APPENDIX A

Regular Expression Metacharacters and Function Parameters

This appendix describes the various regular expression metacharacters available starting with Oracle Database 10*g*. 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 10*g* Release 1 and which in Release 2.

Table A-1. Character-matching metacharacters

Syntax	Initial release	Description
.	10*g*R1	Matches any single character except for newline. Will match newline when the *n* flag is set. On Windows, Linux, and Unix platforms, chr(10) is recognized as the newline.
[...]	10*g*R1	Defines a *matching list* 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 *nonmatching list* (see the next entry).
[^ ...]	10*g*R1	Matches any character not listed between the brackets. Referred to as a *nonmatching list*.
[:*class*:]	10*g*R1	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. Table A-5 lists the valid character class names.
[.*coll*.]	10*g*R1	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 *ch*. Table A-4 lists the valid collation elements.

Syntax	Initial release	Description
[=*char*=]	10*g*R1	Matches all characters that share the same base character as char. May be used only within a matching list. For example, [[=e=]] matches any of: "eéëèÉËÈE".
\d	10*g*R2	Matches any digit. Equivalent to [[:digit:]].
\D	10*g*R2	Matches any nondigit. Equivalent to [^[:digit:]].
\w	10*g*R2	Matches any *word character*. Word characters are defined to be alphabetic characters, numeric characters, and the underscore.
\W	10*g*R2	Matches any nonword character.
\s	10*g*R2	Matches any whitespace character. Equivalent to [[:space:]].
\S	10*g*R2	Matches nonwhitespace characters. Equivalent to [^[:space:]].

Table A-2. Quantifiers

Syntax	Initial release	Description
?	10*g*R1	Zero or one.
*	10*g*R1	Zero or more.
+	10*g*R1	One or more.
{*m*}	10*g*R1	Exactly *m* occurrences.
{*m*,}	10*g*R1	At least *m* occurrences.
{*m*,*n*}	10*g*R1	At least *m*, and at most *n* occurrences.
+?	10*g*R2	One or more, but nongreedy.
??	10*g*R2	Zero or one, but nongreedy.
{*m*}?	10*g*R2	The same as {*m*}.
{*m*,}?	10*g*R2	At least *m* occurrences, but nongreedy and stops as soon as *m* occurrences are reached.
{*m*,*n*}?	10*g*R2	At least *m*, and at most *n* occurrences, but nongreedy; when possible, *m* occurrences are matched.

Table A-3. Other metacharacters

Syntax	Initial release	Description
|	10*g*R1	Specifies an alternation. An alternation within a subexpression doesn't extend beyond the subexpression.
(...)	10*g*R1	Defines a subexpresson.
n	10*g*R1	References the text matched by the *n*th subexpression. Backreferences may range from \1 through \9.
\	10*g*R1	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.
^	10*g*R1	Anchors an expression to the beginning of the string (in multiline mode, to the beginning of a line).
$	10*g*R1	Anchors an expression to the end of the string (in multiline mode, to the end of a line).
\A	10*g*R2	Anchors an expression to the beginning of the string regardless of whether multiline mode is specified.

Syntax	Initial release	Description
\Z	10*g*R2	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	10*g*R2	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.

APPENDIX B
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 *text minimum*. 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 *n*th power, where *n* 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.

APPENDIX C
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 Oracle8*i* 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.

Index

Symbols

We'd like to hear your suggestions for improving our indexes. Send email to index@oreilly.com.

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z

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.