



BSI Standards Publication

Information technology — Guidance for the use of database language SQL

Part 5: Row pattern recognition

National foreword

This British Standard is the UK implementation of ISO/IEC 19075-5:2021. It supersedes PD ISO/IEC TR 19075-5:2016, which is withdrawn.

The UK participation in its preparation was entrusted to Technical Committee IST/40, Data management and interchange.

A list of organizations represented on this committee can be obtained on request to its committee manager.

Contractual and legal considerations

This publication has been prepared in good faith, however no representation, warranty, assurance or undertaking (express or implied) is or will be made, and no responsibility or liability is or will be accepted by BSI in relation to the adequacy, accuracy, completeness or reasonableness of this publication. All and any such responsibility and liability is expressly disclaimed to the full extent permitted by the law.

This publication is provided as is, and is to be used at the recipient's own risk.

The recipient is advised to consider seeking professional guidance with respect to its use of this publication.

This publication is not intended to constitute a contract. Users are responsible for its correct application.

© The British Standards Institution 2021
Published by BSI Standards Limited 2021

ISBN 978 0 539 13082 9

ICS 35.060

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 30 September 2021.

Amendments/corrigenda issued since publication

Date	Text affected
------	---------------

**Information technology — Guidance
for the use of database language
SQL —**

**Part 5:
Row pattern recognition**

*Technologies de l'information — Recommandations pour l'utilisation
du langage de base de données SQL —*

Partie 5: Reconnaissance de formes de lignes



Reference number
ISO/IEC 19075-5:2021(E)

AVA Documents
Document Sharing Hub
© ISO/IEC 2021



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2021

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier; Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword.....	vii
Introduction.....	ix
1 Scope.....	1
2 Normative references.....	2
3 Terms and definitions.....	3
4 Row pattern recognition: FROM clause.....	4
4.1 Context of row pattern recognition.....	4
4.2 Introduction to the FROM clause in row pattern recognition.....	4
4.3 Example of ONE ROW PER MATCH.....	4
4.4 Example of ALL ROWS PER MATCH.....	7
4.5 Summary of the syntax.....	9
4.6 The row pattern input table.....	10
4.6.1 Introduction to the row pattern input table.....	10
4.6.2 The row pattern input name.....	11
4.6.3 The row pattern input declared column list.....	12
4.7 MATCH_RECOGNIZE.....	13
4.8 PARTITION BY.....	13
4.9 ORDER BY.....	13
4.10 Row pattern variables.....	13
4.11 MEASURES.....	14
4.12 ONE ROW PER MATCH vs ALL ROWS PER MATCH.....	15
4.12.1 Introduction to use of ROWS PER MATCH.....	15
4.12.2 Handling empty matches.....	15
4.12.3 Handling unmatched rows.....	19
4.13 AFTER MATCH SKIP.....	21
4.14 PATTERN.....	22
4.14.1 Introduction to the PATTERN syntax.....	22
4.14.2 PERMUTE.....	23
4.14.3 Excluding portions of the pattern.....	24
4.15 SUBSET.....	25
4.16 DEFINE.....	26
4.17 The row pattern output table.....	27
4.17.1 Introduction to the row pattern output table.....	27
4.17.2 Row pattern output name.....	28
4.17.3 Row pattern output declared column list.....	28
4.18 Prohibited nesting.....	29
4.18.1 Introduction to prohibited nesting.....	29
4.18.2 Row pattern recognition nested within another row pattern recognition.....	30

4.18.3	Outer references within a row pattern recognition query.	30
4.18.4	Conventional query nested within row pattern recognition query.	31
4.18.5	Recursion.	32
4.18.6	Concatenated row pattern recognition.	32
5	Expressions in MEASURES and DEFINE.	33
5.1	Introduction to the use of expressions in MEASURES and DEFINE.	33
5.2	Row pattern column references.	33
5.3	Running vs. final semantics.	34
5.4	RUNNING vs.FINAL keywords.	38
5.5	Aggregates.	39
5.6	Row pattern navigation operations.	39
5.6.1	The four operations.	39
5.6.2	PREV and NEXT.	39
5.6.3	FIRST and LAST.	41
5.6.4	Nesting FIRST and LAST within PREV or NEXT.	42
5.7	Ordinary row pattern column references reconsidered.	43
5.8	MATCH_NUMBER function.	44
5.9	CLASSIFIER function.	44
6	Row pattern recognition: WINDOW clause.	48
6.1	Introduction to the WINDOW clause.	48
6.2	Example of row pattern recognition in a window.	48
6.3	Summary of the syntax.	50
6.3.1	Syntax components.	50
6.3.2	Syntactic comparison to windows without row pattern recognition.	51
6.3.3	Syntactic comparison to MATCH_RECOGNIZE.	52
6.4	Row pattern input table.	52
6.5	Row pattern variables and other range variables.	52
6.6	Windows defined on windows.	54
6.7	PARTITION BY.	55
6.8	ORDER BY.	55
6.9	MEASURES.	55
6.10	Full window frame and reduced window frame.	55
6.10.1	Introduction to window framing.	55
6.10.2	ROWS BETWEEN CURRENT ROW AND.	56
6.10.3	EXCLUDE NO OTHERS.	56
6.11	AFTER MATCH SKIP.	56
6.12	INITIAL vs SEEK.	57
6.13	PATTERN.	57
6.14	SUBSET.	57
6.15	DEFINE.	57
6.16	Empty matches and empty reduced window frames.	57
6.17	Prohibited nesting.	59
6.17.1	Restrictions on nesting.	59
6.17.2	Row pattern recognition nested within another row pattern recognition.	60
6.17.3	Outer references within a row pattern recognition query.	60
6.17.4	Conventional query nested within row pattern recognition query.	61
6.17.5	Recursion.	61

6.17.6	Concatenated row pattern recognition.	61
7	Pattern matching rules.	63
7.1	Regular expression engines.	63
7.2	Parenthesized language and preferment.	64
7.2.1	Introduction to parenthesized language and preferment.	64
7.2.2	Alternation.	65
7.2.3	Concatenation.	65
7.2.4	Quantification.	66
7.2.5	Exclusion.	67
7.2.6	Anchors.	68
7.2.7	The empty pattern.	68
7.2.8	Infinite repetitions of empty matches.	68
7.3	Pattern matching in theory and practice.	70
Index.		73

Tables

Table	Page
1 Sample data.	7
2 Results of ONE ROW PER MATCH.	7
3 Results of ALL ROWS PER MATCH.	8
4 Row pattern recognition syntax summary.	9
5 Analysis of sample data permitting empty matches.	16
6 Result of query permitting empty matches.	16
7 Results of query using SHOW EMPTY ROWS.	18
8 Results of query using OMIT EMPTY ROWS.	18
9 Results of ALL ROWS PER MATCH.	20
10 Original and renamed column names.	29
11 Ordered row pattern partition of data.	35
12 RUNNING and FINAL in MEASURES.	36
13 Ordered row pattern partition of data.	37
14 Ordered row pattern partition of data.	37
15 Example data set and mappings for FIRST and LAST.	41
16 Data set and mappings for nesting example.	43
17 Window example query results.	50
18 Row pattern recognition in windows — syntax summary.	51
19 Results for empty match and no match.	58
20 Computation of matches and window function results.	59
21 Input data.	71
22 Mapping of first element.	71
23 Mapping of second element.	72
24 Mapping of third element.	72

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.

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 or www.iec.ch/members_experts/refdocs).

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), or the IEC list of patent declarations received (see patents.iec.ch).

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

For an explanation of the voluntary nature of standards, 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 www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange*.

This first edition of ISO/IEC 19075-5 cancels and replaces ISO/IEC TR 19075-5:2016.

This document is intended to be used in conjunction with the following editions of the parts of the ISO/IEC 9075 series:

- ISO/IEC 9075-1, sixth edition or later;
- ISO/IEC 9075-2, sixth edition or later;
- ISO/IEC 9075-3, sixth edition or later;
- ISO/IEC 9075-4, seventh edition or later;
- ISO/IEC 9075-9, fifth edition or later;
- ISO/IEC 9075-10, fifth edition or later;
- ISO/IEC 9075-11, fifth edition or later;
- ISO/IEC 9075-13, fifth edition or later;
- ISO/IEC 9075-14, sixth edition or later;
- ISO/IEC 9075-15, second edition or later;
- ISO/IEC 9075-16, first edition or later.

ISO/IEC 19075-5:2021(E)

A list of all parts in the ISO/IEC 19075 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/-national-committees.

Introduction

This document discusses the syntax and semantics for recognizing patterns in rows of a table, as defined in [ISO/IEC 9075-2](#).

The organization of this document is as follows:

- 1) [Clause 1, “Scope”](#), specifies the scope of this document.
- 2) [Clause 2, “Normative references”](#), identifies additional standards that, through reference in this document, constitute provisions of this document.
- 3) [Clause 3, “Terms and definitions”](#), defines the terms and definitions used in this document.
- 4) [Clause 4, “Row pattern recognition: FROM clause”](#), discusses Feature R010, “Row pattern recognition: FROM clause”.
- 5) [Clause 5, “Expressions in MEASURES and DEFINE”](#), discusses scalar expression syntax in row pattern matching.
- 6) [Clause 6, “Row pattern recognition: WINDOW clause”](#), discusses Feature R020, “Row pattern recognition: WINDOW clause”. [Clause 6, “Row pattern recognition: WINDOW clause”](#), does not duplicate material already presented in [Clause 4, “Row pattern recognition: FROM clause”](#) and [Clause 5, “Expressions in MEASURES and DEFINE”](#), which should be read even if the reader is only interested in Feature R020, “Row pattern recognition: WINDOW clause”.
- 7) [Clause 7, “Pattern matching rules”](#), discusses the formal rules of pattern matching.

Information technology — Guidance for the use of database language SQL —

Part 5:

Row pattern recognition

1 Scope

This document discusses the syntax and semantics for recognizing patterns in rows of a table, as defined in [ISO/IEC 9075-2](#), commonly called “SQL/RPR”.

SQL/RPR 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.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements 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.

ISO/IEC 9075-1, *Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework)*

ISO/IEC 9075-2, *Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation)*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 9075-1 apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- IEC Electropedia: available at <http://www.electropedia.org/>
- ISO Online browsing platform: available at <http://www.iso.org/obp>

4 Row pattern recognition: FROM clause

4.1 Context of row pattern recognition

The requirements for the material discussed in this document shall be as specified in [ISO/IEC 9075-1](#) and [ISO/IEC 9075-2](#).

4.2 Introduction to the FROM clause in row pattern recognition

Feature R010, “Row pattern recognition: FROM clause” of SQL/RPR 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 document.

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.

4.3 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 1 — All examples in this document 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)  
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-deterministic. 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 SQL/RPR, discussed in [Subclause 5.6, “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 shall 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

4.3 Example of ONE ROW PER MATCH

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 7, “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).

[Table 1, “Sample data”](#), illustrates 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.

Table 1 — Sample data

SYMBOL	TRADEDAY	PRICE		
XYZ	2009-06-08	50		
XYZ	2009-06-09	60	→ A	}
XYZ	2009-06-10	49	→ B	
XYZ	2009-06-11	40	→ B	} first match
XYZ	2009-06-12	35	→ B	
XYZ	2009-06-15	45	→ C	}
XYZ	2009-06-16	45		
XYZ	2009-06-17	45	→ A	}
XYZ	2009-06-18	43	→ B	
XYZ	2009-06-19	47	→ C	} second match
XYZ	2009-06-22	52	→ C	
XYZ	2009-06-23	70	→ C	}
XYZ	2009-06-24	60		

The result of the example for this row pattern partition is shown in Table 2, “Results of ONE ROW PER MATCH”.

Table 2 — Results of ONE ROW PER MATCH

SYMBOL	MATCHNO	STARTP	BOTTOMP	ENDP	AVGP
XYZ	1	60	35	45	45.8
XYZ	2	45	43	70	51.4

4.4 Example of ALL ROWS PER MATCH

The previous example can be modified slightly to illustrate ALL ROWS PER MATCH, as follows:

```
SELECT M.Symbol, /* ticker symbol */
       M.Matchno, /* sequential match number */
       M.Tradeday, /* day of trading */
       M.Price, /* price on day of trading */
       M.Classy, /* classifier */
       M.Startp, /* starting price */
       M.Bottomp, /* bottom price */
```

4.4 Example of ALL ROWS PER MATCH

```

M.Endp, /* ending price */
M.Avgp /* average price */
FROM Ticker
MATCH_RECOGNIZE (
  PARTITION BY Symbol
  ORDER BY Tradeday
  MEASURES MATCH_NUMBER() AS Matchno,
             CLASSIFIER() AS Classy,
             A.Price AS Startp,
             FINAL LAST (B.Price) AS Bottomp,
             FINAL LAST (C.Price) AS Endp,
             FINAL AVG (U.Price) AS Avgp
  ALL ROWS PER MATCH
  AFTER MATCH SKIP PAST LAST ROW
  PATTERN (A B+ C+)
  SUBSET U = (A, B, C)
  DEFINE /* A defaults to True, matches any row */
         B AS B.Price < PREV (B.Price),
         C AS C.Price > PREV (C.Price)
) AS M

```

Note that the second row pattern measure column in this example shows the use of the special function `CLASSIFIER()`, which returns the name of the row pattern variable to which a row is mapped. `CLASSIFIER` is discussed in [Subclause 5.9, “CLASSIFIER function”](#).

The result of this query on the sample data is shown in [Table 3, “Results of ALL ROWS PER MATCH”](#).

Table 3 — Results of ALL ROWS PER MATCH

SYM BOL	MA TC HN O	TRADEDAY	PRICE	CLAS SY	STAR TP	BOT TOMP	ENDP	AVGP
XYZ	1	2009-06-09	60	A	60	35	45	45.8
XYZ	1	2009-06-10	49	B	60	35	45	45.8
XYZ	1	2009-06-11	40	B	60	35	45	45.8
XYZ	1	2009-06-12	35	B	60	35	45	45.8
XYZ	1	2009-06-15	45	C	60	35	45	45.8
XYZ	2	2009-06-17	45	A	45	43	70	51.4
XYZ	2	2009-06-18	43	B	45	43	70	51.4
XYZ	2	2009-06-19	47	C	45	43	70	51.4
XYZ	2	2009-06-22	52	C	45	43	70	51.4
XYZ	2	2009-06-23	70	C	45	43	70	51.4

ALL ROWS PER MATCH differs from ONE ROW PER MATCH in the following respects:

- 1) ALL ROWS PER MATCH returns one row for each row of each match of the pattern.

- 2) The row pattern output table has a column corresponding to every column of the row pattern input table, not just the row pattern partitioning columns. (Note the column M.Price in the SELECT list. This is a column of the row pattern input table, not a row pattern measure column.)
- 3) The MEASURES clause supports two semantics for expression evaluation, running semantics and final semantics, indicated by the keywords RUNNING and FINAL.
- 4) ALL ROWS PER MATCH provides three suboptions for handling empty matches and unmatched rows. These options are not illustrated in this example; see [Subclause 4.12.2, “Handling empty matches”](#), and [Subclause 4.12.3, “Handling unmatched rows”](#), for examples of these options.

4.5 Summary of the syntax

The complete syntax for row pattern recognition in the FROM clause involves the components shown in [Table 4, “Row pattern recognition syntax summary”](#).

Table 4 — Row pattern recognition syntax summary

Syntactic component	Optional?	Default	Cross reference
row pattern input table	no	—	Subclause 4.6, “The row pattern input table”
row pattern input name	yes	implementation-dependent	Subclause 4.6.2, “The row pattern input name”
row pattern input declared column list	yes	none	Subclause 4.6.3, “The row pattern input declared column list”
MATCH_RECOGNIZE	no	—	Subclause 4.7, “MATCH_RECOGNIZE”
PARTITION BY	yes	row pattern input table constitutes one row pattern partition	Subclause 4.8, “PARTITION BY”
ORDER BY	yes	non-deterministic ordering in each row pattern partition	Subclause 4.9, “ORDER BY”
MEASURES	yes	none	Subclause 4.11, “MEASURES”
ONE ROW PER MATCH or ALL ROWS PER MATCH	yes	ONE ROW PER MATCH	Subclause 4.12, “ONE ROW PER MATCH vs ALL ROWS PER MATCH”
AFTER MATCH SKIP	yes	AFTER MATCH SKIP PAST LAST ROW	Subclause 4.13, “AFTER MATCH SKIP”
PATTERN	no	—	Subclause 4.14, “PATTERN”
SUBSET	yes	no explicit union row pattern variables	Subclause 4.15, “SUBSET”

Syntactic component	Optional?	Default	Cross reference
DEFINE	no	—	Subclause 4.16, “DEFINE”
row pattern output name	yes	implementation-dependent	Subclause 4.17.2, “Row pattern output name”
row pattern output declared column list	yes	none	Subclause 4.17.3, “Row pattern output declared column list”

4.6 The row pattern input table

4.6.1 Introduction to the row pattern input table

The row pattern input table is the input argument to MATCH_RECOGNIZE. In the examples above, the row pattern input table was Ticker, which is a table or view, or perhaps a named query (defined in a WITH clause). The row pattern input table can also be a derived table (also known as in-line view). For example:

```
FROM ( SELECT S.Name, T.Tradeday, T.Price
        FROM Ticker T, SymbolNames S
        WHERE T.Symbol = S.Symbol )
MATCH_RECOGNIZE ( ... ) AS M
```

The row pattern input table shall not be a <joined table>. The work-around is to use a derived table, such as:

```
FROM ( SELECT * FROM A LEFT OUTER JOIN B ON (A.X = B.Y) )
MATCH_RECOGNIZE ( ... ) AS M
```

Note that column names in the row pattern input table are unambiguous, since it is impossible to use range variables within the MATCH_RECOGNIZE clause to disambiguate. If the row pattern input table is a base table or a view, this is not a problem, since SQL does not allow ambiguous column names in a base table or view. This is only an issue when the row pattern input table is a derived table.

For example, consider a join of two tables, Emp and Dept, each of which has a column called Name. The following is a syntax error:

```
FROM ( SELECT D.Name, E.Name, E.Empno, E.Salary
        FROM Dept D, Emp E
        WHERE D.Deptno = E.Deptno )
MATCH_RECOGNIZE (
    PARTITION BY D.Name
    ... )
```

The preceding example is an error because the range variable D is not visible within the MATCH_RECOGNIZE (the scope of D is just the derived table). Rewriting like this is no help:

```
FROM ( SELECT D.Name, E.Name, E.Empno, E.Salary
        FROM Dept D, Emp E
        WHERE D.Deptno = E.Deptno )
MATCH_RECOGNIZE (
```

```
PARTITION BY Name
... )
```

This rewrite eliminates the use of the range variable D within the MATCH_RECOGNIZE. However, now the error is that Name is ambiguous, because there are two columns of the derived table called Name. The way to handle this is to disambiguate the column names within the derived table itself, like this:

```
FROM ( SELECT D.Name AS DName, E.Name AS EName,
            E.Empno, E.Salary
        FROM Dept D, Emp E
        WHERE D.Deptno = E.Deptno )
MATCH_RECOGNIZE (
  PARTITION BY DName
  ... )
```

4.6.2 The row pattern input name

Optionally, a correlation name for the row pattern input table may be declared, as in this example (equivalent to the example in [Subclause 4.3, “Example of ONE ROW PER MATCH”](#)):

```
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 AS T
  MATCH_RECOGNIZE (
    PARTITION BY T.Symbol
    ORDER BY T.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)
    DEFINE /* A defaults to True, matches any row */
           B AS B.Price < PREV (B.Price),
           C AS C.Price > PREV (C.Price)
  ) AS M
```

The row pattern input name in this example is T, as defined by the syntax “Ticker AS T”. It is also possible to omit the noise word AS, like this: “Ticker T”.

Specifying the row pattern input name is optional. The examples in [Subclause 4.3, “Example of ONE ROW PER MATCH”](#), and [Subclause 4.4, “Example of ALL ROWS PER MATCH”](#), do not show an explicit row pattern input name.

When the row pattern input name is not specified, the following defaults apply:

- 1) If the row pattern input table is a base table, view, or query name (the name of a query defined in a WITH clause), then the table name, view name or query name is the default row pattern input name.
- 2) Otherwise, an implementation-dependent row pattern input name, different from any other range variable in the query, is implicit. In practice, this means that the row pattern input name is unknowable and cannot be referenced elsewhere in the query.

4.6 The row pattern input table

The scope of the row pattern input name is the PARTITION BY and ORDER BY clauses of the MATCH_RECOGNIZE clause. This means that the row pattern input name can be used in the following contexts:

- 1) To qualify column names in the PARTITION BY clause.
- 2) To qualify column names in the ORDER BY clause.

The example above illustrates both of these uses.

The row pattern input name cannot be referenced in the MEASURES or DEFINE clauses, nor elsewhere in the query, such as the WHERE clause or the SELECT list.

4.6.3 The row pattern input declared column list

If an explicit row pattern input name is specified, it may be followed by a parenthesized list of column names, as in this example:

```
SELECT M.Sym,          /* 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 AS T (Sym, Td, Pr)
  MATCH_RECOGNIZE (
    PARTITION BY T.Sym
    ORDER BY T.Td
    MEASURES MATCH_NUMBER() AS Matchno,
              A.Pr AS Startp,
              LAST (B.Pr) AS Bottomp,
              LAST (C.Pr) AS Endp,
              AVG (U.Pr) AS Avgp
    ONE ROW PER MATCH
    AFTER MATCH SKIP PAST LAST ROW
    PATTERN (A B+ C+)
    SUBSET U = (A, B, C)
    DEFINE /* A defaults to True, matches any row */
           B AS B.Pr < PREV (B.Pr),
           C AS C.Pr > PREV (C.Pr)
  ) AS M
```

The parenthesized list of column names (Sym, Td, Pr) is called the row pattern input declared column list. The row pattern input declared column list may be used to change the names of the columns of the row pattern input table. There shall be exactly the same number of column names in the list as there are columns in the row pattern input table. In this example, Symbol has been renamed to Sym, Tradeday has been renamed to Td, and Price has been renamed to Pr. Consequently, the columns cannot be referenced as Symbol, Tradeday, or Price within the MATCH_RECOGNIZE; instead, they are referenced by their new names, Sym, Td, and Pr. Note that this also changes the default names of the columns in the row pattern output table. Thus, in the SELECT list, the first item is M.Sym, because the input column names Symbol was renamed to Sym, which becomes the name of the corresponding output column.

4.7 MATCH_RECOGNIZE

MATCH_RECOGNIZE is the keyword that introduces the syntax for row pattern recognition in the FROM clause. Syntactically, MATCH_RECOGNIZE is a postfix operator following the row pattern input table. The MATCH_RECOGNIZE keyword is followed by a parenthesized list of syntactic components that collectively describe the row pattern recognition operation.

4.8 PARTITION BY

PARTITION BY is used to specify that the rows of the row pattern input table are to be partitioned by one or more columns. Note that the column names in the PARTITION BY may be unqualified, or they may be qualified by the row pattern input name. See the examples in [Subclause 4.6.2, “The row pattern input name”](#).

If there is no PARTITION BY, then all rows of the row pattern input table constitute a single row pattern partition.

4.9 ORDER BY

ORDER BY is used to specify the order of rows within a row pattern partition. The ORDER BY clause of a MATCH_RECOGNIZE is similar to the ORDER BY clause of a cursor. As with the PARTITION BY clause, column names may be unqualified, or they may be qualified by the row pattern input name. See the examples in [Subclause 4.6.2, “The row pattern input name”](#).

If the order of two rows in a row pattern partition is not determined by the ORDER BY, then the result of MATCH_RECOGNIZE is non-deterministic.

NOTE 2 — Syntactically, the row pattern output table is always regarded as non-deterministic because there is no way for the query engine to deduce at compile time whether the ordering is total. This means that MATCH_RECOGNIZE cannot be used in contexts that are required to be deterministic, such as check constraints and assertions. However, the query author can use ORDER BY to insure that the query is sufficiently deterministic for the author’s intended purpose.

4.10 Row pattern variables

Row pattern variables are range variables whose scope is limited to a MATCH_RECOGNIZE clause. As range variables, row pattern variables are used to qualify column references, in either the scalar expression of a row pattern measure column, or the Boolean condition of a DEFINE.

There are two kinds of row pattern variables:

- 1) Primary row pattern variables, which are declared in the PATTERN and defined by an associated Boolean condition specified in the DEFINE clause.
- 2) Union row pattern variables, which are declared in the SUBSET clause as a union of a list of primary row pattern variables. The primary row pattern variables are called components of the union row pattern variable.

A row pattern variable shall not be both a primary row pattern variable and a union row pattern variable. This means that a row pattern variable that is declared in PATTERN shall not also be declared on the left hand side of a SUBSET.

Informally, a match consists of a set of contiguous rows in a row pattern partition of the row pattern input table. (For a more formal treatment, see [Clause 7, “Pattern matching rules”](#).) Each row of the match is mapped to a primary row pattern variable. The mapping of rows to primary row pattern variables conforms to the regular expression in the PATTERN clause, and is further constrained to insure that all Boolean conditions in the DEFINE clause are true.

Thus rows are mapped to row pattern variables. Conversely, each row pattern variable *RPV* has a set of rows that are mapped to *RPV*. For example, given:

```
PATTERN (A+ (B+ | C+) D)
SUBSET S = (B, D)
```

Suppose that consecutive rows R_3 , R_4 , R_5 , R_6 , and R_7 are mapped as follows:

$R_3 \rightarrow A$

$R_4 \rightarrow A$

$R_5 \rightarrow B$

$R_6 \rightarrow B$

$R_7 \rightarrow D$

Then:

- the set of rows mapped to A is $\{ R_3, R_4 \}$,
- the set of rows mapped to B is $\{ R_5, R_6 \}$,
- the set of rows mapped to C is empty, and
- the set of rows mapped to D is $\{ R_7 \}$.

The set of rows mapped to a union row pattern variable *URPV* can be obtained as the set union of rows mapped to each component of *URPV*. In this example:

- the set of rows mapped to S is $\{ R_5, R_6 \} \cup \{ R_7 \} = \{ R_5, R_6, R_7 \}$.

There is always one implicit union row pattern variable, called the universal row pattern variable, defined as the union of all primary row pattern variables. Thus, every row of a match is mapped to the universal row pattern variable. The universal row pattern variable is used to implicitly qualify unqualified column reference within the MEASURES or DEFINE clauses. There is no syntax available to the user to denote the universal row pattern variable. The query writer may, of course, define an explicit union row pattern variable that is the union of all primary row pattern variables. (The example in [Subclause 4.3, “Example of ONE ROW PER MATCH”](#), illustrates this technique.)

4.11 MEASURES

The MEASURES clause defines row pattern measure columns, which are columns of the row pattern output table whose value is computed by evaluating an expression related to a particular match. Note that SQL/RPR extends the scalar expression syntax of [ISO/IEC 9075-2](#), and provides special semantics for evaluating scalar expressions in the context of a row pattern match. This is discussed in [Clause 5, “Expressions in MEASURES and DEFINE”](#).

NOTE 3 — The MEASURES clause in a window definition does not define columns; instead, it defines named expressions which are accessed using a variant of the window function syntax, called row pattern measure functions. “Row pattern

measure” is the generic term for row pattern measure columns and row pattern measure functions, whose values are computed using the same rules.

4.12 ONE ROW PER MATCH vs ALL ROWS PER MATCH

4.12.1 Introduction to use of ROWS PER MATCH

ONE ROW PER MATCH indicates that the result has one row for each match. Columns of this row are defined by the PARTITION and MEASURES clauses. This is the default.

ALL ROWS PER MATCH indicates that the result has one row for each row of each match. (It is possible to exclude some rows using the exclusion syntax { - - } in the PATTERN; see [Subclause 4.14.3, “Excluding portions of the pattern”](#).)

ALL ROWS PER MATCH has three suboptions:

- ALL ROWS PER MATCH SHOW EMPTY MATCHES
- ALL ROWS PER MATCH OMIT EMPTY MATCHES
- ALL ROWS PER MATCH WITH UNMATCHED ROWS

These options are explained in the following subsections.

4.12.2 Handling empty matches

Some patterns permit empty matches. For example:

PATTERN (A*)

can be matched by zero or more rows that are mapped to A.

An empty match does not map any rows to primary row pattern variables; nevertheless, an empty match has a starting row. For example, there can be an empty match at the first row of a row pattern partition, an empty match at the second row of a row pattern partition, etc. An empty match is assigned a sequential match number, based on the ordinal position of its starting row, the same as any other match.

When using ONE ROW PER MATCH, an empty match results in one row of the row pattern output table. The row pattern measures for an empty match are computed as follows:

- The value of MATCH_NUMBER() is the sequential match number of the empty match.
- Any COUNT is 0.
- Any other aggregate, row pattern navigation operation, or ordinary row pattern column reference is null.

For example, the example in [Subclause 4.3, “Example of ONE ROW PER MATCH”](#), can be modified to permit empty matches, as follows:

```
SELECT M.Symbol, /* ticker symbol */
       M.Matchno, /* sequential match number */
       M.Firstp, /* starting price */
       M.Lastp /* ending price */
FROM Ticker
```