INTERNATIONAL STANDARD ISO/IEC 9075-4:1999
TECHNICAL CORRIGENDUM 2

Published 2003-06-01

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

Information technology — Database languages — SQL —

Part 4:
Persistent Stored Modules (SQL/PSM)

TECHNICAL CORRIGENDUM 2

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

Partie 4: Modules stockés persistants (SQL/PSM)

RECTIFICATIF TECHNIQUE 2

Technical Corrigendum 2 to ISO/IEC 9075-4:1999 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange*. ISO/IEC 9075-4:1999/Cor. 2:2003 cancels and replaces ISO/IEC 9075-4:1999/Cor. 1:2000.

Statement of purpose for rationale:

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

Notes on numbering:

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

ICS 35.060 **Ref. No. ISO/IEC 9075-4:1999/Cor.2:2003(E)**

Published in Switzerland

Contents

Page

Information technology — Database languages — SQL —

Part 4:
Persistent Stored Modules (SQL/PSM)

TECHNICAL CORRIGENDUM 2

3.3.1.1 Exceptions

1. Rationale: Delete definition of an unused term.

Delete Subclause 3.3.1.1.

3.3.1.2 Other terms

1. Rationale: Delete what is moved to Part 2.

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

Insert this paragraph An SQL-statement *S1* may be said to be executed as a direct result of executing an <SQL control statement> *S2* if *S2* contains *S1*.

3.3.2.1 Clause, Subclause, and Table relationships

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

Insert the following row into Table 1, "Clause, Subclause, and Table relationships":

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Subclause 4.10.2a "Preparable and immediately executable SQL-statements"	Subclause 4.6.5 Preparable and immediately executable SQL-statements	ISO/IEC 9075-5

2. Rationale: Correct cross reference.

Replace the following rows in Table 1, "Clause, Subclause, and Table relationships":

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence 4.10.7 "SQL-statement atomicity"
Subclause 4.10.7 "SQL-statement atomicity"	*(none)*	*(none)*

with:

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence

Subclause 4.10.7 “SQL-statement atomicity”	Subclause 4.30.4 “SQL-statement atomicity”	ISO/IEC 9075-2

3. Rationale: Address requirement for multiple diagnostics areas

Insert the following row into Table 1, "Clause, Subclause, and Table relationships":

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Subclause 4.11, “SQL-sessions”	Subclause 4.34, “SQL-sessions”	ISO/IEC 9075-2

4. Rationale: Correct cross reference.

Replace the following rows in Table 1, "Clause, Subclause, and Table relationships":

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Subclause 4.11.1, ‘‘Privileges’’	Subclause 4.31, “Basic security model”	ISO/IEC 9075-2
Clause 11, “SQL-client module”	Subclause 13.1, “<SQL-client module definition>”	ISO/IEC 9075-2

with:

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Subclause 4.11.1, ‘‘Privileges’’	Subclause 4.31.2, ‘‘Privileges’’	ISO/IEC 9075-2
Clause 11, “SQL-client module”	Subclause 13, “SQL-client modules”	ISO/IEC 9075-2

4.2 SQL-invoked routines

1. Rationale: Correct the Part Merge instructions

Replace the 2nd paragraph with:

Replace 32nd paragraph An SQL-invoked routine has a *routine SQL-path*, which is inherited from its containing SQL-server module or schema, the current SQL-session, or the containing SQL-client module.

Replace the 3rd paragraph with:

Insert in the 34th paragraph — If the SQL-invoked routine is not a schema-level routine, then the <SQL-server module name> of the SQL-server module that includes the SQL-invoked routine and the <schema name> of the schema that includes the SQL-server module.

4.7 Diagnostics areas

1. Rationale: Address requirement for multiple diagnostics areas

Replace the entire Subclause with:

Replace paragraph 8 The <get diagnostics statement> is used to obtain information from an occupied condition area, referenced by its ordinal position within the specified diagnostics area. Normally, only the first (i.e., current) area may be specified. However, if a handler is active, it is also possible to reference the second (i.e., most recently stacked) area, in order to obtain information about the condition that caused the handler to become active.

Insert this paragraph Information about a completion or exception condition that causes a handler to be activated is placed into one or more condition areas of the first diagnostics area before any handler is activated. The diagnostics area stack is then pushed so that the handler can access that information even while its own execution is causing the first diagnostics area to be modified.

Insert this paragraph The first diagnostics area is emptied during the execution of a <signal statement>. Information is added to the first diagnostics area during the execution of a <resignal statement>.

4.8 Cursors

1. Rationale: Fix bug in processing results sets in PSM.

Replace the 1st paragraph with:

Insert this paragraph For every <declare cursor> in a <compound statement>, a cursor is effectively created each time the <compound statement> is executed, and destroyed when that execution completes, unless the cursor is an open result set cursor.

4.9 Condition handling

1. Rationale: Address requirement for multiple diagnostics areas

Replace the 5th paragraph with:

A condition represents an error or informational state caused by execution of an <SQL procedure statement>. Conditions are raised to provide information in a diagnostics area about the execution of an <SQL procedure statement>.

2. Rationale: Address requirement for multiple diagnostics areas. Simplify and clarify where execution resumes.

Replace the 12th, 13th and 14th paragraphs, including NOTE 2, with:

If a handler type specifies CONTINUE, then, when the handler is activated, it will:

— Push the diagnostics area stack.

— Execute the handler action.

— Pop the diagnostics area stack.

— Cause the SQL-session to continue as it would have done if execution of the innermost executing statement that raised the condition had completed.

If a handler type specifies EXIT, then, when the handler is activated, it will:

— Push the diagnostics area stack.

— Execute the handler action.

— Pop the diagnostics area stack.

— Implicitly LEAVE the <compound statement> for which the handler was created with no active exception condition.

If a handler type specifies UNDO, then, when the handler is activated, it will:

— Push the diagnostics area stack.

— Roll back all of the changes to SQL-data or to schemas by the execution of every SQL-statement contained in the SQL-statement list of the <compound statement> at the scope of the handler and cancel any <SQL procedure statement>s triggered by the execution of such statements.

— Execute the handler action.

— Pop the diagnostics area stack.

— Cause the SQL-session to continue as it would have done if execution of the <compound statement> for which the handler was created had completed.

4.10 SQL-statements

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

Insert the following Subclause after Subclause 4.10.2, "Embeddable SQL-statements":

4.10.2a Preparable and immediately executable SQL-statements

Insert this paragraph Consequently, the following SQL-control statements are not preparable:

— <compound statement>

— <case statement>

— <if statement>

— <iterate statement>

— <leave statement>

— <loop statement>

— <while statement>

— <repeat statement>

— <for statement>

— <assignment statement>

Insert this paragraph Consequently, the following SQL-control declarations are not preparable:

— <condition declaration>

— <handler declaration>

— <SQL variable declaration>

4.10.1 SQL-statements classified by function

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

Delete the 2nd bullet from the 2nd paragraph.

4.10.7 SQL-statement atomicity

1. Rationale: Not all SQL-statements are atomic.

Insert the following paragraph:

Insert this paragraph The following are non-atomic SQL-statements:

— <assignment statement>

— <compound statement>, unless BEGIN ATOMIC is specified

— <case statement>

— <if statement>

— <loop statement>

— <while statement>

— <repeat statement>

— <for statement>

4.11 SQL-sessions

1. Rationale: Address requirement for multiple diagnostics areas

Insert the following Subclause:

4.11 SQL-sessions

Insert this paragraph Certain operations during an SQL-session *SS* are possible only when *SS* is in *condition handling mode*. This mode becomes in effect when execution of an SQL-statement has completed to the extent that all diagnostics information pertaining to that execution is recorded in the first diagnostics area. Condition handling mode ceases to be in effect when execution of the next SQL-statement begins.

5.1 <token> and <separator>

1. Rationale: Editorial - Correct reserved and non-reserved word lists.

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

```
| REDO
```

6.2 <identifier chain>

1. Rationale: Correct definition of possible scope tags. Adjust rule numbering to accommodate changes in Part 2.

Replace Syntax Rule 2) with:

2) Replace SR7)a)ii) Otherwise *IC* shall be contained within the scope of one or more exposed <table or query name>s or <correlation name>s whose associated tables include a column whose <identifier> is equivalent to I_1 or within the scope of a <routine name> whose associated <SQL parameter declaration list> includes an SQL parameter whose <identifier> is equivalent to I_1 or within the scope of one or more <beginning label>s whose associated <local declaration list> includes an SQL variable whose <identifier> is equivalent to I_1. Let the phrase *possible scope tags* denote those exposed <table or query name>s, <correlation name>s, <routine name>s and <beginning label>s.

Replace Syntax Rule 3) with:

3) Insert after SR7)a)ii)1)B) If *IPST* is a <beginning label>, then let *SV* be the SQL variable whose <identifier> is equivalent to I_1. PIC_1 is the basis of *IC*, the basis length is 1 (one), the basis scope is the scope of *SP*, and the basis referent is *SV*.

8.1 <routine invocation>

1. *Rationale: The formulation "<SQL parameter name> of an SQL parameter of an SQL-invoked routine" remains due to historical reasons and is a synonym to <SQL parameter reference>, which is now used in the BNF definition of <target specification>.*

Replace Syntax Rule 6) with:

6) Replace SR8)c)i)4)B) If A_i is an <SQL variable reference>, an <SQL parameter reference>, or a <column reference>, then P_i shall be assignable to A_i, according to the Syntax Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2, with A_i and P_i as *TARGET* and *VALUE*, respectively.
NOTE 6.1 — The <column reference> can only be a new transition column reference.

2. *Rationale: General Rule 3) covers the case of specifying an <SQL variable name>, so General Rule 2) is redundant. Correct tagging in General Rule 3). The formulation "<SQL parameter name> of an SQL parameter of an SQL-invoked routine" remains due to historical reasons and is a synonym to <SQL parameter reference>, which is now used in the BNF definition of <target specification>.*

Delete General Rule 2).

Replace General Rule 3) with:

3) Replace GR10)b)ii) If TS_i is an <SQL variable reference>, an <SQL parameter reference>, or a <column reference>, then CPV_i is assigned to TS_i according to the rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2.
NOTE 7.1 — The <column reference> can only be a new transition column reference.

9.12 <drop collation statement>

1. *Rationale: Standardise terminology.*

Replace the Function with:

Destroy a collation.

2. *Rationale: Add missing rule.*

Insert the following Syntax Rule:

1) Insert after SR4 If RESTRICT is specified, then *C* shall not be referenced in the module descriptor of any SQL-server module.

9.13 <drop transliteration statement>

1. Rationale: Add missing rule.

Insert the following Syntax Rule:

1) Insert after SR5 If RESTRICT is specified, then *C* shall not be referenced in the module descriptor of any SQL-server module.

9.18 <SQL-server module definition>

1. Rationale: Editorial.

In the Format, replace the production <SQL-server module definition> with:

```
<SQL-server module definition> ::=
     CREATE MODULE <SQL-server module name>
     [ <SQL-server module character set specification> ]
     [ <SQL-server module schema clause> ]
     [ <SQL-server module path specification> ]
     [ <temporary table declaration>... ]
     <SQL-server module contents>...
     END MODULE
```

9.19 <drop module statement>

1. Rationale: Remove redundant and confusing references to assertion descriptor.

Replace Syntax Rule 3) c) as follows:

3) c) The <search condition> of any constraint descriptor.

10.2 <revoke statement>

1. Rationale: There is no <row value expression> immediately contained in a <set clause>.

Replace Syntax Rule 6) f) with:

6) f) SELECT privileges on every <table reference> and <column reference> contained in a <value expression> simply contained in an <update target> contained in the <SQL routine body> of any SQL-invoked routine included in *SSM*.

2. Rationale: Correct the rules for dependency on USAGE privilege for a user-defined type.

Replace Syntax Rule 6) j) with:

6) j) USAGE privilege on every domain, every collation, every character set, and every transliteration whose name is contained in the <routine body> of any SQL-invoked routine included in *SSM*.

j.1) USAGE privilege on every user-defined type *UDT* such that there is a <data type> contained in the

<routine body> of any SQL-invoked routine included in *SSM* that is usage-dependent on *UDT*.

3. *Rationale: Replace usage of undefined tag RD.*

Replace Syntax Rules 6) l), m) and n) with:

6) l) SELECT privilege WITH HIERARCHY OPTION on at least one supertable of the scoped table of any <reference resolution> that is contained in any <query expression> contained in the <SQL routine body> of any SQL-invoked routine included in *SSM*.

m) SELECT privilege WITH HIERARCHY OPTION on at least one supertable of the scoped table of any <reference resolution> that is contained in any <table expression> or <select list> immediately contained in a <select statement: single row> contained in the <SQL routine body> of any SQL-invoked included in *SSM*.

n) 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> contained in a <delete statement: searched>, an <update statement: searched>, or a <merge statement> contained in the <SQL routine body> of any SQL-invoked routine included in *SSM*.

4. *Rationale: Replace usage of undefined tag RD and there is no <row value expression> immediately contained in a <set clause>.*

Replace Syntax Rule 6) o) with:

6) o) SELECT privilege WITH HIERARCHY OPTION on at least one supertable of the scoped table of any <reference resolution> that is contained in any <value expression> simply contained in an <update target> contained in the <SQL routine body> of any SQL-invoked included in *SSM*.

11.1 Calls to an <externally-invoked procedure

1. *Rationale: Address requirement for multiple diagnostics areas*

In Syntax Rule 1) add the following text to the package definition for SQLSTATE_CODES:

```
DIAGNOSTICS_EXCEPTION_STACKED_DIAGNOSTICS_AREA_ACCESSED_WHEN_NO_HANDLER_ACTIVE:
    constant SQLSTATE_TYPE := "0Z002"
```

11.2 <SQL procedure statement>

1. *Rationale: Address requirement for multiple diagnostics areas*

Insert the following General Rules:

1) Replace GR5)a)iii)6) If *S* is not a <compound statement>, then the first diagnostics area is emptied.

2) Insert after GR7) Condition handling mode becomes in effect in the SQL-session.

3) Insert after GR7) The General Rules of Subclause 13.2, "<handler declaration>", are applied.

4) Insert this rule Condition handling mode ceases to be in effect in the SQL-session.

2. *Rationale: Editorial - to correct reference.*

Replace Conformance Rule 1) with:

1) Without Feature P001, "Stored modules", an <SQL procedure statement> shall not be an <SQL server module definition> or a <drop module statement>.

11.1 Calls to an externally-invoked procedure

1. *Rationale: Remove undefined constants.*

Replace Syntax Rules 1) with:

1) Replace SR2)e)

```
CASE_NOT_FOUND_FOR_CASE_STATEMENT_NO_SUBCLASS:
    constant SQLSTATE_TYPE :="20000";
DATA_EXCEPTION_NULL_VALUE_IN_FIELD_REFERENCE:
    constant SQLSTATE_TYPE :="22006";
RESIGNAL_WHEN_HANDLER_NOT_ACTIVE_NO_SUBCLASS:
    constant SQLSTATE_TYPE :="0K000";
UNHANDLED_USER_DEFINED_EXCEPTION_NO_SUBCLASS:
    constant SQLSTATE_TYPE :="45000";
```

12.2 <fetch statement>

1. *Rationale: Replace misleading text.*

Replace Syntax Rules 2) and 3) with:

2) Replace SR6)a)i) If *TS* is an <SQL variable reference> or an <SQL parameter reference>, then the Syntax Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9072-2, apply to *TS* and the row type of table *T* as *TARGET* and *VALUE*, respectively.

3) Replace SR6)b)ii) For each <target specification> *TS1* that is an <SQL variable reference> or an <SQL parameter reference>, the Syntax Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9072-2, apply to *TS1* and the corresponding column of table *T* as *TARGET* and *VALUE*, respectively.

Replace General Rules 1) with:

1) Replace GR7)a)i) If *TS* is an <SQL variable reference> or an <SQL parameter reference>, then the General Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9072-2, apply to *TS* and the current row as *TARGET* and *VALUE*, respectively.

2. *Rationale: Replace misleading text and correct symbol.*

Replace General Rule 2) with:

2) Replace GR7)b)i) If *TV* is an <SQL variable reference> or an <SQL parameter reference>, then the General Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9072-2, apply to *TV* and *SV* as *TARGET* and *VALUE*, respectively.

13.1 <compound statement>

1. Rationale: Remove confusion of duplicate tags.

Replace Syntax Rule 8) with:

8) Let *CON* be the <condition name> immediately contained in a <condition declaration> contained in a <local declaration list>. The *declared local name* of the <condition declaration> is *CON*.

2. Rationale: Editorial.

Replace General Rule 3) b) with:

3) b) The General Rules of Subclause 13.5, "<SQL procedure statement>", in ISO 9075-2, are evaluated with S_i as the *executing statement*.

3. Rationale: Correct editorial error.

Replace General Rule 3) c) with:

3) c) If the execution of S_i terminates with exception conditions, then:

i) The following <resignal statement> is effectively executed without further Syntax Rule checking:

RESIGNAL

ii) If there are unhandled exception conditions or completion conditions other than *successful completion* at the completion of the execution of a handler (if any), then the execution of *CS* is terminated immediately.

1) For every open cursor *CR* that is declared in the <local declaration list> of *CS*, the following SQL-statement is effectively executed:

CLOSE *CR*

2) The SQL variables, cursors, and handlers specified in the <local declaration list>, the <local cursor declaration list>, and the <local handler declaration list> of *CS* are destroyed.

4. Rationale: Fix bug in processing results sets in PSM.

Replace General Rule 4) with:

4) For every open cursor *CR* that is not a result set cursor that is declared in the <local cursor declaration list> of *CS*, the following SQL-statement is effectively executed:

CLOSE *CR*

NOTE 16.1 — Result set cursor is defined in Subclause 4.29, Cursors in ISO/IEC 9075-2.

Replace General Rule 5) with:

5) The SQL variables, cursors that are not open result set cursors, and handlers specified in <local declaration list>, the <local cursor declaration list>, and the <local handler declaration list> of *CS* are destroyed.

13.2 <handler declaration>

1. Rationale: Simplify and clarify where execution resumes.

Replace General Rule 2) with:

2) Let *CS* be the <compound statement> simply containing *HD*.

2. Rationale: Fix bug in processing results sets in PSM, simplify and clarify where execution resumes and address the requirement for multiple diagnostics areas.

Replace General Rule 3) with:

3) When *H* is activated,

Case:

a) If *HD* specifies CONTINUE, then:

i) The General Rules of Subclause 22.2, "Pushing and popping diagnostics areas" are applied, with PUSH as *OPERATION* and the diagnostics area stack as *STACK*.

ii) *HA* is executed.

iii) Case:

A) If there is an unhandled condition other than *successful completion* at the completion of *HA*, then

I) The General Rules of Subclause 22.2, "Pushing and popping diagnostics areas" are applied, with PUSH as *OPERATION* and the diagnostics area stack as *STACK*.

II) The following <resignal statement> is effectively executed:

RESIGNAL

B) Otherwise:

I) The General Rules of Subclause 22.2, "Pushing and popping diagnostics areas" are applied, with POP as *OPERATION* and the diagnostics area stack as *STACK*.

II) *HA* completes with completion condition *successful completion* and the SQL-session continues as it would have done if execution of the innermost executing statement that raised the condition had completed.

b) If *HD* specifies EXIT, then:

i) The General Rules of Subclause 22.2, "Pushing and popping diagnostics areas" are applied,

with PUSH as *OPERATION* and the diagnostics area stack as *STACK*.

ii) *HA* is executed.

iii) For every open cursor *CR* that was declared in *CS* and that is not a result set cursor, the following statement is implicitly executed:

CLOSE *CR*

NOTE 17.1 — Result set cursor is defined in Subclause 4.29, Cursors in ISO/IEC 9075-2.

iv) Case:

A) If there is an unhandled condition other than *successful completion* at the completion of *HA*, then

I) The General Rules of Subclause 22.2, "Pushing and popping diagnostics areas" are applied, with PUSH as *OPERATION* and the diagnostics area stack as *STACK*.

II) The following <resignal statement> is effectively executed:

RESIGNAL

B) Otherwise:

I) The General Rules of Subclause 22.2, "Pushing and popping diagnostics areas" are applied, with POP as *OPERATION* and the diagnostics area stack as *STACK*.

II) *HA* completes with completion condition *successful completion* and the SQL-session continues as it would have done if execution of *CS* had completed.

c) If *HD* specifies UNDO, then:

i) The General Rules of Subclause 22.2, "Pushing and popping diagnostics areas" are applied, with PUSH as *OPERATION* and the diagnostics area stack as *STACK*.

ii) All changes made to SQL-data or schemas by the execution of SQL-statements contained in the <SQL statement list> of *CS* and any <SQL procedure statement>s triggered by the execution of any such statements are cancelled.

iii) For every open cursor *CR* that was declared in *CS*, the following statement is implicitly executed:

CLOSE *CR*

iv) *HA* is executed.

v) Case:

A) If there is an unhandled condition other than *successful completion* at the completion of *HA*, then

I) The General Rules of Subclause 22.2, "Pushing and popping diagnostics areas" are applied, with PUSH as *OPERATION* and the diagnostics area stack as *STACK*.

II) The following <resignal statement> is effectively executed:

RESIGNAL

B) Otherwise:

I) The General Rules of Subclause 22.2, "Pushing and popping diagnostics areas" are applied, with POP as *OPERATION* and the diagnostics area stack as *STACK*.

II) *HA* completes with completion condition *successful completion* and the SQL-session continues as it would have done if execution of *CS* had completed.

13.5 <assignment statement>

1. Rationale: Correct the reference to a non-existent Format element.

In the Format replace the production for <multiple variable assignment> with:

```
<multiple variable assignment> ::=
     SET <assignment target list> <equals operator> <assigned row>
```

2. Rationale: Prohibit assigning to new transition variables in AFTER triggers

Insert the following Syntax Rule:

2.1) If the <assignment statement> is contained in a <triggered SQL statement> of an AFTER trigger, then the <modified field target> or a <mutated target specification> contained in the <assignment target> shall not be a <column reference>.

13.7 <if statement>

1. Rationale: Use the correct form of containment

Replace Syntax Rule 1) with:

1) If one or more <if statement elseif clause>s are specified, then the <if statement> is equivalent to an <if statement> that does not contain ELSEIF by performing the following transformation recursively:

```
IF <search condition>
  <if statement then clause>
  <if statement elseif clause 1>
  [ <if statement elseif clause>... ]
  [ <if statement else clause> ]
END IF
```

is equivalent to

```
IF <search condition>
  <if statement then clause>
ELSE
```

```
  IF <search condition 1>
    THEN <statement list 1>
    [ <if statement elseif clause>... ]
    [ <if statement else clause> ]
  END IF
END IF
```

where <search condition 1> is the <search condition> simply contained in <if statement elseif clause 1> and <statement list 1> is the <SQL statement list> simply contained in <if statement elseif clause 1>.

13.13 <for statement>

1. Rationale: Remove incompatibility with 9075-4:1996

Replace Syntax Rule 6) with:

6) Let *BL*, *FLVN*, and *SLL* be the <beginning label>, <for loop variable name>, and <SQL statement list> of *FS*, respectively.

a) Let *AT_END* be an implementation-dependent <SQL variable name> that is not equivalent to any other <SQL variable name> or any <SQL parameter name> contained in the outermost containing <SQL-server module definition>, <SQL-invoked routine>, or <compound statement>.

b) Let *NOT_FOUND* be an implementation-dependent <condition name> that is not equivalent to any other <condition name> contained in the outermost containing <SQL-server module definition>, <SQL-invoked routine>, or <compound statement>.

c) Let *CS* be the explicit or implicit <cursor sensitivity>. The <for statement> is equivalent to:

```
BL: BEGIN NOT ATOMIC
  FLVN: BEGIN NOT ATOMIC
    DECLARE CN CS CURSOR FOR FCS;
    DECLARE V1 DT1;
    DECLARE V2 DT2;
    .
    .
    .
    DECLARE VN DTN;
    DECLARE AT_END BOOLEAN DEFAULT FALSE;
    DECLARE NOT_FOUND CONDITION FOR SQLSTATE '02000';
    BEGIN NOT ATOMIC
      DECLARE CONTINUE HANDLER FOR NOT_FOUND
      SET AT_END = TRUE;
      OPEN CN;
      FETCH CN INTO V1, V2, ... VN;
      WHILE NOT AT_END DO
        SLL;
        BEGIN NOT ATOMIC
          FETCH CN INTO V1, V2, ... VN;
        END;
      END WHILE;
      CLOSE CN;
    END;
  END FLVN;
END BL
```

15.1 <embedded SQL host program>

1. Rationale: Remove redundant rule that refers to a non-existent BNF term.

Delete Syntax Rule 4).

16.1 <get diagnostics statement>

1. Rationale: Delete statement codes for declarations.

Delete the following rows in Table 3, "SQL-statement codes":

SQL-statement	Identifier	Code
<handler declaration>	HANDLER	87
<SQL variable declaration>	DECLARE VARIABLE	96

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

Replace the following rows from Table 3, "SQL-statement codes for use in the diagnostics area":

SQL-statement	Identifier	Code
<temporary table declaration>	TEMPORARY TABLE	93
<alter module statement>	ALTER MODULE	95

with:

SQL-statement	Identifier	Code
<iterate statement>	ITERATE	102

3. Rationale: Correct the signal and resignal statements.

Insert the following General Rule:

2) Insert before GR 3)p) If COMMAND_FUNCTION or DYNAMIC_FUNCTION identifies a <signal statement> or <resignal statement>, then the values of CLASS_ORIGIN, SUBCLASS_ORIGIN, CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME, CATALOG_NAME, SCHEMA_NAME, TABLE_NAME, COLUMN_NAME, CURSOR_NAME, MESSAGE_TEXT, MESSAGE_LENGTH, and MESSAGE_OCTET_LENGTH are not set as specified in ISO/IEC 9075-2, but instead are set as specified in 13.15, "<signal statement>" and 13.16, "<resignal statement>" in this part of ISO/IEC 9075.