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

TECHNICAL CORRIGENDUM 1

Published 2000-12-15

Information technology — Database languages — SQL —

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

TECHNICAL CORRIGENDUM 1

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

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

RECTIFICATIF TECHNIQUE 1

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

Statement of purpose for rationale:

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

Notes on numbering:

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

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

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

4.8.2.3 Locators

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

Replace the 8th paragraph with:

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

Replace 9th paragraph with:

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

4.11.2 SQL-statements classified by function

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

Add a list element to the 1st paragraph:

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

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

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

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

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

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

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

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

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

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

Delete the 2nd bullet from the 1st paragraph

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

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

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

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

Add the following bullets to the 1st paragraph:

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

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

5.6.5.1 Additional functional classes of SQL-statements

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

Replace the bullet of the 1st paragraph with:

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

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

Replace the 2nd paragraph with:

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

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

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

Replace the entire Subclause with:

6.2.5 Relationships of incremental parts within ISO/IEC 9075

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

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

6.2.5.1 New and modified Clauses, Subclauses, and Annexes

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

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

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

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

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

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

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

6.2.5.2 New and modified Format items

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

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

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

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

Format items in new Subclauses are unmarked.

6.2.5.3 New and modified paragraphs and rules

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

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

$\boxed{\text{Replace the 5th paragraph}}$ means that the following text is to replace the fifth paragraph of the corresponding Subclause in ISO/IEC 9075-2.

$\boxed{\text{Replace SR6) b) ii)}}$ means that the following text is to replace Syntax Rule 6)b)ii) of the corresponding Subclause in ISO/IEC 9075-2.

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

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

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

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

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

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

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

6.2.5.4 New and modified tables

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

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

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

Annex B SQL Packages

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

In Table 2 — SQL Packages, delete line 3:

Delete the entire Annex B.3, "OLAP facilities".

Annex B.9 SQL/MM Support

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

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

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

3.1.1 Definitions taken from ISO/IEC 10646

1. *Rationale: Remove unused definitions.*

Replace the Subclause with:

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

a) character

3.1.2 Definitions taken from Unicode

1. *Rationale: Remove unused definitions.*

Replace the Subclause with:

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

a) control character

3.1.5 Definitions provided in Part 2

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

Replace item b) with:

b) **assignment**: The operation whose effect is to ensure that the value at a site *T* (known as the target) is identical to a given value *S* (known as the source). Assignment is frequently indicated by the use of the phrase "*T* is set to *S*" or "the value of *T* is set to *S*".

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

Replace item i) with:

i) **comparable** (of a pair of values): Capable of being compared, according to the rules of Subclause 8.2 "<comparison predicate>". In most, but not all, cases the values of a data type can be compared one with another. For the specification of comparability of individual data types, see Subclauses 4.2 to 4.11. Further, if a value of one data type can be compared with a value of another data type, then the two data types are said to be (mutually) comparable, see Subclause 4.12, "Type conversions and mixing of data types".

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

Replace item l) with:

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

Two null values of comparable type are not distinct.

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

For two nonnull values, the following rules apply:

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

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

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

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

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

NOTE 0.1 — The result of evaluating whether or not two comparable values are distinct is never *unknown*. The result of evaluating whether or not two values that are not comparable, for example, values of a user-defined type that has no comparison type, is not defined.

4. Rationale: Define "equals".

Insert the following definition:

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

5. Rationale: Define "identical"

Insert the following definition:

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

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

In item vv) delete the following bullet items:

— U+200C, Zero Width Non-Joiner

— U+200D, Zero Width Joiner

— U+200E, Left-To-Right Mark

— U+200F, Right-To-Left Mark

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

In item vv) add the following bullet item:

— U+202F, Narrow No-Break Space

4.1 Data types

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

Replace the 3rd paragraph with:

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

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

Replace the 11th paragraph with:

> A structured type *ST* is *directly based on* a data type *DT* if any of the following are true:
>
> a) *DT* is the declared type of some attribute of *ST*.
>
> b) *DT* is the direct supertype of *ST*.
>
> c) *DT* is a direct subtype of *ST*.
>
> d) *DT* is compatible with *ST*.

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

Add the following paragraph immediately after the 15th paragraph:

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

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

Add the following paragraphs after the last paragraph:

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

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

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

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

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

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

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

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

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

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

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

A type *T* is *DT-EC-ordered* if *T* is *DTE*-ordered, where *DTE* is the set of distinct types with EQUALS ONLY comparison form (DT-EC stands for “distinct type - equality comparison”)

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

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

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

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

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

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

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

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

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

4.2.1 Character strings and collating sequences

1. Rationale: Remove redundant definition.

Replace the 4th paragraph with:

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

4.2.4 Named character sets

1. Rationale: Correct grammatical error.

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

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

2. Rationale: Editorial.

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

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

3. Rationale: Editorial.

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

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

4.3.1 Binary string comparison

1. *Rationale: Remove redundant definition.*

Replace the paragraph with:

All binary strings are mutually comparable.

4.4.1 Bit string comparison and assignment

1. *Rationale: Remove redundant definition.*

Replace the paragraph with:

All bit strings are mutually comparable.

4.7.1 Datetimes

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

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

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

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

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

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

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

4.7.3 “Operations involving datetimes and intervals”

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

Replace the 5th paragraph with:

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

2. *Rationale: Editorial - typographical error.*

Replace the 7th paragraph with:

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

4.8 User-defined types

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

Replace the 3rd paragraph with:

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

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

Replace the 4th paragraph with:

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

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

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

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

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

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

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

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

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

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

4.8.1 Observers and mutators

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

Replace the 3rd paragraph with:

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

4.8.2 Constructors

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

Replace the 1st and 2nd paragraphs with:

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

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

For every structured type *ST* that is instantiable, zero or more SQL-invoked constructor methods can be specified. The name of those methods must be equivalent to the name of the type for which they are specified.
NOTE 6.1 — SQL-invoked constructor methods are original methods that cannot be overloaded. An SQL-invoked constructor method and a regular function may exist such that both have equivalent function names, the types of the first parameters (if any) of the method's augmented parameter list and the function's parameter list are the same, and the types of the corresponding remaining parameters (if any) are identical according to the Syntax Rules of Subclause 10.14, "Data type identity".

4.8.4 User-defined type comparison and assignment

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

Replace the 6th paragraph with:

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

4.10 Reference types

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

Replace the 2nd paragraph with:

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

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

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

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

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

— The name of the referenced type.

— The length of the referenced type.

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

Replace the 8th paragraph with:

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

4.13 Data conversions

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

Replace the 1st paragraph with:

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

4.16 Tables

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

Replace the 7th paragraph with:

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

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

Replace the 11th paragraph with:

Every table descriptor includes:

— The column descriptor of each column in the table.

— The name, if any, of the structured type, if any, associated with the table.

— An indication of whether the table is insertable-into or not.

— An indication of whether the base table is a referenceable table or not, and an indication of whether the self-referencing column is a system-generated, a user-generated, or a derived self-referencing column.

— A list, possibly empty, of the names of its direct supertables.

— A list, possibly empty, of the names of its direct subtables.

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

Replace the 13th paragraph with:

A derived table descriptor describes a derived table. In addition to the components of every table descriptor, a derived table descriptor includes:

— The <query expression> that defines how the table is to be derived.

— An indication of whether the derived table is updatable or not.

4.16.3 Operations involving tables

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

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

> Let *T* be a table whose value is *TV1*. For every row *RR*, let n_{RR} be the number of rows in *TV1* that are identical to *RR*.
>
> The effect of deleting a row *R* from *T* is to replace the value *TV1* of *T* with the value *TV2* such that, for *RR* identical to *R*, the number of rows in *TV2* that are identical to *RR* is n_{RR} - 1 (one) and for *RR* not identical to *R*, the number of rows in *TV2* that are not identical to *RR* is n_{RR}. The primary effect of a <delete statement: positioned> on *T* is to delete exactly one specified row from *T*. The primary effect of a <delete statement: searched> on *T* is to delete zero or more rows from *T*.
>
> The effect of replacing a row *R1* with the row *R2* in *T* is to replace the value *TV1* of *T* with the value *TV2* such that, if *R1* is not identical to *R2*, then, for *RR* identical to *R1*, the number of rows in *TV2* that are identical to *RR* is n_{RR} - 1 (one), for *RR* identical to *R2*, the number of rows in *TV2* that are identical to *RR* is n_{RR} + 1 (one), and for *RR* identical to neither *R1* nor *R2*, the number of rows in *TV2* that are identical to *RR* is n_{RR}. Otherwise *R1* is identical to *R2* and the number of rows in *TV2* that are identical to *RR* is n_{RR} for all *RR*.
>
> The primary effect of an <update statement: positioned> on *T* is to replace exactly one specified row in *T* with some specified row. The primary effect of an <update statement: searched> on *T* is to replace zero or more rows in *T*.
>
> If *T*, as well as being updatable, is insertable-into, then rows can be inserted into it. The effect of inserting a row *R* into *T* is to replace the value *TV1* of *T* with the value *TV2* such that, for *RR* identical to *R*, the number of rows in *TV2* that are identical to *RR* is n_{RR} + 1 (one) and, for *RR* not identical to *R*, the number of rows in *TV2* that are identical to *RR* is n_{RR}. The primary effect of an <insert statement> on *T* is to insert into *T* each of the zero or more rows contained in a specified table.

4.18.1 General rules and definitions

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

Replace the 2nd paragraph with:

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

2. *Rationale: Correct erroneous symbol.*

Replace the 8th paragraph with:

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

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

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

Replace the 1st paragraph with:

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

4.20 SQL-schemas

1. *Rationale: Cleanup of imprecise wording.*

Replace the 5th paragraph with:

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

4.21 SQL-client modules

1. *Rationale: Editorial.*

Replace NOTE 29 with:

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

4.23 SQL-invoked routines

1. *Rationale: Editorial.*

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4. *Rationale: Editorial.*

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

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

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

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

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

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

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

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

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

6. *Rationale: Missing word.*

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

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

4.24 Built-in functions

1. *Rationale: Remove undefined term.*

Replace the 1st paragraph with:

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

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

4.25 SQL-paths

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

Replace the 3rd paragraph with:

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

4.26.4 Locators

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

Replace 5th paragraph with:

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

Replace 6th paragraph with:

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

4.30.1 Classes of SQL-statements

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

Replace the 2nd paragraph with:

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

4.30.2 SQL-statements classified by functions

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

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

> — <grant privilege statement>

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

Add the following bullets to the 2nd paragraph:

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

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

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

4.30.2a SQL-statements and SQL-data access indication

The following SQL-statements possibly contain SQL:

— SQL-control statements

— SQL-control declaration statements

— SQL-session statements

— SQL-diagnostics statements

— <free locator statement>

— <hold locator statement>

The following SQL-statements possibly read SQL-data:

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

— SQL-statements that simply contain a <subquery>

The following SQL-statements possibly modify SQL-data:

— SQL-schema statements

— SQL-data change statements

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

4.30.3 SQL-statements and transaction states

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

Replace the 3rd paragraph with:

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

— <return statement>

— <call statement>

4.30.4 SQL-statement atomicity

1. *Rationale: Add missing concept.*

Add the following paragraphs after the 1st paragraph:

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

4.31.1 Authorization identifiers

1. *Rationale: Editorial.*

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

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

4.34.1 Execution contexts

1. *Rationale: Correct cross reference.*

Replace the 1st paragraph with:

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

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

Replace the 2nd paragraph with:

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

4.35.2 Execution of triggers

1. *Rationale: Editorial.*

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

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

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

Replace the 9th paragraph with:

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

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

Replace the 12th paragraph with:

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

5.2 <token> and <separator>

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

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

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

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

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

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

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

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

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

in the correct alphabetical order.

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

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

```
| ADMIN
| AGGREGATE
| ALIAS
| CLASS
| COMPLETION
| DESTROY
| DESTRUCTOR
| DICTIONARY
| EVERY
| HOST
| IGNORE
| INITIALIZE
| ITERATE
| LESS
| LIMIT
| MODIFY
| OFF
| OPERATION
| PARAMETERS
| POSTFIX
| PREFIX
| PREORDER
| SCOPE
| SEQUENCE
| STATEMENT
| STRUCTURE
| TERMINATE
| THAN
| VARIABLE
```

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

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

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

in the correct alphabetical order.

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

Delete NOTE 24.

Delete Syntax Rule 23)

Replace Syntax Rule 24) with:

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

Replace Syntax Rules 25) and 26) with:

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

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

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

Replace Conformance Rule 1) with:

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

5.3 <literal>

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

Replace Conformance Rule 7) with:

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

2. *Rationale: Editorial.*

Replace Conformance Rules 8) and 9) with:

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

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

5.4 Names and identifiers

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

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

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

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

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

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

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

Replace Conformance Rule 12) with:

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

6.1 <data type>

1. *Rationale: Delete a redundant rule.*

Delete General Rule 11)

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

Replace Conformance Rule 1) with:

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

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

Replace Conformance Rule 8) with:

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

6.3 <value specification> and <target specification>

1. *Rationale: Editorial.*

Replace Syntax Rule 7) with:

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

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

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

Case:

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

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

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

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

Case:

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

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

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

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

Case:

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

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

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

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

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

Case:

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

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

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

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

6.5 <identifier chain>

1. Rationale: Take query names into account.

Replace Syntax Rule 7) a) with:

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

Case:

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

Case:

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

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

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

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

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

Replace Syntax Rule 11) with:

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

6.11 <method invocation>

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

In the Format add the following production:

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

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

Delete Syntax Rule 3).

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

Replace Syntax Rule 5) with:

5) Case:

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

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

4. *Rationale: Provide correct logical structure.*

Replace Syntax Rule 6) with

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

```
MN ( VEP AS DT AL )
```

   b) Otherwise,

      Case:

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

```
MN ( VEP AL )
```

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

```
MN ( VEP AL )
```

6.16 <set function specification>

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

Add the following Syntax Rule:

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

Delete Syntax Rule 7).

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

Replace Syntax Rule 8) with:

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

Delete Syntax Rule 10).

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

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

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

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

Replace Conformance Rule 7) with:

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

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

Replace Conformance Rule 8) with:

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

Add the following Conformance Rule:

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

6.18 <string value function>

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

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

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

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

Replace Syntax Rule 4) with:

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

   a) Case:

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

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

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

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

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

Replace Syntax Rule 5) with:

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

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

   b) Case:

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

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

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

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

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

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

Replace Syntax Rule 7) with:

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

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

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

   c) Case:

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

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

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

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

Replace Syntax Rule 8) with:

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

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

b) Case:

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

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

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

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

Replace Syntax Rule 9) with:

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

a) If FROM is specified, then:

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

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

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

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

```
TRIM ( SRC )
```

is equivalent to

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

c) Case:

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

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

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

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

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

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

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

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

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

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

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

is true.

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

g) If the predicate

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

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

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

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

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

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

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

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

8. Rationale: Editorial.

Replace Conformance Rule 3) with:

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

6.20 <interval value function>

1. Rationale: Editorial - typographical error.

Replace Syntax Rule 1) with:

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

Replace General Rule 1) with:

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

Case:

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

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

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

2. Rationale: Editorial.

Replace Conformance Rule 1) with:

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

6.22 <cast specification>

1. Rationale: Editorial.

Replace Syntax Rule 4) with:

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

2. Rationale: Editorial.

Replace the introductory paragraph of Syntax Rule 6) with:

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

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

3. *Rationale: Editorial.*

Replace Syntax Rule 13) c) with:

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

4. *Rationale: Improve wording.*

Replace Syntax Rule 8) d) with:

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

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

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

TV is the result of

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

Case:

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

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

Replace Syntax Rule 8) d) with:

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

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

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

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

Case:

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

ii) Otherwise, *TV* is the result of

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

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

Replace Syntax Rule 11) with:

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

Case:

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

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

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

Replace Syntax Rule 12) with:

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

Case:

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

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

5. *Rationale: Editorial.*

Replace General Rule 16) d) with:

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

6.23 <value expression>

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

Replace Syntax Rule 2) with:

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

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

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

Replace General Rule 5) with:

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

6.24 <new specification>

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

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

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

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

Replace Syntax Rule 3) with:

3) Case:

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

```
NEW RN( )
```

then:

Case:

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

```
RN( )
```

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

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

b) Otherwise, the <new specification>

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

is equivalent to the <new invocation>

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

3. *Rationale: Editorial.*

Replace Conformance Rule 1) with:

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

6.25 <subtype treatment>

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

Replace the Format with:

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

<subtype operand> ::= <value expression>

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

2. *Rationale: Editorial.*

Replace Conformance Rule 1) with:

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

6.26 <numeric value expression>

1. *Rationale: Editorial.*

Replace General Rule 5) with:

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

Case:

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

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

6.27 <string value expression>

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

Replace Syntax Rule 3) with:

3) Case:

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

Case:

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

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

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

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

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

Replace General Rule 2) with:

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

Case:

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

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

Case:

i) If the most specific type of either *S1* or *S2* is large object character string, then let *LOL* be the implementation-defined maximum length of a large object character string.

   Case:

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

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

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

ii) If the most specific type of either *S1* or *S2* is variable-length character string, then let *VL* be the implementation-defined maximum length of a variable-length character string.

   Case:

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

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

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

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

6.30 <boolean value expression>

1. *Rationale: Correct use of undefined non-terminal.*

Replace Syntax Rule 1) and 2) with:

1) The declared type of a <nonparenthesized value expression primary> shall be boolean.

2) If NOT is specified in a <boolean test>, then let *BP* be the contained <boolean primary> and let *TV* be the contained <truth value>. The <boolean test> is equivalent to:

```
( NOT ( BP IS TV ) )
```

Replace Syntax Rule 3) b) with:

b) If *BVE* is a <parenthesized boolean value expression> and the immediately contained <boolean value expression> is a known-not-null condition for *X*, then *BVE* is a *known-not-null condition* for *X*.

b.1) If *BVE* is a <nonparenthesized value expression primary>, then *BVE* is not a known-not-null condition for *X*.

7.1 <row value constructor>

1. Rationale: Specify Syntax Rules for "contextually typed" terms.

Add the following Syntax Rules:

4.1) A <contextually typed row value constructor element> immediately contained in a <contextually typed row value constructor> shall not be a <value expression> of the form "<left paren> <value expression> <right paren>"

NOTE 93.1 — This Rule removes a syntactic ambiguity. A <contextually typed row value constructor> of this form is permitted, but is parsed in the form "<left paren> <contextually typed row value constructor list> <right paren>".

4.2) A <contextually typed row value constructor element> immediately contained in a <contextually typed row value constructor> shall not be a <value expression> that is a <row value expression>.

NOTE 93.2 — This Rule removes a syntactic ambiguity, since otherwise a <contextually typed row value constructor> could be a <row value expression>, and a <row value expression> could be a <contextually typed row value constructor>.

4.3) Let *CTRVC* be the <contextually typed row value constructor>. The declared type of *CTRVC* is a row type described by a sequence of (<field name>, <data type>) pairs, corresponding in order to each <contextually typed row value constructor element> *X* simply contained in *CTRVC*. The data type is the declared type of *X* and the <field name> is implementation-dependent and not equivalent to the <column name> name of any column or field, other than itself, of a table referenced by any <table reference> contained in the SQL-statement.

4.4) The degree of a <contextually typed row value constructor> is the degree of its declared type.

2. Rationale: Supply missing Conformance Rules for <contextually typed row value constructor>

Add the following Conformance Rules:

5) Without Feature F641, "Row and table constructors", a <contextually typed row value constructor> that is not simply contained in a <contextually typed table value constructor> shall not contain more than one <row value constructor element>.

6) Without Feature F641, "Row and table constructors", a <contextually typed row value constructor> shall not be a <row subquery>.

7.3 <table value constructor>

1. Rationale: Use the correct non-terminal symbols.

Replace Conformance Rule 1) with:

1) Without Feature F641, "Row and table constructors", the <contextually typed row value expression list> of a <contextually typed table value constructor> shall contain exactly one <contextually typed row value constructor> *RVE*. *RVE* shall be of the form "(<contextually typed row value constructor element list>)".

2. *Rationale: Supply missing Conformance Rules for <contextually typed table value constructor>*

Add the following Conformance Rules:

3) Without Feature F641, “Row and table constructors”, the <contextually typed row value expression list> of a <contextually typed table value constructor> shall contain exactly one <contextually typed row value constructor> *RVE*. *RVE* shall be of the form “(<contextually typed row value constructor element list>)”.

4) Without Feature F641, “Row and table constructors”, conforming SQL language shall not contain any <contextually typed table value constructor>.

7.6 <table reference>

1. *Rationale: Use correct containment.*

Replace Syntax Rules 1) and 2) with:

1) If a <table reference> *TR* simply contains a <collection derived table> *CDT*, then let *C* be the <collection value expression> simply contained in *CDT*, let *CN* be the <correlation name> simply contained in *TR*, and let *TEMP* be an <identifier> that is not equivalent to *CN* nor to any other <identifier> contained in *TR*.

   a) Case:

      i) If *TR* specifies a <derived column list> *DCL*, then

         Case:

         1) If *CDT* specifies WITH ORDINALITY, then *DCL* shall contain 2 <column name>s. Let *N1* and *N2* be respectively the first and second of those <column name>s.

         2) Otherwise, *DCL* shall contain 1 (one) <column name>; let *N1* be that <column name>. Let *N2* be a <column name> that is not equivalent to *N1*, *CN*, *TEMP*, or any other <identifier> contained in *TR*.

      ii) Otherwise, let *N1* and *N2* be two <column name>s that are not equivalent to one another nor to *CN*, *TEMP*, or any other <identifier> contained in *TR*.

   b) Let *RECQP* be:

```
WITH RECURSIVE TEMP ( N1, N2 ) AS
  ( SELECT C[1] AS N1, 1 AS N2
      FROM ( VALUES( 1 ) ) AS CN
     WHERE 0 < CARDINALITY( C )
    UNION
    SELECT C[N2+1] AS N1, N2+1 AS N2
      FROM TEMP
     WHERE N2 < CARDINALITY( C )
  )
```

   c) Case:

      i) If *TR* specifies a <derived column list> *DCL*, then let *PDCLP* be:

```
( DCL )
```

ii) Otherwise, let *PDCLP* be a zero-length string.

d) Case:

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

```
LATERAL ( RECQP SELECT * FROM TEMP AS CN PDCLP )
```

ii) Otherwise, let *ELDT* be:

```
LATERAL ( RECQP SELECT N1 FROM TEMP AS CN PDCLP )
```

e) *CDT* is equivalent to the <lateral derived table> *ELDT*.

2) A <correlation name> simply contained in a <table reference> *TR* is *exposed* by *TR*. A <table or query name> simply contained in a <table reference> *TR* is *exposed* by *TR* if and only if *TR* does not specify a <correlation name>.

2. *Rationale: The scope of an exposed table name or correlation name must include the <order by clause> of a simple table query. Use correct containment.*

Replace Syntax Rule 3) with:

3) Case:

a) If a <table reference> *TR* is contained in a <from clause> *FC* with no intervening <query expression>, then the *scope clause SC* of *TR* is the <select statement: single row> or innermost <query specification> that contains *FC*. The scope of the exposed <correlation name> or exposed <table or query name> of *TR* is the <select list>, <where clause>, <group by clause>, and <having clause> of *SC*, together with every <lateral derived table> that is simply contained in *FC* and is preceded by *TR*, and every <collection derived table> that is simply contained in *FC* and is preceded by *TR*, and the <join condition> of all <joined table>s contained in *SC* that contain *TR*. If *SC* is the <query specification> that is the <query expression body> of a simple table query *STQ*, then the scope of the exposed <correlation name> or exposed <table or query name> also includes the <order by clause> of *STQ*.

b) Otherwise, the *scope clause SC* of *TR* is the outermost <joined table> that contains *TR* with no intervening <query expression>. The scope of the exposed <correlation name> or exposed <table or query name> of *TR* is the <join condition> of *SC* and of all <joined table>s contained in *SC* that contain *TR*.

3. *Rationale: supply correct rules for equivalency of <table name>s, <query name>s and <correlation name>s.*

Replace Syntax Rules 4) and 5) with:

4) A <table or query name> that is a <table name> that is exposed by a <table reference> *TR* shall not be equivalent to any other <table or query name> that is a <table name> that is exposed by a <table reference> with the same scope clause as *TR*.

4.1) A <table or query name> that is a <query name> that is exposed by a <table reference> *TR* shall not be equivalent to the <qualified identifier> of any <table or query name> that is a <table name> that is exposed by a <table reference> with the same scope clause as *TR* , and shall not be equivalent to any other <table or query name> that is a <query name> that is exposed by a <table reference> with the same scope clause as *TR*.

5) A <correlation name> that is exposed by a <table reference> *TR* shall not be equivalent to any other <correlation name> that is exposed by a <table reference> with the same scope clause as *TR* and shall not be equivalent to the <qualified identifier> of any <table or query name> that is a <table name> that is exposed by a <table reference> with the same scope clause as *TR*, and shall not be equivalent to any <table or query name> that is a <query name> that is exposed by a <table reference> with the same scope clause as *TR*.

4. Rationale: Use correct containment.

Replace Syntax Rules 6) and 7) with:

6) A <table or query name> simply contained in a <table reference> *TR* has a scope clause and scope defined by that <table reference> if and only if the <table or query name> is exposed by *TR*.

7) If *TR* simply contains <only spec> *OS* and the table identified by the <table or query name> *TN* is not a typed table, then *OS* is equivalent to *TN*.

Replace Syntax Rule 11) with:

11) Case:

a) If no <derived column list> is specified, then the row type *RT* of the <table reference> is the row type of its simply contained <table or query name>, <derived table>, <lateral derived table>, or <joined table>.

b) Otherwise, the row type *RT* of the <table reference> is described by a sequence of (<field name>, <data type>) pairs, where the <field name> in the *i*-th pair is the *i*-th <column name> in the <derived column list> and the <data type> in the *i*-th pair is the declared type of the *i*-th column of the <derived table>, <joined table>, or <lateral derived table>, of the table identified by the <table or query name> simply contained in the <table reference>.

Replace Syntax Rule 12) and 13) with:

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

13) A <derived table> or <lateral derived table> is an *insertable-into derived table* if and only if the <query expression> simply contained in *TR* is insertable-into.

Replace Syntax Rule 15) c) with:

15) c) If *TR* immediately contains a <derived table> or <lateral derived table>, then every updatable column of the table identified by the <query expression> simply contained in <derived table> or <lateral derived table> is called an *updatable column* of *TR*.

Replace Syntax Rule 16) with:

16) If the <table or query name> simply contained in <table reference> is not a query name in scope, then let *T* be the table identified by the <table name> immediately contained in <table or query name>. If the <table reference> is not contained in a <schema definition>, then the schema identified by the explicit or implicit qualifier of the <table name> shall include the descriptor of *T*. If the <table reference> is contained in a <schema definition> *S*, then the schema identified by the explicit or implicit qualifier of the <table name> shall include the descriptor of *T*,or *S* shall contain a <schema element> that creates the descriptor of *T*.
NOTE 94 — "*query name in scope*" is defined in Subclause 7.12, "<query expression>".

5. *Rationale: Use correct containment and correct the <only spec> and untyped tables.*

Replace Access Rule 1) with:

1) If <table reference> simply contains a <table or query name> that is a <table name>, then:

   a) Let *T* be the table identified by the <table name> immediately contained in the <table or query name> simply contained in <table reference>.

   b) If *T* is a base table or a viewed table and the <table reference> is contained in any of:

      — A <query expression> simply contained in a <cursor specification>, a <view definition>, or an <insert statement>.

      — A <table expression> or <select list> immediately contained in a <select statement: single row>.

      — A <search condition> immediately contained in a <delete statement: searched> or an <update statement: searched>.

      — A <value expression> simply contained in a <row value expression> immediately contained in a <set clause>.

      then

      Case:

      i) If <table reference> is contained in an <SQL schema statement> then, the applicable privileges of the <authorization identifier> that owns the containing schema shall include SELECT on at least one column of *T*.

      ii) Otherwise, the current privileges shall include SELECT on at least one column of *T*.
      NOTE 95 — “applicable privileges” and “current privileges” are defined in Subclause 10.5, “<privileges>”.

   c) If the <table reference> is contained in a <query expression> simply contained in a <view definition> then the applicable privileges of the <authorization identifier> that owns the view shall include SELECT for at least one column of *T*.

   d) If *TR* simply contains <only spec> and *TR* identifies a typed table, then

      Case:

      i) If <table reference> is contained in a <schema definition>, then the applicable privileges of the <authorization identifier> that owns the containing schema shall include SELECT WITH HIERARCHY OPTION on at least one supertable of *T*.

      ii) Otherwise, the current privileges shall include SELECT WITH HIERARCHY OPTION on at least one supertable of *T*.

      NOTE 96 — “applicable privileges” and *current privileges* are defined in Subclause 10.5, “<privileges>”.

6. *Rationale: Use correct containment.*

Replace General Rule 1) with:

1) A <correlation name> or exposed <table or query name> simply contained in a <table reference> defines that <correlation name> or <table or query name> to be an identifier of the table identified by the <table or query name> or <derived table> or <lateral derived table> of that <table reference>.

7.7 <joined table>

1. *Rationale: NATURAL joins should be subject to the same syntactic restrictions as <named column join>s. Provide consistent Syntax Rules for comparison operations.*

Replace Syntax Rule 7) c) with:

7) c) Let C_1 and C_2 be a pair of corresponding join columns contained in T_1 and T_2 , respectively. C_1 and C_2 shall be comparable, and the declared type of C_1 or C_2 shall not be LOB-ordered or array_ordered.

2. *Rationale: Named column joins implicitly perform <comparison predicate>s, and should be subject to the same Conformance Rules. Provide consistent Conformance Rules for comparison operations.*

Add the following Conformance Rule:

4.1) Without Feature S024, "Enhanced structured types", if NATURAL or <named column join> is specified, and if *C* is a corresponding join column, then the declared type of *C* shall not be an ST-ordered type.

7.8 <where clause>

1. *Rationale: Clarification.*

Replace Syntax Rule 2) with:

2) If a <value expression> directly contained in the <search condition> is a <set function specification>, then the <where clause> shall be contained in a <having clause> or <select list>, the <set function specification> shall contain a column reference, and every column reference contained in the <set function specification> shall be an outer reference .

7.9 <group by clause>

1. *Rationale: Disambiguate BNF non-terminal names; provide a unified treatment of CUBE and ROLLUP.*

In the Format, replace the production for <group by clause> with:

```
<group by clause> ::= GROUP BY <grouping element list>
```

Insert the following production:

```
<grouping element list> ::=
    <grouping element> [ {<comma> <grouping element>}...]
```

In the Format, replace the production for <grouping specification> with:

```
<grouping element> ::=
        <ordinary grouping set>
      | <rollup list>
      | <cube list>
      | <grouping sets specification>
      | <grand total>
```

In the Format, replace the production for <grouping sets list> with:

```
<grouping sets specification> ::=
     GROUPING SETS <left paren> <grouping set list> <right paren>
```

Replace the production for <grouping set> with:

```
<grouping set> ::=
        <ordinary grouping set>
      | <rollup list>
      | <cube list>
      | <grouping sets specification>
      | <grand total>
```

2. *Rationale: Extend the restrictions on the use of columns with types based on LOB and array types. Provide consistent Syntax Rules for comparison operations.*

Delete Syntax Rule 2).

3. *Rationale: Provide a correct, unified treatment of CUBE and ROLLUP.*

Replace Syntax Rule 3) with:

3) Let *QS* be the <query specification> that simply contains the <group by clause>, and let *SL*, *FC*, *WC*, *GBC* and *HC* be the <select list>, the <from clause>, the <where clause> if any, the <group by clause> and the <having clause> if any, respectively, that are simply contained in *QS*.

Delete Syntax Rule 4).

4. *Rationale: Extend the restrictions on the use of columns with types based on LOB and array types. Provide consistent Syntax Rules for comparison operations.*

Replace Syntax Rule 5) with:

5) The declared type of a grouping column shall not be LOB-ordered, array-ordered, UDT-EC-ordered, or UDT-NC-ordered.

5. *Rationale: Provide a correct, unified treatment of CUBE and ROLLUP*

Insert the following Syntax Rules:

6.1) A <grouping set list> shall not contain a <grouping sets specification>.

6.2) If a <group by clause> simply contains a <grouping sets specification> *GSS*, then *GSS* shall be the only <grouping element> simply contained in the <group by clause>.

6.3) Let *SL1* be obtained from *SL* by replacing every <asterisk> and <asterisked identifier chain> using the syntactic transformations in the Syntax Rules of Subclause 7.12 “<query specification>”.

6.4) A <group by clause> is *primitive* if it does not contain a <rollup list>, <cube list>, <grouping sets specification>, or <grouping column reference list>, and does not contain both a <grouping column reference> and an <empty grouping set>.

6.5) A <group by clause> is *simple* if it does not contain a <rollup list>, <cube list> or <grouping sets specification>.

6.6) If *GBC* is a simple <group by clause> that is not primitive, then *GBC* is transformed into a primitive <group by clause> as follows:

a) Let *NSGB* be the number of <grouping column reference>s contained in *GBC*.

b) Case:

i) If *NSGB* is 0 (zero), then *GBC* is replaced by

```
GROUP BY ()
```

ii) Otherwise:

1) Let $SGCR_1$, ... $SGCR_{NSGB}$ be an enumeration of the <grouping column reference>s contained in *GBC*.

2) *GBC* is replaced by

GROUP BY $SGCR_1$, ... , $SGCR_{NSGB}$

NOTE 101.1 — that is, a simple <group by clause> that is not primitive may be transformed into a primitive <group by clause> by deleting all parentheses, and deleting extra <comma>s as necessary for correct syntax. If there are no grouping columns at all (for example, GROUP BY (), ()) this is transformed to the canonical form GROUP BY ().

6.7) If *GBC* is a primitive <group by clause>, then let *SLNEW* and *HCNEW* be obtained from *SL1* and *HC*, respectively, by replacing every <grouping operation> by the exact numeric literal 0 (zero). *QS* is equivalent to

```
SELECT SLNEW FC WC GBC HCNEW
```

Replace Syntax Rules 7), 8), 9), 10) and 11) with:

7) If *RL* is a <rollup list>, then let GCR_i range over the *n* <grouping column reference>s contained in *RL*. *RL* is equivalent to

GROUPING SETS (
(GCR_1, GCR_2 , ..., GCR_n),
(GCR_1 , GCR_2 , ..., GCR_{n-1}),
(GCR_1 , GCR_2 , ..., GCR_{n-2}),
...
(GCR_1),
()
)

NOTE 102 — The result of the transform is to replace *RL* with a <grouping sets specification> that contains a <grouping set> for every initial sublist of the <grouping column reference list> of the <rollup list> , obtained by dropping elements from the right, one by one , and regarding <empty grouping set> as the shortest such initial sublist.

8) If *CL* is a <cube list>, then let GCR_i range over the *n* <grouping column reference>s contained in *CL*. *CL* is transformed as follows:

   a) Let $M = 2^n - 1$.

   b) For each *i* between 0 (zero) and *M*:

      i) Let BSL_i be the <bit string literal> containing *n* <bit>s whose binary value is *i*.

      ii) For each *j* between 1 (one) and *n*, let $B_{i,j}$ be the *j*-th <bit>, counting from left to right, in BSL_i.

      iii) For each *j* between 1 (one) and *n*, let $GSLCR_{i,j}$ be

      Case:

      1) If $B_{i,j}$ is 0 (zero), then the zero-length string.

      2) If $B_{i,j}$ is 1 (one), and $B_{i,k}$ is 0 (zero) for all $k < j$, then GCR_j.

      3) Otherwise, <comma> GCR_j.

      iv) Let GSL_i be the <ordinary grouping set>

      ($GSLCR_{i,1}$ $GSLCR_{i,2}$ . . . $GSLCR_{i,n}$)

   c) *CL* is equivalent to:

   GROUPING SETS (GSL_M , GSL_{M-1} , . . . GSL_0)

   NOTE 103 — The result of the transform is to replace *CL* with a <grouping sets specification> that contains a <grouping set> for all possible subsets of the set of grouping columns in the <grouping column reference list> of the <cube list>, including <empty grouping set> as the empty subset with no grouping columns. For example, CUBE (A, B, C) is equivalent to:

```
GROUPING SETS ( /* BSLi */
( A, B, C),    /* 111 */
( A, B    ),      /* 110 */
( A,    C),    /* 101 */
( A       ),   /* 100 */
(    B, C),    /* 011 */
(    B    ),   /* 010 */
(       C),    /* 001 */
(         )    /* 000 */
)
```

9) If <grouping sets specification> *GSSA* simply contains another <grouping sets specification> *GSSB*, then *GSSA* is transformed as follows:

   a) Let *NA* be the number of <grouping set>s simply contained in *GSSA*, and let *NB* be the number of <grouping set>s simply contained in *GSSB*.

   b) Let GSA_i be an enumeration of the <grouping set>s simply contained in *GSSA*, for *i* between 1 (one) and *NA*.

   c) Let GSB_i be an enumeration of the <grouping set>s simply contained in *GSSB*, for *i* between 1 (one) and *NB*.

   d) Let *k* be the index such that $GSSB = GSA_k$.

e) *GSSA* is equivalent to:

```
GROUPING SETS
  ( GSA_1 , GSA_2 , ... GSA_k-1 ,
    GSB_1 , ... , GSB_NB,
    GSA_k+1 , ..., GSA_NA
  )
```

NOTE 103.1 — Thus the nested <grouping sets specification> is removed by simply “promoting” each of its <grouping set>s to be a <grouping set> of the encompassing <grouping sets specification>.

10) If *CGB* is a <group by clause> that is not simple, then *CGB* is transformed as follows:

a) Previous Syntax Rules are applied repeatedly to eliminate any <grouping set specification> that is nested in another <grouping set specification>, as well as any <rollup list> and any <cube list>.

NOTE 103.2 — As a result, *CGB* is a list of two or more <grouping set>s, each of which is an <ordinary grouping set>, an <empty grouping set> or a <grouping sets specification> that contains only <ordinary grouping set>s and <empty grouping set>s. There are no remaining <rollup list>s, <cube list>s, or nested <grouping sets specification>s.

b) Any <grouping element> *GS* that is an <ordinary grouping set> or an <empty grouping set> is replaced by the <grouping sets specification>:

```
GROUPING SETS ( GS )
```

NOTE 103.3 — As a result, *CGB* s a list of two or more <grouping sets specification>s.

c) Let *GSSX* and *GSSY* be the first two <grouping sets specification>s in *CGB*. *CGB* is transformed by replacing *GSSX* <comma> *GSSY* as follows:

i) Let *NX* be the number of <grouping set>s in *GSSX* and let *NY* be the number of <grouping set>s in *GSSY*.

ii) Let GSX_i for *i* between 1 (one) and *NX* be the <grouping set>s contained in *GSSX*, and let GSY_i for *i* between 1 (one) and *NY* be the <grouping set>s contained in *GSSY*.

iii) Let *MX(i)* be the number of <grouping column reference>s in GSX_i, and let *MY(i)* be the number of <grouping column reference>s in GSY_i.

NOTE 103.4 — If GSX_i is <empty grouping set>, then *MX(i)* is 0 (zero); and similarly for *GSY i*.

iv) Let $GCRX_{i,j}$ for *j* between 1 (one) and *MX(i)* be the <grouping column reference>s contained in GSX_i, and let $GCRY_{i,j}$ for *j* between 1(one) and *MY(i)* be the <grouping column reference>s contained in GSY_i.

NOTE 103.5 — If GSX_i is <empty grouping set>, then there are no $GCRX_{i,j}$; and similarly for GSY_i.

v) For each *a* between 1 (one) and *NX* and each *b* between 1 (one) and *NY*, let $GST_{a,b}$ be

```
( GCRX_a,1 , ... , GCRX_a,MX(a), GCRY_b,1 , ... , GCRY_b,MY(b) )
```

that is, an <ordinary grouping set> consisting of $GCRA_{a,j}$ for all *j* between 1 (one) and *MX(a)* followed by $GCRY_{b,j}$ for all *j* between 1 (one) and *MY(b)*.

vi) *CGB* is transformed by replacing *GSSX* <comma> *GSSY* with:

```
GROUPING SETS
  ( GST_{1,1} , ... GST_{1,NY} ,
    GST_{2,1} , ... GST_{2,NY} ,
    . . .
    GST_{NX,1} , ... GST_{NX,NY}
  )
```

NOTE 103.6 — Thus each <ordinary grouping set> in *GSSA* is "concatenated" with each <ordinary grouping set> in *GSSB*. For example,

```
GROUP BY GROUPING SETS ( ( A, B ), ( C ) ),
GROUPING SETS ( ( X, Y ), ( ) )
```

is transformed to.

```
GROUP BY GROUPING SETS (
  ( A, B, X, Y ),
  ( A, B ),
  ( C, X, Y ), ( C ) )
```

d) The previous subrule of this Syntax Rule is applied repeatedly until *CGB* consists of a single <grouping sets specification>.

11) If <grouping specification> consists of a single <grouping sets specification> *GSS* that contains only <ordinary grouping set>s or <grand total>, then:

a) Let *m* be the number of <grouping set>s contained in *GSS*.

b) Let G_{si},1 <= *i* <= *m*, range over the <grouping set>s contained in *GSS*.

c) Let *p* be the number of distinct <column reference>s that are contained in *GSS*.

d) Let *PC* be an ordered list of these <column reference>s ordered according to their left-to-right occurrence in the list.

e) Let PC_k,1 <= *k* <= *p*, be the *k*-th <column reference> in *PC*.

f) Let $DTPC_k$ be the declared type of the column identified by PC_k.

g) Let $CNPC_k$ be the column name of the column identified by $_{Pck}$.

h) For each GS_i:

i) If GS_i is a <grand total>, then let *n(i)* be 0 (zero). If $GS_i$is a <grouping column reference>, then let *n(i)* be 1 (one). Otherwise, let *n(i)* be the number of <grouping column reference>s contained in the <grouping column reference list>.

ii) Let $GCR_{i;j}$,1 <= *j* <= *n(i)*, range over the <grouping column reference>s contained in GS_i.

iii) Case:

1) If GS_i is an <ordinary grouping set>, then

A) Transform *SL1* to obtain *SL2*, and transform *HC* to obtain *HC2*, as follows:

For every PC_k:

If there is no j such that $PC_k = GCR_{i,j}$, then make the following replacements in *SL1* and *HC*:

I) Replace each <derived column> in *SL1* that is a <column reference> that references PC_k by:

```
CAST ( NULL AS DTPCk ) AS CNPCk
```

II) Replace each <column reference> in *SL1* and *HC* that references PC_k and that is not an entire <derived column> by:

```
CAST ( NULL AS DTPCk )
```

III) Replace each <grouping operation> in *SL1* and *HC* that contains a <column reference> that references PC_k by the <literal> 1 (one).

B) Transform *SL2* to obtain *SLNEW*, and transform *HC2* to obtain *HCNEW* by replacing each <grouping operation> that remains in *SL2* and *HC2* by the <literal> 0 (zero).

NOTE 103.7 — Thus the value of a <grouping operation> is 0 (zero) if the grouping column referenced by the <grouping operation> is among the $GCR_{i,j}$ and 1(one) if it is not.

C) Let $GSSQL_i$ be:

```
SELECT SLNEW FC WC
GROUP BY GCRi,1 , ..., GCRi,n(i)
HCNEW
```

2) If G_{si} is a <grand total>, then

A) Transform *SL1* to obtain *SLNEW*, and transform *HC* to obtain *HCNEW*, as follows:

For every k, 1 <= k <= p:

I) Replace each <derived column> in *SL1* that is a <column reference> that references PC_k by:

```
CAST( NULL AS DTPCk ) AS CNPCk
```

II) Replace each <column reference> in *SL1* and *HC* that references PC_k and that is not an entire <derived column> by:

```
CAST ( NULL AS DTPCk ) AS CNPCk
```

III) Replace each <grouping operation> that contains a <column reference> that references PC_k by the <literal> 1 (one).

B) Let $GSSQL_i$ be:

```
SELECT SLNEW FC WC
GROUP BY ( )
HCNEW
```

i) *QS* is equivalent to:

```
GSSQL_1
UNION ALL
GSSQL_2
UNION ALL
.
.
.
UNION ALL
GSSQL_m
```

Delete Syntax Rule 12).

6. *Rationale: Adapting the Conformance Rules of <group by clause> to the Feature name and broader scope of Feature T431, "Extended grouping capabilities".*

Replace Conformance Rule 1) with:

1) Without Feature T431, "Extended grouping capabilities", conforming SQL language shall not specify ROLLUP, CUBE, GROUPING SETS, or <grand total>.

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

Replace Conformance Rule 3) with:

3) Without Feature S024, “Enhanced structured types”, a grouping column shall not be of an ST-ordered declared type.

8. *Rationale: Adapting the Conformance Rules of <group by clause> to the Feature name and broader scope of Feature T431, "Extended grouping capabilities".*

Add the following Conformance Rule:

3.1) Without Feature T431, "Extended grouping capabilities", an <ordinary grouping set> shall be a <grouping column reference>.

7.10 <having clause>

1. *Rationale: Supply missing application of functional dependencies to <having clause>.*

Add the following Conformance Rules:

1) Without Feature T301, "Functional dependencies", each column reference directly contained in the <search condition> shall be one of the following:

   a) an unambiguous reference to a grouping column of *T*, or

   b) an outer reference.

2) Without Feature T301, "Functional dependencies", each column reference contained in a <subquery> in the <search condition> that references a column of *T* shall be one of the following:

   a) an unambiguous reference to a grouping column of *T*, or

   b) contained in a <set function specification>.

7.11 <query specification>

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

Replace Syntax Rule 5) with:

5) If a <set quantifier> DISTINCT is specified, then no column of *T* shall have a declared type that is LOB-ordered, array-ordered, UDT-EC-ordered, or UDT-NC-ordered.

2. *Rationale: Clarify which <value expression>s are meant.*

Replace Syntax Rule 10) with:

10) Each column reference directly contained in each <value expression> simply contained in the <select list> and each column reference contained in a <set function specification> directly contained in each <value expression> simply contained in the <select list> shall unambiguously reference a column of *T*.

3. *Rationale: Additional circumstance for indeterminacy.*

Replace Syntax Rule 11) d) ii) with:

11) d) ii) The functional dependency $G \mapsto C$, where *G* is the set consisting of the grouping columns of *T*, holds in *T*.

4. *Rationale: Clarify which <value expression>s are meant.*

Replace Syntax Rule 13) with:

13) If *T* is a grouped table, then let *G* be the set consisting of every column referenced by a <column reference> contained in the <group by clause> immediately contained in <table expression>. In each <value expression> contained in the <select list>, each <column reference> that references a column of *T* shall reference some column *C* that is functionally dependent on *G* or shall be contained in a <set function specification>.

5. *Rationale: Clarify which <value expression>s are meant.*

Replace Conformance Rule 3) with:

3) Without Feature T301, “Functional dependencies”, if *T* is a grouped table, then in each <value expression> contained in the <select list>, each <column reference> that references a column of *T* shall reference a grouping column or be specified in a <set function specification>.

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

Replace Conformance Rule 4) with:

4) Without Feature S024, “Enhanced structured types”, if any column in the result of a <query specification> is of an ST-ordered declared type, then DISTINCT shall not be specified or implied.

7.12 <query expression>

1. *Rationale: Provide consistent Syntax Rules for comparison operations.*

Replace Syntax Rule 11) c) with:

11) c) If the set operator is UNION DISTINCT, EXCEPT DISTINCT, EXCEPT ALL, INTERSECT DISTINCT or INTERSECT ALL, then the declared type of each column of *T1* and *T2* shall not be LOB-ordered, array-ordered, UDT-EC-ordered, or UDT-NC-ordered.

Delete Syntax Rules 21) and 22).

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

Replace Conformance Rule 8) with:

8) Without Feature S024, “Enhanced structured types”, if any column in the result of a <query expression> is of ST-ordered declared type, then DISTINCT shall not be specified or implied, and neither INTERSECT nor EXCEPT shall be specified.

8.2 <comparison predicate>

1. *Rationale: Provide consistent Syntax Rules for comparison operations.*

Replace Syntax Rule 5) a) with:

5) a) If the declared type of X_i or Y_i is LOB-ordered, array-ordered, reference-ordered, or UDT-EC-ordered, then <comp op> shall be either <equals operator> or <not equals operator>.

2. *Rationale: Comparison forms and categories do not have to be the same throughout a subtype family.*

Replace NOTE 126 with:

NOTE 126 — The comparison form and comparison categories included in the user-defined type descriptors of both *UDT1* and *UDT2* are constrained to be the same, and the same as those of all their supertypes. If the comparison category is either STATE or RELATIVE, then the comparison functions of *UDT1* and *UDT2* are constrained to be equivalent; if the comparison category is MAP, they are not constrained to be equivalent.

3. *Rationale: SR 5) b) i) 2) was incorrectly nested.*

Delete Syntax Rule 5) b) i) 2).

Delete Syntax Rule 5) b) i) 3) and the accompanying NOTE.

Add Syntax Rule 5) b) i.1):

5) b) i.1) If the declared types of X_i and Y_i are reference types, then the referenced type of the declared type of X_i and the referenced type of the declared type of Y_i shall have a common supertype.

4. *Rationale: clarify that the choice of an ordering function is not determined by subject routine determination rules.*

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

1) b) iii) 1) If the comparison category of UDT_x is MAP, then let *HF1* be the <routine name> with explicit <schema name> of the comparison function of UDT_x and let *HF2* be the <routine name> with explicit <schema name> of the comparison function of UDT_y. If *HF1* is an SQL-invoked function that is a method, then let *HFX* be *X.HF1*; otherwise, let *HFX* be *HF1* (*X*). If *HF2* is an SQL-invoked function that is a method, then let *HFY* be *Y.HF2*; otherwise, let *HFY* be *HF2* (*Y*).

X <comp op> *Y*

has the same result as

HFX <comp op> *HFY*

Replace General Rule 1) b) iii) 2) A) with:

1) b) iii) 2) A) Let *RF* be the <routine name> with explicit <schema name> of the comparison function of UDT_x.

5. *Rationale: Editorial - typographical error.*

Replace General Rule 4) with:

4) The comparison of two binary string values, *X* and *Y*, is determined by comparison of their octets with the same ordinal position. If X_i and Y_i are the values of the *i*-th octets of *X* and *Y*, respectively, and if L_x is the length in octets of *X* and L_y is the length in octets of *Y*, then *X* is equal to *Y* if and only if $L_x = L_y$ and if $X_i = Y_i$ for all *i*.

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

Replace Conformance Rule 1) with:

1) Without Feature T042, “Extended LOB data type support”, no field of the declared row type of a <row value expression> that is simply contained in a <comparison predicate> shall be of a LOB-ordered declared type.

Replace Conformance Rule 2) with:

2) Without Feature S024, “Enhanced structured types”, no field of the declared type of a <row value expression> that is simply contained in a <comparison predicate> shall be of a declared type that is ST-ordered.

8.3 <between predicate>

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

Replace Conformance Rule 2) with:

2) Without Feature S024, “Enhanced structured types”, no field of the declared type of a <row value expression> that is simply contained in a <between predicate> shall be of a declared type that is ST-ordered.

8.4 <in predicate>

1. *Rationale: Resolve an ambiguous case, in which <in predicate value> might be interpreted either as a <table subquery> or as a <scalar subquery>, by choosing the former interpretation.*

Add the following Syntax Rule:

0.1) If <in value list> consists of a single <row value expression>, it shall not be a <scalar subquery>.

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

Replace Conformance Rules 2), 3) and 4) with:

2) Without Feature T042, “Extended LOB data type support”, no field of the declared row type of a <row value expression> or a <table subquery> contained in an <in predicate> shall be of a LOB-ordered declared type.

3) Without Feature S024, “Enhanced structured types”, no field of the declared row type of a <row value expression> or a <table subquery> that is simply contained in an <in predicate> shall be of an ST-ordered declared type.

8.6 <similar predicate>

1. *Rational: “P” is referenced in General Rules 5) a), 6) e), 6) g), and 6) i) but is not defined.*

Replace General Rule 5) a) with:

5) a) If the enumeration is specified in the form “<character specifier> <minus sign> <character specifier>”, then the set of all characters that collate greater than or equal to the character represented by the left <character specifier> and less than or equal to the character represented by the right <character specifier>, according to the collating sequence of the pattern *PCV*.

Replace General Rule 6) e) with:

6) e) L (<percent>)

is the set of all strings of any length (zero or more) from the character set of the pattern *PCV*.

Replace General Rule 6) g) with:

6) g) L (<underscore>)

is the set of all strings of length 1 (one) from the character set of the pattern *PCV*.

Replace General Rule 6) i) with:

6) i) L (<left bracket> <circumflex> <character enumeration> <right bracket>)

is the set of all strings of length 1 (one) with characters from the character set of the pattern *PCV* that are not contained in the set of characters in the <character enumeration>.

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

Add the following Conformance Rule:

2) Without Feature T042, "Extended LOB data type support", a <character value expression> contained in a <similar predicate> shall not be of declared type CHARACTER LARGE OBJECT.

8.8 <quantified comparison predicate>

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

Replace the Conformance Rules 1) and 2) with:

1) Without Feature T042, "Extended LOB data type support", no field of the declared row type of a <row value expression> or a <table subquery> contained in a <quantified comparison predicate> shall be of a LOB-ordered declared type.

2) Without Feature S024, "Enhanced structured types", no field of the declared row type of a <row value expression> shall be of an ST-ordered declared type.

8.10 <unique predicate>

1. *Rationale: Clarify that unordered types do not support <unique predicate>.*

In Syntax Rules, replace "None" with:

1) Each column of user-defined type in the result of the <table subquery> shall have a comparison type.

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

Add the following Syntax Rule:

2) The declared type of a column identified by a <column name> in the <unique column list> shall not be LOB-ordered, array-ordered, UDT-EC-ordered, or UDT-NC-ordered.

3. *Rationale: Use the notion of distinct rather than equality*

Replace General Rule 2) with:

2) If there are no two rows in *T* such that the value of each column in one row is non-null and is not distinct from the value of the corresponding column in the other row, then the result of the <unique predicate> is *true*; otherwise, the result of the <unique predicate> is *false*.

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

Replace Conformance Rule 2) with:

2) Without Feature S024, "Enhanced structured types", no column of the result of the <table subquery> shall be of ST-ordered declared type.

8.11 <match predicate>

1. Rationale: Editorial - typographical error.

Replace Syntax Rule 1) with:

1) The row type of the <row value expression> and the row type of the <table subquery> shall be comparable.

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

Replace Conformance Rule 2) with:

2) Without Feature S024, “Enhanced structured types”, no field of the declared row type of the <row value expression> shall be of an ST-ordered declared type and no column of the result of the <table subquery> shall be of an ST-ordered declared type.

3) Without Feature T042, “Extended LOB data type support”, no field of the declared row type of the <row value expression> shall be of a LOB-ordered declared type.

8.12 <overlaps predicate>

1. Rationale: Create a separate Feature for <overlaps predicate>.

Replace Conformance Rule 1) with:

1) Without Feature F053, “OVERLAPS predicate”, conforming SQL language shall not contain any <overlaps predicate>.

8.13 <distinct predicate>

1. Rationale: Use the notion of distinct.

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

1) The result of <distinct predicate> is *true* if the value of <row value expression 3> is distinct from the value of <row value expression 4>; otherwise the result is *false*.
NOTE 136.1 — “distinct” is defined in Subclause 3.1.5, “Definitions provided in Part 2”.

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

Replace Conformance Rule 2) with:

2) Without Feature S024, “Enhanced structured types”, no field of the declared row type of either <row value expression> shall be of an ST-ordered declared type.

3) Without Feature T042, “Extended LOB data type support”, no field of the declared row type of either <row value expression> shall be of a LOB-ordered declared type.

9.0 Determination of identical values

1. *Rationale: Specify the meaning of "identical"*
   Source: BHX-049R3

Insert the following subclause:

9.0 Determination of identical values

Function

Determine whether two instances of values are identical, that is to say are occurrences of the same value.

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let *V1* and *V2* be two values specified in an application of this subclause.

   NOTE 137.1 — This subclause is invoked implicitly wherever the word *identical* is used of two values.

2) Case:

   a) If *V1* and *V2* are both null, then *V1* is identical to *V2*.

   b) If *V1* is null and *V2* is not null or *V1* is not null and *V2* is null, then *V1* is not identical to *V2*.

   c) If *V1* and *V2* are of comparable predefined types, then

      Case:

      i) If *V1* and *V2* are character strings then

         Case:

         1) If *V1* is not distinct from *V2* and CAST (*V1* AS BIT (BIT_LENGTH (*V1*)) is not distinct from CAST (*V2* AS BIT (BIT_LENGTH (*V2*)), then *V1* is identical to *V2*.

         2) Otherwise, *V1* is not identical to *V2*.

      ii) If *V1* and *V2* are time with time zone or timestamp with time zone and are not distinct and their time zone fields are not distinct, then *V1* is identical to *V2*.

      iii) Otherwise, *V1* is identical to *V2* if and only if *V1* is not distinct from *V2*.

   d) If *V1* and *V2* are of constructed types, then

      Case:

i) If *V1* and *V2* are rows and their respective fields are identical, then *V1* is identical to *V2*.

ii) If *V1* and *V2* are arrays and have the same cardinality and elements in the same ordinal position in the two arrays are identical, then *V1* is identical to *V2*.

iii) If *V1* and *V2* are references and *V1* is not distinct from *V2*, then *V1* is identical to *V2*.

iv) Otherwise, *V1* is not identical to *V2*.

e) If *V1* and *V2* are of the same most specific type, *MST*, and *MST* is a user-defined type, then:

Case:

i) If *MST* is a distinct type, whose source type is *SDT* and the results of *SDT* (*V1*) and *SDT* (*V2*) are identical, then *V1* is identical to *V2*.

ii) If *MST* is a structured type and, for every observer function *O* defined for *MST*, the results of the invocations *O* (*V1*) and *O* (*V2*) are identical, then *V1* is identical to *V2*.

iii) Otherwise, *V1* is not identical to *V2*.

f) Otherwise, *V1* is not identical to *V2*.

Conformance Rules

None.

9.1 Retrieval assignment

1. *Rationale: Restore the facility of assignments involving character strings if there is a user-defined cast function.*

Replace Syntax Rule 3) b) with:

3) b) Otherwise, the declared type of *V* shall be a mutually assignable character string type or an appropriate user-defined cast function *UDCF* shall be available to assign *V* to *T*.

9.3 Data types of results of aggregations

1. *Rationale: Editorial.*

Replace Syntax Rule 3) i) with:

i) If any data type in *DTS* is a row type, then each data type in *DTS* shall be a row type with the same degree and the data type of each field in the same ordinal position of every row type shall be comparable. The result data type is a row defined by an ordered sequence of (<field name>, data type) pairs FD_i , where data type is the data type resulting from the application of this Subclause to the set of data types of fields in the same ordinal position as FD_i in every row type in *DTS* and <field name> is determined as follows:

Case:

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

ii) Otherwise, the <field name> in FD_i is implementation-dependent.

10.4 <routine invocation>

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

Replace Syntax Rule 4) with:

4) Case:

a) If *RI* is immediately contained in a <call statement>, then an SQL-invoked routine *R* is a *possibly candidate routine* for *RI* (henceforth, simply “possibly candidate routine”) if *R* is an SQL-invoked procedure and the <qualified identifier> of the <routine name> of *R* is equivalent to the <qualified identifier> of *RN*.

b) If *RI* is immediately contained in a <method selection>, then an SQL-invoked routine *R* is a *possibly candidate routine* for *RI* if *R* is an instance SQL-invoked method and the <qualified identifier> of the <routine name> of *R* is equivalent to the <qualified identifier> of *RN*.

c) If *RI* is immediately contained in a <constructor method selection>, then an SQL-invoked routine *R* is a *possibly candidate routine* for *RI* if *R* is an SQL-invoked constructor method and the <qualified identifier> of the <routine name> of *R* is equivalent to the <qualified identifier> of *RN*.

d) If *RI* is immediately contained in a <static method selection>, then an SQL-invoked routine *R* is a possibly candidate routine for *RI* if *R* is a static SQL-invoked method and the <qualified identifier> of the <routine name> of *R* is equivalent to the <qualified identifier> of *RN* and *MSD* is included in a user-defined type descriptor for *UDTSM* or for some supertype of *UDTSM*.

e) Otherwise, an SQL-invoked routine *R* is a *possibly candidate routine* for *RI* if *R* is an SQL-invoked function that is not an SQL-invoked method and the <qualified identifier> of the <routine name> of *R* is equivalent to the <qualified identifier> of *RN*.

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

Replace Syntax Rule 6) b) iii) 2) A) with:

6) b) iii) 2) A) If the declared type of P_i is a user-defined type, then:

I) Let ST_i be the set of subtypes of the declared type of A_i .

II) The type designator of the declared type of P_i shall be in the type precedence list of the data type of some type in ST_i .

NOTE 149 — “type precedence list” is defined in Subclause 9.5, “Type precedence list determination”.

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

Replace Syntax Rule 7) b) i) 1) A) with:

7) b) i) 1) A) If *RI* is immediately contained in a <method selection>, <static method selection> or a <constructor method selection>, then let *DP* be *TP*.

4. *Rationale: Editorial.*

Replace Syntax Rule 7) b) i) 2) C) II) with:

7) b) i) 2) C) II) If the routine descriptor of *R1* includes a STATIC indication, then there is no other invocable routine *R2* for which the user-defined type described by the user-defined type descriptor that includes the routine descriptor of *R2* is a subtype of the user-defined type described by the user-defined type descriptor that includes the routine descriptor of *R1*.

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

Replace Syntax Rule 8) b) i) 1) A) with:

8) b) i) 1) A) If *RI* is immediately contained in a <method selection>, <static method selection> or a <constructor method selection>, then let *DP* be *TP*.

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

Replace Syntax Rule 9) with:

9) If *SR* is a constructor function, then *RI* shall be simply contained in a <new invocation>.

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

Replace General Rule 2) b) iii) 1) A) IV) with:

2) b) iii) 1) A) IV) The declared type of PM_j ,$2 \leq j \leq N$, is compatible with the declared type of PSR_j.
NOTE 152 — *SR* is an element of the set *SM*.

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

Replace General Rule 6) e) with:

6) e) Case:

i) If the SQL-data access indication of *CSC* specifies possibly contains SQL and *R* possibly reads SQL-data, or *R* possibly modifies SQL-data, then

1) If *R* is an external routine, then an exception condition is raised: *external routine exception — reading SQL-data not permitted.*

2) Otherwise, an exception condition is raised: *SQL routine exception — reading SQL-data not permitted.*

ii) If the SQL-data access indication of *CSC* specifies possibly reads SQL-data and *R* possibly modifies SQL-data, then:

1) If *R* is an external routine, then an exception condition is raised: *external routine exception — modifying SQL-data not permitted.*

2) Otherwise, an exception condition is raised: *SQL routine exception — modifying SQL-data not permitted.*

Add the following General Rule:

6) e.1) Case:

i) If *R* does not possibly contain SQL, then set the SQL-data access indication in the routine execution context of *RSC* to does not possibly contain SQL.

ii) If *R* possibly contains SQL, then set the SQL-data access indication in the routine execution context of *RSC* to possibly contains SQL.

iii) If *R* possibly reads SQL-data, then set the SQL-data access indication in the routine execution context of *RSC* to possibly reads SQL-data.

iv) If *R* possibly modifies SQL-data, then set the SQL-data access indication in the routine execution context of *RSC* to possibly modifies SQL-data.

Delete General Rule 7).

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

Replace General Rule 8) a) with:

8) a) If *R* is an SQL-invoked method whose routine descriptor does not include a STATIC indication and if CPV_1 is the null value, then:

i) Let *RV* be the null value.

ii) If *R* is a mutator, then an exception condition is raised: *data exception — null instance used in mutator function*.

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

Replace General Rule 8) c) iv) with:

8) c) iv) If the SQL-data access indication of *RSC* specifies possibly contains SQL, and, before the completion of the execution of the <SQL routine body> of *R*, an attempt is made to execute an SQL-statement that possibly reads SQL-data, or an attempt is made to execute an SQL-statement that possibly modifies SQL-data, then an exception condition is raised: *SQL routine exception — reading SQL-data not permitted*.

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

Replace General Rule 8) c) v) with:

8) c) v) If the SQL-data access indication of *RSC* specifies possibly reads SQL-data, and, before the completion of the execution of the <SQL routine body> of *R*, an attempt is made to execute an SQL statement that possibly modifies SQL-data, then an exception condition is raised: *SQL routine exception — modifying SQL-data not permitted*.

12. *Rationale: Fix the subscripts in the references to CPV in General Rule 9) b) i) 1) C) and General Rule 9) b) i) 2) B).*

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

9) b) i) 1) C) For *i* ranging from (*PN*+1)+1 to (*PN*+1)+*N*, the *i*-th entry in *ESPL* is the SQL indicator argument corresponding to $\text{CPV}_{i\text{-}(PN+1)}$.

13. *Rationale: Fix the subscripts in the references to CPV in General Rule 9) b) i) 1) C) and General Rule 9) b) i) 2) B).*

Replace General Rule 9) b) i) 2) B) with:

9) b) i) 2) B) For *i* ranging from *PN*+1 to *PN*+*N*, the *i*-th entry in *ESPL* is the SQL indicator argument corresponding to $CPV_{i\text{-}PN}$.

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

Add the following General Rule:

9) f) iii) 2.1) If the SQL-data access indication of *RSC* specifies does not possibly contain SQL, and before the completion of any execution of *P*, an attempt is made to execute an SQL-statement, then an exception condition is raised: *external routine exception — containing SQL not permitted*.

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

Delete General Rule 9) f) iii) 4).

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

Replace General Rule 9) f) iii) 5) with:

9) f) iii) 5) If the SQL-data access indication of *RSC* specifies possibly contains SQL, and, before the completion of any execution of *P*, an attempt is made to execute an SQL-statement that possibly reads SQL-data, or an attempt is made to execute an SQL-statement that possibly modifies SQL-data, then an exception condition is raised: *external routine exception — reading SQL-data not permitted*.

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

Replace General Rule 9) f) iii) 6) with:

9) f) iii) 6) If the SQL-data access indication of *RSC* specifies possibly reads SQL-data, and, before the completion of any execution of *P*, an attempt is made to execute an SQL-statement that possibly modifies SQL-data, then an exception condition is raised: *external routine exception — modifying SQL-data not permitted*.

18. *Rationale: Check for the most specific type of the returned value from type-preserving functions.*

Replace General Rule 10) with:

10) Case:

a) If *R* is an SQL-invoked function, then:

i) If *R* is type preserving, then:

1) Let *MAT* be the most specific type of the value of the argument substituted for the result SQL parameter of *R*.

2) If the most specific type of *RV* is not compatible with *MAT*, then an exception condition is raised: *data exception — most specific type mismatch*.

ii) Let *ERDT* be the effective returns data type of the <routine invocation>.

iii) Let the result of the <routine invocation> be the result of assigning *RV* to a target of declared type *ERDT* according to the rules of Subclause 9.2, “Store assignment”.

b) Otherwise, for each SQL parameter P_i of *R* that is an output SQL parameter or both an input SQL parameter and an output SQL parameter, let TS_i be the <target specification> of the corresponding <SQL argument> A_i.

Case:

i) If TS_i is a <host parameter specification>, then CPV_i is assigned to TS_i according to the rules of Subclause 9.1, “Retrieval assignment”.

ii) If TS_i is the <SQL parameter name> of an SQL parameter of an SQL-invoked routine, then CPV_i is assigned to TS_i according to the rules of Subclause 9.2, “Store assignment”.

19. Rationale: Ensure that the complete result set is available for scrollable cursors

Replace General Rule11) e) with:

11) e) Case:

i) If $FRCN_i$ is a scrollable cursor, then let NXT_i, $1 \leq i \leq RTN$, be 1 (one).

ii) Otherwise, let NXT_i,$1 \leq i \leq RTN$, be the ordinal number of the row of RS_i that would be retrieved if the following SQL-statement were executed:

```
FETCH NEXT FROM FRCNi INTO . . .
```

10.5 <privileges>

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

Replace Conformance Rule 9) with:

9) Without Feature F461, “Named character sets”, in conforming SQL language, an <object name> shall not specify CHARACTER SET.

10.6 <character set specification>

1. *Rationale: Align Syntax Rules with the Concepts .*

   3) The <standard character set name>s shall include: SQL_CHARACTER and all those specified in Subclause 4.2.4, “Named character sets”as defined by other standards.

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

Replace Conformance Rule 1) with:

   1) Without Feature F461, “Named character sets”, conforming SQL language shall not contain a <character set specification>.

10.7 <specific routine designator>

1. *Rationale: Use the correct BNF (<user-defined type name> instead of <user-defined type>).*

In the Format, replace the production for <specific routine designator> with:

```
<specific routine designator> ::=
       SPECIFIC <routine type> <specific name>
     | <routine type> <member name> [ FOR <user-defined type name> ]
```

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

In the Format, replace the production for <routine type> with:

```
<routine type> ::=
       ROUTINE
     | FUNCTION
     | PROCEDURE
     | [ INSTANCE | STATIC | CONSTRUCTOR ] METHOD
```

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

Replace Syntax Rule 2) with:

   2) If <routine type> specifies METHOD and neither INSTANCE nor STATIC nor CONSTRUCTOR is specified, then INSTANCE is implicit.

4. *Rationale: use the correct BNF (<user-defined type name> instead of <user-defined type>) and correct namespace problems associated with methods used to initialize newly-constructed structured type values. Replace incorrectly used concept of data type identity of routine parameters with the concept of data type compatibility.*

Replace Syntax Rule 3) with:

   3) If a <member name> *MN* is specified, then:

      a) If <user-defined type name> is specified, then <routine type> shall specify METHOD. If METHOD is specified, then <user-defined type name> shall be specified.

b) Let *RN* be the <schema qualified routine name> of *MN* and let *SCN* be the <schema name> of *MN*.

c) Case:

i) If *MN* contains a <data type list>, then:

1) If <routine type> specifies FUNCTION, then there shall be exactly one SQL-invoked function that is not an SQL-invoked method in the schema identified by *SN* whose <schema qualified routine name> is *RN* such that for all *i* the declared type of its *i*-th SQL parameter is compatible with the *i*-th <data type> in the <data type list> of *MN*. The <specific routine designator> identifies that SQL-invoked function.

2) If <routine type> specifies PROCEDURE, then there shall be exactly one SQL-invoked procedure in the schema identified by *SN* whose <schema qualified routine name> is *RN* such that for all *i* the declared type of its *i*-th SQL parameter is compatible with the *i*-th <data type> in the <data type list> of *MN*. The <specific routine designator> identifies that SQL-invoked function.

3) If <routine type> specifies METHOD, then

Case:

A) If STATIC is specified, then there shall be exactly one static SQL-invoked method of the type identified by <user-defined type name> such that for all *i*, the declared data type of its *i*-th SQL parameter is compatible with the *i*-th <data type> in the <data type list> of *MN*. The <specific routine designator> identifies that static SQL-invoked method.

B) If CONSTRUCTOR is specified, then there shall be exactly one SQL-invoked constructor method of the type identified by <user-defined type name> such that for all *i*, the declared data type of its *i*-th SQL parameter in the unaugmented <SQL parameter declaration list> is compatible with the *i*-th <data type> in the <data type list> of *MN*. The <specific routine designator> identifies that SQL-invoked constructor method.

C) Otherwise, there shall be exactly one instance SQL-invoked method of the type identified by <user-defined type name> such that for all *i*, the declared data type of its *i*-th SQL parameter in the unaugmented <SQL parameter declaration list> is compatible with the *i*-th <data type> in the <data type list> of *MN*. The <specific routine designator> identifies that SQL-invoked method.

4) If <routine type> specifies ROUTINE, then there shall be exactly one SQL-invoked routine in the schema identified by *SN* whose <schema qualified routine name> is *RN* such that for all *i* the declared type of its *i*-th SQL parameter is compatible with the *i*-th <data type> in the <data type list> of *MN*. The <specific routine designator> identifies that SQL-invoked routine.

ii) Otherwise:

1) If <routine type> specifies FUNCTION, then there shall be exactly one SQL-invoked function in the schema identified by *SN* whose <schema qualified routine name> is *RN*. The <specific routine designator> identifies that SQL-invoked function.

2) If <routine type> specifies PROCEDURE, then there shall be exactly one SQL-invoked procedure in the schema identified by *SN* whose <schema qualified routine name> is *RN*. The <specific routine designator> identifies that SQL-invoked procedure.

3) If <routine type> specifies METHOD, then

   Case:

   A) If STATIC is specified, then there shall be exactly one static SQL-invoked method of the user-defined type identified by <user-defined type name>. The <specific routine designator> identifies that static SQL-invoked method.

   B) If CONSTRUCTOR is specified, then there shall be exactly one SQL-invoked constructor method of the user-defined type identified by <user-defined type name>. 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 <user-defined type name> that is not a static SQL-invoked method. The <specific routine designator> identifies that SQL-invoked method.

4) If <routine type> specifies ROUTINE, then there shall be exactly one SQL-invoked routine in the schema identified by *SN* whose <schema qualified routine name> is *RN*. The <specific routine designator> identifies that SQL-invoked routine.

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

Replace Syntax Rule 4) with:

4) If FUNCTION is specified, then the SQL-invoked routine that is identified shall be an SQL-invoked function that is not an SQL-invoked method. If PROCEDURE is specified, then the SQL-invoked routine that is identified shall be an SQL-invoked procedure. If STATIC METHOD is specified, then the SQL-invoked routine that is identified shall be a static SQL-invoked method. If CONSTRUCTOR METHOD is specified, then the SQL-invoked routine shall be an SQL-invoked constructor method. If INSTANCE METHOD is specified or implicit, then the SQL-invoked routine shall be an instance SQL-invoked method. If ROUTINE is specified, then the SQL-invoked routine that is identified is either an SQL-invoked function or an SQL-invoked procedure.

10.12 Execution of triggers

1. *Rationale: Editorial.*

Replace General Rule 4) b) with:

4) b) If the execution of *TSS* is not successful, then an exception condition is raised: *triggered action exception*. The exception information associated with *TSS* is entered into the diagnostics area in a location other than the location corresponding to condition number 1 (one).

10.13 Execution of array-returning functions

1. Rationale: Editorial.

Replace General Rule 6) with:

6) If the call type data item has a value of -1 (indicating "*open call*"), then *P* is executed with a list of *EN* parameters PD_i whose parameter names are PN_i and whose values are set as follows:

a) Depending on whether the language of *R* specifies ADA, C, COBOL, FORTRAN, MUMPS, PASCAL, or PLI, let the *operative data type correspondences* table be Table 18, "Data type correspondences for Ada", Table 19, "Data type correspondences for C", Table 20, "Data type correspondences for COBOL", Table 21, "Data type correspondences for Fortran", Table 22, "Data type correspondences for MUMPS", Table 23, "Data type correspondences for Pascal", or Table 24, "Data type correspondences for PL/I", respectively. Refer to the two columns of the operative data type correspondences table as the "SQL data type" column and the "host data type" column.

b) For *i* varying from 1 (one) to *EN*, the <data type> DT_i of PD_i is the data type listed in the host data type column of the row in the data type correspondences table whose value in the SQL data type column corresponds to the data type of ESP_i.

c) The value of PD_i is set to the value of ESP_i.

Replace General Rule 8) a) with:

8) a) *P* is executed with a list of *EN* parameters PD_i whose parameter names are PN_i and whose values are set as follows:

i) Depending on whether the language of *R* specifies ADA, C, COBOL, FORTRAN, MUMPS, PASCAL, or PLI, let the *operative data type correspondences* table be Table 18, "Data type correspondences for Ada", Table 19, "Data type correspondences for C", Table 20, "Data type correspondences for COBOL", Table 21, "Data type correspondences for Fortran", Table 22, "Data type correspondences for MUMPS", Table 23, "Data type correspondences for Pascal", or Table 24, "Data type correspondences for PL/I", respectively. Refer to the two columns of the operative data type correspondences table as the "SQL data type" column and the "host data type" column.

ii) For *i* varying from 1 (one) to *EN*, the <data type> DT_i of PD_i is the data type listed in the host data type column of the row in the data type correspondences table whose value in the SQL data type column corresponds to the data type of ESP_i.

c) The value of PD_i is set to the value of ESP_i.

Replace General Rule 8) b) iii) with:

8) b) iii) Otherwise, set the value of the call type data item to 1 (one) (indicating *close call*).

Replace General Rule 9) with:

9) If the call type data item has a value of 1 (indicating "*close call*"), then *P* is executed with a list of *EN* parameters PD_i whose parameter names are PN_i and whose values are set as follows:

a) Depending on whether the language of *R* specifies ADA, C, COBOL, FORTRAN, MUMPS, PASCAL, or PLI, let the *operative data type correspondences* table be Table 18, "Data type correspondences for Ada”, Table 19, “Data type correspondences for C”, Table 20, “Data type correspondences for COBOL”, Table 21, “Data type correspondences for Fortran”, Table 22, “Data type correspondences for MUMPS”, Table 23, “Data type correspondences for Pascal”, or Table 24, “Data type correspondences for PL/I”, respectively. Refer to the two columns of the operative data type correspondences table as the “SQL data type” column and the “host data type” column.

b) For *i* varying from 1 (one) to *EN*, the <data type> DT_i of PD_i is the data type listed in the host data type column of the row in the data type correspondences table whose value in the SQL data type column corresponds to the data type of ESP_i.

c) The value of PD_i is set to the value of ESP_i.

11.1 <schema definition>

1. *Rationale: Remove redundant item. <grant role statement> is defined in <grant statement>.*

In the Format, replace the production for <schema element> with:

```
<schema element> ::=
        <table definition>
      | <view definition>
      | <domain definition>
      | <character set definition>
      | <collation definition>
      | <translation definition>
      | <assertion definition>
      | <trigger definition>
      | <user-defined type definition>
      | <schema routine>
      | <grant statement>
      | <role definition>
      | <user-defined cast definition>
      | <user-defined ordering definition>
      | <transform definition>
```

2. *Rationale: Correct the references to schema which do not treat it as a descriptor.*

Replace General Rules 1) with:

1) A <schema definition> creates an SQL-schema *S* in a catalog. *S* includes:

a) A schema name that is equivalent to the explicit or implicit <schema name>.

b) A schema authorization identifier that is equivalent to the explicit or implicit <authorization identifier>.

c) A schema character set name that is equivalent to the explicit or implicit <schema character set specification>.

d) A schema SQL-path that is equivalent to the explicit or implicit <schema path specification>.

e) The descriptor created by every <schema element> of the <schema definition>.

Delete General Rule 4).

3. *Rationale: Remove redundant item. <grant role statement> is defined in <grant statement>.*

Delete Conformance Rule 2).

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

Replace Conformance Rules 8) and 9) with:

8) Without Feature F461, “Named character sets”, a <schema character set specification> shall not be specified.

9) Without Feature F451, “Character set definition”, conforming SQL language shall not contain any <character set definition>.

11.2 <drop schema statement>

1. *Rationale: Correct the references to schema which do not treat it as a descriptor.*

Replace General Rule 13) with:

13) *S* is destroyed.

11.3 <table definition>

1. *Rationale: Incorrect symbol in syntax substitute for <like clause>.*

Replace Syntax Rule 6) b) with:

6) b) Let *nt* be the number of columns in *T1*. Let C_i , 1 (one) ≤ *i* ≤ *nt*, be the columns of *T1*, in the order in which they appear in *T1*, let CN_i be the column name included in the column descriptor of C_i, and let DT_i be the data type included in the column descriptor of C_i. Let CD_1 be:

CN_1 DT_1

If *nt* is greater than 1 (one), then let CD_i ,2 ≤ *i* ≤ *nt*, be:

, CN_i DT_i

The <like clause> is effectively replaced by CD_i, 1 (one) ≤ *i* ≤ *nt*.
NOTE 169 – <column constraint>s are not included in columns; <column constraint>s are effectively transformed to <table constraints>s and are thereby excluded.

2. *Rationale: Incomplete rule—should prohibit <column definition>s as well as <like clause>s.*

Replace Syntax Rule 10) b) with:

10) b) *TEL1* shall not contain a <like clause> or a <column definition>.

3. *Rationale: Align the rules with the BNF in the Format,*

Replace Syntax Rule 16) with:

16) If *TEL1* contains a <column options>, then *TD* shall specify OF <user-defined type> and any <column definition> shall be contained before the first <column options>.

4. *Rationale: Align the rules with the BNF in the Format.*

Replace Access Rule 4) with:

4) If "OF <user-defined type>" is specified, then the applicable privileges of *A* shall include USAGE on *ST*.

5. *Rationale: Editorial - typographical error.*

Replace General Rule 3) with:

3) For each <column options> CO, if CO contains a <scope clause> SC, then let CD be the column descriptor identified by the <column name> specified in CO. The <table name> specified in SC is included in the reference type descriptor that is included in CD.

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

Add the following General Rule:

5) l.1) Case:

   i) If OF <user-defined type> is not specified, then an indication that *T* is insertable-into.

   ii) Otherwise,

   Case:

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

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

7. *Rationale: Correct privileges for <temporary tables>s.*

Replace General Rule 8) with:

8) A set of privilege descriptors is created that define the privileges INSERT, SELECT, UPDATE, DELETE, TRIGGER, and REFERENCES on this table and SELECT, INSERT, UPDATE, and REFERENCES for every <column definition> in the table definition. If OF <user-defined type> is specified, then a table/method privilege descriptor is created on this table for every method of the structured type identified by the <user-defined type> and the table SELECT privilege has the WITH HIERARCHY OPTION. These privileges are grantable. The grantor for each of these privilege descriptors is set to the special grantor value "_SYSTEM". The grantee is <authorization identifier> *A*.

8. *Rationale: Editorial.*

Replace General Rule 10) with:

10) The row type *RT* of the table *T* defined by the <table definition> is the set of pairs (<field name>, <data type>) where <field name> is the name of a column *C* of *T* and <data type> is the declared type of *C*. This set of pairs contains one pair for each column of *T*, in the order of their ordinal position in *T*.

9. *Rationale: Permit tables of structured type without requiring Feature S043, "Enhanced reference types".*

Replace Conformance Rule 5) with:

5) Without Feature S043, "Enhanced reference types", a <self-referencing column specification> shall specify SYSTEM GENERATED.

11.4 <column definition>

1. *Rationale: Provide consistent Syntax Rules for comparison operations.*

Add the following Syntax Rule:

9.1) If <data type> is a <reference type> that identifies a reference type that is LOB-ordered, array-ordered, UDT-EC-ordered or UDT-NC-ordered, then REFERENCES ARE CHECKED shall not be specified.

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

Add the following Conformance Rule:

7) Without Feature S024, "Enhanced structured types", if <data type> is a <reference type> that identifies a reference type that is ST-ordered, then REFERENCES ARE CHECKED shall not be specified.

11.7 <unique constraint definition>

1. *Rationale: Provide consistent Syntax Rules for comparison operations.*

Replace Syntax Rule 1) with:

1) The declared type of no column identified by any <column name> in the <unique column list> shall be LOB-ordered, array-ordered, UDT-EC-ordered, or UDT-NC-ordered.

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

Add the following Conformance Rule:

3) Without Feature S024, "Enhanced structured types", the declared type of no column identified by any <column name> in the <unique column list> shall be of ST-ordered declared type.

11.8 <referential constraint definition>

1. Rationale: Add a missing prefix.

Replace Syntax Rule 3) b) ii) with:

3) b) ii) For a given row in the referenced table, every matching row for that given row that is a matching row only to the given row in the referenced table for the referential constraint is a *unique matching row*. For a given row in the referenced table, a matching row for that given row that is not a unique matching row for that given row for the referential constraint is a *non-unique matching row*.

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

Add the following Syntax Rule:

8.1) No referencing column shall have a declared type that is LOB-ordered, array-ordered, UDT-EC-ordered, or UDT-NC-ordered.

3. Rationale: Provide consistent Conformance Rules for comparison operations.

Add the following Conformance Rule:

6) Without Feature S024, “Enhanced structured types”, no referencing column shall be of ST-ordered declared type.

11.17 <drop column definition>

1. Rationale: Remove unneeded misleading text.

Delete NOTE 193

Delete General Rule 2)

11.21 <view definition>

1. Rationale: Replace incorrect non-terminal.

Replace Syntax Rule 21) e) with:

21) e) If <subview clause> is not specified, then <self-referencing column specification> shall be specified.

Replace Syntax Rule 21) i) with:

21) i) If <self-referencing column specification> is specified, then:

i) <subview clause> shall not be specified.

ii) SYSTEM GENERATED shall not be specified.

iii) Let *RST* be the reference type REF(*ST*).

Case:

1) If USER GENERATED is specified, then:

   A) *RST* shall have a user-defined representation.

   B) Let *m* be 1 (one).

2) If DERIVED is specified, then:

   A) *RST* shall have a derived representation.

   B) Let *m* be 0 (zero).

Replace Syntax Rule 21) s) with:

21) s) If <self-referencing column specification> is specified, then

Case:

i) If *RST* has a user-defined representation, then:

   1) *TQN* shall have a candidate key consisting of a single column *RC*.

   2) Let *SS* be the first <select sublist> in the <select list> of *QS*.

   3) *SS* shall consist of a single <cast specification> *CS* whose leaf column is *RC*.

   NOTE 202 — "Leaf column of a <cast specification>" is defined in Subclause 6.22, "<cast specification>".

   4) The declared type of F_1 shall be REF(*ST*).

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

   1) Let C_i, 1 (one) $\leq i \leq n$, be the columns of *V* that correspond to the attributes of the derived representation of *RST*.

   2) *TQN* shall have a candidate key consisting of some subset of the underlying columns of C_i, 1 (one) $\leq i \leq n$.

11.30 <character set definition>

1. Rationale: Replace undefined non-terminals .

Replace Syntax Rule 3) with:

3) The character set identified by the <character set specification> contained in <character set source> shall have associated with it a privilege descriptor that was effectively defined by the <grant statement>

```
GRANT USAGE ON CHARACTER SET CS TO PUBLIC
```

where *CS* is the <character set name>.

2. *Rationale: Delete unnecessary and incorrect rule .*

Delete General Rule 4).

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

Replace Conformance Rules 1) and 2) with:

1) Without Feature F451, “Character set definition”, conforming SQL language shall not specify any <character set definition>.

2) Without Feature F451, “Character set definition”, and Feature F691, “Collation and translation”, <collation source> shall specify DEFAULT.

11.31 <drop character set statement>

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

Replace Conformance Rule 1) with:

1) Without Feature F451, “Character set definition”, conforming SQL language shall contain no <drop character set statement>.

11.38 <trigger definition>

1. *Rationale: Clarify that the subject table of a <trigger definition> cannot be a declared local temporary table.*

Replace Syntax Rule 5) with:

5) *T* shall be a base table that is not a declared local temporary table.

2. *Rationale: Provide missing Access Rule.*

Add the following Access Rule:

3) If an <old or new values alias> has been specified, then the applicable privileges for *A* shall include SELECT on *T*.

3. *Rationale: Editorial.*

Replace Conformance Rule 1) with:

1) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not contain a <trigger definition>.

11.40 <user-defined type definition>

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

In the Format, replace the production of <user-defined type body> with:

```
<user-defined type body> ::=
     <user-defined type name>
     [ <subtype clause> ]
     [ AS <representation> ]
     [ <instantiable clause> ]
     <finality>
     [ <reference type specification> ]
     [ <ref cast option> ]
     [ <cast option> ]
     [ <method specification list> ]
```

In the Format, replace the production of <user-defined representation> with:

```
<user-defined representation> ::= REF USING <predefined type>
```

2. *Rationale: Correct namespace problems associated with methods used to initialize newly-constructed structured type values. Correct use of <specific name> for method specifications by replacing with <specific method name>.*

In the Format, replace the production for <partial method specification> with:

```
<partial method specification> ::=
     [ INSTANCE | STATIC | CONSTRUCTOR ] METHOD <method name>
     <SQL parameter declaration list>
     <returns clause>
     [ SPECIFIC <specific method name> ]
```

3. *Rationale: <method characteristic> should not include <transform group specification>.*

In the Format, replace the production for <method characteristic> with:

```
<method characteristic> ::=
       <language clause>
     | <parameter style clause>
     | <deterministic characteristic>
     | <SQL-data access indication>
     | <null-call clause>
```

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

In the Format, insert the following production:

```
<specific method name> ::=
     [ <schema name> <period> ] <qualified identifier>
```

5. *Rationale: Remove unneeded rule during process of correcting the oversight in recursive type definition. Move check to <attribute definition>.*

Delete Syntax Rule 6) g) and associated NOTE 225.

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

Replace Syntax Rule 6) i) i) with:

6) i) i) Let *BT* be <predefined type>. *BT* is the *representation type of the referencing type* of *UDT*.

Add the following Syntax Rule:

6) k) iii.1) If the user-defined type descriptor of *SST* indicates that the referencing type of *SST* has a user-defined representation, then let *BT* be the data type described by the data type descriptor of the representation type of the referencing type of *SST* included in the user-defined type descriptor of *SST*.

1) If <cast to ref> is specified, then let *FNREF* be <cast to ref identifier>; otherwise, let *FNREF* be the <qualified identifier> of *UDTN*.

2) Case:

A) If <cast to type> is specified, then let *FNTYP* be <cast to type identifier>.

B) Otherwise, the Syntax Rules of Subclause 9.7, “Type name determination”, are applied to *BT*, yielding an <identifier> *FNTYP*.

Add the following Syntax Rule:

6) k.1) If <ref cast option> is specified, then exactly one of the following shall be true:

i) <user-defined representation> is specified.

ii) <subtype clause> is specified and the user-defined type descriptor of the direct supertype of *UDT* indicates that the referencing type of the direct supertype of *UDT* has a user-defined representation.

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

Replace Syntax Rule 7) a) with:

7) a) Let *M* be the number of <method specification>s MS_i, 1 (one) ≤ i ≤ M, contained in <method specification list>. Let MN_i be the <method name> of MS_i.

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

Delete Syntax Rule 7) b) i)

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

Delete Syntax Rule 7) b) ii)

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

Replace Syntax Rule 7) b) iii) with:

7) b) iii) If MS_i does not specify INSTANCE, CONSTRUCTOR or STATIC, then INSTANCE is implicit.

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

Add the following Syntax Rule:

7) b) iv.1) If MS_i specifies CONSTRUCTOR, then:

1) SELF AS RESULT shall be specified.

2) OVERRIDING shall not be specified.

3) MN_i shall be equivalent to the <qualified identifier> of *UDTN*.

4) The <returns data type> shall specify *UDTN*.

5) MS_i specifies an SQL-invoked constructor method.

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

Replace Syntax Rules 7) b) vi), 7) b) vii), 7) b) viii) with:

7) b) vi) If <specific method name> is not specified, then an implementation-dependent <specific method name> whose <schema name> is equivalent to *SN* is implicit.

7) b) vii) If <specific method name> contains a <schema name>, then that <schema name> shall be equivalent to *SN*. If <specific method name> does not contain a <schema name>, then the <schema name> of *SN* is implicit

7) b) viii) The schema identified by the explicit or implicit <schema name> of the <specific method name> shall not include a routine descriptor whose specific name is equivalent to <specific method name> or a user-defined type descriptor that includes a method specification descriptor whose specific method name is equivalent to <specific method name>.

13. *Rationale: Remove the anomaly in <SQL parameter namer>s.*

Replace Syntax Rule 7) b) ix) with:

7) b) ix) Let PDL_i be the <SQL parameter declaration list> contained in MS_i.

1) No two <SQL parameter name>s contained in PDL_i shall be equivalent.

2) No <SQL parameter name> contained in PDL_i shall be equivalent to SELF.

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

Add the following Syntax Rule:

7) b) xiii) 1.1) Both MS_i and MS_k either specify CONSTRUCTOR, or both do not specify CONSTRUCTOR.

15. *Rationale: <method characteristic> should not include <transform group specification>.*

Replace Syntax Rule 7) b) xv) 1) with:

7) b) xv) 1) The <method characteristics> of MS_i shall contain at most one <language clause>, at most one <parameter style clause>, at most one <deterministic characteristic>, at most one <SQL-data access indication>, and at most one <null-call clause>.

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

Replace Syntax Rule 7) b) xv) 4) with:

7) b) xv) 4) <SQL-data access indication> shall be specified.

17. *Rationale: <method characteristic> should not include <transform group specification>.*

Replace Syntax Rule 7) b) xv) 6) A) III) with:

7) b) xv) 6) A) III) <parameter style clause> shall not be specified.

18. *Rationale: <method characteristic> should not include <transform group specification>.*

Delete Syntax Rule 7) b) xv) 6) B) II)

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

Add the following Syntax Rule:

7) b) xv) 7) D.1) MS_i does not specify CONSTRUCTOR.

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

Replace Syntax Rule 7) b) xv) 10):

7) b) xv) 10) If MS_i does not specify STATIC or CONSTRUCTOR, then there shall be no SQL-invoked function F that satisfies all the following conditions:

A) The routine name of *F* and RN_i have equivalent <qualified identifier>s.

B) If *F* is not a static method, then *F* has AN_i SQL parameters; otherwise, *F* has (AN_i - 1) SQL parameters.

C) The data type being defined is a proper subtype of
Case:

I) If *F* is not a static method, then the declared type of the first SQL parameter of *F*.

II) Otherwise, the user-defined type whose user-defined type descriptor includes the routine descriptor of *F*.

D) The declared type of the *i*-th SQL parameter in *NPLi* ,$2 \leq i \leq AN_i$ is compatible with

Case:

I) If *F* is not a static method, then the declared type of *i*-th SQL parameter of *F*.

II) Otherwise, the declared type of the (*i*-1)-th SQL parameter of *F*.

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

Replace Syntax Rule 7) b) xvi) 1) with:

7) b) xvi) 1) MS_i shall not specify STATIC or CONSTRUCTOR.

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

Add the following Syntax Rule:

7) b) xvi) 3) D.1) The descriptor of COMS shall not include an indication that STATIC or CONSTRUCTOR has been specified.

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

Replace Syntax Rule 7) b) xvi) 5):

7) b) xvi) 5) COMS shall not be the corresponding method specification of a mutator or observer function.

24. *Rationale: For overriding method specifications, provide missing syntax rules to enforce proper <locator indication> in <SQL parameter declaration list>s. Also provide missing syntax rules to enforce that the parameter names are properly aligned with the parameter names of the corresponding parameters of the corresponding original method specification.*

Replace Syntax Rule 7) b) xvi) 6) with:

7) b) xvi) 6) For *j* ranging from 1 (one) to N_i, all of the following shall be true:

A) If the $POVMS_j$ contains an <SQL parameter name> *PNM1*, then $PCOMS_j$ contains an <SQL parameter name> that is equivalent to *PNM1*.

B) If $PCOMS_j$ contains an <SQL parameter name> *PNM2*, then $POVMS_j$ contains an <SQL parameter name> that is equivalent to *PNM2*.

C) If the $POVMS_j$ contains a <locator indication>, then $PCOMS_j$ contains a <locator indication>.

D) If *PCOMS*$_j$ contains a <locator indication>, then *POVMS*$_j$ contains a <locator indication>.

25. *Rationale: Correct the first <user-defined cast definition> and the <transform definition>.*

Replace General Rule 1) b) with:

1) b) The following SQL-statements are executed without further Access Rule checking:

```
CREATE FUNCTION SN.FNUDT ( SDTP SDT )
RETURNS UDTN
LANGUAGE SQL
DETERMINISTIC
RETURN RV1

CREATE METHOD SN.FNSDT ( )
RETURNS SDT
FOR UDTN
LANGUAGE SQL
DETERMINISTIC
RETURN RV2

CREATE CAST ( UDTN AS SDT )
WITH METHOD SN.FNSDT ( ) FOR UDTN
AS ASSIGNMENT

CREATE CAST ( SDT AS UDTN )
WITH FUNCTION SN.FNUDT ( SDT )
AS ASSIGNMENT

CREATE TRANSFORM FOR UDTN
FNUDT ( FROM SQL WITH METHOD SN.FNSDT ( ) FOR UDTN,
TO SQL WITH FUNCTION SN.FNUDT ( SDT ) )
```

where: *SN* is the explicit or implicit <schema name> of *UDTN*; *RV1* is an implementation-dependent <value expression> such that for every invocation of *SN.FNUDT* with argument value *AV1*, *RV1* evaluates to the representation of *AV1* in the data type identified by *UDTN*; *RV2* is an implementation-dependent <value expression> such that for every invocation of *SN.FNSDT* with argument value *AV2*, *RV2* evaluates to the representation of *AV2* in the data type *SDT*, and *SDTP* is an <SQL parameter name> arbitrarily chosen.

26. *Rationale: Provide appropriate declarations for the system-generated ordering for distinct type whose source type is LOB*

Add the following General Rule:

1) c) Case:

i) If *SDT* is not a large object type, then the following SQL-statement is executed without further Access Rule checking:

```
CREATE ORDERING FOR UDTN
ORDER FULL BY MAP
WITH METHOD     FNSDT ( ) FOR UDTN
```

ii) If *SDT* is a large object type, and the SQL-implementation supports Feature T042, “Extended LOB data type support”, then the following SQL-statement is executed without further Access Rule checking:

```
CREATE ORDERING FOR UDTN
ORDER EQUALS ONLY BY MAP
WITH METHOD FNSDT ( ) FOR UDTN
```

NOTE 228 — If *SDT* is a large object type, and the SQL-implementation does not support Feature T042, “Extended LOB data type support”, then no ordering for *UDTN* is created.

27. Rationale: Editorial.

Replace General Rule 2) b) with:

2) b) If INSTANTIABLE is specified, then let *V* be a value of the most specific type *UDT* such that, for every attribute *ATT* of *UDT*, invocation of the corresponding observer function on *V* yields the default value for *ATT*. The following <SQL-invoked routine> is effectively executed:

```
CREATE FUNCTION UDTN ( ) RETURNS UDTN
RETURN V
```

This SQL-invoked function is the *constructor function* for *UDT*.

28. Rationale: Correct the <SQL-invoked routine> statement that specifies SN.FNTYP and correct the last <user-defined cast definition>. Correct the specification of implicit generation of cast functions when reference values are generated via <user-defined representation>.

Replace General Rule 2) c) with:

2) c) If <user-defined representation> is specified or if <subtype clause> is specified and the user-defined type descriptor of the direct supertype of *UDT* indicates that the referencing type of the direct supertype of *UDT* has a user-defined representation, then the following SQL-statements are executed without further Access Rule checking:

```
CREATE FUNCTION SN.FNREF ( BTP BT )
  RETURNS REF ( UDTN )
  LANGUAGE SQL
  DETERMINISTIC
  STATIC DISPATCH
  RETURN RV1

CREATE FUNCTION SN.FNTYP ( UDTNP REF( UDTN ))
  RETURNS BT
  LANGUAGE SQL
  DETERMINISTIC
  STATIC DISPATCH
  RETURN RV2

CREATE CAST ( BT AS REF ( UDTN ) )
  WITH FUNCTION SN.FNREF ( BT )

CREATE CAST ( REF( UDTN ) AS BT )
  WITH FUNCTION SN.FNTYP ( REF( UDTN ) )
```

where: *SN* is the explicit or implicit <schema name> of *UDTN*; *RV1* is an implementation-dependent <value expression> such that for every invocation of *SN.FNREF* with argument value *AV1*, *RV1* evaluates to the representation of *AV1* in the data type identified by REF (*UDTN*); *RV2* is an implementation-dependent <value expression> such that for every invocation of *SN.FNTYP* with argument value *AV2*, *RV2* evaluates to the representation of *AV2* in the data type *BT*; and *UDTNP* is an <SQL parameter name> arbitrarily chosen.

29. Rationale: Redundant subrule.

Replace General Rule 5) e) with:

5) e) If *UDT* is a distinct type, then the data type descriptor of *SDT*.

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

Replace General Rule 5) f) v) with:

5) f) v) Case:

1) If <user-defined representation> is specified, then an indication that the referencing type of *UDT* has a user-defined representation and the data type descriptor of the representation type of the referencing type of *UDT*.

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

3) Otherwise, an indication that the referencing type of *UDT* has a system-defined representation.

31. Rationale: Ordering form and category for a structured type.

Add the following General Rules:

5) f) v.1) The ordering form NONE.

v.2) The ordering category STATE.

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

Add the following General Rule:

5) f) v.3) If <subtype clause> is specified, then let *SUDT* be the direct supertype of *UDT* and let *DSUDT* be the user-defined type descriptor of *SUDT*. Let *RUDT* be the referencing type of *UDT* and let *RSUDT* be the referencing type of *SUDT*.

Case:

1) If *DSUDT* indicates that *RSUDT* has a user-defined representation, then an indication that *RUDT* has a user-defined representation and the data type descriptor of the representation type of *RSUDT* included in *DSUDT*.

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

3) If *DSUDT* indicates that *RSUDT* has a system-defined representation, then an indication that *RUDT* has a system-defined representation.

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

Replace General Rule 5) g) ii) with:

5) g) ii) An indication of whether STATIC or CONSTRUCTOR is specified.

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

Replace General Rule 5) g) iv) with:

5) g) iv) The <specific method name> of *ORMS*.

35. *Rationale: <method characteristic> should not include <transform group specification>.*

Delete General Rule 5) g) x)

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

Replace General Rule 5) g) xii) with:

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

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

Replace General Rule 5) h) iii) with:

5) g) iii) The <specific method name> of *OVMS*.

38. *Rationale: <method characteristic> should not include <transform group specification>.*

Delete General Rule 5) h) ix)

39. *Rationale: Clarify that Feature S023 also comprises <method specification list>.*

Add the following Conformance Rule:

8.1) Without Feature S023, “Basic structured types”, conforming SQL language shall not specify <method specification list>.

11.41 <attribute definition>

1. *Rationale: Correct oversight in rules for recursive type definition.*

Add the following Syntax Rule:

6.1) *DT* shall not be based on *UDT*.

NOTE 229.1 — The notion of one data type being based on another data type is defined in Subclause 4.1, “Data types”.

11.43 <add attribute definition>

1. *Rationale: Correct oversight in rules for recursive type definition.*

Replace Syntax Rule 3) with:

3) The declared type of a column of a base table shall not be SPRD, SBRD, SPAD, or SBAD.

2. *Rationale: Correct oversight in rules for recursive type definition.*

Replace Syntax Rule 4) with:

4) The declared type of a column of a base table shall not be based on D.

11.44 <drop attribute definition>

1. *Rationale: Editorial.*

Replace Syntax Rule 4) with:

4) Let *SPD* be any supertype of *D*. Let *SBD* be any subtype of *D*. Let *RD* be the reference type whose referenced type is *D*. Let *SPRD* be any supertype of *RD*. Let *SBRD* be any subtype of *RD*. Let *AD* be the array type whose element type is *D*. Let *SPAD* be any array type whose element type is *SPD* or *SPRD*. Let *SBAD* be any array type whose element type is *SBD* or *SBRD*.

2. *Rationale: Correct oversight in rules for recursive type definition.*

Replace Syntax Rule 5) with:

5) The declared type of any column of any base table shall not be *SPRD*, *SBRD*, *SPAD*, or *SBAD*.

3. *Rationale: Correct oversight in rules for recursive type definition.*

Replace Syntax Rule 6) with:

6) The declared type of any column of any base table shall not be based on *D*.

4. *Rationale: Dependency on a user-defined ordering requires only the existence of the ordering..*

Replace Syntax Rule 8) c) with:

8) c) *R1* or *R2* shall not be the ordering function in the descriptor of any user-defined type..

11.45 <add original method specification>

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

Delete Syntax Rule 4)

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

Delete Syntax Rule 5)

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

Replace Syntax Rule 6) with:

6) If *PORMS* does not specify INSTANCE, CONSTRUCTOR or STATIC, then INSTANCE is implicit.

6.1) If *PORMS* specifies CONSTRUCTOR, then:

a) SELF AS RESULT shall be specified.

b) *MN* shall be equivalent to the <qualified identifier> of *DN*.

c) The <returns data type> shall specify *DN*.

d) *PORMS* specifies an SQL-invoked constructor method.

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

Replace Syntax Rule 9) with:

9) Case:

a) If *PORMS* does not specify <specific method name>, then an implementation-dependent <specific method name> is implicit whose <schema name> is equivalent to *SN*.

b) Otherwise:

Case:

i) If <specific method name> contains a <schema name>, then that <schema name> shall be equivalent to *SN*.

ii) Otherwise, the <schema name> *SN* is implicit.

The schema identified by the explicit or implicit <schema name> of the <specific method name> shall not include a routine descriptor whose specific name is equivalent to <specific method name> or a user-defined type descriptor that includes a method specification descriptor whose specific method name is equivalent to <specific method name>.

5. *Rationale: <method characteristic> should not include <transform group specification>.*

Replace Syntax Rule 10) with:

10) *MCH* shall contain at most one <language clause>, at most one <parameter style clause>, at most one <deterministic characteristic>, at most one <SQL-data access indication>, and at most one <null-call clause>.

a) If <language clause> is not specified in *MCH*, then LANGUAGE SQL is implicit.

b) Case:

i) If LANGUAGE SQL is specified or implied, then:

1) <parameter style clause> shall not be specified.

2) <SQL-data access indication> shall not specify NO SQL.

3) Every <SQL parameter declaration> contained in <SQL parameter declaration list> shall contain an <SQL parameter name>.

4) The <returns clause> shall not specify a <result cast>.

ii) Otherwise:

1) If <parameter style clause> is not specified, then PARAMETER STYLE SQL is implicit.

2) If a <result cast> is specified, then let *V* be some value of the <data type> specified in the <result cast> and let *RT* be the <returns data type>. The following shall be valid according to the Syntax Rules of Subclause 6.22, "<cast specification>":
   CAST (*V* AS *RT*)

3) If <result cast from type> *RCT* simply contains <locator indication>, then *RCT* shall be either binary large object type, character large object type, array type, or user-defined type.

c) If <deterministic characteristic> is not specified in *MCH*, then NOT DETERMINISTIC is implicit.

d) If <SQL-data access indication> is not specified, then CONTAINS SQL is implicit.

e) If <null call clause> is not specified in *MCH*, then CALLED ON NULL INPUT is implicit.

6. *Rationale: Remove the anomaly in <SQL parameter name>s.*

Replace Syntax Rule 11) with:

11) No two <SQL parameter name>s contained in *MPDL* shall be equivalent.

11.1) No <SQL parameter name> contained in *MPDL* shall be equivalent to SELF.

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

Replace Syntax Rule 15) with:

15) Case:

a) If *ORMS* specifies CONSTRUCTOR, then let a *conflicting method specification CMS* be a method specification whose descriptor is included in the descriptor of *D*, such that the following are all true:

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

ii) Let $PCMS_j$, 1 (one) $\leq j \leq N$, be the *j*-th SQL parameter in the unaugmented SQL parameter declaration list of *CMS*. Let PMS_j, 1 (one) $\leq j \leq N$, be the *j*-th SQL parameter in the unaugmented SQL parameter declaration list *MPDL*.

iii) For *j* varying from 1 (one) to *N*, the Syntax Rules of Subclause 10.14, “Data type identity”, are applied with the declared type of $PCMS_j$ and the declared type of PMS_j.

iv) *CMS* is an SQL-invoked constructor method.

b) Otherwise, let a *conflicting method specification CMS* be a method specification whose descriptor is included in the descriptor of some *SPD* or *SBD*, such that the following are all true:

i) *MN* and the method name included in the descriptor of *CMS* are equivalent.

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

iii) Let $PCMS_j$, 1 (one) ≤ j ≤ *N*, be the *j*-th SQL parameter in the unaugmented SQL parameter declaration list of *CMS*. Let PMS_j, 1 (one) ≤ *j* ≤ *N*, be the *j*-th SQL parameter in the unaugmented SQL parameter declaration list *MPDL*.

iv) For *j* varying from 1 (one) to *N*, the Syntax Rules of Subclause 10.14, “Data type identity”, are applied with the declared type of $PCMS_j$ and the declared type of PMS_j.

v) *CMS* and *ORMS* either both are instance methods or one of *CMS* and *ORMS* is a static method and the other is an instance method.

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

Replace Syntax Rule 18) with:

18) If *PORMS* does not specify STATIC or CONSTRUCTOR, then there shall be no SQL-invoked function *F* that satisfies all the following conditions:

a) *F* is not an SQL-invoked method.

b) The <routine name> of *F* and *RN* have equivalent <qualified identifier>s.

c) *F* has *AN* SQL parameters.

d) *D* is a subtype or supertype of the declared type of the first SQL parameter of *F*.

e) The declared type of the *i*-th SQL parameter in *NPL*,2 ≤ *i* ≤ *AN* is compatible with the declared type of *i*-th SQL parameter of *F*.

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

Replace General Rule 1 b) with:

1) b) An indication of whether STATIC or CONSTRUCTOR is specified.

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

Replace General Rule 1) d) with:

1) d) The <specific method name> of *PORMS*.

11. *Rationale: <method characteristic> should not include <transform group specification>.*

Delete General Rule 1) j)

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

Replace General Rule 1) n) with:

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

11.46 <add overriding method specification>

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

Replace Syntax Rule 2) with:

2) Let *POVMS* be the <partial method specification> immediately contained in *OVMS*. *POVMS* shall not specify STATIC or CONSTRUCTOR.

2. *Rationale: Remove syntax rule that conflicts with format rules.*

Delete Syntax Rule 5)

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

Replace Syntax Rule 7) with:

7) Case:

a) If *POVMS* does not specify <specific method name>, then an implementation-dependent <specific method name> is implicit whose <schema name> is equivalent to *SN*.

b) Otherwise:

Case:

i) If <specific method name> contains a <schema name>, then that <schema name> shall be equivalent to *SN*.

ii) Otherwise, the <schema name> *SN* is implicit.

The schema identified by the explicit or implicit <schema name> of the <specific method name> shall not include a routine descriptor whose specific name is equivalent to <specific method name> or a user-defined type descriptor that includes a method specification descriptor whose specific method name is equivalent to <specific method name>.

4. *Rationale: Correct improper use of terms in Syntax Rule 9) a).*

Replace Syntax Rule 9) a) with:

9) a) *MN* and the <method name> of *COMS* are equivalent.

5. *Rationale: Remove improper use of "shall".*

Replace Syntax Rule 9) b) with:

9) b) Let *N* be the number of elements of the augmented SQL parameter declaration list *UPCOMS* generally included in the descriptor of *COMS*. *MPDL* contains (*N*-1) SQL parameter declarations.

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

Add the following Syntax Rule:

9) c.1) The descriptor of *COMS* shall not include an indication that STATIC or CONSTRUCTOR has been specified.

7. *Rationale: Rectify misplacement of Syntax Rule 9) d); correct the subrules i) and iv) to enforce correct match of parameter names and locator indications.*

Delete Syntax Rule 9) d)

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

Delete Syntax Rule 11)

9. *Rationale: Rectify misplacement of Syntax Rule 9) d); correct the subrules i) and iv) to enforce correct match of parameter names and locator indications.*

Add the following Syntax Rule:

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

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

b) If the descriptor of the *i*-th parameter of the augmented <SQL parameter declaration list> of *UPCOMS* includes a 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 descriptor of the *i*-th parameter of the augmented <SQL parameter declaration list> of *UPCOMS* shall include a <locator indication>.

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

10. *Rationale: Add missing syntax rule that prevents the addition of a conflicting overriding method specification to some structured type.*

Add the following Syntax Rule:

13.1) Let a *conflicting overriding method specification COVMS* be an overriding method specification that is included in the descriptor of *D*, such that the following are all true:

a) *MN* and the method name of *COVMS* are equivalent.

b) The augmented SQL parameter declaration list of *COVMS* contains *N* elements.

c) For *i* varying from 2 to *N*, the data types of the SQL parameter *POVMSi-1* and the SQL parameter *PCOVMSi* of *COVMS* are compatible.

There shall be no conflicting overriding method specification *COVMS*.

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

Add the following Syntax Rule:

16) b) ii) 3.1) MSD_i does not include an indication that CONSTRUCTOR has been specified.

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

Replace General Rule 1) c) with:

1) c) The <specific method name> of *POVMS*.

13. *Rationale: <method characteristic> should not include <transform group specification>.*

Delete General Rule 1) j)

11.47 <drop method specification>

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

Replace the Format with:

```
<drop method specification> ::=
     DROP <specific method specification designator> RESTRICT

<specific method specification designator> ::=
       SPECIFIC METHOD <specific method name>
     | [ INSTANCE | STATIC | CONSTRUCTOR] METHOD <method name>
         [ <data type list> ]
```

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

Replace the Syntax Rules with:

1) Let *D* be the user-defined type identified by the <user-defined type name> *DN* immediately contained in the <alter type statement> containing the <drop method specification> *DORMS*. Let *DSN* be the explicit or implicit <schema name> of *DN*. Let *SMSD* be the <specific method specification designator> immediately contained in *DORMS*.

2) If *SMSD* immediately contains a <specific method name> *SMN*, then

   a) If <specific method name> contains a <schema name>, then that <schema name> shall be equivalent to *DSN*. If <specific method name> does not contain a <schema name>, then the <schema name> *DSN* is implicit.

   b) The descriptor of *D* shall include a method specification descriptor *DOOMS* whose specific method name is equivalent to *SMN*.

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

   d) Let *MN* be the <method name> included in *DOOMS*.

3) If *SMSD* immediately contains a <method name> *ME*, then:

   a) If none of INSTANCE, STATIC, or CONSTRUCTOR is immediately contained in *SMSD*, then INSTANCE is implicit.

   b) The descriptor of *D* shall include a method specification descriptor *DOOMS* whose method name *MN* is equivalent to *ME*.

   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* which includes:

         A) An indication that the method specification is STATIC.

         B) An indication that the method specification is original.

         C) An augmented parameter list *PDL* such that for all *i*, the declared type of its *i*-th parameter 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* which includes:

         A) An indication that the method specification is CONSTRUCTOR.

         B) An indication that the method specification is original.

         C) An augmented parameter list *PDL* such that for all *i* > 1, the declared type of its *i*-th parameter 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:

A) 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.

B) *DOOMS* includes an augmented parameter list *PDL* such that for all $i > 1$, the declared type of its *i*-th parameter 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*.

d) If *SMSD* does not immediately contain a <data type list>, then:

Case:

i) If STATIC is specified, the descriptor of *D* shall include exactly one method specification descriptor *DOOMS* which includes an indications that the method specification is both original and STATIC.

ii) If CONSTRUCTOR is specified, the descriptor of *D* shall include exactly one method specification descriptor *DOOMS* which includes an indications that the method specification is both original and CONSTRUCTOR.

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

4) Case:

a) If *DOOMS* includes an indication that the method specification is original, then:

Case:

i) If *DOOMS* includes an indication that the method specification specified STATIC, then there shall be no SQL-invoked function *F* that satisfies all of the following conditions:

1) The <routine name> of *F* and *MN* have equivalent <qualified identifier>s.

2) If *N* is the number of elements in *PDL*, then *F* has *N* SQL parameters.

3) The declared type of the first SQL parameter of *F* is *D*.

4) The declared type of the *i*-th element of *PDL*, $1 <= i <= N$, is compatible with the declared type of SQL parameter P_i of *F*.

5) *F* is an SQL-invoked method.

6) *F* includes an indication that STATIC is specified.

ii) If *DOOMS* includes an indication that the method specification specified CONSTRUCTOR, then there shall be no SQL-invoked function *F* that satisfies all of the following conditions:

1) The <routine name> of *F* and *MN* have equivalent <qualified identifier>s.

2) If *N* is the number of elements in *PDL*, then *F* has *N* SQL parameters.

3) The declared type of the first SQL parameter of *F* is *D*.

4) The declared type of the *i*-th element of *PDL*, 2 <= *i* <= *N*, is compatible with the declared type of SQL parameter P_i of *F*.

5) *F* is an SQL-invoked method.

6) *F* includes an indication that CONSTRUCTOR is specified.

iii) Otherwise:

1) There shall be no proper subtype *PSBD* of *D* whose descriptor includes the descriptor *DOVMS* of an overriding method specification such that all of the following is true:

A) *MN* and the < method name> included in *DOVMS* have equivalent <qualified identifier>s.

B) If *N* is the number of elements in *PDL*, then the augmented SQL parameter declaration list *APDL* included in *DOVMS* has *N* SQL parameters.

C) *PSBD* is the declared type the first SQL parameter of *APDL*.

D) The declared type of the *i*-th element of *PDL*, 2 <= *i* <= *N*, is compatible with the declared type of SQL-parameter P_i of *APDL*.

2) There shall be no SQL-invoked function *F* that satisfies all of the following conditions:

A) The <routine name> of *F* and *MN* have equivalent <qualified identifier>s.

B) If *N* is the number of elements in *PDL*, then *F* has *N* SQL parameters.

C) The declared type of the first SQL parameter of *F* is *D*.

D) The declared type of the *i*-th element of *PDL*, 2 <= *i* <= *N*, is compatible with the declared type of SQL parameter P_i of *F*.

E) *F* is an SQL-invoked method.

F) *F* does not include an indication that either STATIC or CONSTRUCTOR is specified.

b) Otherwise, there shall be no SQL-invoked function *F* that satisfies all of the following conditions:

i) The <routine name> of *F* and *MN* have equivalent <qualified identifier>s.

ii) If *N* is the number of elements in *PDL*, then *F* has *N* SQL parameters.

iii) The declared type of the first SQL parameter of *F* is *D*.

iv) The declared type of the *i*-th element of *PDL*, 2 <= *i* <= *N*, is compatible with the declared type of SQL parameter P_i of *F*.

v) *F* is an SQL-invoked method.

vi) *F* does not include an indication that either STATIC or CONSTRUCTOR is specified.

Replace the General Rules with:

1) Let *STDS* be the descriptor of *D*.

2) *DOOMS* is removed from *STDS*.

3) *DOOMS* is destroyed.

11.48 <drop data type statement>

1. *Rationale: Correct oversight in rules for recursive type definition.*

Replace Syntax Rule 4) a) with:

4) a) The declared type of no column, field, or attribute whose descriptor is not included in the descriptor of *D* shall be based on *SRD*, or *SAD*.

2. *Rationale: Correct oversight in rules for recursive type definition.*

Replace Syntax Rule 4) b) with.

4) b) The declared type of no column, field, or attribute shall be based on *D*.

3. *Rationale: Dependency on a user-defined ordering requires only the existence of the ordering.*

Replace Syntax Rule 4) h) iii) with:

4) h) iii) *R* shall not be the ordering function included in the descriptor of any user-defined type.

Delete Syntax Rule 4) h) iv).

11.49 <SQL-invoked routine>

1. *Rationale: Clarify that implicit schema of user-defined type is schema name of either the <schema definition> or the <SQL-client module definition> in which the corresponding SQL statement appears. Correct namespace problems associated with methods used to initialize newly-constructed structured type values.*

In the Format, replace the production for <method specification designator> with:

```
<method specification designator> ::=
     [ INSTANCE | STATIC | CONSTRUCTOR ] METHOD <method name>
     <SQL parameter declaration list>
     [ <returns clause> ]
     FOR <user-defined type name>
```

2. *Rationale: <method characteristic> should not include <transform group specification>.*

In the Format, replace the production for <routine characteristic> with:

```
<routine characteristic> ::=
       <language clause>
     | <parameter style clause>
```

```
        | SPECIFIC <specific name>
        | <deterministic characteristic>
        | <SQL-data access indication>
        | <null-call clause>
        | <dynamic result sets characteristic>
```

3. *Rationale: <method characteristic> should not include <transform group specification>.*

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

```
<external body reference> ::=
     EXTERNAL [ NAME <external routine name> ]
     [ <parameter style clause> ]
     [ <transform group specification> ]
     [ <external security clause> ]
```

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

Replace Syntax Rule 3) with:

3) An <SQL-invoked routine> specified as an <SQL-invoked procedure> is called an *SQL-invoked procedure*; an <SQL-invoked routine> specified as an <SQL-invoked function> is called *an SQL-invoked function*. An <SQL-invoked function> that specifies a <method specification designator> is further called an *SQL-invoked method*. An SQL-invoked method that specifies STATIC is called a *static SQL-invoked method*. An SQL-invoked method that specifies CONSTRUCTOR is called an *SQL -invoked constructor method*.

5. *Rationale: Clarify that implicit schema of user-defined type is schema name of either the <schema definition> or the <SQL-client module definition> in which the corresponding SQL statement appears.*

Replace Syntax Rule 4) a) with:

4) a) Let *UDTN* be the <user-defined type name> immediately contained in <method specification designator>. Let *UDT* be the user-defined type identified by *UDTN*.

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

Replace Syntax Rule 4) b) with:

4) b) There shall exist a method specification descriptor *DMS* in the descriptor of *UDT* such that the <method name> of *DMS* is equivalent to the <method name>, *DMS* indicates STATIC if and only if the <method specification designator> specifies STATIC, *DMS* indicates CONSTRUCTOR if and only if the <method specification designator> specifies CONSTRUCTOR and the declared type of every SQL parameter in the unaugmented SQL parameter declaration list in *DMS* is compatible with the declared type of the corresponding SQL parameter in the <SQL parameter declaration list> contained in the <method specification designator>. *DMS* identifies the corresponding method specification of the <method specification designator>.

7. *Rationale: Remove the anomaly in <SQL parameter name>s.*

Replace Syntax Rule 4) e) with:

4) e) For *i* varying from 1 (one) to *MN*, the <SQL parameter name>s contained in $PCOMS_i$ and $POVMS_i$ shall be equivalent.

8. *Rationale: <method characteristic> should not include <transform group specification>.*

Delete Syntax Rule 4) j).

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

Replace Syntax Rule 4) k) with:

4) k) Let *SPN* be the <specific method name> in *DMS*. *SPN* is the <specific name> of *R*.

10. *Rationale: <method characteristic> should not include <transform group specification>.*

Replace Syntax Rule 5) a) with:

5) a) <routine characteristics> shall contain at most one <language clause>, at most one <parameter style clause>, at most one <specific name>, at most one <deterministic characteristic>, at most one <SQL-data access indication>, at most one <null-call clause>, and at most one <dynamic result sets characteristic>.

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

Replace Syntax Rule 5) g) with:

5) g) <SQL-data access indication> shall be specified.

12. *Rationale: <method characteristic> should not include <transform group specification>.*

Replace Syntax Rule 5) j) iii) with:

5) j) iii) <parameter style clause> shall not be specified.

13. *Rationale: Remove the anomaly in <SQL parameter name>s.*

Replace Syntax Rule 10) with:

10) No two <SQL parameter name>s contained in *NPL* shall be equivalent.

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

Add the following Syntax Rule:

15) b) i) 1.1) If *R* is an SQL-invoked constructor method, then let *SCR* be the set containing every SQL-invoked constructor method of type *UDT*, including *R*, whose <schema qualified routine name> is equivalent to *RN* and whose number of SQL parameters is *PN*.

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

Replace Syntax Rule 15) b) i) 2) with:

15) b) i) 2) Otherwise, let *SCR* be the set containing every SQL-invoked function in *S* that is not a static SQL-invoked method or an SQL-invoked constructor method, including *R*, whose

<schema qualified routine name> is equivalent to *RN* and whose number of SQL parameters is *PN*.

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

Replace Syntax Rule 18) c) with:

18) c) If READS SQL DATA is specified, then it is implementation-defined whether the <SQL routine body> shall not contain an <SQL procedure statement> *S* that satisfies at least one of the following:

i) *S* is an <SQL data change statement>.

ii) *S* contains a <routine invocation> whose subject routine is an SQL-invoked routine that possibly modifies SQL-data.

iii) *S* contains an <SQL procedure statement> that is an <SQL data change statement>.

Replace Syntax Rule 18) d) with:

18) d) If CONTAINS SQL is specified, then it is implementation-defined whether the <SQL routine body> shall not contain an <SQL procedure statement> *S* that satisfies at least one of the following:

i) *S* is an <SQL data statement> other than <free locator statement> and <hold locator statement>.

ii) *S* contains a <routine invocation> whose subject routine is an SQL-invoked routine that possibly modifies SQL-data or possibly reads SQL-data.

iii) *S* contains an <SQL procedure statement> that is an <SQL data statement> other than <free locator statement> and <hold locator statement>.

17. Rationale: Clarify that the implicit schema of user-defined type used as a return type or parameter type is resolved using the SQL-path.

Replace Syntax Rule 19) d) iii) 2) B) II) with:

19) d) iii) 2) B) II) If *RT* specifies a <user-defined type> without a <locator indication>

1) Case:

a) If *R* is an SQL-invoked method that is an overriding method, then the Syntax Rules of Subclause 10.18, “Determination of a to-sql function for an overriding method”, are applied with *R* as *ROUTINE*. There shall be an applicable to-sql function *TSF*.

b) Otherwise, the Syntax Rules of Subclause 10.17, “Determination of a to-sql function”, are applied with the data type identified by *RT* and the <group name> contained in the <group specification> that contains *RT* as *TYPE* and *GROUP*, respectively. There shall be an applicable to-sql function *TSF*.

2) *TSF* is called the to-sql function associated with the result.

3) Case:

a) If *TSF* is an SQL-invoked method, then *PT* is the <parameter type> of the second SQL parameter of *TSF*.

b) Otherwise, *PT* is the <parameter type> of the first SQL parameter of *TSF*.

18. *Rationale: Clarify that the implicit schema of user-defined type used as a return type or parameter type is resolved using the SQL-path.*

Replace Syntax Rule 19) d) iv) 1) B) with:

19) d) iv) 1) B) If the <parameter type> T_i simply contained in the *i*-th <SQL parameter declaration> is a <user-defined type> without a <locator indication>, then:

I) Case:

1) If the <parameter mode> immediately contained in the *i*-th <SQL parameter declaration> is IN, then:

a) The Syntax Rules of Subclause 10.15, “Determination of a from-sql function”, are applied with the data type identified by T_i and the <group name> contained in the <group specification> that contains T_i as *TYPE* and *GROUP*, respectively. There shall be an applicable from-sql function FSF_i . FSF_i is called the from-sql function associated with the *i*-th SQL parameter.

b) The *i*-th effective SQL parameter list entry is the *i*-th <SQL parameter declaration> with the <parameter type> replaced by the <returns data type> of FSF_i.

2) If the <parameter mode> immediately contained in the *i*-th <SQL parameter declaration> is OUT, then:

a) The Syntax Rules of Subclause 10.17, “Determination of a to-sql function”, are applied with the data type identified by T_i and the <group name> contained in the <group specification> that contains T_i as *TYPE* and *GROUP*, respectively. There shall be an applicable to-sql function TSF_i. TSF_i is called the to-sql function associated with *i*-th SQL parameter.

b) The *i*-th effective SQL parameter list entry is the *i*-th <SQL parameter declaration> with the <parameter type> replaced by

Case:

i) If TSF_i is an SQL-invoked method, then the <parameter type> of the second SQL parameter of TSF_i .

ii) Otherwise, the <parameter type> of the first SQL parameter of *TSFi* .

3) Otherwise:

a) The Syntax Rules of Subclause 10.15, “Determination of a from-sql function”, are applied with the data type identified by T_i and the <group name> contained in the <group specification> that contains T_i as *TYPE* and *GROUP*, respectively. There shall be an applicable from-sql function FSF_i . FSF_i is called the from-sql function associated with the *i*-th SQL parameter.

b) The Syntax Rules of Subclause 10.17, “Determination of a to-sql function”, are applied with the data type identified by T_i and the <group name> contained in the <group specification> that contains T_i as *TYPE* and *GROUP*, respectively. There shall be an applicable to-sql function TSF_i. TSF_i is called the to-sql function associated with the *i*-th SQL parameter.

c) The *i*-th effective SQL parameter list entry is the *i*-th <SQL parameter declaration> with the <parameter type> replaced by the <returns data type> of FSF_i.

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

Replace Syntax Rule 20) with:

20) Case:

a) If <method specification designator> is specified, then:

i) If *DMS* indicates that the method is deterministic, then *R* is deterministic; otherwise, *R* is possibly non-deterministic.

ii) If the SQL-data access indication of *DMS* indicates that the method possibly modifies SQL-data, then *R* possibly modifies SQL-data. If the SQL-data access indication of *DMS* indicates that the method possibly reads SQL-data, then *R* possibly reads SQL-data. If the SQL-data access indication of *DMS* indicates that the method possibly contains SQL, then *R* possibly contains SQL. Otherwise, *R* does not possibly contain SQL.

b) Otherwise:

i) If DETERMINISTIC is specified, then *R* is deterministic; otherwise, it is possibly non-deterministic.

ii) An <SQL-invoked routine> possibly modifies SQL-data if and only if <SQL-data access indication> specifies MODIFIES SQL DATA.

iii) An <SQL-invoked routine> possibly reads SQL-data if and only if <SQL-data access indication> specifies READS SQL DATA.

iv) An <SQL-invoked routine> possibly contains SQL if and only if <SQL-data access indication> specifies CONTAINS SQL.

Replace General Rule 3) o) with:

3) o) The SQL-invoked routine descriptor includes an indication whether the SQL-invoked routine does not possibly contain SQL, possibly contains SQL, possibly reads SQL-data, or possibly modifies SQL-data.

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

Replace General Rule 4) d) with:

4) d) The routine descriptor includes an indication that the SQL-invoked routine is an SQL-invoked function that is an SQL-invoked method, an indication of the user-defined type *UDT*, and an indication of whether STATIC or CONSTRUCTOR was specified.

21. *Rationale: Editorial.*

Replace Conformance Rules 1) and 4) with:

1) Without Feature T471, “Result sets return value”, conforming SQL language shall not specify <dynamic result sets characteristic>.

4) Without Feature S241, “Transform functions”, conforming SQL language shall not specify <transform group specification>.

22. *Rationale: Remove useless item from Core.*

Add the following Conformance Rule:

10.1 Without at least one of the features:
S231 “Structured Type locators”,
S232 “Array Locators”,
T041 “Basic LOB data type”
conforming SQL language shall not specify <locator indication>.

11.50 <alter routine statement>

1. *Rationale: Dependency on a user-defined ordering requires only the existence of the ordering..*

Replace Syntax Rule 4) a) with:

4) a) *SR* shall not be the ordering function included in the descriptor of any user-defined type *UDT*.

11.51 <drop routine statement>

1. *Rationale: Dependency on a user-defined ordering requires only the existence of the ordering.*

Replace Syntax Rule 4) with:

4) If RESTRICT is specified, then *SR* shall not be the ordering function included in the descriptor of any user-defined type *UDT*.

2. *Rationale: When dropping a routine that is an order function, it is also necessary to perform a <drop user-defined ordering statement>.*

Add the following General Rule:

3.1) If *SR* is the ordering function included in the descriptor of any user-defined type *UDT*, then let *UDTN* be a <user-defined type name> that identifies *UDT*. The following <drop user-defined ordering statement> is effectively executed without further Access Rule checking:

DROP ORDERING FOR *UDTN* CASCADE

11.52 <user-defined cast definition>

1. *Rationale: Clarify that implicit schema of user-defined type is schema name of either the <schema definition> or the <SQL-client module definition> in which the corresponding SQL statement appears.*

Replace Syntax Rules 4) and 5) with:

4) At least one of *SDT* or *TDT* shall contain a <user-defined type name> or a <reference type>.

5) If *SDT* contains a <user-defined type name>, then let *SSDT* be the schema that includes the descriptor of the user-defined type identified by *SDT*.

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

Add the following Syntax Rules:

6) a) i.1) EQUALS ONLY shall be specified.

i.2) The declared type of each attribute of *UDT* shall not be UDT-NC-ordered.

3. *Rationale: Clarify that implicit schema of user-defined type is schema name of either the <schema definition> or the <SQL-client module definition> in which the corresponding SQL statement appears.*

Replace Syntax Rule 7) with:

7) If *TDT* contains a <user-defined type name>, then let *STDT* be the schema that includes the descriptor of the user-defined type identified by *TDT*.

4. *Rationale: Provide consistent Syntax Rules for comparison operations.*

Add the following Syntax Rule:

8) c) If the result data type of *F* is a large object type, then ORDER FULL shall not be specified.

5. *Rationale: Clarify that implicit schema of user-defined type is schema name of either the <schema definition> or the <SQL-client module definition> in which the corresponding SQL statement appears.*

Replace Syntax Rule 9) with:

9) If both *SDT* and *TDT* contain a <user-defined type name> or a <reference type>, then the <authorization identifier> that owns *SSDT* and the <authorization identifier> that owns *STDT* shall be equivalent.

Replace Syntax Rule 10) c) with:

10) c) The <authorization identifier> that owns *SSDT* or *STDT* (both, if both *SDT* and *TDT* are <user-defined type name>s) shall own the schema that includes the SQL-invoked routine descriptor of *F*.

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

Add the following Conformance Rules:

2) Without Feature T042, “Extended LOB data type support”, if <state category> is specified, then the declared type of each attribute of *UDT* shall not be LOB-ordered.

3) Without Feature T042, “Extended LOB data type support”, if <map category> is specified, the result type of *F* shall not be a large object type.

11.54 <user-defined ordering definition>

1. *Rationale: Clarify that implicit schema of user-defined type is schema name of either the <schema definition> or the <SQL-client module definition> in which the corresponding SQL statement appears.*

In the Format, replace the production for <user-defined ordering definition> with:

```
<user-defined ordering definition> ::=
     CREATE ORDERING FOR <user-defined type name> <ordering form>
```

Replace Syntax Rule 1) with:

1) Let *UDTN* be the <user-defined type name>. Let *UDT* be the user-defined type identified by *UDTN*.

2. *Rationale: Editorial.*

Replace Syntax Rule 2) with:

2) The descriptor of *UDT* shall include an ordering form that specifies NONE.

3. *Rationale: Provide consistent Syntax Rules for comparison operations.*

Add the following Syntax Rules:

6) i.1) EQUALS ONLY shall be specified.

   i.2) The declared type of each attribute of *UDT* shall not be UDT-NC-ordered.

Append the following Syntax Rules:

8) c) If the result data type of *F* is a large object type, then ORDER FULL shall not be specified.

4. *Rationale: Editorial.*

Replace Access Rule 1) with:

1) The enabled authorization identifiers shall include the <authorization identifier> that owns the schema that includes the descriptor of *UDT* and the schema that includes the routine descriptor of *F*.

5. *Rationale: Incorrect informative note.*

Delete NOTE 264

6. *Rationale: Editorial.*

Replace General Rule 4) with:

4) The <specific routine designator> identifying the ordering function, depending on the ordering category, in the descriptor of *UDT* is set to *SRD*.

7. *Rationale: Editorial.*

Replace Conformance Rule 1) with:

1) Without Feature S251, “User-defined orderings”, conforming SQL shall contain no <user-defined ordering definition>.

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

Add the following Conformance Rules:

2) Without Feature T042, “Extended LOB data type support”, if <state category> is specified, then the declared type of each attribute of *UDT* shall not be LOB-ordered.

3) Without Feature T042, “Extended LOB data type support”, if <map category> is specified, the result type of *F* shall not be a large object type.

11.55 <drop user-defined ordering statement>

1. *Rationale: Clarify that implicit schema of user-defined type is schema name of either the <schema definition> or the <SQL-client module definition> in which the corresponding SQL statement appears.*

In the Format, replace the production for <drop user-defined ordering statement> with:

```
<drop user-defined ordering statement> ::=
     DROP ORDERING FOR <user-defined type name> <drop behavior>
```

Replace Syntax Rule 1) with:

1) Let *UDTN* be the <user-defined type name>. Let *UDT* be the user-defined type identified by *UDTN*.

2. *Rationale: Editorial.*

Replace Syntax Rule 2) with:

2) The descriptor of *UDT* shall include an ordering form that specifies EQUALS or FULL.

3. *Rationale: Recognize that a <distinct predicate> may depend on a user-defined ordering. Provide the correct set of dependencies for a user-defined ordering.*

Replace Syntax Rule 4) with:

4) Let *S* be the set of types whose comparison type is *UDT*. Let *DEP* designate any of the following:

   a) A <comparison predicate>, <between predicate>, <in predicate>, <quantified comparison predicate>, <match predicate>, or <distinct predicate> that simply contains a <row value expression> whose declared type is S-ordered.

b) A <table subquery> simply contained in a <quantified comparison predicate>, <in predicate>, <unique predicate>, or <match predicate> that has a column whose declared type is S-ordered.

c) A <set function specification> that specifies DISTINCT, MAX, or MIN and that simply contains a <value expression> whose declared type is S-ordered.

d) A <group by clause> that contains a <column reference> to a column whose declared type is S-ordered.

e) A <query specification> that specifies the <set quantifier> DISTINCT and that has a column whose declared type is S-ordered.

f) A <query expression> that specifies or implies UNION DISTINCT and that has a column whose declared type is S-ordered.

g) A <query expression> that specifies INTERSECT or EXCEPT and that has a column whose declared type is S-ordered.

j) A <joined table> that specifies NATURAL or USING and that has a corresponding join column whose declared type is S-ordered.

Replace Syntax Rule 5) with:

5) If RESTRICT is specified, then *DEP* shall not be specified in any of the following:

a) The <SQL routine body> of any routine descriptor.

b) The <query expression> of any view descriptor.

c) The <search condition> of any constraint descriptor or assertion descriptor.

d) The triggered action of any trigger descriptor.

NOTE 267 — If CASCADE is specified, then such referencing objects will be dropped by the execution of the <revoke statement> specified in the General Rules of this Subclause.

4. *Rationale: Provide the correct set of dependencies for a user-defined ordering.*

Replace General Rules 1), 2), 3), 4), 5) and 6) with:

1) Let *R* be any SQL-invoked routine that contains *DEP* in its <SQL routine body>. Let *SN* be the specific name of *R*. The following <drop routine statement> is effectively executed without further Access Rule checking:
DROP SPECIFIC ROUTINE *SN* CASCADE

2) Let *V* be any view that contains *DEP* in its <query expression>. Let *VN* be the <table name> of *V*. The following <drop view statement> is effectively executed without further Access Rule checking:
DROP VIEW *VN* CASCADE

3) Let *T* be any table that contains *DEP* in the <search condition> of any constraint *C* whose constraint descriptor is included in the table descriptor of *T*. Let *TN* be the <table name> of *T*. Let *TCN* be the <constraint name> of *C*. The following <alter table statement> is effectively executed without further Access Rule checking:
ALTER TABLE *TN* DROP CONSTRAINT *TCN* CASCADE

4) Let *A* be any assertion that contains *DEP* in its <search condition>. Let *AN* be the <constraint name> of *A*. The following <drop assertion statement> is effectively executed without further Access Rule checking:
DROP ASSERTION *AN* CASCADE

5) Let *D* be any domain that contains *DEP* in the <search condition> of any constraint descriptor or in the <default option> included in the domain descriptor of *D*. Let *DN* be the <domain name> of *D*. The following <drop domain statement> is effectively executed without further Access Rule checking:
DROP DOMAIN *DN* CASCADE

6) Let *T* be any trigger that contains *DEP* in its triggered action. Let *TN* be the <trigger name> of *T*. The following <drop trigger statement> is effectively executed without further Access Rule checking:
DROP TRIGGER *TN* CASCADE

5. *Rationale: Editorial.*

Replace General Rule 7) with:

7) In the user-defined data type descriptor of *UDT*, the ordering form is set to NONE and the ordering category is set to STATE. No ordering function is included in the user-defined data type descriptor of *UDT*.

11.56 <transform definition>

1. *Rationale: Clarify that implicit schema of user-defined type is schema name of either the <schema definition> or the <SQL-client module definition> in which the corresponding SQL statement appears.*

In the Format, replace the production for <transform definition> with:

```
<transform definition> ::=
     CREATE { TRANSFORM | TRANSFORMS } FOR
        <user-defined type name> <transform group>...
```

11.57 <drop transform statement>

1. *Rationale: Clarify that implicit schema of user-defined type is schema name of either the <schema definition> or the <SQL-client module definition> in which the corresponding SQL statement appears.*

In the Format, replace the production for <drop transform statement> with:

```
<drop transform statement> ::=
     DROP { TRANSFORM | TRANSFORMS } <transforms to be dropped>
        FOR <user-defined type name> <drop behavior>
```

12.2 <grant privilege statement>

1. *Rationale: Correct the references to <grant statement>.*

Replace Syntax Rule 5) with:

5) If the <grant privilege statement> is not contained in a <schema definition>, then the schema identified by the explicit or implicit qualifier of the <object name> shall include the descriptor of *O*. If <grant privilege statement> is contained in a <schema definition> S, then the schema identified by the

explicit or implicit qualifier of the <object name> shall include the descriptor of *O* or *S* shall include a <schema element> that creates the descriptor of *O*.

2. *Rationale: Editorial.*

Replace General Rule 7) with:

7) Let *SWH* be the set of privilege descriptors in *CPD* whose action is SELECT WITH HIERARCHY OPTION, and let *ST* be the set of subtables of *O*, then for every grantee *G* in *SWH* and for every table *T* in *ST*, the following <grant statement> is effectively executed without further Access Rule checking:

GRANT SELECT ON *T* TO *G* GRANTED BY *A*

3. *Rationale: Correct the references to <grant statement>.*

Replace Conformance Rule 1) with:

1) Without Feature S024, " Enhanced structured types", a <specific routine designator> contained in a <grant privilege statement> shall not identify a method.

12.4 <select statement: single row>

1. *Rationale: Replace incorrect non-terminal.*

Replace Syntax Rule 1) with:

1) Insert after SR4) For each <target specification> *TS* that is an <SQL variable reference>, the Syntax Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2, apply to *TS* and the corresponding element of the <select list>, as *TARGET* and *VALUE*, respectively.

Replace General Rule 1) with:

1) Insert after GR5) For each <target specification> *TS* that is an <SQL variable reference>, the corresponding value in the row of *Q* is assigned to *TS* according to the General Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2, as *VALUE* and *TARGET*, respectively. The assignment of values to targets in the <select target list> is in an implementation-dependent order.

12.6 <revoke statement>

1. *Rationale: Correct the references to schema which do not treat it as a descriptor.*

Replace Syntax Rule 30) with:

30) *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*.

Replace General Rule 10) with:

10) For every lost schema *S1*, the default character set of that schema is modified to include the name of the implementation-defined <character set specification> that would have been this schema's default character set had the <schema definition> not specified a <schema character set specification>.

13.1 <SQL-client module definition>

1. *Rationale: Editorial - supply missing ellipsis.*

In the Format, replace the production <SQL-client module definition> with:

```
<SQL-client module definition> ::=
     <module name clause>
     <language clause>
     <module authorization clause>
     [ <module path specification> ]
     [ <module transform group specification> ]
     [ <temporary table declaration>... ]
     <module contents>...
```

13.2 <module name clause>

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

Replace Conformance Rule 1) with:

1) Without Feature F461, “Named character sets”, <module character set specification> shall not be specified.

13.3 <externally-invoked procedure>

1. *Rationale: Complete removal of deprecated feature .*

In the Format, replace the production for <host parameter declaration setup> with:

```
<host parameter declaration setup> ::= <host parameter declaration list>
```

Delete NOTE 274.

13.5 <SQL procedure statement>

1. *Rationale: Remove redundant item. <grant role statement> is defined in <grant statement>.*

In the Format, replace the production for <SQL schema definition statement> with:

```
<SQL schema definition statement> ::=
       <schema definition>
     | <table definition>
     | <view definition>
     | <SQL-invoked routine>
     | <grant statement>
     | <role definition>
     | <domain definition>
     | <character set definition>
     | <collation definition>
     | <translation definition>
     | <assertion definition>
     | <trigger definition>
     | <user-defined type definition>
     | <user-defined cast definition>
```

```
        | <user-defined ordering definition>
        | <transform definition>
```

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

Delete Syntax Rule 4).

Delete Syntax Rule 5).

Delete Syntax Rule 6).

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

Replace Conformance Rule 3) and 10) with:

3) Without Feature F451, “Character set definition”, an <SQL schema definition statement> shall not be a <character set definition>.

10) Without Feature F451, “Character set definition”, an <SQL schema definition statement> shall not be a <drop character set statement>.

14.1 <declare cursor>

1. *Rationale: Eliminate redundant <collate clause>.*

In the Format, replace the production for <sort specification> with:

```
<sort specification> ::= <sort key> [ <ordering specification> ]
```

2. *Rationale: Correct non-terminals.*

Replace Syntax Rules 7), 8) and 9) with:

7) If <cursor scrollability> is not specified, then NO SCROLL is implicit.

8) If <cursor holdability> is not specified, then WITHOUT HOLD is implicit.

9) If <cursor returnability> is not specified, then WITHOUT RETURN is implicit.

3. *Rationale: Prohibit ordering on data types that do not support ordering. Provide consistent Conformance Rules for comparison operations.*

Replace Syntax Rule 18) c) with:

18) c) *DT* shall not be LOB-ordered, array-ordered, UDT-EC-ordered, UDT-NC-ordered, or reference-ordered.

4. *Rationale: A <sort key> of character type need not be a column reference.*

Replace Syntax Rule 19) with:

19) If a <sort specification> contains a <collate clause>, then the declared type of the <value expression> contained in the <sort specification> shall be character string. The column descriptor of the

corresponding column in the result has the collating sequence specified in <collate clause> and the coercibility characteristic Explicit.

5. *Rationale: Editorial.*

Replace Syntax Rule 20) with:

20) If an <updatability clause> of FOR UPDATE without a <column name list> is specified or implicit, then a <column name list> that consists of the <column name> of every column of *LUT* is implicit.

6. *Rationale: Syntax Rule 22) states the wrong restriction on <sort key>s of user-defined type. The correct restriction is already present in Syntax Rule 18)c).*

Delete Syntax Rule 22).

7. *Rationale: Clarify that nulls sort as if they were equal.*

Replace General Rule 2) c) with:

2) c) Let *P* be any row of *TS* and let *Q* be any other row of *TS*, and let PV_i and QV_i be the values of K_i in these rows, respectively. The relative position of rows *P* and *Q* in the result is determined by comparing PV_i and QV_i according to the rules of Subclause 8.2, “<comparison predicate>”, where the <comp op> is the applicable <comp op> for K_i , with the following special treatment of null values. A sort key value that is null is considered equal to another sort key value that is null. Whether a sort key value that is null is considered greater or less than a non-null value is implementation-defined, but all sort key values that are null shall either be considered greater than all non-null values or be considered less than all non-null values. PV_i is said to precede QV_i if the value of the <comparison predicate> “PV_i <comp op> QV_i “ is true for the applicable <comp op>. If PV_i and QV_i are not null and the result of “PV_i <comp op> QV_i “ is <u>*unknown*</u> , then the relative ordering of PV_i and QV_i is implementation-dependent.

8. *Rationale: Use the final <sort specification>.*

Replace General Rule 2) d) with:

2) d) In *TS*, the relative position of row *P* is before row *Q* if PV_n precedes QV_n for some *n* greater than 0 (zero) and less than or equal to the number of <sort specification>s and $PV_i = QV_i$ for all $i < n$. The relative order of two rows that are not distinct with respect to the <sort specification>s are implementation-dependent.

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

Replace Conformance Rule 9) with:

9) Without Feature S024, “Enhanced structured types”, a <value expression> that is a <sort key> shall not be of an ST-ordered declared type.

14.3 <fetch statement>

1. *Rationale: Correct use of non-terminals.*

Replace Syntax Rule 6) a) i) with:

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

Replace Syntax Rule 6) b) ii) with:

6) ii) For each <target specification> *TS1* that is an <SQL parameter reference>, the Syntax Rules of Subclause 9.2, “Store assignment”, apply to *TS1* and the corresponding column of table *T* as *TARGET* and *VALUE*, respectively.

Replace Syntax Rule 6) b) iii) with:

6) iii) For each <target specification> *TS2* that is a <host parameter specification>, the Syntax Rules of Subclause 9.1, “Retrieval assignment”, apply to each *TS2* and the corresponding column of table *T*, as *TARGET* and *VALUE*, respectively.

Replace General Rule 7) a) i) with:

7) i) If *TS* is an <SQL parameter reference>, then the General Rules of Subclause 9.2, “Store assignment”, apply to *TS* and the current row as *TARGET* and *VALUE*, respectively.

Replace General Rule 7) b) i) with:

7) i) If *TV* is an <SQL parameter reference>, then the General Rules of Subclause 9.2, “Store assignment”, apply to *TS* and *SV* as *TARGET* and *VALUE*, respectively.

2. *Rationale: Add missing Syntax and General Rules for <fetch statement>.*

Replace General rule 7) b) ii) with:

7) b) ii) If *TV* is a <host parameter specification>, then the General Rules of Subclause 9.1, “Retrieval assignment” are applied to *TV* and *SV*, as *TARGET* and *VALUE*, respectively.

14.5 <select statement: single row>

1. *Rationale: Correct use of non-terminals.*

Replace Syntax Rules 3) and 4) with:

3) For each <target specification> *TS* that is an <SQL parameter reference>, the Syntax Rules of Subclause 9.2, “Store assignment”, apply to *TS* and the corresponding element of the <select list>, as *TARGET* and *VALUE*, respectively.

4) For each <target specification> *TS* that is a <host parameter specification>, the Syntax Rules of Subclause 9.1, “Retrieval assignment”, apply to *TS* and the corresponding element of the <select list>, as *TARGET* and *VALUE*, respectively.

2. *Rationale: Supply the correct definition for non-determinism.*

Delete Syntax Rule 5).

Add the following Syntax Rule:

6.1) The <select statement: single row> is *possibly non-deterministic* if *S* is possibly non-deterministic.

3. *Rationale: Correct use of non-terminals.*

Replace Syntax Rules 4) and 5) with:

4) For each <target specification> *TS* that is an <SQL parameter reference>, the corresponding value in the row of *Q* is assigned to *TS* according to the General Rules of Subclause 9.2, "Store assignment", as *VALUE* and *TARGET*, respectively. The assignment of values to targets in the <select target list> is in an implementation-dependent order.

5) For each <target specification> *TS* that is a <host parameter specification>, the corresponding value in the row of *Q* is assigned to *TS* according to the General Rules of Subclause 9.1, "Retrieval assignment", as *VALUE* and *TARGET*, respectively. The assignment of values to targets in the <select target list> is in an implementation-dependent order.

14.6 <delete statement: positioned>

1. *Rationale: If ONLY is specified, then parentheses in <target table> are required, otherwise they are optional.*

In the Format, replace the production for <target table> with:

```
<target table> ::=
        <table name>
      | ONLY <left paren> <table name> <right paren>
```

2. *Rationale: Use the notion of identical to make specification of effect of <delete statement: positioned> more precise.*

Replace General Rule 7) with:

7) Let *R* be the current row of *CR*. Exactly one row *R1* in *LUT* such that each field in *R* is identical to the corresponding field in *R1* is identified for deletion from *LUT*.

3. *Rationale: Use of ONLY requires Feature S111, "ONLY in query expressions".*

Add the following Conformance Rule:

1) Without Feature S111, "ONLY in query expressions", a <target table> shall not specify ONLY.

14.8 <insert statement>

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

Delete Syntax Rule 3)

2. *Rationale: Remove a syntactic ambiguity, that VALUES (1) may be parsed either as <insert columns and source> ::= <from subquery> ::= <query expression> ::= <table value constructor> ::= VALUES (1) or <insert columns and source> ::= <from constructor> ::= <contextually typed table value constructor> ::= VALUES (1).*

Add the following Syntax Rule:

8.1) A <query expression> simply contained in a <from subquery> shall not be a <table value constructor>.

3. *Rationale: Use the correct non-terminal symbols.*

Replace Syntax Rule 9) with:

9) An <insert columns and source> that specifies DEFAULT VALUES is implicitly replaced by an <insert columns and source> that specifies a <contextually typed table value constructor> of the form
```
   VALUES (DEFAULT, DEFAULT, . . . , DEFAULT)
```
where the number of “DEFAULT” entries is equal to the number of columns of T.

4. *Rationale: Cater for self-referencing column in underlying table of T.*

Replace General Rules 7) b), c) and d) with:

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

Case:

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

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

c) For each object column in the candidate row, let C_i be the object column identified by the *i*-th <column name> in the <insert column list> and let $_{Svi}$ be the *i*-th value of *R*.

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

i) C_i is not a self-referencing column of *T;*

ii) C_i is a user-generated self-referencing column of *T;*

iii) C_i is a self-referencing column of *T* and OVERRIDING SYSTEM VALUE is specified;

the General Rules of Subclause 9.2, "Store assignment", are applied to C_i and $_{Svi}$ as *TARGET* and *SOURCE*, respectively.

14.9 <update statement: positioned>

1. *Rationale: Remove undefined SET ROW functionality.*

In the Format, replace the production for <update target> with:

```
<update target> ::=
       <object column>
     | <object column>
         <left bracket or trigraph>
         <simple value specification>
         <right bracket or trigraph>
```

2. *Rationale: Remove undefined SET ROW functionality.*

Delete Syntax Rule 9).

3. *Rationale: Remove undefined SET ROW functionality.*

Replace General Rule 15) with:

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

4. *Rationale: Use the notion of identical to make specification of <update statement: positioned> more precise.*

Replace General Rule 16) with:

16) Let *R1* be the candidate new row. The current row *R* of *CR* is replaced by *R1*. Exactly one row *TR* in *T* such that each field in *R* is identical to the corresponding field in *TR* is identified for replacement in *T*. Let *TR1* be a row consisting of the fields of *R1* and the fields of *TR* that have no corresponding fields in *R1*, ordered according to the order of their corresponding columns in *T*. *TR1* is the replacement row for *TR* and { (*TR*, *TR1*) } is the replacement set for *T*.

5. *Rationale: Remove undefined SET ROW functionality.*

Delete Conformance Rule 3).

14.10 <update statement: searched>

1. *Rationale: Remove undefined SET ROW functionality.*

Delete Syntax Rule 4).

2. *Rationale: Remove undefined SET ROW functionality.*

Replace General Rule 13) with:

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

14.11<temporary table declaration>

1. Rationale: Correct privileges for <temporary tables>s.

Replace General Rule 4) with:

4) A set of privilege descriptors is created that define the privileges INSERT, SELECT, UPDATE, DELETE, and REFERENCES on this table and INSERT, SELECT, UPDATE, and REFERENCES for every <column definition> in the table definition to *A*. These privileges are not grantable. The grantee is “PUBLIC”.

14.18 Effect of inserting a table into a derived table

1. Rationale: Editorial.

Replace General Rule 2) b) iii) with:

2) b) iii) The candidate rows are added to the corresponding S_i.

15.2 <return statement>

1. Rationale: Move the check for the most specific type of the returned value to <routine invocation>.

Replace General Rules 1) and 2) with:

1) Let *RV* be the value of *VE*. Let *RT* be an item of data type *RDT*, arbitrarily chosen.

16.4 <savepoint statement>

1. Rationale: Remove the anomaly in <savepoint specifier>.

In the Format, replace the production for <savepoint specifier> with:

```
<savepoint specifier> ::= <savepoint name>
```

Delete Syntax Rule 1)

Replace General Rules 1) and 2) with:

1) Let *S* be the <savepoint name>.

Delete General Rule 5)

16.5 <release savepoint statement>

1. Rationale: Remove the anomaly in <savepoint specifier>.

Delete Syntax Rule 1)

Replace General Rule 1) with:

1) Let *S* be the <savepoint name>.

16.7 <rollback statement>

1. *Rationale: Remove the anomaly in <savepoint specifier>.*

Delete Syntax Rule 1)

Replace General Rule 3) a) with:

3) a) Let *S* be the <savepoint name>.

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

Replace General Rule 3) f) with:

3) f) Every valid locator generated in the current SQL-transaction subsequent to the establishment of *S* is marked invalid.

19.1 <get diagnostics statement>

1. *Rationale: Editorial. Non-reserved word - DYNAMIC_FUNCTION - not defined. Only defined in Part 5.*

Replace General Rule 3) p) with:

3) p) The values of CONNECTION_NAME and SERVER_NAME are respectively

Case:

i) If COMMAND_FUNCTION identifies an <SQL connection statement>, then the <connection name> and the <SQL-server name> specified by or implied by the <SQL connection statement>.

ii) Otherwise, the <connection name> and <SQL-server name> of the SQL-session in which the condition was raised.

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

Replace the following row in Table 26, "SQL-statement codes":

SQL-statement	Identifier	Code
<grant statement>	GRANT	48

with:

SQL-statement	Identifier	Code
<grant privilege statement>	GRANT	48

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

Add the following rows to Table 26, "SQL-statement codes":

SQL-statement	Identifier	Code
<user-defined cast definition>	CREATE CAST	52

<drop-user-defined cast definition>	DROP CAST	78
<temporary table declaration>	TEMPORARY TABLE	93

20.11 ATTRIBUTES view

1. *Rationale: Add missing column to the definition of the ATTRIBUTES view.*

In the Definition, replace the view definition with:

```
CREATE VIEW ATTRIBUTES AS
SELECT DISTINCT
       UDT_CATALOG, UDT_SCHEMA, UDT_NAME, A.ATTRIBUTE_NAME, ORDINAL_POSITION,
       CASE WHEN EXISTS ( SELECT *
                          FROM DEFINITION_SCHEMA.SCHEMATA AS S
                          WHERE ( UDT_CATALOG, UDT_SCHEMA )
                                = ( S.CATALOG_NAME, S.SCHEMA_NAME )
                                AND
                                ( SCHEMA_OWNER IN ( 'PUBLIC', CURRENT_USER )
                                  OR
                                  SCHEMA_OWNER IN ( SELECT ROLE_NAME
                                                    FROM ENABLED_ROLES ) ) )
                THEN ATTRIBUTE_DEFAULT
              ELSE NULL
         END AS ATTRIBUTE_DEFAULT,
       IS_NULLABLE, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH,
       CHARACTER_OCTET_LENGTH, C1.CHARACTER_SET_CATALOG,
       C1.CHARACTER_SET_SCHEMA, C1.CHARACTER_SET_NAME,
       D1.COLLATION_CATALOG, D1.COLLATION_SCHEMA, D1.COLLATION_NAME,
       NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
       DATETIME_PRECISION, INTERVAL_TYPE, INTERVAL_PRECISION,
       D1.USER_DEFINED_TYPE_CATALOG AS ATTRIBUTE_UDT_CATALOG,
       D1.USER_DEFINED_TYPE_SCHEMA AS ATTRIBUTE_UDT_SCHEMA,
       D1.USER_DEFINED_TYPE_NAME AS ATTRIBUTE_UDT_NAME,
       D1.SCOPE_CATALOG, D1.SCOPE_SCHEMA, D1.SCOPE_NAME,
       MAXIMUM_CARDINALITY, A.DTD_IDENTIFIER, CHECK_REFERENCES,
       IS_DERIVED_REFERENCE_ATTRIBUTE
  FROM ( DEFINITION_SCHEMA.ATTRIBUTES AS A
         LEFT JOIN ( DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D1
                     LEFT JOIN DEFINITION_SCHEMA.COLLATIONS AS C1
                       ON ( ( C1.COLLATION_CATALOG, C1.COLLATION_SCHEMA,
                              C1.COLLATION_NAME )
                            = ( D1.COLLATION_CATALOG, D1.COLLATION_SCHEMA,
                                D1.COLLATION_NAME ) ) )
           ON ( ( A.UDT_CATALOG, A.UDT_SCHEMA, A.UDT_NAME,
                  'USER-DEFINED TYPE', A.DTD_IDENTIFIER )
                = ( D1.OBJECT_CATALOG, D1.OBJECT_SCHEMA, D1.OBJECT_NAME,
                    D1.OBJECT_TYPE, D1.DTD_IDENTIFIER ) ) )
 WHERE ( A.UDT_CATALOG, A.UDT_SCHEMA, A.UDT_NAME )
           IN ( SELECT UDT_CATALOG, UDT_SCHEMA, UDT_NAME
                  FROM DEFINITION_SCHEMA.USER_DEFINED_TYPE_PRIVILEGES
                 WHERE ( SCHEMA_OWNER IN ( 'PUBLIC', CURRENT_USER )
                         OR
                         SCHEMA_OWNER IN ( SELECT ROLE_NAME
                                           FROM ENABLED_ROLES ) ) )
       AND
       A.UDT_CATALOG = ( SELECT CATALOG_NAME
                         FROM INFORMATION_SCHEMA_CATALOG_NAME );
```

20.18 COLUMNS view

1. *Rationale: Editorial - typographical errors.*

```
CREATE VIEW COLUMNS AS
SELECT DISTINCT
       TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
       C.COLUMN_NAME, ORDINAL_POSITION,
       CASE WHEN EXISTS ( SELECT *
                            FROM DEFINITION_SCHEMA.SCHEMATA AS S
                          WHERE ( TABLE_CATALOG, TABLE_SCHEMA )
                                  = (S.CATALOG_NAME, S.SCHEMA_NAME )
                                AND
                                ( SCHEMA_OWNER IN ( 'PUBLIC', CURRENT_USER )
                                  OR
                                  SCHEMA_OWNER IN ( SELECT ROLE_NAME
                                                      FROM ENABLED_ROLES ) ) )
              THEN COLUMN_DEFAULT
            ELSE NULL
         END AS COLUMN_DEFAULT,
       IS_NULLABLE,
       COALESCE ( D1.DATA_TYPE, D2.DATA_TYPE ) AS DATA_TYPE,
       COALESCE ( D1.CHARACTER_MAXIMUM_LENGTH, D2.CHARACTER_MAXIMUM_LENGTH )
         AS CHARACTER_MAXIMUM_LENGTH,
       COALESCE ( D1.CHARACTER_OCTET_LENGTH, D2.CHARACTER_OCTET_LENGTH )
         AS CHARACTER_OCTET_LENGTH,
       COALESCE ( D1.NUMERIC_PRECISION, D2.NUMERIC_PRECISION )
         AS NUMERIC_PRECISION,
       COALESCE ( D1.NUMERIC_PRECISION_RADIX, D2.NUMERIC_PRECISION_RADIX )
         AS NUMERIC_PRECISION_RADIX,
       COALESCE ( D1.NUMERIC_SCALE, D2.NUMERIC_SCALE )
         AS NUMERIC_SCALE,
       COALESCE ( D1.DATETIME_PRECISION, D2.DATETIME_PRECISION )
         AS DATETIME_PRECISION,
       COALESCE ( D1.INTERVAL_TYPE, D2.INTERVAL_TYPE )
         AS INTERVAL_TYPE,
       COALESCE ( D1.INTERVAL_PRECISION, D2.INTERVAL_PRECISION )
         AS INTERVAL_PRECISION,
       COALESCE ( C1.CHARACTER_SET_CATALOG, C2.CHARACTER_SET_CATALOG )
         AS CHARACTER_SET_CATALOG,
       COALESCE ( C1.CHARACTER_SET_SCHEMA, C2.CHARACTER_SET_SCHEMA )
         AS CHARACTER_SET_SCHEMA,
       COALESCE ( C1.CHARACTER_SET_NAME, C2.CHARACTER_SET_NAME )
         AS CHARACTER_SET_NAME,
       COALESCE ( D1.COLLATION_CATALOG, D2.COLLATION_CATALOG )
         AS COLLATION_CATALOG,
       COALESCE ( D1.COLLATION_SCHEMA, D2.COLLATION_SCHEMA )
         AS COLLATION_SCHEMA,
       COALESCE ( D1.COLLATION_NAME, D2.COLLATION_NAME )
         AS COLLATION_NAME,
       DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME,
       COALESCE ( D1.USER_DEFINED_TYPE_CATALOG, D2.USER_DEFINED_TYPE_CATALOG )
         AS UDT_CATALOG,
       COALESCE ( D1.USER_DEFINED_TYPE_SCHEMA, D2.USER_DEFINED_TYPE_SCHEMA )
         AS UDT_SCHEMA,
       COALESCE ( D1.USER_DEFINED_TYPE_NAME, D2.USER_DEFINED_TYPE_NAME )
         AS UDT_NAME,
       COALESCE ( D1.SCOPE_CATALOG, D2.SCOPE_CATALOG ) AS SCOPE_CATALOG,
       COALESCE ( D1.SCOPE_SCHEMA, D2.SCOPE_SCHEMA ) AS SCOPE_SCHEMA,
       COALESCE ( D1.SCOPE_NAME, D2.SCOPE_NAME ) AS SCOPE_NAME,
       COALESCE ( D1.MAXIMUM_CARDINALITY, D2.MAXIMUM_CARDINALITY )
         AS MAXIMUM_CARDINALITY,
       COALESCE ( D1.DTD_IDENTIFIER, D2.DTD_IDENTIFIER ) AS DTD_IDENTIFIER,
       IS_SELF_REFERENCING
  FROM ( DEFINITION_SCHEMA.COLUMNS AS C
         LEFT JOIN ( DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D1
                     LEFT JOIN DEFINITION_SCHEMA.COLLATIONS AS C1
```

```
                              ON ( ( C1.COLLATION_CATALOG, C1.COLLATION_SCHEMA,
                                     C1.COLLATION_NAME )
                                     = ( D1.COLLATION_CATALOG, D1.COLLATION_SCHEMA,
                                         D1.COLLATION_NAME ) ) )
                  ON ( ( C.TABLE_CATALOG, C.TABLE_SCHEMA, C.TABLE_NAME,
                         'TABLE', C.DTD_IDENTIFIER )
                         = ( D1.OBJECT_CATALOG, D1.OBJECT_SCHEMA, D1.OBJECT_NAME,
                             D1.OBJECT_TYPE, D1.DTD_IDENTIFIER ) ) )
              LEFT JOIN ( DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D2
                          LEFT JOIN DEFINITION_SCHEMA.COLLATIONS AS C2
                            ON ( ( C2.COLLATION_CATALOG, C2.COLLATION_SCHEMA,
                                   C2.COLLATION_NAME )
                                   = ( D2.COLLATION_CATALOG, D2.COLLATION_SCHEMA,
                                       D2.COLLATION_NAME ) ) )
                ON ( ( C.DOMAIN_CATALOG, C.DOMAIN_SCHEMA, C.DOMAIN_NAME,
                       'DOMAIN', C.DTD_IDENTIFIER )
                       = ( D2.OBJECT_CATALOG, D2.OBJECT_SCHEMA, D2.OBJECT_NAME,
                           D2.OBJECT_TYPE, D2.DTD_IDENTIFIER ) )
        WHERE ( C.TABLE_CATALOG, C.TABLE_SCHEMA, C.TABLE_NAME, C.COLUMN_NAME )
                IN ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME
                       FROM DEFINITION_SCHEMA.COLUMN_PRIVILEGES
                      WHERE ( SCHEMA_OWNER IN ( 'PUBLIC', CURRENT_USER )
                              OR
                              SCHEMA_OWNER IN ( SELECT ROLE_NAME
                                                  FROM ENABLED_ROLES ) ) )
              AND
              C.TABLE_CATALOG = ( SELECT CATALOG_NAME
                                    FROM INFORMATION_SCHEMA_CATALOG_NAME );
```

20.19 CONSTRAINT_COLUMN_USAGE view

1. *Rationale: Editorial - typographical errors.*

In the Definition, replace the <from clause> with:

```
  FROM ( ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME,
                  CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
             FROM DEFINITION_SCHEMA.CHECK_COLUMN_USAGE )
         UNION
         ( SELECT PK.TABLE_CATALOG, PK.TABLE_SCHEMA, PK.TABLE_NAME,
                  PK.COLUMN_NAME,
                  FK.CONSTRAINT_CATALOG, FK.CONSTRAINT_SCHEMA, FK.CONSTRAINT_NAME
             FROM DEFINITION_SCHEMA.REFERENTIAL_CONSTRAINTS AS FK
                  JOIN DEFINITION_SCHEMA.KEY_COLUMN_USAGE AS PK
                    ON ( FK.UNIQUE_CONSTRAINT_CATALOG,
                         FK.UNIQUE_CONSTRAINT_SCHEMA,
                         FK.UNIQUE_CONSTRAINT_NAME )
                       = ( PK.CONSTRAINT_CATALOG, PK.CONSTRAINT_SCHEMA,
                           PK.CONSTRAINT_NAME ) )
         UNION
         ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME,
                  CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
             FROM DEFINITION_SCHEMA.KEY_COLUMN_USAGE
                  NATURAL JOIN        DEFINITION_SCHEMA.TABLE_CONSTRAINTS
            WHERE CONSTRAINT_TYPE IN ( ’UNIQUE’, ’PRIMARY KEY’ ) ) )
       JOIN DEFINITION_SCHEMA.SCHEMATA
             ON ( ( TABLE_CATALOG, TABLE_SCHEMA )
                  = ( CATALOG_NAME, SCHEMA_NAME ) )
```

20.21 DATA_TYPE_PRIVILEGES view

1. *Rationale: Editorial - typographical error and missing grant statement.*

Replace the Definition with:

```
CREATE VIEW DATA_TYPE_PRIVILEGES
( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
  OBJECT_TYPE, DTD_IDENTIFIER )
AS
SELECT USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
       USER_DEFINED_TYPE_NAME, 'USER-DEFINED TYPE', DTD_IDENTIFIER
  FROM ATTRIBUTES
UNION
SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
       'TABLE', DTD_IDENTIFIER
  FROM COLUMNS
UNION
SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
       'DOMAIN', DTD_IDENTIFIER
  FROM DOMAINS
UNION
SELECT USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
       USER_DEFINED_TYPE_NAME, 'USER-DEFINED TYPE', DTD_IDENTIFIER
  FROM METHOD_SPECIFICATIONS
UNION
SELECT USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
       USER_DEFINED_TYPE_NAME, 'USER-DEFINED TYPE', DTD_IDENTIFIER
  FROM METHOD_SPECIFICATION_PARAMETERS
UNION
SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
       'ROUTINE', DTD_IDENTIFIER
  FROM PARAMETERS
UNION
SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
       'ROUTINE', DTD_IDENTIFIER
  FROM ROUTINES
UNION
SELECT USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
       USER_DEFINED_TYPE_NAME, 'USER-DEFINED TYPE', SOURCE_DTD_IDENTIFIER
  FROM USER_DEFINED_TYPES
UNION
SELECT USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
       USER_DEFINED_TYPE_NAME, 'USER-DEFINED TYPE', REF_DTD_IDENTIFIER
  FROM USER_DEFINED_TYPES;

GRANT SELECT ON TABLE DATA_TYPE_PRIVILEGES
  TO PUBLIC WITH GRANT OPTION;
```

20.26 DOMAINS view

1. *Rationale: Editorial - typographical error.*

In the Definition, replace the <where clause> with:

```
WHERE ( ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME, 'DOMAIN' )
        IN ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, OBJECT_TYPE
               FROM DEFINITION_SCHEMA.USAGE_PRIVILEGES
             WHERE ( SCHEMA_OWNER IN ( 'PUBLIC', CURRENT_USER )
                     OR
                     SCHEMA_OWNER IN ( SELECT ROLE_NAME
                                         FROM ENABLED_ROLES ) ) )
        OR
        ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME )
```

```
            IN ( SELECT DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME
                   FROM COLUMNS ) )
        AND
        DOMAIN_CATALOG = ( SELECT CATALOG_NAME
                             FROM INFORMATION_SCHEMA_CATALOG_NAME );
```

20.27 ELEMENT_TYPES view

1. *Rationale: Remove irrelevant column from the definition of the ELEMENT_TYPES view.*

In the Definition, replace the view definition with:

```
CREATE VIEW ELEMENT_TYPES AS
  SELECT DISTINCT
         E.OBJECT_CATALOG, E.OBJECT_SCHEMA, E.OBJECT_NAME,
         E.OBJECT_TYPE, ARRAY_TYPE_IDENTIFIER, DATA_TYPE,
         CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
         CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME,
         COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,
         NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
         DATETIME_PRECISION, INTERVAL_TYPE, INTERVAL_PRECISION,
         USER_DEFINED_TYPE_CATALOG AS UDT_CATALOG,
         USER_DEFINED_TYPE_SCHEMA AS UDT_SCHEMA,
         USER_DEFINED_TYPE_NAME AS UDT_NAME,
         SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME,
         MAXIMUM_CARDINALITY, E.DTD_IDENTIFIER
    FROM DEFINITION_SCHEMA.ELEMENT_TYPES AS E
         JOIN ( DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D
                LEFT JOIN DEFINITION_SCHEMA.COLLATIONS AS S
                  USING ( COLLATION_CATALOG, COLLATION_SCHEMA,
                          COLLATION_NAME ) )
           ON ( ( E.OBJECT_CATALOG, E.OBJECT_SCHEMA, E.OBJECT_NAME,
                  E.OBJECT_TYPE, E.DTD_IDENTIFIER )
                  = ( D.OBJECT_CATALOG, D.OBJECT_SCHEMA, D.OBJECT_NAME,
                      D.OBJECT_TYPE, D.DTD_IDENTIFIER ) )
   WHERE ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
           OBJECT_TYPE, ROOT_DTD_IDENTIFIER )
           IN ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                       OBJECT_TYPE, DTD_IDENTIFIER
                  FROM INFORMATION_SCHEMA.DATA_TYPE_PRIVILEGES );
```

20.29 FIELDS view

1. *Rationale: Remove irrelevant columns from the definition of the FIELDS view.*

In the Definition, replace the view definition with:

```
CREATE VIEW FIELDS AS
SELECT DISTINCT
       F.OBJECT_CATALOG, F.OBJECT_SCHEMA, F.OBJECT_NAME,
       F.OBJECT_TYPE, ROW_IDENTIFIER, FIELD_NAME,
       ORDINAL_POSITION, DATA_TYPE,
       CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
       CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME,
       COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,
       NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
       DATETIME_PRECISION, INTERVAL_TYPE, INTERVAL_PRECISION,
       USER_DEFINED_TYPE_CATALOG AS UDT_CATALOG,
       USER_DEFINED_TYPE_SCHEMA AS UDT_SCHEMA,
       USER_DEFINED_TYPE_NAME AS UDT_NAME,
       SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME,
       MAXIMUM_CARDINALITY, F.DTD_IDENTIFIER
  FROM DEFINITION_SCHEMA.FIELDS AS F
```

```
        JOIN ( DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D
               LEFT JOIN DEFINITION_SCHEMA.COLLATIONS AS S
                 USING ( COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) )
          ON ( ( F.OBJECT_CATALOG, F.OBJECT_SCHEMA, F.OBJECT_NAME,
                 F.OBJECT_TYPE, F.DTD_IDENTIFIER )
                 = ( D.OBJECT_CATALOG, D.OBJECT_SCHEMA, D.OBJECT_NAME,
                     D.OBJECT_TYPE, D.DTD_IDENTIFIER ) )
  WHERE ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
          OBJECT_TYPE, ROOT_DTD_IDENTIFIER )
          IN ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                      OBJECT_TYPE, DTD_IDENTIFIER
                 FROM INFORMATION_SCHEMA.DATA_TYPE_PRIVILEGES );
```

20.31 METHOD_SPECIFICATION_PARAMETERS view

1. *Rationale: Editorial - typographical error and missing grant statement.*

In the Definition, replace the <where clause> with:

```
WHERE ( M.USER_DEFINED_TYPE_CATALOG, M.USER_DEFINED_TYPE_SCHEMA,
        M.USER_DEFINED_TYPE_NAME )
       IN ( SELECT USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                   USER_DEFINED_TYPE_NAME
              FROM DEFINITION_SCHEMA.USER_DEFINED_TYPE_PRIVILEGES
             WHERE ( SCHEMA_OWNER IN ( 'PUBLIC', CURRENT_USER )
                     OR
                     SCHEMA_OWNER IN ( SELECT ROLE_NAME
                                         FROM ENABLED_ROLES ) ) )
      AND
     M.USER_DEFINED_TYPE_CATALOG = ( SELECT CATALOG_NAME
                                       FROM INFORMATION_SCHEMA_CATALOG_NAME );
```

In the Definition, replace the <grant statement> with:

```
GRANT SELECT ON TABLE METHOD_SPECIFICATION_PARAMETERS
  TO PUBLIC WITH GRANT OPTION;
```

20.32 METHOD_SPECIFICATIONS view

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

In the Definition, replace the view definition with:

```
CREATE VIEW METHOD_SPECIFICATIONS AS
  SELECT M.SPECIFIC_CATALOG, M.SPECIFIC_SCHEMA, M.SPECIFIC_NAME,
         M.USER_DEFINED_TYPE_CATALOG AS UDT_CATALOG,
         M.USER_DEFINED_TYPE_SCHEMA AS UDT_SCHEMA,
         M.USER_DEFINED_TYPE_NAME AS UDT_NAME,
         M.METHOD_NAME, IS_STATIC, IS_OVERRIDING,
         IS_CONSTRUCTOR,
         D.DATA_TYPE, D.CHARACTER_MAXIMUM_LENGTH, D.CHARACTER_OCTET_LENGTH,
         C.CHARACTER_SET_CATALOG, C.CHARACTER_SET_SCHEMA,
         C.CHARACTER_SET_NAME,
         D.COLLATION_CATALOG, D.COLLATION_SCHEMA, D.COLLATION_NAME,
         D.NUMERIC_PRECISION, D.NUMERIC_PRECISION_RADIX, D.NUMERIC_SCALE,
         D.DATETIME_PRECISION, D.INTERVAL_TYPE, D.INTERVAL_PRECISION,
         D.USER_DEFINED_TYPE_CATALOG AS RETURN_UDT_CATALOG,
         D.USER_DEFINED_TYPE_SCHEMA AS RETURN_UDT_SCHEMA,
         D.USER_DEFINED_TYPE_NAME AS RETURN_UDT_NAME,
         D.SCOPE_CATALOG, D.SCOPE_SCHEMA, D.SCOPE_NAME,
         D.MAXIMUM_CARDINALITY, D.DTD_IDENTIFIER, M.METHOD_LANGUAGE,
         M.PARAMETER_STYLE, M.IS_DETERMINISTIC, M.SQL_DATA_ACCESS,
         M.IS_NULL_CALL,
         M.TO_SQL_SPECIFIC_CATALOG, M.TO_SQL_SPECIFIC_SCHEMA,
         M.TO_SQL_SPECIFIC_NAME,
         M.CREATED, M.LAST_ALTERED
    FROM ( DEFINITION_SCHEMA.METHOD_SPECIFICATIONS M
           JOIN ( DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR D
                  LEFT JOIN DEFINITION_SCHEMA.COLLATIONS AS C
                    ON ( C.COLLATION_CATALOG, C.COLLATION_SCHEMA,
                         C.COLLATION_NAME )
                         = ( D.COLLATION_CATALOG, D.COLLATION_SCHEMA,
                             D.COLLATION_NAME ) )
             ON ( M.USER_DEFINED_TYPE_CATALOG, M.USER_DEFINED_TYPE_SCHEMA,
                  M.USER_DEFINED_TYPE_NAME, 'USER-DEFINED TYPE',
                  DTD_IDENTIFIER )
                  = ( D.OBJECT_CATALOG, D.OBJECT_SCHEMA, D.OBJECT_NAME,
                      D.OBJECT_TYPE, D.DTD_IDENTIFIER ) )
   WHERE ( M.USER_DEFINED_TYPE_CATALOG, M.USER_DEFINED_TYPE_SCHEMA,
           M.USER_DEFINED_TYPE_NAME )
           IN ( SELECT USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
                       USER_DEFINED_TYPE_NAME
                  FROM DEFINITION_SCHEMA.USER_DEFINED_TYPE_PRIVILEGES
                 WHERE ( SCHEMA_OWNER IN ( 'PUBLIC', CURRENT_USER )
                       OR
                       SCHEMA_OWNER IN ( SELECT ROLE_NAME
                                           FROM ENABLED_ROLES ) ) )
         AND
         M.USER_DEFINED_TYPE_CATALOG
         = ( SELECT CATALOG_NAME
               FROM INFORMATION_SCHEMA_CATALOG_NAME );
```

20.33 PARAMETERS view

1. *Rationale: Editorial - typographical error.*

In the Definition, replace the <from clause> with:

```
FROM ( DEFINITION_SCHEMA.PARAMETERS P1
       LEFT JOIN    ( DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR D1
                   LEFT JOIN DEFINITION_SCHEMA.COLLATIONS C1
                     ON ( ( C1.COLLATION_CATALOG, C1.COLLATION_SCHEMA,
                            C1.COLLATION_NAME )
                          = ( D1.COLLATION_CATALOG, D1.COLLATION_SCHEMA,
                              D1.COLLATION_NAME ) ) )
         ON ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
              'ROUTINE', P1.DTD_IDENTIFIER )
            = ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                OBJECT_TYPE, D1.DTD_IDENTIFIER ) )
       JOIN DEFINITION_SCHEMA.ROUTINES R1
         ON ( ( P1.SPECIFIC_CATALOG, P1.SPECIFIC_SCHEMA, P1.SPECIFIC_NAME )
              = ( R1.SPECIFIC_CATALOG, R1.SPECIFIC_SCHEMA, R1.SPECIFIC_NAME ) )
```

20.34 REFERENCED_TYPES view

1. *Rationale: Remove irrelevant column from the definition of the REFERENCED_TYPES view.*

In the Definition, replace the view definition with:

```
CREATE VIEW REFERENCED_TYPES AS
  SELECT DISTINCT
         R.OBJECT_CATALOG, R.OBJECT_SCHEMA, R.OBJECT_NAME,
         R.OBJECT_TYPE, REFERENCE_TYPE_IDENTIFIER, DATA_TYPE,
         CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
             CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME,
             COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,
             NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
             DATETIME_PRECISION, INTERVAL_TYPE, INTERVAL_PRECISION,
             USER_DEFINED_TYPE_CATALOG AS UDT_CATALOG,
             USER_DEFINED_TYPE_SCHEMA AS UDT_SCHEMA,
             USER_DEFINED_TYPE_NAME AS UDT_NAME,
             SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME,
             MAXIMUM_CARDINALITY, R.DTD_IDENTIFIER
        FROM ( DEFINITION_SCHEMA.REFERENCED_TYPES AS R
               JOIN ( DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D
                      LEFT JOIN DEFINITION_SCHEMA.COLLATIONS AS S
                        USING ( COLLATION_CATALOG, COLLATION_SCHEMA,
                                COLLATION_NAME ) )
                 ON ( ( R.OBJECT_CATALOG, R.OBJECT_SCHEMA,
                        R.OBJECT_NAME,R.OBJECT_TYPE, R.DTD_IDENTIFIER )
                        = ( D.OBJECT_CATALOG, D.OBJECT_SCHEMA, D.OBJECT_NAME,
                            D.OBJECT_TYPE, D.DTD_IDENTIFIER ) ) )
       WHERE ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
               OBJECT_TYPE, ROOT_DTD_IDENTIFIER )
               IN ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                           OBJECT_TYPE, DTD_IDENTIFIER
                      FROM INFORMATION_SCHEMA.DATA_TYPE_PRIVILEGES );
```

20.38 ROLE_TABLE_GRANTS view

1. *Rationale: Editorial - typographical error.*

In the Definition, replace the <where clause> with:

```
  WHERE ( GRANTEE IN ( SELECT ROLE_NAME
                         FROM ENABLED_ROLES )
          OR
          GRANTOR IN ( SELECT ROLE_NAME
                         FROM ENABLED_ROLES ) )
     AND TABLE_CATALOG = ( SELECT CATALOG_NAME
                             FROM INFORMATION_SCHEMA_CATALOG_NAME );
```

20.56 TABLES view

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

Replace the Definition with:

```
  CREATE VIEW TABLES AS
    SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
           TABLE_TYPE, SELF_REFERENCING_COLUMN_NAME, REFERENCE_GENERATION,
           USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
           USER_DEFINED_TYPE_NAME, IS_INSERTABLE_INTO
      FROM DEFINITION_SCHEMA.TABLES
     WHERE ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME )
             IN ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
                    FROM DEFINITION_SCHEMA.TABLE_PRIVILEGES
                   WHERE GRANTEE IN ( 'PUBLIC', CURRENT_USER )
                  UNION
                  SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
                    FROM DEFINITION_SCHEMA.COLUMN_PRIVILEGES
                   WHERE ( SCHEMA_OWNER IN ( 'PUBLIC', CURRENT_USER )
                           OR
                           SCHEMA_OWNER IN ( SELECT ROLE_NAME
                                               FROM ENABLED_ROLES ) ) )
             AND
             TABLE_CATALOG = ( SELECT CATALOG_NAME
                                             FROM INFORMATION_SCHEMA_CATALOG_NAME );

        GRANT SELECT ON TABLE TABLES
          TO PUBLIC WITH GRANT OPTION;
```

20.62 TRIGGERS view

1. *Rationale: Extend TRIGGERS view to allow for the representation of all possible elements of <old or new values alias list>, and delete stray reference to " CONDITION_TIMING," and editorially remove erroneous text.*

Replace the Definition with:

```
CREATE VIEW TRIGGERS AS
  SELECT TRIGGER_CATALOG, TRIGGER_SCHEMA, TRIGGER_NAME,
         EVENT_MANIPULATION,
         EVENT_OBJECT_CATALOG, EVENT_OBJECT_SCHEMA, EVENT_OBJECT_TABLE,
         ACTION_ORDER, ACTION_CONDITION, ACTION_STATEMENT,
         ACTION_ORIENTATION, ACTION_TIMING,
         ACTION_REFERENCE_OLD_TABLE, ACTION_REFERENCE_NEW_TABLE,
         ACTION_REFERENCE_OLD_ROW, ACTION_REFERENCE_NEW_ROW, CREATED
    FROM DEFINITION_SCHEMA.TRIGGERS
         JOIN DEFINITION_SCHEMA.SCHEMATA S
           ON ( TRIGGER_CATALOG, TRIGGER_SCHEMA )
                  = ( S.CATALOG_NAME, S.SCHEMA_NAME )
   WHERE ( SCHEMA_OWNER = CURRENT_USER
           OR
           SCHEMA_OWNER IN ( SELECT ROLE_NAME
                               FROM ENABLED_ROLES ) )
         AND
         TRIGGER_CATALOG = ( SELECT CATALOG_NAME
                               FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE TRIGGERS
  TO PUBLIC WITH GRANT OPTION;
```

20.68 VIEWS view

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

Replace the Definition with:

```
CREATE VIEW VIEWS AS
  SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
         CASE WHEN EXISTS ( SELECT *
                              FROM DEFINITION_SCHEMA.SCHEMATA AS S
                            WHERE ( TABLE_CATALOG, TABLE_SCHEMA )
                                  = ( S.CATALOG_NAME, S.SCHEMA_NAME )
                                  AND
                                  ( SCHEMA_OWNER = CURRENT_USER
                                    OR
                                    SCHEMA_OWNER IN ( SELECT ROLE_NAME
                                                        FROM ENABLED_ROLES )
                                  ) )
                  THEN VIEW_DEFINITION
                ELSE NULL
           END AS VIEW_DEFINITION,
         CHECK_OPTION, IS_UPDATABLE
    FROM DEFINITION_SCHEMA.VIEWS
   WHERE ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME )
           IN ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
                  FROM TABLES )
         AND
         TABLE_CATALOG = ( SELECT CATALOG_NAME
                            FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE VIEWS
  TO PUBLIC WITH GRANT OPTION;
```

20.69 Short name views

1. *Rationale: Add missing column to the definition of the ATTRIBUTES view.*

In the Definition, replace the view definition for ATTRIBUTES_S with:

```
CREATE VIEW ATTRIBUTES_S
  ( UDT_CATALOG, UDT_SCHEMA, UDT_NAME,
    ATTRIBUTE_NAME, ORDINAL_POSITION, ATTRIBUTE_DEFAULT,
    IS_NULLABLE, DATA_TYPE, CHAR_MAX_LENGTH,
    CHAR_OCTET_LENGTH, CHAR_SET_CATALOG, CHAR_SET_SCHEMA,
    CHARACTER_SET_NAME, COLLATION_CATALOG, COLLATION_SCHEMA,
    COLLATION_NAME, NUMERIC_PRECISION, NUMERIC_PREC_RADIX,
    NUMERIC_SCALE, DATETIME_PRECISION, INTERVAL_TYPE,
    INTERVAL_PRECISION, DOMAIN_CATALOG, DOMAIN_SCHEMA,
    DOMAIN_NAME, ATT_UDT_CAT, ATT_UDT_SCHEMA,
    ATT_UDT_NAME, SCOPE_CATALOG, SCOPE_SCHEMA,
    SCOPE_NAME, MAX_CARDINALITY, DTD_IDENTIFIER,
    CHECK_REFERENCES, IS_DERIVED_REF_ATT ) AS
  SELECT UDT_CATALOG, UDT_SCHEMA, UDT_NAME,
         ATTRIBUTE_NAME, ORDINAL_POSITION, COLUMN_DEFAULT,
         IS_NULLABLE, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH,
         CHARACTER_OCTET_LENGTH, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
         CHARACTER_SET_NAME, COLLATION_CATALOG, COLLATION_SCHEMA,
         COLLATION_NAME, NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
         NUMERIC_SCALE, DATETIME_PRECISION, INTERVAL_TYPE,
         INTERVAL_PRECISION, DOMAIN_CATALOG, DOMAIN_SCHEMA,
         DOMAIN_NAME, ATTRIBUTE_UDT_CATALOG, ATTRIBUTE_UDT_SCHEMA,
         ATTRIBUTE_UDT_NAME, SCOPE_CATALOG, SCOPE_SCHEMA,
         SCOPE_NAME, MAXIMUM_CARDINALITY, DTD_IDENTIFIER, CHECK_REFERENCES,
         IS_DERIVED_REFERENCE_ATTRIBUTE
    FROM INFORMATION_SCHEMA.ATTRIBUTES;
```

2. *Rationale: Remove irrelevant column from the definition of the ELEMENT_TYPES view.*

In the Definition, replace the view definition for ELEMENT_TYPES_S with:

```
CREATE VIEW ELEMENT_TYPES_S
  ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
    OBJECT_TYPE, ARRAY_TYPE_ID, DATA_TYPE,
    CHAR_MAX_LENGTH, CHAR_OCTET_LENGTH, CHAR_SET_CATALOG,
    CHAR_SET_SCHEMA, CHARACTER_SET_NAME, COLLATION_CATALOG,
    COLLATION_SCHEMA, COLLATION_NAME, NUMERIC_PRECISION,
    NUMERIC_PREC_RADIX, NUMERIC_SCALE, DATETIME_PRECISION,
    INTERVAL_TYPE, INTERVAL_PRECISION,
    UDT_CATALOG, UDT_SCHEMA, UDT_NAME,
    SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME,
    MAX_CARDINALITY, DTD_IDENTIFIER ) AS
  SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
         OBJECT_TYPE, ARRAY_TYPE_IDENTIFIER, DATA_TYPE,
         CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
         CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME,
         COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,
         NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
         DATETIME_PRECISION, INTERVAL_TYPE, INTERVAL_PRECISION,
         UDT_CATALOG, UDT_SCHEMA, UDT_NAME,
         SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME,
         MAXIMUM_CARDINALITY, DTD_IDENTIFIER
    FROM INFORMATION_SCHEMA.ELEMENT_TYPES;
```

3. *Rationale: Remove irrelevant columns from the definition of the FIELDS view.*

In the Definition, replace the view definition for the FIELDS_S view with:

```
CREATE VIEW FIELDS_S
  ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
    OBJECT_TYPE, ROW_IDENTIFIER, FIELD_NAME,
    ORDINAL_POSITION, DATA_TYPE,
    CHAR_MAX_LENGTH, CHAR_OCTET_LENGTH, CHAR_SET_CATALOG,
    CHAR_SET_SCHEMA, CHARACTER_SET_NAME, COLLATION_CATALOG,
    COLLATION_SCHEMA, COLLATION_NAME, NUMERIC_PRECISION,
    NUMERIC_PREC_RADIX, NUMERIC_SCALE, DATETIME_PRECISION,
    INTERVAL_TYPE, INTERVAL_PRECISION,
    UDT_CATALOG, UDT_SCHEMA, UDT_NAME,
    SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME,
    MAX_CARDINALITY, DTD_IDENTIFIER ) AS
  SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
         OBJECT_TYPE, ROW_IDENTIFIER, FIELD_NAME,
         ORDINAL_POSITION, DATA_TYPE,
         CHARACTER_MAX_LENGTH, CHARACTER_OCTET_LENGTH, CHARACTER_SET_CATALOG,
         CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, COLLATION_CATALOG,
         COLLATION_SCHEMA, COLLATION_NAME, NUMERIC_PRECISION,
         NUMERIC_PRECISION_RADIX, NUMERIC_SCALE, DATETIME_PRECISION,
         INTERVAL_TYPE, INTERVAL_PRECISION, DOMAIN_DEFAULT,
         UDT_CATALOG, UDT_SCHEMA, UDT_NAME,
         SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME,
         MAXIMUM_CARDINALITY, DTD_IDENTIFIER
    FROM INFORMATION_SCHEMA.FIELDS;
```

4. *Rationale: Editorial - remove erroneous column and other text.*

Replace the Definition of the REFERENCED_TYPES_S view with:

```
CREATE VIEW REFERENCED_TYPES_S
  ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
    OBJECT_TYPE, REFERENCE_TYPE_ID, DATA_TYPE,
    CHAR_MAX_LENGTH, CHAR_OCTET_LENGTH, CHAR_SET_CATALOG,
    CHAR_SET_SCHEMA, CHARACTER_SET_NAME, COLLATION_CATALOG,
    COLLATION_SCHEMA, COLLATION_NAME, NUMERIC_PRECISION,
    NUMERIC_PREC_RADIX, NUMERIC_SCALE, DATETIME_PRECISION,
    INTERVAL_TYPE, INTERVAL_PRECISION,
    UDT_CATALOG, UDT_SCHEMA, UDT_NAME,
    SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME,
    MAX_CARDINALITY, DTD_IDENTIFIER ) AS
  SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
         OBJECT_TYPE, REFERENCE_TYPE_IDENTIFIER, DATA_TYPE,
         CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
         CHARACTER_SET_CATALOG,
         CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, COLLATION_CATALOG,
         COLLATION_SCHEMA, COLLATION_NAME, NUMERIC_PRECISION,
         NUMERIC_PRECISION_RADIX, NUMERIC_SCALE, DATETIME_PRECISION,
         INTERVAL_TYPE, INTERVAL_PRECISION,
         UDT_CATALOG, UDT_SCHEMA, UDT_NAME,
         SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME,
         MAXIMUM_CARDINALITY, DTD_IDENTIFIER
    FROM INFORMATION_SCHEMA.REFERENCED_TYPES;
```

5. *Rationale: Editorial.*

Replace the Definition of the ROL_TAB_METH_GRNTS view with:

```
CREATE VIEW ROL_TAB_METH_GRNTS
  ( GRANTOR, GRANTEE, TABLE_CATALOG,
    TABLE_SCHEMA, TABLE_NAME, SPECIFIC_CATALOG,
    SPECIFIC_SCHEMA, SPECIFIC_NAME, IS_GRANTABLE ) AS
  SELECT GRANTOR, GRANTEE, TABLE_CATALOG,
         TABLE_SCHEMA, TABLE_NAME, SPECIFIC_CATALOG,
         SPECIFIC_SCHEMA, SPECIFIC_NAME, IS_GRANTABLE
    FROM DEFINITION_SCHEMA.ROLE_TABLE_METHOD_GRANTS;
```

6. *Rationale: Replace use of reserved word as column name.*

Replace the Definition of the SQL_LANGUAGES_S view with:

```
CREATE VIEW SQL_LANGUAGES_S
  ( SOURCE, SQL_LANGUAGE_YEAR, CONFORMANCE,
    INTEGRITY, IMPLEMENTATION, BINDING_STYLE,
    PROGRAMMING_LANGUAGE ) AS
  SELECT SQL_LANGUAGE_SOURCE, SQL_LANGUAGE_YEAR, SQL_LANGUAGE_CONFORMANCE,
         SQL_LANGUAGE_INTEGRITY, SQL_LANGUAGE_IMPLEMENTATION,
         SQL_LANGUAGE_BINDING_STYLE, SQL_LANGUAGE_PROGRAMMING_LANGUAGE
    FROM INFORMATION_SCHEMA.SQL_LANGUAGES;
```

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

Replace the Definition of the TABLES_S view with:

```
CREATE VIEW TABLES_S
  ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
    TABLE_TYPE, SELF_REF_COLUMN, REF_GENERATION,
    UDT_CATALOG, UDT_SCHEMA, UDT_NAME, IS_INSERTABLE_INTO) AS
  SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
         TABLE_TYPE, SELF_REFERENCING_COLUMN, REFERENCE_GENERATION,
         USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
         USER_DEFINED_TYPE_NAME, IS_INSERTABLE_INTO
    FROM INFORMATION_SCHEMA.TABLES;
```

21.3 EQUAL_KEY_DEGREES assertion

1. *Rationale: Replace use of reserved word as correlation name.*

Replace the Definition with:

```
CREATE ASSERTION EQUAL_KEY_DEGREES
  CHECK     ( NOT EXISTS ( SELECT *
                          FROM ( SELECT COUNT ( DISTINCT FK.COLUMN_NAME ),
                                        COUNT ( DISTINCT PK.COLUMN_NAME )
                                   FROM KEY_COLUMN_USAGE AS FK,
                                        REFERENTIAL_CONSTRAINTS AS RF,
                                        KEY_COLUMN_USAGE AS PK
                                  WHERE ( FK.CONSTRAINT_CATALOG,
                                          FK.CONSTRAINT_SCHEMA,
                                          FK.CONSTRAINT_NAME )
                                        = ( RF.CONSTRAINT_CATALOG,
                                            RF.CONSTRAINT_SCHEMA,
                                            RF.CONSTRAINT_NAME )
                                        AND
                                        ( PK.CONSTRAINT_CATALOG,
                                          PK.CONSTRAINT_SCHEMA,
                                          PK.CONSTRAINT_NAME )
                                          = ( RF.UNIQUE_CONSTRAINT_CATALOG,
                                              RF.UNIQUE_CONSTRAINT_SCHEMA,
                                              RF.UNIQUE_CONSTRAINT_NAME )
                                    GROUP BY RF.CONSTRAINT_CATALOG,
                                             RF.CONSTRAINT_SCHEMA,
                                             RF.CONSTRAINT_NAME )
                                   AS R ( FK_DEGREE, PK_DEGREE )
                           WHERE FK_DEGREE <> PK_DEGREE ) )
```

21.6 ASSERTIONS base table

1. *Rationale: Remove irrelevant column from the definition of the ASSERTIONS base table.*

Delete the column definition for CHECK_TIME.

2. *Rationale: Editorial.*

Replace the check constraint ASSERTIONS_DEFERRED_CHECK with:

```
CONSTRAINT ASSERTIONS_DEFERRED_CHECK
  CHECK ( ( IS_DEFERRABLE, INITIALLY_DEFERRED )
          IN ( VALUES ( 'NO', 'NO' ),
                      ( 'YES', 'NO' ),
                      ( 'YES', 'YES' ) ) )
```

21.7 ATTRIBUTES base table

1. *Rationale: Editorial.*

Replace the check constraint ATTRIBUTES_IS_DERIVED_REFERENCE_ATTRIBUTE_CHECK with:

```
CONSTRAINT ATTRIBUTES_IS_DERIVED_REFERENCE_ATTRIBUTE_CHECK
  CHECK ( IS_DERIVED_REFERENCE_ATTRIBUTE IN ( 'YES', 'NO' ) ),
```

Replace the check constraint ATTRIBUTES_CHECK_DATA_TYPE with:

```
CONSTRAINT ATTRIBUTES_CHECK_DATA_TYPE
  CHECK ( ( UDT_CATALOG, UDT_SCHEMA, UDT_NAME,
            'USER-DEFINED TYPE', DTD_IDENTIFIER )
           IN ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                       OBJECT_TYPE, DTD_IDENTIFIER
                FROM DATA_TYPE_DESCRIPTOR ) ),
```

21.8 CHARACTER_SETS base table

1. *Rationale: Correct too hasty amendment of a change proposal.*
   Source: BHX-144

Replace Description 2) with:

2) The value of FORM_OF_USE is the null value.

2. *Rationale: Correct misapplication of a change proposal.*
   Source: BHX-144

Replace Description 3) with:

3) The value of NUMBER_OF_CHARACTERS is the null value.

3. *Rationale: Add missing Description.*
   Source: GBR-STC-028

Insert the following Description:

7) There is a row in this table for the character set INFORMATION_SCHEMA.SQL_CHARACTER. In that row:

   a) CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, and CHARACTER_SET_NAME are the name of the catalog, 'INFORMATION_SCHEMA', and 'SQL_IDENTIFIER', respectively.

   b) DEFAULT_COLLATE_CATALOG, DEFAULT_COLLATE_SCHEMA, and DEFAULT_COLLATE_NAME are the name of the catalog, 'INFORMATION_SCHEMA', and 'SQL_CHARACTER', respectively.

21.12 COLLATIONS base table

1. *Rationale: Editorial.*

Replace the primary key constraint COLLATIONS_PAD_PRIMARY_KEY with:

```
CONSTRAINT COLLATIONS_PRIMARY_KEY
     PRIMARY KEY ( COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ),
```

Replace the foreign key constraint COLLATIONS_PAD_FOREIGN_KEY_SCHEMATA with:

```
CONSTRAINT COLLATIONS_FOREIGN_KEY_SCHEMATA
     FOREIGN KEY ( COLLATION_CATALOG, COLLATION_SCHEMA )
        REFERENCES SCHEMATA,
```

2. *Rationale: Correct too hasty amendment of a change proposal.*
   Source: BHX-144

Replace Description 3) with:

3) The values of COLLATION_TYPE, COLLATION_DICTIONARY, and COLLATION_DEFINITION are the null value.

21.14 COLUMNS base table

1. *Rationale: Editorial - typographical error.*

Replace the check constraint COLUMN_CHECK_DATA TYPE with:

```
CONSTRAINT COLUMNS_CHECK_DATA_TYPE
  CHECK ( DOMAIN_CATALOG NOT IN( SELECT CATALOG_NAME
                                   FROM SCHEMATA )
          OR
( ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME ) IS NOT NULL
  AND
  ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
    'TABLE', DTD_IDENTIFIER )
    NOT IN ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                    OBJECT_TYPE, DTD_IDENTIFIER
               FROM DATA_TYPE_DESCRIPTOR ) )
OR
( ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME ) IS NULL
  AND
  ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
    'COLUMN', COLUMN_NAME )
    IN ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                OBJECT_TYPE, DTD_IDENTIFIER
           FROM DATA_TYPE_DESCRIPTOR ) ) )
```

21.15 DATA_TYPE_DESCRIPTOR base table

1. *Rationale: Provide missing text in Function section.*

Replace the Function with:

The DATA_TYPE_DESCRIPTOR table has one row for each domain, one row for each column (in each table) and for each attribute (in each structured type) that is defined as having a data type rather than a domain, one row for each distinct type, one row for the result type of each SQL-invoked function, one row for each SQL parameter of each SQL-invoked routine, one row for the result type of each method specification, one row for each parameter of each method specification, and one row for each structured type whose associated reference type has a user-defined representation. It effectively contains a representation of the data type descriptors.

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

Replace the check constraint DATA_TYPE_DESCRIPTOR_DATA_TYPE_CHECK_COMBINATIONS with:

```
CONSTRAINT DATA_TYPE_DESCRIPTOR_DATA_TYPE_CHECK_COMBINATIONS
  CHECK ( ( DATA_TYPE IN ( 'CHARACTER', 'CHARACTER VARYING',
                           'CHARACTER LARGE OBJECT' )
              AND
              ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
                COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NOT NULL
              AND
              ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
                DATETIME_PRECISION, USER_DEFINED_TYPE_CATALOG,
                USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME ) IS NULL
              AND
              ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
              AND
              ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
              AND
              MAXIMUM_CARDINALITY IS NULL )
          OR
          ( DATA_TYPE IN ( 'BIT', 'BIT VARYING' )
              AND
              ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH ) IS NOT NULL
              AND
              ( COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
              AND
              ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
                DATETIME_PRECISION, USER_DEFINED_TYPE_CATALOG,
                USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME ) IS NULL
              AND
              ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
              AND
  ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
  AND
  MAXIMUM_CARDINALITY IS NULL )
OR
( DATA_TYPE IN ( 'BINARY LARGE OBJECT' )
  AND
  ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH ) IS NOT NULL
  AND
  ( COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME,
    NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
    DATETIME_PRECISION, USER_DEFINED_TYPE_CATALOG,
    USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME ) IS NULL
  AND
  ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
  AND
  ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
  AND
  MAXIMUM_CARDINALITY IS NULL )
OR
( DATA_TYPE IN ( 'INTEGER', 'SMALLINT' )
  AND
  ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
    COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
  AND
  NUMERIC_PRECISION_RADIX IN (2,10 )
  AND
  NUMERIC_PRECISION IS NOT NULL
  AND
  NUMERIC_SCALE = 0
  AND
  DATETIME_PRECISION IS NULL
  AND
  ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
  AND
  ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
  AND
  MAXIMUM_CARDINALITY IS NULL )
OR
( DATA_TYPE IN ( 'NUMERIC', 'DECIMAL' )
  AND
  ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
    COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
  AND
  NUMERIC_PRECISION_RADIX = 10
  AND
  ( NUMERIC_PRECISION, NUMERIC_SCALE ) IS NOT NULL
  AND
  DATETIME_PRECISION IS NULL
  AND
  ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
  AND
  ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
  AND
  MAXIMUM_CARDINALITY IS NULL )
OR
( DATA_TYPE IN ( 'REAL', 'DOUBLE PRECISION', 'FLOAT' )
  AND
  ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
    COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
  AND
  NUMERIC_PRECISION IS NOT NULL
  AND
  NUMERIC_PRECISION_RADIX = 2
  AND
  NUMERIC_SCALE IS NULL
  AND
  DATETIME_PRECISION IS NULL
  AND
```

```
    ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
      USER_DEFINED_TYPE_NAME ) IS NULL
    AND
    ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
    AND
    ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
    AND
    MAXIMUM_CARDINALITY IS NULL )
  OR
  ( DATA_TYPE IN ( 'DATE', 'TIME', 'TIMESTAMP', 'TIME WITH TIME ZONE',
                   'TIMESTAMP WITH TIME ZONE' )
    AND
    ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
      COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
    AND
    ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX ) IS NOT NULL
    AND
    NUMERIC_SCALE IS NULL
    AND
    DATETIME_PRECISION IS NOT NULL
    AND
    ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
      USER_DEFINED_TYPE_NAME ) IS NULL
    AND
    ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
    AND
    ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
    AND
    MAXIMUM_CARDINALITY IS NULL )
  OR
  ( DATA_TYPE = 'INTERVAL'
    AND
    ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
      COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
    AND
    ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX ) IS NOT NULL
    AND
    NUMERIC_SCALE IS NULL
    AND
    DATETIME_PRECISION IS NOT NULL
    AND
    ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
      USER_DEFINED_TYPE_NAME ) IS NULL
    AND
    INTERVAL_TYPE IN ( 'YEAR', 'MONTH', 'DAY', 'HOUR', 'MINUTE',
                       'SECOND', 'YEAR TO MONTH', 'DAY TO HOUR',
                       'DAY TO MINUTE', 'DAY TO SECOND',
                       'HOUR TO MINUTE', 'HOUR TO SECOND',
                       'MINUTE TO SECOND' )
    AND
    INTERVAL_PRECISION IS NOT NULL
    AND
    ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
    AND
    MAXIMUM_CARDINALITY IS NULL )
  OR
  ( DATA_TYPE = 'BOOLEAN'
    AND
    ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
      COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IS NULL
    AND
    ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX ) IS NULL
    AND
    NUMERIC_SCALE IS NULL
    AND
    DATETIME_PRECISION IS NULL
    AND
    ( INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
    AND
    ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
      USER_DEFINED_TYPE_NAME ) IS NULL
    AND
    ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
    AND
    MAXIMUM_CARDINALITY IS NULL )
  OR
  ( DATA_TYPE = 'USER-DEFINED'
    AND
    ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
      DATETIME_PRECISION, CHARACTER_OCTET_LENGTH,
      CHARACTER_MAXIMUM_LENGTH, INTERVAL_TYPE, INTERVAL_PRECISION,
      SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
    AND
    ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
      USER_DEFINED_TYPE_NAME ) IS NOT NULL
    AND
    MAXIMUM_CARDINALITY IS NULL )
  OR
  ( DATA_TYPE = 'REF'
    AND
    ( CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH ) IS NOT NULL
      AND
      ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
        DATETIME_PRECISION, INTERVAL_TYPE, INTERVAL_PRECISION ) IS NULL
      AND
      ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
        USER_DEFINED_TYPE_NAME ) IS NOT NULL
      AND
      MAXIMUM_CARDINALITY IS NULL )
    OR
    ( DATA_TYPE = 'ARRAY'
      AND
      ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
        DATETIME_PRECISION, CHARACTER_OCTET_LENGTH,
        CHARACTER_MAXIMUM_LENGTH, INTERVAL_TYPE,
        INTERVAL_PRECISION )         IS NULL
      AND
      ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
        USER_DEFINED_TYPE_NAME ) IS NULL
      AND
      ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
      AND
      MAXIMUM_CARDINALITY IS NOT NULL )
    OR
    ( DATA_TYPE = 'ROW'
      AND
      ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
        DATETIME_PRECISION, CHARACTER_OCTET_LENGTH,
        CHARACTER_MAXIMUM_LENGTH, INTERVAL_TYPE,
        INTERVAL_PRECISION )         IS NULL
      AND
      ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
        USER_DEFINED_TYPE_NAME ) IS NULL
      AND
      ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
      AND
      MAXIMUM_CARDINALITY IS NULL )
    OR
    ( DATA_TYPE NOT IN ( 'CHARACTER', 'CHARACTER VARYING',
                         'CHARACTER LARGE OBJECT',
                         'BINARY LARGE OBJECT', 'BIT', 'BIT VARYING',
                         'INTEGER', 'SMALLINT', 'NUMERIC', 'DECIMAL',
                         'REAL', 'DOUBLE PRECISION', 'FLOAT',
                         'DATE', 'TIME', 'TIMESTAMP',
                         'INTERVAL', 'BOOLEAN', 'USER-DEFINED',
                         'REF', 'ARRAY', 'ROW' ) ) ),
```

3. *Rationale: Align Description 6) with the Definition section.*

Replace Description 6) with:

6) Case:

a) If DATA_TYPE is 'USER-DEFINED', then the values of USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are the qualified name of the user-defined type being described.

b) If the DATA_TYPE is 'REF', then the values of USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are the qualified name of the referenced structured type of the reference type being described.

c) Otherwise, the values of USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are the null value.

21.16 DIRECT_SUPERTABLES

1. *Rationale: Correct DIRECT_SUPERTABLES_CHECK_NO_REFLEXITIVITY constraint.*

Replace the DIRECT_SUPERTABLE_CHECK_NO_REFLEXITIVTY constraint with the following:

```
CONSTRAINT DIRECT_SUPERTABLES_CHECK_NO_REFLEXITIVITY
  CHECK ( ( TABLE_CATALOG, TABLE_SCHEMA,
            SUPERTABLE_NAME, TABLE_NAME )
           NOT IN ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
                           SUPERTABLE_NAME
                      FROM DIRECT_SUPERTABLES ) ),
```

21.23 METHOD_SPECIFICATION_PARAMETERS

1. *Rationale: Delete reference to METHOD_NAME column.*

Delete the column definition for METHOD_NAME.

2. *Rationale: Correct METHOD_SPECIFICATION_PARAMETERS_PRIMARY_KEY constraint.*

Replace the METHOD_SPECIFICATION_PARAMETERS_PRIMARY_KEY constraint with:

```
CONSTRAINT METHOD_SPECIFICATION_PARAMETERS_PRIMARY_KEY
  PRIMARY KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
                               ORDINAL_POSITION ),
```

3. *Rationale: Correct METHOD_SPECIFICATION_PARAMETERS_FOREIGN_KEY constraint.*

Replace the METHOD_SPECIFICATION_PARAMETERS_PRIMARY_KEY constraint with:

```
CONSTRAINT METHOD_SPECIFICATION_PARAMETERS_FOREIGN_KEY
  FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME )
    REFERENCES METHOD_SPECIFICATIONS,
```

21.24 METHOD_SPECIFICATIONS base table

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

Add the following column definition after the column IS_OVERRIDING:

```
IS_CONSTRUCTOR                         INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT METHOD_SPECIFICATION_IS_CONSTRUCTOR_CHECK
    CHECK ( IS_CONSTRUCTOR IN ('YES', 'NO') )
  CONSTRAINT METHOD_SPECIFICATION_IS_CONSTRUCTOR_COMBINATION_CHECK
    CHECK ( IS_CONSTRUCTOR = 'YES'
            AND
            IS_OVERRIDING = 'NO' ),
```

2. *Rationale: The specification of SQL_DATA_ACCESS is incorrect.*

Replace the definition of SQL_DATA_ACCESS with:

```
SQL_DATA_ACCESS                        INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT METHOD_SPECIFICATIONS_SQL_DATA_ACCESS_NOT_NULL NOT NULL
  CONSTRAINT METHOD_SPECIFICATIONS_SQL_DATA_ACCESS_CHECK
    CHECK ( SQL_DATA_ACCESS IN ( 'NO SQL', 'CONTAINS SQL',
                                 'READS SQL DATA', 'MODIFIES SQL DATA' ) ),
```

3. *Rationale: Correct METHOD_SPECIFICATIONS_LANGUAGE_CHECK constraint.*

Replace the check constraint METHOD_SPECIFICATIONS_LANGUAGE_CHECK with:

```
CONSTRAINT METHOD_SPECIFICATIONS_LANGUAGE_CHECK
  CHECK ( ROUTINE_BODY IN ( 'SQL', 'ADA', 'C', 'COBOL', 'FORTRAN', 'MUMPS',
                            'PASCAL', 'PLI' ) ),
```

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

Add the following Description Rule:

5.1) The values of IS_CONSTRUCTOR have the following meanings:

YES	The SQL-invoked method is an SQL-invoked constructor method.
NO	The SQL-invoked method is not an SQL-invoked constructor method.

5. *Rationale: The specification of SQL_DATA_ACCESS is incorrect.*

Replace Description Rule 10) with:

10) The values of SQL_DATA_ACCESS have the following meanings:

NO SQL	The SQL-invoked routine does not possibly contain SQL.
CONTAINS SQL	The SQL-invoked routine possibly contains SQL.
READS SQL DATA	The SQL-invoked routine possibly reads SQL-data.
MODIFIES SQL DATA	The SQL-invoked routine possibly modifies SQL-data.

21.25 PARAMETERS base table

1. *Rationale: Editorial - typographical error.*

Replace the foreign key constraint PARAMETERS_FOREIGN_KEY_SCHEMATA with:

```
CONSTRAINT PARAMETERS_FOREIGN_KEY_SCHEMATA
  FOREIGN KEY ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA )
    REFERENCES SCHEMATA,
```

21.33 ROUTINES base table

1. *Rationale: The specification of SQL_DATA_ACCESS is incorrect.*

Replace the definition of SQL_DATA_ACCESS with:

```
SQL_DATA_ACCESS                        INFORMATION_SCHEMA.CHARACTER_ DATA
  CONSTRAINT ROUTINES_SQL_DATA_ACCESS_NOT_NULL NOT NULL
  CONSTRAINT ROUTINES_SQL_DATA_ACCESS_CHECK
    CHECK ( SQL_DATA_ACCESS IN ( 'NO SQL', 'CONTAINS SQL',
                                 'READS SQL DATA', 'MODIFIES SQL DATA' ) ),
```

Replace Description Rule 8) with:

8) The values of SQL_DATA_ACCESS have the following meanings:

NO SQL	The SQL-invoked routine does not possibly contain SQL.
CONTAINS SQL	The SQL-invoked routine possibly contains SQL.
READS SQL DATA	The SQL-invoked routine possibly reads SQL-data.
MODIFIES SQL DATA	The SQL-invoked routine possibly modifies SQL-data.

21.34 SCHEMATA base table

1. *Rationale: Editorial - typographical error.*

Replace the Definition with:

```
CREATE TABLE SCHEMATA (
  CATALOG_NAME                         INFORMATION_SCHEMA.SQL_IDENTIFIER,
  SCHEMA_NAME                          INFORMATION_SCHEMA.SQL_IDENTIFIER,
  SCHEMA_OWNER                         INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT SCHEMA_OWNER_NOT_NULL NOT NULL,
  DEFAULT_CHARACTER_SET_CATALOG        INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT DEFAULT_CHARACTER_SET_CATALOG_NOT_NULL NOT NULL,
  DEFAULT_CHARACTER_SET_SCHEMA         INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT DEFAULT_CHARACTER_SET_SCHEMA_NOT_NULL NOT NULL,
  DEFAULT_CHARACTER_SET_NAME           INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT DEFAULT_CHARACTER_SET_NAME_NOT_NULL NOT NULL,
  SQL_PATH                             INFORMATION_SCHEMA.CHARACTER_DATA,
  CONSTRAINT SCHEMATA_PRIMARY_KEY
    PRIMARY KEY ( CATALOG_NAME, SCHEMA_NAME ),
  CONSTRAINT SCHEMATA_FOREIGN_KEY
    FOREIGN KEY ( SCHEMA_OWNER )
      REFERENCES USERS
  )
```

21.35 SQL_FEATURES base table

1. *Rationale: Editorial - typographical errors.*

Replace the Definition with:

```
CREATE TABLE SQL_FEATURES (
  FEATURE_ID                           INFORMATION_SCHEMA.CHARACTER_DATA,
  FEATURE_NAME                         INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT SQL_FEATURES_FEATURE_NAME_NOT_NULL NOT NULL,
  /* Zero for SUB_FEATURE_ID indicates a feature or a package*/
  SUB_FEATURE_ID                       INFORMATION_SCHEMA.CHARACTER_DATA,
  /* Zero-length string for SUB_FEATURE_NAME indicates a feature or a package */
  SUB_FEATURE_NAME                     INFORMATION_SCHEMA.CHARACTER_DATA,
  FEATURE_SUBFEATURE_PACKAGE_CODE      INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT_SQL_FEATURES_FEATURE_SUBFEATURE_PACKAGE_CODE_CHECK
      CHECK ( FEATURE_SUBFEATURE_PACKAGE_CODE IN ( 'FEATURE', 'SUBFEATURE',
                                                   'PACKAGE' ) )
    CONSTRAINT SQL_FEATURES_SUB_FEATURE_NAME_NOT_NULL NOT NULL,
  IS_SUPPORTED                         INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT SQL_FEATURES_IS_SUPPORTED_NOT_NULL NOT NULL
    CONSTRAINT SQL_FEATURES_IS_SUPPORTED_CHECK
      CHECK ( IS_SUPPORTED IN ( 'YES', 'NO' ) ),
  IS_VERIFIED_BY                       INFORMATION_SCHEMA.CHARACTER_DATA,
  COMMENTS                             INFORMATION_SCHEMA.CHARACTER_DATA,
  CONSTRAINT SQL_FEATURES_PRIMARY_KEY
    PRIMARY KEY ( FEATURE_ID, SUB_FEATURE_ID ),
  CONSTRAINT SQL_FEATURES_CHECK_SUPPORTED_VERIFIED
    CHECK ( IS_SUPPORTED = 'YES'
            OR
            IS_VERIFIED_BY IS NULL )
  )
```

21.36 SQL_IMPLEMENTATION_INFO base table

1. *Rationale: Editorial - typographical error.*

Replace the Definition with:

```
CREATE TABLE SQL_IMPLEMENTATION_INFO (
IMPLEMENTATION_INFO_ID                 INFORMATION_SCHEMA.CHARACTER_DATA,
IMPLEMENTATION_INFO_NAME               INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT SQL_IMPLEMENTATION_INFO_NAME_NOT_NULL NOT NULL,
INTEGER_VALUE                          INFORMATION_SCHEMA.CARDINAL_NUMBER,
CHARACTER_VALUE                        INFORMATION_SCHEMA.CHARACTER_DATA,
COMMENTS                               INFORMATION_SCHEMA.CHARACTER_DATA,
CONSTRAINT SQL_IMPLEMENTATION_INFO_PRIMARY_KEY
  PRIMARY KEY ( IMPLEMENTATION_INFO_ID )
)
```

21.38 SQL_SIZING base table

1. *Rationale: Editorial - typographical error.*

Replace the Definition with:

```
CREATE TABLE SQL_SIZING (
  SIZING_ID                        INFORMATION_SCHEMA.CARDINAL_NUMBER,
  SIZING_NAME                      INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT SQL_SIZING_SIZING_NAME_NOT_NULL NOT NULL,
  /* If SUPPORTED_VALUE is the null value, that means that the item */
  /* being described is not applicable in the implementation. */
```

```
    /* If SUPPORTED_VALUE is 0 (zero), that means that the item */
    /* being described has no limit or the limit cannot be determined.*/
    SUPPORTED_VALUE                  INFORMATION_SCHEMA.CARDINAL_NUMBER,
    COMMENTS                         INFORMATION_SCHEMA.CHARACTER_DATA,
    CONSTRAINT SQL_SIZING_PRIMARY_KEY
      PRIMARY KEY (SIZING_ID )
    )
```

21.39 SQL_SIZING_PROFILES base table

1. *Rationale: Editorial - typographical errors.*

Replace the Function with:

The SQL_SIZING base table has one row for each sizing item defined in ISO/IEC 9075.

Replace the Definition with:

```
  CREATE TABLE SQL_SIZING_PROFILES (
  SIZING_ID                          INFORMATION_SCHEMA.CARDINAL_NUMBER,
  SIZING_NAME                        INFORMATION_SCHEMA.CHARACTER_DATA
    CONSTRAINT SQL_SIZING_SIZING_NAME_NOT_NULL NOT NULL,
  PROFILE_ID                         INFORMATION_SCHEMA.CHARACTER_DATA,
  /* If REQUIRED_VALUE is the null value, that means that the item */
  /* being described is not applicable in the profile. */
  /* If REQUIRED_VALUE is 0 (zero), that means that the profile */
  /* does not set a limit for this sizing item. */
  REQUIRED_VALUE                     INFORMATION_SCHEMA.CARDINAL_NUMBER,
  COMMENTS                           INFORMATION_SCHEMA.CHARACTER_DATA,
  CONSTRAINT SQL_SIZING_PROFILE_PRIMARY_KEY
    PRIMARY KEY ( SIZING_ID, PROFILE_ID )
  )
```

21.43 TABLES base table

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

In the Definition, add the following column definition:

```
    IS_INSERTABLE_INTO                   INFORMATION_SCHEMA.CHARACTER_DATA
      CONSTRAINT IS_INSERTABLE_INTO_NOT_NULL NOT NULL
      CONSTRAINT IS_INSERTABLE_INTO_CHECK
        CHECK ( IS_INSERTABLE_INTO IN ( 'YES', 'NO' ) ),
```

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

Add the following Description Rule:

6.1) The values of IS_INSERTABLE_INTO have the following meanings:

YES	The table is insertable-into.
NO	The table is not insertable-into

21.44 TRANSFORMS base table

1. *Rationale: Editorial - typographical error.*

Replace the column definition TRANSFORM_TYPE with:

```
TRANSFORM_TYPE                         INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT TRANSFORM_TYPE_NOT_NULL NOT NULL
  CONSTRAINT TRANSFORM_TYPE_CHECK
    CHECK ( TRANSFORM_TYPE IN ('TO SQL', 'FROM SQL') ),
```

21.45 TRANSLATIONS base table

1. *Rationale: Correct too hasty amendment of a change proposal.*

In the Definition, delete the column constraint:

```
CONSTRAINT TRANSLATION_DEFINITION_NOT_NULL NOT NULL
```

Replace Description 4) with:

4) The value of TRANSLATION_DEFINITION is the null value.

21.47 TRIGGER_COLUMN_USAGE base table

1. *Rationale: Enable recording of column dependencies for UPDATE triggers.*

Replace the Function with:

The TRIGGER_COLUMN_USAGE base table has one row for each column of a table identified by a <table name> contained in a <table reference> that is contained in the <search condition> of a <triggered action> or explicitly or implicitly referenced in a <triggered SQL statement> of a <trigger definition> of the trigger being described.

2. *Rationale: Enable recording of column dependencies for UPDATE triggers.*

In the Definition, delete the constraint TRIGGER_COLUMN_USAGE_EVENT_NOT_UPDATE_CHECK.

3. *Rationale: Enable recording of column dependencies for UPDATE triggers.*

Replace Description 3) with:

3) The values of TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, and COLUMN_NAME are the catalog name, unqualified schema name, qualified identifier, and column name, respectively, of a column of a table identified by a <table name> contained in the <search condition> of a <triggered action>, or explicitly or implicitly referenced in a <triggered SQL statement> of a <trigger definition> of the trigger being described.

21.49 TRIGGERS base table

1. *Rationale: Extend TRIGGERS view to allow for the representation of all possible elements of <old or new values alias list>, and delete stray reference to " CONDITION_TIMING,".*

Replace the definitions of the columns CONDITION_TIMING, CONDITION_REFERENCE_OLD_TABLE and CONDITION_REFERENCE_NEW_TABLE with:

```
ACTION_TIMING                          INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT TRIGGERS_ACTION_TIMING_CHECK
    CHECK ( ACTION_TIMING IN ( 'BEFORE', 'AFTER' ) ),
ACTION_REFERENCE_OLD_TABLE             INFORMATION_SCHEMA.SQL_IDENTIFIER,
ACTION_REFERENCE_NEW_TABLE             INFORMATION_SCHEMA.SQL_IDENTIFIER,
ACTION_REFERENCE_OLD_ROW               INFORMATION_SCHEMA.SQL_IDENTIFIER,
ACTION_REFERENCE_NEW_ROW               INFORMATION_SCHEMA.SQL_IDENTIFIER,
```

2. *Rationale: Remove references to non-existing column.*

Delete the constraint EVENT_MANIPULATION_UPDATE_CHECK.

3. *Rationale: Editorial.*

Replace the definition of EVENT_MANIPULATION with:

```
EVENT_MANIPULATION                     INFORMATION_SCHEMA.CHARACTER_DATA
CONSTRAINT TRIGGERS_EVENT_MANIPULATION_CHECK
  CHECK ( EVENT_MANIPULATION IN
          ( 'INSERT', 'DELETE', 'UPDATE' ) ),
```

4. *Rationale: Extend TRIGGERS view to allow for the representation of all possible elements of <old or new values alias list>, and delete stray reference to " CONDITION_TIMING,".*

Replace Descriptions 4), 5) and 6) with:

4) The values of ACTION_TIMING have the following meaning:

BEFORE	The <trigger action time> is BEFORE.
AFTER	The <trigger action time> is AFTER.

5) The value of ACTION_REFERENCE_OLD_TABLE is the <old values table alias> of the trigger being described.

6) The value of ACTION_REFERENCE_NEW_TABLE is the <new values table alias> of the trigger being described.

6.1) The value of ACTION_REFERENCE_OLD_ROW is the <old values correlation name> of the trigger being described.

6.2) The value of ACTION_REFERENCE_NEW_ROW is the <new values correlation name> of the trigger being described.

5. *Rationale: Editorial.*

Replace Description Rule 7) with:

7) The value of ACTION_ORDER is the ordinal position of the trigger in the list of triggers with the same EVENT_OBJECT_CATALOG, EVENT_OBJECT_SCHEMA, EVENT_OBJECT_TABLE, EVENT_MANIPULATION, ACTION_TIMING, and ACTION_ORIENTATION.

6. *Rationale: Description Rule 9) refers to a non-existent BNF non-terminal, <triggered SQL statement list>.*

Replace Description Rule 9) with:

9) ACTION_STATEMENT is a character representation of the <triggered SQL statement> in the <triggered action> of the trigger being described.

7. *Rationale: Editorial.*

Replace Description Rule 10) with:

10) The values of ACTION_ORIENTATION have the following meanings:

ROW	The <triggered action> specifies FOR EACH ROW.
STATEMENT	The <triggered action> specifies FOR EACH STATEMENT.

21.52 USER_DEFINED_TYPES base table

1. *Rationale: Editorial - typographical error.*

Replace the column definition ORDERING_FORM with:

```
ORDERING_FORM                         INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT USER_DEFINED_TYPES_ORDERING_FORM_NOT_NULL NOT NULL
  CONSTRAINT USER_DEFINED_TYPES_ORDERING_FORM_CHECK
    CHECK ( ORDERING_FORM IN ( 'NONE', 'FULL', 'EQUALS' ) ),
```

2. *Rationale: Editorial - typographical error.*

Replace the column definition REFERENCE_TYPE with:

```
REFERENCE_TYPE                        INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT USER_DEFINED_TYPES_REFERENCE_TYPE_CHECK
    CHECK ( REFERENCE_TYPE IN ( 'SYSTEM GENERATED', 'USER GENERATED',
                                'DERIVED' ) ),
```

3. *Rationale: Editorial*

Replace the from clause with:

```
FROM ( DEFINITION_SCHEMA.USER_DEFINED_TYPES AS U
       LEFT JOIN ( DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D1
                   LEFT JOIN DEFINITION_SCHEMA.COLLATIONS AS C1
                     ON ( ( C1.COLLATION_CATALOG, C1.COLLATION_SCHEMA,
                            C1.COLLATION_NAME )
                          = ( D1.COLLATION_CATALOG, D1.COLLATION_SCHEMA,
                              D1.COLLATION_NAME ) ) )
         ON ( ( U.USER_DEFINED_TYPE_CATALOG, U.USER_DEFINED_TYPE_SCHEMA,
                U.USER_DEFINED_TYPE_NAME, 'USER-DEFINED TYPE',
                U.SOURCE_DTD_IDENTIFIER )
              = ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                  OBJECT_TYPE, D1.DTD_IDENTIFIER )
              OR
              ( U.USER_DEFINED_TYPE_CATALOG, U.USER_DEFINED_TYPE_SCHEMA,
                U.USER_DEFINED_TYPE_NAME, 'USER-DEFINED TYPE',
                U.REF_DTD_IDENTIFIER )
              = ( OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
                  OBJECT_TYPE, D1.DTD_IDENTIFIER ) ) )
```

21.56 VIEWS base table

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

In the Definition, delete the column definition for IS_INSERTABLE_INTO.

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

Delete Description 6)

22.1 SQLSTATE

1. *Rationale: The SQLSTATE Class codes allocated for SQL/MM no longer needed since they are not used by ISO/IEC 13249.*

Delete the rows in Table 27, "SQLSTATE class and subclass values" for the Class values "H1", "H2", "H3", "H4", "H5", "H6", "H7", "H8", "H9", "HA", "HB", "HC", "HD", "HE" and "HF".

22.3 SQL Multimedia and Application Package SQLSTATE Subclasses

1. *Rationale: The SQLSTATE Class codes allocated for SQL/MM no longer needed since they are not used by ISO/IEC 13249.*

Delete Subclause 22.3, "SQL Multimedia and Application Package SQLSTATE Subclasses" including Table 29, "SQLSTATE class codes for SQL/MM)".

23.1 General conformance requirements

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

Add the following row to “Table 30 — Implied feature relationships”:

Feature ID	**Feature Description**	**Implied Feature ID**	**Implied Feature Description**
F451	Character set definition	F461	Named character sets

Annex A SQL conformance summary

1. *Rationale: Editorial.*

Replace Item 4) d) i) with:

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

2. *Rationale: Create a separate Feature for <overlaps predicate>.*

Delete Items 4) h) and 4) i).

Add the following Item:

4.1) Specifications for Feature F053, “OVERLAPS predicate”:

a) Subclause 8.1, “<predicate>”:

i) Without Feature F053, “OVERLAPS predicate”, conforming SQL language shall not contain any <overlaps predicate>.

b) Subclause 8.12, “<overlaps predicate>”:

i) Without Feature F053, “OVERLAPS predicate”, conforming SQL language shall not contain any <overlaps predicate>.

3. *Rationale: There is no syntax "OF NEW TYPE" in <table definition>.*

Delete Item 20)

4. *Rationale: Editorial.*

Replace Item 23) a) i) with:

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

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

Replace Items 27) and 28) with:

27) Specifications for Feature F451, “Character set definition”:

a) Subclause 11.1, “<schema definition>”:

i) Without Feature F451, “Character set definition”, conforming SQL language shall not contain any <character set definition>.

b) Subclause 11.30, “<character set definition>”:

i) Without Feature F451, “Character set definition”, conforming SQL language shall not specify any <character set definition>.

ii) Without Feature F451, “Character set definition”, and Feature F691, “Collation and translation”, <collation source> shall specify DEFAULT.

c) Subclause 11.31, “<drop character set statement>”:

i) Without Feature F451, “Character set definition”, conforming SQL language shall contain no <drop character set statement>.

d) Subclause 13.5, “<SQL procedure statement>”:

i) Without Feature F451, “Character set definition”, an <SQL schema definition statement> shall not be a <character set definition>.

ii) Without Feature F451, “Character set definition”, an <SQL schema definition statement> shall not be a <drop character set statement>.

28) Specifications for Feature F461, “Named character sets”:

a) Subclause 5.3, “<literal>”:

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

b) Subclause 5.4, “Names and identifiers”:

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

c) Subclause 6.1, “<data type>”:

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

d) Subclause 10.5, “<privileges>”:

i) Without Feature F461, “Named character sets”, in conforming SQL language, an <object name> shall not specify CHARACTER SET.

e) Subclause 10.6, “<character set specification>”:

i) Without Feature F461, “Named character sets”, conforming SQL language shall not contain a <character set specification>.

f) Subclause 11.1, “<schema definition>”:

i) Without Feature F461, “Named character sets”, a <schema character set specification> shall not be specified.

g) Subclause 13.2, “<module name clause>”:

i) Without Feature F461, “Named character sets”, <module character set specification> shall not be specified.

6. *Rationale: Supply missing Conformance Rules for <contextually typed row value constructor>*

Add Item 38) a) iii):

38) a) iii) Without Feature F641, “Row and table constructors”, a <contextually typed row value constructor> that is not simply contained in a <contextually typed table value constructor> shall not contain more than one <row value constructor element>.

Add Item 38) a) iv):

38) a) iv) Without Feature F641, “Row and table constructors”, a <contextually typed row value constructor> shall not be a <row subquery>.

7. *Rationale: Use the correct non-terminal symbols.*

Replace Item 38) b) i) with:

38) b) i) Without Feature F641, “Row and table constructors”, the <contextually typed row value expression list> of a <contextually typed table value constructor> shall contain exactly one <contextually typed row value constructor> *RVE*. *RVE* shall be of the form “(<contextually typed row value constructor element list>)”.

8. *Rationale: Supply missing Conformance Rules for <contextually typed table value constructor>*

Add Item 38) b) iii):

38) b) iii) Without Feature F641, “Row and table constructors”, the <contextually typed row value expression list> of a <contextually typed table value constructor> shall contain exactly one <contextually typed row value constructor> *RVE*. *RVE* shall be of the form “(<contextually typed row value constructor element list>)”.

Add Item 38) b) iv):

38) b) iv) Without Feature F641, “Row and table constructors”, conforming SQL language shall not contain any <contextually typed table value constructor>.

9. *Rationale: Editorial.*

Replace Item 56) d) i) with:

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

10. *Rationale: Named column joins implicitly perform <comparison predicate>s, and should be subject to the same Conformance Rules.*

Add the following Conformance Rules to Item 57):

57) b.1) Without Feature S024, "Enhanced structured types", if NATURAL or <named column join> is specified, and if *C* is a corresponding join column, then the declared type of *C* shall not be based on a structured type.

11. *Rationale: Correct the references to <grant statement>.*

Replace Item 57) t) with:

57) t) Subclause 12.2, “< grant privilege statement> “:

i) Without Feature S024, “ Enhanced structured types”, a <specific routine designator> contained in a <grant privilege statement> shall not identify a method.

12. *Rationale: Clarify that Feature S023 also comprises <method specification list>.*

Add the following Item:

56) g) i.1) Without Feature S023, “Basic structured types”, conforming SQL language shall not specify <method specification list>.

13. Rationale: permit tables of structured type without requiring Feature S043, "Enhanced reference types".

Replace Item 59) f) ii) with:

59) f) ii) Without Feature S043, "Enhanced reference types", a <self-referencing column specification> shall specify SYSTEM GENERATED.

14. Rationale: Use of ONLY requires Feature S111, "ONLY in query expressions".

Add the following Item:

66) a.1) Subclause 14.6, "<delete statement: positioned>":

i) Without Feature S111, "ONLY in query expressions", a <target table> shall not specify ONLY.

15. Rationale: Editorial.

Replace Item 68) b) i) with:

68) b) i) Without Feature S161, "Subtype treatment", conforming SQL Language shall contain no <subtype treatment>.

16. Rationale: Editorial.

Replace Item 71) a) i) with:

71) a) i) Without Feature S241, "Transform functions", conforming SQL language shall not specify <transform group specification>.

17. Rationale: Editorial.

Replace Item 72) a) i) with:

72) a) i) Without Feature S251, "User-defined orderings", conforming SQL shall contain no <user-defined ordering definition>.

18. Rationale: Editorial.

Replace Item 76) a) i) with:

76) a) i) Without Feature T041, "Basic LOB data type support", conforming SQL language shall not contain any <binary string literal>.

19. Rationale: Editorial.

Replace Item 77) a) ii) with:

77) a) ii) Without Feature T042, "Extended LOB data type support", conforming SQL language shall not contain any <blob value function>.

20. Rationale: Editorial.

Replace Item 77) d) ii) with:

77) d) ii) Without Feature F421, “National character”, and Feature T042, “Extended LOB data type support”, neither operand of <concatenation> shall be of declared type NATIONAL CHARACTER LARGE OBJECT.

21. Rationale: Editorial.

Replace Item 87) b) i) with:

87) b) i) Without Feature T211, “Basic trigger capability”, conforming SQL language shall not contain a <trigger definition>.

22. Rationale: Supply missing application of functional dependencies to <having clause>.

Add the following to Item:

95) a.0) Subclause 7.10, "<having clause>"

a) Without Feature T301, "Functional dependencies", each column reference directly contained in the <search condition> shall be one of the following:

i) an unambiguous reference to a grouping column of *T*, or

ii) an outer reference.

b) Without Feature T301, "Functional dependencies", each column reference contained in a <subquery> in the <search condition> that references a column of *T* shall be one of the following:

i) an unambiguous reference to a grouping column of *T*, or

ii) contained in a <set function specification>.

23. Rationale: Remove redundant item. <grant role statement> is defined in <grant statement>.

Delete Item 99) b) ii).

24. Rationale: Remove undefined SET ROW functionality.

Delete Item 102).

25. *Rationale: Adapting the Conformance Rules to the Feature name and broader scope of Feature T431, "Extended grouping capabilities".*

Replace Item 103) with:

103) Specifications for Feature T431, "Extended grouping capabilities":

a) Sublause 6.16, "<set function specification>":

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

b) Subclause 7.9, "<group by clause>":

i) Without Feature T431, "Extended grouping capabilities", conforming SQL language shall not specify ROLLUP, CUBE, GROUPING SETS, or <grand total>.

ii) Without Feature T431, "Extended grouping capabilities", an <ordinary grouping set> shall be a <grouping column reference>.

26. *Rationale: Editorial.*

Replace Item 106) a) i) with:

106) a) i) Without Feature T471, "Result sets return value", conforming SQL language shall not specify <dynamic result sets characteristic>.

Annex B Implementation-defined elements

1. *Rationale: Insert missing item.*

Add the following Item:

1.1) Subclause 4.2.4 "Named character sets":

a) The collation and form-of-use of the named character set SQL_CHARACTER are implementation-defined.

2. *Rationale: The implementation-defined collation used to compare identifiers has not been noted.*

Add the following Item:

12.1) Subclause 5.2, "<token> and <separator>": equivalence of two <regular identifier>s, or a <regular identifier> and a <delimited identifier>, is determined using an implementation-defined collation that is sensitive to case.

3. *Rationale: Remove undefined non-terminals .*

Delete Item 30)

4. *Rationale: Missing entry for SQL-data access indication.*

Add the following Item:

33.1) Subclause 11.49 “<SQL-invoked routine>”:

a) If READS SQL DATA is specified, then it is implementation-defined whether the <SQL routine body> shall not contain an <SQL procedure statement> *S* that satisfies at least one of the following:

i) *S* is an <SQL data change statement>.

ii) *S* contains a <routine invocation> whose subject routine is an SQL-invoked routine that possibly modifies SQL-data.

iii) *S* contains an <SQL procedure statement> that is an <SQL data change statement>.

b) If CONTAINS SQL is specified, then it is implementation-defined whether the <SQL routine body> shall not contain an <SQL procedure statement> *S* that satisfies at least one of the following:

i) *S* is an <SQL data statement> other than <free locator statement> and <hold locator statement>.

ii) *S* contains a <routine invocation> whose subject routine is an SQL-invoked routine that possibly modifies SQL-data or possibly reads SQL-data.

iii) *S* contains an <SQL procedure statement> that is an <SQL data statement> other than <free locator statement> and <hold locator statement>.

5. *Rationale: Delete spurious items.*

Delete Items 51) and 52).

Annex C Implementation-dependent elements

1. *Rationale: The notion of distinct may be implementation-dependent if a RELATIVE or MAP ordering function is incorrectly defined.*

Add the following item:

0.1) Subclause 3.1.5, “Definitions provided in Part 2”

a) Whether two nonnull values of user-defined type whose comparison form is RELATIVE or MAP that result in <u>*unknown*</u> when tested for equality according to the rules of Subclause 8.2, “<comparison predicate>” are distinct or not is implementation-dependent.

2. *Rationale: There is no syntax “OF NEW TYPE” in <table definition>. Fixing bugs connected with the use of <user-defined type> syntax.*

Delete Item 20)

Annex E Incompatibilities with ISO/IEC 9075:1992 and ISO/IEC 9075-4:1996

1. *Rationale: Editorial.*

Replace point 1) with:

1) In ISO/IEC 9075:1992, Subclause 12.3, “<procedure>”, a <parameter declaration list> had an alternative “<parameter declaration> . . . “ (that is, a parameter list not surrounded by parentheses and with the individual component parameter declarations not separated by commas). This option was listed in ISO/IEC 9075:1992, as a deprecated feature. In ISO/IEC 9075-2:1999, the equivalent Subclause 13.3, “<externally-invoked procedure>”, does not contain this option. SQL-client modules that used this deprecated feature may be converted to conforming SQL by inserting a comma between each pair of <parameter declaration>s and placing a left parenthesis before and a right parenthesis after the entire parameter list.

2. *Rationale: Editorial. Correct the list of incompatible new <reserved word>s.*

In point 14) delete the texts:

— ABS
— ACTION
— AGGREGATE
— ALIAS
— CARDINALITY
— COMPLETION
— DESTROY
— DICTIONARY
— EVERY
— FACTOR
— HOST
— IGNORE
— INITIALIZE
— ITERATE
— LESS
— LIMIT
— MOD
— MODIFY
— NO
— OFF
— OPERATION
— OPERATOR
— OVERLAY
— PARAMETERS
— PREORDER
— RELATIVE
— REPLACE
— SENSITIVE
— SEQUENCE
— SESSION
— SPACE
— STRUCTURE
— SUBLIST
— SYMBOL
— TERM
— TERMINATE

— THE
— TYPE
— VARIABLE

In Item 14) add the following list elements:

— CURRENT_ROLE
— DYNAMIC
— FUNCTION
— INOUT
— LATERAL
— LOCALTIME
— LOCALTIMESTAMP
— METHOD
— OUT
— RELEASE
— UNNEST

Annex F SQL feature and package taxonomy

1. *Rationale: Editorial - typographic error.*

Replace row 20 in "Table 31 — SQL/Foundation feature taxonomy and definition for Core SQL" with:

Feature	ID	Feature Name	Feature Description
20	E021-12	Character comparison	— Subclause 8.2, "<comparison predicate>": For the CHARACTER and CHARACTER VARYING data types, without support for <table subquery> and without support for Feature F131, "Grouped operations"

2. *Rationale: Feature F031-13 should reference <drop table statement> rather than <drop column definition>.*

Replace row 118 in "Table 31 — SQL/Foundation feature taxonomy and definition for Core SQL" with:

Feature	ID	Feature Name	Feature Description
118	F031-13	DROP TABLE statement: RESTRICT clause	— Subclause 11.20 <drop table statement>: With a <drop behavior> of RESTRICT

3. *Rationale: Create a separate Feature for <overlaps predicate>.*

Add the following row into "Table 32—SQL/Foundation feature taxonomy for features outside Core SQL"

Feature	ID	Feature Name
7.1	F053	OVERLAPS predicate

4. *Rationale: Remove undefined SET ROW functionality.*

In Table 32, delete row 158, Feature ID T411, Feature Name “UPDATE STATEMENT: SET ROW option

5. *Rationale: Renaming of Feature T431 is necessary to reflect its broader scope.*

In Table 32 replace row 159 with:

Feature	**ID**	**Feature Name**
159	T431	Extended grouping capabilities