INTERNATIONAL STANDARD ISO/IEC 9075:1992
Technical Corrigendum 3
ISO/IEC 9075-3:1995
Technical Corrigendum 1
ISO/IEC 9075-4:1996
Technical Corrigendum 1

TECHNICAL CORRIGENDUM

Published 1999-11-15

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

Information technology — Database languages — SQL

TECHNICAL CORRIGENDUM

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

RECTIFICATIF TECHNIQUE

Technical Corrigendum 3 to International Standard ISO/IEC 9075:1992, Technical Corrigendum 1 to International Standard ISO/IEC 9075-3:1995 and Technical Corrigendum 1 to International Standard ISO/IEC 9075-4:1996 were prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management services*.

Relation to previous technical corrigenda:

This Corrigendum contains the cumulative set of corrections to ISO/IEC 9075:1992, ISO/IEC 9075-3:1995 and ISO/IEC 9075-4:1996.
It completely subsumes and replaces all previous corrigenda for ISO/IEC 9075.

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 rule numbering:

This Corrigendum introduces some new Syntax, Access, General and Leveling Rules. The new Rules in this Corrigendum have been numbered as follows:
Rules inserted between, for example, Rules 7) and 8) (in ISO/IEC 9075:1992) 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.

ICS 35.060 **Ref. No. ISO/IEC 9075/Cor:1999(E)**

Contents

Page

ISO/IEC 9075-4:1996
Database Languages - SQL-Part 4:Persistent Stored Modules
(SQL/PSM)

ISO/IEC 9075:1992
Database Languages - SQL

Introduction

1. *Rationale: In the list of significant new features, the wording incorrectly implies that all the examples listed in item 10) are referential integrity facilities.*

On page xiv, in Significant new feature 10), replace "referential integrity" with "integrity".

2 Normative references

1. *Rationale: Editorial. The (non-extended) Pascal standard should be identified as ISO/IEC 7185 rather than ISO 7185. The designation was changed in 1990 when the standard was revised.*

Change "ISO 7185:1990" to "ISO/IEC 7185:1990"

2. *Rationale: Editorial.*

Add the following reference after the reference to "ISO 8601:1988":

— ISO 8649:1988, *Information Processing Systems — Open Systems Interconnection — Service Definition for the Association Control Service Element.*

3. *Rationale: The newly revised Ada language standard (ISO/IEC-8652:1995, Information technology — Programming languages — Ada) contains support for decimal-encoded numeric data and variable length character strings. The revised interface allows newly written applications in the revised Ada language access to these features of SQL; previously written Ada applications, conformant with the earlier Ada interface, are conformant with the revised interface.*

Replace the reference to ISO/IEC 8652:1987) with:

— ISO/IEC 8652:1995, *Information technology — Programming languages — Ada.*

4. *Rationale: Editorial.*

Add the following reference after the reference to ISO/IEC 8824:1990:

— ISO/IEC 9579-1:1993, *Information technology — Open Systems Interconnection — Remote Database Access, Part 1: Generic Model, Service, and Protocol.*

Add the following reference after the reference to ISO/IEC 9899:

— ISO/IEC 10026-21, *Information technology — Open Systems Interconnection — Distributed Transaction Processing — Part 2: Service Definition.*

3.3.4.3 Terms denoting rule requirements

1. *Rationale: The following unifies the SQLSTATE returned for the different ways of invoking an SQL statement.*

In the first and second paragraphs, replace "*syntax error or access rule violation* (if this situation occurs during dynamic execution of an SQL-statement, then the exception that is raised is *syntax error or access rule violation in dynamic SQL statement*; if the situation occurs during direct invocation of an SQL-statement, then the exception that is raised is *syntax or access rule violation in direct SQL statement*)" with "*syntax error or access rule violation*".

4.2 Character strings

1. *Rationale: Editorial.*

In the second paragraph, replace the last sentence with:

> Character sets defined by standards or by implementations reside in the Information Schema (named INFORMATION_SCHEMA) in each catalog, as do collations and translations defined by standards and collations, translations, and form-of-use conversions defined by implementations.

4.2.1 Character strings and collating sequences

1. *Rationale: The following changes make the definitions of character set and collation descriptors more precise.*

Replace the text on page 17 that occurs after the first paragraph with:

> A character set is described by a character set descriptor. A character set descriptor includes:
>
> — the name of the character set or character repertoire,
>
> — if the character set is a character repertoire, then the name of the form-of-use,
>
> — an indication of what characters are in the character set, and
>
> — whether or not the character set uses the DEFAULT collation for its character repertoire, and,
>
> — if the character set does not utilize the DEFAULT collation for its character repertoire, then the <translation name> contained in the character set's <translation collation>, if any, the <collation name> contained in the character set's <collate clause> or <limited collation definition>, if any, and, whether or not DESC was specified in the reference to the collation
>
> For every character set, there is at least one collation. A collation is described by a collation descriptor. A collation descriptor includes:
>
> — the name of the collation,
>
> — the name of the character repertoire on which the collation operates,
>
> — whether the collation has the NO PAD or the PAD SPACE attribute, and
>
> — whether or not this collation utilizes the DEFAULT collation for its character repertoire,

— if the collation does not utilize the DEFAULT collation for its character repertoire, then the <translation name> contained in the collation's <translation collation>, if any, the <collation name> contained in the collation's <collation source>, if any, and whether or not DESC was specified in the definition of the collation.

4.2.3 Rules determining collating sequence usage

1. *Rationale: Editorial. The second bullet of the first paragraph (When columns are involved (e.g.., comparing ...)) predicates the possibility of just one column being involved, but the sentence ends with words that imply at least two columns. In addition, it is not clear whether the intent is that default collating sequence of a column overrides the collating sequence (necessarily the default for the relevant repertoire) for a literal.*

Replace the second bullet with:

— When one or more columns are involved (e.g., comparing two columns, or comparing a column to a literal), then provided that all columns involved have the same default collating sequence and there is no explicit specification of a collating sequence, that default collating sequence is used.

4.3.1 Bit string comparison and assignment

1. *Rationale: The second paragraph can be read to mean that all source bits are assigned successively to the most significant bit position of the receiving (target) string. Note that there does not appear to be a corresponding rule for character string assignment in the relevant subclause of "Concepts". Relevant rules do exist in clause 9, but do not address the order in which bits (or characters) are assigned.*

Delete the second paragraph of the subclause.

4.4 Numbers

1. *Rationale: Clarification.*

Add the following sentence immediately before the heading of Subclause 4.4.1 Characteristics of Numbers:

A value described by a numeric data type descriptor is always signed.

4.4.1 Characteristics of numbers

1. *Rationale: In several paragraphs in this subclause, phrases similar to "values representable in the data type" are used when the meaning is "members of the data type". A data type is defined as a set of values, so the term is unnecessarily complicated. Since there are rules defining mappings from all numbers within the range of a data type onto that data type, the meaning of "representable" is anyway somewhat ambiguous. Simpify the convoluted wording used to specify that rounding is always towards zero. Use numeric instead of numerical for consistency.*

Replace the fourth, fifth, sixth, seventh and eighth paragraphs with:

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

An approximation obtained by rounding of a numeric value N for an <exact numeric type> T is a value V in T such that the absolute value of the difference between N and the numeric value of V is not greater than half

the absolute value of the difference between two successive numeric values in T. If there are more than one such values V, then it is implementation-defined which one is taken.

All numeric values between the smallest and the largest value, inclusive, in a given exact numeric type have an approximation obtained by rounding or truncation for that type; it is implementation-defined which other numeric values have such approximations.

An approximation obtained by truncation or rounding of a numeric value N for an <approximate numeric type> T is a value V in T such that there is no value in T that lies between the V and N.

If there are more than one such values V then it is implementation-defined which one is taken. It is implementation-defined which numeric values have approximations obtained by rounding or truncation for a given approximate numeric type.

4.5 Datetimes and intervals

1. *Rationale: Clarification.*

Add the following sentence before the paragraph starting "Every datetime ...":

A value described by an interval data type descriptor is always signed.

4.5.1 Datetimes

1. *Rationale: Editorial.*

Replace the first paragraph with:

Table 4, "Fields in datetime items", specifies the fields that can make up a datetime value; a datetime value is made up of a subset of those fields. Not all of the fields shown are required to be in the subset, but every field that appears in the table between the first included primary field and the last included primary field shall also be included. If either timezone field is in the subset, then both of them shall be included.

2. *Rationale: Clarify the treatment of time zones.*

Replace the sixth, seventh, eighth and ninth paragraphs with:

A datetime data type that specifies WITH TIME ZONE is a data type that is *datetime with time zone*.

The surface of the earth is divided into zones, called time zones, in which every correct clock tells the same time, known as *local time*. Local time is equal to UTC (Coordinated Universal Time) plus the *time zone displacement*, which is a value of INTERVAL HOUR TO MINUTE, between -'12:59' and +'13:00'. The time zone displacement is constant throughout a time zone, changing at the beginning and end of Daylight Saving Time, where applicable.

A datetime value, of data type TIME or TIMESTAMP, may represent a local time or UTC. A data item may be defined to contain either a datetime value only, or a datetime value together with a time zone displacement.

For the convenience of users, whenever a datetime value with time zone is to be implicitly derived from one without (for example, in a simple assignment operation), SQL assumes the value without time zone to be local, subtracts the default session time zone displacement from it to give UTC, and associates that time zone displacement with the result.

Conversely, whenever a datetime value without time zone is to be implicitly derived from one with, SQL assumes the value with time zone to be UTC, adds the time zone displacement to it to give local, and the result, without any time zone displacement, is local.

4.5.2 Intervals

1. *Rationale: Editorial*

Replace the fifth paragraph with:

Within an item of type interval, the first field is constrained only by the <interval leading field precision> of the associated <interval qualifier>. Table 7, "Valid values for fields in INTERVAL items", specifies the constraints on subsequent field values.

2. *Rationale: Editorial.*

In Table 7, replace "<interval leading field precision" with "<interval leading field precision>" (two occurrences).

3. *Rationale: Clarify the precision of interval fields.*

Replace the sixth paragraph (that following Table 7) with:

Values in interval fields other than SECOND are integers, and have precision 2 when not the first field. SECOND, however, can be defined to have an <interval fractional seconds precision> that indicates the number of decimal digits maintained following the decimal point in the seconds value. When not the first field, SECOND has a precision of 2 places before the decimal point.

4. *Rationale: The wording seemed to imply that extra fields effectively added to a day-time interval for the purposes of operations between two fields are all added at the same end. However, comparison of an HOUR interval with a DAY-MINUTE interval would require extension of the HOUR interval at both ends.*

In the ninth paragraph, change "either the most significant end or the least significant end of one or both day-time intervals" to "either the most significant end of one interval, or the least significant end of one interval, or both".

4.5.3 Operations involving datetimes and intervals

1. *Rationale: Editorial. Table 8, Valid operators involving datetimes and intervals, specifies not the results of arithmetic operations involving datetime and interval operands, but the result types of operations between operands of those types.*

Replace the first paragraph with:

Table 8, "Valid operators involving datetimes and intervals", specifies the data types of the results of arithmetic operations involving datetime and interval operands.

4.6 Type conversion and mixing of data types

1. *Rationale: Correct an inconsistency in style between the third sentence of the third paragraph and similar statements elsewhere in the standard.Also correct the grammar of the last sentence.*

Replace the third paragraph with:

> Values corresponding to the data types BIT and BIT VARYING are always mutually comparable and are mutually assignable. If a store assignment would result in the loss of bits due to truncation, then an exception condition is raised. When values of unequal length are compared, if the shorter is a prefix of the longer, then the shorter is less than the longer; otherwise, the longer is effectively truncated to the length of the shorter for the purposes of comparison. When values of equal length are to be compared, then a bit-by-bit comparison is made. A 0-bit is less than a 1-bit.

4.8 Columns

1. *Rationale: Editorial.*

Replace the third paragraph and the lead-in to the bullet list of the fourth paragraph with:

> Every column has a nullability characteristic that indicates whether any attempt to store a null value into that column will inevitably cause an exception condition to be raised, and whether any attempt to retrieve a value from that column can ever result in a null value. The possible values of the nullability characteristic are known not nullable and possibly nullable.
>
> A column *C* with <column name> *CN* of a base table *T* has a nullability characteristic that is known not nullable if any only if either:

2. *Rationale: Editorial.*

In the penultimate paragraph on page 28, replace "<row value constructor expression>" with "<row value constructor element>".

4.9 Tables

1. *Rationale: Editorial.*

Replace the first sentence of the third paragraph with:

> A table is either a base table or a derived table.

2. *Rationale: There is no named derived table other than a viewed table.*

After the paragraph that begins with "A derived table descriptor describes a derived table.", delete the first item ("— if the table is named, then the name of the table;").

3. *Rationale: There is no named derived table other than a viewed table.*

After the paragraph that begins with "A view descriptor describes a view.", insert "— the name of the view, and" before the existing item.

4.10.2 Table constraints

1. *Rationale: Editorial.*

In the **Note**, replace "<match option>" with "<match type>".

2. *Rationale: Editorial.*

In the paragraph that begins with "A referential constraint is satisfied", replace "<match option>" with "<match type>".

4.18.1 Status parameters

1. *Rationale: To insure that the value returned to the user in SQLSTATE is representative of the actual state of the transaction or SQL-statement.*

Add the following as the last paragraph:

> For the purpose of choosing status parameter values to be returned, *exceptions* for transaction rollback have precedence over *exceptions* for statement failure. Similarly, completion condition *no data* has precedence over *warning*, which has precedence over *successful completion*. All *exceptions* have precedence over all completion conditions. The values assigned to SQLSTATE shall obey these precedence rules.

4.21 Cursors

1. *Rationale: Define "dynamic cursor" and "extended dynamic cursor".*

Add the following after the first sentence of the first paragraph:

> A cursor specified by a <dynamic declare cursor> is a *declared dynamic cursor*. A cursor specified by an <allocate cursor statement> is an *extended dynamic cursor*. A *dynamic cursor* is either a declared dynamic cursor or an extended dynamic cursor.

2. *Rationale: Correct concepts section regarding when cursors are destroyed.*

Replace the second paragraph with:

> For every <declare cursor> or <dynamic declare cursor> in a <module>, a cursor is effectively created when an SQL-transaction (see Subclause 4.28, "SQL-transactions") referencing the <module> is initiated, and destroyed when that SQL-transaction is terminated. An extended dynamic cursor is effectively created when an <allocate cursor statement> is executed within an SQL-transaction and destroyed when that SQL-transaction is terminated. In addition, a dynamic cursor is destroyed when a <deallocate prepared statement> is executed that deallocates the prepared statement on which the cursor is based.

4.22.6 SQL-statements and transaction states

1) *Rationale: No statement can be both transaction-initiating and not transaction-initiating.*

In the first dashed list (of transaction-initiating SQL-statements), in the bulleted sublist of SQL-data statements, delete the entry for <dynamic select statement>.

4.24 SQL dynamic statements

1) *Rationale: Editorial.*

In the fourth paragraph, replace "<target specification>s" with "<simple value specification>s".

2) *Rationale: Editorial.*

In the eighth paragraph, replace the first occurrence of "<SQL statement>s" with "<SQL procedure statement>s", and replace the second occurrence of "<SQL statement>s" with "<embedded SQL statement>s".

4.26 Privileges

1) *Rationale: Editorial.*

In the fourth paragraph on page 52, replace "<module authorization identifier> is" with "<schema authorization identifier> is".

2) *Rationale: Provide missing rules that cover the acquisition of the necessary privileges to acquire the WITH GRANT OPTION on views through a grant to PUBLIC.*

Add the following before the antepenultimate paragraph of this Subclause:

> The phrase *user privileges* refers to the set of privileges defined by the privilege descriptors whose grantee is either the identified <authorization identifier> or PUBLIC.

4.28 SQL-transactions

1) *Rationale: Clarification.*

In the paragraph that begins "In some environments (e.g., remote database access)", replace all occurrences of "SQL-environment" with "SQL-implementation".

4.29 SQL-connections

1) *Rationale: Editorial.*

Replace the second paragraph with :

> An SQL-connection is an *active SQL-connection* if any SQL-statement that initiates or requires an SQL-transaction has been executed at its SQL-server via that SQL-connection during the current SQL-transaction.

2. *Rationale: Clarification.*

In the last sentence of the penultimate paragraph, replace "SQL-environment" with "SQL-implementation".

4.31 Client-server operation

1. *Rationale: Clarification.*

Replace the first sentence with :

> As perceived by an SQL-agent, an SQL-implementation consists of one or more SQL-servers and one SQL-client through which SQL-connections can be made to them.

5.2 <token> and <separator>

1. *Rationale: The maximum length of an <identifier> is intended to be 128 characters.*

Replace Syntax Rule 8) with :

8) In a <regular identifier>, the number of <underscore>s plus the number of <identifier part>s shall be less than 128.

2. *Rationale: A <regular identifier> shall not contain any <quote> or <double quote>. Thus, a <delimited identifier> with a <delimited identifier body> containing a <quote> or <double quote> is not equivalent to any <regular identifier>.*

In Syntax Rule 13), delete the expression "(with all occurrences of <quote> replaced by <quote symbol> and all occurrences of <doublequote symbol> replaced by <double quote>)".

3. *Rationale: Correct the incorrect references to "<quote>" and "<quote symbol>" and delete the redundant references to "<double quote>"s and "<double quote symbol>"s in Syntax Rule 14.*

In Syntax Rule 14), delete "(with all occurrences of <quote> replaced by <quote symbol> and all occurrences of <doublequote symbol> replaced by <doublequote>)".

4. *Rationale: A <character representation> does not appear in a <regular identifier> or in a <delimited identifier body>.*

Replace Leveling Rule 2) a) with :

a) The number of <underscore>s plus the number of <identifier part>s contained in a <regular identifier> shall be less than 18.

Insert the following Leveling Rule 2) a.1):

a.1) The <delimited identifier body> of a <delimited identifier> shall not comprise more than 18 <delimited identifier part>s.

5.3 <literal>

1. *Rationale: Support changes to Subclause 6.10.*

In Format, replace the BNF for <date string>, <time string>, <timestamp string> with :

<date string> ::= <quote> <unquoted date string> <quote>

<time string> ::= <quote> <unquoted time string> <quote>

<timestamp string> ::= <quote> <unquoted timestamp string> <quote>

2. *Rationale: CLI requires the <sign> of an <interval literal> to appear within <unquoted interval string>.*

In Format, replace the BNF for <interval literal> with :

<interval literal> ::=
INTERVAL <interval string> <interval qualifier>

3. *Rationale: Support changes to Subclause 6.10.*

In Format, replace the BNF for <interval string> with :

<interval string> ::= <quote> <unquoted interval string> <quote>

Add the following to Format:

<unquoted date string> ::= <date value>

<unquoted time string> ::= <time value> [<time zone interval>]

<unquoted timestamp string> ::= <unquoted date string> <space> <unquoted time string>

<unquoted interval string> ::= <sign> { <year-month literal> | <day-time literal> }

4. *Rationale: Ellipses were placed both in the definition of <separator> and in the usage of <separator> in several types of string literals. This leads to a small ambiguity when two separator characters follow one another. The following changes remove the ellipses in the latter of these two places:*

In Format, in each of the productions for <character string literal>, <national character string literal>, <bit string literal>, <hex string literal>, replace "<separator> ..." with "<separator>".

In Syntax Rules 1), 2) and 3) replace "<separator> ..." with "<separator>".

5. *Rationale: Stipulate the precision of fields in interval literals.*

Insert a Syntax Rule as follows:

20.1) Each datetime component shall have the precision specified by the <interval qualifier>.

6. Rationale: Refrain from applying a time zone displacement unless required.

Replace General Rule 8), including the Note, with:

8) If <time zone interval> is specified, then the time and timestamp values in <time literal> and <timestamp literal> represent a datetime in the specified time zone. If <date value> is specified, it is interpreted as a date in the Gregorian calendar. If <time value> is specified, it is interpreted as a time of day. Let *DV* be the value of the <datetime literal>, disregarding <time zone interval>.

Case:

a) If <time zone interval> is specified, then let *TZI* be the value of the interval denoted by it; the value of the <datetime literal> is *DV* - *TZI*, with time zone displacement *TZI*.

b) Otherwise, the value of the <datetime literal> is *DV*.

Note: If <time zone interval> is specified, a <time literal> or <timestamp literal> is interpreted as local time with the specified time zone displacement. However, it is effectively converted to UTC, while retaining the original time zone displacement. If <time zone interval> is not specified, no assumption is made about time zone displacement. However, should a time zone displacement be required during subsequent processing, the default SQL-session time zone displacement will be applied at that time.

5.4 Names and identifiers

1. Rationale: Editorial.

In Syntax Rule 1), replace "identified" with "indicated".

2. Rationale: Editorial. In most places where equivalence of lower case and upper case letters is discussed, the phrase "replaced by the equivalent upper case letter or letters" is used. However, in Syntax Rule 5, (An <SQL language identifier> ...), the words "or letters" are omitted.

In Syntax Rule 5), in the first sentence, replace "upper-case letter)," with "upper-case letter or letters),".

3. Rationale: Add the rule for the comparison of <character set name>s missing since <character set name> was decoupled from <qualified name>.

Add the following Syntax Rule:

10.5) Two <character set names> are equal if they have the same <SQL language identifier> and the same <schema name>, regardless of whether the <schema name>s are implicit or explicit.

6.1 <data type>

1. Rationale: Clarify the error condition that should be returned if a datetime overflow occurs.

Delete General Rule 6).

2. Rationale: Clarification.

Add the following Note after Syntax Rule 28):

Note: The length of interval data types is specified in the General Rules of 10.1, "<interval qualifier>".

6.2 <value specification> and <target specification>

1. *Rationale: Identify possible upward incompatibility.*

Add the following at the end of Syntax Rule 3):

Note: In an environment where the SQL-implementation conforms to Entry SQL, conforming SQL language that contains either:

a) a specified or implied <comparison predicate> that compares the <value specification> USER with a <value specification> other than USER, or

b) a specified or implied assignment in which the "value" (as defined in Subclause 9.2 Store assignment) contains the <value specification> USER

will become non-conforming in an environment where the SQL-implementation conforms to Intermediate SQL or Full SQL, unless the character repertoire of the implementation-defined character set in that environment is identical to the character repertoire of SQL_TEXT.

6.3 <table reference>

1. *Rationale: Editorial.*

In Syntax Rule 2) a), replace "with no intervening" with "without an intervening".

2. *Rationale: Editorial.*

In Syntax Rule 2) a), replace "The scope clause of the exposed" with "The scope of the exposed".

3. *Rationale: Editorial.*

In Syntax Rule 2) b), replace "with no intervening" with "without an intervening".

4. *Rationale: Clarify that references to non-existing objects is only allowed within the same <schema definition>.*

Add the following Syntax Rule:

8.1) Let *T* be the table identified by the <table name> immediately contained in <table reference>. 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 include a <schema element> that creates the descriptor of *T*.

5. *Rationale: Editorial.*

Delete the first sentence of Access Rule 1).

6. *Rationale: Editorial. Leveling Rule 2)a) is redundant, as Leveling Rule 2)a) of 7.5, <joined table>, specifies a general prohibition of the specification of <joined table> in Entry SQL.*

Delete Leveling Rule 2)a).

6.4 <column reference>

1. *Rationale: The first sentence of Syntax Rule 9 re-declares some of the notational symbols that were originally declared in Syntax Rule 1. The re-declaration of CR is identical to the original declaration, but the redeclaration of C is subtly different. According to Syntax Rule 1, C is the column identified by the <column name>. However, it takes two further rules to explain how the table associated with the <column name> is to be determined. According to Syntax Rule 9, C is the column identified by the column reference. Syntax Rule 4, meanwhile, has made two statements of quite different nature, one about the data type of the column reference, and the other a purely syntactic requirement that the <column name> should identify a single column. Syntax Rule 4 should be split into two rules, and the declaration of C is probably best made in one of those rules. The last declaration of Syntax Rule 1 and the first sentence of Syntax Rule 9 should both be deleted.*

Replace Syntax Rule 1) with:

1) Let *CR* be the <column reference>. Let *CN* be the <column name> contained in *CR*.

Replace Syntax Rule 4) with:

4) *CN* shall uniquely identify a column of *T*. Let *C* be that column.

4.1) The data type of *CR* is the data type of *C*.

2. *Rationale: Remove unnecessary Syntax Rules.*

Delete Syntax Rules 6) and 7).

3. *Rationale: The first sentence of Syntax Rule 9 re-declares some of the notational symbols that were originally declared in Syntax Rule 1. The re-declaration of CR is identical to the original declaration, but the redeclaration of C is subtly different. According to Syntax Rule 1, C is the column identified by the <column name>. However, it takes two further rules to explain how the table associated with the <column name> is to be determined. According to Syntax Rule 9, C is the column identified by the column reference. Syntax Rule 4, meanwhile, has made two statements of quite different nature, one about the data type of the column reference, and the other a purely syntactic requirement that the <column name> should identify a single column. Syntax Rule 4 should be split into two rules, and the declaration of C is probably best made in one of those rules. The last declaration of Syntax Rule 1 and the first sentence of Syntax Rule 9 should both be deleted.*

Replace Syntax Rule 9) with:

9) *C* is an underlying column of *CR*. If *CR* is a <derived column>, then every underlying column of *C* is an underlying column of *CR*.

4. *Rationale: Remove unnecessary General Rule.*

Delete General Rule 2).

6.5 <set function specification>

1. *Rationale: Editorial. The Note following Syntax Rule 2 refers the reader to Subclause 7.8, <having clause>, for a definition of "argument source". The Note is both misleading and unnecessary. It is misleading because the definition in 7.8 has a restricted scope, and for a full definition reference to 7.9 is also necessary. The Note is unnecessary because the immediately preceding normative text refers the reader to both 7.8 and 7.9 for the necessary specification.*

Delete the Note following Syntax Rule 2).

6.6 <numeric value function>

1. *Rationale: Correct Format for <position expression> to support <bit value expression>s in addition to <character value expression>s.*

In the Format replace the production for <position expression> with :

<position expression> ::=
POSITION <left paren> <string value expression>
IN <string value expression> <right paren>

Replace Syntax Rule 1) with :

1) If <position expression> is specified, then both <string value expression>s shall be <bit value expression>s or both shall be <character value expression>s having the same character repertoire.

Replace General Rules 1) and 2) with :

1) If <position expression> is specified and neither <string value expression> is the null value, then

   Case:

   a) If the first <string value expression> has a length 0, then the result is 1.

   b) If the value of the first <string value expression> is equal to an identical-length substring of contiguous characters or bits from the value of the second <string value expression>, then the result is 1 greater than the number of characters or bits within the value of the second <string value expression> preceding the start of the first such substring.

   c) Otherwise, the result is 0.

2) If <position expression> is specified and either <string value expression> is the null value, then the result is the null value.

6.7 <string value function>

1. *Rationale: define the data type, coercibility and collating sequence of <string value function>, <character value function> and <bit value function>.*

Insert the following Syntax Rules:

0.1) The data type of <string value function> is the data type of the simply contained <character value function> or <bit value function>. If <string value function> is <character value function>, the coercibility and collating sequence of <string value function> is the coercibility and collating sequence of the simply contained <character value function>.

0.2) The data type, coercibity and collating sequence of <character value function> are the data type, coercibility and collating sequence of the simply contained <character substring function>, <fold>, <form-of-use conversion>, <character translation>, or <trim function>.

0.3) The data type of <bit value function> is the data type of the simply contained <bit substring function>.

2. *Rationale: a keyword does not have an operand.*

In Syntax Rule 2) b), replace "SUBSTRING" with "<character substring function>".

3. *Rationale: Define the result of <string value function>, <character value function> and <bit value function>*

Insert the following General Rules:

0.1) The result of <string value function> is the result of the simply contained <character value function> or <bit value function>.

0.2) The result of <character value function> is the result of the simply contained <character substring function>, <fold>, <form-of-use conversion>, <character translation>, or <trim function>.

0.3) The result of <bit value function> is the result of the simply contained <bit substring function>.

6.8 <datetime value function>

1. *Rationale: The following change clarifies that <datetime value function> is effectively evaluated only once per SQL statement.*

Replace General Rule 3) with :

3) If an SQL-statement causes the evaluation of one or more <datetime value function>s, then all such evaluations are effectively performed simultaneously. The time of evaluation of the <datetime value function> during the execution of the SQL-statement is implementation-dependent.

6.10 <cast specification>

1. *Rationale: Clarify that references to non-existing objects is only allowed within the same <schema definition>.*

Add the following Syntax Rule:

8.1) If <domain name> is specified, then let *D* be the domain identified by the <domain name>. If the <cast specification> is not contained in a <schema definition>, then the schema identified by the explicit or implicit qualifier of the <domain name> shall include the descriptor of *D*. If the <cast specification> is contained in a <schema definition> *S*, then the schema identified by the explicit or implicit qualifier of the <domain name> shall include the descriptor of *D*, or *S* shall include a <schema element> that creates the descriptor of *D*.

2. *Rationale: Stipulate the position of the minus sign in the result of a cast of a negative interval to character string.*

Replace the first paragraph of General Rule 5) e) with :

e) If *SD* is a datetime data type or an interval data type, then let *Y* be the shortest character string that conforms to the definition of <literal> in Subclause 5.3, "<literal>", and such that the interpreted value of *Y* is *SV* and the interpreted precision of *Y* is the precision of *SD*. If *SV* is a negative interval, then <sign> shall be specified within <unquoted interval string> in the literal *Y*.

Replace the first paragraph of General Rule 6) e) with :

e) If *SD* is a datetime data type or an interval data type, then let *Y* be the shortest character string that conforms to the definition of <literal> in Subclause 5.3, "<literal>", and such that the interpreted value of *Y* is *SV* and the interpreted precision of *Y* is the precision of *SD*. If *SV* is a negative interval, then <sign> shall be specified within <unquoted interval string> in the literal *Y*.

3. *Rationale: Legalize a simpler format for character strings being cast to temporal data types.*

In General Rule 9) a) i), replace "rules for <literal>" with "rules for <literal> or for <unquoted date string>".

4. *Rationale: Clarify the error condition that should be returned if a datetime overflow occurs.*

Replace General Rule 9) a) ii) with :

ii) Otherwise,

1) If a <datetime value> does not conform to the natural rules for dates or times according to the Gregorian calendar, an exception condition is raised: *data exception — invalid datetime format.*

2) Otherwise, an exception condition is raised: *data exception — invalid character value for cast*.

5. *Rationale: Legalize a simpler format for character strings being cast to temporal data types, clarify the error condition that should be returned if a datetime overflow occurs and refrain from applying a time zone displacement unless required.*

Replace General Rules 10) and 11) with:

10) Let *STZD* be default time zone displacement of the SQL-session.

11) If *TD* is the datetime data type TIME, then

a) Case:

i) If *SD* is character string, then *SV* is replaced by

TRIM (BOTH ' ' FROM *SV*)

Case:

1) If the rules for <literal> or for <unquoted time string> in Subclause 5.3, "<literal>", can be applied to *SV* to determine a valid value of the data type *TD*, then *TV* is set to that value.

2) If a <datetime value> does not conform to the natural rules for dates or times according to the Gregorian calendar, then an exception condition is raised: *data exception invalid datetime format*.

3) Otherwise, an exception condition is raised: *data exception - invalid character value for cast*.

ii) If *SD* is a TIME, then *TV* is *SV*.

iii) If *SD* is a TIMESTAMP, then *TV* is the hour, minute, and second <datetime field>s of *SV*.

b) Case:

i) If the data type of *SV* is datetime with time zone, then let *STZ* be the time zone displacement of *SV*.

Case:

1) If *TV* is WITH TIME ZONE, the time zone displacement of *TV* is *STZ*.

2) Otherwise, *TV* is replaced by *TV* + *STZ*.

Note: The result is local.

ii) Otherwise, if *TV* is WITH TIME ZONE, *TV* is replaced by *TV* - *STZD*, with time zone displacement *STZD*.

Note: The result is UTC.

11.1) If *TD* is the datetime data type TIMESTAMP, then

a) Case:

i) If *SD* is character string, then *SV* is replaced by

TRIM (BOTH ' ' FROM *SV*)

Case:

1) If the rules for <literal> or for <unquoted time string> in Subclause 5.3, "<literal>", can be applied to *SV* to determine a valid value of the data type *TD*, then *TV* is set to that value.

2) If a <datetime value> does not conform to the natural rules for dates or times according to the Gregorian calendar, then an exception condition is raised: *data exception - invalid datetime format.*

3) Otherwise, an exception condition is raised: *data exception - invalid character value for cast.*

ii) If *SD* is DATE, then the <datetime field>s hour, minute, and second of *TV* are set to 0 and the <datetime field>s year, month, and day of *TV* are set to their respective values in *SV*.

iii) If *SD* is TIME, then the <datetime field>s year, month, and day of *TV* are set to their respective values in an execution of CURRENT_DATE and the <datetime field>s hour, minute, and second of *TV* are set to their respective values in *SV*.

iv) If *SD* is TIMESTAMP, then *TV* is *SV*.

b) Case:

i) If the data type of *SV* is datetime with time zone, then let *STZ* be the time zone displacement of *SV*.

Case:

1) If *TV* is WITH TIME ZONE, the time zone displacement of *TV* is *STZ*.

2) Otherwise, *TV* is replaced by *TV* + *STZ*.

Note: The result is local.

ii) Otherwise, if *TV* is WITH TIME ZONE, *TV* is replaced by *TV* - *STZD*, with time zone displacement *STZD*.

Note: The result is UTC.

6. Rationale: Legalize a simpler format for character strings being cast to temporal data types.

In General Rule 12) b) i), replace "rules for <literal>" with "rules for <literal> or for <unquoted interval string>".

7. *Rationale: Clarify the error condition that should be returned if a datetime overflow occurs.*

Replace General Rule 12) b) ii) with :

ii) Otherwise,

1) If a <datetime value> does not conform to the natural rules for intervals according to the Gregorian calendar, an exception condition is raised: *data exception — invalid interval format.*

2) Otherwise, an exception condition is raised: *data exception — invalid character value for cast*.

6.11 <value expression>

1. *Rationale: the data type of <value expression primary> should be defined.*

Insert the following Syntax Rule:

1.1) The data type of a <value expression primary> is the data type of the immediately contained <unsigned value specification>, <column reference>, <set function specification>, <scalar subquery>, <value expression>, or <cast spec-ification>.

2. *Rationale: the value of a <scalar subquery> is defined in Subclause 7.11 "<scalar subquery>, <row subquery> and <table subquery>".*

Delete General Rule 2).

6.12 <numeric value expression>

1. *Rationale: Editorial.*

In General Rule 2), replace "then the value of" with "then the result of".

6.14 <datetime value expression>

1. *Rationale: The expression "X AT TIME ZONE Y - Z" may be parsed as either "(X AT TIME ZONE Y) - Z" or "X AT TIME ZONE (Y - Z)". The following clarifies that it should be the former.*

Replace the Format for <time zone specifier> with:

```
<time zone specifier> ::=
    LOCAL
  | TIME ZONE <interval primary>
```

2. *Rationale: Clarify the data type of the result, and refrain from applying a time zone displacement unless required.*

Replace Syntax Rules 2), 3), 4) and 5) with:

2) If the <datetime value expression> immediately contains neither <plus sign> nor <minus sign>, then the precision of the result of the <datetime value expression> is the precision of the <value expression primary> or <datetime value function> that it simply contains.

3) If the data type of the <datetime primary> is DATE, then <time zone> shall not be specified.

4) Case:

a) If <time zone> is specified and the data type of <datetime primary> is TIMESTAMP or TIME, then the data type of <datetime term> is respectively TIMESTAMP WITH TIME ZONE or TIME WITH TIME ZONE, with the same fractional seconds precision as <datetime primary>.

b) Otherwise the data type of <datetime term> is the same as the data type of <datetime primary>.

5) If the <datetime value expression> immediately contains either <plus sign> or <minus sign>, then:

a) The <interval value expression> or <interval term> shall contain only <datetime field>s that are contained within the <datetime value expression> or <datetime term>,

b) The result of the <datetime value expression> contains the same <datetime field>s as are contained within the <datetime value expression> or <datetime term>, with a fractional seconds precision that is the greater of the fractional seconds precisions, if any, of either the <datetime value expression> and <interval term>, or the <datetime term> and <interval value expression> that it simply contains.

5.1) The data type of the <interval primary> immediately contained in a <time zone specifier> shall be INTERVAL HOUR TO MINUTE.

3. *Rationale: If <time zone> is implied, then <time zone specifier> shall not be specified.*

Replace General Rule 2) with:

2) If <time zone> is specified and the <interval primary> immediately contained in <time zone specifier> is the null value, then the result of the <datetime value expression> is the null value.

4. *Rationale: General Rule 3) refers to the operators + and - directly, whereas the convention adopted in other clauses of the standard is to refer to them indirectly through the syntactic elements <plus sign> and <minus sign>.*

In the lead-in to General Rule 3 and in the first sentence of General Rule 3) a) i) change "+" to "<plus sign>" and "-" to "<minus sign>".

5. *Rationale: Refrain from applying a time zone displacement unless required.*

Replace General Rule 4) with:

4) Let *DT* be the data type, *DV* the value, and *TZD* the time zone displacement, if any, of the <datetime primary> simply contained in the <datetime value expression>; let *STZD* be the current default time zone displacement of the SQL-session.

Case:

a) If the data type of the result is not datetime with time zone, then

Case:

i) If *DT* is not datetime with time zone, then the value of the <datetime term> is *DV*.

ii) Otherwise, the value of the <datetime term> is *DV* + *TZD*.

b) If the data type of the result is datetime with time zone, then:

i) Case:

A) If *DT* is datetime with time zone, then let *TZV* be *TZD*.

B) Otherwise, *DT* is replaced by *DT* - *STZD*; let *TZV* be *STZD*.

ii) Case:

A) If LOCAL is specified, then let *TZ* be *STZD*.

B) If TIME ZONE is specified then, if the value of the <interval primary> immediately contained in <time zone specifier> is less than INTERVAL -'12:59' or greater than INTERVAL +'13:00', then an exception condition is raised: *data exception-invalid time zone displacement value*. Otherwise, let *TZ* be the value of the <interval value expression> simply contained in <time zone>.

C) Otherwise, let *TZ* be *TZV*.

iii) The value of the <datetime term> is *DV*, with time zone displacement *TZ*.

6.15 <interval value expression>

1. Rationale: Clarification of the data type of a result.

Replace Syntax Rule 3) with:

3) Case:

a) If the <interval value expression> simply contains an <interval qualifier> *IQ*, then the data type of the result is
INTERVAL *IQ*

b) If the <interval value expression> is an <interval term>, then the result of the <interval value expression> contains the same interval fields as the <interval primary>

c) If <interval term 1> is specified, then the result contains every interval field that is contained in the result of either <interval value expression 1> or <interval term 1>, and, if both contain a seconds field, then the fractional seconds precision of the result is the greater of the two fractional seconds precisions.

Note: Interval fields are effectively defined by Table 5 - Fields in year-month INTERVAL items, and Table 6 Fields in day-time INTERVAL items.

7.1 <row value constructor>

1. Rationale: To clarify the value of a <default specification> by reference to the General Rules of 11.5 <default clause> as is also done in 13.9.

Replace General Rule 2) with:

2) The value of a <default specification> is determined according to the General Rules of Subclause 11.5, "<default clause>".

2. Rationale: Editorial.

In Leveling Rule 1) a), replace "<table value constructor>" with "<table value constructor> or an <overlaps predicate>".

3. Rationale: The 1989 standard prohibited the use of a <row subquery> in a predicate. A leveling rule was omitted that would only allow this use of <row subquery> in Full level.

Add the following to Leveling Rule 1):

a.1) A <row value constructor> shall not be a <row subquery>.

7.4 <from clause>

1. *Rationale: The use of the phrase "with no intervening <derived table> or <joined table>" in the preamble of both sub-rules of Syntax Rule 1) and both sub-rules of General Rule 1) is initially confusing, since the Format contains neither <derived table> nor <joined table>. It takes a little thought to realise that the reference is to syntactic elements contained in the elements explicitly present in the Format. The term immediately contains is introduced in subclause 3.3.4.2, Syntactic containment, to describe just this situation.*

Replace Syntax Rule 1) with:

1) Case:

   a) If the <from clause> immediately contains only one <table reference>, then the descriptor of the result of the <from clause> is the same as the descriptor of the table identified by that <table reference>.

   b) If the <from clause> immediately contains more than one <table reference>, then the descriptors of the columns of the result of the <from clause> are the descriptors of the columns of the tables identified by those <table reference>s, in the order in which the <table reference>s appear in the <from clause> and in the order in which the columns are defined within each table.

Replace General Rule 1) with:

1) Case:

   a) If the <from clause> immediately contains only one <table reference>, then the result of the <from clause> is the table identified by that <table reference>.

   b) If the <from clause> immediately contains more than one <table reference>, then the result of the <from clause> is the extended Cartesian product of the tables identified by those <table reference>s.

      The extended Cartesian product, *CP*, is the multiset of all rows *R* such that *R* is the concatenation of a row from each of the identified tables in the order in which they are identified. The cardinality of *CP* is the product of the cardinalities of the identified tables. The ordinal position of a column in *CP* is *N*+*S*, where *N* is the ordinal position of that column in the identified table *T* from which it is derived and *S* is the sum of the degrees of the tables identified before *T* in the <from clause>.

2. *Rationale: Editorial.*

In Leveling Rule 2) a), replace "<table name>" with "<table reference>".

7.5 <joined table>

1. *Rationale: Syntax Rule 6) [If NATURAL ...] does not adequately address the following:*
   - *i) the case when the set of corresponding columns is empty,*
   - *ii) the cases when either set of non-corresponding columns is empty,*
   - *iii) the case when a set of non-corresponding join columns contains columns with the same <column name>.*

   Syntax Rule 6) also conflicts with Syntax Rules 7), 8) and 9) with respect to the definition of the nullability characteristics of a column.

In Syntax Rule 6) d), replace "Let" by "If there is at least one corresponding join column, then let"

Replace Syntax Rule 6) e) with:

e) 1) If T_1 contains at least one column that is not a corresponding join column, then let SLT_1 be a <select list> of <derived column>s of the form:

TA.C

for every column *C* of T_1 that is not a corresponding join column, taken in order of their ordinal position in T_1.

2) If T_2 contains at least one column that is not a corresponding join column, then let SLT_2 be a <select list> of <derived column>s of the form:

TB.C

for every column *C* of T_2 that is not a corresponding join column, taken in order of their ordinal position in T_2.

Replace Syntax Rule 6) f) with:

f) Let the <select list> *SL* be defined as:

Case:

i) If all the columns of T_1 and T_2 are corresponding join columns, then let *SL* be *SLCC*.

ii) If T_1 contains no corresponding join columns and T_2 contains no corresponding join columns, then let *SL* be SLT_1, SLT_2.

iii) If T_1 contains no columns other than corresponding join columns, then let *SL* be *SLCC*, SLT_2.

iv) If T_2 contains no columns other than corresponding join columns, then let *SL* be *SLCC*, SLT_1

v) Otherwise, let *SL* be *SLCC*, SLT_1, SLT_2

The descriptors of the columns of the result of the <joined table>, with the possible exception of the nullability characteristics of the column, are the same as the descriptors of the columns of the result of

SELECT *SL* FROM TR_1, TR_2

2. *Rationale: General Rule 6) a) [If NATURAL is ...] uses the <select list> definitions SLCC, SLT_1 and SLT_2 which are not applicable to the table SN.*

Replace General Rule 6) a) with:

a) If NATURAL is specified or a <named columns join> is specified, then:

i) Let CS_i be a name for the *i*-th column of *S*. Column CS_i of *S corresponds* to the *i*-th column of T_1 if *i* is less than or equal to D_1. Column CS_j of *S corresponds* to the $(j\text{-}D_1)$-th column of T_2 for *j* greater than D_1.

ii) Let *SLN* be the <select list> derived from *SL* by replacing each <derived column> of the form
TA.C or *TB.C*
in *SL* by the name CS_i of the column of *S* that corresponds to the column *C* of T_1 or T_2, respectively.

iii) The result of the <joined table> is the multiset of rows resulting from
SELECT *SLN* FROM *SN*

7.6 <where clause>

1. *Rationale: Syntax Rule 2), by use of the definite article, implies that a <set function specification> contains exactly one <column reference>.*

In Syntax Rule 2) replace "and the <column reference>" with "and every <column reference> contained".

2. *Rationale: Remove the ambiguity in Leveling Rule 2).*

Replace Leveling Rule 2) with:

2) The following restrictions apply for Entry SQL in addition to any Intermediate SQL restriction:

a) A <value expression> directly contained in the <search condition> shall not contain a <column reference> that references a <derived column> that generally contains a <set function specification>.

7.7 <group by clause>

1. *Rationale: General Rule 1) states that within a group no two values of a grouping column are distinct. General Rule 2 states that such values are equal. Is there any significance in the use of different terms? Since General Rule 3c of 8.2, <comparison predicate>, page 170, recognises that two character strings may compare as equal even if they are of different lengths or contain different characters, it is possible to interpret General Rule 1) as saying that identity, not merely equality comparison, is required for assignment of rows to the same group. If the intention is that the rules of <comparison predicate> should be used in the determination of groups, then that should be stated in 7.7.*

Replace General Rules 1) and 2) with:

1) The result of the <group by clause> is a partitioning of the rows of *T* into the minimum number of groups such that, for each grouping column of each group, no two values of that grouping column are distinct.

2) When a <search condition> or <value expression> is applied to a group, a reference to a grouping column is a reference to the common value in that column of the rows in that group.

Note: Where application of the General Rules of 8.2, <comparison predicate>, has resulted in the formation of a group with values of different lengths or containing different sequences of characters in the same grouping column, the value selected as the common value of that grouping column in that group is implementation-dependent.

7.8 <having clause>

1. *Rationale: The notational symbol T is declared twice, first in Syntax Rule 1), then again (as not quite the same thing) in General Rule 1). A simple solution to the elimination of the redundancy would be to delete the second definition. The second sentence of General Rule 1) would then require amendment, since it contains a reference to the current first sentence. That second sentence provides a definition (of the meaning of grouping column when there is no effective <group by clause>) that is required for the full understanding of Syntax Rule 1), so its content should in any event be moved to Syntax Rule 1).*

Replace Syntax Rule 1) with:

1) Let *HC* be the <having clause>. Let *TE* be the <table expression> that immediately contains *HC*.

Case:

a) If *TE* immediately contains neither a <where clause> nor a <group by clause>, then let *T* be the descriptor of the table defined by the <from clause> *FC* immediately contained in *TE* and let *R* be the result of *FC*.

b) If *TE* immediately contains a <where clause> *WC* but not a <group by clause>, then let *T* be the descriptor of the table defined by *WC* and let *R* be the result of *WC*.

c) Otherwise let *T* be the descriptor of the table defined by the <group by clause> *GBC* immediately contained in *TE* and let *R* be the result of *GBC*.

If *TE* does not immediately contain a <group by clause>, the table described by *T* has no grouping columns.

1.1) Each <column reference> directly contained in the <search condition> shall unambiguously reference a grouping column of *T* or be an outer reference.

Note: *Outer reference* is defined in Subclause 6.4, "<column reference>".

2. *Rationale: Datetime data types with time zone are as much a cause of non-determinism as character strings.*

Replace Syntax Rule 3) with:

3) The <having clause> is *possibly non-deterministic* if it contains a reference to a column *C* of *T* that has a data type of character string or datetime with time zone and:

a) *C* is specified within a <set function specification> that specifies MIN or MAX, or

b) *C* is a grouping column of *T*.

3. *Rationale: The notational symbol T is declared twice, first in Syntax Rule 1), then again (as not quite the same thing) in General Rule 1). A simple solution to the elimination of the redundancy would be to delete the second definition. The second sentence of General Rule 1) would then require amendment, since it contains a reference to the current first sentence. That second sentence provides a definition (of the*

meaning of grouping column when there is no effective <group by clause>) that is required for the full understanding of Syntax Rule 1), so its content should in any event be moved to Syntax Rule 1).

Replace all four General Rules with:

1) If *TE* does not immediately contain a <group by clause>, then *R* consists of a single group.

2) The <search condition> is applied to each group of *R*. The result of the <having clause> is a grouped table containing those groups of *R* for which the result of the <search condition> is true.

3) When the <search condition> is applied to a given group of *R*, that group is the argument or argument source of each <set function specification> directly contained in the <search condition> unless the <column reference> in the <set function specification> is an outer reference.

Note: *Outer reference* is defined in Subclause 6.4, "<column reference>".

4) Each <subquery> in the <search condition> is effectively evaluated for each group of *R* and the result used in the application of the <search condition> to the given group of *R*. If any evaluated <subquery> contains an outer reference to a column of *T*, then the reference is to the values of that column in the given group of *R*.

Note: *Outer reference* is defined in Subclause 6.4, "<column reference>".

7.9 <query specification>

1. *Rationale: The rules for expansion of "<qualifier>.*" need to limit the expansion to the table identified by the <qualifier>, rather than including the entire <table expression>.*

Replace Syntax Rule 4) with:

4) If the <select sublist> "<qualifier>.*" is specified, then let *Q* be the <qualifier> of that <select sublist>. *Q* shall be a <table name> or <correlation name> exposed by a <table reference> immediately contained in the <from clause> of *T*. Let *TQ* be the table associated with *Q*. That <select sublist> is equivalent to a <value expression> sequence in which each <value expression> is a <column reference> *CR* that references a column of *TQ* that is not a common column of a <joined table>. Each column of *TQ* that is not a common column of a <joined table> shall be referenced exactly once. The columns shall be referenced in the ascending order of their ordinal positions with *TQ*.

Note: *common column of a <joined table>* is defined in Subclause 7.5, "<joined table>".

2. *Rationale: Clarification of the definition of outer reference.*

Replace Syntax Rule 7) with:

7) If *T* is a grouped table, then in each <value expression>, each <column reference> that references a column of *T* shall reference a grouping column or be specified within a <set function specification>. If *T* is not a grouped table and any <value expression> contains a <set function specification> that contains a reference to a column of *T* or any <value expression> directly contains a <set function specification> that does not contain an outer reference, then in each <value expression>, each <column reference> that references a column of *T* shall be specified within a <set function specification>.

3. *Rationale: Editorial.*

In Syntax Rule 9) a), replace the two occurrences of "*C*" with "*CN*".

In Syntax Rule 9) b), replace "*C*" with "the <column name> of the column designated by the <column reference>".

4. *Rationale: A <dynamic parameter specification> is possibly nullable and the syntax of the CAST required parentheses.*

Replace Syntax Rule 10) with:

10) A column of the table that is the result of a <query specification> is possibly nullable if and only if it contains a <column reference> for a column C that is possibly nullable, an <indicator parameter>, an <indicator variable>, a <dynamic parameter specification>, a <subquery>, CAST (NULL AS X) (where X represents a <data type> or a <domain name>), SYSTEM_USER, or a <set function specification> that does not contain COUNT.

5. *Rationale: Datetime data types with time zone are as much a cause of non-determinism as character strings.*

Replace Syntax Rule 13) with:

13) A <query specification> is *possibly non-deterministic* if any of the following conditions are true:

a) The <set quantifier> DISTINCT is specified and one of the columns of *T* has a data type of either character string or datetime with time zone; or

b) The <query specification> directly contains a <having clause> that is possibly non-deterministic; or

c) The <select list> contains a reference to a column *C* of *T* that has a data type of either character string or datetime with time zone and either

i) *C* is specified with a <set function specification> that specifies MIN or MAX, or

ii) *C* is a grouping column of *T*.

6. *Rationale: Editorial.*

In General Rule 1) a) ii) 1), replace "the table is the result of the <query specification>" with "the result of the <query specification> is that table".

In General Rule 1) b) ii) 1), replace "the table is the result of the <query specification>" with "the result of the <query specification> is that table".

7.10 <query expression>

1. *Rationale: Editorial.*

Replace Syntax Rule 14) with:

14) The *simply underlying tables* of a <query expression> are the tables identified by those <table name>s, <query specification>s, and <derived table>s contained in the <query expression> without an intervening <derived table> or an intervening <join condition>.

2. *Rationale: Datetime data types with time zone are as much a cause of non-determinism as character strings.*

Replace Syntax Rule 15) with:

15) A <query expression> is *possibly non-deterministic* if

a) it contains a set operator UNION and ALL is not specified, or if it contains EXCEPT or INTERSECT; and

b) the first or second operand contains a column that has a data type of character string or datetime with time zone.

3. *Rationale: Correct the definitions of non-determinisim.*

Add a Syntax Rule 15.1):

15.1) A <query expression> is *possibly non-deterministic* if the first or second operand is possibly non-deterministic.

7.11 <scalar subquery>, <row subquery>, and <table subquery>

1. *Rationale: define the value of a <scalar subquery>.*

Insert the following General Rule:

1.1) Let *SS* be a <scalar subquery>.
Case:
a) If the cardinality of *SS* is 0 then the value of the <scalar subquery> is the null value.
b) Otherwise, let *C* be the column of the <query expression> simply contained in *SS*. The value of *SS* is the value of *C* in the unique row of the result of the <scalar subquery>.

2. *Rationale: Editorial.*

Replace Leveling Rule 1) with:

1) The following restrictions apply for Intermediate SQL:

None.

3. *Rationale: SQL-89 prohibited the use of a subquery with degree greater than one as the argument of an <exists predicate>. It was intended to relax this restriction in Intermediate SQL, but the required Leveling Rule for Entry SQL was inadvertently omitted.*

Add the following Leveling Rule:

b.1) If a <table subquery> is simply contained in an <exists predicate>, then the <select list> of the <query specification> directly contained in the <table subquery> shall comprise either an <asterisk> or a single <derived column>.

8.2 <comparision predicate>

1. Rationale: Editorial.

Replace General Rule 3) c):

c) Depending on the collating sequence, two strings may compare as equal even if they are of different lengths or contain different sequences of characters. When any of the operations MAX, MIN, and DISTINCT reference a grouping column, and the UNION, EXCEPT, and INTERSECT operators refer to character strings, the specific value selected by these operations from a set of such equal values is implementation-dependent.

8.3 <between predicate>

1. Rationale: Editorial.

Replace Syntax Rule 2) and Syntax Rule 3) with the single Syntax Rule:

2) In the three <row value constructor>s, the data types of values with the same ordinal position shall be comparable.

8.7 <quantified comparison predicate>

1. Rationale: Editorial.

In Syntax Rule 2), replace "columns" with "corresponding columns".

8.11 <overlaps predicate>

1. Rationale: In Syntax Rule 2, the final phrase does not state explicitly what the data type of the first column of the <row value constructor> should be comparable with.

Replace Syntax Rule 2) with:

2) The data types of the first column of <row value constructor 1> and the first column of <row value constructor 2> shall both be datetime data types, and these data types shall be comparable.

9.1 Retrieval assignment

1. *Rationale: General Rule 3) k) appears to prescribe that every numeric value should be rounded or truncated before assignment to a target, even if its exact value is a member of the target data type. In addition, where an approximation is selected, there is no requirement that it should be any particular, or even a close, approximation.*

Replace General Rule 3) k) with:

k) If the data type of *T* is numeric then:

Case:

i) If *V* is a member of the data type of *T*, then *T* is set to *V*.

ii) If a member of the data type of *T* can be obtained from *V* by rounding or truncation, then *T* is set to that value. If the data type of *T* is exact numeric, then it is implementation-defined whether the approximation is obtained by rounding or by truncation.

iii) Otherwise an exception is raised: *data exception — numeric value out of range*.

2. *Rationale: Clarify the treatment of datetimes and refrain from applying a time zone displacement unless required.*

Replace General Rule 3) l) with:

l) If the data type, *DT*, of *T* is datetime then

i) If only one of *DT* and the data type of *V* is datetime with time zone, then *V* is effectively replaced by CAST (*V* TO *DT*).

ii) Case:

1) If V is a member of the data type of T, then T is set to V.

2) If a member of the data type of T can be obtained from V by rounding, then T is set to that value.

3) Otherwise, an exception condition is raised: *data exception - datetime field overflow*.

3. *Rationale: General Rule 3) m) appears to state that neither rounding nor truncation is permitted in assignment of an interval value. Was this intended? A user might be quite happy to lose some fractional seconds, for example.*

Replace General Rule 3) m) with:

m) If the data type of *T* is interval then:

Case:

i) If *V* is a member of the data type of *T*, then *T* is set to *V*.

ii) If a member of the data type of *T* can be obtained from *V* by rounding or truncation, then *T* is set to that value. It is implementation-defined whether the approximation is obtained by rounding or by truncation.

iii) Otherwise an exception is raised: *data exception — interval field overflow*.

9.2 Store assignment

1. *Rationale: General Rule 1) is redundant in SQL 9075:1992 and is incorrect for SQL/PSM DIS 9075-4:1996*

Delete General Rule 1).

2. *Rationale: Editorial. Both Syntax Rule 1) and General Rule 3) contain declarations of V, and General Rule 2) refers to the value of V, when V is already declared to be a value.*

In General Rule 2), replace the phrase "If the value of *V* is" with "If *V* is".

In General Rule 3), replace "Otherwise, let *V* denote a non-null value of *T*." with "If *V* is not the null value, then:

3. *Rationale: General Rule 3) k) appears to prescribe that every numeric value should be rounded or truncated before assignment to a target, even if it exact value is a member of the target data type. In addition, where an approximation is selected, there is no requirement that it should be any particular, or even a close, approximation.*

Replace General Rule 3) k) with:

k) If the data type of *T* is numeric then:

Case:

i) If *V* is a member of the data type of *T*, then *T* is set to *V*.

ii) If a member of the data type of *T* can be obtained from *V* by rounding or truncation, then *T* is set to that value. If the data type of *T* is exact numeric, then it is implementation-defined whether the approximation is obtained by rounding or by truncation.

iii) Otherwise an exception is raised: *data exception — numeric value out of range*.

4. *Rationale: Clarify the treatment of datetimes and refrain from applying a time zone displacement unless required.*

Replace General Rule 3) l) with:

l) If the data type, *DT*, of *T* is datetime then

i) If only one of *DT* and the data type of *V* is datetime with time zone, then *V* is effectively replaced by CAST (*V* TO *DT*).

ii) Case:

1) If V is a member of the data type of T, then T is set to V.

2) If a member of the data type of T can be obtained from V by rounding, then T is set to that value.

3) Otherwise, an exception condition is raised: *data exception - datetime field overflow*.

5. *Rationale: General Rule 3) m) appears to state that neither rounding nor truncation is permitted in assignment of an interval value. Was this intended? A user might be quite happy to lose some fractional seconds, for example.*

Replace General Rule 3) m) with:

m) If the data type of *T* is interval then:

Case:

i) If *V* is a member of the data type of *T*, then *T* is set to *V*.

ii) If a member of the data type of *T* can be obtained from *V* by rounding or truncation, then *T* is set to that value. It is implementation-defined whether the approximation is obtained by rounding or by truncation.

iii) Otherwise an exception is raised: *data exception — interval field overflow*.

9.3 Set operation result data types

1. *Rationale: Clarify the time zone and fractional seconds precision of a datetime result.*

Replace Syntax Rule 3) e), with:

e) If some data type in *DTS* is a datetime data type, then every data type in *DTS* shall be a datetime data type having the same datetime fields. The result data type is a datetime data type having the same datetime fields, whose fractional seconds precision is the largest of the fractional seconds precisions in *DTS*. If some data type in *DTS* is WITH TIME ZONE, then the result is WITH TIME ZONE, otherwise the result is without time zone.

10.1 <interval qualifier>

1. *Rationale: Editorial.*

Replace "Syntax rules" with "Syntax Rules".

2. *Rationale: Clarification of the definition of "significance" of fractional seconds precision.*

Replace Syntax Rule 1) with:

1) There is a significance of ordering of <datetime field>s. In order from most significant to least significant, the ordering is: YEAR, MONTH, DAY, HOUR, MINUTE, and SECOND. A <start field> or <single datetime field> with an <interval leading field precision> *i* is more significant than a <start field> or <single datetime field> with an <interval leading field precision> *j* if *i*<*j*. An <end field> or <single datetime field> with an <interval fractional seconds precision> *i* is less significant than an <end field> or <single datetime field> with an <interval fractional seconds precision> *j* if *i*>*j*.

3. *Rationale: There is a contradiction between Syntax Rule 2) a) and General Rule 4. The Syntax Rule requires the <start field> to be more significant than the <end field> while the General Rule allows them to be the same. The Syntax Rule is modified to remove the contradiction.*

Replace Syntax Rule 2) with:

2) If TO is specified, then:

a) <start field> shall be more significant than, or equally significant to, the <end field>,
b) if <start field> specified MONTH, then <end field> shall specify MONTH, and
c) if <start field> specified YEAR, then <end field> shall specify either YEAR or MONTH.

4. *Rationale: Stipulate the precision of fields in an <interval qualifier>.*

Insert a Syntax Rule as follows:

6.1) The precision of a field other than the <start field> or <single datetime field> is
Case:
a) if the field is not SECOND, then 2
b) otherwise, 2 digits before the decimal point and the explicit or implicit <interval fractional seconds precision> digits after the decimal point.

5. *Rationale: Corrects incorrect computation of the length in positions of an item of type interval, when <end field> is SECOND.*

In General Rule 5) b)iii) 1), replace "plus 1 (the length in positions of the <colon> preceding the <end field>)" with "plus 3 (the length in positions of the <end field> other than any <fractional seconds precision>, plus the length in positions of its preceding <colon>)".

10.2 <language clause>, Table 16

1. *Rationale: Editorial. The (non-extended) Pascal standard should be identified as ISO/IEC 7185 rather than ISO 7185. The designation was changed in 1990 when the standard was revised.*

In Table 16, change "ISO 7185:1990" to "ISO/IEC 7185:1990".

10.4 <character set specification>

1. *Rationale: Clarify that references to non-existing objects is only allowed within the same <schema definition>.*

Add the following Syntax Rule:

3.1) Let *C* be the <character set name> contained in the <character set specification>. If the <character set specification> is not contained in a <schema definition>, then the schema identified by the explicit or implicit qualifier of the <character set name> shall include the descriptor of *C*. If the <character set specification> is contained in a <schema definition> *S*, then the schema identified by the explicit or implicit qualifier of the <character set name> shall include the descriptor of *C*, or *S* shall include a <schema element> that creates the descriptor of *C*.

10.5 <collate clause>

1. *Rationale: Clarify that references to non-existing objects are only allowed within the same <schema definition>.*

Add the following Syntax Rule:

0.1) Let *C* be the <collation name> contained in the <collate clause>. If the <collate clause> is not contained in a <schema definition>, then the schema identified by the explicit or implicit qualifier of the <collation name> shall include the descriptor of *C*. If the <collate clause> is contained in a

<schema definition> *S*, then the schema identified by the explicit or implicit qualifier of the <collation name> shall include the descriptor of *C*, or *S* shall include a <schema element> that creates the descriptor of *C*.

10.6 <constraint name definition> and <constraint attributes>

1. *Rationale: Editorial.*

In the **Note** following General Rule 3), replace "<SQL statement>" with "SQL-statement".

2. *Rationale: Editorial.*

In Leveling Rule 2) a), replace "Intermediate" with "Entry".

11.1 <schema definition>

1. *Rationale: Editorial.*

In Leveling Rule 2) a), replace "Intermediate" with "Entry".

11.2 <drop schema statement>

1. *Rationale: Editorial.*

In General Rule 5), replace "collation definition" with "collating sequence".

11.4 <column definition>

1. *Rationale: Clarify that references to non-existing objects is only allowed within the same <schema definition>.*

Add the following Syntax Rule:

4.1) If the <column definition> is not contained in a <schema definition>, then the schema identified by the explicit or implicit qualifier of the <domain name> shall include the descriptor of *D*. If the <column definition> is contained in a <schema definition> *S*, then the schema identified by the explicit or implicit qualifier of the <domain name> shall include the descriptor of *D*, or *S* shall include a <schema element> that creates the descriptor of *D*.

2. *Rationale: Editorial.*

In Access Rule 1), replace "*D*.." with "*D*.".

3. *Rationale: Editorial.*

In Leveling Rule 2) c), replace "Intermediate" with "Entry".

11.5 <default clause>

1. *Rationale: Editorial.*

In Syntax Rule 1), replace "<alter domain definition>" with "<alter domain statement>".

2. *Rationale: The following changes correct the fact that the current rules cause the value of, for example, CURRENT_USER to be evaluated at the time of the <table definition> or <column definition> and forever more used as the default value for the associated column in the table. The value of CURRENT_USER, etc., and the <datetime value function>s are being put directly into the column or domain descriptor at the time that the <default clause> is processed. It was intended that those things be evaluated at the time that the row was inserted into the table (or updated in the table). Additionally the new rule 2.1 defines how a default value of null is represented in COLUMN_DEFAULT in the DOMAINS and COLUMNS base table. The new Case c) of General Rule 2) covers the case of a derived table.*

Replace General Rules 1) and 2) with the following General Rules:

1) The default value inserted in the column descriptor, if the <default clause> is to apply to a column, or in the domain descriptor, if the <default clause> is to apply to a domain, is the <default option>. If the subject data type is bit string with fixed length, the <default clause> specifies a <bit string literal>, and the length of the <bit string literal> is less than the fixed length of the column, then a completion condition is raised: *warning — implicit zero-bit padding*.

2) The default value of a column is

   Case:

   a) If the column descriptor indicates a default value derived from a <default option>, then the value in the column descriptor;

   b) If the column descriptor includes a domain name that identifies a domain descriptor that includes a default value derived from a <default option>, then the value in the domain descriptor;

   c) If the default value is for a column *C* of a candidate row for insertion into or update of a derived table *DT* and *C* has a single counterpart column *CC* in a leaf generally underlying table of *DT*, then the default value of *CC* obtained by applying the General Rules of this Subclause.

   d) Otherwise, the null value.

2.1) When a default value is required for a column in a row of a table, the default value for the column is derived from the default value in the appropriate column descriptor or domain descriptor. Let *D* be that descriptor.

   Case:

   a) If *D* contains NULL, then the null value.

   b) If *D* contains a <literal>, then

      Case:

      i) If the subject data type is numeric, then the numeric value of the <literal>.

      ii) If the subject data type is character string with variable length, then the value of the <literal>.

iii) If the subject data type is character string with fixed length, then the value of the <literal>, extended as necessary on the right with <space>s to the length in characters of the subject data type.

iv) If the subject data type is bit string with variable length, then the value of the <literal>.

v) If the subject data type is bit string with fixed length, then the value of the <literal> extended as necessary on the right with 0-valued bits to the length of the subject data type.

vi) If the subject data type is datetime or interval, then the value of the <literal>.

c) If the column or domain descriptor specifies CURRENT_USER, SESSION_USER, or SYSTEM_USER, then

Case:

i) If the subject data type is character string with variable length, then the value obtained by an evaluation of CURRENT_USER, SESSION_USER, or SYSTEM_USER at the time that the default value is required.

ii) If the subject data type is character string with fixed length, then the value obtained by an evaluation of CURRENT_USER, SESSION_USER, or SYSTEM_USER at the time that the default value is required, extended as necessary on the right with <space>s to the length in characters of the subject data type.

d) If the column or domain descriptor contains a <datetime value function>, then the value of an evaluation of the <datetime value function> at the time that the default value is required.

e) Otherwise, the null value.

3. *Rationale: The following helps define the action taken when a default value cannot be represented in the Information Schema.*

Add the following General Rule:

3.1) If the <default clause> is contained in a <schema definition>, and if the character representation of the <default option> cannot be represented in the Information Schema without truncation, then a completion indication is raised: *warning — default option too long for information schema.*

11.6 <table constraint definition>

1. *Rationale: Clarify that table constraints have <schema name>s identical to that of the table that the constraint is built on.*

Add the following Syntax Rule:

1.1) If <constraint name definition> is specified, and if its <constraint name> contains a <schema name>, then that <schema name> shall be the same as the explicit or implicit <schema name> of the <table name> of the table identified by the containing <table definition> or <alter table statement>.

2. *Rationale: Editorial.*

In Leveling Rule 2) a), replace "Intermediate" with "Entry".

11.8 <referential constraint definition>

1. *Rationale: Whilst the intent of Syntax Rule 2) a) is reasonably clear, the actual wording is devoid of meaning. The "unique columns" of a unique constraint is a set of columns. It is not, and does not contain, a set of column names.*

In the first sentence of Syntax Rule 2) a) replace "the set of column names of that <reference column list> shall be equal to the set of column names in the unique columns" with "the set of <column name>s contained in that <reference column list> shall be equal to the set of <column name>s contained in the <unique column list>".

2. *Rationale: Clarify that references to non-existing objects are only allowed within the same <schema definition>.*

Add the following Syntax Rule:

9.1) Let *T* be the referenced table. If the <referential constraint definition> 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 <referential constraint definition> 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 include a <schema element> that creates the descriptor of *T*.

3. *Rationale: The General Rules that specify the completion condition 01001 (cursor operation conflict) pertain only to SQL3 and were inadvertently included in ISO/IEC 9075:1992. These rules need therefore to be deleted.*

Delete General Rule 8).

11.9 <check constraint definition>

1. *Rationale: The prohibition of <target specification> in the <search condition> of a <check constraint definition> is not sufficient to prevent a <parameter name> or <embedded variable name> as desired.*

Replace Syntax Rule 1) with:

1) The <search condition> shall not contain a <parameter name>, a <embedded variable name>, or a <dynamic parameter specification>.

11.11 <add column definition>

1. *Rationale: Editorial.*

Add the following General Rule:

0.1) Let *T* be the table identified by the <table name> immediately contained in the containing <alter table statement>.

11.15 <drop column definition>

1. Rationale: Editorial.

Replace Syntax Rule 4) with:

4) If RESTRICT is specified, then *C* shall not be referenced in the <query expression> of any view descriptor or in the <search condition> of any constraint descriptor other than a table constraint descriptor that contains references to no other column and that is included in the table descriptor of *T*.

Note: A <drop column definition> that does not specify CASCADE will fail if there are any references to that column resulting from the use of CORRESPONDING, NATURAL, SELECT * (except where contained in an <exists predicate>), or REFERENCES without a <reference column list> in its <referenced table and columns>.

Note: If CASCADE is specified, then any such dependent object will be dropped by the execution of the <revoke statement> specified in the General Rules of this Subclause.

11.16 <add table constraint definition>

1. Rationale: Provide General Rules for maintaining the nullability characteristic in a column definition.

Add the following General Rule:

2.1) Let *TC* be the table constraint added to *T*. If *TC* causes some column *CN* to be known not nullable and no other constraint causes *CN* to be known not nullable, then the nullability characteristic of the column descriptor of *CN* is changed to known not nullable.

Note: The nullability characteristic is defined in Subclause 4.8, “Columns”.

11.17 <drop table constraint definition>

1. *Rationale: Provide General Rules for maintaining the "nullability characteristic" in a column definition.*

Add the following General Rule:

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

Note: The nullability characteristic is defined in Subclause 4.8, "Columns".

11.18 <drop table statement>

1. *Rationale: A <drop behavior> of RESTRICT shall not be allowed if there are referential constraints that reference the table being dropped, and shall not consider check constraints of the table being dropped.*

Replace Syntax Rule 4) with:

4) If RESTRICT is specified, then *T* shall not be referenced in the <query expression> of any view descriptor, the <search condition> of any table check constraint descriptor of any table other than *T*, or the referenced table of any referential constraint descriptor of any table other than *T*.

Note: 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.

11.19 <view definition>

1. *Rationale: The prohibition of <target specification> in the <query specification> of a <view definition> is not sufficient to prevent a <parameter name> or <embedded variable name> as desired.*

Replace Syntax Rule 1) with:

1) The <query expression> shall not contain a <parameter name>, <embedded variable name>, or <dynamic parameter specification>.

2. *Rationale: A predicate specified for LOCAL CHECK OPTION is tested only for rows that would pass all lower level predicates. This does not reflect the correct semantics for LOCAL. Predicates specified in a derived table must also be considered by the WITH CHECK OPTION.*

Replace General Rule 9) with:

9) Let *V1* be an updatable view. Let *V2* be the simply underlying table of the simply underlying table of the <query expression> of *V1*. Let *S* be the set of check conditions of *V1* that is used for the validation of WITH CHECK OPTION. Each check condition of *S* is a primary <search condition> of a view. Let *TE* be the <table expression> directly contained in the <query expression> of the view. The primary <search condition> of a view is the <search condition> in the <where clause> of *TE* and the <search condition> immediately contained in every <where clause> simply contained in every <derived table> contained in *TE* without an intervening <where clause> or <having clause>. If the <table expression> does not contain a <where clause>, then the view has an implicit primary <search condition> that is always true.

   Case:

   a) If *V2* is a base table and the descriptor of *V1* does not include WITH CHECK OPTION, then *S* is empty.

   b) If *V2* is a base table and the descriptor of *V1* includes any form of WITH CHECK OPTION, then *S* consists of the primary <search condition> of *V1*.

   c) If *V2* is a view and the descriptor of *V1* does not include WITH CHECK OPTION, then *S* is identical to the set of check conditions of *V2*.

   d) If *V2* is a view and the descriptor of *V1* contains WITH CASCADED CHECK OPTION, then *S* consists of the primary <search condition> of *V1* and the primary <search condition> of each view that is a generally underlying table of *V1*.

   e) If *V2* is a view and the descriptor of *V1* contains WITH LOCAL CHECK OPTION, then *S* consists of the primary <search condition> of *V1* and the set of check conditions of *V2*.

Replace General Rule 11) with:

11) During an update operation on *V1*, every check condition in *S* is applied to every inserted or updated row. If any check condition is not true, then an exception condition is raised: *with check option violation*.

11.21 <domain definition>

1. *Rationale: Clarify that domain constraints have <schema name>s identical to that of the domain that the constraint is built on.*

Add the following Syntax Rule:

1.1) If <constraint name definition> is specified, and if its <constraint name> contains a <schema name>, then that <schema name> shall be the same as the explicit or implicit <schema name> of the <domain name> of the domain identified by the containing <domain definition> or <alter domain statement>.

2. *Rationale: In General Rule 3), a domain constraint descriptor is referenced, but it has not been created.*

Insert the following General Rule:

2.1) For every <domain constraint> immediately contained in the <domain definition>:
A domain constraint descriptor is created that describes the domain constraint being defined. The domain constraint descriptor includes the <constraint name> contained in the explicit or implicit <constraint name definition>. The domain constraint descriptor includes an indication of whether the constraint is deferrable or not deferrable, and whether the initial constraint mode of the constraint is *deferred* or *immediate*. The domain constraint descriptor includes the <search condition> immediately contained in the <check constraint definition> immediately contained in the <domain constraint>.

3. *Rationale: The following change specifies that when overriding the collation of a contained character set, a grantable USAGE privilege on that overriding collation is not needed to acquire a grantable USAGE privilege on the domain.*

Replace General Rule 4) with:

4) A privilege descriptor is created that defines the USAGE privilege on this domain to the <authorization identifier> of the <schema> or <module> in which the <domain definition> appears. This privilege is grantable if and only if the applicable privileges include a grantable REFERENCES privilege for each <column reference> included in the domain descriptor and a grantable USAGE privilege for each <domain name>, <collation name>, <character set name>, and <translation name> contained in the <search condition> of any domain constraint descriptor included in the domain descriptor. The grantor of the privilege descriptor is set to the special grantor value "_SYSTEM".

11.25 <add domain constraint definition>

1. *Rationale: Provide General Rules for maintaining the "nullability characteristic" in a column definition.*

Add the following General Rule:

1.1) If *DC* causes some column *CN* to be known not nullable and no other constraint causes *CN* to be known not nullable, then the nullability characteristic of the column descriptor of *CN* is changed to known not nullable.

Note: The nullability characteristic is defined in Subclause 4.8, "Columns".

11.26 <drop domain constraint definition>

1. *Rationale: Provide General Rules for maintaining the "nullability characteristic" in a column definition.*

Add the following General Rule:

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

Note: The nullability characteristic is defined in Subclause 4.8, "Columns".

11.27 <drop domain statement>

1. *Rationale: When a domain is destroyed it is necessary to destroy its domain constraint descriptors.*

Replace General Rule 3) with:

3) The identified domain is destroyed by destroying its descriptor, its data type descriptor, and its domain constraint descriptors.

11.28 <character set definition>

1. *Rationale: Editorial.*

In Syntax Rule 3), replace "character set descriptor" with "character set".

11.29 <drop character set statement>

1. *Rationale: The following change adds missing restrictions to deal with the convention that absence of a <drop behavior> production in the BNF of a schema manipulation statement indicates that the respective schema manipulation statement cannot drop objects while other objects depend on their existence.*

Replace Syntax Rule 3) with:

3) *C* shall not be referenced in the <query expression> of any view descriptor or in the <search condition> of any constraint descriptor, nor be included in any collation descriptor or translation descriptor or schema descriptor, nor be referenced in the <module character set specification> of any module, nor be included in any column's or any domain's data type descriptor.

11.30 <collation definition>

1. *Rationale: Clarify that references to non-existing objects is only allowed within the same <schema definition>.*

Add the following Syntax Rule:

10.1) If <translation name> is specified and the <collation definition> is not contained in a <schema definition>, then the schema identified by the explicit or implicit qualifier of the <translation name> shall include the descriptor of *T*. If the <collation definition> is contained in a <schema definition> *S*, then the schema identified by the explicit or implicit qualifier of the <translation name> shall include the descriptor of *T*, or *S* shall include a <schema element> that creates the descriptor of *T*.

11.31 <drop collation statement>

1. *Rationale: By convention, the absence of a <drop behavior> in a schema manipulation statement that executes a drop implies that no object that is dependent upon a dropped object will be dropped. In the absence of the following, constraints and views would be dropped when the <revoke statement> effectively executed by the General Rules of 11.31 <drop collation statement> is executed. Additionally, the COLLATION may not be dropped if it is used in a view definition or in a constraint definition.*

Add the following Syntax Rule:

2.1) *C* shall not be referenced in the <query expression> included in any view descriptor or in the <search condition> included in any constraint descriptor.

2. *Rationale: The following changes address the problems that string manipulations are specified on descriptors that are not defined to contain character strings, and that a convention is violated that absence of a <drop behavior> production implies that the Schema Manipulation Language fails if it causes other objects to be dropped. To achieve this, the defined contents of descriptors must be altered. Then, General Rules are reordered such that the effectively executed revoke statement happens after any domain descriptors that depend upon the collation being dropped have been altered to no longer have that dependency.*

Replace all the General Rules with:

1) For every collation descriptor *CD* that includes *CN*, *CD* is modified such that it does not include *CN*. If *CD* does not include any translation name, then *CD* is modified to indicate that it utilizes the DEFAULT collation for its character repertoire.

2) For every character set descriptor *CSD* that includes *CN*, *CSD* is modified such that it does not include *CN*. If *CSD* does not include any translation name, then *CSD* is modified to indicate that it utilizes the DEFAULT collation for its character repertoire.

3) For every column descriptor or domain descriptor *DD* that includes *CN*, *DD* is modified such that it does not contain *CN*.

Note: This causes the column or domain that *DD* is the descriptor for to revert to the default collation for its character set.

4) Let *A* be the current <authorization identifier>. The following <revoke statement> is effectively executed with a current <authorization identifier> of "_SYSTEM" and without further Access Rule checking:

REVOKE USAGE ON COLLATION *CN* FROM A CASCADE

5) The descriptor of *C* is destroyed.

11.32 <translation definition>

1. *Rationale: Syntax Rule 5) appears to be redundant when one consults the relevant production (which leads to <translation name>) and General Rule 24) of 5.4, Names and identifiers. There is a difference, since one rule refers to a translation and the other to a translation descriptor, but there is generally a close correspondence between such entities. The intention of Syntax Rule 5) may, however, be to require that the identified translation has already been defined. If this is so, and this supposition seems to be supported by the change to this rule made by TC1, then the rule should be modified to be both more explicit and to permit reference to another translation descriptor being created through the same <schema definition>. TC1 makes the intent more explicit, but does not address the possibility of the referenced <schema translation name> being created by the same <create schema> as contains the <translation definition>.*

Replace Syntax Rule 5) with:

5) Let *TD* be the translation descriptor identified by <schema translation name>. If the <translation definition> is not contained in a <schema definition>, then the schema identified by the explicit or implicit qualifier of the <schema translation name> shall include *TD*. If the <translation definition> is contained in a <schema definition> S, then the schema identified by the explicit or implicit qualifier of the <schema translation name> shall include *TD*, or *S* shall include a <schema element> that creates *TD*.

11.34 <assertion definition>

1. *Rationale: The prohibition of <target specification> in the <search condition> of an <assertion definition> is not sufficient to prevent a <parameter name> or <embedded variable name> as desired.*

Replace Syntax Rule 4) with:

4) The <search condition> shall not contain a <parameter name>, a <embedded variable name>, or a <dynamic parameter specification>.

11.36 <grant statement>

1. *Rationale: Clarify that references to non-existing objects is only allowed within the same <schema definition>.*

Add the following Syntax Rule:

3.1) If the <grant 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 the <grant 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.*

In General Rule 7), replace "<grantor>" with "grantor" and replace "<object>" with "object".

3. *Rationale: Provide missing rules that cover the acquisition of the necessary privileges to acquire the WITH GRANT OPTION on views through a grant to PUBLIC.*

In General Rule 7), replace "if *G* has been granted" with "if the user privileges of *G* contain the".

4. *Rationale: The following changes addresses the fact that <grant statement> and <revoke statement> are inconsistent with <collation definition> with regard to their treatment of the grantability of USAGE privilege on collations. The first change makes grantability of USAGE privilege against a domain more consistent with that of the grantability of USAGE privilege against a column, while the second change makes a collation USAGE privilege grantable only when the owner of that collation has a grantable USAGE privilege on both its contained translation, if any, and on its contained character set. For both changes, missing rules are provided that cover the acquisition of the necessary privileges to acquire the WITH GRANT OPTION on views through a grant to PUBLIC.*

Replace General Rule 8) with:

8) For every <grantee> *G* and for every domain *D1* owned by *G*, if the user privileges of *G* contain the REFERENCES privilege WITH GRANT OPTION on every column referenced in the <search condition> included in a domain constraint descriptor included in the domain descriptor of *D1* and a grantable USAGE privilege on all domains, character sets, collations, and translations whose <domain name>s, <character set name>s, <collation name>s, and <translation name>s, respectively, are included in the domain descriptor, then for every privilege descriptor with <privileges> USAGE, a grantor of "_SYSTEM", object *D1*, and <grantee> *G* that is not grantable, the following <grant statement> is effectively executed with a current <authorization identifier> of "_SYSTEM" and without further Access Rule checking:

GRANT USAGE ON DOMAIN *D1* TO *G* WITH GRANT OPTION

Replace General Rule 9) with:

9) For every <grantee> *G* and for every collation *C1* owned by *G*, if the user privileges of *G* contain a grantable USAGE privilege for the character set name included in the collation descriptor of *C1* and a grantable USAGE privilege for the translation name, if any, included in the collation descriptor of *C1*, then for every privilege descriptor with a <privileges> USAGE, a grantor of "_SYSTEM", object of *C1*, and <grantee> *G* that is not grantable, the following <grant statement> is effectively executed with a current <authorization identifier> of "_SYSTEM" and without further Access Rule checking:

GRANT USAGE ON COLLATION *C1* TO *G* WITH GRANT OPTION

In General Rule 10), replace "USAGE privilege of *G* for every character set identified by a <character set specification> contained in the <translation definition> of *T1* is grantable" with "user privileges of *G* contain a grantable USAGE privilege for every character set identified by a <character set specification> contained in the <translation definition> of *T1*".

5. *Rationale: Editorial.*

In General Rule 10), replace "<grantor>" with "grantor" and replace "<object>" with "object".

6. Rationale: Editorial.

Replace General Rule 11) with:

11) If <table name> is specified, then for each view *V* owned by some <grantee> *G* such that *T* or some column *CT* of *T* is referenced in the <query expression> of *V*, let RT_i, for *i* ranging from 1 to the number of tables identified by the <table reference>s contained in the <query expression> of *V*, be the <table name>s of those tables. For every column *CV* of *V*:

a) Let CRT_{ij}, for *j* ranging from 1 to the number of columns of RT_i that are underlying columns of *CV*, be the <column name>s of those columns.

b) If WITH GRANT OPTION was specified, then let *WGO* be "WITH GRANT OPTION"; otherwise, let *WGO* be a zero-length string.

c) If, following successful execution of the <grant statement>, the user privileges of *G* will contain a REFERENCES(CRT_{ij}) privilege for all *i* and for all *j*, and will contain a REFERENCES privilege on some column of RT_i, for all *i*, then the following <grant statement> is effectively executed as though the current <authorization identifier> were "_SYSTEM" and without further Access Rule checking:

GRANT REFERENCES (*CV*) ON *V* TO *G WGO*

11.37 <revoke statement>

1. Rationale: Editorial.

In Syntax Rule 6), replace "A privilege descriptor is" with "A privilege descriptor *P* is".

2. Rationale: The following change address problems in the definition of when a privilege descriptor D is allowed to be created by a grant permitted by P, so as to properly deal with REFERENCES privilege against views.

Replace Syntax Rule 7) b) iv) 2) with:

2) *P* and *D* are both column privilege descriptors, the privilege descriptor *D* identifies a <column name> *CVN* explicitly or implicitly contained in the <view column list> of a <view definition> *V* and either:

A) *V* is an updatable view. For each column *CV* identified by a <column name> *CVN*, there is a corresponding column in the underlying table of the <query expression> *TN*. Let *CTN* be the <column name> of the column of the <query expression> from which *CV* is derived. The action for *P* is UPDATE or INSERT and the identified column of *P* is *TN.CTN*; or

B) For every table *T* identified by a <table reference> contained in the <query expression> of *V* and for every column *CT* that is a column of *T* and an underlying column of *CV*, the action for *P* is REFERENCES and either the identified column of *P* is *CT* or the identified table of *P* is *T*.

3. *Rationale: The following change modifies the definition of "allowed to be created by" to reflect a collations dependency upon contained translations.*

Replace Syntax Rule 7) b) iv) 4) with:

4) The privilege descriptor *D* identifies the <collation name> of a <collation definition> *CO* and the identified character set name of *P* is included in the collation descriptor for *CO* or the identified translation name of *P* is included in the collation descriptor for *CO*.

4. *Rationale: The following change fixes the definition of when a privilege descriptor D is allowed to be created by a grant permitted by P, as that definition doesn't deal with USAGE privilege against domains properly.*

Add the following to Syntax Rule 7):

b.1) The following conditions hold:

i) The privilege descriptor for *D* indicates that its grantor is the special grantor value "_SYSTEM";

ii) The grantee of *P* is the owner of the domain identified by *D*, or the grantee of *P* is PUBLIC, and

iii) The privilege descriptor *D* identifies the <domain name> of a <domain definition> *DO* and either the column privilege descriptor *P* has an action of REFERENCES and identifies a column referenced in the <search condition> included in the domain descriptor for *DO*, or the privilege descriptor *P* has an action of USAGE and identifies a domain, collation, character set, or translation whose <domain name>, <collation name>, <character set name> or <translation name>, respectively, is contained in the <search condition> of the domain descriptor for *DO*.

5. *Rationale: The following changes deal with the need that when determining whether or not the owner of a view has a grantable REFERENCES privilege on a particular column of a view, the owner must have a grantable REFERENCES privilege on every column that is an underlying column of that column.*
This rule updates the definition of a modified privilege descriptor to deal with the case where revocation of REFERENCES privilege on one or more columns of a table RTi impacts one or more columns CVN of view V.

Replace Syntax Rule 10) c) with:

c) Case:

i) The following conditions hold:

1) *P* is not a REFERENCES column privilege descriptor that identifies a <column name> *CVN* explicitly or implicitly contained in the <view column list> of a <view definition> *V*, and;

2) Let *XO* and *XA* respectively be the identifier of the object identified by a privilege descriptor *X* and the action of *X*. Within the set of privilege descriptors upon which *P* directly depends, there exists some *XO* and *XA* for which the set of identified privilege descriptors unioned with the set of modified privilege descriptors include all privilege descriptors specifying the grant of *XA* on *XO* WITH GRANT OPTION.

ii) The following conditions hold:

1) *P* is a REFERENCES column privilege descriptor that identifies a column *CV* named by a <column name> *CVN* explicitly or implicitly contained in the <view column list> of a <view definition> *V*, and;

2) Let *SP* be the set of privileges upon which *P* directly depends. For every table *T* identified by a <table reference> contained in the <query expression> of *V*, let *RT* be the <table name> of *T*. There exists a column *CT* whose <column name> is *CRT*, such that:

A) *CT* is a column of *T* and an underlying column of *CV*, and

B) Every privilege descriptor *PD* that is the descriptor of some member of *SP* that specifies REFERENCES on *CRT* WITH GRANT OPTION is either an identified privilege descriptor for *CRT* or a modified privilege descriptor for *CRT*.

and

6. *Rationale: The following rule deals with the case where, due to a <revoke statement>, all REFERENCE privileges are lost against some underlying table of V, because in that instance all REFERENCE privileges should be lost against view V. The change above only deals with loss of grantability of REFERENCE against particular columns of V. To deal with this problem, it is necessary to define a new case where a privilege descriptor may become abandoned. [new item c)]. Also Syntax Rule 11) is structured as a list, which implies that all of the items in the list must be true for the privilege to be abandoned. This is inappropriate in that item b) is mutually exclusive with the new item c) i). Also indicate that the definition of abandoned doesn't depend upon whether or not GRANT OPTION FOR was specified.*

Replace Syntax Rule 11) with:

11) A privilege descriptor *P* is *abandoned* if:

Case:

a) It is not an independent node, and *P* is not itself an identified or a modified privilege descriptor, and there exists no path to *P* from any independent node other than paths that include an identified privilege descriptor or a modified privilege descriptor.

b) *P* is a SELECT column privilege descriptor and there exists a SELECT table privilege descriptor *X* with the same grantee, grantor, catalog name, schema name, and table name, and *X* is an abandoned privilege descriptor.

c) The following conditions hold:

i) *P* is a REFERENCES column privilege descriptor that identifies a <column name> *CVN* explicitly or implicitly contained in the <view column list> of a <view definition> *V*; and,

ii) Letting *SP* be the set of privileges upon which *P* directly depends, either:

1) There exists some table name *RT* such that:

A) *RT* is the name of the table identified by some <table reference> contained in the <query expression> of *V*, and

B) For every column privilege descriptor *CPD* that is the descriptor of some member of *SP* that specifies REFERENCES on *RT*, *CPD* is either an identified privilege descriptor for *RT* or an abandoned privilege descriptor for *RT*.

or

2) There exists some column name *CRT* such that:

A) *CRT* is the name of some column of the table of some <table reference> contained in the <query expression> of *V*, and

B) For every column privilege descriptor *CPD* that is the descriptor of some member of *SP* that specifies REFERENCES on *CRT*, *CPD* is either an identified privilege descriptor for *CRT* or an abandoned privilege descriptor for *CRT*.

7. *Rationale: The following changes address an inconsistency in that to have grantable USAGE privilege on a column that specifies <collate clause>, its owner doesn't need grantable USAGE privilege on that overriding collation. But, to do the same for a domain, the owner needs grantable USAGE privilege on the overriding collation. The following changes also provide a more consistent treatment of the <drop behavior> of the <revoke statement>.*

Replace Syntax Rule 17) with:

17) For every domain descriptor *DO* contained in *S1*, *DO* is said to be *lost* if the destruction of all abandoned privileged descriptors, and, if GRANT OPTION FOR is not specified, all identified privilege descriptors would result in *A1* no longer having USAGE privilege on any character set included in the data type descriptor included in *DO*.

Replace Syntax Rule 18) with:

18) For every table descriptor *TD* contained in *S1*, for every column descriptor *CD* included in *TD*, *CD* is said to be *lost* if either:

a) The destruction of all abandoned privilege descriptors, and, if GRANT OPTION FOR is not specified, all identified privilege descriptors would result in *A1* no longer having USAGE privilege on any character set included in the data type descriptor included in *CD*, or

b) The name of the domain *DN* included in *CD*, if any, identifies a lost domain descriptor and the destruction of all abandoned privilege descriptors, and, if GRANT OPTION FOR is not specified, all identified privilege descriptors, would result in *A1* no longer having USAGE privilege on any character set included in the data type descriptor of the domain descriptor of *DN*.

Add the following Syntax Rules:

18.1) For every module *MO*, let *G* be the <module authorization identifier> that owns *MO*. *MO* is said to be *lost* if the destruction of all abandoned privilege descriptors, and, if GRANT OPTION FOR is not specified, all identified privilege descriptors, would result in *G* no longer having USAGE privilege on the character set referenced in the <module character set specification> of *MO*.

18.2) Let *SD* be the descriptor of the schema *S1*. *SD* is said to be *lost* if the destruction of all abandoned privilege descriptors, and, if GRANT OPTION FOR is not specified, all identified privilege descriptors, would result in *A1* no longer having USAGE privilege on the default character set included in the schema descriptor *SD*.

18.3) For every domain descriptor *DO* contained in *S1*, *DO* is said to be *impacted* if *DO* is not lost, and the destruction of all abandoned privilege descriptors and, if GRANT OPTION FOR is not specified, all identified privilege descriptors, would result in *A1* no longer having USAGE privilege on the collation whose <collation name> is contained in *DO*.

18.4) For every column descriptor *CD* contained in a table descriptor contained in *S1*, *CD* is said to be *impacted* if *CD* is not lost and the destruction of all abandoned privilege descriptors and, if GRANT OPTION FOR is not specified, all identified privilege descriptors, would result in *A1* no longer having USAGE privilege on the collation whose <collation name> is contained in *CD*.

18.5) For every collation descriptor CN contained in S1, CN is said to be impacted if the destruction of all abandoned privilege descriptors and, if GRANT OPTION FOR is not specified, all identified privilege descriptors, would result in A1 no longer having USAGE privilege on the collation whose <collation name> is contained in CN as the name of the collation's <collation source>.

18.6) For every character set descriptor *CSD* contained in *S1*, *CSD* is said to be *impacted* if the destruction of all abandoned privilege descriptors and, if GRANT OPTION FOR is not specified, all identified privilege descriptors, would result in *A1* no longer having USAGE privilege on the collation whose <collation name> is contained in *CSD*.

18.7) If RESTRICT is specified, then there shall be no abandoned privilege descriptors, abandoned view descriptors, abandoned table constraint descriptors, abandoned assertion descriptors, abandoned domain constraint descriptors, lost domain descriptors, lost column descriptors, lost module descriptors, lost schema descriptors, impacted domain descriptors, impacted column descriptors, impacted collation descriptors, or impacted character set descriptors.

18.8) If CASCADE is specified, then the impact on a module that is determined to be a lost module is implementation-defined.

Delete General Rule 7).

8. Rationale: The following changes specify the necessary actions against lost objects, given the changes above.

Add the following General Rules:

7.1) For every lost column descriptor *CD*, let *S1.TN* be the <table name> of the table whose descriptor includes the descriptor *CD* and let *CN* be the <column name> of *CD*. The following <alter table statement> is effectively executed without further Access Rule checking:

ALTER TABLE *S1.TN* DROP COLUMN *CN* CASCADE

7.2) For every lost domain descriptor *DO*, let *S1.DN* be the <domain name> of *DO*. The following <drop domain statement> is effectively executed without further Access Rule checking:

DROP DOMAIN *S1.DN* CASCADE

7.3) For every lost schema descriptor *SD*, 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>.

9. *Rationale: The following changes describe the actions to take against impacted domains, columns, collations, and character sets.*

Add the following General Rules:

7.4) If the object identified by *O* is a collation, let *OCN* be the name of that collation.

7.5) For every impacted domain descriptor *DO*, *DO* is modified such that it does not include *OCN*.

7.6) For every impacted column descriptor *CD*, *CD* is modified such that it does not include *OCN*.

7.7) For every impacted collation descriptor *CN*, *CN* is modified such that it does not include *OCN*. If *CN* does not include any translation name, then *CN* is modified to indicate that it utilizes the DEFAULT collation for its character repertoire.

7.8) For every impacted character set descriptor *CSD*, *CSD* is modified such that it does not include *OCN*. If *CSD* does not include any translation name, then *CSD* is modified to indicate that it utilizes the DEFAULT collation for its character repertoire.

12.3 <procedure>

1. *Rationale: <get diagnostics statement> input parameters must also be established.*

Add the following General Rule 6) b) i.1):

i.1) The values of all input parameters to the <procedure> are established.

2. *Rationale: Remove the incorrect suggestion that the <set transaction statement> directly affects the constraint mode of any constraint.*

Replace General Rule 6) c) iii) 2) with:

2) Case:

A) If a <set transaction statement> has been executed since the termination of the last SQL-transaction in the SQL-session, then the access mode and isolation level of the SQL-transaction are set as specified by the <set transaction statement>. If a <set constraints mode statement> *SCM* has been executed since the termination of the last SQL-transaction in the SQL-session, then the constraint modes of constraints specified in *SCM* are set as specified in *SCM*.

B) Otherwise, the access mode of that SQL-transaction is read-write and the isolation level of that SQL-transaction is SERIALIZABLE, and the constraint modes for all constraints in that SQL-transaction are set to their initial states.

3. *Rationale: Resolve inconsistency regarding "no data" and exception condition.*

Delete General Rules 8) and 9).

4. *Rationale: Clarify with respect to the language added to Subclause 4.18.1; to permit implementation-defined 01 warnings to be returned in SQLSTATE when no exceptions are specified by the Standard.*

Replace General Rule 11).a).ii) with:

ii) Either a completion condition is raised: *successful completion*, or a completion condition is raised: *warning*, or a completion condition is raised: *no data*.

5. *Rationale: The Leveling Rules for <procedure> specify that in Entry SQL a <parameter declaration> shall not specify a <data type> that is CHARACTER VARYING. However, the Leveling Rules for <data type> (subclause 6.1) are more restrictive, prohibiting the variable length forms of both character and national character, <datetime type> and <interval type>, <national character string type> (thus getting variable length national character strings in two separate ways), and types that specify CHARACTER SET. For Intermediate SQL, <procedure> prohibits only specification of BIT and BIT VARYING, whilst <data type> makes the same prohibition through reference to <bit string type>, and also prohibits specification of <time precision> and <timestamp precision>. There is no obvious reason why the restrictions specified for <procedure> should be weaker than those specified for <data type>. The relevant rules for <procedure> should therefore either be made an exact copy of those for <data type>, or deleted. In the interests of economy and the avoidance of update anomalies, it seems preferable to delete them.*

Replace Leveling Rule 1) a) with:

None.

Replace Leveling Rule 2) a) with:

None.

12.4 Calls to a <procedure>

1. *Rationale: The newly revised Ada language standard (ISO/IEC-8652:1995, Information technology — Programming languages — Ada) contains support for decimal-encoded numeric data and variable length character strings. The revised interface allows newly written applications in the revised Ada language access to these features of SQL. However, support for programs written against the old Ada interface is preserved.*

Replace Syntax Rule 1) with:

1) If the subject <language clause> specifies ADA then:

   a) The SQL-implementation shall generate the source code of an Ada library unit package the name of which shall be identical to the <module name> of the <module> if <module name> is a valid Ada identifier and is otherwise implementation-defined. For each <procedure> of the <module> there shall appear within the package defined above a subprogram declaration declaring a procedure; the name of that procedure shall be <procedure name> if <procedure name> is a valid Ada identifier and is otherwise implementation-defined; the parameters in the Ada procedure declarations shall appear in the same order as the <parameter declarations> of the <procedure> and shall have the same <parameter names> as the <parameter declarations> if those names are valid Ada identifiers, otherwise the names are implementation-defined; the parameter modes and subtype marks used in the parameter specifications are constrained by the remaining paragraphs of this subclause.

   b) Any <data type> in a <parameter declaration> shall specify CHARACTER, CHARACTER VARYING, NATIONAL CHARACTER, NATIONAL CHARACTER VARYING, BIT, BIT VARYING, NUMERIC, SMALLINT, INTEGER, REAL, DOUBLE PRECISION.

   c) The types of parameter specifications within the Ada subprogram declarations shall be taken from the library unit package, `Interfaces.SQL` and its children: `Numerics` and `Varying` and optional children *`Adacsn`* and *`Adacsn`*`.Varying`.

      i) The declaration of the library unit package, `Interfaces.SQL`, shall conform to the following template:

```
package Interfaces.SQL is
  -- the declarations of CHAR and NCHAR
  -- may be subtype declarations
  type CHAR is see the rules
  type NCHAR is see the rules
  type BIT is array (POSITIVE range <>) of BOOLEAN;
  type SMALLINT is range bs..ts;
  type INT is range bi..ti;
  type REAL is digits dr;
  type DOUBLE_PRECISION is digits dd;
  type SQLCODE_TYPE is range bsc..tsc;
  subtype SQL_ERROR is SQLCODE_TYPE range SQLCODE_TYPE'FIRST .. -1;
  subtype NOT_FOUND is SQLCODE_TYPE range 100..100;
  subtype INDICATOR_TYPE is t;
  type SQLSTATE_TYPE is new CHAR (1 .. 5);
  package SQLSTATE_CODES is
    AMBIGUOUS_CURSOR_NAME_NO_SUBCLASS:
      constant SQLSTATE_TYPE :="3C000";
    CARDINALITY_VIOLATION_NO_SUBCLASS:
      constant SQLSTATE_TYPE :="21000";
    CONNECTION_EXCEPTION_NO_SUBCLASS:
      constant SQLSTATE_TYPE :="08000";
```

```
CONNECTION_EXCEPTION_CONNECTION_DOES_NOT_EXIST:
  constant SQLSTATE_TYPE :="08003";
CONNECTION_EXCEPTION_CONNECTION_FAILURE:
  constant SQLSTATE_TYPE :="08006";
CONNECTION_EXCEPTION_CONNECTION_NAME_IN_USE:
  constant SQLSTATE_TYPE :="08002";
CONNECTION_EXCEPTION_SQLCLIENT_UNABLE_TO_ESTABLISH_SQLCONNECTION:
  constant SQLSTATE_TYPE :="08001";
CONNECTION_EXCEPTION_SQLSERVER_REJECTED_ESTABLISHMENT_OF_SQLCONNECTION:
  constant SQLSTATE_TYPE :="08004";
CONNECTION_EXCEPTION_TRANSACTION_RESOLUTION_UNKNOWN:
  constant SQLSTATE_TYPE :="08007";
DATA_EXCEPTION_NO_SUBCLASS:
  constant SQLSTATE_TYPE :="22000";
DATA_EXCEPTION_CHARACTER_NOT_IN_REPERTOIRE:
  constant SQLSTATE_TYPE :="22021";
DATA_EXCEPTION_DATETIME_FIELD_OVERFLOW:
  constant SQLSTATE_TYPE :="22008";
DATA_EXCEPTION_DIVISION_BY_ZERO:
  constant SQLSTATE_TYPE :="22012";
DATA_EXCEPTION_ERROR_IN_ASSIGNMENT:
  constant SQLSTATE_TYPE :="22005";
DATA_EXCEPTION_INDICATOR_OVERFLOW:
  constant SQLSTATE_TYPE :="22022";
DATA_EXCEPTION_INTERVAL_FIELD_OVERFLOW:
  constant SQLSTATE_TYPE :="22015";
DATA_EXCEPTION_INVALID_CHARACTER_VALUE_FOR_CAST:
  constant SQLSTATE_TYPE :="22018";
DATA_EXCEPTION_INVALID_DATETIME_FORMAT:
  constant SQLSTATE_TYPE :="22007";
DATA_EXCEPTION_INVALID_ESCAPE_CHARACTER:
  constant SQLSTATE_TYPE :="22019";
DATA_EXCEPTION_INVALID_ESCAPE_SEQUENCE:
  constant SQLSTATE_TYPE :="22025";
DATA_EXCEPTION_INVALID_INTERVAL_FORMAT:
  constant SQLSTATE_TYPE := "22006”;
DATA_EXCEPTION_INVALID_PARAMETER_VALUE:
  constant SQLSTATE_TYPE :="22023";
DATA_EXCEPTION_INVALID_TIME_ZONE_DISPLACEMENT_VALUE:
  constant SQLSTATE_TYPE :="22009";
DATA_EXCEPTION_NULL_VALUE_NO_INDICATOR_PARAMETER:
  constant SQLSTATE_TYPE :="22002";
DATA_EXCEPTION_NUMERIC_VALUE_OUT_OF_RANGE:
  constant SQLSTATE_TYPE :="22003";
DATA_EXCEPTION_STRING_DATA_LENGTH_MISMATCH:
  constant SQLSTATE_TYPE :="22026";
DATA_EXCEPTION_STRING_DATA_RIGHT_TRUNCATION:
  constant SQLSTATE_TYPE :="22001";
DATA_EXCEPTION_SUBSTRING_ERROR:
  constant SQLSTATE_TYPE :="22011";
DATA_EXCEPTION_TRIM_ERROR:
  constant SQLSTATE_TYPE :="22027";
DATA_EXCEPTION_UNTERMINATED_C_STRING:
  constant SQLSTATE_TYPE :="22024";
DEPENDENT_PRIVILEGE_DESCRIPTORS_STILL_EXIST_NO_SUBCLASS:
  constant SQLSTATE_TYPE :="2B000";
DYNAMIC_SQL_ERROR_NO_SUBCLASS:
  constant SQLSTATE_TYPE :="07000";
DYNAMIC_SQL_ERROR_CURSOR_SPECIFICATION_CANNOT_BE_EXECUTED:
  constant SQLSTATE_TYPE :="07003";
DYNAMIC_SQL_ERROR_INVALID_DESCRIPTOR_COUNT:
  constant SQLSTATE_TYPE :="07008";
DYNAMIC_SQL_ERROR_INVALID_DESCRIPTOR_INDEX:
  constant SQLSTATE_TYPE :="07009";
DYNAMIC_SQL_ERROR_PREPARED_STATEMENT_NOT_A_CURSOR_SPECIFICATION:
  constant SQLSTATE_TYPE :="07005";
```

```
      DYNAMIC_SQL_ERROR_RESTRICTED_DATA_TYPE_ATTRIBUTE_VIOLATION:
        constant SQLSTATE_TYPE :="07006";
      DYNAMIC_SQL_ERROR_USING_CLAUSE_DOES_NOT_MATCH_DYNAMIC_PARAMETER_SPEC:
        constant SQLSTATE_TYPE :="07001";
      DYNAMIC_SQL_ERROR_USING_CLAUSE_DOES_NOT_MATCH_TARGET_SPEC:
        constant SQLSTATE_TYPE :="07002";
      DYNAMIC_SQL_ERROR_USING_CLAUSE_REQUIRED_FOR_DYNAMIC_PARAMETERS:
        constant SQLSTATE_TYPE :="07004";
      DYNAMIC_SQL_ERROR_USING_CLAUSE_REQUIRED_FOR_RESULT_FIELDS:
        constant SQLSTATE_TYPE :="07007";
      FEATURE_NOT_SUPPORTED_NO_SUBCLASS:
        constant SQLSTATE_TYPE :="0A000";
      FEATURE_NOT_SUPPORTED_MULTIPLE_ENVIRONMENT_TRANSACTIONS:
        constant SQLSTATE_TYPE :="0A001";
      INTEGRITY_CONSTRAINT_VIOLATION_NO_SUBCLASS:
        constant SQLSTATE_TYPE :="23000";
      INVALID_AUTHORIZATION_SPECIFICATION_NO_SUBCLASS:
        constant SQLSTATE_TYPE :="28000";
      INVALID_CATALOG_NAME_NO_SUBCLASS:
        constant SQLSTATE_TYPE :="3D000";
      INVALID_CHARACTER_SET_NAME_NO_SUBCLASS:
        constant SQLSTATE_TYPE :="2C000";
      INVALID_CONDITION_NUMBER_NO_SUBCLASS:
        constant SQLSTATE_TYPE :="35000";
      INVALID_CONNECTION_NAME_NO_SUBCLASS:
        constant SQLSTATE_TYPE :="2E000";
      INVALID_CURSOR_NAME_NO_SUBCLASS:
        constant SQLSTATE_TYPE :="34000";
      INVALID_CURSOR_STATE_NO_SUBCLASS:
        constant SQLSTATE_TYPE :="24000";
      INVALID_SCHEMA_NAME_NO_SUBCLASS:
        constant SQLSTATE_TYPE :="3F000";
      INVALID_SQL_DESCRIPTOR_NAME_NO_SUBCLASS:
        constant SQLSTATE_TYPE :="33000";
      INVALID_SQL_STATEMENT_NAME_NO_SUBCLASS:
        constant SQLSTATE_TYPE :="26000";
      INVALID_TRANSACTION_STATE_NO_SUBCLASS:
        constant SQLSTATE_TYPE :="25000";
      INVALID_TRANSACTION_TERMINATION_NO_SUBCLASS:
        constant SQLSTATE_TYPE :="2D000";
      NO_DATA_NO_SUBCLASS:
        constant SQLSTATE_TYPE :="02000";
      REMOTE_DATABASE_ACCESS_NO_SUBCLASS:
        constant SQLSTATE_TYPE :="HZ000”;
      REMOTE_DATABASE_ACCESS_REMOTE_DATABASE_ACCESS_ACCESS_CONTROL_VIOLATION:
        constant SQLSTATE_TYPE :="HZ010”;
      REMOTE_DATABASE_ACCESS_REMOTE_DATABASE_ACCESS_BAD_REPETITION_COUNT:
        constant SQLSTATE_TYPE :="HZ020”;
      REMOTE_DATABASE_ACCESS_REMOTE_DATABASE_ACCESS_COMMAND_HANDLE_UNKNOWN:
        constant SQLSTATE_TYPE :="HZ030”;

REMOTE_DATABASE_ACCESS_REMOTE_DATABASE_ACCESS_CONTROL_AUTHENTICATION_FAILURE:
        constant SQLSTATE_TYPE :="HZ040”;
      REMOTE_DATABASE_ACCESS_CONTROL_SERVICES_NOT_ALLOWED:
        constant SQLSTATE_TYPE :="HZ230”;
      REMOTE_DATABASE_ACCESS_DATA_RESOURCE_HANDLE_NOT_SPECIFIED:
        constant SQLSTATE_TYPE :="HZ050”;
      REMOTE_DATABASE_ACCESS_DATA_RESOURCE_HANDLE_UNKNOWN:
        constant SQLSTATE_TYPE :="HZ060”;
      REMOTE_DATABASE_ACCESS_DATA_RESOURCE_NAME_NOT_SPECIFIED:
        constant SQLSTATE_TYPE :="HZ070”;
      REMOTE_DATABASE_ACCESS_DATA_RESOURCE_NOT_AVAILABLE_PREMANENT:
        constant SQLSTATE_TYPE :="HZ080”;
      REMOTE_DATABASE_ACCESS_DATA_RESOURCE_NOT_AVAILABLE_TRANSIENT:
```

```
    constant SQLSTATE_TYPE :="HZ081";
REMOTE_DATABASE_ACCESS_DATA_RESOURCE_ALREADY_OPEN:
    constant SQLSTATE_TYPE :="HZ090";
REMOTE_DATABASE_ACCESS_DATA_RESOURCE_UNKNOWN:
    constant SQLSTATE_TYPE :="HZ100";
REMOTE_DATABASE_ACCESS_DIALOGUE_ID_UNKNOWN:
    constant SQLSTATE_TYPE :="HZ110";
REMOTE_DATABASE_ACCESS_DUPLICATE_COMMAND_HANDLE:
    constant SQLSTATE_TYPE :="HZ120";
REMOTE_DATABASE_ACCESS_DUPLICATE_DATA_RESOURCE_HANDLE:
    constant SQLSTATE_TYPE :="HZ130";
REMOTE_DATABASE_ACCESS_DUPLICATE_DIALOGUE_ID:
    constant SQLSTATE_TYPE :="HZ140";
REMOTE_DATABASE_ACCESS_DUPLICATE_OPERATION_ID:
    constant SQLSTATE_TYPE :="HZ150";
REMOTE_DATABASE_ACCESS_INVALID_SEQUENCE:
    constant SQLSTATE_TYPE :="HZ160";
REMOTE_DATABASE_ACCESS_DIALOGUE_ALREADY_ACTIVE:
    constant SQLSTATE_TYPE :="HZ161";
REMOTE_DATABASE_ACCESS_DIALOGUE_INITIALIZING:
    constant SQLSTATE_TYPE :="HZ162";
REMOTE_DATABASE_ACCESS_DIALOGUE_NOT_ACTIVE:
    constant SQLSTATE_TYPE :="HZ163";
REMOTE_DATABASE_ACCESS_DIALOGUE_TERMINATING:
    constant SQLSTATE_TYPE :="HZ164";
REMOTE_DATABASE_ACCESS_TRANSACTION_NOT_OPEN:
    constant SQLSTATE_TYPE :="HZ165";
REMOTE_DATABASE_ACCESS_TRANSACTION_OPEN:
    constant SQLSTATE_TYPE :="HZ166";
REMOTE_DATABASE_ACCESS_TRANSACTION_TERMINATING:
    constant SQLSTATE_TYPE :="HZ167";
REMOTE_DATABASE_ACCESS_NO_DATA_RESOURCE_AVAILABLE:
    constant SQLSTATE_TYPE :="HZ170";
REMOTE_DATABASE_ACCESS_OPERATION_ABORTED_PERMANENT:
    constant SQLSTATE_TYPE :="HZ180";
REMOTE_DATABASE_ACCESS_OPERATION_ABORTED_TRANSIENT:
    constant SQLSTATE_TYPE :="HZ181";
REMOTE_DATABASE_ACCESS_OPERATION_CANCELLED:
    constant SQLSTATE_TYPE :="HZ190";
REMOTE_DATABASE_ACCESS_SERVICE_NOT_NEGOTIATED:
    constant SQLSTATE_TYPE :="HZ200";
REMOTE_DATABASE_ACCESS_TRANSACTION_ROLLED_BACK:
    constant SQLSTATE_TYPE :="HZ210";
REMOTE_DATABASE_ACCESS_USER_AUTHENTICATION_FAILURE:
    constant SQLSTATE_TYPE :="HZ220";
REMOTE_DATABASE_ACCESS_HOST_IDENTIFIER_ERROR:
    constant SQLSTATE_TYPE :="HZ300";
REMOTE_DATABASE_ACCESS_INVALID_SQL_CONFORMANCE_LEVEL:
    constant SQLSTATE_TYPE :="HZ310";
REMOTE_DATABASE_ACCESS_RDA_TRANSACTION_NOT_OPEN:
    constant SQLSTATE_TYPE :="HZ320";
REMOTE_DATABASE_ACCESS_RDA_TRANSACTION_OPEN:
    constant SQLSTATE_TYPE :="HZ325";
REMOTE_DATABASE_ACCESS_SQL_ACCESS_CONTROL_VIOLATION:
    constant SQLSTATE_TYPE :="HZ330";
REMOTE_DATABASE_ACCESS_SQL_DATABASE_RESOURCE_ALREADY_OPEN:
    constant SQLSTATE_TYPE :="HZ340";
REMOTE_DATABASE_ACCESS_SQL_DBL_ARGUMENT_COUNT_MISMATCH:
    constant SQLSTATE_TYPE :="HZ350";
REMOTE_DATABASE_ACCESS_SQL_DBL_ARGUMENT_TYPE_MISMATCH:
    constant SQLSTATE_TYPE :="HZ360";
REMOTE_DATABASE_ACCESS_SQL_DBL_NO_CHAR_SET:
    constant SQLSTATE_TYPE :="HZ365";
REMOTE_DATABASE_ACCESS_SQL_DBL_TRANSACTION_STATEMENT_NOT_ALLOWED:
    constant SQLSTATE_TYPE :="HZ370";
REMOTE_DATABASE_ACCESS_SQL_USAGE_MODE_VIOLATION:
```

```
  constant SQLSTATE_TYPE :="HZ380";
REMOTE_DATABASE_ACCESS_ABORT_FAILURE_SERVICE_PROVIDER:
  constant SQLSTATE_TYPE :="HZ410";
REMOTE_DATABASE_ACCESS_ABORT_FAILURE_SERVICE_USER:
  constant SQLSTATE_TYPE :="HZ411";
REMOTE_DATABASE_ACCESS_ASSOCIATION_FAILURE_PERMANENT:
  constant SQLSTATE_TYPE :="HZ420";
REMOTE_DATABASE_ACCESS_ASSOCIATION_FAILURE_TRANSIENT:
  constant SQLSTATE_TYPE :="HZ421";
REMOTE_DATABASE_ACCESS_RELEASE_FAILURE:
  constant SQLSTATE_TYPE :="HZ430";
REMOTE_DATABASE_ACCESS_BEGIN_DIALOGUE_REJECTED_PROVIDER:
  constant SQLSTATE_TYPE :="HZ450";
REMOTE_DATABASE_ACCESS_BEGIN_DIALOGUE_REJECTED_USER:
  constant SQLSTATE_TYPE :="HZ451";
REMOTE_DATABASE_ACCESS_HEURISTIC_HAZARD:
  constant SQLSTATE_TYPE :="HZ460";
REMOTE_DATABASE_ACCESS_HEURISTIC_MIX:
  constant SQLSTATE_TYPE :="HZ461";
REMOTE_DATABASE_ACCESS_PABORT_ROLLBACK_FALSE:
  constant SQLSTATE_TYPE :="HZ470";
REMOTE_DATABASE_ACCESS_PABORT_ROLLBACK_TRUE:
  constant SQLSTATE_TYPE :="HZ471";
REMOTE_DATABASE_ACCESS_ROLLBACK:
  constant SQLSTATE_TYPE :="HZ480";
REMOTE_DATABASE_ACCESS_UABORT_ROLLBACK_FALSE:
  constant SQLSTATE_TYPE :="HZ490";
REMOTE_DATABASE_ACCESS_UABORT_ROLLBACK_TRUE:
  constant SQLSTATE_TYPE :="HZ491";
REMOTE_DATABASE_ACCESS_UERROR:
  constant SQLSTATE_TYPE :="HZ4A0";
SUCCESSFUL_COMPLETION_NO_SUBCLASS:
  constant SQLSTATE_TYPE :="00000";
SYNTAX_ERROR_OR_ACCESS_RULE_VIOLATION_NO_SUBCLASS:
  constant SQLSTATE_TYPE :="42000";
TRANSACTION_ROLLBACK_NO_SUBCLASS:
  constant SQLSTATE_TYPE :="40000";
TRANSACTION_ROLLBACK_INTEGRITY_CONSTRAINT_VIOLATION:
  constant SQLSTATE_TYPE :="40002";
TRANSACTION_ROLLBACK_SERIALIZATION_FAILURE:
  constant SQLSTATE_TYPE :="40001";
TRANSACTION_ROLLBACK_STATEMENT_COMPLETION_UNKNOWN:
  constant SQLSTATE_TYPE :="40003";
TRIGGERED_DATA_CHANGE_VIOLATION_NO_SUBCLASS:
  constant SQLSTATE_TYPE :="27000";
WARNING_NO_SUBCLASS:
  constant SQLSTATE_TYPE :="01000";
WARNING_CURSOR_OPERATION_CONFLICT:
  constant SQLSTATE_TYPE :="01001";
WARNING_DEFAULT_OPTION_TOO_LONG_FOR_INFORMATION_SCHEMA:
  constant SQLSTATE_TYPE := "0100B";
WARNING_DISCONNECT_ERROR:
  constant SQLSTATE_TYPE :="01002";
WARNING_IMPLICIT_ZERO_BIT_PADDING:
  constant SQLSTATE_TYPE :="01008";
WARNING_INSUFFICIENT_ITEM_DESCRIPTOR_AREAS:
  constant SQLSTATE_TYPE :="01005";
WARNING_NULL_VALUE_ELIMINATED_IN_SET_FUNCTION:
  constant SQLSTATE_TYPE :="01003";
WARNING_PRIVILEGE_NOT_GRANTED:
  constant SQLSTATE_TYPE :="01007";
WARNING_PRIVILEGE_NOT_REVOKED:
  constant SQLSTATE_TYPE :="01006";
WARNING_QUERY_EXPRESSION_TOO_LONG_FOR_INFORMATION_SCHEMA:
```

```
        constant SQLSTATE_TYPE :="0100A";
      WARNING_SEARCH_CONDITION_TOO_LONG_FOR_INFORMATION_SCHEMA:
        constant SQLSTATE_TYPE :="01009";
      WARNING_STRING_DATA_RIGHT_TRUNCATION_WARNING:
        constant SQLSTATE_TYPE :="01004";
      WITH_CHECK_OPTION_VIOLATION_NO_SUBCLASS:
        constant QLSTATE_TYPE :="44000";
    end SQLSTATE_CODES;
end Interfaces.SQL;
```

where *bs*, *ts*, *bi*, *ti*, *dr*, *dd*, *bsc* and *tsc* are implementation-defined integral values; *t* is INT or SMALLINT corresponding to the implementation-defined <exact numeric type> of indicator parameters.

ii) The library unit package `Interfaces.SQL.Numerics` shall contain a sequence of decimal fixed point type declarations of the following form.

```
type Scale_s is delta 10.0 ** -s digits max_p;
```

where *s* is an integer ranging from 0 to an implementation-defined maximum value and *max_p* is an implementation-defined integer maximum precision.

iii) The library unit package `Interfaces.SQL.Varying`, shall contain type or subtype declarations with the defining identifiers CHAR, NCHAR and BIT.

iv) Let *SQLcsn* be a <character set name> and let *Adacsn* be the result of replacing <period>'s in *SQLcsn* with <underscore>. If *Adacsn* is a valid Ada identifier, then the library unit packages `Interfaces.SQL.`*`Adacsn`* and `Interfaces.SQL.`*`Adacsn`*`.Varying` shall contain a type or subtype declaration with defining identifier CHAR. If *Adacsn* is not a valid Ada identifier, the names of these packages shall be implementation-defined.

v) `Interfaces.SQL` and its children may contain context clauses and representation items as needed. These packages may also contain declarations of Ada character types as needed to support the declarations of the types CHAR and NCHAR.

Note: If the implementation-defined character set specification used by default with a CHARACTER data type is Latin1, then the declaration

```
subtype CHAR is String;
```

within `Interfaces.SQL` and the declaration

```
subtype CHAR is Ada.Strings.Unbounded.Unbounded_String;
```

within `Interfaces.SQL.Varying` (assuming the appropriate context clause), conform to the requirements of this paragraph of this subclause. If the character set underlying NATIONAL CHARACTER is supported by an Ada package specification *Host_Char_Pkg* that declares a type *String_Type* that stores strings over the given character set, and furthermore the package specification *Host_Char_Pkg_Varying* (not necessary distinct from *Host_Char_Pkg*) declares a type *String_Type_Varying* that reproduces the functionality of Ada.Strings.Unbounded.Unbounded_String over the national character type (rather than Latin1), then the declaration

```
subtype NCHAR is Host_Char_Pkg.String_Type;
```

within Interfaces.SQL and the declaration

```
subtype NCHAR is Host_Char_Pkg_Varying.String_Type_Varying;
```

within `Interfaces.SQL.Varying` conform to the requirements of this paragraph. Similar comments apply to other character sets and the packages `Interfaces.SQL.`*`Adacsn`* and `Interfaces.SQL.`*`Adacsn`*`.Varying`.

vi) The library unit package `Interfaces.SQL` shall contain declarations of the following form:

```
package CHARACTER_SET renames Interfaces.SQL.Adacsn;
  subtype CHARACTER_TYPE is CHARACTER_SET.cst;
```

where *cst* is a data type capable of storing a single character from the default character set. The package `Interfaces.SQL.`*`Adacsn`* shall contain the necessary declaration for *cst*.

Note: If the default character set is Latin1, then a declaration of the form:

```
package CHARACTER_SET is
  subtype cst is Character;
end CHARACTER_SET;
```

may be substituted for the renaming declaration of CHARACTER_SET.

d) The base type of the SQLCODE parameter shall be `Interfaces.SQL.SQLCODE_TYPE`. The base type of the SQLSTATE parameter shall be `Interfaces.SQL.SQLSTATE_TYPE`.

Note: SQLSTATE is the preferred status parameter. The SQLCODE status parameter is a deprecated feature that is supported for compatibility with earlier versions of this International Standard. See Annex D, "Deprecated Features".

e) The Ada parameter mode of the SQLCODE parameter is **out**. The Ada parameter mode of the SQLSTATE parameter is **out**.

f) If the *i*-th <parameter declaration> specifies a <data type> that is

i) CHARACTER(*L*) for some *L*, then the subtype mark in the *i*-th parameter declaration shall specify `Interfaces.SQL.CHAR`;

ii) CHARACTER VARYING(*L*) for some *L*, then the subtype mark in the *i*-th parameter declaration shall specify `Interfaces.SQL.VARYING.CHAR;`

iii) NATIONAL CHARACTER(*L*) for some *L*, then the subtype mark in the *i*-th parameter declaration shall specify `Interfaces.SQL.NCHAR;`

iv) NATIONAL CHARACTER VARYING(*L*) for some *L*, then the subtype mark in the *i*-th parameter declaration shall specify `Interfaces.SQL.VARYING.NCHAR;`

v) CHARACTER(*L*) CHARACTER SET *csn* for some *L* and some character set name, *csn*, then the subtype mark in the *i*-th parameter declaration shall specify `Interfaces.SQL.`*`Adacsn`*`.CHAR;`

vi) CHARACTER VARYING(*L*) CHARACTER SET *csn* for some *L* and some character set name, *csn*, then the subtype mark in the *i*-th parameter declaration shall specify `Interfaces.SQL.`*`Adacsn`*`.VARYING.CHAR`

If *P* is an actual parameter associated with the *i*-th parameter in a call to the encompassing procedure, then *P* shall be sufficient to hold a character string of length *L* in the appropriate character set.

Note: If a character set uses fixed length encodings then the definition of the subtype CHAR for fixed length strings may be an array type whose element type is an Ada character type. If that character type is defined so as to use the number of bits per character used by the SQL encoding, then the restriction on *P* is precisely P'LENGTH = *L*. For variable length strings using fixed length encodings, if the definition of CHAR in the appropriate VARYING package is based on the type Ada.Strings.Unbounded.Unbounded_String, there is no restriction on *P*. Otherwise, a precise statement of the restriction on *P* is implementation-defined.

g) If the *i*-th <parameter declaration> specifies a <data type> that is

i) BIT(*L*) for some length *L*, then the subtype mark in the *i*-th parameter declaration shall specify `Interfaces.SQL.BIT`. If *P* is an actual parameter associated with the *i*-th parameter in a call to the encompassing procedure, then P'LENGTH shall be equal to *L*.

ii) BIT VARYING(*L*) for some length *L*, then the subtype mark in the *i*-th parameter declaration shall specify `Interfaces.SQL.Varying.BIT`. If *P* is an actual parameter associated with the *i*-th parameter in a call to the encompassing procedure, then *P* shall be sufficient to hold a bit string of length *L*.

h) If the *i*-th <parameter declaration> specifies a <data type> that is NUMERIC(*P*,*S*) for some <precision> and <scale> *P* and *S*, then the Ada library unit package generated for the encompassing module shall contain a declaration equivalent to

```
subtype Numeric_p_s is Interfaces.SQL.Numerics.Scale_s
  digits p;
```

The subtype mark in the *i*-th parameter specification shall specify this subtype.

i) If the *i*-th <parameter declaration> specifies a <data type> that is SMALLINT, then the subtype mark in the *i*-th parameter declaration shall specify `Interfaces.SQL.SMALLINT`.

j) If the *i*-th <parameter declaration> specifies a <data type> that is INTEGER, then the subtype mark in the *i*-th parameter declaration shall specify `Interfaces.SQL.INT`.

k) If the *i*-th <parameter declaration> specifies a <data type> that is REAL, then the subtype mark in the *i*-th parameter declaration shall specify `Interfaces.SQL.REAL`.

l) If the *i*-th <parameter declaration> specifies a <data type> that is DOUBLE_PRECISION, then the subtype mark in the *i*-th parameter declaration shall specify `Interfaces.SQL.DOUBLE_PRECISION`.

m) For every parameter,

Case:

i) If the parameter is an input parameter but not an output parameter, then the Ada parameter mode is **in**.

ii) If the parameter is an output parameter but not an input parameter, then the Ada parameter mode is **out**.

iii) If the parameter is both an input parameter and an output parameter, then the Ada parameter mode is **in out**.

iv) Otherwise, the Ada parameter mode is **in**, **out**, or **in out**.

n) The following Ada library unit renaming declaration exists:

```
with Interfaces.SQL;
package SQL_Standard renames Interfaces.SQL.
```

2. *Rationale: When CHARACTER or CHARACTER VARYING data is exchanged between host programs written in the C language and an SQL database system, the C convention of terminating strings with a "null character" is used on the C side of the interface. The binding rules incorrectly assume that the "null character" is always equivalent to a single "char" in C (i.e., a single byte). However, so-called wide character sets like Unicode sometimes encode characters with one of the two bytes equal to 0; this requires that the C convention be modified so that a number of bytes equivalent to the size of the "widest" character*

all be 0. That, in turn, requires that the SQL/C binding provide sufficient storage for those additional bytes.

Replace Syntax Rule 2) c) with:

c) If the i-th <parameter declaration> specifies a <data type> *PDT* that is CHARACTER(*L*) or CHARACTER VARYING(*L*) for some <length> *L*, then the type of the *i*-th parameter *P* shall be C char with length *k* greater than the maximum possible length in octets of *PDT*, where *k* is the size in octets of the largest character in the character set of *PDT*.

3. Rationale: Editorial.

In Syntax Rule 2) d), replace "(*L*/(*B*+1))" with "(*L*/*B*) plus one".

4. Rationale: Fix an incorrect PL/I specification.

In Syntax Rule 7) f), delete the word "REAL".

In Syntax Rule 7) g), delete the word "REAL".

5. Rationale: The following rule defines a number of the symbols that are used later in the Subclause. Its deletion (as 12.3 General Rule 8)) in Technical Corrigendum 1 without replacement was in error.

Insert the following General Rule:

0.1) When a <procedure> is called by an SQL-agent, let PD_i be the <parameter declaration> of the *i*-th parameter and let DT_i and PN_i be the <data type> and the <parameter name> specified in PD_i, respectively. Let PI_i be the *i*-th parameter in the procedure call.

6. Rationale: A <select statement: single row> or a <fetch statement> may return a null value. General Rule 1) of 9.1 "Retrieval assignment" assigns a value to an indicator parameter, but not its corresponding data parameter. Ada requires that all output parameters be assigned values by a subprogram.

Replace General Rule 1) with:

1) If the subject <language clause> specifies ADA, then:

a) Where P_i is used as an input parameter whose value is evaluated, a reference to PN_i in a <general value specification> has the value PI_i.

b) Where P_i is used as an output parameter, a reference to PN_i that assigns a value SV_i to PN_i implicitly assigns the value SV_i to PI_i.

c) If P_i is used as an output parameter and no value has been assigned to PL_i, then an implementation-dependent value is assigned PI_i.

7. *Rationale: When CHARACTER or CHARACTER VARYING data is exchanged between host programs written in the C language and an SQL database system, the C convention of terminating strings with a "null character" is used on the C side of the interface. The binding rules incorrectly assume that the "null character" is always equivalent to a single "char" in C (i.e., a single byte). However, so-called wide character sets like Unicode sometimes encode characters with one of the two bytes equal to 0; this requires that the C convention be modified so that a number of bytes equivalent to the size of the "widest" character all be 0. That, in turn, requires that the SQL/C binding provide sufficient storage for those additional bytes.*

Replace the first sentence of General Rule 2) b) ii) with:

ii) Let CL_i be k greater than the maximum possible length in octets of PN_i, where k is the size in octets of the largest character in the character set of DT_i.

Add the following note at the end of General Rule 2) b):

Note: In the preceding Rule, the phrase "implementation-defined null character that terminates a C character string" implies one or more octets all of whose bits are zero and whose number is equal to the number of octets in the largest character of the character set of DT_i.

12.5 <SQL procedure statement>

1. *Rationale: Editorial.*

In Leveling Rule 2) a) replace "<SQL schema definition statement>" with "<SQL schema statement>".

13.1 <declare cursor>

1. *Rationale: Editorial.*

Replace Syntax Rule 8) with:

8) The *simply underlying table* of the <cursor specification> is *T*.

2. *Rationale: There is a contradiction between Syntax Rules 13) and 14) of Subclause 13.1 and Syntax Rule 8) and Access Rule 2) of Subclause 13.9.*

Replace Syntax Rule 13) with:

13) If an <updatability clause> of FOR UPDATE without a <column name list> is specified or implicit, then a <column name list> that includes the <column name> of every column of the simply underlying table of the simply underlying table of *T* is implicit.

Replace Syntax Rule 14) with:

14) If an <updatability clause> of FOR UPDATE with a <column name list> is specified, then each <column name> in the <column name list> shall be the <column name> of a column of the simply underlying table of the simply underlying table of *T*.

Delete Syntax Rule 9) and insert it as General Rule 0.1).

13.2 <open statement>

1. *Rationale: CR was used ambiguously to reference the syntactic construct <declare cursor> and the cursor itself. This change disambiguates the references. The change also accommodates the use of the rules of this Subclause for dynamic cursors.*

Replace Syntax Rule 1) with:

1) The containing <module> shall contain a <declare cursor> *DC* whose <cursor name> is the same as the <cursor name> in the <open statement>. Let *CR* be the cursor specified by *DC*. *CR* is *associated* with the <cursor specification> contained in *DC*.

Replace General Rule 2) with:

2) Let *S* be the <cursor specification> associated with *CR*.

13.3 <fetch statement>

1. *Rationale: CR was used ambiguously to reference the syntactic construct <declare cursor> and the cursor itself. This change disambiguates the references.*

Replace Syntax Rule 2) with:

2) The containing <module> shall contain a <declare cursor> *DC* whose <cursor name> is the same as the <cursor name> in the <fetch statement>. Let *T* be the table defined by the <cursor specification> of *DC*. Let *CR* be the cursor specified by *DC*.

Replace Syntax Rule 3) with:

3) If the implicit or explicit <fetch orientation> is not NEXT, then the <declare cursor> *DC* shall specify SCROLL.

2. *Rationale: Editorial. The statement of General Rule 3) c) iii) is unnecessarily complicated. The introduction of the concept of "the next row of a row" is unnecessary.*

Replace General Rule 3) c) iii) with:

iii) If the position of *CR* is before a row *R* that is not the first row of *T*, then let T_t contain all rows of *T* ordered before row *R*, preserving their order in *T*.

3. *Rationale: Remove an imprecise rule that is adequately covered by the penultimate paragraph of Subclause 4.21 Cursors.*

Delete General Rule 7).

4. Rationale: Resolve inconsistency regarding "no data" and exception condition and add explanatory note.

Replace General Rule 9) with:

9) If an exception condition occurs during the assignment of a value to a target, then the values of all targets are implementation-dependent and *CR* remains positioned on the current row.

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

13.4 <close statement>

1. Rationale: CR was used ambiguously to reference the syntactic construct <declare cursor> and the cursor itself. This change disambiguates the references.

Replace Syntax Rule 1) with:

1) The containing <module> shall contain a <declare cursor> *DC* whose <cursor name> is the same as the <cursor name> in the <close statement>. Let *CR* be the cursor specified by *DC*.

Replace General Rule 2) with:

2) Cursor *CR* is placed in the closed state and the copy of the <cursor specification> of *DC* is destroyed.

13.5 <select statement: single row>

1. Rationale: Resolve inconsistency regarding "no data" and exception condition.

Delete General Rule 5).

Add the following General Rule:

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

13.6 <delete statement: positioned>

1. Rationale: Clarify that references to non-existing objects is only allowed within the same <schema definition>.

Add the following Syntax Rule:

3.1) The schema identified by the explicit or implicit qualifier of the <table name> shall include the descriptor of *T*.

2. Rationale: The General Rules that specify the completion condition 01001 (cursor operation conflict) pertain only to SQL3 and were inadvertently included in ISO/IEC 9075:1992. These rules need therefore to be deleted.

Delete General Rule 4).

13.7 <delete statement: searched>

1. *Rationale: Clarify that references to non-existing objects is only allowed within the same <schema definition>.*

Add the following Syntax Rule:

2.1) The schema identified by the explicit or implicit qualifier of the <table name> shall include the descriptor of *T*.

2. *Rationale: The General Rules that specify the completion condition 01001 (cursor operation conflict) pertain only to SQL3 and were inadvertently included in ISO/IEC 9075:1992. These rules need therefore to be deleted.*

Delete General Rule 3).

13.8 <insert statement>

1. *Rationale: Clarify that references to non-existing objects is only allowed within the same <schema definition>.*

Add the following Syntax Rule:

6.1) The schema identified by the explicit or implicit qualifier of the <table name> shall include the descriptor of *T*.

2. *Rationale: Editorial.*

In Access Rule 1) a), replace "<privileges>" with "privileges".

In Leveling Rule 2) a), replace "<insert statement>" with "<insert columns and source>".

3. *Rationale: Editorial.*

In Leveling Rule 2) a), replace "<value specification>" with "<value specification> or a <null specification>".

13.9 <update statement: positioned>

1. *Rationale: Clarify that references to non-existing objects is only allowed within the same <schema definition>.*

Add the following Syntax Rule:

10.1) The schema identified by the explicit or implicit qualifier of the <table name> shall include the descriptor of *T*.

2. *Rationale: There is a contradiction between Syntax Rules 13) and 14) of Subclause 13.1 and Syntax Rule 8) and Access Rule 2) of Subclause 13.9.*

Delete Access Rule 2) and insert it as Syntax Rule 7.1).

3. *Rationale: The General Rules that specify the completion condition 01001 (cursor operation conflict) pertain only to SQL3 and were inadvertently included in ISO/IEC 9075:1992. These rules need therefore to be deleted.*

Delete General Rule 4).

4. *Rationale: The following change is needed for the same reason that the changes to Subclause 11.5, "<default clause>" were necessary.*

Replace General Rule 5) with:

5) The value associated with DEFAULT is the default value for the <object column> in the containing <set clause>, as indicated in the General Rules of Subclause 11.5, "<default clause>".

5. *Rationale: Correct accidental omission of a rule approved for inclusion in ISO/IEC 9075:1992.*

Append the following text to General Rule 9):

If an exception condition occurs during the assignment of *SV* to *C*, then *CR* remains positioned on its current row.

13.10 <update statement: searched>

1. *Rationale: Clarify that references to non-existing objects is only allowed within the same <schema definition>.*

Add the following Syntax Rule:

5.1) The schema identified by the explicit or implicit qualifier of the <table name> shall include the descriptor of *T*.

2. *Rationale: The General Rules that specify the completion condition 01001 (cursor operation conflict) pertain only to SQL3 and were inadvertently included in ISO/IEC 9075:1992. These rules need therefore to be deleted.*

Delete General Rule 3).

13.11 <temporary table declaration>

1. *Rationale: Editorial. Syntax Rule 1) defines T as the name of a table. Syntax Rule 3) refers to the "name of T", effectively to the name of a name.*

In Syntax Rule 3) Replace "includes the name of T" with "includes T".

14.1 <set transaction statement>

1. *Rationale: Clarify the scope of the <set transaction statement>.*

Add the following sentence to the end of the paragraph under Function:

Note: This statement has no effect on any SQL-transaction subsequent to the next SQL-transaction.

14.2 <set constraints mode statement>

1. *Rationale: Clarify the scope of the <set transaction statement>.*

Add the following sentence to the end of the paragraph under Function:

This statement has no effect on any SQL-transaction subsequent to this SQL-transaction.

14.3 <commit statement>

1. *Rationale: Clarification.*

In General Rule 4), replace "were executed" with "were executed for each active SQL-connection".

15.1 <connect statement>

1. *Rationale: Editorial.*

In General Rule 9) a), replace "SQL-server" with "SQL-session".

2. *Rationale: Editorial.*

Replace General Rule 2) with:

2) If <user name> is specified, then let *S* be the character string that is the value of <user name> and let *V* be the character string that is the value of

TRIM (BOTH ' ' FROM *S*)

15.2 <set connection statement>

1. *Rationale: Editorial.*

In General Rule 4), replace "SQL-server" with "SQL-session".

15.3 <disconnect statement>

1. *Rationale: Editorial.*

In General Rule 5), replace "SQL-connection, if any" with "SQL-connection".

16.5 <set local time zone statement>

1. *Rationale: Editorial.*

In Leveling Rule 1), replace "SQL;" with "SQL:".

In Leveling Rule 2), replace "SQL;" with "SQL, in addition to any Intermediate SQL restrictions:".

17.1 Description of SQL item descriptor areas

1. *Rationale: Editorial.*

In Syntax Rule 3) e), change "date" to "data".

2. *Rationale: Syntax Rule 3) applies only to data types that are not implementation-defined.*

Add the following to Syntax Rule 2):

l) TYPE indicates an implementation-defined data type and *T* satisfies the implementation-defined rules for matching that data type.

Add the following to Syntax Rule 3):

i) TYPE indicates an implementation-defined data type.

17.2 <allocate descriptor statement>

1. *Rationale: Editorial.*

In General Rule 2), replace "<scope clause>" with "<scope option>".

17.3 <deallocate descriptor statement>

1. *Rationale: Editorial.*

In General Rule 1), replace "<scope clause>" with "<scope option>".

2. *Rationale: Editorial.*

In Leveling Rule 2) a), replace "shall contain no" with "shall not contain any".

17.4 <get descriptor statement>

1. *Rationale: Clarification of wording.*

In General Rule 7) b) replace "datetime data type, interval qualifier" with "datetime data type or interval qualifier as appropriate, interval leading field precision" and replace "character set of" with "character set, respectively, of".

2. *Rationale: The behavior of <get descriptor statement> should be consistent with the behavior of <using clause>.*

Replace the last sentence of General Rule 7) c) with:

For a <dynamic parameter specification>, the value of UNNAMED is 1 and the value of NAME is implementation-dependent.

17.5 <set descriptor statement>

1. *Rationale: Disallow the use of literals to specify a value for the DATA field of the SQL descriptor due to the serious problems that arise if one attempts to enforce the existing match rules against literals.*

Add the following Syntax Rule:

3.1) If the <descriptor item name> is DATA, then <simple value specification 2> shall not be a <literal>.

17.6 <prepare statement>

1. *Rationale: Introduce missing <scope option> to <preparable dynamic delete statement: positioned>.*

Replace General Rule 2) with:

2) If *P* is a <preparable dynamic delete statement: positioned> or a <preparable dynamic update statement: positioned>, let *CN* be the <cursor name> contained in *P*.

Case:

a) If *P* contains a <scope option> whose value is GLOBAL, then if there exists an extended dynamic cursor with an <extended cursor name> having a global scope and a <cursor name> that is *CN*, then that is the cursor referenced by *P*, otherwise an exception condition is raised: *invalid cursor name*.

b) If *P* contains a <scope option> whose value is LOCAL, or if no <scope option> is specified, then the *potentially referenced cursors* of *P* include:

i) every declared dynamic cursor whose <cursor name> is *CN* with a scope of the containing module

ii) every extended dynamic cursor with an <extended cursor name> having a scope of the containing module and a <cursor name> that is *CN*

Case:

i) If the number of potentially referenced cursors is greater than 1, then an exception condition is raised: *ambiguous cursor name*

ii) If the number of potentially referenced cursors is less than 1, then an exception condition is raised: *invalid cursor name*

iii) Otherwise, *CN* refers to the only potentially referenced cursor of *P*.

2. *Rationale: The following unifies the SQLSTATE returned for the different ways of invoking an SQL statement.*

In General Rule 4), replace "*syntax error or access rule violation in dynamic SQL statement*" with "*syntax error or access rule violation*".

3. *Rationale: Clarify that the operands referred to collectively as E1 may have different syntactic forms.*

Replace General Rule 4) c) with:

c) *P* contains *E1* and *E2* as operands of a single dyadic operator.

4. *Rationale: Clarify the use of <dynamic parameter specification> in COALESCE.*

Replace General Rule 4) f) with:

f) *P* contains a <null predicate> whose <row value constructor> simply contains *E1* unless *E1* is directly contained in a <value expression> that is an operand of COALESCE.

5. *Rationale: Editorial.*

In General Rule 4) g), replace "where *E1* is the" with "where the" and replace "of the <overlaps predicate>" with "of the <overlaps predicate> is *E1*".

6. *Rationale: Clarify the use of a <dynamic parameter specification> in COALESCE and in a <case expression>.*

Replace General Rule 4) h) with:

h) *P* contains a <simple case> where <case operand> and each <when operand> meets the criteria for *E1*, or *P* contains a <case abbreviation> where each operand meets the criteria for *E1*.

7. *Rationale: Clarify the use of a <dynamic parameter specification> in a <between predicate>.*

Replace General Rule 4) i) with:

i) *P* contains a <comparison predicate> or <between predicate> where, for some *i*, the *i*-th <value expression> in all of the immediately contained <row value constructor>s meets the criteria for *E1*.

8. *Rationale: Clarify that the operands referred to collectively as E1 may have different syntactic forms.*

Replace General Rule 4) j) with:

j) *P* contains a <table value constructor> in which the i-th <value expression> in each <row value constructor> meets the criteria for *E1* and either:

   i) *P* is not an <insert statement>, or

   ii) *P* is an insert statement> and the <table value expression> is not the <query expression> simply contained in the <insert statement>.

9. *Rationale: Clarify the use of <dynamic parameter specification>s in <in value list>.*

Replace General Rule 4) k) with:

k) *P* contains an <in predicate> with an <in value list> in which the <row value constructor> and each <value expression> of the <in value list> meet the criteria for *E1*.

10. *Rationale: Define the use of <dynamic parameter specification>s in <result expression>s of <case expression>s.*

Add the following to General Rule 4):

r) *P* contains a <case expression> where at least one <result> is *E1* and each other <result> is either NULL or meets the criteria for *E1*.

11. Rationale: Clarify the use of <dynamic parameter specification> in COALESCE.

Replace General Rule 5) j) with:

j) If one or more operands of COALESCE are *E1*, then the data type of *E1* is the data type determined by applying Subclause 9.3, "Set operation result data types", to the operands that do not meet the criteria for *E1*.

12. Rationale: Clarify the use of a <dynamic parameter specification> in a <case expression>.

Replace General Rule 5) k) with:

k) If one or more of the operands of the <case operand> and the <when operand>s in a <case specification> are *E1*, then the data type of *E1* is the data type determined by applying Subclause 9.3, "Set operation result data types", to the <case operand> and all the <when operand>s that do not meet the criteria for *E1*.

13. Rationale: Clarify the use of a <dynamic parameter specification> in a <between predicate>.

Replace General Rule 5) m) with:

m) In the first and second operands of a <comparison predicate>, if the *i*-th value of one operand is *E1*, then the data type of *E1* is the data type of the *i*-th value of the other operand.

14. Rationale: Define the type of <dynamic parameter specification>s in <table value constructor>s.

Replace General Rule 5) o) with:

o) In a <table value constructor> that is not a <query expression> simply contained in an <insert statement>, in which the i-th <value expression> of some <row value constructor> is *E1*, the data type determined by applying Subclause 9.3, "Set operation result data types", to the i-th <value expression>s of the other <row value constructor>s that are not *E1*.

In General Rule 5) p), replace "each" with "some".

15. Rationale: Clarify the use of <dynamic parameter specification>s in <in value list>.

Replace General Rule 5) r) with:

r) In an <in predicate> that specifies an <in value list>, if the <row value constructor> is not *E1*, then let *D* be its date type. Otherwise, let *D* be the data type determined by applying Subclause 9.3, "Set operation result data types", to the <value specification>s of the <in value list> that are not *E1*. The data type of any *E1* in the <in predicate> is assumed to be *D*.

16. Rationale: Define the use of <dynamic parameter specification>s in <result expression>s of <case specification>s.

Add the following to General Rule 5):

w.1) If any <result expression> in a <case specification> is *E1*, then the data type of *E1* is the data type that is determined by applying Subclause 9.3, "Set operation result data types", to all the <result expression>s that are not *E1*.

17. Rationale: Clarify the use of a <dynamic parameter specification> in a <between predicate>.

Add the following to General Rule 5):

w.2) In an operand of a <between predicate>, if the *i*-th value of one operand is *E1*, then the data type of *E1* is the data type determined by applying Subclause 9.3, "Set operation result data types" using the *i*-th value of the other operands.

18. Rationale: Editorial.

In General Rule 8), replace "<simple target specification>" with "<simple value specification>".

19. Rationale: The following corrects an incorrectly structured General Rule.

Replace General Rule 9) b) with:

b) If <statement name> is specified for the <SQL statement name>, then

Case:

i) If *P* is not a <cursor specification> and <statement name> is associated with a cursor *C* through a <dynamic declare cursor>, then an exception is raised: *dynamic SQL error — prepared statement is not a cursor specification*.

ii) Otherwise:

1) If <statement name> is not associated with a cursor and either *P* is not a <cursor specification> or *P* is a <cursor specification> that conforms to the Format and Syntax Rules of a <dynamic single row select statement>, then the same <statement name> shall be specified for each <execute statement> that is to be associated with this prepared statement.

2) If *P* is a <cursor specification> and <statement name> is associated with a cursor *C* through a <dynamic declare cursor>, then an association is made between *C* and *P*. The association is preserved until the prepared statement is destroyed.

17.9 <using clause>

1. Rationale: Set the LENGTH in the descriptor area for <interval type>s.

In General Rule 3) e) iv) 6), insert "LENGTH is set to the length in positions of the interval type, " before "DATETIME_INTERVAL_CODE".

17.10 <execute statement>

1. Rationale: Rule improperly fails to address <parameter using clause> option.

In General Rule 6), delete the phrase “that is <using descriptor>”.

17.11 <execute immediate statement>

1. Rationale: Introduce missing <scope option> to <preparable dynamic delete statement: positioned>.

Replace General Rule 2) with:

2) If *P* is a <preparable dynamic delete statement: positioned> or a <preparable dynamic update statement: positioned>, let *CN* be the <cursor name> contained in *P*.

Case:

a) If *P* contains a <scope option> whose value is GLOBAL, then if there exists an extended dynamic cursor with an <extended cursor name> having a global scope and a <cursor name> that is *CN*, then that is the cursor referenced by *P*, otherwise an exception condition is raised: *invalid cursor name*.

b) If *P* contains a <scope option> whose value is LOCAL, or if no <scope option> is specified, then the *potentially referenced cursors* of *P* include:

i) every declared dynamic cursor whose <cursor name> is *CN* with a scope of the containing module

ii) every extended dynamic cursor with an <extended cursor name> having a scope of the containing module and a <cursor name> that is *CN*

Case:

i) If the number of potentially referenced cursors is greater than 1, then an exception condition is raised: *ambiguous cursor name*.

ii) If the number of potentially referenced cursors is less than 1, then an exception condition is raised: *invalid cursor name*.

iii) Otherwise, *CN* refers to the only potentially referenced cursor of *P*.

2. Rationale: The following unifies the SQLSTATE returned for the different ways of invoking an SQL statement.

In General Rule 3), replace "*syntax error or access rule violation in dynamic SQL statement*" with "*syntax error or access rule violation*".

17.15 <dynamic fetch statement>

1. Rationale: Editorial.

Replace Leveling Rule 1) with:

1) The following restrictions apply for Intermediate SQL:

None.

17.18 <dynamic update statement: positioned>

1. *Rationale: Editorial.*

In Format, replace "<set clause> [{ <comma> <set clause> } ...]" with "<set clause list>".

Delete Syntax Rules 7) and 8).

In Syntax Rule 10) replace "<update statement: positioned>" with "<dynamic update statement: positioned>".

Delete Syntax Rule 11).

17.19 <preparable dynamic delete statement: positioned>

1. *Rationale: Introduce missing <scope option> to <preparable dynamic delete statement: positioned>.*

Replace the Format with:

```
<preparable dynamic delete statement: positioned> ::=
        DELETE [ FROM <table name> ]
            WHERE CURRENT OF [ <scope option> ] <cursor name>
```

17.20 <preparable dynamic update statement: positioned>

1. *Rationale: Introduce missing <scope option> to <preparable dynamic delete statement: positioned>.*

Replace the Format with:

```
<preparable dynamic update statement: positioned> ::=
        UPDATE [ <table name> ]
            SET <set clause list>
            WHERE CURRENT OF [ <scope option> ] <cursor name>
```

18.1 <get diagnostics statement>

1. *Rationale: DYNAMIC_FUNCTION failed to provide information about errors discovered during execution of a <prepare statement> about the statement being prepared. The values of COMMAND_FUNCTION and DYNAMIC_FUNCTION fail to account for implementation-defined statements and statements that are completely unrecognized by the implementation.*

In General Rule 1) d), in the first sentence, replace "the prepared statement executed" with "the statement being prepared or executed dynamically".

In Table 22, "SQL-statement character codes for use in the diagnostics area", add the following two entries at the end of the table:

```
| Implementation-defined statements | An implementation-defined character     |
|                                   | string value different from the value   |
|                                   | associated with any other SQL-statement |

| Unrecognized statements           | A zero-length string                    |
```

2. *Rationale: The General Rules that specify the completion condition 01001 (cursor operation conflict) pertain only to SQL3 and were inadvertently included in ISO/IEC 9075:1992. These rules need therefore to be deleted.*

Delete General Rule 3) e).

3. *Rationale: CATALOG_NAME, SCHEMA_NAME, and TABLE_NAME do not always have applicable values in GET DIAGNOSTICS for a syntax error or access rule violation.*

In General Rule 3) f) ii), in the last sentence, replace "<space>s" with "a zero-length string".

In General Rule 3) f) ii) 3), replace "<space>s" with "zero-length strings".

4. *Rationale: The following unifies the SQLSTATE returned for the different ways of invoking an SQL statement.*

Replace the first paragraph of General Rule 3) g) with:

g) If the value of RETURNED_SQLSTATE corresponds to *syntax error or access rule violation*, then:

5. *Rationale: CATALOG_NAME, SCHEMA_NAME, and TABLE_NAME do not always have applicable values in GET DIAGNOSTICS for a syntax error or access rule violation. The rule is also restructured to remove an internal ambiguity.*

Replace General Rule 3) g) i) with:

i) Case:

1) If the syntax error or access rule violation was caused by reference to a specific table, then the values of CATALOG_NAME, SCHEMA_NAME, and TABLE_NAME are:

Case:

i) If the specific table referred to was not a declared local temporary table, then <catalog name>, the <unqualified schema name> of the <schema name> of the schema that contains the table that caused the syntax error or access rule violation, and the <qualified identifier>, respectively.

ii) Otherwise, a zero length string, "MODULE" and the <local table name>, respectively.

2) Otherwise, CATALOG_NAME, SCHEMA_NAME and TABLE_NAME contain a zero-length string.

In General Rule 3) g) ii), replace "<space>s" with "a zero-length string".

6. *Rationale: The following unifies the SQLSTATE returned for the different ways of invoking an SQL statement.*

Replace the first paragraph of General Rule 3) j) with:

j) If the value of RETURNED_SQLSTATE does not correspond to *syntax error or access rule violation*, then:

7. *Rationale: In addition to <SQL connection statement>s, both <commit statement> and <rollback statement> can affect multiple servers and potentially give rise to condition information for more than one server.*

Replace General Rule 3) n) with:

n) The values of CONNECTION_NAME and SERVER_NAME are respectively:

Case:

i) If COMMAND_FUNCTION or DYNAMIC_FUNCTION identifies an <SQL-connection statement>, then the <connection name> and <SQL-server name> specified 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.

Note: If COMMAND_FUNCTION or DYNAMIC_FUNCTION identifies an SQL-statement that can affect more than one SQL-session (e.g., <commit statement>, <rollback statement>, DISCONNECT ALL), then there may be one or more conditions for each of the SQL-servers involved.

8. *Rationale: Editorial.*

In Leveling Rule 1), replace "SQL;" with "SQL:".

In Leveling Rule 2), replace "SQL;" with "SQL in addition to any Intermediate SQL restrictions:".

19.1 <embedded SQL host program>

1. *Rationale: Prohibit multiple <embedded character set declaration>s, the effects of which are undefined.*

Replace Syntax Rule 6) with:

6) An <embedded SQL host program> shall contain no more than one <embedded character set declaration>. If an <embedded character set declaration> is not specified, then an <embedded character set declaration> that specifies an implementation-defined character set that contains at least every character that is in <SQL language character> is implicit.

19.3 <embedded SQL Ada program>

1. *Rationale: Editorial*

In the Format Rule for "<Ada variable definition>, replace the colon character with "<colon>".

19.5 <embedded SQL COBOL program>

1. *Rationale: Incorrect use of PICTURE B in COBOL syntax.*

In Format, replace the BNF for <COBOL bit type> with:

```
<COBOL bit type> ::=
    { PIC | PICTURE } [ IS ]
    { X [ <left paren> <length> <right paren> ] } ...
    USAGE IS BIT
```

Replace Syntax Rule 5) b) with:

b) The syntax "USAGE IS BIT" shall be deleted in any <COBOL bit type>.

20.1 <direct SQL statement>

1. *Rationale: The following unifies the SQLSTATE returned for the different ways of invoking an SQL statement.*

In General Rule 5), replace "*syntax error or access rule violation in direct SQL statement*" with "*syntax error or access rule violation* ".

2. *Rationale: Clarify with respect to the language added to Subclause 4.18.1; to permit implementation-defined 01 warnings to be returned in Direct SQL; to correct typos.*

Replace General Rule 8) with:

8) Case:

a) If *S* executed successfully, then either a completion condition is raised: *successful completion*, or a completion condition is raised: *warning*, or a completion condition is raised: *no data*.

b) If *S* did not execute successfully, then all changes made to SQL-data or schemas by the execution of *S* are cancelled and an exception condition is raised.

Note: The method of raising a condition is implementation-defined.

21.1 Introduction

1. *Rationale: Clarification of the extensibility of the tables in the Information Schema base tables and views.*

Replace the fourth paragraph with:

An implementation may define objects that are associated with INFORMATION_SCHEMA that are not defined in theis Clause. An implementation, or any future version of this International Standard, may also add columns to tables that are defined in this Clause.

2. *Rationale: Clarification of representation of <identifier>s in the Information Schema base tables and views.*

Replace the last paragraph with:

The representation of an <identifier> in the base tables and views of the Information Schema is by a character string corresponding to its <identifier body> (in the case of a <regular identifier>) or its <delimited identifier body> (in the case of a <delimited identifier>). Within this character string, any lower case letter appearing in a <regular identifier> is replaced by the corresponding upper case letter, and any <doublequote symbol> appearing in a <delimited identifier body> is replaced by a <double quote>. Where an <actual identifier> has multiple forms that are equal according to the rules of Subclause 8.2, "<comparison predicate>", the form stored is that encountered at definition time.

21.2.2 INFORMATION_SCHEMA_CATALOG_NAME base table

1. *Rationale: This base table is supposed to contain a single row; however, its primary key does not properly reflect this constraint.*

Replace Definition with:

```
CREATE TABLE INFORMATION_SCHEMA_CATALOG_NAME
     ( CATALOG_NAME SQL_IDENTIFIER,
     CONSTRAINT INFORMATION_SCHEMA_CATALOG_NAME_PRIMARY_KEY
      PRIMARY KEY ( CATALOG_NAME ),
     CONSTRAINT INFORMATION_SCHEMA_CATALOG_NAME_CHECK
      CHECK ( ( SELECT COUNT (*) FROM INFORMATION_SCHEMA_CATALOG_NAME ) = 1 ) )
```

2. *Rationale: Editorial.*

Replace the second occurrence of "Definition" with "Description".

21.2.3 INFORMATION_SCHEMA_CATALOG_NAME_CARDINALITY assertion

1. *Rationale: Editorial.*

Delete the Description section.

Add the following at the end of the Subclause:

Leveling Rules

1) The following restrictions apply for Intermediate SQL:

None.

2) The following restrictions apply for Entry SQL in addition to any Intermediate SQL restrictions:

a) Conforming Entry SQL language shall not reference the Information Schema.

21.2.4 SCHEMATA view

1. *Rationale: Editorial. The Function suggests that the view contains rows for all schemata owned by a specified user. However, the view definition restricts the rows to those that describe schemata in the current catalog that are owned by the current user.*

Replace the Function with:

Identify the schemata in the catalog that are owned by the current user.

21.2.5 DOMAINS view

1. *Rationale: In support of changes to 21.3.5.*

In the Definition, replace "`DATETIME_PRECISION, DOMAIN_DEFAULT`" with "`DATETIME_PRECISION, INTERVAL_TYPE, INTERVAL_PRECISION, DOMAIN_DEFAULT`".

21.2.6 DOMAIN_CONSTRAINTS view

1. *Rationale: Editorial.*

In Definition, replace "`( S.CATALOG_NAME, SCHEMA_NAME S)`" with "`( S.CATALOG_NAME, S.SCHEMA_NAME )`".

21.2.9 COLUMNS view

1. *Rationale: In support of changes to 21.3.5.*

In the Definition, insert the following lines immediately after the line that reads "`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,
```

21.2.17 ASSERTIONS view

1. *Rationale: The ASSERTIONS view should appear in Full SQL along with <assertion definition>.*

Replace Leveling Rules 1) and 2) with:

1) The following restrictions apply for Intermediate SQL:

   a) Conforming Intermediate SQL language shall not reference the ASSERTIONS view.

2) The following restrictions apply for Entry SQL in addition to any Intermediate SQL restrictions:

   None.

21.2.23 CONSTRAINT_TABLE_USAGE view

1. *Rationale: CONSTRAINT_TABLE_USAGE is not returning unique constraints as indicated in its Function.*

Replace the Definition with:

```
CREATE VIEW CONSTRAINT_TABLE_USAGE
        AS
          SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
                 CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME FROM
        (
          ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
                   CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
              FROM DEFINITION_SCHEMA.CHECK_TABLE_USAGE )
        UNION
```

```
        ( SELECT PK.TABLE_CATALOG, PK.TABLE_SCHEMA, PK.TABLE_NAME,
                 FK.CONSTRAINT_CATALOG, FK.CONSTRAINT_SCHEMA,
                 FK.CONSTRAINT_NAME
            FROM
            DEFINITION_SCHEMA.REFERENTIAL_CONSTRAINTS AS FK
            JOIN
            DEFINITION_SCHEMA.TABLE_CONSTRAINTS 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,
                 CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME
            FROM DEFINITION_SCHEMA.TABLE_CONSTRAINTS
               WHERE CONSTRAINT_TYPE IN ( 'UNIQUE', 'PRIMARY KEY' )
        )
      )
      JOIN
        DEFINITION_SCHEMA.SCHEMATA S
           ON
            ( ( TABLE_CATALOG, TABLE_SCHEMA )
            = ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
        WHERE S.SCHEMA_OWNER = CURRENT_USER
          AND CONSTRAINT_CATALOG
            = ( SELECT CATALOG_NAME FROM INFORMATION_SCHEMA_CATALOG_NAME )
```

21.2.24 CONSTRAINT_COLUMN_USAGE view

1. *Rationale: CONSTRAINT_COLUMN_USAGE is not returning referential constraints as indicated in its Function.*

Replace the Definition with:

```
CREATE VIEW CONSTRAINT_COLUMN_USAGE
     AS
       SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME,
              CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME 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 ) )
    WHERE SCHEMA_OWNER = CURRENT_USER
    AND CONSTRAINT_CATALOG
      = ( SELECT CATALOG_NAME FROM INFORMATION_SCHEMA_CATALOG_NAME )
```

21.2.27 SQL_IDENTIFIER domain

1. *Rationale: Clarification of the representation of <identifier>s in the Information Schema base tables and views.*

Replace the Function with:

Define a domain that contains all valid <identifier body>s and <delimited identifier body>s.

Replace Description 1) with:

1) This domain specifies all variable-length character values that conform to the rules for formation and representation of an <identifier body> or an <delimited identifier body>.

Note: There is no way in SQL to specify a <domain constraint> that would be true for the body of any valid <regular identifier> or <delimited identifier> and false for all other character string values.

Replace Description 2) with:

2) *L* is the implementation-defined maximum length of <identifier body> and <delimited identifier body>.

21.3.5 DATA_TYPE_DESCRIPTOR base table

1. *Rationale: The descriptions of the existing columns do not allow for the fact that the interval type has two precisions: the fractional seconds precision and the interval leading field precision. In addition, the interval type also specifies the interval qualifier which must be encoded into the DATA_TYPE_DESCRIPTOR. Finally, numeric precision should not be allowed to be null for INTEGER, SMALLINT, NUMERIC, DECIMAL, DATE, and INTERVAL data types.*
   Also because DECIMAL and NUMERIC always have a radix of 10, even when the scale is 0 and to permit implementation-defined data types to be represented in the Information Schema changes are made to the TABLE_OR_DOMAIN_CHECK_COMBINATIONS.
   Also correct TABLE_OR_DOMAIN_CHECK_COMBINATIONS constraints.

Replace Definition with:

```
CREATE TABLE DATA_TYPE_DESCRIPTOR
  (
  TABLE_OR_DOMAIN_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_OR_DOMAIN_SCHEMA  INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_OR_DOMAIN_NAME                INFORMATION_SCHEMA.SQL_IDENTIFIER,
  COLUMN_NAME             INFORMATION_SCHEMA.SQL_IDENTIFIER,
  DATA_TYPE               INFORMATION_SCHEMA.SQL_IDENTIFIER
    CONSTRAINT TABLE_OR_DOMAIN_DATA_TYPE_NOT_NULL NOT NULL,
  CHARACTER_MAXIMUM_LENGTH            INFORMATION_SCHEMA.CARDINAL_NUMBER,
  CHARACTER_OCTET_LENGTH  INFORMATION_SCHEMA.CARDINAL_NUMBER,
```

```
  COLLATION_CATALOG       INFORMATION_SCHEMA.SQL_IDENTIFIER,
  COLLATION_SCHEMA        INFORMATION_SCHEMA.SQL_IDENTIFIER,
  COLLATION_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,
  NUMERIC_PRECISION       INFORMATION_SCHEMA.CARDINAL_NUMBER,
  NUMERIC_PRECISION_RADIX INFORMATION_SCHEMA.CARDINAL_NUMBER,
  NUMERIC_SCALE           INFORMATION_SCHEMA.CARDINAL_NUMBER,
  DATETIME_PRECISION                    INFORMATION_SCHEMA.CARDINAL_NUMBER,
  INTERVAL_TYPE           INFORMATION_SCHEMA.CHARACTER_DATA,
  INTERVAL_PRECISION                    INFORMATION_SCHEMA.CARDINAL_NUMBER,
  CONSTRAINT TABLE_OR_DOMAIN_CHECK_COMBINATIONS
    CHECK ( DATA_TYPE NOT IN ('CHARACTER', 'CHARACTER VARYING', 'BIT',
                              'BIT VARYING', 'REAL', 'DOUBLE PRECISION',
                              'FLOAT', 'INTEGER', 'SMALLINT', 'NUMERIC',
                              'DECIMAL', 'DATE', 'TIME', 'TIMESTAMP',
                              'TIME WITH TIME ZONE',
                              'TIMESTAMP WITH TIME ZONE', 'INTERVAL')
        OR (
             DATA_TYPE IN ( 'CHARACTER', 'CHARACTER VARYING' )
             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 ) IS NULL
             AND ( INTERVAL_TYPE, INTERVAL_PRECISION ) 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,
                   NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX,
                   NUMERIC_SCALE, DATETIME_PRECISION ) IS NULL
             AND ( INTERVAL_TYPE, INTERVAL_PRECISION ) 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
             ( INTERVAL_TYPE, INTERVAL_PRECISION ) 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
           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
          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 NULL
            AND
            NUMERIC_SCALE IS NULL
            AND
            DATETIME_PRECISION IS NOT NULL
            AND
            ( INTERVAL_TYPE, INTERVAL_PRECISION ) 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 NULL
            AND
            NUMERIC_SCALE IS NULL
            AND
            DATETIME_PRECISION IS NOT 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
          ) ),

  CONSTRAINT DATA_TYPE_DESCRIPTOR_PRIMARY_KEY
    PRIMARY KEY ( TABLE_OR_DOMAIN_CATALOG, TABLE_OR_DOMAIN_SCHEMA,
                  TABLE_OR_DOMAIN_NAME, COLUMN_NAME ),

  CONSTRAINT DATA_TYPE_CHECK_REFERENCES_COLLATION
    CHECK ( COLLATION_CATALOG NOT IN ( SELECT CATALOG_NAME FROM SCHEMATA )
        OR
          ( COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME ) IN
          ( SELECT COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME
              FROM COLLATIONS ) ),

  CONSTRAINT DATA_TYPE_DESCRIPTOR_CHECK_USED
    CHECK ( ( TABLE_OR_DOMAIN_CATALOG, TABLE_OR_DOMAIN_SCHEMA,
              TABLE_OR_DOMAIN_NAME, COLUMN_NAME )
            IN ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME
                   FROM COLUMNS
                 UNION
                 SELECT DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME, ' '
                   FROM DOMAINS )
          )
  )
```

2. *Rationale: Editorial.*

In Description 3), replace "the precision if it is numeric type, and, and the precision if it is a datetime or interval type" with "the precision if it is numeric type, the scale if it is a numeric type, and the fractional seconds precision if it is a datetime or interval type".

3. *Rationale: The descriptions of the existing columns do not allow for the fact that the interval type has two precisions: the fractional seconds precision and the interval leading field precision. In addition, the interval type also specifies the interval qualifier which must be encoded into the DATA_TYPE_DESCRIPTOR. Finally, numeric precision should not be allowed to be null for INTEGER, SMALLINT, NUMERIC, DECIMAL, DATE, and INTERVAL data types.*

Add the following Descriptions:

3.1) If DATA_TYPE is 'INTERVAL', then the values of INTERVAL_TYPE are the value for <interval qualifier> for the data type being described, excluding any <interval leading field precision> and an <interval fractional seconds precision>; otherwise, INTERVAL_TYPE is the null value.

3.2) If DATA_TYPE is 'INTERVAL', then the values of INTERVAL_PRECISION are the interval leading field precision of the data type being described; otherwise, INTERVAL_PRECISION is the null value.

21.3.6 DOMAINS base table

1. *Rationale: Correct CHECK constraints.*

Replace the constraint "DOMAINS_CHECK_DATA_TYPE" with:

```
CONSTRAINT DOMAINS_CHECK_DATA_TYPE
  CHECK ( DOMAIN_CATALOG NOT IN ANY ( SELECT CATALOG_NAME FROM SCHEMATA )
     OR
        ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME, '' ) IN
          ( SELECT TABLE_OR_DOMAIN_CATALOG, TABLE_OR_DOMAIN_SCHEMA,
                   TABLE_OR_DOMAIN_NAME, COLUMN_NAME
              FROM DATA_TYPE_DESCRIPTOR ) )
```

2. *Rationale: The following helps define the action taken when a default value cannot be represented in the Information Schema.*

In Description, replace the first sentence of Item 3) with:

The value of DOMAIN_DEFAULT is null if the domain being described has no explicit default value; the value of DOMAIN_DEFAULT is TRUNCATED if the character representation of the <default option> cannot be represented without truncation.

In Description, add the following at the end of Item 6):

Note: TRUNCATED is different from other values like CURRENT_USER, SYSTEM_USER, or CURRENT_DATE in that it is not an SQL keyword and does not correspond to a defined value in SQL.

21.3.8 TABLES base table

1. *Rationale: Editorial.*

In Definition, in the last constraint, change "EXCEPTIONEXISTS" to "EXISTS".

21.3.10 COLUMNS base table

1. *Rationale: Provide General Rules for maintaining the nullability characteristic in a column definition.*

In Definition, insert the following immediately after the column "IS_NULLABLE":

```
CONSTRAINT IS_NULLABLE_NOT_NULL NOT NULL
CONSTRAINT IS_NULLABLE_CHECK
     CHECK ( IS_NULLABLE IN ( 'YES', 'NO' ) ),
```

2. *Rationale: Correct CHECK constraints.*

Replace the constraint "COLUMNS_CHECK_REFERENCES_DOMAIN" with:

```
CONSTRAINT COLUMNS_CHECK_REFERENCES_DOMAIN
  CHECK ( DOMAIN_CATALOG NOT IN ( SELECT CATALOG_NAME FROM SCHEMATA )
     OR
        ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME ) IN
          ( SELECT DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME
              FROM DOMAINS ) ),
```

Replace the constraint "COLUMNS_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, COLUMN_NAME ) NOT IN
          ( SELECT TABLE_OR_DOMAIN_CATALOG, TABLE_OR_DOMAIN_SCHEMA,
                   TABLE_OR_DOMAIN_NAME, COLUMN_NAME
              FROM DATA_TYPE_DESCRIPTOR )
     OR
        ( DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME ) IS NULL
        AND
        ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME ) IN
          ( SELECT TABLE_OR_DOMAIN_CATALOG, TABLE_OR_DOMAIN_SCHEMA,
                   TABLE_OR_DOMAIN_NAME, COLUMN_NAME
              FROM DATA_TYPE_DESCRIPTOR )
        ) )
```

3. *Rationale: The following helps define the action taken when a default value cannot be represented in the Information Schema.*

In Description, replace the first sentence of Item 6) with:

The value of COLUMN_DEFAULT is null if the column being described has no explicit default value or if its default value comes only from a domain; the value of COLUMN_DEFAULT is TRUNCATED if the character representation of the <default option> cannot be represented without truncation.

Note: TRUNCATED is different from other values like CURRENT_USER, SYSTEM_USER, or CURRENT_DATE in that it is not an SQL keyword and does not correspond to a defined value in SQL.

21.3.11 VIEW_TABLE_USAGE base table

1. *Rationale: Clearly define which referenced tables belong in VIEW_TABLE_USAGE.*

In Function, replace "referenced in" with "identified by a <table name> simply contained in a <table reference> that is contained in".

2. *Rationale: Correct CHECK constraints.*

Replace the constraint "VIEW_TABLE_USAGE_CHECK_REFERENCES_TABLES" with:

```
CONSTRAINT VIEW_TABLE_USAGE_CHECK_REFERENCES_TABLES
  CHECK ( TABLE_CATALOG NOT IN ( SELECT CATALOG_NAME FROM SCHEMATA )
     OR
        ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) IN
          ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
              FROM TABLES ) ),
```

3. *Rationale: Clearly define which referenced tables belong in VIEW_TABLE_USAGE.*

In Description 2), replace "view requires" with "identified by a <table name> simply contained in a <table reference> that is contained in the <query expression> of the view being described".

21.3.12 VIEW_COLUMN_USAGE base table

1. *Rationale: Clearly define which referenced columns belong in VIEW_COLUMN_USAGE.*

In Function, replace "referenced by a view" with "identified by a <table name> simply contained in a <table reference> that is contained in the <query expression> of the view being described".

2. *Rationale: Correct CHECK constraints.*

Replace the constraint "VIEW_COLUMN_USAGE_CHECK_REFERENCES_COLUMNS" with:

```
CONSTRAINT VIEW_COLUMN_USAGE_CHECK_REFERENCES_COLUMNS
  CHECK ( TABLE_CATALOG NOT IN ( SELECT CATALOG_NAME FROM SCHEMATA )
     OR
        ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME ) IN
          ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME
              FROM COLUMNS ) ),
```

3. *Rationale: Clearly define which referenced columns belong in VIEW_COLUMN_USAGE.*

In Description 2), replace "identifier, respectively, of a column" with "column name, respectively, of a column of a table identified by a <table name> simply contained in a <table reference> that is contained in the <query expression> of the view".

21.3.13 TABLE_CONSTRAINTS base table

1) *Rationale: In the TABLE_CONSTRAINTS base table, the table constraint TABLE_CONSTRAINTS_CHECK_REFERENCES_TABLES specifies that if TABLE_CATALOG is referenced in the SCHEMATA base table, then the table specified by TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME must be defined in the TABLES base table. The table constraint TABLE_CONSTRAINTS_CHECK_VIEWS specifies that if TABLE_CATALOG is referenced in the SCHEMATA base table, then not only must the table specified by TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME be defined in the TABLES base table, it must also not be a view. Since any row that satisfies the table constraint TABLE_CONSTRAINT_CHECK_VIEWS will also satisfy the table constraint TABLE_CONSTRAINT_CHECK_TABLES, the table constraint TABLE_CONSTRAINT_CHECK_TABLES is redundant.*

In Definition, delete the constraint `TABLE_CONSTRAINTS_CHECK_REFERENCES_TABLES`.

2) *Rationale: Correct CHECK constraints.*

Replace the constraint "TABLE_CONSTRAINTS_CHECK_VIEWS" with:

```
CONSTRAINT TABLE_CONSTRAINTS_CHECK_VIEWS
CHECK ( TABLE_CATALOG NOT IN ( SELECT CATALOG_NAME FROM SCHEMATA )
   OR
      ( ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) IN
          ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
              FROM TABLES
              WHERE TABLE_TYPE <> 'VIEW' ) ),
```

21.3.15 REFERENTIAL_CONSTRAINTS base table

1) *Rationale: Editorial.*

Change the first line of the last CHECK to the following:

```
CHECK ( UNIQUE_CONSTRAINT_CATALOG NOT IN ( SELECT CATALOG_NAME FROM
SCHEMATA )
```

2) *Rationale: Editorial.*

In Description 4), replace "update rule" with "<update rule>".

21.3.17 CHECK_TABLE_USAGE base table

1) *Rationale: Clearly define which referenced tables belong in CHECK_TABLE_USAGE.*

In Function, replace "table referenced by" with "table identified by a <table name> simply contained in a <table reference> contained in".

2. *Rationale: Correct CHECK constraints.*

Replace the constraint "CHECK_TABLE_USAGE_CHECK_REFERENCES_TABLES" with:

```
CONSTRAINT CHECK_TABLE_USAGE_CHECK_REFERENCES_TABLES
  CHECK ( TABLE_CATALOG NOT IN ( SELECT CATALOG_NAME FROM SCHEMATA )
     OR
        ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) IN
          ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
              FROM TABLES ) )
```

3. *Rationale: Clearly define which referenced tables belong in CHECK_TABLE_USAGE.*

In Description 2), replace "table that is referenced by" with "table identified by a <table name> simply contained in a <table reference> contained in the <search condition> of".

21.3.18 CHECK_COLUMN_USAGE base table

1. *Rationale: Clearly define which referenced tables belong in CHECK_COLUMN_USAGE.*

In Function, replace "column referenced by" with "column identified by a <column reference> contained in".

2. *Rationale: Correct CHECK constraints.*

Replace the constraint "CHECK_TABLE_USAGE_CHECK_REFERENCES_TABLES" with:

```
CONSTRAINT CHECK_COLUMN_USAGE_CHECK_REFERENCES_COLUMNS
  CHECK ( TABLE_CATALOG NOT IN ( SELECT CATALOG_NAME FROM SCHEMATA )
     OR
        ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME ) IN
          ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME
              FROM COLUMNS ) )
```

3. *Rationale: Clearly define which referenced tables belong in CHECK_COLUMN_USAGE.*

In Description 2), replace "and qualified name, respectively, of a column that is referenced by" with "qualified identifier, and column name, respectively, of a column identified by a <column reference> explicitly or implicitly contained in the <search condition> of".

21.3.21 COLUMN_PRIVILEGES base table

1. *Rationale: Editorial.*

In Description 4), for each of the four values for PRIVILEGE_TYPE, replace "CATALOG.TABLE", "SCHEMA.TABLE", "NAME.COLUMN" with "CATALOG, TABLE", "SCHEMA, TABLE", "NAME, and COLUMN", respectively.

21.3.22 USAGE_PRIVILEGES base table

1) *Rationale: Editorial.*

In the Definition, replace "`OBJECT_TYPE INFORMATION_SCHEMA.SQL_IDENTIFIER`" with "`OBJECT_TYPE INFORMATION_SCHEMA.CHARACTER_DATA`".

21.3.23 CHARACTER_SETS base table

1) *Rationale: Correct CHECK constraints.*

Replace the constraint "CHARACTER_SETS_CHECK_REFERENCES_COLLATIONS" with:

```
CONSTRAINT CHARACTER_SETS_CHECK_REFERENCES_COLLATIONS
  CHECK ( DEFAULT_COLLATE_CATALOG NOT IN ( SELECT CATALOG_NAME FROM SCHEMATA )
     OR
        ( DEFAULT_COLLATE_CATALOG, DEFAULT_COLLATE_SCHEMA,
          DEFAULT_COLLATE_NAME ) IN
          ( SELECT COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME
              FROM COLLATIONS ) )
```

21.3.24 COLLATIONS base table

1) *Rationale: Correct CHECK constraints.*

Replace the constraint "COLLATIONS_CHECK_REFERENCES_CHARACTER_SETS" with:

```
CONSTRAINT COLLATIONS_CHECK_REFERENCES_CHARACTER_SETS
CHECK ( CHARACTER_SET_CATALOG NOT IN ( SELECT CATALOG_NAME FROM SCHEMATA )
   OR
      ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME ) IN
        ( SELECT CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
                 CHARACTER_SET_NAME
            FROM CHARACTER_SETS ) )
```

2) *Rationale: Editorial.*

In Description 1), replace "defined" with "described".

In Description 4), replace "A row always exists in this table for the collation SQL_TEXT." with "There is a row in this table for the collation INFORMATION_SCHEMA.SQL_TEXT.".

21.3.25 TRANSLATIONS base table

1) *Rationale: Correct CHECK constraints.*

Replace the constraints "TRANSLATIONS_CHECK_REFERENCES_SOURCE" and "TRANSLATIONS_CHECK_REFERENCES_TARGET" with:

```
CONSTRAINT TRANSLATIONS_CHECK_REFERENCES_SOURCE
  CHECK ( SOURCE_CHARACTER_SET_CATALOG NOT IN
            ( SELECT CATALOG_NAME FROM SCHEMATA )
     OR
        ( SOURCE_CHARACTER_SET_CATALOG, SOURCE_CHARACTER_SET_SCHEMA,
          SOURCE_CHARACTER_SET_NAME ) IN
           ( SELECT CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
```

```
                    CHARACTER_SET_NAME
                FROM CHARACTER_SETS ) ),

CONSTRAINT TRANSLATIONS_CHECK_REFERENCES_TARGET
  CHECK ( TARGET_CHARACTER_SET_CATALOG NOT IN
          ( SELECT CATALOG_NAME FROM SCHEMATA )
      OR
         ( TARGET_CHARACTER_SET_CATALOG, TARGET_CHARACTER_SET_SCHEMA,
           TARGET_CHARACTER_SET_NAME ) IN
            ( SELECT CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA,
                     CHARACTER_SET_NAME
                FROM CHARACTER_SETS ) )
```

21.3.26 SQL_LANGUAGES base table

1. *Rationale: Editorial.*

In Definition, replace "`SQL_LANGUAGE_YEAR = '1989' AND AND`" with "`SQL_LANGUAGE_YEAR = '1989' AND`".

2. *Rationale: Editorial.*

In Definition, replace "`SQL_LANGUAGE_PROGRAMMING_LANGUGE`" with "`SQL_LANGUAGE_PROGRAMMING_LANGUAGE`".

22.1 SQLSTATE

1. *Rationale: Clarify that 00, 01, and 02 are the only SQLSTATE classes for completion conditions.*

In the first sentence of the third paragraph, replace "implementation-specified conditions" with "implementation-specified exception conditions".

Add the following to the end of the third paragraph:

> An implementation-defined completion condition shall be indicated by returning an implementation-defined subclass in conjunction with the *successful completion*, the *warning*, or the *no data* classes.

2. *Rationale: Remove confusion regarding whether or not an implementation may provide information that distinguishes between syntax and access rule violation.*

Add the following note after the 4th paragraph:

> **Note:** One consequence of this is that an implementation may, but is not required by this standard to, provide subcodes for exception condition "syntax error or access rule violation" (Class 42) that distinguish between the syntax and access rule violation cases.

3. *Rationale: Clarify with respect to the language added to Subclause 4.18.1.*

Insert the following paragraph after the (new) note after the fourth paragraph:

> If multiple completion conditions: *warning* or exception conditions, including implementation-defined conditions, are raised, it is implementation-dependent which of the corresponding SQLSTATE values is returned in the SQLSTATE parameter, provided that the precedence rules in Subclause 4.18.1, "Status parameters", are obeyed. Any number of applicable condition values, in addition to the one returned in SQLSTATE, may be returned in the diagnostics area.

4. *Rationale: Clarify that an implementation-specified condition duplicates a standard-defined condition, even if it is a more specific specialization of the standard-defined condition.*

In the last paragraph, replace “may duplicate a condition” with “may duplicate, in whole or in part, a condition”.

5. *Rationale: Editorial.*

Insert the following row into Table 23, after the row with Condition "data exception" and Subcondition "invalid escape sequence":

```
|                                    |     | invalid interval format | 006 |
```

6. *Rationale: Editorial.*

In Table 23, change the Subcondition column for the entry with Condition "Remote Database Access" to read "(See Table 25 for subclass codes)".

7. *Rationale: The following unifies the SQLSTATE returned for the different ways of invoking an SQL statement.*

Delete the following rows from Table 23:

```
| syntax error or access rule violation in       | 2A | (no subclass) | 000 |
| direct SQL statement                           |    |               |     |

| syntax error or access rule violation in       | 37 | (no subclass) | 000 |
| dynamic SQL statement                          |    |               |     |
```

8. *Rationale: The General Rules that specify the completion condition 01001 (cursor operation conflict) pertain only to SQL3 and were inadvertently included in ISO/IEC 9075:1992. These rules and the completion code itself need therefore to be deleted.*

Delete the following row from Table 23:

```
|                                    |     | cursor operation conflict | 001 |
```

9. *Rationale: The following helps define the action taken when a default value cannot be represented in the Information Schema.*

Add the following rows to Table 23, between the second and third item with Condition "warning":

```
|                                    |     | default option too long | 00B |
|                                    |     | long for information    |     |
|                                    |     | schema                  |     |
```

22.2 SQLCODE

1. *Rationale: Completion conditions were added to SQL-92, but the values to be returned in the SQLCODE parameter were not specified. Since various implementations have used both 0 and positive values other than 100 to return warnings both of these alternatives are explicitly allowed.*

In Table 24-SQLCODE values, replace the Condition "successful completion" with "successful completion or *warning*".

Add the following row at the end of Table 24 — SQLCODE values:

| +m[1] | warning |

[1]m must be greater than 0 and not equal to 100

2. *Rationale: Errors that prevent the execution of an SQL-statement may occur in RDA service elements, and should be reported to the SQL-agent through SQLSTATE and subclass codes.*

Add a new subclause 22.3, "Remote Database Access SQLSTATE Subclasses":

22.3 Remote Database Access SQLSTATE Subclasses

This International Standard reserves SQLSTATE class 'HZ' for Remote Database Access errors, which may occur when an SQL-client interacts with an SQL-server across a communications network using an RDA Application Context. ISO/IEC 9579-1, ISO/IEC 9579-2, ISO 8649, and ISO/IEC 10026-2 define a number of exception conditions that must be detected in a conforming ISO RDA implementation. This subclause defines SQLSTATE subclass codes for each such condition out of the set of codes reserved for International Standards.

If an implementation using RDA reports a condition shown in Table 25 for a given exception condition, then it shall use the SQLSTATE class code 'HZ' and the subclass codes shown, and shall set the values of CLASS_ORIGIN to 'ISO 9075' and SUBCLASS_ORIGIN as indicated in Table 25 when those exceptions are retrieved by a <get diagnostics statement>.

An implementation using client-server communications other than RDA may report conditions corresponding to the conditions shown in Table 25 using the SQLSTATE class code 'HZ' and the corresponding subclass codes shown. It may set the values of CLASS_ORIGIN to 'ISO 9075' and SUBCLASS_ORIGIN as indicated in Table 25. Any other communications error shall be returned with a subclass code from the implementation-defined range, with CLASS_ORIGIN set to 'ISO 9075' and SUBCLASS_ORIGIN set to an implementation-defined character string.

Table 25 — SQLSTATE Subclasses for Class 'HZ'

RDA Generic Condition Subclass origin 'ISO/IEC 9579-1'	SQLSTATE subclass
(no subclass)	000
access control violation	010
bad repetition count	020
command handle unknown	030
control authentication failure	040
control services not allowed	230
data resource handle not specified	050
data resource handle unknown	060
data resource name not specified	070
data resource not available, permanent	080
data resource not available, transient	081
data resource already open	090
data resource unknown	100
dialogue id unknown	110
duplicate command handle	120
duplicate data resource handle	130
duplicate dialogue id	140
duplicate operation id	150
invalid sequence	160
* dialogue already active	161
* dialogue initializing	162
* dialogue not active	163
* dialogue terminating	164
* transaction not open	165
* transaction open	166
* transaction terminating	167
no data resource available	170
operation aborted, permanent	180
operation aborted, transient	181
operation cancelled	190
service not negotiated	200
transaction rolled back	210
user authentication failure	220

* **Note:** These are subcomponents of invalid sequence.

RDA SQL Specialization Condition Subclass origin 'ISO/IEC 9579-2'	SQLSTATE subclass
host identifier error	300
invalid SQL conformance level	310
RDA transaction not open	320
RDA transaction open	325
SQL access control violation	330
SQL database resource already open	340
SQL DBL argument count mismatch	350
SQL DBL argument type mismatch	360
SQL DBL no char set	365
SQL DBL transaction statement not allowed	370
SQL usage mode violation	380

Association Control Service Element Condition Subclass origin 'ISO 8649'	SQLSTATE subclass
abort failure service provider	410
abort failure service user	411
association failure, permanent	420
association failure, transient	421
release failure	430

Distributed Transaction Processing Condition Subclass origin 'ISO/IEC 10026-2'	SQLSTATE subclass
begin dialogue rejected provider	450
begin dialogue rejected user	451
heuristic hazard	460
heuristic mix	461
PAbort rollback false	470
PAbort rollback true	471
rollback	480
UAbort rollback false	490
UAbort rollback true	491
UError	4A0

Note: The error conditions given in this table correspond to ASN.1 error types formed by eliminating blanks between individual words and, in some cases, selecting one of a set of enumerated subparameters.

A Remote Database Access exception condition may also result in an SQL completion condition defined in Table 23 (such as '40000 Transaction Rollback); if such a condition occurs, then the 'HZ' class SQLSTATE value shall not be returned in the SQLSTATE parameter but may be returned in the Diagnostics Area.

23.2 Claims of Conformance

1. *Rationale: The newly revised Ada language standard (ISO/IEC-8652:1995, Information Technology — Programming Language — Ada) contains support for decimal-encoded numeric data and variable length character strings. The revised interface allows newly written applications in the revised Ada language access to these features of SQL; previously written Ada applications, conformant with the earlier Ada interface, are conformant with the revised interface.*

Add the following sentence to the end of item 3:

If binding to Ada is supported, then whether or not support is offered for library unit packages `Interfaces.SQL.Numerics`, `Interfaces.SQL.Varying`, and the packages supporting other character sets, as well as the capabilities supported by those packages.

23.3 Extensions and options

1. Rationale: Clarification.

In the first sentence of the first paragraph, replace "this International Standard" with "the level of this International Standard to which conformance is claimed".

A.1 Intermediate SQL Specifications

1. Rationale: Editorial.

Delete Item 1) a).

Delete Item 1) b).

Replace Item 2) a) with:

a) A <general literal> shall not be a <bit string literal> or a <hex string literal>.

Delete Item 2) c).

Replace Item 3) b) with:

b) Conforming Intermediate SQL language shall not contain any explicit <catalog name>, <connection name>, <collation name>, <translation name>, <form-of-use conversion name>, or <qualified local table name>.

Delete Item 3) e).

Delete Item 14).

Delete Item 15).

Replace Item 18) a) with:

a) The <set qualifier> DISTINCT shall not be specified more than once in a <query specification>, excluding any <subquery> of that <query specification>.

Delete Item 18) b).

Delete Item 20).

Replace Item 28) a) with:

a) Conforming Intermediate SQL language shall not specify TEMPORARY and shall not reference any global or local temporary able.

Delete Item 29) a).

Delete Item 30).

Add the following to Item 31):

b.1) The order of the column names in a <reference column list> shall be the same as the order of column names of the corresponding unique constraint of the referenced table.

Add the following Item:

39.1) Subclause 11.28, "<character set definition>":

a) In Conforming Intermediate SQL language, <collation source> shall specify DEFAULT.

Delete Item 48).

Delete Item 49).

Delete Item 51).

Delete Item 52).

Delete Item 53).

Replace Item 54) a) with:

a) No leaf generally underlying table of *T* shall be an underlying table of any <query expression> generally contained in the <search condition>.

Replace Item 55) a) with:

a) The leaf generally underlying table of *T* shall not be generally contained in the <query expression> immediately contained in he <insert columns and source> except as the <qualifier> of a <column reference>.

Replace Item 56) a) with:

a) CR shall not be an ordered cursor.

Replace Item 57) a) with:

a) No leaf generally underlying table of *T* shall be an underlying table of *T* shall be an underlying table of any <query expression> generally contained in the <search condition> or in any <value expression> immediately contained in any <update source> contained in the <set clause list>.

Delete Item 70).

Delete Item 77).

Delete Item 78).

Delete Item 79).

Delete Item 82).

Delete Item 89).

Delete Item 90).

2. Rationale: Intermediate SQL should not reference the ASSERTIONS view

Add the following Item:

90.1) Conforming Intermediate SQL shall not reference the ASSERTIONS view.

3. Rationale: Editorial.

Add the following Item:

90.2) Subclause 21.2.19, "COLLATIONS view".

a) Conforming Intermediate SQL language shall not reference the COLLATIONS view.

Add the following Item:

90.3) Subclause 21.2.20, "TRANSLATIONS view".

a) Conforming Intermediate SQL language shall not reference the TRANSLATIONS view.

A.2 Entry SQL Specifications

1. Rationale: Editorial.

Replace Item 2) a) with:

a) No <regular identifier> or <delimited identifier body> shall contain more than 18 <character representation>s.

Replace Item 3) with:

3) Subclause 5.3, "<literal>":

a) A <general literal> shall not be a <national character string literal>.

b) A <general literal> shall not be a <datetime literal> or an <interval literal>.

c) A <character string literal> shall contain at least one <character representation>.

d) Conforming SQL language shall contain exactly one repetition of <character representation> (that is, it shall contain exactly one sequence of "<quote> <character representation>... <quote>").

e) A <character string literal> shall not specify a <character set specification>.

Replace Item 4) a) with:

a) Conforming Entry SQL language shall not contain any <domain name>, <SQL statement name>, <dynamic cursor name>, <constraint name>, <descriptor name>, or <character set name>.

Add the following to Item 4):

a.1) An <identifier> shall not specify a <character set specification>.

Delete Item 7) a).

Delete Item 18) a).

Add the following Item:

18.1) Subclause 7.3, "<table expression>":

a) If the table identified in the <from clause> is a grouped view, then the <table expression> shall not contain a <where clause>, <group by clause>, or <having clause>.

Add the following Item:

18.2) Subclause 7.4, "<from clause>":

a) If the table identified by <table name> is a grouped view, then the <from clause> shall contain exactly one <table reference>.

2. *Rationale: Remove the ambiguity in Leveling Rule 2).*

Replace Item 20) a) with:

a) A <value expression> directly contained in the <search condition> shall not contain a <column reference> that references a <derived column> that generally contains a <set function specification>.

3. *Rationale: Editorial.*

Replace Item 21) b) with:

b) A <select sublist> shall be a <derived column>.

Add the following to Item 21):

b.1) If the <table expression> of the <query specification> is a grouped view, then the <select list> shall not contain a <set function specification>.

4. *Rationale: Support changes to Subclause 7.11.*

Add the following to Item 23):

a.1) The <query expression> contained in a <subquery> shall be a <query specification>.

5. *Rationale: SQL-89 prohibited the use of a subquery with degree greater than one as the argument of an <exists predicate>. It was intended to relax this restriction in Intermediate SQL, but the required Leveling Rule for Entry SQL was inadvertently omitted.*

Add the following to Item 23):

b.1) If a <table subquery> is simply contained in an <exists predicate>, then the <select list> of the <query specification> directly contained in the <table subquery> shall comprise either an <asterisk> or a single <derived column>.

6. Rationale: Editorial.

Add the following Item:

31.1) Subclause 10.6, "<constraint name definition> and <constraint attributes>":

a) Conforming Entry SQL language shall contain no <constraint name definition>.

7. Rationale: Editorial.

Replace Item 32) a) with:

a) Conforming Entry SQL language shall not contain any <domain definition>.

8. Rationale: Editorial

Replace Item 32) d) with:

d) Conforming Entry SQL language shall not contain any <character set specification>.

Add the following to Item 56):

b.1) A <module authorization clause> shall specify AUTHORIZATION and shall not specify SCHEMA.

Delete Item 58).

Replace Item 59) with:

59) Subclause 12.5, "<SQL procedure statement>":

a) An <SQL procedure statement> shall not be an <SQL schema statement>.

Add the following to Item 61):

b.1) A <fetch statement> shall not contain a <fetch orientation>.

Add the following to Item 62):

a.1) The <table expression> shall not include a <group by clause> or a <having clause> and shall not identify a grouped view.

9. Rationale: Editorial

Replace Item 63) a) with:

a) The <query expression> that is contained in an <insert columns and source> shall be a <query specification> or it shall be a <table value constructor> that contains exactly one <row value constructor> of the form "<left paren> <row value constructor list> <right paren>", and each <row value constructor element> of that <row value constructor list> shall be a <value specification> or a <null specification>.

10. Rationale: Editorial

In Item 69) a) replace "Intermediate" with "Entry".

Delete Item 91) a).

Replace Item 91) c) with:

c) No two <host variable definition>s shall specify the same variable name.

Replace Item 93) with:

93) Subclause 19.4. "<embedded SQL C program>":

a) A <C derived variable> shall not to be a <C VARCHAR variable>.

b) A <C variable definition> shall not contain a <character set specification>.

11. Rationale: Editorial

Delete Item 94) b).

12. Rationale: Editorial

Add the following Item:

95.1) Subclause 19.6, "<embedded SQL MUMPS program>":

a) Conforming Entry SQL language shall contain no <embedded SQL MUMPS program>.

Delete Item 98) b).

Delete Item 98) c).

Annex B: Implementation-defined elements

1. Rationale: Clarification.

In Item 24), replace "SQL-environment and create" with "SQL-server and initiate".

In Item 25), replace "both occurrences of "SQL-environment" with "SQL-server".

In Item 26), replace "environment" with "SQL-implementation".

2. Rationale: Changes made to 9.1 and 9.2 require a change in Annex B.

Add the following Items:

78.1) Subclause 9.1, "Retrieval assignment": If a value *V* is interval with a greater precision than a target *T*, then it is implementation-defined whether the approximation of *V* retrieved into *T* is obtained by rounding or by truncation.

79.1) Subclause 9.2, "Store assignment": If a value *V* is interval with a greater precision than a target *T*, then it is implementation-defined whether the approximation of *V* stored into *T* is obtained by rounding or by truncation.

3. Rationale: Changes made to 11.37 require a change in Annex B.

Add the following Item:

90.1) Subclause 11.37, "<revoke statement>": When loss of the USAGE privilege on a character set causes a module to be determined to be a lost module, the impact on that module is implementation-defined.

4. Rationale: Clarification.

In Item 112), replace "SQL-environment" with "SQL-server".

In Item 113), replace "SQL-environment" with "SQL-server".

5. Rationale: Insert missing items.

118.1) Subclause 17.6, "<prepare statement>": The maximum length L for the CHARACTER VARYING(L) data type assumed for a value specified in the <match value> in <pattern> or <escape character> of a <like predicate> is implementation-defined.

118.2) Subclause 17.6, "<prepare statement>": The maximum length L for the CHARACTER VARYING(L) data type assumed for a value specified in a <value specification> in <set catalog statement>, <set schema statement>, <set names statement> or <set session authorization identifier statement> is implementation-defined.

118.3) Subclause 17.6, "<prepare statement>": The maximum length L for the CHARACTER VARYING(L) data type assumed for a value specified in a <simple value specification> in <set local time zone statement> is implementation-defined.

Annex C Implementation-dependent elements

1. Rationale: Editorial.

Delete Item 41).

Add the following Item:

42.1) Subclause 12.1, "<module>": If the SQL-agent performs calls of <procedure>s from more than one Ada task, then the results are implementation-dependent.

Add the following Item:

42.2) Subclause 12.1, "<module>": After the execution of the last <procedure>, if an unrecoverable error has not occurred, and the SQL-agent did not terminate unexpectedly, and no constraint is not satisfied, then the choice of whether to perform a <commit statement> or a <rollback statement> is implementation-dependent. The determination of whether an SQL-agent has terminated unexpectedly is implementation-dependent.

Delete Item 44).

Insert the following as the first sentence of Item 72):

A <rollback statement> or a <commit statement> is effectively executed.

2. *Rationale: Resolve inconsistency regarding "no data" and exception condition.*

Replace Item 49) with:

49) Subclause 13.3 "<fetch statement>": If an exception condition occurs during the assignment of a value to a target, then the values of all targets are implementation-dependent.

3. *Rationale: The behavior of <get descriptor statement> should be consistent with the behavior of <using clause>.*

Replace Item 57) with:

57) For a <dynamic parameter specification>, the value of UNNAMED is 1 and the value of NAME is implementation-dependent.

4. *Rationale: Support changes to Subclause 22.1.*

Add the following Item:

75.1) If multiple completion conditions: *warning* or exception conditions, including implementation-defined conditions, are raised, it is implementation-dependent which of the corresponding SQLSTATE values is returned in the SQLSTATE parameter, provided that the precedence rules in Subclause 4.18.1, “Status parameters", are obeyed.

Annex E Incompatibilities with ISO/IEC 9075:1989

1. *Rationale: Editorial.*

In Item 10), delete "— BETWEEN" and insert "— CURRENT" between "— CROSS" and "CURRENT_DATE".

2. *Rationale: Document the intent to change the SQL standard so that the General Rules of <declare cursor> are applied each time that an <open statement> is executed.*

Add the following Item:

9.1) In ISO/IEC 9075:1989, although General Rules were specified for the non-executable <declare cursor>, it was not stated that the General Rules specified for the non-executable <declare cursor> were to be applied each time that a cursor is opened. This International Standard adds a General Rule to Subclause 13.2, “<open statement>”, and to Subclause 17.14, “<dynamic open statement>”, to explicitly state this association.

3. *Rationale: Completion conditions were added to ISO/IEC 9075:1992, but the values to be returned in the SQLCODE parameter were not specified.*

Add the following Item:

9.2) In ISO/IEC 9075:1989, the SQLCODE status parameter could only return a negative number to indicate an exception condition, 0 to indicate successful completion, and 100 to indicate no data. In this International Standard, a completion condition of *warning* will cause a value of 0 or a value that is positive and not equal to 100 to be returned for SQLCODE.

4. Rationale: Document an incompatibility with ISO/IEC 9075:1989

Add the following Item:

9.3) In ISO/IEC 9075:1989, it was valid to write <check constraint definition>s that generally contained the <value specification> USER. In this International Standard, there are prohibitions against <check constraint definition> that generally contain the <value specification> CURRENT_USER, which is the syntactic equivalent of USER.

Annex F Maintenance and interpretation of SQL

1. Rationale: Editorial.

Insert the following before the last paragraph:

In addition, this International Standard introduces some corrections to defects in ISO/IEC 9075-1989. These corrections do not cause incompatibilities with that International Standard.

1) In ISO/IEC 9075-1989, <value expression> can be a unary + or - followed by a <value specification> that can be a <literal> that includes a <signed numeric literal> that can have a leading + or -. This makes it ambiguous whether the unary + or - is part of the <value expression>. In this International Standard, <value expression> uses <unsigned value specification> that uses <unsigned literal> that does not permit a unary + or -. Thus, a unary + or - in a <value expression> will be a part of the <value expression>, and not a part of the <unsigned literal>. This change does not affect the expressiveness of the SQL language or add any restriction in the use of unary + or -.

(Blank page)

ISO/IEC 9075-3:1995
Database Languages - SQL-Part 3:Call-Level Interface (SQL/CLI)

Contents

1. *Rationale: Provide a mechanism for obtaining unbound output parameter data. and make it possible to invoke PSM procedures from CLI.*

Replace the seventh and eighth paragraphs with:

When a <dynamic select statement> or <dynamic single row select statement> is prepared or executed immediately, a description of the resultant columns is automatically provided in the applicable CLI implementation descriptor area. In this case, the application may additionally retrieve information by using the DescribeCol and/or the ColAttribute routine to obtain a description of a single resultant column and by using the NumResultCols routine to obtain a count of the number of resultant columns. The application sets values in the CLI application descriptor area for the description of the corresponding target specifications either explicitly using the routines SetDescField and SetDescRec or implicitly using the routine BindCol.

When an SQL-statement is prepared or executed immediately, a description of the dynamic parameters is automatically provided in the applicable CLI implementation descriptor area if this facility is supported by the current SQL-connection. An attribute associated with the allocated SQL-connection indicates whether this facility is supported. The value of the attribute may be retrieved using the routine GetConnectAttr. Regardless of whether automatic description is supported, all dynamic input (or input/ouput) parameters must be defined in the CLI application descriptor area before SQL-statement execution. This can be done either explicitly using the routines SetDescField and SetDescRec or implicitly using the routine BindParameter. The value of a dynamic input (or input/ouput) parameter may be established before SQL-statement execution (immediate parameter value) or may be provided during SQL-statement execution (deferred parameter value). Its description in the CLI descriptor area determines which method is in use. The ParamData routine is used to cycle through and process deferred input (or input/ouput) parameter values. The PutData routine is used to provide the deferred values. The PutData routine also enables the values of character string input (or input/output) parameters to be provided in pieces.

Before a <call statement> is prepared or executed immediately, the application may choose whether or not to bind any dynamic output parameters in the CLI application descriptor area. This can be done either explicitly using the routines SetDescField and SetDescRec or implicitly using the routine BindParameter. After execution of the statement, values for unbound output (or input/ouput) parameters can be individually retrieved using the GetParamData routine. The GetParamData routine also enables the values of character and binary string output (or input/ouput) parameters to be retrieved piece by piece.

4.1 Introduction

1. *Rationale: The following changes make it possible to invoke PSM procedures from CLI.*

In the eighth paragraph replace "BindParam" with "BindParameter".

4.3 Diagnostics areas

1. *Rationale: SQL/PSM [2] defines a new SQL diagnostics field CONDITION_IDENTIFIER which contains the <condition name> of a user-defined exception when the RETURNED_SQLSTATE field corresponds to unhandled user-defined exception. This paper proposes to add support for this new diagnostics field to SQL/CLI.*

Add the following entry to Table 1 "Fields in CLI diagnostics area" (in alphabetical order by field name):

Fields in status records	
Field	Data type
CONDITION_IDENTIFIER	CHARACTER VARYING (L)

2. *Rationale: SQL/PSM [2] defines a new SQL diagnostics field CONDITION_IDENTIFIER which contains the <condition name> of a user-defined exception when the RETURNED_SQLSTATE field corresponds to unhandled user-defined exception. This paper proposes to add support for this new diagnostics field to SQL/CLI.*

Change the sentence following Table 1 "All diagnostics area fields" to read:

> All diagnostics area fields specified in other parts of ISO/IEC 9075 that are not included in this table are not applicable to SQL/CLI.

4.4.7 CLI descriptor areas

1. *Rationale: The following changes make it possible to invoke PSM procedures from CLI.*

In the third paragraph replace "BindParam" with "BindParameter".

5.1 <CLI routine>

1. *Rationale: The following changes make it possible to invoke PSM procedures from CLI.*

In the Format in the production for <CLI generic name> replace "BindParam" with "BindParameter".

2. *Rationale: Provide a mechanism for obtaining unbound output parameter data.*

In the Format, in the production for <CLI generic name> insert the following in alphabetic sequence:

| GetParamData

3. *Rationale: The following changes make it possible to invoke PSM procedures from CLI.*

In Table 3, "Abbreviated CLI generic names" replace "BindParam" with "BindParameter".

4. *Rationale: Provide a mechanism for obtaining unbound output parameter data.*

Insert the following entry into Table 3 "Abbreviated CLI generic names" (in alphabetical order by Generic Name):

```
| GetParamData      | GPD                                                    |
```

5. *Rationale: In Syntax Rule 12) various phrases are used inwith double quotes. Other similar phrases in this clause useitalics instead of double quotes.*

In Syntax Rule 12), rewrite the phrases with double quotes, "operative data type correspondence table", "SQL data type column" and "host data type column" in italics and without double quotes.

6. *Rational: Editorial*

In Syntax Rule 12), replace the ", Refer" with ". Refer".

5.2 <CLI routine> invocation

1. *Rationale: In this clause, SQL/CLI routines are referred with single quotes. But in General Rule 3) b) 6), those are not referred without single quotes just like, "...Cancel nor ParamData nor PutData...".*

CLI routines should be referred with single quotes just like, "...'Cancel' nor 'ParamData' nor 'PutData'...".

5.3 SQL/CLI common elements

1. *Rationale: SQL/PSM [2] defines a new SQL diagnostics field CONDITION_IDENTIFIER which contains the <condition name> of a user-defined exception when the RETURNED_SQLSTATE field corresponds to unhandled user-defined exception. This paper proposes to add support for this new diagnostics field to SQL/CLI.*

Add the following entry to Table 12 "Codes used for diagnostics fields" (in alphabetical order by field name):

Fields in status records	
Field	Code
CONDITION_IDENTIFIER	25

2. *Rationale: SQL/PSM [2] defines a new SQL diagnostics field CONDITION_IDENTIFIER which contains the <condition name> of a user-defined exception when the RETURNED_SQLSTATE field corresponds to unhandled user-defined exception. This paper proposes to add support for this new diagnostics field to SQL/CLI.*

Change the sentence following Table 12 "All diagnostics area fields" to read:

All diagnostics area fields specified in other parts of ISO/IEC 9075 that are not included in this table are not applicable to SQL/CLI.

5.3.3 Implicit using clause

1. Rationale: Add support for PSM output parameters

Replace General Rule 9) with:

9) If *T* is 'EXECUTE' or 'OPEN', then *IPD* and *APD* describe the <dynamic parameter specification>s and <dynamic parameter specification> values, respectively, for the statement being executed. Let *D* be the number of <dynamic parameter specification>s in *S*.

a) If the value of COUNT for *APD* is less than zero, then an exception condition is raised: *dynamic SQL error — invalid descriptor count*.

b) If the value of COUNT for *IPD* is less than zero, then an exception condition is raised: *dynamic SQL error — invalid descriptor count*.

c) If the value of COUNT for *IPD* is less than *D*, then an exception condition is raised: *dynamic SQL error — using clause does not match dynamic parameter specifications*.

d) If the value of COUNT for *IPD* is greater than *D*, then it is implementation-defined whether an exception condition is raised: *dynamic SQL error — using clause does not match dynamic parameter specifications*.

e) If the first *D* item descriptor areas of *IPD* are not valid as specified in Subclause 5.3.8, "Description of CLI item descriptor areas", then an exception condition is raised: *dynamic SQL error — using clause does not match dynamic parameter specifications*.

f) Let *AD* be the minimum of the value of COUNT for *APD* and the value of COUNT for *IPD*.

g) For each of the first *AD* item descriptor areas of *APD*, if TYPE indicates DEFAULT, then:

i) Let *TP*, *P*, and *SC* be the values of TYPE, PRECISION, and SCALE, respectively, for the corresponding item descriptor area of *IPD*.

ii) The data type, precision, and scale of the described <dynamic parameter specification> value are set to *TP*, *P*, and *SC*, respectively, for the purposes of this invocation only.

h) If the first *AD* item descriptor areas of *APD* are not valid as specified in Subclause 5.3.8, "Description of CLI item descriptor areas", then an exception condition is raised: *dynamic SQL error — using clause does not match dynamic parameter specifications*.

i) For each item descriptor area for which DEFERRED is false in the first *AD* item descriptor areas of *APD* and whose corresponding <dynamic parameter specification> has a <parameter mode> of PARAM MODE IN or PARAM MODE INOUT, refer to the corresponding <dynamic parameter specification> value as an immediate parameter value and refer to the corresponding <dynamic parameter specification> as an immediate parameter.

j) For the *i*-th immediate parameter value as described by the corresponding item descriptor area *IDA* of *APD*:

i) If NULL is false for *IDA*, then:

1) Let *V* be the value of the host variable addressed by DATA_POINTER.

2) Case:

A) If TYPE indicates CHARACTER, then

Case:

I) If OCTET_LENGTH_POINTER is zero or if OCTET_LENGTH_POINTER is not zero and the value of the host variable addressed by OCTET_LENGTH_ POINTER indicates NULL TERMINATED, then let *L* be the number of characters of *V* that precede the implementation-defined null character that terminates a C character string.

II) Otherwise, let *Q* be the value of the host variable addressed by OCTET_LENGTH_POINTER and let *L* be the number of characters wholly contained in the first *Q* octets of *V*.

B) Otherwise, let *L* be zero.

3) Let *SV* be *V* with effective data type *SDT*, as represented by the length value *L* and by the values of TYPE, PRECISION, and SCALE.

ii) Otherwise, let *SV* be the null value.

k) Let *TDT* be the effective data type of the *i*-th immediate parameter as represented by the values of TYPE, LENGTH, PRECISION, SCALE, DATETIME_INTERVAL_CODE, DATETIME_INTERVAL_PRECISION, CHARACTER_SET_CATALOG, CHARACTER_SET_ SCHEMA, and CHARACTER_SET_NAME in the corresponding item descriptor area of *IPD*.

l) If the <cast specification>

```
CAST (SV AS TDT)
```

violates the Syntax Rules of Subclause 6.10, "<cast specification>", of ISO/IEC 9075:1992, then an exception condition is raised: *dynamic SQL error — restricted data type attribute violation*.

m) If the <cast specification>

```
CAST (SV AS TDT)
```

violates the General Rules of Subclause 6.10, "<cast specification>", of ISO/IEC 9075:1992, then an exception condition is raised in accordance with the General Rules of Subclause 6.10, "<cast specification>", of ISO/IEC 9075:1992.

n) Let *TV* be the value obtained, with data type *TDT*, by effectively performing the <cast specification>

```
CAST (SV AS TDT)
```

o) Let *ADT* be the effective data type of the actual *i*-th immediate parameter, defined to be the data type represented by the values of TYPE, LENGTH, PRECISION, SCALE, DATETIME_ INTERVAL_CODE, DATETIME_INTERVAL_PRECISION, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, and CHARACTER_SET_NAME that would automatically be set in the corresponding item descriptor area of *IPD* if POPULATE IPD was *true* for C.

p) If the <cast specification>

```
CAST (TV AS ADT)
```

violates the Syntax Rules of Subclause 6.10, "<cast specification>", of ISO/IEC 9075:1992, then an exception condition is raised: *dynamic SQL error — restricted data type attribute violation*.

q) If the <cast specification>
CAST (*TV* AS *ADT*)
violates the General Rules of Subclause 6.10, "<cast specification>", of ISO/IEC 9075:1992, then an exception condition is raised in accordance with the General Rules of Subclause 6.10, "<cast specification>", of ISO/IEC 9075:1992.

r) The <cast specification>
CAST (*TV* AS *ADT*)
is effectively performed and is the value of the *i*-th immediate parameter.

s) If DEFERRED is true for at least one of the first *AD* item descriptor areas of *APD*, then

i) Let *PN* be the parameter number associated with the first such item descriptor area.

ii) *PN* becomes the deferred parameter number associated with *AS*.

iii) If *T* is 'EXECUTE', then *S* becomes the statement source associated with *AS*.

iv) An exception condition is raised: *CLI-specific condition — dynamic parameter value needed*.

2. Rationale: Add support for PSM output parameters

Add a new General Rule as follows:

9.1) If *T* is 'CALL', then *IPD* and *APD* describe the <dynamic parameter specification>s and <dynamic parameter specification> values, respectively, for the <call statement> being executed. Let *D* be the number of <dynamic parameter specification>s in *S*.

a) If the value of COUNT for *APD* is less than zero, then an exception condition is raised: *dynamic SQL error — invalid descriptor count*.

b) Refer to a <dynamic parameter specification> value whose corresponding item descriptor area has a non-zero DATA_POINTER and whose corresponding <dynamic parameter specification> has a <parameter mode> of PARAM MODE OUT or PARAM MODE INOUT as a *bound target* and refer to the corresponding <dynamic parameter specification> as a *bound parameter*.

c) If every item descriptor area corresponding to a bound target in the first *D* item descriptor areas of *APD* is not valid as specified in Subclause 5.3.8, "Description of CLI item descriptor areas", then an exception condition is raised: *dynamic SQL error — using clause does not match target specifications*.

d) Let *SDT* be the effective data type of the i-th bound parameter as represented by the values of TYPE, LENGTH, PRECISION, SCALE, DATETIME_INTERVAL_CODE, DATETIME_INTERVAL_PRECISION, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA and CHARACTER_SET_NAME in the corresponding item descriptor area of *IPD*. Let *SV* be the value of the output parameter, with data type *SDT*.

e) Let *T1*, *OL*, *DP*, *IP*, and *LP* be the values of TYPE, OCTET_LENGTH, DATA_POINTER, INDICATOR_POINTER, and OCTET_LENGTH_POINTER, respectively, in the item descriptor area corresponding to the *i*-th bound target.

f) Case:

i) If *T1* indicates CHARACTER, then:

1) Let *UT* be the code value corresponding to CHARACTER VARYING as specified in Table 6, "Codes used for implementation data types in SQL/CLI".

2) Let *LV* be the implementation-defined maximum length for a CHARACTER VARYING data type.

ii) Otherwise, let *UT* be *T1* and let *LV* be 0.

g) Let *TDT* be the effective data type of the *i*-th bound target as represented by the type *UT*, the length value *LV*, and the values of PRECISION, SCALE, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, and CHARACTER_SET_NAME in the corresponding item descriptor area of *APD*.

h) If the <cast specification>
`CAST (`*SV* `AS` *TDT*`)`
does not conform to the Syntax Rules of Subclause 6.10, "<cast specification>", in ISO/IEC 9075:1992, then an exception condition is raised: *dynamic SQL error — restricted data type attribute violation.*

i) If the <cast specification>
`CAST (`*SV* `AS` *TDT*`)`
does not conform to the General Rules of Subclause 6.10, "<cast specification>", in ISO/IEC 9075:1992, then an exception condition is raised in accordance with the General Rules of Subclause 6.10, "<cast specification>", of ISO/IEC 9075:1992.

j) The <cast specification>
`CAST (`*SV* `AS` *TDT*`)`
is effectively performed and the result is the value *TV* of the *i*-th bound target.

k) If *TV* is the null value, then

Case:

i) If *IP* is zero, then an exception condition is raised: *data exception — null value, no indicator parameter*.

ii) Otherwise, the value of the host variable addressed by *IP* is set to -1.

l) If *TV* is not the null value, then:

i) If *IP* is not zero, then the value of the host variable addressed by *IP* is set to 0.

ii) Case:

1) If *T1* does not indicate CHARACTER, then the value of the host variable addressed by *DP* is set to *TV*.

2) Otherwise, the General Rules of Subclause 5.3.4, "Character string retrieval", are applied with *DP*, *TV*, *OL*, and *LP* as TARGET, VALUE, TARGET OCTET LENGTH, and RETURNED OCTET LENGTH, respectively.

5.3.4 Character string retrieval

1. *Rationale: General Rule 5) a) ii) covers the case in which the completion condition "warning — string data, right truncation" is raised but the rules defines an action which includes all of the octets of V.*

Replace General Rule 5) a) ii) with:

ii) Otherwise, *T* is set to the first *TL* octets of *V* and a completion condition is raised: *warning — string data, right truncation*.

5.3.7 CLI-specific status codes

1. *Rationale: The following changes make it possible to invoke PSM procedures from CLI.*

Insert the following subcondition into Table 4, "SQLSTATE class and subclass values for CLI-specific conditions":

```
|            |      | invalid parameter mode            | 105 |
```

2. *Rationale: Delete unused CLI-specific condition.*

Delete from Table 4 "SQLSTATE class and subclass values for CLI-specific conditions" the following entry:

```
|            |      | descriptor invalid on indirect    | 023 |
|            |      | reference                         |     |
```

5.3.8 Description of CLI item descriptor areas

1. *Rationale: Editorial: <interval leading precision> is defined neither in SQL/CLI nor in SQL:1992.*

Replace Syntax Rule 2) h) with:

h) TYPE indicates an <interval type>, DATETIME_INTERVAL_CODE is one of the code values in Table 9, "Codes associated with <interval qualifier> in SQL/CLI", to indicate the <interval qualifier> of the interval data type, DATETIME_INTERVAL_PRECISION is a valid <interval leading field precision>, and PRECISION is a valid value for <interval fractional seconds precision>, if applicable.

2. *Rational: Editorial*

In Syntax Rule 3), replace the ", Refer" with ". Refer".

3. *Rationale: The descriptor consistency check should apply to implementation parameter descriptors.*

In Syntax Rule 4) replace the text "neither an implementation row descriptor nor an implementation parameter descriptor" with "not an implementation row descriptor".

4. *Rationale: Add support for PSM output parameters*

Replace Syntax Rule 5) with:

5) Let *IDA* be a CLI item descriptor area in an application parameter descriptor. Let *IDA1* be the corresponding item descriptor area in the implementation parameter descriptor.

5. *Rationale: Syntax Rule 9) c) v) 3) defines a variable HV to be the value of the host variable addressed by OCTET_LENGTH_POINTER and then uses it as if it were the host variable addressed by DATA_POINTER and to add support for PSM output parameters*

Replace Syntax Rule 9) c) v) with:

v) TYPE indicates CHARACTER and one of the following is true:

1) NULL is true.

2) DEFERRED is true.

3) OCTET_LENGTH_POINTER is not zero, PARAMETER_MODE in *IDA1*is PARAM MODE IN or PARAM MODE INOUT, the value *V* of the host variable addressed by OCTET_LENGTH_POINTER is greater than zero, and the number of characters wholly contained in the first *V* octets of the host variable addressed by DATA_POINTER is a valid length value for a CHARACTER data type.

4) OCTET_LENGTH_POINTER is not zero, PARAMETER_MODE in *IDA1*is PARAM MODE IN or PARAM MODE INOUT, the value of the host variable addressed by OCTET_LENGTH_POINTER indicates NULL TERMINATED, and the number of characters of the value of the host variable addressed by DATA_POINTER that precede the implementation-defined null character that terminates a C character string is a valid length value for a CHARACTER data type.

5) OCTET_LENGTH_POINTER is zero, PARAMETER_MODE in *IDA1* is PARAM MODE IN or PARAM MODE INOUT, and the number of characters of the value of the host variable addressed by DATA_POINTER that precede the implementation-defined null character that terminates a C character string is a valid length value for a CHARACTER data type.

6) PARAMETER_MODE in *IDA1* is PARAM MODE OUT.

6. *Rationale: Add support for PSM output parameters*

Replace Syntax Rule 9) d) iii) with:

iii) DATA_POINTER is not zero and one of the following is true:

1) NULL is true.

2) DEFERRED is true.

3) PARAMETER_MODE in *IDA1* is PARAM MODE IN or PARAM MODE INOUT and the value of the host variable addressed by DATA_POINTER is a valid value of the data type indicated by TYPE.

4) PARAMETER_MODE in *IDA1* is PARAM MODE OUT.

7. *Rationale: Add support for PSM output parameters*

Add a new Table as follows:

Table 9.1 — Codes associated with <parameter mode> in SQL/CLI

Parameter mode	Code
PARAM MODE IN	1
PARAM MODE INOUT	2
PARAM MODE OUT	4

5.3.9 <CLI routine>

1. *Rationale: Provide a mechanism for obtaining unbound output parameter data.*

Insert the following entry into Table 22 "Miscellaneous codes used in CLI" (in alphabetical order by Code within Context):

```
| Data type          | -99       | APD TYPE                        |
```

2. *Rationale: Provide a mechanism for obtaining unbound output parameter data.*

Change the name of Table 23 to "Codes used for GetData and GetParamData data types" and change all references to this table to use the same name.

3. *Rationale: The following changes make it possible to invoke PSM procedures from CLI.*

In Table 24, "Codes used to identify SQL/CLI routines", replace the entry:

```
| BindParam       | 1002                                  |
```

with

```
| BindParameter   | 72                                    |
```

4. *Rationale: Provide a mechanism for obtaining unbound output parameter data.*

Insert the following entry into Table 24 "Codes used to identify SQL/CLI routines" (in alphabetical order by Generic Name):

```
| GetParamData    | 1025                                  |
```

5. *Rationale: Provide a mechanism for obtaining unbound output parameter data.*

Insert the following entry into Table 25 "Codes and data types for implementation information" (in alphabetical order by Information Type):

```
| GETPARAMDATA EXTENSIONS  | 20000     | INTEGER         |
```

6. *Rationale: Define maximum statement lengths in octets instead of characters.*

In Table 25 "Codes and data types for implementation information", replace:

```
| MAXIMUM STATEMENT LENGTH           | 105    | SMALLINT |
```

with:

```
| MAXIMUM STATEMENT OCTETS           | 20000  | INTEGER |
| MAXIMUM STATEMENT OCTETS DATA      | 20001  | INTEGER |
| MAXIMUM STATEMENT OCTETS SCHEMA    | 20002  | INTEGER |
```

7. *Rationale: Provide a mechanism for obtaining unbound output parameter data.*

Insert the following new table:

Table 29.1 — Values for GETPARAMDATA EXTENSIONS with GetInfo

Implementation Type	Value
ANY PARAMETER	1
ANY ORDER	2

5.4 Data type correspondences

1. *Rationale: In Table 42, the value of "K" is used for null string length. it depends on each character set. But the description of "K" in the table's footnote 2 is ambiguous.*

In Table 42 replace "L+k" with "(L+1)*k".

6.4 AllocStmt

1. *Rationale: Editorial: The two parameters should be separated by a "," and not by a ")" as shown in the Definition.*

Replace the Definition with:

```
AllocStmt (
     ConnectionHandle            IN    INTEGER,
     StatementHandle                   OUT    INTEGER )
     RETURNS SMALLINT
```

6.5 BindCol

1. *Rationale: Editorial: The Definition identifies a parameter named "StrLen_or_Len". It should be "StrLen_or_Ind" to match up with its use in General Rules 16 and 17.*

In the Definition replace "StrLen_or_Len" by "StrLen_or_Ind".

2. Rationale: Delete unnecessary test of descriptor validity

Replace General Rule 3) with:

3) Let *ARD* be the allocated CLI descriptor area identified by *HV* and let *N* be the value of COUNT for *ARD*.

3. Rational: Editorial

In General Rule 7), replace the ", Refer" with ". Refer".

6.6 BindParam

1. Rationale: The following changes make it possible to invoke PSM procedures from CLI.

Change the title of the Subclause from "BindParam" to "BindParameter".

Replace the Definition with:

```
BindParameter (
     StatementHandle      IN       INTEGER,
     ParameterNumber      IN       SMALLINT,
     InputOutputMode      IN       SMALLINT,
     ValueType            IN       SMALLINT,
     ParameterType        IN       SMALLINT,
     ColumnSize           IN       INTEGER,
     DecimalDigits        IN       SMALLINT,
     ParameterValue       DEF      ANY,
     BufferLength         INOUT    INTEGER,
     StrLen_or_Ind        DEF      INTEGER )
     RETURNS SMALLINT
```

2. Rationale: Delete unnecessary test of descriptor validity

Replace GeneralRule 3) with:

3) Let *APD* be the allocated CLI descriptor area identified by *HV* and let *N* be the value of COUNT for *APD*.

3. Rationale: Editorial: The second and third sentences are separated by a comma instead of a period.

Replace General Rule 7) with:

7) Let *HL* be the standard programming language of the invoking host program. Let *operative data type correspondence table* be the data type correspondence table for *HL* as specified in Subclause 5.4, "Data type correspondences". Refer to the two columns of the operative data type correspondence table as the *SQL data type column* and the *host data type column*.

4. Rationale: The following changes make it possible to invoke PSM procedures from CLI.

Replace General Rule 13) with:

13) Let *SC* be the value of ColumnSize, let *DD* be the value of DecimalDigits, and let *BL* be the value of BufferLength.

5. Rationale: Correct computation of a dynamic parameters interval leading field precision.

Replace General Rule 14) b) vi) with:

vi) Case:

1) If *DIC* indicates SECOND, DAY TO SECOND, HOUR TO SECOND, or MINUTE TO SECOND, then the interval leading precision of the <dynamic parameter specification> described by *IDA1* is set to *CS* - *IL* - *DD* - *DP* - 1.

2) Otherwise, the internval leading field precision of the <dynamic parameter specification> described by *IDA1* is set to *CS* - *IL* - 1.

6. Rationale: The following changes make it possible to invoke PSM procedures from CLI.

Insert the following General Rules:

15.1) Let *IOM* be the value of InputOutputMode.

15.2) If *IOM* is not one of the code values in Table 9.1, "Code associated with <parameter mode> in SQL/CLI", then an exception condition is raised: *CLI-specific condition — invalid parameter mode*.

15.3) The parameter mode of the <dynamic parameter specification> described by *IDA2* is set to *IOM*.

Replace General Rule 19) with:

19) The address of the host variable that is to define the length (in octets) of the <dynamic parameter specification> value described by *IDA2* is set to the address of StrLen_or_Ind.

Insert the following General Rules:

19.1) If *IOM* is PARAM MODE OUT or PARAM MODE INOUT and *BL* is not greater than zero, then an exception condition is raised: *CLI-specific condition — invalid string length or buffer length*.

19.2) The length in octets of the <dynamic parameter specification> value described by *IDA2* is set to *BL*.

6.9 ColAttribute

1. Rationale: General Rule 7) a) uses "prepared statement is not a cursor specification" as dynamic SQL error. But this dynamic SQL error is not supported in SQL92.

Replace "prepared statement is not a cursor specification" with "prepared statement not a cursor specification". And amend the description in Index 7 (page 7).

6.13 DescribeCol

1. *Rationale: Editorial: In the Definition the parameters ColName and BufferLength are not separated by a comma.*

Replace the Definition with:

```
DescribeCol (
     StatementHandle    IN     INTEGER,
     ColumnNumber       IN     SMALLINT,
     ColumnName         OUT    CHARACTER(L),
     BufferLength       IN     SMALLINT,
     NameLength         OUT    SMALLINT,
     DataType           OUT    SMALLINT,
     ColumnSize         OUT    INTEGER,
     DecimalDigits      OUT    SMALLINT,
     Nullable           OUT    SMALLINT )
     RETURNS SMALLINT
```

6.17 ExecDirect

1. *Rationale: The Definition defines the data type of the TextLength parameter as "SMALLINT". It should be "INTEGER".*

In the Definition change the data type of the TextLength parameter from "SMALLINT" to "INTEGER".

2. *Rationale: Add support for PSM output parameters*

Add a new General Rule 11) b) iii.1) as follows:

iii.1) If *P* is a <call statement>, then the General Rules of Subclause 5.3.3, ``Implicit using clause", are applied to 'CALL', *P*, *S*, as TYPE, SOURCE, and ALLOCATED STATEMENT, respectively.

6.18 Execute

1. *Rationale: Add support for PSM output parameters*

Add a new General Rule 5) b) ii.1) as follows:

ii.1) If *P* is a <call statement>, then the General Rules of Subclause 5.3.3, "Implicit using clause", are applied to 'CALL', *P*, *S*, as TYPE, SOURCE, and ALLOCATED STATEMENT, respectively.

6.24 FreeStmt

1. *Rationale: The Definition defines the data type of the Option parameter as "INTEGER". It should be "SMALLINT".*

In the Definition change the data type of the Option parameter from "INTEGER" to "SMALLINT".

6.27 GetData

1. *Rationale: In General Rule 2) b), "StringLength" is used for a parameter of this routine, but in the definition of this clause, that is not defined.*

Omit ", StringLength," in General Rule 2) b).

2. *Rationale: Editorial.*

Replace "S" with "SC" in General Rule 18) b).

6.30 GetDiagField

1. *Rationale: Editorial: General Rule 11) f) is a case but the word "Case" is missing.*

Replace General Rule 11) f) with:

f) If *DI* indicates MORE, then the value retrieved is:

Case:

i) If more conditions were raised during execution of *R* than have been stored in the diagnostics area, then 'Y'.

ii) If all the conditions that were raised during execution of *R* have been stored in the diagnostics area, then 'N'.

2. *Rationale: Editorial: General Rule 12) i) is a case but the word "Case" is missing. Also General Rule 12) i) ii) 3) references TABLE_NAME, CATALOG_NAME and SCHEMA_NAME instead of the values of these items.*

Replace General Rule 12) i) with:

i) If *DI* indicates CURSOR_NAME, CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME, CATALOG_NAME, SCHEMA_NAME, TABLE_NAME, or COLUMN_NAME, then the values retrieved are:

Case:

i) If the value of SQLSTATE corresponds to *warning — cursor operation conflict*, then the value of CURSOR_NAME is the name of the cursor that caused the completion condition to be raised.

ii) If the value of SQLSTATE corresponds to *integrity constraint violation*, *transaction rollback — integrity constraint violation*, or *triggered data change violation*, then:

1) The values of CONSTRAINT_CATALOG and CONSTRAINT_SCHEMA are the <catalog name> and the <unqualified schema name> of the <schema name> of the schema containing the constraint or assertion. The value of CONSTRAINT_NAME is the <qualified identifier> of the constraint or assertion.

2) Case:

A) If the violated integrity constraint is a table constraint, then the values of CATALOG_NAME, SCHEMA_NAME, and TABLE_NAME are the <catalog name>, the <unqualified schema name> of the <schema name>, and the <qualified identifier> or <local table name>, respectively, of the table in which the table constraint is contained.

B) If the violated integrity constraint is an assertion and if only one table referenced by the assertion has been modified as a result of executing the SQL-statement, then the values of CATALOG_NAME, SCHEMA_NAME, and TABLE_NAME are the <catalog name>, the <unqualified schema name> of the <schema name>, and the <qualified identifier> or <local table name>, respectively, of the modified table.

C) Otherwise, the values of CATALOG_NAME, SCHEMA_NAME, and TABLE_NAME are <space>s.

3) If the value of TABLE_NAME identifies a declared local temporary table, then the value of CATALOG_NAME is <space>s and the value of SCHEMA_NAME is 'MODULE'.

iii) If the value of SQLSTATE corresponds to *syntax error or access rule violation*, then:

1) The values of CATALOG_NAME, SCHEMA_NAME, and TABLE_NAME are the <catalog name>, the <unqualified schema name> of the <schema name> of the schema that contains the table that caused the syntax error or the access rule violation and the <qualified identifier> or <local table name>, respectively. If TABLE_NAME refers to a declared local temporary table, then CATALOG_NAME is <space>s and SCHEMA_NAME contains "MODULE".

2) If the syntax error or the access rule violation was for an inaccessible column, then the value of COLUMN_NAME is the <column name> of that column. Otherwise, the value of COLUMN_NAME is <space>s.

iv) If the value of SQLSTATE corresponds to *invalid cursor state*, then the value of CURSOR_NAME is the name of the cursor that is in the invalid state.

v) If the value of SQLSTATE corresponds to *with check option violation*, then the values of CATALOG_NAME, SCHEMA_NAME, and TABLE_NAME are the <catalog name> and the <unqualified schema name> of the <schema name> of the schema that contains the view that caused the violation of the WITH CHECK OPTION, and the <qualified identifier> of that view, respectively.

vi) If the value of SQLSTATE does not correspond to *syntax error or access rule violation*, then:

1) If the values of CATALOG_NAME, SCHEMA_NAME, TABLE_NAME, and COLUMN_NAME identify a column for which no privileges are granted to the current <authorization identifier>, then the value of COLUMN_NAME is replaced by a zero-length string.

2) If the values of CATALOG_NAME, SCHEMA_NAME, and TABLE_NAME identify a table for which no privileges are granted to the current <authorization identifier>, then the values of CATALOG_NAME, SCHEMA_NAME, and TABLE_NAME are replaced by a zero-length string.

3) If the values of CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, and CONSTRAINT_NAME identify a <table constraint> for some table *T* and if no privileges

for *T* are granted to the current <authorization identifier>, then the values of CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, and CONSTRAINT_NAME are replaced by a zero-length string.

4) If the values of CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, and CONSTRAINT_NAME identify an assertion contained in some schema *S* and if the owner of *S* is not the <authorization identifier> of the current SQL-session, then the values of CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, and CONSTRAINT_NAME are replaced by a zero-length string.

3. *Rationale: SQL/PSM [2] defines a new SQL diagnostics field CONDITION_IDENTIFIER which contains the <condition name> of a user-defined exception when the RETURNED_SQLSTATE field corresponds to unhandled user-defined exception. This paper proposes to add support for this new diagnostics field to SQL/CLI.*

Add the following General Rule 12) j.1):

j.1) If DI indicates CONDITION_IDENTIFIER, then the value retrieved is:

Case:

i) If the value of RETURNED_SQLSTATE corresponds to unhandled user-defined exception, then the <condition name> of the user-defined exception.

ii) Otherwise, a zero-length string.

6.31 GetDiagRec

1. *Rationale: Editorial: General Rule 9) c) ii) used an undefined term V instead of the defined term MT.*

Replace General Rule 9) c) ii) with:

ii) Otherwise, an implementation-defined character string is retrieved. Let *MT* be the implementation-defined character string that is retrieved and let *L* be the length in octets of *MT*. If *BL* is not greater than zero, then an exception condition is raised: *CLI-specific condition — invalid string length or buffer length*. If TextLength is not a null pointer, then TextLength is set to *L*.

Case:

1) If null termination is *false* for the current SQL-environment, then:

A) If *L* is not greater than *BL*, then the first *L* octets of MessageText are set to *MT* and the values of the remaining octets of MessageText are implementation-dependent.

B) Otherwise, MessageText is set to the first *BL* octets of *MT*.

2) Otherwise, let *k* the number of octets in a null terminator in the character set of MessageText and let the phrase "implementation-defined null character that terminates a C character string" imply *k* octets, all of whose bits are zero.

A) If *L* is not greater than (*BL*-*k*), then the first (*L*+*k*) octets of MessageText are set to *MT* concatenated with a single implementation-defined null character that terminates a C

character string. The values of the remaining characters of MessageText are implementation-dependent.

B) Otherwise, MessageText is set to the first (*BL*-*k*) octets of *MT* concatenated with a single implementation-defined null character that terminates a C character string.

NOTE 31 – An implementation may provide <space>s or a zero-length string or a character string that describes the status condition.

6.34 GetInfo

1. *Rationale: Provide a mechanism for obtaining unbound output parameter data.*

Insert the following new subrule into General Rule 9):

p+1) If *IT* indicates GETPARAMDATA EXTENSIONS, then *V* is set as follows to indicate whether the implementation supports certain extensions to the GetParamData routine:

i) Let *V* be 0.

ii) If GetParamData can be called to obtain parameters that precede the last bound output parameter, then add the numeric value for ANY PARAMETER from Table 29.1, "Values for GETPARAMDATA EXTENSIONS with GetInfo", to *V*.

iii) If GetParamData can be called for parameters in any order, then add the numeric value for ANY ORDER from Table 29.1, "Values for GETPARAMDATA EXTENSIONS with GetInfo", to *V*.

NOTE 33.1 - This also means that a parameter can be accessed by GetParamData even though previous calls retrieved all the data for that parameter.

2. *Rationale: When zero is returned for the InfoTypes:*
a) MAXIMUM COLUMNS IN GROUP BY,
b) MAXIMUM COLUMNS IN ORDER BY,
c) MAXIMUM COLUMNS IN SELECT,
d) MAXIMUM COLUMNS IN TABLE,
the appropriate General Rule states returns zero if there is no maximum. This value should also be returned if the maximum cannot be determined.

In the General Rules 9) r), 9) s), 9) t) and 9) u) replace the sentence "If there is no maximum, then let *V* be 0." with "If there is no maximum or if the maximum cannot be determined, then let *V* be 0.".

3. *Rationale: Define maximum statement lengths in octets instead of characters and also return zero if the maximum cannot be determined.*

Replace General Rule 9) v) with:

v) If *IT* indicates MAXIMUM STATEMENT OCTETS, then let *V* be the maximum number of octets that *SS* supports for an <SQL statement variable>. If there is no maximum or if the maximum cannot be determined, then let *V* be 0.

v.1) If *IT* indicates MAXIMUM STATEMENT OCTETS DATA, then let *V* be the maximum number of octets that *SS* supports for an <SQL data statement>. If there is no maximum or if the maximum cannot be determined, then let *V* be 0.

v.2) If *IT* indicates MAXIMUM STATEMENT OCTETS SCHEMA, then let *V* be the maximum number of octets that *SS* supports for an SQL <schema definition>. If there is no maximum or if the maximum cannot be determined, then let *V* be 0.

4. *Rationale: When zero is returned for the InfoTypes:*
   a) MAXIMUM TABLES IN SELECT, or
   b) MAXIMUM USER NAME LENGTH
   the appropriate General Rule states returns zero if there is no maximum. This value should also be returned if the maximum cannot be determined.

In the General Rules 9)w) and 9)x) replace the sentence "If there is no maximum, then let *V* be 0." with "If there is no maximum or if the maximum cannot be determined, then let *V* be 0.".

6.34.1 GetParamData

1. *Rationale: Provide a mechanism for obtaining unbound output parameter data.*

Insert this new subclause:

6.34.1 GetParamData

Function

Retrieve the value of a dynamic output parameter.

Definition

```
GetParamData (
      StatementHandle       IN    INTEGER,
      ParameterNumber       IN    SMALLINT,
      TargetType            IN    SMALLINT,
      TargetValue           OUT   ANY,
      BufferLength          IN    INTEGER,
      StrLen_or_Ind         OUT   INTEGER )
    RETURNS SMALLINT
```

General Rules

1) Let *S* be the allocated SQL-statement identified by StatementHandle.

2) If there is no executed statement associated with *S*, then an exception condition is raised: *CLI-specific condition — function sequence error*. Otherwise, let *P* be the statement that was prepared.

3) If *P* is not a <call statement>, then an exception condition is raised: *CLI-specific condition* — function sequence error.

4) Let *APD* be the current application parameter descriptor for *S* and let *N* be the value of COUNT for *APD*.

5) If *N* is less than zero, then an exception condition is raised: *dynamic SQL error — invalid descriptor count*.

6) Let *PN* be the value of ParameterNumber.

7) If *PN* is less than 1 or greater than *N* , then an exception condition is raised: *dynamic SQL error — invalid descriptor index.*

8) If DATA_POINTER is non-zero for at least one of the first *N* item descriptor areas of *APD*, then let *BPN* be the parameter number associated with such an item descriptor area and let *HBPN* be the value of MAX(*BPN*). Otherwise, let *HBPN* be zero.

9) Let *IDA* be the item descriptor area of *APD* specified by *PN* and let *IDA1* be the item descriptor area of *IPD* specified by *PN*.

10) Let *PM* be the value of PARAMETER_MODE in *IDA1*.

11) If *PM* is PARAM MODE IN, then an exception condition is raised: *dynamic SQL error — invalid descriptor index*.

12) If *PN* is not greater than *HBPN* , then
Case:
a) If DATA_POINTER for *IDA* is not zero, then an exception condition is raised: *dynamic SQL error — invalid descriptor index.*
b) If DATA_POINTER for *IDA* is zero, then it is implementation-defined whether an exception condition is raised: *dynamic SQL error — invalid descriptor index.*

NOTE 33.2 - This implementation-defined feature determines whether parameters before the highest bound parameter can be accessed by GetParamData.

13) If there is a *fetched parameter number* associated with *S*, then let *FPN* be that parameter number; otherwise, let *FPN* be zero.

Note 33.3 - *fetch parameter number* is the ParameterNumber value used with the previous invocation (if any) of the GetParamData routine with *S*. See General Rules 24 and 25 where this value is set.

14) Case:
a) If *FPN* is greater than zero and *PN* is not greater than *FPN*, then it is implementation-defined whether an exception condition is raised: *dynamic SQL error — invalid descriptor index.*

NOTE 33.4 - This implementation-defined feature determines whether GetParamData can only access parameters in ascending parameter number order.

b) If *FPN* is less than zero, then:

i) Let *AFPN* be the absolute value of *FPN*.

ii) Case:

1) If *PN* is less than *AFPN* , then it is implementation-defined whether an exception condition is raised: *dynamic SQL error — invalid descriptor index.*

NOTE 33.5 - This implementation-defined feature determines whether GetData can only access parameters in ascending parameter number order.

2) If *PN* is greater than *AFPN*, then let *FPN* be *AFPN*.

15) Let *T* be the value of TargetType.

16) Let *HL* be the standard programming language of the invoking host program. Let operative data type correspondence table be the data type correspondence table for *HL* as specified in Subclause 5.4,