TECHNICAL REPORT

ISO/IEC TR 19075-5

First edition
2016-12-15

Information technology — Database languages — SQL Technical Reports —

Part 5: Row Pattern Recognition in SQL

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

Partie 5: Reconnaissance de formes de lignes dans SQL

Reference number
ISO/IEC TR 19075-5:2016(E)

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Contents

Page

Foreword. vii
Introduction. viii
1 Scope. 1
2 Normative references. 3
2.1 ISO and IEC standards. 3
3 Row pattern recognition: FROM clause. 5
3.1 Example of ONE ROW PER MATCH. 5
3.2 Example of ALL ROWS PER MATCH. 8
3.3 Summary of the syntax. 10
3.4 The row pattern input table. 11
3.4.1 The row pattern input name. 12
3.4.2 The row pattern input declared column list. 13
3.5 MATCH_RECOGNIZE. 14
3.6 PARTITION BY. 14
3.7 ORDER BY. 14
3.8 Row pattern variables. 14
3.9 MEASURES. 16
3.10 ONE ROW PER MATCH *vs.* ALL ROWS PER MATCH. 16
3.10.1 Handling empty matches. 16
3.10.2 Handling unmatched rows. 20
3.11 AFTER MATCH SKIP. 22
3.12 PATTERN. 24
3.12.1 PERMUTE. 25
3.12.2 Excluding portions of the pattern. 26
3.13 SUBSET. 27
3.14 DEFINE. 28
3.15 The row pattern output table. 29
3.15.1 Row pattern output name. 30
3.15.2 Row pattern output declared column list. 30
3.16 Prohibited nesting. 31
3.16.1 Row pattern recognition nested within another row pattern recognition. 32
3.16.2 Outer references within a row pattern recognition query. 32
3.16.3 Conventional query nested within row pattern recognition query. 33
3.16.4 Recursion. 34
3.16.5 Concatenated row pattern recognition. 34

4 Expressions in MEASURES and DEFINE. 35
4.1 Row pattern column references. 35
4.2 Running *vs.* final semantics. 36
4.3 RUNNING *vs.* FINAL keywords. 40
4.4 Aggregates. 41
4.5 Row pattern navigation operations. 42
4.5.1 PREV and NEXT. 42
4.5.2 FIRST and LAST. 43
4.5.3 Nesting FIRST and LAST within PREV or NEXT. 45
4.6 Ordinary row pattern column references reconsidered. 46
4.7 MATCH_NUMBER function. 47
4.8 CLASSIFIER function. 47
5 Row pattern recognition: WINDOW clause. 51
5.1 Example of row pattern recognition in a window. 51
5.2 Summary of the syntax. 53
5.2.1 Syntactic comparison to windows without row pattern recognition. 54
5.2.2 Syntactic comparison to MATCH_RECOGNIZE. 55
5.3 Row pattern input table. 55
5.4 Row pattern variables and other range variables. 56
5.5 Windows defined on windows. 57
5.6 PARTITION BY. 58
5.7 ORDER BY. 58
5.8 MEASURES. 58
5.9 Full window frame and reduced window frame. 59
5.9.1 ROWS BETWEEN CURRENT ROW AND. 59
5.9.2 EXCLUDE NO OTHERS. 59
5.10 AFTER MATCH SKIP. 60
5.11 INITIAL *vs.* SEEK. 60
5.12 PATTERN. 60
5.13 SUBSET. 61
5.14 DEFINE. 61
5.15 Empty matches and empty reduced window frames. 61
5.16 Prohibited nesting. 63
5.16.1 Row pattern recognition nested within another row pattern recognition. 63
5.16.2 Outer references within a row pattern recognition query. 63
5.16.3 Conventional query nested within row pattern recognition query. 64
5.16.4 Recursion. 64
5.16.5 Concatenated row pattern recognition. 65
6 Pattern matching rules. 67
6.1 Regular expression engines. 67
6.2 Parenthesized language and preferment. 68
6.2.1 Alternation. 69
6.2.2 Concatenation. 69

6.2.3 Quantification. 70
6.2.4 Exclusion. 71
6.2.5 Anchors. 72
6.2.6 The empty pattern. 72
6.2.7 Infinite repetitions of empty matches. 72
6.3 Pattern matching in theory and practice. 75
Index. 79

Tables

Table

Page

1 Sample Data. . . . 8
2 Results of ONE ROW PER MATCH. . . . 8
3 Results of ALL ROWS PER MATCH. . . . 9
4 Row pattern recognition syntax summary. . . . 10
5 Analysis of sample data permitting empty matches. . . . 17
6 Result of query permitting empty matches. . . . 18
7 Results of query using SHOW EMPTY ROWS. . . . 19
8 Results of query using OMIT EMPTY ROWS. . . . 20
9 Results of ALL ROWS PER MATCH. . . . 21
10 Original and renamed column names. . . . 31
11 Ordered row pattern partition of data. . . . 37
12 RUNNING and FINAL in MEASURES. . . . 38
13 Ordered row pattern partition of data. . . . 39
14 Ordered row pattern partition of data. . . . 40
15 Example data set and mappings for FIRST and LAST. . . . 44
16 Data set and mappings for nesting example. . . . 45
17 Window Example Query Results. . . . 53
18 Row pattern recognition in windows — syntax summary. . . . 54
19 Results for empty match and no match. . . . 61
20 Computation of matches and window function results. . . . 62
21 Input data. . . . 75
22 Mapping of first element. . . . 76
23 Mapping of second element. . . . 76
24 Mapping of third element. . . . 77

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.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

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. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: www.iso.org/iso/foreword.html.

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, SC 32, *Data management and interchange*.

A list of all the parts in the ISO 19075 series, can be found on the ISO website.

Introduction

This Technical Report discusses the syntax and semantics for recognizing patterns in rows of a table, as defined in [ISO9075-2].

[ISO9075-2] defines two features regarding row pattern recognition:

— Feature R010, “Row pattern recognition: FROM clause”

— Feature R020, “Row pattern recognition: WINDOW clause”

These two features have considerable syntax and semantics in common, the principle difference being whether the syntax is placed in the FROM clause or in the WINDOW clause.

The organization of this Technical Report is as follows:

1) Clause 1, “Scope”, specifies the scope of this Technical Report.

2) Clause 2, “Normative references”, identifies standards that are referenced by this Technical Report.

3) Clause 3, “Row pattern recognition: FROM clause”, discusses Feature R010, “Row pattern recognition: FROM clause”.

4) Clause 4, “Expressions in MEASURES and DEFINE”, discusses scalar expression syntax in row pattern matching.

5) Clause 5, “Row pattern recognition: WINDOW clause”, discusses Feature R020, “Row pattern recognition: WINDOW clause”. Clause 5, “Row pattern recognition: WINDOW clause”, does not duplicate material already presented in Clause 3, “Row pattern recognition: FROM clause” and Clause 4, “Expressions in MEASURES and DEFINE”, which should be read even if the reader is only interested in Feature R020, “Row pattern recognition: WINDOW clause”.

6) Clause 6, “Pattern matching rules”, discusses the formal rules of pattern matching.

Information technology — Database languages — SQL Technical Reports —

Part 5:
Row Pattern Recognition in SQL

1 Scope

This Technical Report discusses the syntax and semantics for recognizing patterns in rows of a table, as defined in [ISO9075-2].

(Blank page)

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

2.1 ISO and IEC standards

[ISO9075-2] ISO/IEC 9075-2:2016, *Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation)*.

(Blank page)

3 Row pattern recognition: FROM clause

Feature R010, “Row pattern recognition: FROM clause” of [ISO9075-2] enhances the capability of the FROM clause with a MATCH_RECOGNIZE clause to specify a row pattern. The syntax and semantics of a row pattern is discussed through examples presented throughout this Clause of this Technical Report.

There are two principal variants of the MATCH_RECOGNIZE clause:

1) ONE ROW PER MATCH, which returns a single summary row for each match of the pattern (the default).

2) ALL ROWS PER MATCH, which returns one row for each row of each match. There are three suboptions, to control whether to also return empty matches or unmatched rows.

3.1 Example of ONE ROW PER MATCH

The following example illustrates MATCH_RECOGNIZE with the ONE ROW PER MATCH option. Let Ticker (Symbol, Tradeday, Price) be a table with three columns representing historical stock prices. Symbol is a character column, Tradeday is a date column, and Price is a numeric column.

NOTE 2 — All examples in this Technical Report use mixed-case identifiers for the names of tables, columns, *etc.*, whereas SQL key words are shown in uppercase. Unquoted identifiers are actually equivalent to uppercase, so the column headings of sample results will be shown with the identifiers converted to uppercase.

It is desired to partition the data by Symbol, sort it into increasing Tradeday order, and then detect maximal “V” patterns in Price: a strictly falling price, followed by a strictly increasing price. For each match to a V pattern, it is desired to report the starting price, the price at the bottom of the V, the ending price, and the average price across the entire pattern.

The following query may be used to solve this pattern matching problem:

```
SELECT M.Symbol, /* ticker symbol */
       M.Matchno, /* sequential match number */
       M.Startp, /* starting price */
       M.Bottomp, /* bottom price */
       M.Endp, /* ending price */
       M.Avgp /* average price */
FROM Ticker
       MATCH_RECOGNIZE (
         PARTITION BY Symbol
         ORDER BY Tradeday
         MEASURES MATCH_NUMBER() AS Matchno,
                  A.Price AS Startp,
                  LAST (B.Price) AS Bottomp,
                  LAST (C.Price) AS Endp,
                  AVG (U.Price) AS Avgp
         ONE ROW PER MATCH
         AFTER MATCH SKIP PAST LAST ROW
         PATTERN (A B+ C+)
         SUBSET U = (A, B, C)
```

3.1 Example of ONE ROW PER MATCH

```
        DEFINE /* A defaults to True, matches any row */
               B AS B.Price < PREV (B.Price),
               C AS C.Price > PREV (C.Price)
    ) AS M
```

In the example above, the principal syntactic elements of MATCH_RECOGNIZE are presented on separate lines. In this example:

— Ticker is the name of the row pattern input table. In this example, the row pattern input table is a table or view. The row pattern input table may also be a derived table (in-line view).

— MATCH_RECOGNIZE introduces the syntax for row pattern recognition.

— PARTITION BY specifies how to partition the row pattern input table. The PARTITION BY clause is a list of columns of the row pattern input table. This clause is optional; if omitted, there are no row pattern partitioning columns, and the entire row pattern input table constitutes a single row pattern partition.

— ORDER BY specifies how to order the rows within row pattern partitions. The ORDER BY clause is a list of columns of the row pattern input table. This clause is optional; if omitted, the order of rows in row pattern partitions is completely non-determinstic. However, since non-deterministic ordering will defeat the purpose of most row pattern recognition, the ORDER BY clause will usually be specified.

— MEASURES specifies row pattern measure columns, whose values are calculated by evaluating expressions related to the match. The first row pattern measure column in this example uses the special nullary function MATCH_NUMBER(), whose value is the sequential number of a match within a row pattern partition. The third and fourth row pattern measure columns in this example use the LAST operation, which obtains the value of an expression in the last row that is mapped by a row pattern match to a row pattern variable. LAST is one of the row pattern navigation operations introduced by [ISO9075-2], discussed in Subclause 4.5, “Row pattern navigation operations”.

The result of the MATCH_RECOGNIZE clause is called the row pattern output table. When ONE ROW PER MATCH is specified, as in this example, the row pattern output table has one column for each row pattern partitioning column and one column for each row pattern measure column.

— ONE ROW PER MATCH specifies that the row pattern output table will have a single row for each match that is found in the row pattern input table.

— AFTER MATCH SKIP clause specifies where to resume looking for the next row pattern match after successfully finding a match. In this example, AFTER MATCH SKIP PAST LAST ROW specifies that pattern matching will resume after the last row of a successful match.

— PATTERN specifies the row pattern that is sought in the row pattern input table. A row pattern is a regular expression using primary row pattern variables. In this example, the row pattern has three primary row pattern variables (A, B, and C).

— SUBSET defines the union row pattern variable U as the union of the primary row pattern variables A, B, and C.

— DEFINE specifies the Boolean condition that defines a primary row pattern variable; a row must satisfy the Boolean condition in order to be mapped to a particular primary row pattern variable. This example uses PREV, a row pattern navigation operation that evaluates an expression in the previous row. If a primary row pattern variable is not defined in the DEFINE clause, then the definition defaults to a condition that is always true, meaning that any row can be mapped to the primary row pattern variable.

— AS M defines the range variable M to associate with the row pattern output table. This clause is optional; if omitted, then an implementation-dependent range variable is used. Since an implementation-dependent

range variable is unknowable to the query writer, the AS clause should not be omitted if there are any other tables in the FROM clause aside from the MATCH_RECOGNIZE.

The processing of MATCH_RECOGNIZE is as follows:

1) The row pattern input table is partitioned according to the PARTITION BY clause. Each row pattern partition consists of the set of rows of the row pattern input table that are equal (more precisely, not distinct) on the row pattern partitioning columns.

2) Each row pattern partition is ordered according to the ORDER BY clause.

3) Each ordered row pattern partition is searched for matches to the PATTERN.

4) Pattern matching operates by seeking the match at the earliest row, considering the rows in a row pattern partition in the order specified by the ORDER BY. When there is more than one match at a row, then the most preferred match is taken. The precise rules of pattern matching are discussed in Clause 6, “Pattern matching rules”.

5) After a match is found, row pattern matching calculates the row pattern measure columns, which are expressions defined by the MEASURES clause.

6) Using ONE ROW PER MATCH, as shown in the example, row pattern recognition generates one row for each match that is found.

7) The AFTER MATCH SKIP clause determines where row pattern matching resumes within a row pattern partition after a non-empty match has been found. In the example above, row pattern matching resumes at the next row after the rows mapped by a match (AFTER MATCH SKIP PAST LAST ROW).

Here is sample data for one row pattern partition of Ticker, shown sorted according to the ORDER BY clause. The sample data contains two matches to the pattern, indicated by arrows showing the mapping to primary row pattern variables in each match.