INTERNATIONAL STANDARD ISO/IEC 9075-2:2016
TECHNICAL CORRIGENDUM 2

Published 2022-06

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 2

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

RECTIFICATIF TECHNIQUE 2

Technical Corrigendum 2 to ISO/IEC 9075-2:2016 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange*.

ICS 35.060 **Ref. No. ISO/IEC 9075-2:2016/Cor.2:2022(E)**

Published in Switzerland

2 Normative references

2.2 Other international standards

1. *Rationale: Replace invalid URI.*

   Replace the normative reference for RFC 7159 with:

   [RFC7159] Internet Engineering Task Force, RFC 7159, *The JavaScript Object Notation (JSON) Data Interchange Format*, March 2014; https://datatracker.ietf.org/doc/rfc7159/

3 Definitions, notations, and conventions

3.1 Definitions

3.1.6 Definitions provided in Part 2

1. *Rationale: Correct the definition of whitespace.*

   Replace definition 3.1.6.77 with:

   3.1.6.1 white space
   sequence of one or more characters that have the Unicode property White_Space

   NOTE 1 — White space is typically used to separate <nondelimiter token>s from one another in SQL text, and is always permitted between two tokens in SQL text.

4 Concepts

4.3 Character strings

4.3.1 Introduction to character strings

1. *Rationale: Correct the definition of truncating whitespace.*

   Replace the 6th paragraph with:

   With two exceptions, a character string expression is assignable only to sites of a character string type whose character set is the same. The exceptions are as specified in Subclause 4.2.8, "Universal character sets", and such other cases as may be implementation-defined. If a store assignment would result in the loss of non-<truncating whitespace> characters due to truncation, then an exception condition is raised. If a retrieval assignment or evaluation of a <cast specification> would result in the loss of characters due to truncation, then a warning condition is raised.

4.6 Datetimes and intervals

4.6.2 Datetimes

1. *Rationale: Clarify the precision of timestamps.*

   Add the following paragraph after the 13th paragraph:

   When a timestamp value is called for as part of a descriptor or context definition in this Clause (for example, "The creation timestamp" in a descriptor), and no additional information as to the specific type or precision is supplied, an implementation-defined timestamp type is implicit.

4.7 User-defined types

4.7.3 Structured types

4.7.3.3 Constructors

1. *Rationale: Use the correct term.*

   Replace Note 33 with:

   > NOTE 33 — SQL-invoked constructor methods are original methods that cannot be overloaded. An SQL-invoked constructor method and a regular SQL-invoked function may exist such that they have equivalent routine names, the types of the first parameter of the method's augmented SQL parameter declaration list and the function's parameter list are the same, and the types of the corresponding remaining parameters (if any) are identical according to the Syntax Rules of Subclause 9.24, "Data type identity".

4.23 Integrity constraints

4.23.3 Table constraints

4.23.3.2 Unique constraints

1. *Rationale: Clarify uniqueness in the presence of nulls.*

   Replace the 2nd, 3rd and 4th paragraphs with:

   Let *T* be a table and let *R1* and *R2* be two rows of *T*. If there is some column for which the corresponding values from *R1* and *R2* are not equal, *R1* and *R2* are *unique with nulls distinct*. If there is some column for which the corresponding values from *R1* and *R2* are distinct, *R1* and *R2* are *unique with nulls not distinct*. If all column values of both *R1* and *R2* are the null value, it is implementation-defined whether *R1* and *R2* are unique with nulls distinct, or unique with nulls not distinct. It is implementation-defined whether the *implementation uniqueness rule* is unique with nulls distinct or unique with nulls not distinct.

   If the table descriptor for base table *T* includes a unique constraint descriptor indicating that the unique constraint was defined with PRIMARY KEY, then the columns of that unique constraint constitute the *primary key* of *T*. A table that has a primary key cannot have a proper supertable.

Let *T* be a base table and let *R1* and *R2* be two rows of *T*. Let *UC* be a unique constraint on *T*. If *T* is a system-versioned table, let *R1* and *R2* be two current system rows of *T*. If *UC* includes a <without overlap specification> *WOS*, let *ATPN* be the <application time period name> contained in *WOS*; *R1* and *R1* must additionally be such that the *ATPN* period values of *R1* and *R2* overlap. Let *UR1* and *UR2* be rows containing only the unique columns of *R1* and *R2* respectively.

UC is satisfied if and only if for every such *UR1* and *UR2*, the implementation uniqueness rule holds.

4.39 SQL-statements

4.39.5 SQL-statement atomicity and statement execution contexts

This Subclause is modified by Subclause 4.10.4, "SQL-statement atomicity and statement execution contexts", in ISO/IEC 9075-4.

1. *Rationale: Clarify delta tables.*

   Add the following bullet to the bullet list of the 10th paragraph:

   — A set of old delta tables and new delta tables.

2. *Rationale: Correct the algorithm for CHECK OPTION enforcement.*

   Add the following bullet to the bullet list of the 10th paragraph:

   — A set of views to be checked.

4.40 Basic security model

4.40.3 Roles

1. *Rationale: Correct and clarify the granting of privileges to roles.*

   Replace the 3rd paragraph with:

   A role is granted to one or more authorization identifiers by executing a <grant role statement>. The granting of a role to an authorization identifier *A* is called a *role authorization* (for *A*).

   If the authorization identifier *A* to which a role R is granted is a user identifier, then *A* is able by means of a <set role statement> to temporarily also acquire the privileges of *R*.

2. *Rationale: Correct the definition.*

   Replace the 2nd paragraph with:

   A privilege *P* is *applicable for* an authorization identifier *A* if its grantee is PUBLIC, or *A*, or, if *A* is a role name, an applicable role for *A*.

4.43 SQL-sessions

4.43.3 SQL-session properties

This Subclause is modified by Subclause 4.16.1, "SQL-session properties", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 4.8.1, "SQL-session properties", in ISO/IEC 9075-14.

1. *Rationale: Clarify the precision of timestamps.*

   Replace the 27th bullet of the 13th paragraph with:

   — A *statement timestamp*: either "not set", or a datetime value of type TIMESTAMP(*n*) WITH TIME ZONE, *n* being the implementation-defined maximum <timestamp precision>, which is the date and time at which every <datetime value function> in the statement is effectively evaluated; initially "not set".

4.43.6 Routine execution context

1. *Rationale: Use the correct term.*

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

   — An indication of whether the active SQL-invoked routine is an SQL-invoked function.

5 Lexical elements

5.2 <token> and <separator>

This Subclause is modified by Subclause 5.1, "<token> and <separator>", in ISO/IEC 9075-4.
This Subclause is modified by Subclause 5.1, "<token> and <separator>", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 5.2, "<token> and <separator>", in ISO/IEC 9075-10.
This Subclause is modified by Subclause 5.1, "<token> and <separator>", in ISO/IEC 9075-13.
This Subclause is modified by Subclause 5.1, "<token> and <separator>", in ISO/IEC 9075-14.
This Subclause is modified by Subclause 7.1, "<token> and <separator>", in ISO/IEC 9075-15.

1. *Rationale: Delete words incorrectly classified as <non-reserved word>s.*

   Remove the following from the list of <non-reserved word>s:

   — DESCRIBE_CATALOG
   — DESCRIBE_NAME
   — DESCRIBE_PROCEDURE_SPECIFIC_CATALOG
   — DESCRIBE_PROCEDURE_SPECIFIC_NAME
   — DESCRIBE_PROCEDURE_SPECIFIC_SCHEMA
   — DESCRIBE_SCHEMA

- FINISH_CATALOG
- FINISH_NAME
- FINISH_PROCEDURE_SPECIFIC_CATALOG
- FINISH_PROCEDURE_SPECIFIC_NAME
- FINISH_PROCEDURE_SPECIFIC_SCHEMA
- FINISH_SCHEMA
- FULFILL_CATALOG
- FULFILL_NAME
- FULFILL_PROCEDURE_SPECIFIC_CATALOG
- FULFILL_PROCEDURE_SPECIFIC_NAME
- FULFILL_PROCEDURE_SPECIFIC_SCHEMA
- FUFILL_SCHEMA
- HAS_PASS_THROUGH_COLUMNS
- HAS_PASS_THRU_COLS
- IS_PRUNABLE
- JSON
- PRIVATE_PARAMETERS
- PRIVATE_PARAMS_S
- RETURNS_ONLY_PASS_THROUGH
- RET_ONLY_PASS_THRU
- START_CATALOG
- START_NAME
- START_PROCEDURE_SPECIFIC_CATALOG
- START_PROCEDURE_SPECIFIC_NAME
- START_PROCEDURE_SPECIFIC_SCHEMA
- START_SCHEMA
- TABLE_SEMANTICS

2. *Rationale: Add words that should have been classified as <non-reserved word>s.*

   Add the following to the list of <non-reserved word>s:

   - MEASURES

— NULL_ORDERING
— OCCURRENCE
— PERMUTE
— PIPE
— PREV
— SEMANTICS
— SORT_DIRECTION
— UNMATCHED

3. *Rationale: Add words that should have been classified as <reserved word>s.*

   Add the following to the list of <reserved word>s:

   — ABSENT
   — JSON

4. *Rationale: Remove duplicate words from the list of <reserved word>s.*

   Delete the second occurrence of the following words from the list of <reserved word>s:

   — CURRENT_PATH
   — CURRENT_ROLE

5. *Rationale: Correct the definition of truncating whitespace.*

   Add the following to the Format:

```
<truncating whitespace> ::=
  !! See the Syntax Rules.
```

6. *Rationale: Correct the definition of truncating whitespace.*

   Insert the following Syntax Rule:

   3.1) <truncating whitespace> is an implementation-defined subset of the characters included in the <white space> but the subset shall always include at least <space>.

   NOTE 127.1 — Since the Unicode definition of White_Space may, over time, acquire additional characters, this definition prevents an existing conforming implementation from being made non-conforming by such a change. However, implementations are expected to align themselves with the most recent Unicode definition in a timely manner.

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.
This Subclause is modified by Subclause 8.1, "<data type>", in ISO/IEC 9075-15.

1. *Rationale: Correctly handle invalid datatime field values.*

   Replace General Rule 6) with:

   6) Table 11, "Valid values for datetime fields", specifies the constraints on the values of the <primary datetime field>s in datetime values.

2. *Rationale: Correctly handle invalid datatime field values.*

   Insert the following General Rules:

   6.1) If the value of any one or more of the <primary datetime field>s in a datetime value does not satisfy the constraints specified in Table 11, "Valid values for datetime fields", then an exception condition is raised: *data exception — datetime field overflow*.

   6.2) If The values of TIMEZONE_HOUR and TIMEZONE_MINUTE have opposite signs (one negative and one positive) or the values of either or both TIMEZONE_HOUR or TIMEZONE_MINUTE do not satisfy the constraints specified in Table 11, "Valid values for datetime fields", then an exception condition is raised: *data exception — invalid time zone displacement value*.

   8.1) If the value of any one or more of the <primary datetime field>s within an interval data type do not satisfy the constraints specified in Table 12, "Valid absolute values for interval fields", then an exception condition is raised: *data exception — datetime field overflow*.

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: Correct the definition of truncating whitespace.*

   Replace General Rule 12) with:

   12) The value specified by CURRENT_PATH is a <schema name list> where <catalog name>s are <delimited identifier>s and the <unqualified schema name>s are <delimited identifier>s. Each <schema name> is separated from the preceding <schema name> by a <comma> with no intervening <separator>s. The schemas referenced in this <schema name list> are those referenced in the SQL-path of the current SQL-session context, in the order in which they appear in that SQL-path.

6.10 <window function>

1. *Rationale: Correct syntactic transformation rules.*

   Replace Syntax Rule 6) a) with:

   6) ...

      a) If <ntile function>, <lead or lag function>, or <rank function type> is specified, then the window ordering clause *WOC* of *WDX* shall be present.

2. *Rationale: Correct syntactic transformation rules.*

   Replace Syntax Rule 6) d) with:

   6) ...

      d) `RANK( ) OVER` *WNS* is equivalent to:

```
( COUNT(*) OVER (WNS1 GROUPS BETWEEN UNBOUNDED PRECEDING
                 AND 1 PRECEDING) + 1 )
```

3. *Rationale: Correct syntactic transformation rules.*

   Replace Syntax Rule 6) e) ii) with:

   6) ...

      e) ...

         ii) `DENSE_RANK( ) OVER` *WNS* is equivalent to the <window function>:

```
COUNT (DISTINCT ROW ( VE1, ..., VEN ) )
OVER (WNS1 GROUPS UNBOUNDED PRECEDING)
```

4. *Rationale: Correct syntactic transformation rules.*

   Replace Syntax Rule 6) h) with:

   6) ...

      h) Let *ANT2* be an approximate numeric type with implementation-defined precision. `CUME_DIST( ) OVER` *WNS* is equivalent to:

```
( CAST ( COUNT(*) OVER
     ( WNS1 GROUPS UNBOUNDED PRECEDING ) AS ANT2 ) /
   COUNT(*) OVER ( WNS1 GROUPS BETWEEN UNBOUNDED PRECEDING
                                   AND UNBOUNDED FOLLOWING ) )
```

5. *Rationale: Correct the classification of the first argument of LISTAGG.*

   Replace Syntax Rule 12) with:

   12) If the window ordering clause or the window framing clause of the window structure descriptor that describes the <window name or specification> is present, then no <aggregate function> simply contained in <window function> shall specify DISTINCT, <listagg set function>, or <ordered set function>.

6. *Rationale: <ntile function> must be computed over the entire window partition.*

   Replace General Rule 1) a) ii) 3) C) with:

   1) ...
      a) ...
         ii) ...
            3) ...
               C) Let *WDX1* be the window structure descriptor that describes the window defined by the <window specification>

```
( WNS RANGE BETWEEN UNBOUNDED PRECEDING
                    AND UNBOUNDED FOLLOWING )
```

               C.1) Let *T* be the collection of rows in the window frame of *R* as defined by *WDX1*, as specified by the General Rules of Subclause 7.15, "<window clause>".

7. *Rationale: Determine the correct sizes of the subdivisions of the window partition used by <ntile function>.*

   Replace General Rule 1) a) ii) 3) E) II) with:

   1) ...
      a) ...
         ii) ...
            3) ...
               E) ...
                  II) Otherwise, for each i, 1 (one) ≤ i ≤ MOD(CT, NT), let NQ_i be CEILING(CAST(CT AS REAL) / NT), and for each i, MOD(CT, NT) < i ≤ NT, let NQ_i be FLOOR(CAST(CT AS REAL) / NT).

8. *Rationale: <lead or lag function> (particularly LEAD) must be computed over the entire window partition.*

   Replace General Rule 1) b) ii) with:

1) ...

b) ...

ii) Let *WDX1* be the window structure descriptor that describes the window defined by the <window specification>

```
( WNS RANGE BETWEEN UNBOUNDED PRECEDING
                    AND UNBOUNDED FOLLOWING )
```

ii.1) Let *T* be the collection of rows in the window frame of *R* as defined by *WDX1*, as specified by the General Rules of Subclause 7.15, "<window clause>".

9. *Rationale: <lead or lag function> (particularly LEAD) must be computed over the entire window partition.*

Replace General Rule 1) b) iii) 1) with:

1) ...

b) ...

iii) ...

1) Case:

A) If *NTREAT* is RESPECT NULLS, then let *TX* be the sequence of values that is the result of applying *VE1* to each row of *T* that follows the current row, ordered according to the window ordering of *WDX1*.

B) Otherwise, let *TX* be the sequence of values that is the result of applying *VE1* to each row of *T* that follows the current row and eliminating null values, ordered according to the window ordering of *WDX1*.

6.12 <case expression>

This Subclause is modified by Subclause 6.4, "<case expression>", in ISO/IEC 9075-14.
This Subclause is modified by Subclause 8.6, "<case expression>", in ISO/IEC 9075-15.

1. *Rationale: Use correct terminology.*

Replace Syntax Rule 2) h) with:

2) ...

h) If <else clause> is specified, then let *CEEC* be that <else clause>; otherwise, let *CEEC* be the zero-length character string.

6.13 <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.
This Subclause is modified by Subclause 8.7, "<cast specification>", in ISO/IEC 9075-15.

1. *Rationale: Correct the definition of truncating whitespace.*

   Replace the lead text of General Rule 8) b) with:

   8) ...

      b) If *SD* is character string, then *SV* is replaced by *SV* with any leading or trailing <truncating whitespace>s removed.

         Case:

2. *Rationale: Correct the definition of truncating whitespace.*

   Replace the lead text of General Rule 9) b) with:

   9) ...

      b) If *SD* is character string, then *SV* is replaced by *SV* with any leading or trailing <truncating whitespace>s removed.

         Case:

3. *Rationale: Correct the definition of truncating whitespace.*

   Replace the lead text of General Rule 10) b) with:

   10) ...

      b) If *SD* is character string, then *SV* is replaced by *SV* with any leading or trailing <truncating whitespace>s removed.

         Case:

4. *Rationale: Correct the definition of truncating whitespace.*

   Replace the lead text of General Rule 11) d) ii) with:

   11) ...

      d) ...

         ii) If the length in characters of *SV* is larger than *LTD*, then *TV* is the first *LTD* characters of *SV*. If any of the remaining characters of *SV* are non-<truncating whitespace> characters, then a completion condition is raised: *warning — string data, right truncation*.

5. *Rationale: Correct the definition of truncating whitespace.*

   Replace the lead text of General Rule 12) d) ii) with:

   12) ...

      d) ...

ii) If the length in characters of *SV* is larger than *MLTD*, then *TV* is the first *MLTD* characters of *SV*. If any of the remaining characters of *SV* are non-<truncating whitespace> characters, then a completion condition is raised: *warning — string data, right truncation*.

6.19 <new specification>

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

1. *Rationale: Use the correct term.*

   Replace Syntax Rule 3) a) i) with:

   3) ...

      a) ...

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

6.31 <string value expression>

1. *Rationale: Correct the definition of truncating whitespace.*

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

   2) ...

      b) ...

         ii) ...

            3) Case:

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

                  Case:

                  I) If *M* is less than or equal to *LOL*, then the result of the <concatenation> is *S* with length *M*.

                  II) If *M* is greater than *LOL* and the right-most *M*–*LOL* characters of *S* are all the <truncating whitespace> character, then the result of the <concatenation> is the first *LOL* characters of *S* with length *LOL*.

                  III) Otherwise, an exception condition is raised: *data exception — string data, right truncation*.

B) If the most specific type of at least one of *S1* and *S2* is variable-length character string, then let *VL* be the implementation-defined maximum length of variable-length character strings.

Case:

I) If *M* is less than or equal to *VL*, then the result of the <concatenation> is *S* with length *M*.

II) If *M* is greater than *VL* and the right-most *M*–*VL* characters of *S* are all the <truncating whitespace> character, then the result of the <concatenation> is the first *VL* characters of *S* with length *VL*.

III) Otherwise, an exception condition is raised: *data exception — string data, right truncation*.

C) If the most specific types of both *S1* and *S2* are fixed-length character string, then the result of the <concatenation> is *S*.

6.32 <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.
This Subclause is modified by Subclause 8.9, "<string value function>", in ISO/IEC 9075-15.

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

   Replace Syntax Rule 19) a) ii) with:

   19) ...

   a) ...

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

   ```
   TRIM ( SRC )
   ```

   is equivalent to:

   ```
   TRIM ( BOTH X'00' FROM SRC )
   ```

2. *Rationale: Correct an error in the rule structuring.*

   Replace General Rule 6) with:

   6) If <regular expression substring function> is specified, then let *C* be the result of the first <character value expression>, let *R* be the result of the second <character value expression>, and let *E* be the result of the <escape character>.

   Case:

   a) If at least one of *C*, *R*, and *E* is the null value, then the result of the <regular expression substring function> is the null value.

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

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

d) Otherwise, let *R1*, *R2*, and *R3* be the substrings of *R*, such that

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

is *True*.

Case:

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

ii) If the predicate

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

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

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

1) Let *S1* be the shortest initial substring of *C* such that there is a substring *S23* of *C* such that the value of the following <search condition> is *True*:

```
'C' = 'S1' || 'S23' AND
'S1' SIMILAR TO 'R1' ESCAPE 'E' AND
'S23' SIMILAR TO '(R2R3)' ESCAPE 'E'
```

2) Let *S3* be the shortest final substring of *S23* such that there is a substring *S2* of *S23* such that the value of the following <search condition> is *True*:

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

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

3. *Rationale: Correct the definition of truncating whitespace.*

Replace the lead text of General Rule 7) g) ii) with:

7) ...

g) ...

ii) If *FRL* is greater than *FRML*, then the result of the <fold> is the first *FRML* characters of *FR* with length *FRML*. If any of the right-most (*FRL* – *FRML*) characters of *FR* are

not <truncating whitespace> characters, then a completion condition is raised: *warning — string data, right truncation*.

4. *Rationale: Use the correct BNF term.*

   Replace General Rule 15) a) with:

   15) ...

      a) Let *S* be the value of the <binary trim source>.

6.33 <JSON value constructor>

1. *Rationale: Resolve a syntatic ambiguity with <JSON name and value>.*

   Replace the BNF for <JSON name and value> with:

```
<JSON name and value> ::=
    <JSON name and value 1>
  | <JSON name and value 2>

<JSON name and value 1> ::=
  [ KEY ] <JSON name> VALUE <JSON value expression>

<JSON name and value 2> ::=
  <JSON name> <colon> <JSON value expression>
```

2. *Rationale: Resolve a syntatic ambiguity with <JSON name and value>.*

   Insert the following Syntax Rule:

   4.1) If <JSON name and value 1> *JNV1* is specified and *JNV1* does not immediately contain KEY, then the <JSON name> immediately contained in *JNV1* shall not be a <routine invocation> that immediately contains a <routine name> that is a <regular identifier> that is equivalent to KEY.

   NOTE 202.1 — This Syntax Rule resolves an ambiguity. In a <JSON name and value> like "KEY(x) VALUE y", KEY could be a key word or the beginning of a routine invocation. This Syntax Rule specifies that it is a key word.

3. *Rationale: Resolve an ambiguity in <JSON value constructor>.*

   Insert a new Syntax Rule:

   6) ...

   c.1) If <JSON array constructor by enumeration> immediately contains exactly 1 (one) <JSON value expression>, then that <JSON value expression> shall not be a <scalar subquery>.

   NOTE 202.1 — This Syntax Rule resolves an ambiguity in which a <JSON array constructor> (e.g., `JSON_ARRAY( ( SELECT a FROM t ) )`) otherwise might be interpreted either as a <JSON array constructor by enumeration> or as a <JSON array constructor by query>. The ambiguity is resolved by adopting the interpretation that such a <JSON array constructor> is a <JSON array constructor by query>.

4. *Rationale: Resolve a syntatic ambiguity with <JSON name and value>.*

   Replace General Rule 2) d) ii) 1) B) with:

   2) ...

      d) ...

         ii) ...

            1) ...

               B) Let JN_i be the <JSON name> simply contained in JNV_i.

5. *Rationale: Resolve a syntatic ambiguity with <JSON name and value>.*

   Replace General Rule 2) d) ii) 1) E) with:

   2) ...

      d) ...

         ii) ...

            1) ...

               E) Let VJE_i be the <JSON value expression> simply contained in JNV_i and let $VJVE_i$ be the value of VJE_i.

6. *Rationale: Resolve a syntatic ambiguity with <JSON name and value>.*

   Replace Conformance Rule 4) with:

   4) Without Feature T814, "Colon in JSON_OBJECT or JSON_OBJECTAGG", conforming SQL language shall not contain a <JSON name and value 2>.

7 Query expressions

7.6 <table reference>

This Subclause is modified by Subclause 7.1, "<table reference>", in ISO/IEC 9075-4.
This Subclause is modified by Subclause 7.1, "<table reference>", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 7.1, "<table reference>", in ISO/IEC 9075-14.
This Subclause is modified by Subclause 9.1, "<table reference>", in ISO/IEC 9075-15.

1. *Rationale: Correct the calculation of the maximum of the cardinalities of arrays.*

   Replace Syntax Rule 7) g) with:

   7) ...

      g) Let $MCARD_1$ be $CARD_1$. Let $MCARD_j$, $2 \leq j \leq NCV$, be

```
CASE
  WHEN CARD_j > MCARD_j-1
    THEN CARD_j
  ELSE MCARD_j-1
END
```

2. *Rationale: Clarify which variant of CHECK OPTION is intended.*

   Replace Syntax Rule 10) b) i) with:

   10) ...

   b) ...

   i) The view descriptor of *TT* shall indicate CASCADED CHECK OPTION.

3. *Rationale: Clarify delta tables.*

   Insert the following General Rule:

   5) ...

   d) All old delta tables and all new delta tables in the most recent statement execution context are destroyed.

7.11 <JSON table>

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

   Replace Syntax Rule 1) f) ix) with:

   1) ...

   f) ...

   ix) If *JTQCD* does not contain <JSON table formatted column error behavior>, then

   Case:

   1) If *JTEB* is ERROR ON ERROR, then the implicit <JSON table formatted column error behavior> of *JTQCD* is ERROR ON ERROR.

   2) Otherwise, the implicit <JSON table formatted column error behavior> of *JTQCD* is NULL ON ERROR.

7.13 <group by clause>

1. *Rationale:*

   Replace SR 17) o) with:

17) ...

o) Let *COR* be an implementation-dependent <correlation name> that is not equivalent to any other <identifier> contained in *QS*. *QS* is equivalent to

```
SELECT QSSQ ZN1 AS DC1, ... , ZNNDC AS DCNDC
FROM ( GU ) AS COR
```

7.15 <window clause>

1. *Rationale: Resolve an ambiguity between a key word and an <unsigned value specification>.*

Insert the following Syntax Rules:

7.1) If *WDEF* specifies <window frame preceding>, then the <unsigned value specification> immediately contained in <window frame preceding> shall not be the <non-reserved word> UNBOUNDED.

7.2) If *WDEF* specifies <window frame following>, then the <unsigned value specification> immediately contained in <window frame following> shall not be the <non-reserved word> UNBOUNDED.

NOTE 6 — The preceding two rules resolve an ambiguity in which UNBOUNDED might be interpreted either as a key word or as an <unsigned value specification>. This ambiguity is resolved by adopting the interpretation as a key word.

7.16 <query specification>

This Subclause is modified by Subclause 7.2, "<query specification>", in ISO/IEC 9075-4.
This Subclause is modified by Subclause 9.2, "<query specification>", in ISO/IEC 9075-15.

1. *Rationale: Correct over-restriction of updatability.*

Replace Syntax Rule 24) a) with:

24) ...

a) *QS* does not immediately contain a <set quantifier> that specifies DISTINCT.

7.17 <query expression>

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

1. *Rationale: Avoid relying on a missing definition of possibly non-deterministic for <query primary>.*

Replace Syntax Rule 24) c) with:

24) ...

c) The <query expression> contains a <query specification>, <table value constructor> or <explicit table> that is possibly non-deterministic.

7.18 <search or cycle clause>

1. *Rationale: Restructure invalid case construction.*

   Replace Syntax Rule 2) with:

   2) Let *WQN* be the <query name>, *WCL* the <with column list>, and *WQE* the <query expression> simply contained in *WLEC*. Let *WQEB* be the <query expression body> immediately contained in *WQE*. Let *OP* be the set operator immediately contained in *WQEB*. Let *TLO* be the <query expression body> that constitutes the first operand of *OP* and let *TRO* be the <query specification> that (necessarily) constitutes the second operand of *OP*.

   3) Let *TROSL* be the <select list> immediately contained in *TRO*. Let *WQNTR* be the <table reference> simply contained in the <from clause> immediately contained in the <table expression> *TROTE* immediately contained in *TRO* such that *WQNTR* immediately contains *WQN*.

      Case:

      a) If *WQNTR* simply contains a <correlation name>, then let *WQNCRN* be that <correlation name>.

      b) Otherwise, let *WQNCRN* be *WQN*.

   4) If *WLEC* simply contains a <search clause> *SC*, then let *SQC* be the <sequence column> and *SO* be the <recursive search order> immediately contained in *SC*. Let *CNL* be the <column name list> immediately contained in *SO*.

      a) *WCL* shall not contain a <column name> that is equivalent to *SQC*.

      b) Every <column name> of *CNL* shall be equivalent to some <column name> contained in *WCL*. No <column name> shall be contained more than once in *CNL*.

      c) Case:

         i) If *SO* immediately contains DEPTH, then let *SCEX1* be:

            ```
            WQNCRN.SQC
            ```

            let *SCEX2* be:

            ```
            SQC || ARRAY [ROW(CNL)]
            ```

            and let *SCIN* be:

            ```
            ARRAY [ROW(CNL)]
            ```

         ii) If *SO* immediately contains BREADTH, then let *SCEX1* be:

            ```
            ( SELECT OC.*
              FROM ( VALUES (WQNCRN.SQC) )
              OC(LEVEL, CNL) )
            ```

            let SCEX2 be:

            ```
            ROW(SQC.LEVEL + 1, CNL)
            ```

and let *SCIN* be:

```
ROW(0, CNL)
```

5) If *WLEC* simply contains a <cycle clause> *CC*, then let *CCL* be the <cycle column list>, let *CMC* be the <cycle mark column>, let *CMV* be the <cycle mark value>, let *CMD* be the <non-cycle mark value>, and let *CPA* be the <path column> immediately contained in *CC*.

   a) Every <column name> of *CCL* shall be equivalent to some <column name> contained in *WCL*. No <column name> shall be contained more than once in *CCL*.

   b) *CMC* and *CPA* shall not be equivalent to each other and not equivalent to any <column name> of *WCL*.

   c) The declared type of *CMV* and *CMD* shall be character string of length 1 (one). *CMV* and *CMD* shall be literals and *CMV* shall not be equal to *CMD*.

   d) Let *CCEX1* be:

```
WQNCRN.CMC, WQNCRN.CPA
```

   Let *CCEX2* be:

```
CASE WHEN ROW(CCL) IN
  (SELECT P.* FROM UNNEST(CPA) P)

    THEN CMV ELSE CMD END,

CPA || ARRAY [ROW(CCL)]
```

   Let *CCIN* be:

```
CMD, ARRAY [ROW(CCL)]
```

   Let *NCCON1* be:

```
CMC <> CMV
```

6) Case:

   a) If *WLEC* simply contains a <search clause> and does not simply contain a <cycle clause>, then let *EWCL* be:

```
WCL, SQC
```

   Let *ETLOSL* be:

```
WCL, SCIN
```

   Let *ETROSL* be:

```
WCL, SCEX2
```

   Let *ETROSL1* be:

```
TROSL, SCEX1
```

Let *NCCON* be:

```
TRUE
```

b) If *WLEC* simply contains a <cycle clause> and does not simply contain a <search clause>, then let *EWCL* be:

```
WCL, CMC, CPA
```

Let *ETLOSL* be:

```
WCL, CCIN
```

Let *ETROSL* be:

```
WCL, CCEX2
```

Let *ETROSL1* be:

```
TROSL, CCEX1
```

Let *NCCON* be:

```
NCCON1
```

c) If *WLEC* simply contains both a <search clause> and a <cycle clause> *CC*, then:

i) The <column name>s *SQC*, *CMC*, and *CPA* shall not be equivalent to each other.

ii) Let *EWCL* be:

```
WCL, SQC, CMC, CPA
```

Let *ETLOSL* be:

```
WCL, SCIN, CCIN
```

Let *ETROSL* be:

```
WCL, SCEX2, CCEX2
```

Let *ETROSL1* be:

```
TROSL, SCEX1, CCEX1
```

iii) Let *NCCON* be:

```
NCCON1
```

7) *WLEC* is equivalent to the expanded <with list element>:

```
WQN(EWCL) AS
  ( SELECT ETLOSL FROM (TLO) TLOCRN(WCL)
    OP
    SELECT ETROSL
    FROM (SELECT ETROSL1 TROTE) TROCRN(EWCL)
```

```
    WHERE NCCON
)
```

8 Predicates

8.1 <predicate>

This Subclause is modified by Subclause 8.1, "<predicate>", in ISO/IEC 9075-14.

1. *Rationale: Provide the missing definition of possibly non-deterministic for <predicate>.*

   Insert the following Syntax Rule:

   1) A <predicate>*P* is *possibly non-deterministic* if any of the following are true:

      a) *P* contains a <value expression>, <nonparenthesized value expression primary> or <query expression> that is possibly non-deterministic.

      b) *P* contains a <comparison predicate> , <overlaps predicate>, or <distinct predicate> simply containing <row value predicand>s *RVP1* and *RVP2* such that the declared types of *RVP1* and *RVP2* have corresponding constituents such that one constituent is datetime with time zone and the other is datetime without time zone.

      c) *P* contains a <quantified comparison predicate> or a <match predicate> simply containing a <row value predicand> *RVP* and a <table subquery> *TS* such that the declared types of *RVP* and *TS* have corresponding constituents such that one constituent is datetime with time zone and the other is datetime without time zone.

      d) *P* contains a <member predicate> simply containing a <row value predicand> *RVP* and a <multiset value expression> *MVP* such that the declared type of the only field *F* of *RVP* and the element type of *MVP* have corresponding constituents such that one constituent is datetime with time zone and the other is datetime without time zone.

      e) *P* contains a <submultiset predicate> simply containing a <row value predicand> *RVP* and a <multiset value expression> *MVP* such that the declared type of the only field *F* of *RVP* and the declared type of *MVP* have corresponding constituents such that one constituent is datetime with time zone and the other is datetime without time zone.

      f) *P* contains a <multiset value expression> that specifies or implies MULTISET UNION, MULTISET EXCEPT, or MULTISET INTERSECT such that the element types of the operands have corresponding constituents such that one constituent is datetime with time zone and the other is datetime without time zone.

8.11 <unique predicate>

1. *Rationale: Clarify uniqueness in the presence of nulls.*

   Replace General Rule 2) with:

   2) Case:

      a) If there are zero or one rows in *T*, the result of the <unique predicate> is *True*.

b) If for every two rows *R1* and *R2* in *T*, the implementation uniqueness rule holds, the result of the <unique predicate> is *True*.

c) Otherwise, the result of the <unique predicate> is *False*.

9 Additional common rules

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.
This Subclause is modified by Subclause 11.2, "Store assignment", in ISO/IEC 9075-15.

1. *Rationale: Correct the definition of truncating whitespace.*

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

   3) ...

      b) ...

         ii) ...

            2) If one or more of the rightmost *M*–*L* characters of *V* are not <truncating whitespace>s, then an exception condition is raised: *data exception — string data, right truncation*.

2. *Rationale: Correct the definition of truncating whitespace.*

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

   3) ...

      b) ...

         v) ...

            2) If one or more of the rightmost *M*–*L* characters of *V* are not <truncating whitespace>s, then an exception condition is raised: *data exception — string data, right truncation*.

3. *Rationale: Correct the definition of truncating whitespace.*

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

   3) ...

      b) ...

         vii) ...

2) If one or more of the rightmost *M*–*L* characters of *V* are not <truncating whitespace>s, then an exception condition is raised: *data exception — string data, right truncation*.

9.18 Compilation of an invocation of a polymorphic table function

1. *Rationale: Use correct symbol.*

   Replace General Rules 2) b) iii) 2) B) and 2) b) iii) 2) C) with:

   2) ...

      b) ...

         iii) ...

            2) ...

               B) The COUNT and TOP_LEVEL_COUNT components of the header of *ODA* are set to *NOC*.

               C) The number and maximum number of SQL item descriptors of *ODA* are set to *NOC*.

9.38 SQL/JSON path language: lexical elements

1. *Rationale: Use the correct tokens for the "not equals" comparison operator in the SQL/JSON path language.*

   Replace the BNF for <SQL/JSON special symbol> with:

```
<SQL/JSON special symbol> ::=
    <alternative not equals operator>
  | <asterisk>
  | <at sign>
  | <comma>
  | <dollar sign>
  | <double ampersand>
  | <double equals>
  | <double vertical bar>
  | <exclamation mark>
  | <greater than operator>
  | <greater than or equals operator>
  | <left bracket>
  | <left paren>
  | <less than operator>
  | <less than or equals operator>
  | <minus sign>
  | <not equals operator>
  | <percent>
  | <period>
  | <plus sign>
  | <question mark>
  | <right bracket>
  | <right paren>
  | <solidus>
```

```
<alternative not equals operator> ::=
  !=
```

9.39 SQL/JSON path language: syntax and semantics

1. *Rationale: Use the correct tokens for the "not equals" comparison operator in the SQL/JSON path language.*

   Replace the BNF for <JSON comp op> with:

```
<JSON comp op> ::=
    <double equals>
  | <JSON path not equals operator>
  | <less than operator>
  | <greater than operator>
  | <less than or equals operator>
  | <greater than or equals operator>
```

NOTE 451 — Equality operators have the same precedence as inequality comparison operators, unlike [ECMAScript].

```
<JSON path not equals operator> ::=
    <not equals operator>
  | <alternative not equals operator>
```

2. *Rationale: Use the correct symbols.*

   Replace General Rule 11) g) ii) 3) F) II) 1) with:

   11) ...

   g) ...

   ii) ...

   3) ...

   F) ...

   II) ...

   1) Let $TJSFROM_i$ be the result of implementation-defined truncation or rounding of $RJSFROM_i$ and let $TJSTO_i$ be the result of implementation-defined truncation or rounding of $RJSTO_i$.

3. *Rationale: Use the correct symbols.*

   Replace General Rule 11) g) ii) 3) F) III) 1) with:

   11) ...

   g) ...

   ii) ...

   3) ...

F) ...

III) ...

1) Let *RJSU* be the list of integers formed by concatenating RJS_i, 1 (one) ≤ i ≤ s, in that order.

4. *Rationale: Use the correct tokens for the "not equals" comparison operator in the SQL/JSON path language.*

Replace General Rule 12) c) iii) 2) D) I) with:

12) ...

c) ...

iii) ...

2) ...

D) ...

I) If *JCO* is <double equals> or <JSON path not equals operator>, and AI_i and BI_j are not equality-comparable, then let *ERR* be *True*.

5. *Rationale: Use the correct tokens for the "not equals" comparison operator in the SQL/JSON path language.*

Replace the lead text of General Rule 12) c) iii) 2) D) IV) with:

12) ...

c) ...

iii) ...

2) ...

D) ...

IV) If *JCO* is <JSON path not equals operator> then

Case:

10 Additional common elements

10.4 <routine invocation>

This Subclause is modified by Subclause 9.1, "<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 9.4, "<routine invocation>", in ISO/IEC 9075-13.
This Subclause is modified by Subclause 11.1, "<routine invocation>", in ISO/IEC 9075-14.
This Subclause is modified by Subclause 12.1, "<routine invocation>", in ISO/IEC 9075-15.

1. *Rationale: Supply the correct parameters.*

   Replace Syntax Rule 9) h) iii) 5) with:

   9) ...

      h) ...

         iii) ...

            5) 04 If XA_i is an <SQL parameter reference>, a <column reference>, or a <target array element specification>, then the Syntax Rules of Subclause 9.2, "Store assignment", are applied with XA_i as *TARGET* and P_i as *VALUE*.

               NOTE 480 — The <column reference> can only be a new transition variable column reference.

2. *Rationale: Supply the correct arguments.*

   Replace Syntax Rule 9) h) v) with:

   9) ...

      h) ...

         v) For each SQL parameter P_i, 1 (one) ≤ i ≤ *SRNP*, that is an input SQL parameter or both an input SQL parameter and an output SQL parameter, and each XA_i that is not a <contextually typed value specification>, the Syntax Rules of Subclause 9.2, "Store assignment", are applied with P_i as *TARGET* and XA_i as *VALUE*.

3. *Rationale: Correct handling of invalid locators.*

   Replace General Rule 8) i) i) 4) B) with:

   8) ...

      i) ...

         i) ...

            4) ...

               B) Case:

                  I) If the routine descriptor of *R* indicates that the return value is a locator, then

                  Case:

                  1) If *ERDI* is an invalid locator, then an exception condition is raised: *locator exception — invalid specification* and evaluation of GR 8) is terminated immediately and evaluation continues with GR 11).

2) If *RT* is a binary large object type, then let *RDI* be the binary large object value corresponding to *ERDI*.

3) If *RT* is a character large object type, then let *RDI* be the large object character string corresponding to *ERDI*.

4) If *RT* is an array type, then let *RDI* be the array value corresponding to *ERDI*.

5) If *RT* is a multiset type, then let *RDI* be the multiset value corresponding to *ERDI*.

6) If *RT* is a user-defined type, then let *RDI* be the user-defined type value corresponding to *ERDI*.

II) Otherwise, if *R* specifies <result cast>, then let *CRT* be the <data type> specified in <result cast>; otherwise, let *CRT* be the <returns data type> of *R*.

Case:

1) If *R* specifies <result cast> and the routine descriptor of *R* indicates that the <result cast> has a locator indication, then

Case:

a) If *CRT* is a binary large object type, then let *RDI* be the binary large object value corresponding to *ERDI*.

b) If *CRT* is a character large object type, then let *RDI* be the large object character string corresponding to *ERDI*.

c) If *CRT* is an array type, then let *RDI* be the array value corresponding to *ERDI*.

d) If *CRT* is a multiset type, then let *RDI* be the multiset value corresponding to *ERDI*.

e) If *CRT* is a user-defined type, then let *RDI* be the user-defined type value corresponding to *ERDI*.

2) Otherwise,

Case:

a) If *CRT* is a user-defined type, then:

i) Let *TSF* be the SQL-invoked routine identified by the specific name of the to-sql function associated with the result of *R*.

ii) Case:

1) If *TSF* is an SQL-invoked method, then:

A) If *R* is a type-preserving function, then let *MAT* be the most specific type of the value of the argument substituted

for the result SQL parameter of *R*; otherwise, let *MAT* be *CRT*.

B) The General Rules of Subclause 10.4, "<routine invocation>", are applied with a static SQL argument list whose first element is the value returned by the invocation of `MAT()` and whose second element is *ERDI* as *STATIC SQL ARG LIST* and *TSF* as *SUBJECT ROUTINE*.

2) The General Rules of Subclause 10.4, "<routine invocation>", are applied with *ERDI* as *STATIC SQL ARG LIST* and *TSF* as *SUBJECT ROUTINE*.

iii) Let *RDI* be the result of invocation of *TSF*.

b) 14 Otherwise, let *RDI* be *ERDI*.

4. *Rationale: Correct handling of invalid locators.*

Replace General Rule 8) j) iii) with:

8) ...

j) ...

iii) Otherwise:

NOTE 9 — In this case, either *R* specifies PARAMETER STYLE SQL and entry (*PN*+*i*) in *ESPL* (that is, the *i*-th SQL indicator argument corresponding to CPV_i) is not negative and a value was assigned to the *i*-th entry in *ESPL*, or else *R* specifies PARAMETER STYLE GENERAL.

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

2) Case:

A) If P_i is a locator parameter, then

Case:

I) If EV_i is an invalid locator, then an exception condition is raised: *locator exception — invalid specification* and evaluation of GR 8) is terminated immediately and evaluation continues with GR 11).

II) If T_i is a binary large object type, then CPV_i is set to the binary large object value corresponding to EV_i.

III) If T_i is a character large object type, then CPV_i is set to the large object character string corresponding to EV_i.

IV) If T_i is an array type, then CPV_i is set to the array value corresponding to EV_i.

V) If T_i is a multiset type, then CPV_i is set to the multiset value corresponding to EV_i.

VI) If T_i is a user-defined type, then CPV_i is set to the user-defined type value corresponding to EV_i.

B) If T_i is a user-defined type, then:

I) Let TSF_i be the SQL-invoked function identified by the specific name of the to-sql function associated with P_i in the routine descriptor of *R*.

II) Case:

1) If TSF_i is an SQL-invoked method, then the General Rules of Subclause 10.4, "<routine invocation>", are applied with a static SQL argument list whose first element is the value returned by the invocation of TSF_i ( ) and whose second element is EV_i as *STATIC SQL ARG LIST* and TSF_i as *SUBJECT ROUTINE*.

2) Otherwise, the General Rules of Subclause 10.4, "<routine invocation>", are applied with EV_i as *STATIC SQL ARG LIST* and TSF_i as *SUBJECT ROUTINE*.

III) CPV_i is set to the result of an invocation of TSF_i.

C) 14 Otherwise, CPV_i is set to EV_i.

10.6 <specific routine designator>

1. *Rationale: Use the correct term.*

Replace Syntax Rules 3) c) i) 3) B) and 3) c) i) 3) C) with:

3) ...

c) ...

i) ...

3) ...

B) If CONSTRUCTOR is specified, then there shall be exactly one SQL-invoked constructor method of the user-defined type identified by <schema-resolved user-defined type name> whose <method name> is *METH* such that, for all *i*, 1 (one) ≤ *i* ≤ the number of arguments, when the when the Syntax Rules of Subclause 9.24, "Data type identity", are applied with declared type of its *i*-th SQL parameter in the unaugmented SQL parameter declaration list as *TYPE1* and the *i*-th <data type> in the <data type list> of *MN* as *TYPE2*, those Syntax Rules are satisfied. The <specific routine designator> identifies that SQL-invoked constructor method.

C) Otherwise, there shall be exactly one instance SQL-invoked method of the user-defined type identified by <schema-resolved user-defined type name> whose <method name> is *METH* such that, for all *i*, 1 (one) ≤ *i* ≤ the number of arguments, when the when the Syntax Rules of Subclause 9.24, "Data type identity", are applied with declared type of its *i*-th SQL parameter in the unaugmented SQL parameter declaration list as *TYPE1* and the *i*-th <data type> in the <data type list> of *MN* as *TYPE2*, those Syntax Rules are satisfied. The <specific routine designator> identifies that instance SQL-invoked method.

10.9 <aggregate function>

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

1. *Rationale: Correct the classification of the first argument of LISTAGG.*

   Replace the BNF for <aggregate function> and <ordered set function> with:

```
<aggregate function> ::=
    COUNT <left paren> <asterisk> <right paren> [ <filter clause> ]
  | <general set function> [ <filter clause> ]
  | <binary set function> [ <filter clause> ]
  | <ordered set function> [ <filter clause> ]
  | <array aggregate function> [ <filter clause> ]
  | <row pattern count function> [ <filter clause> ]
  | <JSON aggregate function> [ <filter clause> ]
  | <listagg set function> [ <filter clause> ]

<ordered set function> ::=
    <hypothetical set function>
  | <inverse distribution function>
```

2. *Rationale: Correct the determination of the declared type of the result when the result is an interval type.*

   Replace Syntax Rule 7) g) vi) with:

   7) ...

      g) ...

         vi) If *DT* is interval, then the declared type of the result is interval with the same fields as *DT* and an implementation-defined <interval leading field precision> not less than that of *DT*.

3. *Rationale: Correct the classification of the first argument of LISTAGG.*

   Replace Syntax Rule 14) with:

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

4. *Rationale: Correct the ordering of the General Rules.*

   Replace General Rules 11) a), b) and c) with:

   11) ...

      a) Let *T2* be *T1* ordered by the <sort specification list> as specified in the General Rules of Subclause 10.10, "<sort specification list>".

      b) Let *TX* be the single-column table that is the result of applying the <character value expression> to each row of *T2* and eliminating null values. If one or more null values are eliminated, then a completion condition is raised: *warning — null value eliminated in set function*.

      c) Case:

         i) If DISTINCT is specified, then let *TXA* be the result of eliminating redundant duplicate values from *TX*, using the comparison rules specified in Subclause 8.2, "<comparison predicate>", to identify the redundant duplicate values.

         ii) Otherwise, let *TXA* be *TX*.

10.11 <JSON aggregate function>

1. *Rationale: Resolve a syntatic ambiguity with <JSON name and value>.*

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

   4) ...

      b) ...

         ii) ...

            1) Let *JVE* be the <JSON value expression> simply contained in <JSON name and value>.

2. *Rationale: Resolve a syntatic ambiguity with <JSON name and value>.*

   Replace General Rule 4) b) ii) 2) A) with:

   4) ...

      b) ...

         ii) ...

            2) ...

               A) Let VJN_i be the value of <JSON name> simply contained in <JSON name and value> evaluated for R_i.

3. *Rationale: Resolve a syntatic ambiguity with <JSON name and value>.*

   Replace Conformance Rule 4) with:

   4) Without Feature T814, "Colon in JSON_OBJECT or JSON_OBJECTAGG", conforming SQL language shall not contain a <JSON name and value 2>.

11 Schema definition and manipulation

11.4 <column definition>

This Subclause is modified by Subclause 10.4, "<column definition>", in ISO/IEC 9075-4.
This Subclause is modified by Subclause 12.1, "<column definition>", in ISO/IEC 9075-14.
This Subclause is modified by Subclause 13.1, "<column definition>", in ISO/IEC 9075-15.

1. *Rationale: Avoid relying on a missing definition of possibly non-deterministic for <generation expression>.*

   Replace Syntax Rule 10) d) with:

   10) ...

      d) *GE* shall not contain a possibly non-deterministic <value expression>.

2. *Rationale: Use the correct degree of containment.*

   Replace Syntax Rule 10) e) with:

   10) ...

      e) *GE* shall not generally contain a <routine invocation> whose subject routine possibly reads SQL-data.

11.28 <drop table period definition>

1. *Rationale: Use correct terminology.*

   Replace General Rule 1) b) with:

   1) ...

      b) Let *ECN* be the <column name> of the system-time period end column of *T*. The following <alter table statement> is executed without further Access Rule checking:

```
ALTER TABLE TN DROP COLUMN ECN CASCADE
```

11.32 <view definition>

This Subclause is modified by Subclause 10.11, "<view definition>", in ISO/IEC 9075-4.
This Subclause is modified by Subclause 10.3, "<view definition>", in ISO/IEC 9075-13.
This Subclause is modified by Subclause 12.4, "<view definition>", in ISO/IEC 9075-14.

This Subclause is modified by Subclause 13.2, "<view definition>", in ISO/IEC 9075-15.

1. *Rationale: Clarify which variant of CHECK OPTION is intended.*

   Replace Syntax Rule 16) with:

   16) If CASCADED CHECK OPTION is specified or implied, then the viewed table shall be effectively updatable.

2. *Rationale: LOCAL CHECK OPTION cannot be enforced except on simply updatable views.*

   Insert a new Syntax Rule:

   16.1) If LOCAL CHECK OPTION is specified, then the viewed table shall be simply updatable.

3. *Rationale: A generally underlying table of a view that specifies CHECK OPTION may not have an INSTEAD OF trigger.*

   Insert a new Syntax Rule:

   16.2) If CHECK OPTION is specified, then no generally underlying table of *QE* shall be a view whose view descriptor includes an indication that the view is trigger updatable, trigger insertable-into or trigger deletable.

4. *Rationale: Clarify a rule by removing redundant specification.*

   Replace Syntax Rule 18) with:

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

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

5. *Rationale: Differentiate the restrictions for CASCADED CHECK OPTION and LOCAL CHECK OPTION in view hierarchies.*

   Replace Syntax Rule 21) t) iv) with:

   21) ...

   t) ...

   iv) If the view descriptor of *SUPERT* or any supertable of *SUPERT* includes an indication that CASCADED CHECK OPTION was specified, then *QS* shall not be possibly non-deterministic.

   iv.1) If the view descriptor of *SUPERT* or any supertable of *SUPERT* includes an indication that LOCAL CHECK OPTION was specified, then *QS* shall not generally contain a <query expression> *QE2* that is possibly non-deterministic unless *QE2* is generally

contained in the hierarchical <query expression> of a viewed table that is a leaf underlying table specification of *QS*.

6. *Rationale: Differentiate the restrictions for CASCADED CHECK OPTION and LOCAL CHECK OPTION; both rules must include all supertables.*

   Replace Syntax Rule 21) t) v) with:

   21) ...

   t) ...

   v) If *SUPERT* or any supertable of *SUPERT* is referenced in any check constraint descriptor, assertion descriptor, or the original <query expression> of any view having CASCADED CHECK OPTION, then *QS* shall not be possibly non-deterministic.

   v.1) If *SUPERT* or any supertable of *SUPERT* is referenced, except as a leaf underlying table, in the original <query expression> of any view having LOCAL CHECK OPTION, then *QS* shall not be possibly non-deterministic.

7. *Rationale: LOCAL CHECK OPTION cannot be enforced except on simply updatable views.*

   Insert the following Syntax Rule:

   21) ...

   t) ...

   v.2) If the view descriptor of *SUPERT* or any supertable of *SUPERT* includes an indication that LOCAL CHECK OPTION was specified, then *QS* shall be simply updatable.

8. *Rationale: Correct undefined term.*

   Replace General Rule 8) b) with:

   8) ...

   b) For every method *M* of the structured type identified by <path-resolved user-defined type name>, a privilege descriptor is created that defines the SELECT privilege on the table/method pair consisting of table *V* and method *M* to *A*. That privilege is grantable.

11.49 <trigger definition>

This Subclause is modified by Subclause 10.19, "<trigger definition>", in ISO/IEC 9075-4.

1. *Rationale: Use the correct degree of containment.*

   Replace Syntax Rule 11) with:

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

11.56 <add original method specification>

1. *Rationale: Use correct term.*

   Replace Syntax Rule 15) a) i) with:

   15) ...

   a) ...

   i) *MPDL* and the unaugmented SQL parameter declaration list of *CMS* have the same number *N* of SQL parameters.

2. *Rationale: Use correct term.*

   Replace Syntax Rule 15) b) ii) with:

   15) ...

   b) ...

   ii) *MPDL* and the unaugmented SQL parameter declaration list of *CMS* have the same number *N* of SQL parameters.

11.58 <drop method specification>

1. *Rationale: Use correct term.*

   Replace Syntax Rule 2) c) with:

   2) ...

   c) Let *PDL* be the augmented SQL parameter declaration list included in *DOOMS*.

2. *Rationale: Use correct term.*

   Replace Syntax Rule 3) c) with:

   3) ...

   c) If *SMSD* immediately contains a <data type list> *DTL*, then

   Case:

   i) If STATIC is specified, then the descriptor of *D* shall include exactly one method specification descriptor *DOOMS* that includes:

   1) An indication that the method specification is STATIC.

   2) An indication that the method specification is original.

   3) An augmented SQL parameter declaration list *PDL* such that the declared type of its *i*-th parameter, for all *i*, is identical to the *i*-th declared type in *DTL*.

ii) If CONSTRUCTOR is specified, then the descriptor of *D* shall include exactly one method specification descriptor *DOOMS* that includes:

1) An indication that the method specification is CONSTRUCTOR.

2) An indication that the method specification is original.

3) An augmented SQL parameter declaration list *PDL* such that the declared type of its *i*-th parameter, for all *i* > 1 (one), is identical to the (*i*–1)-th declared type in *DTL* and the declared type of the first parameter of *PDL* is identical to *DN*.

iii) Otherwise, the descriptor of *D* shall include exactly one method specification descriptor *DOOMS* for which:

1) If *DOOMS* includes an indication that the method specification is original, then *DOOMS* shall not include an indication that the method specification is either STATIC or CONSTRUCTOR.

2) *DOOMS* includes an augmented SQL parameter declaration list *PDL* such that the declared type of its *i*-th parameter, for all *i* > 1 (one), is identical to the (*i*–1)-th declared type in *DTL* and the declared type of the first parameter of *PDL* is identical to *DN*.

11.60 <SQL-invoked routine>

This Subclause is modified by Subclause 10.24, "<SQL-invoked routine>", in ISO/IEC 9075-4.
This Subclause is modified by Subclause 11.11, "<SQL-invoked routine>", in ISO/IEC 9075-9.
This Subclause is modified by Subclause 10.8, "<SQL-invoked routine>", in ISO/IEC 9075-13.
This Subclause is modified by Subclause 12.8, "<SQL-invoked routine>", in ISO/IEC 9075-14.
This Subclause is modified by Subclause 13.4, "<SQL-invoked routine>", in ISO/IEC 9075-15.

1. *Rationale: <locator indication> is specified in the parameter lists of PTF component procedures but not in the parameter list of the polymorphic table function itself. The default value of a private parameter must be assignable to the declared type of the default parameter.*

Replace Syntax Rule 10) e) with:

10) ...

e) If <PTF private parameters> *PTFPRIV* is specified, then for each <SQL parameter declaration> *PTFPRIVPD* contained in *PTFPRIV*:

i) *PTFPRIVPD* shall not specify <parameter mode>, <locator indication>, RESULT, or a <parameter type> that is ROW, <generic table parameter type> or <descriptor parameter type>.

ii) If *PTFPRIVPD* contains a <parameter default> *PDEF*, then let *PV* be a variable whose declared type is specified by the <data type> contained in *PTFPRIVPD*. Syntax Rules of Subclause 9.2, "Store assignment", are applied with *PV* as *TARGET* and *PDEF* as *VALUE*.

iii) The SQL parameter declared by *PTFPRIVPD* is a *private parameter* of *R*.

2. *Rationale: <locator indication> is specified in the parameter lists of PTF component procedures but not in the parameter list of the polymorphic table function itself.*

   Insert a new Syntax Rule:

   10) ...

      g) ...

         i.0) <locator indication> shall not be specified.

3. *Rationale: <locator indication> is specified in the parameter lists of PTF component procedures but not in the parameter list of the polymorphic table function itself.*

   Replace the lead text of Syntax Rule 10) h) with:

   10) ...

      h) In the following rules, a parameter list *ACTUALPL* is said to *conform* to a parameter list *TEMPLATEPL* if they have the same number of parameters, and, for each parameter APL_i of *ACTUALPL*, the <parameter type> of APL_i is the same as the <parameter type>, disregarding the presence or absence of <locator indication>, of the parameter TPL_i at the same ordinal position in *TEMPLATEPL*.

4. *Rationale: NPL must be defined for polymorphic table functions.*

   Insert a new Syntax Rule:

   10) ...

      m.1) Let *NPL* be the <SQL parameter declaration list> contained in the <function specification>.

12 Access Control

12.2 <grant privilege statement>

This Subclause is modified by Subclause 11.1, "<grant privilege statement>", in ISO/IEC 9075-13.

1. *Rationale: Grantor determination must be done in the Syntax Rules.*

   Insert the following Syntax Rules:

   3.1) Case:

      a) If GRANTED BY is omitted, then let *G* be OMITTED.

      b) Otherwise, let *G* be <grantor>.

   3.2) Let *A* be the result of applying the Syntax Rules of Subclause 12.8, "Grantor determination", with *G* as *GRANTOR*. *A* is the *grantor* of the <grant privilege statement>.

2. *Rationale: Supply missing Access Rules.*

   Insert the following Access Rules:

   1) The applicable privileges for *A* shall include a privilege identifying *O*, or, if *O* is a table, a column of *O* or a table/method pair whose table is *O*.

   2) If <privileges> contains a <privilege column list> *PCL*, then for every <column name> *CN* contained in *PCL*, the applicable privileges shall include a column privilege whose object is the column identified by *CN*.

   3) If <privileges> contains a <privilege method list> *PML*, then for every <specific routine designator> *SRD* contained in *PML*, the applicable privileges shall include a privilege whose object is the method identified by *SRD*.

3. *Rationale: Grantor determination is now done in the Syntax Rules; privilege checking is now done in the Access Rules.*

   Delete General Rules 1), 2) and 3).

12.3 <privileges>

This Subclause is modified by Subclause 11.2, "<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 11.2, "<privileges>", in ISO/IEC 9075-13.

1. *Rationale: Replace the use of "may" and clarify.*

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

   3) If *T* is a temporary table, then <object privileges> shall specify ALL PRIVILEGES.

   4) If <action> *AC* is specified, then

      Case:

      a) 09 13 If *ON* specifies a <domain name>, <collation name>, <character set name>, <transliteration name>, <schema-resolved user-defined type name>, or <sequence generator name>, then *AC* shall specify USAGE.

      b) If *T* is a base table or a viewed table, then *AC* shall specify SELECT, DELETE, INSERT, UPDATE, REFERENCES, or TRIGGER.

      c) If *ON* specifies a <schema-resolved user-defined type name> that identifies a structured type or specifies a <table name>, then *AC* shall specify UNDER.

      d) 04 If the object identified by *ON* is an SQL-invoked routine, then *AC* shall specify EXECUTE.

2. *Rationale: Clarify the grantor in the case of ALL PRIVILEGES.*

   Replace Syntax Rule 10) a) with the following:

   10) ...

a) If ALL PRIVILEGES is specified, then let *A* be the grantor of the <grant privilege statement> or the <revoke statement> that simply contains <privileges>. ALL PRIVILEGES specifies the set union of the following sets of privilege descriptor kernels:

i) The set of all privilege descriptor kernels whose object is *O* and whose action is one of the actions on *O* for which *A* has grantable privilege descriptors.

ii) If *O* is a table, then all privilege descriptor kernels whose object *O2* is a column of *O* and whose action is one of the actions on *O2* for which *A* has grantable column privilege descriptors.

iii) If *O* is a table, then all privilege descriptor kernels whose object *O3* is a table/method pair in which the table is *O* and whose action is one of the actions on *O3* for which *A* has grantable column privilege descriptors.

3. *Rationale: Delete redundant and misleading rules.*

Delete General Rules 2), 3), 4), 5), 6) and 7).

12.4 <role definition>

1. *Rationale: Grantor determination must be done in the Syntax Rules.*

Add the following Syntax Rules:

2) Case:

a) If WITH ADMIN is omitted, then let *G* be OMITTED.

b) Otherwise, let *G* be <grantor>.

3) Let *A* be the result of applying the Syntax Rules of Subclause 12.8, "Grantor determination", with *G* as GRANTOR.

2. *Rationale: Grantor determination must be done in the Syntax Rules.*

Delete General Rules 2) and 3).

12.5 <grant role statement>

1. *Rationale: Grantor determination must be done in the Syntax Rules.*

Add the following Syntax Rules:

2) Case:

a) If <grantor> is omitted, then let *G* be OMITTED.

b) Otherwise, let *G* be <grantor>.

3) Let *A* be the result of applying the Syntax Rules of Subclause 12.8, "Grantor determination", with *G* as GRANTOR.

2. *Rationale: Grantor determination must be done in the Syntax Rules.*

   Delete General Rules 1) and 2).

12.6 <drop role statement>

1. *Rationale: <drop role statement> needs <drop behavior>.*

   Replace the Format with:

```
<drop role statement> ::=
  DROP ROLE <role name> <drop behavior>
```

2. *Rationale: <drop role statement> needs <drop behavior>.*

   Add the following Syntax Rule:

   2) Let *DB* be the <drop behavior>.

3. *Rationale: <drop role statement> needs <drop behavior>.*

   Replace General Rules 1) and 2) with:

   1) For every <authorization identifier> *A* identified by a role authorization descriptor as having been granted to *R*, the following <revoke role statement> is effectively executed without further Access Rule checking:

      ```
      REVOKE R FROM A DB
      ```

12.7 <revoke statement>

This Subclause is modified by Subclause 11.3, "<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 11.3, "<revoke statement>", in ISO/IEC 9075-13.

1. *Rationale: Grantor determination must be done in the Syntax Rules.*

   Insert the following Syntax Rules:

   2) Case:

      a) If GRANTED BY is omitted, then let *G* be OMITTED.

      b) Otherwise, let *G* be <grantor>.

   3) Let *A* be the result of applying the Syntax Rules of Subclause 12.8, "Grantor determination", with *G* as GRANTOR. *A* is the *grantor* of the <revoke statement>.

2. *Rationale: Supply missing Access Rules.*

   Insert the following Access Rules:

   1) Case:

      a) If the <revoke statement> is a <revoke privilege statement>, then:

         i) The applicable privileges for *A* shall include a privilege identifying *O*, or if *O* is a table, a column of *O* or a table/method pair whose table is *O*.

         ii) If <privileges> contains a <privilege column list> PCL, then for every <column name> *CN* contained in *PCL*, the applicable privileges shall include a column privilege whose object is the column identified by *CN*.

         iii) If <privileges> contains a <privilege method list> PML, then for every <specific routine designator> *SRD* contained in *PML*, the applicable privileges shall include a privilege whose object is the method identified by *SRD*.

      b) If the <revoke statement> is a <revoke role statement>, then, for every role *R* identified by a <role revoked>, there shall exist a grantable role authorization descriptor whose role name is *R*, and whose grantee is *A* or an applicable role of *A*.

3. *Rationale: Grantor determination must be done in the Syntax Rules.*

   Delete General Rules 1), 2) and 3).

12.8 Grantor determination

1. *Rationale: Grantor determination must be done in the Syntax Rules.*

   Replace the Subclause Signature and add a new Syntax Rule:

Subclause Signature

```
"Grantor determination" [Syntax Rules] (
  Parameter: "GRANTOR"
)
```

   1) Let *G* be the *GRANTOR* in an application of the Rules of this Subclause.

2. *Rationale: Grantor determination must be done in the Syntax Rules.*

   Insert the following Syntax Rule:

   2) The grantor *A* is derived from *G* as follows.

      Case:

      a) If *G* is OMITTED, then

         Case: