INTERNATIONAL STANDARD ISO/IEC 9075-1:1999
ISO/IEC 9075-2:1999
ISO/IEC 9075-3:1999
ISO/IEC 9075-4:1999
ISO/IEC 9075-5:1999

TECHNICAL CORRIGENDUM 1

Published 2000-12-15

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION • МЕЖДУНАРОДНАЯ ОРГАНИЗАЦИЯ ПО СТАНДАРТИЗАЦИИ • ORGANISATION INTERNATIONALE DE NORMALISATION
INTERNATIONAL ELECTROTECHNICAL COMMISSION • МЕЖДУНАРОДНАЯ ЭЛЕКТРОТЕХНИЧЕСКАЯ КОМИССИЯ • COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

Information technology — Database languages — SQL —

Part 1: Framework (SQL/Framework)
Part 2: Foundation (SQL/Foundation)
Part 3: Call-Level Interface (SQL/CLI)
Part 4: Persistent Stored Modules (SQL/PSM)
Part 5: Host Language Bindings (SQL/Bindings)

TECHNICAL CORRIGENDUM 1

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

Partie 1: Charpente (SQL/Charpente)
Partie 2: Fondations (SQL/Fondations)
Partie 3: Interface de niveau d'appel (SQL/CLI)
Partie 4: Modules stockés persistants (SQL/PSM)
Partie 5: Liants de langage d'hôte (SQL/Liants)

RECTIFICATIF TECHNIQUE 1

Technical Corrigendum 1 to parts 1 to 5 of International Standard ISO/IEC 9075:1999 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange*.

Statement of purpose for rationale:

A statement indicating the rationale for each change to ISO/IEC 9075 is included. This is to inform the users of that standard as to the reason why it was judged necessary to change the original wording. In many cases the reason is editorial or to clarify the wording; in some cases it is to correct an error or an omission in the original wording.

Notes on numbering:

Where this Corrigendum introduces new Syntax, Access, General and Conformance Rules, the new rules have been numbered as follows:
Rules inserted between, for example, Rules 7) and 8) are numbered 7.1), 7.2), etc. [or 7) a.1), 7) a.2), etc.].
Those inserted before Rule 1) are numbered 0.1), 0.2), etc.
Where this Corrigendum introduces new subclauses, the new subclauses have been numbered as follows:
Subclauses inserted between, for example, subclause 4.3.2 and 4.3.3 are numbered 4.3.2a, 4.3.2b, etc.
Those inserted before, for example, 4.3.1 are numbered 4.3.0, 4.3.0a, etc.

ICS 35.060 Ref. No. ISO/IEC 9075 (parts 1 to 5):1999/Cor.1:2000(E)

© ISO/IEC 2000 – All rights reserved

Printed in Switzerland

Documents
Document Sharing Hub

Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

Page

ISO/IEC 9075-1:1999
Database Languages - SQL-Part 1:Framework . . . 9
4.8.2.3 Locators . . . 9
4.11.2 SQL-statements classified by function . . . 9
5.3.3 SQL-statements specified in ISO/IEC 9075-2 . . . 9
5.5.1 SQL-statements specified in ISO/IEC 9075-4 . . . 9
5.6.5.1 Additional functional classes of SQL-statements . . . 10
6.2.5 Relationships of incremental parts to ISO/IEC 9075-2, Foundation . . . 10
Annex B SQL Packages . . . 13
Annex B.9 SQL/MM Support . . . 13

ISO/IEC 9075-2:1999
Database Languages - SQL-Part 2:Foundation . . . 15
3.1.1 Definitions taken from ISO/IEC 10646 . . . 15
3.1.2 Definitions taken from Unicode . . . 15
3.1.5 Definitions provided in Part 2 . . . 15
4.1 Data types . . . 17
4.2.1 Character strings and collating sequences . . . 19
4.2.4 Named character sets . . . 19
4.3.1 Binary string comparison . . . 20
4.4.1 Bit string comparison and assignment . . . 20
4.7.1 Datetimes . . . 20
4.7.3 “Operations involving datetimes and intervals” . . . 21
4.8 User-defined types . . . 21
4.8.1 Observers and mutators . . . 22
4.8.2 Constructors . . . 22
4.8.4 User-defined type comparison and assignment . . . 23
4.10 Reference types . . . 23
4.13 Data conversions . . . 24
4.16 Tables . . . 24
4.16.3 Operations involving tables . . . 25
4.18.1 General rules and definitions . . . 25
4.18.9 Known functional dependencies in the result of <having clause> . . . 26
4.20 SQL-schemas . . . 26
4.21 SQL-client modules . . . 26
4.23 SQL-invoked routines . . . 26
4.24 Built-in functions . . . 29
4.25 SQL-paths . . . 29
4.26.4 Locators . . . 30
4.30.1 Classes of SQL-statements . . . 30
4.30.2 SQL-statements classified by functions . . . 30
4.30.3 SQL-statements and transaction states . . . 31
4.30.4 SQL-statement atomicity . . . 32
4.31.1 Authorization identifiers . . . 32
4.34.1 Execution contexts . . . 32
4.35.2 Execution of triggers . . . 33
5.2 <token> and <separator> . . . 33
5.3 <literal> . . . 36
5.4 Names and identifiers . . . 36
6.1 <data type> . . . 37
6.3 <value specification> and <target specification> . . . 37
6.5 <identifier chain> . . . 38
6.11 <method invocation> . . . 39

© ISO/IEC 2000 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

6.16 <set function specification> 40
6.18 <string value function> 41
6.20 <interval value function> 45
6.22 <cast specification> 45
6.23 <value expression> 47
6.24 <new specification> 48
6.25 <subtype treatment> 49
6.26 <numeric value expression> 49
6.27 <string value expression> 50
6.30 <boolean value expression> 51
7.1 <row value constructor> 52
7.3 <table value constructor> 52
7.6 <table reference> 53
7.7 <joined table> 57
7.8 <where clause> 57
7.9 <group by clause> 57
7.10 <having clause> 64
7.11 <query specification> 65
7.12 <query expression> 66
8.2 <comparison predicate> 66
8.3 <between predicate> 67
8.4 <in predicate> 68
8.6 <similar predicate> 68
8.8 <quantified comparison predicate> 69
8.10 <unique predicate> 69
8.11 <match predicate> 70
8.12 <overlaps predicate> 70
8.13 <distinct predicate> 70
9.0 Determination of identical values 71
9.1 Retrieval assignment 72
9.3 Data types of results of aggregations 72
10.4 <routine invocation> 73
10.5 <privileges> 77
10.6 <character set specification> 78
10.7 <specific routine designator> 78
10.12 Execution of triggers 80
10.13 Execution of array-returning functions 81
11.1 <schema definition> 82
11.2 <drop schema statement> 83
11.3 <table definition> 83
11.4 <column definition> 85
11.7 <unique constraint definition> 85
11.8 <referential constraint definition> 86
11.17 <drop column definition> 86
11.21 <view definition> 86
11.30 <character set definition> 87
11.31 <drop character set statement> 88
11.38 <trigger definition> 88
11.40 <user-defined type definition> 89
11.41 <attribute definition> 97
11.43 <add attribute definition> 98
11.44 <drop attribute definition> 98
11.45 <add original method specification> 98
11.46 <add overriding method specification> 102
11.47 <drop method specification> 104

© ISO/IEC 2000 – All rights reserved

Documents
Document Sharing Hub

Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

11.48 <drop data type statement> . . . 108
11.49 <SQL-invoked routine> . . . 108
11.50 <alter routine statement> . . . 114
11.51 <drop routine statement> . . . 114
11.52 <user-defined cast definition> . . . 115
11.54 <user-defined ordering definition> . . . 116
11.55 <drop user-defined ordering statement> . . . 117
11.56 <transform definition> . . . 119
11.57 <drop transform statement> . . . 119
12.2 <grant privilege statement> . . . 119
12.4 <select statement: single row> . . . 120
12.6 <revoke statement> . . . 120
13.1 <SQL-client module definition> . . . 121
13.2 <module name clause> . . . 121
13.3 <externally-invoked procedure> . . . 121
13.5 <SQL procedure statement> . . . 121
14.1 <declare cursor> . . . 122
14.3 <fetch statement> . . . 124
14.5 <select statement: single row> . . . 124
14.6 <delete statement: positioned> . . . 125
14.8 <insert statement> . . . 126
14.9 <update statement: positioned> . . . 127
14.10 <update statement: searched> . . . 127
14.11<temporary table declaration> . . . 128
14.18 Effect of inserting a table into a derived table . . . 128
15.2 <return statement> . . . 128
16.4 <savepoint statement> . . . 128
16.5 <release savepoint statement> . . . 128
16.7 <rollback statement> . . . 129
19.1 <get diagnostics statement> . . . 129
20.11 ATTRIBUTES view . . . 130
20.18 COLUMNS view . . . 131
20.19 CONSTRAINT_COLUMN_USAGE view . . . 132
20.21 DATA_TYPE_PRIVILEGES view . . . 133
20.26 DOMAINS view . . . 133
20.27 ELEMENT_TYPES view . . . 134
20.29 FIELDS view . . . 134
20.31 METHOD_SPECIFICATION_PARAMETERS view . . . 135
20.32 METHOD_SPECIFICATIONS view . . . 135
20.33 PARAMETERS view . . . 136
20.34 REFERENCED_TYPES view . . . 136
20.38 ROLE_TABLE_GRANTS view . . . 137
20.56 TABLES view . . . 137
20.62 TRIGGERS view . . . 138
20.68 VIEWS view . . . 140
20.69 Short name views . . . 141
21.3 EQUAL_KEY_DEGREES assertion . . . 143
21.6 ASSERTIONS base table . . . 144
21.7 ATTRIBUTES base table . . . 144
21.8 CHARACTER_SETS base table . . . 144
21.12 COLLATIONS base table . . . 145
21.14 COLUMNS base table . . . 145
21.15 DATA_TYPE_DESCRIPTOR base table . . . 146
21.16 DIRECT_SUPERTABLES . . . 150
21.23 METHOD_SPECIFICATION_PARAMETERS . . . 150

© ISO/IEC 2000 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

21.24 METHOD_SPECIFICATIONS base table 151
21.25 PARAMETERS base table 152
21.33 ROUTINES base table 152
21.34 SCHEMATA base table 152
21.35 SQL_FEATURES base table 153
21.36 SQL_IMPLEMENTATION_INFO base table 153
21.38 SQL_SIZING base table 153
21.39 SQL_SIZING_PROFILES base table 154
21.43 TABLES base table 154
21.44 TRANSFORMS base table 155
21.45 TRANSLATIONS base table 155
21.47 TRIGGER_COLUMN_USAGE base table 155
21.49 TRIGGERS base table 156
21.52 USER_DEFINED_TYPES base table 157
21.56 VIEWS base table 158
22.1 SQLSTATE 158
22.3 SQL Multimedia and Application Package SQLSTATE Subclasses 158
23.1 General conformance requirements 158
Annex A SQL conformance summary 158
Annex B Implementation-defined elements 164
Annex C Implementation-dependent elements 165
Annex E Incompatibilities with ISO/IEC 9075:1992 and ISO/IEC 9075-4:1996 166
Annex F SQL feature and package taxonomy 167

ISO/IEC 9075-3:1999
Database Languages - SQL-Part 3:Call Level Interface 169
3.1.1.Definitions provided in Part 3 169
3.3.2.1 Clause, subclause and table relationships 169
4.4.5 Connection attributes 170
5.1 <CLI routine> 170
5.3 Description of CLI item descriptor areas 170
5.5 Implicit DESCRIBE USING clause 171
5.11 Deferred parameter check 171
5.13 Description of CLI item descriptor areas 171
5.14 Other tables associated with CLI 172
5.15 Data type correspondences 173
6.5 BindCol 174
6.6 BindParameter 174
6.9 ColAttribute 174
6.10 ColumnPrivileges 175
6.11 Columns 175
6.12 Connect 175
6.14 DataSources 176
6.15 DescribeCol 176
6.17 EndTran 177
6.18 Error 178
6.19 ExecDirect 178
6.21 Fetch 178
6.22 FetchScroll 179
6.23 ForeignKeys 179
6.28 GetConnectAttr 180
6.29 GetCursorName 180
6.30 GetData 180
6.32 GetDescRec 180
6.34 GetDiagRec 181

© ISO/IEC 2000 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

6.36 GetFeatureInfo . . . 181
6.40 GetParamData . . . 181
6.50 Prepare . . . 182
6.51 PrimaryKeys . . . 182
6.54 SetConnectAttr . . . 182
6.55 SetCursorName . . . 182
6.56 SetDescField . . . 183
6.60 SpecialColumns . . . 183
6.62 TablePrivileges . . . 183
6.63 Tables . . . 184
A.1 C header file SQLCLI.H . . . 184
A.2 COBOL library item SQLCLI . . . 185

ISO/IEC 9075-4:1999
Database Languages - SQL-Part 4:Persistent Stored Modules . . . 187
3.3.2.1 Clause, Subclause, and Table relationships . . . 187
4.8 Cursors . . . 187
4.10 SQL-statements . . . 187
4.10.1 SQL-statements classified by function . . . 188
5.1 <token> and <separator> . . . 188
6.2 <identifier chain> . . . 188
9.18 <SQL-server module definition> . . . 189
11.2 <SQL procedure statement> . . . 189
12.2 <fetch statement> . . . 189
13.1 <compound statement> . . . 190
13.6 <resignal statement> . . . 190
13.7 <if statement> . . . 190
15.1 <embedded SQL host program> . . . 191
16.1 <get diagnostics statement> . . . 191
16.3 <resignal statement> . . . 191
17.2 MODULE_PRIVILEGES view . . . 191
17.4 MODULES view . . . 192
Annex A SQL Conformance Summary . . . 192
Annex E Incompatibilities with ISO/IEC 9075:1992 . . . 193
Annex F SQL Feature Taxonomy . . . 193

ISO/IEC 9075-5:1999
Database Languages - SQL-Part 5:Bindings . . . 195
4.6.1 Classes of SQL-statements . . . 195
4.6.4 Embeddable SQL-statement . . . 195
4.6.5 Preparable and immediately executable SQL-statements . . . 195
4.6.6 Directly executable SQL-statements . . . 196
5.1 <token> and <separator> . . . 196
10.5 <SQL-invoked routine> . . . 197
11.2 Calls to an <externally-invoked procedure> . . . 197
11.3 <SQL procedure statement> . . . 197
12.0 <fetch statement> . . . 197
12.1 <select statement: single row> . . . 198
12.2 <free locator statement> . . . 198
14.3 <set names statement> . . . 199
15.1 Description of SQL descriptor areas . . . 199
15.6 <prepare statement> . . . 199
15.8 <describe statement> . . . 200
15.13 <dynamic declare cursor> . . . 200
15.14 <allocate cursor statement> . . . 200

© ISO/IEC 2000 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

16.1 <embedded SQL host program> 201
16.3 <embedded SQL Ada program> 203
16.4 <embedded SQL C program> 204
16.5 <embedded SQL COBOL program> 204
16.6 <embedded SQL FORTRAN program> 205
16.7 <embedded SQL MUMPS program> 206
16.8 <embedded SQL Pascal program> 206
16.9 <embedded SQL PL/I program> 207
19.1 <get diagnostics statement> 208
20.1 SQLSTATE 208
Annex A SQL Conformance Summary 209
Annex E Incompatibilities with ISO/IEC 9075:1992 211

© ISO/IEC 2000 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

© ISO/IEC 2000 – All rights reserved

Documents
Document Sharing Hub

Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

ISO/IEC 9075-1:1999
Database Languages - SQL-Part 1:Framework

4.8.2.3 Locators

1. *Rationale: Correct the specification of which locators are marked invalid when an SQL-transaction ends.*

Replace the 8th paragraph with:

> A non-holdable locator remains valid until the end of the SQL-transaction in which it was generated, unless it is explicitly made invalid by the execution of a <free locator statement> or a <rollback statement> that specifies a <savepoint clause> is executed before the end of that SQL-transaction if the locator was generated subsequent to the establishment of the savepoint identified by the <savepoint clause>.

Replace 9th paragraph with:

> A holdable locator may remain valid beyond the end of the SQL-transaction in which it is generated. A holdable locator becomes invalid whenever a <free locator statement> identifying that locator is executed or the SQL-transaction in which it is generated or any subsequent SQL-transaction is rolled back. All locators remaining valid at the end of an SQL-session are marked invalid when that SQL-session terminates.

4.11.2 SQL-statements classified by function

1. *Rationale: Correct the classification of SQL-statements.*

Add a list element to the 1st paragraph:

— SQL-dynamic statements, which support the preparation and execution of dynamically generated SQL-statements, and obtaining information about them.

5.3.3 SQL-statements specified in ISO/IEC 9075-2

1. *Rationale: Correct the classification of SQL-statements.*

Replace the 4th bullet of the 1st paragraph with:

— Two SQL-control statements (CALL and RETURN), which can be used to invoke a procedure and specify a value to be returned by a function.

5.5.1 SQL-statements specified in ISO/IEC 9075-4

1. *Rationale: Correct the classification of SQL-statements.*

Replace the 1st bullet of the 1st paragraph with:

— Additional SQL-control statements which may be used to control the execution of an SQL routine.

© ISO/IEC 2000 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

2. *Rationale: Correct the classification of SQL-statements.*

Delete the 2nd bullet from the 1st paragraph

3. *Rationale: Correct the classification of SQL-statements.*

Replace the 3rd bullet of the 1st paragraph with:

— Additional SQL-diagnostics statements, which may be used to signal exceptions.

4. *Rationale: Correct the classification of SQL-statements.*

Add the following bullets to the 1st paragraph:

— SQL-control declaration statements which may be used to declare variables and exception handlers.

— Additional SQL-schema statements, which may be used to create and drop modules.

5.6.5.1 Additional functional classes of SQL-statements

1. *Rationale: Correct the classification of SQL-statements.*

Replace the bullet of the 1st paragraph with:

— SQL-dynamic statements, which support the preparation and execution of dynamically generated SQL-statements, and obtaining information about them.

2. *Rationale: Correct the classification of SQL-statements.*

Replace the 2nd paragraph with:

A number of SQL-data statements are also added, most of which contain the word "dynamic" in their names. They are not to be confused with SQL-dynamic statements.

6.2.5 Relationships of incremental parts to ISO/IEC 9075-2, Foundation

1. *Rationale: To permit the modification of Parts 1, 3 and 10 as well as Parts 2 and 5 in other Parts, since Part 1 needs to be updated by the Conformance clauses of other Parts and Parts 3 and 10 are analogous in functionality to Part 5.*

Replace the entire Subclause with:

6.2.5 Relationships of incremental parts within ISO/IEC 9075

Parts of ISO/IEC 9075 other than this part of ISO/IEC 9075 and ISO/IEC 9075-2 depend on ISO/IEC 9075-1, ISO/IEC 9075-2 and its Technical Corrigenda and are referenced as incremental parts. Each incremental part is to be used as though it were merged with the text of ISO/IEC 9075. This Subclause describes the conventions used to specify the merger.

The merger described also accounts for the Technical Corrigenda that have been published to correct ISO/IEC 9075. This accommodation is typically indicated by the presence of a phrase like “in the Technical Corrigenda” or “in the TC”.

© ISO/IEC 2000 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

6.2.5.1 New and modified Clauses, Subclauses, and Annexes

Where a Clause (other than Clause 1, “Scope”, and Clause 2, “Normative references”), Subclause, or Annex in any incremental part of ISO/IEC 9075 has a name identical to a Clause, Subclause, or Annex in ISO/IEC 9075-1, ISO/IEC 9075-2, ISO/IEC 9075-3, ISO/IEC 9075-5 or ISO/IEC 9075-10 (unless the incremental part is itself ISO/IEC 9075-3, ISO/IEC 9075-5 or ISO/IEC 9075-10), it supplements the Clause, Subclause, or Annex, respectively, in ISO/IEC 9075-1 and/or ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10, regardless of whether or not the number or letter of the Clause, Subclause, or Annex corresponds. It typically does so by adding or replacing paragraphs, Format items, or Rules.

In each incremental part, Table 1, "Clause, Subclause, and Table relationships", identifies the relationships between each Clause, Subclause, and Annex in that incremental part and the corresponding Clause, Subclause, or Annex in ISO/IEC 9075-1 and/or ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10.

Where a Clause, Subclause, or Annex in an incremental part has a name that is not identical to the name of some Clause, Subclause, or Annex in ISO/IEC 9075-1 and/or ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10, it provides language specification particular to that part. A Subclause that is part of a Clause or Subclause identified as new is inherently new and is not marked.

The Clauses, Subclauses, and Annexes in each incremental part appear in the order in which they are intended to appear in the merged document. In the absence of other explicit instructions regarding its placement, any new Clause, Subclause, or Annex is to be positioned as follows: Locate the prior Clause, Subclause, or Annex in ISO/IEC 9075-1 and/or ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10 whose name is identical to the name of a corresponding Clause, Subclause, or Annex that appears in the incremental part of ISO/IEC 9075. The new Clause, Subclause, or Annex shall immediately follow that Clause, Subclause, or Annex. If there are multiple new Clauses, Subclauses, or Annexes with no intervening Clause, Subclause, or Annex that modifies an existing Clause, Subclause, or Annex, then those new Clauses, Subclauses, or Annexes appear in order, following the prior Clause, Subclause, or Annex whose name was matched.

When an incremental part performs a modification to the Clause, Subclause, or Annex in ISO/IEC 9075-1 and/or ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10, then the modifications are applied in the following sequence:

1) All modifications to ISO/IEC 9075-1 from the incremental part.
2) All modifications to ISO/IEC 9075-3 from the incremental part.
3) All modifications to ISO/IEC 9075-5 from the incremental part.
4) All modifications to ISO/IEC 9075-10 from the incremental part.
5) All modifications to ISO/IEC 9075-2 from ISO/IEC 9075-3, including all modifications that were added, augmented, or replaced as a result of step 2.
6) All modifications to ISO/IEC 9075-2 from ISO/IEC 9075-5, including all modifications that were added, augmented, or replaced as a result of step 2.
7) All modifications to ISO/IEC 9075-2 from ISO/IEC 9075-10, including all modifications that were added, augmented, or replaced as a result of step 2.
8) All modifications to ISO/IEC 9075-2 from the incremental part. Note that modifications in this third step may augment or replace modifications applied as a result of steps 2, 3 and 4.

Modifications to ISO/IEC 9075-1 and/or ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10 by more than one incremental part do not interact. The modifications made by an incremental part only have influence on the language specification of that part and those specifications are not influenced by modifications made by any other incremental part.

© ISO/IEC 2000 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

6.2.5.2 New and modified Format items

In a modified Subclause, a Format item that defines a BNF non-terminal symbol (that is, the BNF non-terminal symbol appears on the left-hand side of the ::= mark) either modifies a Format item whose definition appears in ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10, or replaces a Format item whose definition appears in ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10, or defines a new Format item that does not have a definition at all in ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10. Those Format items in the incremental part that modify a Format item whose definition appears in ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10 are identified by the existence of a "Format comment" such as:

```
<modified item> ::=
       !! All alternatives from ISO/IEC 9075-2
     | <new alternative>
```

By contrast, Format items that completely replace Format items in ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10 have BNF non-terminal symbols identical to BNF non-terminal symbols of Format items in ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10, but do not state that they include any alternatives from ISO/IEC 9075-2 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10.

New Format items that have no correspondence to any Format item in ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10 are not distinguished in the incremental part.

Format items in new Subclauses are unmarked.

6.2.5.3 New and modified paragraphs and rules

In modified Subclauses, each paragraph or Rule is marked to indicate whether it is a modification of a paragraph or Rule in ISO/IEC 9075-1, ISO/IEC 9075-2, ISO/IEC 9075-3, ISO/IEC 9075-5 or ISO/IEC 9075-10 or is a new paragraph or Rule added by this incremental part.

Modifications of paragraphs or Rules in ISO/IEC 9075-2 are identified by the inclusion of an indicative phrase enclosed in a box.

[Replace the 5th paragraph] means that the following text is to replace the fifth paragraph of the corresponding Subclause in ISO/IEC 9075-2.

[Replace SR6) b) ii)] means that the following text is to replace Syntax Rule 6)b)ii) of the corresponding Subclause in ISO/IEC 9075-2.

[Augments SR3)] means that the following text is to extend or enhance Syntax Rule 3). In most instances, the augmentation is the addition of a new alternative meant to support new syntax. New paragraphs or Rules in an incremental part is marked to indicate where it is to be inserted.

[Insert before 2nd paragraph] means that the following text is to be read as though it were inserted immediately before the second paragraph of the corresponding Subclause in ISO/IEC 9075-2.

[Insert before GR4)] means that the following text is to be read as though it were inserted immediately before General Rule 4) of the corresponding Subclause in ISO/IEC 9075-2.

If no specific insertion point is indicated, as in [Insert this paragraph] or [Insert this GR], then the following text is to be read as though it were appended at the end of the appropriate section (the General Rules, for example) of the corresponding Subclause in ISO/IEC 9075-2.

© ISO/IEC 2000 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

Modifications of paragraphs or Rules in ISO/IEC 9075-1 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10 are identified in the same way as for modifications of ISO/IEC 9075-2, except that “in Part 1”, “in Part 3 ”, “in Part 5 ” or “in Part 10” is appended to the indicative phrase, as appropriate.

In such indications, “SR” is used to mean “Syntax Rule”, “AR” is used to mean “Access Rule”, “GR” is used to mean “General Rule”, and “CR” is used to mean “Conformance Rule”. “Desc.” is used to mean “Description” and “Func.” is used to mean “Function”.

All paragraphs, Format items, and Rules in new Clauses or Subclauses are also new and are therefore unmarked.

6.2.5.4 New and modified tables

If the name of a table in an incremental part is identical to that of a table in ISO/IEC 9075-1 and/or ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10, then the table supplements the table in ISO/IEC 9075-1 and/or ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10, typically by adding or replacing one or more table entries; otherwise, it is a new table.

In each incremental part, there is a table, Table 1, "Clause, Subclause, and Table relationships", that identifies the relationships between tables in that incremental part and the corresponding tables in ISO/IEC 9075-1 and/or ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10.

The rows in modified tables are generally new rows to be effectively inserted into the corresponding table in ISO/IEC 9075-1 and/or ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10, though in rare cases a row already in a table in ISO/IEC 9075-1 and/or ISO/IEC 9075-2 and/or ISO/IEC 9075-3 and/or ISO/IEC 9075-5 and/or ISO/IEC 9075-10 is effectively replaced by a row in the table in the incremental part. Such replacement is required wherever the value in the first column of the corresponding table is the same.

Annex B SQL Packages

1. *Rationale: Withdraw the package called “OLAP facilities”, which is being superseded by the package called “OLAP” as defined in Amendment 1 of ISO/IEC 9075:1999.*

In Table 2 — SQL Packages, delete line 3:

Delete the entire Annex B.3, “OLAP facilities”.

Annex B.9 SQL/MM Support

1. *Rationale: The package "SQL/MM support" is no longer needed since it is now defined explicitly in ISO/IEC 13249-1.*

Delete the entire Annex B.9, "SQL/MM support".

© ISO/IEC 2000 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

© ISO/IEC 2000 – All rights reserved

Documents
Document Sharing Hub

Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

ISO/IEC 9075-2:1999
Database Languages - SQL-Part 2:Foundation

3.1.1 Definitions taken from ISO/IEC 10646

1. *Rationale: Remove unused definitions.*

Replace the Subclause with:

This part of ISO/IEC 9075 makes use of the following terms defined in ISO/IEC 10646:

a) character

3.1.2 Definitions taken from Unicode

1. *Rationale: Remove unused definitions.*

Replace the Subclause with:

This part of ISO/IEC 9075 makes use of the following terms defined in The Unicode Standard:

a) control character

3.1.5 Definitions provided in Part 2

1. *Rationale: Clarify the definition of "assignment"*

Replace item b) with:

b) **assignment**: The 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). Assignment is frequently indicated by the use of the phrase "*T* is set to *S*" or "the value of *T* is set to *S*".

2. *Rationale: Clarify the definition of "comparable"*

Replace item i) with:

i) **comparable** (of a pair of values): Capable of being compared, according to the rules of Subclause 8.2 "<comparison predicate>". 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 Subclauses 4.2 to 4.11. Further, if a value of one data type can be compared with a value of another data type, then the two data types are said to be (mutually) comparable, see Subclause 4.12, "Type conversions and mixing of data types".

© ISO/IEC 2000 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

3. *Rationale: Correct and clarify the notion of "distinct".*

Replace item l) with:

l) **distinct**: (of a pair of comparable values): Informally: not equal or not both null or having a pair of corresponding components that are distinct. Formally:

Two null values of comparable type are not distinct.

A null value and a nonnull value of comparable type are distinct.

For two nonnull values, the following rules apply:

— Two values of predefined type or reference type are distinct if and only if they are not equal.

— If two values *V1* and *V2* are of a user-defined type whose comparison form is RELATIVE or MAP and the result of comparing them for equality according to Subclause 8.2, "<comparison predicate>" is *unknown*, then it is implementation-dependent whether they are distinct or not; otherwise, they are distinct if and only if they are not equal.

— If two values *V1* and *V2* are of a structured type whose comparison form is STATE, then they are distinct if their most specific types are different, or if there is an attribute *A* of their common most specific type such that the value of *A* in *V1* and the value of *A* in *V2* are distinct.

— Two rows (or partial rows) are distinct if and only if at least one of their pairs of respective fields is distinct.

— Two arrays are distinct if the arrays do not have the same cardinality or if, for arrays of the same cardinality, elements in the same ordinal position in the two arrays are distinct.

NOTE 0.1 — 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, is not defined.

4. *Rationale: Define "equals".*

Insert the following definition:

o.1) **equal** (of a pair of comparable values): Yielding *true* if passed as arguments in a <comparison predicate>. See Subclause 8.2, "<comparison predicate>".

5. *Rationale: Define "identical"*

Insert the following definition:

s.1) **identical** (of a pair of values): Indistinguishable, in the sense that it is impossible, by any means available in SQL, to detect any difference between them. For the full definition, see Subclause 9.0 "Determination of identical values".

6. *Rationale: Remove <identifier ignorable character>s from the definition of <white space>.*

In item vv) delete the following bullet items:

— U+200C, Zero Width Non-Joiner

— U+200D, Zero Width Joiner

© ISO/IEC 2000 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

— U+200E, Left-To-Right Mark

— U+200F, Right-To-Left Mark

7. *Rationale: Add missing character to the definition of <white space>.*

In item vv) add the following bullet item:

— U+202F, Narrow No-Break Space

4.1 Data types

1. *Rationale: Change "user-defined data type" to "user-defined type".*

Replace the 3rd paragraph with:

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

2. *Rationale: Correct oversight in definition of directly based on.*

Replace the 11th paragraph with:

A structured type *ST* is *directly based on* a data type *DT* if any of the following are true:

a) *DT* is the declared type of some attribute of *ST*.

b) *DT* is the direct supertype of *ST*.

c) *DT* is a direct subtype of *ST*.

d) *DT* is compatible with *ST*.

3. *Rationale: Add definition of non-referentially based on.*

Add the following paragraph immediately after the 15th paragraph:

A data type *DT1* is non-referentially based on a data type *DT2* if *DT1* is not a reference type and is either directly based on *DT2* or is directly based on some data type that is non-referentially based on *DT2*.

4. *Rationale: Provide consistent rules for comparison operations.*

Add the following paragraphs after the last paragraph:

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 this:

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

© ISO/IEC 2000 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

1. *T* is a member of *S*.

2. *T* is a row type and the declared type of some field of *T* is *S*-ordered.

3. *T* is an array type and the element type of *T* is *S*-ordered.

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

5. *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.

6. *T* is a reference type with a derived representation and the declared type of some attribute enumerated by the <derived representation> is *S*-ordered.

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

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

A type *T* is *array-ordered* if *T* is *ARR*-ordered, where *ARR* is the set of array types.

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

A type *T* is *DT-EC-ordered* if *T* is *DTE*-ordered, where *DTE* 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 *DTF*-ordered, where *DTF* is the set of distinct types with FULL comparison form.

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

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

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

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

A type *T* is *ST-ordered* if *T* is either 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

© ISO/IEC 2000 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

4.2.1 Character strings and collating sequences

1. *Rationale: Remove redundant definition.*

Replace the 4th paragraph with:

> The collating sequence used for a particular comparison is determined as in Subclause 4.2.3, “Rules determining collating sequence usage”.

4.2.4 Named character sets

1. *Rationale: Correct grammatical error.*

Replace the 1st bullet of the 1st paragraph with:

— SQL_CHARACTER specifies the name of a character repertoire that consists of the 88 <SQL language character>s as specified in Subclause 5.1, “<SQL terminal character>”. It consists of the 52 uppercase and lowercase simple latin characters, 10 digits, and 26 <SQL special character>s, including: <space>, <double quote>, <percent>, <ampersand>, <quote>, <left paren>, <right paren>, <asterisk>, <plus sign>, <comma>, <minus sign>, <period>, <solidus>, <colon>, <semicolon>, <less than operator>, <equals operator>, <greater than operator>, <question mark>, <underscore>, <vertical bar>, <left bracket>, <right bracket>, <circumflex>, <left brace>, and <right brace>. The 88 characters specified as <SQL language character>s are all included in the ISO International Reference Version (IRV) characters specified in ISO 646:1991.
The characters in IRV are included in many other international character set definitions. In addition, 82 of these characters (all except <vertical bar>, <left bracket>, <right bracket>, <circumflex>, <left brace>, and <right brace>) are in the most stable subset of IRV that, by ISO convention, is included in every latin-based ISO standard set of characters. As far as can be determined, <vertical bar>, <left bracket>, <right bracket>, <circumflex>, <left brace>, and <right brace> are included in most character sets that enjoy world-wide use. Thus, the SQL_CHARACTER repertoire is the most universal of the character sets named herein. The collation and form-of-use of SQL_CHARACTER are implementation-defined.

2. *Rationale: Editorial.*

Replace the 6th bullet of the 1st paragraph with:

— UTF16 and ISO10646 specify the name of a character repertoire that consists of every character represented by The Unicode Standard Version 2.0 and by ISO/IEC 10646 UTF-16, where each character is encoded using the UTF-16 encoding, occupying either 2 or 4 octets.

3. *Rationale: Editorial.*

Replace the 11th bullet of the 1st paragraph with:

> NOTE 4.1 — The character sets SQL_ CHARACTER, GRAPHIC_ IRV (or ASCII_ GRAPHIC), LATIN1, ISO8BIT (or ASCII_ FULL), and UNICODE (or ISO10646) have both a “ floor” and “ ceiling” requirement to consist of exactly the characters specified. Any character data type associated with one of these character sets has an implied integrity constraint limiting a value of the data type to be a character string consisting only of characters from the specified character set. The SQL_ TEXT and SQL_ IDENTIFIER character sets have a similar “ floor” requirement in that they must contain all characters that are in other character sets supported by the implementation (for SQL- data and for <identifier> s, respectively); however, SQL_ TEXT and SQL_ IDENTIFIER do not have a “ ceiling” requirement.

© ISO/IEC 2000 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

4.3.1 Binary string comparison

1. *Rationale: Remove redundant definition.*

Replace the paragraph with:

All binary strings are mutually comparable.

4.4.1 Bit string comparison and assignment

1. *Rationale: Remove redundant definition.*

Replace the paragraph with:

All bit strings are mutually comparable.

4.7.1 Datetimes

1. *Rationale: Correct and clarify Table 5.*

Add the following legend to Table 5, “Datetime data type conversions”:

SV	Source value
TV	Target value
.UTC	UTC component of *SV* or *TV* (if source or target has time zone)
.TZ	Time zone component of *SV* or *TV* (if source or target has time zone)
STZD	SQL-session default time zone displacement
=>	Cast from the type before the arrow to the type after the arrow
TIME w/ TZ	TIME WITH TIME ZONE
TIME w/o TZ	TIME WITHOUT TIME ZONE
TS w/ TZ	TIMESTAMP WITH TIME ZONE
TS w/o TZ	TIMESTAMP WITHOUT TIME ZONE

In the table cell for the conversion from TIME WITHOUT TIME ZONE to TIMESTAMP WITHOUT TIME ZONE, change “*TZ*” to “*SV*”.

In the table cell for the conversion from TIME WITH TIME ZONE to TIMESTAMP WITHOUT TIME ZONE, replace the cell contents with: *SV* => TS w/ TZ => TS w/o TZ.

In the table cell for the conversion from TIMESTAMP WITHOUT TIME ZONE to TIME WITHOUT TIME ZONE, change “*TZ*” to “*SV*”

In the table cell for the conversion from TIMESTAMP WITHOUT TIME ZONE to TIME WITH TIME ZONE, change “TS w/o TZ” to “TIME w/ TZ”.

© ISO/IEC 2000 – All rights reserved
AVA Documents Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

4.7.3 "Operations involving datetimes and intervals"

1. *Rationale: Clarify the meaning of the <overlaps predicate>.*

Replace the 5th paragraph with:

> <overlaps predicate> uses the operator OVERLAPS to determine whether or not two chronological periods overlap in time. A chronological period is specified either as a pair of datetimes (starting and ending) or as a starting datetime and an interval. If the length of the period is greater than 0, then the period consists of all points of time greater than or equal to the lower endpoint, and less than the upper endpoint. If the length of the period is 0, the period consists of a single point in time, the lower endpoint. Two periods overlap if they have at least one point in common.

2. *Rationale: Editorial - typographical error.*

Replace the 7th paragraph with:

> <interval absolute value function> operates on an interval argument and returns its absolute value in the same most specific type.

4.8 User-defined types

1. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values and correct incorrect terminology and <method characteristic> should not include <transform group specification>. Correct use of <specific name> for method specifications by replacing with <specific method name>.*

Replace the 3rd paragraph with:

> The definition of a user-defined type may include a <method specification list> consisting of one or more <method specification>s. A <method specification> is either an <original method specification> or an <overriding method specification>. Each <original method specification> specifies the <method name>, the <specific method name>, the <SQL parameter declaration list>, the <returns data type>, the <result cast from type> (if any), whether the method is type-preserving, the <language clause>, the <parameter style> if the language is not SQL, whether STATIC or CONSTRUCTOR is specified, whether the method is deterministic, whether the method possibly modifies SQL-data, possibly reads SQL-data, possibly contains SQL, or does not possibly contain SQL, and whether the method should be evaluated as the null value whenever any argument is the null value, without actually invoking the method.

2. *Rationale: Correct use of <specific name> for method specifications by replacing with <specific method name>.*

Replace the 4th paragraph with:

> Each <overriding method specification> specifies the <method name>, the <specific method name>, the <SQL parameter declaration list> and the <returns data type>. For each <overriding method specification>, there must be an <original method specification> with the same <method name> and <SQL parameter declaration list> in some proper supertype of the user-defined type. Every SQL-invoked method in a schema must correspond to exactly one <original method specification> or <overriding method specification> associated with some user-defined type existing in that schema.

© ISO/IEC 2000 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

3. *Rationale: Correct the specification of implicit generation of cast functions when reference values are generated via <user-defined representation>.*

In the 8th paragraph, replace the 12th bullet with:

— If the user-defined type is a structured type, then whether the referencing type of the structured type has a user-defined representation, a derived representation, or a system-defined representation, and the type descriptor of the representation type of the referencing type of the structured type if user-defined representation is specified or the list of attributes if derived representation is specified. NOTE 5 — "user-defined representation", "derived representation", and "system-defined representation" of a reference type are defined in Subclause 4.10, "Reference types".

4. *Rationale: Correct use of <specific name> for method specifications by replacing with <specific method name>.*

Replace the 1st sub-bullet in the 13th bullet of the 8th paragraph with:

- The <method name>.
- The <specific method name>.

5. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

Replace the 8th sub-bullet in the 13th bullet of the 8th paragraph with:

- If the <method specification> is an <original method specification>, then an indication of whether STATIC or CONSTRUCTOR is specified.

4.8.1 Observers and mutators

1. *Rationale: Attribute values might be null. Clarify the effect of observers and mutuators.*

Replace the 3rd paragraph with:

Let *V* be a value in data type *T* and let *AV* be a value in data type *AT*. The invocation *A*(*V*, *AV*) returns *MV* such that *A*(*MV*) is identical to from *AV* and for every attribute *A'*(*A'* ≠ *A*)of *T*, *A'*(*MV*) is identical to *A'*(*V*). The most specific type of *MV* is the most specific type of *V*.

4.8.2 Constructors

1. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

Replace the 1st and 2nd paragraphs with:

Associated with each structured type *ST* is one *constructor function*, implicitly defined when *ST* is defined. The constructor function is defined if and only if *ST* is instantiable.

Let *TN* be the name of a structured type *T*. The signature of the constructor function for *T* is *TN*( ) and its result data type is *T*. The invocation *TN*( ) returns a value *V* such that *V* is not null and, for every attribute *A* of *T*, *A*(*V*) returns the default value of *A*. The most specific type of *V* is *T*.

© ISO/IEC 2000 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

For every structured type *ST* that is instantiable, zero or more SQL-invoked constructor methods can be specified. The name of those methods must be equivalent to the name of the type for which they are specified.

NOTE 6.1 — SQL-invoked constructor methods are original methods that cannot be overloaded. An SQL-invoked constructor method and a regular function may exist such that both have equivalent function names, the types of the first parameters (if any) of the method's augmented parameter list and the function's parameter list are the same, and the types of the corresponding remaining parameters (if any) are identical according to the Syntax Rules of Subclause 10.14, "Data type identity".

4.8.4 User-defined type comparison and assignment

1. *Rationale: Values do not have declared types. Every user-defined type has a comparison type. Comparison forms and categories do not have to be the same throughout a subtype family.*

Replace the 6th paragraph with:

Two values *V1* and *V2* whose most specific types are user-defined types *T1* and *T2* are comparable if and only if *T1* and *T2* are in the same subtype family and each have some comparison type, *CT1* and *CT2* respectively. *CT1* and *CT2* constrain the comparison forms and comparison categories of *T1* and *T2* to be the same. Their respective comparison types *CT1* and *CT2* constrain the comparison forms and comparison categories of *T1* and *T2* to be the same, and the same as those of all their supertypes. If the comparison category is either STATE or RELATIVE, then the comparison functions of *T1* and *T2* are constrained to be equivalent; if the comparison category is MAP, they are not constrained to be equivalent.

4.10 Reference types

1. *Rationale: Correct the specification of implicit generation of cast functions when reference values are generated via <user-defined representation>.*

Replace the 2nd paragraph with:

Given a structured type *T*, the REF values that can reference rows in typed tables defined on *T* collectively form a certain data type *RT* known as a *reference type*. *RT* is the *referencing type* of *T*. *T* is the referenced type of *RT*.

2. *Rationale: Correct definition of the length of <reference type>s.*

Replace the 7th paragraph and its associated bullet items with:

A reference type is described by a reference type descriptor. A reference type descriptor includes:

— The name of the referenceable table, if any, that is the scope of the reference type.

— The name of the referenced type.

— The length of the referenced type.

3. *Rationale: Correct definition of the length of <reference type>s.*

Replace the 8th paragraph with:

In a host variable, a REF value is materialized as an *N*-octet value, where *N* is the length of the <reference type>.

© ISO/IEC 2000 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

4.13 Data conversions

1. *Rationale: To clarify the concept of user-defined casts.*

Replace the 1st paragraph with:

> Explicit data conversions can be specified by a *CAST operator*. A CAST operator defines how values of a source data type are converted into a value of a target data type according to the Syntax Rules and General Rules of Subclause 6.22, "<cast specification>". Data conversions between predefined data types and between constructed types are defined by the rules of this part of ISO/IEC 9075. Data conversions between a user-defined type and another data type are defined by a user-defined cast.

4.16 Tables

1. *Rationale: Clean up typed table insertability property for non-instantiable types.*

Replace the 7th paragraph with:

> All base tables are *updatable*. Derived tables are either updatable or not updatable. The operations of update and delete are permitted for updatable tables, subject to constraining Access Rules. Some updatable tables, including all base tables whose row type is not derived from a user-defined type that is not instantiable, are also *insertable-into*, in which case the operation of insert is also permitted, again subject to Access Rules.

2. *Rationale: Clean up typed table insertability property for non-instantiable types.*

Replace the 11th paragraph with:

> Every table descriptor includes:
>
> — The column descriptor of each column in the table.
>
> — The name, if any, of the structured type, if any, associated with the table.
>
> — An indication of whether the table is insertable-into or not.
>
> — An indication of whether the base table is a referenceable table or not, and an indication of whether the self-referencing column is a system-generated, a user-generated, or a derived self-referencing column.
>
> — A list, possibly empty, of the names of its direct supertables.
>
> — A list, possibly empty, of the names of its direct subtables.

3. *Rationale: Clean up typed table insertability property for non-instantiable types.*

Replace the 13th paragraph with:

> A derived table descriptor describes a derived table. In addition to the components of every table descriptor, a derived table descriptor includes:
>
> — The <query expression> that defines how the table is to be derived.
>
> — An indication of whether the derived table is updatable or not.

© ISO/IEC 2000 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

4.16.3 Operations involving tables

1. *Rationale: Correct the description of the effect of deleting, updating and inserting a row in a table.*

Replace the 4th, 5th and 6th paragraphs with:

Let *T* be a table whose value is *TV1*. For every row *RR*, let n_{RR} be the number of rows in *TV1* that are identical to *RR*.

The effect of deleting a row *R* from *T* is to replace the value *TV1* of *T* with the value *TV2* such that, for *RR* identical to *R*, the number of rows in *TV2* that are identical to *RR* is n_{RR} - 1 (one) and for *RR* not identical to *R*, the number of rows in *TV2* that are not identical to *RR* is n_{RR}. The primary effect of a <delete statement: positioned> on *T* is to delete exactly one specified row from *T*. The primary effect of a <delete statement: searched> on *T* is to delete zero or more rows from *T*.

The effect of replacing a row *R1* with the row *R2* in *T* is to replace the value *TV1* of *T* with the value *TV2* such that, if *R1* is not identical to *R2*, then, for *RR* identical to *R1*, the number of rows in *TV2* that are identical to *RR* is n_{RR} - 1 (one), for *RR* identical to *R2*, the number of rows in *TV2* that are identical to *RR* is n_{RR} + 1 (one), and for *RR* identical to neither *R1* nor *R2*, the number of rows in *TV2* that are identical to *RR* is n_{RR}. Otherwise *R1* is identical to *R2* and the number of rows in *TV2* that are identical to *RR* is n_{RR} for all *RR*.

The primary effect of an <update statement: positioned> on *T* is to replace exactly one specified row in *T* with some specified row. The primary effect of an <update statement: searched> on *T* is to replace zero or more rows in *T*.

If *T*, as well as being updatable, is insertable-into, then rows can be inserted into it. The effect of inserting a row *R* into *T* is to replace the value *TV1* of *T* with the value *TV2* such that, for *RR* identical to *R*, the number of rows in *TV2* that are identical to *RR* is n_{RR} + 1 (one) and, for *RR* not identical to *R*, the number of rows in *TV2* that are identical to *RR* is n_{RR}. The primary effect of an <insert statement> on *T* is to insert into *T* each of the zero or more rows contained in a specified table.

4.18.1 General rules and definitions

1. *Rationale: Use the notion of distinct.*

Replace the 2nd paragraph with:

Let "*T*: *A* → *B*" (read "in *T*, *A* determines *B*" or "*B* is functionally dependent on *A* in *T*") denote the functional dependency of *B* on *A* in *T*, which is true if, for any possible value of *T*, any two rows that are not distinct for every column in *A* also are not distinct for every column in *B*. When the table *T* is understood from context, the abbreviation "*A* → *B*" may also be used.

2. *Rationale: Correct erroneous symbol.*

Replace the 8th paragraph with:

In the following Subclauses, let a column *C1* be a *counterpart* of a column *C2* under qualifying table *QT* if *C1* is specified by a column reference (or by a <value expression> that is a column reference) that references *C2* and *QT* is the qualifying table of *C2*. If *C1* is a counterpart of *C2* under qualifying table *QT1* and *C2* is a counterpart of *C3* under qualifying table *QT2*, then *C1* is a counterpart of *C3* under *QT2*.

© ISO/IEC 2000 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

4.18.9 Known functional dependencies in the result of <having clause>

1. *Rationale: More than one <search condition> may be directly contained in a <having clause>.*

Replace the 1st paragraph with:

> Let *T1* be the table that is the operand of the <having clause>, let *SC* be the <search condition> simply contained in the <having clause>, and let *R* be the result of the <having clause>.

4.20 SQL-schemas

1. *Rationale: Cleanup of imprecise wording.*

Replace the 5th paragraph with:

> Base tables and views are identified by <table name>s. A <table name> consists of a <schema name> and an <identifier>. The <schema name> identifies the schema in which a persistent base table or view identified by the <table name> is defined. Base tables and views defined in different schemas can have <identifier>s that are equal according to the General Rules of Subclause 8.2, “comparison predicate>”.

4.21 SQL-client modules

1. *Rationale: Editorial.*

Replace NOTE 29 with:

> NOTE 29 — The <module character set specification> has no effect on the SQL language contained in the SQL-client module and exists only for compatibility with ISO/IEC 9075:1992. It may be used to document the character set of the SQL-client module.

4.23 SQL-invoked routines

1. *Rationale: Editorial.*

Replace the 4th, 5th and 6th paragraphs with:

> A <predicate> *P* is said to be dependent on an SQL-invoked routine *SR* if and only if *SR* is the ordering function included in the descriptor of a user-defined type *UDT* and one of the following conditions is true:
>
> — *P* is a <comparison predicate> that immediately contains a <row value expression> whose declared type is some user-defined type *T1* whose comparison type is *UDT*.
>
> — *P* is a <quantified comparison predicate> that immediately contains a <row value expression> that has some field whose declared type is some user-defined type *T1* whose comparison type is *UDT*.
>
> — *P* is a <unique predicate> that immediately contains a <table subquery> that has a column whose declared type is some user-defined type *T1* whose comparison type is *UDT*.
>
> — *P* is a <match predicate> that immediately contains a <row value expression> that has some field whose declared type is some user-defined type *T1* whose comparison type is *UDT*.
>
> — *P* is a <comparison predicate> with some corresponding value whose declared type is some array type whose element type is a user-defined type *T1* whose comparison type is *UDT*.

© ISO/IEC 2000 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

— *P* is a <quantified comparison predicate> that immediately contains a <row value expression> that has some field whose declared type is some array type whose element type is a user-defined type *T1* whose comparison type is *UDT*.

— *P* is a <unique predicate> that immediately contains a <table subquery> that has a column whose declared type is some array type whose element type is a user-defined type *T1* whose comparison type is *UDT*.

— *P* is a <match predicate> that immediately contains a <row value expression> that has some field whose declared type is some array type whose element type is a user-defined type *T1* whose comparison type is *UDT*.

NOTE 30 — "Comparison type" is defined in Subclause 4.8.4, "User-defined type comparison and assignment".

A <set function specification> *SFS* is said to be dependent on an SQL-invoked routine *SR* if and only if all the following are true:

— *SR* is the ordering function included in the descriptor of a user-defined type *UDT*.

— *SFS* is a <general set function> whose <set function type> *SFS* is MAX or MIN or *SFS* is a <general set function> whose <set qualifier> is DISTINCT.

— The declared type of the <value expression> of *SFS* is *UDT*.

A <group by clause> *GBC* is said to be dependent on an SQL-invoked routine *SR* if and only if all the following are true:

— *SR* is the ordering function included in the descriptor of a user-defined type *UDT*.

— The declared type of a grouping column of *GBC* is *UDT*.

2. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

Replace the 9th paragraph that begins with "An SQL-invoked method is an SQL-invoked function ..." with:

An *SQL-invoked method* is an SQL-invoked function that is specified by <method specification designator> (see Subclause 11.49, "<SQL-invoked routine>"). There are three kinds of SQL-invoked methods: *instance SQL-invoked methods*, *SQL-invoked constructor methods*, and *static SQL-invoked methods*. All SQL-invoked methods are associated with a user-defined type, also known as the *type of the method*. The <method characteristic>s of an SQL-invoked method are specified by a <method specification> contained in the <user-defined type definition> of the type of the method. An instance SQL-invoked method and an SQL-invoked constructor method both satisfy the following conditions:

— Its first parameter, called the *subject parameter*, has a declared type that is a user-defined type. The type of the subject parameter is the type of the method. A parameter other than the subject parameter is called an *additional parameter*.

— Its descriptor is in the same schema as the descriptor of the data type of its subject parameter.

An SQL-invoked constructor method satisfies the following additional condition:

— Its <method name> is equivalent to the <qualified identifier> simply contained in the <user-defined type name> included in the user-defined type descriptor of the type of the method.

© ISO/IEC 2000 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

3. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

Replace the 16th paragraph which begins with "SQL-invoked routines are invoked differently..." with:

SQL-invoked routines are invoked differently depending on their form. SQL-invoked procedures are invoked by <call statement>s. SQL-invoked regular functions are invoked by <routine invocation>s. Instance SQL-invoked methods are invoked by <method invocation>s, while SQL-invoked constructor methods are invoked by <new invocation>s and static SQL-invoked methods are invoked by <static method invocation>s. An invocation of an SQL-invoked routine specifies the <routine name> of the SQL-invoked routine and supplies a sequence of argument values corresponding to the <SQL parameter declaration>s of the SQL-invoked routine. A *subject routine* of an invocation is an SQL-invoked routine that may be invoked by a <routine invocation>. After the selection of the subject routine of a <routine invocation>, the SQL arguments are evaluated and the SQL-invoked routine that will be executed is selected. If the subject routine is an instance SQL-invoked method, then the SQL-invoked routine that is executed is selected from the set of overriding methods of the subject routine. (The term “set of overriding methods” is defined in the General Rules of Subclause 10.4, “<routine invocation>”.) The overriding method that is selected is the overriding method with a subject parameter the type designator of whose declared type precedes that of the declared type of the subject parameter of every other overriding method in the type precedence list of the most specific type of the value of the SQL argument that corresponds to the subject parameter. See the General Rules of Subclause 10.4, “<routine invocation>”. If the subject routine is not an SQL-invoked method, then the SQL-invoked routine executed is that subject routine. After the selection of the SQL-invoked routine for execution, the values of the SQL arguments are assigned to the corresponding SQL parameters of the SQL-invoked routine and its <routine body> is executed. If the SQL-invoked routine is an SQL routine, then the <routine body> is an <SQL procedure statement> that is executed according to the General Rules of <SQL procedure statement>. If the SQL-invoked routine is an external routine, then the <routine body> identifies a program written in some standard programming language other than SQL that is executed according to the rules of that standard programming language.

4. *Rationale: Editorial.*

Replace the 20th paragraph, which begins “If a <routine invocation> is contained in a <query expression> of a view, a check constraint, or an assertion, ...”, with:

If a <routine invocation> is contained in a <query expression> of a view, a check constraint, or an assertion, the <triggered action> of a trigger, or in an <SQL-invoked routine>, then the subject routine for that invocation is determined at the time the view is created, the check constraint is defined, the assertion is created, the trigger is created, or the SQL-invoked routine is created. If the subject routine is an SQL-invoked procedure, an SQL-invoked regular function, or a static SQL-invoked method, then the same SQL-invoked routine is executed whenever the view is used, the check constraint or assertion is evaluated, the trigger is executed, or the SQL-invoked routine is invoked. If the subject routine is an instance SQL-invoked method, then the SQL-invoked routine that is executed is determined whenever the view is used, the check constraint or assertion is evaluated, the trigger is executed, or the SQL-invoked routine is invoked, based on the most specific type of the value resulting from the evaluation of the SQL argument that correspond to the subject parameter. See the General Rules of Subclause 10.4, “<routine invocation>”.

5. *Rationale: Clarify the semantics of SQL-data access indication.*

Replace the 23rd paragraph, which begins “An external routine either *does not possibly contain SQL* or *possibly contains SQL*...”, with:

An external routine *does not possibly contain SQL*, *possibly contains SQL*, *possibly reads SQL-data*, or *possibly modifies SQL-data*. Only an external routine that possibly contains SQL, or possibly reads SQL-data or possibly modifies SQL-data may execute SQL-statements during its invocation.

© ISO/IEC 2000 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

Replace the 24th paragraph, which begins "An SQL-invoked routine may or may not *possibly read SQL-data*.", with:

> An SQL routine *possibly contains SQL*, *possibly reads SQL-data*, or *possibly modifies SQL-data*. Only an SQL-invoked routine that possibly reads SQL-data or possibly modifies SQL-data may read SQL-data during its invocation.

Replace the 25th paragraph, which begins "An SQL-invoked routine may or may not *possibly modify SQL-data*.", with:

> Only an SQL-invoked routine that *possibly modifies SQL-data* may modify SQL-data during its invocation.

6. *Rationale: Missing word.*

Replace the 28th paragraph, which begins "The identifiers in this new entry of the authorization stack are then modified", with:

> The identifiers in this new entry of the authorization stack are then modified depending on whether the SQL-invoked routine is an SQL routine or an external routine. If the SQL-invoked routine is an SQL routine, then, if the routine authorization identifier is a user identifier, the user identifier is set to the routine authorization identifier and the role name is set to null; otherwise, the role name is set to the routine authorization identifier and the user identifier is set to null.

4.24 Built-in functions

1. *Rationale: Remove undefined term.*

Replace the 1st paragraph with:

> Certain operators whose invocation employs syntax similar to that of <routine invocation> are designated built-in functions. Those that are so designated are those that appear in the result of the following <query expression>:

```
SELECT DISTINCT ROUTINE_NAME
  FROM INFORMATION_SCHEMA.ROUTINES
 WHERE SPECIFIC_SCHEMA = 'INFORMATION_SCHEMA'
```

4.25 SQL-paths

1. *Rationale: Use the correct BNF non-terminal (<user-defined type> rather than <user-defined type name>).*

Replace the 3rd paragraph with:

— The user-defined type when the <user-defined type> does not contain a <schema name>.

© ISO/IEC 2000 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

4.26.4 Locators

1. *Rationale: Correct the specification of which locators are marked invalid when an SQL-transaction ends.*

Replace 5th paragraph with:

> A non-holdable locator remains valid until the end of the SQL-transaction in which it was generated, unless it is explicitly made invalid by the execution of a <free locator statement> or a <rollback statement> that specifies a <savepoint clause> is executed before the end of that SQL-transaction if the locator was generated subsequent to the establishment of the savepoint identified by the <savepoint clause>.

Replace 6th paragraph with:

> A holdable locator may remain valid beyond the end of the SQL-transaction in which it is generated. A holdable locator becomes invalid whenever a <free locator statement> identifying that locator is executed or the SQL-transaction in which it is generated or any subsequent SQL-transaction is rolled back. All locators remaining valid at the end of an SQL-session are marked invalid when that SQL-session terminates.

4.30.1 Classes of SQL-statements

1. *Rationale: Clarify the semantics of SQL-data access indication.*

Replace the 2nd paragraph with:

> In this part of ISO/IEC 9075, there are at least three ways of classifying SQL-statements:
>
> — According to their effect on SQL objects, whether persistent objects, i.e., SQL-data, SQL-client modules, and schemas, or transient objects, such as SQL-sessions and other SQL-statements.
>
> — According to whether or not they start an SQL-transaction, or can, or must, be executed when no SQL-transaction is active.
>
> — According to whether they do not possibly contain SQL, possibly contain SQL, possibly read SQL-data, or possibly modify SQL-data.

4.30.2 SQL-statements classified by functions

1. *Rationale: Correct the classification of SQL-statements.*

Replace the 13th bullet of the 2nd paragraph with:

> — <grant privilege statement>

2. *Rationale: Correct the classification of SQL-statements.*

Add the following bullets to the 2nd paragraph:

> — <user-defined cast definition>
>
> — <drop user-defined cast statement>

© ISO/IEC 2000 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

3. *Rationale: Clarify the semantics of SQL-data access indication.*

Add the following Subclause after Subclause 4.30.2, “SQL-statements classified by functions”:

4.30.2a SQL-statements and SQL-data access indication

The following SQL-statements possibly contain SQL:

— SQL-control statements

— SQL-control declaration statements

— SQL-session statements

— SQL-diagnostics statements

— <free locator statement>

— <hold locator statement>

The following SQL-statements possibly read SQL-data:

— SQL-data statements other than SQL-data change statements, <free locator statement>, or and <hold locator statement>

— SQL-statements that simply contain a <subquery>

The following SQL-statements possibly modify SQL-data:

— SQL-schema statements

— SQL-data change statements

NOTE 31.1 — SQL-transaction statements, SQL-connection statements and SQL-dynamic statements are not included in the above classification since they are not permitted to occur in SQL-invoked routines.

4.30.3 SQL-statements and transaction states

1. *Rationale: Correct the classification of SQL-statements.*

Replace the 3rd paragraph with:

The following SQL-statements are possibly transaction-initiating SQL-statements:

— <return statement>

— <call statement>

© ISO/IEC 2000 – All rights reserved

Documents Document Sharing Hub

Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

4.30.4 SQL-statement atomicity

1. *Rationale: Add missing concept.*

Add the following paragraphs after the 1st paragraph:

> Depending on the type of SQL-statement being executed , a statement execution context may be said to be either atomic or non-atomic.
>
> A *non-atomic execution context* is said to be active during the execution of a non-atomic SQL-statement.

4.31.1 Authorization identifiers

1. *Rationale: Editorial.*

Replace the 1st paragraph that begins with “An <authorization identifier> identifies a set of privileges. ...” with:

> An <authorization identifier> identifies a set of privileges. An <authorization identifier> can be either a <user identifier> or a <role name>. A <user identifier> represents a user of the database system. The mapping of <user identifier>s to operating system users is implementation-dependent.
> A <role name> represents a role.

4.34.1 Execution contexts

1. *Rationale: Correct cross reference.*

Replace the 1st paragraph with:

> Execution contexts augment an SQL-session context to cater for certain special circumstances that might pertain from time to time during invocations of SQL-statements. An execution context is either a trigger execution context or a routine execution context. There is always a statement execution context, a routine execution context, and zero or more trigger execution contexts. For certain SQL-statements, the execution context is always atomic. A routine execution context is either atomic or non-atomic. Every trigger execution context is atomic. Statement execution contexts are described in Subclause 4.30.4, “SQL-statement atomicity”. Trigger execution contexts are described in Subclause 4.35, “Triggers”.

2. *Rationale: Clarify the semantics of SQL-data access indication.*

Replace the 2nd paragraph with:

> A routine execution context consists of:
>
> — An indication as to whether or not an SQL-invoked routine is active.
>
> — An SQL-data access indication, which indicates what SQL-statements, if any, are allowed during the execution of an SQL-invoked routine. The SQL-data access indication is one of the following: does not possibly contain SQL, possibly contains SQL, possibly reads SQL-data, or possibly modifies SQL-data.
>
> — An identification of the SQL-invoked routine that is active.
>
> — The routine SQL-path derived from the routine SQL-path if the SQL-invoked routine that is active is an SQL routine and from the external routine SQL-path if the SQL-invoked routine that is active is an external routine.

© ISO/IEC 2000 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

4.35.2 Execution of triggers

1. *Rationale: Editorial.*

Replace the 2nd paragraph that begins with "A trigger execution context consists of a set of ..." with:

> A trigger execution context consists of a set of *state changes*. Within a trigger execution context, each state change is uniquely identified by a *trigger event*, a *subject table*, and a *column list*. The trigger event can be DELETE, INSERT, or UPDATE. A state change *SC* contains a *set of transitions*, a set of statement-level triggers *considered as executed* for *SC*, and a set of row-level triggers, each paired with the set of rows in *SC* for which it is considered as executed.

2. *Rationale: Clarify the notion of state changes.*

Replace the 9th paragraph with:

> When execution of an SQL-data change statement S_i causes a trigger execution context *TEC* to come into existence, the set of state changes SSC_i is empty. Let *TE* be one of DELETE, INSERT, or UPDATE contained in S_i. Let *ST* be the subject table of S_i. If *TE* is INSERT or DELETE, then let *PSC* be a set whose only element is the empty set. If *TE* is UPDATE, then let *CL* be the list of columns being updated by S_i, and let *OC* be the set of column names identifying the columns in *CL*. Let *PSC* be the set consisting of the empty set and every subset of the set of column names of *ST* that has at least one column name that is in *OC*. Let *PSCN* be the number of elements in *PSC*. A state change SC_{ij} for *j* varying from 1 (one) to *PSCN*, identified by *TE*, *ST*, and the *j*-th element in *PSC*, is added to SSC_i, provided SSC_i does not already contain a state change corresponding to SC_{ij}. A transition T_{ijk} is added to SC_{ij} when a row is inserted into, updated in, or deleted from *ST* during the execution of S_i or the checking of referential constraints according to the General Rules of Subclause 11.8, "<referential constraint definition>", Subclause 14.6, "<delete statement: positioned>", Subclause 14.7, "<delete statement: searched>", Subclause 14.8, "<insert statement>", Subclause 14.9, "<update statement: positioned>", and Subclause 14.10, "<update statement: searched>".

3. *Rationale: Correct the use of unidentified identifiers.*

Replace the 12th paragraph with:

> When a state change SC_{ij} arises in SSC_i, one or more triggers are activated by SC_{ij}. A trigger *TR* is activated by SC_{ij} if and only if the subject table of *TR* is the subject table of SC_{ij}, the trigger event of *TR* is the trigger event of SC_{ij}, and the set of column names listed in the trigger column list of *TR* is the equivalent to the set of column names of SC_{ij}.
> NOTE 33.1 — The trigger column list is included in the descriptor of *TR*; it is empty if the trigger event is DELETE or INSERT. The trigger column list is also empty if the trigger event is UPDATE, but the <trigger event> of the <trigger definition> that defined *TR* does not specify a <trigger column list>.

5.2 <token> and <separator>

1. *Rationale: <underscore> is already defined in <identifier part>.*

In the Format, replace the production for <identifier body> with:

```
<identifier body> ::= <identifier start> [ { <identifier part> }... ]
```

© ISO/IEC 2000 – All rights reserved

Documents Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

2. *Rationale: Editorial. Misplaced and incorrect reserved words.*

In the Format, in the production for <non-reserved word>, delete the texts:

```
| BETWEEN
| BITVAR
| DYNAMIC_FUNCTION
| DYNAMIC_FUNCTION_CODE
| EXISTING
| EXISTS
| HOLD
| INFIX
| METHOD
| OVERLAPS
| SIMILAR
| SUBLIST
| USER_DEFINED_TYPE_CATALOG
| USER_DEFINED_TYPE_NAME
| USER_DEFINED_TYPE_SCHEMA
```

3. *Rationale: Editorial. Misplaced and incorrect reserved words.*

In the Format, in the production for <non-reserved word>, add the texts:

```
| ADMIN
| ATTRIBUTE
| CHARACTERISTICS
| EVERY
| DERIVED
| ORDERING
| SCOPE
| STATEMENT
```

in the correct alphabetical order.

4. *Rationale: Editorial. Misplaced and incorrect reserved words.*

In the Format, in the production for <reserved word>, delete the texts:

```
| ADMIN
| AGGREGATE
| ALIAS
| CLASS
| COMPLETION
| DESTROY
| DESTRUCTOR
| DICTIONARY
| EVERY
| HOST
| IGNORE
| INITIALIZE
| ITERATE
| LESS
| LIMIT
| MODIFY
| OFF
| OPERATION
| PARAMETERS
| POSTFIX
| PREFIX
```


© ISO/IEC 2000 – All rights reserved
Documents
Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

```
| PREORDER
| SCOPE
| SEQUENCE
| STATEMENT
| STRUCTURE
| TERMINATE
| THAN
| VARIABLE
```

5. *Rationale: Editorial. Misplaced and incorrect reserved words.*

In the Format, in the production for <reserved word>, add the texts:

```
| BETWEEN
| EXISTS
| HOLD
| METHOD
| OVERLAPS
| RELEASE
| SIMILAR
```

in the correct alphabetical order.

6. *Rationale: Use case-normal forms consistently to define equivalent identifiers.*

Delete NOTE 24.

Delete Syntax Rule 23)

Replace Syntax Rule 24) with:

24) The case- normal form of a <regular identifier> shall not be equal, according to the comparison rules in Subclause 8.2, "< comparison predicate>", to any <reserved word> (with every letter that is a lower- case letter replaced by the corresponding upper- case letter or letters), treated as the repetition of a <character string literal> that specifies a <character set specification> of SQL_ IDENTIFIER.

Replace Syntax Rules 25) and 26) with:

25) Two <regular identifier>s are equivalent if the case-normal forms of their <identifier body>s, considered as the repetition of a <character string literal> that specifies a <character set specification> of SQL_IDENTIFIER and an implementation-defined collation IDC that is sensitive to case, compare equally according to the comparison rules in Subclause 8.2, “<comparison predicate>”.

26) A <regular identifier> and a <delimited identifier> are equivalent if the case-normal form of the <identifier body> of the <regular identifier> and the <delimited identifier body> of the <delimited identifier> (with all occurrences of <quote> replaced by <quote symbol> and all occurrences of <doublequote symbol> replaced by <double quote>), considered as the repetition of a <character string literal> that specifies a <character set specification> of SQL_IDENTIFIER and collation IDC, compare equally according to the comparison rules in Subclause 8.2, “<comparison predicate>”.

7. *Rationale: <underscore> is already defined in <identifier part>.*

Replace Conformance Rule 1) with:

1) Without Feature F391, "Long identifiers", in a <regular identifier>, the number of <identifier part>s shall be less than 18.

© ISO/IEC 2000 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

5.3 <literal>

1. *Rationale: Specify explicitly the implication that F451, "Character set definition" depends on F461, "Named character sets" .*

Replace Conformance Rule 7) with:

7) Without Feature F461, "Named character sets", a <character string literal> shall not specify a <character set specification>.

2. *Rationale: Editorial.*

Replace Conformance Rules 8) and 9) with:

8) Without Feature F411, “Time zone specification”, conforming SQL shall not specify a <time zone interval>.

9) Without Feature T041, “Basic LOB data type support”, conforming SQL language shall not contain any <binary string literal>.

5.4 Names and identifiers

1. *Rationale: Use “equivalent” instead of “same” when comparing identifiers.*

Replace Syntax Rules 13), 14) and 15) with:

13) Two <schema qualified name>s are equivalent if and only if their <qualified identifier>s are equivalent and their <schema name>s are equivalent, regardless of whether the <schema name>s are implicit or explicit.

13.1) Two <local or schema qualified name>s are equivalent if and only if their <qualified identifier>s are equivalent and either they both specify MODULE or they both specify or imply <schema name>s that are equivalent.

14) Two <character set name>s are equivalent if and only if their <SQL language identifier>s are equivalent and their <schema name>s are equivalent, regardless of whether the <schema name>s are implicit or explicit.

15) Two <schema name>s are equivalent if and only if their <unqualified schema name>s are equivalent and their <catalog name>s are equivalent, regardless of whether the <catalog name>s are implicit or explicit.

2. *Rationale: Specify explicitly the implication that F451, "Character set definition" depends on F461, "Named character sets" .*

Replace Conformance Rule 12) with:

12) Without Feature F461, “Named character sets”, conforming SQL language shall not contain any <character set name>.

© ISO/IEC 2000 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

6.1 <data type>

1. *Rationale: Delete a redundant rule.*

Delete General Rule 11)

2. *Rationale: Use the correct BNF non-terminal (<user-defined type> rather than <user-defined type name>).*

Replace Conformance Rule 1) with:

1) Without Feature S023, “Basic structured types”, <user-defined type>, if specified, shall not identify a structured type.

3. *Rationale: Specify explicitly the implication that F451, "Character set definition" depends on F461, "Named character sets" .*

Replace Conformance Rule 8) with:

8) Without Feature F461, “Named character sets”, a <data type> shall not specify CHARACTER SET.

6.3 <value specification> and <target specification>

1. *Rationale: Editorial.*

Replace Syntax Rule 7) with:

7) A <target specification> or <simple target specification> that is a <column reference> shall be a new transition variable column reference.
NOTE 65 — “New transition variable column reference” is defined in Subclause 6.5, “<identifier chain>”.

8) Let *X* denote either a column *C* or the <key word> VALUE. Given a <boolean value expression> *BVE* and *X*, the notion “*BVE* is a known-not-null condition for *X*” is defined recursively as follows:

   a) If *BVE* is a <predicate>, then

      Case:

      i) If *BVE* is a <predicate> of the form “*RVE* IS NOT NULL”, where *RVE* is a <row value expression> that simply contains a <row value constructor element> that is a <column reference> that references *C*, then *BVE* is a *known-not-null condition* for *C*.

      ii) If *BVE* is the <predicate> “VALUE IS NOT NULL”, then *BVE* is a *known-not-null condition* for VALUE.

      iii) Otherwise, *BVE* is not a known-not-null condition for *X*.

   b) If *BVE* is a <value expression primary>, then

      Case:

      i) If *BVE* is of the form “<left paren> <value expression> <right paren>” and the <value expression> is a known-not-null condition for *X*, then *BVE* is a *known-not-null condition* for *X*.

© ISO/IEC 2000 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

ii) Otherwise, *BVE* is not a known-not-null condition for *X*.

c) If *BVE* is a <boolean test>, then let *BP* be the <boolean primary> immediately contained in *BVE*. If *BP* is a known-not-null condition for *X*, and <truth value> is not specified, then *BVE* is a *known-not-null condition* for *X*. Otherwise, *BVE* is not a known-not-null condition for *X*.

d) If *BVE* is of the form "NOT *BT*", where *BT* is a <boolean test>, then

Case:

i) If *BT* is "*CR* IS NULL", where *CR* is a column reference that references column *C*, then *BVE* is a *known-not-null condition* for *C*.

ii) If *BT* is "VALUE IS NULL", then *BVE* is a *known-not-null condition* for VALUE.

iii) Otherwise, *BVE* is not a known-not-null condition for *X*.

NOTE 66 — For simplicity, the rules do not attempt to analyze conditions such as "NOT NOT A IS NULL", or "NOT (A IS NULL OR NOT (B = 2))"

e) If *BVE* is of the form "*BVE1* AND *BVE2*", then

Case:

i) If either *BVE1* or *BVE2* is a known-not-null condition for *X*, then *BVE* is a *known-not-null condition* for *X*.

ii) Otherwise, *BVE* is not a known-not-null condition for *X*.

f) If *BVE* is of the form "*BVE1* OR *BVE2*", then *BVE* is not a known-not-null condition for *X*.

NOTE 67 — For simplicity, this rule does not detect cases such as "A IS NOT NULL OR A IS NOT NULL", which might be classified as a known-not-null condition.

6.5 <identifier chain>

1. Rationale: Take query names into account.

Replace Syntax Rule 7) a) with:

7) a) If $N = 1$ (one), then *IC* shall be contained within the scope of one or more exposed <table or query name>s or <correlation name>s whose associated tables include a column whose <identifier> is equivalent to I_1 or within the scope of a <routine name> whose associated <SQL parameter declaration list> includes an SQL parameter whose <identifier> is equivalent to I_1. Let the phrase *possible scope tags* denote those exposed <table or query name>s, <correlation name>s, and <routine name>s.

Case:

i) If the number of possible scope tags in the innermost scope containing a possible scope tag is 1 (one), then

Case:

© ISO/IEC 2000 – All rights reserved

Documents Document Sharing Hub

Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

1) If the innermost possible scope tag is a <table or query name> or <correlation name>, then let *T* be the table associated with the possible scope tag, and let *C* be the column of *T* whose <identifier> is equivalent to I_1. *PIC1* is the basis of *IC*, the basis length is 1 (one), the basis scope is the scope of *T*, and the basis referent is *C*.

2) If the innermost possible scope tag is a <routine name>, then let *SP* be the SQL parameter whose <identifier> is equivalent to I_1. *PIC1* is the basis of *IC*, the basis length is 1 (one), the basis scope is the scope of *SP*, and the basis referent is *SP*.

ii) Otherwise, each possible scope tag shall be a <table or query name> or a <correlation name> of a <table reference> that is directly contained in a <joined table> *JT*. I_1 shall be a common column name in *JT*. Let *C* be the column of *JT* that is identified by I_1. *PIC1* is the basis of *IC*, the basis length is 1 (one), and the basis referent is *C*.

NOTE 72 — "Common column name" is defined in Subclause 7.7, "<joined table>".

2. *Rationale: Editorial. Add missing text.*

Replace Syntax Rule 11) with:

11) A <basic identifier chain> whose basis referent is a column is a *column reference*. If the basis length is 2, and the basis scope is a <trigger definition> whose <trigger action time> is BEFORE, and I_1 is equivalent to the <new values correlation name> of the <trigger definition>, then the column reference is a *new transition variable column reference*.

6.11 <method invocation>

1. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

In the Format add the following production:

<constructor method selection> ::= <routine invocation>

2. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

Delete Syntax Rule 3).

3. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

Replace Syntax Rule 5) with:

5) Case:

a) If <method invocation> is immediately contained in <new invocation>, then let *TP* be an SQL-path containing the <schema name> of schema that includes the descriptor of *UDT*.

b) Otherwise, let *TP* be an SQL-path, arbitrarily defined, containing the <schema name> of every schema that includes a descriptor of a supertype or subtype of *UDT*.

© ISO/IEC 2000 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

4. Rationale: Provide correct logical structure.

Replace Syntax Rule 6) with

6) Case:
   a) If <generalized invocation> is specified, then let *DT* be the <data type> simply contained in the <generalized invocation>. Let *RI* be the following <method selection>:

      MN (*VEP* AS *DT AL*)

   b) Otherwise,

      Case:

      i) If <method invocation> is immediately contained in <new invocation>, then let *RI* be the following <constructor method selection>:

         MN (*VEP AL*)

      ii) Otherwise, let *RI* be the following <method selection>:

         MN (*VEP AL*)

6.16 <set function specification>

1. Rationale: Disallow DISTINCT on columns with types based on LOB and array types. Provide consistent Syntax Rules for comparison operations.

Add the following Syntax Rule:

6.1) If the <set function specification> specifies a <general set function> whose <set quantifier> is DISTINCT, then *DT* shall not be LOB-ordered, array-ordered, UDT-EC-ordered, or UDT-NC-ordered.

Delete Syntax Rule 7).

2. Rationale: Provide consistent Syntax Rules for comparison operations.

Replace Syntax Rule 8) with:

8) If the <set function specification> specifies a <set function type> that is MAX or MIN, then DT shall not be LOB-ordered, array-ordered, reference-ordered, UDT-EC-ordered, UDT-NC-ordered, or a row type.

Delete Syntax Rule 10).

3. Rationale: Changes to <group by clause> have changed the treatment of <grouping operation>. Editorial - Erroneous text not deleted.

Replace General Rules 3) h) and 3) i) with:

NOTE 78.1 — the value of <grouping operation> is computed by means of syntactic transformations in 7.9 “<group by clause>”.

© ISO/IEC 2000 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

4. *Rationale: Correcting the Feature Name of Feature T431 in the Conformance Rules of <set function specification>.*

Replace Conformance Rule 7) with:

7) Without Feature T431, "Extended grouping capabilities", conforming SQL language shall not contain a <set function specification> that is a <grouping operation>.

5. *Rationale: Provide consistent Conformance Rules for comparison operations.*

Replace Conformance Rule 8) with:

8) Without Feature S024, “Enhanced structured types”, in a <general set function>, if MAX or MIN is specified, then the <value expression> shall not be of an ST-ordered declared type.

Add the following Conformance Rule:

10) Without Feature S024, “Enhanced structured types”, if a <set qualifier> of DISTINCT is specified, then the <value expression> shall not be of ST-ordered declared type.

6.18 <string value function>

1. *Rationale: Provide more meaningful keywords for the <regular expression substring function>.*

In the Format, replace the production for <regular expression substring function> with:

```
<regular expression substring function> ::=
     SUBSTRING <left paren> <character value expression>
     SIMILAR <character value expression>
     ESCAPE <escape character> <right paren>
```

2. *Rationale: A <character value function> that operates on a <character value expression> of character large object should return character large object.*

Replace Syntax Rule 4) with:

4) If <character substring function> is specified, then:

a) Case:

i) If the declared type of <character value expression> is fixed-length character string or variable-length character string, then the declared type of the <character substring function> is variable-length character string with maximum length equal to the fixed length or maximum length of the <character value expression>.

ii) Otherwise, the declared type of the <character substring function> is large object character string with a maximum length equal to the maximum length of the <character value expression>.

a.1) The character repertoire and form-of-use of the <character substring function> are the same as the character repertoire and form-of-use of the <character value expression>.

b) The collating sequence and the coercibility characteristic are determined as specified for monadic operators in Subclause 4.2.3, “Rules determining collating sequence usage”, where the first operand of <character substring function> plays the role of the monadic operand.

© ISO/IEC 2000 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

3. *Rationale: A <character value function> that operates on a <character value expression> of character large object should return character large object.*

Replace Syntax Rule 5) with:

5) If <regular expression substring function> is specified, then:

a) The declared types of the <escape character> and the <character value expression>s of the <regular expression substring function> shall be character string with the same character repertoire.

b) Case:

i) If the declared type of <character value expression> is fixed-length character string or variable-length character string, then the declared type of the <regular expression substring function> is variable-length character string with maximum length equal to the maximum length of the first <character value expression>.

ii) Otherwise, the declared type of the <regular expression substring function> is large object character string with a maximum length equal to the maximum length of the first <character value expression>.

b.1) The character repertoire and form-of-use of <regular expression substring function> are the same as the character repertoire and form-of-use of the first <character value expression>.

c) The value of the <escape character> shall have length 1 (one).

d) The collating sequence and the coercibility characteristic are determined as specified for monadic operators in Subclause 4.2.3, “Rules determining collating sequence usage”, where the first operand of <regular expression substring function> plays the role of the monadic operand.

4. *Rationale: A <character value function> that operates on a <character value expression> of character large object should return character large object.*

Replace Syntax Rule 7) with:

7) If <form-of-use conversion> is specified, then:

a) <form-of-use conversion> shall be simply contained in a <value expression> that is immediately contained in a <derived column> that is immediately contained in a <select sublist> or shall immediately contain either a <simply value specification> that is a <host parameter name> or a <value specification> that is a <host parameter specification>.

b) A <form-of-use conversion name> shall identify a form-of-use conversion.

c) Case:

i) If the declared type of <character value expression> is fixed-length character string or variable-length character string, then the declared type of the result is variable-length character string with implementation-defined maximum length.

ii) Otherwise, the declared type of the result is large object character string with implementation-defined maximum length.

© ISO/IEC 2000 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

c.1) The character set of the result is the same as the character repertoire of the <character value expression> and form-of-use determined by the form-of-use conversion identified by the <form-of-use conversion name>. Let *CR* be that character repertoire. The result has the *Implicit* coercibility characteristic and its collating sequence is the default collating sequence of *CR*.

5. *Rationale: A <character value function> that operates on a <character value expression> of character large object should return character large object.*

Replace Syntax Rule 8) with:

8) If <character translation> is specified, then:

a) A <translation name> shall identify a character translation.

b) Case:

i) If the declared type of <character value expression> is fixed-length character string or variable-length character string, then the declared type of the <character translation> is variable-length character string with implementation-defined maximum length**.**

ii) Otherwise, the declared type of the <character translation> is large object character string with implementation-defined maximum length.

b.1) The declared type of the <character translation> has a character repertoire equal to the character repertoire of the target character set of the translation. Let *CR* be that character repertoire. The result has the *Implicit* coercibility characteristic and its collating sequence is the default collating sequence of *CR*.

6. *Rationale: A <character value function> that operates on a <character value expression> of character large object should return character large object.*

Replace Syntax Rule 9) with:

9) If <trim function> is specified, then;

a) If FROM is specified, then:

i) Either <trim specification> or <trim character> or both shall be specified.

ii) If <trim specification> is not specified, then BOTH is implicit.

iii) If <trim character> is not specified, then ' ' is implicit.

b) Otherwise, let *SRC* be <trim source>.

```
TRIM ( SRC )
```

is equivalent to

```
TRIM ( BOTH ' ' FROM SRC )
```

c) Case:

i) If the declared type of <character value expression> is fixed-length character string or variable-length character string, then the declared type of the <trim function> is variable-

© ISO/IEC 2000 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

length character string with maximum length equal to the fixed length or maximum length of the <trim source>.

ii) Otherwise, the declared type of the <trim function> is large object character string with maximum length equal to the maximum length of the <trim source>.

d) If a <trim character> is specified, then <trim character> and <trim source> shall be comparable.

e) The character repertoire and form-of-use of the <trim function> are the same as those of the <trim source>.

f) The collating sequence and the coercibility characteristic are determined as specified for monadic operators in Subclause 4.2.3, "Rules determining collating sequence usage", where the <trim source> plays the role of the monadic operand.

7. Rationale: Clarify the algorithm for <regular expression substring function>.

Replace General Rules 4)c), 4)d) and 4)e) with:

4) c) If the length in characters of *E* is not equal to 1 (one), then an exception condition is raised: *data exception — invalid escape character*.

d) If *R* does not contain exactly two occurrences of the two-character sequence consisting of *E* followed by the <double quote> character, then an exception condition is raised: *data exception — invalid use of escape character*.

e) Let *R1*, *R2*, and *R3* be the substrings of *R*, such that

```
'R' = 'R1' || 'E' || '"' || 'R2' || 'E' || '"' || 'R3'
```

is true.

f) If any one of *R1*, *R2* or *R3* is not a zero-length string and does not have the format of a <regular expression>, then an exception condition is raised: *data exception — invalid regular expression*.

g) If the predicate

```
'C' SIMILAR TO '(R1)' || '(R2)' || '(R3)' ESCAPE 'E'
```

is not true, then the result of the <regular expression substring function> is the null value.

h) Otherwise, the result of the <regular expression substring function> is computed as follows:

i) Let *S1* be the shortest initial substring of *C* such that there is a sub-string *S23* of *C* such that the following <search condition> is true:

```
'C' = 'S1' || 'S23' AND
'S1' SIMILAR TO 'R1' ESCAPE 'E' AND
'S23' SIMILAR TO '(R2)(R3)' ESCAPE 'E'
```

ii) Let *S3* be the shortest final substring of *S23* such that there is a sub-string *S2* of *S23* such that the following <search condition> is true:

```
'S23' = 'S2' || 'S3' AND
'S2' SIMILAR TO 'R2' ESCAPE 'E' AND
'S3' SIMILAR TO 'R3' ESCAPE 'E'
```


© ISO/IEC 2000 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

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

8. Rationale: Editorial.

Replace Conformance Rule 3) with:

3) Without Feature T042, “Extended LOB data type support”, conforming SQL language shall not contain any <blob value function>.

6.20 <interval value function>

1. Rationale: Editorial - typographical error.

Replace Syntax Rule 1) with:

1) If <interval absolute value function> is specified, then the declared type of the result is the declared type of the <interval value expression>.

Replace General Rule 1) with:

1) If <interval absolute value function> is specified, then let *N* be the value of the <interval value expression>.

Case:

a) If *N* is the null value, then the result is the null value.

b) If $N \geq 0$ (zero), then the result is *N*.

c) Otherwise, the result is -1**N*.

2. Rationale: Editorial.

Replace Conformance Rule 1) with:

1) Without Feature F052, “Intervals and datetime arithmetic”, conforming SQL shall contain no <interval value function>.

6.22 <cast specification>

1. Rationale: Editorial.

Replace Syntax Rule 4) with:

4) Let *C* be some column and let *CO* be the <cast operand> of a <cast specification> *CS*. *C* is a leaf column of *CS* if *CO* consists of a single column reference that identifies *C* or of a single <cast specification> *CS1* of which *C* is a leaf column.

2. Rationale: Editorial.

Replace the introductory paragraph of Syntax Rule 6) with:

6) If the <cast operand> is a <value expression>, then the valid combinations of *TD* and *SD* in a <cast specification> are given by the following table. “Y” indicates that the combination is syntactically

© ISO/IEC 2000 – All rights reserved

Documents Document Sharing Hub

Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

valid without restriction; “M” indicates that the combination is valid subject to other Syntax Rules in this Subclause being satisfied; and “N” indicates that the combination is not valid:

3. Rationale: Editorial.

Replace Syntax Rule 13) c) with:

13) c) CAST (*VALUE* AS *ETD*)
where *VALUE* is a <value expression> of declared type *ESD*, shall be a valid <cast specification>.

4. Rationale: Improve wording.

Replace Syntax Rule 8) d) with:

8) d) If *SD* is a fixed-length bit string or variable-length bit string, then let *LSV* be the value of BIT_LENGTH(*SV*) and let *B* be the BIT_LENGTH of the character with the smallest BIT_LENGTH in the form-of-use of *TD*. Let *PAD* be the value of the remainder of the division *LSV* / *B*. Let *NC* be a character whose bits all have the value 0 (zero).

If *PAD* is not 0 (zero), then append (*B* - *PAD*) 0-valued bits to the least significant end of *SV*; a completion condition is raised: *warning — implicit zero-bit padding*.

Let *SVC* be the possibly padded value of *SV* regarded as a character string without regard to valid character encodings and let *LTDS* be a character string of *LTD* characters of value *NC* characters in the form-of-use of *TD*.

TV is the result of

```
SUBSTRING ( SVC || LTDS FROM 1 FOR LTD )
```

Case:

i) If the length of *TV* is less than the length of *SVC*, then a completion condition is raised: *warning — string data, right truncation*.

ii) If the length of *TV* is greater than the length of *SVC*, then a completion condition is raised: *warning — implicit zero-bit padding*.

Replace Syntax Rule 8) d) with:

9) d) If *SD* is a fixed-length bit string or variable-length bit string, then let *LSV* be the value of BIT_LENGTH(*SV*) and let *B* be the BIT_LENGTH of the character with the smallest BIT_LENGTH in the form-of-use of *TD*.

Let *PAD* be the value of the remainder of the division *LSV* / *B*.

If *PAD* is not 0 (zero), then append (*B* - *PAD*) 0-valued bits to the least significant end of *SV*; a completion condition is raised: *warning — implicit zero-bit padding*.

Let *SVC* be the possible padded value of *SV* regarded as a character string without regard to valid character encodings.

Case:

i) If CHARACTER_LENGTH(*SVC*) is not greater than *MLTD*, then *TV* is *SVC*.

© ISO/IEC 2000 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

ii) Otherwise, *TV* is the result of

```
SUBSTRING ( SVC FROM 1 FOR MLTD )
```

If the length of *TV* is less than the length of *SVC*, then a completion condition is raised: *warning — string data, right truncation*.

Replace Syntax Rule 11) with:

11) If *TD* is fixed-length bit string, then let *LTD* be the length in bits of *TD*. Let *BLSV* be the result of BIT_LENGTH(*SV*).

Case:

a) If *BLSV* is equal to *LTD*, then *TV* is *SV* regarded as a bit string with a length in bits of *BLSV*.

b) If *BLSV* is larger than *LTD*, then *TV* is the first *LTD* bits of *SV* regarded as a bit string with a length in bits of *LTD*, and a completion condition is raised: *warning — string data, right truncation*.

c) If *BLSV* is smaller than *LTD*, then *TV* is *SV* regarded as a bit string extended on the right with *LTD* - *BLSV* bits whose values are all 0 (zero) and a completion condition is raised: *warning — implicit zero-bit padding*.

Replace Syntax Rule 12) with:

12) If *TD* is variable-length bit string, then let *MLTD* be the maximum length in bits of *TD*. Let *BLSV* be the result of BIT_LENGTH(*SV*).

Case:

a) If *BLSV* is less than or equal to *MLTD*, then *TV* is *SV* regarded as a bit string with a length in bits of *BLSV*.

b) If *BLSV* is larger than *MLTD*, then *TV* is the first *MLTD* bits of *SV* regarded as a bit string with a length in bits of *MLTD* and a completion condition is raised: *warning — string data, right truncation*.

5. Rationale: Editorial.

Replace General Rule 16) d) with:

16) d) If *SD* is TIMESTAMP WITH TIME ZONE, then the UTC component of *TV* is the hour, minute, and second <primary datetime field>s of *SV*, with implementation-defined rounding or truncation if necessary, and the time zone component of *TV* is the time zone displacement of *SV*.

6.23 <value expression>

1. Rationale: Correct the declared data type and value of <value expression primary>.

Replace Syntax Rule 2) with:

2) The declared type of a <value expression primary> is the declared type of the simply contained <unsigned value specification>, <column reference>, <set function specification>, <scalar subquery>, <case expression>, <value expression>, <cast specification>, <subtype treatment>, <attribute or

© ISO/IEC 2000 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

method reference>, <reference resolution>, <collection value constructor>, <field reference>, <element reference>, <method invocation>, or <static method invocation>, or the effective returns type of the immediately contained <routine invocation>, respectively.

2. *Rationale: Correct the declared data type and value of <value expression primary>.*

Replace General Rule 5) with:

5) The value of a <value expression primary> is the value of the simply contained <unsigned value specification>, <column reference>, <set function specification>, <scalar subquery>, <case expression>, <value expression>, <cast specification>, <subtype treatment>, <collection value constructor>, <field reference>, <element reference>, <method invocation>, <static method invocation>, <routine invocation>, or <attribute or method reference>.

6.24 <new specification>

1. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

In the Format, replace the production for <new invocation> with:

```
<new invocation> ::=
        <method invocation>
      | <routine invocation>
```

2. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

Replace Syntax Rule 3) with:

3) Case:

a) If the <new specification> is of the form

```
NEW RN( )
```

then:

Case:

i) If *S* does not include the descriptor of a constructor method whose method name is equivalent to *MN* and whose unaugmented parameter list is empty, then the <new specification> is equivalent to the <new invocation>

```
RN( )
```

ii) Otherwise, the <new specification> is equivalent to the <new invocation>

```
RN( ).MN( )
```


© ISO/IEC 2000 – All rights reserved

Documents Document Sharing Hub

Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

b) Otherwise, the <new specification>

```
NEW RN( a1, a2, ..., an )
```

is equivalent to the <new invocation>

```
RN().MN( a1, a2, ..., an )
```

3. Rationale: Editorial.

Replace Conformance Rule 1) with:

1) Without Feature S023, “Basic structured types”, conforming SQL language shall not contain any <new specification>.

6.25 <subtype treatment>

1. Rationale: Resolve incorrect duplication of BNF non-terminal symbol.

Replace the Format with:

```
<subtype treatment> ::=
     TREAT <left paren> <subtype operand>
       AS <target subtype> <right paren>

<subtype operand> ::= <value expression>

<target subtype> ::=
     <user-defined type>
```

2. Rationale: Editorial.

Replace Conformance Rule 1) with:

1) Without Feature S161, “Subtype treatment”, conforming SQL Language shall contain no <subtype treatment>.

6.26 <numeric value expression>

1. Rationale: Editorial.

Replace General Rule 5) with:

5) If the most specific type of the result of an arithmetic operation is exact numeric, then

Case:

a) If the operator is not division and the mathematical result of the operation is not exactly representable with the precision and scale of the result data type, then an exception condition is raised: *data exception — numeric value out of range*.

© ISO/IEC 2000 – All rights reserved

Documents
Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000

b) If the operator is division and the approximate mathematical result of the operation represented with the precision and scale of the result data type loses one or more leading significant digits after rounding or truncating if necessary, then an exception condition is raised: *data exception — numeric value out of range*. The choice of whether to round or truncate is implementation-defined.

6.27 <string value expression>

1. *Rationale: Include character large object in the description of the effects of <concatenation>.*

Replace Syntax Rule 3) with:

3) Case:

a) If <concatenation> is specified, then:
Let *D1* be the declared type of the <character value expression> and let *D2* be the declared type of the <character factor>. Let *M* be the length in characters of *D1* plus the length in characters of *D2*. Let *VL* be the implementation-defined maximum length of a variable-length character string, let *LOL* be the implementation-defined maximum length of a large object character string, and let *FL* be the implementation-defined maximum length of a fixed-length character string.

Case:

i) If the declared type of the <character value expression> or <character value expression> or <character factor> is large object character string, then the declared type of the <concatenation> is large object character string with maximum length equal to the lesser of *M* and *LOL*.

ii) If the declared type of the <character value expression> or <character factor> is variable-length character string, then the declared type of the <concatenation> is variable-length character string with maximum length equal to the lesser of *M* and *VL*.

iii) If the declared type of the <character value expression> and <character factor> is fixed-length character string, then *M* shall not be greater than *FL* and the declared type of the <concatenation> is fixed-length character string with length *M*.

b) Otherwise, the declared type of the <character value expression> is the declared type of the <character factor>.

2. *Rationale: Include character large object in the description of the effects of <concatenation>.*

Replace General Rule 2) with:

2) If <concatenation> is specified, then let *S1* and *S2* be the result of the <character value expression> and <character factor>, respectively.

Case:

a) If either *S1* or *S2* is the null value, then the result of the <concatenation> is the null value.

b) Otherwise, let *S* be the string consisting of *S1* followed by *S2* and let *M* be the length of *S*.

Case:

© ISO/IEC 2000 – All rights reserved
Documents Document Sharing Hub
Full document details: ISO_IEC 9075-5_1999_Cor 1_2000