INTERNATIONAL STANDARD ISO/IEC 9075-2:2011
TECHNICAL CORRIGENDUM 2

Published 2015-10-01

Information technology — Database languages — SQL —

Part 2:
Foundation (SQL/Foundation)

TECHNICAL CORRIGENDUM 2

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

Partie 2: Fondations (SQL/Foundation)

RECTIFICATIF TECHNIQUE 2

Technical Corrigendum 2 to ISO/IEC 9075-2:2011 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange*.

ICS 35.060 **Ref. No. ISO/IEC 9075-2:2011/Cor.2:2015(E)**

Contents

Page

4 Concepts . . . 1
4.1 Data types . . . 1
4.1.4 Data type terminology . . . 1
4.2 Character strings . . . 1
4.2.4 Character repertoires . . . 1
4.4 Numbers . . . 2
4.4.3 Operations involving numbers . . . 2
4.7 User-defined types . . . 2
4.7.7 User-defined type descriptor . . . 2
4.15 Tables . . . 2
4.15.2 Base tables . . . 2
4.15.2.3 Temporary tables . . . 2
4.15.6 Table updatability . . . 3
4.15.7 Table descriptors . . . 4
4.15.8 Relationships between tables . . . 4
4.15.9 Referenceable tables, subtables and supertables . . . 5
4.15.10 Operations involving tables . . . 5
4.15.11 Identity columns . . . 5
4.18 Integrity constraints . . . 6
4.18.3 Table constraints . . . 6
4.18.3.2 Unique constraints . . . 6
4.18.3.3 Referential constraints . . . 7
4.18.3.4 Table check constraints . . . 8
4.19 Functional dependencies . . . 9
4.19.8 Known functional dependencies in a <joined table> . . . 9
4.19.8 Known functional dependencies in a <table primary> . . . 10
4.19.9 Known functional dependencies in a <table factor> . . . 10
4.25 Dynamic SQL concepts . . . 11
4.25.2 Dynamic SQL statements and descriptor areas . . . 11
4.36 SQL-transactions . . . 11
4.36.3 Properties of SQL-transactions . . . 11
4.36.7 Encompassing transactions . . . 11
4.38 SQL-sessions . . . 12
4.38.4 SQL-session context management . . . 12
5 Lexical elements . . . 12
5.3 <literal> . . . 12

6 Scalar expressions. . . . 13
6.1 <data type>. . . . 13
6.6 <identifier chain>. . . . 13
6.10 <window function>. . . . 14
6.11 <nested window function>. . . . 15
6.23 <reference resolution>. . . . 15
6.28 <numeric value function>. . . . 16
6.29 <string value expression>. . . . 16
6.30 <string value function>. . . . 16

7 Query expressions. . . . 17
7.6 <table reference>. . . . 17
7.7 <joined table>. . . . 18
7.11 <window clause>. . . . 19
7.12 <query specification>. . . . 20
7.13 <query expression>. . . . 21

8 Predicates. . . . 24
8.2 <comparison predicate>. . . . 24
8.5 <like predicate>. . . . 25
8.7 <regex like predicate>. . . . 25
8.17 <submultiset predicate>. . . . 26

9 Additional common rules. . . . 26
9.3 Passing a value from a host language to the SQL-server 26
9.4 Passing a value from the SQL-server to a host language 27
9.5 Result of data type combinations 27
9.12 Grouping operations. . . . 28
9.15 Collation determination. . . . 28
9.16 Execution of array-returning functions. . . . 28
9.18 XQuery regular expression matching. . . . 29
9.30 Determination of view component privileges. . . . 29

10 Additional common elements. . . . 31
10.4 <routine invocation>. . . . 31
10.9 <aggregate function>. . . . 34

11 Schema definition and manipulation. . . . 35
11.2 <drop schema statement>. . . . 35
11.3 <table definition>. . . . 36
11.4 <column definition>. . . . 36
11.7 <unique constraint definition>. . . . 37
11.8 <referential constraint definition>. . . . 37
11.16 <drop column not null clause>. . . . 38
11.19 <alter column data type clause>. . . . 38
11.26 <drop table constraint definition>. . . . 39
11.27 <add table period definition>. . . . 40

11.32 <view definition>. 40
11.34 <domain definition>. 43
11.49 <trigger definition>. 44
11.50 <drop trigger statement>. 44
11.51 <user-defined type definition>. 44
11.56 <add original method specification>. 46
11.57 <add overriding method specification>. 46
11.60 <SQL-invoked routine>. 46
11.61 <alter routine statement>. 48

12 Access control. 48
12.7 <revoke statement>. 48

13 SQL-client modules. 49
13.4 <SQL procedure statement>. 49
13.5 Data type correspondences. 53

14 Data manipulation. 55
14.3 <cursor specification>. 55
14.5 <fetch statement>. 55
14.8 <delete statement: positioned>. 56
14.11 <insert statement>. 56
14.12 <merge statement>. 58
14.13 <update statement: positioned>. 63
14.14 <update statement: searched>. 64
14.15 <set clause list>. 65
14.16 <temporary table declaration>. 65

15 Additional data manipulation rules. 65
15.1 Effect of opening a cursor. 65
15.5 Effect of a positioned delete. 66
15.6 Effect of a positioned update. 66
15.7 Effect of deleting rows from base tables. 66
15.10 Effect of inserting tables into base tables. 67
15.12 Effect of inserting a table into a viewed table. 68
15.13 Effect of replacing rows in base tables. 68
15.15 Effect of replacing some rows in a viewed table. 70
15.17 Execution of referential actions. 70
15.19 Execution of triggers. 71

16 Control statements. 72
16.2 <return statement>. 72

20 Dynamic SQL. 72
20.2 <allocate descriptor statement>. 72
20.4 <get descriptor statement>. 73
20.5 <set descriptor statement>. 73
20.6 <prepare statement>. 73

20.8 <deallocate prepared statement> . . . 76
20.15 <allocate extended dynamic cursor statement> . . . 76
20.17 <dynamic open statement> . . . 77
20.21 <dynamic delete statement: positioned> . . . 77
20.22 <dynamic update statement: positioned> . . . 77
20.23 <preparable dynamic delete statement: positioned> . . . 77
20.25 <preparable dynamic update statement: positioned> . . . 78
21 Embedded SQL . . . 79
21.1 <embedded SQL host program> . . . 79
21.3 <embedded SQL Ada program> . . . 79
21.4 <embedded SQL C program> . . . 80
21.5 <embedded SQL COBOL program> . . . 82
21.6 <embedded SQL Fortran program> . . . 84
21.7 <embedded SQL MUMPS program> . . . 85
21.8 <embedded SQL Pascal program> . . . 85
21.9 <embedded SQL PL/I program> . . . 86
22 Direct invocation of SQL . . . 87
22.1 <direct SQL statement> . . . 87
25 Conformance . . . 88
25.3 Implied feature relationships of SQL/Foundation . . . 88

Tables

Table **Page**

16 Data type correspondences for Ada. 53
17 Data type correspondences for C. 53
17 Data type correspondences for C. 54
18 Data type correspondences for COBOL. 54
19 Data type correspondences for Fortran. 54
19 Data type correspondences for Fortran. 54
35 Implied feature relationships of SQL/Foundation. 88

Information technology — Database languages — SQL —

Part 2:
Foundation (SQL/Foundation)

TECHNICAL CORRIGENDUM 1

4 Concepts

4.1 Data types

4.1.4 Data type terminology

This Subclause is modified by Subclause 4.1.2, "Data type terminology", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 4.1.2, "Data type terminology", in ISO/IEC 9075-14.

1. *Rationale: To clarify that the attributes of the derived representation are a distinguished subset of the attributes of the structured type.*

   Replace the 6th bullet of the 10th paragraph with:

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

2. *Rationale: Resolve potentially conflicting definitions of array-ordered and multiset-ordered.*

   Replace the 2nd, 3rd, 4th and 5th bullets of the 11th paragraph with:

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

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

4.2 Character strings

4.2.4 Character repertoires

1. *Rationale: Correct the number of the standard.*

   Replace the lead text of the 3rd paragraph with:

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

4.4 Numbers

4.4.3 Operations involving numbers

1. *Rationale: Remove erroneous see reference.*

   Replace the 5th bullet of the 1st paragraph with:

   — <max cardinality expression> (see Subclause 4.10.5, "Operations involving arrays") operates on an array argument and returns an integer.

4.7 User-defined types

4.7.7 User-defined type descriptor

This Subclause is modified by Subclause 4.8.3, "User-defined type descriptor", in ISO/IEC 9075-13.

1. *Rationale: To clarify that the attributes of the derived representation are a distinguished subset of the attributes of the structured type.*

   Replace the 2nd bullet of the 11th bullet of the 1st paragraph with:

   — ...

   - If user-defined representation is specified, then the type descriptor of the representation type of the referencing type of the structured type; otherwise, if derived representation is specified, then the list of derivational attributes of the derived representation.

4.15 Tables

4.15.2 Base tables

4.15.2.3 Temporary tables

This Subclause is modified by Subclause 4.2.1.1, "Temporary tables", in ISO/IEC 9075-4.

1. *Rationale: Use the correct term "temporary table".*

   Replace the 1st paragraph with:

   A temporary table is either a global temporary table, a created local temporary table, or a declared local temporary table.

2. *Rationale: Direct SQL may access a created local temporary table.*

   Insert after the 2nd paragraph.

   For purposes of temporary tables, if an SQL-session is executing direct SQL, then the <direct SQL statement>s executed during the SQL-session effectively comprise a separate SQL-client module.

3. *Rationale: Direct SQL may access a created local temporary table.*

   Replace the 6th paragraph with:

   An SQL-client module declared local temporary table is a declared local temporary table defined in an <SQL-client module definition> or in a <direct SQL statement>. (Note that the <direct SQL statement>s executed by an SQL-session that is executing direct SQL effectively comprise a distinct SQL-client module.). An SQL-client module declared local temporary table is effectively materialized the first time any <externally-invoked procedure> in the <SQL-client module definition> that contains the <temporary table declaration> is executed, or when the <temporary table declaration> is executed as a <direct SQL statement>, and it persists for that SQL-session. Every SQL-client module in every SQL-session that references an SQL-client module declared local temporary table causes a distinct instance of that declared local temporary table (*i.e.*, a multiset of rows that is visible only to that SQL-client module during that SQL-session) to be materialized. That is, the multiset of rows that is referenced by the <table name> of an SQL-client module declared local temporary table cannot be shared between SQL-sessions, nor between SQL-client modules that execute during an SQL-session.

4.15.6 Table updatability

1. *Rationale: Tables identified by a <query name> can be insertable-into.*

   Replace the 2nd paragraph with:

   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*. Transition tables are not insertable-into. A table identified by a <query name> may be insertable-into, as specified by the Syntax Rules of Subclause 7.13, “<query expression>”. The Syntax Rules of Subclause 7.6, “<table reference>”, determine whether a derived table *T* is insertable-into. A view is *insertable-into* if the derived table that defines the view is insertable-into. A view is *trigger insertable-into* if an insert INSTEAD OF trigger is defined on that view.

4.15.7 Table descriptors

This Subclause is modified by Subclause 4.10.4, “Table descriptors”, in ISO/IEC 9075-9.

1. *Rationale: Use the correct term "descriptor".*

   Replace the 3rd item in the base table descriptor with:

   — The descriptor of each period defined for the table.

2. *Rationale: The original <query expression> defines rows with no subrows in proper subtables.*

   Replace the 3rd item in the view descriptor with:

   — The original <query expression> of the view. (This <query expression> is used to find the rows of the view that have no subrows in proper subtables of the view.)

4.15.8 Relationships between tables

This Subclause is modified by Subclause 4.10.5, “Relationships between tables”, in ISO/IEC 9075-9.

1. *Rationale: Define target underlying tables*

   Append the following text:

   The terms *target simply underlying table*, *target underlying table*, *target leaf underlying table*, and *target leaf generally underlying table* also define a relationship between a derived table or cursor and other tables.

   The target simply underlying tables of effectively updatable derived tables are defined in the Syntax Rules of Subclause 7.12, “<query specification>”, and Subclause 7.13, “<query expression>”. The target simply underlying table specification of a <table or query name> is the <table name> or <query name> simply contained in the <table or query name>. (A <transition table name> is not a target underlying table.) A <table name> has no target simply underlying tables. The target simply underlying table of a <query name> *QN* is the <query expression body> *QEB* of the <with list element> identified by *QN*. The target simply underlying table of an updatable cursor is the <cursor specification> included in the cursor's result set descriptor (or, for standing cursors, the <cursor specification> included in the cursor declaration descriptor).

   The target underlying tables of an effectively updatable derived table or updatable cursor *DTC* are the target simply underlying tables of *DTC* and the target underlying tables of the target simply underlying tables of *DTC*.

   The target leaf underlying tables of *DTC* are the target underlying tables of *DTC* that do not themselves have any target underlying tables.

   The target generally underlying tables of *DTC* are the target underlying tables of *DTC* and, for each target underlying table of *DTC* that is a <table name> *VN* that identifies a view *V*, the hierarchical <query expression> *QEV* included in the view descriptor of *V* and the target generally underlying tables of *QEV*.

The target leaf generally underlying tables of *DTC* are the target generally underlying tables of *DTC* that do not themselves have any target generally underlying tables.

4.15.9 Referenceable tables, subtables and supertables

1. *Rationale: Define proper supertables.*

   Replace the 6th paragraph with:

   A table T_b is called a *supertable* of a table T_a if T_a is a subtable of T_b. If T_a is a direct subtable of T_b, then T_b is called a *direct supertable* of T_a. If T_a is a proper subtable of T_b, then T_b is called a *proper supertable* of T_a. A table that is not a subtable of any other table is called a *maximal supertable*.

4.15.10 Operations involving tables

1. *Rationale: Use the correct term "regular base table".*

   Replace the 6th paragraph with:

   The primary effect of a <delete statement: positioned> on a regular base table *T* or a derived table *T* is to delete exactly one specified row from *T*. The primary effect of a <delete statement: searched> on a regular base table *T* or a derived table *T* is to delete zero or more rows from *T*.

2. *Rationale: Separate the description of the primary effect of a searched delete statement from that of a positioned delete statement on a system-versioned table because the number of rows for deletion is different between those two types of delete statement.*

   Replace the 7th paragraph with:

   The primary effect of a <delete statement: positioned> on a system-versioned table *T* is to replace exactly one specified row of *T* with its system-time period end column value set to the transaction timestamp of the SQL-transaction in which the <delete statement: positioned> executes.

   The primary effect of a <delete statement: searched> on a system-versioned table *T* is to replace zero or more rows of *T* with their system-time period end column values set to the transaction timestamp of the SQL-transaction in which the <delete statement: searched> executes.

4.15.11 Identity columns

1. *Rationale: There is no text describing the situation in which a value for an identity column is supplied in the <insert statement> or the <merge statement>.*

   Replace the 1st paragraph with:

The columns of a base table *BT* can optionally include not more than one *identity column*. The declared type of an identity column is either an exact numeric type with scale 0 (zero), INTEGER for example, or a distinct type whose source type is an exact numeric type with scale 0 (zero). An identity column has a *start value*, an *increment*, a *maximum value*, a *minimum value*, and a *cycle option*. An identity column is associated with an internal sequence generator *SG*. Let *IC* be the identity column of *BT*. When a row *R* is presented for insertion into *BT*, if *R* does not contain a column corresponding to *IC*, then the General Rules of Subclause 9.25, "Generation of the next value of a sequence generator", in ISO/IEC 9075-2, are applied with *SG* as *SEQUENCE*; let *V* be the *RESULT* returned from the application of those General Rules. The value for *IC* is *V*. If *R* contains a column corresponding to *IC* and the <insert statement> or the <merge statement> that inserts *R* specifies OVERRIDING SYSTEM VALUE, then the value in *R* is assigned to *IC*. If *R* contains a column corresponding to *IC* and the <insert statement> or the <merge statement> that inserts *R* specifies OVERRIDING USER VALUE, then the value in *R* is ignored and *IC* is assigned *V*. The definition of an identity column may specify GENERATED ALWAYS or GENERATED BY DEFAULT.

4.18 Integrity constraints

4.18.3 Table constraints

4.18.3.2 Unique constraints

1. *Rationale: The condition for satisfaction of an unique constraint for system-versioned tables was missing.*

   Replace the 3rd paragraph with:

   Let *T* be a base table that is not a system-versioned table. A unique constraint that does not include a <without overlap specification> on *T* is satisfied if and only if there do not exist two rows *R1* and *R2* of *T* such that *R1* and *R2* have the same non-null values in the unique columns. If a unique constraint *UC* on *T* includes a <without overlap specification> *WOS*, then let *ATPN* be the <application time period name> contained in *WOS*. *UC* is satisfied if and only if there do not exist two rows *R1* and *R2* of *T* such that *R1* and *R2* have the same non-null values in the unique columns and the *ATPN* period values of *R1* and *R2* overlap.

   Let *T* be a system-versioned table. A unique constraint that does not include a <without overlap specification> on *T* is satisfied if and only if there do not exist two current system rows *R1* and *R2* of *T* such that *R1* and *R2* have the same non-null values in the unique columns. If a unique constraint *UC* on *T* includes a <without overlap specification> *WOS*, then let *ATPN* be the <application time period name> contained in *WOS*. *UC* is satisfied if and only if there do not exist two current system rows *R1* and *R2* of *T* such that *R1* and *R2* have the same non-null values in the unique columns and the *ATPN* period values of *R1* and *R2* overlap.

   In addition, if the unique constraint was defined with PRIMARY KEY, then it requires that none of the values in the specified column or columns be a null value.

4.18.3.3 Referential constraints

1. *Rationale: The condition for satisfaction of referential constraints for system-versioned tables was missing.*

   Replace the 2nd and 3rd paragraph with:

   The choices for <match type> are MATCH SIMPLE, MATCH PARTIAL, and MATCH FULL; MATCH SIMPLE is the default. There is no semantic difference between these choices if there is only one referencing column (and, hence, only one referenced column). There is also no semantic difference if all referencing columns are not nullable. If there is more than one referencing column, at least one of which is nullable, and if no <referencing period specification> is specified, then the various <match type>s have the following semantics:

   — MATCH SIMPLE: if at least one referencing column is null, then the row of the referencing table passes the constraint check. If all referencing columns are not null and the referenced table is not a system-versioned table, then the row passes the constraint check if and only if there is a row of the referenced table that matches all the referencing columns. If all referencing columns are not null and the referenced table is a system-versioned table, then the row passes the constraint check if and only if there is a current system row of the referenced table that matches all the referencing columns.

   — MATCH PARTIAL: if all referencing columns are null, then the row of the referencing table passes the constraint check. If at least one referencing columns is not null and the referenced table is not a system-versioned table, then the row passes the constraint check if and only if there is a row of the referenced table that matches all the non-null referencing columns. If at least one referencing columns is not null and the referenced table is a system-versioned table, then the row passes the constraint check if and only if there is a current system row of the referenced table that matches all the non-null referencing columns.

   — MATCH FULL: if all referencing columns are null, then the row of the referencing table passes the constraint check. If all referencing columns are not null and the referenced table is not a system-versioned table, then the row passes the constraint check if and only if there is a row of the referenced table that matches all the referencing columns. If all referencing columns are not null and the referenced table is a system-versioned table, then the row passes the constraint check if and only if there is a current system row of the referenced table that matches all the referencing columns. If some referencing column is null and another referencing column is nonnull, then the row of the referencing table violates the constraint check.

   If there is more than one referencing column, at least one of which is nullable, and if <referencing period specification> is specified, then let *CATPN* be the <application time period name> contained in <referencing period specification> and let *PATPN* be the <application time period name> contained in the <referenced period specification>; the various <match type>s have the following semantics:

   — MATCH SIMPLE: if at least one referencing column is null, then the row of the referencing table passes the constraint check. If all referencing columns are not null and the referenced table is not a system-versioned table, then a row *R* of the referencing table passes the constraint check if and only if there is a non-empty set *S* of rows of the referenced table such that every row in *S* matches all the referencing columns of *R* and the *CATPN* period value of *R* is a subset of the union of the *PATPN* period values of the rows in *S*. If all referencing columns are not null and the referenced table is a system-versioned table, then a row *R* of the referencing table passes the constraint check if and only if there is a non-empty set *S* of current system rows of the referenced table such that every row in *S* matches all the referencing columns of *R* and the *CATPN* period value of *R* is a subset of the union of the *PATPN* period values of the rows in *S*.

— MATCH PARTIAL: if all referencing columns are null, then the row of the referencing table passes the constraint check. If at least one referencing columns is not null and the referenced table is not a system-versioned table, then a row *R* of the referencing table passes the constraint check if and only if there is a non-empty set *S* of rows of the referenced table such that every row in *S* matches all the non-null referencing columns of *R* and the *CATPN* period value of *R* is a subset of the union of the *PATPN* period values of the rows in *S*. If at least one referencing columns is not null and the referenced table is a system-versioned table, then a row *R* of the referencing table passes the constraint check if and only if there is a non-empty set *S* of current system rows of the referenced table such that every row in *S* matches all the non-null referencing columns of *R* and the *CATPN* period value of *R* is a subset of the union of the *PATPN* period values of the rows in *S*.

— MATCH FULL: if all referencing columns are null, then the row of the referencing table passes the constraint check. If all referencing columns are not null and the referenced table is not a system-versioned table, then a row *R* of the referencing table passes the constraint check if and only if there is a non-empty set *S* of rows of the referenced table such that every row in *S* matches all the referencing columns of *R* and the *CATPN* period value of *R* is a subset of the union of the *PATPN* period values of the rows in *S*. If all referencing columns are not null and the referenced table is a system-versioned table, then a row *R* of the referencing table passes the constraint check if and only if there is a non-empty set *S* of current system rows of the referenced table such that every row in *S* matches all the referencing columns of *R* and the *CATPN* period value of *R* is a subset of the union of the *PATPN* period values of the rows in *S*. If some referencing column is null and another referencing column is non-null, then the row of the referencing table violates the constraint check.

NOTE 51 — In the case that <referencing period specification> is specified, even though for a given row *R* in the referencing table there may be more than one corresponding row in the referenced table, there is at most one corresponding row in the referenced table at any given point in time in the *CATPN* period value of *R*.

4.18.3.4 Table check constraints

1. *Rationale: The condition for satisfaction of an table check constraint for system-versioned tables was missing.*

   Replace the 1st paragraph with:

   Let *T* be a base table that is not a system-versioned table. A table check constraint on *T* is satisfied if and only if the specified <search condition> evaluates to *True* or *Unknown* for every row of *T*.

   Let *T* be a system-versioned table. A table check constraint on *T* is satisfied if and only if the specified <search condition> evaluates to *True* or *Unknown* for every current system row of *T*.

   NOTE 55 — Consequently, an empty table satisfies every table check constraint that applies to it.

4.19 Functional dependencies

4.19.8 Known functional dependencies in a <joined table>

1. *Rationale: Correct the statement of functional dependencies in a <joined table>*

   Replace the 2nd bullet of the 5th paragraph with:

   — RIGHT or FULL is specified and at least one of the following is true:

   - At least one column in *A* is known not nullable and not a join partitioning column.
   - All columns in *A* and *B* are join partitioning columns.

2. *Rationale: Correct the statement of functional dependencies in a <joined table>*

   Replace the 2nd bullet of the 6th paragraph with:

   — LEFT or FULL is specified and at least one of the following is true:

   - At least one column in *A* is known not nullable and not a join partitioning column.
   - All columns in *A* and *B* are join partitioning columns.

3. *Rationale: Correct the statement of functional dependencies in a <joined table>*

   Replace the 7th paragraph with:

   If INNER is specified, *AP* is an equality AND-component of the <search condition>, one comparand of *AP* is a column reference *CR*, and the other comparand of *AP* is a <literal>, then let *CRC* be the counterparts of *CR* in *R*. Let {} denote the empty set. {} ↦ {*CRC*} is a *known functional dependency* in *R*.

4. *Rationale: Correct the statement of functional dependencies in a <joined table>*

   Replace the 8th paragraph with:

   If INNER is specified, *AP* is an equality AND-component of the <search condition>, one comparand of *AP* is a column reference *CRA*, and the other comparand of *AP* is a column reference *CRB*, then let *CRAC* and *CRBC* be the counterparts of *CRA* and *CRB* in *R*. Then {*CRAC*} ↦ {*CRBC*} is a *known functional dependency* in *R*.

5. *Rationale: Correct the statement of functional dependencies in a <joined table>*

   Delete Note 64.

6. *Rationale: Correct the statement of functional dependencies in a <joined table>*

   Insert before the 9th paragraph.

   If LEFT is specified, <join condition> is specified, and the <search condition> *SC* is deterministic, then let *DJA* be the set of <column reference>s contained in SC that reference columns of *T1* and let *PCB* be the set of join partitioning columns of *TRB*, if any. If *AP* is an equality AND-component of *SC*, one comparand of *AP* is a <literal> or a column reference to a column of *T1*, and the other comparand of *AP* is a column reference *CRB* to a column of *T2*, then let *CRBC* be the counterpart of *CRB* in *R*. Then *DJA* ∪ *PCB* ↦ {*CRBC*} is a known functional dependency in *R*.

   If RIGHT is specified, <join condition> is specified, and the <search condition> *SC* is deterministic, then let *DJB* be the set of <column reference>s contained in *SC* that reference columns of *T2* and let *PCA* be the set of join partitioning columns of *TRA*, if any. If *AP* is an equality AND-component of *SC*, one comparand of *AP* is a <literal> or a column reference to a column of *T2*, and the other comparand of *AP* is a column reference *CRA* to a column of *T1*, then let *CRAC* be the counterpart of *CRA* in *R*. Then *DJB* ∪ *PCA* ↦ {*CRAC*} is a known functional dependency in *R*.

4.19.8 Known functional dependencies in a <table primary>

1. *Rationale: Associate symbol with the <table or query name> in and <only spec>.*

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

   — If *TP* immediately contains a <table or query name> *TQN* or an <only spec> that immediately contains a <table or query name> *TQN*, then the counterparts of the BPK-sets and BUC-sets of *TQN* are the BPK-sets and BUC-sets, respectively, of *R*. If *A* ↦ *B* is a known functional dependency in the result of *TQN*, and *AC* and *BC* are the counterparts of *A* and *B*, respectively, then *AC* ↦ *BC* is a *known functional dependency* in *R*.

4.19.9 Known functional dependencies in a <table factor>

1. *Rationale: Editorial.*

   Replace the text with:

   Let *R* be the result of <table factor> *TF*. Let *S* be the result of <table primary> immediately contained in *TF*. The counterparts of the BPK-sets and BUC-sets of *S* are the BPK-sets and BUC-sets, respectively, of *R*. If *A* ↦ *B* is a functional dependency in *S*, and *AC* and *BC* are the counterparts of *A* and *B*, respectively, then *AC* ↦ *BC* is a *known functional dependency* in *R*.

4.25 Dynamic SQL concepts

4.25.2 Dynamic SQL statements and descriptor areas

1. *Rationale: Editorial correction.*

   Replace the 2nd paragraph with:

   For a statement other than a <dynamic select statement>, an <execute statement> is used to associate parameters with the prepared statement and execute it as though it had been coded when the program was written. For a <dynamic select statement>, the prepared <cursor specification> is associated with a declared dynamic cursor via a <dynamic declare cursor> or with an extended dynamic cursor via an <allocate extended dynamic cursor statement>. The dynamic cursor can be opened and dynamic parameters can be associated with the dynamic cursor with a <dynamic open statement>. Operations on an open dynamic cursor are described in Subclause 4.33.2, "Operations on and using cursors".

4.36 SQL-transactions

4.36.3 Properties of SQL-transactions

This Subclause is modified by Subclause 4.15.1, "Properties of SQL-transactions", in ISO/IEC 9075-9.

1. *Rationale: Editorial.*

   Replace the 4th paragraph with:

   An SQL-transaction has a *transaction timestamp*, a value of an implementation-defined timestamp type that is used to set the values of system-time period start and system-time period end columns of rows, if any, modified by the execution of an SQL-data change statement in this SQL-transaction. The transaction timestamp is set by an SQL-implementation before any SQL-data change statement is executed in that transaction and, once set, remains unchanged during that SQL-transaction.

4.36.7 Encompassing transactions

1. *Rationale: Use the correct term "transation isolation level".*

   Replace the 1st paragraph of the 2nd bullet of the 4th paragraph with:

   — If the SQL-transaction has a transaction isolation level of READ UNCOMMITTED, then the branch of the SQL-transaction may have a transation isolation level of READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ, or SERIALIZABLE.

4.38 SQL-sessions

4.38.4 SQL-session context management

1. *Rationale: Use the correct terms “SQL-session context”, "<module authorization clause>” and "user identifier".*

   Replace the 3rd, 4th and 5th paragraphs with:

   Each SQL-session context contains an “authorization stack” Each cell of the authorization stack is a pair, a user identifier, and a role name. Whenever an SQL-session context is created, the base of its authorization stack is initialized with the SQL-session user identifier for the user identifier and the null value for the role name.

   The <value specification>s CURRENT_USER and CURRENT_ROLE identify the user identifier and role name on top of the authorization stack in the top of the stack of SQL-session contexts. The <value specification> SESSION_USER is the value of the SQL-session user identifier in the latest SQL-session context.

   Invocation of an <externally-invoked procedure> pushes a cell on the authorization stack. If the <externally-invoked procedure> is invoked using invoker’s rights, then the new cell is a copy of the one beneath it. If it is invoked using definer's rights, the new cell is taken from <module authorization clause>, which supplies either a user identifier or a role name, the other being set to the null value.

2. *Rationale: Use the correct term "user identifier".*

   Replace the 8th paragraphs with:

   The <set role statement> changes the role name on top of the authorization stack in the latest SQL-session context, but does not change the user identifier.

5 Lexical elements

5.3 <literal>

1. *Rationale: For consistency with the rest of the document.*

   Replace General Rule 5) with:

   5) Let *ANL* be an <approximate numeric literal>. Let *ANDT* be the declared type of *ANL*. Let *ANV* be the product of the exact numeric value represented by the <mantissa> of *ANL* and the number obtained by raising the number 10 to the power of the exact numeric value represented by the <exponent> of *ANL*. If *ANV* is a value of *ANDT*, then the value of *ANL* is *ANV*; otherwise, the value of *ANL* is a member of *ANDT* obtained from *ANV* by rounding or truncation. The choice of whether to round or truncate in implementation-defined.

6 Scalar expressions

6.1 <data type>

This Subclause is modified by Subclause 6.1, "<data type>", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 6.1, "<data type>", in ISO/IEC 9075-14.

1. *Rationale: Avoid problems in the Syntax Rules by changing the BNF non-terminal.*

   Replace the BNF for "large object length" with:

```
<large object length> ::=
    <unsigned integer> [ <multiplier> ]
  | <large object length token>
```

2. *Rationale: Editorial.*

   Replace Syntax Rule 43) with:

   43) An <array type> *AT* specifies an *array type*. The <data type> immediately contained in *AT* is the *element type* of the array type. The value of the <maximum cardinality> immediately contained in *AT* is the *maximum cardinality* of a site of data type *AT*. If the maximum cardinality is not specified, then an implementation-defined maximum cardinality is implicit.

6.6 <identifier chain>

This Subclause is modified by Subclause 6.2, "<identifier chain>", in ISO/IEC 9075-4.

1. *Rationale: Handle <query expression>s containing UNION, INTERSECT and EXCEPT correctly.*

   Replace Syntax Rule 8) a) i) with:

   8) ...

      a) ...

         i) If *IC* is contained in an <order by clause> simply contained in a <query expression> *QE*, and the result of *QE* has a column *DC* whose column name is equivalent to *IC*, then PIC_1 is a candidate basis, the scope of PIC_1 is *QE*, and the referent of PIC_1 is the column *DC*.

6.10 <window function>

Subclause Signature

```
"<window function>" (
  Parameter: "WINFUNC"
) Returns: "TRANSFORM"
```

1. *Rationale: Cater for entry to the Subclause either by BNF or signature.*

   Replace Syntax Rule 1) with:

   1) Case:

      a) If this Subclause has been invoked as a "subroutine Subclause" from another Subclause, then: Let *OF* be the *WINFUNC* in an application of the Rules of this Subclause. The result of the application of this Subclause is *QSX* or *SSSRX*, as appropriate, which is returned as *TRANSFORM*.

      b) Otherwise, let *OF* be the <window function>.

2. *Rationale: Cater for entry to the Subclause either by BNF or signature.*

   Delete Syntax Rule 3).

3. *Rationale: Clarify <ntile function>.*

   Replace GR 1) a) ii) 3) E) II) with:

   1) ...

      a) ...

         ii) ...

            3) ...

               E) ...

                  II) Otherwise, for each i, 1 (one) $\leq i \leq$ MOD(CT, NT), let NQ_i be CEILING(CT / NT), and for each i, MOD(CT, NT) $< i \leq NT$, let NQ_i be FLOOR(CT / NT).

4. *Rationale: Editorial.*

   Replace GR 1) a) ii) 3) F) with:

   1) ...

      a) ...

         ii) ...

3) ...

F) Let END_0 be 0 (zero). For each i, 1 (one) $\leq i \leq NT$, let $START_i$ be $(END_{(i-1)}$ + 1) and let END_i be $(END_{(i-1)} + NQ_i)$.

5. *Rationale: Editorial.*

Replace GR 1) b) iv) 1) with:

1) ...

b) ...

iv) ...

1) Case:

A) If *NTREAT* is RESPECT NULLS, then let *TX* be the sequence of values that is the result of applying *VE1* to each row of *T* that precedes the current row, ordered according to the window ordering of *WDX*.

B) Otherwise, let *TX* be the sequence of values that is the result of applying *VE1* to each row of *T* that precedes the current row and eliminating null values, ordered according to the window ordering of *WDX*.

6.11 <nested window function>

1. *Rationale: Use the defined symbol.*

Replace General Rule 4) e) ii)

4) ...

e) ...

ii) Otherwise, the value of *NWF* is the value of *VE*, evaluated in ROW_M.

6.23 <reference resolution>

1. *Rationale: Use the correct term.*

Replace the Function with:

Obtain the value referenced by a REF value.

6.28 <numeric value function>

1. *Rationale: Add missing variants of national character type.*

   Replace Conformance Rule 4) with:

   4) Without Feature F421, “National character”, conforming SQL language shall not contain a <length expression> that simply contains a <string value expression> that has a declared type of NATIONAL CHARACTER, NATIONAL CHARACTER VARYING, or NATIONAL CHARACTER LARGE OBJECT.

6.29 <string value expression>

1. *Rationale: Editorial.*

   Replace the lead text of General Rule 2) b) ii) 3) A) with:

   2) ...

      b) ...

         ii) ...

            3) ...

               A) If the most specific type of at least one of *S1* and *S2* is a character large object type, then let *LOL* be the implementation-defined maximum length of large object character strings.

                  Case:

6.30 <string value function>

This Subclause is modified by Subclause 6.4, “<string value function>”, in ISO/IEC 9075-9.
This Subclause is modified by Subclause 6.8, “<string value function>”, in ISO/IEC 9075-14.

1. *Rationale: Specify the result correctly.*

   Replace General Rule 9) b) v) 1) C) with:

   9) ...

      b) ...

         v) ...

            1) ...

C) Otherwise, let *MV* be the *OCC*-th match vector in *LOMV*. The General Rules of Subclause 9.19, "XQuery regular expression replacement", in ISO/IEC 9075-2, are applied with *MV* as *MATCH*, *STR* as *STRING*, *PAT* as *PATTERN*, *REP* as *REPLACEMENT*, and *FL* as *FLAG*; let the result of <regex transliteration> be the *RESULT* returned from the application of those General Rules.

2. *Rationale: Editorial.*

Replace Conformance Rule 10) with:

10) Without Feature F421, "National character", conforming SQL language shall not contain a <character value function> that has a declared type of NATIONAL CHARACTER, NATIONAL CHARACTER VARYING, or NATIONAL CHARACTER LARGE OBJECT.

7 Query expressions

7.6 <table reference>

This Subclause is modified by Subclause 7.1, "<table reference>", in ISO/IEC 9075-4.
This Subclause is modified by Subclause 7.1, "<table reference>", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 7.1, "<table reference>", in ISO/IEC 9075-14.

1. *Rationale: There can be multiple occurrences of <search condition>, <set clause list> and <merge insert value list> in a MERGE statement.*

Replace Syntax Rule 9) b) with:

9) ...

b) If *TF* is simply contained in a <merge statement> *MS*, then the *scope clause SC* of *TR* is *MS*. The scope of the range variable of *TF* and of *TR* is the <search condition>s, <set clause list>s, and <merge insert value list>s of *SC*.

2. *Rationale: INSTEAD OF triggers on target generally underlying tables must be prohibited with FINAL TABLE. Also, an INSTEAD OF DELETE trigger might insert or update rows of a target table, and must be prohibited with FINAL TABLE applied to a MERGE.*

Replace Syntax Rule 15) b) ii) with:

15) ...

b) ...

ii) If FINAL is specified, then

1) If either *S* is an <insert statement> or *S* is a <merge statement> that contains a <merge insert specification>, then *TT* shall not be trigger insertable-into and no target generally underlying table of *TT* shall be trigger insertable-into.

2) If either *S* is an <update statement: searched> or *S* is a <merge statement> that contains a <merge update specification>, then *TT* shall not be trigger updatable and no target generally underlying table of *TT* shall be trigger updatable.

3) If *S* is a <merge statement> that contains a <merge delete specification>, then *TT* shall not be trigger deletable and no target generally underlying table of *TT* shall be trigger deletable.

3. *Rationale: Correctly define and use the terms "old delta table of merge operation" and "new delta table of merge operation".*

Replace General Rule 3) b) iii) with:

3) ...

b) ...

iii) If *S* is a <merge statement>, then

Case:

1) If OLD is specified, then the result of *DCDT* is the old delta table of merge operation on *TT*.

NOTE 182 — "old delta table of merge operation" is defined in Subclause 14.12, "<merge statement>".

2) Otherwise, the result of *DCDT* is the new delta table of merge operation on *TT*.

NOTE 183 — "new delta table of merge operation" is defined in Subclause 14.12, "<merge statement>".

7.7 <joined table>

1. *Rationale: Use the correct symbol RTB.*

Replace Syntax Rule 9) e) ii) 3) B) with:

9) ...

e) ...

ii) ...

3) ...

B) Otherwise, let RVB_i be some range variable that is not equivalent to any range variable in the outermost <query specification> containing the <joined table>, nor to any other range variable created by these rules. RVB_i is effectively associated with the field of *RTB* whose name is $CJCN_i$.

2. *Rationale: Insert missing ≤ symbol.*

   Replace Syntax Rule 9) e) iii) with:

   9) ...

      e) ...

         iii) Let *SLCC* be a <select list> of <derived column>s of the form

         `COALESCE (` RVA_i`.`$CJCN_i$`,` RVB_i`.`$CJCN_i$ `) AS` $CJCN_i$

         for every *i*, 1 (one) ≤ *i* ≤ *N*, in ascending order.

7.11 <window clause>

1. *Rationale: Editorial.*

   Replace General Rule 5) b) i) 1) with:

   5) ...

      b) ...

         i) ...

            1) In the following subrules, when performing addition or subtraction to combine a datetime and a year-month interval, if the result would raise the exception condition *data exception — datetime field overflow* because the <primary datetime field> DAY is not valid for the computed value of the <primary datetime field>s YEAR and MONTH, then the <primary datetime field> DAY is set to the last day that is valid for the <primary datetime field>s YEAR and MONTH, and no exception condition is raised.

2. *Rationale: Use correct symbols.*

   Replace General Rule 5) b) iii) 3) with:

   5) ...

      b) ...

         iii) ...

            3) The *distance* between two window ordering groups *WOG1* and *WOG2* is number of window ordering groups between *WOG1* and *WOG2*, inclusive, minus 1 (one).

7.12 <query specification>

This Subclause is modified by Subclause 7.2, "<query specification>", in ISO/IEC 9075-4.

1. *Rationale: Correct the transformation of <query specification>s containing <in-line windows specification>s.*

   Replace Syntax Rule 13) a) with:

   13) ...

      a) Case:

         i) If *GWQ* contains an <in-line window specification>, then:

            1) Let *NWF* be the number of <window function>s simply contained in *GWQ*.

            2) For each <window function> WF_i, 0 (zero) ≤ *i* ≤ *NWF*, simply contained in *GWQ*:

               A) The Syntax Rules of Subclause 6.10, "<window function>", in ISO/IEC 9075-2, are applied with WF_i as *WINFUNC*; let *GWQ1* be the *TRANSFORM* returned from the application of those Syntax Rules;

               B) Let *GWQ* be *GWQ1*.

         ii) Otherwise, let *GWQ1* be *GWQ*.

2. *Rationale: Remove redundant sentence.*

   Replace Syntax Rule 13) b) with:

   13) ...

      b) If the <select list> of *GWQ1* immediately contains <asterisk> or simply contains <qualified asterisk>, then Syntax Rules of Subclause 7.12, "<query specification>", in ISO/IEC 9075-2, are applied with *GWQ1* as *TBLEXP*; let *GWQ2* be the *TBLEXP2* returned from the application of those Syntax Rules; otherwise, let *GWQ2* be *GWQ1*.

3. *Rationale: Correct the definition of simply underlying tables.*

   Replace Syntax Rule 22) with:

   22) After applying all relevant syntactic transformations, let *TREF* be the <table reference>s that are simply contained in the <from clause> of the <table expression>. The *simply underlying tables* of the <query specification> are the <table or query name>s and <query expression>s contained in *TREF* without an intervening <query expression> or <data change delta table>.

4. *Rationale: Define target underlying tables.*

   Insert a new Syntax Rule.

27.1) Let *QS* be an updatable <query specification>, and let *SUT* be a simply underlying table of *QS*. Then *SUT* is a *target simply underlying table* of *QS* if at least one of the following is true:

a) *QS* is simply updatable.

b) *QS* is one-to-one with *SUT* and the SQL-implementation supports Feature T111, "Updatable joins, unions, and columns".

5. *Rationale: Clarify the definition of insertable-into tables.*

Replace Syntax Rule 28) with:

28) A <query specification> *QS* is *insertable-into* if *QS* is updatable and every simply underlying table of *QS* is insertable-into.

7.13 <query expression>

This Subclause is modified by Subclause 7.2, "<query expression>", in ISO/IEC 9075-14.

1. *Rationale: Remove an ambiguity.*

Replace Syntax Rule 3) i) ii) 2) D)

3) ...

i) ...

ii) ...

2) ...

D) WQE_i shall not contain a <query specification> *QS* such that *QS* immediately contains a <table expression> *TE* that contains a <query name> referencing WQN_j and either of the following is true:

I) *TE* immediately contains a <having clause> that contains a <set function specification>.

II) *QS* immediately contains a <select list> *SL* that contains a <window function> or a <set function specification>.

NOTE 243 — If a <window function> is contained in an <order by clause>, then the syntactic transformation in this Subclause that moves the <window function> to a <select sublist> is effectively applied before applying this rule.

2. *Rationale: Clarify that recursive queries are not updatable.*

Replace the lead text of Syntax Rule 8) with:

8) A <query expression> *QE1* is *simply updatable* if it is not contained in a <with clause> that specifies RECURSIVE, it does not specify a <result offset clause> or a <fetch first clause> and, for every

<query expression> or <query specification> *QE2* that is simply contained in the <query expression body> of *QE1*, all of the following are true:

3. *Rationale: Clarify that recursive queries are not updatable.*

   Replace the lead text of Syntax Rule 9) with:

   9) A <query expression> *QE1* is *updatable* if it is not contained in a <with clause> that specifies RECURSIVE, it does not specify a <result offset clause> or a <fetch first clause> and, for every <simple table> *QE2* that is simply contained in *QE1*, all of the following are true:

4. *Rationale: Remove incorrect use of the plural.*

   Replace Syntax Rule 19) c) with:

   19) ...

   c) If the SQL-implementation supports Feature T101, “Enhanced nullability determination” and the *i*-th column of at least one of *T1* and *T2* is known not nullable, then the *i*-th column of *TR* is known not nullable; otherwise, the *i*-th column of *TR* is possibly nullable.

5. *Rationale: Correct the definition of simply underlying tables.*

   Replace Syntax Rule 22) with:

   22) The *simply underlying table* of *QE* is the <query expression body> immediately contained in *QE*.

   22.1) After performing all relevant syntactic transformations, the *simply underlying tables* of a <query expression body> *QEB* are the <query specification>s and <query expression>s contained in *QEB* without an intervening <query expression>.

6. *Rationale: Define target underlying tables.*

   Insert a new Syntax Rule.

   22.2) If *QE* or *QEB* is effectively updatable, then the *target simply underlying tables* of *QE* or *QEB* are the simply underlying tables of *QE* or *QEB*, respectively.

7. *Rationale: Define common column names as fully qualified.*

   Replace Syntax Rule 28) d) i) 3) with:

   28) ...

   d) ...

   i) ...

   3) Let *SL* be the <select list> of *QS*. Let *SLT* be obtained from *SL* by replacing each <column reference> with its fully qualified equivalent; in the case of

common column names, each common column name is regarded as fully qualified.

8. *Rationale: Clarify how the ordering of the table specified by a <query expression> is obtained.*

Replace General Rule 5) b) with:

5) ...

b) If *QE* immediately contains an <order by clause>, then the ordering of rows in *T* is the same as the ordering of rows in the sort table of *QE* and its extended order by clause as determined by the General Rules of Subclause 10.10, "<sort specification list>". The table specified by *QE* is effectively the sort table of *QE* with all extended sort key columns (if any) removed.

NOTE 248 — "extended sort key column" and "extended order by clause" are defined in the Syntax Rules of this Subclause.

9. *Rationale: Define the cardinality of the table.*

Replace General Rule 7) d) ii) with:

7) ...

d) ...

ii) Otherwise, rows other than the first *FFRC* rows in order as specified by General Rule 6) of this Subclause are removed from *T* and the cardinality of *T* is *FFRC*.

10. *Rationale: Make Conformance Rules 19) and 20) consistent with Conformance Rules 12), 13), 14), and 16).*

Replace Conformance Rules 19) and 20) with:

19) Without Feature F862, "<result offset clause>in subqueries", in conforming SQL language, a <query expression> contained in another <query expression> shall not immediately contain a <result offset clause>.

20) Without Feature F863, "Nested <result offset clause>in <query expression>", in conforming SQL language, a <query primary> shall not immediately contain a <result offset clause>.

11. *Rationale: Insert the missing phrase "in conforming SQL language".*

Replace Conformance Rules 22) and 23) with:

22) Without Feature F866, "FETCH FIRST clause: PERCENT option", in conforming SQL language, <fetch first clause> shall not contain <fetch first percentage>.

23) Without Feature F867, "FETCH FIRST clause: WITH TIES option", in conforming SQL language, <fetch first clause> shall not contain WITH TIES.

8 Predicates

8.2 <comparison predicate>

This Subclause is modified by Subclause 7.1, "<comparison predicate>", in ISO/IEC 9075-13.

1. *Rationale: Editorial.*

   Replace the lead text of General Rule 1) b) ii) with:

   1) ...

      b) ...

         ii) If the declared types of *XV* and *YV* are array types or distinct types whose source types are array types and the cardinalities of *XV* and *YV* are *N1* and *N2*, respectively, then let X_i, 1 (one) ≤ *i* ≤ *N1*, denote a <value expression> whose value and declared type is that of the *i*-th element of *XV* and let Y_i, 1 (one) ≤ *i* ≤ *N2*, denote a <value expression> whose value and declared type is that of the *i*-th element of *YV*. The result of

```
X <comp op> Y
```

         is determined as follows:

2. *Rationale: Editorial.*

   Replace General Rule 1) b) ii) 3) with:

   1) ...

      b) ...

         ii) ...

            3) *X* = *Y* is *False* if and only if one of the following is true:

               A) *N1* ≠ *N2*

               B) *N1* = *N2* and for some *i*, 1 (one) ≤ *i* ≤ *N1*, NOT (X_i = Y_i) is *True*.

3. *Rationale: Use the correct term.*

   Replace General Rule 9) with:

   9) The result of comparing two REF values *X* and *Y* is determined by the comparison of their octets with the same ordinal position. Let L_x be the length in octets of *X* and let L_y be the length in octets of *Y*. Let X_i and Y_i, 1 (one) ≤ *i* ≤ L_x, be the values of the *i*-th octets of *X* and *Y*, respectively. *X* is equal to *Y* if and only if $L_x = L_y$ and, for all *i*, $X_i = Y_i$.

8.5 <like predicate>

1. *Rationale: Correct the matching of substrings.*

   Replace Syntax Rule 3) d) with:

   3) ...

      d) The Syntax Rules of Subclause 9.15, “Collation determination”, in ISO/IEC 9075-2, are applied with set of declared types of *CVE* and *PC* as *TYPESET*; let *LC* be the *COLL* returned from the application of those Syntax Rules. Let *LCN* be a collation equivalent to *LC* with the NO PAD characteristic.

         It is implementation-defined which collations can be used as collations for the <like predicate>.

2. *Rationale: Correct the matching of substrings.*

   Replace General Rule 3) d) ii) 4) with:

   3) ...

      d) ...

         ii) ...

             4) If the *i*-th substring specifier of *PCV*, PCV_i, is a sequence of single character specifiers, then the *i*-th substring of *MCV*, MCV_i, contains 1 (one) or more characters and

                `'`PCV_i`' = '`MCV_i`' COLLATE `*LCN*

                is *True*.

8.7 <regex like predicate>

1. *Rationale: Replace undefined symbols with the correct symbols or values.*

   Replace General Rule 1) b) with:

   1) ...

      b) The General Rules of Subclause 9.18, “XQuery regular expression matching”, in ISO/IEC 9075-2, are applied with *CVE* as *STRING*, *PAT* as *PATTERN*, *1* as *POSITION*, *CHARACTERS* as *UNITS*, and *FL* as *FLAG*; let *LOMV* be the *LIST* returned from the application of those General Rules.

8.17 <submultiset predicate>

1. *Rationale: Correct <submultiset predicate>.*

   Replace General Rules 3) c) ii) and 3) c) iii) with:

   3) ...

      c) ...

         ii) If there exists an enumeration CE_i for 1 (one) ≤ i ≤ M of the elements of CV and an enumeration ME_j for 1 (one) ≤ j ≤ N of the elements of MV such that for all i, 1 (one) ≤ i ≤ M, $CE_i = ME_i$, then the <submultiset predicate> is *True*.

         iii) If there exist an enumeration CE_i for 1 (one) ≤ i ≤ M of the elements of CV and an enumeration ME_j for 1 (one) ≤ j ≤ N of the elements of MV such that for all i, 1 (one) ≤ i ≤ M, $CE_i = ME_i$ is either *True* or *Unknown*, then the <submultiset predicate> is *Unknown*.

9 Additional common rules

9.3 Passing a value from a host language to the SQL-server

1. *Rationale: Correct typography and use the correct symbols for the programming languages.*

   Replace General Rule 4) a) with:

   4) ...

      a) If *DT* contains <locator indication>, then

         Case:

         i) If *LANG* is ADA, then `Interfaces.SQL.INT`.

         ii) If *LANG* is C, then `unsigned long`.

         iii) If *LANG* is COBOL, then `PIC S9(9) USAGE IS BINARY`.

         iv) If *LANG* is FORTRAN, then `INTEGER`.

         v) If *LANG* is M, then character.

         vi) If *LANG* is PASCAL, then `INTEGER`.

         vii) If *LANG* is PLI, then `FIXED BINARY(31)`.

9.4 Passing a value from the SQL-server to a host language

1. *Rationale: Correct typography and use the correct symbols for the programming languages.*

   Replace General Rule 3) a) with:

   3) ...

      a) If *DT* contains <locator indication>, then

         Case:

         i) If *LANG* is ADA, then `Interfaces.SQL.INT`.

         ii) If *LANG* is C, then `unsigned long`.

         iii) If *LANG* is COBOL, then `PIC S9(9) USAGE IS BINARY`.

         iv) If *LANG* is FORTRAN, then `INTEGER`.

         v) If *LANG* is M, then character.

         vi) If *LANG* is PASCAL, then `INTEGER`.

         vii) If *LANG* is PLI, then `FIXED BINARY(31)`.

2. *Rationale: Remove incorrect word "respectively".*

   Replace General Rule 5) a) with:

   5) ...

      a) If *DT* contains <locator indication>, then let the value of *PI* be the binary large object locator value, the character large object locator value, the array locator value, the multiset locator value, or the user-defined type locator value that uniquely identifies *SV*.

9.5 Result of data type combinations

This Subclause is modified by Subclause 9.3, "Result of data type combinations", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 10.3, "Result of data type combinations", in ISO/IEC 9075-14.

1. *Rationale: Add missing word "field".*

   Replace Syntax Rule 3) h) i) with:

   3) ...

      h) ...

         i) If the field in the same ordinal position as FD_i in every row type in *DTS* have the same name *F*, then the <field name> in FD_i is *F*.

9.12 Grouping operations

This Subclause is modified by Subclause 9.7, “Grouping operations”, in ISO/IEC 9075-9.
This Subclause is modified by Subclause 10.9, “Grouping operations”, in ISO/IEC 9075-14.

1. *Rationale: The referencing column and the corresponding referenced column of a referential constraint must have the same collation.*

   Replace Syntax Rule 2) j) with:

   2) ...

      j) A referencing column or a referenced column of a <referential constraint definition>.

9.15 Collation determination

1. *Rationale: The referencing column and the corresponding referenced column of a referential constraint must have the same collation.*

   Replace Syntax Rule 4) a) with:

   4) ...

      a) If the comparison operation is a <referential constraint definition>, then, for each referencing column *RC* and its corresponding referenced column *RFC*, the declared type collation of *RC* and the declared type collation of *RFC* shall be the same collation *COLL* and the collation to be used is *COLL*.

9.16 Execution of array-returning functions

This Subclause is modified by Subclause 8.5, “Execution of array-returning functions”, in ISO/IEC 9075-13.

1. *Rationale: Correct typography.*

   Replace General Rule 7) a) with:

   7) ...

      a) Depending on whether the language of *P* specifies ADA, C, COBOL, FORTRAN, M, PASCAL, or PLI, let the *operative data type correspondences table* be Table 16, “Data type correspondences for Ada”, Table 17, “Data type correspondences for C”, Table 18, “Data type correspondences for COBOL”, Table 19, “Data type correspondences for Fortran”, Table 20, “Data type correspondences for M”, Table 21, “Data type correspondences for Pascal”, or Table 22, “Data type correspondences for PL/I”, respectively. Refer to the two columns of the operative data type correspondences table as the “SQL data type” column and the “host data type” column.

2. *Rationale: Make the notation for all calls the same.*

   Replace the lead text of General Rule 9) with:

   9) [13] If the call type data item has a value of –1 (indicating *open call*), then:

9.18 XQuery regular expression matching

1. *Rationale: Editorial.*

   Replace NOTE 299 with:

   NOTE 299 — For example, if *S* is `'a'`, then there are four position/lengths, namely (1,0), denoting the zero-length substring at the beginning of *S*; (2,0), denoting the zero-length substring at the end of *S*; (1,1), denoting the whole string *S*; and (0,0), denoting no substring of *S*.

9.30 Determination of view component privileges

1. *Rationale: Eliminate recursive WITH clauses as recursive queries are not updateable and so do not need view component privileges.*

   Replace General Rule 3) b) with:

   3) ...

      b) For all *i* and *j* between 1 (one) and *N*, if VC_i is a <query expression> simply contained in a <with list element> that is simply contained in a <with clause> that does not specify RECURSIVE and VC_j references the table defined by VC_i, then $i < j$.

2. *Rationale: A view component that is not updatable has no DML privileges.*

   Replace the lead text of General Rule 4) a) with:

   4) ...

      a) If VC_i is not updatable, or if VC_i is not simply updatable and the SQL-implementation does not support Feature T111, “Updatable joins, unions, and columns”, then there are no view component privilege descriptors that identify VC_i as object.

      a.1) If VC_i is a <query specification>, then:

      Case:

3. *Rationale: Make VC_i a reference rather than a definition.*

   Replace the lead text of General Rule 4) a) i) with:

4) ...

a) ...

i) If the <from clause> of VC_i contains exactly one <table reference> *TR*, then

4. *Rationale: "one-to-one" is not always defined; use "target leaf underlying table" instead.*

Replace General Rule 4) a) ii) 1) with:

4) ...

a) ...

ii) ...

1) If, for every target leaf underlying table *LUT* of VC_i, the applicable privileges for *A* include DELETE on *LUT*, then a view component table privilege descriptor is created whose identified object is VC_i, action is DELETE, grantor is the special grantor value "_SYSTEM", and the grantee is *A*. The privilege is grantable if and only if the applicable privileges for *A* includes grantable DELETE privilege on each such *LUT*. The privilege descriptor is immediately dependent on every privilege descriptor whose identified object is such a target leaf underlying table, the action is DELETE, and grantee is *A*.

5. *Rationale: "one-to-one" is not always defined; use "target leaf underlying table" instead.*

Replace the lead text of General Rule 4) a) ii) 2) with:

4) ...

a) ...

ii) ...

2) For each updatable column *C* of VC_i, let *LUT* be the target leaf underlying table of VC_i that has a counterpart *CC* to *C*.

6. *Rationale: Complete the rules for the determination of immediately dependent.*

Replace General Rules 4) a) i) 1) and 4) a) i) 2) with:

4) ...

a) ...

i) ...

1) Case:

A) If *TR* is a <table name> *TN* or an <only spec> that simply contains a <table name> *TN*, then let *S* be the set of applicable privileges for *A* on the table referenced by *TN*.

B) If *TR* is a <query name> *QN* or an <only spec> that simply contains a <query name> *QN*, then *QN* references some VC_j, where $j < i$. Let *S* be the set of view component privilege descriptors whose identified object is VC_j or a column of VC_j.

NOTE 304.1 — If *QN* is defined in a WITH clause that specifies RECURSIVE, then the table identified by *QN* is not effectively updatable, so the <query specification> VC_i is also not effectively updatable, and therefore VC_i has no view component privilege descriptors, by a prior rule.

C) If *TR* is a <derived table> or <lateral derived table>, then let *QE* be the <query expression> simply contained in *TR*. *TR* is some VC_j, where $j < i$. Let *S* be the set of view component privilege descriptors whose identified object is VC_j or a column of VC_j.

D) If *TR* is a <parenthesized joined table>, then let *JT* be the <joined table> simply contained in *TR*. Let *QS* be the <query specification> obtained from VC_i by replacing *TR* with *JT*. The rules of this Subclause are effectively applied as if VC_i were replaced by *QS*.

NOTE 304.2 — NOTE nnn: This will reclassify VC_i as a <query specification> with a <from clause> containing more than one <table reference>, considered in subsequent rules. The list of view components VC_1, ... VC_N is not changed by this transformation.

NOTE 304.3 — All other cases of <table reference> are not effectively updatable so they cannot arise here.

2) If *S* contains a table privilege descriptor *PD* whose action is DELETE, then a view component table privilege descriptor is created as follows: the identified object is VC_i, the action is DELETE, the grantor is the special grantor value "_SYSTEM", and the grantee is *A*. The privilege is grantable if and only if *PD* indicates a grantable privilege. The privilege descriptor is immediately dependent on *PD*.

10 Additional common elements

10.4 <routine invocation>

This Subclause is modified by Subclause 8.1, "<routine invocation>", in ISO/IEC 9075-4.
This Subclause is modified by Subclause 7.1, "<routine invocation>", in ISO/IEC 9075-10.
This Subclause is modified by Subclause 8.3, "<routine invocation>", in ISO/IEC 9075-13.
This Subclause is modified by Subclause 11.1, "<routine invocation>", in ISO/IEC 9075-14.

Subclause Signature

```
"<routine invocation>" (
  Parameter: "ROUTINE INVOCATION",
  Parameter: "SQLPATH",
  Parameter: “UDT”
)

“<routine invocation>” (
  Parameter: “STATIC SQL ARG LIST”,
  Parameter: “SUBJECT ROUTINE”
)
```

1. *Rationale: Clarify the behaviour of the Subclause when directly invoked.*

   Replace Syntax Rule 1) with:

   1) Case:

      a) If this Subclause has been invoked as a “subroutine Subclause” from another Subclause, then:

         Let *RI* be the *ROUTINE INVOCATION*, let *TP* be the *SQLPATH*, and let *UDTSM* be the *UDT* in an application of the Rules of this Subclause.

      b) Otherwise, let *RI* be the <routine invocation>, let *TP* be undefined, and let *UDTSM* be undefined.

2. *Rationale: Use the correct symbol.*

   Replace the lead text of Syntax Rule 9) with:

   9) If *RI* is immediately contained in a <call statement>, then:

      Case:

3. *Rationale: The wording “shall contain” must be “contains”, because the rule describes a condition.*

   Replace Syntax Rule 9) a) ii) with:

   9) ...

      a) ...

         ii) If the <SQL argument list> does not contain any <SQL argument> that is a <named argument specification>, then for each *k*, $NA < k \leq SIRNA$, the routine descriptor of *SIR* contains an indication that the *k*-th SQL parameter has a default value.

4. *Rationale: Define undefined symbol P_i.*

   Replace the lead text of Syntax Rule 9) g) with:

   9) ...

      g) Let *SRNP* be the number of SQL parameters of *SR*. Let P_i, 1 (one) $\leq i \leq SRNP$, be the *i*-th SQL parameter of *SR*.

5. *Rationale: Use the correct symbol for the correct parameter.*

   Replace Syntax Rule 9) g) iii) 3) with:

   9) ...

      g) ...

         iii) ...

            3) 04 If XA_i is an <SQL parameter reference>, a <column reference>, or a <target array element specification>, then the Syntax Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2, are applied with P_i as *TARGET* and XA_i as *VALUE*.

6. *Rationale: Use the correct symbol.*

   Replace the lead text of Syntax Rule 10) with:

   10) If *RI* is not immediately contained in a <call statement>, then

       Case:

7. *Rationale: Use the correct symbol for the correct parameter.*

   Replace Syntax Rule 10) b) ix) with:

   10) ...

       b) ...

          ix) For each P_i, the Syntax Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2, are applied with P_i as *TARGET* and A_i as *VALUE*.

8. *Rationale: Use the correct symbol.*

   Replace Syntax Rule 12) a) with:

   12) ...

       a) If *RI* is not immediately contained in a <call statement> and <SQL argument list> does not immediately contain at least one <SQL argument>, then let the *static SQL argument list* of *RI* be an empty list of SQL arguments.

9. *Rationale: Clarify the behaviour of the Subclause when directly invoked.*

   Replace General Rule 1) with:

   1) Case:

      a) If this Subclause has been invoked as a "subroutine Subclause" from another Subclause, then:

Let *SAL* be the *STATIC SQL ARG LIST* and let *SR* be the *SUBJECT ROUTINE* in an application of the Rules of this Subclause.

b) Otherwise, let *SAL* be the static argument list determined by the Syntax Rules of this Subclause, and let *SR* subject routine determined by the syntax rules of this Subclause.

10. *Rationale: Initialise "save area data item" with a compatible value.*

Replace General Rule 8) b) i) 1) M) with:

8) ...

b) ...

i) ...

1) ...

M) If *R* is an array-returning external function or a multiset-returning external function, then set the value of the save area data item (that is, SQL argument value list entry (*PN*+*FRN*)+(*N*+*FRN*)+5) to an arbitrary string and set the value of the call type data item (that is, SQL argument value list entry (*PN*+*FRN*)+(*N*+*FRN*)+6) to –1.

10.9 <aggregate function>

This Subclause is modified by Subclause 11.2, "<aggregate function>", in ISO/IEC 9075-14.

1. *Rationale: Define the undefined terms.*

Insert a new Syntax Rule.

11) A <value expression> *VE* simply contained in *AF* is an *aggregated argument* of *AF* if either *AF* is not an <ordered set function> or *VE* is simply contained in a <within group specification>; otherwise, *VE* is a *non-aggregated argument* of *SFE*.

2. *Rationale: Correct the algorithm for PERCENTILE_CONT and define an undefined symbol.*

Replace General Rule 9) h) with:

9) ...

h) Case:

i) If *TXA* is empty, then the result is the null value.

ii) If PERCENTILE_CONT is specified, then:

1) Let *N* be the cardinality of *TXA*.

2) Let *ROW0* be the greatest exact numeric value with scale 0 (zero) that is less than or equal to *NVE**(*N*–1). Let *ROWLIT0* be a <literal> representing *ROW0*.

3) Let *ROW1* be the least exact numeric value with scale 0 (zero) that is greater than or equal to *NVE**(*N*–1). Let *ROWLIT1* be a <literal> representing *ROW1*.

4) Let *FACTOR* be an <approximate numeric literal> representing 1+*NVE**(*N*–1)–*ROW0*.

5) The result is the result of the <scalar subquery>

```
 ( WITH TEMPTABLE(X, Y) AS
       ( SELECT ROW_NUMBER()
                  OVER (ORDER BY WSP) - 1,
                TXCOLNAME
         FROM TXANAME )
SELECT CAST ( T0.Y + FACTOR * (T1.Y - T0.Y) AS DT )
FROM TEMPTABLE T0, TEMPTABLE T1
WHERE T0.X = ROWLIT0
  AND T1.X = ROWLIT1 )
```

NOTE 330 — Although ROW_NUMBER is non-deterministic, the values of T0.Y and T1.Y are determined by this expression. Note that the only column of *TXA* is completely ordered by *WSP*. If *NVE**(*N*–1) is a whole number, then the rows selected from T0 and T1 are the same and the result is just T0.Y. Otherwise, the subquery performs a linear interpolation between the two consecutive values whose row numbers in the ordered set, seen as proportions of the whole, bound the argument of the PERCENTILE_CONT operator.

iii) If PERCENTILE_DISC is specified, then

1) If the <ordering specification> simply contained in *WSP* is DESC, then let *MAXORMIN* be MAX; otherwise let *MAXORMIN* be MIN.

2) Let *NVELIT* be a <literal> representing the value of *NVE*.

3) The result is the result of the <scalar subquery>

```
 ( SELECT MAXORMIN (TXCOLNAME)
  FROM ( SELECT TXCOLNAME,
                CUME_DIST() OVER (ORDER BY WSP)
  FROM TXANAME ) AS TEMPTABLE (TXCOLNAME, CUMEDIST)
  WHERE CUMEDIST >= NVELIT )
```

11 Schema definition and manipulation

11.2 <drop schema statement>

This Subclause is modified by Subclause 9.2, "<drop schema statement>", in ISO/IEC 9075-4.
This Subclause is modified by Subclause 11.2, "<drop schema statement>", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 9.1, "<drop schema statement>", in ISO/IEC 9075-13.

1. *Rationale: Remove redundant word.*

   Replace NOTE 336 with:

   NOTE 336 — If CASCADE is specified, then such objects will be dropped implicitly by the <revoke statement> and/or explicitly by the SQL-schema manipulation statements specified in the General Rules of this Subclause.

11.3 <table definition>

This Subclause is modified by Subclause 9.3, "<table definition>", in ISO/IEC 9075-4.
This Subclause is modified by Subclause 11.3, "<table definition>", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 9.2, "<table definition>", in ISO/IEC 9075-13.

1. *Rationale: To clarify that the attributes of the derived representation are a distinguished subset of the attributes of the structured type.*

   Replace the lead text of Syntax Rule 11) c) vi) with:

   11) ...

   c) ...

   vi) If *RST* has a derived representation, then let *m* be the number of derivational attributes of the derived representation of *RST* and let A_i, 1 (one) ≤ *i* ≤ *m*, be those derivational attributes.

2. *Rationale: A base table descriptor does not include an indication of updatability (all base tables are updatable).*

   Replace General Rule 4) n) with:

   4) ...

   n) Case:

   i) If <typed table clause> is not specified, then an indication that *T* is insertable-into.

   ii) Otherwise,

   Case:

   1) If the data type descriptor of *R* indicates that *R* is instantiable, then an indication that *T* is insertable-into.

   2) Otherwise, an indication that *T* is not insertable-into.

11.4 <column definition>

This Subclause is modified by Subclause 9.4, "<column definition>", in ISO/IEC 9075-4.

This Subclause is modified by Subclause 12.1, "<column definition>", in ISO/IEC 9075-14.

1. *Rationale: Describe the sequence generator descriptor.*

   Insert a new General Rule.

   3.1) If <identity column specification> is specified, then the General Rules of Subclause 9.26, "Creation of a sequence generator", in ISO/IEC 9075-2, are applied with *SGO* as *OPTIONS* and *ICT* as *DATA TYPE*; let the sequence generator descriptor *DSG* be the *SEQGENDESC* returned from the application of those General Rules

2. *Rationale: Describe the sequence generator descriptor.*

   Replace General Rule 4) f) iv) with:

   4) ...

      f) ...

         iv) *DSG*

11.7 <unique constraint definition>

This Subclause is modified by Subclause 11.4, "<unique constraint definition>", in ISO/IEC 9075-9.

1. *Rationale: Correct the updating of the known not nullable characteristic when the Subclause is invoked from <add table constraint definition>.*

   Replace Syntax Rule 7) with:

   7) If the <unique specification> specifies PRIMARY KEY, then for each column descriptor identified by a <column name> in the explicit or implicit <unique column list> for which the nullability characteristic is not known not nullable, the nullability characteristic of the column descriptor is changed to known not nullable.

11.8 <referential constraint definition>

1. *Rationale: The referencing column and the corresponding referenced column of a referential constraint must have the same collation.*

   Replace Syntax Rule 9) with:

   9) Each referencing column and its corresponding referenced column are operands of a grouping operation. The Syntax Rules and Conformance Rules of Subclause 9.12, "Grouping operations", apply.

11.16 <drop column not null clause>

1. *Rationale: A column of the primary key cannot lose its known not nullable nullability characteristic.*

   Insert a new Syntax Rule.

   3.1) Let *CN* be the column name of *C*. *T* shall not include a table constraint descriptor that indicates that the constraint was defined with PRIMARY KEY and that includes *CN* as a name of a unique column.

11.19 <alter column data type clause>

This Subclause is modified by Subclause 11.6, "<alter column data type clause>", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 12.3, "<alter column data type clause>", in ISO/IEC 9075-14.

1. *Rationale: <alter column data type clause> should not change a column's character set or collation.*

   Delete Syntax Rule 11).

2. *Rationale: <alter column data type clause> should not change a column's character set or collation.*

   Insert a new Syntax Rule.

   12.1) Case:

   a) If <data type> contains <character string type> *CST*, then:

      i) Case:

         1) If <data type> contains <character set specification>, then let *CSS* be that <character set specification>.

         2) Otherwise, let *CSS* be a <character set specification> that identifies the character set of *DTC*.

      ii) Case:

         1) If <data type> contains <collation name>, then let *COLL* be that <collation name>.

         2) Otherwise, let *COLL* be a <collation name> that identifies the collation of *DTC*.

      iii) Let *D* be the data type specified by

```
CST CHARACTER SET CSS COLLATION COLL
```

   b) If <data type> contains <national character string type> *NCST*, then

      i) Case:

         1) If <data type> contains <collation name>, then let *COLL* be that <collation name>.

2) Otherwise, let *COLL* be a <collation name> that identifies the collation of *DTC*.

ii) Let *D* be the data type specified by

```
NCST COLLATION COLL
```

c) Otherwise, let *D* be the data type specified by <data type>.

3. *Rationale: <alter column data type clause> should not change a column's character set or collation.*

Insert a new Syntax Rule.

15) ...

a) ...

v) The character set and collation of *D* shall be the character set and collation of *DTC*.

11.26 <drop table constraint definition>

This Subclause is modified by Subclause 9.9, "<drop table constraint definition>", in ISO/IEC 9075-4.

1. *Rationale: To clarify that the attributes of the derived representation are a distinguished subset of the attributes of the structured type.*

Replace Syntax Rule 9) with:

9) If *T* is a referenceable table with a derived self-referencing column, then *TC* shall not be a unique constraint whose unique columns correspond to the derivational attributes of the derived representation of the reference type whose referenced type is the structured type of *T*.

2. *Rationale: It is the action on the <table constraint> which causes the nullability characteristic to change not the <table constraint> itself.*

Replace Syntax Rule 10) b) with:

10) ...

b) Let *ATPN* be the <application time period name> included in *ATPD*. Destruction of *TC* shall not cause the nullability characteristic of the *ATPN* start column of *T* or the *ATPN* end column of *T* to change from known not nullable to possibly nullable.

3. *Rationale: Prevent the destruction of a table constraints, if that destruction leads to certain columns becoming possibly nullable.*

Add a new Syntax Rule.

10.1) Destruction of *TC* shall not cause the nullability characteristic of any of the following columns of *T* to change from known not nullable to possibly nullable:

a) A column that is a constituent of the primary key of *T*, if any,

b) The system-time period start column, if any,

c) The system-time period end column, if any,

d) The identity column, if any.

4. *Rationale: Make the referenced rules more explicit.*

   Replace NOTE 367 with:

   NOTE 367 — If CASCADE is specified, then any such dependent object will be dropped implicitly by the <revoke statement> and/or explicitly by the SQL-schema manipulation statements specified in the General Rules of this Subclause.

11.27 <add table period definition>

1. *Rationale: Reference the descriptor not the object.*

   Replace Syntax Rule 4) a) with:

   4) ...

      a) The table descriptor of *T* shall not include a system-time period descriptor.

11.32 <view definition>

This Subclause is modified by Subclause 9.11, "<view definition>", in ISO/IEC 9075-4.
This Subclause is modified by Subclause 9.3, "<view definition>", in ISO/IEC 9075-13.
This Subclause is modified by Subclause 12.4, "<view definition>", in ISO/IEC 9075-14.

1. *Rationale: Clarification regarding containment.*

   Replace Syntax Rule 18) with:

   18) If WITH LOCAL CHECK OPTION is specified, then *QE* shall not generally contain a <query expression> *QE2* or a <query specification> *QS2* that is possibly non-deterministic unless *QE2* or *QS2* is generally contained in the hierarchical <query expression> of a viewed table that is a leaf underlying table of *QE*.

       If WITH CASCADED CHECK OPTION is specified, then *QE* shall not generally contain a <query expression> or <query specification> that is possibly non-deterministic.

2. *Rationale: Clarify that a referenceable view may not contain an <aggregate function>.*

   Insert a new Syntax Rule.

   21) ...

q.1) *QS* shall not generally contain an <aggregate function>

3. *Rationale: To clarify that the attributes of the derived representation are a distinguished subset of the attributes of the structured type.*

Replace Syntax Rule 21) r) ii) with:

21) ...

r) ...

ii) Otherwise, *RST* has a derived representation.

1) Let *u* be the number of attributes of the derived representation of *RST*. Let C_i, 1 (one) ≤ *i* ≤ *u*, be the columns of *V* that correspond to the attributes of the derived representation of *RST*.

2) For each *i*, 1 (one) ≤ *i* ≤ *u*, the <value expression> simply contained in the <derived column> that defines C_i shall be a column reference.

3) *TQN* shall have a candidate key consisting of some subset of the underlying columns of C_i, 1 (one) ≤ *i* ≤ *u*.

4. *Rationale: Clarify the applicability of Feature T111 when creating a subview.*

Replace Syntax Rule 21) s) ii) with:

21) ...

s) ...

ii) Case:

1) If the SQL-implementation supports Feature T111, “Updatable joins, unions, and columns” and *SUPERT* is updatable, then *QS* shall be updatable.

2) If the SQL-implementation does not support Feature T111, “Updatable joins, unions, and columns” and *SUPERT* is simply updatable, then *QS* shall be simply updatable.

5. *Rationale: Editorial.*

Replace NOTE 384 with:

NOTE 384 — This ensures that the updatable columns of *SUPERT* can be determined when the view is created and will not be subject to change as a result of adding the subview. It also ensures that UPDATE column privileges on *SUPERT* can be established solely by examining the original <query expression> of *SUPERT*, and need not change as the result of adding a subview. In more detail, the rule says that if a column of a referenceable view *RV* is a column reference to a column of the basis table of *RV*, then in every subview *SUBRV*, that column must be a column reference to the corresponding column in the basis table of *SUBRV*.

6. *Rationale: Clarify that adding a subview may not violate a dependency of an SQL-schema object on some property of the subview's supertable.*

   Insert a new Syntax Rule.

   21) ...

   s) ...

   v.1) For every

   1) Original <query expression> *OBJ* of the view descriptor of any view.

   2) <search condition> *OBJ* of any constraint descriptor or assertion descriptor.

   3) <triggered action> *OBJ* of any trigger descriptor.

   4) SQL routine body *OBJ* of any routine descriptor.

   5) <parameter default> *OBJ* of any SQL parameter of any routine descriptor.

   OBJ shall still satisfy the Syntax Rules and Conformance Rules applicable to *OBJ* under the assumption that *V* is created as a subtable of *SUPERT*.

   NOTE 386.1 — This prevents the following general scenario:

   1) create a referenceable view *SUPERT*;

   2) create an SQL-schema object *OBJ* that depends on some property *P* of *SUPERT*;

   3) create a subtable of *SUPERT* that lacks the property *P* on which *OBJ* depends, thereby invalidating *OBJ*.

   Examples of properties *P* are: being possibly non-deterministic; not invoking an SQL-invoked routine that possibly reads SQL-data; or the usage of domains.

7. *Rationale: To clarify that the attributes of the derived representation are a distinguished subset of the attributes of the structured type.*

   Replace Syntax Rule 21) t) i) with:

   21) ...

   t) ...

   i) Let *r* be the number of <view column option>s. For every <view column option> VCO_j, 1 (one) ≤ *j* ≤ *r*, <column name> shall be equivalent to the <attribute name> of some originally-defined attribute of *ST*.

8. *Rationale: Use the correct term.*

   Replace General Rule 1) b) i) with:

   1) ...

   b) ...

i) If *RMSV* has a derived representation, then let *SL* be the <select list> simply contained in *QS*, and let *TE* be the <table expression> simply contained in *QS*. Let *IDV* be an implementation-dependent <value expression> that computes the REF value that references a row of *V*. Let *OQE* be the <query expression>

```
SELECT IDV, SL
TE
```

9. *Rationale: Remove incorrect word.*

Replace General Rule 1) e) ii) 1) A) I)

1) ...

e) ...

ii) ...

1) ...

A) ...

I) If <self-referencing column name> is specified, then <self-referencing column name>.

10. *Rationale: Complete the descrition of the view descriptor.*

Insert new General Rules.

1) ...

k) If *V* is insertable-into, then an indication that *V* is insertable-into.

l) If *V* is updatable, then an indication that *V* is updatable.

m) If *V* is simply updatable, then an indication that *V* is simply updatable.

11.34 <domain definition>

This Subclause is modified by Subclause 11.8, "<domain definition>", in ISO/IEC 9075-9.

1. *Rationale: Descriptors not objects are included in descriptors.*

Replace General Rule 4) with:

4) A privilege descriptor is created that defines the USAGE privilege on this domain to the <authorization identifier> *A* of the schema or SQL-client module in which the <domain definition> appears. This privilege is grantable if and only if the applicable privileges for *A* include a grantable REFERENCES privilege for each <column reference> contained in the <search condition> of every domain constraint descriptor included in the domain descriptor and a grantable USAGE privilege for each <domain name>, <collation name>, <character set name>, and <transliteration name> contained in the <search

condition> of every domain constraint descriptor included in the domain descriptor. The grantor of the privilege descriptor is set to the special grantor value "_SYSTEM".

11.49 <trigger definition>

This Subclause is modified by Subclause 9.19, "<trigger definition>", in ISO/IEC 9075-4.

1. *Rationale: The complete set of restrictions for defining an INSTEAD OF trigger on a view were not enforced.*

   Replace Syntax Rule 15 with:

   15) If INSTEAD OF is specified, then:

       a) <triggered when clause> shall not be specified.

       b) <trigger column list> shall not be specified.

       c) <table name> shall not identify a view that is either recursive or referenceable, or whose view descriptor includes an indication that either WITH LOCAL CHECK OPTION or WITH CASCADED CHECK OPTION has been specified.

       d) <table name> shall not identify a view that is a generally underlying table of the original <query expression> of a view whose view descriptor includes an indication that either WITH LOCAL CHECK OPTION or WITH CASCADED CHECK OPTION has been specified.

       e) <table name> shall not identify a view that is identified by a <target table> or <insertion target> *TT* of a <data change statement> *DCS* contained in a <data change delta table> that specifies FINAL, or is a target generally underlying table of *TT*, where *DCS* is contained in any of the following:

          i) The SQL routine body of any SQL routine

          ii) The triggered action of any trigger descriptor

11.50 <drop trigger statement>

1. *Rationale: Syntax Rule 2) must use the BNF term.*

   Replace Syntax Rule 2) with:

   2) The schema identified by the explicit or implicit <schema name> of *TRN* shall include the descriptor of *TR*.

11.51 <user-defined type definition>

This Subclause is modified by Subclause 11.10, "<user-defined type definition>", in ISO/IEC 9075-9.

This Subclause is modified by Subclause 9.4, "<user-defined type definition>", in ISO/IEC 9075-13.
This Subclause is modified by Subclause 12.6, "<user-defined type definition>", in ISO/IEC 9075-14.

1. *Rationale: To clarify that the attributes of the derived representation are a distinguished subset of the attributes of the structured type.*

   Replace Syntax Rule 8) i) with:

   8) ...

      i) If <derived representation> is specified, then no two <attribute name>s in <list of attributes> shall be equivalent. The attributes identified by the <attribute name>s are the *derivational attributes of the derived representation*.

2. *Rationale: To clarify that the attributes of the derived representation are a distinguished subset of the attributes of the structured type.*

   Replace General Rule 1) f) v) 2) with:

   1) ...

      f) ...

         v) ...

            2) If <derived representation> is specified, then an indication that the referencing type of *UDT* has a derived representation, along with the list of derivational attributes of the derived representation specified by <list of attributes>.

3. *Rationale: To clarify that the attributes of the derived representation are a distinguished subset of the attributes of the structured type.*

   Replace General Rule 1) f) vi) 2) with:

   1) ...

      f) ...

         vi) ...

            2) If *DSUDT* indicates that *RSUDT* has a derived representation, then an indication that *RUDT* has a derived representation and the list of derivational attributes of the derived representation included in *DSUDT*.

4. *Rationale: Use the BNF term.*

   Replace General Rule 1) h) ix) with:

   1) ...

      h) ...

ix) The <locator indication> contained in the <returns clause> included in the *DCMS*, if any.

5. *Rationale: Use the correct name.*

Replace General 1) g) xvi) with:

1) ...

g) ...

xvi) The CURRENT_TIMESTAMP as the value of the last-altered timestamp.

11.56 <add original method specification>

1. *Rationale: Identify the correct object.*

Replace General Rule 1) f) with:

1) ...

f) For every SQL parameter declaration in *NPL*, a locator indication (if specified).

11.57 <add overriding method specification>

1. *Rationale: Use the correct subscript.*

Replace Syntax Rule 9) c) with:

9) ...

c) For i varying from 2 to N, the Syntax Rules of Subclause 9.20, "Data type identity", in ISO/IEC 9075-2, are applied with the declared type of SQL parameter $PCOMS_i$ of *UPCOMS* as *TYPE1* and the declared type of SQL parameter $POVMS_{i-1}$ of *MPDL* as *TYPE2*.

11.60 <SQL-invoked routine>

This Subclause is modified by Subclause 9.24, "<SQL-invoked routine>", in ISO/IEC 9075-4.
This Subclause is modified by Subclause 11.11, "<SQL-invoked routine>", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 9.8, "<SQL-invoked routine>", in ISO/IEC 9075-13.
This Subclause is modified by Subclause 12.8, "<SQL-invoked routine>", in ISO/IEC 9075-14.

1. *Rationale: Clarify the definitions of array-returning external function and multiset-returning external function.*

   Replace Syntax Rules 6) n) and 6) o) with:

   6) ...

      n) An *array-returning external function* is an SQL-invoked function that is an external routine and that satisfies one of the following conditions:

         i) A <result cast from type> is specified that does not contain a <locator indication> but simply contains one of the following:

            1) An <array type>.

            2) A <path-resolved user-defined type name> whose source type is an array type.

         ii) A <result cast from type> is not specified and <returns data type> does not contain a <locator indication> but simply contains one of the following:

            1) An <array type>.

            2) A <path-resolved user-defined type name> whose source type is an array type.

      o) A multiset-returning external function is an SQL-invoked function that is an external routine and that satisfies one of the following conditions:

         i) A <result cast from type> is specified that simply contains a <multiset type> or a <path-resolved user-defined type name> whose source type is a multiset type and does not contain a <locator indication>.

         ii) A <result cast from type> is not specified and <returns data type> simply contains a <multiset type> or a <path-resolved user-defined type name> whose source type is a multiset type and does not contain a <locator indication>.

2. *Rationale: Replace the undefined symbol T with the correct one, T_i.*

   Replace Syntax Rule 20) d) iv) 1) B) I) 2) a) with:

   20) ...

      d) ...

         iv) ...

            1) ...

               B) ...

                  I) ...

                     2) ...

                        a) The Syntax Rules of Subclause 9.23, "Determination of a to-sql function", in ISO/IEC 9075-2, are applied with the data type identified by T_i as *TYPE* and the

<group name> contained in the <group specification> that contains T_i as *GROUP*. There shall be an applicable to-sql function TSF_i. TSF_i is called the *to-sql function associated with i-th SQL parameter*.

11.61 <alter routine statement>

This Subclause is modified by Subclause 9.9, "<alter routine statement>", in ISO/IEC 9075-13.

1. *Rationale: The correct term is "SQL data type column".*

   Replace Syntax Rule 9) b) with:

   9) ...

      b) Depending on whether the <language clause> specifies ADA, C, COBOL, FORTRAN, M, PASCAL, or PLI, let the operative data type correspondences table be Table 16, "Data type correspondences for Ada", Table 17, "Data type correspondences for C", Table 18, "Data type correspondences for COBOL", Table 19, "Data type correspondences for Fortran", Table 20, "Data type correspondences for M", Table 21, "Data type correspondences for Pascal", or Table 22, "Data type correspondences for PL/I", respectively. Refer to the two columns of the operative data type correspondences table as the *SQL data type column* and the *host data type column*.

12 Access control

12.7 <revoke statement>

This Subclause is modified by Subclause 10.3, "<revoke statement>", in ISO/IEC 9075-4.
This Subclause is modified by Subclause 13.2, "<revoke statement>", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 10.3, "<revoke statement>", in ISO/IEC 9075-13.

1. *Rationale: Use the correct form of "SQL routine body" which is <SQL routine body>.*

   Replace General Rule 29) e) with:

   29) ...

      e) SELECT privilege on at least one column of each table identified by a <table reference> contained in a <value expression> simply contained in an <update source> or an <assigned row> contained in the <SQL routine body> of *RD*.

2. *Rationale: Use the correct form of "SQL routine body" which is <SQL routine body>.*

   Replace General Rule 29) g) with:

   29) ...

   g) SELECT privilege on at least one column identified by a <column reference> contained in a <value expression> simply contained in an <update source> or an <assigned row> contained in the <SQL routine body> of *RD*.

3. *Rationale: Use the correct form of "SQL routine body" which is <SQL routine body>.*

   Replace General Rules 29) k) and 29) l) with:

   29) ...

   k) DELETE privilege on the table identified by the <target table> contained in a <merge statement> that contains a <merge delete specification> and that is contained in the <SQL routine body> of *RD*.

   l) USAGE privilege on each domain, collation, character set, transliteration, and sequence generator whose name is contained in the <SQL routine body> of *RD*.

13 SQL-client modules

13.4 <SQL procedure statement>

This Subclause is modified by Subclause 11.2, "<SQL procedure statement>", in ISO/IEC 9075-4.
This Subclause is modified by Subclause 14.3, "<SQL procedure statement>", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 13.2, "<SQL procedure statement>", in ISO/IEC 9075-14.

Subclause Signature

```
“<SQL procedure statement>” (
  Parameter: “EXECUTING STATEMENT”
)
```

1. *Rationale: Cater for entry to the Subclause either by BNF or signature.*

   Replace General Rule 1) with:

   1) Case:

   a) If this Subclause has been invoked as a “subroutine Subclause” from another Subclause, then: Let *S* be the *EXECUTING STATEMENT* in an application of the Rules of this Subclause.

NOTE 454 — *S* is necessarily the innermost executing statement of the SQL-session as defined in Subclause 4.38, “SQL-sessions”.

NOTE 455 — *S* is not necessarily an <SQL procedure statement>.

b) Otherwise, let *S* be the <SQL procedure statement>.

2. *Rationale: Clarify the execution of an SQL-statement.*

Delete General Rule 4).

3. *Rationale: Clarify the execution of an SQL-statement.*

Replace General Rule 5) with:

5.1) If *S* is immediately contained in an <externally-invoked procedure> *EP*, then:

a) Let *LANG* be the caller language of *EP*; let *n* be the number of <host parameter declaration>s specified in *EP*; let PD_i, 1 (one) ≤ *i* ≤ *n*, be the *i*-th such <host parameter declaration>; let PN_i and DT_i be the <host parameter name> and <host parameter data type>, respectively, specified in PD_i; and let P_i be the host parameter corresponding to PD_i. When *EP* is called by an SQL-agent, let PI_i be the *i*-th argument in the procedure call.

b) The SQL-client module that contains *S* is associated with the SQL-agent.

5.2) 04 If *S* is not an <SQL diagnostics statement>, then the first diagnostics area is emptied.

5.3) If *S* is not an <SQL diagnostics statement> or an <SQL connection statement>, and no SQL-session is current for the SQL-agent, then

Case:

a) If the SQL-agent has not executed an <SQL connection statement> and there is no default SQL-session associated with the SQL-agent, then the following <connect statement> is effectively executed:

```
CONNECT TO DEFAULT
```

b) If the SQL-agent has not executed an <SQL connection statement> and there is a default SQL-session associated with the SQL-agent, then the following <set connection statement> is effectively executed:

```
SET CONNECTION DEFAULT
```

c) Otherwise, an exception condition is raised: *connection exception — connection does not exist.*

5.4) If no exception condition has been raised, then:

a) If *S* is immediately contained in an <externally-invoked procedure>, then for each *i*, 1 (one) ≤ *i* ≤ *n*, if P_i is an input host parameter or both an input host parameter and an output host parameter, then the General Rules of Subclause 9.3, “Passing a value from a host language to the SQL-server”, in ISO/IEC 9075-2, are applied with *LANG* as *LANGUAGE*, DT_i as *SQL TYPE*, and PI_i as *HOST VALUE*; let *SV* be the *SQL VALUE* returned from the application of those General Rules. The value of P_i is set to *SV*.

b) If *S* does not conform to the Syntax Rules and Access Rules of an <SQL procedure statement>, then an exception condition is raised: *syntax error or access rule violation*.

NOTE 456 — Subclause 6.3.3.2, "Terms denoting rule requirements", and Subclause 6.3.3.3, "Rule evaluation order", in [ISO9075-1], require that the Syntax Rules and Access Rules remain satisfied during General Rule evaluation; doing so might require initiation of a transaction earlier than the General Rules otherwise suggest.

5.5) If no exception has been raised and no SQL-transaction is active for the SQL-agent and *S* is an SQL-statement that implicitly initiates SQL-transactions, then an SQL-transaction is initiated. If *S* is immediately contained in an <externally-invoked procedure>, then the SQL-client module that contains *S* is associated with the SQL-transaction.

NOTE 457 — If *S* is a <prepare statement>, a transaction may or may not be initiated based on the contents of the <SQL statement variable>. See Subclause 4.34.4, "SQL-statements and transaction states".

5.6) If no exception has been raised and the non-dynamic or dynamic execution of an <SQL data statement>, <SQL dynamic data statement>, <dynamic select statement>, or <dynamic single row select statement> occurs within the same SQL-transaction as the non-dynamic or dynamic execution of an SQL-schema statement and this is not allowed by the SQL-implementation, then an exception condition is raised: *invalid transaction state — schema and data statement mixing not supported*.

5.7) If no exception has been raised and *S* is an <SQL schema statement> and the transaction access mode of the current SQL-transaction is read-only, then an exception condition is raised: *invalid transaction state*.

5.8) If no exception condition has been raised, then:

a) The General Rules of *S* are evaluated.

b) If *S* is immediately contained in an <externally-invoked procedure>, then for each *i*, 1 (one) $\leq i \leq n$, if P_i is an output host parameter that is not the status parameter or both an input host parameter and an output host parameter, then the General Rules of Subclause 9.4, "Passing a value from the SQL-server to a host language", in ISO/IEC 9075-2, are applied with *LANG* as *LANGUAGE*, DT_i as *SQL TYPE*, and the value of P_i as *SQL VALUE*; let *LV* be the *HOST VALUE* returned from the application of those General Rules. The value of PI_i is set to *LV*.

5.9) If prior General Rules successfully initiated or resumed an SQL-session, then subsequent calls to an <externally-invoked procedure> and subsequent invocations of <direct SQL statement>s by the SQL-agent are associated with that SQL-session until the SQL-agent terminates the SQL-session or makes it dormant.

4. *Rationale: Clarify the execution of an SQL-statement.*

Replace General Rule 6) with:

6) If no exception has been raised, then for every enforced referential constraint *RC*, the General Rules of Subclause 15.17, "Execution of referential actions", in ISO/IEC 9075-2, are applied with *RC* as *CONSTRAINT*.

5. *Rationale: Clarify the execution of an SQL-statement.*

Replace the lead text of General Rule 7) with:

7) If no exception has been raised, then for every state change *DSC* in *SSC* whose trigger event is DELETE, let *DSOT* be the set of transitions in *DSC* and let *DBT* be the subject table of *DSC*.

   Case:

6. *Rationale: Clarify the execution of an SQL-statement.*

   Replace the lead text of General Rule 8) with:

   8) If no exception has been raised, then

      Case:

7. *Rationale: Clarify the execution of an SQL-statement.*

   Insert a new General Rule.

   9.1) If *S* is a <select statement: single row> or a <fetch statement> and a completion condition is raised: *no data*, or an exception condition is raised, then the value of each result that is assigned to a <target specification> in *S* is implementation-dependent.

   NOTE 458.1 — If *S* is immediately contained in an <externally-invoked procedure>, then these results are the values PI_i of the output parameters PN_i. Using [ISO9075-4], the results might be the values of SQL variables. If *S* is contained in the body of an SQL routine, the results might be values of SQL parameters. Direct SQL does not support <select statement: single row> or <fetch statement>.

8. *Rationale: Clarify the execution of an SQL-statement.*

   Replace General Rule 10) with:

   10) Case:

      a) If *S* executed successfully, then either a completion condition is raised: *successful completion*, or a completion condition is raised: *warning*, or a completion condition is raised: *no data*, as determined by the General Rules in this and other Subclauses of ISO/IEC 9075.

      b) If *S* did not execute successfully, then:

         i) If *S* is an atomic SQL-statement, then all changes made to SQL-data or schemas by the execution of *S* are canceled.

         ii) The exception condition with which the execution of *S* completed is raised.

9. *Rationale: Clarify the execution of an SQL-statement.*

   Insert a new General Rule.

   10.1) If *S* is immediately contained in an <externally-invoked procedure>, then the status parameter is set to the value specified for the condition in Clause 24, “Status codes”.

13.5 Data type correspondences

This Subclause is modified by Subclause 14.4, "Data type correspondences", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 13.3, "Data type correspondences", in ISO/IEC 9075-14.

1. *Rationale: Correct the SQL data type syntax for CHARACTER VARYING and CHARACTER LARGE OBJECT in Table 16, "Data type correspondences for Ada".*

   In Table 16, "Data type correspondences for Ada" replace the rows for CHARACTER VARYING and CHARACTER LARGE OBJECT with:

09 14 **Table 16 — Data type correspondences for Ada**

SQL Data Type	**Ada Data Type**
CHARACTER VARYING (*L U*) CHARACTER SET *CS*	`Interfaces.SQL.VARYING.CHAR (1..`*L*`*`*k*`)` or `Interfaces.SQL.VARYING.NCHAR (1..`*L*`*`*k*`)` 1 3

CHARACTER LARGE OBJECT (*L U*) CHARACTER SET *CS*	`TYPE HVN IS`
	`  RECORD`
	`    HVN_RESERVED : Interfaces.SQL.INT;`
	`    HVN_LENGTH : Interfaces.SQL.INT;`
	`    HVN_DATA : Interfaces.SQL.CHAR(1..L*k)` 1 2 4`;`
	`  END RECORD;`

2. *Rationale: In the row for CHARACTER LARGE OBJECT remove the duplicate footnote in Table 17, "Data type correspondences for C".*

   In Table 17, "Data type correspondences for C" replace the row for CHARACTER LARGE OBJECT with:

09 14 **Table 17 — Data type correspondences for C**

SQL Data Type	**C Data Type**

CHARACTER LARGE OBJECT (*L U*) CHARACTER SET *CS*	`struct {`
	`    long hvn_reserved;`
	`    unsigned long hvn_length;`
	`    unit hvn_data[L * k];`
	`  } hvn`1 2 3`;`

3. *Rationale: In the 3rd footnote of Table 17, "Data type correspondences for C" replace the undefined symbol AV with the defined symbol CV.*

   In Table 17, "Data type correspondences for C" replace the 3rd footnote with:

Table 17 — Data type correspondences for C

SQL Data Type	C Data Type
[3] In a C value *CV* of this type, the *length portion* of *CV* is the field of *CV* called `hvn_length`, and the *data portion* of *CV* is the field of *CV* called `hvn_data`.	

4. *Rationale:* *In the 4th footnote of Table 18, "Data type correspondences for COBOL" replace the undefined symbol AV with the defined symbol CV.*

In Table 18, "Data type correspondences for COBOL" replace the 4th footnote with:

09 14 Table 18 — Data type correspondences for COBOL

SQL Data Type	COBOL Data Type
[4] In a COBOL value *CV* of this type, the *length portion* of *CV* is the field of *CV* called `hvn-LENGTH`, and the *data portion* of *CV* is the field of *CV* called `hvn-DATA`.	

5. *Rationale:* *In the row for BINARY VARYING in Table 19, "Data type correspondences for Fortran" replace "None" with a declaration as defined by the Syntax Rule of Subclause 21.6, "<embedded SQL Fortran program>" .*

In Table 19, "Data type correspondences for Fortran" replace the row for BINARY VARYING with:

09 14 Table 19 — Data type correspondences for Fortran

SQL Data Type	Fortran Data Type

BINARY VARYING (*L*)	`CHARACTER hvn(L+8)`
	`    INTEGER*4 hvn_RESERVED`
	`    INTEGER*4 hvn_LENGTH`
	`    CHARACTER hvn_DATA`
	`    EQUIVALENCE(hvn(1), hvn_RESERVED)`
	`    EQUIVALENCE(hvn(5), hvn_LENGTH)`
	`    EQUIVALENCE(hvn(9), hvn_DATA)`[1] [3]

6. *Rationale:* *In the 3rd footnote of Table 19, "Data type correspondences for Fortran" replace the undefined symbol AV with the defined symbol FV.*

In Table 19, "Data type correspondences for Fortran" replace the 3rd footnote with:

Table 19 — Data type correspondences for Fortran

SQL Data Type	Fortran Data Type
[3] In a Fortran value *FV* of this type, the *length portion* of *FV* is the field of *FV* called `hvn_LENGTH`, and the *data portion* of *FV* is the field of *FV* called `hvn_DATA`.	

14 Data manipulation

14.3 <cursor specification>

1. *Rationale: "one-to-one" is not always defined; use "target leaf underlying table" instead.*

   Replace Syntax Rule 5) with:

   5) If an <updatability clause> of FOR UPDATE with or without a <column name list> is specified, then *CP* shall not contain INSENSITIVE, *QE* shall be updatable, and *QE* shall have only one target leaf underlying table *LUT*.

14.5 <fetch statement>

This Subclause is modified by Subclause 12.3, "<fetch statement>", in ISO/IEC 9075-4.
This Subclause is modified by Subclause 11.15, "<fetch statement>", in ISO/IEC 9075-10.
This Subclause is modified by Subclause 14.1, "<fetch statement>", in ISO/IEC 9075-14.

1. *Rationale: Replace "arbitrary site" with "temporary site".*

   Replace Syntax Rule 9) b) iii) 1) with:

   9) ...

      b) ...

         iii) ...

              1) If $TS1_i$ contains a <simple value specification>, then the Syntax Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2, are applied with a temorary site whose declared type is the declared type of $TS1_i$ as *TARGET* and CS_i as *VALUE*.

2. *Rationale: Remove an intrusive comma.*

   Replace Syntax Rule 9) b) v) with:

   9) ...

b) ...

v) For each <target specification> $TS2_i$, 1 (one) ≤ i ≤ NTS, that is an <embedded variable specification>, the Syntax Rules of Subclause 9.1, “Retrieval assignment”, in ISO/IEC 9075-2, are applied with $TS2_i$ as *TARGET* and CS_i as *VALUE*.

14.8 <delete statement: positioned>

This Subclause is modified by Subclause 12.6, “<delete statement: positioned>”, in ISO/IEC 9075-4.
This Subclause is modified by Subclause 11.12, “<delete statement: positioned>”, in ISO/IEC 9075-10.

1. *Rationale: “one-to-one” is not always defined; use “target leaf underlying table” instead.*

   Replace Syntax Rule 5) with:

   5) Let *TU* be the simply underlying table of the cursor identified by *CN*. Let *LUT* be the target leaf underlying table of *TU*.

2. *Rationale: The target underlying table is a piece of syntax, not an actual table.*

   Replace Syntax Rule 6) with:

   6) Let *TT* be the <target table> and let *TN* be the <table name> contained in *TT*. *TN* shall be equivalent to *LUT*.

3. *Rationale: Use the more specific <schema name> instead of "qualifier".*

   Replace Syntax Rule 10) with:

   10) The schema identified by the explicit or implicit <schema name> of *TN* shall include the descriptor of *LUT*.

4. *Rationale: Use the correct symbol COR instead of CN.*

   Replace NOTE 466 with:

   NOTE 466 — *COR* has no scope.

14.11 <insert statement>

This Subclause is modified by Subclause 14.4, “<insert statement>”, in ISO/IEC 9075-14.

1. *Rationale: Add the missing verb in the sentence.*

   Replace Syntax Rule 6) with:

6) An <insert columns and source> that specifies DEFAULT VALUES is implicitly replaced by an <insert columns and source> that specifies a <contextually typed table value constructor> of the form

```
VALUES (DEFAULT, DEFAULT, ..., DEFAULT)
```

where the number of instances of "DEFAULT" is equal to the number of columns of *T*.

2. *Rationale: Editorial.*

Replace Syntax Rule 9) with:

9) If *T* is not trigger insertable-into, then *T* shall be an updatable table and each object column shall be an updatable column of *T*.

NOTE 469 — The notion of updatable columns of base tables is defined in Subclause 4.15, "Tables". The notion of updatable columns of viewed tables is defined in Subclause 11.32, "<view definition>".

3. *Rationale: Make the store assignment compatible.*

Replace Syntax Rule 14) with:

14) For each column of *T* identified by the <column name> in the <insert column list>, the Syntax Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2, are applied with the column as *TARGET* and the corresponding column of *QT* as *VALUE*.

4. *Rationale: To clarify that the attributes of the derived representation are a distinguished subset of the attributes of the structured type.*

Replace General Rule 7) d) ii) with:

7) ...

d) ...

ii) If *RC* is a derived self-referencing column, then the value of *RC* is effectively replaced by a value derived from the columns in the candidate row that correspond to the list of attributes of derivational the derived representation of the reference type of *RC* in an implementation-dependent manner.

5. *Rationale: If the target table is trigger insertable-into, then it does not need to be simply updatable.*

Replace Conformance Rule 5) with:

5) Without Feature T111, "Updatable joins, unions, and columns", conforming SQL language shall not contain an <insert statement> that contains an <insertion target> that identifies a table that is not simply updatable and not trigger insertable-into.

14.12 <merge statement>

This Subclause is modified by Subclause 14.5, "<merge statement>", in ISO/IEC 9075-14.

1. *Rationale: Clarify which object columns shall be updatable.*

   Replace Syntax Rule 16) with:

   16) If <merge when not matched clause> is specified and if *T* is not trigger insertable-into, then every insert object column shall identify an updatable column of *T*.

   16.1) If <merge when matched clause> is specified and if *T* is not trigger updatable, then every update object column shall identify an updatable column of *T*.

   NOTE 475 — The notion of updatable columns of base tables is defined in Subclause 4.15, "Tables". The notion of updatable columns of viewed tables is defined in Subclause 11.32, "<view definition>".

2. *Rationale: Correct typography.*

   Replace the lead text of Syntax Rule 19) with:

   19) Let *DSC* be the <search condition> immediately contained in <merge statement>.

3. *Rationale: Use the correct term "current privileges" in the Access Rules.*

   Replace Access Rule 1) b) ii) with:

   1) ...

      b) ...

         ii) If <merge delete specification> is specified, then the current privileges for *A* shall include DELETE for *T*.

4. *Rationale: Correctly define and use the terms "old delta table of merge operation" and "new delta table of merge operation" and correct an incorrect formulation of <correlation name>.*

   Replace General Rule 6) with:

   6) Let *NMWC* be the number of <merge when clause>s immediately contained in the <merge operation specification>.

      a) For each <merge when clause>, MWC_j, *j* varying from 1 (one) to *NMWC*, in the order specified in the <merge operation specification>,

         Case:

         i) If <merge when matched clause> $MWMC_j$ is specified, then:

            1) For each row *R1* of *T*:

A) *SC1* and the <search condition> SC_j immediately contained in $MWMC_j$, if any, are effectively evaluated for *R1* with *CN* bound to *R1* and to each row of *Q* with the exposed <correlation name>s or <table or query name>s of the <table reference> bound to that row. Both *SC1* and SC_j are effectively evaluated for *R1* before updating or deleting any row of *T* and prior to the invocation of any <triggered action> caused by the update or deletion of any row of *T* and before inserting any rows into *T* and prior to the invocation of any <triggered action> caused by the insert of any row of *T*.

Case:

I) If *TT* contains ONLY, then *R1* is a subject row if *R1* has no subrow in a proper subtable of *T* and the result of both *SC1* and SC_j are *True* for some row *R2* of *Q* and *R1* is not a subject row identified by any other <merge when matched clause> that precedes $MWMC_j$ in the <merge operation specification>. *R2* is the matching row.

II) Otherwise, *R1* is a subject row if the result of both *SC1* and SC_j are *True* for some row *R2* of *Q* and *R1* is not a subject row identified by any other <merge when matched clause> that precedes $MWMC_j$ in the <merge operation specification>. *R2* is the matching row.

B) If *R1* is a subject row, then:

I) Let *M* be the number of matching rows in *Q* for *R1*.

II) If *M* is greater than 1 (one), then an exception condition is raised: *cardinality violation*.

III) If <merge update specification> is specified, then:

1) The <update source> of each <set clause> is effectively evaluated for *R1* before any row of *T* is updated and prior to the invocation of any <triggered action> caused by the update of any row of *T*. The resulting value is the update value.

2) A candidate new row is constructed by copying the subject row and updating it as specified by each <set clause> by applying the General Rules of Subclause 14.15, "<set clause list>".

2) Let S_j be the set consisting of every subject row.

3) If *T* is a base table, then each subject row is also an object row; otherwise, an object row is any row of a leaf generally underlying table of *T* from which a subject row is derived.

NOTE 476 — The data values allowable in the object rows may be constrained by a WITH CHECK OPTION constraint. The effect of a WITH CHECK OPTION constraint is defined in the General Rules of Subclause 15.15, "Effect of replacing some rows in a viewed table".

4) If any row in the set of object rows has been marked for deletion by any <delete statement: positioned>, <dynamic delete statement: positioned>, or <preparable dynamic delete statement: positioned> that identifies some open cursor *CR* or updated by any <update statement: positioned>, <dynamic update statement: positioned>, or <preparable dynamic update statement: positioned> that identifies some open cursor, then a completion condition is raised: *warning — cursor operation conflict*.

5) If <merge update specification> is specified, then:

   A) Let *CL* be the columns of *T* identified by the <object column>s contained in the <set clause list>.

   B) Each subject row *SR* is identified for replacement, by its corresponding candidate new row *CNR*, in *T*. The set of (*SR*, *CNR*) pairs is the replacement set for *T*.

      NOTE 477 — Identifying a row for replacement, associating a replacement row with an identified row, and associating a replacement set with a table are implementation-dependent operations.

   C) Case:

      I) If *T* is a base table, then:

         1) Case:

            a) If *TT* specifies ONLY, then *T* is *identified for replacement processing without subtables* with respect to object columns *CL*.

            b) Otherwise, *T* is *identified for replacement processing with subtables* with respect to object columns *CL*.

               NOTE 478 — Identifying a base table for replacement processing, with or without subtables, is an implementation-dependent mechanism. In general, though not here, the list of object columns can be empty.

         2) The General Rules of Subclause 15.13, “Effect of replacing rows in base tables”, are applied.

      II) If *T* is a viewed table, then the General Rules of Subclause 15.15, “Effect of replacing some rows in a viewed table”, in ISO/IEC 9075-2, are applied with *TT* as *VIEW NAME* and the replacement set for *T* as *REPLACEMENT SET FOR VIEW NAME*.

   D) Let ND_j be the new delta table of update operation on *T*, if any.

      NOTE 479 — “new delta table of update operation” is defined in Subclause 15.13, “Effect of replacing rows in base tables”, and Subclause 15.15, “Effect of replacing some rows in a viewed table”.

6) If <merge delete specification> is specified, then:

   A) Each subject row is identified for deletion from *T*.

   B) Case:

I) If *T* is a base table, then:

1) Case:

a) If *TT* specifies ONLY, then *T* is *identified for deletion processing without subtables*.

b) Otherwise, *T* is *identified for deletion processing with subtables*.

NOTE 480 — Identifying a base table for deletion processing, with or without subtables, is an implementation-dependent mechanism.

2) The General Rules of Subclause 15.7, "Effect of deleting rows from base tables", are applied.

II) Otherwise, *T* is a viewed table and the General Rules of Subclause 15.9, "Effect of deleting some rows from a viewed table", in ISO/IEC 9075-2, are applied with *TT* as *VIEW NAME*.

ii) If <merge when not matched clause> $MWNMC_j$ is specified, then:

1) Let *TR1* be the <target table> immediately contained in <merge statement> and let *TR2* be the <table reference> immediately contained in <merge statement>. If <merge correlation name> is specified, then let *MCN* be "AS <merge correlation name>"; otherwise, let *MCN* be a zero-length string. If *MWNMCj* immediately contains a <search condition> SC_j, then let $ONSC_j$ be "`OR NOT` SC_j"; otherwise, let $ONSC_j$ be a zero-length string. Let *S1* be the result of

```
SELECT *
FROM TR1 MCN, TR2
WHERE SC1 ONSCj
```

2) Let *S2* be the collection of rows of *Q* for which there exists in *S1* some row that is the concatenation of some row *R1* of *T* and some row *R2* of *Q*.

3) Let *S3* be the collection of rows of *Q* that are not in *S2*. Let *SN3* be the effective distinct name for *S3*. Let *ER* be the exposed range variable of *TR2*. If *ER* is a <table name>, then let *EN* be the zero-length character string; otherwise, let *EN* be "AS *ER*".

4) Let *S4* be the result of:

```
SELECT EXP1, EXP2, ... , EXPNI
FROM SN3 EN
```

5) Let *S5* be the collection of rows of *S4* for which no candidate rows have been effectively created by any other <merge when not matched clause> that precedes $MWNMC_j$ in the <merge operation specification>.

6) *S5* is effectively evaluated before deletion of any rows from, insertion of any rows into, or update of any rows in *T*.

7) For each row *R* of *S5*:

A) A candidate row of *T* is effectively created in which the value of each column is its default value, as specified in the General Rules of Subclause 11.5, "<default clause>". The candidate row consists of every column of *T*.

B) If *T* has a column *RC* of which some underlying column is a self-referencing column, then

Case:

I) If *RC* is a system-generated self-referencing column, then the value of *RC* is effectively replaced by the REF value of the candidate row.

II) If *RC* is a derived self-referencing column, then the value of *RC* is effectively replaced by a value derived from the columns in the candidate row that correspond to the list of derivational attributes of the derived representation of the reference type of *RC* in an implementation-dependent manner.

C) For each object column in the candidate row, let C_i be the object column identified by the *i*-th <column name> in the <insert column list> and let SV_i be the *i*-th value of *R*.

D) For every C_i for which one of the following conditions is true:

I) C_i is not marked as unassigned and no underlying column of C_i is a self-referencing column.

II) Some underlying column of C_i is a user-generated self-referencing column.

III) Some underlying column of C_i is a self-referencing column and OVERRIDING SYSTEM VALUE is specified.

IV) Some underlying column of C_i is an identity column and the *i*-th column of *R* is not derived from <default specification> and OVERRIDING SYSTEM VALUE is specified.

V) Some underlying column of C_i is an identity column whose descriptor includes an indication that values are generated by default and neither OVERRIDING USER VALUE is specified nor is the *i*-th column derived from <default specification>.

the General Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2, are applied with C_i as *TARGET* and SV_i as *VALUE*. C_i is no longer marked as unassigned.

NOTE 481 — If OVERRIDING USER VALUE is specified, then some columns of the candidate row(s) may continue to be marked as unassigned as a result of the preceding rules. The value of such columns is ultimately determined by the General Rules of Subclause 15.10, "Effect of inserting tables into base tables", which has the effect of overriding user values specified in <insert columns and source>.

NOTE 482 — The data values allowable in the candidate row may be constrained by a WITH CHECK OPTION constraint. The effect of a WITH CHECK OPTION constraint is defined in the General Rules of Subclause 15.12, “Effect of inserting a table into a viewed table”.

8) Let S_j be the table consisting of the candidate rows.

Case:

A) If *T* is a base table, then:

I) *T* is *identified for insertion of source table* S_j.

NOTE 483 — Identifying a base table for insertion of a source table is an implementation-dependent operation.

II) The General Rules of Subclause 15.10, “Effect of inserting tables into base tables”, are applied.

B) If *T* is a viewed table, then the General Rules of Subclause 15.12, “Effect of inserting a table into a viewed table”, in ISO/IEC 9075-2, are applied with *S* as *SOURCE* and *T* as *TARGET*.

iii) Let ND_j be the new delta table of insert operation on *T*, if any.

NOTE 482.1 — “new delta table of insert operation” is defined in Subclause 15.10, “Effect of inserting tables into base tables”, and Subclause 15.12, “Effect of inserting a table into a viewed table”.

b) Let *ODT* be the union of all S_j, where for *j* varying from 1 (one) to *NMWC*, MWC_j immediately contains a <merge when matched clause>. *ODT* is the *old delta table of merge operation* on *T*.

c) Let *NDT* be the union of all ND_j, where for *j* varying from 1 (one) to *NMWC*, MWC_j immediately contains either a <merge when matched clause> that specifies UPDATE or a <merge when not matched clause>. *NDT* is the *new delta table of merge operation* on *T*.

5. *Rationale: The target table of a <merge statement> does not need to be simply updatable if there are INSTEAD OF triggers to perform the specified data changes.*

Replace Conformance Rule 5) with:

5) Without Feature T111, “Updatable joins, unions, and columns”, if <merge when not matched clause> is specified, then *T* shall be simply updatable or trigger insertable-into.

5.1) Without Feature T111, “Updatable joins, unions, and columns”, if <merge update specification> is specified, then *T* shall be simply updatable or trigger updatable.

5.2) Without Feature T111, “Updatable joins, unions, and columns”, if <merge delete specification> is specified, then *T* shall be simply updatable or trigger deletable.

14.13 <update statement: positioned>

This Subclause is modified by Subclause 12.7, “<update statement: positioned>”, in ISO/IEC 9075-4.
This Subclause is modified by Subclause 11.13, “<update statement: positioned>”, in ISO/IEC 9075-10.
This Subclause is modified by Subclause 14.6, “<update statement: positioned>”, in ISO/IEC 9075-14.

1. *Rationale: "one-to-one" is not always defined; use "target leaf underlying table" instead.*

   Replace Syntax Rule 5) with:

   5) Let *TU* be the simply underlying table of the cursor identified by *CN*. Let *LUT* be the target leaf underlying table of *TU*.

2. *Rationale: The target underlying table is a piece of syntax, not an actual table.*

   Replace Syntax Rule 6) with:

   6) Let *TT* be the <target table> and let *TN* be the <table name> contained in *TT*. *TN* shall be equivalent to *LUT*.

3. *Rationale: Use the more specific <schema name> instead of "qualifier".*

   Replace Syntax Rule 10) with:

   10) The schema identified by the explicit or implicit <schema name> of *TN* shall include the descriptor of *LUT*.

14.14 <update statement: searched>

This Subclause is modified by Subclause 14.7, "<update statement: searched>", in ISO/IEC 9075-14.

1. *Rationale: Replace undefined symbol with the correct symbol.*

   Replace Syntax Rule 8) with:

   8) If *USS* is contained in a <triggered SQL statement>, then *SC* shall not contain a <value specification> that specifies a parameter reference.

2. *Rationale: Clarify the table identified by <correlation name>s or <table or query name>.*

   Replace General Rule 5) a) with:

   5) ...

      a) If *TT* contains ONLY, then *SC* is effectively evaluated for each row of *T* with the exposed <correlation name>s or <table or query name>s of *TT* bound to that row, and the subject rows are those rows for which the result of *SC* is *True* and for which there is no subrow in a proper subtable of *T*. *SC* is effectively evaluated for each row of *T* before updating any row of *T*.

14.15 <set clause list>

1. *Rationale: Add a clarification.*

   Add a note after General Rule 1).

   NOTE 481.1 — If <set clause> specifies more than one object column, then the update value is a <contextually typed row value expression> that contains a number of fields equal to the number of object columns.

14.16 <temporary table declaration>

This Subclause is modified by Subclause 12.8, "<temporary table declaration>", in ISO/IEC 9075-4.

1. *Rationale: Direct SQL may access a created local temporary table.*

   Replace Syntax Rule 2) with:

   2) 04 *TTD* shall be contained in an <SQL-client module definition> or a <direct SQL data statement>.

2. *Rationale: Remove the confusion caused by treating the name of a schema as the schema itself.*

   Replace General Rule 1) with:

   1) Let *U* be the implementation-dependent <schema name> of the schema that contains the declared local temporary table such that the schema identified by *U* does not contain a table whose <table name> is equivalent to *TN*.

15 Additional data manipulation rules

15.1 Effect of opening a cursor

This Subclause is modified by Subclause 7.1, "Effect of opening a cursor", in ISO/IEC 9075-3.
This Subclause is modified by Subclause 13.1, "Effect of opening a cursor", in ISO/IEC 9075-4.

1. *Rationale: Editorial correction.*

   Replace General Rule 4) c) with:

   4) ...

      c) If the kind of cursor described by *CDD* is a extended dynamic cursor, then let *S* be the prepared statement indicated by the <statement name> that is the origin of *CDD*.

15.5 Effect of a positioned delete

1. *Rationale: An updatable cursor may have more than one leaf underlying table; use "target leaf underlying table" instead.*

   Replace Syntax Rule 8) with:

   8) Let *T* be the simply underlying table of *CR* and let *LUT* be the target leaf underlying table of *T*.

15.6 Effect of a positioned update

1. *Rationale: An updatable cursor may have more than one leaf underlying table; use "target leaf underlying table" instead.*

   Replace Syntax Rule 14) with:

   14) Let *T* be the simply underlying table of *CR* and let *LUT* be the target leaf underlying table of *T*.

15.7 Effect of deleting rows from base tables

This Subclause is modified by Subclause 15.1, "Effect of deleting rows from base tables", in ISO/IEC 9075-9.

1. *Rationale: Clarify which tables and rows are processed by this subclause.*

   Insert a new General Rule.

   1.1) Let *TT2* be the set consisting of the following tables:

   a) Every supertable of every table in *TT*.

   b) Every subtable of every table in *TT* that is identified for deletion processing with subtables.

2. *Rationale: Clarify which tables and rows are processed by this subclause.*

   Replace General Rule 2) with:

   2) For every row *R* in *S*:

   a) Every superrow *SR* of *R* is identified for deletion from the base table *BT* containing *SR*.

   b) If the table containing *R* is identified for deletion processing with subtables, then every subrow *SR* of *R* is identified for deletion from the base table *BT* containing *SR*.

3. *Rationale: Clarify which tables and rows are processed by this subclause.*

   Replace the lead text of General Rule 4) with:

4) For every table *ST* in *TT2*,

Case:

4. *Rationale: Clarify which tables and rows are processed by this subclause.*

Replace General Rule 6) with:

6) 09 Every row that is identified for deletion in some table in *TT2* is marked for deletion. These rows are no longer identified for deletion, nor are their containing tables identified for deletion processing (with or without subtables).

NOTE 500 — "Marking for deletion" is an implementation-dependent mechanism.

15.10 Effect of inserting tables into base tables

This Subclause is modified by Subclause 15.2, "Effect of inserting tables into base tables", in ISO/IEC 9075-9.

1. *Rationale: Editorial.*

Replace General Rule 2) a) iii) with:

2) ...

a) ...

iii) The General Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2, are applied with *ICS* as *TARGET* and *ICNV* as *VALUE*.

2. *Rationale: Use declared type instead of data type.*

Replace General Rule 2) b) ii) with:

2) ...

b) ...

ii) Let *NV* be the transaction timestamp of the current SQL-transaction. Let *DT* be the declared type of the system-time period start column of *T*.

3. *Rationale: Editorial.*

Replace General Rule 2) b) iii) with:

2) ...

b) ...

iii) The General Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2, are applied with *ICS* as *TARGET* and *NVV* as *VALUE*.

4. *Rationale: Editorial.*

   Replace General Rule 2) c) iii) with:

   2) ...

      c) ...

         iii) The General Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2, are applied with *ICS* as *TARGET* and *NV* as *VALUE*.

15.12 Effect of inserting a table into a viewed table

1. *Rationale: Only target leaf generally underlying tables have new delta tables.*

   Replace General Rule 4) with:

   4) Let *n* be the number of target leaf generally underlying tables of *QE*. Let T_i, 1 (one) ≤ *i* ≤ *n*, be the target leaf generally underlying tables of *QE*. Let NT_i, 1 (one) ≤ *i* ≤ *n*, be the new delta table of insert operation on T_i. Let *S* be the result of evaluating *QE* with every reference to T_i, 1 (one) ≤ *i* ≤ *n*, being replaced with a reference to NT_i. *S* is the *new delta table of insert operation* on *T*.

15.13 Effect of replacing rows in base tables

This Subclause is modified by Subclause 15.3, "Effect of replacing rows in base tables", in ISO/IEC 9075-9.

1. *Rationale: Clarify which tables, rows and object columns are processed by this subclause.*

   Insert a new General Rule.

   1.1) Let *TT2* be the set comprising the following tables:

      a) Every supertable of every table in *TT*.

      b) Every subtable of every table in *TT* that is identified for replacement processing with subtables.

2. *Rationale: Clarify which tables, rows and object columns are processed by this subclause.*

   Replace General Rule 2) with:

   2) For every base table *T* in *TT*:

      a) Let *OC*(*T*) be the set consisting of the name of every object column with respect to which *T* is identified for replacement processing and the name of every generated column of *T* that depends on at least one of these object columns.

b) For every table *ST* in *TT2* that is a subtable or supertable of *T*, let *OC*(*ST*) be the intersection (possibly empty) of *OC*(*T*) and the set of names of the columns of *ST*.

3. *Rationale: Clarify which tables, rows and object columns are processed by this subclause.*

Replace the lead text of General Rule 5) with:

5) For every table *ST* in *TT2*:

4. *Rationale: Clarify which tables, rows and object columns are processed by this subclause.*

Replace the lead text of General Rule 5) b) with:

5) ...

b) Let *TL* be the set consisting of the names of the columns of *ST*. For every subset *STL* of *TL* such that either *STL* is empty or the intersection of *STL* and *OC*(*ST*) is not empty,

Case:

5. *Rationale: Clarify which tables, rows and object columns are processed by this subclause.*

Replace the lead text of General Rule 8) with:

8) 09 For every table *T* in *TT2*,

Case:

6. *Rationale: Editorial.*

Replace General Rule 8) a) i) with:

8) ...

a) ...

i) Let *START* be the system-time period start column of *T* and let *END* be the system-time period end column of *T*. Let *DT* be the declared type of *START*.

7. *Rationale: Clarify which tables, rows and object columns are processed by this subclause.*

Replace General Rule 8) b) with:

8) ...

b) Otherwise, for every row *R* that is identified for replacement in *T*, *R* is replaced by its corresponding replacement row. *R* is no longer identified for replacement. *T* is no longer identified for replacement processing, with or without subtables. Let *SUP* be the set consisting of every replacement row of every *R*. *SUP* is the *new delta table of update operation* on *T*.