 946
- soft-closed cursors, 1078
- sorting collection contents, 405
- SOUNDEX function, 236
- source code
  - debuggers, 811
  - stored in the database, displaying and searching, 754
- sparse versus dense (collections), 344
- specification (packages), 652, 656
  - example of, 653
  - rules for building, 658
- SPOOL command (SQL\*Plus), 33
- SQL (Structured Query Language), xxv
  - calling a Java method in, 1232
  - calling PL/SQL functions in, 631
  - context switching between PL/SQL and, 869
  - cursor attributes, inability to use in SQL statements, 492
  - DML statements, 463
  - dynamic (see dynamic SQL and dynamic PL/SQL)
  - dynamic and static SQL, 537
  - execution of anonymous PL/SQL block containing SQL, 1042
  - floating-point literals, 251
  - implicit cursor attributes, 466, 498
  - integration with PL/SQL, xxvi, 7
  - name resolution, 64
  - optimizing, 825
  - optimizing PL/SQL function execution in, 14
  - optimizing PL/SQL function performance in, 922
  - running statement in SQL\*Plus, 26
  - sharing SQL statements, 1079
  - SQL semantics for LOBs, 443–447
    - performance impact, 446
    - yielding temporary LOBs, 444
  - statement as loop, 126
  - static and dynamic, 487
  - viewing SQL statements during RLS debugging, 1018
  - working with collections, 398–406
    - CAST pseudofunction, 399
    - COLLECT pseudofunction, 400
    - MULTISET pseudofunction, 400
    - sorting contents, 405
    - TABLE pseudofunction, 402
- SQL Developer IDE, 45
- SQL Guard, 568
- SQL injection (see code injection)
- SQL Navigator IDE, 45
- SQL%FOUND attribute, 467, 498
- SQL%ISOPEN attribute, 466, 498
- SQL%NOTFOUND attribute, 466, 498
- SQL%ROWCOUNT attribute, 467, 498
  - determining number of rows modified by dynamic SQL statements, 541
- SQL\*Plus, 23–37
  - advantages and limitations of, 36
  - current directory, 30
  - editing a statement, 33
  - error handling, 36
  - exiting, 33
  - loading custom environment automatically on startup, 35
  - problems with ampersand (&) in PL/SQL code, 205
  - running a script, 29
  - running a SQL statement, 26

- running PL/SQL program, 27
- saving output to file, 32
- setting preferences, 31
- starting up, 24
- versions of, 23
- SQL\*Plus \_EDITOR variable, 34
- SQLCODE function, 134
  - combined with WHEN OTHERS, 156
  - defined, 144
  - values returned by (Oracle error codes), 137
- SQLERRM function, 144
  - useful applications of, 148
- SQLJ, 47
- SQLNET.ORA file, specifying encryption wallet location, 438
- SQLPATH environment variable, 35
- SSL-encrypted web page, retrieving via HTTPS, 960
- STANDARD package, 1045
  - core features of PL/SQL language and dependencies, 767
  - identifiers from, 69
  - predefined datatypes in, 175
  - predefined exceptions in, 132, 137
  - USER function, 846
- standard time and daylight saving time, 296
- START command (SQL\*Plus), 29
- STARTUP triggers, 722
- statement sharing, using to reduce memory use, 1079
- statement-level triggers, 689
- statements
  - editing in SQL\*Plus, 33
  - running SQL statement in SQL\*Plus, 26
  - termination with semicolon in PL/SQL, 75
- static methods, 1151
- static polymorphism, 625
- static SQL, 487, 537
- static typing, 175
- stepwise refinement, 621
- store table, 345
- stored procedures
  - creating to execute any DDL statement with EXECUTE IMMEDIATE, 540
- stored programs
  - dropping, 43
  - executing, 41
  - hiding source code, 44
  - managing grants and synonyms for, 42
  - showing, 42
- streaming clauses in parallel pipelined functions, 902
- streaming options, ordering data in parallel processes, 902
- streaming table functions, 535, 638
  - creating, 641–644
- string literals
  - case sensitivity, 71
  - embedding single quotes in, 73
- strings, 199–240
  - collections indexed by, 344
  - concatenating, 206
  - converting to and from datetimes
    - from datetimes to strings, 292
    - from strings to datetimes, 289
    - from time zones to character strings, 301
  - using CAST function, 308
  - converting to and from numbers
    - implicit conversions, 268
    - using CAST function, 267
    - using TO\_CHAR function, 261–267
    - using TO\_NUMBER function, 257–261
  - converting to intervals, 305
  - datatypes, 199–203
    - CHAR, 201
    - subtypes, 202
    - VARCHAR2, 200
  - dealing with case, 207
    - capitalizing each word, 209
    - case-insensitivity and indexes, 209
    - forcing string to all upper- or lowercase, 208
    - making comparisons case insensitive, 208
  - empty strings, 227
  - formatting search strings for Oracle Text, 1123
  - function quick reference, 231–240
  - functions for, 1105–1111
  - mixing CHAR and VARCHAR2 values, 229
  - null and zero-length, 72
  - padding, 213
  - regular expression searching, extracting, and replacing with, 216–227
  - sorting order for different languages, 1115–1120
  - specifying string constants, 203
  - string-indexed collections, 380–385

- traditional searching, extracting, and replacing, [210](#)
  - negative string positioning, [213](#)
  - trimming, [215](#)
  - using nonprintable characters, [205](#)
- STRING\_TO\_RAW function (UTL\_I18N), [980](#)
- string\_tracker package, [382](#)
  - extending with multilevel collections, [396](#)
- strong REF CURSOR, [179](#)
- strong type, [522](#)
- Structured Query Language (see SQL)
- subblocks, [58](#)
- SUBMULTISET operator, checking if nested table is contained in another nested table, [409](#)
- subnormal range of values, BINARY\_FLOAT, [252](#)
- subprograms
  - granting roles to, [13](#)
  - nested, sprucing up code with, [623](#)
  - overloading, [624–630](#)
  - PL/SQL, defining in SQL statements, [634](#)
- SUBSTR function, [211, 237](#)
  - negative string positioning, [213](#)
- SUBSTR functions, [1110](#)
  - SUBSTRB, [1111](#)
- SUBTYPE statement, [188](#)
- SUBTYPES, [184](#)
- subtypes
  - creating for object type, [1146](#)
  - invoking supertype methods from (Oracle 11g and later), [1152](#)
  - privilege to create, [1200](#)
  - programmer-defined, [188](#)
  - string, [202](#)
- suspended statements, [688](#)
  - AFTER SUSPEND triggers, [736–743](#)
- symmetric encryption algorithms, [975](#)
- synchronization with database columns, drawbacks of anchored declarations, [187](#)
- synonyms for stored programs, [43](#)
- SYS.STANDARD package, numeric functions, [272](#)
- SYSDATE function, [282](#)
  - date arithmetic with DATE datatypes, [312](#)
  - interpreting two-digit years, [300](#)
  - time-of-day and date values returned, [295](#)
- system exceptions, [130](#)
  - named, [136](#)

- predefining in dynamic SQL package, [136](#)
  - scope of, [139](#)
- system global area (see SGA)
- system-generated OIDs, [1156](#)
- SYSTIMESTAMP function, [280, 282](#)
- SYS\_CONTEXT function, [1021](#)
- SYS\_REFCURSOR type, [522](#)

## T

- table API, [677](#)
- table functions, [405, 465, 637–647](#)
  - calling in a FROM clause, [638](#)
  - creating a pipelined function, [644](#)
  - creating a streaming function, [641–644](#)
  - enabling for parallel execution, [646](#)
  - passing results with a cursor variable, [640](#)
  - pipelined, improving performance with, [888–916](#)
- TABLE pseudofunction, [402](#)
- table-based records, [326](#)
- tables
  - anchoring to, [185](#)
  - caching contents in a package, [848](#)
  - collections as, [343](#)
  - just-in-time caching of table data, [850](#)
- TCP sockets, I/O via, [968](#)
- TDE (transparent data encryption), [438, 995](#)
- templates for common error handling, [167](#)
- temporary LOBs, [439](#)
  - checking if LOB is temporary, [441](#)
  - creating, [440](#)
  - freeing, [441](#)
  - managing, [442](#)
  - yielded by SQL semantics, [444](#)
- termination condition (loop), [107](#)
- testing code, [788–795](#)
  - automated options for PL/SQL, [794](#)
  - general advice for testing PLS/QL code, [793](#)
  - reasons for inadequate testing, [788](#)
  - typical, inadequate testing techniques, [789](#)
- TEXT\_SEARCH\_FUNC function, [1122](#)
- THEN keyword, [84](#)
  - in IF-THEN-ELSE statements, [86](#)
  - in IF-THEN-ELSIF statements, [88](#)
- three-valued logic, [84](#)
- time zones, [278](#)
  - and DATE functions used with TIME-STAMPS, [320](#)
  - converting to character strings, [301](#)

- in datetime conversions, 295
- no standard for names and abbreviations, 298
- Oracle FAQ on, 297
- region names and abbreviations for, 1130
- session and database, 283
- timeout values, getting and setting, using DBMS\_RESUMABLE package, 741
- timer, DBMS\_UTILITY.GET\_CPU\_TIME, 1011
- TIMESTAMP datatypes, 178, 278, 1126
  - computing interval between, 313
  - considerations when choosing, 281
  - conversions to strings with TO\_CHAR function, 292
  - date functions and, 320
  - functions returning current date and time, 282
  - localization with appropriate date/time formatting, 1127–1131
  - mixing with DATES in datetime arithmetic, 316
  - NLS settings for, 309
  - TIMESTAMP type, 279
  - TIMESTAMP WITH LOCAL TIME ZONE type, 278, 279
  - TIMESTAMP WITH TIME ZONE type, 71, 278, 279
  - TIMESTAMP WITH TIME ZONE variable, 301
- timestamp literals, 303
- timestamps
  - for referenced remote procedures, 769
  - functions converting strings to, 290
  - Oracle FAQ on, 297
- TIMEZONE datatypes, using EXTRACT function with, 310
- title case, 235
- TM format model element, 266
- tnsnames.ora file, 1249, 1251
  - modifying to use two multithreaded agents, 1254
- TNS\_ADMIN environment variable, 1251
- Toad IDE, 22, 45
- TOO\_MANY\_ROWS exception
  - as unexpected exception, 161
  - handling with implicit cursors, 497
- top-down design, 621
- TO\_BINARY\_DOUBLE function, 257
- TO\_BINARY\_FLOAT function, 257
- TO\_CHAR function, 238, 261–267
  - converting datetime values to strings, 292
  - dealing with spaces, 265
  - passing NLS settings to, 267
  - rounding numbers, 264
  - using with format model, 262
  - using without format, 262
  - V format element, 263
- TO\_DATE function, 290
  - date format elements, 291
  - easing up on exact format mask matches, 299
  - requiring format mask to match exactly, 298
- TO\_DSINTERVAL function, 305
- TO\_MULTI\_BYTE function, 238
- TO\_NCHAR function, 238
  - converting datetime values to national character set, 293
- TO\_NUMBER function, 258–261
  - passing NLS settings to, 260
  - using with format model, 259
  - using without format string, 258
- TO\_SINGLE\_BYTE function, 238
- TO\_TIMESTAMP function, 290
  - date format elements, 291
- TO\_TIMESTAMP\_TZ function, 290, 295
- TO\_YMINTERVAL function, 305
- tracing PL/SQL execution, 795–808
  - debugging versus tracing, 795
  - guidelines for, 795
  - toggling tracing through conditional compilation flags, 1066
  - using DBMS\_APPLICATION\_INFO package, 801
  - using DBMS\_TRACE package, 804–808
  - using DBMS\_UTILITY.FORMAT\_CALL\_STACK function, 796
  - using opp\_trace utility, 803
  - using UTL\_CALL\_STACK package, 798
- training materials, xxxiii
- transaction API, 677
- transactions, 461
  - ACID principle, 461
  - autonomous, 477–484
    - building autonomous logging mechanism, 482
    - defining, 478
    - rules and restrictions on, 479
    - transaction visibility, 480

- when to use, 481
- defined, 473
- DML triggers' participation in, 690
- full support provided by external procedures, 1248
- integrity of, 6
- management of
  - COMMIT statement, 474
  - LOCK TABLE statement, 476
  - PL/SQL statements for, 473
  - ROLLBACK statement, 474
  - SAVEPOINT statement, 475
  - SET TRANSACTION statement, 476
- TRANSACTIONS parameter, database initialization file, 480
- transformative functions, 535
- TRANSLATE function, 238
- TRANSLATE...USING function, 239
- transparent data encryption (TDE), 438, 995
- transparent tablespace encryption (TTE), 997
- TREAT function, 1160, 1170, 1171
- triggers (see database triggers)
- trigonometric functions, 272
- TRIM function, 215, 239, 267
- TRIM method, 363
- Triple DES (DES3) algorithm, 976
- TRUE and FALSE values, 74
- TRUE values, 178
- TRUNC function, 271, 276
  - use with DATE subtraction, 314
  - use with MONTHS\_BETWEEN function, 315
- TTE (transparent tablespace encryption), 997
- %TYPE attribute
  - NOT NULL declaration constraint for variables declared with, 188
  - use in scalar anchoring, 183
- type evolution, 1162
- TYPE...RECORD statement, 326, 327
- TZH (time zone hour), 296
- TZR (time zone region), 296

## U

- UDF pragma, 637
  - improving performance of functions executed from within SQL, 922
- UGA (user global area), 1041, 1085
  - defined, 1077
  - finding how much current session uses, 1087

- unbounded versus bounded (collections), 344
- unconstrained declarations, 608
- unconstrained subtypes, 189
- UNDER keyword, 1147
- UNDER privilege, 1200
- unexpected exceptions, 159, 161
  - guidelines for dealing with, 162
- unfortunate exceptions, 159, 161
  - guidelines for dealing with, 162
- unhandled exceptions, 131, 149
  - propagation of, 150–153
  - and rollbacks in execution environments, 470
- Unicode, 1100–1111
  - and your environment, 1103
  - character encodings, 1102
  - collation charts, 1116
  - converting strings to, 240
  - defined, 1100
  - functions for, 1105–1111
  - Globalization Support (NLS) parameters, 1104, 1104
  - multilingual sort, 1119
  - national character set datatypes, 1102
  - specifying string characters by Unicode code point, 204
  - UTF-8 character set, 200
- UNION operator, 402
  - combining data from database table and collection, 405
  - MULTISET UNION, 409
- unique indexes, emulating in collections with string indexes, 383
- UNISTR function, 240, 1111
- unit name and line number, referencing, 1069
- unit-qualified names, obtaining with
  - UTL\_CALL\_STACK, 798
- unnamed or anonymous exceptions, 131
- upcasting, 1148
  - while dereferencing, 1170
- UPDATE statements, 7, 464
  - restrictions on record-based updates, 472
  - RETURNING clause, 468
  - using FORALL with, 878
  - using records in, 470, 471
  - WHERE CURRENT OF clause, 518
- updates
  - controlling with RLS, 1000
  - RLS policies and, 1006

- row-level locks to avoid losing, 993
- UPDATING function, 695
- UPPER function, 208
- URI types, 180, 451
- URIType, 180
- UriType, 451
- UROWID type, 179, 417
  - conversions to ROWID, using CAST, 194
- US7ASCII character set, 65
- user administration (GDK), 1138
- USER function, 846
  - caching value returned by, 846
- user global area (see UGA)
- User-Agent header, 957
- user-defined datatypes, 181
- user-defined delimiters, 73
- user-defined functions
  - calling from SQL
    - read consistency and, 633
    - restrictions on, 632
- users
  - granting EXECUTE privilege to, 42
  - identifying nondatabase users with applica-
    - tion contexts, 1026–1028
- USERS table in g11n schema, 1127
- USER\_\* views, 751
- USER\_ARGUMENTS view, 758
- USER\_DEPENDENCIES view, 764
- USER\_ERRORS view, querying, 39
- USER\_IDENTIFIERS view, 760
- USER\_JAVA\_POLICY view, 1211
- USER\_OBJECTS view, 753, 1222
  - invalidated dependent program units, 765
  - object\_name, containing names of Java sche-
    - ma objects, 1223
  - querying for complete list of programs, 42
  - verifying that class is loaded, 1216
- USER\_OBJECT\_SIZE view, 755
- USER\_PLSQL\_OBJECT\_SETTINGS view, 752, 756, 760, 1066
  - compilation environment parameters, 1068
- USER\_PROCEDURES view, 757
- USER\_SOURCE view, 754, 1197
- USER\_TAB\_PRIVS\_MADE data dictionary
  - view, 43
- USER\_TRIGGERS view, 757
- USER\_TRIG\_COLUMNS view, 757
- USING clause
  - EXECUTE IMMEDIATE statement, 539
  - OPEN FOR statement, 543, 547
    - use with dynamic PL/SQL execution, bind
      - variables, 552
- UTC (Coordinated Universal Time), 278
  - defined, 280
  - displacement for time zone, 295, 301
  - time zone as UTC offset, 1130
- UTF-8 Unicode character set, 1101
- UTL\_CALL\_STACK package, 14, 798
  - examples of use, 799
  - important points about, 801
  - subprograms, 798
- UTL\_ENCODE package, base64 conversion
  - with, 953
- UTL\_FILE package, 925, 929–944
  - closing files with FCLOSE or FCLOSE\_ALL, 934
  - copying files with FCOPY, 941
  - deleting files with FREMOVE, 942
  - directories and, 931
  - exceptions defined in, 168
  - exceptions raised by FREMOVE program, 169
  - IS\_OPEN function, 934
  - limitations of, 1236
  - modifying files, procedures for, 933
  - opening files with FOPEN, 932
  - reading from files with GET\_LINE, 935
    - encapsulation for GET\_LINE, 936
    - exceptions, 936
  - renaming and moving files with FRENAME, 943
  - retrieving file attributes with FGETATTR, 943
  - specifying file locations when opening files
    - with FOPEN, 930
  - using in parallel pipelined function, 899
  - UTL\_FILE\_DIR parameter, 929
  - writing to files, procedures for, 933, 938
    - PUT\_LINE procedure, 939
    - writing formatted text with PUTF, 940
- UTL\_HTTP package, 956
  - cookies, 965
  - fetching a LOB, 958
  - HTTPS retrievals, 960
  - redirects, following, 963
  - retrieving a web page in pieces, 956
  - submitting data to web page via GET or
    - POST, 961



- support for HTTP authentication, 959
- UTL\_I18N package, 1133–1136
  - STRING\_TO\_RAW function, 980
- UTL\_IL8N package, globalization support, 982
- UTL\_LMS package, 1136
- UTL\_MAIL package
  - SEND procedure, 944
  - SEND\_ATTACH\_RAW, 953
  - SEND\_ATTACH\_VARCHAR2, 952
  - setting up and using, 945
- UTL\_RECOMP package, 776
- UTL\_REF package, 1168
- UTL\_SMTP package, 945
  - sending short, plain-text message, 947
- UTL\_TCP package, 945, 968
- UTL\_URL.ESCAPE function, 962

## V

- V number format element, 263
- V\$OPEN\_CURSOR view, 1078
- V\$PARAMETER view, 1114
- V\$SQLAREA view, 1082
- V\$TEMPORARY\_LOBS view, 442
- V\$TIMEZONE\_NAMES view, 1130
- valid and invalid names for, 67
- VALIDATE clause, DROP TYPE statement, 1164
- VALUE function, 1157
- VALUES clause, INSERT statement
  - using record variables, 472
- VALUES OF clause, FORALL statement, 879, 885, 887
- VALUE\_ERROR exception, 183
- VARCHAR subtype, 203
- VARCHAR2 type, 176, 200
  - assigning zero-length string to, 72
  - conversion to ROWID, 193
  - converting NUMBER to and from, using
    - CAST function, 267
  - converting to RAW, 980
  - declaration, 200
  - empty strings considered as NULLs, 228
  - maximum length, 201
  - mixing with CHAR values in strings, 229
  - string-indexed collections, 380
  - use in argument definitions, 758
  - using CLOBs interchangeably with, 442
  - UTL\_MAIL.SEND\_ATTACH\_VARCHAR2, 952

- variable attribute notation, 1158
- variables
  - assigning value of string constant to, 204
  - collection, 369–374
  - creating and manipulating in SQL\*Plus, 31
  - declaring, 181
    - anchored declarations, 183
    - NOT NULL clause for default value, 183
  - fetching from explicit cursors, 506
  - fetching into, using NDS, 546
  - in program data, 173
  - naming, 174
  - qualifying references to, 59
  - scope, 58
  - visibility of, 62–65
- VARRAYs, 455
  - accessing data in, 380
  - collection type, 343, 345
    - changing VARRRAY characteristics, 369
    - comparison to associative arrays and nested tables, 354
    - declaring, 368
    - example of, 348
  - database-to-PL/SQL integration, 373
  - declaring and initializing collection variables of type, 370
  - difference between nested tables as column datatypes, 353
- versions
  - DBMS\_DB\_VERSION package information
    - on installed database, 1067
  - Oracle database, xxxiii
  - Oracle databases, xxvii
- views
  - BEQUEATH CURRENT\_USER, 1061
  - BEQUEATH\_CURRENT\_USER, 13
  - database trigger, 745
  - dynamic performance views, 862
  - for external procedures, 1272
  - object, 1184–1197
    - data dictionary entries for, 1197
  - VPD (virtual private database), 1018
- virtual columns, 507
- virtual private database (VPD), 1000
  - (see also row-level security)
  - function result caching and, 865, 865
  - views, 1018
- visibility (transactions), 480
- visibility of variables, 62–65



visible identifiers, [62](#)

Visual Basic, PL/SQL called from, [52](#)

## W

WAIT keyword, appending to FOR UPDATE clause, [516](#)

wallets, [438](#), [960](#), [995](#)

warnings

compile-time (see compile-time warnings)

performance-related, [837](#)

weak REF CURSOR, [179](#)

weak type, [522](#)

web server (Oracle), built-in, [968](#)

web-based data (HTTP), working with, [956–967](#)  
authentication using HTTP, [959](#)  
disabling cookies or making them persistent, [965](#)

retrieving data from FTP server, [966](#)

retrieving SSL-encrypted web page via HTTPS, [960](#)

retrieving web page in pieces, [956](#)

retrieving web page into a LOB, [958](#)

submitting data to web page via GET or POST, [961–965](#)

using a proxy server, [966](#)

websites for PL/SQL developers, [16](#)

WHEN clause

applying to DML trigger (example), [699](#)

in DML triggers, [692](#)

in CASE expressions, [99](#)

in DML triggers, [690](#)

in exception section

RAISE statement within, [141](#)

trapping only named exceptions, [144](#)

in searched CASE statements, [96](#), [97](#)

in simple CASE statements, [94](#)

multiple row-level BEFORE INSERT triggers  
with mutually exclusive WHEN clauses, [702](#)

of exception section, [133](#)

WHEN OTHERS clause, [138](#), [144](#), [144](#), [146](#), [154](#)

warning about programs ignoring errors, [157](#)

writing code to handle exceptions, [155](#)

WHenever SQLERROR EXIT ROLLBACK command, [33](#)

WHenever SQLERROR EXIT  
SQL.SQLCODE command, [36](#)

WHERE clause

DELETE statement, [465](#)

obtaining count of rows in any table for, using EXECUTE IMMEDIATE, [540](#)

REF-based navigation in, [1167](#)

SELECT statement, [493](#)

UPDATE statement, [464](#)

WHERE CURRENT OF clause, UPDATE and DELETE statements, [518](#)

WHILE loops, [107](#), [112](#)

boundary and body of, [107](#)

exiting, [124](#)

whitelisting, using to control access to program units, [816](#)

whitespace

dealing with spaces in number to character string conversions, [265](#)

in regular expressions, [221](#)

keywords and, [70](#)

wide, denormalized records, [910](#), [915](#)

widening or upcasting, [1148](#)

while dererencing, [1170](#)

WITH FUNCTION clause, [637](#), [922](#)

word, defined, [233](#), [235](#)

WORLD\_LEXER, [1122](#)

WRAP and CREATE\_WRAPPED programs of DBMS\_DDL package, [819](#)

wrap executable, [819](#)

wrap utility, [44](#)

WRAPPED keyword, [821](#)

wrapping code, [818](#)

building PL/SQL wrapper for Java code, [1217](#)

dynamic wrapping with DBMS\_DDL, [819](#)

guidelines for working with wrapped code, [821](#)

restrictions on and limitations of wrapping, [818](#)

## X

X format element, [291](#)

XDBUriType, [451](#)

XML

datatypes for working with, [180](#)

working with, documentation, [451](#)

XMLType, [180](#), [448–451](#)

creating table to hold XML data, [448](#)

indexing columns for retrieval of XML documents, [451](#)

- querying XML data in a table, 449
- retrieving data into PL/SQL variable of XMLType, 450

- XPath, 448
  - defined, 449
- XQuery, 448

## Y

- years, interpreting two-digit years, 299

- YMINTERVAL\_UNCONSTRAINED type, 319

## Z

- zip/compression functionality using a Java class, 1240

## About the Authors

---

Steven Feuerstein is considered one of the world's leading experts on the Oracle PL/SQL language. He is the author or coauthor of *Oracle PL/SQL Programming*, *Oracle PL/SQL Best Practices*, *Oracle PL/SQL Programming: Guide to Oracle8i Features*, *Oracle PL/SQL Developer's Workbook*, *Oracle Built-in Packages*, *Advanced Oracle PL/SQL Programming with Packages*, and several pocket reference books (all from O'Reilly Media). Steven is a Senior Technology Advisor with Quest Software, has been developing software since 1980, and worked for Oracle Corporation from 1987 to 1992.

Bill Pribyl is the primary author of *Learning Oracle PL/SQL* and the coauthor of *Oracle PL/SQL Programming* and its companion pocket reference, all from O'Reilly Media. Currently heading (and learning from) a small team of programmers at an international trading firm, Bill has used PL/SQL to write TCP/IP networking clients, tnsping callouts, near-real-time commodity price loaders, and transcendental functions. With a degree in physics from Rice University, Bill's non-working hours are largely invested in supporting his wife, who is surviving inflammatory breast cancer.

## Colophon

---

Ants are featured on the cover of *Oracle PL/SQL Programming*, Sixth Edition. At least 8,000 different species of ants can be found everywhere on Earth except the north and south poles. Ants preserved in amber suggest that these insects existed 50 million years before humans.

Humans have long been fascinated by ants because these tiny insects are accomplished builders, nurses, miners, and even farmers. Fables such as "The Ant and the Grasshopper" extol the virtues of hardworking, forward-looking ants. (Hail ants!) It is true that individual ants are able to perform amazing feats; an ant can carry up to 50 times its body weight, can travel the human equivalent of 40 miles a day, and can climb vertical heights the equivalent of Mount Everest. However, the greatest accomplishments of ants are those performed together for the good of their community.

Queen ants establish new communities, or nests, after their mating flight. On this flight, the queen mates with several males. After mating, the males fall to Earth and die. The queen then finds an uninhabited nest, settles into it, and pulls her wings off. She will never fly again, and after removing her wings she is able to absorb the wing muscles as nutrients for her eggs. She will continue to lay eggs, thousands of them, for years. During the three-stage development process, which takes about two months, the eggs, larvae, and pupae are cared for by the nurse ants who feed, clean, and carefully move the young to warmer or cooler places in the nest, depending on the temperature. These nurse ants are, in turn, cared for by other worker ants, who feed the nurses with regurgitated food. The workers and the nurses will fight together to defend the young against enemies if the nest is invaded, either by another group of ants or by a larger animal.

The cover image is a 19th century engraving from the Dover Pictorial Archive. The cover fonts are URW Typewriter and Guardian Sans. The text font is Adobe Minion Pro; the heading font is Adobe Myriad Condensed; and the code font is Dalton Maag's Ubuntu Mono.