ISO/IEC JTC 1/SC 32 N 1208

Date: 2004-12-16

REPLACES: --

ISO/IEC JTC 1/SC 32

Data Management and Interchange

Secretariat: United States of America (ANSI)
Administered by Pacific Northwest National Laboratory on behalf of ANSI

DOCUMENT TYPE	Final Text Submitted for COR Publication
TITLE	Technical Corrigenda – ISO/IEC 9075-02 Information technology - Database Languages - SQL - Part 2: Foundation (SQL/Foundation)
SOURCE	SC 32 Secretariat
PROJECT NUMBER	1.32.03.05.99.00
STATUS	This is sent to ITTF for publication
REFERENCES	
ACTION ID.	ITTF
REQUESTED ACTION	
DUE DATE	
Number of Pages	137
LANGUAGE USED	English
DISTRIBUTION	P & L Members SC Chair WG Conveners and Secretaries

Douglas Mann, Secretariat, ISO/IEC JTC 1/SC 32

Pacific Northwest National Laboratory *, 13667 Legacy Circle Apt H, Herndon, VA, 20171, United States of America

Telephone: +1 202-566-2126; Facsimile; +1 202-566-1639; E-mail: MannD@battelle.org

available from the JTC 1/SC 32 WebSite http://www.jtc1sc32.org/

*Pacific Northwest National Laboratory (PNL) administers the ISO/IEC JTC 1/SC 32 Secretariat on behalf of ANSI

ISO/IEC JTC 1/SC 32

Date: 2004-12-10

COR ISO/IEC 9075-02:2004 (E)

ISO/IEC JTC 1/SC 32/WG 3

Nederlands Normalisatie Instituut (NNI)

Information technology — Database languages — SQL —

Part 2:
Foundation (SQL/Foundation)

TECHNICAL CORRIGENDUM 1

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

Partie 2: Fondations (SQL/Fondations)

RECTIFICATIE TECHNIQUE 1

Document type: Corridenga
Document subtype: Technical Corrigendum (COR)
Document stage: (5) IS Publication
Document language: English

Statement of purpose for rationale:

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

Notes on numbering:

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

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

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

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

Contents

Page

Tables

Table

Page

Information technology — Database languages — SQL —

Part 2:
Foundation (SQL/Foundation)

TECHNICAL CORRIGENDUM 1

Foreword

1. *Rationale: Correct intent of this second edition.*

   Replace the 6th paragraph with:

   This second edition cancels and replaces the first editions of ISO/IEC 9075-2:1999 and ISO/IEC 9075-5:1999, which have been technically revised. It also incorporates the amendments ISO/IEC 9075-2:1999/Amd.1:2001 and ISO/IEC 9075-5:1999/Amd.1:2001 and the Technical Corrigenda ISO/IEC 9075-2:1999/Cor.1:2001, ISO/IEC 9075-2:1999/Amd.1:2001/Cor.1:2003, ISO/IEC 9075-2:1999/Cor.2:2001, ISO/IEC 9075-5:1999/Cor.1:2001, ISO/IEC 9075-5:1999/Amd.1:2001/Cor.1:2003 and ISO/IEC 9075-5:1999/Cor.2:2001. As a result of the revision some material has been separated out and now forms ISO/IEC 9075-11:2003.

2. *Rationale: Remove incorrect reference to obsolete part.*

   In the 7th paragraph, delete the 5th bullet.

2 Normative references

2.1 JTC1 standards

1. *Rationale: Correct references to IS rather than FCD documents.*

   Replace the references [Framework] and [Schemata] with:

   [Framework] ISO/IEC 9075-1:2003, *Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework).*

   [Schemata] ISO/IEC 9075-11:2003, *Information technology — Database languages — SQL — Part 11: Information and Definition Schemas (SQL/Schemata)*

3 Definitions, notations, and conventions

3.1 Definitions

3.1.6 Definitions provided in Part 2

1. *Rationale: Editorial.*

   Replace the definition of "assignable" with:

   3.1.6.1 **assignable (of types, taken pairwise):** The characteristic of a data type *T1* that permits a value of *T1* to be assigned to a site of a specified data type *T2* (where *T1* and *T2* may be the same data type).

2. *Rationale: Restore the correct definition of "assignment".*

   Replace the definition of "assignment" with:

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

3. *Rationale: Remove erroneous word "comparable" from definition.*

   Replace the definition of "identical" with:

   3.1.6.15 **identical (of a pair of values):** Indistinguishable, in the sense that it is impossible, by any means specified in ISO/IEC 9075, to detect any difference between them. For the full definition, see Subclause 9.8, "Determination of identical values".

4. *Rationale: Definitions required for correction to dynamic result sets.*

   Insert the following definitions:

   3.1.6.29.1 **result set:** A sequence of rows brought into existence by opening a cursor and ranged over by that cursor.

   3.1.6.29.2 **result set sequence:** A sequence of returned result sets.

   3.1.6.29.3 **returned result set:** A result set created during execution of an SQL-invoked procedure and not destroyed when that execution terminates. Such a result set can be accessed by using a cursor other than the one that brought it into existence.

   3.1.6.42.1 **with-return cursor:** A cursor that, when opened, creates a result set that is capable of becoming a returned result set. The WITH RETURN option of <declare cursor> and <allocate cursor statement> specifies a with-return cursor.

4 Concepts

4.13 Columns, fields and attributes

1. *Rationale: Missing item in the column descriptor.*

   In the 6th paragraph, insert the following item to the list of items in a column descriptor:

   — An indication of whether *C* is updatable or not.

4.14 Tables

4.14.2 Types of tables

1. *Rationale: Editorial.*

   Replace the 1st paragraph with:

   A table is either a base table, a derived table, a transient table, or a viewed table. A base table is either a persistent base table, a global temporary table, a created local temporary table, or a declared local temporary table.

2. *Rationale: Delete obsolete reference to element type of a <query expression>.*

   Replace the 3rd paragraph with:

   A derived table is a table derived directly or indirectly from one or more other tables by the evaluation of a <query expression>.

3. *Rationale: Editorial.*

   Append the following text to the 4th paragraph:

   Base tables and views are identified by <table name>s. The same <table name>, in its fully qualified form, cannot be used for both a base table and a view.

4. *Rationale: Define view components, for use in view component privilege descriptors.*

   Insert the following paragraph after the 4th paragraph:

   A <query specification>, <table value constructor>, <explicit table>, or <query expression> contained in a <view definition> is called a *view component*.

4.14.3 Table descriptors

1. *Rationale: Track simple updatability in derived table descriptors.*

   In the 5th paragraph, insert the following item to the list of items in a derived table descriptor:

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

4.17 Integrity constraints

4.17.1 Overview of integrity constraints

1. *Rationale: Every constraint descriptor is to include a <search condition>.*

   Insert the following at the end of the 1st paragraph:

   — An applicable <search condition>.

   NOTE 28.1 — The applicable <search condition> included in the descriptor is not necessarily the <search condition> that might be contained in the SQL-statement whose execution brings the constraint descriptor into existence. The General Rules for the SQL-statement in question specify the applicable <search condition> to be included in the constraint descriptor, in some cases deriving it from a given <search condition>. For example, the syntax for table constraints allows universal quantification over the rows of the table in question to be implicit; in the applicable <search condition> included in the descriptor, that universal quantification is made explicit, to allow for uniform treatment of all types of constraint.

4.17.2 Checking of constraints

1. *Rationale: Clarify when constraints are checked.*

   Replace the text of the Subclause with:

   Every constraint is either *deferrable* or *non-deferrable*. Within an SQL-transaction, every constraint has a constraint mode; if a constraint is nondeferrable, then its constraint mode is always *immediate*; otherwise, it is either immediate or *deferred*. Every constraint has an initial constraint mode that specifies the constraint mode for that constraint at the start of each SQL-transaction and immediately after definition of that constraint. If a constraint is deferrable, then its constraint mode within the current SQL-transaction may be changed (from immediate to deferred, or from deferred to immediate) by execution of a <set constraints mode statement>.

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

or schemas resulting from executing that statement are canceled. (See the General Rules of Subclause 13.5, "<SQL procedure statement>".)

NOTE 29 — This includes SQL-statements that are executed as a direct result or an indirect result of executing a different SQL-statement. It also includes statements whose effects explicitly or implicitly include setting the constraint mode to immediate.

The constraint mode can be set to immediate either explicitly by execution of a <set constraints mode statement>, or implicitly at the end of the current SQLtransaction.

When a <commit statement>is executed, all constraints are effectively checked and, if any constraint is not satisfied, then an exception condition is raised and the SQL-transaction is terminated by an implicit <rollback statement>.

4.17.3 Table constraints

1. *Rationale: Every constraint descriptor includes a <search condition>.*

   Replace the 1st paragraph with:

   A constraint whose definition is part of some base table definition is a *table constraint*. Being part of a particular table definition allows for convenient syntactic shorthands in which universal quantification over the rows of the table in question is implied.

   A table constraint is either a unique constraint, a referential constraint, or a table check constraint. A table constraint descriptor is either a unique constraint descriptor, a referential constraint descriptor, or a table check constraint descriptor, respectively.

2. *Rationale: Every constraint descriptor includes a <search condition>.*

   Delete the 11th and 12th paragraphs

4.17.4 Domain constraints

1. *Rationale: Every constraint descriptor includes a <search condition>, and clarify how domain constraints are applied.*

   Replace the text of the subclause with:

   A domain constraint is a constraint that is specified for a domain. It is applied to all columns that are based on that domain, and to all values cast to that domain.

   A domain constraint is described by a domain constraint descriptor. In addition to the components of every constraint descriptor, a domain constraint descriptor includes:

   — The template <search condition> for the generation of domain constraint usage <search condition>s.

   — A possibly empty set of domain constraint usages.

A domain constraint usage descriptor is created implicitly by the evaluation of a <column definition> whose <data type or domain name> is a <domain name>. If *C* is such a column and *D* is the domain identified by the <domain name>, then every domain constraint *DC* defined for *D* implies a domain constraint usage, to the effect that each value in *C* satisfies *DC*.

In addition to the components of every constraint descriptor, a domain constraint usage descriptor includes:

— The name of the applicable column.

— The applicable <search condition> that evaluates whether each value in *C* satisfies *DC*.

A domain constraint is satisfied by SQL-data if and only if, for every table *T* that has a column named *C* based on that domain, the applicable <search condition> recorded in the appropriate domain constraint usage evaluates to *True* or *Unknown*.

A domain constraint is satisfied by the result of a <cast specification> if and only if the specified template <search condition>, with each occurrence of the <general value specification> VALUE replaced by that result, evaluates to *True* or *Unknown*.

4.17.5 Assertions

1. *Rationale: Every constraint descriptor includes a <search condition>.*

   Replace the text of the subclause with:

   An assertion is a constraint whose descriptor is an independent schema component not included in any table descriptor.

4.18 Functional dependencies

4.18.6 Known functional dependencies in a <joined table>

1. *Rationale: Rules for inheriting funtional dependencies in <joined table> were omitted.*

   Insert the following after the 5th paragraph:

   If $A \mapsto B$ is a known functional dependency in *T1*, *CA* is the counterpart of *A* in *R*, and *CB* is the counterpart of *B* in *R*, then $CA \mapsto CB$ is a known functional dependency in *R* when one of the following is true:

   — CROSS, INNER, or LEFT is specified.

   — RIGHT or FULL is specified and at least one column in *A* is known not nullable.

   If $A \mapsto B$ is a known functional dependency in *T2*, *CA* is the counterpart of *A* in *R*, and *CB* is the counterpart of *B* in *R*, then $CA \mapsto CB$ is a known functional dependency in *R* when one of the following is true:

   — CROSS, INNER, or RIGHT is specified.

— LEFT or FULL is specified and at least one column in *A* is known not nullable.

4.18.15 Known functional dependencies in a <query expression>

1. *Rationale: Remove incorrect reference to <joined table>.*

   Replace the 2nd paragraph with:

   A <query expression> that is a <query term> that is a <query primary> that is a <simple table> is covered by previous Subclauses of this Clause.

4.20 SQL-Schemas

1. *Rationale: Editorial.*

   Replace the 5th paragraph with:

   Base tables and views are identified by <table name>s. A <table name> consists of a <schema name>, followed by a <period>, followed by an <identifier>. The <schema name> identifies the schema that includes the table descriptor of the base table or view identified by the <table name>. The <table name>s of base tables and views defined in different schemas can have equivalent <identifier>s.

   NOTE 44.1 — Equivalence of <identifier>s is defined in Subclause 5.2, "<token> and <separator>".

4.27 SQL-invoked routines

4.27.3 Execution of SQL-invoked routines

1. *Rationale: The treatment of the authorization stack is inconsistent with Subclauses 4.34.1.1 and 10.4.*

   Replace the 1st, 2nd, 3rd, 4th, 5th, 6th and 7th paragraphs with:

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

4.27.4 Routine descriptors

1. *Rationale: Change of terminology.*

   In the 1st paragraph, replace the 8th bullet with:

   — If the SQL-invoked routine is an SQL-invoked procedure, then the maximum number of returned result sets.

2. *Rationale: An external routine does not have two authorization identifiers.*

   In the 1st paragraph, in the 15th bulleted item, delete the 5th bulleted subitem ("The external routine authorization identifier ... ").

4.27.5 Result sets returned by SQL-invoked procedures

1. *Rationale: New subclause for result sets returned by SQL-invoked procedures.*

4.27.5 Result sets returned by SQL-invoked procedures

NOTE 48.1 — Subclause 3.1.6, "Definitions provided in Part 2", gives definitions of the terms result set, result set sequence, returned result set, and with-return cursor, used in this Subclause.

An invocation of an SQL-invoked procedure *SIP1* might bring into existence a result set sequence *RSS*. *RSS* consists of the result sets of with-return cursors opened by *SIP1* and remaining open when *SIP1* terminates, placed in the order in which those result sets are created during the execution of *SIP1*.

NOTE 48.2 — If the same cursor is opened more than once during the execution of *SIP1*, then it is the last opening that is considered to create the result set, even if the result set in question is identical to that created by some earlier opening.

RSS is available, by executing a particular form of <allocate cursor statement>, only to the *invoker INV* of *SIP1*. If *SIP1* is invoked by an SQL-invoked procedure *SIP2*, then *INV* is *SIP2*. If *SIP1* is invoked by an externally-invoked procedure *EIP*, then *INV* is the SQL-client module containing *EIP*. Otherwise, *RSS* is not available.

NOTE 48.3 — Only the immediate invoker is considered. For example, if an externally-invoked procedure *EIP* executes a <call statement> invoking an SQL-invoked procedure *SIP3* that invokes *SIP1*, then the result set sequence returned by *SIP1* is available only to *SIP3*, until either *SIP3* returns control to *EIP* or another invocation of *SIP1* by *SIP3* is given before *SIP3* returns. There is no mechanism whereby *SIP3* can return *SIP1*'s result set sequence to the invoker of *SIP3*, even if *SIP3* is defined to be able to return a result set sequence.

The invocation of *SIP1* by *INV* destroys any existing result set sequence that might have arisen from some previous invocation of *SIP1* by *INV*. All result set sequences available to *INV* are destroyed when *INV* terminates.

A returned result set consists of a sequence of rows called the *constituent rows* and an *initial cursor* position.

The constituent rows and initial cursor position of each returned result set *RS* in *RSS* are determined when *SIP1* returns to *INV*. If the with-return cursor *C* for *RS* is scrollable, then the constituent rows of *RS* are

those of the result set of *C* as it exists when *SIP1* terminates; otherwise, the constituent rows are as for scrollable cursors, except that rows preceding the current cursor position of *C* are excluded. If *C* is scrollable, then the initial cursor position of *RS* is the position of *C* when *SIP1* terminates; otherwise, the initial cursor position of *RS* is before the first row.

NOTE 48.4 — The result set of *C* as it exists when *SIP1* terminates might differ from that generated when *C* was opened, if, for example, any <delete statement: positioned>s or <update statement: positioned>s are executed by *SIP1* before it terminates.

The maximum number of returned result sets that may form a result set sequence is specified by the <dynamic result sets characteristic> contained in the <SQL-invoked routine> defining *SIP1*. If the actual number of with-return cursors that remain open when *SIP1* returns is greater than the maximum number of returned result sets specified in the <dynamic result sets characteristic> clause, then a warning condition is raised. It is implementation-dependent whether or not result sets whose positions are greater than that maximum number are returned.

4.32 Cursors

4.32.1 General description of cursors

1. *Rationale: Clarify definition of result sets.*

   Replace the 7th paragraph with:

   A cursor in the open state identifies a result set and a position relative to the ordering of that result set. If the <declare cursor> does not contain an <order by clause>, or contains an <order by clause> that does not specify the order of the rows completely, then the rows of the result set have an order that is defined only to the extent that the <order by clause> specifies an order and is otherwise implementation-dependent.

   NOTE 50.1 — A definition of “result set” is given in Subclause 3.1.6, “Definitions provided in Part 2”.

2. *Rationale: Correct the definition of updatable cursor.*

   Replace the 10th paragraph with:

   A cursor is either *updatable* or *not updatable*. If FOR UPDATE OF is specified for the cursor, or if the table identified by the cursor is simply updatable and FOR READ ONLY, SCROLL, and ORDER BY are not specified for the cursor, then the cursor is updatable; otherwise, the cursor is not updatable. The operations of update and delete are permitted for updatable cursors, subject to constraining Access Rules and Conformance Rules.

4.32.2 Operations on and using cursors

1. *Rationale: Clarify definition of returned result sets.*

   Replace the 11th paragraph with:

A <declare cursor> *DC* that specifies WITH RETURN defines a with-return cursor. The <cursor specification> *CR* contained in *DC* defines a returned result set. A with-return cursor, if declared in an SQL-invoked procedure and in the open state when the procedure returns to its invoker, yields a returned result set that can be accessed by the invoker of the procedure that generates it.

NOTE 50.2 — Definitions of "returned result set" and "with-return cursor" are given in Subclause 3.1.6, "Definitions provided in Part 2".

4.33 SQL-statements

4.33.3 SQL-statements and SQL-data access indication

1. *Rationale: All queries read SQL data, not just <subquery>s.*

   Replace the 2nd paragraph with:

   The following SQL-statements possibly read SQL-data:

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

   — SQL-statements that contain a <query expression> and are not SQL-statements that possibly modify SQL-data.

4.33.4 SQL-statements and transaction states

1. *Rationale: SQL-transaction statements can initiate SQL-transactions.*

   Replace the 1st, 2nd and 3rd paragraphs with:

   The following SQL-statements are transaction-initiating SQL-statements, Thus, if there is no current SQL-transaction, and a statement of this class is executed, then an SQL-transaction is initiated:

   — All SQL-schema statements

   — The following SQL-transaction statements:

   - <start transaction statement>.
   - <savepoint statement>.
   - <commit statement>.
   - <rollback statement>.

   — The following SQL-data statements:

   - <open statement>.

- <close statement>.
- <fetch statement>.
- <select statement: single row>.
- <insert statement>.
- <delete statement: searched>.
- <delete statement: positioned>.
- <update statement: searched>.
- <update statement: positioned>.
- <merge statement>.
- <allocate cursor statement>.
- <dynamic open statement>.
- <dynamic close statement>.
- <dynamic fetch statement>.
- <direct select statement: multiple rows>.
- <dynamic single row select statement>.
- <dynamic delete statement: positioned>.
- <preparable dynamic delete statement: positioned>.
- <dynamic update statement: positioned>.
- <preparable dynamic update statement: positioned>.
- <free locator statement>.
- <hold locator statement>.

— The following SQL-dynamic statements

- <describe input statement>.
- <describe output statement>.
- <allocate descriptor statement>.
- <deallocate descriptor statement>.
- <get descriptor statement>.
- <set descriptor statement>.
- <prepare statement>.
- <deallocate prepared statement>.

With the exception of <start transaction statement>, every transaction-initiating SQL-statement *implicitly initiates a transaction* if there is no current SQL-transaction when it is executed, in which case the execution of the initiating statement is included in the initiated SQL-transaction. In the case of <start transaction statement>, transaction initiation is the primary effect of executing the statement itself.

Whether or not an <execute immediate statement> starts a transaction depends on the content of the <SQL statement variable> referenced by the <execute immediate statement> at the time it is executed. Whether or not an <execute statement> starts a transaction depends on the content of the <SQL statement variable> referenced by the <prepare statement> at the time the prepared statement referenced by the <execute statement> was prepared. In both cases, if the content of the <SQL statement variable> was a transaction-initiating SQL-statement, then the <execute immediate statement> or <execute statement> is treated as a transaction-initiating statement; otherwise, it is not treated as a transaction-initiating statement.

The following SQL-statements are not transaction-initiating SQL-statements, Thus, if there is no current SQL-transaction, then executing a statement of this class does not change that state of affairs.

— All SQL-transaction statements except <start transaction statement>s, <savepoint statement>s, <commit statement>s, and <rollback statement>s.

— All SQL-connection statements.

— All SQL-session statements.

— All SQL-diagnostics statements.

— SQL embedded exception declarations.

— The following SQL-data statements:

  • <temporary table declaration>.

  • <declare cursor>.

  • <dynamic declare cursor>.

  • <dynamic select statement>.

2. *Rationale: All queries need a transaction to have been started, not just <subquery>s.*

   Replace the final (6^{th}) paragraph with:

   If an <SQL control statement> causes the evaluation of a <query expression> and there is no current SQL-transaction, then an SQL-transaction is initiated before evaluation of the <query expression>.

4.34 Basic security model

4.34.1 Authorization identifiers

1. *Rationale: Inappropriate angle-brackets, implementation-dependent mappings of user identifiers are of no possible use, so they must be implementation-defined, and use of database system instead of SQL-implementation.*

   Replace the 1st paragraph with:

   An authorization identifier identifies a set of privileges. An authorization identifier is either a user identifier or a role name. A user identifier represents a user of the SQL-implementation. Any mapping of user identifiers to operating system users is implementation-defined. A role name represents a role.

4.34.1.1 SQL-session authorization identifiers

1. *Rationale: Inappropriate angle-brackets.*

   Replace the 1st paragraph with:

   An SQL-session has a user identifier called the *SQL-session user identifier*. When an SQL-session is initiated, the SQL-session user identifier is determined in an implementation-defined manner, unless the session is initiated using a <connect statement>. The value of the SQL-session user identifier can never be the null value. The SQL-session user identifier can be determined by using SESSION_USER.

4.34.2 Privileges

1. *Rationale: Define view component privilege descriptors. Only <grant privilege statement> uses GRANT OPTION.*

   Replace the 1st paragraph with:

   A privilege authorizes a given category of <action> to be performed by a specified <authorization identifier> on a specified base table, view, view component, column, domain, character set, collation, transliteration, user-defined type, table/method pair, SQL-invoked routine, or sequence generator.

2. *Rationale: Define view component privilege descriptors.*

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

— The identification of the base table, view, view component, column, domain, character set, collation, transliteration, user-defined type, table/method pair, SQL-invoked routine, or sequence generator that the descriptor describes.

3. *Rationale: Define view component privilege descriptors.*

Replace the 4th and 5th paragraphs with:

A privilege descriptor with an <action> of INSERT, UPDATE, DELETE, SELECT, TRIGGER, or REFERENCES is called a *table privilege descriptor* and identifies the existence of a privilege on the table or view component identified by the privilege descriptor. If a table privilege descriptor identifies a view component, the privilege descriptor is called a *view component table privilege descriptor*.

A privilege descriptor with an <action> of SELECT (<column name list>), INSERT (<column name list>), UPDATE (<column name list>), or REFERENCES (<column name list>) is called a *column privilege descriptor* and identifies the existence of a privilege on the columns in the table or view component identified by the privilege descriptor. If a column privilege descriptor identifies a view component, the privilege descriptor is called a *view component column privilege descriptor*.

4. *Rationale: Define view component privilege descriptors.*

Insert the following paragraph after the 10th paragraph:

A *view privilege dependency descriptor* is a descriptor that includes two privilege descriptors, called the *supporting privilege descriptor* and the *dependent privilege descriptor*. A view privilege dependency descriptor is a record that an INSERT, UPDATE, or DELETE privilege of a view, or a column of a view, is directly dependent on another privilege.

5. *Rationale: Only <grant privilege statement> uses GRANT OPTION.*

Replace the 11th paragraph with:

A grantable privilege is a privilege, associated with a schema object, that may be granted by a <grant privilege statement>. The WITH GRANT OPTION clause of a <grant privilege statement> specifies whether the <authorization identifier> recipient of a privilege (acting as a grantor) may grant it to others.

4.34.3 Roles

1. *Rationale: Various editorial corrections and additions, and removal of an unnecessary definition.*

Replace the entire Subclause with:

A role, identified by a role name, is, like a user, a potential grantee and grantor of privileges and other roles. Also like a user, a role can additionally own schemas and other objects.

A role is created by executing a <role definition> and destroyed by executing a <drop role statement>.

A role is granted to one or more authorization identifiers by executing a <grant role statement>, thus conferring on the grantees all the privileges of that role. The granting of a role to an authorization identifier *A* is called a *role authorization* (for *A*).

The privileges of a role with role name *R* are the union of the privileges whose grantee is *R* and the sets of privileges for the role names defined by the role authorizations for *R*. Cycles of role authorizations are prohibited.

The WITH ADMIN OPTION clause of the <grant role statement> for role R specifies that each grantee may grant *R* to others, revoke *R* from others, and destroy *R*.

Each role authorization is described by a *role authorization descriptor*. A role authorization descriptor includes:

— The role name of the role.

— The authorization identifier of the grantor.

— The authorization identifier of the grantee.

— An indication of whether or not the role authorization is grantable.

4.34.4 Security model definitions

1. *Rationale: Insertion of one missing definition and simplification of others.*

   Replace the entire subclause with:

   A role *R* is *applicable for* an authorization identifier *A* if there exists a role authorization descriptor whose role name is *R* and whose grantee is PUBLIC, or *A*, or an applicable role for *A*.

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

   NOTE 52.1 — *applicable for* is a persistent relationship between persistent objects. Thus, it in no way depends on any SQL-session.

   An authorization identifier is *enabled* if it is the current user identifier, the current role name, or a role name that is applicable for the current role name.

   A privilege *P* is *current* if *P* is applicable for an enabled authorization identifier.

   NOTE 52.2 — *enabled* and *current* apply to (transient) elements of the current SQL-session context.

4.37 SQL-sessions

4.37.3 SQL-session properties

1. *Rationale: Clarify how enduring transaction characteristics are altered.*

   Replace the 10th and 11th paragraphs with:

   An SQL-session has the following enduring characteristics:

   — *enduring transaction characteristics*

   Each of the enduring characteristics can be altered at any time in an SQL-session by executing an appropriate <set session characteristics statement>.

2. *Rationale: Avoid misleading references to "current transaction".*

   In the 12th paragraph, replace the 7th dashed item with:

   — The current access mode.

3. *Rationale: Avoid misleading references to "current transaction".*

   In the 12th paragraph, replace the 9th dashed item with:

   — The current isolation level.

4. *Rationale: Add result set sequences to SQL-session context.*

   In the 12th paragraph, insert the following after the 24th bullet:

   — Each currently available result set sequence *RSS*, along with the specific name of an SQL-invoked procedure *SIP* and the name of the invoker of *SIP* for the invocation causing *RSS* to be brought into existence.

   NOTE 54.1 — Result set sequences are defined in Subclause 4.27.5, "Result sets returned by SQL-invoked procedures".

4.38 Triggers

4.38.2 Trigger execution

1. *Rationale: Remove incorrect information about trigger execution contexts.*

   Delete the 2nd paragraph.

2. *Rationale: Correct the information about where transitions are specified.*

   Replace the 17th paragraph with:

   Let *PSCN* be the number of elements in *PSC*. A state change $SC_{i,j}$, for *j* varying from 1 (one) to *PSCN*, identified by *TE*, *ST*, and the *j*-th element in *PSC*, is added to SSC_i, provided that SSC_i does not already contain a state change corresponding to $SC_{i,j}$. Transitions are added to $SC_{i,j}$ as specified by the General Rules of Subclause 14.16, "Effect of deleting rows from base tables", Subclause 14.19, "Effect of inserting tables into base tables", and Subclause 14.22, "Effect of replacing rows in base tables".

5 Lexical elements

5.1 <SQL terminal character>

1. *Rationale: Correct character being defined.*

   Replace the production for <right bracket> with:

   ```
   <right bracket> ::= ]
   ```

5.2 <token> and <separator>

1. *Rationale: Explicitly make lower case "u" permissable.*

   Insert the following Syntax Rule

   14.1) In a <Unicode delimited identifier>, the introductory 'U' may be represented either in upper case (as 'U') or in lower case (as 'u').

2. *Rationale: Editorial.*

   Replace Syntax Rules 15) and 16) with:

   15) <Unicode escape character> shall be a single character from the source language character set other than a <hexit>, <plus sign>, <quote>, <double quote>, or <white space>.

   16) If the source language character set contains <reverse solidus>, then let *DEC* be <reverse solidus>; otherwise, let *DEC* be an implementation-defined character from the source language character set that is not a <hexit>, <plus sign>, <quote>, <double quote>, or <white space>.

5.3 <literal>

1. *Rationale: Explicitly make lower case "u" permissable.*

   Insert the following Syntax Rule

   2.1) In a <Unicode character string literal>, the introductory 'U' may be represented either in upper case (as 'U') or in lower case (as 'u').

2. *Rationale: Explicitly make lower case "x" permissable.*

   Insert the following Syntax Rule

   3.1) In a <binary string literal>, the introductory 'X' may be represented either in upper case (as 'X') or in lower case (as 'x').

3. *Rationale: Explicitly make lower case "n" permissable.*

   Insert the following Syntax Rule

   5.1) In a <national character string literal>, the introductory 'N' may be represented either in upper case (as 'N') or in lower case (as 'n').

4. *Rationale: Explicitly make lower case "e" permissable.*

   Insert the following Syntax Rule

   19.1) In an <approximate numeric literal>, the exponent indicator 'E' may be represented either in upper case (as 'E') or in lower case (as 'e').

5.4 Names and identifiers

1. *Rationale: Repair an omission and remove unnecessary definition of <local qualified name>.*

   Replace Syntax Rule 7) with:

   7) Let *CN* be a <cursor name>. *CN* shall be contained in an <SQL-client module definition> whose <module contents> contain a <declare cursor> or <dynamic declare cursor> whose <cursor name> is *CN*.

2. *Rationale: Correct misspelling.*

   Replace Syntax Rule 9) with the following:

   9) Two <user-defined type name>s are equivalent if and only if they have equivalent <qualified identifier>s and equivalent <schema name>s, regardless of whether the <schema name>s are implicit or explicit.

3. *Rationale: Determine the implicit schema name of a constraint name correctly.*

   Replace Syntax Rule 13) with:

   13) If a <schema qualified name> *SQN* other than a <transcoding name> does not contain a <schema name>, then

   Case:

   a) If any of the following is true:

      i) *SQN* is immediately contained in a <collation name> that is not immediately contained in a <collation definition> or in a <drop collation statement>.

      ii) *SQN* is immediately contained in a <transliteration name> that is not immediately contained in a <transliteration definition> or in a <drop transliteration statement>.

   then <schema name> INFORMATION_SCHEMA is implicit.

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

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

   d) Otherwise,

   Case:

   i) If *SQN* is contained, without an intervening <schema definition>, in a <preparable statement> that is prepared in the current SQL-session by an <execute immediate statement> or a <prepare statement> or in a <direct SQL statement> that is invoked directly, then the default <unqualified schema name> for the SQL-session is implicit.

ii) If *SQN* is contained in a <schema definition>, then the <schema name> that is specified or implicit in the <schema definition> is implicit.

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

4. *Rationale: Prohibit the use of LOCAL in places where it makes no sense.*

Replace Syntax Rule 28) with:

28) In a <descriptor name>, <extended statement name>, or <extended cursor name>, if a <scope option> is not specified, then a <scope option> of LOCAL is implicit. If a <scope option> is contained in an <SQL-schema statement> then it shall not contain LOCAL.

5. *Rationale: Clarification of the method of identification of dynamic objects.*

Replace General Rule 31) with:

31) If a prepared statement *PSX* is created in SQL-session *SS* by executing a <prepare statement> *PS1* that contains an <extended statement name> *ESN1* whose value at the time of execution is *V*, then, for as long as it exists, *PSX* can be identified by an <extended statement name> *ESN2* in an <SQL procedure statement> *PS2* executed in *SS* if the value of *ESN2* at the time of execution is *V* and the <scope option> of *ESN2* is the same as the <scope option> of *ESN1*. If the <scope option> of *ESN1* is LOCAL, then *ESN2* identifies *PSX* only if *PS2* is contained in the same <SQL-client module definition> as *PS1*.

6. *Rationale: Clarification of the method of identification of dynamic objects.*

Replace General Rule 34) with:

34) If a cursor *CSR* is created in SQL-session *SS* by executing an <allocate cursor statement> *ACS* that contains an <extended statement name> *ESN1* whose value at the time of execution is *V*, then, for as long as it exists, *CSR* can be identified by an <extended statement name> *ESN2* in an <SQL procedure statement> *PS2* executed in *SS* if the value of *ESN2* at the time of execution is *V* and the <scope option> of *ESN2* is the same as the <scope option> of *ESN1*. If the <scope option> of *ESN1* is LOCAL, then *ESN2* identifies *CSR* only if *PS2* is contained in the same <SQL-client module definition> as *ACS*.

7. *Rationale: Clarification of the method of identification of dynamic objects.*

Replace General Rule 35) with:

35) If an SQL descriptor area *SDA* is created in SQL-session *SS* by executing an <allocate descriptor statement> *ADS* that contains an <extended statement name> *ESN1* whose value at the time of execution is *V*, then, for as long as it exists, *SDA* can be identified by an <extended statement name> *ESN2* in an <SQL procedure statement> *PS2* executed in *SS* if the value of *ESN2* at the time of execution is *V* and the <scope option> of *ESN2* is the same as the <scope option> of *ESN1*. If the <scope option> of *ESN1* is LOCAL, then *ESN2* identifies *SDA* only if *PS2* is contained in the same <SQL-client module definition> as *ADS*.

6 Scalar expressions

6.1 <data type>

1. *Rationale: <char length units> is not appropriate to a <binary large object string type>.*

   Insert the following Syntax Rule:

   5.1) A <binary large object string type> shall not contain a <char length units>.

2. *Rationale: Correct name of the type.*

   Replace Syntax Rule 6) with:

   6) If <char length units> is specified, then the character repertoire of the explicit or implicit character set of the character string type shall be UCS.

3. *Rationale: <char length units> is not appropriate to a <binary large object string type>.*

   Replace Syntax Rule 7) with:

   7) If <data type> contains a <character string type> or <character large object type> and does not contain a <char length units>, then CHARACTERS is implicit.

6.4 <value specification> and <target specification>

1. *Rationale: Omission of <target array reference>.*

   Replace Syntax Rule 7) with:

   7) A <target specification>, <target array reference>, or <simple target specification> that is a <column reference> shall be a new transition variable column reference.

2. *Rationale: Inappropriate use of the definite article, omission of array element, and ambiguity.*

   Replace General Rule 3) with:

   3) A <target specification> specifies a target that is a host parameter, an output SQL parameter, a column of a new transition variable, an element of a target whose declared type is an array type, a parameter used in a dynamically prepared statement, or a host variable, according to whether the <target specification> is a <host parameter specification>, an <SQL parameter reference>, a <column reference>, a <target array element specification>, a <dynamic parameter specification>, or an <embedded variable specification>, respectively.

3. *Rationale: Omission of embedded variable, and presence of non-normative text.*

   Replace General Rule 13) with:

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

   NOTE 91.1 — A <simple target specification> can never be assigned the null value.

4. *Rationale: The word "specific"" is not necessary.*

   Replace Conformance Rule 5) with:

   5) Without Feature F611, "Indicator data types", in conforming SQL language, the declared types of <indicator parameter>s and <indicator variable>s shall be the same implementation-defined data type.

6.9 <set function specification>

1. *Rationale: "<subquery>" should be "<query expression>".*

   Replace Syntax Rule 1) with:

   1) If <aggregate function> specifies a <general set function>, then the <value expression> simply contained in the <general set function> shall not contain a <set function specification> or a <query expression>.

2. *Rationale: "<subquery>" should be "<query expression>".*

   Replace Syntax Rule 2) with:

   2) If <aggregate function> specifies <binary set function>, then neither the <dependent variable expression> nor the <independent variable expression> simply contained in the <binary set function> shall contain a <set function specification> or a <query expression>.

6.10 <window function>

1. *Rationale: "<subquery>" should be "<query expression>".*

   Replace Syntax Rule 4) with:

   4) *OF* shall not contain an outer reference or a <query expression>.

6.12 <cast specification>

1. *Rationale: Need to specify the returned value.*

   Replace General Rules 1) and 2) with:

   1) Let *CS* be the <cast specification>. If the <cast operand> is a <value expression> *VE*, then let *SV* be the value of *VE*.

   2) Case:

      a) If the <cast operand> specifies NULL, then the result of *CS* is the null value and no further General Rules of this Subclause are applied.

      b) If the <cast operand> specifies an <empty specification>, then the result of *CS* is an empty collection of declared type *TD* and no further General Rules of this Subclause are applied.

      c) If *SV* is the null value, then the result of *CS* is the null value and no further General Rules of this Subclause are applied.

2. *Rationale: Use the correct symbol.*

   Replace General Rule 3) a) with:

   3) ...

      a) If *TD* is a supertype of *SD*, then *TV* is *SV*.

3. *Rationale: Correct use of values in syntactic substitutions.*

   Delete General Rule 5) b).

4. *Rationale: Correct use of values in syntactic substitutions.*

   Replace General Rule 5) c) with:

   5) ...

      c) For *i* varying from 1 (one) to *SC*, the following <cast specification> is applied:

         `CAST ( VE[i] AS ETD )`

         yielding value TVE_i.

5. *Rationale: Correct use of values in syntactic substitutions.*

   Delete General Rules 6) b) and 6) c).

6. *Rationale: Correct use of values in syntactic substitutions.*

   Replace General Rule 6) d) and 6) e) with:

6) ...

d) If *TD* is an array type, then let *TC* be the maximum cardinality of *TD*.

Case:

i) If *SC* is greater than *TC*, then an exception condition is raised: *data exception — array data, right truncation*.

ii) Otherwise, *TV* is the array resulting from the execution of:

```
ARRAY (
 ( SELECT CAST ( M.E AS ETD ) FROM UNNEST ( VE ) AS M(E) )
)
```

e) If *TD* is a multiset type, then *TV* is the multiset resulting from the execution of:

```
MULTISET ( ( SELECT CAST ( M.E AS ETD ) FROM UNNEST ( VE ) AS M(E) ) )
```

7. *Rationale: Correct use of values in syntactic substitutions.*

Replace General Rule 7) with:

7) If *SD* is a <row type>, then *TV* is the row resulting from the execution of:

```
ROW ( CAST ( VE.FSD1 AS TFTD1), CAST ( VE.FSD2 AS TFTD2), ...
  CAST ( VE.FSDDSD AS TFTDDSD ) )
```

8. *Rationale: Delete ill-formed rule and make all invalid character strings for CAST to datetime and interval types result in the same exception.*

Delete General Rule 13) a) ii).

9. *Rationale: Delete ill-formed rule and make all invalid character strings for CAST to datetime and interval types result in the same exception.*

Replace General Rule 13) b) with:

13) ...

b) If *SD* is the datetime data type DATE, then *TV* is *SV*.

10. *Rationale: Correct use of values in syntactic substitutions.*

Replace General Rule 13) d) with:

13) ...

d) If *SD* is the datetime data type TIMESTAMP WITH TIME ZONE, then *TV* is computed by:

```
CAST ( CAST ( VE AS TIMESTAMP WITHOUT TIME ZONE ) AS DATE )
```

11. *Rationale: Correct use of values in syntactic substitutions.*

    Replace General Rule 15) a) ii) with:

    15) ...

        a) ...

            ii) 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 TIME(*TSP*) WITH TIME ZONE, then *TV* is:

```
CAST ( CAST ( VE AS TIME(TSP) WITH TIME ZONE )
  AS TIME(TSP) WITHOUT TIME ZONE )
```

12. *Rationale: Delete ill-formed rule and make all invalid character strings for CAST to datetime and interval types result in the same exception.*

    Replace General Rule 15) a) iii) and General Rule 15) a) iv) with:

    15) ...

        a) ...

            iii) Otherwise, an exception condition is raised: *data exception — invalid datetime format*.

13. *Rationale: Correct use of values in syntactic substitutions.*

    Replace General Rule 15) e) with:

    15) ...

        e) If *SD* is TIMESTAMP WITH TIME ZONE, then *TV* is:

```
CAST ( CAST ( VE AS TIMESTAMP(TSP) WITHOUT TIME ZONE )
  AS TIME(TSP) WITHOUT TIME ZONE )
```

14. *Rationale: Correct use of values in syntactic substitutions.*

    Replace General Rule 16) a) ii) with:

    16) ...

        a) ...

            ii) 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 TIME(*TSP*) WITHOUT TIME ZONE, then *TV* is:

```
CAST ( CAST ( VE AS TIME(TSP) WITHOUT TIME ZONE )
  AS TIME(TSP) WITH TIME ZONE )
```


15. *Rationale: Delete ill-formed rule and make all invalid character strings for CAST to datetime and interval types result in the same exception.*

Replace General Rule 16) a) iii) and General Rule 16) a) iv) with:

16) ...

a) ...

iii) Otherwise, an exception condition is raised: *data exception — invalid datetime format*.

16. *Rationale: Correct use of values in syntactic substitutions.*

Replace General Rule 16) e) with:

16) ...

e) If *SD* is TIMESTAMP WITHOUT TIME ZONE, then *TV* is:

```
CAST ( CAST ( VE AS TIMESTAMP(TSP) WITH TIME ZONE )
  AS TIME(TSP) WITH TIME ZONE )
```

17. *Rationale: Correct use of values in syntactic substitutions.*

Replace General Rule 17) a) ii) with:

17) ...

a) ...

ii) 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 TIMESTAMP(*TSP*) WITH TIME ZONE, then *TV* is:

```
CAST ( CAST ( VE AS TIMESTAMP(TSP) WITH TIME ZONE )
  AS TIMESTAMP(TSP) WITHOUT TIME ZONE )
```

18. *Rationale: Delete ill-formed rule and make all invalid character strings for CAST to datetime and interval types result in the same exception.*

Replace General Rule 17) a) iii) and General Rule 17) a) iv) with:

17) ...

a) ...

iii) Otherwise, an exception condition is raised: *data exception — invalid datetime format*.

19. *Rationale: Correct use of values in syntactic substitutions.*

Replace General Rule 17) d) with:

17) ...

d) If *SD* is TIME WITH TIME ZONE, then *TV* is:

```
CAST ( CAST ( VE AS TIMESTAMP(TSP) WITH TIME ZONE )
  AS TIMESTAMP(TSP) WITHOUT TIME ZONE )
```

20. *Rationale: Correct use of values in syntactic substitutions.*

Replace General Rule 18) a) ii) with:

18) ...

a) ...

ii) 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 TIMESTAMP(*TSP*) WITHOUT TIME ZONE, then *TV* is:

```
CAST ( CAST ( VE AS TIMESTAMP(TSP) WITHOUT TIME ZONE )
  AS TIMESTAMP(TSP) WITH TIME ZONE )
```

21. *Rationale: Delete ill-formed rule and make all invalid character strings for CAST to datetime and interval types result in the same exception.*

Replace General Rule 18) a) iii) and General Rule 18) a) iv) with:

18) ...

a) ...

iii) Otherwise, an exception condition is raised: *data exception — invalid datetime format*.

22. *Rationale: Correct use of values in syntactic substitutions.*

Replace General Rule 18) b) and 18) c) with:

18) ...

b) If *SD* is a date, then *TV* is computed by:

```
CAST ( CAST ( VE AS TIMESTAMP WITHOUT TIME ZONE )
  AS TIMESTAMP WITH TIME ZONE )
```

c) If *SD* is TIME WITHOUT TIME ZONE, then *TV* is:

```
CAST ( CAST ( VE AS TIMESTAMP WITHOUT TIME ZONE )
  AS TIMESTAMP WITH TIME ZONE )
```


23. *Rationale: Delete ill-formed rule and make all invalid character strings for CAST to datetime and interval types result in the same exception.*

    Replace General Rule 19) b) ii)with:

    19) ...

        b) ...

            ii) Otherwise, an exception condition is raised: *data exception — invalid interval format*.

24. *Rationale: Need to specify the returned value.*

    Insert the following General Rule:

    21.1) The result of *CS* is *TV*.

6.13 <next value expression>

1. *Rationale: Define the data type of the <next value expression>.*

    Insert the following Syntax Rule:

    3) The declared type of <next value expression> is the data type described by the data type descriptor included in the sequence generator descriptor identified by <sequence generator name>.

6.27 <numeric value function>

1. *Rationale: CHARACTER_LENGTH cannot support binary string arguments.*

    In the Format, replace the production for <char length expression> with:

```
<char length expression> ::=
    { CHAR_LENGTH | CHARACTER_LENGTH } <left paren> <character value expression>
    [ USING <char length units> ] <right paren>
```

2. *Rationale: Correct the definition of MOD function. The functionality was intended to be aligned with that of the FORTRAN MOD function and the C fmod function and % operator.*

    Replace General Rule 9) c) lead-in paragraph, with:

    9) ...

        c) Otherwise, the result is the unique exact numeric value *R* with scale 0 (zero) such that all of the following are true:

6.29 <string value function>

1. *Rationale: Remove test on value from Syntax Rules.*

   Delete Syntax Rule 7) d).

7 Query expressions

7.6 <table reference>

1. *Rationale: Define simply updatable derived table.*

   Insert the following Syntax Rule:

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

2. *Rationale: Define simply updatable derived table.*

   Replace Syntax Rule 17) with:

   17) A <collection derived table> is not updatable and not simply updatable.

3. *Rationale: Move the rule pertaining to <sample clause> from Subclause 7.15, "<subquery>" to this Subclause with appropriate rewording.*

   Insert the following General Rule:

   3) ...

      a) ...

         v) If *TF* contains outer references, then a table with identical rows is generated every time *TF* is evaluated with a given set of values for outer references.

            NOTE 130.1 — "Outer reference" is defined in Subclause 6.7, "<column reference>".

7.7 <joined table>

1. *Rationale: Corresponding join columns belong to the operands of a <joined table>, not to the result.*

   Replace Syntax Rules 10), 11) and 12) with:

10) For every column *CR* of the result of the <joined table> that corresponds to a column C_1 of T_1 that is not a corresponding join column, *CR* is *possibly nullable* if any of the following conditions are true:

    a) RIGHT or FULLis specified.

    b) INNER, LEFT, or CROSS JOIN is specified or implicit and C_1 is possibly nullable.

11) For every column *CR* of the result of the <joined table> that corresponds to a column C_2 of T_2 that is not a corresponding join column, *CR* is *possibly nullable* if any of the following conditions are true:

    a) LEFT or FULL is specified.

    b) INNER, RIGHT, or CROSS JOIN is specified or implicit and C_2 is possibly nullable.

12) For every column *CR* of the result of the <joined table> that corresponds to a corresponding join column C_1 of T_1 and a corresponding join column C_2 of T_2, *CR* is *possibly nullable* if any of the following conditions are true:

    a) LEFT or FULL is specified and C_1 is possibly nullable.

    b) RIGHT or FULL is specified and C_2 is possibly nullable.

2. *Rationale: Incorrect use of "subrow", incorrect subscript, wrong side.*

   Insert the following General Rule:

   1.1) Let D_1 and D_2 be the degrees of TR_1 and TR_2, respectively.

3. *Rationale: Incorrect use of "subrow", incorrect subscript, wrong side.*

   Replace General Rule 4) with:

   4) Let P_1 be the collection of rows of T_1 for which there exist some row R in TR and some row R_1 in T_1 such that the values of the first D_1 fields of R are identical to the values of the corresponding fields of R_1.

4. *Rationale: Incorrect use of "subrow", incorrect subscript, wrong side.*

   Replace General Rule 6) with:

   6) Let X_1 be U_1 extended with D_2 columns containing the null value.

5. *Rationale: Remove incorrect use "subrow", incorrect subscript, wrong side.*

   Replace General Rule 8) b) with:

   8) ...

b) Let P_2 be the collection of rows of T_2 for which there exists some row R in TR and some row R_2 in T_2 such that the values of the last D_2 fields of R are identical to the values of the corresponding fields of R_2.

6. *Rationale: Incorrect use of "subrow", incorrect subscript, wrong side.*

Replace General Rule 8) d) with:

8) ...

d) Let X_2 be U_2 extended on the left with *D1* columns containing the null value.

7.8 <where clause>

1. *Rationale: "<subquery>" should be "<query expression>".*

Replace Syntax Rule 2) with:

2) The <search condition> shall not contain a <window function> without an intervening <query expression>.

7.10 <having clause>

1. *Rationale: "<subquery>" should be "<query expression>".*

Replace Syntax Rule 4) with:

4) The <search condition> shall not contain a <window function> without an intervening <query expression>.

2. *Rationale: "<subquery>" should be "<query expression>".*

Replace Conformance Rule 2) with:

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

a) An unambiguous reference to a grouping column of *T*.

b) Contained in an aggregated argument of a <set function specification>.

7.11 <window clause>

1. *Rationale: "<subquery>" should be "<query expression>".*

   Replace Syntax Rule 7) with:

   7) A <window clause> shall not contain a <window function> without an intervening <query expression>.

7.12 <query specification>

1. *Rationale: "<subquery>" should be "<query expression>".*

   Replace Syntax Rule 12) j) with:

   12) ...

   j) Let *N2* be the number of <column reference>s that are contained in *SL* or *WIC* without an intervening <query expression> or <set function specification>.

2. *Rationale: "<subquery>" should be "<query expression>".*

   Replace Syntax Rule 12) k) with:

   12) ...

   k) Let CR_j, 1 (one) ≤ j ≤ *N2*, be an enumeration of the <column reference>s that are contained in *SL* or *WIC* without an intervening <query expression> or <set function specification>.

3. *Rationale: "<subquery>" should be "<query expression>".*

   Replace Syntax Rule 12) r) with:

   12) ...

   r) Let *SLNEW* be the <select list> obtained from *SL2* by replacing each simply contained <set function specification> SFS_i by $GWQN.SFSI_i$ and replacing each <column reference> CR_j that is contained without an intervening <query expression> or <set function specification> by $GWQN.CRI_j$.

4. *Rationale: "<subquery>" should be "<query expression>".*

   Replace Syntax Rule 18) a) vii) with:

   18) ...

   a) ...

   vii) A <query expression>.

5. *Rationale: Clarify how to decide if two column references are the same.*

   Replace Syntax Rule 21) b) with:

   21) ...

      b) Of those <derived column>s in the <select list> that are column references that have a counterpart in a base table, no column of a base table is referenced more than once in the <select list>.

6. *Rationale: Correct the definition of insertable-into.*

   Delete Syntax Rule 22).

7. *Rationale: Make simply updatable equivalent to updatable in SQL-92.*

   Replace Syntax Rules 23), 24), and 25) with:

   23) If a <query specification> *QS* is potentially updatable, then

      Case:

      a) If the <from clause> of the <table expression> specifies exactly one <table reference>, then a column of *QS* is said to be a *potentially updatable column* if it has a counterpart in *TR* that is updatable.

         NOTE 169 — The notion of updatable columns of table references is defined in Subclause 7.6, "<table reference>".

      b) Otherwise, a column of *QS* is said to be a *potentially updatable column* if it has a counterpart in some updatable column of some simply underlying table *UT* of *QS* such that *QS* is one-to-one with respect to *UT*.

   24) A <query specification> is *updatable* if it is potentially updatable and it has at least one potentially updatable column.

   25) A <query specification> *QS* is *simply updatable* if all of the following conditions hold:

      a) *QS* is updatable.

      b) The <from clause> immediately contained in the <table expression> immediately contained in *QS* contains exactly one <table reference>, and the table referenced by that <table reference> is simply updatable.

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

      d) If the <table expression> immediately contained in *QS* immediately contains a <where clause> *WC*, then no leaf generally underlying table of *QS* shall be a generally underlying table of any <query expression> contained in *WC*.

   25.1) A column *C* of *QS* is *updatable* if at least one of the following is true:

      a) *QS* is simply updatable.

      b) *QS* is updatable, *C* is potentially updatable, and the SQL-implementation supports Feature T111, "Updatable joins, unions and columns".

8. *Rationale: Correct the definition of insertable-into.*

   Insert the following Syntax Rule:

   25.2) A <query specification> *QS* is *insertable-into* if it is updatable and every simply underlying table of *QS* is insertable-into.

9. *Rationale: Clarify the evaluation of <set function specification>s and outer references in <subquery>s.*

   Replace General Rule 1) with:

   1) If *QS* is contained in a <subquery> *SQ*, then certain <set function specification>s and outer references are resolved, such that their values are constant for every row in the result of *QS*, as follows:

      Case:

      a) If *SQ* is being evaluated for a group, then let *G* be that group.

         i) Let *TE* be the <table expression> whose result includes *G*.

         ii) For every <set function specification> *SFS* contained in *QS* whose aggregation query simply contains *TE*, the value of *SFS* is the result of evaluating *SFS* for *G*.

            NOTE 170 — The circumstances in which a <subquery> is evaluated for a given group, rather than a given row, are defined in the General Rules of this Subclause and the General Rules of Subclause 7.10, "<having clause>".

      b) Otherwise, let *R* be the row for which *SQ* is being evaluated. For every <column reference> *CR* contained in *SQ* that is an outer reference whose qualifying scope is simply contained in a <subquery> that contains *SQ*, the value of *CR* is the value of the field in *R* corresponding to the column referenced by *CR*.

         NOTE 171 — An expression having been resolved under this rule is not resolved again in the case where it is contained in a <query expression> contained in *SQ*.

10. *Rationale: "<subquery>" should be "<query expression>".*

    Replace Conformance Rule 1) with:

    1) Without Feature F801, "Full set function", conforming SQL language shall not contain a <query specification> *QS* that contains more than one <set quantifier> containing DISTINCT, unless such <set quantifier> is contained in a <query expression> contained in *QS*.

11. *Rationale: Eliminate redefinition of updatable in Conformance Rules.*

    Delete Conformance Rule 4).

7.13 <query expression>

1. *Rationale: "<subquery>" should be "<query expression>".*

   Replace Syntax Rule 2) g) ii) 2) A) III) with:

   2) ...

      g) ...

         ii) ...

            2) ...

               A) ...

                  III) WQE_i does not contain a <table expression> that immediately contains a <from clause> that contains WQN_k, and immediately contains a <where clause> containing a <query expression> that contains a <query name> referencing WQN_l.

2. *Rationale: Define simply updatable <query expression>.*

   Insert this Syntax Rule:

   6.1) <query expression> *QE1* is *simply updatable* if, for every <simple table> *QE2* that is simply contained in the <query expression body> of *QE1*, all of the following are true:

      a) *QE2* is not a <table value constructor>.

      b) *QE1* contains *QE2* without an intervening <query expression body> that specifies UNION ALL, UNION DISTINCT, EXCEPT ALL, or EXCEPT DISTINCT.

      c) *QE1* contains *QE2* without an intervening <query term> that specifies INTERSECT.

      d) *QE2* is simply updatable.

3. *Rationale: Make simply updatable equivalent to updatable in SQL-92.*

   Replace the lead-in paragraph of Syntax Rule 7) with:

   7) <query expression> *QE1* is *updatable* if for every <simple table> *QE2* that is simply contained in the <query expression body> of *QE1*:

4. *Rationale: Make simply updatable equivalent to updatable in SQL-92.*

   Insert the following subrule to Syntax Rule 7):

   7) ...

      a.1) *QE2* is not a <table value constructor>.

5. *Rationale: Make simply updatable equivalent to updatable in SQL-92.*

   Replace Syntax Rule 7) b) i) with:

   7) ...

      b) ...

         i) *QEB* immediately contains a <query expression body> *LO* and a <query term> *RO* such that no leaf generally underlying table of *LO* is also a leaf generally underlying table of *RO*.

6. *Rationale: Complete the definition of simply updatable.*

   Insert the following Syntax Rule:

   8.1) A table specified by a <query name> immediately contained in a <with list element> *WLE* is *simply updatable* if and only if the <query expression> simply contained in *WLE* is simply updatable.

7. *Rationale: Correct the definition of insertable-into.*

   Replace Syntax Rule 9) with:

   9) <query expression> *QE1* is *insertable-into* if the <query expression body> of *QE1* is a <query primary> that is one of the following:

      a) An insertable-into <query specification>.

      b) An <explicit table> that identifies a table that is insertable-into.

      c) Of the form <left paren> <query expression body> <right paren>, where the parenthesized <query expression body> recursively satisfies this condition.

8. *Rationale: A <table value constructor> has no updatable columns.*

   Replace Syntax Rule 22) a) with:

   22) ...

      a) A column of a <table value constructor> has no underlying columns and no updatable columns.

9. *Rationale: Conditionalize the definition of updatable column, depending on support for Feature T111, "Updatable joins, unions and columns".*

   Replace Syntax Rule 22) d) i) with:

   22) ...

      d) ...

         i) If the SQL-implementation supports Feature T111, "Updatable joins, unions and columns", a set operator UNION ALL is specified, and both underlying columns of

the *i*-th column of *QE* are updatable, then the *i*-th column of *QE* is an updatable column of *QE*.

10. *Rationale: Remove reference to non-existent option.*

    Replace General Rule 4) a) with:

    4) ...

       a) If no set operator is specified, then *T* is the result of the specified <simple table>.

11. *Rationale: "<subquery>" should be "<query expression>".*

    Replace Conformance Rule 2) with:

    2) Without Feature T122, "WITH (excluding RECURSIVE) in subquery", in conforming SQL language, a <query expression> contained in a <query expression> shall not contain a <with clause>.

12. *Rationale: "<subquery>" should be "<query expression>".*

    Replace Conformance Rule 4) with:

    4) Without Feature T132, "Recursive query in subquery", in conforming SQL language, a <query expression> contained in a <query expression> shall not contain RECURSIVE.

13. *Rationale: Eliminate redefinition of updatable in Conformance Rules.*

    Delete Conformance Rule 10).

7.14 <search or cycle clause>

1. *Rationale: Remove the ability to specify non-relevant and potentially misleading syntax.*

   In the Format replace the production for "recursive search order" with:

```
<recursive search order> ::=
    DEPTH FIRST BY <column name list>
  | BREADTH FIRST BY <column name list>
```

2. *Rationale: Remove the ability to specify non-relevant and potentially misleading syntax.*

   Replace Syntax Rule 2) b) i) with:

   2) ...

      b) ...

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

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

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

3) Case:

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

```
WQNCRN.SQC
```

let *SCEX2* be:

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

and let *SCIN* be:

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

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

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

let *SCEX2* be:

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

and let *SCIN* be:

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

7.15 <subquery>

1. *Rationale: Move the rule pertaining to <sample clause> from this Subclause to Subclause 7.6, "<table reference>" with appropriate rewording.*

Delete General Rule 5) and Note 177.

8 Predicates

8.2 <comparison predicate>

1. *Rationale: Cardinality is a dynamic aspect of value, not a declared aspect.*

   Replace the lead-text of General Rule 1) b) ii) with:

   1) ...

      b) ...

         ii) If the declared types of *XV* and *YV* are array types and the cardinalities of *XV* and *YV* are *N1* and *N2*, respectively, then let X_i, 1 (one) $\le i \le$ *N1*, denote a <value expression> whose value and declared type is that of the *i*-th element of *XV* and let Y_i denote a <value expression> whose value and declared type is that of the *i*-th element of *YV*. The result of

         X `<comp op>` *Y*

         is determined as follows:

2. *Rationale: Cardinality is a dynamic aspect of value, not a declared aspect.*

   Replace the lead-text of General Rule 1) b) iii) with:

   1) ...

      b) ...

         iii) If the declared types of *XV* and *YV* are multiset types and the cardinalities of *XV* and *YV* are *N1* and *N2*, respectively, then the result of

         X `<comp op>` *Y*

         is determined as follows:

         Case:

3. *Rationale: Allow for mulitsets being empty.*

   Insert the following General Rule:

   1) ...

      b) ...

         iii) ...

            0.1) *X* = *Y* is *True* if *N1* = 0 (zero) and *N2* = 0 (zero).

9 Additional common rules

9.24 Determination of view and view component privileges

1. *Rationale: Determine INSERT, UPDATE, and DELETE privileges on views correctly.*

   Insert the following new Subclause:

9.24 Determination of view and view component privileges

Function

Determine view component privilege descriptors for all view components of a view, and the privilege descriptors of the view whose <action> is INSERT, UPDATE, or DELETE. Also determine the view privilege dependency descriptors of the view.

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let *V* be the *VIEW* in an application of this Subclause.

2) Let *A* be the <authorization identifier> that owns the schema identified by the <table name> of *V*.

3) Let $VC_1, \ldots, VC_N$ be an enumeration of the view components of *V*. The enumeration shall have the following properties:

   a) For all *i* and *j* between 1 (one) and *N*, if VC_i is contained in VC_j, then $i < j$.

   b) For all *i* and *j* between 1 (one) and *N*, if VC_i is a <query expression> simply contained in a <with list element> and VCj references the table defined by VC_i, then $i < j$.

   NOTE 213.1 — A depth-first left-to-right traversal of the BNF of the <view definition> of *V* is one way to obtain such an enumeration.

4) If *V* is effectively updatable, then the following subrules are performed to recursively create certain view component privilege descriptors. The following subrules also recursively define when a view component privilege is *immediately dependent* on another privilege descriptor.

   For each *i* between 1 (one) and *N*, in that order,

Case:

a) If VC_i is an updatable <query specification>, then

i) If the <from clause> of VC_i has exactly one <table reference> *TR*, then

1) Case:

A) If *TR* is a <table name>, then let *S* be the set of applicable privileges for *A* on the table referenced by *TR*.

B) Otherwise, *TR* references some VC_j, where $j < i$. Let *S* be the set of view component privilege descriptors whose identified object is VC_j or a column of VC_j.

2) If *S* contains a table privilege descriptor *PD* whose action is DELETE, then a view component table privilege descriptor is created, as follows: the identified object is VC_i, the action is DELETE, the grantor is the special grantor value "_SYSTEM", and the grantee is *A*. The privilege is grantable if and only if *PD* indicates a grantable privilege.

3) For each updatable column *C* of VC_i, let *CC* be the counterpart of *C* in the table identified by *TR*.

A) If *S* contains a column privilege descriptor *PD* whose action is UPDATE (*CC*), then a view component table privilege descriptor is created, as follows: the identified object is *C*, the action is UPDATE, the grantor is the special grantor value "_SYSTEM", and the grantee is *A*. The privilege is grantable if and only if *PD* indicates a grantable privilege. The privilege descriptor is immediately dependent on *PD*.

B) If *V* is insertable-into, and *S* contains a column privilege descriptor *PD* whose action is INSERT (*CC*), then a view component table privilege descriptor is created, as follows: the identified object is *C*, the action is INSERT, the grantor is the special grantor value "_SYSTEM", and the grantee is *A*. The privilege is grantable if and only if *PD* indicates a grantable privilege. The privilege descriptor is immediately dependent on *PD*.

ii) Otherwise:

1) If, for every leaf underlying table *LUT* of VC_i such that VC_i is one-to-one with *LUT*, the applicable privileges for *A* include DELETE on *LUT*, then a view component table privilege descriptor is created whose identified object is VC_i, <action> is DELETE, grantor is the special grantor value "_SYSTEM", and the grantee is *A*. The privilege is grantable if and only if the applicable privileges for *A* includes grantable DELETE privilege on each such *LUT*. The privilege descriptor is immediately dependent on every privilege descriptor whose identified object is such a leaf underlying table, <action> is DELETE, and grantee is *A*.

2) For each updatable column *C* of VC_i, let *LUT* be the leaf underlying table of VC_i that has a counterpart *CC* to *C*.

A) If the applicable privileges for *A* include UPDATE (*CC*) on *LUT*, then a view component column privilege descriptor *VCCPD* is created, as follows: the identified object is *C*, the action is UPDATE, the grantor is the special grantor value "_SYSTEM", and the grantee is *A*. The privilege is grantable if and only if the applicable privilege for *A* includes grantable UPDATE (*CC*) privilege on *LUT*. The privilege descriptor is immediately dependent on every privilege descriptor whose identified object is *CC*, <action> is UPDATE, and grantee is *A*.

B) If *V* is insertable-into, and the applicable privileges for *A* include INSERT (*CC*) on *LUT*, then a view component column privilege descriptor *VCCPD* is created, as follows: the identified object is *C*, the action is INSERT, the grantor is the special grantor value "_SYSTEM", and the grantee is *A*. The privilege is grantable if and only if the applicable privilege for *A* includes grantable INSERT (*CC*) privilege on *LUT*. The privilege descriptor is immediately dependent on every privilege descriptor whose identified object is *CC*, <action> is INSERT, and grantee is *A*

b) If VC_i is a <table value constructor>, then there are no view component privilege descriptors that identify VC_i as object.

c) If VC_i is an <explicit table>, then let *T* be the table identified by the <explicit table>.

i) If the applicable privileges for *A* include DELETE on *T*, then a view component table privilege descriptor is created whose identified object is VC_i, <action> is DELETE, grantor is the special grantor value "_SYSTEM", and the grantee is *A*. The privilege is grantable if and only if the applicable privileges for *A* includes grantable DELETE privilege on *T*. The privilege descriptor is immediately dependent on the privilege descriptor whose identified object is *T*, <action> is DELETE, and grantee is *A*.

ii) For each updatable column *C* of VC_i, let *CC* be the counterpart to *C* in *T*. If the applicable privileges for *A* include UPDATE (*CC*) on *T*, then a view component column privilege descriptor *VCCPD* is created, as follows: the identified object is *C*, the action is UPDATE, the grantor is the special grantor value "_SYSTEM", and the grantee is *A*. The privilege is grantable if and only if the applicable privilege for *A* includes grantable UPDATE (*CC*) privilege on *T*. The privilege descriptor is immediately dependent on the privilege descriptor whose identified object is *CC*, <action> is UPDATE, and grantee is *A*.

iii) For each updatable column *C* of VC_i, let *CC* be the counterpart to *C* in *T*. If the applicable privileges for A include INSERT (*CC*) on *T*, then a view component column privilege descriptor *VCCPD* is created, as follows: the identified object is *C*, the action is INSERT, the grantor is the special grantor value "_SYSTEM", and the grantee is *A*. The privilege is grantable if and only if the applicable privilege for *A* includes grantable INSERT (*CC*) privilege on *T*. The privilege descriptor is immediately dependent on the privilege descriptor whose identified object is *CC*, <action> is INSERT, and\ grantee is *A*.

d) If VC_i is a <query expression>, then

Case:

i) If VC_i is a <simple table> *ST*, then there exists a $j < i$ such that *ST* is VC_j.

1) For each view component table privilege descriptor *VCTPD* of VC_j, a new view component table privilege descriptor is created, as follows: the identified object is VC_i, the grantor is the special grantor value "_SYSTEM", the grantee is *A*, and the <action>, and the indication of whether the privilege is grantable is the same as in *VCTPD*. The privilege descriptor is immediately dependent on *VCTPD*.

2) For each view component column privilege descriptor *VCCPD* of VC_j, a new view component column privilege descriptor of VC_i is created, as follows: the identified object is the column of VC_i that is the counterpart of the column of VC_j identified by *VCCPD*, the grantor is the special grantor value "_SYSTEM", the grantee is *A*, and the indication of whether the privilege is grantable is the same as in *VCCPD*. The privilege descriptor is immediately dependent on *VCTPD*.

ii) Otherwise, VC_i immediately contains UNION ALL. Let VC_l and VC_r be the left and right operands of VC_i, respectively.

1) If there is a view component table privilege descriptor $VCTPD_l$ whose identified object is VC_l and whose <action> is DELETE, and there is a view component table privilege descriptor $VCTPD_r$ whose identified object is VC_r and whose <action> is DELETE, then a new view component table privilege descriptor is created as follows: the identified object is VC_i, the <action> is DELETE, the grantor is the special grantor value "_SYSTEM", the grantee is *A*, and the privilege is grantable if and only if both $VCTPD_l$ and $VCTPD_r$ indicate that the privilege is grantable. The privilege descriptor is immediately dependent on $VCTPD_l$ and $VCTPD_r$.

2) For each updatable column *C* of VC_i, let C_l and C_r be the counterparts of *C* in VC_l and VC_r, respectively. If there is a view component column privilege descriptor $VCTPD_l$ whose identified object is C_l and whose <action> is UPDATE, and there is a view component column privilege descriptor $VCTPD_r$ whose identified object is C_r and whose <action> is UPDATE, then a new view component table privilege descriptor is created as follows: the identified object is *C*, the <action> is UPDATE, the grantor is the special grantor value "_SYSTEM", the grantee is *A*, and the privilege is grantable if and only if both $VCTPD_l$ and $VCTPD_r$ indicate that the privilege is grantable. The privilege descriptor is immediately dependent on $VCTPD_l$ and $VCTPD_r$.

5) A view component privilege descriptor *SPD* is *simply dependent* on another privilege descriptor *PD* if *SPD* is immediately dependent on *PD*, or if there is a view component privilege descriptor *SPD2* such that *SPD* is immediately dependent on *SPD2* and *SPD2* is simply dependent on *PD*.

6) *VCN* is the view component that is the <query expression> immediately contained in the <view definition>.

a) For each view component table privilege descriptor *VCTPD* whose identified object is *VCN*:

i) A privilege descriptor *PD1* is created, as follows: the identified object is *V*, the <action> is the same as the <action> of *VCTPD*, the grantor is the special grantor

value "_SYSTEM", the grantee is *A*, and the privilege is grantable if and only if *VCTPD* is grantable.

ii) For each privilege descriptor *PD* such that *VCTPD* is simply dependent on *PD*, and such that the object of *PD* is not a view component or a column of a view component, a view privilege dependency descriptor is created, as follows: the supporting privilege descriptor is *PD* and the dependent privilege descriptor is *PD1*.

b) For each view component column privilege descriptor *VCCPD* whose identified object is *VCN*:

i) A privilege descriptor *PD2* is created, as follows: the identified object is the column of *V* that is the counterpart of the column identified by *VCCPD*, the <action> is the same as the <action> of *VCCPD*, the grantor is the special grantor value "_SYSTEM", the grantee is *A*, and the privilege is grantable if and only if *VCTPD* is grantable.

ii) For each privilege descriptor *PD* such that *VCCPD* is simply dependent on *PD*, and such that the object of *PD* is not a view component or a column of a view component, a view privilege dependency descriptor is created, as follows: the supporting privilege descriptor is *PD* and the dependent privilege descriptor is *PD2*.

7) If, for every column *C* of *V*, there is a column privilege descriptor *CPD* whose identified object is *C*, <action> is UPDATE, grantor is the special grantor value "_SYSTEM", and grantee is *A*, then it is implementation-defined whether a table privilege descriptor *TPD* is created whose identified object is *V*, <action> is UPDATE, grantor is the special grantor value "_SYSTEM", grantee is *A*. If *TPD* is created, then a collection of view privilege dependency descriptors is created, one for each column *C* of *V*, in which the supporting privilege descriptor is *CPD* and the dependent privilege descriptor is *TPD*.

8) If, for every column *C* of *V*, there is a column privilege descriptor *CPD* whose identified object is *C*, <action> is INSERT, grantor is the special grantor value "_SYSTEM", grantee is *A*, then it is implementation-defined whether a table privilege descriptor *TPD* is created whose identified object is *V*, <action> is INSERT, grantor is the special grantor value "_SYSTEM", grantee is A. If such a table privilege descriptor is created, then it is directly dependent on every such column privilege descriptor, and it has an indication that the privilege is grantable if every such column privilege descriptor has an indication that it is grantable. If *TPD* is created, then a collection of view privilege dependency descriptors is created, one for each column *C* of *V*, in which the supporting privilege descriptor is *CPD* and the dependent privilege descriptor is *TPD*.

9) All view component privilege descriptors are destroyed.

Conformance Rules

None.

10 Additional common elements

10.4 <routine invocation>

1. *Rationale: Avoid misleading references to "current transaction".*

   Replace General Rule 5) b) with:

   5) ...

      b) Set the values of the current SQL-session identifier, the identities of all instances of global temporary tables, the current constraint mode for each integrity constraint, the current access mode, the current isolation level, and the current condition area limit to their values in *CSC*.

2. *Rationale: Clarify the effect on the authorization stack.*

   Replace General Rule 5) j) ii) 1) with:

   5) ...

      j) ...

         ii) ...

            1) If the SQL security characteristic of *R* is DEFINER, then the top cell of the authorization stack of *RSC* is set to contain only the routine authorization identifier of *R*.

3. *Rationale: Clarify the specifications concerning returned result sets.*

   Insert the following General Rule:

   6) ...

      j.1) If the subject routine is an SQL-invoked procedure *SIP*, then let *INV* be the invoker of *SIP*. An empty result set sequence *RSS*, for SQL-invoked procedure *SIP* and invoker *INV*, is added to *RSC*.

4. *Rationale: Clarify the specifications concerning returned result sets.*

   Delete General Rule 8) g) i) 1).

5. *Rationale: Exclude an array-returning function or a multiset-returning function.*

   Replace General Rule 8) h) i) 3) with:

   8) ...

      h) ...

i) ...

3) If *R* is not a null-call function and *R* is not an array-returning function or a multiset-returning function,*R* specifies PARAMETER STYLE SQL, and entry (*PN*+1)+*N*+1 in *ESPL* (that is, SQL indicator argument *N*+1 corresponding to the result data item) is negative, then let *RDI* be the null value.

6. *Rationale: Clarify the specifications concerning returned result sets.*

Replace General Rule 10) with:

10) If *RSS* is not empty, then let *PR* be the descriptor of *SIP*.

a) Let *MAX* be maximum number of returned result sets included in *PR*.

b) Let *OPN* be the actual number of returned result sets included in *RSS*.

c) Case:

i) If *OPN* is greater than *MAX*, then:

1) Let *RTN* be *MAX*.

2) A completion condition is raised: *warning — attempt to return too many result sets*.

ii) Otherwise, let *RTN* be *OPN*.

d) For each *i*, $1 \le i \le RTN$, let FRC_i be the with-return cursor of the *i*-th returned result set RS_i in *RSS* and let $FRCN_i$ be the <cursor name> that identifies FRC_i.

e) Case:

i) If FRC_i is a scrollable cursor, then the initial cursor position of RS_i is the current cursor position of FRC_i.

ii) Otherwise,

Case:

1) If an SQL-statement beginning with:
`FETCH NEXT FROM` $FRCN_i$

would position the cursor on or before some row in RS_i, then let *RN* be the ordinal position of that row in RS_i.

2) Otherwise, let *RN* be one greater than the number of rows in RS_i.

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

f) A completion condition is raised: *warning — result sets returned*.

10.1) If *R* is an external routine, then it is implementation-defined whether, for every open cursor *CR* that is associated with *RSC* and defined by a <declare cursor> that is contained in the <SQL-client module definition> of *P*:

a) The following SQL-statement is effectively executed:
`CLOSE` *`CR`*

b) *CR* is destroyed.

7. *Rationale: Avoid misleading references to "current transaction".*

Replace General Rule 11) b), c) and d) with:

11) ...

b) Set the value of the current access mode in *CSC* to the value of the current access mode in *RSC*.

c) Set the value of the current isolation level in *CSC* to the value of the current isolation level in *RSC*.

d) Set the value of the current condition area limit in *CSC* to the value of the current condition area limit *CAL* in *RSC*.

8. *Rationale: Clarify the specifications concerning returned result sets.*

Insert the following General Rule:

11) ...

g.1) If the subject routine is an SQL-invoked procedure, then the result set sequence *RSS* is added to *CSC*.

NOTE 225.1 — *RSS* is now available for reference by an <allocate cursor statement> containing FOR PROCEDURE.

10.8 <constraint name definition> and <constraint characteristics>

1. *Rationale: Determine the implicit schema name of a constraint name correctly.*

Delete Syntax Rules 1) and 2).

2. *Rationale: Determine the implicit schema name of a constraint name correctly.*

Replace General Rule 1) with:

1) Let *C* be the constraint identified by <constraint name>.

3. *Rationale: Remove specification of when and how constraints are checked.*

Delete General Rules 4) and 5).

10.9 <aggregate function>

1. *Rationale: "<subquery>" should be "<query expression>".*

   Replace Syntax Rule 8) with:

   8) A <filter clause> shall not contain a <query expression>, a <window function>, or an outer reference.

2. *Rationale: "<subquery>" should be "<query expression>".*

   Replace Syntax Rule 10) a) with:

   10) ...

      a) The <hypothetical set function> shall not contain a <window function>, a <set function specification>, or a <query expression>.

3. *Rationale: "<subquery>" should be "<query expression>".*

   Replace Syntax Rule 11) b) with:

   11) ...

      b) The <inverse distribution function> shall not contain a <window function>, a <set function specification>, or a <query expression>.

11 Schema definition and manipulation

11.2 <drop schema statement>

1. *Rationale: A role is not a schema object.*

   Delete General Rule 12).

11.3 <table definition>

1. *Rationale: Prohibit <host parameter specification>s, <SQL parameter reference>s, <dynamic parameter specification>s, and <embedded variable specification>s in DDL.*

   Insert the following Syntax Rule:

0.1) The <table content source> shall not contain a <host parameter specification>, an <SQL parameter reference>, a <dynamic parameter specification>, or an <embedded variable specification>.

2. *Rationale: Editorial.*

Replace Syntax Rule 3) with:

3) *TN* shall not identify an existing table descriptor.

11.4 <column definition>

1. *Rationale: Prohibit <host parameter specification>s, <SQL parameter reference>s, <dynamic parameter specification>s, and <embedded variable specification>s in DDL.*

Insert the following Syntax Rule:

0.1) The <column definition> shall not contain a <host parameter specification>, an <SQL parameter reference>, a <dynamic parameter specification>, or an <embedded variable specification>.

2. *Rationale: "<subquery>" should be "<query expression>".*

Replace Syntax Rule 6) f) with:

6) ...

f) *GE* shall not contain a <query expression>.

3. *Rationale: Record updatability in the column descriptor.*

Insert the following subrule to General Rule 4):

4) ...

h.1) An indication that the column is updatable.

4. *Rationale: Clarify how domain constraints are handled.*

Insert the following General Rule:

4.1) If <domain name> is specified, then, for every domain constraint descriptor *DCD* included in the domain descriptor of *D*, let *DSC* be the template <search condition> included in *DCD*. Let *CSC* be a copy of *DSC* in which every instance of the <general value specification> VALUE is replaced by *C*. A domain constraint usage descriptor is created and added to the set of domain constraint usage descriptors included in *DCD*. The domain constraint usage descriptor created includes:

a) The name of the applicable column.

b) The control <search condition>
`( SELECT EVERY ( CSC ) FROM T )`

NOTE 249.1 — This is a <scalar subquery> of declared type BOOLEAN.

11.6 <table constraint definition>

1. *Rationale: Check for uniqueness of constraint names.*

   Insert the following Syntax Rule:

   3.1) Let *S* be the schema identified by the explicit or implicit <schema name> of the <constraint name>. *S* shall not include a constraint descriptor whose constraint name is <constraint name>.

2. *Rationale: Remove specification of when and how constraints are checked.*

   Replace General Rules 2) and 3) with:

   2) A table constraint descriptor is created that describes the table constraint being defined. The table constraint descriptor includes

      a) The <constraint name> contained in the explicit or implicit <constraint name definition>.

      b) An indication of whether the constraint is deferrable or not deferrable.

      c) An indication of whether the initial constraint mode of the constraint is deferred or immediate.

      d) The <search condition> as specified in the General Rules applicable to the particular kind of <table constraint> contained in the <table constraint definition>, and any additional contents specified by those General Rules.

   3) If the <table constraint> is a <check constraint definition>, then let *SC* be the <search condition> immediately contained in the <check constraint definition> and let *T* be the table name included in the corresponding table constraint descriptor; the <search condition> included in the table constraint descriptor is
   `( SELECT EVERY ( SC ) FROM T )`

   NOTE 253.1 — This is a <scalar subquery> of declared type BOOLEAN.

11.7 <unique constraint definition>

1. *Rationale: Include a <search condition> in the constraint descriptor and delete incorrect information about when constraints are checked.*

   Replace Syntax Rule 3) c) i) with:

   3) ...

      c) ...

         i) If the <unique specification> specifies PRIMARY KEY, then let *SC* be the <search condition>:

```
UNIQUE ( SELECT UCL FROM TN )
AND
( SELECT EVERY ( ( UCL ) IS NOT NULL ) FROM TN )
```

NOTE 253.2 — The second operand of AND in this expression is a <scalar subquery> of declared type BOOLEAN.

2. *Rationale: Include a <search condition> in the constraint descriptor and delete incorrect information about when constraints are checked.*

Replace the General Rules with:

1) A <unique constraint definition> defines a unique constraint.

2) The <search condition> included in the table constraint descriptor created as a result of execution of the containing <table constraint definition> is *SC*. This descriptor additionally includes:

   a) The <unique specification>, indicating whether the constraint is defined with PRIMARY KEY or UNIQUE.

   b) The names of the unique columns specified in the <unique column list>.

11.8 <referential constraint definition>

1. *Rationale: Include a <search condition> in the constraint descriptor and delete incorrect information about when constraints are checked.*

Replace General Rules 1) and 2) with:

1) A <referential constraint definition> defines a referential constraint.

2) Let R_f be the referencing columns in the referencing table *U* and let R_t be the referenced columns in the referenced table *T*. Let *MP* be

   Case:

   a) If SIMPLE is specified or implicit, then
   R_f MATCH SIMPLE (SELECT R_t FROM *T*)

   b) If PARTIAL is specified, then
   R_f MATCH PARTIAL (SELECT R_t FROM *T*)

   c) If FULL is specified, then
   R_f MATCH FULL (SELECT R_t FROM *T*)

2. *Rationale: Include a <search condition> in the constraint descriptor and delete incorrect information about when constraints are checked.*

Insert the following General Rule:

2.1) The <search condition> included in the table constraint descriptor created as a result of executing the containing <table constraint definition> is:
`( SELECT EVERY ( MP ) FROM U )`

NOTE 256.1 — This is a <scalar subquery> of declared type BOOLEAN.

This descriptor additionally includes:

a) A list of the names of the referencing columns specified in the <referencing columns>.

b) The name of the referenced table specified in the <referenced table and columns>.

c) A list of the names of the referenced columns specified in the <referenced table and columns>.

d) The value of the <match type>, if specified.

e) The <referential triggered action>, if specified.

3. *Rationale: Avoid duplicate specification of subtable and supertable handling.*

Replace General Rule 6) with:

6) Let *F* be the referencing table

4. *Rationale: Handle deletions and triggers in Clause 14.*

Replace General Rule 7) a) i) to iii) with:

7) ...

a) ...

i) If the <delete rule> specifies CASCADE, then:

1) *F* is identified for deletion processing and every matching row in *F* is identified for deletion.

2) The General Rules of Subclause 14.16, "Effect of deleting rows from base tables", are applied.

ii) If the <delete rule> specifies SET NULL, then:

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

2) *F* is identified for replacement processing with subtables with respect to the referencing columns.

3) The General Rules of Subclause 14.22, "Effect of replacing rows in base tables", are applied.

iii) If the <delete rule> specifies SET DEFAULT, then:

1) Each matching row *MR* in *F* is paired with the candidate replacement row *NMR*, formed by copying *MR* and setting each referencing column in the copy to the default value specified in the General Rules of Subclause 11.5, "<default

clause>". *MR* is identified for replacement by *NMR* in *F*. The set of (*MR*, *NMR*) pairs is the replacement set for *F*.

2) *F* is identified for replacement processing with subtables with respect to the referencing columns.

3) The General Rules of Subclause 14.22, "Effect of replacing rows in base tables", are applied.

5. *Rationale: Handle deletions and triggers in Clause 14.*

Replace General Rule 7) b) i) to iii) with:

7) ...

b) ...

i) If the <delete rule> specifies CASCADE, then:

1) *F* is identified for deletion processing and every unique matching row in *F* is identified for deletion.

2) The General Rules of Subclause 14.16, "Effect of deleting rows from base tables", are applied.

ii) If the <delete rule> specifies SET NULL, then:

1) Each unique matching row *UMR* in *F* is paired with the candidate replacement row *NUMR*, formed by copying *UMR* and setting each referencing column in the copy to the null value. *UMR* is identified for replacement by *NUMR* in *F*. The set of (*UMR*, *NUMR*) pairs is the replacement set for *F*.

2) *F* is identified for replacement processing with subtables with respect to the referencing columns.

3) The General Rules of Subclause 14.22, "Effect of replacing rows in base tables", are applied.

iii) If the <delete rule> specifies SET DEFAULT, then:

1) Each unique matching row *UMR* in *F* is paired with the candidate replacement row *NUMR*, formed by copying *UMR* and setting each referencing column in the copy to the new value of that referenced column. *UMR* is identified for replacement by *NUMR* in *F*. The set of (*UMR*, *NUMR*) pairs is the replacement set for *F*.

2) *F* is identified for replacement processing with subtables with respect to the referencing columns.

3) The General Rules of Subclause 14.22, "Effect of replacing rows in base tables", are applied.

6. *Rationale: Handle deletions and triggers in Clause 14.*

Replace General Rule 8) a) i) with:

8) ...

a) ...

i) If the <update rule> specifies CASCADE, then:

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

2) *F* is identified for replacement processing with subtables with respect to the referencing columns.

3) The General Rules of Subclause 14.22, "Effect of replacing rows in base tables", are applied.

7. *Rationale: Handle deletions and triggers in Clause 14.*

Replace General Rule 8) a) ii) 1) and 2) with:

8) ...

a) ...

ii) ...

1) If SIMPLE is specified or implicit, then:

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

B) *F* is identified for replacement processing with subtables with respect to the referencing columns.

C) The General Rules of Subclause 14.22, "Effect of replacing rows in base tables", are applied.

2) If <match type> specifies FULL, then:

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

B) *F* is identified for replacement processing with subtables with respect to the referencing columns.

C) The General Rules of Subclause 14.22, "Effect of replacing rows in base tables", are applied.

8. *Rationale: Handle deletions and triggers in Clause 14.*

Replace General Rule 8) a) iii) with:

8) ...

a) ...

iii) If the <update rule> specifies SET DEFAULT, then:

1) Each matching row *MR* in *F* is paired with the candidate replacement row *NMR*, formed by copying *MR* and setting each referencing column in the copy that corresponds with a referenced column to the default value specified in the General Rules of Subclause 11.5, "<default clause>". *MR* is identified for replacement by *NMR* in *F*. The set of (*MR*, *NMR*) pairs is the replacement set for *F*.

2) *F* is identified for replacement processing with subtables with respect to the referencing columns.

3) The General Rules of Subclause 14.22, "Effect of replacing rows in base tables", are applied.

9. *Rationale: Handle deletions and triggers in Clause 14.*

Replace General Rule 8) b) i) to iii) with:

8) ...

b) ...

i) If the <update rule> specifies CASCADE, then:

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

NOTE 260 — Because of the Rules of Subclause 8.2, "<comparison predicate>", on which the definition of "distinct" relies, the values in *C2* may have been updated to values that are not distinct, yet are not identical. Which of these non-distinct values is used for the cascade operation is implementation-dependent.

2) *F* is identified for replacement processing with subtables with respect to the referencing columns.

3) The General Rules of Subclause 14.22, "Effect of replacing rows in base tables", are applied.

ii) If the <update rule> specifies SET NULL, then:

1) Each unique matching row *UMR* in *F* that contains a non-null value in the referencing column in *F* that corresponds with the updated referenced column is paired with the candidate replacement row *NUMR*, formed by copying *UMR* and setting that referencing column in the copy to the null value. *UMR* is identified for replacement by *NUMR* in *F*. The set of (*UMR*, *NUMR*) pairs is the replacement set for *F*.

2) *F* is identified for replacement processing with subtables with respect to the referencing columns.

3) The General Rules of Subclause 14.22, "Effect of replacing rows in base tables", are applied.

iii) If the <update rule> specifies SET DEFAULT, then:

1) Each unique matching row *UMR* in *F* that contains a non-null value in the referencing column in *F* that corresponds with the updated referenced column is paired with the candidate replacement row *NUMR*, formed by copying *UMR* and setting that referencing column in the copy to the default value specified in the General Rules of Subclause 11.5, "<default clause>". *UMR* is identified for replacement by *NUMR* in *F*. The set of (*UMR*, *NUMR*) pairs is the replacement set for *F*.

2) *F* is identified for replacement processing with subtables with respect to the referencing columns.

3) The General Rules of Subclause 14.22, "Effect of replacing rows in base tables", are applied.

10. *Rationale: Handle deletions and triggers in Clause 14.*

Delete General Rules 15) and 16).

11.9 <check constraint definition>

1. *Rationale: Replace inappropriate reference to <target specification>.*

Replace Syntax Rule 1) with:

1) The <search condition> shall not contain a <host parameter specification>, an <SQL parameter reference>, a <dynamic parameter specification>, or an <embedded variable specification>.

2. *Rationale: "<subquery>" should be "<query expression>".*

Replace Syntax Rule 2) with:

2) The <search condition> shall not contain a <set function specification> that is not contained in a <query expression>.

3. *Rationale: "<subquery>" should be "<query expression>".*

   Replace Syntax Rule 4) with:

   4) If the <check constraint definition> is contained in a <table definition> that defines a temporary table and specifies ON COMMIT PRESERVE ROWS or a <temporary table declaration> that specifies ON COMMIT PRESERVE ROWS, then the <search condition> shall not contain a reference to a temporary table defined by a <table definition> or a <temporary table declaration> that specifies ON COMMIT DELETE ROWS.

4. *Rationale: "<subquery>" should be "<query expression>".*

   Replace Conformance Rule 1) with:

   1) Without Feature F671, "Subqueries in CHECK constraints", conforming SQL language shall not contain a <search condition> contained in a <check constraint definition> that contains a <query expression>.

5. *Rationale: Remove specification of when and how constraints are checked.*

   Delete NOTE 265.

11.18 <drop column definition>

1. *Rationale: Clarify the affect of <drop column definition> when restrict is specified, the the affected column is referenced by a multi-column <unique constraint definition>.*

   Replace Note 271 with:

   NOTE 271 — A <drop column definition> whose <drop behaviour> is RESTRICT 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>), a multi-column <unique constraint definition>, or REFERENCES without a <reference column list> in its <referenced table and columns>. Also note that a <drop column definition> whose <drop behaviour> is RESTRICT that references a column that is referenced by a multi-column <unique constraint definition> will fail while dropping a column that is referenced by a single-column <unique constraint definition> will succeed.

2. *Rationale: Clarify how domain constraints are handled.*

   Replace General Rule 4) with:

   4) If the column is not based on a domain, then its data type descriptor is destroyed; otherwise, all domain constraint usages which reference *C* are destroyed.

11.22 <view definition>

1. *Rationale: Deletion of rule referring to a non-existent concept.*

   Delete Syntax Rule 1)

2. *Rationale: Deletion of rule that prohibits what is impossible anyway.*

   Delete Syntax Rule 2)

3. *Rationale: Addition of <host parameter specification> to list of prohibited items.*

   Replace Syntax Rule 3) with

   3) The <query expression> shall not contain a <host parameter specification>, an <SQL parameter reference>, a <dynamic parameter specification>, or an <embedded variable specification>.

4. *Rationale: For SQL-92 compatability, define simply updatable views and effectively updatable views.*

   Insert the following Syntax Rules:

   10.1) The viewed table is *simply updatable* if the <query expression> is simply updatable.

   10.2) The viewed table is *effectively updatable* if it is simply updatable, or if the SQL-implementation supports Feature T111, "Updatable joins, unions and columns" and the viewed table is updatable.

5. *Rationale: Indicate which columns of a view are updatable.*

   Insert the following subrule to General Rule 1):

   1) ...

      c.1) In each column descriptor, an indication that the column is updatable if *V* is effectively updatable and the corresponding column of *QE* is updatable.

6. *Rationale: Use the correct rules to determine INSERT, UPDATE and DELETE privileges on views.*

   Replace General Rule 6) with:

   6) If *V* is effectively updatable, then a set of privilege descriptors is created by applying the General Rules of Subclause 9.24, "Determination of view and view component privileges", with *V* as the *VIEW*.

7. *Rationale: SQL-92 compatability.*

   Insert the following Conformance Rule:

   5) Without Feature T111, "Updatable joins, unions and columns", in conforming SQL language, if WITH CHECK OPTION is specified, then the viewed table shall be simply updatable.

11.24 <domain definition>

1. *Rationale: Check for uniqueness of constraint names.*

   Insert the following Syntax Rule:

   8) ...

      b.1) Let *S* be the schema identified by the <schema name> of the explicit or implicit <constraint name> contained in <domain constraint>. *S* shall not include a constraint descriptor whose constraint name is <constraint name>.

2. *Rationale: Clarify how domain constraints are handled.*

   Replace General Rules 5) and 6) with:

   5) A domain constraint descriptor is created that describes the domain constraint being defined. The domain constraint descriptor includes:

      a) The <constraint name> contained in the explicit or implicit <constraint name definition>.

      b) An indication of whether the constraint is deferrable or not deferrable.

      c) An indication of whether the initial constraint mode of the constraint is deferred or immediate.

      d) The <search condition> contained in the <domain definition> as the template <search condition>.

      e) The appropriate <search condition>:
      `( 1=1 )`

11.30 <drop domain statement>

1. *Rationale: Clarify how domain constraints are handled.*

   Replace General Rule 1) d) with:

   1) ...

      d) For every domain constraint descriptor *DCD* included in the domain descriptor of *D* whose <constraint name> is not contained in the excluded constraint list then: for every domain constraint usage descriptor *DCU* included in *DCD*:

         i) Let *SC* be the appropriate <search condition> included in *DCU*.

         ii) Let *TCD* be a <table constraint definition> consisting of a <constraint name definition> whose <constraint name> is implementation-dependent, and whose <constraint characteristics> are the <constraint characteristics> of the domain constraint descriptor, and whose <table constraint> is
         `CHECK ( SC )`

iii) If the applicable privileges for *UA* include all of the privileges necessary for *UA* to successfully execute the <alter table statement>
`ALTER TABLE` *`TN`* `ADD` *`TCD`*
then the following <alter table statement> is effectively executed with a current authorization identifier of *UA*:
`ALTER TABLE` *`TN`* `ADD` *`TCD`*

11.37 <assertion definition>

1. *Rationale: Delete a redundant rule.*

   Delete Syntax Rule 8).

2. *Rationale: Remove specification of when and how constraints are checked.*

   Delete NOTE 289.

3. *Rationale: Remove specification of when and how constraints are checked.*

   Replace General Rules 3) and 4) with:

   4) An assertion descriptor is created that describes the assertion being defined. This descriptor includes:

      a) <constraint name>.

      b) Whether the constraint is deferrable, as specified in <constraint characteristics>.

      c) The initial constraint mode specified in <constraint characteristics>.

      d) The applicable <search condition> *SC*.

11.41 <user-defined type definition>

1. *Rationale: Correct an erroneous mix-up of <cast to distinct>, <cast to source>, <cast to ref>, and <cast to type>.*

   Replace Syntax Rule 7) f) with:

   7) ...

      f) Neither <cast to ref> nor <cast to type> shall be specified.

2. *Rationale: Correct an erroneous mix-up of <cast to source> and <cast to type>.*

   Replace Syntax Rule 8) a) with:

8) ...

a) Neither <cast to distinct> nor <cast to source> shall be specified.

3. *Rationale: Correct an erroneous mix-up of <cast to distinct>, <cast to source>, <cast to ref>, and <cast to type>.*

Replace the lead text of Syntax Rule 8) k) with:

8) ...

k) If <cast to ref> or <cast to type> is specified, then exactly one of the following shall be true:

11.42 <attribute definition>

1. *Rationale: Remove an incorrect double negative.*

Replace Conformance Rule 4) with:

4) Without Feature S026, "Self-referencing structured types", conforming SQL language shall not contain a <data type> simply contained in an <attribute definition> that is a <reference type> whose <referenced type> is equivalent to the <schema-resolved user-defined type name> simply contained in the <user-defined type definition> that contains <attribute definition>.

11.50 <SQL-invoked routine>

1. *Rationale: Clarify the specifications concerning returned result sets.*

Replace Syntax Rule 6) d) with:

6) ...

d) If the SQL-invoked routine is an SQL-invoked procedure and <dynamic result sets characteristic> is not specified, then DYNAMIC RESULT SETS 0 (zero) is implicit.

2. *Rationale: Clarify the specifications concerning returned result sets.*

Replace General Rule 3) d) with:

3) ...

d) The maximum number of result sets included in the routine descriptor is

Case:

i) If the SQL-invoked routine is an SQL-invoked procedure, then the explicit or implicit value of <maximum dynamic result sets>.

ii) Otherwise, 0 (zero).

3. *Rationale: An external routine does not have two authorization identifiers.*

   Delete General Rule 6) a) i) 2).

4. *Rationale: An external routine does not have two authorization identifiers.*

   Replace General Rule 6) a) ii) with:

   6) ...

      a) ...

         ii) Otherwise, the external routine SQL-path is implementation-defined.

11.51 <alter routine statement>

1. *Rationale: Reference the correct BNF term, <dynamic result sets characteristic>.*

   Replace Syntax Rules 7) and 8) with:

   7) <alter routine characteristics> shall contain at most one <language clause>, at most one <parameter style clause>, at most one <SQL-data access indication>, at most one <null-call clause>, at most one <dynamic result sets characteristic>, and at most one <external routine name>.

   8) If <dynamic result sets characteristic> is specified, then *SR* shall be an SQL-invoked procedure.

12 Access control

12.1 <grant statement>

1. *Rationale: Editorial.*

   Replace General Rule 4) e) iii) with:

   4) ...

      e) iii) The following <grant statement> is effectively executed as though the current user identifier were "_SYSTEM" and without further Access Rule checking:
      `GRANT REFERENCES ON V TO G WGO`

2. *Rationale: Use the correct rules to determine INSERT, UPDATE and DELETE privileges on views.*

   Replace General Rule 5) with:

   5) Following the successful execution of the <grant statement>, for every table *T* specified by some involved privilege descriptor and for every updatable view *V* owned by some grantee *G* such that *T* is some leaf underlying table of the <query expression> of *V*, the General Rules of Subclause 9.24, "Determination of view and view component privileges" are applied, with *V* as the *VIEW*.

3. *Rationale: Use the correct rules to determine INSERT, UPDATE and DELETE privileges on views.*

   Insert a new General Rule:

   11) Redundant duplicate view privilege dependency descriptors are removed from the collection of all view privilege dependency descriptors.

12.2 <grant privilege statement>

1. *Rationale: Inappropriate reference to run-time values in Syntax Rules.*

   Delete Syntax Rules 2), 3), and 4).

2. *Rationale: Inappropriate reference to run-time values in Syntax Rules.*

   Delete Access Rule 1).

3. *Rationale: Inappropriate reference to run-time values in Syntax Rules.*

   Insert the following General Rules:

   0.1) Case:

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

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

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

   0.3) If the applicable privileges for *A* do not include a privilege identifying *O*, then an exception condition is raised: *privilege not granted*.

   0.4) A set of privilege descriptors is identified. The privilege descriptors identified are those defining, for each <action> explicitly or implicitly in <privileges>, that <action> on *O* held by *A* with grant option.

4. *Rationale: Editorial.*

   Replace General Rule 7) a) with:

7) ...

a) If the privilege is grantable, then let *WGO* be "WITH GRANT OPTION".

12.3 <privileges>

1. *Rationale: Inappropriate reference to run-time values in Syntax Rules.*

   Delete General Rule 1).

12.4 <role definition>

1. *Rationale: A role is not a schema object.*

   Replace General Rule 1) with:

   1) A role descriptor whose role name is <role name> is created in the SQL-environment.

2. *Rationale: Inappropriate reference to run-time values in Syntax Rules.*

   Delete General Rules 2) and 3).

3. *Rationale: Inappropriate reference to run-time values in Syntax Rules.*

   Insert the following General Rules:

   0.1) Case:

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

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

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

12.5 <grant role statement>

1. *Rationale: Correction of terminology in Syntax Rule.*

   Replace Syntax Rule 1) with:

   1) No role identified by a specified <grantee> shall be applicable for any role identified by a specified <role granted>; that is, no cycles of role authorizations are allowed.

2. *Rationale: Inappropriate reference to run-time values in Syntax Rules.*

   Delete Syntax Rules 2) and 3).

3. *Rationale: Inappropriate reference to run-time values in Syntax Rules.*

   Delete Access Rule 1).

4. *Rationale: Inappropriate reference to run-time values in Syntax Rules.*

   Insert the following General Rules:

   0.1) Case:

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

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

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

   0.3) For each <role granted> *R*, if no grantable role authorization descriptor exists whose role name is *R* and whose grantee is *A* or an applicable role for *A*, then an exception condition is raised: *invalid role specification*.

5. *Rationale: Use of inappropriate terminology.*

   Replace General Rules 1), 2), 3) and 4) with:

   1) For each <grantee>, *GEE*, for each <role granted>, *R*, a role authorization descriptor is created with role name *R*, grantee *GEE*, and grantor *A*.

   2) If WITH ADMIN OPTION is specified, then each role authorization descriptor is grantable.

   3) If two role authorization descriptors are identical except that one is grantable and the other is not, then both role authorization descriptors are set to indicate that the role authorization is grantable.

   4) Redundant duplicate role authorization descriptors are destroyed.

6. *Rationale: Correction of terminology in General Rule.*

   Replace General Rule 6) with:

   6) The set of involved grantees is the union of the set of <grantee>s and the set of <role name>s for which at least one of the <role name>s that is possibly specified as a <grantee> is applicable.

12.6 <drop role statement>

1. *Rationale: Use more appropriate terminology.*

   Replace Access Rule 1) with:

   1) There shall exist at least one grantable role authorization descriptor whose role name is *R* and whose grantee is an enabled authorization identifier.

12.7 <revoke statement>

1. *Rationale: Inappropriate reference to run-time values in Syntax Rules.*

   Delete Syntax Rules 3), 4), and 9) to 37).

2. *Rationale: Inappropriate reference to run-time values in Syntax Rules.*

   Delete Access Rule 1).

3. *Rationale: Inappropriate reference to run-time values in Syntax Rules.*

   Insert the following General Rules:

   0.1) Case:

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

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

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

   0.3) Case:

      a) If the <revoke statement> is a <revoke privilege statement>, and the applicable privileges for *A* do not include a privilege identifying *O*, then an exception condition is raised: *invalid grantor*.

      b) If the <revoke statement> is a <revoke role statement>, then, for every role *R* identified by a <role revoked>, if the applicable roles of *A* do not include a role *AR* such that there exists a grantable role authorization descriptor with role *R* and grantee *A*, then an exception condition is raised: *invalid grantor*.

   0.4) Case:

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

NOTE 337 — Column privilege descriptors become identified when <action> explicitly or implicitly contains a <privilege column list>. Table/method descriptors become identified when <action> explicitly or implicitly contains a <privilege method list>.

b) If the <revoke statement> is a <revoke role statement>, then, for every <grantee> specified, a set of role authorization descriptors is identified. A role authorization descriptor is said to be *identified* if it defines the grant of any of the specified <role revoked>s to <grantee> with grantor *A*.

4. *Rationale: Use the correct rules to determine INSERT, UPDATE, and DELETE privileges on views. Inappropriate reference to run-time values in Syntax Rules.*

Insert the following General Rule:

0.5) A privilege descriptor *D* is said to be *directly dependent* on another privilege descriptor *P* if

Case:

a) If *D* identifies a view or a column of a view and the <action> of *D* is INSERT, UPDATE, or DELETE, then *D* is directly dependent on *P* if there exists a view privilege dependency descriptor whose supporting privilege descriptor is *P* and whose dependent privilege descriptor is *D*.

b) Otherwise, one of the following is true:

i) All of the following conditions hold:

1) *P* indicates that the privilege that it represents is grantable.

2) The grantee of *P* is the same as the grantor of *D*, or the grantee of *P* is PUBLIC, or, if the grantor of *D* is a <role name>, the grantee of *P* is an applicable role for the grantor of *D*.

3) Case:

A) *P* and *D* are both column privilege descriptors. The action and the identified column of *P* are the same as the action and identified column of *D*, respectively.

B) *P* and *D* are both table privilege descriptors. The action and the identified table of *P* are the same as the action and the identified table of *D*, respectively.

C) *P* and *D* are both execute privilege descriptors. The action and the identified SQL-invoked routine of *P* are the same as the action and the identified SQL-invoked routine of *D*, respectively.

D) *P* and *D* are both usage privilege descriptors. The action and the identified domain, character set, collation, transliteration, user-defined type, or sequence generator of *P* are the same as the action and the identified domain, character set, collation, transliteration, user-defined type, or sequence generator of *D*, respectively.

E) *P* and *D* are both under privilege descriptors. The action and the identified user-defined type or table of *P* are the same as the action and the identified user-defined type or table of *D*, respectively.

F) *P* and *D* are both table/method privilege descriptors. The action and the identified method and table of *P* are the same as the action and the identified method and table of *D*, respectively.

ii) All of the following conditions hold:

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

2) The action of *P* is the same as the action of *D*.

3) The grantee of *P* is the owner of the table, collation, or transliteration identified by *D* or the grantee of *P* is PUBLIC.

4) One of the following conditions hold:

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

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

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

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

C) The privilege descriptor *D* identifies the <transliteration name> of a <transliteration definition> *TD* and the identified character set name of *P* is contained in the <source character set specification>, or the <target character set specification> immediately contained in *TD*.

iii) All of the following conditions hold:

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

2) The grantee of *P* is the owner of the domain identified by *D* or the grantee of *P* is PUBLIC.

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

<domain name>, <collation name>, <character set name> or <transliteration name>, respectively, is contained in the <search condition> of the domain descriptor for *DO*.

5. *Rationale: Inappropriate reference to run-time values in Syntax Rules.*

Insert the following General Rules:

0.6) The *privilege dependency graph* is a directed graph such that all of the following are true:

a) Each node represents a privilege descriptor.

b) Each arc from node *P1* to node *P2* represents the fact that *P2* directly depends on *P1*.

An *independent node* is a node that has no incoming arcs.

0.7) A privilege descriptor *P* is said to be *modified* if all of the following are true:

a) *P* indicates that the privilege that it represents is grantable.

b) *P* directly depends on an identified privilege descriptor or a modified privilege descriptor.

c) Case:

i) If *P* is neither a SELECT nor a REFERENCES column privilege descriptor that identifies a <column name> *CVN* explicitly or implicitly contained in the <view column list> of a <view definition> *V*, then let *XO* and *XA* respectively be the identifier of the object identified by a privilege descriptor *X* and the action of *X*. Within the set of privilege descriptors upon which *P* directly depends, there exist some *XO* and *XA* for which the set of identified privilege descriptors unioned with the set of modified privilege descriptors include all privilege descriptors specifying the grant of *XA* on *XO* WITH GRANT OPTION.

ii) If *P* is a column privilege descriptor that identifies a column *CV* identified by a <column name> *CVN* explicitly or implicitly contained in the <view column list> of a <view definition> *V* with an action *PA* of REFERENCES or SELECT, then let *SP* be the set of privileges upon which *P* directly depends. For every table *T* identified by a <table reference> contained in the <query expression> of *V*, let *RT* be the <table name> of *T*. There exists a column *CT* whose <column name> is *CRT*, such that all of the following are true:

1) *CT* is a column of *T* and an underlying column of *CV*.

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

d) At least one of the following is true:

i) GRANT OPTION FOR is specified and the grantor of *P* is the special grantor value "_SYSTEM".

ii) There exists a path to *P* from an independent node that includes no identified or modified privilege descriptors. *P* is said to be a *marked modified privilege descriptor*.

iii) *P* directly depends on a marked modified privilege descriptor, and the grantor of *P* is the special grantor value "_SYSTEM". *P* is said to be a *marked modified privilege descriptor*.

6. *Rationale: Correction of Syntax Rules. Inappropriate reference to run-time values in Syntax Rules.*

Insert the following General Rules:

0.8) A role authorization descriptor *D* is said to be *directly dependent* on another role authorization descriptor *RD* if all of the following conditions hold:

a) *RD* indicates that the role that it represents is grantable.

b) The role name of *D* is the same as the role name of *RD*.

c) The grantee of *RD* is the same as the grantor of *D*, or the grantee of *RD* is PUBLIC, or, if the grantor of *D* is a <role name>, the grantee of *RD* is an applicable role for the grantor of *D*.

7. *Rationale: Inappropriate reference to run-time values in Syntax Rules.*

Insert the following General Rules:

0.9) The *role dependency graph* is a directed graph such that all of the following are true:

a) Each node represents a role authorization descriptor.

b) Each arc from node *R1* to node *R2* represents the fact that *R2* directly depends on *R1*.

An independent node is one that has no incoming arcs.

0.10) A role authorization descriptor *RD* is said to be *abandoned* if it is not an independent node, and it is not itself an identified role authorization descriptor, and there exists no path to *RD* from any independent node other than paths that include an identified role authorization descriptor.

8. *Rationale: Correction of Syntax Rules. Inappropriate reference to run-time values in Syntax Rules.*

Insert the following General Rules:

0.11) An arc from a node *P* to a node *D* of the privilege dependency graph is said to be *unsupported* if all of the following are true:

a) The grantor of *D* and the grantee of *P* are both <role name>s.

b) The destruction of all abandoned role authorization descriptors and, if ADMIN OPTION FOR is not specified, all identified role authorization descriptors would result in the grantee of *P* no longer being an applicable role for the grantor of *D*.

9. *Rationale: Inappropriate reference to run-time values in Syntax Rules.*

Insert the following General Rules:

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

Case:

a) It is not an independent node, and *P* is not itself an identified or a modified privilege descriptor, and there exists no path to *P* from any independent node other than paths that include an identified privilege descriptor or a modified privilege descriptor or an unsupported arc, and, if <revoke statement> specifies WITH HIERARCHY OPTION, then *P* has the WITH HIERARCHY OPTION.

b) All of the following conditions hold:

   i) *P* is a column privilege descriptor that identifies a <column name> *CVN* explicitly or implicitly contained in the <view column list> of a <view definition> *V*, with an action *PA* of REFERENCES or SELECT.

   ii) Letting *SP* be the set of privileges upon which *P* directly depends, at least one of the following is true:

      1) There exists some table name *RT* such that all of the following are true:

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

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

      2) There exists some column name *CRT* such that all of the following are true:

         A) *CRT* is the name of some column of the table identified by some <table reference> contained in the <query expression> of *V*.

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

10. *Rationale: Inappropriate reference to run-time values in Syntax Rules. <revoke role statement> uses ADMIN instead of GRANT.*

Insert the following General Rules:

0.13) The *revoke destruction action* is defined as

Case:

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

   Case:

   i) If the <revoke statement> specifies the WITH HIERARCHY OPTION, then the removal of the WITH HIERARCHY OPTION from all identified and abandoned privilege descriptors.

   ii) Otherwise, the destruction of all abandoned privilege descriptors and, if GRANT OPTION FOR is not specified, all identified privilege descriptors.

b) If the <revoke statement> is a <revoke role statement>, then the destruction of all abandoned role authorization descriptors, all abandoned privilege descriptors and, if ADMIN OPTION FOR is not specified, all identified role authorization descriptors.

11. *Rationale: Inappropriate reference to run-time values in Syntax Rules.*

Insert the following General Rules:

0.14) Let *S1* be the name of any schema and *A1* be the <authorization identifier> that owns the schema identified by *S1*.

0.15) Let *V* be any view descriptor included in *S1*. Let *QE* be the <query expression> of *V*. *V* is said to be *abandoned* if the revoke destruction action would result in *A1* no longer having in its applicable privileges any of the following:

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

b) SELECT privilege on every column identified by a <column reference> contained in *QE*.

c) USAGE privilege on every domain, every collation, every character set, and every transliteration whose names are contained in *QE*.

d) USAGE privilege on any user-defined type *UDT* such that some <data type> contained in *V* is usage-dependent on *UDT*.

e) EXECUTE privilege on every SQL-invoked routine that is the subject routine of any <routine invocation>, <method invocation>, <static method invocation>, or <method reference> that is contained in *QE*.

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

g) SELECT privilege on any column identified by a <column reference> contained in the <scalar subquery> that is equivalent to some <dereference operation> contained in *QE*.

h) SELECT privilege WITH HIERARCHY OPTION on at least one supertable of the scoped table of any <reference resolution> that is contained in *QE*.

i) SELECT privilege on the scoped table of any <reference resolution> that is contained in *QE*.

j) If *V* is the descriptor of a referenceable table, then USAGE privilege on the structured type associated with the view described by *V*.

k) UNDER privilege on every direct supertable of the view described by *V*.

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

0.16) Let *T* be any table descriptor included in *S1*. *T* is said to be *abandoned* if the revoke destruction action would result in *A1* no longer having any of the following:

a) If *T* is the descriptor of a referenceable table, then USAGE privilege on the structured type associated with the table described by *T*.

b) UNDER privilege on every direct supertable of the table described by *T*.

12. *Rationale: Clarify how items might be included in constraint descriptors. Inappropriate reference to run-time values in Syntax Rules.*

Insert the following General Rules:

0.17) Let *TC* be any table constraint descriptor included in *S1*. *TC* is said to be *abandoned* if the revoke destruction action would result in *A1* no longer having in its applicable privileges any of the following:

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

b) REFERENCES privilege on every column identified by a <column reference> contained in the <search condition> of *TC*.

c) USAGE privilege on every domain, every collation, every character set, and every transliteration whose names are contained in any <search condition> of *TC*.

d) USAGE privilege on any user-defined type *UDT* such that some <data type> contained in the <search condition> of *TC* is usage-dependent on *UDT*.

e) EXECUTE privilege on every SQL-invoked routine that is the subject routine of any <routine invocation>, <method invocation>, <static method invocation>, or <method reference> that is contained in any <search condition> of *TC*.

f) The table/method privilege on every table *T1* and every method *M* such that there is a <method reference> *MR* contained in any <search condition> of *TC* such that *T1* is in the scope of the <value expression primary> of *MR* and *M* is the subject routine of *MR*.

g) SELECT privilege on any column identified by a <column reference> contained in the <scalar subquery> that is equivalent to some <dereference operation> contained in any <search condition> of *TC*.

h) SELECT privilege WITH HIERARCHY OPTION on at least one supertable of the scoped table of any <reference resolution> that is contained in any <search condition> of *TC*.

i) SELECT privilege on the scoped table of any <reference resolution> that is contained in any <search condition> of *TC*.

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

0.18) Let *AX* be any assertion descriptor included in *S1*. *AX* is said to be *abandoned* if the revoke destruction action would result in *A1* no longer having in its applicable privileges any of the following:

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

b) REFERENCES privilege on every column identified by a <column reference> contained in the <search condition> of *AX*.

c) USAGE privilege on every domain, every collation, every character set, and every transliteration whose names are contained in any <search condition> of *AX*.

d) USAGE privilege on any user-defined type UDT such that some <data type> contained in the <search condition> of *AX* is usage-dependent on *UDT*.

e) EXECUTE privilege on eery SQL-invoked routine that is the subject routine of any <routine invocation>, <method invocation>, <static method invocation>, or <method reference> that is contained in any <search condition> of *AX*.

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

g) SELECT privilege on any column identified by a <column reference> contained in the <scalar subquery> that is equivalent to some <dereference operation> contained in any <search condition> of *AX*.

h) SELECT privilege WITH HIERARCHY OPTION on at least one supertable of the scoped table of any <reference resolution> that is contained in any <search condition> of *AX*.

i) SELECT privilege on the scoped table of any <reference resolution> that is contained in any <search condition> of *AX*.

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

13. *Rationale: Inappropriate reference to run-time values in Syntax Rules.*

Insert the following General Rules:

0.19) Let *TR* be any trigger descriptor included in *S1*. *TR* is said to be *abandoned* if the revoke destruction action would result in *A1* no longer having in its applicable privileges any of the following:

a) TRIGGER privilege on the subject table of *TR*.

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

c) SELECT privilege on every column identified by a <column reference> contained in any <search condition> of *TR*.

d) USAGE privilege on every domain, collation, character set, and transliteration whose name is contained in any <search condition> of *TR*.

e) USAGE privilege on any user-defined type *UDT* such that some <data type> contained in any <search condition> of *TR* is usage-dependent on *UDT*.

f) The table/method privilege on every table *T1* and every method *M* such that there is a <method reference> *MR* contained in any <search condition> of *TR* such that *T1* is in the scope of the <value expression primary> of *MR* and *M* is the subject routine of *MR*.

g) EXECUTE privilege on the SQL-invoked routine that is the subject routine of any <routine invocation>, <method invocation>, <static method invocation>, or <method reference> that is contained in any <search condition> of *TR*.

h) EXECUTE privilege on the SQL-invoked routine that is the subject routine of any <routine invocation>, <method invocation>, <static method invocation>, or <method reference> that is contained in the <triggered SQL statement> of *TR*.

i) SELECT privilege on at least one column of every table identified by a <table reference> contained in a <query expression> simply contained in a <cursor specification>, an <insert statement>, or a <merge statement> contained in the <triggered SQL statement> of *TR*.

j) SELECT privilege on at least one column of every table identified by a <table reference> contained in a <table expression> or <select list> immediately contained in a <select statement: single row> contained in the <triggered SQL statement> of *TR*.

k) SELECT privilege on at least one column of every table identified by a <table reference> and <column reference> contained in a <search condition> contained in a <delete statement: searched>, an <update statement: searched>, or a <merge statement> contained in the <triggered SQL statement> of *TR*.

l) SELECT privilege on at least one column of every table identified by a <table reference> and <column reference> contained in a <value expression> simply contained in a an <update source> or an <assigned row>contained in the <triggered SQL statement> of *TR*.

m) INSERT privilege on every column

   Case:

   i) Identified by a <column name> contained in the <insert column list> of an <insert statement> or a <merge statement> contained in the <triggered SQL statement> of *TR*.

   ii) Of the table identified by the <table name> immediately contained in an <insert statement> that does not contain an <insert column list> and that is contained in the <triggered SQL statement> of *TR*.

   iii) Of the table identified by the <target table> contained in a <merge statement> that contains a <merge insert specification> and that does not contain an <insert column list> and that is contained in the <triggered SQL statement> of *TR*.

n) UPDATE privilege on every column identified by a <column name> is contained in an <object column> contained in either an <update statement: positioned>, an <update statement: searched>, or a <merge statement> contained in the <triggered SQL statement> of *TR*.

o) DELETE privilege on every table identified by a <table name> contained in either a <delete statement: positioned> or a <delete statement: searched> contained in the <triggered SQL statement> of *TR*.

p) USAGE privilege on every domain, collation, character set, transliteration, and sequence generator whose name is contained in the <triggered SQL statement> of *TR*.

q) USAGE privilege on any user-defined type *UDT* such that some <data type> contained in the <triggered SQL statement> of *TR* is usage-dependent on *UDT*.

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

s) SELECT privilege on any column identified by a <column reference> contained in the <scalar subquery> that is equivalent to some <dereference operation> contained in any of the following:

   i) A <search condition> of *TR*.

   ii) A <query expression> simply contained in a <cursor specification>, an <insert statement>, or a <merge statement> contained in the <triggered SQL statement> of *TR*.

   iii) A <table expression> or <select list> immediately contained in a <select statement: single row> contained in the <triggered SQL statement> of *TR*.

   iv) A <search condition> contained in a <delete statement: searched>, an <update statement: searched>, or a <merge statement> contained in the <triggered SQL statement> of *TR*.

   v) A <value expression> contained in an <update source> or an <assigned row> contained in the <triggered SQL statement> of *TR*.

t) SELECT privilege WITH HIERARCHY OPTION on at least one supertable of the scoped table of any <reference resolution> that is contained in any of the following:

   i) A <search condition> of *TR*.

   ii) A <query expression> simply contained in a <cursor specification>, an <insert statement>, or a <merge statement> contained in the <triggered SQL statement> of *TR*.

   iii) A <table expression> or <select list> immediately contained in a <select statement: single row> contained in the <triggered SQL statement> of *TR*.

   iv) A <search condition> contained in a <delete statement: searched>, an <update statement: searched>, or a <merge statement> contained in the <triggered SQL statement> of *TR*.

   v) A <value expression> contained in an <update source> or an <assigned row> contained in the <triggered SQL statement> of *TR*.

u) SELECT privilege on the scoped table of any <reference resolution> contained in any of the following:

   i) A <search condition> of *TR*.

   ii) A <query expression> simply contained in a <cursor specification>, an <insert statement>, or a <merge statement> contained in the <triggered SQL statement> of *TR*.

   iii) A <table expression> or <select list> immediately contained in a <select statement: single row> contained in the <triggered SQL statement> of *TR*.

   iv) A <search condition> contained in a <delete statement: searched>, an <update statement: searched>, or a <merge statement> contained in the <triggered SQL statement> of *TR*.

   v) A <value expression> contained in an <update source> or an <assigned row> contained in the <triggered SQL statement> of *TR*.

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

0.20) Let *DC* be any domain constraint descriptor included in *S1*. *DC* is said to be *abandoned* if the revoke destruction action would result in *A1* no longer having in its applicable privileges any of the following:

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

b) REFERENCES privilege on every column identified by a <column reference> contained in the <search condition> of *DC*.

c) USAGE privilege on every domain, every user-defined type, every collation, every character set, and every transliteration whose names are contained in any <search condition> of *DC*.

d) USAGE privilege on any user-defined type *UDT* such that some <data type> contained in any <search condition> of *DC* is usage-dependent on *UDT*.

e) EXECUTE privilege on every SQL-invoked routine that is the subject routine of any <routine invocation>, <method invocation>, <static method invocation>, or <method reference> that is contained in any <search condition> of *DC*.

f) The table/method privilege on every table *T1* and every method *M* such that there is a <method reference> *MR* contained in any <search condition> of *DC* such that *T1* is in the scope of the <value expression primary> of *MR* and *M* is the subject routine of *MR*.

g) SELECT privilege on any column identified by a <column reference> contained in a <scalar subquery> that is equivalent to some <dereference operation> contained in any <search condition> of *DC*.

h) SELECT privilege WITH HIERARCHY OPTION on at least one supertable of the scoped table of any <reference resolution> that is contained in any <search condition> of *DC*.

i) SELECT privilege on the scoped table of any <reference resolution> that is contained in contained in any <search condition> of *DC*.

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

0.21) For every domain descriptor *DO* included in *S1*, *DO* is said to be *lost* if the revoke destruction action would result in *A1* no longer having in its applicable privileges USAGE privilege on every character set included in the data type descriptor included in *DO*.

0.22) For every table descriptor *TD* contained in *S1*, for every column descriptor *CD* included in *TD*, *CD* is said to be *lost* if any of the following are true:

a) The revoke destruction action would result in *A1* no longer having in its applicable privileges USAGE privilege on any character set included in the data type descriptor included in *CD*.

b) The revoke destruction action would result in *A1* no longer having in its applicable privileges USAGE privilege on any user-defined type *UDT* such that a data type descriptor included in *CD* describes a type that is usage-dependent on *UDT*.

c) The name of the domain *DN* included in *CD*, if any, identifies a lost domain descriptor and the revoke destruction action would result in *A1* no longer having in its applicable privileges USAGE privilege on any character set included in the data type descriptor of the domain descriptor of *DN*.

0.23) For every SQL-client module *MO*, let *G* be the <module authorization identifier> that owns *MO*. *MO* is said to be *lost* if the revoke destruction action would result in *G* no longer having in its applicable privileges USAGE privilege on the character set referenced in the <module character set specification> of *MO*.

0.24) For every user-defined type descriptor *DT* included in *S1*, *DT* is said to be *abandoned* if any of the following are true:

a) The revoke destruction action would result in *A1* no longer having in its applicable privileges USAGE privilege on any user-defined type *UDT* such that a data type descriptor included in *DT* describes a type that is usage-dependent on *UDT*.

b) The revoke destruction action would result in *A1* no longer having in its applicable privileges the UNDER privilege on any user-defined type that is a direct supertype of *DT*.

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

0.26) For every collation descriptor *CN* contained in *S1*, *CN* is said to be *impacted* if the revoke destruction action would result in *A1* no longer having in its applicable privileges USAGE privilege on the collation whose name is contained in the <existing collation name> of *CN*.

0.27) For every character set descriptor *CSD* contained in *S1*, *CSD* is said to be *impacted* if the revoke destruction action would result in *A1* no longer having in its applicable privileges USAGE privilege on the collation whose name is contained in *CSD*.

0.28) For every descriptor included in *S1* that includes a data type descriptor *DTD*, *DTD* is said to be *impacted* if the revoke destruction action would result in *A1* no longer having in its applicable privileges USAGE privilege on the collation whose name is included in *DTD*.

0.29) Let *RD* be any routine descriptor with an SQL security characteristic of DEFINER that is included in *S1*. *RD* is said to be *abandoned* if the revoke destruction action would result in *A1* no longer having in its applicable privileges all of the following:

a) EXECUTE privilege on the SQL-invoked routine that is the subject routine of any <routine invocation>, <method invocation>, <static method invocation>, or <method reference> that is contained in the <routine body> of *RD*.

b) SELECT privilege on at least one column of each table identified by a <table reference> contained in a <query expression> simply contained in a <cursor specification>, an <insert statement>, or a <merge statement> contained in the SQL routine body of *RD*.

c) SELECT privilege on at least one column of each table identified by a <table reference> contained in a <table expression> or <select list> immediately contained in a <select statement: single row> contained in the SQL routine body of *RD*.

d) SELECT privilege on at least one column of each table identified by a <table reference> contained in a <search condition> contained in a <delete statement: searched>, an <update statement: searched>, or a <merge statement> contained in the SQL routine body of *RD*.

e) SELECT privilege on at least one column of each table identified by a <table reference> contained in a <value expression> simply contained in an <update source> or an <assigned row> contained in the SQL routine body of *RD*.

f) SELECT privilege on at least one column identified by a <column reference> contained in a <search condition> contained in a <delete statement: searched>, an <update statement: searched>, or a <merge statement> contained in the <SQL routine body> of *RD*.

g) SELECT privilege on at least one column identified by a <column reference> contained in a <value expression> simply contained in an <update source> or an <assigned row> contained in the SQL routine body of *RD*.

h) INSERT privilege on each column

Case:

i) Identified by a <column name> contained in the <insert column list> of an <insert statement> or a <merge statement> contained in the SQL routine body of *RD*.

ii) Of the table identified by the <table name> immediately contained in an <insert statement> that does not contain an <insert column list> and that is contained in the SQL routine body of *RD*.

iii) Of the table identified by the <target table> immediately contained in a <merge statement> that contains a <merge insert specification> and that does not contain an <insert column list> and that is contained in the SQL routine body of *RD*.

i) UPDATE privilege on each column whose name is contained in an <object column> contained in either an <update statement: positioned>, an <update statement: searched>, or a <merge statement> contained in the SQL routine body of *RD*.

j) DELETE privilege on each table whose name is contained in a <table name> contained in either a <delete statement: positioned> or a <delete statement: searched> contained in the SQL routine body of *RD*.

k) USAGE privilege on each domain, collation, character set, transliteration, and sequence generator whose name is contained in the SQL routine body of *RD*.

l) USAGE privilege on each user-defined type *UDT* such that a declared type of any SQL parameter, returns data type, or result cast included in *RD* is usage-dependent on *UDT*.

m) USAGE privilege on each user-defined type *UDT* such that some <data type> contained in the SQL routine body of *RD* is usage-dependent on *UDT*.

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

o) SELECT privilege on any column identified by a <column reference> contained in a <scalar subquery> that is equivalent to a <dereference operation> contained in any of the following:

i) A <query expression> simply contained in a <cursor specification>, an <insert statement>, or a <merge statement> contained in the <SQL routine body> of *RD*.

ii) A <table expression> or <select list> immediately contained in a <select statement: single row> contained in the <SQL routine body> of *RD*.

iii) A <search condition> contained in a <delete statement: searched>, an <update statement: searched>, or a <merge statement> contained in the <SQL routine body> of *RD*.

iv) A <value expression> contained in an <update source> or an <assigned row> contained in the <SQL routine body> of *RD*.

p) SELECT privilege WITH HIERARCHY OPTION on at least one supertable of the scoped table of any <reference resolution> that is contained in any of the following:

i) A <query expression> simply contained in a <cursor specification>, an <insert statement>, or a <merge statement> contained in the SQL routine body of *RD*.

ii) A <table expression> or <select list> immediately contained in a <select statement: single row> contained in the SQL routine body of *RD*.

iii) A <search condition> contained in a <delete statement: searched>, an <update statement: searched>, or a <merge statement> contained in the SQL routine body of *RD*.

iv) A <value expression> simply contained in an <update source> or an <assigned row> contained in the SQL routine body of *RD*.

q) SELECT privilege on the scoped table of any <reference resolution> that is contained in any of the following:

i) A <query expression> simply contained in a <cursor specification>, an <insert statement>, or a <merge statement> contained in the <SQL routine body> of *RD*.

ii) A <table expression> or <select list> immediately contained in a <select statement: single row> contained in the <SQL routine body> of *RD*.

iii) A <search condition> contained in a <delete statement: searched>, an <update statement: searched>, or a <merge statement> contained in the <SQL routine body> of *RD*.

iv) A <value expression> contained in an <update source> or an <assigned row> contained in the <SQL routine body> of *RD*.

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

0.30) For every table descriptor *TD* included in *S1*, for every column descriptor *CD* included in *TD*, *CD* is said to be *contaminated* if *CD* includes one of the following:

a) A user-defined type descriptor that describes a supertype of a user-defined type described by an abandoned user-defined type descriptor.

b) A reference type descriptor that includes a user-defined type descriptor that describes a supertype of a user-defined type described by an abandoned user-defined type descriptor.

c) A collection type descriptor that includes a user-defined type descriptor that describes a supertype of a user-defined type described by an abandoned user-defined type descriptor.

d) A collection type descriptor that includes a reference type descriptor that includes a user-defined type descriptor that describes a supertype of a user-defined type described by an abandoned user-defined type descriptor.

0.31) If RESTRICT is specified, and there exists an abandoned privilege descriptor, abandoned view, abandoned table constraint, abandoned assertion, abandoned domain constraint, lost domain, lost column, lost schema, or a descriptor that includes an impacted data type descriptor, impacted collation,

impacted character set, abandoned user-defined type, forsaken column descriptor, forsaken domain descriptor, or abandoned routine descriptor, then an exception condition is raised: *dependent privilege descriptors still exist*.

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

12.8 Grantor determination

1. *Rationale: Inappropriate reference to run-time values in Syntax Rules.*

Insert the following Subclause:

12.8 Grantor determination

Function

Determine the grantor of a privilege or role authorization, or the intended owner of a role.

Syntax Rules

None.

Access Rules

None.

General Rules

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

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

Case:

a) If *G* is *OMITTED*, then

Case:

i) If there is a current user identifier, then *A* is the current user identifier.

ii) Otherwise, *A* is the current role name.

b) If *G* is *CURRENT_USER*, then

Case:

i) If there is no current user identifier, then an exception condition is raised: *invalid grantor*.

ii) Otherwise, *A* is the current user identifier.

c) If *G* is *CURRENT_ROLE*, then

Case:

i) If there is no current role name, then an exception condition is raised: *invalid grantor*.

ii) Otherwise, *A* is the current role name.

Conformance Rules

None.

13 SQL-client modules

13.4 Calls to an <externally-invoked procedure>

1. *Rationale: Change of terminology.*

In Syntax Rule 2) e), enhance the definition of the SQLSTATES_CODES package with:

2) ...

e)

```
NO_DATA_NO_ADDITIONAL_RESULT_SETS_RETURNED:
  constant SQLSTATE_TYPE :="02001";
WARNING_RESULT_SETS_RETURNED:
  constant SQLSTATE_TYPE := "0100C";
```

13.5 <SQL procedure statement>

1. *Rationale: Clarification.*

Insert the following note after General Rule 1):

NOTE 350.1 — *S* is not necessarily an <SQL procedure statement>.

2. *Rationale: Clarify when constraints are checked, trigger execution contexts are created and AFTER triggers are processed.*

   Insert the following General Rules:

   2.1) The current trigger execution context *CTEC*, if any, is preserved and new trigger execution context *NTEC* is created with an empty set of state changes *SSC*.

3. *Rationale: Rule associating statement with transaction is redundant.*

   Delete General Rule 4) a) iii) 3).

4. *Rationale: Detailing the properties of a new transaction is not needed.*

   Replace General Rule 4) a) iii) 4) with:

   4) ...

      a) ...

         iii) ...

            4) If no SQL-transaction is active for the SQL-agent and *S* is an SQL-statement that implicitly initiates SQL-transactions, then an SQL-transaction is initiated. The SQL-client module that contains S is associated with the SQL-transaction.

5. *Rationale: Rule associating statement with transaction is redundant.*

   Delete General Rule Rule 4) b) i).

6. *Rationale: Detailing the properties of a new transaction is not needed.*

   Replace General Rule 4) b) ii) with:

   4) ...

      b) ...

         ii) If no SQL-transaction is active for the SQL-agent and *S* is an SQL-statement that implicitly initiates SQL-transactions, then an SQL-transaction is initiated.

7. *Rationale: Clarify when constraints are checked, trigger execution contexts are created and AFTER triggers are processed.*

   Insert the following General Rules:

   4.1) For every referential constraint descriptor *RCD* whose mode is immediate, the General Rules of Subclause 11.8, "<referential constraint definition>", are applied.

   4.2) For every state change *DSC* in *SSC* whose trigger event is DELETE, let *DSOT* be the set of transitions in *DSC* and let *DBT* be the subject table of *DSC*. Each row that is marked for deletion from *DBT* is deleted from *DBT*.

4.3) Case:

a) If, for some constraint *C*, the constraint mode of *C* in the current SQL-session is immediate and a control <search condition> included in *CD* evaluates to *False*, then an exception is raised: *integrity constraint violation*.

NOTE 350.1 — This rule implies a re-evaluation of the control <search condition>s of referential constraints. This is needed to cater for

a) referential constraints specifying NO ACTION and

b) referential constraints that are not satisfied after application of SET DEFAULT.

b) Otherwise, the Syntax Rules and General Rules of Subclause 14.26, "Execution of AFTER triggers", are applied with *SSC* as the *SET OF STATE CHANGES*.

4.4) *NTEC*, together with all of its contents, is destroyed and *CTEC*, if present, is restored to become the current trigger execution context.

8. *Rationale: Distinguish properly between atomic statements and non-atomic ones.*

Replace GR 5) a) ii) with:

5) ...

a) ...

ii) If *S* is an atomic SQL-statement, then all changes made to SQL-data or schemas by the execution of *S* are canceled.

NOTE 350.2 — Atomic and non-atomic SQL-statements are defined in Subclause 4.33.5, "SQL-statement atomicity and statement execution contexts".

9. *Rationale: Distinguish properly between atomic statements and non-atomic ones.*

Replace GR 5) b) ii) 1) with:

5) ...

b) ...

ii) ...

1) If *S* is an atomic SQL-statement, then all changes made to SQL-data or schemas by the execution of *S* are canceled.

NOTE 350.3 — Atomic and non-atomic SQL-statements are defined in Subclause 4.33.5, "SQL-statement atomicity and statement execution contexts".

14 Data manipulation

14.1 <declare cursor>

1. *Rationale: An updatable cursor must have only one leaf underlying table that is one-to-one.*

   Replace Syntax Rule 11) with:

   11) If an <updatability clause> of FOR UPDATE with or without a <column name list> is specified, then INSENSITIVE shall not be specified, *QE* shall be updatable, and *QE* shall have only one leaf underlying table *LUT* such that *QE* is one-to-one with respect to *LUT*.

2. *Rationale: An updatable cursor must have only one leaf underlying table that is one-to-one.*

   Replace Syntax Rule 13) with:

   13) If *CS* is updatable, then let *LUTN* be a <table name> that references *LUT*. *LUTN* is an exposed <table or query name> whose scope is <updatability clause>.

3. *Rationale: Clarify the specifications concerning dynamic result sets.*

   Replace Syntax Rule 16) with:

   16) If WITH RETURN is specified, then the cursor specified by the <cursor specification> is a with-return cursor.

   NOTE 354 — "with-return cursor" is defined in Subclause 3.1.6, "Definitions provided in Part 2".

4. *Rationale: Clarify the reference to sort table.*

   Replace Syntax Rule 18) e) with:

   18) ...

   e) *ST* is said to be the sort table of *CS*.

5. *Rationale: Cursor ordering is specified in Subclause 10.10, "<sort specification list>".*

   Replace General Rule 2) with:

   2) If an <order by clause> is specified, then the ordering of rows of the result is determined by the the General Rules of Subclause 10.10, "<sort specification list>". The result table specified by the <cursor specification> is the sort table of *CS* with all extended sort key columns (if any) removed.

6. *Rationale: SQL-92 compatability.*

   Insert the following Conformance Rule:

9) Without Feature T111, “Updatable joins, unions and columns”, in conforming SQL language, if FOR UPDATE is specified, then *QE* shall be simply updatable.

14.2 <open statement>

1. *Rationale: Factor out what is common to all cursors.*

   Replace all the General Rules with:

   1) The General Rules of Subclause 14.28, “Effect of opening a cursor”, are applied with *CR* as *CURSOR*.

14.3 <fetch statement>

1. *Rationale: Factor out what is common to all cursors.*

   Replace General Rules 2) through 5) with:

   2) The General Rules of Subclause 14.29, “Determination of the current row of a cursor”, are applied with *CR* as *CURSOR* and <fetch orientation> as *FETCH ORIENTATION*.

14.4 <close statement>

1. *Rationale: Clarification of the method of identification of dynamic objects.*

   Insert the following Syntax Rule:

   1) Let *CR* be the cursor identified by <cursor name>.

2. *Rationale: Clarification of the method of identification of dynamic objects.*

   Delete General Rule 1).

3. *Rationale: Factor out what is common to all cursors.*

   Replace General Rules 2) through 5) with:

   2) The General Rules of Subclause 14.30, “Effect of closing a cursor”, are applied with *CR* as *CURSOR*.

14.6 <delete statement: positioned>

1. *Rationale: An updatable cursor has only one leaf underlying table that is one-to-one.*

   Replace Syntax Rule 1) with:

   1) Let *CR* be the cursor denoted by the <cursor name>. *CR* shall be an updatable cursor.

2. *Rationale: An updatable cursor has only one leaf underlying table that is one-to-one.*

   Replace Syntax Rule 4) with:

   4) Let *T* be the simply underlying table of *CR*. *T* is the *subject table* of the <delete statement: positioned>. Let *LUT* be the leaf underlying table of *T* such that *T* is one-to-one with respect to *LUT*.

3. *Rationale: Move the rule that deletes rows from base tables to the correct place.*

   Replace General Rule 9) a) with:

   9) ...

      a) If *LUT* is a base table, then:

         i) Case:

            1) If <target table> specifies ONLY, then *LUT* is *identified for deletion processing without subtables*.

            2) Otherwise, *LUT* is *identified for deletion processing with subtables*.

               NOTE 364 — Identifying a base table for deletion processing, with or without subtables, is an implementation-dependent mechanism.

         ii) The General Rules of Subclause 14.16, "Effect of deleting rows from base tables", are applied.

4. *Rationale: Move the rule that deletes rows from base tables to the correct place.*

   Delete General Rule 10).

14.7 <delete statement: searched>

1. *Rationale: Move the rule that deletes rows from base tables to the correct place.*

   Replace General Rule 7) a) with:

   7) ...

      a) If *T* is a base table, then:

i) Case:

1) If <target table> specifies ONLY, then *T* is *identified for deletion processing without subtables*.

2) Otherwise, *T* is *identified for deletion processing with subtables*.

NOTE 367 — Identifying a base table for deletion processing, with or without subtables, is an implementation-dependent mechanism.

ii) The General Rules of Subclause 14.16, "Effect of deleting rows from base tables", are applied.

2. *Rationale: Move the rule that deletes rows from base tables to the correct place.*

Delete General Rule 8).

3. *Rationale: Clarify when deletions are effected.*

Delete General Rule 11).

4. *Rationale: Delete spurious rule.*

Delete General Rule 12).

5. *Rationale: Correct the definition of when no deletions occur.*

Replace General Rule 13) with:

13) If no rows are marked for deletion, then a completion condition is raised: *no data*.

6. *Rationale: SQL-92 compatability.*

Insert the following Conformance Rule:

2) Without Feature T111, "Updatable joins, unions and columns", conforming SQL language shall not contain a <delete statement: searched> that contains a <target table> that identifies a table that is not simply updatable.

14.8 <insert statement>

1. *Rationale: Add the description that identity column and self-referencing column cease to be marked as unassigned.*

7) ...

d) For every C_i for which one of the following conditions is true:

i) C_i is not marked as unassigned and no underlying column of C_i is a self-referencing column.

ii) Some underlying column of C_i is a user-generated self-referencing column.

iii) Some underlying column of C_i is a self-referencing column and OVERRIDING SYSTEM VALUE is specified.

iv) Some underlying column of C_i is an identity column and OVERRIDING SYSTEM VALUE is specified.

the General Rules of Subclause 9.2, "Store assignment", are applied with C_i and SV_i as *TARGET* and *SOURCE*, respectively. C_i is no longer marked as unassigned.

2. *Rationale: Move the rule that inserts into base tables to the correct place.*

Replace General Rule 8) a) with:

8) ...

a) If *T* is a base table, then:

i) *T* is *identified for insertion of source table S.*

NOTE 374 — Identifying a base table for insertion of a source table is an implementation-dependent operation.

ii) The General Rules of Subclause 14.19, "Effect of inserting tables into base tables", are applied.

3. *Rationale: Move the rule that inserts into base tables to the correct place.*

Delete General Rule 9).

4. *Rationale: SQL-92 compatability.*

Insert the following Conformance Rule:

5) Without Feature T111, "Updatable joins, unions and columns", conforming SQL language shall not contain an <insert statement> that contains an <insertion target> that identifies a table that is not simply updatable.

14.9 <merge statement>

1. *Rationale: Move the rules that update base tables to their correct places.*

Replace General Rule 6) a) vi) 1) with:

6) ...

a) ...

vi) ...

1) If *T* is a base table, then:

A) Case:

I) If <target table> specifies ONLY, then *T* is *identified for replacement processing without subtables* with respect to object columns *CL*.

II) Otherwise, *T* is *identified for replacement processing with subtables* with respect to object columns *CL*.

NOTE 380 — Identifying a base table for replacement processing, with or without subtables, is an implementation-dependent mechanism. In general, though not here, the list of object columns can be empty.

B) The General Rules of Subclause 14.22, "Effect of replacing rows in base tables", are applied.

2. *Rationale: Move the rules that update base tables to their correct places.*

Delete General Rule 6) a) vii).

3. *Rationale: Add the description that identity column and self-referencing column cease to be marked as unassigned.*

6) ...

b) iv) ...

4) For every C_i for which one of the following conditions is true:

A) C_i is not marked as unassigned and no underlying column of C_i is a self-referencing column.

B) Some underlying column of C_i is a user-generated self-referencing column.

C) Some underlying column of C_i is a self-referencing column and OVERRIDING SYSTEM VALUE is specified.

D) Some underlying column of C_i is an identity column and OVERRIDING SYSTEM VALUE is specified.

the General Rules of Subclause 9.2, "Store assignment", are applied with C_i and SV_i as *TARGET* and *SOURCE*, respectively. C_i is no longer marked as unassigned.

4. *Rationale: Move the rules that update base tables to their correct places.*

Replace General Rule 6) b) vii) 1) with:

6) ...

b) ...

vii) ...

1) If *T* is a base table, then:

A) *T* is *identified for insertion of source table S*.

NOTE 382 — Identifying a base table for insertion of a source table is an implementation-dependent operation.

B) The General Rules of Subclause 14.19, "Effect of inserting tables into base tables", are applied.

5. *Rationale: Move the rules that update base tables to their correct places.*

Delete General Rule 6) b) viii).

6. *Rationale: SQL-92 compatability.*

Insert the following Conformance Rule:

5) Without Feature T111, "Updatable joins, unions and columns", conforming SQL language shall not contain a <merge statement> that contains a <target table> that identifies a table that is not simply updatable.

14.10 <update statement: positioned>

1. *Rationale: Correct the definition of the correspondence between a row of the cursor and a row in a base table.*

Replace General Rule 14) with:

14) Let *R1* be the candidate new row and let *R* be the current row of *CR*. Exactly one row *TR* in *T*, such that the value of each field in *R* that is derived from one or more fields in *TR* is identical to the corresponding value derived from the same one or more fields in *TR*, is identified for replacement in *T*. The current row *R* of *CR* is replaced by *R1*. Let *TR1* be a row consisting of the fields of *R1* and the fields of *TR* that have no corresponding fields in *R1*, ordered according to the order of their corresponding columns in *T*. *TR1* is the replacement row for *TR* and
`{ ( TR, TR1 ) }`
is the replacement set for *T*.

2. *Rationale: Move the rule that updates base tables to the correct place.*

Replace General Rule 15) a) with:

15) ...

a) If LUT is a base table, then:

i) Case:

1) If <target table> specifies ONLY, then *LUT* is *identified for replacement processing without subtables* with respect to object columns *CL*.

2) Otherwise, *LUT* is *identified for replacement processing with subtables* with respect to object columns *CL*.

NOTE 388 — Identifying a base table for replacement processing, with or without subtables, is an implementation-dependent mechanism. In general, though not here, the list of object columns can be empty.

ii) The General Rules of Subclause 14.22, "Effect of replacing rows in base tables", are applied.

3. *Rationale: Move the rule that updates base tables to the correct place.*

Delete General Rule 16).

4. *Rationale: An updatable cursor has only one leaf underlying table that is one-to-one.*

Replace Syntax Rules 1) and 2) with:

1) Let *CR* be the cursor denoted by the <cursor name>. *CR* shall be an updatable cursor.

2) Let *TU* be the simply underlying table of *CR*. *TU* is the *subject table* of the <update statement: positioned>. Let *LUT* be the leaf underlying table of *T* such that *T* is one-to-one with respect to *LUT*.

14.11 <update statement: searched>

1. *Rationale: Move the rule that updates base tables to the correct place.*

Replace General Rule 13) a) with:

13) ...

a) If *T* is a base table, then:

i) Case:

1) If <target table> specifies ONLY, then *T* is *identified for replacement processing without subtables* with respect to object columns *CL*.

2) Otherwise, *T* is *identified for replacement processing with subtables* with respect to object columns *CL*.

NOTE 393 — Identifying a base table for replacement processing, with or without subtables, is an implementation-dependent mechanism. In general, though not here, the list of object columns can be empty.

ii) The General Rules of Subclause 14.22, "Effect of replacing rows in base tables", are applied.

2. *Rationale: Move the rule that updates base tables to the correct place.*

Delete General Rule 14).

3. *Rationale: SQL-92 compatability.*

Insert the following Conformance Rule:

2) Without Feature T111, "Updatable joins, unions and columns", conforming SQL language shall not contain an <update statement: searched> that contains a <target table> that identifies a table that is not simply updatable.

14.16 Effect of deleting rows from base tables

1. *Rationale: Clarify when constraints are checked, trigger execution contexts are created and AFTER triggers are processed.*

Replace General Rules 3) and 4) with:

3) Let *SSC* be the set of state changes in the current trigger execution context.

4) For every table *T* in *TT*, for every table *ST* that is a supertable of *T* or, unless *T* is identified for deletion processing with subtables, a subtable of *T*,

Case:

a) If a state change *SC* exists in *SSC* with subject table *ST* and trigger event DELETE, then one copy each of every row of *ST* that is identified for deletion in *ST* is added to the set of transitions of *SC*.

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

i) The set of transitions of *SC* consists of one copy each of every row of *ST* that is identified for deletion in *ST*.

ii) The trigger event of *SC* is DELETE.

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

iv) The column list of *SC* is empty.

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

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

2. *Rationale: Clarify when constraints are checked, trigger execution contexts are created and AFTER triggers are processed.*

   Delete General Rules 7), 8), 9), and 10).

14.17 Effect of deleting some rows from a derived table

1. *Rationale: Move the rule specifying deletions from base tables to the correct place.*

   Insert the following General Rule:

   2) ...

      c) ...

         iv.1) The General Rules of Subclause 14.16, "Effect of deleting rows from base tables", are applied.

14.19 Effect of inserting tables into base tables

1. *Rationale: Clarify when constraints are checked, trigger execution contexts are created and AFTER triggers are processed.*

   Replace General Rule 1) with:

   1) ...

      a) Let *SSC* be the set of state changes in the current trigger execution context.

2. *Rationale: Clarify when constraints are checked, trigger execution contexts are created and AFTER triggers are processed.*

   Replace General Rule 3) with:

   3) For every base table *BT* that is identified for insertion,

      Case:

      a) If a state change *SC* exists in *SSC* with subject table *ST* and trigger event INSERT, then the rows in the source table for *BT* are added to the set of transitions of *SC*.

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

         i) The set of transitions of *SC* consists of the rows in the source table for *BT*.

         ii) The trigger event of *SC* is INSERT.

         iii) The subject table of *SC* is *BT*.

iv) The column list of *SC* is empty.

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

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

3. *Rationale: Clarify when constraints are checked, trigger execution contexts are created and AFTER triggers are processed.*

   Delete NOTE 402.

4. *Rationale: Clarify when constraints are checked, trigger execution contexts are created and AFTER triggers are processed.*

   Delete GR5) c).

5. *Rationale: Clarify when constraints are checked, trigger execution contexts are created and AFTER triggers are processed.*

   Delete GR6) and NOTE 403.

6. *Rationale: Clarify when constraints are checked, trigger execution contexts are created and AFTER triggers are processed.*

   Delete GR7).

14.20 Effect of inserting a table into a derived table.

1. *Rationale: Move the rule specifying insertions to base tables to the correct place.*

   Insert the following General Rule:

   2) ...

      b) ...

         iv.1) The General Rules of Subclause 14.19, "Effect of inserting tables into base tables", are applied.

14.21 Effect of inserting a table into a viewed table

1. *Rationale: Correction to processing of WITH CHECK OPTION.*

   Replace General Rules 2) and 3) with:

2) Case:

a) If *TD* indicates WITH CHECK OPTION, then:

i) Case:

1) If *TD* specifies LOCAL, then let *VD* be a view descriptor derived from *TD* by removing the WITH CHECK OPTION indication.

2) Otherwise, let *VD* be a view descriptor derived from *TD* as follows:

A) The WITH CHECK OPTION indication is removed.

B) Every reference contained in *QE* to an underlying table *UV* of *QE* that is a viewed table is replaced by a reference to a view whose descriptor is identical to that of *UV* except that WITH CASCADED CHECK OPTION is indicated.

ii) The General Rules of this Subclause are applied with *S* as *SOURCE* and the view described by *VD* as *TARGET*.

iii) If the result of

```
EXISTS ( SELECT * FROM S
         EXCEPT ALL
         SELECT * FROM T )
```

is *True*, then an exception condition is raised: *with check option violation*.

b) Otherwise, the General Rules of Subclause 14.20, "Effect of inserting a table into a derived table", are applied, with *S* as *SOURCE* and *QE* as *TARGET*.

14.22 Effect of replacing rows in base tables

1. *Rationale: Clarify when constraints are checked, trigger execution contexts are created and AFTER triggers are processed.*

Replace General Rules 4) and 5) with:

4) Let *SSC* be the set of state changes in the current trigger execution context.

5) For every table *ST* that is identified for replacement processing, let *TL* be the set consisting of the names of the columns of *ST*. For every subset *STL* of *TL* such that either *STL* is empty or the intersection of *STL* and *OC* is not empty:

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

i) Let *NV* be the result of applying the General Rules of Subclause 9.21, "Generation of the next value of a sequence generator", with the sequence descriptor included in the column descriptor of *IC* as *SEQUENCE*.

Case:

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

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

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

b) All sites in *ST* that are marked as unassigned cease to be so marked.

c) Case:

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

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

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

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

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

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

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

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

2. *Rationale: Clarify when constraints are checked, trigger execution contexts are created and AFTER triggers are processed.*

Delete General Rule 9).

3. *Rationale: Clarify when constraints are checked, trigger execution contexts are created and AFTER triggers are processed.*

Delete General Rule 10) and NOTE 404.

4. *Rationale: Clarify when constraints are checked, trigger execution contexts are created and AFTER triggers are processed.*

Delete General Rule 11).

14.23 Effect of replacing some rows in a derived table

1. *Rationale: Move the rule that updates base table to the correct place.*

   Insert the following General Rule:

   2) ...

      c) ...

         iv.1) The General Rules of Subclause 14.22, "Effect of replacing rows in base tables", are applied.

14.24 Effect of replacing some rows in a viewed table

1. *Rationale: Correction to processing of WITH CHECK OPTION.*

   Replace General Rules 2) and 3) with:

   2) Case:

      a) If *TD* indicates WITH CHECK OPTION, then:

         i) Case:

            1) If *TD* specifies LOCAL, then let *VD* be a view descriptor derived from *TD* by removing the WITH CHECK OPTION indication.

            2) Otherwise, let *VD* be a view descriptor derived from *TD* as follows:

               A) The WITH CHECK OPTION indication is removed.

               B) Every reference contained in *QE* to an underlying table *UV* of *QE* that is a viewed table is replaced by a reference to a view whose descriptor is identical to that of *UV* except that WITH CASCADED CHECK OPTION is indicated.

         ii) The General Rules of this Subclause are applied with the view *V* described by *VD* as *VIEW NAME* and *RS* as the *replacement set for V*>.

         iii) Let *S* be the table consisting of the candidate new rows of *RS*. If the result of

```
EXISTS ( SELECT * FROM S
         EXCEPT ALL
         SELECT * FROM T )
```

         is *True*, then an exception condition is raised: *with check option violation*.

      b) Otherwise, the General Rules of Subclause 14.23, "Effect of replacing some rows in a derived table", are applied with *QE* as *TABLE* and *RS* as the *replacement set for QE*.

14.27 Execution of triggers

1. *Rationale: Correct variable symbol references.*

   Replace General Rule 3) a) with:

   3) ...

      a) If *TR* is a row-level trigger, then, for each transition *T* in *ST* for which *TR* is not considered as executed, *TA* is invoked and *TR* is considered as executed for *T*. The order in which the transitions in *ST* are taken is implementation-dependent.

14.28 Effect of opening a cursor

1. *Rationale: Factor out what is common to all cursors. Deletion of redundant rule and clarification of replacement of BNF term by a "value".*

   Insert a new Subclause as follows:

14.28 Effect of opening a cursor.

Function

Specify the effect of opening a cursor

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let *CR* be *CURSOR* specified in an application of this Subclause. If *CR* is not in the closed state, then an exception condition is raised: *invalid cursor state*.

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

3) *CR* is opened in the following steps:

   a) A copy *CS* of *S* is effectively created in which:

i) Each <embedded variable specification>, <host parameter specification>, <SQL parameter reference>, and <dynamic parameter specification> is replaced by a <literal> denoting the value resulting from evaluating the <embedded variable specification>, <host parameter specification>, <SQL parameter reference>, and <dynamic parameter specification>, respectively, with all such evaluations effectively done at the same instance in time.

ii) Each <value specification> generally contained in *S* that is CURRENT_USER, CURRENT_ROLE, SESSION_USER, SYSTEM_USER, CURRENT_PATH, CURRENT_DEFAULT_TRANSFORM_GROUP, or CURRENT_TRANSFORM_GROUP_FOR_TYPE <path-resolved user-defined type name> is replaced by a <literal> denoting the value of CURRENT_USER, CURRENT_ROLE, SESSION_USER, SYSTEM_USER, CURRENT_PATH, CURRENT_DEFAULT_TRANSFORM_GROUP, or CURRENT_TRANSFORM_GROUP_FOR_TYPE <path-resolved user-defined type name>, respectively, with all such evaluations effectively done at the same instant in time.

iii) Each <datetime value function> generally contained in *S* is replaced by a <literal> denoting the value resulting from evaluation of that <datetime value function>, with all such evaluations effectively done at the same instant in time.

NOTE 410.1 — Multiple appearances in *S* of equivalent <embedded variable specification>s, <host parameter specification>s, or <SQL parameter reference>s are also effectively evaluated at the same instant in time.

b) Let *T* be the table specified by *CS*. *T* is the result set of *CR*.

c) A table descriptor for *T* is effectively created.

d) The General Rules of Subclause 14.1, "<declare cursor>", are applied.

e) Case:

i) If *S* specifies INSENSITIVE, then a copy of *T* is effectively created and *CR* is placed in the open state and its position is before the first row of the copy of *T*.

ii) Otherwise, *CR* is placed in the open state and its position is before the first row of *T*.

4) If *CR* specifies INSENSITIVE, and the SQL-implementation is unable to guarantee that significant changes will be invisible through *CR* during the SQL-transaction in which *CR* is opened and every subsequent SQL-transaction during which it may be held open, then an exception condition is raised: *cursor sensitivity exception — request rejected*.

5) If *CR* specifies SENSITIVE, and the SQL-implementation is unable to guarantee that significant changes will be visible through *CR* during the SQL-transaction in which *CR* is opened, then an exception condition is raised: *cursor sensitivity exception — request rejected*.

NOTE 410.2 — The visibility of significant changes through a sensitive holdable cursor during a subsequent SQL-transaction is implementation-defined.

6) Whether an SQL-implementation is able to disallow significant changes that would not be visible through a currently open cursor is implementation-defined.

7) If *CR* is a with-return cursor, then let *SIP* be the active SQL-invoked procedure, let *INV* be the invoker of *SIP*, and let *RSS* be the result set sequence for *SIP* and *INV* in the active SQL-session context. *T* is added to the end of *RSS*.

Conformance Rules

None.

14.29 Determination of the current row of a cursor

1. *Rationale: Factor out what is common to all cursors.*

   Insert a new Subclause as follows:

14.29 Determination of the current row of a cursor

Function

Specify how the current row of a cursor is determined.

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let *CR* be *CURSOR* specified in an application of this Subclause and let *FO* be *FETCH ORIENTATION*. If *CR* is not in the open state, then an exception condition is raised: *invalid cursor state*.

2) Case:

   a) If *FO* contains a <simple value specification>, then let *J* be the value of that <simple value specification>.

   b) If *FO* specifies NEXT or FIRST, then let *J* be +1.

   c) If *FO* specifies PRIOR or LAST, then let *J* be -1.

3) Let T_t be a table of the same degree as *T*.

   Case:

   a) If *FO* specifies ABSOLUTE, FIRST, or LAST, then let T_t contain all rows of *T*, preserving their order in *T*.

b) If *FO* specifies NEXT or specifies RELATIVE with a positive value of *J*, then:

i) If the table *T* identified by *CR* is empty or if the position of *CR* is on or after the last row of *T*, then let T_t be a table of no rows.

ii) If the position of *CR* is on a row *R* that is other than the last row of *T*, then let T_t contain all rows of *T* ordered after row *R*, preserving their order in *T*.

iii) If the position of *CR* is before a row *R*, then let T_t contain row *R* and all rows of *T* ordered after row *R*, preserving their order in *T*.

c) If *FO* specifies PRIOR or specifies RELATIVE with a negative value of *J*, then:

i) If the table *T* identified by *CR* is empty or if the position of *CR* is on or before the first row of *T*, then let T_t be a table of no rows.

ii) If the position of *CR* is on a row *R* that is other than the first row of *T*, then let T_t contain all rows of *T* ordered before row *R*, preserving their order in *T*.

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

iv) If the position of *CR* is after the last row of *T*, then let T_t contain all rows of *T*, preserving their order in *T*.

d) If RELATIVE is specified with a zero value of *J*, then:

i) If the position of *CR* is on a row of *T*, then let T_t be a table comprising that one row.

ii) Otherwise, let T_t be an empty table.

4) Let *N* be the number of rows in T_t. If *J* is positive, then let *K* be *J*. If *J* is negative, then let *K* be *N*+*J*+1. If *J* is zero and ABSOLUTE is specified, then let *K* be zero; if *J* is zero and RELATIVE is specified, then let *K* be 1.

5) Case:

a) If *K* is greater than 0 (zero) and not greater than *N*, then *CR* is positioned on the *K*-th row of T_t and the corresponding row of *T*. That row becomes the current row of *CR*.

b) Otherwise, a completion condition is raised: *no data*.

Case:

i) If *FO* specifies RELATIVE with *J* equal to zero, then the position of *CR* is unchanged.

ii) If *FO* implicitly or explicitly specifies NEXT, specifies ABSOLUTE or RELATIVE with *K* greater than *N*, or specifies LAST, then *CR* is positioned after the last row.

iii) Otherwise, *FO* specifies PRIOR, FIRST, or ABSOLUTE or RELATIVE with *K* not greater than *N* and *CR* is positioned before the first row.

Conformance Rules

None.

14.30 Effect of closing a cursor

1. *Rationale: Factor out what is common to all cursors.*

   Insert a new Subclause as follows:

14.30 Effect of closing a cursor

Function

Effect of closing a cursor.

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let *CR* be *CURSOR* specified in an application of this Subclause.

2) If *CR* is not in the open state, then an exception condition is raised: *invalid cursor state*.

3) Let *RS* be the result set of *CR*.

4) *CR* is placed in the closed state and the copy of the <cursor specification> of the <declare cursor> that specified CR is destroyed.

5) Case:

   a) If *RS* is a returned result set in some existing result set sequence *RSS*, then:

      i) Let *RTN* be the number of result sets in *RSS*.

      ii) Let *RSN* be the ordinal position of *RS* in *RSS*.

      iii) If *RS* is not the last result set in *RSS*, then let *NRS* be the result set immediately following *RS* in *RSS*.

      iv) Case:

         1) If *CR* is a with-return cursor, then *RS* is deleted from *RSS*.

NOTE 410.3 — If *CR* is a with-return cursor, then it must be being closed by the SQL-invoked procedure that generates it, not by the invoker of that procedure. Its result set will not now become a returned one unless the cursor is opened again.

2) If *RSN* = *RTN*, then a completion condition is raised: *no data — no additional result sets returned*.

3) Otherwise:

A) *CR* is placed in the open state.

B) *NRS* is the result set of *CR*.

C) The cursor position of *CR* is the initial cursor position of *NRS*.

D) *CR* is not scrollable.

E) A completion condition is raised: *warning — additional result sets returned*.

b) Otherwise, *RS* is destroyed.

Conformance Rules

None.

16 Transaction management

16.1 <start transaction statement>

1. *Rationale: Factor out common material and address various minor problems.*

Replace the Format and the Syntax and General Rules with:

Format

```
<start transaction statement> ::=
    START TRANSACTION [ <transaction characteristics> ]
```

Syntax Rules

None.

General Rules

1) If an SQL-transaction is currently active, then an exception condition is raised: *invalid transaction state — active SQL-transaction*.

2) Case:

   a) If <transaction characteristics> is omitted, then let *TC* be
   `ECAM ISOLATION LEVEL ECIL DIAGNOSTICS SIZE ECNC`
   where *ECAM*, *ECIL*, and *ECNC* are the access mode, isolation level and number of conditions, respectively, of the enduring transaction characteristics of the current SQL-session.

   b) Otherwise, let *TC* be the <transaction characteristics>.

3) If <number of conditions> is specified and is less than 1 (one), then an exception condition is raised: *invalid condition number*.

4) The <set transaction statement>
   `SET TRANSACTION TC`
   is effectively executed.

   NOTE 411 — The characteristics of a transaction begun by a <start transaction statement> are as specified here regardless of the characteristics specified by any preceding <set transaction statement>. That is, even if one or more characteristics are omitted by the <start transaction statement>, the defaults specified in the Syntax Rules of this Subclause and Subclause 16.8, "<transaction characteristics>", are effective and are not affected by any (preceding) <set transaction statement> in the same SQL-session.

5) An SQL-transaction is initiated.

2. *Rationale: Factor out common material and address various minor problems*

   Delete Conformance Rules 2) and 3)

16.2 <set transaction statement>

1. *Rationale: Factor out common material and address various minor problems.*

   Replace the Format with:

Format

```
<set transaction statement> ::=
    SET [ LOCAL ] TRANSACTION <transaction characteristics>
```

2. *Rationale: Factor out common material and address various minor problems.*

   Replace the Syntax Rules with:

   1) If LOCAL is specified, then <transaction characteristics> shall not contain <number of conditions>.

3. *Rationale: Factor out common material and address various minor problems.*

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

   3) Let *TC* be <transaction characteristics>. Let *CSC* be the current SQL-session context.

   4) If the explicit or implicit <transaction access mode> contained in *TC* contains READ ONLY, then the current access mode of *CSC* is set to *read-only*. Otherwise, the current access mode of *CSC* is set to *read-write*.

   5) The current isolation level of *CSC* is set to an implementation-defined isolation level that will not exhibit any of the phenomena that the explicit or implicit <level of isolation> contained in *TC* would not exhibit, as specified in Table 8, "SQL-transaction isolation levels and the three phenomena".

   6) The current condition area limit of *CSC* is set to the explicit or implicit <number of conditions> contained in *TC*.

16.3 <set constraints mode statement>

1. *Rationale: Misused key word.*

   Replace Syntax Rule 2) with:

   2) The constraint identified by <constraint name> shall be deferrable.

2. *Rationale: Avoid reference to possibly nonexistent object and misuse of key word.*

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

   1) Let *CSC* be the current SQL-session context.

   2) If IMMEDIATE is specified, then

      Case:

      a) If ALL is specified, then the constraint mode in *CSC* of all constraints that are deferrable is set to immediate.

      b) Otherwise, the constraint mode in *CSC* for the constraints identified by the <constraint name>s in the <constraint name list> is set to immediate.

   3) If DEFERRED is specified, then

      Case:

      a) If ALL is specified, then the constraint mode in *CSC* of all constraints that are deferrable is set to deferred.

      b) Otherwise, the constraint mode in *CSC* for the constraints identified by the <constraint name>s in the <constraint name list> is set to deferred.

16.6 <commit statement>

1. *Rationale: Inappropriate symbol used as a cursor name.*

   Replace General Rule 3) with:

   3) For every open cursor *CR* that is not a holdable cursor in any SQL-client module associated with the current SQL-transaction, the General Rules of Subclause 14.30, "Effect of closing a cursor", are applied with *CR* as *CURSOR*.

2. *Rationale: Restore the SQL-session's enduring transaction characteristics.*

   Replace General Rule 9) with:

   9) The current SQL-transaction is terminated.

   9.1) Case:

      a) If <commit statement> contains AND CHAIN, then an SQL-transaction is initiated. Any branch transactions of the SQL-transaction are initiated with the same access mode, isolation level, and diagnostics area limit as the corresponding branch of the SQL-transaction just terminated.

      b) Otherwise:

         i) The current access mode, current isolation level, and current condition area limit of the current SQL-session context are set to the access mode, isolation level, and condition area limit, respectively, of the enduring transaction characteristics of the current SQL-session.

         ii) For every constraint *C*, the constraint mode of *C* in the current SQL-session context is set to the initial constraint mode included in the constraint descriptor for *C*.

3. *Rationale: Clarify whether prepared statements may be deallocated.*

   Replace General Rule 10) with:

   10) The prepared statement of every held cursor remains in existence. It is implementation-dependent whether or not any other prepared statement is deallocated.

16.7 <rollback statement>

1. *Rationale: Inappropriate symbol used as a cursor name.*

   Replace General Rule 2) e) with:

   2) ...

e) For every open cursor *CR* that is not a holdable cursor in any SQL-client module associated with the current SQL-transaction, the General Rules of Subclause 14.30, "Effect of closing a cursor", are applied with *CR* as *CURSOR*.

2. *Rationale: Restore the SQL-session's enduring transaction characteristics.*

Replace General Rule 2) f) with:

2) ...

f) The current SQL-transaction is terminated.

f.1) Case:

i) If <rollback statement> contains AND CHAIN, then an SQL-transaction is initiated. Any branch transactions of the SQL-transaction are initiated with the same access mode, isolation level, and diagnostics area limit as the corresponding branch of the SQL-transaction just terminated.

ii) Otherwise:

1) The current access mode, current isolation level, and current condition area limit of the current SQL-session context are set to the access mode, isolation level, and condition area limit, respectively, of the enduring transaction characteristics of the current SQL-session.

2) For every constraint *C*, the constraint mode of *C* in the current SQL-session context is set to the initial constraint mode included in the constraint descriptor for *C*.

3. *Rationale: Inappropriate symbol used as a cursor name.*

Replace General Rule 3) g) with:

3) ...

g) For every open cursor *CR* that is not a holdable cursor in any SQL-client module associated with the current SQL-transaction, the General Rules of Subclause 14.30, "Effect of closing a cursor", are applied with *CR* as *CURSOR*.

16.8 <transaction characteristics>

1. *Rationale: Factor out common material and address various minor problems.*

Insert the following new Subclause:

16.8 <transaction characteristics>

Function

Specify transaction characteristics

Format

```
<transaction characteristics> ::=
    <transaction mode> [ { <comma> <transaction mode> }... ]

<transaction mode> ::=
   <isolation level>
  |<transaction access mode>
  |<diagnostics size>

<transaction access mode> ::=
   READ ONLY
  |READ WRITE

<isolation level> ::= ISOLATION LEVEL <level of isolation>

<level of isolation> ::=
   READ UNCOMMITTED
  |READ COMMITTED
  |REPEATABLE READ
  |SERIALIZABLE

<diagnostics size> ::= DIAGNOSTICS SIZE <number of conditions>

<number of conditions> ::= <simple value specification>
```

Syntax Rules

1) Let *TC* be the <transaction characteristics>.

2) *TC* shall contain at most one <isolation level>, at most one <transaction access mode>, and at most one <diagnostics size>.

3) If *TC* does not contain an <isolation level>, then ISOLATION LEVEL SERIALIZABLE is implicit.

4) If <transaction access mode> is READ WRITE, then the <level of isolation> shall not be READ UNCOMMITTED.

5) If *TC* does not contain a <transaction access mode>, then

   Case:

   a) If <isolation level> contains READ UNCOMMITTED, then READ ONLY is implicit.

   b) Otherwise, READ WRITE is implicit.

6) The declared type of <number of conditions> shall be exact numeric with scale 0 (zero).

7) If *TC* does not contain a <diagnostics size>, then DIAGNOSTICS SIZE *n* is implicit, where *n* is an implementation-dependent value not less than 1 (one).

Access Rules

None.

General Rules

None.

Conformance Rules

1) Without Feature F111, "Isolation levels other than SERIALIZABLE ", conforming SQL language shall not contain an <isolation level> that contains a <level of isolation> other than SERIALIZABLE.

2) Without Feature F121, "Basic diagnostics management", conforming SQL language shall not contain a <diagnostics size>.

17 Connection management

17.1 <connect statement>

1. *Rationale: Initialize the SQL-session's enduring and current transaction. characteristics*

   Insert the following General Rule:

   9) ...

      c) The access mode, isolation level, and condition area limit of the enduring transaction characteristics of the current SQL-session are read-write, SERIALIZABLE, and an implementation-dependent value not less than 1 (one), respectively.

2. *Rationale: Initialize the SQL-session's enduring and current transaction. characteristics*

   Replace General Rule 13) with:

   13) The current SQL-session context of the current SQL-session is initialized as follows:

      a) The authorization stack is set to a single cell containing the user identifier <connection user name>.

b) The current access mode, current isolation level, and current condition area limit are set to the access mode, isolation level, and condition area limit, respectively, of the enduring transaction characteristics of the current SQL-session.

18 Session management

18.1 <set session characteristics statement>

1. *Rationale: Avoid allowing repetition of TRANSACTION.*

   Replace the production for <session characteristic> with:

```
<session characteristic> ::=
    <session transaction characteristics>

<session transaction characteristics> ::=
    TRANSACTION <transaction mode> [ { <comma> <transaction mode>... } ]
```

2. *Rationale: Avoid allowing repetition of TRANSACTION.*

   Replace Syntax Rule 1) with:

   1) <session characteristic list> shall contain at most one <isolation level>, at most one <transaction access mode>, and at most one <diagnostics size>.

3. *Rationale: Clarify the General Rules.*

   Replace General Rule 1) with:

   1) Let *SCL* be the <session transaction list>. Let *ESC* be the enduring session characteristics of the current SQL-session.

   1.1) If *SCL* contains an <isolation level> *IL*, then the isolation level of *ESC* is set to the <level of isolation> contained in *IL*.

   1.2) If *SCL* contains an <access mode> *AM*, then the access mode of *ESC* is set to read-only or read-write according to whether *AM* contains READ ONLY or READ WRITE, respectively.

   1.3) If *SCL* contains a <diagnostics size> *DS*, then the condition area limit of *ESC* is set to the <number of conditions> contained in *DS*.

4. *Rationale: Delete redundant Conformance Rule.*

   Delete Conformance Rule 2)

18.2 <set session user identifier statement>

1. *Rationale: Clarify intent of General Rule 4).*

   Replace General Rule 4) with:

   4) If *V* is not equal to the current value of the SQL-session user identifier of the current SQL-session context, then restrictions on the permissible values *V* are implementation-defined.

18.3 <set role statement>

1. *Rationale: Set the SQL-session role name as well as the current one. Avoid evaluating too many General Rules when NONE is specified.*

   Replace the Function with:

   ### Function

   Set the SQL-session role name and the current role name for the current SQL-session context.

2. *Rationale: Set the SQL-session role name as well as the current one. Avoid evaluating too many General Rules when NONE is specified.*

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

   2) If there is no current user identifier, then an exception condition is raised: *invalid role specification*.

   3) If <role specification> contains a <value specification>, then:

      a) Let *S* be <value specification> and let *V* be the character string that is the value of `TRIM ( BOTH ' ' FROM S )`

      b) If *V* does not conform to the Format and Syntax Rules of a <role name>, then an exception condition is raised: *invalid role specification*.

      c) If no role authorization descriptor exists that indicates that the role identified by *V* has been granted to either the current user identifier or to PUBLIC, then an exception condition is raised: *invalid role specification*.

      d) The SQL-session role name and the current role name are set to *V*.

   4) If NONE is specified, then the current role name is removed.

19 Dynamic SQL

19.2 <allocate descriptor statement>

1. *Rationale: Clarification of the method of identification of dynamic objects.*

   Replace General Rules 1) and 2) with:

   1) Let *S* be the <simple value specification> that is immediately contained in <descriptor name> and let *V* be the character string that is the result of
   `TRIM ( BOTH ' ' FROM` *S* `)`

      Case:

      a) If *V* does not conform to the Format and Syntax Rules of an <identifier>, then an exception condition is raised: *invalid SQL descriptor name*.

      b) Otherwise, let *DN* be the <descriptor name>. The value of *DN* is *V*.

   2) Case:

      a) If *DN* identifies an SQL descriptor area, then an exception condition is raised: *invalid SQL descriptor name*.

      b) Otherwise, an SQL descriptor area is created that is identified by *DN*. The SQL descriptor area will have at least <occurrences> number of SQL item descriptor areas. The value of LEVEL in each of the item descriptor areas is set to 0 (zero). The value of every other field in the SQL descriptor area is implementation-dependent.

19.3 <deallocate descriptor statement>

1. *Rationale: Clarification of the method of identification of dynamic objects.*

   Replace General Rule 1) with:

   1) Case:

      a) If <descriptor name> does not identify an SQL descriptor area, then an exception condition is raised: *invalid SQL descriptor name*.

      b) Otherwise, the SQL descriptor area identified by <descriptor name> is destroyed.

19.4 <get descriptor statement>

1. *Rationale: Clarification of the method of identification of dynamic objects.*

   Replace General Rule 1) with:

   1) If <descriptor name> does not identify an SQL descriptor area, then an exception condition is raised: *invalid SQL descriptor name*.

19.5 <set descriptor statement>

1. *Rationale: Clarification of the method of identification of dynamic objects.*

   Replace General Rule 1) with:

   1) If <descriptor name> does not identify an SQL descriptor area, then an exception condition is raised: *invalid SQL descriptor name*.

19.6 <prepare statement>

1. *Rationale: Clarification of the method of identification of dynamic objects.*

   Replace General Rule 11) with:

   11) If <SQL statement name> identifies a prepared statement *PS*, then an implicit
   `DEALLOCATE PREPARE` *SSN*
   is executed, where *SSN* is an <SQL statement name> that identifies *PS*.

2. *Rationale: Remove one redundant rule and one rule in the wrong place.*

   Delete General Rules 13) and 14).

19.8 <deallocate prepared statement>

1. *Rationale: Delete spurious Syntax Rule.*

   Replace Syntax Rule 1) with:

   1) If <SQL statement name> is a <statement name>, then

      Case:

a) If the <deallocate prepared statement> is contained in an <SQL-invoked routine>, then the innermost containing <SQL-invoked routine> shall contain a <prepare statement> whose <statement name> is equivalent to the <statement name> of the <deallocate prepared statement>.

b) Otherwise, the containing <SQL-client module definition> shall contain a <prepare statement> whose <statement name> is equivalent to the <statement name> of the <deallocate prepared statement>

2. *Rationale: Clarification of the method of identification of dynamic objects.*

Replace General Rules 1) and 2) with:

1) If <SQL statement name> does not identify a prepared statement, then an exception condition is raised: *invalid SQL statement name*.

2) If <SQL statement name> identifies a prepared statement that is the <cursor specification> of an open cursor, then an exception condition is raised: *invalid cursor state*.

19.9 <describe statement>

1. *Rationale: Delete spurious Syntax Rule.*

Replace Syntax Rule 1) with:

1) If <SQL statement name> is a <statement name>, then

Case:

a) If the <describe statement> is contained in an <SQL-invoked routine>, then the innermost containing <SQL-invoked routine> shall contain a <prepare statement> whose <statement name> is equivalent to the <statement name> of the <describe statement>.

b) Otherwise, the containing <SQL-client module definition> shall contain a <prepare statement> whose <statement name> is equivalent to the <statement name> of the <describe statement>

2. *Rationale: Clarification of the method of identification of dynamic objects.*

Replace General Rules 1) to 4) with:

1) If <SQL statement name> is specified and does not identify a prepared statement, then an exception condition is raised: *invalid SQL statement name*.

2) If <extended cursor name> is specified and does not identify a cursor, then an exception condition is raised: *invalid cursor name*.

3) If <descriptor name> does not identify an SQL descriptor area, then an exception condition is raised: *invalid SQL descriptor name*.

19.10 <input using clause>

1. *Rationale: Clarification of the method of identification of dynamic objects.*

   Replace General Rule 1) with:

   1) If <descriptor name> does not identify an SQL descriptor area, then an exception condition is raised: *invalid SQL descriptor name*.

19.11 <output using clause>

1. *Rationale: Clarification of the method of identification of dynamic objects.*

   Replace General Rule 1) with:

   1) If <descriptor name> does not identify an SQL descriptor area, then an exception condition is raised: *invalid SQL descriptor name*.

19.12 <execute statement>

1. *Rationale: Delete spurious Syntax Rule.*

   Replace Syntax Rule 1) with:

   1) If <SQL statement name> is a <statement name>, then

      Case:

      a) If the <execute statement> is contained in an <SQL-invoked routine>, then the innermost containing <SQL-invoked routine> shall contain a <prepare statement> whose <statement name> is equivalent to the <statement name> of the <execute statement>.

      b) Otherwise, the containing <SQL-client module definition> shall contain a <prepare statement> whose <statement name> is equivalent to the <statement name> of the <execute statement>

2. *Rationale: Clarification of the method of identification of dynamic objects.*

   Replace General Rule 1) with:

   1) If <SQL statement name> does not identify a prepared statement *P*, then an exception condition is raised: *invalid SQL statement name*.

3. *Rationale: Reliably distinguish a <dynamic single row select statement> from a <dynamic select statement>.*

   Replace General Rule 3) with:

3) If *PS* is a <dynamic select statement> then:

Case:

a) If *PS* does not conform to the Format and Syntax Rules of a <dynamic single row select statement>, then an exception condition is raised: *dynamic SQL error — cursor specification cannot be executed*.

b) Otherwise, *PS* is treated as a <dynamic single row select statement>.

4. *Rationale: Erroneous and redundant General Rules.*

Delete General Rules 10) and 11).

19.13 <execute immediate statement>

1. *Rationale: Avoid forcing implementations to define useless values.*

Replace General Rule 3) with:

3) Let *SV* be <SQL statement variable>. <execute immediate statement> is equivalent to the following:

```
PREPARE IMMEDIATE_STMT FROM SV ;
EXECUTE IMMEDIATE_STMT ;
DEALLOCATE PREPARE IMMEDIATE_STMT ;
```

where *IMMEDIATE_STMT* is an implementation-dependent <statement name> that does not identify any existing prepared statement.

19.14 <dynamic declare cursor>

1. *Rationale: Correct a Syntax Rule to reference only a <prepare statement> not in a <schema statement>.*

Replace Syntax Rule 2) with:

2) The containing <SQL-client module definition> shall contain, without an intervening <schema statement>, a <prepare statement> whose <statement name> is equivalent to the <statement name> of the <dynamic declare cursor>.

19.15 <allocate cursor statement>

1. *Rationale: Clarify the specifications concerning dynamic result sets.*

Replace the Function with:

Function

Define a cursor based on a prepared statement for a <cursor specification> or assign a cursor to the result set sequence returned from an SQL-invoked procedure.

2. *Rationale: Clarification of the method of identification of dynamic objects.*

   Replace General Rule 1) with:

   1) Let *S* be the <simple value specification> immediately contained in <extended cursor name>. Let *V* be the character string that is the result of
   `TRIM ( BOTH ' ' FROM` *`S`* `)`

      Case:

      a) If *V* does not conform to the Format and Syntax Rules of an <identifier>, then an exception condition is raised: *invalid cursor name*.

      b) Otherwise let *ECN* be the <extended cursor name>. The value of *ECN* is *V*.

   2) If the *ECN* identifies a cursor then an exception condition is raised: *invalid cursor name*.

3. *Rationale: Clarification of the method of identification of dynamic objects.*

   Replace General Rule 3) a) and b) with:

   3) ...

      a) If <extended statement name> does not identify a prepared statement, then an exception condition is raised: *invalid SQL statement name*.

      b) If the prepared statement identified by <extended statement name> is not a <cursor specification>, then an exception condition is raised: *dynamic SQL error — prepared statement not a cursor specification*.

4. *Rationale: Clarify the specifications concerning dynamic result sets.*

   Replace General Rule 4) with:

   4) If <result set cursor> is specified, then:

      a) Let *SIP* be the SQL-invoked procedure identified by <specific routine designator>. Let *INV* be the active SQL-invoked routine of the current routine execution context.

      b) If the SQL-session context of the current SQL-session does not include a result set sequence *RSS* brought into existence by an invocation of *SIP* by *INV*, then an exception condition is raised: *invalid SQL-invoked procedure reference*.

      c) If *RSS* is empty, then an exception condition is raised: *no data — no additional result sets returned*.

      d) An association is made between the <extended cursor name> and the first result set *FRS* in *RSS*.

e) Cursor *CR* is placed in the open state.

f) *FRS* is the result set of *CR*.

g) The position of *CR* is the initial cursor position of *FRS*.

h) *CR* is not scrollable.

i) *CR* is not updatable.

19.16 <dynamic open statement>

1. *Rationale: Clarification of the method of identification of dynamic objects.*

   Replace Syntax Rule 1) with:

   1) If <dynamic cursor name> *DCN* is a <cursor name> *CN*, then the containing <SQL-client module definition> shall contain a <dynamic declare cursor> *DDC* whose <cursor name> is *CN*.

2. *Rationale: Clarification of the method of identification of dynamic objects.*

   Replace General Rules 1) and 2) with:

   1) Case:

      a) If *DCN* is a <cursor name> and the <statement name> contained in *DDC* does not identify a prepared statement that is a <cursor specification>, then an exception condition is raised: *invalid SQL statement name*.

      b) Otherwise, if *DCN* does not identify a cursor, then an exception condition is raised: *invalid cursor name*.

   2) Let CR be the cursor identified by DCN.

3. *Rationale: Factor out what is common to all cursors.*

   Replace General Rule 6) with:

   6) The General Rules of Subclause 14.28, "Effect of opening a cursor", are applied with *CR* as *CURSOR*.

19.17 <dynamic fetch statement>

1. *Rationale: Clarification of the method of identification of dynamic objects.*

   Delete Syntax Rule 3).

2. *Rationale: Clarification of the method of identification of dynamic objects.*

   Insert the following new General Rule before General Rule 1):

   0.1) If *DCN* does not identify a cursor then an exception condition is raised: *invalid cursor name*.

   0.2) Let *CR* be the cursor identified by *DCN*.

3. *Rationale: Factor out what is common to all cursors.*

   Replace General Rule 1) with:

   1) The General Rules of Subclause 14.29, "Determination of the current row of a cursor", are applied with *CR* as *CURSOR* and <fetch orientation> as *FETCH ORIENTATION*.

   1.1) If a completion condition *no data* has been raised, then no further General Rules of this Subclause are applied.

4. *Rationale: Factor out what is common to all cursors.*

   Insert the following General Rule after General Rule 2)

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

19.19 <dynamic close statement>

1. *Rationale: Clarification of the method of identification of dynamic objects.*

   Insert the following new General Rules before General Rule 1).

   0.1) If *DCN* does not identify a cursor then an exception condition is raised: *invalid cursor name*.

   0.2) Let *CR* be the cursor identified by *DCN*.

2. *Rationale: Factor out what is common to all cursors.*

   Replace General Rule 1) with:

   1) The General Rules of Subclause 14.30, "Effect of closing a cursor", are applied with *CR* as *CURSOR*

19.20 <dynamic delete statement: positioned>

1. *Rationale: Clarification of the method of identification of dynamic objects.*

   Replace General Rules 1) and 2) with:

1) If *DCN* does not identify a cursor, then an exception condition is raised: *invalid cursor name*.

2. *Rationale: An updatable cursor may has exactly one leaf underlying table that is one-to-one.*

   Replace General Rule 5) with:

   5) Let *T* be the simply underlying table of *CR*. *T* is the *subject table* of the <dynamic delete statement: positioned>. Let *LUT* be the leaf underlying table such that *T* is one-to-one with respect to *LUT*. Let *LUTN* be a <table name> that identifies *LUT*.

19.21 <dynamic update statement: positioned>

1. *Rationale: Clarification of the method of identification of dynamic objects.*

   Replace General Rules 1) and 2) with:

   1) If *DCN* does not identify a cursor, then an exception condition is raised: *invalid cursor name*.

2. *Rationale: An updatable cursor may has exactly one leaf underlying table that is one-to-one.*

   Replace General Rule 5) with:

   5) Let *T* be the simply underlying table of *CR*. *T* is the *subject table* of the <dynamic delete statement: positioned>. Let *LUT* be the leaf underlying table such that *T* is one-to-one with respect to *LUT*. Let *LUTN* be a <table name>that identifies *LUT*.

19.22 <preparable dynamic delete statement: positioned>

1. *Rationale: An updatable cursor may has exactly one leaf underlying table that is one-to-one.*

   Replace Syntax Rule 1) lead-in paragraph with:

   1) If <target table> is not specified, then let *QE* be the <query expression> simply contained in the <cursor specification> identified by <cursor name>. Let *LUT* be the leaf underlying table of *QE* such that *QE* is one-to-one with respect to *QE*. Let *TN* be the name of *LUT*.

19.23 <preparable dynamic update statement: positioned>

1. *Rationale: An updatable cursor may has exactly one leaf underlying table that is one-to-one.*

   Replace Syntax Rule 1) lead-in paragraph with:

1) If <target table> is not specified, then let *QE* be the <query expression> simply contained in the <cursor specification> identified by <cursor name>. Let *LUT* be the leaf underlying table of *QE* such that *QE*is one-to-one with respect to *QE*. Let *TN* be the name of *LUT*.

22 Diagnostics management

22.1 <get diagnostics statement>

1. *Rationale: Assign a value rather than a keyword.*

   Replace General Rule 6) with:

   6) The General Rules of Subclause 9.2, "Store assignment", apply to <simple target specification> and the value of whichever of <statement information item name> or <condition information item name> is specified, as *TARGET* and *VALUE*, respectively.

23 Status codes

23.1 SQLSTATE

1. *Rationale: Change of terminology.*

   Insert the following two rows to Table 32, "SQLSTATE class and subclass values".

Table 32 — SQLSTATE class and subclass values

Category	Condition	Class	Subcondition	Subclass
N	*no data*	02	*no additional result sets returned*	001
W	*warning*	01	*result sets returned*	00C

24 Conformance

24.3 Implied feature relationships of SQL/Foundation

1. *Rationale: Feature F691 has been deleted.*

   Delete these rows from Table 34, "Implied feature relationships of SQL/Foundation":

Table 34 — Implied feature relationships of SQL/Foundation

Feature ID	Feature Name	Implied Feature ID	Implied Feature Name
F691	Collation and translation	F695	Translation support
F691	Collation and translation	F690	Collation support

2. *Rationale: Add some missing implied feature relationships.*

   Insert the following rows to Table 34:

Table 34 — Implied feature relationships of SQL/Foundation

Feature ID	Feature Name	Implied Feature ID	Implied Feature Name
F053	OVERLAPS predicate	F052	Intervals and date arithmetic
T152	DISTINCT predicate with negation	T151	DISTINCT predicate
T272	Enhanced savepoint management	T271	Savepoints
T432	Nested and concatenated GROUPING SETS	T431	Extended grouping capabilities
T433	Multiargument GROUPING function	T431	Extended grouping capabilities
T652	SQL-dynamic statements in SQL routines	B031	Basic dynamic SQL
T654	SQL-dynamic statements in external routines	B031	Basic dynamic SQL

Annex A

(informative)

SQL Conformance Summary

1. *Rationale: The word "specific"" is not necessary.*

   Replace item 71) a) i) with:

   71) ...

   a) ...

   i) Without Feature F611, "Indicator data types", in conforming SQL language, the declared types of <indicator parameter>s and <indicator variable>s shall be the same implementation-defined data type.

Annex B

(informative)

Implementation-defined elements

1. *Rationale: Implementation-dependent mappings of user identifiers are of no possible use, so they must be implementation-defined.*

   Insert the following Subitem:

   19.1) Subclause 4.34, "Basic security model"

   a) The mapping of <authorization identifier>s to operating system users is implementation-defined.

2. *Rationale: Add an item previously omitted.*

   Insert the following Subitem:

   20) ...

   c.1) It is implementation-defined whether or not, or how, a <rollback statement> that references a <savepoint specifier> affects diagnostics area contents, the contents of SQL descriptor areas, and the status of prepared statements.

3. *Rationale: "Intermediate SQL" conformance level is no longer defined.*

   Replace item 28) e) with:

   28) ...

      e) Without Feature F611, "Indicator data types", in conforming SQL language, the declared types of <indicator parameter>s and <indicator variable>s shall be the same implementation-defined data type.

4. *Rationale: Document the new implementation-defined elements in the new Subclause 9.24, "Determination of view and view component privileges."*

   Insert the following list item:

   43.1) Subclause 9.24, "Determination of view and view component privileges", with *V* as the *VIEW*:

      a) If an <authorization identifier> *A* has UPDATE privilege on every column of a view *V*, it is implementation-defined whether *A* has UPDATE privilege on *V*.

      b) If an <authorization identifier> *A* has INSERT privilege on every column of a view *V*, it is implementation-defined whether *A* has INSERT privilege on *V*.

5. *Rationale: Cursor ordering is specified in Subclause 10.10, "<sort specification list>".*

   Delete Item 58) a).

Annex C

(informative)

Implementation-dependent elements

1. *Rationale: Record a previously unrecorded item.*

   Insert the following Item:

   4.1) Whether or not result sets whose positions are greater than that maximum number are returned is implementation-dependent.

2. *Rationale: Implementation-dependent mappings of user identifiers are of no possible use, so they must be implementation-defined.*

   Delete Item 8)

3. *Rationale: Clarify whether prepared statements may be deallocated.*

   Insert the following Item:

   36.1) Whether a prepared statement, other than the prepared statement of a held cursor, remains in existence is implementation-dependent.

4. *Rationale: Remove a reference to a rule that has been moved.*

   Delete Item 40).

5. *Rationale: Record a previously unrecorded item.*

   Insert the following Item:

   42.1) The <statement name> of the statement prepared when an <execute immediate statement> is executed is implementation-dependent.

Annex E

(informative)

Incompatibilities with ISO/IEC 9075-2:1999

1. *Rationale: The definition of simply updatable has been narrowed.*

   Insert the following item:

   18) It was intended that ISO/IEC 9075-2:1999 the definition of *simply updatable* would be equivalent to the ISO/IEC 9075:1992 definition of *updatable*. However, it was found that ISO/IEC 9075-2:1999's definition of *simply updatable* was broader than that. In this edition of ISO/IEC 9075, the definition of *simply updatable* has been tightened, in order to make it equivalent to the ISO/IEC 9075:1992 definition of *updatable*.

2. *Rationale: Acknowledge an incompatability.*

   Insert the following item:

   19) Feature T411, “UPDATE statement: SET ROW option”, which provided the ability in an UPDATE statement to specify the new value of the entire row by specifying SET ROW = <row value expression>, has been deleted.

Annex F

(informative)

SQL feature taxonomy

1. *Rationale: Remove some incorrect conformance features.*

   Delete Feature F691 and Feature F696 from Table 36, “Feature taxonomy for optional features”

2. *Rationale: Add missing Conformance feature to the table.*

   Insert the following row to Table 36, “Feature taxonomy for optional features”.

Table 4 — Feature taxonomy for optional features

	Feature ID	Feature Name
1081	**F690**	**Collation support**