INTERNATIONAL STANDARD

ISO/IEC 9075-16

First edition
2023-06

Information technology — Database languages SQL —

Part 16: Property Graph Queries (SQL/PGQ)

Reference number
ISO/IEC 9075-16:2023(E)

Documents
Document Sharing Hub
© ISO/IEC 2023

Full document details: ISO IEC 09075-16-2023

COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2023

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

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub

Full document details: ISO IEC 09075-16-2023

Contents

Page

Foreword . . . viii
Introduction . . . x
1 Scope . . . 1
2 Normative references . . . 2
3 Terms and definitions . . . 3
4 Concepts . . . 8
4.1 Notations and conventions . . . 8
4.1.1 Notations . . . 8
4.2 Columns, fields, and attributes . . . 8
4.3 SQL-statements . . . 8
4.3.1 SQL-statements classified by function . . . 8
4.3.1.1 SQL-schema statements . . . 8
4.4 Basic security model . . . 8
4.4.1 Privileges . . . 9
4.5 SQL-property graphs . . . 9
4.5.1 Introduction to SQL-property graphs . . . 9
4.5.2 Pure property graph . . . 9
4.5.3 Tabular property graph . . . 11
4.6 Operations involving property graphs . . . 12
4.7 Graph pattern matching . . . 13
4.7.1 Summary of graph pattern matching . . . 13
4.7.2 Paths . . . 13
4.7.3 Path patterns . . . 14
4.7.4 Graph pattern variables . . . 15
4.7.5 References to graph pattern variables . . . 16
4.7.6 Path pattern matching . . . 17
4.7.7 Path modes . . . 18
4.7.8 Selective path search prefixes . . . 19
4.7.9 Match modes . . . 19
5 Lexical elements . . . 20
5.1 <SQL terminal character> . . . 20
5.2 <token> and <separator> . . . 21
5.3 Names and identifiers . . . 24
6 Scalar expressions . . . 26
6.1 <value expression primary> . . . 26
6.2 <identifier chain> . . . 28
6.3 <set function specification> . . . 30
6.4 <case expression> . . . 32

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

6.5 <property reference>. 33
6.6 <element_id function>. 35
6.7 <graphical match number function>. 36
6.8 <graphical path name function>. 37
6.9 <graphical element number function>. 38
6.10 <graphical path length function>. 39

7 Query expressions. 40
7.1 <table reference>. 40
7.2 <query specification>. 50

8 Predicates. 51
8.1 <predicate>. 51
8.2 <directed predicate>. 52
8.3 <labeled predicate>. 53
8.4 <source/destination predicate>. 54
8.5 <all_different predicate>. 56
8.6 <same predicate>. 57
8.7 <bound predicate>. 58
8.8 <property_exists predicate>. 60

9 Additional common rules. 61
9.1 Potential sources of non-determinism. 61
9.2 Contextual inference of a set of labels. 62
9.3 Expansion of an <all properties reference>. 65
9.4 Satisfaction of a <label expression> by a defined label set. 67
9.5 Converting a tabular property graph to a pure property graph. 69
9.6 Machinery for graph pattern matching. 72
9.7 Evaluation of a <path pattern expression>. 77
9.8 Evaluation of a selective <path pattern>. 82
9.9 Applying bindings to evaluate an expression. 86
9.10 Applying bindings to evaluate a subexpression of an aggregate. 89
9.11 Applying bindings to generate a row. 91
9.12 Creation of an element table descriptor. 94
9.13 Creation of a vertex table descriptor. 98
9.14 Creation of an edge table descriptor. 99
9.15 Consistency check of a tabular property graph descriptor. 102
9.16 Deriving a pure property graph descriptor from a tabular property graph descriptor. 104

10 Additional common elements. 106
10.1 <aggregate function>. 106
10.2 <sort specification list>. 108
10.3 <graph reference>. 109
10.4 <graph pattern>. 110
10.5 <path pattern prefix>. 116
10.6 <path pattern expression>. 120
10.7 <graph pattern quantifier>. 129
10.8 <label expression>. 131
10.9 <simplified path pattern expression>. 133
10.10 <element reference>. 138

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

10.11 <path reference>. 140
11 Schema definition and manipulation. 141
11.1 <schema definition>. 141
11.2 <drop schema statement>. 142
11.3 <table definition>. 143
11.4 <drop column definition>. 144
11.5 <drop table statement>. 145
11.6 <view definition>. 146
11.7 <drop view statement>. 147
11.8 <drop domain statement>. 148
11.9 <drop character set statement>. 149
11.10 <drop collation statement>. 150
11.11 <drop transliteration statement>. 151
11.12 <drop attribute definition>. 152
11.13 <drop data type statement>. 153
11.14 <alter routine statement>. 154
11.15 <drop routine statement>. 155
11.16 <drop user-defined cast statement>. 156
11.17 <drop user-defined ordering statement>. 157
11.18 <drop sequence generator statement>. 158
11.19 <property graph definition>. 159
11.20 <alter property graph statement>. 164
11.21 <add element table definition>. 165
11.22 <drop element table definition>. 167
11.23 <alter element table definition>. 170
11.24 <add element table label clause>. 171
11.25 <drop element table label clause>. 174
11.26 <alter element table label properties>. 176
11.27 <drop property graph statement>. 180
12 Access control. 182
12.1 <grant statement>. 182
12.2 <privileges>. 183
12.3 <revoke statement>. 184
13 SQL-client modules. 187
13.1 <externally-invoked procedure>. 187
13.2 <SQL procedure statement>. 188
14 Diagnostics management. 189
14.1 <get diagnostics statement>. 189
15 Information Schema. 191
15.1 Information Schema digital artifact. 191
15.2 PG_DEFINED_LABEL_SETS view. 191
15.3 PG_DEFINED_LABEL_SET_LABELS view. 192
15.4 PG_EDGE_DEFINED_LABEL_SETS view. 193
15.5 PG_EDGE_TABLE_COMPONENTS view. 194
15.6 PG_EDGE_TRIPLETS view. 195
15.7 PG_ELEMENT_TABLE_KEY_COLUMNS view. 196

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

15.8 PG_ELEMENT_TABLE_LABELS view. . . . 197
15.9 PG_ELEMENT_TABLE_PROPERTIES view. . . . 198
15.10 PG_ELEMENT_TABLES view. . . . 199
15.11 PG_LABEL_PROPERTIES view. . . . 200
15.12 PG_LABELS view. . . . 201
15.13 PG_PROPERTY_DATA_TYPES view. . . . 202
15.14 PG_PROPERTY_GRAPH_PRIVILEGES view. . . . 203
15.15 PG_VERTEX_DEFINED_LABEL_SETS view. . . . 204
15.16 PROPERTY_GRAPHS view. . . . 205

16 Definition Schema. . . . 206
16.1 Definition Schema digital artifact. . . . 206
16.2 DATA_TYPE_DESCRIPTOR base table. . . . 206
16.3 TABLES base table. . . . 207
16.4 PG_DEFINED_LABEL_SETS base table. . . . 208
16.5 PG_DEFINED_LABEL_SET_LABELS base table. . . . 209
16.6 PG_EDGE_DEFINED_LABEL_SETS base table. . . . 210
16.7 PG_EDGE_TABLE_COMPONENTS base table. . . . 212
16.8 PG_EDGE_TRIPLETS base table. . . . 214
16.9 PG_ELEMENT_TABLE_KEY_COLUMNS base table. . . . 216
16.10 PG_ELEMENT_TABLE_LABELS base table. . . . 217
16.11 PG_ELEMENT_TABLE_PROPERTIES base table. . . . 218
16.12 PG_ELEMENT_TABLES base table. . . . 219
16.13 PG_LABEL_PROPERTIES base table. . . . 221
16.14 PG_LABELS base table. . . . 222
16.15 PG_PROPERTY_DATA_TYPES base table. . . . 223
16.16 PG_PROPERTY_GRAPH_PRIVILEGES base table. . . . 225
16.17 PG_VERTEX_DEFINED_LABEL_SETS base table. . . . 227
16.18 PROPERTY_GRAPHS base table. . . . 228

17 Status codes. . . . 229
17.1 SQLSTATE. . . . 229

18 Conformance. . . . 230
18.1 Claims of conformance to SQL/PGQ. . . . 230
18.2 Additional conformance requirements for SQL/PGQ. . . . 230
18.3 Implied feature relationships of SQL/PGQ. . . . 230

Annex A (informative) **SQL conformance summary. . . . 234**

Annex B (informative) **Implementation-defined elements. . . . 249**

Annex C (informative) **Implementation-dependent elements. . . . 251**

Annex D (informative) **SQL optional feature taxonomy. . . . 254**

Annex E (informative) **Deprecated features. . . . 259**

Annex F (informative) **Incompatibilities with ISO/IEC 9075:2016. . . . 260**

Annex G (informative) **Defect Reports not addressed in this edition of this document. . . . 261**

Index. . . . 262

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

Tables

Table

Page

1 Conversion of simplified syntax delimiters to default edge delimiters. 135
2 Data types of <condition information item name>s. 189
3 SQL-statement codes. 190
4 SQLSTATE class and subclass codes. 229
5 Implied feature relationships of SQL/PGQ. 230
A.1 Feature definitions outside of Conformance Rules. 234
D.1 Feature taxonomy for optional features. 254

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub

Full document details: ISO IEC 09075-16-2023

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).

ISO and IEC draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO and IEC take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO and IEC have not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents and https://patents.iec.ch. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

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 Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange*.

This first edition of ISO/IEC 9075-16 is designed to be used in conjunction with the following editions of other parts of the ISO/IEC 9075 series, all published in 2023:

— ISO/IEC 9075-1, sixth edition;

— ISO/IEC 9075-2, sixth edition;

— ISO/IEC 9075-3, sixth edition;

— ISO/IEC 9075-4, seventh edition;

— ISO/IEC 9075-9, fifth edition;

— ISO/IEC 9075-10, fifth edition;

— ISO/IEC 9075-11, fifth edition;

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

— ISO/IEC 9075-13, fifth edition;

— ISO/IEC 9075-14, sixth edition;

— ISO/IEC 9075-15, second edition.

A list of all parts in the ISO/IEC 9075 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.

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub

Full document details: ISO IEC 09075-16-2023

Introduction

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, "Concepts", presents concepts used in the definition of property graph queries.

5) Clause 5, "Lexical elements", defines a number of lexical elements used in the definition of property graph queries.

6) Clause 6, "Scalar expressions", defines a number of scalar expressions used in the definition of property graph queries.

7) Clause 7, "Query expressions", defines the elements of the language that produce rows and tables of data as used in property graph queries.

8) Clause 8, "Predicates", defines the predicates of the language.

9) Clause 9, "Additional common rules", specifies additional rules for implicit or explicit invocation from other places in this document.

10) Clause 10, "Additional common elements", defines additional common elements used in the definition of property graph queries.

11) Clause 11, "Schema definition and manipulation", defines the schema definition and manipulation statements associated with the definition of property graph queries.

12) Clause 12, "Access control", defines facilities for controlling access to SQL-data.

13) Clause 13, "SQL-client modules", defines the facilities for using property graph queries.

14) Clause 14, "Diagnostics management", defines the diagnostics management facilities.

15) Clause 15, "Information Schema", defines viewed tables that contain schema information.

16) Clause 16, "Definition Schema", defines base tables on which the viewed tables containing schema information depend.

17) Clause 17, "Status codes", defines SQLSTATE values related to property graph queries.

18) Clause 18, "Conformance", defines the criteria for conformance to this document.

19) Annex A, "SQL conformance summary", is an informative Annex. It summarizes the conformance requirements of the SQL language.

20) Annex B, "Implementation-defined elements", is an informative Annex. It lists those features for which the body of this document states that the syntax, the meaning, the returned results, the effect on SQL-data and/or schemas, or other aspect is partly or wholly implementation-defined.

21) Annex C, "Implementation-dependent elements", is an informative Annex. It lists those features for which the body of this document states that the syntax, the meaning, the returned results, the effect on SQL-data and/or schemas, or other aspect is partly or wholly implementation-dependent.

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

22) Annex D, "SQL optional feature taxonomy", is an informative Annex. It identifies features of the SQL language specified in this document by an identifier and a short descriptive name. This taxonomy is used to specify conformance.

23) Annex E, "Deprecated features", is an informative Annex. It lists features that the responsible Technical Committee intends not to include in a future edition of this document.

24) Annex F, "Incompatibilities with ISO/IEC 9075:2016", is an informative Annex. It lists incompatibilities with the previous edition of this document.

25) Annex G, "Defect Reports not addressed in this edition of this document", is an informative Annex. It describes the Defect Reports that were known at the time of publication of this document. Each of these problems is a problem carried forward from the previous edition of the ISO/IEC 9075 series. No new problems have been created in the drafting of this edition of this document.

In the text of this document, in Clause 5, "Lexical elements", through Clause 18, "Conformance", Subclauses begin new pages. Any resulting blank space is not significant.

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub

Full document details: ISO IEC 09075-16-2023

Documents
Document Sharing Hub

Full document details: ISO IEC 09075-16-2023

Information technology — Database language SQL —

Part 16:
Property Graph Queries (SQL/PGQ)

1 Scope

This document defines ways for the SQL language to represent property graphs and to interact with them.

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

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)*

ISO/IEC 9075-4, *Information technology — Database languages — SQL — Part 4: Persistent Stored Modules (SQL/PSM)*

ISO/IEC 9075-11, *Information technology — Database languages — SQL — Part 11: Information and Definition Schemas (SQL/Schemata)*

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 9075-1, ISO/IEC 9075-2, and the following apply.

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

— ISO Online browsing platform: available at `https://www.iso.org/obp`

— IEC Electropedia: available at `https://www.electropedia.org/`

3.1
destination vertex
vertex (3.37) that is distinguished as the destination of a *directed edge* (3.4)

3.2
destination vertex table
table that contains data representing the *destination vertices* (3.1) of a *directed edge* (3.4)

3.3
defined label set
<pure property graph> distinguished set of labels

Note 1 to entry: Subclause 4.5.2, "Pure property graph", specifies six kinds of defined label sets: vertex defined label set, edge defined label set, edge triplet defined label set, source vertex defined label set, destination vertex defined label set, and endpoint vertex defined label set.

3.4
directed edge
edge (3.6) that distinguishes one of its *endpoints* (3.14) as its *source vertex* (3.30) and one of its endpoints as its *destination vertex* (3.1)

Note 1 to entry: A directed edge expresses a relationship that is asymmetric.

Note 2 to entry: The antonym is *undirected edge* (3.35).

3.5
directed graph
graph in which every *edge* (3.6) is directed

Note 1 to entry: The antonym is *undirected graph* (3.36).

3.6
edge
relationship
connection between two *vertices* (3.37)

Note 1 to entry: Both terms, "edge" and "relationship", are used in the real world to denote the same concept. Without judgement or prejudice, this document uses only the term "edge". In BNF productions, wherever the keyword EDGE is allowed, the keyword RELATIONSHIP can equally well be used instead.

3.7
edge pattern
path pattern (3.24) that matches a single *edge* (3.6)

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

3.8
edge table
table that contains data representing *edges* (3.6)

3.9
edge table alias
identifier that uniquely identifies an *edge table* (3.8) within an *SQL-property graph* (3.32)

3.10
edge variable
element variable (3.13) that is declared in an *edge pattern* (3.7) or as the second *iterator variable* (3.18) in <one row per step>

Note 1 to entry: An edge variable may be bound to a list of *edges* (3.6).

3.11
element pattern
edge pattern (3.7) or *vertex pattern* (3.38)

3.12
element table
edge table (3.8) or *vertex table* (3.39)

3.13
element variable
variable that may be bound to a list of *graph elements* (3.15)

Note 1 to entry: An element variable is either a *primary variable* (3.27) or an *iterator variable* (3.18).

Note 2 to entry: An element variable is either a *vertex variable* (3.41) or an *edge variable* (3.10).

3.14
endpoint
one of the two *vertices* (3.37) connected by an *edge* (3.6)

Note 1 to entry: Both endpoints of an edge may be the same vertex.

3.15
graph element
vertex (3.37) or *edge* (3.6)

3.16
graph pattern
set of one or more *path patterns* (3.24)

3.17
graph pattern variable
path variable (3.25), *subpath variable* (3.34), or *element variable* (3.13)

3.18
iterator variable
element variable (3.13) that is declared in <one row per iteration>

3.19
label
identifier associated with a *vertex* (3.37) or *edge* (3.6)

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

3.20
label expression
expression composed from *label* (3.19) names using disjunction, conjunction, and negation

Note 1 to entry: Disjunction, conjunction, and negation are denoted respectively by a vertical bar "|", ampersand "&" and exclamation mark "!", with parentheses for grouping. See Subclause 10.8, "<label expression>" for more details.

3.21
mixed graph
graph that allows both directed and undirected *edges* (3.6)

3.22
multigraph
graph that allows more than one *edge* (3.6) connecting two *vertices* (3.37)

3.23
path
sequence of an odd number of *graph elements* (3.15)

Note 1 to entry: A path always starts and ends with a *vertex* (3.37) and alternates between vertices and *edges* (3.6), such that each edge resides in the path between its *endpoints* (3.14). See Subclause 4.7.2, "Paths", for a fuller discussion of paths.

Note 2 to entry: A path may consist of a single vertex.

Note 3 to entry: A vertex or an edge may be contained multiple times in a path, including via self-loops.

3.24
path pattern
pattern (3.26) that matches a *path* (3.23)

3.25
path variable
graph pattern variable (3.17) that may be bound to a path binding that is matched by a *path pattern* (3.24)

Note 1 to entry: The extracted path of the path binding is a *path* (3.23).

Note 2 to entry: Extracted path and path binding are defined in Subclause 9.6, "Machinery for graph pattern matching".

3.26
pattern
syntax for specifying requirements on the bindings of variables

3.27
primary variable
element variable (3.13) that is declared in an *element pattern* (3.11)

3.28
property
<edge or vertex> (property name, data type) pair

Note 1 to entry: The value of a property is a value of its data type.

3.29
property graph
data structure consisting of zero or more *vertices* (3.37) and zero or more *edges* (3.6)

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

3.30
source vertex
vertex (3.37) that is distinguished as the source of a *directed edge* (3.4)

3.31
source vertex table
table that contains data representing the *source vertices* (3.30) of a *directed edge* (3.4)

3.32
SQL-property graph
SQL-schema object that represents *vertices* (3.37) and *edges* (3.6) using tables

3.33
subpath
path (3.23) fully contained in another path

Note 1 to entry: A subpath may be identical to its containing path.

3.34
subpath variable
graph pattern variable (3.17) that is declared at the head of a <parenthesized path pattern expression>

3.35
undirected edge
edge (3.6) that does not distinguish between its *endpoints* (3.14)

Note 1 to entry: An undirected edge expresses a relationship that is necessarily symmetric.

Note 2 to entry: The antonym is *directed edge* (3.4).

3.36
undirected graph
graph in which every *edge* (3.6) is undirected

Note 1 to entry: The antonym is *directed graph* (3.5).

3.37
vertex
node
fundamental unit of which a *property graph* (3.29) is formed

Note 1 to entry: plural: vertices or nodes.

Note 2 to entry: Both terms, “vertex” and “node”, are used in the real world to denote the same concept. Without judgement or prejudice, this document uses only the term “vertex”. In BNF productions, wherever the keyword VERTEX is allowed, the keyword NODE can equally well be used instead.

3.38
vertex pattern
path pattern (3.24) that matches a single *vertex* (3.37)

3.39
vertex table
table that contains data representing *vertices* (3.37)

3.40
vertex table alias
identifier that uniquely identifies a *vertex table* (3.39) within an *SQL-property graph* (3.32)

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

3.41
vertex variable
element variable (3.13) that is declared in a *vertex pattern* (3.38) or as an *iterator variable* (3.18) in <one row per vertex> or as the first or third iterator variable in <one row per step>

Note 1 to entry: A vertex variable may be bound to a list of *vertices* (3.37).

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

4 Concepts

This Clause modifies Clause 4, "Concepts", in ISO/IEC 9075-2.

4.1 Notations and conventions

This Subclause modifies Subclause 4.1, "Notations and conventions", in ISO/IEC 9075-2.

4.1.1 Notations

This Subclause modifies Subclause 4.1.1, "Notations", in ISO/IEC 9075-2.

The notations used in this document are defined in ISO/IEC 9075-1.

The syntax defined in this document is available from the ISO website as a "digital artifact". The syntax defined in this document is available from the ISO website as a "digital artifact". See `https://standards.iso.org/iso-iec/9075/-16/ed-1/en/` to download digital artifacts for this document. To download the syntax defined in a plain-text format, select the file named `ISO_IEC_9075-16(E)_PGQ.bnf.txt`. To download the syntax defined in an XML format, select the file named `ISO_IEC_9075-16(E)_PGQ.bnf.xml`.

4.2 Columns, fields, and attributes

This Subclause modifies Subclause 4.15, "Columns, fields, and attributes", in ISO/IEC 9075-2.

Insert after the 5th paragraph: The nullability characteristic of a column of <graph table> is defined by the Syntax Rules of Subclause 7.1, "<table reference>".

4.3 SQL-statements

This Subclause modifies Subclause 4.41, "SQL-statements", in ISO/IEC 9075-2.

4.3.1 SQL-statements classified by function

This Subclause modifies Subclause 4.41.2, "SQL-statements classified by function", in ISO/IEC 9075-2.

4.3.1.1 SQL-schema statements

This Subclause modifies Subclause 4.41.2.1, "SQL-schema statements", in ISO/IEC 9075-2.

Insert into the 1st paragraph, after the last list item:

— <property graph definition>.

— <alter property graph statement>.

— <drop property graph statement>.

4.4 Basic security model

This Subclause modifies Subclause 4.42, "Basic security model", in ISO/IEC 9075-2.

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

4.4.1 Privileges

This Subclause modifies Subclause 4.42.2, "Privileges", in ISO/IEC 9075-2.

Insert into the 1st paragraph, after the last list item:

— An SQL-property graph.

Insert after the 8th paragraph: A privilege descriptor with an object that is an SQL-property graph *PG* and an action of SELECT is called a *property graph privilege descriptor* and identifies the existence of a privilege on *PG*.

4.5 SQL-property graphs

4.5.1 Introduction to SQL-property graphs

An SQL-property graph is described by an SQL-property graph descriptor. This descriptor includes the following.

— The schema-qualified name of the SQL-property graph.

— A pure property graph descriptor describing the pure property graph.

— A tabular property graph descriptor describing the tabular property graph.

The data making up an SQL-property graph is the data exposed through the vertex and edge tables that are part of the tabular property graph.

4.5.2 Pure property graph

A pure property graph is a mixed multigraph and consists of the following.

— A set of vertices. Each vertex consists of the following.

- An identifier that is unique within the pure property graph.

  NOTE 1 — The value of a vertex identifier is implementation-dependent (UV021) and is not required to be accessible. This unique identifier is used for definitional purposes to establish the identity of the vertex.

- A set of zero or more labels (also known as a *vertex defined label set*) that determines the properties of the vertex. A label has a name (i.e., an <identifier>).

- Zero or more properties. For each property:

  — its name;

  — its declared type, which can be any SQL data type,

  — its value.

— A set of edges. Each edge consists of the following.

- An identifier that is unique within the pure property graph.

  NOTE 2 — The value of an edge identifier is implementation-dependent (UV022) and is not required to be accessible. This unique identifier is used for definitional purposes to establish the identity of the edge.

- A set of zero or more labels (also known as an *edge defined label set*) that determines the properties of the edge. A label has a name (i.e., an <identifier>).

- Zero or more properties. For each property:

  — its name;

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

— its declared type, which can be any SQL data type;

— its value.

- An indication, whether the edge is directed or undirected.

  Case:

  — If the edge is directed, then an ordered pair of vertices, as follows.

    - A source vertex.
    - A destination vertex.

      NOTE 3 — The direction of an edge is from its source vertex to its destination vertex.

      NOTE 4 — Both the destination vertex and the source vertex of an edge are part of the same pure property graph as the edge.

      NOTE 5 — The source vertex and the destination vertex can be the same vertex.

  — If the edge is undirected, then an unordered pair of vertices, the endpoints of the undirected edge.

    NOTE 6 — The pure property graph model supports undirected edges, while the tabular graph model does not. An SQL-implementation can provide implementation-defined means to populate an undirected or mixed graph. This document specifies query syntax (<edge pattern> in Subclause 10.6, "<path pattern expression>") and semantics (the General Rules regarding <edge pattern> in Subclause 9.7, "Evaluation of a <path pattern expression>") to query undirected edges, if an SQL-implementation supports undirected edges.

    NOTE 7 — The two endpoints of an undirected edge can be the same vertex.

    NOTE 8 — The case of a directed edge whose source and destination vertex are the same is distinguished from an undirected edge whose endpoints are the same.

A pure property graph is described by a *pure property graph descriptor*. This descriptor includes the following.

— A set of zero or more labels. Each label consists of the following.

- An <identifier> that is unique within the pure property graph descriptor.

  NOTE 9 — The <identifier> of a label is also known as the name of the label.

- A set of zero or more properties. Each property consists of the following.

  — Its name, which is an <identifier> that is unique within the label.

  — Its declared type.

— A possibly empty set of vertex defined label sets.

— A possibly empty set of edge defined label sets.

- For each edge defined label set, an indication whether edges that have that edge defined label set are always directed, always undirected, or possibly directed or undirected.

— A possibly empty set of *edge triplets*. An edge triplet consists of the following.

- An *edge triplet defined label set*, which is a subset of an edge defined label set.
- An indication whether the edge triplet describes directed edges or undirected edges.

  Case:

  — If the edge triplet describes directed edges, then:

    - a *source vertex defined label set*, which is a subset of a vertex defined label set;

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

- a *destination defined vertex label set*, which is a subset of a vertex defined label set.

— If the edge triplet describes undirected edges, then:

- a pair of *endpoint vertex defined label sets*, each of which is a subset of a vertex defined label set.

4.5.3 Tabular property graph

A *tabular property graph* comprises the following.

— Zero or more vertex tables.

— Zero or more edge tables.

A tabular property graph *TPG* is described by a tabular property graph descriptor. This descriptor includes the following.

— The descriptors of zero or more vertex tables. Each vertex table descriptor includes the following.

- If the vertex table is based on a persistent base table or viewed table, then the schema-qualified name of that table *V*; otherwise, a <query expression> defining the vertex table.
- The vertex table alias that identifies *V* as a vertex table within the SQL-property graph that contains *TPG*.
- A list of columns that uniquely identify a row in *V*.
- One or more labels. Each label consists of the following.

  — An <identifier>.

  — A set of zero or more properties. Each property consists of the following.

  - A name, which is an <identifier> that is unique within the label.
  - A declared type.
  - A <value expression> associated with the property.

  NOTE 10 — While a pure property graph allows a vertex to have zero labels, in a tabular property graph a vertex has at least one label.

— The descriptors of zero or more edge tables. Each edge table descriptor includes the following.

- If the edge table is based on a persistent base table or viewed table, then the schema-qualified name of that table *E*; otherwise, a <query expression> defining the edge table.
- The edge table alias that identifies *E* as an edge table within the SQL-property graph that contains *TPG*.
- A list of columns that uniquely identify a row in *E*.
- One or more labels. Each label consists of the following.

  — An <identifier>.

  — A set of zero or more properties. Each property consists of the following.

  - A name, which is an <identifier> that is unique within the label.
  - A declared type.
  - A <value expression> associated with the property.

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

NOTE 11 — While a pure property graph allows an edge to have zero labels, in a tabular property graph an edge has at least one label.

- The name of the *source vertex table*.
- The *edge source key*: one or more columns (of the edge table) referencing one or more rows in the source vertex table.
- The *source vertex key*: one or more columns (of the source vertex table) that correspond to the edge source key
- The name of the *destination vertex table*.
- The *edge destination key*: one or more columns (of the edge table) referencing one or more rows in the destination vertex table.
- The *destination vertex key*: one or more columns (of the destination vertex table) that correspond to the edge destination key.

4.6 Operations involving property graphs

<graph table> is a kind of <table primary> that applies a <graph pattern> to a property graph and formats the results as a table.

<directed predicate> is a predicate to test whether an edge variable is bound to a directed edge.

NOTE 12 — <directed predicate> is specified for use in mixed graphs; however, this document does not provide a means to create a mixed graph.

<labeled predicate> is a predicate to test whether the graph element bound to an element variable satisfies a <label expression>.

<source/destination predicate> is a pair of predicates, to test whether a vertex variable is bound to the source or destination of the edge that is bound to an edge variable.

<all_different predicate> is a predicate to test whether a list of element variables are bound to pairwise distinct graph elements.

<same predicate> is a predicate to test whether a list of element variables are bound to the same graph element.

<bound predicate> is a predicate to test whether an element variable is bound to at least one graph element. Optional syntax of <bound predicate> tests whether an iterator variable is bound to the same position in a path binding as a primary variable.

<property_exists predicate> is a predicate to determine if the graph element bound to a singleton element variable has a particular property.

<element_id function> is a function to obtain a value that encapsulates the identity of a graph element. The declared type of <element_id function> is an implementation-defined (IV120) type that is suitable for equality operations and grouping operations. The value returned by <element_id function> encapsulates the identity of the graph element for at least the duration of the innermost executing SQL-statement.

<graphical match number function> is a function to obtain a unique implementation-dependent (UV016) number identifying the current match being returned by <graph table>.

<graphical path name function> is a function to obtain the name of the path variable of the path that the iterator variable(s) are currently iterating over.

<graphical element number function> is a function to obtain the 1-relative sequential number of the graph element in a path binding that an iterator variable is positioned on.

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

A *within-match aggregate* is a <set function specification> contained in a <graph table> without an intervening <query expression>. A within-match aggregate computes a value from the group of graph elements referenced by an element reference whose degree of reference is effectively bounded group.

4.7 Graph pattern matching

4.7.1 Summary of graph pattern matching

Graph pattern matching is the process of applying a list of special-purpose regular expressions called a <graph pattern> on a pure property graph *PG* in order to return a set of reduced matches. Each reduced match is a list of path bindings; each path binding is a function that maps the symbols in a word of a regular language to a path of *PG*. The regular language and related concepts such as path binding and reduced match are specified in Subclause 9.6, "Machinery for graph pattern matching", and Subclause 10.4, "<graph pattern>".

A <graph pattern> is a list of <path pattern>s. Each <path pattern> in the list is matched to *PG* to detect a possibly-empty set of path bindings in *PG* that correspond to the <path pattern>.

The cross-product of these sets is reduced by natural joins over those global singleton element variables that are exposed by each <path pattern> and that are bound to the same graph element in *PG*. The remaining tuples are called reduced matches; the set of these reduced matches may be empty, but cannot be infinite because of syntactic restrictions to guard against infinite cycling.

The behavior of the graph pattern matching is defined in this document.

NOTE 13 — A more detailed summary can be found in Subclause 4.7.6, "Path pattern matching".

Additional qualifying parameters (predicates, a selective <path search prefix>, a <path mode>, and the <different edges match mode>) that restrict the result may be supplied.

If *PG* contains cycles, then a match to a <path pattern> having an unbounded quantifier might return an infinite set of paths. However, this possibility is prevented by Syntax Rules that require the use of selective <path search prefix>s, restrictive <path mode>s, or the <different edges match mode> (or any combination thereof) to prevent infinitely-sized result sets.

4.7.2 Paths

A *path P* is a sequence of *n* graph elements of a property graph *PG*, such that:

— *n* is an odd number;

— *n* = 1 (one), then *P* has no edges;

— the graph element at each odd index is a vertex and the graph element at each even index is an edge that connects the pair of vertices immediately before and after the edge in the sequence.

A path contains a non-empty sequence of vertices and a (possibly empty) sequence of edges. If there are two or more vertices, then the path is a sequence of graph elements that starts with a vertex and is followed by a sequence of ordered (edge, vertex) pairs.

If *PG* is a multigraph, then an edge in the path between a pair of vertices is one of possibly several edges between those vertices in *PG*.

Every edge *E* in the path has an orientation. If *E* is undirected, then the orientation of *E* is *undirected*. Let *V1* be the vertex that immediately precedes *E* in the path, and let *V2* be the vertex that immediately follows *E*. If the source of *E* is *V1* and the destination of *E* is *V2*, then the orientation of *E* is *left to right*. It is also said that the orientation of *E* is *directed pointing right*. If the source of *E* is *V2* and the destination of *E* is *V1*, then the orientation of *E* is *right to left*. It is also said that the orientation of *E* is *directed pointing left*.

NOTE 14 — A directed self-edge (i.e., when *E* is directed and *V1* and *V2* are the same vertex), is oriented both left to right and right to left.

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

If no edge from *PG* appears more than once in a path, then the path is called a *trail*. If no vertex from *PG* appears more than once in a path, except possibly as the first and last vertices of the path, then the path is called *simple*. If no vertex from *PG* appears more than once in a path, then the path is called *acyclic*.

NOTE 15 — The term "path" is used in more than one way in mathematical graph theory and in informal technical presentations and discussions. In this document, the term path always denotes what a graph-theoretic work could call a partially-oriented walk in a pure property graph. Such a graph is a mixed multigraph: edges can be directed or undirected, and there can be multiple edges between two vertices.

4.7.3 Path patterns

A <path pattern> is an expression built from the following syntactic elements, governed by the Format and Syntax Rules of Subclause 10.4, "<graph pattern>", and other Subclauses.

— An optional <path variable declaration>, to declare a path variable to be bound to a path binding.

— An optional <path pattern prefix>.

NOTE 16 — <path pattern prefix> is described in Subclause 4.7.7, "Path modes", and Subclause 4.7.8, "Selective path search prefixes", and specified in Subclause 10.5, "<path pattern prefix>".

— A mandatory <path pattern expression>.

A <path pattern expression> is an expression built recursively from <element pattern>s, governed by the Format and Syntax Rules of Subclause 10.6, "<path pattern expression>", and other Subclauses.

An <element pattern> is a pattern to match a single graph element. There are two kinds of <element pattern>s.

— <vertex pattern>:

A <vertex pattern> is a pattern to match a single vertex. A <vertex pattern> comprises at a minimum a matching pair of parentheses, which may contain optional <element pattern filler>, described subsequently.

— <edge pattern>:

An <edge pattern> is a pattern to match a single edge. An <edge pattern> is either a <full edge pattern> (which optionally permits <element pattern filler>) or an <abbreviated edge pattern> (which does not support <element pattern filler>). These two major classes of <edge pattern> have seven variants each, for the seven possible non-empty combinations of the three edge orientations (the individual edge orientations being directed pointing left, undirected, or directed pointing right). Thus there are fourteen varieties of <edge pattern>.

<element pattern filler> provides three optional components of <vertex pattern>s and <full edge pattern>s.

— <element variable declaration>, to declare an element variable to be bound to a graph element by the <element pattern>.

— <label expression>, a predicate regarding the labels of the graph element that is bound by the <element pattern>. A <label expression> is an expression formed from <label name>s and the <wildcard label> "%", using the operation signs <vertical bar> "|" for disjunction, <ampersand> "&" for conjunction, <exclamation mark> "!" for negation, and balanced pairs of parentheses for grouping.

— <element pattern where clause>, a <search condition> to be satisfied by the graph element that is bound by the <element pattern>.

<path pattern expression>s are regular expressions built recursively from <element pattern>s using the following operations, governed by the Format and Syntax Rules of Subclause 10.6, "<path pattern expression>", and other Subclauses.

— concatenation, indicated syntactically by string concatenation (i.e., no operation sign).

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

NOTE 17 — <element pattern>s and more complex <path pattern expression>s can be concatenated in ways that appear to juxtapose either two <vertex pattern>s or two <edge pattern>s. These topologically inconsistent patterns are understood during pattern matching as follows.

- Two consecutive <vertex pattern>s bind to the same vertex.
- Two consecutive <edge pattern>s conceptually have an implicit <vertex pattern> between them.

— Grouping, using matching pairs of parentheses to form a <parenthesized path pattern expression>. A <parenthesized path pattern expression> may optionally contain the following.

- A <subpath variable declaration> to declare a subpath variable.
- A <search condition> to constrain matches.

— Alternation, indicated by <vertical bar> or <multiset alternation operator>.

NOTE 18 — Alternation with <vertical bar> provides set semantics using deduplication of redundant equivalent reduced matches, whereas alternation with <multiset alternation operator> provides multiset semantics, with no deduplication.

— Quantification, indicated by a postfix <graph pattern quantifier>, which may be affixed to an <edge pattern> or a <parenthesized path pattern expression>.

— <questioned path primary>, indicated by a postfix <question mark> affixed to an <edge pattern> or a <parenthesized path pattern expression>.

NOTE 19 — Unlike the situation in many regular expression languages, the <question mark> operator in <path pattern expression>s is not the same as {0,1}, the difference being that <questioned path primary> exposes its singleton element or subpath variables as conditional singletons, whereas {0, 1}, in common with all other quantifiers, exposes all element or subpath variables as a group.

4.7.4 Graph pattern variables

A *graph pattern variable GPV* is a site identified by an <identifier> (the *name* of the graph pattern variable) and having a value determined by a multi-path binding *MPB* to a <graph pattern>.

NOTE 20 — "Site" is defined in ISO/IEC 9075-1 Subclause 4.7.1, "Sites".

There are four kinds of graph pattern variables.

— Vertex variables; the value of a vertex variable is a list of vertices.

— Edge variables; the value of an edge variable is a list of edges.

— Path variables; the value of a path variable is a path binding.

— Subpath variables.

NOTE 21 — Subpath variables are not bound to a value in this edition of this document. They serve to assure multiset semantics in <path multiset alternation>.

In a <graph table> *GT*, an <identifier> shall not identify more than one graph pattern variable; thus *GT* defines a namespace in which there is a one-to-one correspondence between the names of graph pattern variables and the graph pattern variables that they name.

NOTE 22 — Because of this one-to-one correspondence, certain terms that are defined for graph pattern variables are also defined for their names, so that the name of the variable and the variable itself can be used interchangeably in the rules. These terms include "declare" and "expose".

Vertex variables and edge variables are collectively called *element variables*. An element variable is declared in either an <element pattern> (in which case it is a *primary element variable*) or in a <one row per iteration> in which case it is an *iterator variable*).

A primary vertex variable *VV* and its name are declared by an <element variable declaration> simply contained in a <vertex pattern>.

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

A primary edge variable *EV* and its name are declared by an <element variable declaration> simply contained in a <full edge pattern>.

A primary variable may be declared in more than one <element pattern>. A multiply declared primary variable *MDPV* expresses a natural equijoin in two circumstances.

— If *MDPV* is declared in both operands of a <path concatenation>.

— If *MPDV* is declared in two or more <path pattern>s of a <graph pattern>.

NOTE 23 — Declaring an element variable in two operands of a <path pattern union> or <path multiset alternation> does not express a natural equijoin.

<one row per vertex> declares a single iterator vertex variable and its name.

<one row per step> declares an iterator vertex variable, an iterator edge variable, and another iterator vertex variable, as well as their names.

Iterator element variables cannot be multiply declared.

A path variable *PV* and its name are declared by a <path variable declaration> simply contained in a <path pattern>. A path variable may only be declared once.

A subpath variable *SV* and its name are declared by a <subpath variable declaration> simply contained in a <parenthesized path pattern expression>. More than one operand of a <path pattern union> may declare *SV*; otherwise *SV* cannot be multiply declared.

4.7.5 References to graph pattern variables

Graph pattern variables are visible within the <graph table> *GT* in which they are declared. They may be referenced in scalar expressions and <search condition>s within *GT*.

Iterator variables and path variables may only be referenced in <graph table column definition>s.

Primary variables may be referenced in the following BNF non-terminals.

— <element pattern where clause>.

— <parenthesized path pattern where clause>.

— <graph pattern where clause>.

— <graph table column definition>.

If a primary element variable *PEV* is declared in a <quantified path primary> *QPP*, then it may bind to more than one graph element. References to *PEV* are interpreted contextually: if the reference occurs outside *QPP*, then the reference is to the complete list of graph elements that are bound to *PEV*. In this circumstance *PEV* is said to have *group degree of reference*. If the reference does not cross a quantifier, then the reference has *singleton degree of reference* and references at most one graph element, even if the multi-path binding binds *PEV* multiple times.

NOTE 24 — For example,

```
(X) -[E WHERE E.P > 1]->{1,10} (Y) WHERE SUM(E.P) < 100
```

This example references primary edge variable E twice: once in the <edge pattern> and once outside the <edge pattern>. Within the <edge pattern>, E has singleton degree of reference and the <property reference> E.P references a property of a single edge. On the other hand, the reference within the SUM aggregate has group degree of reference (because of the quantifier {1,10}) and references the list of edges that are bound to E.

A reference *R* to a graph pattern variable *GPV* is termed *local* in the following circumstances.

— If *GPV* is declared in an <element pattern> *EP* and *R* is contained in the <element pattern where clause> of *EP*.

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

— If *GPV* is declared in a <parenthesized path pattern expression> *PPPE* and *R* is contained in the <parenthesized path pattern where clause> of *PPPE*.

— If *R* is in a <graph pattern where clause> or a <graph table column definition>.

R has a degree of reference, determined by the Syntax Rules of Subclause 10.10, "<element reference>" and Subclause 10.11, "<path reference>". The degree of reference of R is one of the following: unconditional singleton, conditional singleton, or effectively bounded group.

NOTE 25 — Subpath variables cannot be referenced by SQL language defined by this document, though it is possible they will be referenceable in a future edition. At present their only use is to distinguish operands of a <path multiset alternation>.

NOTE 26 — In general, an element variable that is declared within a <quantified path primary> is bound by a multi-path binding to a list of graph elements. The reference *R*, on the other hand, can reference a proper subset of this list, based on the syntactic context in which *R* is placed. The degree of reference expresses the cardinality of the list that *R* references, as follows: an unconditional singleton references a list of length 1 (one), a conditional singleton references a list of length 0 (zero) or 1 (one), an effectively bounded group references a finite list. Syntax rules prohibit the possibility of referencing an infinite list.

A reference to a path variable always has unconditional singleton degree of reference.

References to graph pattern variables are subject to the following constraints.

— An operand *OP* of <path pattern union> or <path multiset alternation> *U* may only reference element variables declared in *OP*, or outside of *U*.

— A non-local reference shall have singleton degree of reference.

— A group reference shall be contained in an aggregated argument of a <set function specification>. The group references in an aggregated argument of a <set function specification> shall reference the same graph pattern variable. All other references to graph pattern variables in a <set function specification> shall have singleton degree of reference.

— A selective <path pattern> *SPP* shall not reference a graph pattern variable that is not declared in *SPP*.

— Iterator variables shall only be referenced in <graph table column definition>s.

If *GT* has a <graph table export clause> other than EXPORT NO SINGLETONS, then certain singleton <graph pattern variable>s (the exported <graph pattern variable>s) are visible in the <query specification> whose <from clause> simply contains *GT*. A syntactic transformation defines an equivalent <query specification> in which the <graph table> specifies EXPORT NO SINGLETONS, and all references to exported <graph pattern variable>s have been placed in <graph table column definition>s.

4.7.6 Path pattern matching

Path pattern matching is performed by Subclause 10.4, "<graph pattern>", which in turn may invoke Subclause 9.7, "Evaluation of a <path pattern expression>", and Subclause 9.8, "Evaluation of a selective <path pattern>", as well as other Subclauses incidentally invoked for expression evaluation.

In more detail, a <path pattern> can be seen as a regular expression built from <vertex pattern>s and <edge pattern>s. To formalize this, an alphabet is formed (in Subclause 9.6, "Machinery for graph pattern matching"), comprising the element variable names, plus additional special symbols for the anonymous vertex symbol, the anonymous edge symbol, bracket symbols to indicate the beginning and ending of bindings to <parenthesized path pattern expression>s, and subpath symbols to mark the beginning and ending of subpaths. The precise specification of the corresponding regular language is found in Subclause 9.7, "Evaluation of a <path pattern expression>".

Each <path pattern> of a <graph pattern> is evaluated independently of each other, resulting in a set of path bindings. A path binding is a list of elementary bindings; each elementary binding is an ordered pair (*LET*, *GE*), where *LET* is a member of the alphabet, and *GE* is

Case:

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

— if *LET* is a vertex variable or the anonymous vertex symbol, then a vertex;

— if *LET* is an edge variable or the anonymous edge symbol, then an edge;

— Otherwise, equal to *LET*.

NOTE 27 — In the formal semantics, <label expression>s are evaluated at this stage, but <search condition>s are not; hence, it is possible that some path bindings will be subsequently rejected because they fail a <search condition>. Implementations are, of course, free to "push down" predicate evaluation provided that the ultimate results are the same as prescribed by the formal semantics.

Projecting the elementary bindings of a path binding to the first component yields a word of the regular language of the <path pattern>. Projecting to the second component yields an annotated path, which is a path interspersed with mark-up to indicate the beginning and ending of <parenthesized path pattern expression>s and the beginning and ending of subpaths.

If a <path pattern> has an unbounded quantifier that is not in the scope of a restrictive <path mode> or the <different edges match mode>, then there may be infinitely many path bindings. Such a <path pattern> must have a selective <path search prefix> *SPSP*. Subclause 9.8, "Evaluation of a selective <path pattern>", is invoked to reduce this potentially infinite set of path bindings to a finite set. All <search condition>s contained in the selective <path pattern> are evaluated to reduce the set of path bindings prior to making the final selection according to *SPSP*.

NOTE 28 — An implementation cannot generate an infinite set and then apply <search condition>s; instead, it must enumerate the search space in a fashion enabling it to arrive at the same result as specified by the formal semantics. The formal semantics do not specify the algorithm for this enumeration, only the result.

At this point, there is a finite set of path bindings for each <path pattern> in a <graph pattern>. The cross product of these sets is formed; a member of the cross product is called a multi-path binding. A multi-path binding does not violate any restrictive <path mode> or <different edges match mode> that may be in force. The set of multi-path bindings is reduced by enforcing natural equijoins on the unconditional singleton variables exposed by the <path pattern>s.

Then, the <search condition>s in the <graph pattern> are evaluated, potentially further reducing the set of multi-path bindings. Those that remain satisfy all the selective <path search prefix>es and all the <search condition>s of the <graph pattern>.

Then, the function *REDUCE* (defined in Subclause 9.6, "Machinery for graph pattern matching") is applied to the remaining multi-path bindings. *REDUCE* removes the elementary bindings of the bracket symbols, and collapses adjacent anonymous vertex symbol bindings into a single elementary binding. The results, now called reduced matches, are deduplicated according to set semantics.

NOTE 29 — If there is a <path pattern union> in the <graph pattern>, then duplicates can arise.

4.7.7 Path modes

A <path mode> may be specified for any <parenthesized path pattern expression> or any <path pattern> from the following choices.

— WALK, the default <path mode>, is the absence of any filtering implied by the other <path mode>s.

— TRAIL, where path bindings with repeated edges are not returned.

— ACYCLIC, where path bindings with repeated vertices are not returned.

— SIMPLE, where path bindings with repeated vertices are not returned unless these repeated vertices are the first and the last in the path.

Using trail, acyclic, or simple matching path modes for all unbounded quantifiers guarantees that the result set of a graph pattern matching will be finite.

NOTE 30 — There is no implied hierarchy between path modes. Specifically, path mode SIMPLE does not imply path mode TRAIL. While such an implication holds for directed graphs, it does not hold for graphs with undirected edges. For instance,

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

the pattern ( ) - ( ) - ( ) on a graph with two vertices and one undirected edge between them returns a match under SIMPLE but does not return any matches under TRAIL.

4.7.8 Selective path search prefixes

The set of path bindings resulting from a graph pattern match can be further restricted by a selective <path search prefix> *SPSP*. *SPSP* is defined by partitioning the potentially infinite set of path bindings by the endpoints, which are the first and last vertices bound by the path bindings.

NOTE 31 — This partitioning is crucial to the definition of *SPSP*. *SPSP* makes a selection of some number of path bindings from each partition. For example, a path binding is "shortest" if its length is minimal within its partition. A "shortest" path binding in one partition can be longer than a "shortest" path binding in another partition.

SPSP can constrain the result set in the following ways.

— <any path search>: to non-deterministically pick some number of path bindings from each partition; the number is specified by a <simple value specification>.

— <all shortest path search>: to pick all the shortest path bindings in each partition.

— <counted shortest path search>: to non-deterministically pick some number of shortest path bindings from each partition; the number is specified by a <simple value specification>.

— <counted shortest group search>: to group each partition into groups of path bindings having the same length, order the groups by path length, and pick all path bindings in some number of groups from the front of each partition; the number is specified by a <simple value specification>.

The specification of *SPSP* guarantees that the result set of a graph pattern matching will be finite.

4.7.9 Match modes

A <graph pattern> *GP* may optionally specify a <match mode> that applies to all <path pattern>s simply contained in *GP*.

There are two <match mode>s.

— DIFFERENT EDGES: A matched edge is not permitted to bind to more than one edge variable. There are no restrictions on matched vertices.

— REPEATABLE ELEMENTS: There are no restrictions on matched edges or matched vertices.

If a <match mode> is not specified, then an implementation-defined (ID022) <match mode> is implicit.

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

5 Lexical elements

This Clause modifies Clause 5, "Lexical elements", in ISO/IEC 9075-2.

5.1 <SQL terminal character>

This Subclause modifies Subclause 5.1, "<SQL terminal character>", in ISO/IEC 9075-2.

Function

Define the terminal symbols of the SQL language and the elements of strings.

Format

```
<SQL special character> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <tilde>

<tilde> ::=
  ~ !! U+007E
```

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

5.2 <token> and <separator>

This Subclause modifies Subclause 5.2, "<token> and <separator>", in ISO/IEC 9075-2.

Function

Specify lexical units (tokens and separators) that participate in SQL language.

Format

```
<delimiter token> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <bracket right arrow>
  | <bracket tilde right arrow>
  | <left arrow>
  | <left arrow bracket>
  | <left arrow tilde>
  | <left arrow tilde bracket>
  | <left minus right>
  | <left minus slash>
  | <left tilde slash>
  | <minus left bracket>
  | <minus slash>
  | <multiset alternation operator>
  | <right bracket minus>
  | <right bracket tilde>
  | <slash minus>
  | <slash minus right>
  | <slash tilde>
  | <slash tilde right>
  | <tilde left bracket>
  | <tilde right arrow>
  | <tilde slash>

<bracket right arrow> ::=
  ]-> !! <U+005D, U+002D, U+003E>

<bracket tilde right arrow> ::=
  ]~> !! <U+005D, U+007E, U+003E>

<left arrow> ::=
  <- !! <U+003C, U+002D>

<left arrow bracket> ::=
  <-[ !! <U+003C, U+002D, U+005B>

<left arrow tilde> ::=
  <~ !! <U+003C, U+007E>

<left arrow tilde bracket> ::=
  <~[ !! <U+003C, U+007E, U+005B>

<left minus right> ::=
  <-> !! <U+003C, U+002D, U+003E>

<left minus slash> ::=
  <-/ !! <U+003C, U+002D, U+002F>

<left tilde slash> ::=
  <~/ !! <U+003C, U+007E, U+002F>
```

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

```
<minus left bracket> ::=
  -[ !! <U+002D, U+005B>

<minus slash> ::=
  -/ !! <U+002D, U+002F>

<multiset alternation operator> ::=
  |+| !! <U+007C, U+002B, U+007C>

<right bracket minus> ::=
  ]- !! <U+005D, U+002D>

<right bracket tilde> ::=
  ]~ !! <U+005D, U+007E>

<slash minus> ::=
  /- !! <U+002F, U+002D>

<slash minus right> ::=
  /-> !! <U+002F, U+002D, U+003E>

<slash tilde> ::=
  /~ !! <U+002F, U+007E>

<slash tilde right> ::=
  /~> !! <U+002F, U+007E, U+003E>

<tilde left bracket> ::=
  ~[ !! <U+007E, U+005B>

<tilde right arrow> ::=
  ~> !! <U+007E, U+003E>

<tilde slash> ::=
  ~/ !! <U+007E, U+002F>

<reserved word> ::=
    !! All alternatives from ISO/IEC 9075-2

  | ALL_DIFFERENT

  | BINDING_COUNT

  | ELEMENT_ID | ELEMENT_NUMBER | EXPORT

  | GRAPH | GRAPH_TABLE

  | MATCHNUM

  | PATH_LENGTH | PATH_NAME | PROPERTY_EXISTS

  | SAME

<non-reserved word> ::=
    !! All alternatives from ISO/IEC 9075-2

  | ACYCLIC

  | BINDINGS | BOUND

  | DESTINATION | DIFFERENT | DIRECTED

  | EDGE | EDGES | ELEMENTS

  | LABEL | LABELED
```


© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

```
  | NODE

  | PATHS | PROPERTIES | PROPERTY | PROPERTY_GRAPH_CATALOG | PROPERTY_GRAPH_NAME
  | PROPERTY_GRAPH_SCHEMA

  | RELATIONSHIP | RELATIONSHIPS

  | SHORTEST | SINGLETONS | STEP

  | TABLES | TRAIL

  | VERTEX

  | WALK

<edge synonym> ::=
    EDGE | RELATIONSHIP

<edges synonym> ::=
    EDGES | RELATIONSHIPS

<vertex synonym> ::=
    NODE | VERTEX
```

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

5.3 Names and identifiers

This Subclause modifies Subclause 5.4, "Names and identifiers", in ISO/IEC 9075-2.

Function

Specify names.

Format

```
<destination vertex table alias> ::=
  <vertex table alias>

<edge table alias> ::=
  <identifier>

<element table name> ::=
  <table name>

<element table alias> ::=
  <identifier>

<graph pattern variable> ::=
    <element variable>
  | <path or subpath variable>

<path or subpath variable> ::=
    <path variable>
  | <subpath variable>

<element variable> ::=
  <identifier>

<label name> ::=
  <identifier>

<path variable> ::=
  <identifier>

<subpath variable> ::=
  <identifier>

<property graph name> ::=
  <schema qualified name>

<property name> ::=
  <identifier>

<source vertex table alias> ::=
  <vertex table alias>

<vertex table alias> ::=
  <identifier>
```

Syntax Rules

No additional Syntax Rules.

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

Access Rules

No additional Access Rules.

General Rules

1) Insert after the last GR: An <element variable> identifies an element variable, which may be bound to a list of graph elements.

2) Insert after the last GR: A <path variable> identifies a path variable, which is bound to a path binding that is matched by a <path pattern>.

3) Insert after the last GR: A <subpath variable> identifies a subpath variable.

4) Insert after the last GR: A <property name> identifies a property in an SQL-property graph.

5) Insert after the last GR: A <label name> identifies a label in an SQL-property graph.

6) Insert after the last GR: A <destination vertex table alias> identifies a destination vertex table in an SQL-property graph.

7) Insert after the last GR: An <edge table alias> identifies an edge table in an SQL-property graph.

8) Insert after the last GR: An <element table alias> identifies an element table in an SQL-property graph.

9) Insert after the last GR: An <element table name> identifies a table.

10) Insert after the last GR: A <property graph name> identifies an SQL-property graph.

11) Insert after the last GR: A <source vertex table alias> identifies a source vertex table in an SQL-property graph.

12) Insert after the last GR: A <vertex table alias> identifies a vertex table in an SQL-property graph.

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

6 Scalar expressions

This Clause modifies Clause 6, "Scalar expressions", in ISO/IEC 9075-2.

6.1 <value expression primary>

This Subclause modifies Subclause 6.3, "<value expression primary>", in ISO/IEC 9075-2.

Function

Specify a value that is syntactically self-delimited.

Format

```
<non-parenthesized value expression primary> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <property reference>
  | <graphical value expression primary>

<graphical value expression primary> ::=
    <element_id function>
  | <graphical match number function>
  | <graphical path name function>
  | <graphical element number function>
  | <graphical path length function>
```

Syntax Rules

1) Augment SR 1) by adding "<property reference> and <graphical value expression primary>" to the list of BNF non-terminals simply contained in <value expression primary>.

2) Insert after SR 1): The declared type of <graphical value expression primary> is the declared type of the simply contained <element_id function>, <graphical match number function>, <graphical path name function>, <graphical element number function>, or <graphical path length function>.

Access Rules

No additional Access Rules.

General Rules

1) Augment GR 1) by adding "<property reference> and <graphical value expression primary>" to the list of BNF non-terminals simply contained in <value expression primary>.

2) Insert after GR 1): The value of <graphical value expression primary> is the value of the simply contained <element_id function>, <graphical match number function>, <graphical path name function>, <graphical element number function>, or <graphical path length function>.

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

6.2 <identifier chain>

This Subclause modifies Subclause 6.6, "<identifier chain>", in ISO/IEC 9075-2.

Function

Disambiguate a period-separated chain of identifiers.

Format

No additional Format items.

Syntax Rules

1) Insert after SR 7): A property *PR* is said to be *refinable* if the declared type of *PR* is a row type, a JSON type, or a structured type.

2) Augment the lead text of SR 9) by adding "a property *PR*" to the list of possible referents of a basis.

3) Augment SR 9)b)iv) by adding "<element variable> " as alternatives to both occurrences of <correlation name>.

4) Insert after SR 9)b)iv)1): If $N = 2$ and *EN* is an exposed <element variable>, then

   a) Let *GT* be the <graph table> that is the scope of *EN*. Let *SPG* be the SQL-property graph identified by the <graph reference> simply contained in *GT*, and let *GP* be the <graph pattern> simply contained in *GT*. Let *EV* be the element variable that is identified by *EN*.

   b) The Syntax Rules of Subclause 9.2, "Contextual inference of a set of labels", are applied with *SPG* as *SQL-PROPERTY GRAPH*, *GP* as *GRAPH PATTERN*, *EV* as *ELEMENT VARIABLE*, and the <graph table column definition>, <parenthesized path pattern where clause> or <graph pattern where clause> that simply contains *IC* as *CONTEXT*; let *SOL* be the *SET OF LABELS* returned from the application of those Syntax Rules.

   c) Let *SOPN* be the set of property names of the properties of the labels of *SOL*.

   d) If I_2 is a property name in *SOPN*, then PIC_2 is a candidate basis of *IC*, the scope of PIC_2 is the scope of *EN*, and the referent of PIC_2 is the property *PR* named by I_2.

5) Insert after SR 9)b)iv)2): If $N > 2$ and *EN* is an exposed <element variable>, then

   a) Let *GT* be the <graph table> that is the scope of *EN*. Let *SPG* be the SQL-property graph identified by the <graph reference> simply contained in *GT*, and let *GP* be the <graph pattern> simply contained in *GT*. Let *EV* be the element variable that is identified by *EN*.

   b) The Syntax Rules of Subclause 9.2, "Contextual inference of a set of labels", are applied with *SPG* as *SQL-PROPERTY GRAPH*, *GP* as *GRAPH PATTERN*, *EV* as *ELEMENT VARIABLE*, and the <graph table column definition>, <parenthesized path pattern where clause>, or <graph pattern where clause> that simply contains *IC* as *CONTEXT*; let *SOL* be the *SET OF LABELS* returned from the application of those Syntax Rules.

   c) Let *SOPN* be the set of property names of the properties of the labels of *SOL*.

   d) If I_2 is a property name in *SOPN* and the property *PR* named by I_2 is refinable, then PIC_2 is a candidate basis of *IC*, the scope of PIC_2 is the scope of *EN*, and the referent of PIC_2 is the property *PR* named by I_2.

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

6) Insert after SR 16): A <basic identifier chain> whose basis referent is a property is a *property reference*.

Access Rules

No additional Access Rules.

General Rules

1) Insert after GR 4): If *BIC* is a property reference, then *BIC* references the property that is the basis referent of *BIC*.

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

6.3 <set function specification>

This Subclause modifies Subclause 6.9, "<set function specification>", in ISO/IEC 9075-2.

Function

Specify a value derived by the application of a function to an argument.

Format

No additional Format items.

Syntax Rules

1) Insert after SR 3): An <element reference> simply contained in an <element variable count function> is an *aggregated argument* of *SFE*.

2) Insert after SR 4): If <set function specification> *SFS* is contained in <graph table> *GT* without an intervening <query expression>, then:

   a) <grouping operation> shall not be specified.

   b) *SFS* is a *within-match aggregate*.

   c) The aggregated arguments of *SFS* shall reference exactly one element variable *AEV* whose degree of reference is group. The degree of reference of *AEV* shall be effectively bounded group. *AEV* is called the *aggregated element variable* of *SFS*.

      NOTE 32 — The aggregated element variable can be referenced more than once; for example, SUM (P.X * P.Y) references the element variable P twice. Since P is the only element variable referenced in the aggregated argument of SUM, the degree of reference of P will be effectively bounded group.

   d) Every element variable other than *AEV* that is referenced in *SFS* shall have singleton degree of reference.

      NOTE 33 — Assuming that *P* and *Q* are element variables, in the example

      ```
      PERCENTILE_CONT (P.X) WITHIN GROUP (ORDER BY Q.Y)
      ```

      P.X is a non-aggregated argument and therefore P must have singleton degree of reference. Q.Y is an aggregated argument and therefore Q is the aggregated element variable and must have effectively bounded group degree of reference.

      As another example:

      ```
      SUM (P.X * Q.Y)
      ```

      one of the variables P and Q must have group degree of reference (thereby being the aggregated element variable) and the other must have singleton degree of reference.

3) Insert after SR 6)a): If *SFS* is a within-match aggregate, then the aggregation query of *SFS* is *GT*.

4) Convert SR 7) to be: Case:

   a) If *SFS* is a within-match aggregate, then *SFS* shall be contained in a <parenthesized path pattern where clause>, <graph pattern where clause> or <graph table columns clause> simply contained in *GT*

   b) Otherwise, the original SR 7)

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

1) Insert after the last CR: Without Feature G120, "Within-match aggregates", conforming SQL language shall not contain a within-match aggregate.

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

6.4 <case expression>

This Subclause modifies Subclause 6.12, "<case expression>", in ISO/IEC 9075-2.

Function

Specify a conditional value.

Format

```
<case operand> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <element reference>

<when operand> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <directed predicate part 2>
  | <labeled predicate part 2>
  | <source predicate part 2>
  | <destination predicate part 2>
  | <bound predicate part 2>
```

Syntax Rules

1) Insert after SR 2)c): If any <when operand> is <directed predicate part 2>, <labeled predicate part 2>, <source predicate part 2>, <destination predicate part 2>, or <bound predicate part 2>, then *CO* shall be <element reference> and every <when operand> shall be <directed predicate part 2>, <labeled predicate part 2>, <source predicate part 2>, <destination predicate part 2>, or <bound predicate part 2>; otherwise, *CO* shall not be <element reference> and no <when operand> shall be <directed predicate part 2>, <labeled predicate part 2>, <source predicate part 2>, <destination predicate part 2>, or <bound predicate part 2>.

NOTE 34 — This rule disambiguates whether a single <identifier> is interpreted as an <element reference> or a <row value predicand> that is a single <identifier>.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

6.5 <property reference>

Function

Reference a property of a graph element.

Format

```
<property reference> ::=
  <basic identifier chain>
```

Syntax Rules

1) The <basic identifier chain> *BIC* simply contained in a <property reference> *PR* shall be a property reference.

2) The declared type of *PR* is the declared type of *BIC*.

3) Let *EV* be the first <identifier> simply contained in *BIC*. *EV* is an <element variable>. Let *ER* be the <element reference>:

   EV

   ER shall be valid according to the Syntax Rules of Subclause 10.10, "<element reference>".

Access Rules

None.

General Rules

1) Let *LOE* be the list of graph elements bound to *ER*, and let *PN* be the property name contained in *ER*.

   NOTE 35 — The list of graph elements bound to *ER* is determined by the General Rules of Subclause 9.9, "Applying bindings to evaluate an expression", which is invoked prior to evaluating a <value expression> or <search condition> that contains *PR*.

2) If the degree of reference of *ER* is singleton, then

   Case:

   a) If *LOE* is empty, then the result of *PR* is the null value.

   b) Otherwise, let *E* be the sole graph element in *LOE*.

      Case:

      i) If *E* does not have a property whose name is *PN*, then the result of *PR* is the null value.

      ii) If the property of *E* whose name is *PN* is flagged as unusable, then an exception condition *EC* is raised where *EC* is the exception condition recorded as additional information when the property was marked as unusable.

         NOTE 36 — The exception condition is recorded as additional information in GR 6)b)iii)3)F) of Subclause 9.5, "Converting a tabular property graph to a pure property graph".

      iii) Otherwise, the result of *PR* is the value of the property of *E* whose name is *PN*.

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

NOTE 37 — A <property reference> that references a singleton element variable is resolved by this Subclause. References to a group element variable must be contained in an aggregated argument of a <set function specification>. The <set function specification> contains an <aggregate function>; the General Rules of Subclause 10.1, "<aggregate function>", invoke the General Rules of Subclause 9.10, "Applying bindings to evaluate a subexpression of an aggregate", to decompose such group references into a list of singleton references, so this Subclause only needs to handle singletons.

Conformance Rules

1) Without Feature G090, "Property reference", conforming SQL language shall not contain a <property reference>.

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

6.6 <element_id function>

Function

Generate a unique identifier for a graph element.

Format

```
<element_id function> ::=
  ELEMENT_ID <left paren> <element reference> <right paren>
```

Syntax Rules

1) The <element reference> shall have singleton degree of reference.

2) The declared type of <element_id function> is an implementation-defined (IV120) type that is permitted as the declared type of an operand of an equality operation according to the Syntax Rules of Subclause 9.11, "Equality operations", in ISO/IEC 9075-2 and as the declared type of an operand of a grouping operation according to the Syntax Rules of Subclause 9.12, "Grouping operations", in ISO/IEC 9075-2.

Access Rules

1) Access Rules for <element_id function> are implementation-defined (IA033).

General Rules

1) Let *LOE* be the list of graph elements bound to the <element reference>.

2) Case:

   a) If *LOE* is empty, then the value of <element_id function> is the null value.

   b) Otherwise, let *GE* be the sole graph element in *LOE*. The value of <element_id function> is an implementation-dependent (UV017) value that encapsulates the identity of *GE* in the SQL-property graph that contains *GE* for the duration of the innermost executing SQL-statement.

Conformance Rules

1) Without Feature G100, "ELEMENT_ID function", conforming SQL language shall not contain an <element_id function>.

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

6.7 <graphical match number function>

Function

Return a number that uniquely identifies a match in a set of matches.

Format

```
<graphical match number function> ::=
  MATCHNUM <left paren> <right paren>
```

Syntax Rules

1) <graphical match number function> *GMNF* shall be contained in the <graph table columns clause> of a <graph table> *GT*, or in the scope of the exported <graph pattern variable>s of a <graph table> *GT*.

2) The declared type of the result of *GMNF* is implementation-defined (IV121) exact numeric with scale 0 (zero).

Access Rules

None.

General Rules

1) The value of the result of *GMNF* is the current match number.

NOTE 38 — The current match number is specified in the General Rules of Subclause 7.1, "<table reference>".

Conformance Rules

1) Without Feature G803, "MATCHNUM function", conforming SQL language shall not contain <graphical match number function>.

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

6.8 <graphical path name function>

Function

Return the name of the path that the iterator variables are currently iterating over.

Format

```
<graphical path name function> ::=
  PATH_NAME <left paren> <right paren>
```

Syntax Rules

1) <graphical path name function> *GPNF* shall be contained in the <graph table columns clause> of a <graph table> *GT*, or in the scope of the exported <graph pattern variable>s of a <graph table> *GT*.

2) *GT* shall specify <one row per iteration>.

3) The declared type of the result of *GPNF* is character string with implementation-defined (IL005) maximum length, character set SQL_IDENTIFIER, and collation SQL_IDENTIFIER.

Access Rules

None.

General Rules

1) The value of the result of *GPNF* is

   Case:

   a) If there is a current path name, then a character string whose value is equivalent to the current path name.

   b) Otherwise, the null value.

   NOTE 39 — The current path name is specified by the General Rules of Subclause 9.11, "Applying bindings to generate a row".

Conformance Rules

1) Without Feature G800, "PATH_NAME function", conforming SQL language shall not contain <graphical path name function>.

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-16-2023

6.9 <graphical element number function>

Function

Return the sequential element number of the graph element binding that an iterator variable is currently positioned on.

Format

```
<graphical element number function> ::=
  ELEMENT_NUMBER <left paren> <element reference> <right paren>
```

Syntax Rules

1) <graphical element number function> *GENF* shall be contained in the <graph table columns clause> of a <graph table> *GT*, or in the scope of the exported <graph pattern variable>s of a <graph table> *GT*.

2) *GT* shall specify <one row per iteration> *ORPI*, and <element reference> shall reference an iterator variable *IV* declared by *ORPI*.

3) The declared type of the result of *GENF* is implementation-defined (IV122) exact numeric with scale 0 (zero).

Access Rules

None.

General Rules

1) The value of the result of *GENF* is the current element number of *IV*.

NOTE 40 — The current element number of an iterator variable is specified by the General Rules of Subclause 9.11, "Applying bindings to generate a row".

Conformance Rules

1) Without Feature G801, "ELEMENT_NUMBER function", conforming SQL language shall not contain <graphical element number function>.

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-16-2023