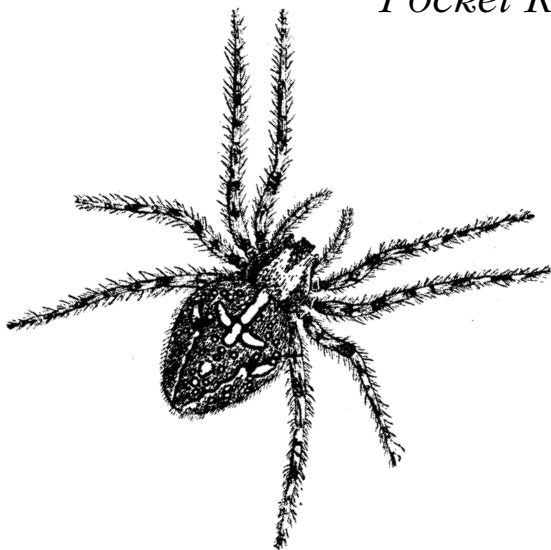


Tutorial and Quick Reference



Oracle Regular Expressions

Pocket Reference



O'REILLY®

*Jonathan Gennick
& Peter Linsley*

Oracle Regular Expressions

Pocket Reference

Jonathan Gennick and Peter Linsley

O'REILLY®

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

Tutorial

A *regular expression* (often known as a *regex*) is a sequence of characters that describe a pattern in text. Regular expressions use a syntax that has evolved over a number of years, and that is now codified as part of the POSIX standard.

Regular expressions are extremely useful, because they allow you to work with text in terms of patterns. For example, you can use regular expressions to search the park table and identify any park with a description containing text that looks like a phone number. You can then use the same regular expression to extract that phone number from the description.

NOTE

This tutorial will get you started using regular expressions, but we can only begin to cover the topic in this small book. If you want to learn about regular expressions in depth, see Jeffrey Friedl's excellent book *Mastering Regular Expressions* (O'Reilly).

Patterns

The simplest type of *pattern* is simply an exact string of characters that you are searching for, such as the string in the following WHERE clause:

```
SELECT *
FROM park
WHERE park_name='Mackinac Island State Park';
```

However, the string 'Mackinac Island State Park' isn't what most people think of when you mention the word "pattern." The expectation is that a pattern will use so-called *metacharacters* that allow for matches when you know only the general pattern of text you are looking for.

Standard SQL has long had rather limited support for pattern matching in the form of the LIKE predicate. For example, the

following query attempts to return the names of all state parks:

```
SELECT park_name
FROM park
WHERE park_name LIKE '%State Park%';
```

The percent (%) characters in this pattern specify that any number of characters are allowed on either side of the string 'State Park'. Any number of characters may be zero characters, so strings in the form 'xxx State Park' fit the pattern. There! I've just used a pattern to describe the operation of a pattern.

NOTE

Humans have long used patterns as a way to organize and describe text. Look no further than your address and phone number for examples of commonly used patterns.

Handy as it is at times, LIKE is an amazingly weak predicate, supporting only two expression metacharacters that don't even begin to address the range of patterns you might need to describe in your day-to-day work. You need more. You need a richer and more expressive language for describing patterns. You need regular expressions.

Regular Expressions

Regular expressions is the answer to the question: "How do I describe a pattern of text?" Regular expressions first became widely used on the Unix platform, supported by such utilities as *ed*, *grep*, and (notably) Perl. Regular expressions have gone on to become formalized in the IEEE POSIX standard, and regular expressions are widely supported across an ever-growing range of editors, email clients, programming languages, scripting languages, and now Oracle SQL and PL/SQL.

Let's revisit the earlier problem of finding state parks in the park table. We performed that task using LIKE to search for the words 'State Park' in the park_name column. Following is the regular expression solution to the problem:

```
SELECT park_name
FROM park
WHERE REGEXP_LIKE(park_name, 'State Park');
```

REGEXP_LIKE is a new Oracle predicate that searches, in this case, the park_name column to see whether it contains a string matching the pattern 'State Park'. REGEXP_LIKE is similar to LIKE, but differs in one major respect: LIKE requires its pattern to match the entire column value, whereas REGEXP_LIKE looks for its pattern anywhere within the column value.

There are no metacharacters in the regular expression 'State Park', so it's not a terribly exciting pattern. Following is a more interesting example that attempts to identify parks with descriptions containing phone numbers:

```
SELECT park_name
FROM park
WHERE REGEXP_LIKE(description, '...-....');
```

This query uses the regular expression metacharacter period (.), which matches any character. The expression does not look for three periods followed by a dash followed by four periods. It looks for any three characters, followed by a dash, followed by any four characters. Because it matches any character, the period is a very commonly used regular expression metacharacter.

NOTE

You can create function-based indexes to support queries using REGEXP_LIKE and other REGEXP functions in the WHERE clause.

It can be inconvenient to specify repetition by repeating the metacharacter, so regular expression syntax allows you to follow a metacharacter with an indication of how many times you want that metacharacter to be repeated. For example:

```
SELECT park_name
FROM park
WHERE REGEXP_LIKE(description, '{3}-{4}');
```

The *interval quantifier* {3} is used to specify that the first period should repeat three times. The {4} following the second period indicates a repeat count of four. This pattern of a regular expression syntax element followed by a quantifier is one you'll frequently encounter and use when working with regular expressions.

Table 1 illustrates the types of quantifiers you can use to specify repetition in a regular expression. The later “Regular Expression Quick Reference” section describes each of these quantifiers in detail.

Table 1. Quantifiers used to specify repetition in a pattern

Pattern	Matches
.*	Zero or more characters
.+	One or more characters
.?	Zero or one character
{3,4}	Three to four characters
{3,}	Three or more characters
{3}	Exactly three characters