TECHNICAL REPORT

ISO/IEC TR 19075-1

First edition
2011-07-15

Information technology — Database languages — SQL Technical Reports —

Part 1:
XQuery Regular Expression Support in SQL

Technologies de l'information — Langages de base de données — Rapport techniques SQL —

Partie 1: Support d'expressions régulières de XQuery en SQL

Reference number
ISO/IEC TR 19075-1:2011(E)

Contents

(Blank page)

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

In exceptional circumstances, when the joint technical committee has collected data of a different kind from that which is normally published as an International Standard ("state of the art", for example), it may decide to publish a Technical Report. A Technical Report is entirely informative in nature and shall be subject to review every five years in the same manner as an International Standard.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC TR 19075-1 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange*.

ISO/IEC TR 19075 consists of the following parts, under the general title *Information technology — Database languages — SQL Technical Reports*:

— *Part 1: XQuery Regular Expression Support in SQL*

Introduction

The organization of this part of ISO/IEC TR 19075 is as follows:

1) Clause 1, “Scope”, specifies the scope of this part of ISO/IEC TR 19075.
2) Clause 2, “XQuery regular expressions”, explains how XQuery regular expressions are formed.
3) Clause 3, “Operators using regular expressions”, explains how the SQL operators use regular expressions.

Information technology — Database languages — SQL Technical Reports —

Part 1:
XQuery Regular Expression Support in SQL

1 Scope

This Technical Report describes the regular expression support in SQL adopted from the regular expression syntax of [XQuery F&O], which is derived from Perl. This Technical Report discusses five operators using this regular expression syntax:

— LIKE_REGEX predicate, to determine the existence of a match to a regular expression.

— OCCURRENCES_REGEX numeric function, to determine the number of matches to a regular expression.

— POSITION_REGEX function, to determine the position of a match.

— SUBSTRING_REGEX function, to extract a substring matching a regular expression.

— TRANSLATE_REGEX function, to perform replacements using a regular expression.

(Blank page)

2 XQuery regular expressions

XQuery regular expression syntax is specified in [XQuery F&O], section 7.6.1, “Regular expression syntax”. This paper references the XQuery specification, with two small modifications (required since character strings in an RDBMS are not necessarily normalized according to XML conventions). The following subsections provide an overview of this syntax.

The XQuery regular expression syntax is itself a modification of another regular expression syntax found in [XML Schema: Datatypes].

This section presents an overview of the capabilities of XQuery regular expression syntax. In the process, this section will illustrate some of the SQL operators. The SQL operators themselves are presented in the next section.

The following discussion does not cover every aspect of XQuery regular expressions; for this, [XQuery F&O] is the reference (though hardly a tutorial; a variety of popular works contain detailed treatments of regular expressions).

2.1 Matching a specific character

Perhaps the most elementary pattern matching requirement is the ability to match a single character or string. For most characters, this is done by simply writing the character in the regular expression. For example, suppose you want to know if a string S contains the letters “`xyz`”. This could be done with the following predicate:

```
S LIKE_REGEX 'xyz'
```

Note that the SQL LIKE predicate would require an exact match for “`xyz`”. However, the convention with regular expressions is that S need only contain a substring that is “`xyz`”. For example, all of the following values of S would yield *True* for the predicate above:

```
xyz
abcxyz123
1 xyz 2 xyz 3 xyz
```

Note that in the last example, there are actually three occurrences of the regular expression “`xyz`” within the tested value. The user may wish to know the number of occurrences of a match. This can be done with OCCURRENCES_REGEX. For example:

```
OCCURRENCES_REGEX ('xyz' IN '1 xyz 2 xyz 3 xyz') = 3
```

The user might also wish to know the position of a specific match. This can be done using POSITION_REGEX. For example, to learn the starting character position of the second occurrence,

```
POSITION_REGEX ( 'xyz' IN '1 xyz 2 xyz 3 xyz' OCCURRENCE 2 ) = 9
```

It is also possible to ask for the character position of the first character after the match. For example:

```
   POSITION_REGEX ( AFTER 'xyz' IN '1 xyz 2 xyz 3 xyz' OCCURRENCE 2 ) = 12
```

If AFTER is used and the last character of the subject string is consumed, then the result is the length of the string plus 1 (one):

```
   POSITION_REGEX ( AFTER 'xyz' IN 'xyz' ) = 4
```

2.2 Metacharacters and escape sequences

As mentioned, most characters can be matched by simply writing the character in the regular expression. However, certain characters are reserved as *metacharacters*. The complete list of metacharacters is:

```
   . \ ? * + { } ( ) | [ ] ^ $
```

The use of each of these metacharacters will be explained later. If you want to match a metacharacter, then you need to use an *escape sequence*, consisting of a backslash (“`/`”) followed by the metacharacter. For example, to test whether a string contains a dollar sign, you could write

```
   S LIKE_REGEX '\$'
```

In particular, the escape sequence representing a backslash is two consecutive backslashes. There are various other defined escape sequences, matching either a single character, or any of a group of characters. The *single character escape sequence*s are:

\n	newline (U+000A)
\r	return (U+000D)
\t	tab (U+0009)
\-	minus sign ('-')

The so-called *category escape*s are exemplified by “`\p{L}`” or “`\p{Lu}`”. A category escape begins with “`\p{`” followed by one uppercase letter, optionally a lowercase letter, and then the closing brace. In these example, “`\p{L}`” matches any letter (as defined by Unicode) and “`\p{Lu}`” matches any uppercase letter. Some interesting category escapes are listed below:

\p{L}	Any letter.
\p{Lu}	Any uppercase letter.
\p{Ll}	Any lowercase letter.
\p{Nd}	Any decimal digit.
\p{P}	Any punctuation mark.
\p{Z}	Any separator (space, line, paragraph, *etc*.).

The complete list of category escapes is found in [XML Schema: Datatypes], section F.1.1, “Character class escapes”.

There are also *complementary category escape*s, which are exemplified by “`\P{L}`” or “`\P{Lu}`”. A complementary category escape matches any character that would not be matched by the corresponding category escape. The difference is that the (positive) character escape is written with a lowercase “`p`” whereas the complementary character escape is written with an uppercase “`P`”.

The so-called *block escape*s match any character in a block of Unicode, that is, a predefined consecutive range of code points. For example, “`\p{IsBasicLatin}`” matches the ASCII character set. There are also *complementary block escape*s, such as “`\P{IsBasicLatin}`”, which matches any single character that is not an ASCII character.

Finally, there are the following *multi-character escape sequence*s:

\s	As defined by [XML Schema: Datatypes], this escape matches space (U+0020), tab (U+0009), newline (U+000A), or return (U+000D). Since character strings in an RDBMS have not undergone XML line termination normalization, we broaden it to include any character or two-character sequence that is recognized by [Unicode18] as a line terminator. Subclause 2.5, “Line terminators”, discusses this issue further.
\S	Any single character not matched by `\s`.
\i	Underscore (“_”), colon (“`:`”) or letter (this is a lot more than just the Latin letters; see [XML 1.0] appendix B, rule [84]).
\I	Any single character not matched by `\i`.
\c	Any single character matched by NameChar, as defined in [XML 1.0] section 2.3, rule [4].
\C	Any single character not matched by `\c`.
\d	Any single digit
\D	Any single character not matched by `\d`.
\w	Any single Unicode character except those classified as “punctuation”, “separator”, or “other”.
\W	The complement of `\w`.

2.3 Dot

Dot (period, “`.`”) is a metacharacter that is used to match any single character (the same behavior as “_” in LIKE predicates), or any single character that is not a line terminator. The default is to match anything except a line terminator. The alternative, called *dot-all mode*, is specified using a flag that contains a lowercase “`s`”.

For example

```
S LIKE_REGEX 'a.b'
```

matches the following:

```
'xa0by'
```

but not the following:

```
'xa
by'
```

because the character between the “`a`” and the “`b`” is a line terminator. However, using dot-all mode like this:

```
S LIKE_REGEX 'a.b' FLAG 's'
```

would match both examples.

2.4 Anchors

We have seen that regular expressions look for a match anywhere within a string, without needing to match the entire string. But what if you want to require a match of the entire string? For this, you can use *anchors*. The anchors are the metacharacters "`^`" for the start of a string (or line), and "`$`" for the end of a string (or line). For example:

```
S LIKE_REGEX '^xyz$'
```

can only match a string that is precisely 'xyz'.

Anchors may be used separately to require a "begins with" or "end with" match. For example

```
S LIKE_REGEX '^xyz'
```

matches any string that begins with "`xyz`", and

```
S LIKE_REGEX 'xyz$'
```

matches any string that ends with "`xyz`".

Instead of matching the begin or end of the string, the anchors may be used to anchor a match to the begin or end of a line, by performing the match in *multi-line mode*. Multi-line mode is specified using a flag containing a lowercase "`m`". For example:

```
S LIKE_REGEX '^xyz' FLAG 'm'
```

performs an anchored search in multi-line mode, matching any string containing a line that begins with "`xyz`". The example above would match the following string:

```
'line one
xyz
line three'
```

2.5 Line terminators

The metacharacters "`.`", "`^`", and "`$`" and the multi-character escape sequences "`\s`" and "`\S`" are defined in terms of a "line terminator". What counts as a line terminator? [XQuery F&amp;O] only recognizes a line feed (U+000A) as a line terminator. This definition works well for XQuery, because XML normalizes the line terminators on various platforms to a line feed.

A closer look shows that XML has two definitions of line handling, in section 2.11, "End-of-line handling", of [XML 1.0] and [XML 1.1]. So which should we use for SQL?

A first stop in answering this is to look at [SQL/XML WD] Subclause 6.17, "<XML query>", which requires XML 1.0 as a basic level of support, and permits XML 1.1 support in the form of Feature X211, "XML 1.1