



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 — Langues 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.

Contents

Page

ISO/IEC 9075:1992

Database Languages - SQL	7
Introduction	7
2 Normative references	7
3.3.4.3 Terms denoting rule requirements	8
4.2 Character strings	8
4.2.1 Character strings and collating sequences	8
4.2.3 Rules determining collating sequence usage	9
4.3.1 Bit string comparison and assignment	9
4.4 Numbers	9
4.4.1 Characteristics of numbers	9
4.5 Datetimes and intervals	10
4.5.1 Datetimes	10
4.5.2 Intervals	11
4.5.3 Operations involving datetimes and intervals	11
4.6 Type conversion and mixing of data types	12
4.8 Columns	12
4.9 Tables	12
4.10.2 Table constraints	13
4.18.1 Status parameters	13
4.21 Cursors	13
4.22.6 SQL-statements and transaction states	14
4.24 SQL dynamic statements	14
4.26 Privileges	14
4.28 SQL-transactions	14
4.29 SQL-connections	14
4.31 Client-server operation	15
5.2 <token> and <separator>	15
5.3 <literal>	16
5.4 Names and identifiers	17
6.1 <data type>	17
6.2 <value specification> and <target specification>	18
6.3 <table reference>	18
6.4 <column reference>	19
6.5 <set function specification>	20
6.6 <numeric value function>	20
6.7 <string value function>	21
6.8 <datetime value function>	21
6.10 <cast specification>	22
6.11 <value expression>	25
6.12 <numeric value expression>	25
6.14 <datetime value expression>	25
6.15 <interval value expression>	27
7.1 <row value constructor>	28
7.4 <from clause>	29
7.5 <joined table>	30
7.6 <where clause>	31
7.7 <group by clause>	31
7.8 <having clause>	32
7.9 <query specification>	33
7.10 <query expression>	35
7.11 <scalar subquery>, <row subquery>, and <table subquery>	36

8.2 <comparison predicate>	36
8.3 <between predicate>	36
8.7 <quantified comparison predicate>	37
8.11 <overlaps predicate>	37
9.1 Retrieval assignment	37
9.2 Store assignment	38
9.3 Set operation result data types	39
10.1 <interval qualifier>	40
10.2 <language clause>, Table 16	41
10.4 <character set specification>	41
10.5 <collate clause>	41
10.6 <constraint name definition> and <constraint attributes>	41
11.1 <schema definition>	42
11.2 <drop schema statement>	42
11.4 <column definition>	42
11.5 <default clause>	42
11.6 <table constraint definition>	44
11.8 <referential constraint definition>	45
11.9 <check constraint definition>	45
11.11 <add column definition>	45
11.15 <drop column definition>	46
11.16 <add table constraint definition>	47
11.17 <drop table constraint definition>	47
11.18 <drop table statement>	48
11.19 <view definition>	48
11.21 <domain definition>	49
11.25 <add domain constraint definition>	49
11.26 <drop domain constraint definition>	50
11.27 <drop domain statement>	50
11.28 <character set definition>	50
11.29 <drop character set statement>	51
11.30 <collation definition>	51
11.31 <drop collation statement>	51
11.32 <translation definition>	52
11.34 <assertion definition>	53
11.36 <grant statement>	53
11.37 <revoke statement>	55
12.3 <procedure>	60
12.4 Calls to a <procedure>	61
12.5 <SQL procedure statement>	71
13.1 <declare cursor>	72
13.2 <open statement>	72
13.3 <fetch statement>	72
13.4 <close statement>	73
13.5 <select statement: single row>	73
13.6 <delete statement: positioned>	74
13.7 <delete statement: searched>	74
13.8 <insert statement>	74
13.9 <update statement: positioned>	75
13.10 <update statement: searched>	75
13.11 <temporary table declaration>	76
14.1 <set transaction statement>	76
14.2 <set constraints mode statement>	76
14.3 <commit statement>	76

15.1 <connect statement>	76
15.2 <set connection statement>	77
15.3 <disconnect statement>	77
16.5 <set local time zone statement>	77
17.1 Description of SQL item descriptor areas	77
17.2 <allocate descriptor statement>	77
17.3 <deallocate descriptor statement>	78
17.4 <get descriptor statement>	78
17.5 <set descriptor statement>	78
17.6 <prepare statement>	78
17.9 <using clause>	82
17.10 <execute statement>	82
17.11 <execute immediate statement>	82
17.15 <dynamic fetch statement>	83
17.18 <dynamic update statement: positioned>	83
17.19 <preparable dynamic delete statement: positioned>	83
17.20 <preparable dynamic update statement: positioned>	84
18.1 <get diagnostics statement>	84
19.1 <embedded SQL host program>	86
19.3 <embedded SQL Ada program>	86
19.5 <embedded SQL COBOL program>	86
20.1 <direct SQL statement>	86
21.1 Introduction	87
21.2.2 INFORMATION_SCHEMA_CATALOG_NAME base table	87
21.2.3 INFORMATION_SCHEMA_CATALOG_NAME_CARDINALITY assertion	88
21.2.4 SCHEMATA view	88
21.2.5 DOMAINS view	88
21.2.6 DOMAIN_CONSTRAINTS view	88
21.2.9 COLUMNS view	89
21.2.17 ASSERTIONS view	89
21.2.23 CONSTRAINT_TABLE_USAGE view	89
21.2.24 CONSTRAINT_COLUMN_USAGE view	90
21.2.27 SQL_IDENTIFIER domain	91
21.3.5 DATA_TYPE_DESCRIPTOR base table	91
21.3.6 DOMAINS base table	94
21.3.8 TABLES base table	94
21.3.10 COLUMNS base table	95
21.3.11 VIEW_TABLE_USAGE base table	96
21.3.12 VIEW_COLUMN_USAGE base table	96
21.3.13 TABLE_CONSTRAINTS base table	97
21.3.15 REFERENTIAL_CONSTRAINTS base table	97
21.3.17 CHECK_TABLE_USAGE base table	97
21.3.18 CHECK_COLUMN_USAGE base table	98
21.3.21 COLUMN_PRIVILEGES base table	98
21.3.22 USAGE_PRIVILEGES base table	99
21.3.23 CHARACTER_SETS base table	99
21.3.24 COLLATIONS base table	99
21.3.25 TRANSLATIONS base table	99
21.3.26 SQL_LANGUAGES base table	100
22.1 SQLSTATE	100
22.2 SQLCODE	102
22.3 Remote Database Access SQLSTATE Subclasses	102
23.2 Claims of Conformance	104
23.3 Extensions and options	105
A.1 Intermediate SQL Specifications	105

A.2 Entry SQL Specifications	107
Annex B: Implementation-defined elements	110
Annex C Implementation-dependent elements	111
Annex E Incompatibilities with ISO/IEC 9075:1989	112
Annex F Maintenance and interpretation of SQL	113
ISO/IEC 9075-3:1995	
Database Languages - SQL-Part 3:Call-Level Interface (SQL/CLI)	
.....	115
Contents	115
4.1 Introduction	116
4.3 Diagnostics areas	116
4.4.7 CLI descriptor areas	116
5.1 <CLI routine>	116
5.2 <CLI routine> invocation	117
5.3 SQL/CLI common elements	117
5.3.3 Implicit using clause	118
5.3.4 Character string retrieval	122
5.3.7 CLI-specific status codes	122
5.3.8 Description of CLI item descriptor areas	122
5.3.9 <CLI routine>	124
5.4 Data type correspondences	125
6.4 AllocStmt	125
6.5 BindCol	126
6.6 BindParam	126
6.9 ColAttribute	128
6.13 DescribeCol	128
6.17 ExecDirect	128
6.18 Execute	128
6.24 FreeStmt	129
6.27 GetData	129
6.30 GetDiagField	129
6.31 GetDiagRec	131
6.34 GetInfo	132
6.34.1 GetParamData	133
6.36 GetTypeInfo	137
6.38 ParamData	138
6.39 Prepare	139
6.40 PutData	139
6.41 RowCount	139
6.38 ParamData	140
6.42 SetConnectAttr	140
6.43 SetCursorName	140
6.44 SetDescField	141
6.45 SetDescRec	143
A.1 C Header File SQLCLI.H	143
A.2 COBOL Library Item SQLCLI	148
B.1 Create table, insert, select	150
B.2 Interactive Query	152
B.3 Providing long dynamic arguments at Execute() time	153
Annex C	154
Annex D Implementation-dependent elements	154
Index	154

ISO/IEC 9075-4:1996

Database Languages - SQL-Part 4:Persistent Stored Modules
(SQL/PSM)

.....	155
2 Normative references	155
3.3 Conventions	155
6.2 <column reference>	155
6.3 <item reference>	155
6.6 <value expression>	156
7.3 <query specification>	156
9.1 <routine invocation>	157
10.3 <default clause>	157
10.7 <drop table statement>	158
10.4 <check constraint definition>	158
10.8 <view definition>	158
10.9 <drop view statement>	158
10.13 <drop translation statement>	159
10.16 <SQL-server module definition>	159
10.18 <SQL-invoked routine>	159
10.20 <grant statement>	159
10.21 <revoke statement>	160
11.4 <SQL procedure statement>	162
12.2 <open statement>	162
13.2 <return statement>	163
13.3 <compound statement>	163
13.9 <if statement>	163
13.14 <for statement>	163
17.3 <resignal statement>	164
19.1.5 ROUTINES view	164
19.2.4 ROUTINES base table	165
Annex A	165
Annex B	167

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. Simplify 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 and 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, coercibility 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 specification>.
- 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, SLT₁ and SLT₂ 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*-*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-determinism.*

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 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*.

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*.