INTERNATIONAL STANDARD

ISO/IEC 9075-2

Sixth edition
2023-06

Information technology — Database languages SQL —

Part 2: Foundation (SQL/Foundation)

Technologies de l'information — Langages de base de données SQL — Partie 2: Fondations (SQL/Fondations)

Reference number
ISO/IEC 9075-2:2023(E)

AVA Documents
Document Sharing Hub
© ISO/IEC 2023

Full document details: ISO IEC 09075-2-2023

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 09075-2-2023

Contents

Page

Foreword . . . xix
Introduction . . . xxi
1 Scope . . . 1
2 Normative references . . . 2
3 Terms and definitions . . . 4
3.1 Definitions taken from ISO/IEC 10646:2020 . . . 4
3.2 Definitions taken from ISO/IEC 14651:2020 . . . 5
3.3 Definitions taken from ISO 8601-1:2019 . . . 5
3.4 Definitions taken from XQuery and XPath Functions and Operators 3.1 . . . 7
3.5 Definitions provided in this document . . . 7
4 Concepts . . . 15
4.1 Notations and conventions . . . 15
4.1.1 Notations . . . 15
4.1.2 Use of terms . . . 15
4.2 Data types . . . 16
4.2.1 General introduction to data types . . . 16
4.2.2 Naming of predefined types . . . 17
4.2.3 Host language data types . . . 18
4.2.4 Data type terminology . . . 18
4.2.5 Properties of distinct . . . 20
4.3 Character strings . . . 21
4.3.1 Introduction to character strings . . . 21
4.3.2 Comparison of character strings . . . 21
4.3.3 Operations involving character strings . . . 22
4.3.3.1 Regular expression syntaxes . . . 22
4.3.3.2 Operators that operate on character strings and return character strings . . . 22
4.3.3.3 Other operators involving character strings . . . 25
4.3.3.4 Operations involving large object character strings . . . 26
4.3.4 Character repertoires . . . 27
4.3.5 Character encoding forms . . . 28
4.3.6 Collations . . . 28
4.3.7 Character sets . . . 29
4.3.8 Universal character sets . . . 31
4.4 Binary strings . . . 31
4.4.1 Introduction to binary strings . . . 31
4.4.2 Binary string comparison . . . 31
4.4.3 Operations involving binary strings . . . 31
4.4.3.1 Operators that operate on binary strings and return binary strings . . . 31

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

4.4.3.2 Other operators involving binary strings. . . . 32
4.5 Numbers. . . . 32
4.5.1 Introduction to numbers. . . . 32
4.5.2 Characteristics of numbers. . . . 33
4.5.3 Operations involving numbers. . . . 34
4.6 Boolean types. . . . 35
4.6.1 Introduction to Boolean types. . . . 35
4.6.2 Comparison and assignment of Booleans. . . . 36
4.6.3 Operations involving Booleans. . . . 36
4.6.3.1 Operations on Booleans that return Booleans. . . . 36
4.6.3.2 Other operators involving Booleans. . . . 36
4.7 Datetimes and intervals. . . . 36
4.7.1 Introduction to datetimes and intervals. . . . 36
4.7.2 Datetimes. . . . 37
4.7.3 Intervals. . . . 39
4.7.4 Operations involving datetimes and intervals. . . . 41
4.8 JSON types. . . . 41
4.8.1 Introduction to JSON types. . . . 41
4.8.2 Comparison and assignment of JSON values. . . . 42
4.8.3 Operations involving JSON values. . . . 43
4.9 User-defined types. . . . 43
4.9.1 Introduction to user-defined types. . . . 43
4.9.2 Distinct types. . . . 44
4.9.3 Structured types. . . . 44
4.9.3.1 Introduction to structured types. . . . 44
4.9.3.2 Observer functions and mutator functions. . . . 44
4.9.3.3 Constructors. . . . 44
4.9.3.4 Subtypes and supertypes. . . . 45
4.9.4 Methods. . . . 46
4.9.5 User-defined type comparison and assignment. . . . 47
4.9.6 Transforms for user-defined types. . . . 48
4.9.7 User-defined type descriptor. . . . 48
4.10 Row types. . . . 50
4.11 Reference types. . . . 50
4.11.1 Introduction to reference types. . . . 50
4.11.2 Operations involving references. . . . 51
4.12 Collection types. . . . 51
4.12.1 Introduction to collection types. . . . 51
4.12.2 Arrays. . . . 52
4.12.3 Multisets. . . . 52
4.12.4 Collection comparison and assignment. . . . 52
4.12.5 Operations involving arrays. . . . 53
4.12.5.1 Operators that operate on array values and return array elements. . . . 53
4.12.5.2 Operators that operate on array values and return array values. . . . 53
4.12.5.3 Operators that operate on array values and return numbers. . . . 53
4.12.6 Operations involving multisets. . . . 53
4.12.6.1 Operators that operate on multisets and return multiset elements. . . . 53

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

4.12.6.2 Operators that operate on multisets and return multisets. . . . 53
4.12.6.3 Operators that operate on multiset values and return numbers. . . . 54
4.13 Data conversions. . . . 54
4.14 Domains. . . . 55
4.15 Columns, fields, and attributes. . . . 55
4.16 Periods. . . . 57
4.16.1 Introduction to periods. . . . 57
4.16.2 Operations involving periods. . . . 58
4.17 Tables. . . . 58
4.17.1 Introduction to tables. . . . 58
4.17.2 Base tables. . . . 58
4.17.2.1 Introduction to base tables. . . . 58
4.17.2.2 Regular persistent base tables. . . . 59
4.17.2.3 System-versioned tables. . . . 59
4.17.2.4 Temporary tables. . . . 59
4.17.3 Derived tables. . . . 60
4.17.4 Transient tables. . . . 61
4.17.5 Unique identification of tables. . . . 61
4.17.6 Table updatability. . . . 61
4.17.7 Table descriptors. . . . 62
4.17.8 Syntactic analysis of derived tables and cursors. . . . 63
4.17.9 Referenceable tables, subtables, and supertables. . . . 65
4.17.10 Operations involving tables. . . . 66
4.17.11 Range variables. . . . 69
4.17.12 Identity columns. . . . 70
4.17.13 Base columns and generated columns. . . . 70
4.17.14 Grouped tables. . . . 70
4.17.15 Windowed tables. . . . 71
4.18 Data analysis operations. . . . 72
4.18.1 Introduction to data analysis operations. . . . 72
4.18.2 Group functions. . . . 72
4.18.3 Window functions. . . . 73
4.18.4 Aggregate functions. . . . 75
4.18.5 Row pattern measures. . . . 77
4.19 Row pattern matching. . . . 78
4.19.1 Introduction to row pattern matching. . . . 78
4.19.2 Matching rows with a pattern. . . . 78
4.19.3 Row pattern matching illustrated. . . . 79
4.19.4 Row pattern partitioning. . . . 83
4.19.5 Row ordering. . . . 83
4.19.6 Row pattern measure columns. . . . 83
4.19.7 Number of rows per match. . . . 83
4.19.8 Skipping rows after matching. . . . 84
4.20 Row patterns. . . . 84
4.21 Unions of row pattern variables. . . . 85
4.22 Defining Boolean conditions. . . . 85
4.23 Scalar expressions in row pattern matching. . . . 86

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

4.23.1 Introduction to scalars in row pattern matching. . . . 86
4.23.2 Running vs. final semantics. . . . 86
4.23.3 Row pattern navigation operations. . . . 87
4.23.4 Row pattern classifier function. . . . 87
4.23.5 Row pattern match number function. . . . 87
4.24 Determinism. . . . 87
4.25 Integrity constraints. . . . 88
4.25.1 Overview of integrity constraints. . . . 88
4.25.2 Checking of constraints. . . . 88
4.25.3 Table constraints. . . . 89
4.25.3.1 Introduction to table constraints. . . . 89
4.25.3.2 Unique constraints. . . . 89
4.25.3.3 Referential constraints. . . . 90
4.25.3.4 Table check constraints. . . . 93
4.25.4 Domain constraints. . . . 93
4.25.5 Assertions. . . . 93
4.26 Functional dependencies. . . . 94
4.26.1 Overview of functional dependency rules and notations. . . . 94
4.26.2 General rules and definitions. . . . 94
4.26.3 Known functional dependencies in a base table. . . . 95
4.26.4 Known functional dependencies in a viewed table. . . . 96
4.26.5 Known functional dependencies in a transition table. . . . 96
4.26.6 Known functional dependencies in <table value constructor>. . . . 96
4.26.7 Known functional dependencies in a <joined table>. . . . 96
4.26.8 Known functional dependencies in a <table primary>. . . . 98
4.26.9 Known functional dependencies in a <table factor>. . . . 99
4.26.10 Known functional dependencies in a <table reference>. . . . 99
4.26.11 Known functional dependencies in the result of a <from clause>. . . . 99
4.26.12 Known functional dependencies in the result of a <where clause>. . . . 99
4.26.13 Known functional dependencies in the result of a <group by clause>. . . . 100
4.26.14 Known functional dependencies in the result of a <having clause>. . . . 100
4.26.15 Known functional dependencies in a <query specification>. . . . 100
4.26.16 Known functional dependencies in a <query expression>. . . . 101
4.27 Candidate keys. . . . 101
4.28 SQL-schemas. . . . 102
4.29 Sequence generators. . . . 103
4.29.1 General description of sequence generators. . . . 103
4.29.2 Operations involving sequence generators. . . . 104
4.30 SQL-client modules. . . . 104
4.31 Embedded syntax. . . . 105
4.32 Dynamic SQL concepts. . . . 106
4.32.1 Introduction to dynamic SQL. . . . 106
4.32.2 Overview of dynamic SQL for constructed SQL-statements. . . . 106
4.32.3 Overview of dynamic SQL for polymorphic table functions. . . . 107
4.32.4 Dynamic SQL statements and descriptor areas. . . . 107
4.33 Direct invocation of SQL. . . . 109
4.34 Externally-invoked procedures. . . . 109

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

4.35 SQL-invoked routines. . . . 109
4.35.1 Overview of SQL-invoked routines. . . . 109
4.35.2 Characteristics of SQL-invoked routines. . . . 111
4.35.3 Execution of conventional SQL-invoked routines. . . . 114
4.35.4 Invocation of polymorphic table functions. . . . 115
4.35.5 Routine descriptors. . . . 119
4.35.6 Result sets returned by SQL-invoked procedures. . . . 122
4.36 SQL-paths. . . . 123
4.37 Host parameters. . . . 123
4.37.1 Overview of host parameters. . . . 123
4.37.2 Status parameters. . . . 123
4.37.3 Data parameters. . . . 124
4.37.4 Indicator parameters. . . . 124
4.37.5 Locators. . . . 124
4.38 Diagnostics area. . . . 125
4.39 Host languages. . . . 126
4.40 Cursors. . . . 127
4.40.1 General description of cursors. . . . 127
4.40.2 Operations on and using cursors. . . . 131
4.41 SQL-statements. . . . 132
4.41.1 Classes of SQL-statements. . . . 132
4.41.2 SQL-statements classified by function. . . . 133
4.41.2.1 SQL-schema statements. . . . 133
4.41.2.2 SQL-data statements. . . . 134
4.41.2.3 SQL-data change statements. . . . 135
4.41.2.4 SQL-transaction statements. . . . 136
4.41.2.5 SQL-connection statements. . . . 136
4.41.2.6 SQL-control statements. . . . 136
4.41.2.7 SQL-session statements. . . . 136
4.41.2.8 SQL-diagnostics statements. . . . 137
4.41.2.9 SQL-dynamic statements. . . . 137
4.41.2.10 SQL embedded exception declaration. . . . 137
4.41.3 SQL-statements and SQL-data access indication. . . . 137
4.41.4 SQL-statements and transaction states. . . . 138
4.41.5 SQL-statement atomicity and statement execution contexts. . . . 140
4.41.6 Embeddable SQL-statements. . . . 141
4.41.7 Preparable and immediately executable SQL-statements. . . . 142
4.41.8 Directly executable SQL-statements. . . . 144
4.42 Basic security model. . . . 145
4.42.1 Authorization identifiers. . . . 145
4.42.1.1 Introduction to authorization identifiers. . . . 145
4.42.1.2 SQL-session authorization identifiers. . . . 145
4.42.1.3 SQL-client module authorization identifiers. . . . 146
4.42.1.4 SQL-schema authorization identifiers. . . . 146
4.42.2 Privileges. . . . 146
4.42.3 Roles. . . . 149
4.42.4 Security model definitions. . . . 149

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

4.43 SQL-transactions. . . . 149
4.43.1 General description of SQL-transactions. . . . 149
4.43.2 Savepoints. . . . 150
4.43.3 Properties of SQL-transactions. . . . 151
4.43.4 Isolation levels of SQL-transactions. . . . 151
4.43.5 Implicit rollbacks. . . . 152
4.43.6 Effects of SQL-statements in an SQL-transaction. . . . 153
4.43.7 Encompassing transactions. . . . 153
4.43.7.1 Encompassing transaction belonging to an external agent. . . . 153
4.43.7.2 Encompassing transaction belonging to the SQL-agent. . . . 153
4.44 SQL-connections. . . . 154
4.45 SQL-sessions. . . . 155
4.45.1 General description of SQL-sessions. . . . 155
4.45.2 SQL-session identification. . . . 156
4.45.3 SQL-session properties. . . . 156
4.45.4 SQL-session context management. . . . 159
4.45.5 Execution contexts. . . . 159
4.45.6 Routine execution context. . . . 159
4.46 Triggers. . . . 160
4.46.1 General description of triggers. . . . 160
4.46.2 Trigger execution. . . . 161
4.47 Client-server operation. . . . 163
4.48 JSON data handling in SQL. . . . 163
4.48.1 Introduction. . . . 163
4.48.2 Implied JSON data model. . . . 164
4.48.3 SQL/JSON data model. . . . 165
4.48.4 SQL/JSON functions. . . . 166
4.48.5 Overview of SQL/JSON path language. . . . 167

5 Lexical elements. . . . 169
5.1 <SQL terminal character>. . . . 169
5.2 <token> and <separator>. . . . 173
5.3 <literal>. . . . 183
5.4 Names and identifiers. . . . 194

6 Scalar expressions. . . . 206
6.1 <data type>. . . . 206
6.2 <field definition>. . . . 219
6.3 <value expression primary>. . . . 221
6.4 <value specification> and <target specification>. . . . 223
6.5 <contextually typed value specification>. . . . 228
6.6 <identifier chain>. . . . 230
6.7 <column reference>. . . . 234
6.8 <SQL parameter reference>. . . . 237
6.9 <set function specification>. . . . 238
6.10 <window function>. . . . 241
6.11 <nested window function>. . . . 247
6.12 <case expression>. . . . 250
6.13 <cast specification>. . . . 254

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

6.14 <next value expression>. 271
6.15 <greatest or least function>. 273
6.16 <field reference>. 275
6.17 <subtype treatment>. 276
6.18 <method invocation>. 278
6.19 <static method invocation>. 280
6.20 <new specification>. 282
6.21 <attribute or method reference>. 284
6.22 <dereference operation>. 285
6.23 <method reference>. 286
6.24 <reference resolution>. 288
6.25 <array element reference>. 290
6.26 <multiset element reference>. 291
6.27 <row pattern navigation operation>. 292
6.28 <JSON value function>. 296
6.29 <value expression>. 298
6.30 <numeric value expression>. 300
6.31 <numeric value function>. 302
6.32 <string value expression>. 316
6.33 <string value function>. 321
6.34 <JSON value constructor>. 341
6.35 <JSON query>. 348
6.36 <JSON simplified accessor>. 351
6.37 <JSON serialize>. 354
6.38 <JSON value expression>. 355
6.39 <JSON typed value function>. 356
6.40 <JSON parse>. 357
6.41 <JSON scalar>. 359
6.42 <datetime value expression>. 360
6.43 <datetime value function>. 363
6.44 <interval value expression>. 365
6.45 <interval value function>. 369
6.46 <boolean value expression>. 370
6.47 <array value expression>. 374
6.48 <array value function>. 376
6.49 <array value constructor>. 378
6.50 <multiset value expression>. 380
6.51 <multiset value function>. 383
6.52 <multiset value constructor>. 384

7 Query expressions. 386

7.1 <row value constructor>. 386
7.2 <row value expression>. 389
7.3 <table value constructor>. 391
7.4 <table expression>. 393
7.5 <from clause>. 394
7.6 <table reference>. 397
7.7 <row pattern recognition clause>. 415

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub

Full document details: ISO IEC 09075-2-2023

7.8 <row pattern measures>. 420
7.9 <row pattern common syntax>. 422
7.10 <joined table>. 427
7.11 <JSON table>. 437
7.12 <where clause>. 451
7.13 <group by clause>. 452
7.14 <having clause>. 461
7.15 <window clause>. 463
7.16 <query specification>. 478
7.17 <query expression>. 486
7.18 <search or cycle clause>. 503
7.19 <subquery>. 507
8 Predicates. 509
8.1 <predicate>. 509
8.2 <comparison predicate>. 511
8.3 <between predicate>. 519
8.4 <in predicate>. 520
8.5 <like predicate>. 522
8.6 <similar predicate>. 528
8.7 <regex like predicate>. 534
8.8 <null predicate>. 536
8.9 <quantified comparison predicate>. 538
8.10 <exists predicate>. 540
8.11 <unique predicate>. 541
8.12 <normalized predicate>. 543
8.13 <match predicate>. 545
8.14 <overlaps predicate>. 548
8.15 <distinct predicate>. 550
8.16 <member predicate>. 553
8.17 <submultiset predicate>. 555
8.18 <set predicate>. 557
8.19 <type predicate>. 558
8.20 <period predicate>. 560
8.21 <search condition>. 565
8.22 <JSON predicate>. 566
8.23 <JSON exists predicate>. 568
9 Additional common rules. 570
9.1 Retrieval assignment. 570
9.2 Store assignment. 576
9.3 Passing a value from a host language to the SQL-server. 582
9.4 Passing a value from the SQL-server to a host language. 586
9.5 Result of data type combinations. 590
9.6 Subject routine determination. 594
9.7 Type precedence list determination. 596
9.8 Host parameter mode determination. 600
9.9 Type name determination. 602
9.10 Determination of identical values. 604

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

9.11 Equality operations. . . . 606
9.12 Grouping operations. . . . 608
9.13 Multiset element grouping operations. . . . 610
9.14 Ordering operations. . . . 612
9.15 Collation determination. . . . 614
9.16 Potential sources of non-determinism. . . . 616
9.17 Executing an <SQL procedure statement>. . . . 620
9.18 Invoking an SQL-invoked routine. . . . 625
9.19 Processing a method invocation. . . . 655
9.20 Transformation of query specifications. . . . 657
9.21 Execution of array-returning external functions. . . . 660
9.22 Execution of multiset-returning external functions. . . . 664
9.23 Evaluation and transformation of <window function>. . . . 665
9.24 Compilation of an invocation of a polymorphic table function. . . . 669
9.25 Execution of an invocation of a polymorphic table function. . . . 674
9.26 Signatures of PTF component procedures. . . . 685
9.27 Invocation of a PTF component procedure. . . . 688
9.28 XQuery regular expression matching. . . . 691
9.29 XQuery regular expression replacement. . . . 694
9.30 Data type identity. . . . 696
9.31 Determination of a from-sql function. . . . 698
9.32 Determination of a from-sql function for an overriding method. . . . 699
9.33 Determination of a to-sql function. . . . 700
9.34 Determination of a to-sql function for an overriding method. . . . 701
9.35 Generation of the next value of a sequence generator. . . . 702
9.36 Creation of a sequence generator. . . . 704
9.37 Altering a sequence generator. . . . 707
9.38 Generation of the hierarchical <query expression> of a view. . . . 710
9.39 Determination of view privileges. . . . 711
9.40 Determination of view component privileges. . . . 713
9.41 Row pattern recognition in a sequence of rows. . . . 717
9.42 Parsing JSON text. . . . 721
9.43 Serializing an SQL/JSON item. . . . 724
9.44 Converting an SQL/JSON sequence to an SQL/JSON item. . . . 726
9.45 SQL/JSON path language: lexical elements. . . . 729
9.46 SQL/JSON path language: syntax and semantics. . . . 733
9.47 Processing <JSON API common syntax>. . . . 764
9.48 Casting an SQL/JSON sequence to an SQL type. . . . 765
9.49 Serializing an SQL/JSON sequence to an SQL string type. . . . 768
9.50 Converting a datetime to a formatted character string. . . . 772
9.51 Converting a formatted character string to a datetime. . . . 776
9.52 Datetime templates. . . . 784

10 Additional common elements. . . . 788

10.1 <interval qualifier>. . . . 788
10.2 <language clause>. . . . 792
10.3 <path specification>. . . . 794
10.4 <routine invocation>. . . . 795

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

10.5 <character set specification>. 799
10.6 <specific routine designator>. 801
10.7 <collate clause>. 804
10.8 <constraint name definition> and <constraint characteristics>. 805
10.9 <aggregate function>. 807
10.10 <sort specification list>. 823
10.11 <JSON aggregate function>. 825
10.12 <JSON input expression>. 831
10.13 <JSON output clause>. 833
10.14 <JSON API common syntax>. 835

11 Schema definition and manipulation. 837

11.1 <schema definition>. 837
11.2 <drop schema statement>. 840
11.3 <table definition>. 843
11.4 <column definition>. 855
11.5 <default clause>. 861
11.6 <table constraint definition>. 865
11.7 <unique constraint definition>. 867
11.8 <referential constraint definition>. 870
11.9 <check constraint definition>. 876
11.10 <alter table statement>. 878
11.11 <add column definition>. 879
11.12 <alter column definition>. 881
11.13 <set column default clause>. 883
11.14 <drop column default clause>. 884
11.15 <set column not null clause>. 885
11.16 <drop column not null clause>. 886
11.17 <add column scope clause>. 887
11.18 <drop column scope clause>. 888
11.19 <alter column data type clause>. 890
11.20 <alter identity column specification>. 894
11.21 <drop identity property clause>. 896
11.22 <drop column generation expression clause>. 897
11.23 <drop column definition>. 898
11.24 <add table constraint definition>. 900
11.25 <alter table constraint definition>. 901
11.26 <drop table constraint definition>. 902
11.27 <add table period definition>. 905
11.28 <drop table period definition>. 908
11.29 <add system versioning clause>. 913
11.30 <drop system versioning clause>. 914
11.31 <drop table statement>. 916
11.32 <view definition>. 919
11.33 <drop view statement>. 930
11.34 <domain definition>. 933
11.35 <alter domain statement>. 935
11.36 <set domain default clause>. 936

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

11.37 <drop domain default clause>. . . . 937
11.38 <add domain constraint definition>. . . . 938
11.39 <drop domain constraint definition>. . . . 939
11.40 <drop domain statement>. . . . 940
11.41 <character set definition>. . . . 942
11.42 <drop character set statement>. . . . 944
11.43 <collation definition>. . . . 946
11.44 <drop collation statement>. . . . 948
11.45 <transliteration definition>. . . . 950
11.46 <drop transliteration statement>. . . . 952
11.47 <assertion definition>. . . . 954
11.48 <drop assertion statement>. . . . 956
11.49 <trigger definition>. . . . 959
11.50 <drop trigger statement>. . . . 965
11.51 <user-defined type definition>. . . . 968
11.52 <attribute definition>. . . . 984
11.53 <alter type statement>. . . . 986
11.54 <add attribute definition>. . . . 987
11.55 <drop attribute definition>. . . . 989
11.56 <add original method specification>. . . . 991
11.57 <add overriding method specification>. . . . 996
11.58 <drop method specification>. . . . 1001
11.59 <drop data type statement>. . . . 1005
11.60 <SQL-invoked routine>. . . . 1008
11.61 <alter routine statement>. . . . 1037
11.62 <drop routine statement>. . . . 1040
11.63 <user-defined cast definition>. . . . 1042
11.64 <drop user-defined cast statement>. . . . 1044
11.65 <user-defined ordering definition>. . . . 1047
11.66 <drop user-defined ordering statement>. . . . 1050
11.67 <transform definition>. . . . 1053
11.68 <alter transform statement>. . . . 1056
11.69 <add transform element list>. . . . 1057
11.70 <drop transform element list>. . . . 1059
11.71 <drop transform statement>. . . . 1061
11.72 <sequence generator definition>. . . . 1064
11.73 <alter sequence generator statement>. . . . 1066
11.74 <drop sequence generator statement>. . . . 1067

12 Access control. . . . 1068

12.1 <grant statement>. . . . 1068
12.2 <grant privilege statement>. . . . 1073
12.3 <privileges>. . . . 1076
12.4 <role definition>. . . . 1080
12.5 <grant role statement>. . . . 1081
12.6 <drop role statement>. . . . 1083
12.7 <revoke statement>. . . . 1084
12.8 Grantor determination. . . . 1103

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

13 SQL-client modules. 1105
13.1 <SQL-client module definition>. 1105
13.2 <module name clause>. 1109
13.3 <externally-invoked procedure>. 1110
13.4 <SQL procedure statement>. 1125
13.5 Data type correspondences. 1128
14 Data manipulation. 1139
14.1 <declare cursor>. 1139
14.2 <cursor properties>. 1141
14.3 <cursor specification>. 1143
14.4 <open statement>. 1146
14.5 <fetch statement>. 1147
14.6 <close statement>. 1151
14.7 <select statement: single row>. 1152
14.8 <delete statement: positioned>. 1156
14.9 <delete statement: searched>. 1158
14.10 <truncate table statement>. 1162
14.11 <insert statement>. 1164
14.12 <merge statement>. 1169
14.13 <update statement: positioned>. 1179
14.14 <update statement: searched>. 1181
14.15 <set clause list>. 1185
14.16 <temporary table declaration>. 1189
14.17 <free locator statement>. 1191
14.18 <hold locator statement>. 1192
15 Additional data manipulation rules. 1193
15.1 Effect of opening a cursor. 1193
15.2 Effect of receiving a result set. 1196
15.3 Determination of the current row of a cursor. 1197
15.4 Effect of closing a cursor. 1199
15.5 Evaluating a <set clause list>. 1201
15.6 Effect of a positioned delete. 1203
15.7 Effect of a positioned update. 1205
15.8 Effect of deleting rows from base tables. 1208
15.9 Effect of deleting some rows from a derived table. 1211
15.10 Effect of deleting some rows from a viewed table. 1213
15.11 Effect of inserting tables into base tables. 1215
15.12 Effect of inserting a table into a derived table. 1218
15.13 Effect of inserting a table into a viewed table. 1220
15.14 Effect of replacing rows in base tables. 1222
15.15 Effect of replacing some rows in a derived table. 1226
15.16 Effect of replacing some rows in a viewed table. 1229
15.17 Checking of views that specify CHECK OPTION. 1231
15.18 Execution of referential actions. 1234
15.19 Execution of BEFORE triggers. 1240
15.20 Execution of AFTER triggers. 1241
15.21 Execution of triggers. 1242

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

16 Control statements. 1245
16.1 <call statement>. 1245
16.2 <return statement>. 1246
17 Transaction management. 1247
17.1 <start transaction statement>. 1247
17.2 <set transaction statement>. 1249
17.3 <transaction characteristics>. 1251
17.4 <set constraints mode statement>. 1253
17.5 <savepoint statement>. 1255
17.6 <release savepoint statement>. 1256
17.7 <commit statement>. 1257
17.8 <rollback statement>. 1259
18 Connection management. 1261
18.1 <connect statement>. 1261
18.2 <set connection statement>. 1264
18.3 <disconnect statement>. 1266
19 Session management. 1268
19.1 <set session characteristics statement>. 1268
19.2 <set session user identifier statement>. 1270
19.3 <set role statement>. 1271
19.4 <set local time zone statement>. 1272
19.5 <set catalog statement>. 1273
19.6 <set schema statement>. 1274
19.7 <set names statement>. 1275
19.8 <set path statement>. 1276
19.9 <set transform group statement>. 1277
19.10 <set session collation statement>. 1278
20 Dynamic SQL. 1280
20.1 Description of SQL descriptor areas. 1280
20.2 <allocate descriptor statement>. 1290
20.3 <deallocate descriptor statement>. 1292
20.4 <get descriptor statement>. 1293
20.5 <set descriptor statement>. 1297
20.6 <copy descriptor statement>. 1302
20.7 <prepare statement>. 1305
20.8 <cursor attributes>. 1316
20.9 <deallocate prepared statement>. 1317
20.10 <describe statement>. 1318
20.11 <input using clause>. 1324
20.12 <output using clause>. 1328
20.13 <execute statement>. 1333
20.14 <execute immediate statement>. 1335
20.15 <dynamic declare cursor>. 1336
20.16 <descriptor value constructor>. 1337
20.17 <allocate extended dynamic cursor statement>. 1339
20.18 <allocate received cursor statement>. 1341

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

20.19 <dynamic open statement>. . . . 1343
20.20 <dynamic fetch statement>. . . . 1345
20.21 <dynamic single row select statement>. . . . 1346
20.22 <dynamic close statement>. . . . 1347
20.23 <dynamic delete statement: positioned>. . . . 1348
20.24 <dynamic update statement: positioned>. . . . 1349
20.25 <preparable dynamic delete statement: positioned>. . . . 1351
20.26 <preparable dynamic cursor name>. . . . 1353
20.27 <preparable dynamic update statement: positioned>. . . . 1355
20.28 <pipe row statement>. . . . 1357

21 Embedded SQL. . . . 1359
21.1 <embedded SQL host program>. . . . 1359
21.2 <embedded exception declaration>. . . . 1370
21.3 <embedded SQL Ada program>. . . . 1374
21.4 <embedded SQL C program>. . . . 1382
21.5 <embedded SQL COBOL program>. . . . 1391
21.6 <embedded SQL Fortran program>. . . . 1398
21.7 <embedded SQL MUMPS program>. . . . 1405
21.8 <embedded SQL Pascal program>. . . . 1409
21.9 <embedded SQL PL/I program>. . . . 1415

22 Direct invocation of SQL. . . . 1422
22.1 <direct SQL statement>. . . . 1422
22.2 <direct select statement: multiple rows>. . . . 1425

23 Diagnostics management. . . . 1426
23.1 <get diagnostics statement>. . . . 1426
23.2 Pushing and popping the diagnostics area stack. . . . 1442

24 Status codes. . . . 1443
24.1 SQLSTATE. . . . 1443
24.2 Remote Database Access SQLSTATE Subclasses. . . . 1453

25 Conformance. . . . 1454
25.1 Claims of conformance to SQL/Foundation. . . . 1454
25.2 Additional conformance requirements for SQL/Foundation. . . . 1455
25.3 Implied feature relationships of SQL/Foundation. . . . 1455

Annex A (informative) **SQL conformance summary. . . . 1466**
Annex B (informative) **Implementation-defined elements. . . . 1555**
Annex C (informative) **Implementation-dependent elements. . . . 1606**
Annex D (informative) **SQL optional feature taxonomy. . . . 1624**
Annex E (informative) **Deprecated features. . . . 1640**
Annex F (informative) **Incompatibilities with ISO/IEC 9075:2016. . . . 1641**
Annex G (informative) **Defect Reports not addressed in this edition of this document. . . . 1642**
Annex H (informative) **SQL mandatory feature taxonomy. . . . 1643**
Bibliography. . . . 1660
Index. . . . 1662

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

Tables

Table

Page

1 Overview of character sets. . . . 30
2 Fields in datetime values. . . . 37
3 Datetime data type conversions. . . . 38
4 Fields in year-month INTERVAL values. . . . 39
5 Fields in day-time INTERVAL values. . . . 40
6 Valid values for fields in INTERVAL values. . . . 40
7 Valid operators involving datetimes and intervals. . . . 41
8 Result with ONE ROW PER MATCH. . . . 81
9 Result with ALL ROWS PER MATCH. . . . 82
10 Schematic diagram of effective parameter lists of PTF component procedures. . . . 118
11 SQL-transaction isolation levels and the three phenomena. . . . 152
12 Interpretation of datetime components. . . . 189
13 Valid values for datetime fields. . . . 214
14 Valid absolute values for interval fields. . . . 215
15 Truth table for the AND Boolean operator. . . . 373
16 Truth table for the OR Boolean operator. . . . 373
17 Truth table for the IS Boolean operator. . . . 373
18 <null predicate> semantics. . . . 537
19 SQL/JSON and ECMAScript correspondences. . . . 731
20 Standard programming languages. . . . 792
21 Data type correspondences for Ada. . . . 1128
22 Data type correspondences for C. . . . 1130
23 Data type correspondences for COBOL. . . . 1131
24 Data type correspondences for Fortran. . . . 1133
25 Data type correspondences for M. . . . 1134
26 Data type correspondences for Pascal. . . . 1136
27 Data type correspondences for PL/I. . . . 1137
28 Data types of <key word>s used in the header of SQL descriptor areas. . . . 1284
29 Data types of <key word>s used in SQL item descriptor areas. . . . 1284
30 Codes used for SQL data types in Dynamic SQL. . . . 1286
31 Codes associated with datetime data types in Dynamic SQL. . . . 1287
32 Codes used for <interval qualifier>s in Dynamic SQL. . . . 1288
33 Codes used for input/output SQL parameter modes in Dynamic SQL. . . . 1288
34 Codes associated with user-defined types in Dynamic SQL. . . . 1289
35 Codes associated with sort direction. . . . 1289
36 Codes associated with null ordering. . . . 1289
37 Data types of <statement information item name>s. . . . 1428
38 Data types of <condition information item name>s. . . . 1428
39 SQL-statement codes. . . . 1430
40 SQLSTATE class and subclass codes. . . . 1444
41 SQLSTATE class codes for RDA. . . . 1453
42 Implied feature relationships of SQL/Foundation. . . . 1455
A.1 Feature definitions outside of Conformance Rules. . . . 1466
D.1 Feature taxonomy for optional features. . . . 1624
H.1 Feature taxonomy and definition for mandatory features. . . . 1643

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub

Full document details: ISO IEC 09075-2-2023

Figures

Figure

Page

1 Operation of <regular expression substring function>. 24
2 Illustration of WIDTH_BUCKET Semantics. 35
3 Illustration of important concepts in example query. 81
4 Taxonomy of SQL-invoked routines. 110
5 Flow of information during the invocation of a polymorphic table function. 117
6 Architecture of SQL/JSON path language usage. 167
7 Diagram of COLTREE. 443
8 Diagram of a plan tree. 444

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

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 sixth edition cancels and replaces the fifth edition (ISO/IEC 9075-2:2016), which has been technically revised. It also incorporates the Technical Corrigenda ISO/IEC 9075-2:2016/Cor.1:2019 and ISO/IEC 9075-2:2016/Cor.2:2022.

The main changes are as follows:

— native JSON datatype;

— more types of numeric literals;

— additional SQL functions:

  — ANY_VALUE,
  — BTRIM,
  — GREATEST,
  — JSON_SCALAR,
  — JSON_SERIALIZE,

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

  - LEAST,
  - LPAD,
  - LTRIM,
  - RPAD,
  - RTRIM;

- improve the presentation and accuracy of the summaries of implementation-defined and implementation-dependent aspects of this document;

- introduction of several digital artifacts;

- alignment with updated ISO house style and other guidelines for creating standards.

This sixth edition of ISO/IEC 9075-2 is designed to be used in conjunction with the following editions of other parts of the ISO/IEC 9075 series, all published 2023:

- ISO/IEC 9075-1, 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 9075-13, fifth edition;

- ISO/IEC 9075-14, sixth edition;

- ISO/IEC 9075-15, second edition;

- ISO/IEC 9075-16, first 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 09075-2-2023

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 SQL.

5) Clause 5, "Lexical elements", defines the lexical elements of the language.

6) Clause 6, "Scalar expressions", defines the elements of the language that produce scalar values.

7) Clause 7, "Query expressions", defines the elements of the language that produce rows and tables of data.

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

9) Clause 9, "Additional common rules", specifies the rules for assignments that retrieve data from or store data into SQL-data, and formation rules for set operations.

10) Clause 10, "Additional common elements", defines additional language elements that are used in various parts of the language.

11) Clause 11, "Schema definition and manipulation", defines facilities for creating and managing a schema.

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

13) Clause 13, "SQL-client modules", defines SQL-client modules and externally-invoked procedures.

14) Clause 14, "Data manipulation", defines the data manipulation statements.

15) Clause 15, "Additional data manipulation rules", defines additional rules for data manipulation.

16) Clause 16, "Control statements", defines the SQL-control statements.

17) Clause 17, "Transaction management", defines the SQL-transaction management statements.

18) Clause 18, "Connection management", defines the SQL-connection management statements.

19) Clause 19, "Session management", defines the SQL-session management statements.

20) Clause 20, "Dynamic SQL", defines the SQL dynamic statements.

21) Clause 21, "Embedded SQL", defines the host language embeddings.

22) Clause 22, "Direct invocation of SQL", defines direct invocation of SQL language.

23) Clause 23, "Diagnostics management", defines the diagnostics management facilities.

24) Clause 24, "Status codes", defines values that identify the status of the execution of SQL-statements and the mechanisms by which those values are returned.

25) Clause 25, "Conformance", defines the criteria for conformance to this document.

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

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

27) 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.

28) 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.

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

30) 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.

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

32) 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.

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

In the text of this document, in Clause 5, "Lexical elements", through Clause 24, "Status codes", 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 09075-2-2023

Information technology — Database language SQL —

Part 2:
Foundation (SQL/Foundation)

1 Scope

This document defines the data structures and basic operations on SQL-data. It provides functional capabilities for creating, accessing, maintaining, controlling, and protecting SQL-data.

This document specifies the syntax and semantics of a database language:

— for specifying and modifying the structure and the integrity constraints of SQL-data;

— for declaring and invoking operations on SQL-data and cursors;

— for declaring database language procedures;

— for embedding SQL-statements in a compilation unit that is otherwise written in a particular programming language (host language);

— for deriving an equivalent compilation unit in the host language. In that equivalent compilation unit, each embedded SQL-statement has been replaced by one or more statements in the host language, some of which invoke an SQL externally-invoked procedure that, when executed, has an effect equivalent to executing the SQL-statement;

— for direct invocation of SQL-statements;

— to support dynamic preparation and execution of SQL-statements.

This document provides a vehicle for portability of data definitions and compilation units between SQL-implementations.

This document provides a vehicle for interconnection of SQL-implementations.

Implementations of this document can exist in environments that also support application programming languages, end-user query languages, report generator systems, data dictionary systems, program library systems, and distributed communication systems, as well as various tools for database design, data administration, and performance optimization.

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

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. [1]

ISO/IEC 646, *Information technology — ISO 7-bit coded character set for information interchange*

ISO/IEC 1539-1:2018, *Information technology — Programming languages — Fortran — Part 1: Base language*

ISO 1989:2014, *Information technology — Programming languages — COBOL*

ISO 6160:1979, *Programming languages — PL/I (Endorsement of ANSI X3.53-1976)*

ISO/IEC 6429, *Information technology — Control functions for coded character sets*

ISO 7185:1990, *Information technology — Programming languages — Pascal*

ISO 8601-1:2019, *Date and time — Representations for information interchange — Part 1: Basic rules*

ISO/IEC 8652:2012, *Information technology — Programming languages — Ada*

ISO/IEC 8652:2012/Cor.1:2016, *Information technology — Programming languages — Ada — Technical Corrigendum 1*

ISO/IEC 8859-1, *Information technology — 8-bit single-byte coded graphic character sets — Part 1: Latin alphabet No. 1*

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

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

ISO/IEC 9579, *Information technology — Remote database access for SQL with security enhancement*

ISO/IEC 9899:2018, *Information technology — Programming languages — C*

ISO/IEC 10206:1991, *Information technology — Programming languages — Extended Pascal*

ISO/IEC 10646:2020, *Information technology — Universal Multi-Octet Coded Character Set (UCS)*

ISO/IEC 11756:1999, *Information technology — Programming languages — M*

ISO/IEC 14651:2020, *Information technology — International string ordering and comparison — Method for comparing character strings and description of the common template tailorable ordering*

ECMA International. *ECMA-262 — ECMAScript® Language Specification 5.1 Edition* [online]. [Place of publication unknown]: Available at `https://262.ecma-international.org/5.1/-ECMA-262.pdf`

1 In this document, ECMAScript Language Specification 5.1 Edition is referenced for the purpose of defining the lexical elements of the SQL/JSON path language specified in Subclause 9.45, "SQL/JSON path language: lexical elements", and Subclause 9.46, "SQL/JSON path language: syntax and semantics". There are no intentions to update this reference to a newer edition of ECMA-262.

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

Internet Engineering Task Force (IETF) RFC 8259 *The JavaScript Object Notation (JSON) Data Interchange Format*. Edited by: Miller, Matthew December 2018
Available at: https://datatracker.ietf.org/doc/rfc8259/

The Unicode Consortium. *Unicode Regular Expressions* [online]. 21. Mountain View, California, USA: The Unicode Consortium, 2020-06-17 . Available at https://www.unicode.org/reports/tr18/-tr18-21.html

W3C XQuery and XPath Functions and Operators 3.1 *XQuery and XPath Functions and Operators 3.1, W3C Recommendation*. Edited by: Malhotra, Ashok et al. 21 March 2017
Available at: https://www.w3.org/TR/xpath-functions/

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 9075-1 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 Definitions taken from ISO/IEC 10646:2020

3.1.1
character encoding form
form that determines how each UCS *code point* (3.1.2) for a UCS character is to be expressed as one or more code units used by the encoding form

[SOURCE: ISO/IEC 10646:2020, 3.22]

3.1.2
code point
value in the UCS codespace

[SOURCE: ISO/IEC 10646:2020, 3.9]

3.1.3
code unit
minimal bit combination that can represent a unit of encoded text for processing or interchange

[SOURCE: ISO/IEC 10646:2020, 3.10]

3.1.4
control character
control function the coded representation of which consists of a single *code point* (3.1.2)

[SOURCE: ISO/IEC 10646:2020, 3.17]

3.1.5
noncharacter code point
code point (3.1.2) that consists of FDD0-FDEF and any *code point* (3.1.2) ending in the value FFFE or FFFF

[SOURCE: ISO/IEC 10646:2020, 7.3.7]

3.1.6
normalization form
mechanisms allowing the selection of a unique coded representation among alternative but equivalent coded text representations of the same text

[SOURCE: ISO/IEC 10646:2020, 22]

3.1.7
repertoire
specified set of characters that are represented in a coded character set

[SOURCE: ISO/IEC 10646:2020, 3.46]

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

3.2 Definitions taken from ISO/IEC 14651:2020

3.2.1
collation
process by which, given two strings, it is determined whether the first one is less than, equal to, or greater than the second one

[SOURCE: ISO/IEC 14651:2020, 3.7]

3.3 Definitions taken from ISO 8601-1:2019

3.3.1
UTC
Coordinated Universal Time
Time scale with the same rate as International Atomic Time (TAI), but differing from TAI only by an integral number of seconds

[SOURCE: ISO 8601-1:2019, 3.1.1.12]

3.3.2
date
time on the calendar time scale

[SOURCE: ISO 8601-1:2019, 3.1.1.1]

3.3.3
calendar day of month
ordinal number of a calendar day within a calendar month

[SOURCE: ISO 8601-1:2019, 3.1.2.13]

3.3.4
calendar day of year
ordinal number of a calendar day within a calendar year

[SOURCE: ISO 8601-1:2019, 3.1.2.14]

3.3.5
calendar month
time scale unit resulting from a defined division of a calendar year, each containing a specific number of calendar days

[SOURCE: ISO 8601-1:2019, 3.1.2.19]

3.3.6
calendar year
time scale unit defined by the calendar system

[SOURCE: ISO 8601-1:2019, 3.1.2.21]

3.3.7
clock hour
time scale unit whose duration is one hour

[SOURCE: ISO 8601-1:2019, 3.1.2.6]

3.3.8
clock minute
time scale unit whose duration is one minute

[SOURCE: ISO 8601-1:2019, 3.1.2.4]

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

3.3 Definitions taken from ISO 8601-1:2019

3.3.9
clock second
time scale unit whose duration is one second

[SOURCE: ISO 8601-1:2019, 3.1.2.2]

3.3.10
day
duration of a calendar day

[SOURCE: ISO 8601-1:2019, 3.1.2.10]

3.3.11
Gregorian calendar
calendar in general use that defines a calendar year that closely approximates the tropical year

[SOURCE: ISO 8601-1:2019, 3.1.1.19]

3.3.12
hour
duration of 60 minutes

[SOURCE: ISO 8601-1:2019, 3.1.2.5]

3.3.13
leap second
intentional time step of one second to adjust UTC to ensure appropriate agreement with UT1, a time scale based on the rotation of the Earth

[SOURCE: ISO 8601-1:2019, 3.1.1.24]

3.3.14
minute
duration of 60 seconds

[SOURCE: ISO 8601-1:2019, 3.1.2.3]

3.3.15
month
duration of a calendar month

[SOURCE: ISO 8601-1:2019, 3.1.2.18]

3.3.16
second
base unit of duration measurement in the International System of Units (SI)

[SOURCE: ISO 8601-1:2019, 3.1.2.1]

3.3.17
year
duration of a calendar year

[SOURCE: ISO 8601-1:2019, 3.1.2.20]

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

3.4 Definitions taken from XQuery and XPath Functions and Operators 3.1

3.4.1
XQuery option flag
valid value of the `$flags` argument of `fn:matches`, to set options for the interpretation of the regular expression

[SOURCE: XQuery and XPath Functions and Operators 3.1, 5.6.2]

3.4.2
XQuery regular expression
regular expression syntax used by functions as defined in terms of the regular expression syntax specified in XML Schema

[SOURCE: XQuery and XPath Functions and Operators 3.1, 5.6.1]

3.4.3
XQuery regular expression parenthesized subexpression
regular expression parenthesized subexpression

[SOURCE: XQuery and XPath Functions and Operators 3.1, 5.6.1.3]

3.4.4
XQuery replacement string
valid value of the `$replacement` argument of `fn:replace`

[SOURCE: XQuery and XPath Functions and Operators 3.1, 5.6.4]

3.5 Definitions provided in this document

3.5.1
assignable, adj.
<of data types, taken pairwise> characteristic of a data type *T1* that permits a value of *T1* to be assigned to a site of a specified data type *T2*, where *T1* and *T2* may be the same data type

3.5.2
assignment
operation whose effect is to ensure that the value at a site *T* (known as the *target*) is identical to a given value *S* (known as the *source*)

Note 1 to entry: Assignment is frequently indicated by the use of the phrase "*T* is set to *S*" or "the value of *T* is set to *S*".

3.5.3
attribute
component of a *structured type* (3.5.68)

Note 1 to entry: Each value *V* in structured type *T* has exactly one attribute value for each attribute *A* of *T*. The characteristics of an attribute are specified by an attribute descriptor. The value of an attribute may be retrieved as the result of the invocation *A*(*V*) of the *observer function* (3.5.38) for that attribute.

3.5.4
cardinality
<of a collection> number of elements in that collection

Note 1 to entry: Those elements need not necessarily have distinct values. The objects to which this concept applies include tables and the values of collection types.

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

3.5.5
comparable, adj.
<of a pair of values> capable of being compared

Note 1 to entry: In most, but not all, cases, the values of a data type can be compared one with another. For the specification of comparability of individual data types, see Subclause 4.3, "Character strings", through Subclause 4.12, "Collection types".

3.5.6
constructor function
niladic (3.5.37) *SQL-invoked function* (3.5.62) of which exactly one is implicitly specified for every *structured type* (3.5.68)

Note 1 to entry: An invocation of the constructor function for data type *T* returns a value *V* of the most specific type of *T* such that *V* is not the null value and, for every *observer function* (3.5.38) *O* defined for *T*, the invocation *O*(*V*) returns the default value of the *attribute* (3.5.3) corresponding to *O*.

3.5.7
data model
<general> definition of the kinds of data that belong to a particular universe of discourse, including the operations on those kinds of data

3.5.8
declared type
<of an expression denoting a value or anything that can be referenced to denote a value, such as, for example, a parameter, column, or variable > unique data type that is common to every value that might result from evaluation of that expression

3.5.9
distinct, adj.
<of a pair of comparable values> capable of being distinguished within a given context

Note 1 to entry: Informally, two values are distinct if neither is null and the values are not equal. A null value and a non-null value are distinct. Two null values are not distinct. See Subclause 4.2.5, "Properties of distinct", and the General Rules of Subclause 8.15, "<distinct predicate>".

3.5.10
distinct type
user-defined type (3.5.76) derived from a predefined type or a collection type

3.5.11
duplicates
two or more members of a multiset that are not distinct

3.5.12
dyadic, adj.
<of operators, functions, and procedures> having exactly two operands or parameters

Note 1 to entry: An example of a dyadic operator in this document is "−", specifying the subtraction of the right operand from the left operand. An example of a dyadic function is POSITION.

3.5.13
element type
<of a collection type> declared type *DT* specified in the definition of a collection type *CT*

Note 1 to entry: The declared type of every element of every value of type *CT* is *DT*.

3.5.14
empty row pattern match
row pattern match (3.5.51) that matches no rows

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

3.5.15
external routine
SQL-invoked routine (3.5.64) whose routine body is an external body reference that identifies a program written in a programming language other than SQL

3.5.16
fixed-length, adj.
characteristic of the declared type of sites whose types are character string types or binary string types that restricts values in those sites to contain exactly one number of characters or octets, respectively, known as the length in characters or octets, respectively, of the site

3.5.17
fully qualified, adj.
<of a name of some SQL object> with all optional components specified explicitly

Note 1 to entry: A fully qualified name does not necessarily identify an object uniquely. For example, although a fully qualified specific name, consisting of a catalog name, a schema name and a specific name, uniquely identifies a routine, a fully qualified routine name does not necessarily do so.

3.5.18
identical, adj.
<of a pair of values> indistinguishable, in the sense that it is impossible, by any means specified in this document, to detect any difference between them

Note 1 to entry: For the full definition, see Subclause 9.10, "Determination of identical values".

3.5.19
interface
<of an *user-defined type* (3.5.76)> set comprising every function such that the declared type of at least one of its parameters or its result is that *user-defined type* (3.5.76)

3.5.20
JSON array
structure represented by a "[", zero or more elements separated by ",", and "]"

3.5.21
JSON Boolean
JSON literal (3.5.24) "**true**" or *JSON literal* (3.5.24) "**false**"

3.5.22
JSON element
JSON text (3.5.30) fragment that is a *JSON value* (3.5.33) in a *JSON array* (3.5.20)

3.5.23
JSON data model
implicit data model associated with JSON

3.5.24
JSON literal
JSON text fragment (3.5.31) that is any of the key words "true", "false", or "null"

3.5.25
JSON member
JSON string (3.5.29) followed by a colon followed by a *JSON value* (3.5.33) in a *JSON object* (3.5.28)

Note 1 to entry: This is also known as a "name-value pair"; the name is sometimes called a "key" and the second value is sometimes called a "bound value".

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

3.5.26
JSON null
JSON literal (3.5.24) "null"

Note 1 to entry: A JSON null is distinct from an SQL null value and from an SQL/JSON null.

3.5.27
JSON number
Unicode character string comprising an integer part, optionally followed by a fractional part and/or an exponent part

3.5.28
JSON object
structure represented by a "{", zero or more members separated by ",", and "}"

3.5.29
JSON string
Unicode character string

Note 1 to entry: Some characters must be "escaped" by preceding them with a reverse solidus ("\"), while any or all characters can be represented in "Unicode notation" comprising the string "\u" followed by four hexadecimal digits or two such strings representing the UTF-16 surrogate pairs representing characters not on the Basic Multilingual Plane (strings are surrounded by double-quote characters, which are not part of the value of the strings).

Note 2 to entry: This applies only to *JSON tokens* (3.5.32) in *JSON text* (3.5.30).

3.5.30
JSON text
sequence of *JSON tokens* (3.5.32), which must be encoded in Unicode (UTF-8 by default); insignificant whitespace may be used anywhere in *JSON text* (3.5.30) except within strings (where all whitespace is significant), numbers, and literals

3.5.31
JSON text fragment
substring of a *JSON text* (3.5.30) that conforms to any BNF non-terminal in RFC 8259

3.5.32
JSON token
one of six structural characters ("{", "}", "[", "]", ":", ","), *JSON strings* (3.5.29), *JSON numbers* (3.5.27), and *JSON literals* (3.5.24)

3.5.33
JSON value
JSON object (3.5.28), *JSON array* (3.5.20), *JSON number* (3.5.27), *JSON string* (3.5.29), or one of three *JSON literals* (3.5.24)

3.5.34
monadic, adj.
<of operators, functions, and procedures> having exactly one operand or parameter

Note 1 to entry: An example of a monadic arithmetic operator in this document is "−", specifying the negation of the operand. An example of a monadic function is CHARACTER_LENGTH, specifying the length in characters of the argument.

3.5.35
most specific type
<of a value> unique data type of which every data type of that value is a supertype

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

3.5.36
mutator function
dyadic, type-preserving *SQL-invoked function* (3.5.62) implicitly defined by the definition of an *attribute* (3.5.3) of a *structured type* (3.5.68) that, when invoked, modifies the value of the *attribute* (3.5.3) with which it is associated

3.5.37
niladic, adj.
<of functions and procedures> having no parameters

3.5.38
observer function
monadic *SQL-invoked function* (3.5.62) associated with an *attribute* (3.5.3) of a *structured type* (3.5.68) that, when invoked, returns the value of the *attribute* (3.5.3) with which it is associated

Note 1 to entry: The observer function is implicitly defined by the definition of the attribute.

3.5.39
primary row pattern variable
row pattern variable (3.5.58) that appears in a <row pattern>

3.5.40
redundant duplicates
all except one of any collection of duplicate values or rows

3.5.41
REF value
value that references some site

3.5.42
reference type
data type all of whose values are potential references to sites of one specified data type

3.5.43
referenced type
declared type of the values at sites referenced by values of a particular *reference type* (3.5.42)

3.5.44
result data type
declared type of the result of an *SQL-invoked function* (3.5.62)

3.5.45
result set
sequence of rows specified by a <cursor specification> that is brought into existence by opening a cursor and ranged over by that cursor

3.5.46
result set sequence
sequence of returned *result sets* (3.5.45)

3.5.47
result SQL parameter
SQL parameter (3.5.66), the *most specific type* (3.5.35) of the value of which determines the *most specific type* (3.5.35) of the result of the *SQL-invoked function* (3.5.62) of which it is a parameter

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

3.5.48
returned result set
result set (3.5.45) created during execution of an *SQL-invoked procedure* (3.5.63) and not destroyed when that execution terminates

Note 1 to entry: Such a *result set* (3.5.45) can be accessed by using a cursor other than the one that brought it into existence (a received cursor).

3.5.49
row pattern
form of regular expressions that specifies criteria (patterns) used to identify collections of rows in a table based on their relationships to one another

3.5.50
row pattern input table
table that is the input operand of a <row pattern recognition clause> or of a <window definition> that specifies window row pattern recognition

3.5.51
row pattern match
sequence of consecutive rows in a *row pattern partition* (3.5.57) that collectively meet the match criteria of a <row pattern common syntax>

3.5.52
row pattern matching
process of identifying collections of rows in a table based on the patterns specified in *row patterns* (3.5.49)

3.5.53
row pattern measure
named scalar expression defined with respect to the *row pattern variables* (3.5.58)

3.5.54
row pattern measure column
exported column of the *row pattern output table* (3.5.56), the value for which is specified by a *row pattern measure* (3.5.53)

3.5.55
row pattern measure function
window function whose value is specified by a *row pattern measure* (3.5.53)

3.5.56
row pattern output table
table that is the result of a <row pattern recognition clause> applied to its *row pattern input table* (3.5.50)

3.5.57
row pattern partition
maximal collection of rows of the *row pattern input table* (3.5.50) in which all rows have non-distinct values for a set of one or more columns, called the *row pattern partitioning columns*; or a window partition in a window whose descriptor has window row pattern recognition

3.5.58
row pattern variable
primary row pattern variable (3.5.39) or an *union row pattern variable* (3.5.74)

3.5.59
savepoint
point within an SQL-transaction, identified by a savepoint name, to which that SQL-transaction may be restored

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

3.5.60
signature
<of an *SQL-invoked routine* (3.5.64)> name of an *SQL-invoked routine* (3.5.64), the position and declared type of each of its *SQL parameters* (3.5.66), and an indication of whether it is an *SQL-invoked function* (3.5.62) or an *SQL-invoked procedure* (3.5.63)

3.5.61
SQL argument
expression denoting a value to be substituted for an *SQL parameter* (3.5.66) in an invocation of an *SQL-invoked routine* (3.5.64)

3.5.62
SQL-invoked function
function that is allowed to be invoked only from within SQL

3.5.63
SQL-invoked procedure
procedure that is allowed to be invoked only from within SQL

3.5.64
SQL-invoked routine
routine that is allowed to be invoked only from within SQL

3.5.65
SQL/JSON data model
data model created for operating on JSON data within the SQL language

3.5.66
SQL parameter
parameter declared as part of the signature of an *SQL-invoked routine* (3.5.64)

3.5.67
SQL routine
SQL-invoked routine (3.5.64) whose routine body is written in SQL

3.5.68
structured type
user-defined type (3.5.76) specified as a set of *attributes* (3.5.3)

3.5.69
subfield
<of a row type> field that is a field of a row type *RT* or a field of a row type *RT2* that is the declared type of a field that is a subfield of *RT*

3.5.70
subtype
<of a data type> data type *T2* such that every value of *T2* is also a value of data type *T1*

Note 1 to entry: If *T1* and *T2* are not compatible, then *T2* is a *proper subtype* of *T1*. "Compatible" is defined in Subclause 4.2.4, "Data type terminology". See also *supertype* (3.5.71).

3.5.71
supertype
<of a data type> data type *T1* such that every value of *T2* is also a value of data type *T1*

Note 1 to entry: If *T1* and *T2* are not compatible, then *T1* is a *proper supertype* of *T2*. "Compatible" is defined in Subclause 4.2.4, "Data type terminology". See also *subtype* (3.5.70).

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

3.5.72
transliteration
method of translating characters in one character set into characters of the same or a different character set

3.5.73
type-preserving function
SQL-invoked function (3.5.62), one of whose parameters is a result *SQL parameter* (3.5.66)

Note 1 to entry: The *most specific type* (3.5.35) of the value returned by an invocation of a type-preserving function is identical to the *most specific type* (3.5.35) of the SQL argument value substituted for the result *SQL parameter* (3.5.66).

3.5.74
union row pattern variable
row pattern variable (3.5.58) that is defined as a union of *primary row pattern variables* (3.5.39) in a <row pattern subset clause>

3.5.75
universal row pattern variable
implementation-defined *union row pattern variable* (3.5.74) that is defined implicitly as the union of all *primary row pattern variables* (3.5.39)

3.5.76
user-defined type
data type whose interface is user-defined

3.5.77
variable-length, adj.
characteristic of the declared type of sites whose types are character string types or binary string types that allows values in those sites to contain any number of characters or octets, respectively, between 0 (zero) and some maximum number, known as the maximum length in characters or octets, respectively, of the site

3.5.78
with-return cursor
cursor that, when opened, creates a result set that is capable of becoming a returned result set

Note 1 to entry: The WITH RETURN option of <declare cursor>, <dynamic declare cursor>, and <allocate extended dynamic cursor statement> specifies a with-return cursor. WITH RETURN may also be specified in the content of an <attributes variable> in a <prepare statement>, to indicate that the prepared statement, when opened as a dynamic cursor, creates a with-return cursor.

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

4 Concepts

This Clause is modified by Clause 4, "Concepts", in ISO/IEC 9075-3.
This Clause is modified by Clause 4, "Concepts", in ISO/IEC 9075-4.
This Clause is modified by Clause 4, "Concepts", in ISO/IEC 9075-9.
This Clause is modified by Clause 4, "Concepts", in ISO/IEC 9075-10.
This Clause is modified by Clause 4, "Concepts", in ISO/IEC 9075-11.
This Clause is modified by Clause 4, "Concepts", in ISO/IEC 9075-13.
This Clause is modified by Clause 4, "Concepts", in ISO/IEC 9075-14.
This Clause is modified by Clause 4, "Concepts", in ISO/IEC 9075-15.
This Clause is modified by Clause 4, "Concepts", in ISO/IEC 9075-16.

4.1 Notations and conventions

This Subclause is modified by Subclause 4.1, "Notations and conventions", in ISO/IEC 9075-3.
This Subclause is modified by Subclause 4.1, "Notations and conventions", in ISO/IEC 9075-4.
This Subclause is modified by Subclause 4.1, "Notations and conventions", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 4.1, "Notations and conventions", in ISO/IEC 9075-10.
This Subclause is modified by Subclause 4.1, "Notations and conventions", in ISO/IEC 9075-11.
This Subclause is modified by Subclause 4.1, "Notations and conventions", in ISO/IEC 9075-13.
This Subclause is modified by Subclause 4.1, "Notations and conventions", in ISO/IEC 9075-14.
This Subclause is modified by Subclause 4.1, "Notations and conventions", in ISO/IEC 9075-15.
This Subclause is modified by Subclause 4.1, "Notations and conventions", in ISO/IEC 9075-16.

4.1.1 Notations

This Subclause is modified by Subclause 4.1.1, "Notations", in ISO/IEC 9075-3.
This Subclause is modified by Subclause 4.1.1, "Notations", in ISO/IEC 9075-4.
This Subclause is modified by Subclause 4.1.1, "Notations", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 4.1.1, "Notations", in ISO/IEC 9075-10.
This Subclause is modified by Subclause 4.1.1, "Notations", in ISO/IEC 9075-11.
This Subclause is modified by Subclause 4.1.1, "Notations", in ISO/IEC 9075-13.
This Subclause is modified by Subclause 4.1.1, "Notations", in ISO/IEC 9075-14.
This Subclause is modified by Subclause 4.1.1, "Notations", in ISO/IEC 9075-15.
This Subclause is modified by Subclause 4.1.1, "Notations", in ISO/IEC 9075-16.

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". See `https://standards.iso.org/iso-iec/9075/-2/ed-6/en/` to download digital artifacts for this document. To download the syntax defined in a plain-text format, select the file named `ISO_IEC_9075-2(E)_Foundation.bnf.txt`. To download the syntax defined in an XML format, select the file named `ISO_IEC_9075-2(E)_Foundation.bnf.xml`.

4.1.2 Use of terms

This Subclause is modified by Subclause 4.1.2, "Use of terms", in ISO/IEC 9075-4.

An SQL-statement *S1* is said to be executed as a *direct result of executing an SQL-statement* if *S1* is the SQL-statement contained in an <externally-invoked procedure> or <SQL-invoked routine> that has been executed.

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

04 An SQL-statement *S1* is said to be executed as a *direct result of executing an SQL-statement* if *S1* is the value of an <SQL statement variable> referenced by an <execute immediate statement> contained in an <externally-invoked procedure> that has been executed, or if *S1* was the value of the <SQL statement variable> that was associated with an <SQL statement name> by a <prepare statement> and that same <SQL statement name> is referenced by an <execute statement> contained in an <externally-invoked procedure> that has been executed.

4.2 Data types

This Subclause is modified by Subclause 4.2, "Data types", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 4.8, "Data types", in ISO/IEC 9075-13.
This Subclause is modified by Subclause 4.2, "Data types", in ISO/IEC 9075-14.
This Subclause is modified by Subclause 4.2, "Data types", in ISO/IEC 9075-15.

4.2.1 General introduction to data types

This Subclause is modified by Subclause 4.2.1, "General introduction to data types", in ISO/IEC 9075-15.

A data type is a set of representable values. Every representable value belongs to at least one data type and some belong to several data types. The physical representation of a value of a data type is implementation-dependent (UV060).

Exactly one of the data types of a value *V*, namely the most specific type of *V*, is a subtype of every data type of *V*. A <value expression> *E* has exactly one declared type, common to every possible result of evaluating *E*. Items that can be referenced by name, such as SQL parameters, columns, fields, attributes, and variables, also have declared types.

SQL supports three sorts of data types: *predefined data types*, *constructed types*, and *user-defined types*. Predefined data types are sometimes called "built-in data types", though not in this document. User-defined types can be defined by a standard, by an SQL-implementation, or by an application.

15 A constructed type is specified using data type constructors. The constructed types and the associated data type constructors are:

— 15 array type: ARRAY;

— multiset type: MULTISET;

— reference type: REF;

— row type: ROW.

15 Array types and multiset types are known generically as *collection types*. SQL defines a category of data types known as *collection types*. The collection types are:

— array type;

— multiset type.

Every predefined data type is a subtype of itself and of no other data types. It follows that every predefined data type is a supertype of itself and of no other data types. The predefined data types are individually described in each of Subclause 4.3, "Character strings", through Subclause 4.8, "JSON types".

Row types, reference types, and collection types are described in Subclause 4.10, "Row types", Subclause 4.11, "Reference types", Subclause 4.12, "Collection types", respectively.

A user-defined type is either a *distinct type* or a *structured type*. User-defined types are described in Subclause 4.9, "User-defined types".

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

4.2.2 Naming of predefined types

This Subclause is modified by Subclause 4.2.1, "Naming of predefined types", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 4.2.1, "Naming of predefined types", in ISO/IEC 9075-14.

09 14 SQL defines predefined data types named by the following <key word>s:

CHARACTER

CHARACTER VARYING

CHARACTER LARGE OBJECT

BINARY

BINARY VARYING

BINARY LARGE OBJECT

NUMERIC

DECIMAL

SMALLINT

INTEGER

BIGINT

FLOAT

REAL

DOUBLE PRECISION

DECFLOAT

BOOLEAN

DATE

TIME

TIMESTAMP

INTERVAL

JSON

These names are used in the *type designators* that constitute the *type precedence lists* specified in Subclause 9.7, "Type precedence list determination". In addition, TABLE is the type designator for the generic table parameter type, and DESCRIPTOR is the type designator for the descriptor parameter type.

NOTE 1 — The generic table parameter type and the descriptor parameter type only occur as parameter types, and only in the parameter lists of polymorphic table functions; they are not available as types for columns, domains, or parameters of non-polymorphic table functions.

09 14 For reference purposes:

— The data types CHARACTER, CHARACTER VARYING, and CHARACTER LARGE OBJECT are collectively referred to as *character string types* and the values of character string types are known as *character strings*.

— The data types BINARY, BINARY VARYING, and BINARY LARGE OBJECT are referred to as *binary string types* and the values of binary string types are referred to as *binary strings*.

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

— The data types CHARACTER LARGE OBJECT and BINARY LARGE OBJECT are collectively referred to as *large object string types* and the values of large object string types are referred to as *large object strings*.

— Character string types and binary string types are collectively referred to as *string types* and values of string types are referred to as *strings*.

— The data types NUMERIC, DECIMAL, SMALLINT, INTEGER, and BIGINT are collectively referred to as *exact numeric types*.

— The data types FLOAT, REAL, and DOUBLE PRECISION are collectively referred to as *approximate numeric types*.

— The data type DECFLOAT is referred to as the *decimal floating-point type*.

— Exact numeric types, approximate numeric types, and the decimal floating-point type are collectively referred to as *numeric types*. Values of numeric types are referred to as *numbers*.

— The data types TIME WITHOUT TIME ZONE and TIME WITH TIME ZONE are collectively referred to as *time types* (or, for emphasis, as time with or without time zone).

— The data types TIMESTAMP WITHOUT TIME ZONE and TIMESTAMP WITH TIME ZONE are collectively referred to as *timestamp types* (or, for emphasis, as timestamp with or without time zone).

— The data types DATE, TIME, and TIMESTAMP are collectively referred to as *datetime types*.

— Values of datetime types are referred to as *datetimes*.

— The data type INTERVAL is referred to as an *interval type*. Values of interval types are called *intervals*.

Each data type has an associated data type descriptor; the contents of a data type descriptor are determined by the specific data type that it describes. A data type descriptor includes an identification of the data type and all information needed to characterize a value of that data type.

Subclause 6.1, "<data type>", describes the semantic properties of each data type.

4.2.3 Host language data types

This Subclause is modified by Subclause 4.8.1, "Host language data types", in ISO/IEC 9075-13.

13 Each host language has its own data types, which are separate and distinct from SQL data types, even though similar names may be used to describe the data types. Mappings of SQL data types to data types in host languages are described in:

— Subclause 11.60, "<SQL-invoked routine>"

— Subclause 21.1, "<embedded SQL host program>"

Not every SQL data type has a corresponding data type in every host language.

4.2.4 Data type terminology

This Subclause is modified by Subclause 4.2.2, "Data type terminology", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 4.2.2, "Data type terminology", in ISO/IEC 9075-14.
This Subclause is modified by Subclause 4.2.2, "Data type terminology", in ISO/IEC 9075-15.

The notion of a *constituent* of a declared type *DT* is defined recursively as follows:

— *DT* is a constituent of *DT*;

— If *DT* is a row type, then the declared type of each field of *DT* is a constituent of *DT*;

— If *DT* is a collection type, then the element type of *DT* is a constituent of *DT*;

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

— Every constituent of a constituent of *DT* is a constituent of *DT*.

A data type *TY* is *usage-dependent* on a user-defined type *UDT* if and only if exactly one of the following is true:

— *TY* is *UDT*;

— *TY* is a reference type whose referenced type is *UDT*;

— *TY* is a row type, and the declared type of some field of *TY* is usage-dependent on *UDT*;

— *TY* is a collection type, and the declared element type of *TY* is usage-dependent on *UDT*.

A structured type *ST* is *directly based on* a data type *DT* if and only if at least one of the following is true:

— *DT* is the declared type of some attribute of *ST*;

— *DT* is a direct supertype of *ST*;

— *DT* is a direct subtype of *ST*;

— *DT* is compatible with *ST*.

A collection type *CT* is *directly based on* a data type *DT* if *DT* is the element type of *CT*.

A row type *RT* is *directly based on* a data type *DT* if *DT* is the declared type of some field (or the data type of the domain of some field) whose descriptor is included in the descriptor of *RT*.

A data type *DT1* is *based on* a data type *DT2* if *DT1* is compatible with *DT2*, *DT1* is directly based on *DT2*, or *DT1* is directly based on some data type that is based on *DT2*.

Two data types, *T1* and *T2*, are said to be *compatible* if *T1* is assignable to *T2*, *T2* is assignable to *T1*, and their descriptors include the same data type name. If they are row types, it shall further be the case that the declared types of their corresponding fields are pairwise compatible. If they are collection types, it shall further be the case that their element types are compatible. If they are reference types, it shall further be the case that their referenced types are compatible.

NOTE 2 — The data types "CHARACTER(*n*) CHARACTER SET *CS1*" and "CHARACTER(*m*) CHARACTER SET *CS2*", where *CS1* ≠ *CS2*, have descriptors that include the same data type name (CHARACTER), but are not mutually assignable; therefore, they are not compatible.

Ordering and comparison of values of the predefined data types requires knowledge only about those predefined data types. However, to be able to compare and order values of constructed types or of user-defined types, additional information is required. We say that some type *T* is *S*-ordered, for some set of types *S*, if, in order to compare and order values of type *T*, information about ordering at least one of the types in *S* is first required. A definition of *S*-ordered is required for several *S* (that is, for several sets of types), but not for all possible such sets.

The general definition of *S*-ordered is as follows.

Let *T* be a type and let *S* be a set of types. *T* is *S-ordered* if and only if exactly one of the following is true:

— *T* is a member of *S*;

— *T* is a row type and the declared type of some field of *T* is *S*-ordered;

— *T* is a collection type and the element type of *T* is *S*-ordered;

— *T* is a structured type whose comparison form is STATE and the declared type of some attribute of *T* is *S*-ordered;

— *T* is a user-defined type whose comparison form is MAP and the return type of the SQL-invoked function that is identified by the <map function specification> is *S*-ordered;

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

— *T* is a reference type with a derived representation and the declared type of some derivational attribute of the derived representation is *S*-ordered.

09 14 15 The notion of *S*-ordered is applied in the following definitions:

— A type *T* is *JSON-ordered* if *T* is *S*-ordered, where *S* is the set of JSON types.

— A type *T* is *LOB-ordered* if *T* is *S*-ordered, where *S* is the set of large object types.

— A type *T* is *array-ordered* if *T* is *S*-ordered, where *S* is the union of the set of array types and the set of distinct types whose source type is an array type.

— A type *T* is *multiset-ordered* if *T* is *S*-ordered, where *S* is the union of the set of multiset types and the set of distinct types whose source type is a multiset type.

— A type *T* is *row-ordered* if *T* is *S*-ordered, where *S* is the set of row types.

— A type *T* is *reference-ordered* if *T* is *S*-ordered, where *S* is the set of reference types.

— A type *T* is *DT-EC-ordered* if *T* is *S*-ordered, where *S* is the set of distinct types with EQUALS ONLY comparison form (DT-EC stands for "distinct type-equality comparison").

— A type *T* is *DT-FC-ordered* if *T* is *S*-ordered, where *S* is the set of distinct types with FULL comparison form.

— A type *T* is *DT-NC-ordered* if *T* is *S*-ordered, where *S* is the set of distinct types with no comparison form.

— A type *T* is *ST-EC-ordered* if *T* is *S*-ordered, where *S* is the set of structured types with EQUALS ONLY comparison form.

— A type *T* is *ST-FC-ordered* if *T* is *S*-ordered, where *S* is the set of structured types with FULL comparison form.

— A type *T* is *ST-NC-ordered* if *T* is *S*-ordered, where *S* is the set of structured types with no comparison form.

— A type *T* is *ST-ordered* if *T* is ST-EC-ordered, ST-FC-ordered, or ST-NC-ordered.

— A type *T* is *UDT-EC-ordered* if *T* is either DT-EC-ordered or ST-EC-ordered (UDT stands for "user-defined type").

— A type *T* is *UDT-FC-ordered* if *T* is either DT-FC-ordered or ST-FC-ordered

— A type *T* is *UDT-NC-ordered* if *T* is either DT-NC-ordered or ST-NC-ordered.

4.2.5 Properties of distinct

Two comparable values are distinct if they are capable of being distinguished within a given context.

Two null values are not distinct.

A null value and a non-null value are distinct.

Two non-null values are distinct if the General Rules of Subclause 8.15, "<distinct predicate>", return *True*.

The result of evaluating whether or not two comparable values are distinct is never *Unknown*. The result of evaluating whether or not two values that are not comparable (for example, values of a user-defined type that has no comparison type) are distinct is not defined.

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

4.3 Character strings

This Subclause is modified by Subclause 4.2, "Character strings", in ISO/IEC 9075-10.

4.3.1 Introduction to character strings

A character string is a sequence of characters. All the characters in a character string are taken from a single character set. A character string has a length, which is the number of characters in the sequence. The length is 0 (zero) or a positive integer.

A *character string type* is described by a character string type descriptor. A character string type descriptor contains:

— The name of the specific character string type (CHARACTER, CHARACTER VARYING, or CHARACTER LARGE OBJECT; NATIONAL CHARACTER, NATIONAL CHARACTER VARYING, and NATIONAL CHARACTER LARGE OBJECT are represented as CHARACTER, CHARACTER VARYING, and CHARACTER LARGE OBJECT, respectively);

— The length or maximum length in characters of the character string type;

— The catalog name, schema name, and character set name of the character set of the character string type;

— The catalog name, schema name, and collation name of the collation of the character string type.

A *character large object type* is a character string type where the name of the specific character string type is CHARACTER LARGE OBJECT. A value of a character large object type is a *large object character string*.

The character set of a character string type may be specified explicitly or implicitly.

The <key word>s NATIONAL CHARACTER are used to specify the character type with an implementation-defined (IV099) character set. Special syntax (`N'string'`) is provided for representing literals in that character set.

With two exceptions, a character string expression is assignable only to sites of a character string type whose character set is the same. The exceptions are as specified in Subclause 4.3.8, "Universal character sets", and such other cases as may be implementation-defined (IA014). If a store assignment would result in the loss of non-<truncating whitespace> characters due to truncation, then an exception condition is raised. If a retrieval assignment or evaluation of a <cast specification> would result in the loss of characters due to truncation, then a warning condition is raised.

Character sets fall into three categories: those defined by national or international standards, those defined by SQL-implementations, and those defined by applications. The character sets defined by ISO/IEC 10646:2020 are known as *Universal Character Sets* (UCS) and their treatment is described in Subclause 4.3.8, "Universal character sets". Every character set contains the <space> character (equivalent to U+0020). An application defines a character set by assigning a new name to a character set from one of the first two categories. They can be defined to "reside" in any schema chosen by the application. Character sets defined by standards or by SQL-implementations reside in the Information Schema (named INFORMATION_SCHEMA) in each catalog, as do collations defined by standards and collations, transliterations, and transcodings defined by SQL-implementations.

NOTE 3 — The Information Schema is defined in ISO/IEC 9075-11.

4.3.2 Comparison of character strings

Two character strings are comparable if and only if either they have the same character set or there exists at least one collation that is applicable to both their respective character sets (which is possible only if the character sets share the same repertoire).

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

NOTE 4 — There are syntactic possibilities in which a comparison operation has two equally worthy collations available to compare two character string operands. Such syntax is a syntax error, even though the operands could be compared if one or the other of the two collations was designated as the collation to use. For example, if *T.C1* is a column whose collation is *COLLATION1* and *T.C2* is a column whose collation is some other collation *COLLATION2*, then

```
T.C1 > T.C2
```

is a syntax error, whereas as

```
T.C1 > T.C2 COLLATE COLLATION2
```

is not a syntax error.

A *collation* is defined by ISO/IEC 14651:2020 as "a process by which two strings are determined to be in exactly one of the relationships of less than, greater than, or equal to one another". Each collation known in an SQL-environment is applicable to one or more character sets, and for each character set, one or more collations are applicable to it, one of which is associated with it as its *character set collation*.

Anything that has a declared type can, if that type is a character string type, be associated with a collation applicable to its character set; this is known as a *declared type collation*. Every declared type that is a character string type has a collation derivation, this being either *none*, *implicit*, or *explicit*. The collation derivation of a declared type with a declared type collation that is explicitly or implicitly specified by a <data type> is *implicit*. If the collation derivation of a declared type that has a declared type collation is not *implicit*, then it is *explicit*. The collation derivation of an expression of character string type that has no declared type collation is *none*.

An operation that explicitly or implicitly involves character string comparison is a *character comparison operation*. At least one of the operands of a character comparison operation shall have a declared type collation.

There may be an SQL-session collation for some or all of the character sets known to the SQL-implementation (see Subclause 4.45, "SQL-sessions").

The collation used for a particular character comparison is specified by Subclause 9.15, "Collation determination".

The comparison of two character string expressions depends on the collation used for the comparison (see Subclause 9.15, "Collation determination"). When values of unequal length are compared, if the collation for the comparison has the NO PAD characteristic and the shorter value is equal to some prefix of the longer value, then the shorter value is considered less than the longer value. If the collation for the comparison has the PAD SPACE characteristic, for the purposes of the comparison, the shorter value is effectively extended to the length of the longer by concatenation of <space>s on the right.

For every character set, there is at least one collation.

4.3.3 Operations involving character strings

4.3.3.1 Regular expression syntaxes

This document utilizes two syntaxes for regular expressions, a POSIX-based regular expression syntax and the XQuery regular expression syntax.

The POSIX-based regular expression syntax is similar to the syntax specified in ISO/IEC/IEEE 9945, but is normatively defined in the General Rules of Subclause 8.6, "<similar predicate>".

The XQuery regular expression syntax is normatively defined in XQuery and XPath Functions and Operators 3.1.

4.3.3.2 Operators that operate on character strings and return character strings

<concatenation operator> is an operator, | |, that returns the character string made by joining its character string operands in the order given.

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

<character substring function> is a triadic function, SUBSTRING, that returns a string extracted from a given string according to a given numeric starting position and a given numeric length.

<regular expression substring function> is a triadic function, SUBSTRING, distinguished by the keywords SIMILAR and UESCAPE. It has three parameters: a source character string, a pattern string, and an escape character. It returns a result string extracted from the source character string by pattern matching using POSIX-based regular expressions.

— **Step 1**: The escape character is exactly one character in length. As indicated in Figure 1, "Operation of <regular expression substring function>", the escape character precedes two instances of <double quote> that are used to partition the pattern string into three subpatterns (identified as *R1*, *R2*, and *R3*).

— **Step 2**: If the source string *S* does not satisfy the predicate

```
'S' SIMILAR TO 'R1' || 'R2' || 'R3'
```

then the result is the null value.

— **Step 3**: Otherwise, *S* is partitioned into two substrings *S1* and *S23* such that *S1* is the shortest initial substring of *S* such that the following is true:

```
'S1' SIMILAR TO 'R1' AND
  'S23' SIMILAR TO '(' || 'R2' || 'R3' || ')'
```

— **Step 4**: Next, *S23* is partitioned into two substrings *S2* and *S3* such that *S3* is the shortest final substring such that the following is true:

```
'S2' SIMILAR TO 'R2' AND 'S3' SIMILAR TO 'R3'
```

The result of the <regular expression substring function> is *S2*.

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

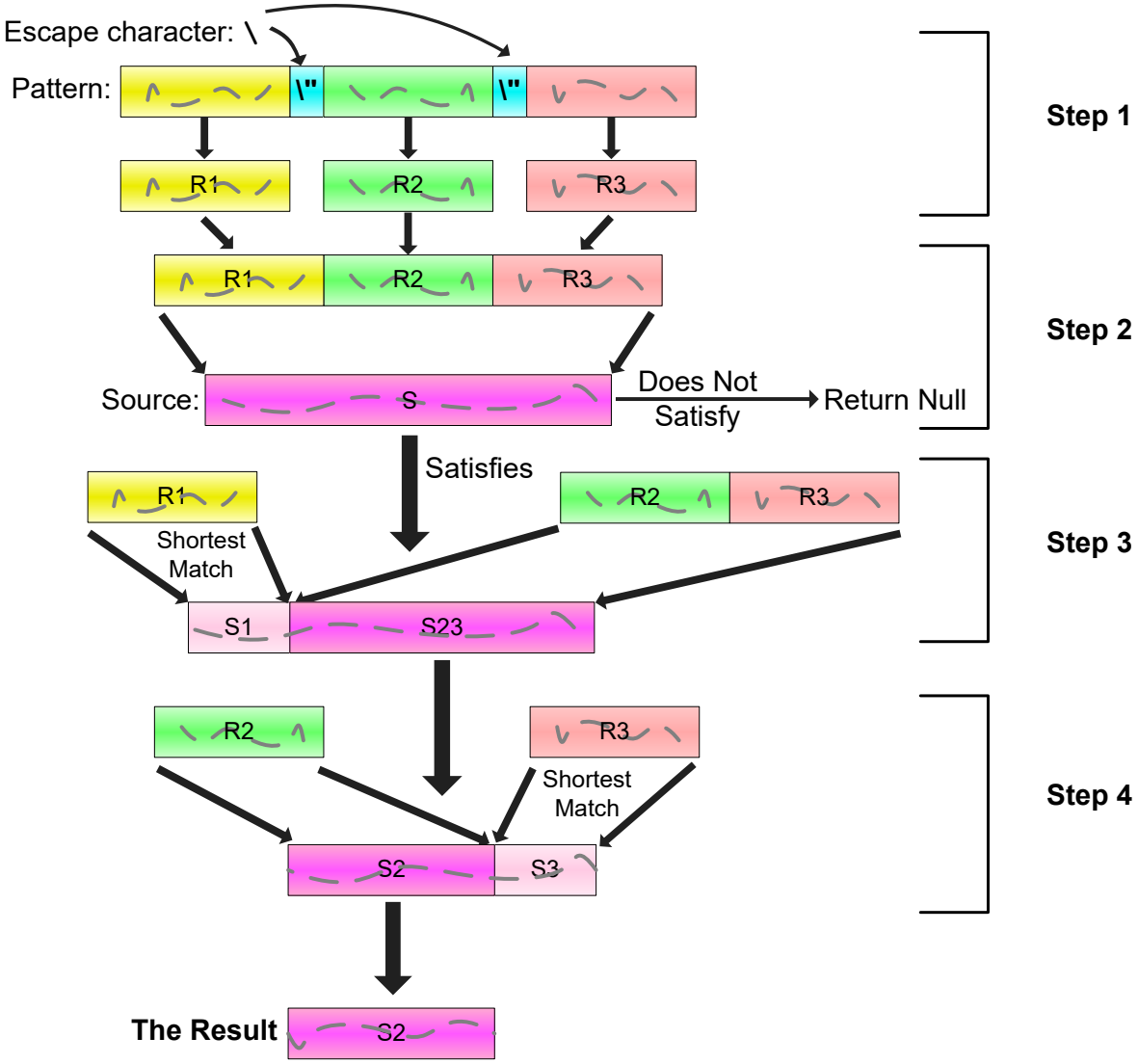

Figure 1 — Operation of <regular expression substring function>

<regex substring function> is a function, SUBSTRING_REGEX, that searches a string for an XQuery regular expression pattern and returns one occurrence of the matching substring.

<regex transliteration> is a function, TRANSLATE_REGEX, that searches a string for an XQuery regular expression pattern and returns the string with either one or every occurrence of the XQuery regular expression pattern replaced using an XQuery replacement string.

<character overlay function> is a function, OVERLAY, that modifies a string argument by replacing a given substring of the string, which is specified by a given numeric starting position and a given numeric length, with another string (called the replacement string). When the length of the substring is zero, nothing is removed from the original string and the string returned by the function is the result of inserting the replacement string into the original string at the starting position.

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

<fold> is a pair of functions for converting all the lower-case and title case characters in a given string to upper-case (UPPER) or all the upper-case and title case characters to lower-case (LOWER). A lower-case character is a character in the Unicode General Category class "Ll" (lower-case letters). An upper-case character is a character in the Unicode General Category class "Lu" (upper-case letters). A title case character is a character in the Unicode General Category class "Lt" (title-case letters).

NOTE 5 — Case correspondences are not always one-to-one: the result of case folding is sometimes of a different length in characters than the source string. For example, U+00DF, "ß", Latin Small Letter Sharp S, becomes "SS" when folded to upper-case.

<transcoding> is a function that invokes an installation-supplied transcoding to return a character string *S2* derived from a given character string *S1*. It is intended, though not enforced by this document, that *S2* be exactly the same sequence of characters as *S1*, but encoded according to some different character encoding form. A typical use might be to convert a character string from two-octet UCS to one-octet Latin1 or *vice versa*.

<single-character trim function> is a function that returns its first string argument with leading and/or trailing pad characters removed. The second argument indicates whether leading, or trailing, or both leading and trailing pad characters shall be removed. The third argument specifies the pad character that is to be removed.

<multi-character trim function> is a function that returns its first string argument with leading pad characters, trailing pad characters, or both leading and trailing pad characters removed. The second argument specifies the pad characters that are to be removed. In contrast to the <single-character trim function>, this function accepts a sequence of characters as the second argument and will remove all leading, trailing, or both occurrences of any of these characters, regardless of their ordering.

<pad function> is a function that returns its first string argument padded to the total length of characters specified by the second argument with the sequence of characters specified by the third argument.

<character transliteration> is a function for changing each character of a given string according to some many-to-one or one-to-one mapping between two not necessarily distinct character sets. The mapping, rather than being specified as part of the function, is some external function identified by a <transliteration name>.

For any pair of character sets, there are zero or more transliterations that may be invoked by a <character transliteration>. A transliteration is described by a transliteration descriptor. A transliteration descriptor includes:

— The name of the transliteration;

— The name of the character set from which it translates;

— The name of the character set to which it translates;

— The specific name of the SQL-invoked routine that performs the transliteration.

4.3.3.3 Other operators involving character strings

<length expression> returns the length of a given character string, as an exact numeric value, in characters or octets according to the choice of function.

<position expression> determines the first position, if any, at which one string, *S1*, occurs within another, *S2*. If *S1* is of length zero, then it occurs at position 1 (one) for any value of *S2*. If *S1* does not occur in *S2*, then zero is returned. The declared type of a <position expression> is exact numeric.

<like predicate> uses the triadic operator LIKE (or the inverse, NOT LIKE), operating on three character strings and returning a Boolean. LIKE determines whether or not a character string "matches" a given "pattern" (also a character string). The characters <percent> and <underscore> have special meaning when they occur in the pattern. The optional third argument is a character string containing exactly one

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

character, known as the “escape character”, for use when a <percent>, <underscore>, or the “escape character” itself is required in the pattern without its special meaning.

<similar predicate> uses the triadic operator SIMILAR (or the inverse, NOT SIMILAR), operating on three character strings and returning a Boolean. SIMILAR determines whether or not a character string “matches” a given “pattern” (also a character string) using a POSIX-based regular expression. The pattern is in the form of a “regular expression”. In this regular expression, certain characters (<left bracket>, <right bracket>, <left paren>, <right paren>, <vertical bar>, <circumflex>, <minus sign>, <plus sign>, <asterisk>, <underscore>, <percent>, <question mark>, <left brace>) have a special meaning. The optional third argument specifies the “escape character”, for use when one of the special characters or the “escape character” itself is required in the pattern without its special meaning.

<regex occurrences function> is a function, OCCURRENCES_REGEX, that searches a string for an XQuery regular expression pattern and returns an integer indicating the count of occurrences of the matched substring.

<regex position expression> is a function, POSITION_REGEX, that searches a string for an XQuery regular expression pattern and returns an integer indicating the beginning position, or 1 (one) plus the ending position, of one occurrence of the matched substring.

<regex like predicate> is a predicate, LIKE_REGEX, that performs XQuery regular expression matching.

4.3.3.4 Operations involving large object character strings

Large object character strings cannot be operated on by all string operations. Large object character strings can, however, be operated on by the following operations:

— <null predicate>;

— <like predicate>;

— <similar predicate>;

— <position expression>;

— <regex like predicate>;

— <regex position expression>;

— <regex occurrences function>;

— <regex substring function>;

— <regex transliteration>;

— <comparison predicate> with an <equals operator> or <not equals operator>;

— <quantified comparison predicate> with the <equals operator> or <not equals operator>.

As a result of these restrictions, large object character strings cannot be used in (among other places):

— predicates other than those listed above and the <exists predicate>;

— <general set function>;

— <group by clause>;

— <order by clause>;

— <unique constraint definition>;

— <referential constraint definition>;

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

— <select list> of a <query specification> that has a <set quantifier> of DISTINCT;

— UNION, INTERSECT, and EXCEPT;

— columns used for matching when forming a <joined table>.

All the operations described within Subclause 4.3.3.2, "Operators that operate on character strings and return character strings", and Subclause 4.3.3.3, "Other operators involving character strings", are supported for large object character strings.

4.3.4 Character repertoires

An SQL-implementation supports one or more character repertoires. These character repertoires may be defined by a standard or be implementation-defined (IA015).

A character repertoire is described by a character repertoire descriptor. A character repertoire descriptor includes:

— The name of the character repertoire;

— The name of the default collation for the character repertoire.

The following character repertoire names are specified as part of the ISO/IEC 9075 series:

— SQL_CHARACTER is a character repertoire that consists of the <SQL language character>s as specified in Subclause 5.1, "<SQL terminal character>". The name of the default collation is SQL_CHARACTER;

— GRAPHIC_IRV is the character repertoire that consists of the 95-character graphic subset of the International Reference Version (IRV) as specified in ISO/IEC 646. Its repertoire is a proper superset of that of SQL_CHARACTER. The name of the default collation is GRAPHIC_IRV;

— LATIN1 is the character repertoire defined in ISO/IEC 8859-1. The name of the default collation is LATIN1;

— ISO8BIT is the character repertoires formed by combining the character repertoire specified by ISO/IEC 8859-1 and the "control characters" specified by ISO/IEC 6429. The repertoire consists of all 255 characters, each consisting of exactly 8 bits, as, including all control characters and all graphic characters except the character corresponding to the numeric value 0 (zero). The name of the default collation is ISO8BIT;

— UCS is the Universal Character Set repertoire specified by ISO/IEC 10646:2020. It is implementation-defined (ID007) whether the name of the default collation is UCS_BASIC or UNICODE;

— SQL_TEXT is a character repertoire that is an implementation-defined (IA016) subset of the repertoire of the Universal Character Set that includes every <SQL language character> and every character in every character set supported by the SQL-implementation. The name of the default collation is SQL_TEXT;

— SQL_IDENTIFIER is an implementation-defined (IA029) character repertoire consisting of the <SQL language character>s and all other characters that the SQL-implementation supports for use in <regular identifier>s. The name of the default collation is SQL_IDENTIFIER.

Each character repertoire includes one or more characters defined to be whitespace; see Clause 3, "Terms and definitions".

NOTE 6 — The normative provisions of this document impose no requirement that any character repertoire have equivalents for any of these characters except U+0020 (<space>); however, by reference to this definition of whitespace, they do impose the requirement that every equivalent for one of these is recognized as a whitespace character.

NOTE 7 — If and when the set of characters with the Unicode property White_Space is modified to add new characters or remove characters, those modifications can implemented by SQL-implementations without affecting conformance to this document.

© ISO/IEC 2023 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO IEC 09075-2-2023

4.3.5 Character encoding forms

An SQL-implementation supports one or more character encoding forms for each character repertoire that it supports. These character encoding forms may be defined by a standard or be implementation-defined (IA017).

A character encoding form is described by a character encoding form descriptor. A character encoding form descriptor includes:

— The name of the character encoding form;

— The name of the character repertoire to which it is applicable.

The following character encoding form names are specified as part of the ISO/IEC 9075 series:

— SQL_CHARACTER is an implementation-defined (IV116) character encoding form. It is applicable to the SQL_CHARACTER character repertoire;

— GRAPHIC_IRV is the character encoding form in which the coded representation of each character is specified in ISO/IEC 646. It is applicable to the GRAPHIC_IRV character repertoire;

— LATIN1 is the character encoding form specified in ISO/IEC 8859-1. It is applicable to the LATIN1 character repertoire;

— ISO8BIT is the character encoding form specified in ISO/IEC 8859-1, augmented by ISO/IEC 6429. When restricted to the LATIN1 characters, it is the same character encoding form as LATIN1. It is applicable to the ISO8BIT character repertoire;

— UTF32 is the character encoding form specified in ISO/IEC 10646:2020, in which each character is encoded as four octets. It is applicable to the UCS character repertoire;

— UTF16 is the character encoding form specified in ISO/IEC 10646:2020, in which each character is encoded as two or four octets. It is applicable to the UCS character repertoire;

— UTF8 is the character encoding form specified in ISO/IEC 10646:2020, in which each character is encoded as from one to four octets. It is applicable to the UCS character repertoire;

— SQL_TEXT is an implementation-defined (IV117) character encoding form. It is applicable to the SQL_TEXT character repertoire;

— SQL_IDENTIFIER is an implementation-defined (IV118) character encoding form. It is applicable to the SQL_IDENTIFIER character repertoire.

If an SQL-implementation supplies more than one character encoding form for a particular character repertoire, then it shall specify a precedence ordering of the character encoding forms of that character repertoire. The precedence of character encoding forms applicable to the UCS character repertoire and defined in this document is:

$$\text{UTF8} \prec \text{UTF16} \prec \text{UTF32}$$

4.3.6 Collations

An SQL-implementation supports one or more collations for each character repertoire that it supports, and one or more collations for each character set that it supports. A collation is described by a collation descriptor. A collation descriptor includes:

— The name of the collation;

— The name of the character repertoire to which it is applicable;

— A list of the names of the character sets to which the collation can be applied;

© ISO/IEC 2023 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO IEC 09075-2-2023