INTERNATIONAL STANDARD ISO/IEC 9075-2:2008
TECHNICAL CORRIGENDUM 1

Published 2010-06-01

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

Information technology — Database languages — SQL —

Part 2:
Foundation (SQL/Foundation)

TECHNICAL CORRIGENDUM 1

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

Partie 2: Fondations (SQL/Foundation)

RECTIFICATIF TECHNIQUE 1

Technical Corrigendum 1 to ISO/IEC 9075-2:2008 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange*.

Statement of purpose for rationale

A statement indicating the rationale for each change to ISO/IEC 9075-2:2008 is included. This is to inform the users of ISO/IEC 9075-2:2008 why it was judged necessary to change the original wording. In many cases, the reason is editorial or to clarify the wording; in some cases, it is to correct an error or an omission in the original wording.

Notes on numbering

Where this Technical Corrigendum introduces new Syntax, Access, General, and Conformance Rules, the new rules have been numbered as follows:

> Rules inserted between, for example, Rules 7) and 8) are numbered 7.1), 7.2), etc. [or 7)a.1), 7)a.2), etc.]. Those inserted before Rule 1) are numbered 0.1), 0.2), etc.

Where this Technical Corrigendum introduces new subclauses, the new subclauses have been numbered as follows:

> Subclauses inserted between, for example, 4.3.2 and 4.3.3 are numbered 4.3.2a, 4.3.2b, etc. Those inserted before, for example, 4.3.1 are numbered 4.3.0, 4.3.0a, etc.

ICS 35.060 **Ref. No. ISO/IEC 9075-2:2008/Cor.1:2010(E)**

Published in Switzerland

Contents

Tables

Table

Page

Information technology — Database languages — SQL —

Part 2:
Foundation (SQL/Foundation)

TECHNICAL CORRIGENDUM 1

3 Definitions, notations, and conventions

3.1 Definitions

3.1.2 Definitions taken from [ISO14651]

1. *Rationale: Editorial correction.*

   Replace the lead text with:

   For the purposes of this document, the definitions of the following terms given in [ISO14651] apply:

3.1.6 Definitions provided in Part 2

1. *Rationale: Define an undefined symbol.*

   Replace the definition 3.1.6.12 with:

   3.1.6.12 element type (of a collection type)
   The declared type *DT* specified in the definition of a collection type *CT*.

   NOTE 4 — The declared type of every element of every value of type *CT* is *DT*.

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: Row types are not totally ordered.*

   Insert the following bullet after the 4th bullet of the 11th paragraph:

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

4.2 Character strings

4.2.4 Character repertoires

1. *Rationale: Clarify that character repertoire SQL_IDENTIFIER is implementation-defined.*

   In the 3rd paragraph, replace the last bullet in dashed list with:

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

4.4 Numbers

4.4.1 Introduction to numbers

1. *Rationale: Editorial correction.*

   Replace the 5th paragraph with:

   Similarly, an SQL-implementation is permitted to regard certain <approximate numeric type>s as equivalent, as permitted by the Syntax Rules of Subclause 6.1, "<data type>", in which case the SQL-implementation chooses a *normal form* to represent each equivalence class of <approximate numeric type> and the normal form determines the name of the approximate numeric type.

4.4.2 Characteristics of numbers

1. *Rationale: Editorial correction.*

   Replace the 6th paragraph with:

   An approximation obtained by truncation of a numeric value *N* for an <exact numeric type> *T* is a value *V* in *T* such that *N* is not closer to zero than *V* and there is no value in *T* between *V* and *N*.

4.7 User-defined types

4.7.2 User-defined type descriptor

This Subclause is modified by Subclause 4.8.2, "User-defined type descriptor", in ISO/IEC 9075-13.

1. *Rationale: Align the descriptor of a method specification and its initializing rules.*

   Replace the 12th bullet of the 1st paragraph with:

   — If the <method specification list> is specified, then for each <method specification> contained in <method specification list>, a *method specification descriptor* that includes:

   - The <method name>.
   - The <specific method name>.
   - An indication as to whether the <method specification> is an <original method specification> or an <overriding method specification>.
   - If the <method specification> is an <original method specification>, then an indication of whether STATIC or CONSTRUCTOR is specified.
   - The <SQL parameter declaration list> augmented to include the implicit first parameter with parameter name SELF.
   - For every <SQL parameter declaration> in the <SQL parameter declaration list>, a <locator indication>, if any.
   - The <returns data type>.
   - The <result cast from type>, if any.
   - The <locator indication> contained in the <returns clause>, if any.
   - The <language name>.
   - If the <language name> is not SQL, then the <parameter style>.
   - An indication whether the method is deterministic.
   - An indication whether the method possibly modifies SQL-data, possibly reads SQL-data, possibly contains SQL, or does not possibly contain SQL.
   - An indication whether the method should not be invoked if any argument is the null value.
   - The creation timestamp.
   - The last-altered timestamp.

4.14 Tables

4.14.6 Operations involving tables

1. *Rationale: Editorial correction.*

   Replace the lead text of the 11th paragraph with:

   Each of the table updating operations, when applied to a base table *T* or a derived table *T*, can have various secondary effects. Such secondary effects include alteration or reversal of the primary effect. Secondary effects might arise from the existence of:

4.17 Integrity constraints

4.17.2 Checking of constraints

1. *Rationale: Supply a missing right parenthesis.*

   Replace the 2nd paragraph with:

   The checking of a constraint depends on its constraint mode within the current SQL-transaction. Whenever an SQL-statement is executed, every constraint whose mode is immediate is checked, at a certain point after any changes to SQL-data and schemas resulting from that execution have been effected, to see if it is satisfied. A constraint is *satisfied* if and only if the applicable <search condition> included in its descriptor evaluates to *True* or *Unknown*. If any constraint is not satisfied, then any changes to SQL-data or schemas resulting from executing that statement are canceled. (See the General Rules of Subclause 13.5, "<SQL procedure statement>").

   NOTE 36 — This includes SQL-statements that are executed as a direct result or an indirect result of executing a different SQL-statement. It also includes statements whose effects explicitly or implicitly include setting the constraint mode to immediate.

4.18 Functional dependencies

4.18.7 Known functional dependencies in a <table primary>

1. *Rationale: Handle all kinds of <table primary> in the definition of known functional dependencies.*

   Replace this Subclause with:

Let *R* be the result of some <table primary> *TP*. The BPK-sets, BUC-sets, and known functional dependencies of *R* are determined as follows.

Case:

— 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 $A \mapsto B$ is a known functional dependency in the result of *TQN*, and *AC* and *BC* are the counterparts of *A* and *B*, respectively, then $AC \mapsto BC$ is a *known functional dependency* in *R*.

— If *TP* immediately contains a <derived table>or <lateral derived table> *DT*, then the counterparts of the BPK-sets and BUC-sets of *DT* are the BPK-sets and BUC-sets, respectively, of *R*. If $A \mapsto B$ is a known functional dependency in the result of *DT*, and *AC* and *BC* are the counterparts of *A* and *B*, respectively, then $AC \mapsto BC$ is a *known functional dependency* in *R*.

— If *TP* immediately contains a <collection derived table> *CDT*, and WITH ORDINALITY is specified, then let *OC* be the column name of the ordinality column and let *CT* be the set comprising all of the column names of the columns of *CDT*. {*OC*} is a BPK-set and a BUC-set, and $\{OC\} \mapsto CT$ is a *known functional dependency*. If WITH ORDINALITY is not specified, then these rules do not identify any BPK-set, BUC-set, or non-axiomatic known functional dependency.

— If *TP* immediately contains a <table function derived table>, then these rules do not identify any BPK-set, BUC-set, or non-axiomatic known functional dependency.

— If *TP* immediately contains a <parenthesized joined table>, then let *JT* be the <joined table> simply contained in *TP*. The counterparts of the BPK-sets and BUC-sets of *JT* are the BPK-sets and BUC-sets, respectively, of *R*. If $A \mapsto B$ is a known functional dependency in the result of *JT*, and *AC* and *BC* are the counterparts of *A* and *B*, respectively, then $AC \mapsto BC$ is a known functional dependency in *R*.

4.18.10 Known functional dependencies in the result of a <from clause>

1. *Rationale: Editorial correction.*

   Replace the 2nd paragraph with:

   If there is only one <table reference> *TR* in *FC*, then the counterparts of the BPK-sets of *TR* and the counterparts of the BUC-sets of *TR* are the BPK-sets and BUC-sets of *FC*, respectively. Otherwise, these rules do not identify any BPK-sets or BUC-sets in the result of *FC*.

4.22 SQL-client modules

1. *Rationale: Editorial correction.*

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

   — The name, if any, of the SQL-client module.

4.24 Dynamic SQL concepts

4.24.2 Dynamic SQL statements and descriptor areas

1. *Rationale: Editorial correction.*

   Replace the 9th paragraph with:

   The scope of a local extended name *LEN* is the SQL-client module *M* containing the externally-invoked procedure that is being executed when the object *O* identified by *LEN* is brought into existence. This means that, during the existence of *O*, *LEN* can be used to reference *O* by any SQL-statement executed in the same SQL-session by an externally-invoked procedure in *M*.

4.27 SQL-invoked routines

4.27.3 Execution of SQL-invoked routines

This Subclause is modified by Subclause 4.4.2, "Execution of SQL-invoked routines", in ISO/IEC 9075-4.

1. *Rationale: Editorial correction.*

   Replace the 1st paragraph with:

   When an SQL-invoked routine is invoked, a copy of the current SQL-session context is pushed onto the stack and some values are modified (see the General Rules of Subclause 10.4, "<routine invocation>"), before the <routine body> is executed. The treatment of the authorization stack is described in Subclause 4.34.1.1, "SQL-session authorization identifiers".

4.29 Host parameters

4.29.5 Locators

1. *Rationale: Use the correct term.*

   Replace the 1st bullet in the 2nd paragraph with:

   — Binary large object locator, a value of which identifies a binary large object string.

4.32 Cursors

4.32.2 Operations on and using cursors

1. *Rationale: Remove a duplicated definition.*

   Replace the 13th paragraph with:

   If *CR* is a with-return cursor, then the <cursor specification> included in the result set descriptor of *CR* defines a returned result set. A with-return cursor, if declared in an SQL-invoked procedure and in the open state when the procedure returns to its invoker, yields a returned result set that can be accessed by the invoker of the procedure that generates it.

   NOTE 66 — Definitions of "returned result set" and "with-return cursor" are given in Subclause 3.1.6, "Definitions provided in Part 2".

5 Lexical elements

5.3 <literal>

1. *Rationale: Use the correct syntax format.*

   Replace Conformance Rule 6) with:

   6) Without Feature F271, "Compound character literals", in conforming SQL language, a <character string literal> shall contain exactly one repetition of <character representation> (that is, it shall contain exactly one sequence of "<quote> [<character representation>...] <quote>").

5.4 Names and identifiers

This Subclause is modified by Subclause 5.2, "Names and identifiers", in ISO/IEC 9075-4.
This Subclause is modified by Subclause 5.2, "Names and identifiers", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 5.2, "Names and identifiers", in ISO/IEC 9075-13.
This Subclause is modified by Subclause 5.2, "Names and identifiers", in ISO/IEC 9075-14.

1. *Rationale: Editorial correction.*

   Replace Syntax Rule12) b) iii) with:

   12) ...

   b) ...

iii) Otherwise, the <schema name> that is specified or implicit for the <SQL-client module definition> is implicit.

2. *Rationale: Editorial correction.*

Replace Syntax Rule 13) c) with:

13) ...

c) If *SQN* is immediately contained in a <constraint name> that is contained in a <domain definition> or an <alter domain statement>, then the explicit or implicit <schema name> of the <domain name> of the domain identified by the <domain definition> or an <alter domain statement> is implicit.

3. *Rationale: Editorial correction.*

Replace Syntax Rule 23) with:

23) The declared type of <SQL-server name>, <connection name>, and <connection user name> shall be character string with an implementation-defined character set and shall have an octet length of 128 octets or less.

4. *Rationale: Editorial correction.*

Replace Conformance Rule 15) with:

15) Without Feature T601, "Local cursor references", in conforming SQL language, a <cursor name> shall not contain a <local qualifier>.

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: Use the correct term.*

Replace General Rule 1) with:

1) If the result of any specification or operation would be a character string value one of whose characters is not in the character set of its declared type, then an exception condition is raised: *data exception — character not in repertoire*.

2. *Rationale: Clarify the meaning of the notation 9(N).*

Replace Table 10 with:

Table 10 — Valid absolute values for interval fields

Keyword	Valid values of INTERVAL fields
MONTH	0 to 11
HOUR	0 to 23
MINUTE	0 to 59
SECOND	0 to 59.9(*N*) where "9(*N*)" indicates a sequence of *N* instances of the digit "9" and "*N*" indicates the number of digits specified by <interval fractional seconds precision> in the <interval qualifier>.

6.4 <value specification> and <target specification>

This Subclause is modified by Subclause 6.1, "<value specification> and <target specification>", in ISO/IEC 9075-4.
This Subclause is modified by Subclause 6.1, "<value specification> and <target specification>", in ISO/IEC 9075-10.

1. *Rationale: Use the correct syntax element.*

Replace the definition of <simple target specification> in Format with:

```
<simple target specification> ::=
    <host parameter name>
  | <SQL parameter reference>
  | <column reference>
  | <embedded variable name>
```

2. *Rationale: Use the correct syntax element.*

Replace General Rule 15) with:

15) 04 A <simple target specification> specifies a target that is a host parameter, an output SQL parameter, a column of a new transition variable, or a host variable, according to whether the <simple target specification> is a <host parameter name>, an <SQL parameter reference>, a <column reference>, or an <embedded variable name>, respectively.

NOTE 115 — A <simple target specification> can never be assigned the null value.

6.7 <column reference>

1. *Rationale: Add missing Conformance Rule.*

   Insert the following Conformance Rule:

   2) Without Feature T301, "Functional dependencies", in conforming SQL language, if *QCR* is a group-invariant column reference, then *QCR* shall be a reference to a grouping column of the qualifying query of *QCR*.

6.9 <set function specification>

1. *Rationale: Clarify that every <set function specification> shall have an aggregation query.*

   Replace Syntax Rule 7) with:

   7) *SFS* shall have an aggregation query, and shall be contained in the <having clause>, <window clause>, or <select list> of its aggregation query.

6.10 <window function>

1. *Rationale: Use the correct term.*

   Replace Syntax Rule 6) b) with:

   6) ...

      b) The window framing clause of *WDX* shall not be present.

6.11 <case expression>

This Subclause is modified by Subclause 6.4, "<case expression>", in ISO/IEC 9075-14.

1. *Rationale: Conformance Rules should be applied to the syntactic transformation.*

   Replace Syntax Rule 1) b) with:

   1) ...

      b) `NULLIF (`V_1`, `V_2`)` is equivalent to the following <case specification>:

```
CASE WHEN
V1=V2 THEN
```



```
NULL ELSE V1
END
```

The Conformance Rules of Subclause 8.2, "<comparison predicate>" are applied to the result of this syntactic transformation.

2. *Rationale: Conformance Rules should be applied to the syntactic transformation.*

Insert the following Syntax Rule:

2) ...

i.1) The Conformance Rules of Clause 8, "Predicates" are applied to the result of this syntactic transformation.

NOTE 125.1 — The specific Subclauses of Clause 8, "Predicates" are determined by the predicates that are created as a result of the syntactic transformation.

6.12 <cast specification>

This Subclause is modified by Subclause 6.2, "<cast specification>", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 6.5, "<cast specification>", in ISO/IEC 9075-14.

1. *Rationale: Use the correct term.*

Replace General Rule 18) d) with:

18) ...

d) If *SD* is TIME WITH TIME ZONE, then the <primary datetime field>s year, month, and day of *TV* are set to their respective values in an execution of CURRENT_DATE and the <primary datetime field>s hour, minute, and second of *TV* are set to their respective values in *SV*, with implementation-defined rounding or truncation if necessary. The time zone component of *TV* is set to the time zone displacement of *SV*.

2. *Rationale: Use the correct symbol.*

Replace General Rule 19) a) ii) with:

19) ...

a) ...

ii) Let *NDSEN* be the scale of the <exact numeric literal> formed by the <cast specification>

```
CAST (SV AS CHARACTER VARYING(max))
```

where *max* is the implementation-defined maximum precision of the CHARACTER VARYING type. If the number of digits of fractional seconds precision *NDFSP* of *TD* is less than *NDSEN*, then it is implementation-defined whether *TV* is determined

by rounding *SV* to *NDFSP* digits of precision or by truncating *SV* to *NDFSP* digits of precision, as specified in Subclause 4.4.2, "Characteristics of numbers".

3. *Rationale: Supply missing data types.*

   Replace Conformance Rule 2) with:

   2) Without Feature F421, "National character", conforming SQL language shall not contain a <cast operand> whose declared type is NATIONAL CHARACTER, NATIONAL CHARACTER VARYING, or NATIONAL CHARACTER LARGE OBJECT.

6.18 <new specification>

This Subclause is modified by Subclause 6.2, "<new specification>", in ISO/IEC 9075-13.

1. *Rationale: Use of the correct term.*

   Replace Syntax Rule 2) with:

   2) Let *UDT* be the user-defined type identified by *UDTN*. *UDT* shall be instantiable. Let *SN* be the implicit or explicit <schema name> of *UDTN*. Let *S* be the schema identified by *SN*. Let *RN* be *SN.MN*.

6.23 <array element reference>

1. *Rationale: Normalize the description of the definition of a symbol.*

   Replace the lead text of General Rule 2) with:

   2) Let *i* be the value of <numeric value expression>.

      Case:

6.27 <numeric value function>

1. *Rationale: Prevent subsequent General Rules from overwriting the result that is established in General Rule 1).*

   Replace General Rule 1) with:

   1) If the value of one or more <string value expression>s, <datetime value expression>s, <interval value expression>s, and <collection value expression>s that are simply contained in a <numeric value function> is the null value, then the result of the <numeric value function> is the null value and no further General Rules of this Subclause are applied.

2. *Rationale: Editorial correction.*

   Replace General Rule 11) b) with:

   11) ...

      b) Otherwise, the result is *e* (the base of natural logarithms) raised to the power *V*. If the result is not representable in the declared type of the result, then an exception condition is raised: *data exception — numeric value out of range*.

3. *Rationale: Editorial correction.*

   Replace Conformance Rule 5) and 6) with:

   5) Without Feature T441, "ABS and MOD functions", conforming SQL language shall not contain an <absolute value expression>.

   6) Without Feature T441, "ABS and MOD functions", conforming SQL language shall not contain a <modulus expression>.

4. *Rationale: Construct conformance rule for T041 and T021.*

   Insert the following Conformance Rule:

   19) Without Feature T041, "Basic LOB data type support", or Feature T021, "BINARY and VARBINARY data types", conforming SQL language shall not contain a <binary position expression>.

6.28 <string value expression>

1. *Rationale: Construct conformance rules for F421, T041 and T021.*

   Replace the Conformance Rules with:

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

   2) Without Feature T041, "Basic LOB data type support", or Feature T021, "BINARY and VARBINARY data types", conforming SQL language shall not contain a <binary value expression>.

6.29 <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: Editorial correction.*

   Replace the definition of <regex transliteration> in Format with:

```
<regex transliteration> ::=
  TRANSLATE_REGEX <left paren>
     <XQuery pattern> [ FLAG <XQuery option flag> ]
     IN <regex subject string>
      [ WITH <XQuery replacement string> ]
      [ FROM <start position> ]
      [ USING <char length units> ]
      [ OCCURRENCE <regex transliteration occurrence> ]
     <right paren>
```

2. *Rationale: Remove redundant descriptions of a declared type and supply missing descriptions of a character set and collation.*

   Replace of Syntax Rule 7) c) with:

   7) ...

      c) The character set and collation of the <regular expression substring function> are those of the first <character value expression>.

3. *Rationale: Remove redundant descriptions of a declared type and supply missing descriptions of a character set and collation.*

   Replace of Syntax Rule 13) d) with:

   13) ...

      d) The character set and collation of the <trim function> are those of the <trim source>.

4. *Rationale: Editorial correction.*

   Replace General Rule 12) b) i) with:

   12) ...

      b) ...

         i) If *V* is the null value, then let *RV* is the null value.

5. *Rationale: Construct conformance rule for F421.*

   Insert the following Conformance Rule:

   15) 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.

6.31 <datetime value function>

1. *Rationale: Construct conformance rule for F555.*

   Replace Conformance Rules 1) and 2) with:

   1) Without Feature F555, "Enhanced seconds precision", conforming SQL language shall not contain a <current local time value function> that contains a <time precision> that is not 0 (zero) and shall not contain a <current time value function> that contains a <time precision> that is not 0 (zero).

   2) Without Feature F555, "Enhanced seconds precision", conforming SQL language shall not contain a <current local timestamp value function> that contains a <timestamp precision> that is neither 0 (zero) nor 6 and shall not contain a <current timestamp value function> that contains a <timestamp precision> that is neither 0 (zero) nor 6.

6.34 <boolean value expression>

1. *Rationale: Editorial correction.*

   Replace Syntax Rule 5) e) ii) 1) with:

   5) ...

      e) ...

         ii) ...

            1) *BF* is negative and *BF* does not generally contain a possibly non deterministic <value expression>.

6.37 <multiset value expression>

1. *Rationale: Construct conformance rule for S271.*

   Insert the following Conformance Rule:

   0.1) Without Feature S271, "Basic multiset support", conforming SQL language shall not contain a <multiset value expression>.

6.39 <multiset value constructor>

1. *Rationale: Editorial correction.*

   Replace Conformance Rule 2) with:

2) Without Feature T326, “Table functions”, in conforming SQL language, a <multiset value constructor> shall not contain a <table value constructor by query>.

7 Query expressions

7.1 <row value constructor>

1. *Rationale: Editorial correction.*

   Replace Syntax Rule 3) b) with:

   3) ...

      b) Otherwise, the declared type of *ERVC* is a row type described by a sequence of (<field name>, <data type>) pairs, corresponding in order to each <row value constructor element> *X* simply contained in *ERVC*. The <data type> is the declared type of *X* and the <field name> is implementation-dependent.

7.6 <table reference>

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: Supply missing definition of ordinality column.*

   Replace Syntax Rule 3) n) i) with:

   3) ...

      n) ...

         i) If *CDT* specifies WITH ORDINALITY, then let *ELDT* be:

```
LATERAL ( RECQP SELECT SLE1, ..., SLENCV, NORD
                FROM TEMP ) AS CN PDCLP
```

         The column named by *NORD* is called the *ordinality column* of *CDT*.

2. *Rationale: Correct the definition of exposed range variable.*

   Replace Syntax Rules 4) and 5) with:

   4) Let *TF* be a <table factor> and let *TP* be the <table primary> that is immediately contained in *TF*.

5) If *TP* is not a <parenthesized joined table>, and *TP* simply contains a <correlation name>, then let *RV* be that <correlation name>; otherwise, let *RV* be the <table or query name> simply contained in *TF*, if any. *RV* is a range variable. *RV* is exposed by *TF* and by *TR*.

3. *Rationale: Editorial correction.*

Replace Syntax Rule 16) with:

16) A <derived table> or <lateral derived table> is a *simply updatable derived table* if and only if the <query expression> simply contained in the <derived table> or <lateral derived table> is simply updatable.

7.7 <joined table>

1. *Rationale: Not every <table reference> or <table factor> has a range variable.*

Replace Syntax Rule 1) with:

1) Let *TRA* be the <table reference> or <table factor> that is the first operand of the <joined table>, and let *TRB* be the <table reference> or <table factor> that is the second operand of the <joined table>. Let *RTA* and *RTB* be the row types of *TRA* and *TRB*, respectively. Let *CP* be:

```
SELECT *
FROM TRA, TRB
```

2. *Rationale: Not every <table reference> or <table factor> has a range variable.*

Replace Syntax Rules 8) e), 8) f) and 8) g) with:

8) e) If there is at least one corresponding join column, then:

i) Let *N* be the number of corresponding join columns.

ii) For each *i*, 1 (one) ≤ *i* ≤ *N*:

1) Let $CJCN_i$ be the *i*-th common column name, taken in order of their ordinal positions in *RTA*.

2) Case:

A) If the field of *RTA* whose name is $CJCN_i$ is associated with a range variable, let RVA_i be that range variable.

B) Otherwise, let RVA_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. RVA_i is effectively associated with the field of RTA whose name is $CJCN_i$.

NOTE 156.1 — This "effective range variable" is a device to overcome the fact that a common column in the result of a natural join or named column join has no range

variable associated with it. This case is only necessary when natural joins or named column joins are nested, for example (A NATURAL JOIN B) NATURAL JOIN C. There is no syntax to actually associate a range variable to a common column in the result of (A NATURAL JOIN B) in this example.

3) Case:

A) If the field of *RTB* whose name is $CJCN_i$ is associated with a range variable, let RVB_i be that range variable.

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 *RTA* whose name is $CJCN_i$.

NOTE 156.2 — This case is only necessary when natural joins or named column joins are nested, for example A NATURAL JOIN (B NATURAL JOIN C). There is no syntax to actually associate a range variable to a common column in the result of (B NATURAL JOIN C) in the preceding example.

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.

f) If *RTA* contains at least one field that is not a corresponding join column, then:

i) Let *NCA* be the number of fields of *RTA* that are not corresponding join columns taken in order of their ordinal positions in *RTA*.

ii) For each j, 1 (one) ≤ j ≤ *NCA*:

1) Let CA_j be the name of the j-th field that is not a corresponding join column, taken in order of their ordinal positions in *RTA*.

2) Case:

A) If the field of *RTA* whose name is CA_j is associated with a range variable, let $RVCA_j$ be that range variable.

B) Otherwise, let $RVCA_j$ 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. $RVCA_j$ is effectively associated with the field of *RTA* whose name is CA_j.

iii) Let *SLTA* be a <select list> of <derived column>s of the form $RVCA_j.CA_j$ for every j, 1 (one) ≤ j ≤ *NCA*, in ascending order.

g) If *RTB* contains at least one field that is not a corresponding join column, then:

i) Let *NCB* be the number of fields of *RTB* that are not corresponding join columns, taken in order of their ordinal positions in *RTB*.

ii) For each k, 1 (one) ≤ k ≤ *NCB*:

1) Let CB_k be the name of the j-th field that is not a corresponding join column, taken in order of their ordinal positions in *RTA*.

2) Case:

A) If the field of *RTB* whose name is CB_k is associated with a range variable, let $RVCB_k$ be that range variable.

B) Otherwise, let $RVCB_k$ 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. $RVCB_k$ is effectively associated with the field of *RTB* whose name is CB_k.

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

$RVCB_k$. CB_k

for every k, 1 (one) ≤ k ≤ *NCB*, in ascending order.

3. *Rationale: Remove redundant suppositions.*

Replace Syntax Rule 12) with:

12) A column *CR* of the result of the <joined table> is *known not null* if any of the following conditions is true:

a) *CR* is readily known not null.

b) If the SQL-implementation supports Feature T101, "Enhanced nullability determination", then *CR* is not possibly nullable according to the following conditions:

i) For every column *CR* of the result of the <joined table> that corresponds to a field *CA* of *RTA* that is not a corresponding join column or a join partitioning column, *CR* is *possibly nullable* if any of the following conditions are true:

1) RIGHT or FULL is specified.

2) INNER, LEFT, or CROSS JOIN is specified or implicit and *CA* is possibly nullable.

ii) For every column *CR* of the result of the <joined table> that corresponds to a field *CB* of *RTB* that is not a corresponding join column or a join partitioning column, *CR* is *possibly nullable* if any of the following conditions are true:

1) LEFT or FULL is specified.

2) INNER, RIGHT, or CROSS JOIN is specified or implicit and *CB* is possibly nullable.

iii) For every column *CR* of the result of the <joined table> that corresponds to a corresponding join column *CA* of *RTA* and a corresponding join column *CB* of *RTB*, *CR* is *possibly nullable* if any of the following conditions are true:

1) LEFT or FULL is specified and *CA* is possibly nullable, or

2) RIGHT or FULL is specified and *CB* is possibly nullable.

iv) A column *CR* of the result of the <joined table> that corresponds to a join partitioning column *JPC* is *possibly nullable* if *JPC* is possibly nullable.

c) *CR* conforms to an implementation-defined rule that correctly infers that the value of *CR* cannot be null.

4. *Rationale: Not every <table reference> or <table factor> has a range variable.*

Replace General Rule 1) c) i) with:

1) ...

c) ...

i) If there are corresponding join columns, then let *T* be

```
CP
WHERE RVA1.CJCN1 = RVB1.CJCN1
AND ...
AND RVAN.CJCNN = RVBN.CJCNN
```

NOTE 156.3 — Some or all of the RVA_i and RVB_i may be effective range variables that were created in the Syntax Rules. Although these effective range variables are not defined in the <from clause> of *CP*, nevertheless they must be correctly associated with the appropriate fields of *RTA* and/or *RTB*.

5. *Rationale: Use more specific description.*

Replace General Rule 4) c) i) 2) with

4) ...

c) ...

i) ...

2) Otherwise, *TVB* is partitioned into the minimum numbers of partitions such that for each join partitioning column *JPC* of each partition, no two values of *JPC* are distinct. If the declared type of a join partitioning column is a user-defined type and the comparison of that column results in *Unknown* for two rows of *TVB*, then the assignment of those rows to partitions is implementation-dependent. Let *NB* be the number of partitions. Let GB_1, ... , GB_{NB} be an enumeration of the partitions.

7.11 <window clause>

1. *Rationale: Use of the correct term*

Replace General Rules 1) b) i) 4), 1) b) i) 5) and 1) b) i) 6) with:

1) ...

b) ...

i) ...

4) Case:

A) If *WDEF* simply contains <window partition clause> *WDEFWPC*, then *WDESC*'s window partitioning clause is *WDEFWPC*.

B) If <existing window name> is specified, then *WDESC*'s window partitioning clause is the window partitioning clause of *WDX*.

C) Otherwise, *WDESC* has no window partitioning clause.

5) Case:

A) If *WDEF* simply contains <window order clause> *WDEFWOC*, then *WDESC*'s window ordering clause is *WDEFWOC*.

B) If <existing window name> is specified, then *WDESC*'s window ordering clause is the window ordering clause of *WDX*.

C) Otherwise, *WDESC* has no window ordering clause.

6) If *WDEF* simply contains <window frame clause> *WDEFWFC*, then *WDESC*'s window framing clause is *WDEFWFC*; otherwise, *WDESC* has no windows framing clause.

7.12 <query specification>

This Subclause is modified by Subclause 7.1, "<query specification>", in ISO/IEC 9075-4.

1. *Rationale: Editorial correction.*

Replace Syntax Rule 7) g) i) with:

7) ...

g) ...

i) If the basis is a <table or query name> or <correlation name>, then let *TQ* be the table associated with the basis. The <select sublist> is equivalent to a <value expression> sequence in which each <value expression> is a column reference *CR* that references a column of *TQ* that is not a common column of a <joined table>. Each column of *TQ* that is not a common column shall be referenced exactly once. The columns shall be referenced in the ascending sequence of their ordinal positions within *TQ*.

2. *Rationale: Use already defined term.*

Replace Syntax Rule 25) c) with:

25) ...

c) Every result column of *QS* is potentially updatable.

3. *Rationale: Editorial correction.*

Replace Conformance Rule 6) with:

6) Without Feature T285, "Enhanced derived column names", in conforming SQL language, if any <derived column> in a <select list> does not specify an <as clause> and the <value expression> of that <derived column> is not a single column reference, then the <column name> of that column is implementation-dependent.

7.13 <query expression>

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

1. *Rationale: Window functions, like aggregates, must be prohibited in the recursive portion of recursive queries.*

Insert the following Syntax Rule:

2) ...

g) ...

iii) ...

4.1) WQE_i shall not contain a <query specification> *QS* in which:

A) *QS* immediately contains a <table expression> that contains a <query name> referencing WQN_j and

B) *QS* immediately contains a <select list> that contains a <window function>.

2. *Rationale: Correct the rules for impermissible outer joins in recursive queries.*

Replace Syntax Rules 2) g) iii) 6) and 2) g) iii) 7) with:

2) ...

g) ...

iii) ...

6) WQE_i shall not contain a <qualified join> *QJ* in which:

A) *QJ* immediately contains a <join type> that specifies FULL and a <table reference> or <partitioned join table> that contains a <query name> referencing WQN_j.

B) *QJ* immediately contains a <join type> that specifies LEFT and a <table reference> or <partitioned join table> following the <join type> that contains a <query name> referencing WQN_j.

C) *QJ* immediately contains a <join type> that specifies RIGHT and a <table reference> or <partitioned join table> preceding the <join type> that contains a <query name> referencing WQN_j.

7) WQE_i shall not contain a <natural join> *QJ* in which:

A) *QJ* immediately contains a <join type> that specifies FULL and a <table reference>, <table factor> or <partitioned join table> that contains a <query name> referencing WQN_j.

B) *QJ* immediately contains a <join type> that specifies LEFT and a <table factor> or <partitioned join table> following the <join type> that contains a <query name> referencing WQN_j.

C) *QJ* immediately contains a <join type> that specifies RIGHT and a <table reference> or <partitioned join table> preceding the <join type> that contains a <query name> referencing WQN_j.

8 Predicates

8.3 <between predicate>

1. *Rationale: Conformance Rules should be applied to the syntactic transformation.*

   Insert the following Syntax Rule:

   6.1) The Conformance Rules of Subclause 8.2, "<comparison predicate>" are applied to the result of all syntactic transformations of this Subclause.

8.5 <like predicate>

1. *Rationale: Use the correct syntax element.*

   Replace Syntax Rule 2) with:

   2) The <row value predicand> immediately contained in <octet like predicate> shall be a <row value constructor predicand> that is a <common value expression> *OVE*. The declared types of *OVE*, <octet pattern>, and <escape octet> shall be binary string.

2. *Rationale: Supply missing data types.*

   Insert the following Conformance Rule:

   6.1) Without Feature F421, "National character", in conforming SQL language, a <character value expression> simply contained in a <like predicate> shall not be of declared type NATIONAL CHARACTER, NATIONAL CHARACTER VARYING.

9 Additional common rules

9.1 Retrieval assignment

This Subclause is modified by Subclause 9.1, "Retrieval assignment", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 10.1, "Retrieval assignment", in ISO/IEC 9075-14.

Subclause Signature

```
"Retrieval assignment" [Syntax Rules] (
Parameter: "TARGET",
Parameter: "VALUE"
)
```

1. *Rationale: Normalize the description of the definition of a symbol.*

   Replace Syntax Rule 1) with:

Syntax Rules

1) Let *T* be the *TARGET* and let *V* be the *VALUE* in an application of the Syntax Rules of this Subclause.

1.1) Let *TD* and *SD* be the declared types of *T* and *V*, respectively.

9.2 Store assignment

This Subclause is modified by Subclause 9.2, "Store assignment", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 10.2, "Store assignment", in ISO/IEC 9075-14.

Subclause Signature

```
"Store assignment" [Syntax Rules] (
Parameter: "TARGET",
```



```
Parameter: "VALUE"
)
```

1. *Rationale: Normalize the description of the definition of a symbol.*

   Replace Syntax Rule 1) with:

   ### Syntax Rules

   1) Let *T* be the *TARGET* and let *V* be the *VALUE* in an application of the Syntax Rules of this Subclause.

   1.1) Let *TD* and *SD* be the declared types of *T* and *V*, respectively.

2. *Rationale: Editorial correction.*

   Replace Syntax Rule 4) c) with:

   4) ...

      c) Let TT_i, 1 (one) ≤ i ≤ n, be the declared type of the i-th field of *T*, let VT_i be the declared type of the i-th field of *V*, let $T1_i$ be an arbitrary target whose declared type is TT_i, and let $V1_i$ be an arbitrary expression whose declared type is VT_i. For each i, 1 (one) ≤ i ≤ n, the Syntax Rules of this Subclause apply to $T1_i$ and $V1_i$ as *TARGET* and *VALUE*, respectively.

9.9 Equality operations

This Subclause is modified by Subclause 9.6, "Equality operations", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 10.8, "Equality operations", in ISO/IEC 9075-14.

1. *Rationale: Supply the argument in an application of Subclause 9.13, "Collation determination".*

   Replace Syntax Rule 4) with:

   4) Let *VS* be the set of declared types of the operands of an equality operation. If *VS* comprises character string types, then the Syntax Rules of Subclause 9.13, "Collation determination", are applied with *VS* as *VALSET*; let the collation to be used in the equality operation be the *COLL* returned from the application of those Syntax Rules.

9.10 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: Supply the argument in an application of Subclause 9.13, "Collation determination".*

   Replace Syntax Rule 4) with:

   4) Let *VS* be the set of declared types of the operands of a grouping operation. If *VS* comprises character string types, then the Syntax Rules of Subclause 9.13, "Collation determination", are applied with *VS* as *VALSET*; let the collation to be used in the grouping operation be the *COLL* returned from the application of those Syntax Rules.

9.11 Multiset element grouping operations

This Subclause is modified by Subclause 9.8, "Multiset element grouping operations", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 10.10, "Multiset element grouping operations", in ISO/IEC 9075-14.

1. *Rationale: All multiset operands of equality operations are operands of a multiset grouping operation.*

   Replace Syntax Rule 2) e) with:

   2) ...

      e) An operand of an equality operation such that the declared type of the operand is a multiset type.

2. *Rationale: <member predicate> is not a multiset grouping operation.*

   Delete Syntax Rule 2) f).

3. *Rationale: Use correct non terminal.*

   Replace Syntax Rule 2) g) with:

   2) ...

      g) A field of the <row value predicand> simply contained in a <submultiset predicate>.

4. *Rationale: Use correct non terminal.*

   Replace Syntax Rule 2) i) with:

   2) ...

      i) A field of the <row value predicand> simply contained in a <set predicate>.

5. *Rationale: Editorial correction.*

   Replace Syntax Rule 2) j) with:

   2) ...

j) A <value expression> simply contained in a <general set function> that specifies INTERSECTION.

6. *Rationale: Supply the argument in an application of Subclause 9.13, "Collation determination".*

Replace Syntax Rule 4) with:

4) Let *VS* be the set of declared types of the operands of the multiset operands of a multiset element grouping operation. If *VS* comprises character string types, then the Syntax Rules of Subclause 9.13, "Collation determination", are applied with *VS* as *VALSET*; let the collation to be used in the multiset element grouping operation be the *COLL* returned from the application of those Syntax Rules.

9.12 Ordering operations

This Subclause is modified by Subclause 9.9, "Ordering operations", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 10.11, "Ordering operations", in ISO/IEC 9075-14.

1. *Rationale: Supply the argument in an application of Subclause 9.13, "Collation determination".*

Replace Syntax Rule 4) with:

4) Let *VS* be the set of declared types of the operands of an ordering operation. If *VS* comprises character string types, then the Syntax Rules of Subclause 9.13, "Collation determination", are applied with *VS* as *VALSET*; let the collation to be used in the ordering operation be the *COLL* returned from the application of those Syntax Rules. If the declared type of an operand *VS* of an ordering operation is a character string type, then the Syntax Rules of Subclause 9.13, "Collation determination", apply with *VS* as *VALSET*.

9.22 Determination of a to-sql function for an overriding method

1. *Rationale: Use the correct term.*

Replace Syntax Rule 3) with:

3) The *applicable to-sql function* is the to-sql function associated with the result of *OM*, if any.

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.

1. *Rationale: Use of a correct term.*

   Replace Syntax Rule 8) a) with:

   8) ...

      a) The declared type of each <value expression> immediately contained in a <generalized expression> shall be a subtype of the structured type identified by the <user-defined type name> simply contained in the <path-resolved user-defined type name> that is immediately contained in <generalized expression>.

2. *Rationale: Correct the argument and the result in application of Subclause 9.2, "Store assignment".*

   Replace General Rule 3) a) with:

   3) ...

      a) [14] If P_i is an input SQL parameter or both an input SQL parameter and an output SQL parameter, then the General Rules of Subclause 9.2, "Store assignment", are applied with V_i as *VALUE* and a temporary site *ST* whose declared type is T_i as *TARGET*. Let CPV_i be the value of *ST*.

3. *Rationale: Clarify the condition that a null-call function returns a null value.*

   Replace General Rule 8) c) i) with:

   8) ...

      c) ...

         i) If *R* is not a null-call function and, for *i* ranging from 1 (one) to *PN*, any of CPV_i is the null value, then an exception condition is raised: *external routine invocation exception — null value not allowed*.

4. *Rationale: Use of a correct term.*

   Replace General Rule 8) i) iii) 1) with:

   8) ...

      i) ...

         iii) ...

            1) Let EV_i be the *i*-th entry in *ESPL*. Let T_i be the declared type of P_i.

5. *Rationale: Correct the argument and the result in application of Subclause 9.2, "Store assignment".*

   Replace General Rule 9) a) iii) with:

   9) ...

      a) ...

         iii) [14] The General Rules of Subclause 9.2, "Store assignment", are applied with RV_i as *VALUE* and a temporary site *ST* whose declared type is *ERDT* as *TARGET*. Let the result of the <routine invocation> be the value of *ST*.

6. *Rationale: Correct the position of a with-return cursor when it is returned.*

   Replace General Rule 10) e) ii) with:

   10) ...

      e) ...

         ii) Otherwise,

         Case:

         1) FRC_i is positioned before some row in RS_i, then let *RN* be the number of rows ordered before that row in RS_i.

         2) If FRC_i is positioned on some row in RS_i, then let *RN* be the ordinal position of that row in RS_i.

         3) Otherwise, let *RN* be the number of rows in RS_i.

         The first *RN* rows are deleted from RS_i and the initial cursor position of RS_i is before the first row of the rows that remain.

10.9 <aggregate function>

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

1. *Rationale: Row types are not totally ordered.*

   Insert the following Syntax Rule:

   7) ...

      d.1) If *AF* specifies a <set function type> that is MAX or MIN, then *DT* shall not be row-ordered.

2. *Rationale: Row types are not totally ordered.*

   Delete Conformance Rule 15) and the note that follows it.

3. *Rationale: Correct conformance rules for F442.*

   Replace Conformance Rules 16) and 17) with:

   16) Without Feature F442, "Mixed column references in set functions", conforming SQL language shall not contain a <hypothetical set function value expression list> that simply contains a <value expression> that contains more than one column reference, one of which is an outer reference.

   17) Without Feature F442, "Mixed column references in set functions", conforming SQL language shall not contain an <inverse distribution function> that contains an <inverse distribution function argument> that simply contains a <value expression> that contains more than one column reference, one of which is an outer reference.

4. *Rationale: Correct conformance rules for F442.*

   Insert the following Conformance Rule:

   19.1) Without Feature F442, "Mixed column references in set functions", conforming SQL language shall not contain a <within group specification> that simply contains a <value expression> that contains more than one column reference, one of which is an outer reference.

10.10 <sort specification list>

1. *Rationale: Row types are not totally ordered.*

   Replace General Rule 1) e) with:

   1) ...

      e) Let PV_i and QV_i be the values of K_i in P and Q, respectively. The relative position of rows P and Q in the result is determined by comparing PV_i and QV_i as follows:

         i) The comparison is performed according to the General Rules of

            Case:

            1) If the declared type of K_i is a row type, then this Subclause, applied recursively;

            2) Otherwise, Subclause 8.2, "<comparison predicate>" where the <comp op> is the applicable <comp op> for K_i,

         ii) The comparison is performed with the following special treatment of null values.

            Case:

            1) If PV_i and QV_i are both the null value, then they are considered equal to each other.

            2) If PV_i is the null value and QV_i is not the null value, then

               Case:

A) If NULLS FIRST is specified or implied, then PV_i <comp op> QV_i is considered to be *True*.

B) If NULLS LAST is specified or implied, then PV_i <comp op> QV_i is considered to be *False*.

3) If PV_i is not the null value and QV_i is the null value, then

Case:

A) If NULLS FIRST is specified or implied, then PV_i <comp op> QV_i is considered to be *False*.

B) If NULLS LAST is specified or implied, then PV_i <comp op> QV_i is considered to be *True*.

11 Schema definition and manipulation

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: A domain constraint should not tie a persistent base table and a temporary table.*

   Add the following Syntax Rule:

   6.1) If the descriptor of *D* includes any domain constraint descriptors, then *T* shall be a persistent base table.

11.18 <alter identity column specification>

1. *Rationale: Editorial correction.*

   Replace Note 295 in Syntax Rule 2) with:

   NOTE 295 — *OPT* is formulated by removing all instances of the keyword SET from the string corresponding to *AICO*.

11.21 <drop table constraint definition>

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

1. *Rationale: Fix problems with contradictions between some symbol definitions and their usage and the use undefined symbols.*

   Replace Syntax Rules 3), 4), 5), 6), 7), and 8) with:

   3) If *TC* is a unique constraint, *T* is the referenced table of a referential constraint *RC*, and the referenced columns of *RC* are the unique columns of *TC*, then *RC* is said to be *dependent on TC*.

   4) Let *QS* be a <query specification> that contains an implicit or explicit <group by clause> and that contains a column reference to a column *C* in its <select list> that is not contained in an aggregated argument of a <set function specification>. Let *G* be the set of grouping columns of *QS*. If *TC* is needed to conclude that $G \mapsto C$ is a known functional dependency in *QS*, then *QS* is said to be *dependent on TC*.

   5) Let *V1* be a view that contains a <query specification> that is dependent on *TC*. *V1* is said to be *dependent on TC*.

   6) Let *R1* be an SQL routine whose <SQL routine body> contains a <query specification> that is dependent on *TC*. *R1* is said to be *dependent on TC*.

   7) Let *C1* be a constraint or assertion whose <search condition> contains a <query specification> that is dependent on *TC*. *C1* is said to be *dependent on TC*.

   8) Let *TR1* be a trigger whose triggered action contains a <query specification> that is dependent on *TC*. *TR1* is said to be *dependent on TC*.

2. *Rationale: Fix problems with contradictions between some symbol definitions and their usage and the use undefined symbols.*

   Replace General Rules 1), 2), and 3) with:

   1) Let *TCN2* be the <constraint name> of any table constraint that is dependent on *TC* and let *TN2* be the <table name> of the table descriptor that includes *TCN2*. The following <alter table statement> is effectively executed without further Access Rule checking for every *TCN2* and *TN2*:

      ```
      ALTER TABLE TN2 DROP
      CONSTRAINT TCN2 CASCADE
      ```

   2) 04 Let *R* be any SQL-invoked routine whose routine descriptor contains the <constraint name> of *TC* in the SQL routine body or any SQL-invoked routine that is dependent on *TC*. Let *SN* be the specific name of *R*. The following <drop routine statement> is effectively executed without further Access Rule checking for every *R*:

      ```
      DROP SPECIFIC ROUTINE SN CASCADE
      ```

   3) Let *VN* be the table name of any view *V* that is dependent on *TC*. The following <drop view statement> is effectively executed without further Access Rule checkingfor every *V*:

      ```
      DROP VIEW VN CASCADE
      ```


3. *Rationale: Fix problems with contradictions between some symbol definitions and their usage and the use undefined symbols.*

   Delete General Rules 4) and 5).

4. *Rationale: Fix problems with contradictions between some symbol definitions and their usage and the use undefined symbols.*

   Replace General Rules 6), and 7) with:

   6) Let *AN* be the constraint name of any assertion *A* that is dependent on *TC*. The following <drop assertion statement> is effectively executed without further Access Rule checking for every *A*:

      ```
      DROP ASSERTION AN CASCADE
      ```

   7) Let *TRN* be the trigger name of any trigger *TR* that is dependent on *TC*. The following <drop trigger statement> is effectively executed without further Access Rule checking for every *TR*:

      ```
      DROP TRIGGER TRN CASCADE
      ```

5. *Rationale: Fix problems with contradictions between some symbol definitions and their usage and the use undefined symbols.*

   Replace General Rule 9) with:

   9) If *TC* causes some column *COL* to be known not nullable and no other constraint causes *COL* to be known not nullable, then the nullability characteristic of the column descriptor of *COL* is changed to possibly nullable.

      NOTE 10 — The nullability characteristic of a column is defined in Subclause 4.13, "Columns, fields, and attributes".

11.22 <drop table statement>

This Subclause is modified by Subclause 9.10, "<drop table statement>", in ISO/IEC 9075-4.

1. *Rationale: Use the correct term.*

   Replace General Rule 4) b) ii) with:

   4) ...

      b) ...

         ii) If *SOD* is an assertion descriptor, then let *SON* be the name of the constraint included in *SOD*. The following <drop assertion statement> is effectively executed without further Access Rule checking:

         ```
         DROP ASSERTION SON CASCADE
         ```

11.23 <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 12.3, "<view definition>", in ISO/IEC 9075-14.

1. *Rationale: Prohibit circular view definitions.*

   Insert the following Syntax Rule:

   3.1) Let *TN* be the <table name> and let *Q* be the <query expression>. There shall not exist a viewed table *V2* identified by <table name> *TN2* and defined by <query expression> *Q2* such that *TN* is generally contained in *Q2* and *TN2* is generally contained in *Q*.

2. *Rationale: Use of the correct term.*

   Replace Syntax Rule 20) j) iv) with:

   20) ...

   j) ...

   iv) Let *MSV* be the maximal superview of the subtable family of *V*. Let *RMSV* be the reference type REF(*MSV*).

   Case:

   1) If *RMSV* has a user-defined representation, then let *m* be 1 (one).

   2) Otherwise, *RMSV* has a derived representation. Let *m* be 0 (zero).

3. *Rationale: Correct reference to referenceable tables.*

   Replace Syntax Rule 20) q) with:

   20) ...

   q) If *TQN* is a referenceable table , then *TR* shall simply contain ONLY.

11.24 <drop view statement>

This Subclause is modified by Subclause 9.12, "<drop view statement>", in ISO/IEC 9075-4.

1. *Rationale: Use the correct term.*

   Replace General Rule 3) b) ii) with:

   3) ...

b) ...

ii) If *SOD* is an assertion descriptor, then let *SON* be the name of the contraint included in *SOD*. The following <drop assertion statement> is effectively executed without further Access Rule checking:

```
DROP ASSERTION SON CASCADE
```

11.25 <domain definition>

This Subclause is modified by Subclause 11.7, "<domain definition>", in ISO/IEC 9075-9.

1. *Rationale: Clarify the description of the object of a privilege.*

   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 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".

11.29 <add domain constraint definition>

1. *Rationale: A domain constraint should not tie a persistent base table and a temporary table.*

   Add the following Syntax Rule:

   3) *D* shall not be included in the column descriptor of any column of any global temporary table, created local temporary table or declared local temporary table.

11.40 <trigger definition>

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

1. *Rationale: A <triggered when clause> should not possibly modify SQL-data.*

   Add the following Syntax Rule:

   10.1) The <triggered when clause> shall not contain a <routine invocation> whose subject routine is an SQL-invoked routine that possibly modifies SQL-data.

11.41 <drop trigger statement>

1. *Rationale: Add Syntax Rules to require existence of trigger.*

   Insert the following Syntax Rules:

   1) Let *TR* be the trigger identified by the <trigger name> and let *TRN* be the name of *TR*.

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

2. *Rationale: Add Syntax Rules to require existence of trigger.*

   Replace Access Rule 1) with:

   1) Let *A* be the <authorization identifier> that owns the schema identified by the <schema name> of *TR*. The enabled authorization identifiers shall include *A*.

3. *Rationale: Remove remaining view component privilege descriptors.*

   Insert the following General Rule:

   1.1) All view component privilege descriptors are destroyed, if any.

11.42 <user-defined type definition>

This Subclause is modified by Subclause 11.9, "<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 12.5, "<user-defined type definition>", in ISO/IEC 9075-14.

1. *Rationale: Align the descriptor of a method specification and its initializing rules.*

   Replace General Rules 1) g) and 1) h) with:

   1) ...

      g) If <method specification list> is specified, then for every <original method specification> *ORMS* contained in <method specification list>, a method specification descriptor that includes:

         i) The <method name> of *ORMS*.

         ii) The <specific method name> of *ORMS*.

         iii) An indication that the method specification is original.

         iv) An indication of whether STATIC or CONSTRUCTOR is specified.

         v) The <SQL parameter declaration list> contained in *ORMS* (augmented, if STATIC is not specified in *ORMS*, to include the implicit first parameter with parameter name SELF).

vi) For every <SQL parameter declaration> in the <SQL parameter declaration list>, a <locator indication>, if any.

vii) The <returns data type>.

viii) The <result cast from type>, if any.

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

x) The <language name> contained in the explicit or implicit <language clause>.

xi) 13 The explicit or implicit <parameter style> if the <language name> is SQL.

xii) An indication of whether the method is deterministic.

xiii) An indication of whether the method possibly modifies SQL-data, possibly reads SQL-data, possibly contains SQL, or does not possibly contain SQL.

xiv) An indication of whether the method should not be invoked if any argument is the null value.

xv) The CURRENT_TIMESTAMP as the value of the creation timestamp.

xvi) The CURRENT_TIMESTAMP as the value of the last_altered timestamp.

h) If <method specification list> is specified, then for every <overriding method specification> *OVMS* contained in <method specification list> let *DCMS* be the descriptor of the corresponding original method specification. The method specification descriptor of *OVMS* includes:

i) The <method name> of *OVMS*.

ii) The <specific method name> of *OVMS*.

iii) An indication that the method specification is overriding.

iv) The <SQL parameter declaration list> contained in *OVMS* (augmented to include the implicit first parameter with parameter name SELF).

v) For every <SQL parameter declaration> in the <SQL parameter declaration list>, a <locator indication>, if any.

vi) The <returns data type> of *OVMS*.

vii) The <result cast from type> included in *DCMS* (if any).

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

ix) The <language name> included in *DCMS*.

x) If the <lamguage name> is not SQL, the <parameter style> included in *DCMS*.

xi) The determinism indication included in *DCMS*.

xii) The SQL-data access indication included in *DCMS*.

xiii) The indication included in *DCMS*, whether the method should not be invoked if any argument is the null value.

xiv) The CURRENT_TIMESTAMP as the value of the creation timestamp.

xv) The CURRENT_TIMESTAMP as the value of the last_altered timestamp.

11.47 <add original method specification>

1. *Rationale: Align the descriptor of a method specification and its initializing rules.*

   Replace General Rule 1) with:

   1) Let *STDS* be the descriptor of *D*. A method specification descriptor *DOMS* is created for *ORMS*. *DOMS* includes:

      a) The <method name> *MN*.

      b) The <specific method name> contained in *PORMS*.

      c) An indication that the method specification is original.

      d) An indication of whether STATIC or CONSTRUCTOR is specified.

      e) The augmented SQL parameter declaration list *NPL*.

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

      g) The <returns data type> contained in *PORMS*.

      h) The <result cast from type> contained in *PORMS* (if any).

      i) The locator indication, if a <locator indication> is contained in the <returns clause> of *PORMS* (if any).

      j) The <language name> explicitly or implicitly contained in *MCH*.

      k) If the <langauge name> is not SQL, then the explicit or implicit <parameter style> contained in *MCH*.

      l) The determinism indication contained in *MCH*.

      m) An indication of whether the method possibly modifies SQL-data, possibly reads SQL-data, possibly contains SQL, or does not possibly contain SQL.

      n) An indication of whether the method should not be invoked if any argument is the null value.

      o) The CURRENT_TIMESTAMP as the value of the creation timestamp.

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

11.48 <add overriding method specification>

1. *Rationale: Replace references to parameter descriptors with references to method specification descriptors.*

   Replace Syntax Rule 12) with:

12) For *i* varying from 2 to *N*:

a) If $POVMS_{i-1}$ contains an <SQL parameter name> *PNM1*, then the *i*-th element of the augmented SQL parameter declaration list included in the descriptor of *COMS* shall have an SQL parameter name that is equivalent to *PNM1*.

b) If the *i*-th element of the augmented SQL parameter declaration list included in the descriptor of *COMS* has an SQL parameter name *PNM2*, then $POVMS_{i-1}$ shall contain an <SQL parameter name> that is equivalent to *PNM2*.

c) $POVMS_{i-1}$ shall not contain <parameter mode>. A <parameter mode> IN is implicit.

d) $POVMS_{i-1}$ shall not specify RESULT.

e) If the <parameter type> PT_{i-1} immediately contained in $POVMS_{i-1}$ contains a <locator indication>, then the *i*-th element of the augmented SQL parameter declaration list included in the descriptor of *COMS* shall include a <locator indication>.

f) If the *i*-th element of the augmented SQL parameter declaration list included in the descriptor of *COMS* includes a <locator indication>, then the <parameter type> PT_{i-1} immediately contained in $POVMS_{i-1}$ shall contain a <locator indication>.

2. *Rationale: Align the descriptor of a method specification and its initializing rules.*

Replace General Rule 1) with:

1) Let *STDS* be the descriptor of *D*, and let *DCMS* be the descriptor of the corresponding original method specification *COMS*. A method specification descriptor *DOMS* is created for *OVMS*. *DOMS* includes:

a) The <method name> *MN*.

b) The <specific method name> contained in *POVMS*.

c) An indication that the method specification is overriding.

d) The augmented SQL parameter declaration list *APDL*.

e) For every SQL parameter in *APDL*, the locator indication of the corresponding SQL parameter included in *DCMS* (if any).

f) The <returns data type> contained in *POVMS*.

g) The <result cast from type> included in *DCMS* (if any).

h) The locator indication contained in the <returns clause> included in the *DCMS*.

i) The <language name> included in *DCMS*.

j) If the <language name> is not SQL, then the <parameter style> included in *DCMS*.

k) The determinism indication included in *DCMS*.

l) The SQL-data access indication included in *DCMS* (if any).

m) The indication included in *DCMS*, whether the method should be invoked if any argument is the null value.

n) The CURRENT_TIMESTAMP as the value of the creation timestamp.

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

11.52 <alter routine statement>

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

1. *Rationale: Editorial correction for incomplete descriptions.*

   Replace General Rules 1) and 2) with:

   1) If *SR* is not a method, then the routine descriptor of *SR* is modified:

      a) If <returned result sets characteristic> is specified, then the maximum number of returned result sets is the value of <maximum returned result sets>.

      b) If <language clause> is specified, then the name of the language in which the body of the SQL-invoked routine is written is the <language name> contained in the <language clause>.

      c) If <external routine name> is specified, then the external name of the routine descriptor is <external routine name>.

      d) If <parameter style clause> is specified, then the routine descriptor includes an indication of whether the parameter passing style is PARAMETER STYLE SQL or PARAMETER STYLE GENERAL.

      e) If the <SQL-data access indication> is specified, then the routine descriptor includes an indication of whether the SQL-invoked routine's <SQL-data access indication> is READS SQL DATA, MODIFIES SQL DATA, CONTAINS SQL, or NO SQL.

      f) If <null-call clause> is specified, then the routine descriptor includes an indication of whether the SQL-invoked routine is a null-call function.

   2) If *SR* is a method, then let *DMS* be the descriptor of the corresponding method specification. *DMS* is modified:

      a) If <language clause> is specified, then the method specification descriptor includes the <language name> contained in the <language clause>.

      b) If <parameter style clause> is specified, then the method specification descriptor includes an indication of whether the parameter passing style is PARAMETER STYLE SQL or PARAMETER STYLE GENERAL.

      c) If the <SQL-data access indication> is specified, then the method specification descriptor includes an indication of whether the SQL-invoked routine's <SQL-data access indication> is READS SQL DATA, MODIFIES SQL DATA, CONTAINS SQL, or NO SQL.

      d) If <null-call clause> is specified, then the method specification descriptor includes an indication of whether the method should not be invoked if any argument is the null value.

2. *Rationale: Editorial correction.*

   Replace General Rule 3) a) with:

   3) ...

      a) If <external routine name> is specified, then the external name of the routine descriptor is <external routine name>.

      b) If <parameter style clause> is specified, then the routine descriptor includes an indication of whether the parameter passing style is PARAMETER STYLE SQL or PARAMETER STYLE GENERAL.

11.53 <drop routine statement>

This Subclause is modified by Subclause 9.25, "<drop routine statement>", in ISO/IEC 9075-4.
This Subclause is modified by Subclause 11.11, "<drop routine statement>", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 9.10, "<drop routine statement>", in ISO/IEC 9075-13.

1. *Rationale: Editorial correction.*

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

   5) ...

      b) *SN* shall not be included in any of the following:

11.55 <drop user-defined cast statement>

This Subclause is modified by Subclause 9.26, "<drop user-defined cast statement>", in ISO/IEC 9075-4.

1. *Rationale: Use the correct term.*

   Replace Syntax Rule 5) c) with:

   5) ...

      c) The type designator of *SDT* is in the type precedence list of *P*.

11.56 <user-defined ordering definition>

This Subclause is modified by Subclause 11.13, "<user-defined ordering definition>", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 9.11, "<user-defined ordering definition>", in ISO/IEC 9075-13.

1. *Rationale: Editorial correction.*

   Replace General Rule 3) b) with:

   3) ...

      b) If MAP is specified, then the ordering category in the user-defined type descriptor of *UDT* is set to MAP.

12 Access control

12.3 <privileges>

This Subclause is modified by Subclause 10.2, "<privileges>", in ISO/IEC 9075-4.
This Subclause is modified by Subclause 13.1, "<privileges>", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 10.2, "<privileges>", in ISO/IEC 9075-13.

1. *Rationale: Editorial correction.*

   Replace General Rule 8) with:

   8) An <authorization identifier> *B* has the WITH ADMIN OPTION on a role if a role authorization descriptor identifies the role as granted to *B* WITH ADMIN OPTION or a role authorization descriptor identifies it as granted WITH ADMIN OPTION to another applicable role for *B*.

12.4 <role definition>

1. *Rationale: Editorial correction.*

   Replace General Rule 4) with:

   4) A grantable role authorization descriptor is created whose role name is <role name>, whose grantor is "_SYSTEM", and whose grantee is *A*.

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: Correct conditions under which an exception is raised.*

   Replace General Rule 3) b) with:

   3) ...

      b) If the <revoke statement> is a <revoke role statement>, then, for every role *R* identified by a <role revoked>, if there does not exists a grantable role authorization descriptor whose role name is *R*, and whose grantee is *A* or an applicable role of *A*, then an exception condition is raised: *invalid grantor*.

2. *Rationale: Editorial correction.*

   Replace General Rule 4) a) with:

   4) ...

      a) If the <revoke statement> is a <revoke privilege statement>, then, for every <grantee> specified, a set of privilege descriptors is identified. A privilege descriptor *P* is said to be *identified* if it belongs to the set of privilege descriptors that defined, for any <action> explicitly or implicitly contained in <privileges>, that <action> on *O*, or any of the objects in *S*, granted by *A* to <grantee>.

         NOTE 362 — Column privilege descriptors become identified when <action> explicitly or implicitly contains a <privilege column list>. Table/method descriptors become identified when <action> explicitly or implicitly contains a <privilege method list>.

3. *Rationale: Editorial correction.*

   Replace General Rule 15) a) with:

   15) ...

      a) SELECT privilege on at least one column of every table identified by a <table reference> contained in *QE*.

4. *Rationale: Editorial correction.*

   Replace General Rule 15) l) with:

   15) ...

      l) SELECT privilege WITH HIERARCHY OPTION on at least one supertable of every typed table identified by a <table reference> that simply contains an <only spec> and that is contained in *QE*.

5. *Rationale: Correct the description about an abandoned domain constraint.*

   Replace General Rule 20) a) with:

   20) ...

a) REFERENCES privilege on at least one column of every table identified by a <table reference> contained in any <search condition> of *DC*.

6. *Rationale: Correct the description about an abandoned domain constraint.*

Replace General Rule 20) j) with:

20) ...

j) SELECT privilege WITH HIERARCHY OPTION on at least one supertable of every typed table identified by a <table reference> that simply contains an <only spec> and that is contained in any <search condition> of *DC*.

7. *Rationale: Drop generated column if privilege is lost for the invocation of an SQL-invoked routine invoked by the generated column.*

Add the following General Rule:

22) ...

d) *CD* has a generation expression *GE* and the revoke destruction action would result in *A1* no longer having in its applicable privileges EXECUTE privilege on any SQL-invoked routine that is the subject routine of any <routine invocation>, <method invocation>, <static method invocation> or <method reference> that is contained in *GE*.

8. *Rationale: Editorial correction.*

Replace General Rule 25) with:

25) *S1* is said to be *lost* if the revoke destruction action would result in *A1* no longer having in its applicable privileges USAGE privilege on the default character set included in *S1*.

9. *Rationale: Editorial correction.*

Replace General Rule 29) n) with:

29) ...

n) The table/method privilege on every table *T1* and every method *M* such that there is a <method reference> *MR* contained in the <SQL routine body> of *RD* such that *T1* is in the scope of the <value expression primary> of *MR* and *M* is the subject routine of *MR*.

10. *Rationale: Delete unnecessary descriptions for dependent privilege descriptors.*

Replace General Rule 31) with:

31) 09 13 If RESTRICT is specified, and there exists an abandoned privilege descriptor, abandoned view, abandoned table constraint, abandoned assertion, abandoned domain constraint, lost domain, lost column, lost schema, or a descriptor that includes an impacted data type descriptor, impacted collation, impacted character set, abandoned user-defined type, or abandoned routine descriptor, then an exception condition is raised: *dependent privilege descriptors still exist*.

13 SQL-client modules

13.4 Calls to an <externally-invoked procedure>

This Subclause is modified by Subclause 11.1, "Calls to an <externally-invoked procedure>", in ISO/IEC 9075-4.
This Subclause is modified by Subclause 14.3, "Calls to an <externally-invoked procedure>", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 13.1, "Calls to an <externally-invoked procedure>", in ISO/IEC 9075-14.

1. *Rationale: Distinguish data types of a host parameter and its corresponding argument.*

   Replace Syntax Rule 1) with:

   1) Let *n* be the number of <host parameter declaration>s in the <externally-invoked procedure> *EP* being called. Let PD_i, 1 (one) ≤ *i* ≤ *n*, be the *i*-th <host parameter declaration>. Let PDT_i be the <data type> contained in PD_i. Let ADT_i be the data type of the argument corresponding to PD_i on the call to *EP*.

2. *Rationale: Distinguish data types of a host parameter and its corresponding argument.*

   Replace Syntax Rule 3) c) with:

   1) For each *i*, 1 (one) ≤ *i* ≤ *n*, ADT_i shall be the data type listed in the "C data type" column of Table 17, "Data type correspondences for C", for which the corresponding row in the "SQL data type" column is PDT_i.

3. *Rationale: Distinguish data types of a host parameter and its corresponding argument.*

   Replace Syntax Rule 4) c) with:

   1) For each *i*, 1 (one) ≤ *i* ≤ *n*, ADT_i shall be the data type listed in the "COBOL data type" column of Table 18, "Data type correspondences for COBOL", for which the corresponding row in the "SQL data type" column is PDT_i.

4. *Rationale: Distinguish data types of a host parameter and its corresponding argument.*

   Replace Syntax Rule 5) c) with:

   1) For each *i*, 1 (one) ≤ *i* ≤ *n*, ADT_i shall be the data type listed in the "Fortran data type" column of Table 19, "Data type correspondences for Fortran", for which the corresponding row in the "SQL data type" column is PDT_i.

5. *Rationale: Distinguish data types of a host parameter and its corresponding argument.*

   Replace Syntax Rule 6) c) with:

   1) For each *i*, 1 (one) ≤ *i* ≤ *n*, ADT_i shall be the data type listed in the "M data type" column of Table 20, "Data type correspondences for M", for which the corresponding row in the "SQL data type" column is PDT_i.

6. *Rationale: Distinguish data types of a host parameter and its corresponding argument.*

   Replace Syntax Rule 7) c) with:

   1) For each *i*, 1 (one) ≤ *i* ≤ *n*, ADT_i shall be the data type listed in the "Pascal data type" column of Table 21, "Data type correspondences for Pascal", for which the corresponding row in the "SQL data type" column is PDT_i.

7. *Rationale: Distinguish data types of a host parameter and its corresponding argument.*

   Replace Syntax Rule 8) c) with:

   1) For each *i*, 1 (one) ≤ *i* ≤ *n*, ADT_i shall be the data type listed in the "PL/I data type" column of Table 22, "Data type correspondences for PL/I", for which the corresponding row in the "SQL data type" column is PDT_i.

8. *Rationale: Supply rules for Ada and Pascal.*

   Replace General Rule 2) b) with:

   2) ...

      b) If *DT* identifies a CHARACTER(*L*) data type and the caller language of *EP* is either ADA, COBOL, FORTRAN, or PASCAL, or *DT* identifies a CHARACTER VARYING(*L*) data type and the caller language of *EP* is M, or *DT* identifies a CHARACTER(*L*) data type or CHARACTER VARYING(*L*) data type and the caller language of *EP* is PLI, then a reference to *PN* is implicitly treated as a character string type value in the specified character set in which the octets of *PI* are the corresponding octets of that value.

9. *Rationale: Supply rules for Ada and Pascal.*

   Insert the following General Rule:

   2) ...

      f) ...

         0.1) If the caller language of *EP* is ADA, then a reference to *PN* is implicitly treated as

         Case:

1) If *DT* identifies a CHARACTER LARGE OBJECT type, then a large object character string containing the *PN*.*PN*_LENGTH characters of *PN*.*PN*_DATA starting at character number 1 (one) in the same order that the characters appear in *PN*.*PN*_DATA.

2) If *DT* identifies a BINARY LARGE OBJECT type, then a binary large object string containing the *PN*.*PN*_LENGTH octets of *PN*.*PN*_DATA starting at octet number 1 (one) in the same order that the octets appear in *PN*.*PN*_DATA.

10. *Rationale: Supply rules for Ada and Pascal.*

Insert the following General Rule:

2) ...

f) ...

iii.1) If the caller language of *EP* is PASCAL, then a reference to *PN* is implicitly treated as

Case:

1) If *DT* identifies a CHARACTER LARGE OBJECT type, then a large object character string containing the *PN*.*PN*_LENGTH characters of *PN*.*PN*_DATA starting at character number 1 (one) in the same order that the characters appear in *PN*.*PN*_DATA.

2) If *DT* identifies a BINARY LARGE OBJECT type, then a binary large object string containing the *PN*.*PN*_LENGTH octets of *PN*.*PN*_DATA starting at octet number 1 (one) in the same order that the octets appear in *PN*.*PN*_DATA.

11. *Rationale: Supply rules for Ada and Pascal.*

Insert the following General Rule:

2) ...

f.1) If *DT* identifies a BINARY(*L*) or BINARY VARYING(*L*) data type and the caller language of *EP* is ADA, then

Case:

i) If *DT* identifies a BINARY(L) data type, then a reference to *PN* is implicitly treated as a binary string type value in which the octets of *PI* are the corresponding octets of that value.

ii) If *DT* identifies a BINARY VARYING(*L*) data type, then a reference to *PN* is implicitly treated as a binary string containing the *PN*.*PN*_LENGTH octets of *PN*.*PN*_DATA starting at octet number 1 (one) in the same order that the octets appear in *PN*.*PN*_DATA.

12. *Rationale: Supply rules for Ada and Pascal.*

Replace General Rule 2) h) with:

2) ...

h) If *DT* identifies a BINARY(*L*) data type and the caller language of *EP* is either COBOL, FORTRAN, or PASCAL, or *DT* identifies a BINARY(*L*) data type or BINARY VARYING(*L*) data type and the caller language of *EP* is PLI, then a reference to *PN* is implicitly treated as a binary string type value in which the octets of *PI* are the corresponding octets of that value.

13. *Rationale: Supply rules for Ada and Pascal.*

Replace General Rule 3) b) with:

3) ...

b) If *DT* identifies a CHARACTER(*L*) data type and the caller language of *EP* is either ADA, COBOL, FORTRAN, or PASCAL, then let *CL* be the maximum possible length in octets of *PN*. A reference to *PN* that assigns some value *SV* to *PN* implicitly assigns a value that is an SQL CHARACTER(*CL*) data type in which octets of the value are the corresponding octets of *SV*, padded on the right with <space>s as necessary to reach the length *CL*.

14. *Rationale: Define undefined symbol.*

Replace General Rule 3) c) with:

3) ...

c) If *DT* identifies a CHARACTER VARYING(*L*) data type and the caller language of *EP* is M, then let *CL* be the maximum possible length in octets of *PN*. A reference to *PN* that assigns some value *SV* to *PN* implicitly assigns a value that is an SQL CHARACTER VARYING(*ML*) data type in which octets of the value are the corresponding octets of *SV*, padded on the right with <space>s as necessary to reach the length *CL*. *ML* is the implementation-defined maximum length of variable-length character strings.

15. *Rationale: Supply rules for Ada and Pascal.*

Insert the following General Rule:

3) ...

h) ...

0.1) If the caller language of *EP* is ADA, then a reference to *PN* that assigns some value *SV* to *PN* implicitly assigns the value OCTET_LENGTH(*SV*) to *PN*.*PN*_ LENGTH and the value *SV* to *PN*.*PN*_ DATA.

16. *Rationale: Supply rules for Ada and Pascal.*

Insert the following General Rule:

3) ...

h) ...

iii.1) If the caller language of *EP* is PASCAL, then a reference to *PN* that assigns some value *SV* to *PN* implicitly assigns the value OCTET_LENGTH(*SV*) to *PN.PN_* LENGTH and the value *SV* to *PN.PN_* DATA.

17. *Rationale: Supply rules for Ada and Pascal.*

Replace General Rule 3) i) with:

3) ...

i) If *DT* identifies a BINARY(*L*) data type and the caller language of *EP* is either ADA or C, then let *CL* be the maximum possible length in octets of *PN*. A reference to *PN* that assigns some value *SV* to *PN* implicitly assigns a value that is an SQL BINARY(*CL*) data type in which octets of the value are the corresponding octets of *SV*, padded on the right with X'00's as necessary to reach the length *CL*.

j) If *DT* identifies a BINARY VARYING(*L*) data type and the caller language of *EP* is ADA, then a reference to *PN* that assigns some value *SV* to *PN* implicitly assigns the value OCTET_LENGTH(*SV*) to *PN.PN*_LENGTH and the value *SV* to *PN.PN*_DATA.

18. *Rationale: Supply rules for Ada and Pascal.*

Replace General Rule 3) k) with:

3) ...

k) If *DT* identifies a BINARY(*L*) data type and the caller language of *EP* is either COBOL, FORTRAN, or PASCAL then let *CL* be the maximum possible length in octets of *PN*. A reference to *PN* that assigns some value *SV* to *PN* implicitly assigns a value that is an SQL BINARY(*CL*) data type in which octets of the value are the corresponding octets of *SV*, padded on the right with X'00's as necessary to reach the length *CL*.

19. *Rationale: Supply rules for Ada and Pascal.*

Replace General Rule 3) l) i) with:

3) ...

l) ...

i) If *DT* identifies BINARY(*L*), then an SQL BINARY(*CL*) data type in which octets of the value are the corresponding octets of *SV*.

13.6 Data type correspondences

This Subclause is modified by Subclause 14.5, "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: Some editorial corrections and supplement of some Ada Data Types.*

   Replace Table 16 with:

09 14 Table 16 — Data type correspondences for Ada

SQL Data Type	Ada Data Type
SQLSTATE	Interfaces.SQL.SQL_STANDARD.SQLSTATE_TYPE
CHARACTER (*L U*) CHARACTER SET *CS*	Interfaces.SQL..CHAR(1..*L***k*) [1]
CHARACTER VARYING (*L*) CHARACTER SET *CS*	*None*
CHARACTER LARGE OBJECT (*L*) 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] END RECORD;
BINARY(*L*)	Interfaces.SQL.CHAR(1..*L***k*)
BINARY VARYING (*L*)	TYPE *HVN* IS RECORD *HVN*_RESERVED : Interfaces.SQL.INT; *HVN*_LENGTH : Interfaces.SQL.INT; *HVN*_DATA : Interfaces.SQL.CHAR(1..*L*); [2] END RECORD;
BINARY LARGE OBJECT(*L*)	TYPE *HVN* IS RECORD *HVN*_RESERVED : Interfaces.SQL.INT; *HVN*_LENGTH : Interfaces.SQL.INT; *HVN*_DATA : Interfaces.SQL.CHAR(1..*L*); [2] END RECORD;
NUMERIC(*P*,*S*)	*None*
DECIMAL(*P*,*S*)	*None*
SMALLINT	Interfaces.SQL.SMALLINT
INTEGER	Interfaces.SQL.INT
BIGINT	Interfaces.SQL.BIGINT
FLOAT(*P*)	*None*
REAL	Interfaces.SQL.REAL

SQL Data Type	Ada Data Type
DOUBLE PRECISION	Interfaces.SQL.DOUBLE_PRECISION
BOOLEAN	Interfaces.SQL.BOOLEAN
DATE	*None*
TIME(*T*)	*None*
TIMESTAMP(*T*)	*None*
INTERVAL(*Q*)	*None*
user-defined type	*None*
REF	Interfaces.SQL.CHAR(1..*N*)
ROW	*None*
ARRAY	*None*
MULTISET	*None*

[1] If *U* is OCTETS, then *k* is 1 (one); Otherwise, *k* is the maximum number of octets per character in the character set *CS*.
[2] *HVN* is the name of the host variable defined to correspond to the SQL data type.

2. *Rationale: Editorial correction.*

Replace Table 17 with:

09 14 Table 17 — Data type correspondences for C

SQL Data Type	C Data Type
SQLSTATE	`char`, with length 6
CHARACTER (*L U*) CHARACTER SET *CS*	*unit*, with length (*L*+1)**k* [1]
CHARACTER VARYING (*L U*) CHARACTER SET *CS*	*unit*, with length (*L*+1)**k* [1]

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

SQL Data Type	C Data Type
BINARY (*L*)	`char`, with length *L*
BINARY VARYING (*L*)	`struct { long hvn_reserved unsigned long hvn_length char hvn_data[L]; } hvn` [2]
BINARY LARGE OBJECT (*L*)	`struct { long hvn_reserved unsigned long hvn_length char hvn_data[L]; } hvn` [2]
NUMERIC (*P*,*S*)	*None*
DECIMAL (*P*,*S*)	*None*
SMALLINT	pointer to `short`
INTEGER	pointer to `long`
BIGINT	pointer to `long long`
FLOAT (*P*)	*None*
REAL	pointer to `float`
DOUBLE PRECISION	pointer to `double`
BOOLEAN	pointer to `long`
DATE	*None*
TIME (*T*)	*None*
TIMESTAMP (*T*)	*None*
INTERVAL (*Q*)	*None*
user-defined type	*None*
REF	`char`, with length *N*
ROW	*None*
ARRAY	*None*
MULTISET	*None*

SQL Data Type	C Data Type
[1] If *U* is OCTETS, then *unit* is `char` and *k* is 1 (one). Otherwise (that is, if *U* is CHARACTERS) then *unit* is an appropriate implementation-defined C data type (typically `char`, `unsigned char`, `unsigned short`, or `unsigned int`) and *k* is an appropriate sizing factor given the choice of *unit*. The choice of *unit* and *k* is implementation-defined, based on the character set *CS*. For example, for UTF32, *unit* might be `char` and *k* might be 4; or *unit* might be `unsigned int` and *k* might be 1 (one). [2] *hvn* is the name of the host variable defined to correspond to the SQL data type.	

3. *Rationale: Some editorial corrections and supplement of some M Data Types.*

Replace Table 20 with:

09 14 Table 20 — Data type correspondences for M

SQL Data Type	M Data Type
SQLSTATE	character, with maximum length at least 5
CHARACTER (*L U*) CHARACTER SET *CS*	*None*
CHARACTER VARYING (*L U*) CHARACTER SET *CS*	character with maximum length *L*k* [1]
CHARACTER LARGE OBJECT (*L U*) CHARACTER SET *CS*	INT *HVN*_RESERVED INT *HVN*_LENGTH VARCHAR *HVN*_DATA *L*k* [1 2]
BINARY (*L*)	*None*
BINARY VARYING (*L*)	INT *HVN*_RESERVED INT *HVN*_LENGTH VARCHAR *HVN*_DATA *L* [2]
BINARY LARGE OBJECT (*L*)	INT *HVN*_RESERVED INT *HVN*_LENGTH VARCHAR *HVN*_DATA *L* [2]
NUMERIC (*P*,*S*)	character
DECIMAL (*P*,*S*)	character
SMALLINT	*None*
INTEGER	character
BIGINT	*None*
FLOAT (*P*)	*None*
REAL	character

SQL Data Type	M Data Type
DOUBLE PRECISION	*None*
BOOLEAN	*None*
DATE	*None*
TIME (*T*)	*None*
TIMESTAMP (*T*)	*None*
INTERVAL (*Q*)	*None*
user-defined type	*None*
REF	character
ROW	*None*
ARRAY	*None*
MULTISET	*None*

[1] If *U* is OCTETS, then *k* is 1 (one); Otherwise, *k* is the maximum number of octets per character in the character set *CS*.
[2] *hvn* is the name of the host variable defined to correspond to the SQL data type.

4. *Rationale: Some editorial corrections and supplement of some Pascal Data Types.*

Replace Table 21 with:

09 14 Table 21 — Data type correspondences for Pascal

SQL Data Type	Pascal Data Type
SQLSTATE	PACKED ARRAY [1..5] OF CHAR
CHARACTER (1)	CHAR
CHARACTER (*L U*) CHARACTER SET *CS*, *L* > 1	PACKED ARRAY [1..*L***k*] OF CHAR [1]
CHARACTER VARYING (*L U*) CHARACTER SET *CS*	*None*
CHARACTER LARGE OBJECT (*L U*) CHARACTER SET *CS*	VAR *HVN* = RECORD *HVN*_RESERVED : INTEGER; *HVN*_LENGTH : INTEGER; *HVN*_DATA : PACKED ARRAY [1..*L***k*] OF CHAR; END; [1] [2]

SQL Data Type	Pascal Data Type
BINARY (*L*)	PACKED ARRAY [1..*L*] OF CHAR
BINARY VARYING (*L*)	*None*
BINARY LARGE OBJECT (*L*)	VAR *HVN* = RECORD *HVN*_RESERVED : INTEGER; *HVN*_LENGTH : INTEGER; *HVN*_DATA : PACKED ARRAY [1..*L*] OF CHAR; END; [1]
NUMERIC (*P*,*S*)	*None*
DECIMAL (*P*,*S*)	*None*
SMALLINT	*None*
INTEGER	INTEGER
BIGINT	*None*
FLOAT (*P*)	*None*
REAL	REAL
DOUBLE PRECISION	*None*
BOOLEAN	BOOLEAN
DATE	*None*
TIME (*T*)	*None*
TIMESTAMP (*T*)	*None*
INTERVAL (*Q*)	*None*
user-defined type	*None*
REF	PACKED ARRAY[1..*N*] OF CHAR
ROW	*None*
ARRAY	*None*
MULTISET	*None*

[1] If *U* is OCTETS, then *k* is 1 (one); Otherwise, *k* is the maximum number of octets per character in the character set *CS*.
[2] *hvn* is the name of the host variable defined to correspond to the SQL data type.

5. *Rationale: Some editorial corrections and supplement of some missing PL/I Data Types.*

Replace Table 22 with:

09 14 Table 22 — Data type correspondences for PL/I

SQL Data Type	PL/I Data Type
SQLSTATE	CHARACTER(5)
CHARACTER (*L U*) CHARACTER SET *CS*	CHARACTER(*L***k*) [1]
CHARACTER VARYING (*L U*) CHARACTER SET *CS* [1]	CHARACTER (*L***k*) VARYING

CHARACTER LARGE OBJECT (*L U*) CHARACTER SET *CS*	`DCL 01 hvn`
	`  49 hvn_reserved FIXED BINARY (31)`
	`  49 hvn_length FIXED BINARY (31)`
	`  49 hvn_data CHAR (L*k)` [1] [2]`;`

BINARY (*L*)	CHARACTER (*L*)
BINARY VARYING (*L*)	CHARACTER (*L*) VARYING

BINARY LARGE OBJECT (*L*)	`DCL 01 hvn`
	`  49 hvn_reserved FIXED BINARY (31)`
	`  49 hvn_length FIXED BINARY (31)`
	`  49 hvn_data CHAR (n);` [2]

NUMERIC(*P*,*S*)	*None*
DECIMAL (*P*,*S*)	FIXED DECIMAL (*P*,*S*)
SMALLINT	FIXED BINARY(*SPI*), where *SPI* is implementation-defined
INTEGER	FIXED BINARY(*PI*), where *PI* is implementation-defined
BIGINT	FIXED BINARY(*BPI*), where *BPI* is implementation-defined
FLOAT (*P*)	FLOAT BINARY (*P*)
REAL	*None*
DOUBLE PRECISION	*None*
BOOLEAN	BIT(1)
DATE	*None*
TIME (*T*)	*None*

SQL Data Type	PL/I Data Type
TIMESTAMP (*T*)	*None*
INTERVAL (*Q*)	*None*
user-defined type	*None*
REF	CHARACTER(*N*) VARYING
ROW	*None*
ARRAY	*None*
MULTISET	*None*
[1] If *U* is OCTETS, then *k* is 1 (one); Otherwise, *k* is the maximum number of octets per character in the character set *CS*. [2] *hvn* is the name of the host variable defined to correspond to the SQL data type	

14 Data manipulation

14.4 <open statement>

This Subclause is modified by Subclause 12.2, "<open statement>", in ISO/IEC 9075-4.

1. *Rationale: Editorial correction.*

   Replace Syntax Rule 3) with:

   3) Let *CDD* be the cursor declaration descriptor of the standing cursor identified by *CN*.

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: Correct the invocation of Subclause 9.2, "Store assignment".*

   Replace Syntax Rule 9) a) with:

   9) ...

a) If the <fetch target list> contains a single <target specification> *TS* and the degree of *T* is greater than 1 (one), then the declared type of *TS* shall be a row type.

Case:

i) 04 If *TS* is an <SQL parameter reference>, then the Syntax Rules of Subclause 9.2, "Store assignment", apply to *TS* as *TARGET* and an arbitrary value of the row type of *T* as *VALUE*.

ii) Otherwise, the Syntax Rules of Subclause 9.1, "Retrieval assignment", apply to *TS* as *TARGET* and an arbitrary value of the row type of *T* as *VALUE*.

2. *Rationale: Correct the invocation of Subclause 9.2, "Store assignment".*

Replace Syntax Rules 9) b) ii), 9) b) iii),9) b) iv) with:

9) ...

a) ...

i.1) For *i* varying from 1 (one) to *NTS*, let CS_i be an arbitrary value of the declared type of the *i*-th column of *T*.

ii) 04 For each <target specification> $TS1_i$, 1 (one) ≤ *i*, ≤ *NTS*, that is a either an <SQL parameter reference> or a <target array element specification>,

Case:

1) If $TS1_i$ contains a <simple value specification>, then the Syntax Rules of Subclause 9.2, "Store assignment", apply to an arbitrary site whose declared type is the declared type of $TS1_i$ as *TARGET* and CS_i as *VALUE*.

2) Otherwise, the Syntax Rules of Subclause 9.2, "Store assignment", apply to $TS1_i$ as *TARGET* and CS_i as *VALUE*.

iii) 10 For each <target specification> $TS2_i$, 1 (one) ≤ *i*, ≤ *NTS*, that is a <host parameter specification>, the Syntax Rules of Subclause 9.1, "Retrieval assignment", apply to $TS2_i$ as *TARGET* and CS_i as *VALUE*.

iv) 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", apply to $TS2_i$ as *TARGET* and CS_i as *VALUE*.

14.7 <select statement: single row>

This Subclause is modified by Subclause 12.5, "<select statement: single row>", in ISO/IEC 9075-4.
This Subclause is modified by Subclause 11.14, "<select statement: single row>", in ISO/IEC 9075-10.
This Subclause is modified by Subclause 14.2, "<select statement: single row>", in ISO/IEC 9075-14.

1. *Rationale: Correct the invocation of Subclause 9.2, "Store assignment".*

   Replace Syntax Rule 9) with:

   3) ...

      a) If the <select target list> contains a single <target specification> *TS* and the degree of *T* is greater than 1 (one), then the declared type of *TS* shall be a row type.

         Case:

         i) If *TS* is an <SQL parameter reference>, then the Syntax Rules of Subclause 9.2, "Store assignment", apply to *TS* as *TARGET* and an arbitrary value of the row type of *T* as *VALUE*.

         ii) Otherwise, the Syntax Rules of Subclause 9.1, "Retrieval assignment", apply to *TS* as *TARGET* and an arbitrary value of the row type of *T* as *VALUE*.

14.11 <insert statement>

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

1. *Rationale: Correct the use of illegal BNF term.*

   Replace Syntax Rule 1) with:

   1) Let *TN* be the <table name> contained in <insertion target>. Let *T* be the table identified by *TN*.

2. *Rationale: Correct the use of non-BNF term in specifying syntactic containment.*

   Replace Syntax Rule 16) with:

   16) If the <insert statement> is contained in a <triggered SQL statement>, then <insert columns and source> shall not contain a <value specification> that specifies a parameter reference.

14.12 <merge statement>

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

1. *Rationale: Allow for situations where both a <merge when not match clause> and a <merge when matched clause> are present.*

   Replace Syntax Rule 2) with:

   2) If <merge when not matched clause> is specified, then *T* shall be insertable-into or trigger insertable-into.

2.1) If <merge when matched clause> is specified, then *T* shall be updatable or trigger updatable.

14.17 <free locator statement>

1. *Rationale: Use the correct term.*

   Replace Syntax Rule 2) with:

   2) Each host variable identified by the <embedded variable name> immediately contained in <locator reference> shall be a binary large object locator variable, a character large object locator variable, an array locator variable, a multiset locator parameter, or a user-defined type locator variable.

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: Nullary functions other than <datetime value function> should be sensitive to <routine invocation>s.*

   Replace General Rule 4) a) ii) with

   4) ...

      a) ...

         ii) Each <value specification> generally contained in *S* without an intervening <routine invocation> that is CURRENT_USER, CURRENT_ROLE, SESSION_USER, SYSTEM_USER, CURRENT_CATALOG, CURRENT_SCHEMA, CURRENT_PATH, CURRENT_DEFAULT_TRANSFORM_GROUP, or CURRENT_TRANSFORM_GROUP_FOR_TYPE <path-resolved user-defined type name> is replaced by a <literal> denoting the value resulting from evaluation of CURRENT_USER, CURRENT_ROLE, SESSION_USER, SYSTEM_USER, CURRENT_CATALOG, CURRENT_SCHEMA, CURRENT_PATH, CURRENT_DEFAULT_TRANSFORM_GROUP, or CURRENT_TRANSFORM_GROUP_FOR_TYPE <path-resolved user-defined type name>, respectively, with all such evaluations effectively done at the same instant in time.

2. *Rationale: Supply the rule for a cursor position.*

   Insert the following General Rule:

4) ...

h) *CR* is placed in the open state and its position is before the first row of *TT*.

15.6 Effect of a positioned update

1. *Rationale: Updating a column that is used to compute a sort key can leave a cursor in an implementation-dependent position.*

Add the following General Rule:

18) If a column *C* of a base table is modified, and the evaluation of the <value expression> of some <sort key> simply contained in the <query expression> of the <cursor specification> of *CR* referenced *C*, then the position of *CR* is implementation-dependent.

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: Correct surplus changes of an identity column value.*

Replace General Rule 5) with:

5) For every table *ST* that is identified for replacement processing,

a) If some column *IC* of *T* is the identity column of *ST*, and for each row *RR* identified for replacement in *ST*, if the site *ICS* corresponding to *IC* in the replacement row for *RR* is marked as *unassigned*, then:

i) *ICS* is no longer marked as unassigned.

ii) Let *NV* be the result of applying the General Rules of Subclause 9.23, "Generation of the next value of a sequence generator", with the sequence generator descriptor included in the column descriptor of *IC* as *SEQUENCE*.

Case:

1) If the declared type of *IC* is a distinct type *DIST*, then let *ICNV* be *DIST*(*NV*).

2) Otherwise, let *ICNV* be *NV*.

iii) The General Rules of Subclause 9.2, "Store assignment", are applied with *ICNV* as *VALUE*, *ICS* as *TARGET*.

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* is not empty:

c) Case:

i) If a state change *SC* exists in *SSC* with subject table *ST*, trigger event UPDATE, and column list *STL*, then the row pairs formed by pairing each row identified for replacement in *ST* with its corresponding replacement row are added to the set of transitions of *SC*.

ii) Otherwise, a state change *SC* is added to *SSC* as follows:

1) The set of transitions of *SC* consists of row pairs formed by pairing each row identified for replacement in *ST* with its corresponding replacement row.

2) The trigger event of *SC* is UPDATE.

3) The subject table of *SC* is *ST*.

4) The column list of *SC* is *STL*.

5) The set of statement-level triggers for which *SC* is considered as executed is empty.

6) The set of row-level triggers consists of each row-level trigger that is activated by *SC*, paired with the empty set (of rows considered as executed).

2. *Rationale: Make the change of a generated column value reflect to a target table.*

Replace General Rule 8) with:

8) For every table *T* in *TT*, for every table *ST* that is a supertable or a subtable of *T*, for every row *R* that is identified for replacement in *ST*, *R* is replaced by its corresponding replacement row. *R* is no longer identified for replacement. *ST* is no longer identified for replacement processing.

15.14 Effect of replacing some rows in a derived table

1. *Rationale: Editorial correction.*

Replace General Rule 3) c) iii) 2) with:

3) ...

c) ...

iii) ...

2) A candidate row *CNRI* of TT_i 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 includes every column of TT_i.

15.17 Execution of referential actions

1. *Rationale: Add missing text.*

   Replace General Rule 10) a) ii) 2) A) with:

   10) ...

   a) ...

   ii) ...

   2) ...

   A) Each matching row *MR* in *F* is paired with the candidate replacement row *NMR*, formed by copying *MR* and setting each referencing column in the copy that corresponds with a referenced column to the null value. *MR* is identified for replacement by *NMR* in *F*. The set of (*MR*, *NMR*) pairs is the replacement set for *F*.

2. *Rationale: Editorial correction.*

   Replace General Rule10) b) i) 1) with:

   10) ...

   b) ...

   i) ...

   1) Each unique matching row *UMR* in *F* that contains a non-null value in the referencing column *C1* in *F* that corresponds to the updated referenced column *C2* is paired with the candidate replacement row *NUMR*, formed by copying *UMR* and setting *C1* in the copy to the new value *V* of *C2*, provided that, in all updated rows in the referenced table that formerly had, during the same execution of the same innermost SQL-statement, that unique matching row as a matching row, the values in *C2* have all been updated to a value that is not distinct from *V*. If this last condition is not satisfied, then an exception condition is raised: *triggered data change violation*. *UMR* is identified for replacement by *NUMR* in *F*. The set of (*UMR*, *NUMR*) pairs is the replacement set for *F*.

   NOTE 12 — Because of the Rules of Subclause 8.2, "<comparison predicate>", on which the definition of "distinct" relies, the values in *C2* may have been updated to values that are not distinct, yet are not identical. Which of these non-distinct values is used for the cascade operation is implementation-dependent.

15.19 Execution of triggers

1. *Rationale: Editorial correction.*

   Replace General Rule 4) a) i) with:

4) ...

a) ...

i) If *TE* is DELETE, then the old transition table for the invocation of *TA* is *ST*. If *TR* is a row-level trigger, then the value of the old transition variable for the invocation of *TA* is *T*.

20 Dynamic SQL

20.1 Description of SQL descriptor areas

This Subclause is modified by Subclause 17.1, "Description of SQL descriptor areas", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 17.1, "Description of SQL descriptor areas", in ISO/IEC 9075-14.

1. *Rationale: Editorial correction.*

   Replace Syntax Rule 6) m) with:

   6) ...

   m) TYPE indicates REF, LENGTH is the length in octets for the REF type, and USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are a valid fully qualified user-defined type name, and SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME are a valid fully qualified table name.

2. *Rationale: Editorial correction.*

   Replace the lead text of Syntax Rule 7) b) with:

   7) ...

   b) Either TYPE indicates CHARACTER VARYING and *T* is specified by CHARACTER VARYING(*L*) or TYPE indicates CHARACTER LARGE OBJECT and *T* is specified by CHARACTER LARGE OBJECT(*L*), where the <character set specification> formed by the values of CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, and CHARACTER_SET_NAME identifies the character set of *SVT* and

   Case:

20.6 <prepare statement>

This Subclause is modified by Subclause 15.1, "<prepare statement>", in ISO/IEC 9075-4.
This Subclause is modified by Subclause 17.2, "<prepare statement>", in ISO/IEC 9075-9.

This Subclause is modified by Subclause 17.4, "<prepare statement>", in ISO/IEC 9075-14.

1. *Rationale: Editorial correction.*

   Replace the lead text of General Rule 5) with:

   5) 04 Let *MP* be the implementation-defined maximum value of <precision> for the NUMERIC data type. Let *ML* be the implementation-defined maximum length of varying-length character strings. For each <value expression> *DP* in *P* or *PS* that meets the criteria for *DPV*, let *DT* denote its declared type. The syntactic substitutions specified in Subclause 14.15, "<set clause list>", shall not be applied until the data types of <dynamic parameter specification>s are determined by this General Rule.

2. *Rationale: Editorial correction.*

   Replace the lead text General Rules 5) a) v) with:

   5) ...

      a) ...

         v) If *DP* is either *X1* or *X2* in a <position expression> of the form "POSITION (*X1* IN *X2*)", and if *DP* is *X1* (*X2*), then

            Case:

3. *Rationale: Editorial correction.*

   Replace the lead text General Rules 5) a) vi) with:

   5) ...

      a) ...

         vi) If *DP* is either *X2* or *X3* in a <string value function> of the form "SUBSTRING (*X1* FROM *X2* FOR *X3*)" or "SUBSTRING (*X1* FROM *X2*)", then *DT* is NUMERIC (*MP*, 0).

4. *Rationale: Editorial correction.*

   Replace the lead text General Rules 5) a) viii) with:

   5) ...

      a) ...

         viii) If *DP* is any of *X1*, *X2*, *X3*, or X4 in a <string value function> of the form "OVERLAY (*X1* PLACING *X2* FROM *X3* FOR X4)" or "OVERLAY (*X1* PLACING *X2* FROM *X3*)", then

               Case:

5. *Rationale: Clarify the data type of dynamic parameters in <case expression>*

Replace General Rule 5) a) xviii) with:

5) ...

a) ...

xviii) If *DP* is a <case operand> or <when operand> simply contained in a <simple case> *CE* or *DP* represents the value of a subfield *SF* of the declared type of such a <case operand> or <when operand>, then:

1) Let *RT* be the result of applying the Syntax Rules of Subclause 9.3, "Result of data type combinations", to the set union of the following sets of declared types:

A) The set consisting of the declared type of the <case operand>.

B) The set consisting of the declared type of any <when operand> of *CE* that is a <row value predicand>.

C) The set consisting of the declared types of any <row value predicand> simply contained in any <comparison predicate part 2>, <between predicate part 2>, <character like predicate part 2>, <octet like predicate part 2>, <similar predicate part 2>, <regex like predicate part 2>, <overlaps predicate part 2>, <distinct predicate part 2>, or <member predicate part 2> that is simply contained in *CE*.

D) The set consisting of the declared row type of a <table subquery> simply contained in an <in predicate part 2> or <quantified comparison predicate part 2> simply contained in *CE*.

E) The set consisting of the declared types of the <row value expression>s simply contained in an <in value list> simply contained in an <in predicate part 2> simply contained in *CE*.

NOTE 448.1 — The following "part 2" predicates do not have any value expressions with declared type information: <null predicate part 2>, <normalized predicate part 2>, <set predicate part 2>, <type predicate part 2>.

2) Case:

A) If *DP* is a <case operand> or <when operand> simply contained in *CE*, then *DT* is *RT*.

B) Otherwise, *DT* is the declared type of the subfield of *RT* that corresponds to *SF*.

20.9 <describe statement>

This Subclause is modified by Subclause 17.4, "<describe statement>", in ISO/IEC 9075-9.

1. *Rationale: Correct reference to referenceable tables.*

   Replace General Rule 7) d) viii) 2) with:

   7) ...

      d) ...

         viii) ...

            2) SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME are set to the fully qualified name of the referenceable table.

20.17 <dynamic fetch statement>

This Subclause is modified by Subclause 17.11, "<dynamic fetch statement>", in ISO/IEC 9075-9.

1. *Rationale: Make the rule consistent with NOTE 460.*

   Replace General Rule 6) with:

   6) If an exception condition is raised during the assignment of a value to a target, then the values of all targets are implementation-dependent.

      NOTE 460 — It is implementation-dependent whether *CR* remains positioned on the current row when an exception condition is raised during the derivation of any <derived column>.

21 Embedded SQL

21.3 <embedded SQL Ada program>

This Subclause is modified by Subclause 18.1, "<embedded SQL Ada program>", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 18.2, "<embedded SQL Ada program>", in ISO/IEC 9075-14.

1. *Rationale: Supply description of a large object length.*

   Replace Syntax Rule 5) g) with:

   5) ...

      g) The syntax

         ```
         SQL TYPE IS BLOB ( L )
         ```

         for a given <Ada host identifier> *HVN* shall be replaced by