WG3:HBA-005
H2-2003-307
August, 2003

ISO

International Organization for Standardization

ANSI

American National Standards Institute

ANSI TC NCITS H2

ISO/IEC JTC 1/SC 32/WG 3

Database

Title: (ISO-ANSI Working Draft) Persistent Stored Modules (SQL/PSM)

Author: Jim Melton (Editor)

References:

1) WG3:HBA-002 = H2-2003-304 = 5WD-01-Framework-2003-09, *WD 9075-1 (SQL/Framework)*, September, 2003

2) WG3:HBA-003 = H2-2003-305 = 5WD-02-Foundation-2003-09, *WD 9075-2 (SQL/Foundation)*, September, 2003

3) WG3:HBA-004 = H2-2003-306 = 5WD-03-CLI-2003-09, *WD 9075-3 (SQL/CLI)*, September, 2003

4) WG3:HBA-005 = H2-2003-307 = 5WD-04-PSM-2003-09, *WD 9075-4 (SQL/PSM)*, September, 2003

5) WG3:HBA-006 = H2-2003-308 = 5WD-09-MED-2003-09, *WD 9075-9 (SQL/MED)*, September, 2003

6) WG3:HBA-007 = H2-2003-309 = 5WD-10-OLB-2003-09, *WD 9075-10 (SQL/OLB)*, September, 2003

7) WG3:HBA-008 = H2-2003-310 = 5WD-11-Schemata-2003-09, *WD 9075-11 (SQL/Schemata)*, September, 2003

8) WG3:HBA-009 = H2-2003-311 = 5WD-13-JRT-2003-09, *WD 9075-13 (SQL/JRT)*, September, 2003

9) WG3:HBA-010 = H2-2003-312 = 5WD-14-XML-2003-09, *WD 9075-14 (SQL/XML)*, September, 2003

ISO/IEC JTC 1/SC 32

Date: 2003-07-25

ISO/IEC 9075-4:2003 (E)

ISO/IEC JTC 1/SC 32/WG 3

United States of America (ANSI)

Information technology — Database languages — SQL — Part 4: Persistent Stored Modules (SQL/PSM)

Technologies de l'information— Langages de base de données — SQL — Partie 4: Modules stocké persistants (SQL/PSM)

Document type: International standard
Document subtype:
Document stage: (4) Approval
Document language: English

Contents

Page

Tables

Table

Page

This page intentionally left blank.

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75% of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this International Standard may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 9075-4 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange*.

This second edition of this part of ISO/IEC 9075 cancels and replaces the first edition, ISO/IEC 9075-4:1996.

ISO/IEC 9075 consists of the following parts, under the general title *Information technology — Database languages — SQL*:

— Part 1: Framework (SQL/Framework)

— Part 2: Foundation (SQL/Foundation)

— Part 3: Call-Level Interface (SQL/CLI)

— Part 4: Persistent Stored Modules (SQL/PSM)

— Part 9: Management of External Data (SQL/MED)

— Part 10: Object Language Bindings (SQL/OLB)

— Part 11: Information and Definition Schema (SQL/Schemata)

— Part 13: Routines and Types Using the Java™ Programming Language (SQL/JRT)

— Part 14: XML-Related Specifications (SQL/XML)

Annexes A, B, C, D, E, and F of this part of ISO/IEC 9075 are for information only.

Introduction

The organization of this part of ISO/IEC 9075 is as follows:

1) Clause 1, "Scope", specifies the scope of this part of ISO/IEC 9075.

2) Clause 2, "Normative references", identifies additional standards that, through reference in this part of ISO/IEC 9075, constitute provisions of this part of ISO/IEC 9075.

3) Clause 3, "Definitions, notations, and conventions", defines the notations and conventions used in this part of ISO/IEC 9075.

4) Clause 4, "Concepts", presents concepts used in the definition of persistent stored modules.

5) Clause 5, "Lexical elements", defines a number of lexical elements used in the definition of persistent stored modules.

6) Clause 6, "Scalar expressions", defines a number of scalar expressions used in the definition of persistent stored modules.

7) Clause 7, "Query expressions", defines the elements of the language that produce rows and tables of data as used in persistent stored modules.

8) Clause 8, "Additional common elements", defines additional common elements used in the definition of persistent stored modules.

9) Clause 9, "Schema definition and manipulation", defines the schema definition and manipulation statements associated with the definition of persistent stored modules.

10) Clause 10, "Access control", defines facilities for controlling access to SQL-data.

11) Clause 11, "SQL-client modules", defines the facilities for using persistent stored modules.

12) Clause 12, "Data manipulation", defines data manipulation operations associated with persistent stored modules.

13) Clause 13, "Control statements", defines the control statements used with persistent stored modules.

14) Clause 14, "Dynamic SQL", defines the facilities for executing SQL-statements dynamically in the context of persistent stored modules.

15) Clause 15, "Embedded SQL", defines the host language embeddings.

16) Clause 16, "Diagnostics management", defines enhancements to the facilities used with persistent stored modules.

17) Clause 17, "Information Schema", defines the Information and Definition Schema objects associated with persistent stored modules.

18) Clause 18, "Definition Schema", defines base tables on which the viewed tables containing schema information depend.

19) Clause 19, "Status codes", defines SQLSTATE values related to persistent stored modules.

20) Clause 20, “Conformance”, defines the criteria for conformance to this part of ISO/IEC 9075.

21) Annex A, “SQL Conformance Summary”, is an informative Annex. It summarizes the conformance requirements of the SQL language.

22) Annex B, “Implementation-defined elements”, is an informative Annex. It lists those features for which the body of this part of ISO/IEC 9075 states that the syntax, the meaning, the returned results, the effect on SQL-data and/or schemas, or any other behavior is partly or wholly implementation-defined.

23) Annex C, “Implementation-dependent elements”, is an informative Annex. It lists those features for which the body of this part of ISO/IEC 9075 states that the syntax, the meaning, the returned results, the effect on SQL-data and/or schemas, or any other behavior is partly or wholly implementation-dependent.

24) Annex D, “Incompatibilities with ISO/IEC 9075:1999”, is an informative Annex. It lists the incompatibilities between this edition of this part of ISO/IEC 9075 and ISO/IEC 9075-4:1996.

25) Annex E, “Defect reports not addressed in this edition of this part of ISO/IEC 9075”, is an informative Annex. It describes reported defects in the previous edition of this part of ISO/IEC 9075 that remain in this edition.

26) Annex F, “SQL feature taxonomy”, is an informative Annex. It identifies features of the SQL language specified in this part of ISO/IEC 9075 by a numeric identifier and a short descriptive name. This taxonomy is used to specify conformance and may be used to develop other profiles involving the SQL language.

In the text of this part of ISO/IEC 9075, Clauses begin a new odd-numbered page, and in Clause 5, “Lexical elements”, through Clause 20, “Conformance”, Subclauses begin a new page. Any resulting blank space is not significant.

This page intentionally left blank.

Information technology— Database languages —SQL —

Part 4: Persistent Stored Modules (SQL/PSM)

1 Scope

This part of International Standard ISO/IEC 9075 specifies the syntax and semantics of a database language for declaring and maintaining persistent database language routines in SQL-server modules.

The database language for <externally-invoked procedure>s and <SQL-invoked routine>s includes:

— The specification of statements to direct the flow of control.

— The assignment of the result of expressions to variables and parameters.

— The specification of condition handlers that allow SQL-invoked routines to deal with various conditions that arise during their execution.

— The specification of statements to signal and resignal conditions.

— The declaration of local cursors.

— The declaration of local variables.

It also includes the definition of the Information Schema tables that contain schema information pertaining to SQL-server modules and SQL-invoked routines.

This page intentionally left blank.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

2.1 JTC1 standards

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

[Foundation] ISO/IEC FCD 9075-2:2003, *Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation).*

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

This page intentionally left blank.

3 Definitions, notations, and conventions

This Clause modifies Clause 3, "Definitions, notations, and conventions", in ISO/IEC 9075-2.

3.1 Conventions

This Subclause modifies Subclause 3.3, "Conventions", in ISO/IEC 9075-2.

3.1.1 Use of terms

This Subclause modifies Subclause 3.3.1, "Use of terms", in ISO/IEC 9075-2.

3.1.1.1 Other terms

This Subclause modifies Subclause 3.3.1.1, "Other terms", in ISO/IEC 9075-2.

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

Insert this paragraph The phrase "The scope of a <handler declaration> contained in a *Y* is that *Y*, excluding every <SQL schema statement> contained in that *Y*" means that the scope of the <handler declaration> does not extend to SQL-statements contained in such an <SQL schema statement>; it does, however, extend to the <SQL schema statement> itself.

This page intentionally left blank.

4 Concepts

This Clause modifies Clause 4, "Concepts", in ISO/IEC 9075-2.

4.1 SQL-server modules

An *SQL-server module* is a persistent object defined in a schema and identified by an <SQL-server module name>. SQL-server modules are created with <SQL-server module definition>s and destroyed with <drop module statement>s and by <drop schema statement>s that destroy the schemas that contain them.

An <SQL-server module definition> contains an <SQL-server module name>, an optional <SQL-server module character set specification>, an optional <SQL-server module schema clause>, an optional <SQL-server module path specification>, zero or more declared local temporary tables specified by <temporary table declaration>s, and one or more <SQL-invoked routine>s.

The <SQL-server module name> of an SQL-server module is a <schema qualified name>. The character set specified by the <SQL-server module character set specification> identifies the character repertoire used for expressing the names of schema objects used in the <SQL-server module definition>. The <default schema name> specified by the <SQL-server module schema clause> identifies the schema name used for implicit qualification of unqualified names appearing in the <SQL-server module definition>. The SQL-invoked routines of an SQL-server module are invoked only from SQL-statements.

An SQL-server module has an *SQL-server module authorization identifier*, which is set to the authorization identifier of the owner of the schema that contains the SQL-server module at the time the SQL-server module is created. The SQL-server module authorization identifier acts as the current authorization identifier for privilege determination for the SQL objects, if any, contained in the SQL-server module.

An SQL-server module is described by an SQL-server module descriptor. An SQL-server module descriptor includes:

— The SQL-server module name of the SQL-server module.

— The descriptor of the character set in which the SQL-server module is represented.

— The default schema name used for implicit qualification of unqualified names in the SQL-server module.

— The SQL-server module authorization identifier of the SQL-server module.

— The list of schema names contained in the <SQL-server module path specification>.

— The table descriptor of every local temporary table declared in the SQL-server module.

— The descriptor of every SQL-invoked routine contained in the SQL-server module.

— The text of the <SQL-server module definition>.

4.2 Tables

This Subclause modifies Subclause 4.14, "Tables", in ISO/IEC 9075-2.

4.2.1 Types of tables

This Subclause modifies Subclause 4.14.2, "Types of tables", in ISO/IEC 9075-2.

Insert this paragraph A declared local temporary table may be declared in an SQL-server module.

Insert this paragraph A declared local temporary table that is declared in an SQL-server module is a named table defined by a <temporary table declaration> that is effectively materialized the first time any <module routine> in the <SQL-server module definition> that contains the <temporary table declaration> is executed.

Insert this paragraph A declared local temporary table is accessible only by <module routine>s in the <SQL-server module definition> that contains the <temporary table declaration>. The effective <schema name> of the <schema qualified name> of the declared local temporary table may be thought of as the implementation-dependent SQL-session identifier associated with the SQL-session and the name of the <SQL-server module definition> that contains the <temporary table declaration>.

4.3 SQL-invoked routines

This Subclause modifies Subclause 4.27, "SQL-invoked routines", in ISO/IEC 9075-2.

4.3.1 Overview of SQL-invoked routines

This Subclause modifies Subclause 4.27.1, "Overview of SQL-invoked routines", in ISO/IEC 9075-2.

Replace 2nd paragraph An SQL-invoked routine is either a component of an <SQL-server module definition> or an element of an SQL-schema. An SQL-invoked routine that is an element of an SQL-schema is called a schema-level routine.

4.3.2 Execution of SQL-invoked routines

This Subclause modifies Subclause 4.27.3, "Execution of SQL-invoked routines", in ISO/IEC 9075-2.

Replace 5th 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.

Insert in the 7th 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.

4.4 SQL-schemas

This Subclause modifies Subclause 4.20, "SQL-schemas", in ISO/IEC 9075-2.

Insert into the 2nd paragraph

— An SQL-server module descriptor.

4.5 SQL-paths

This Subclause modifies Subclause 4.28, "SQL-paths", in ISO/IEC 9075-2.

Replace 3rd paragraph The value specified by CURRENT_PATH is the value of the SQL-path of the current SQL-session. This SQL-path is used to search for the subject routine of a <routine invocation> whose <routine name> does not contain a <schema name> when the <routine invocation> is contained in <preparable statement>s that are prepared in the current SQL-session by either an <execute immediate statement> or a <prepare statement>, or contained in <direct SQL statement>s that are invoked directly. The definition of SQL-schemas and SQL-server modules specify an SQL-path that is used to search for the subject routine of a <routine invocation> whose <routine name>s do not contain a <schema name> when the <routine invocation> is contained respectively in the <schema definition> or the <SQL-server module definition>.

4.6 Host parameters

This Subclause modifies Subclause 4.29, "Host parameters", in ISO/IEC 9075-2.

4.6.1 Status parameters

This Subclause modifies Subclause 4.29.2, "Status parameters", in ISO/IEC 9075-2.

Insert this paragraph Exception conditions or completion conditions may be raised during the execution of an <SQL procedure statement>. One of the conditions becomes the active condition when the <SQL procedure statement> terminates; the *active condition* is the condition returned in SQLSTATE. If the active condition is an exception condition, then it is called the *active exception condition*. If the active condition is a completion condition, then it is called the *active completion condition*.

Insert this paragraph If the <SQL procedure statement> is a <compound statement>, then the active condition may result from the action of some exception handler specified in the <compound statement>.

4.7 Diagnostics area

This Subclause modifies Subclause 4.30, "Diagnostics area", in ISO/IEC 9075-2.

Insert this paragraph Information about a completion or exception condition 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

This Subclause modifies Subclause 4.32, "Cursors", in ISO/IEC 9075-2.

4.8.1 General description of cursors

This Subclause modifies Subclause 4.32.1, "General description of cursors", in ISO/IEC 9075-2.

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

4.9 Condition handling

Condition handling is the method of handling exception and completion conditions in SQL/PSM. Condition handling provides a <*handler declaration*> to define a handler, specifying its type, the exception and completion conditions it can resolve, and the action it takes to do so. Condition handling also provides the ability to explicitly signal exception and completion conditions.

<handler declaration>s specify the handling of exception and completion conditions. <handler declaration>s are optionally specified in <compound statement>s. The scope of a <handler declaration> specified in <compound statement> *CS* is *CS*, excluding every <SQL schema statement> contained in *CS*.

A <handler declaration> associates one or more conditions with a handler action. The handler action is an <SQL procedure statement>.

A *general* <*handler declaration*> is one that is associated with the <condition value>s SQLEXCEPTION, SQLWARNING, or NOT FOUND. All other <handler declaration>s are *specific* <*handler declaration*>s.

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>.

A <condition declaration> is used to declare a <condition name>, and to optionally associate it with an SQL-STATE value. If a <condition declaration> does not specify an SQLSTATE value, it declares a *user-defined exception condition*. <condition name>s can be used in <handler declaration>s, <signal statement>s, and <resignal statement>s.

When the <compound statement> containing a <handler declaration> is executed, a handler is created for the conditions associated with that <handler declaration>. A created handler is *activated* when it is the most appropriate handler for an exception or completion condition that has been raised by an SQL-statement. Such a handler is an *active* handler.

The *most appropriate* handler is determined during execution of an implicit or explicit <resignal statement>. An implicit <resignal statement> is executed when a <compound statement> or <handler action> completes with a condition other than *successful completion*.

If there is no most appropriate handler and the condition is an exception condition, then the SQL-statement raising the exception condition is terminated with that exception condition. This type of exception condition is called an *unhandled exception condition*. Unhandled exception conditions are examined at the next visible scope for handling. If an exception condition remains unhandled at the outermost <externally-invoked procedure> or <direct SQL statement>, it is seen by the SQL-client. Even if the SQL-client resolves the exception condition, execution is not resumed in the SQL-server where the exception condition was raised.

If there is no most appropriate handler and the condition is a completion condition, then execution is resumed as specified in Subclause 6.3.3.7, "Exceptions", in ISO/IEC 9075-1. This type of completion condition is called an *unhandled completion condition*.

A handler type specifies CONTINUE, EXIT, or UNDO.

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.

— Pop the diagnostics area stack.

— Execute the handler action.

— 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.

If a <handler action> completes with a completion condition: *successful completion*, then it was able to resolve the condition, and execution resumes as specified in Subclause 13.2, "<handler declaration>".

If a <handler action> completes with an exception or completion condition other than *successful completion*, then an implicit <resignal statement> is executed. The <resignal statement> determines whether there is another <handler declaration> that can resolve the condition.

4.10 SQL-statements

This Subclause modifies Subclause 4.33, "SQL-statements", in ISO/IEC 9075-2.

4.10.1 Classes of SQL-statements

This Subclause modifies Subclause 4.33.1, "Classes of SQL-statements", in ISO/IEC 9075-2.

Insert this paragraph The following are additional main classes of SQL-statements:

— SQL-control declarations

4.10.2 SQL-statements classified by function

This Subclause modifies Subclause 4.33.2, "SQL-statements classified by function", in ISO/IEC 9075-2.

4.10.2.1 SQL-schema statements

This Subclause modifies Subclause 4.33.2.1, "SQL-schema statements", in ISO/IEC 9075-2.

Insert this paragraph The following are additional SQL-schema statements:

— <SQL-server module definition>

— <drop module statement>

4.10.2.2 SQL-control statements

This Subclause modifies Subclause 4.33.2.6, "SQL-control statements", in ISO/IEC 9075-2.

The following are additional SQL-control statements:

— <compound statement>

— <case statement>

— <if statement>

— <iterate statement>

— <leave statement>

— <loop statement>

— <while statement>

— <repeat statement>

— <for statement>

— <assignment statement>

4.10.2.3 SQL-control declarations

Insert this paragraph The following are the SQL-control declarations:

— <condition declaration>

— <handler declaration>

— <SQL variable declaration>

4.10.2.4 SQL-diagnostics statements

This Subclause modifies Subclause 4.33.2.8, "SQL-diagnostics statements", in ISO/IEC 9075-2.

Insert this paragraph The following are additional SQL-diagnostics statements:

— <signal statement>

— <resignal statement>

4.10.3 Embeddable SQL-statements

This Subclause modifies Subclause 4.33.6, "Embeddable SQL-statements", in ISO/IEC 9075-2.

Insert this paragraph The following are additional SQL-statements that are embeddable in an <embedded SQL host program> and that may be the <SQL procedure statement> in an <externally-invoked procedure> in an SQL-client module:

— All SQL-control statements

NOTE 1 — SQL-control declarations contained in (for example) <compound statement>s are permitted, even when the containing SQL-statement is embedded in an <embedded SQL host program>.

4.10.4 Preparable and immediately executable SQL-statements

This Subclause modifies Subclause 4.33.7, "Preparable and immediately executable SQL-statements", in ISO/IEC 9075-2.

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>

4.10.5 Directly executable SQL-statements

This Subclause modifies Subclause 4.33.8, "Directly executable SQL-statements", in ISO/IEC 9075-2.

Insert this paragraph The following are additional SQL-statements that may be executed directly:

— All SQL-control statements

4.10.6 Iterated SQL-statements

The following are the iterated SQL-statements:

— <loop statement>

— <while statement>

— <repeat statement>

— <for statement>

4.10.7 SQL-statements and transaction states

This Subclause modifies Subclause 4.33.4, "SQL-statements and transaction states", in ISO/IEC 9075-2.

Insert this paragraph The following additional SQL-statement is a transaction-initiating SQL-statement:

— <for statement>

Insert this paragraph The following additional SQL-statement is not a transaction-initiating SQL-statement:

— <iterate statement>

— <leave statement>

Insert this paragraph The following additional SQL-statements are possibly transaction-initiating SQL-statements:

— SQL-control statements other than:

- <for statement>
- <iterate statement>
- <leave statement>

4.10.8 Compound statements

A compound statement allows a sequence of SQL-statements to be considered as a single SQL-statement. A compound statement also defines a local scope in which SQL-variables, condition handlers, and cursors can be declared. See Subclause 13.1, "<compound statement>".

4.10.9 SQL-statement atomicity and statement execution contexts

This Subclause modifies Subclause 4.33.5, "SQL-statement atomicity and statement execution contexts", in ISO/IEC 9075-2.

Add to the list of non-atomic SQL-statements

— <assignment statement>.

— <case statement>.

— <compound statement>, unless BEGIN ATOMIC is specified.

— <for statement>.

— <if statement>.

— <loop statement>.

— <repeat statement>.

— <while statement>.

4.11 Basic security model

This Subclause modifies Subclause 4.34, "Basic security model", in ISO/IEC 9075-2.

4.11.1 Privileges

This Subclause modifies Subclause 4.34.2, "Privileges", in ISO/IEC 9075-2.

Insert this paragraph A privilege further authorizes a given category of <action> to be performed on a specified SQL-server module by a specified <authorization identifier>.

Insert this paragraph An *execute privilege descriptor* may also identify the existence of a privilege on the SQL-server module identified by the privilege descriptor.

Insert this paragraph The identification included in an EXECUTE privilege descriptor may also identify the SQL-server module described by the descriptor.

Insert this paragraph Individual SQL-invoked routines contained in an SQL-server module cannot be associated with EXECUTE privilege descriptors. Only schema-level routines and SQL-server modules are associated with EXECUTE privilege descriptors.

NOTE 2 — "schema-level routine" is defined in Subclause 11.50, "<SQL-invoked routine>", in ISO/IEC 9075-2.

4.12 SQL-sessions

This Subclause modifies Subclause 4.37, "SQL-sessions", in ISO/IEC 9075-2.

4.12.1 General description of SQL-sessions

This Subclause modifies Subclause 4.37.1, "General description of SQL-sessions", in ISO/IEC 9075-2.

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.

This page intentionally left blank.

5 Lexical elements

This Clause modifies Clause 5, "Lexical elements", in ISO/IEC 9075-2.

5.1 <token> and <separator>

This Subclause modifies Subclause 5.2, "<token> and <separator>", in ISO/IEC 9075-2.

Function

Specify lexical units (tokens and separators) that participate in SQL language.

Format

```
<non-reserved word> ::=
    !! All alternatives from ISO/IEC 9075-2
  | CONDITION_IDENTIFIER

  | EXIT

  | STACKED

  | UNDO

<reserved word> ::=
    !! All alternatives from ISO/IEC 9075-2
  | DO

  | ELSEIF

  | HANDLER

  | IF | ITERATE

  | LEAVE | LOOP

  | REPEAT | RESIGNAL

  | SIGNAL

  | UNTIL

  | WHILE
```

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

5.2 Names and identifiers

This Subclause modifies Subclause 5.4, "Names and identifiers", in ISO/IEC 9075-2.

Function

Specify names.

Format

```
<SQL-server module name> ::= <schema qualified name>

<SQL variable name> ::= <identifier>

<condition name> ::= <identifier>
```

Syntax Rules

1) Replace SR 4)a) If the <local or schema qualified name> is contained, without an intervening <schema definition> or <SQL-server module 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.

2) Insert before SR 4)b) If the <local or schema qualified name> is contained in an <SQL-server module definition> without an intervening <schema definition>, then the <default schema name> that is specified or implicit in the <SQL-server module definition> is implicit.

3) Replace SR 13)b)i) If the <schema qualified name> is contained, without an intervening <schema definition> or <SQL-server module 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.

4) Insert before SR 13)b)ii) If the <schema qualified name> is contained in an <SQL-server module definition> without an intervening <schema definition>, then the <default schema name> that is specified or implicit in the <SQL-server module definition> is implicit.

5) Replace SR 8) If <user-defined type name> *UDTN* with a <qualified identifier> *QI* is specified, then

   Case:

   a) If *UDTN* is simply contained in <path-resolved user-defined type name>, then

      Case:

      i) If *UDTN* contains a <schema name> *SN*, then the schema identified by *SN* shall contain the descriptor of a user-defined type *UDT* such that the <qualified identifier> of *UDT* is equivalent to *QI*. *UDT* is the user-defined type identified by *UDTN*.

      ii) Otherwise:

         1) Case:

A) If *UDTN* is contained, without an intervening <schema definition> or <SQL-server module definition>, in a <preparable statement> that is prepared in the current SQL-session by an <execute immediate statement> or by a <prepare statement> or in a <direct SQL statement> that is invoked directly, then let *DP* be the SQL-path of the current SQL-session.

B) If *UDTN* is contained in an <SQL-server module definition> without in intervening <schema definition>, then let *DP* be the SQL-path of that <SQL-server module definition>.

C) If *UDTN* is contained in a <schema definition> that is not contained in an <SQL-client module definition>, then let *DP* be the SQL-path of that <schema definition>.

D) Otherwise, *UDTN* is contained in an <SQL-client module definition>; let *DP* be the SQL-path of that <SQL-client module definition>.

2) Let *N* be the number of <schema name>s in *DP*. Let S_i, 1 (one) ≤ *i* ≤ *N*, be the *i*-th <schema name> in *DP*.

3) Let the *set of subject types* be the set containing every user-defined type *T* in the schema identified by some S_i, 1 (one) ≤ *i* ≤ *N*, such that the <qualified identifier> of *T* is equivalent to *QI*. There shall be at least one type in the set of subject types.

4) Let *UDT* be the user-defined type contained in the set of subject types such that there is no other type *UDT2* for which the <schema name> of the schema that includes the user-defined type descriptor of *UDT2* precedes in *DP* the <schema name> identifying the schema that includes the user-defined type descriptor of *UDT*. *UDTN* identifies *UDT*.

5) The implicit <schema name> of *UDTN* is the <schema name> of the schema that includes the user-defined type descriptor of *UDT*.

b) If *UDTN* is simply contained in <schema-resolved user-defined type name>, then

Case:

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

ii) If *UDTN* is contained in an <SQL-server module definition> without in intervening <schema definition>, then the implicit <schema name> of *UDTN* is the <schema name> that is specified or implicit in <SQL-server module definition>.

iii) If *UDTN* is contained in a <schema definition> that is not contained in an <SQL-client module definition>, then the implicit <schema name> of *UDTN* is the <schema name> that is specified or implicit in <schema definition>.

iv) Otherwise, *UDTN* is contained in an <SQL-client module definition>; the implicit <schema name> of *UDTN* is the <schema name> that is specified or implicit in <SQL-client module definition>.

Access Rules

No additional Access Rules.

General Rules

1) Insert this GR An <SQL-server module name> identifies an SQL-server module.

2) Insert this GR An <SQL variable name> identifies an SQL variable.

3) Insert this GR A <condition name> identifies an exception condition or a completion condition and optionally a corresponding SQLSTATE value.

Conformance Rules

No additional Conformance Rules.

This page intentionally left blank.

6 Scalar expressions

This Clause modifies Clause 6, "Scalar expressions", in ISO/IEC 9075-2.

6.1 <value specification> and <target specification>

This Subclause modifies Subclause 6.4, "<value specification> and <target specification>", in ISO/IEC 9075-2.

Function

Specify one or more values, host parameters, SQL parameters, dynamic parameters, host variables, or SQL variables.

Format

```
<general value specification> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <SQL variable reference>

<simple value specification> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <SQL variable reference>

<target specification> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <SQL variable reference>

<simple target specification> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <SQL variable reference>

<target array reference> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <SQL variable reference>
```

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

6.2 <identifier chain>

This Subclause modifies Subclause 6.6, "<identifier chain>", in ISO/IEC 9075-2.

Function

Disambiguate a <period>-separated chain of identifiers.

Format

No additional Format items.

Syntax Rules

1) Insert before SR 8) An SQL variable *V* is said to be *refinable* if the declared type of *V* is a row type or a structured type.

2) Replace introductory paragraph of SR 8) For at most one *j* between 1 (one) and *M*, PIC_j is called the *basis* of *IC*, and *j* is called the *basis length* of *IC*. The *referent* of the basis is a column *C* of a table, an SQL parameter *SP*, or an SQL variable *SV*. The basis, basis length, basis scope and basis referent of *IC* are determined as follows:

3) Replace SR 8)a)ii) Otherwise, *IC* shall be contained within the scope of one or more range variables whose associated tables include a column whose <column name> is equivalent to I_1 or within the scope of a <routine name> whose associated <SQL parameter declaration list> includes an SQL parameter whose <SQL parameter name> 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 range variables, <routine name>s, and <beginning label>s.

4) Insert after SR 8)a)ii)1)B) If *IPST* is a <beginning label>, then let *SV* be the SQL variable whose <SQL variable name> 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*.

5) Insert before SR 8)b)iv) If *IC* is contained in the scope of a <beginning label> whose associated <local declaration list> includes an SQL variable *SV* whose <SQL variable name> is equivalent to I_1, then PIC_1 is a candidate basis of *IC*, the scope of PIC_1 is the scope of *SV*, and the referent of PIC_1 is *SV*.

6) Insert before SR 8)b)iv) If $N = 2$ and PIC_1 is equivalent to a <beginning label> *BL* whose scope contains *IC* and whose associated <local declaration list> includes an SQL variable *SV* whose <SQL variable name> is equivalent to I_2, then PIC_2 is a candidate basis of *IC*, the scope of PIC_2 is the scope of *SV*, and the referent of PIC_2 is *SV*.

7) Insert before SR 8)b)iv) If $N > 2$ and PIC_1 is equivalent to a <beginning label> *BL* whose scope contains *IC* and whose associated <local declaration list> includes a refinable SQL variable *SV* whose <SQL variable name> is equivalent to I_2, then PIC_2 is a candidate basis of *IC*, the scope of PIC_2 is the scope of *SV*, and the referent of PIC_2 is *SV*.

8) Replace SR 10) If *BL* < *N*, then let *TIC* be the <value expression primary>:

```
( PIC_BL ) <period> I_BL+1 <period> ... <period> I_N
```

The Syntax Rules of Subclause 6.25, "<value expression>", are applied to *TIC*, yielding a column reference, an SQL parameter reference, or an SQL variable reference, and (*N*–*BL*) <field reference>s, <method invocation>s, <modified field reference>s, and/or <mutator reference>s.

NOTE 3 — In this transformation, (PIC_{BL}) is interpreted as a <value expression primary> of the form <left paren> <value expression> <right paren>. PIC_{BL} is a <value expression> that is a <value expression primary> that is an <unsigned value specification> that is either a <column reference> or an <SQL parameter reference>. The identifiers I_{BL+1}, ..., I_N are parsed using the Syntax Rules of <field reference> and <method invocation>. Alternatively, on the left-hand side of an <assignment statement>, (PIC_{BL}) is interpreted as "<left paren> <target specification> <right paren>", and the identifiers I_{BL+1}, ..., I_N are parsed using the Syntax Rules of <modified field reference> and <mutator reference>.

9) Insert after SR 13) A <basic identifier chain> whose basis referent is an SQL variable is an *SQL variable reference*.

Access Rules

None.

General Rules

1) Insert this GR If *BIC* is an SQL variable reference, then *BIC* references the SQL variable *SV* of a given execution of the <compound statement> whose <local declaration list> contains the <SQL variable declaration> that declares *SV*.

Conformance Rules

1) Without Feature P005, "Qualified SQL variable references", conforming SQL language shall not contain an SQL variable reference whose first <identifier> is the <beginning label> of a <compound statement>.

6.3 <next value expression>

This Subclause modifies Subclause 6.13, "<next value expression>", in ISO/IEC 9075-2.

Function

Return the next value of a sequence generator.

Format

No additional Format items.

Syntax Rules

1) Insert after SR 1)e) An <assignment statement>.

2) Insert this SR <next value expression> shall not be contained without an intervening <SQL statement list> in a <case statement>, an <if statement>, a <loop statement>, a <while statement>, or a <repeat statement>.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

6.4 <SQL variable reference>

Function

Reference an SQL variable.

Format

```
<SQL variable reference> ::= <basic identifier chain>
```

Syntax Rules

1) An <SQL variable reference> shall be a <basic identifier chain> that is an SQL variable reference.

Access Rules

None.

General Rules

None.

Conformance Rules

No additional Conformance Rules.

7 Query expressions

This Clause modifies Clause 7, "Query expressions", in ISO/IEC 9075-2.

7.1 <query specification>

This Subclause modifies Subclause 7.12, "<query specification>", in ISO/IEC 9075-2.

Function

Specify a table derived from the result of a <table expression>.

Format

No additional Format items.

Syntax Rules

1) Insert after SR 7)f)i) If *IC* is contained in the scope of a <beginning label> whose associated <local declaration list> includes an SQL variable *SV* whose <SQL variable name> is equivalent to I_1, then PIC_1 is a candidate basis of *IC* and the scope of PIC_1 is the scope of *SV*.

2) Insert after SR 7)f)i) If $N = 2$ and PIC_1 is equivalent to a <beginning label> *BL* whose scope contains *IC* and whose associated <local declaration list> includes an SQL variable *SV* of row type whose <SQL variable name> is equivalent to *I2*, then PIC_2 is a candidate basis of *IC*, the scope of PIC_2 is the scope of *SV*, and the referent of PIC_2 is *SV*.

3) Insert after SR 7)f)i) If $N > 2$ and PIC_1 is equivalent to a <beginning label> *BL* whose scope contains *IC* and whose associated <local declaration list> includes a refinable SQL variable *SV* whose <SQL variable name> is equivalent to *I2*, then PIC_2 is a candidate basis of *IC*, the scope of PIC_2 is the scope of *SV*, and the referent of PIC_2 is *SV*.

4) Insert after SR 18)a)iv) An SQL variable.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

1) Without Feature P005, "Qualified SQL variable references", conforming SQL language shall not contain an <asterisked identifier chain> whose first <identifier> is the <beginning label> of a <compound statement>.

8 Additional common elements

This Clause modifies Clause 10, "Additional common elements", in ISO/IEC 9075-2.

8.1 <routine invocation>

This Subclause modifies Subclause 10.4, "<routine invocation>", in ISO/IEC 9075-2.

Function

Invoke an SQL-invoked routine.

Format

No additional Format items.

Syntax Rules

1) Replace SR 5) An SQL-invoked routine *R* is an *executable routine* if and only if *R* is a possibly candidate routine and

   Case:

   a) If *RI* is contained in an <SQL schema statement>, then

      Case:

      i) If *RI* is contained in an <SQL-server module definition> *M*, then the applicable privileges for the <authorization identifier> that owns the containing schema include EXECUTE on *M*.

      ii) Otherwise, the applicable privileges for the <authorization identifier> that owns the containing schema include EXECUTE on *R*.

   b) Otherwise,

      Case:

      i) If *RI* is contained in an <SQL-server module definition> *M*, then the current privileges include EXECUTE on *M*.

      ii) Otherwise, the current privileges include EXECUTE on *R*.

2) Insert before SR 7)b)i)1)C)I) If *RI* is contained in an <SQL-server module definition>, then let *DP* be the SQL-path of that <SQL-server module definition>.

3) Replace SR 7)b)i)1)C)I) If *RI* is contained in a <schema definition> without an intervening <SQL-server module definition>, then let *DP* be the SQL-path of that <schema definition>.

4) Insert before SR 8)b)i)1)C)I) If *RI* is contained in an <SQL-server module definition>, then let *DP* be the SQL-path of that <SQL-server module definition>.

5) Replace SR 8)b)i)1)C)I) If *RI* is contained in a <schema definition> without an intervening <SQL-server module definition>, then let *DP* be the SQL-path of that <schema definition>.

6) Replace SR 8)c)i)4)C) If A_i is either an <SQL variable reference>, an <SQL parameter reference>, a <column reference>, or a <target array element specification>, 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 4 — The <column reference> can only be a new transition variable column reference.

Access Rules

No additional Access Rules.

General Rules

1) Insert after GR 5)d)i)

NOTE 5 — The identities of declared local temporary tables that are defined in <SQL-server module>s are not removed.

2) Replace the introductory paragraph of GR 9)b)ii) If TS_i is either an <SQL variable reference>, an <SQL parameter reference>, a <column reference>, or a <target array element specification>, then

NOTE 6 — The <column reference> can only be a new transition variable column reference.

Conformance Rules

No additional Conformance Rules.

8.2 <sqlstate value>

Function

Specify an SQLSTATE value.

Format

```
<sqlstate value> ::= SQLSTATE [ VALUE ] <character string literal>
```

Syntax Rules

1) Let *L* be the <character string literal> contained in <sqlstate value>.

2) The implicit or explicit character set of *L* shall be the implementation-defined character set in which SQLSTATE parameter values are returned.

3) Let *V* be the character string that is the value of

```
TRIM (BOTH '' FROM L )
```

4) *V* shall comprise either:

   a) Five characters of which the first two have the form of a standard-defined class value and the last three have the form of a standard-defined subclass value.

   b) Five characters of which the first two have the form of a standard-defined class value and the last three have the form of an implementation-defined subclass value.

   c) Five characters of which the first two have the form of an implementation-defined class value and the last three have the form of either a standard-defined subclass value or an implementation-defined subclass value.

5) *V* shall not be the SQLSTATE value for the condition *successful completion*.

6) The SQLSTATE value defined by the <sqlstate value> is *V*.

Access Rules

None.

General Rules

None.

Conformance Rules

None.

This page intentionally left blank.

9 Schema definition and manipulation

This Clause modifies Clause 11, "Schema definition and manipulation", in ISO/IEC 9075-2.

9.1 <schema definition>

This Subclause modifies Subclause 11.1, "<schema definition>", in ISO/IEC 9075-2.

Function

Define a schema.

Format

```
<schema element> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <SQL-server module definition>
```

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

9.2 <drop schema statement>

This Subclause modifies Subclause 11.2, "<drop schema statement>", in ISO/IEC 9075-2.

Function

Destroy a schema.

Format

No additional Format items.

Syntax Rules

1) Insert this SR If RESTRICT is specified, then *S* shall not include any SQL-server modules.

Access Rules

No additional Access Rules.

General Rules

1) Insert before GR 8) For every SQL-server module *M* contained in *S*, let *MN* be the <SQL-server module name> of *M*. For every *M*, the following <drop module statement> is effectively executed:

```
DROP MODULE MN CASCADE
```

2) Replace GR 11) Let *R* be any SQL-invoked routine whose routine descriptor contains the <schema name> of *S* in the <SQL routine body>.

Case:

a) If *R* is included in an SQL-server module *M*, then let *MN* be the <SQL-server module name> of *M*. The following <drop module statement> is effectively executed without further Access Rule checking:

```
DROP MODULE MN CASCADE
```

b) Otherwise, let *SN* be the <specific name> of *R*. The following <drop routine statement> is effectively executed without further Access Rule checking:

```
DROP SPECIFIC ROUTINE SN CASCADE
```

3) Insert after GR 11) Let *SSM* be any SQL-server module whose module descriptor includes the <schema name> of *S* and let *MN* be the <SQL-server module name> of *SSM*. The following <drop module statement> is effectively executed without further Access Rule checking:

```
DROP MODULE MN CASCADE
```


Conformance Rules

No additional Conformance Rules.

9.3 <default clause>

This Subclause modifies Subclause 11.5, "<default clause>", in ISO/IEC 9075-2.

Function

Specify the default for a column, domain, or SQL variable.

Format

No additional Format items.

Syntax Rules

1) Replace SR 1) The subject data type of a <default clause> is the data type specified in the descriptor identified by the containing <column definition>, <domain definition>, <attribute definition>, <alter column definition>, or <alter domain statement>, or that defined by the <data type> specified in the containing <SQL variable declaration>.

Access Rules

No additional Access Rules.

General Rules

None.

Conformance Rules

No additional Conformance Rules.

9.4 <drop column scope clause>

This Subclause modifies Subclause 11.16, "<drop column scope clause>", in ISO/IEC 9075-2.

Function

Drop the scope from an existing column of data type REF in a base table.

Format

No additional Format items.

Syntax Rules

1) Insert after SR 5)d) The module descriptor of any SQL-server module.

Access Rules

None.

General Rules

1) Replace GR 1) For every SQL-invoked routine *R* whose routine descriptor includes an <SQL routine body> that contains an impacted dereference operation,

   Case:

   a) If *R* is included in an SQL-server module *M*, then let *MN* be the <SQL-server module name> of *M*. The following <drop module statement> is effectively executed without further Access Rule checking:

      ```
      DROP MODULE MN CASCADE
      ```

   b) Otherwise, let *SN* be the <specific name> of *R*. The following <drop routine statement> is effectively executed for every *R* without further Access Rule checking:

      ```
      DROP SPECIFIC ROUTINE SN CASCADE
      ```

2) Insert after GR 4) Let *SSM* be any SQL-server module whose module descriptor includes an impacted dereference operation, and let *MN* be the <SQL-server module name> of *SSM*. The following <drop module statement> is effectively executed without further Access Rule checking:

   ```
   DROP MODULE MN CASCADE
   ```

Conformance Rules

No additional Conformance Rules.

9.5 <drop column definition>

This Subclause modifies Subclause 11.18, "<drop column definition>", in ISO/IEC 9075-2.

Function

Destroy a column of a base table.

Format

No additional Format items.

Syntax Rules

1) Replace SR 5) If RESTRICT is specified, then *C* shall not be referenced in any of the following:

   a) The <query expression> of any view descriptor.

   b) The <search condition> of any constraint descriptor other than a table constraint descriptor that contains references to no other column and that is included in the table descriptor of *T*.

   c) The <SQL routine body> of any routine descriptor.

   d) Either an explicit trigger column list or a triggered action column set of any trigger descriptor.

   e) The module descriptor of any SQL-server module.

   NOTE 7 — A <drop column definition> that does not specify CASCADE will fail if there are any references to that column resulting from the use of CORRESPONDING, NATURAL, SELECT * (except where contained in an exists predicate>), or REFERENCES without a <reference column list> in its <referenced table and columns>.

   NOTE 8 — If CASCADE is specified, then any such dependent object will be dropped by the execution of the <revoke statement> specified in the General Rules of this Subclause.

   NOTE 9 — *CN* may be contained in an implicit trigger column list of a trigger descriptor.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

9.6 <drop table constraint definition>

This Subclause modifies Subclause 11.20, "<drop table constraint definition>", in ISO/IEC 9075-2.

Function

Destroy a constraint on a table.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

1) Replace GR 2) Let *R* be any SQL-invoked routine whose routine descriptor contains the <constraint name> of *TC* in the <SQL routine body>.

   Case:

   a) If *R* is included in an SQL-server module *M*, then let *MN* be the <SQL-server module name> of *M*. The following <drop module statement> is effectively executed without further Access Rule checking:

      ```
      DROP MODULE MN CASCADE
      ```

   b) Otherwise, let *SN* be the <specific name> of *R*. The following <drop routine statement> is effectively executed without further Access Rule checking:

      ```
      DROP SPECIFIC ROUTINE SN CASCADE
      ```

Conformance Rules

No additional Conformance Rules.

9.7 <drop table statement>

This Subclause modifies Subclause 11.21, "<drop table statement>", in ISO/IEC 9075-2.

Function

Destroy a table.

Format

No additional Format items.

Syntax Rules

1) Insert after SR 7) If RESTRICT is specified, then *T* shall not be referenced in the module descriptor of any SQL-server module.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

9.8 <view definition>

This Subclause modifies Subclause 11.22, "<view definition>", in ISO/IEC 9075-2.

Function

Define a viewed table.

Format

No additional Format items.

Syntax Rules

1) Insert after SR 2)

NOTE 10 — <SQL variable name> is also excluded because of the scoping rules for <SQL variable name>.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

9.9 <drop view statement>

This Subclause modifies Subclause 11.23, "<drop view statement>", in ISO/IEC 9075-2.

Function

Destroy a view.

Format

```
No additional Format items.
```

Syntax Rules

1) Insert after SR 5) If RESTRICT is specified, then *V* shall not be referenced in the module descriptor of any SQL-server module.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

9.10 <drop domain statement>

This Subclause modifies Subclause 11.30, "<drop domain statement>", in ISO/IEC 9075-2.

Function

Destroy a domain.

Format

No additional Format items.

Syntax Rules

1) Insert after SR 2) If RESTRICT is specified, then *D* shall not be referenced in the module descriptor of any SQL-server module.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

9.11 <drop character set statement>

This Subclause modifies Subclause 11.32, "<drop character set statement>", in ISO/IEC 9075-2.

Function

Destroy a character set.

Format

No additional Format items.

Syntax Rules

1) Insert after SR 4) *C* shall not be referenced in the module descriptor of any SQL-server module.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

9.12 <drop collation statement>

This Subclause modifies Subclause 11.34, "<drop collation statement>", in ISO/IEC 9075-2.

Function

Destroy a collation.

Format

No additional Format items.

Syntax Rules

1) Insert after SR 4) If RESTRICT is specified, then *C* shall not be referenced in the module descriptor of any SQL-server module.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

9.13 <drop transliteration statement>

This Subclause modifies Subclause 11.36, "<drop transliteration statement>", in ISO/IEC 9075-2.

Function

Destroy a character transliteration.

Format

```
No additional Format items.
```

Syntax Rules

1) Insert after SR 3) If RESTRICT is specified, then *C* shall not be referenced in the module descriptor of any SQL-server module.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

9.14 <assertion definition>

This Subclause modifies Subclause 11.37, "<assertion definition>", in ISO/IEC 9075-2.

Function

Specify an integrity constraint.

Format

No additional Format items.

Syntax Rules

1) Insert after SR 4)

NOTE 11 — <SQL variable name> is also excluded because of the scoping rules for <SQL variable name>.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

9.15 <drop assertion statement>

This Subclause modifies Subclause 11.38, "<drop assertion statement>", in ISO/IEC 9075-2.

Function

Destroy an assertion.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

1) Replace GR 1) Let *R* be any SQL-invoked routine whose routine descriptor contains the <constraint name> of *A* in the <SQL routine body>.

   Case:

   a) If *R* is included in an SQL-server module *M*, then let *MN* be the <SQL-server module name> of *M*. The following <drop module statement> is effectively executed without further Access Rule checking:

      ```
      DROP MODULE MN CASCADE
      ```

   b) Otherwise, let *SN* be the <specific name> of *R*. The following <drop routine statement> is effectively executed without further Access Rule checking:

      ```
      DROP SPECIFIC ROUTINE SN CASCADE
      ```

Conformance Rules

No additional Conformance Rules.

9.16 <trigger definition>

This Subclause modifies Subclause 11.39, "<trigger definition>", in ISO/IEC 9075-2.

Function

Defined triggered SQL-statements.

Format

```
<triggered SQL statement> ::= <SQL procedure statement>
```

NOTE 12 — The preceding production defining <triggered SQL statement> completely supersedes the definition in ISO/IEC 9075-2.

Syntax Rules

1) Insert this SR If <SQL procedure statement> simply contains a <compound statement> *CS*, then *CS* shall specify ATOMIC.

Access Rules

1) Replace If the <triggered action> *TA* of a <trigger definition> contains an <old transition table name> *OTTN*, an <old transition variable name> *OTVN*, a <new transition table name> *NTTN*, or a <new transition variable name> *NTVN*, then:

   a) If *TA* contains *OTTN*, *OTVN*, or *NTTN*, or if *TA* contains *NTVN* other than as an <assignment target> of an <assignment statement>, then the applicable privileges for *TA* shall include SELECT.

   b) If *TA* contains *NTVN* as an <assignment target> of an <assignment statement>, then the applicable privileges for *TA* shall include UPDATE.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

9.17 <drop user-defined ordering statement>

This Subclause modifies Subclause 11.56, "<drop user-defined ordering statement>", in ISO/IEC 9075-2.

Function

Destroy a user-defined ordering method.

Format

No additional Format items.

Syntax Rules

1) Insert after SR 4)d) The module descriptor of any SQL-server module.

Access Rules

No additional Access Rules.

General Rules

1) Replace GR 1) Let *R* be any SQL-invoked routine that contains *P* in its <SQL routine body>.

   Case:

   a) If *R* is included in an SQL-server module *M* with no intervening <schema definition>, then let *MN* be the <SQL-server module name> of *M*. The following <drop module statement> is effectively executed without further Access Rule checking:

      ```
      DROP MODULE MN CASCADE
      ```

   b) Otherwise, let *SN* be the specific name of *R*. The following <drop routine statement> is effectively executed without further Access Rule checking:

      ```
      DROP SPECIFIC ROUTINE SN CASCADE
      ```

2) Insert after GR 6) Let *SSM* be any SQL-server module whose module descriptor contains *P* and let *MN* be the <SQL-server module name> of *SSM*. The following <drop module statement> is effectively executed without further Access Rule checking:

   ```
   DROP MODULE MN CASCADE
   ```

Conformance Rules

No additional Conformance Rules.

9.18 <SQL-server module definition>

Function

Define an SQL-server module.

Format

```
<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

<SQL-server module character set specification> ::= NAMES ARE <character set specification>

<SQL-server module schema clause> ::= SCHEMA <default schema name>

<default schema name> ::= <schema name>

<SQL-server module path specification> ::= <path specification>

<SQL-server module contents> ::= <SQL-invoked routine> <semicolon>
```

Syntax Rules

1) If an <SQL-server module definition> is contained in a <schema definition> *SD* and the <SQL-server module name> of the <SQL-server module definition> contains a <schema name>, then that <schema name> shall be equivalent to the specified or implicit <schema name> of *SD*.

2) The schema identified by the explicit or implicit <schema name> of the <SQL-server module name> shall not include a module descriptor whose <SQL-server module name> is equivalent to the <SQL-server module name> of the containing <SQL-server module definition>.

3) The SQL-invoked routine specified by <SQL-invoked routine> shall not be a schema-level routine.

   NOTE 13 — "Schema-level routine" is defined in Subclause 11.50, "<SQL-invoked routine>", in ISO/IEC 9075-2.

4) If <SQL-server module path specification> is not specified, then an <SQL-server module path specification> containing an implementation-defined <schema name list> that includes the explicit or implicit <schema name> of the <SQL-server module name> is implicit.

5) The explicit or implicit <catalog name> of each <schema name> contained in the <schema name list> of the <SQL-server module path specification> shall be equivalent to the <catalog name> of the explicit or implicit <schema name> of the <SQL-server module name>.

6) The <schema name list> of the explicit or implicit <SQL-server module path specification> is used as the SQL-path of the SQL-server module. The SQL-path is used to effectively qualify unqualified <routine name>s that are immediately contained in <routine invocation>s that are contained in the <SQL-server module definition>.

7) If <SQL-server module schema clause> is not specified, then an <SQL-server module schema clause> containing the <default schema name> that is equivalent to the explicit or implicit <schema name> of the <SQL-server module name> is implicit.

8) If <SQL-server module character set specification> is not specified, then an <SQL-server module character set specification> containing the <character set specification> that is equivalent to the <schema character set specification> of the schema identified by the explicit or implicit <schema name> of the <SQL-server module name> is implicit.

9) The explicit or implicit <SQL-server module character set specification> is the character set in which the SQL-server module is represented. If the SQL-server module is actually represented in a different character set, then the effects are implementation-dependent.

Access Rules

1) If an <SQL-server module definition> is contained in an <SQL-client module definition> with no intervening <schema definition>, then the enabled authorization identifiers shall include the <authorization identifier> that owns the schema identified by the implicit or explicit <schema name> of the <SQL-server module name>.

General Rules

1) An <SQL-server module definition> defines an SQL-server module.

2) A privilege descriptor is created that defines the EXECUTE privilege on the SQL-server module to the <authorization identifier> that owns the schema identified by the explicit or implicit <schema name> of the <SQL-server module name>. The grantor for the privilege descriptor is set to the special grantor value "_SYSTEM". This privilege is grantable if and only if all of the privileges necessary for the <authorization identifier> to successfully execute the <SQL procedure statement> contained in the <routine body> of every <SQL-invoked routine> contained in the <SQL-server module definition> are grantable.

   NOTE 14 — The necessary privileges include the EXECUTE privilege on every subject routine of every <routine invocation> contained in the <SQL procedure statement>.

3) An SQL-server module descriptor is created that describes the SQL-server module being defined. The SQL-server module descriptor includes:

   a) The SQL-server module name specified by the <SQL-server module name>.

   b) The descriptor of the character set specified by the <SQL-server module character set specification>.

   c) The default schema name specified by the <SQL-server module schema clause>.

   d) The SQL-server module authorization identifier that corresponds to the authorization identifier that owns the schema identified by the explicit or implicit <schema name> of the <SQL-server module name>.

   e) The list of schema names contained in the <SQL-server module path specification>.

   f) The descriptor of every local temporary table declared in the SQL-server module.

   g) The descriptor of every SQL-invoked routine contained in the SQL-server module.

h) The text of the <SQL-server module definition>.

Conformance Rules

1) Without Feature P001, “Stored modules”, conforming SQL language shall not contain an <SQL-server module definition>.

9.19 <drop module statement>

Function

Destroy an SQL-server module.

Format

```
<drop module statement> ::= DROP MODULE <SQL-server module name> <drop behavior>
```

Syntax Rules

1) Let *MN* be the <SQL-server module name> and let *M* be the SQL-server module identified by *MN*.

2) *M* shall be an SQL-server module.

3) If RESTRICT is specified, then the descriptor of *M* shall not include the descriptor of an SQL-invoked routine that is included in the subject routines of a <routine invocation> that is contained in any of the following:

   a) The <SQL routine body> of any routine descriptor not included in the module descriptor of *M*.

   b) The <query expression> of any view descriptor.

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

   d) Any trigger descriptor.

   e) The module descriptor of any SQL-server module other than *M*.

Access Rules

1) The enabled authorization identifiers shall include the <authorization identifier> that owns the schema identified by the <schema name> of *M*.

General Rules

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

   ```
   REVOKE EXECUTE ON MODULE MN FROM A CASCADE
   ```

2) The descriptor of *M* is destroyed.

Conformance Rules

1) Without Feature P001, "Stored modules", conforming SQL language shall not contain a <drop module statement>.

9.20 <drop data type statement>

This Subclause modifies Subclause 11.49, "<drop data type statement>", in ISO/IEC 9075-2.

Function

Destroy a user-defined type.

Format

No additional Format items.

Syntax Rules

1) Insert after SR 4)f)v) The module descriptor of any SQL-server module.

2) Insert after SR 4)h)i)4) The module descriptor of any SQL-server module.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

9.21 <SQL-invoked routine>

This Subclause modifies Subclause 11.50, "<SQL-invoked routine>", in ISO/IEC 9075-2.

Function

Define an SQL-invoked routine.

Format

```
<SQL-invoked routine> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <module routine>

<module routine> ::=
    <module procedure>
  | <module function>

<module procedure> ::= [ DECLARE ] <SQL-invoked procedure>

<module function> ::= [ DECLARE ] <SQL-invoked function>
```

Syntax Rules

1) Replace SR 6)h) Case:

   a) If an <SQL-invoked routine> is contained in an <SQL-server module definition>, and <language clause> is not specified, then a <language clause> that is equivalent to the <language clause> of the <SQL-server module definition> is implicit.

   b) If an <SQL-invoked routine> is not contained in an <SQL-server module definition> and <language clause> is not specified, then LANGUAGE SQL is implicit.

2) Replace SR 6)q) If <SQL-invoked routine> is contained in a <schema definition> without an intervening <SQL-server module definition> and *RN* contains a <schema name> *SN*, then *SN* shall be equivalent to the specified or implicit <schema name> of the containing <schema definition>. Let *S* be the SQL-schema identified by *SN*.

3) Insert after SR 6)q) If <SQL-invoked routine> is contained in an <SQL-server module definition> and if *RN* contains a <schema name> *SN*, then *SN* shall be equivalent to the specified <schema name> of the containing <SQL-server module definition>. Let *S* be the SQL-schema identified by *SN*.

Access Rules

No additional Access Rules.

General Rules

1) Insert after GR 3)v) If the SQL-invoked routine is a schema-level routine, then the schema name of the schema that includes the SQL-invoked routine; otherwise, 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 that SQL-server module.

Conformance Rules

No additional Conformance Rules.

9.22 <drop routine statement>

This Subclause modifies Subclause 11.52, "<drop routine statement>", in ISO/IEC 9075-2.

Function

Destroy an SQL-invoked routine.

Format

No additional Format items.

Syntax Rules

1) Insert after SR 5)a)iv) The module descriptor of any SQL-server module.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

9.23 <drop user-defined cast statement>

This Subclause modifies Subclause 11.54, "<drop user-defined cast statement>", in ISO/IEC 9075-2.

Function

Destroy a user-defined cast.

Format

No additional Format items.

Syntax Rules

1) Insert after SR 7)d) The module descriptor of any SQL-server module.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

This page intentionally left blank.

10 Access control

This Clause modifies Clause 12, "Access control", in ISO/IEC 9075-2.

10.1 <grant statement>

This Subclause modifies Subclause 12.1, "<grant statement>", in ISO/IEC 9075-2.

Function

Define privileges.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

1) Insert this GR For every involved grantee *G* and for every SQL-server module *M1* owned by *G*, if the applicable privileges for *G* contain all of the privileges necessary to successfully execute every <SQL procedure statement> contained in the <routine body> of every SQL-invoked routine contained in *M1* WITH GRANT OPTION, then for every privilege descriptor with a <privileges> EXECUTE, a <grantor> of "_SYSTEM", <object> of *M1*, and <grantee> *G* that is not grantable, the following <grant statement> is executed with a current user identifier of "_SYSTEM" and without further Access Rule checking:

```
GRANT EXECUTE ON M1 TO G WITH GRANT OPTION.
```

NOTE 15 — The privileges necessary include the EXECUTE privilege on every subject routine of every <routine invocation> contained in those <SQL procedure statement>s.

Conformance Rules

No additional Conformance Rules.

10.2 <privileges>

This Subclause modifies Subclause 12.3, "<privileges>", in ISO/IEC 9075-2.

Function

Specify privileges.

Format

```
<object name> ::=
    !! All alternatives from ISO/IEC 9075-2
  | MODULE <module name>
```

Syntax Rules

1) Replace SR 7) If the object identified by <object name> of the <grant statement> or <revoke statement> is an SQL-invoked routine or an SQL-server module, then <privileges> shall specify EXECUTE; otherwise, EXECUTE shall not be specified.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

1) Without Feature P001, "Stored modules", conforming SQL language shall not contain a <privilege> of MODULE.

10.3 <revoke statement>

This Subclause modifies Subclause 12.7, "<revoke statement>", in ISO/IEC 9075-2.

Function

Destroy privileges and role authorizations.

Format

No additional Format items.

Syntax Rules

1) Insert after SR 20)e) EXECUTE privilege on every SQL-server module that includes one or more SQL-invoked routines that are among the subject routines of a <routine invocation> that is generally contained in the <query expression> of *V*.

2) Insert after SR 22)e) EXECUTE privilege on every SQL-server modules that includes one or more SQL-invoked routines that are among the subject routines of a <routine invocation> that is generally contained in any <search condition> of *TC*.

3) Insert after SR 23)e) EXECUTE privilege on every SQL-server module that includes one or more SQL-invoked routines that are among the subject routines of a <routine invocation> that is generally contained in any <search condition> of *AX*.

4) Insert after SR 25)e) EXECUTE privilege on every SQL-server module that includes one or more SQL-invoked routines that are among the subject routines of a <routine invocation> that is generally contained in any <search condition> of *DC*.

5) Insert after SR 34)a) EXECUTE privilege on every SQL-server module that includes one or more SQL-invoked routines that are among the subject routines of a <routine invocation> that is contained in the <routine body> of *RD*.

6) Insert this SR Let *SSM* be any SQL-server module descriptor of an SQL-server module included in *S1*. *SSM* is said to be *abandoned* if the revoke destruction action would result in *A1* no longer having all of the following:

   a) EXECUTE privilege on every schema-level routine that is among the subject routines of a <routine invocation> that is contained in the <routine body> of any SQL-invoked routine included in *SSM*.

   b) EXECUTE privilege on every SQL-server module that includes one or more SQL-invoked routines that are among the subject routines of a <routine invocation> that is contained in the <SQL routine body> of any SQL-invoked routine included in *SSM*.

   c) 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 <routine body> of any SQL-invoked routine with a security characteristic of DEFINER included in *SSM*.

d) 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 <routine body> of any SQL-invoked routine with a security characteristic of DEFINER included in *SSM*.

e) 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: positioned>, an <update statement: searched>, or a <merge statement> contained in the <routine body> of any SQL-invoked routine with a security characteristic of DEFINER included in *SSM*.

f) 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 any SQL-invoked routine with a security characteristic of DEFINER included in *SSM*.

g) 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 any SQL-invoked routine with a security characteristic of DEFINER included in *SSM*.

h) 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 any SQL-invoked routine with a security characteristic of DEFINER included in *SSM*.

i) INSERT privilege on every column

   Case:

   i) Identified by a <column name> contained in the <insert column list> of an <insert statement> contained in the <routine body> of any SQL-invoked routine with a security characteristic of DEFINER included in *SSM*.

   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 any SQL-invoked routine with a security characteristic of DEFINER included in *SSM*.

   iii) Of the table identified by the <target table> immediately contained in an <merge statement> that contains a <merge specification> and that does not contain an <insert column list> and that is contained in the <SQL routine body> of any SQL-invoked routine with a security characteristic of DEFINER included in *SSM*.

j) UPDATE privilege on every 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 any SQL-invoked routine with a security characteristic of DEFINER included in *SSM*.

k) DELETE privilege on every table whose name is contained in a <table name> immediately contained in either a <delete statement: positioned> or a <delete statement: searched> contained in the <SQL routine body> of any SQL-invoked routine with a security characteristic of DEFINER included in *SSM*.

l) 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 with a security characteristic of DEFINER included in *SSM*.

m) 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 with a security characteristic of DEFINER included in *SSM* that 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 any SQL-invoked routine with a security characteristic of DEFINER included in *SSM* 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 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 with a security characteristic of DEFINER included in *SSM*.

p) 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 routine with a security characteristic of DEFINER included in *SSM*.

q) 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 with a security characteristic of DEFINER included in *SSM*.

r) 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 source> or an <assigned row> contained in the <SQL routine body> of any SQL-invoked routine with a security characteristic of DEFINER included in *SSM*.

s) 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 any SQL-invoked routine with a security characteristic of DEFINER included in *SSM*.

Access Rules

No additional Access Rules.

General Rules

1) Insert after GR 16) For every abandoned SQL-server module descriptor *MD*, let *M* be the SQL-server module whose descriptor is *MD*. Let *MN* be the <SQL-server module name> of *M*. The following <drop module statement> is effectively executed without further Access Rule checking:

```
DROP MODULE MN CASCADE
```

Conformance Rules

No additional Conformance Rules.

This page intentionally left blank.

11 SQL-client modules

This Clause modifies Clause 13, "SQL-client modules", in ISO/IEC 9075-2.

11.1 Calls to an <externally-invoked procedure>

This Subclause modifies Subclause 13.4, "Calls to an <externally-invoked procedure>", in ISO/IEC 9075-2.

Function

Define the call to an <externally-invoked procedure> by an SQL-agent.

Syntax Rules

1) Insert into SR 2)e)

```
CASE_NOT_FOUND_FOR_CASE_STATEMENT_NO_SUBCLASS:
  constant SQLSTATE_TYPE :="20000";
DATA_EXCEPTION_NULL_VALUE_IN_FIELD_REFERENCE:
  constant SQLSTATE_TYPE :="2202A";
DIAGNOSTICS_EXCEPTION_STACKED_DIAGNOSTICS_ACCESSED_WITHOUT_ACTIVE_HANDLER:
  constant SQLSTATE_TYPE :="0Z002";
RESIGNAL_WHEN_HANDLER_NOT_ACTIVE_NO_SUBCLASS:
  constant SQLSTATE_TYPE :="0K000";
UNHANDLED_USER_DEFINED_EXCEPTION_NO_SUBCLASS:
  constant SQLSTATE_TYPE :="45000";
```

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

11.2 <SQL procedure statement>

This Subclause modifies Subclause 13.5, "<SQL procedure statement>", in ISO/IEC 9075-2.

Function

Define all of the SQL-statements that are <SQL procedure statement>s.

Format

```
<SQL schema definition statement> ::=
    !! All alternatives from ISO/IEC 9075-2
  |<SQL-server module definition>

<SQL schema manipulation statement> ::=
    !! All alternatives from ISO/IEC 9075-2
  |<drop module statement>

<SQL control statement> ::=
    !! All alternatives from ISO/IEC 9075-2
  |<assignment statement>
  |<compound statement>
  |<case statement>
  |<if statement>
  |<iterate statement>
  |<leave statement>
  |<loop statement>
  |<while statement>
  |<repeat statement>
  |<for statement>

<SQL diagnostics statement> ::=
    !! All alternatives from ISO/IEC 9075-2
  |<signal statement>
  |<resignal statement>
```

Syntax Rules

1) Replace SR 2) An <SQL connection statement> shall not be generally contained in an <SQL control statement>, an <SQL-invoked routine>, or an <SQL-server module definition>.

2) Insert after SR 4)d) *S* is a <compound statement> and *S* contains an <SQL variable declaration> that specifies a <default option> that contains a <datetime value function>, CURRENT_USER, CURRENT_ROLE, SESSION_USER, or SYSTEM_USER.

3) Insert after SR 4)d) *S* is a <compound statement> and *S* contains an <SQL variable declaration> that specifies a <domain name> and the domain descriptor identified by the <domain name> has a default value that contains a <datetime value function>, CURRENT_USER, CURRENT_ROLE, SESSION_USER, or SYSTEM_USER.

Access Rules

No additional Access Rules.

General Rules

1) Replace GR 4)a)iii)7) If *S* is not a <compound statement>, then the first diagnostics area is emptied.

2) Insert before GR 7) Condition handling mode becomes in effect in the SQL-session.

3) Insert before GR 7) The General Rules of Subclause 13.2, "<handler declaration>", are applied.

4) Insert after GR 9) Condition handling mode ceases to be in effect in the SQL-session.

Conformance Rules

No additional Conformance Rules.

This page intentionally left blank.

12 Data manipulation

This Clause modifies Clause 14, "Data manipulation", in ISO/IEC 9075-2.

12.1 <open statement>

This Subclause modifies Subclause 14.2, "<open statement>", in ISO/IEC 9075-2.

Function

Open a cursor.

Format

```
No additional Format items.
```

Syntax Rules

1) Replace SR 1) Let *CN* be the <cursor name> in the <open statement>. *CN* shall be contained within the scope of one or more <cursor name>s that are equivalent to *CN*. If there is more than one such <cursor name>, then the one with the innermost scope is specified. Let *CR* be the cursor specified by *CN*.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

12.2 <fetch statement>

This Subclause modifies Subclause 14.3, "<fetch statement>", in ISO/IEC 9075-2.

Function

Position a cursor on a specified row of a table and retrieve values from that row.

Format

```
No additional Format items.
```

Syntax Rules

1) Replace SR 3) Let *CN* be the <cursor name> in the <fetch statement>. *CN* shall be contained within the scope of one or more <cursor name>s that are equivalent to *CN*. If there is more than one such <cursor name>, then the one with the innermost scope is specified. Let *CR* be the cursor specified by *CN*. Let *T* be the table defined by the <cursor specification> of *CR*. Let *DC* be the <declare cursor> denoted by *CN*.

2) Replace SR 6)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 the introductory paragraph of SR 6)b)ii) For *i* varying from 1 (one) to *NTS*, let $TS1_i$ be the *i*-th <target specification> in the <fetch target list> that is either an <SQL variable name>, an <SQL parameter reference>, or a <target array element specification>, and let CS_i be the *i*-th column of table *T* that corresponds with the <target specification> in the <fetch target list>.

Access Rules

No additional Access Rules.

General Rules

1) Replace GR 7)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) Replace the introductory paragraph of GR 7)b)i) If *TV* is either an <SQL variable reference>, an <SQL parameter reference>, or a <target array element specification>, then for each <target specification> in the <fetch target list>, let TV_i be the *i*-th <target specification> in the <fetch target list> and let SV_i denote the *i*-th corresponding value in the current row of *CR*.

Conformance Rules

No additional Conformance Rules.

12.3 <close statement>

This Subclause modifies Subclause 14.4, "<close statement>", in ISO/IEC 9075-2.

Function

Close a cursor.

Format

```
No additional Format items.
```

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

1) Replace GR 1) Let *CN* be the <cursor name> in the <close statement>. *CN* shall be contained within the scope of one or more <cursor name>s that are equivalent to *CN*. If there is more than one such <cursor name>, then the one with the innermost scope is specified. Let *CR* be the cursor specified by *CN*.

Conformance Rules

No additional Conformance Rules.

12.4 <select statement: single row>

This Subclause modifies Subclause 14.5, "<select statement: single row>", in ISO/IEC 9075-2.

Function

Retrieve values from a specified row of a table.

Format

No additional Format items.

Syntax Rules

1) Replace the introductory paragraph of SR 3)b)ii) For *i* varying from 1 (one) to *NOE*, let TS_i be the *i*-th <target specification> in the <select target list> that is either an <SQL variable reference>, an <SQL parameter reference>, or a <target array element specification>, and let SL_i be the *i*-th element of the <select list> that corresponds with the <target specification> in the <select target list>.

Access Rules

No additional Access Rules.

General Rules

1) Replace the introductory paragraph of GR 4)b)ii) For *i* varying from 1 (one) to *NOE*, let TS_i be the *i*-th <target specification> in the <select target list> that is either an <SQL variable reference>, an <SQL parameter reference>, or a <target array element specification>, and let SL_i denote the corresponding (*i*-th) value in the row of *Q*. The assignment of values to targets in the <select target list> is in an implementation-dependent order.

Conformance Rules

No additional Conformance Rules.

12.5 <delete statement: positioned>

This Subclause modifies Subclause 14.6, "<delete statement: positioned>", in ISO/IEC 9075-2.

Function

Delete a row of a table.

Format

```
No additional Format items.
```

Syntax Rules

1) Replace SR 1) Let *CN* be the <cursor name> in the <delete statement: positioned>. *CN* shall be contained within the scope of one or more <cursor name>s that are equivalent to *CN*. If there is more than one such <cursor name>, then the one with the innermost scope is specified. Let *CR* be the cursor specified by *CN*.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

12.6 <update statement: positioned>

This Subclause modifies Subclause 14.10, "<update statement: positioned>", in ISO/IEC 9075-2.

Function

Update a row of a table.

Format

```
No additional Format items.
```

Syntax Rules

1) Replace SR 1) Let *CN* be the <cursor name> in the <update statement: positioned>. *CN* shall be contained within the scope of one or more <cursor name>s that are equivalent to *CN*. If there is more than one such <cursor name>, then the one with the innermost scope is specified. Let *CR* be the cursor specified by *CN*.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

12.7 <temporary table declaration>

This Subclause modifies Subclause 14.13, "<temporary table declaration>", in ISO/IEC 9075-2.

Function

Replace 1st paragraph Declare a declared local temporary table.

Format

No additional Format items.

Syntax Rules

1) Replace SR 2) *TTD* shall be contained in an <SQL-client module definition> or in an <SQL-server module definition>.

2) Replace SR 4) Case:

   a) If a <temporary table declaration> is contained in an <SQL-client module definition> *M* without an intervening <SQL-server module definition>, then *TN* shall not be equivalent to the <table name> of any other <temporary table declaration> contained without an intervening <SQL-server module definition> in *M*.

   b) Otherwise, *TN* shall not be equivalent to the <table name> of any other <temporary table declaration> contained in *M*.

Access Rules

No additional Access Rules.

General Rules

1) Replace GR 1) Case:

   a) If <temporary table declaration> is contained in an <SQL-client module definition> without an intervening <SQL-server module definition>, then let *U* be the implementation-dependent <schema name> that is effectively derived from the implementation-dependent SQL-session identifier associated with the SQL-session and an implementation-dependent name associated with the SQL-client module that contains the <temporary table declaration>.

   b) Otherwise, let *U* be the implementation-dependent <schema name> that is effectively derived from the implementation-dependent SQL-session identifier associated with the SQL-session and the name associated of the <SQL-server module definition> that contains the <temporary table declaration>.

2) Replace GR 3) Case:

a) If <temporary table declaration> is contained in an <SQL-client module definition> without an intervening <SQL-server module definition>, then the definition of *T* within the <SQL-client module definition> is effectively equivalent to the definition of a persistent base table *U.T*. Within the SQL-client module, any reference to MODULE.*T* that is not contained in an <SQL schema statement> is equivalent to a reference to *U.T*.

b) Otherwise, the definition of *T* within an <SQL-server module definition> is effectively equivalent to the definition of a persistent base table *U.T*. Within the SQL-server module, any reference to MODULE.*T* is equivalent to a reference to *U.T*.

Conformance Rules

No additional Conformance Rules.

This page intentionally left blank.

13 Control statements

This Clause modifies Clause 15, "Control statements", in ISO/IEC 9075-2.

13.1 <compound statement>

Function

Specify a statement that groups other statements together.

Format

```
<compound statement> ::=
    [ <beginning label> <colon> ] BEGIN [ [ NOT ] ATOMIC ]
    [ <local declaration list> ] [ <local cursor declaration list> ]
    [ <local handler declaration list> ]
    [ <SQL statement list> ]
    END [ <ending label> ]

<beginning label> ::= <statement label>

<ending label> ::= <statement label>

<statement label> ::= <identifier>

<local declaration list> ::= <terminated local declaration>...

<terminated local declaration> ::= <local declaration> <semicolon>

<local declaration> ::=
    <SQL variable declaration>
  | <condition declaration>

<local cursor declaration list> ::= <terminated local cursor declaration>...

<terminated local cursor declaration> ::= <declare cursor> <semicolon>

<local handler declaration list> ::= <terminated local handler declaration>...

<terminated local handler declaration> ::= <handler declaration> <semicolon>

<SQL statement list> ::= <terminated SQL statement>...

<terminated SQL statement> ::= <SQL procedure statement> <semicolon>
```

Syntax Rules

1) Let *CS* be the <compound statement>.

2) If *CS* is contained in another <SQL control statement> and *CS* does not specify a <beginning label>, then an implementation-dependent <beginning label> that is not equivalent to any other <statement label> contained in the outermost containing <SQL control statement> is implicit.

3) If an <ending label> is specified, then *CS* shall specify a <beginning label> that is equivalent to that <ending label>.

4) The scope of the <beginning label> is *CS* excluding every <SQL schema statement> contained in *CS* and excluding every <local handler declaration list> contained in *CS*. <beginning label> shall not be equivalent to any other <beginning label>s within that scope.

5) If *CS* specifies neither ATOMIC nor NOT ATOMIC, then NOT ATOMIC is implicit.

6) If *CS* specifies ATOMIC, then the <SQL statement list> shall not contain either a <commit statement> or a <rollback statement> that does not specify a <savepoint clause>.

7) Let *VN* be an <SQL variable name> contained in a <local declaration list>. The *declared local name* of the variable identified by *VN* is *VN*.

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*.

9) Let *CN* be the <cursor name> immediately contained in a <declare cursor> *DC* contained in a <local cursor declaration list>. The *declared local name* of the cursor declared by *DC* is *CN*.

10) No two variables declared in a <local declaration list> shall have equivalent declared local names.

11) No two <condition declaration>s contained in a <local declaration list> shall have equivalent declared local names.

12) No two cursors declared in a <local cursor declaration list> shall have equivalent declared local names.

13) The scope of an <SQL variable name> of an <SQL variable declaration> simply contained in a <local declaration> simply contained in *CS* is the <local cursor declaration list> of *CS*, the <local handler declaration list> *LHDL* of *CS* excluding every <SQL schema statement> contained in *LHDL*, and the <SQL statement list> *SSL* of *CS* excluding every <SQL schema statement> contained in *SSL*.

14) The scope of the <condition name> in a <condition declaration> simply contained in a <local declaration> simply contained in *CS* is the <local handler declaration list> *LHDL* of *CS* excluding every <SQL schema statement> contained in *LHDL* and the <SQL statement list> *SSL* of *CS* excluding every <SQL schema statement> contained in *SSL*.

15) The scope of the <cursor name> in a <declare cursor> simply contained in a <terminated local cursor declaration> simply contained in *CS* is the <local handler declaration list> *LHDL* of *CS* excluding every <SQL schema statement> contained in *LHDL* and the <SQL statement list> *SSL* of *CS* excluding every <SQL schema statement> contained in *SSL*.

16) The scope of a <handler declaration> simply contained in a <local handler declaration list> simply contained in *CS* is the <SQL statement list> *SSL* of *CS* excluding every <SQL schema statement> contained in *SSL*.

17) If the <compound statement> simply contains a <handler declaration> that specifies UNDO, then ATOMIC shall be specified.

Access Rules

None.

General Rules

1) If *CS* specifies ATOMIC, then a new savepoint level is established.

2) The SQL variables, cursors, and handlers specified in the <local declaration list>, <local cursor declaration list>, and the <local handler declaration list> of *CS* are created in an implementation-dependent order.

3) Let *N* be the number of <SQL procedure statement>s contained in the <SQL statement list> that is immediately contained in *CS* without an intervening <SQL control statement>. For *i* ranging from 1 (one) to *N*:

   a) Let S_i be the *i*-th such <SQL procedure statement>.

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

   c) If the execution of S_i terminates with exception conditions or completion conditions other than *successful completion*, then:

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

         ```
         RESIGNAL
         ```

      ii) If there are unhandled exception conditions 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) 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 — "result set cursor" is defined in Subclause 4.32, "Cursors", in ISO/IEC 9075-2.

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.

6) If *CS* specifies ATOMIC, then the current savepoint level is destroyed.

   NOTE 17 — Destroying a savepoint level destroys all existing savepoints that are established at that level.

7) The <condition name> of every <condition declaration> contained in <local declaration list> ceases to be considered to be defined.

Conformance Rules

1) Without Feature P002, “Computational completeness”, conforming SQL language shall not contain a <compound statement>.

13.2 <handler declaration>

Function

Associate a handler with exception or completion conditions to be handled in a module or compound statement.

Format

```
<handler declaration> ::=
    DECLARE <handler type> HANDLER FOR <condition value list> <handler action>

<handler type> ::=
    CONTINUE
  | EXIT
  | UNDO

<handler action> ::= <SQL procedure statement>

<condition value list> ::= <condition value> [ { <comma> <condition value> }... ]

<condition value> ::=
    <sqlstate value>
  | <condition name>
  | SQLEXCEPTION
  | SQLWARNING
  | NOT FOUND
```

Syntax Rules

1) Let *HD* be the <handler declaration>.

2) A <condition name> *CN* specified in a <condition value> of *HD* shall be defined by some <condition declaration> with a scope that contains *HD*. Let *C* be the condition specified by the innermost such <condition declaration>.

3) If a <condition value> specifies SQLEXCEPTION, SQLWARNING, or NOT FOUND, then neither <sqlstate value> nor <condition value> shall be specified.

4) No other <handler declaration> with the same scope as *HD* shall contain in its <condition value list> a <condition value> that represents the same condition as a <condition value> contained in the <condition value list> of *HD*.

5) The <condition value list> shall not contain the same <condition value> or <sqlstate value> more than once, nor shall it contain both the <condition name> of a condition *C* and an <sqlstate value> that represents the SQLSTATE value associated with *C*.

6) SQLEXCEPTION, SQLWARNING, and NOT FOUND correspond to SQLSTATE class values corresponding to categories X, W, and N, respectively, in Subclause 23.1, "SQLSTATE", in ISO/IEC 9075-2.

7) If a <condition value> specifies SQLEXCEPTION, SQLWARNING, or NOT FOUND, then the <handler declaration> is a *general* <*handler declaration*>; otherwise, the <handler declaration> is a *specific* <*handler declaration*>.

8) If there is a general <handler declaration> and a specific <handler declaration> for the same <condition value> in the same scope, then only the specific <handler declaration> is associated with that <condition value>.

9) Let *HA* be the <handler action>.

10) *HA* is associated with every <condition name> specified in the <condition value list> of *HD* and with every SQLSTATE value specified in every <sqlstate value> specified in the <condition value list> of *HD*.

11) If *HA* is associated with a <condition name> and that <condition name> was defined for an SQLSTATE value, then *HA* is also associated with that SQLSTATE value.

12) If *HA* is associated with an SQLSTATE class, then it is associated with each SQLSTATE value of that class.

Access Rules

None.

General Rules

1) When the handler *H* associated with the conditions specified by *HD* is created, it is the *most appropriate handler* for any condition *CN* raised during execution of any SQL-statements that are in the scope of *HD* that has an SQLSTATE value or condition name that is the same as an SQLSTATE value or condition name associated with this handler, until *H* is destroyed. *CN* has a more appropriate handler if, during the existence of *H*, another handler *AH* is created with a scope containing *CN*, and if *AH* is associated with an SQLSTATE value or condition name that is the same as the SQLSTATE value or condition name of *CN*. *AH* replaces *H* as the most appropriate handler for *CN* until *AH* is destroyed. When *AH* is destroyed, *H* is reinstated as the most appropriate handler for *CN*.

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

3) When *H* is activated,

   Case:

   a) If *HD* specifies CONTINUE, then:

      i) The General Rules of Subclause 22.2, "Pushing and popping the diagnostics area stack", in ISO/IEC 9075-2, are applied with "PUSH" as *OPERATION* and the diagnostics area stack as *STACK*.

      ii) *HA* is executed.

      iii) Case:

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

            A) The General Rules of Subclause 22.2, "Pushing and popping the diagnostics area stack", in ISO/IEC 9075-2, are applied with "PUSH" as *OPERATION* and the diagnostics area stack as *STACK*.

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

```
RESIGNAL
```

2) Otherwise:

A) The General Rules of Subclause 22.2, "Pushing and popping the diagnostics area stack", in ISO/IEC 9075-2, are applied with "POP" as *OPERATION* and the diagnostics area stack as *STACK*.

B) *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 the diagnostics area stack", in ISO/IEC 9075-2, 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
```

iv) Case:

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

A) The General Rules of Subclause 22.2, "Pushing and popping the diagnostics area stack", in ISO/IEC 9075-2, are applied with "PUSH" as *OPERATION* and the diagnostics area stack as *STACK*.

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

```
RESIGNAL
```

2) Otherwise:

A) The General Rules of Subclause 22.2, "Pushing and popping the diagnostics area stack", in ISO/IEC 9075-2, are applied with "POP" as *OPERATION* and the diagnostics area stack as *STACK*.

B) *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 the diagnostics area stack", in ISO/IEC 9075-2, 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 canceled.

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:

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

A) The General Rules of Subclause 22.2, "Pushing and popping the diagnostics area stack", in ISO/IEC 9075-2, are applied with "PUSH" as *OPERATION* and the diagnostics area stack as *STACK*.

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

```
RESIGNAL
```

2) Otherwise:

A) The General Rules of Subclause 22.2, "Pushing and popping the diagnostics area stack", in ISO/IEC 9075-2, are applied with "POP" as *OPERATION* and the diagnostics area stack as *STACK*.

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

Conformance Rules

1) Without Feature P002, "Computational completeness", conforming SQL language shall not contain a <handler declaration>.

13.3 <condition declaration>

Function

Declare a condition name and an optional corresponding SQLSTATE value.

Format

```
<condition declaration> ::= DECLARE <condition name> CONDITION [ FOR <sqlstate value> ]
```

Syntax Rules

1) Let *CD* be the <condition declaration>.

2) Let *CN* be the <condition name>. At most one <condition declaration> shall specify a <condition name> that is equivalent to *CN* and has the same scope as *CN*.

3) <condition name> is *considered to be defined*, within its scope, for the SQLSTATE value specified by <sqlstate value>.

Access Rules

None.

General Rules

None.

Conformance Rules

1) Without Feature P002, “Computational completeness”, conforming SQL language shall not contain a <condition declaration>.

13.4 <SQL variable declaration>

Function

Declare one or more variables.

Format

```
<SQL variable declaration> ::=
    DECLARE <SQL variable name list> <data type> [ <default clause> ]

<SQL variable name list> ::= <SQL variable name> [ { <comma> <SQL variable name> }... ]
```

Syntax Rules

1) The specified <data type> is the declared type of each variable declared by the <SQL variable declaration>.

Access Rules

None.

General Rules

1) When the variable associated with the <SQL variable declaration> is created, its default value *DV* is derived according to the General Rules of Subclause 9.3, "<default clause>". Let *SV* be the variable defined by the <SQL variable declaration>. The value of *SV* is set to *DV* by the effective invocation of the following SQL-statement:

```
SET SV = DV
```

Conformance Rules

1) Without Feature P002, "Computational completeness", conforming SQL language shall not contain a <SQL variable declaration>.

13.5 <assignment statement>

Function

Assign a value to an SQL variable, SQL parameter, host parameter, or host variable.

Format

```
<assignment statement> ::=
    <singleton variable assignment>
  | <multiple variable assignment>

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

<assignment target list> ::=
    <left paren> <assignment target> [ { <comma> <assignment target> }... ] <right paren>

<singleton variable assignment> ::=
    SET <assignment target> <equals operator> <assignment source>

<assignment target> ::=
    <target specification>
  | <modified field reference>
  | <mutator reference>

<assignment source> ::=
    <value expression>
  | <contextually typed source>

<contextually typed source> ::=
    <implicitly typed value specification>
  | <contextually typed row value expression>

<modified field reference> ::= <modified field target> <period> <field name>

<modified field target> ::=
    <target specification>
  | <left paren> <target specification> <right paren>
  | <modified field reference>

<mutator reference> ::= <mutated target specification> <period> <method name>

<mutated target specification> ::=
    <target specification>
  | <left paren> <target specification> <right paren>
  | <mutator reference>
```

Syntax Rules

1) An <assignment statement> *A* that contains a <multiple variable assignment> is effectively replaced by a <compound statement> *CS* as follows:

a) Let *ATL* be the <assignment target list> contained in *A*, let *ATN* be the number of <assignment target>s contained in *ATL*, and let *AR* be the <assigned row> contained in *A*.

b) *ATN* shall be equal to the degree of *AR*.

c) Let *X* be an arbitrary <SQL variable name> that is not equivalent to any <target specification> contained in *A*.

d) Let *XT* be the declared type of *AR*.

e) Let AT_i, 1 (one) ≤ *i* ≤ *UTN*, be the *i*-th <assignment target> contained in *ATL* and let FN_i be the <field name> of the *i*-th field of *AR*.

f) *CS* is:

```
BEGIN
  DECLARE X XT ;
  SET X = AR ;
  SET AT1 = X . FN1 ;
  . . .
  SET ATATN = X . FNATN ;
END
```

2) A <column reference> immediately contained in a <modified field target> or a <mutated target specification> shall be a new transition variable column reference.

NOTE 18 — "New transition variable column reference" is defined in Subclause 6.6, "<identifier chain>", in ISO/IEC 9075-2.

3) 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 immediately contain a <column reference>.

4) The declared type of the <target specification> simply contained in a <mutator reference> *MR* shall be a user-defined type.

5) If <assignment target> immediately contains a <mutator reference>, then let *TS* be the <mutated target>, let *FN* be the <method name>, and let *AS* be the <assignment source>. The <assignment statement> is equivalent to:

```
SET TS = TS.FN ( AS )
```

NOTE 19 — The preceding rule is applied recursively until the <assignment target> no longer contains a <mutator reference>.

6) If <assignment target> is a <modified field reference> *FR*, then:

a) Let *F* be the field identified by <field name> simply contained in <assignment target> and not simply contained in <modified field target>.

b) Let *AS* be the <assignment source>.

c) The Syntax Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2 are applied to *F* and *AS* as *TARGET* and *VALUE*, respectively.

7) If the <assignment target> simply contains an <embedded variable name> or a <host parameter specification>, then <assignment source> shall not simply contain an <embedded variable name> or a <host parameter specification>.

8) If the <assignment target> simply contains a <column reference>, an <SQL variable reference>, or an <SQL parameter reference> and the <assignment source> is a <value expression>, then the Syntax Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2 are applied to <assignment target> and <assignment source> as *TARGET* and *VALUE*, respectively.

9) If the <assignment target> simply contains an <embedded variable name> or a <host parameter specification> and the <assignment source> is a <value expression>, then the Syntax Rules of Subclause 9.1, "Retrieval assignment", in ISO/IEC 9075-2 are applied to <assignment target> and <assignment source> as *TARGET* and *VALUE*, respectively.

10) If <target array element specification> is specified, then:

    a) <target array reference> contained in an <assignment target> shall not be <column reference>.

    b) The Syntax Rules of Subclause 9.2, "Store assignment", apply to an arbitrary site whose declared type is the declared type of the <target specification> and the <assignment source> as *TARGET* and *VALUE*, respectively.

11) A <contextually typed row value expression> that is specified as a <contextually typed source> shall not contain a <default specification>.

Access Rules

None.

General Rules

1) If <assignment target> is a <target specification> that is a <column reference> *T*, an <SQL variable reference> to an SQL variable *T*, or an <SQL parameter reference> to an SQL parameter *T* of an SQL-invoked routine, then the value of <assignment source> is assigned to *T* according to the General Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2, with <assignment source> and *T* as *VALUE* and *TARGET*, respectively.

2) If <assignment target> is a <target specification> that is the <embedded variable name> of a host variable *T* or the <host parameter specification> of a host parameter *T*, then the value of <assignment source> is assigned to *T* according to the General Rules of Subclause 9.1, "Retrieval assignment", in ISO/IEC 9075-2, with <assignment source> and *T* as *VALUE* and *TARGET*, respectively.

3) If <assignment target> is a <target specification> that is a new transition variable column reference, then let *C* be the column identified by the <column reference> and let *R* be the row that is to be replaced by that transition variable. For each transition variable *TV* that is a replacement for a subrow of *R* or for a superrow of *R* in a table in which *C* is a column, the value of <assignment source> is assigned to *TV.C* according to the General Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2, with <assignment source> and *TV.C* as *VALUE* and *TARGET*, respectively.

4) If <assignment target> is a <modified field reference> *FR*, then let *T* be the <target specification> simply contained in *FR*. Let F_i be a field identified by each <field name> simply contained in *FR*. Let *FT* be the

field identified by the <field name> that is simply contained in <assignment target> and that is not simply contained in <modified field target>.

Case:

a) If the value of *T* or of any F_i is the null value, then an exception condition is raised: *data exception — null value in field reference*.

b) Otherwise, the value of <assignment source> is assigned to *FT* according to the General Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2, with <assignment source> and *FT* as *VALUE* and *TARGET*, respectively.

5) If <target array element specification> is specified, then

Case:

a) If the value of <target specification>, denoted by *C*, is null, then an exception condition is raised: *data exception — null value in array target*.

b) Otherwise:

i) Let *N* be the maximum cardinality of *C*.

ii) Let *M* be the cardinality of the value of *C*.

iii) Let *I* be the value of the <simple value specification> immediately contained in <target specification>.

iv) Let *EDT* be the element type of *C*.

v) Case:

1) If *I* is greater than zero and less than or equal to *M*, then the value of *C* is replaced by an array *A* with element type *EDT* and cardinality *M* derived as follows:

A) For *j* varying from 1 (one) to *I*–1 and from *I*+1 to *M*, the *j*-th element in *A* is the value of the *j*-th element in *C*.

B) The *I*-th element of *A* is set to the value of the <assignment source>, denoted by *SV*, by applying the General Rules of Subclause 9.2, "Store assignment", to the *I*-th element of *A* and *SV* as *TARGET* and *VALUE*, respectively.

2) If *I* is greater than *M* and less than or equal to *N*, then the value of *C* is replaced by an array *A* with element type *EDT* and cardinality *I* derived as follows:

A) For *j* varying from 1 (one) to *M*, the *j*-th element in *A* is the value of the *j*-th element in *C*.

B) For *j* varying from *M*+1 to *I*, the *j*-th element in *A* is the null value.

C) The *I*-th element of *A* is set to the value of the <assignment source>, denoted by *SV*, by applying the General Rules of Subclause 9.2, "Store assignment", to the *I*-th element of *A* and *SV* as *TARGET* and *VALUE*, respectively.

3) Otherwise, an exception condition is raised: *data exception — array element error*.

Conformance Rules

1) Without Feature P002, “Computational completeness”, conforming SQL language shall not contain a <assignment statement>.

2) Without Feature P006, “Multiple assignment”, conforming SQL language shall not contain a <multiple variable assignment>.

3) Without Feature T051, “Row types”, conforming SQL language shall not contain a <modified field reference>.

13.6 <case statement>

Function

Provide conditional execution based on truth of <search condition>s or on equality of operands.

Format

```
<case statement> ::=
    <simple case statement>
  | <searched case statement>

<simple case statement> ::=
    CASE <case operand>
    <simple case statement when clause>...
    [ <case statement else clause> ]
    END CASE

<searched case statement> ::=
    CASE <searched case statement when clause>...
    [ <case statement else clause> ]
    END CASE

<simple case statement when clause> ::=
    WHEN <when operand list>
    THEN <SQL statement list>

<searched case statement when clause> ::=
    WHEN <search condition>
    THEN <SQL statement list>

<case statement else clause> ::= ELSE <SQL statement list>
```

Syntax Rules

1) If a <case statement> specifies a <simple case statement>, then let *SCO1* be the <case operand>:

   a) *SCO1* shall not generally contain a <routine invocation> whose subject routines include an SQL-invoked routine that is possibly non-deterministic or that possibly modifies SQL-data.

   b) If *SCO1* is <overlaps predicate part 1>, then each <when operand> shall be <overlaps predicate part 2>. If *SCO1* is <row value predicand>, then each <when operand> shall not be <overlaps predicate part 2>.

   c) Let *N* be the number of <simple case statement when clause>s.

   d) For each *i* between 1 (one) and *N*, let WOL_i be the <when operand list> of the *i*-th <simple case statement when clause>. Let *M*(*i*) be the number of <when operand>s simply contained in WOL_i. For each *j* between 1 (one) and *M*(*i*), let $WO_{i,j}$ be the *j*-th <when operand> simply contained in WOL_i.

   e) For each *i* between 1 (one) and *N*, and for each *j* between 1 (one) and *M*(*i*),

Case:

i) If $WO_{i,j}$ is a <row value predicand>, then let $SCO2_{i,j}$ be

```
= WOi,j
```

ii) Otherwise, let $SCO2_{i,j}$ be $WO_{i,j}$.

f) Let SSL_i be the <SQL statement list> of the *i*-th <simple cast statement when clause>.

g) If <case statement else clause> is specified, then let *CSEC* be the <case statement else clause>; otherwise, let *CSEC* be a character string of length 0 (zero).

h) The <simple case statement> is equivalent to a <searched case statement> in which the *i*-th <searched case statement when clause> takes the form:

```
WHEN ( SCO1 SCO2i,j ) OR
. . . OR
( SCO1 SCO2i,M(i) )
THEN SSLi
```

i) The <case statement else clause> of the equivalent <searched case statement> takes the form:

```
CSEC
```

Access Rules

None.

General Rules

1) Case:

a) If the <search condition> of some <searched case statement when clause> in a <case statement> is *True*, then let *SL* be the <SQL statement list> of the first (leftmost) <searched case statement when clause> whose <search condition> is *True*.

b) If the <case statement> simply contains a <case statement else clause>, then let *SL* be the <SQL statement list> of that <case statement else clause>.

c) Otherwise, an exception condition is raised: *case not found for case statement*, and the execution of the <case statement> is terminated immediately.

2) Let *N* be the number of <SQL procedure statement>s simply contained in *SL* without an intervening <SQL control statement>. For *i* ranging from 1 (one) to *N*:

a) Let S_i be the *i*-th such <SQL procedure statement>.

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

c) If the execution of S_i terminates with an unhandled exception condition, then the execution of the <case statement> is terminates with that condition.

Conformance Rules

1) Without Feature P002, "Computational completeness", conforming SQL language shall not contain a <case statement>.

2) Without Feature P004, "Extended CASE statement", in conforming SQL language, both <case operand> immediately contained in a <simple case statement> and a <when operand> immediately contained in a <when operand list> immediately contained in a <simple case statement when clause> shall be a <row value predicand> that is a <row value constructor predicand> that is a single <common value expression> or <boolean value predicand>.

3) Without Feature P008, "Comma-separated predicates in simple CASE statement", in conforming SQL language, a <when operand list> immediately contained in a <when operand list> immediately contained in a <simple case statement when clause> shall simply contain exactly one <when operand>.

13.7 <if statement>

Function

Provide conditional execution based on the truth value of a condition.

Format

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

<if statement then clause> ::= THEN <SQL statement list>

<if statement elseif clause> ::= ELSEIF <search condition> THEN <SQL statement list>

<if statement else clause> ::= ELSE <SQL statement list>
```

Syntax Rules

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>.

Access Rules

None.

General Rules

1) Case:

   a) If the <search condition> immediately contained in the <if statement> evaluates to *True*, then let *SL* be the <SQL statement list> immediately contained in the <if statement then clause>.

   b) Otherwise, if an <if statement else clause> is specified, then let *SL* be the <SQL statement list> immediately contained in the <if statement else clause>.

      NOTE 20 — "Otherwise" means that the <search condition> immediately contained in the <if statement> evaluates to *False* or to *Unknown*.

2) Let *N* be the number of <SQL procedure statement>s simply contained in *SL* without an intervening <SQL control statement>. For *i* ranging from 1 (one) to *N*:

   a) Let S_i be the *i*-th such <SQL procedure statement>.

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

   c) If the execution of S_i terminates with an unhandled exception condition, then the execution of the <if statement> is terminated and the condition remains active.

Conformance Rules

1) Without Feature P002, "Computational completeness", conforming SQL language shall not contain a <if statement>.

13.8 <iterate statement>

Function

Terminate the execution of an iteration of an iterated SQL-statement.

Format

```
<iterate statement> ::= ITERATE <statement label>
```

Syntax Rules

1) <statement label> shall be the <beginning label> of some iterated SQL-statement *IS* that contains <iterate statement> without an intervening <SQL-schema statement>.

2) Let *SSL* be the <SQL statement list> simply contained in *IS*.

Access Rules

None.

General Rules

1) The execution of *SSL* is terminated.

NOTE 21 — If the iteration condition for *IS* is *True* or if *IS* does not have an iteration condition, then the next iteration of *SSL* commences immediately. If the iteration condition for *IS* is *False*, then there is no next iteration of *SSL*.

Conformance Rules

1) Without Feature P002, “Computational completeness”, conforming SQL language shall not contain a <iterate statement>.

13.9 <leave statement>

Function

Continue execution by leaving a labeled statement.

Format

```
<leave statement> ::= LEAVE <statement label>
```

Syntax Rules

1) <statement label> shall be the <beginning label> of some <SQL procedure statement> *S* that contains <leave statement> *L* without an intervening <SQL-schema statement>.

Access Rules

None.

General Rules

1) For every <compound statement> *CS* that is contained in *S* and that contains the <leave statement>:

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

      ```
      CLOSE CR
      ```

   b) The 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.

2) The execution of *S* is terminated.

Conformance Rules

1) Without Feature P002, "Computational completeness", conforming SQL language shall not contain a <leave statement>.

13.10 <loop statement>

Function

Repeat the execution of a statement.

Format

```
<loop statement> ::=
    [ <beginning label> <colon> ]
    LOOP
    <SQL statement list>
    END LOOP [ <ending label> ]
```

Syntax Rules

1) Let *LS* be the <loop statement>.

2) If *LS* is contained in another <SQL control statement> and *LS* does not specify a <beginning label>, then an implementation-dependent <beginning label> that is not equivalent to any other <statement label> contained in the outermost containing <SQL control statement> is implicit.

3) If <ending label> is specified, then a <beginning label> shall be specified that is equivalent to <ending label>.

4) The scope of the <beginning label> is *LS* excluding every <SQL schema statement> contained in *LS*. <beginning label> shall not be equivalent to any other <beginning label> contained in *LS* excluding every <SQL schema statement> contained in *LS*.

Access Rules

None.

General Rules

1) Let *SSL* be the <SQL statement list> and let *CCS* be the <compound statement>

   BEGIN NOT ATOMIC *SSL* END

   The General Rules of Subclause 13.5, "<SQL procedure statement>", of ISO/IEC 9075-2, are evaluated repeatedly with *CCS* as the *executing statement*.

   NOTE 22 — The occurrence of an exception condition or the execution of a <leave statement> may also cause execution of *LS* to be terminated; see Subclause 6.3.3.7, "Exceptions", in ISO/IEC 9075-1, and Subclause 13.9, "<leave statement>", respectively. Some actions taken by a condition handler might also cause execution of *LS* to be terminated; see Subclause 13.2, "<handler declaration>".

Conformance Rules

1) Without Feature P002, “Computational completeness”, conforming SQL language shall not contain a <loop statement>.

13.11 <while statement>

Function

While a specified condition is *True*, repeat the execution of a statement.

Format

```
<while statement> ::=
    [ <beginning label> <colon> ]
    WHILE <search condition> DO
    <SQL statement list>
    END WHILE [ <ending label> ]
```

Syntax Rules

1) Let *WS* be the <while statement>.

2) If *WS* is contained in another <SQL control statement> and *WS* does not specify a <beginning label>, then an implementation-dependent <beginning label> that is not equivalent to any other <statement label> contained in the outermost containing <SQL control statement> is implicit.

3) If <ending label> is specified, then a <beginning label> shall be specified that is equivalent to <ending label>.

4) The scope of the <beginning label> is *WS* excluding every <SQL schema statement> contained in *WS*. <beginning label> shall not be equivalent to any other <beginning label> contained in *WS* excluding every <SQL schema statement> contained in *WS*.

Access Rules

None.

General Rules

1) The <search condition> is evaluated.

2) Case:

   a) If the <search condition> evaluates to *False* or *Unknown*, then execution of *WS* is terminated.

   b) Let *SSL* be the <SQL statement list> and let *CCS* be the <compound statement>

      ```
      BEGIN NOT ATOMIC SSL END
      ```

      If the <search condition> evaluates to *True*, then the General Rules of Subclause 13.5, "<SQL procedure statement>", of ISO/IEC 9075-2, are evaluated with *CCS* as the *executing statement* and the execution of *WS* is repeated.

NOTE 23 — The occurrence of an exception condition or the execution of a <leave statement> may also cause execution of *WS* to be terminated; see Subclause 6.3.3.7, "Exceptions", in ISO/IEC 9075-1, and Subclause 13.9, "<leave statement>", respectively. Some actions taken by a condition handler might also cause execution of *WS* to be terminated; see Subclause 13.2, "<handler declaration>".

Conformance Rules

1) Without Feature P002, "Computational completeness", conforming SQL language shall not contain a <while statement>.

13.12 <repeat statement>

Function

Repeat the execution of a statement.

Format

```
<repeat statement> ::=
    [ <beginning label> <colon> ]
    REPEAT
    <SQL statement list>
    UNTIL <search condition>
    END REPEAT [ <ending label> ]
```

Syntax Rules

1) Let *RS* be the <repeat statement>.

2) If *RS* is contained in another <SQL control statement> and *RS* does not specify a <beginning label>, then an implementation-dependent <beginning label> that is not equivalent to any other <statement label> contained in the outermost containing <SQL control statement> is implicit.

3) If <ending label> is specified, then a <beginning label> shall be specified that is equivalent to <ending label>.

4) The scope of the <beginning label> is *RS* excluding every <SQL schema statement> contained in *RS*. <beginning label> shall not be equivalent to any other <beginning label> contained in *RS* excluding every <SQL schema statement> contained in *RS*.

Access Rules

None.

General Rules

1) Let *SSL* be the <SQL statement list> and let *CCS* be the <compound statement>

   ```
   BEGIN NOT ATOMIC SSL END
   ```

   the General Rules of Subclause 13.5, "<SQL procedure statement>", of ISO/IEC 9075-2, are evaluated with *CCS* as the *executing statement* and then <search condition> is evaluated.

   NOTE 24 — The occurrence of an exception condition or the execution of a <leave statement> may also cause execution of *RS* to be terminated; see Subclause 6.3.3.7, "Exceptions", in ISO/IEC 9075-1, and Subclause 13.9, "<leave statement>", respectively. Some actions taken by a condition handler might also cause execution of *RS* to be terminated; see Subclause 13.2, "<handler declaration>".

2) If the <search condition> evaluates to *False* or *Unknown*, then the execution of *RS* is repeated; otherwise, execution of *RS* is terminated.

Conformance Rules

1) Without Feature P002, “Computational completeness”, conforming SQL language shall not contain a <repeat statement>.

13.13 <for statement>

Function

Execute a statement for each row of a table.

Format

```
<for statement> ::=
    [ <beginning label> <colon> ]
    FOR [ <for loop variable name> AS ]
    [ <cursor name> [ <cursor sensitivity> ] CURSOR FOR ]
    <cursor specification>
    DO <SQL statement list>
    END FOR [ <ending label> ]

<for loop variable name> ::= <identifier>
```

Syntax Rules

1) Let *FCS* be the <cursor specification> of the <for statement> *FS*.

2) If *FS* is contained in another <SQL control statement> and *FS* does not specify a <beginning label>, then an implementation-dependent <beginning label> that is not equivalent to any other <statement label> contained in the outermost containing <SQL control statement> is implicit.

3) If <ending label> is specified, then a <beginning label> shall be specified that is equivalent to <ending label>.

4) If <cursor name> is specified, then let *CN* be that <cursor name>. Otherwise, let *CN* be an implementation-dependent <cursor name> that is not equivalent to any other <cursor name> in the outermost containing <SQL-client module definition> or <SQL-invoked routine>.

5) Let *QE* be the <query expression> of *FCS*. Each column of the table specified by *QE* shall have a <column name> that is not equivalent to any other <column name> in the table specified by *QE*. Let *V1*, *V2*, ..., *Vn* be those <column name>s. Let *DT1*, *DT2*, ..., *DTn* be the declared types of the respective columns.

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>.

7) Let *COMMON_CODE* be:

```
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;
```

Case:

a) If <for loop variable name> is specified, then *FS* is equivalent to:

```
BL: BEGIN NOT ATOMIC
  FLVN: BEGIN NOT ATOMIC
    COMMON_CODE
  END FLVN;
END BL
```

b) Otherwise, *FS* is equivalent to:

```
BL: BEGIN NOT ATOMIC
  COMMON_CODE
END BL
```

8) *SLL* shall not contain, without an intervening <SQL-invoked routine> or <SQL schema statement>, a <leave statement> that specifies *FLVN*.

9) *SLL* shall not contain either a <commit statement> or a <rollback statement>.

10) *SLL* shall not contain without an intervening <SQL-invoked routine> or <SQL schema statement> a <fetch statement>, an <open statement>, or a <close statement> that specifies *CN*.

Access Rules

None.

General Rules

None.

Conformance Rules

1) Without Feature P002, “Computational completeness”, conforming SQL language shall not contain a <for statement>.

This page intentionally left blank.

14 Dynamic SQL

This Clause modifies Clause 19, "Dynamic SQL", in ISO/IEC 9075-2.

14.1 <prepare statement>

This Subclause modifies Subclause 19.6, "<prepare statement>", in ISO/IEC 9075-2.

Function

Prepare a statement for execution.

Format

```
No additional Format items.
```

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

1) Append to the lead paragraph of GR 6) The syntactic substitutions specified in Subclause 13.5, "<assignment statement>", shall not be applied until the data types of <dynamic parameter specification>s are determined by this General Rule.

2) Insert after GR 6)a)xxvii) If *DP* is a <contextually-typed row value expression> simply contained in a <multiple variable assignment> *MVA* of an <assignment statement> or if *DP* represents the value of a subfield *SF* of the declared type of such a <contextually-typed row value expression>, then let *RT* be a row type in which the declared type of the *i*-th field is the declared type of the *i*-th field is the declared type of the <target specification>, <modified field reference>, or <mutator reference> immediately contained in the *i*-th <assignment target> contained in the <assignment target list> of *MVA*.

   Case:

   a) If *DP* is a <contextually-typed row value expression> simply contained in *MVA*, then *DT* is *RT*.

   b) Otherwise, *DT* is the declared type of the subfield of *RT* that corresponds to *SF*.

3) Insert after GR 6)a)xxx) If *DP* is the <assignment target> simply contained in a <singleton variable assignment> *SVA*, then

Case:

a) If the <assignment source> immediately contains a <null specification>, then *DT* is undefined.

b) Otherwise, *DT* is the declared type of the <value expression> simply contained in the <assignment source> of *SVA*.

4) Insert after GR 6)a)xxx) If *DP* is the <value expression> simply contained in an <assignment statement> in a <singleton variable assignment> *SVA* or if *DP* represents the value of a subfield *SF* of the declared type of such a <value expression>, then let *RT* be the declared type of the <assignment target> simply contained in *SVA*.

Case:

a) If *DP* is the <value expression> simply contained in the <assignment source>, then *DT* is *RT*.

b) Otherwise, *DT* is the declared type of the subfield of *RT* that corresponds to *SF*.

5) Insert after GR 6)a)xxx) If *DP* is a <value expression> simply contained in a <simple case operand 1> or a <simple case operand 2> of a <simple case statement> *CS*, or if *DP* represents the value of a subfield *SF* of such a <value expression>, then let *RT* be the result of applying the Syntax Rules of Subclause 9.3, "Data types of results of aggregations", in ISO/IEC 9075-2 to the <value expression>s simply contained in the <simple case operand 1> and all <simple case operand 2>s simply contained in *CS*.

Case:

a) If *DP* is a <value expression> simply contained in the <simple case operand 1> or <simple case operand 2> of *CS*, then *DT* is *RT*.

b) Otherwise, *DT* is the declared type of the subfield of *RT* that corresponds to *SF*.

Conformance Rules

No additional Conformance Rules.

15 Embedded SQL

This Clause modifies Clause 20, "Embedded SQL", in ISO/IEC 9075-2.

15.1 <embedded SQL host program>

This Subclause modifies Subclause 20.1, "<embedded SQL host program>", in ISO/IEC 9075-2.

Function

Specify an <embedded SQL host program>.

Format

No additional Format items.

Syntax Rules

1) Insert this SR An <SQL variable declaration> that is contained in an <embedded SQL host program> shall precede in the text of that <embedded SQL host program> any SQL-statement that references the <SQL variable name> of the <SQL variable declaration>.

2) Insert this SR An <SQL variable name> contained in an <SQL variable declaration> that is immediately contained in an <embedded SQL host program> shall not be equivalent to any other <SQL variable name> or <embedded variable name> contained in any other <SQL variable declaration> or <host variable definition>, respectively, that is immediately contained in the <embedded SQL host program>.

3) Insert this SR If a <handler declaration> is immediately contained in an <embedded SQL host program> with no intervening <compound statement>, then any <condition value> contained in that <handler declaration> shall not be equivalent to the <condition value> of any other <handler declaration> immediately contained in that <embedded SQL host program>.

4) Insert before SR 21)k) *M* contains one <SQL variable declaration> for each <SQL variable declaration> contained in *H*. Each <SQL variable declaration> of *M* is a copy of the corresponding <SQL variable declaration> of *H*.

5) Insert before SR 21)l) *M* contains one <handler declaration> for each <handler declaration> contained in *H*. Each <handler declaration> of *M* is a copy of the corresponding <handler declaration> of *H*.

6) Replace SR 22)c) Each <embedded SQL statement> that contains a <declare cursor>, a <dynamic declare cursor>, an <SQL variable declaration>, an <SQL-invoked routine>, or a <temporary table declaration> has been deleted, and every <embedded SQL statement> that contains an <embedded exception declaration>

has been replaced with statements of the host language that will have the effect specified by the General Rules of Subclause 20.2, "<embedded exception declaration>".

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

16 Diagnostics management

This Clause modifies Clause 22, "Diagnostics management", in ISO/IEC 9075-2.

16.1 <get diagnostics statement>

This Subclause modifies Subclause 22.1, "<get diagnostics statement>", in ISO/IEC 9075-2.

Function

Get exception or completion condition information from the diagnostics area.

Format

```
<get diagnostics statement> ::= GET [ <which area> ] DIAGNOSTICS <SQL diagnostics
information>

<which area> ::=
    CURRENT
  | STACKED

<condition information item name> ::=
    !! All alternatives from ISO/IEC 9075-2
  | CONDITION_IDENTIFIER
```

Syntax Rules

1) Insert this SR If <which area> is not specified, then CURRENT is implicit.

2) *Table 1, "<identifier>s for use with <get diagnostics statement>", modifies Table 30, "<identifier>s for use with <get diagnostics statement>", in ISO/IEC 9075-2.*

Table 1 — <identifier>s for use with <get diagnostics statement>

<identifier>	Data Type
<statement information item name>s	
All alternatives from ISO/IEC 9075-2	
<condition information item name>s	

<identifier>	Data Type
All alternatives from ISO/IEC 9075-2	
CONDITION_IDENTIFIER	character varying (*L*)†
† **Where** *L* is an implementation-defined integer not less than 128.	

Access Rules

No additional Access Rules.

General Rules

1) Replace GR 1) Case:

   a) If <which area> specifies CURRENT, then let *DA* be the first diagnostics area.

   b) If <which area> specifies STACKED, then let *DA* be the second diagnostics area.

2) Insert after GR 1) If <which area> specifies STACKED and no condition handler is activated, then an exception condition is raised: *diagnostics exception — stacked diagnostics accessed without active handler*.

3) *Table 2, "SQL-statement codes", modifies Table 31, "SQL-statement codes", in ISO/IEC 9075-2.*

Table 2 — SQL-statement codes

SQL-statement	Identifier	Code
All alternatives from ISO/IEC 9075-2		
<assignment statement>	ASSIGNMENT	5
<case statement>	CASE	86
<compound statement>	BEGIN END	12
<drop module statement>	DROP MODULE	28
<for statement>	FOR	46
<if statement>	IF	88
<iterate statement>	ITERATE	102
<leave statement>	LEAVE	89

SQL-statement	Identifier	Code
<loop statement>	LOOP	90
<resignal statement>	RESIGNAL	91
<repeat statement>	REPEAT	95
<signal statement>	SIGNAL	92
<SQL-server module definition>	CREATE MODULE	51
<while statement>	WHILE	97

4) Insert before GR 4)n) If the value of the RETURNED_SQLSTATE corresponds to *unhandled user-defined exception*, then the value of CONDITION_IDENTIFIER is the <condition name> of the user-defined exception.

5) Insert before GR 4)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 Subclause 16.2, "<signal statement>", and Subclause 16.3, "<resignal statement>", in this part of ISO/IEC 9075.

Conformance Rules

1) Without Feature P007, "Enhanced diagnostics management", conforming SQL language shall not contain a <which area>.

16.2 <signal statement>

Function

Signal an exception condition.

Format

```
<signal statement> ::=
    SIGNAL <signal value> [ <set signal information> ]

<signal value> ::=
    <condition name>
  | <sqlstate value>

<set signal information> ::= SET <signal information item list>

<signal information item list> ::=
    <signal information item> [ { <comma> <signal information item> }... ]

<signal information item> ::=
    <condition information item name> <equals operator> <simple value specification>
```

Syntax Rules

1) Case:

   a) If <signal value> immediately contains <condition name>, then:

      i) Let *CN* be the <condition name> contained in the <signal statement>.

      ii) The <signal statement> shall be contained in the scope of a <condition name> equivalent to *CN*. Let *CNS* be the innermost such scope.

   b) Otherwise, let *C* be the SQLSTATE value defined by <sqlstate value> and let *CN* be a zero-length string.

2) <condition information item name> shall specify CLASS_ORIGIN, SUBCLASS_ORIGIN, CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME, CATALOG_NAME, SCHEMA_NAME, TABLE_NAME, COLUMN_NAME, CURSOR_NAME, or MESSAGE_TEXT. No alternative for <condition information item name> shall be specified more than once in <set signal information>.

3) The data type of a <condition information item name> contained in <signal information item> shall be the data type specified in Table 1, "<identifier>s for use with <get diagnostics statement>".

Access Rules

None.

General Rules

1) The first diagnostics area is emptied and the following changes are made to statement information items in the first diagnostics area:

   a) NUMBER is set to 1 (one).

   b) MORE is set to 'N'.

   c) COMMAND_FUNCTION is set to 'SIGNAL'.

   d) DYNAMIC_FUNCTION is set to a zero-length string.

2) Let *CA* be the first condition area in the first diagnostics area. The condition information item CONDITION_IDENTIFIER in *CA* is set to *CN*.

   Case:

   a) If *CN* is considered to be defined for an SQLSTATE value, then the condition information item RETURNED_SQLSTATE in *CA* is set to the value for which *CN* is considered to be defined within *CNS*.

      NOTE 25 — "considered to be defined for" is defined in Subclause 13.3, "<condition declaration>".

   b) If <sqlstate value> is specified, then the condition information item RETURNED_SQLSTATE in *CA* is set to <sqlstate value>.

   c) Otherwise, the condition information item RETURNED_SQLSTATE in *CA* is set to a zero-length string.

3) The condition information items CLASS_ORIGIN, SUBCLASS_ORIGIN, CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME, CATALOG_NAME, SCHEMA_NAME, TABLE_NAME, COLUMN_NAME, CURSOR_NAME, and MESSAGE_TEXT in *CA* are set to a zero-length string. The condition information items MESSAGE_LENGTH and MESSAGE_OCTET_LENGTH in *CA* are set to 0 (zero).

4) The General Rules of Subclause 22.2, "Pushing and popping the diagnostics area stack", in ISO/IEC 9075-2, are applied with "PUSH" as *OPERATION* and the diagnostics area stack as *STACK*.

5) Condition handling mode becomes in effect in the SQL-session.

6) If <set signal information> is specified, then let *SSI* be the <set signal information>. Otherwise, let *SSI* be a zero-length string. The following <resignal statement> is effectively executed without further Syntax Rule checking:

```
RESIGNAL SSI
```

Conformance Rules

1) Without Feature P002, "Computational completeness", conforming SQL language shall not contain a <signal statement>.

16.3 <resignal statement>

Function

Resignal an exception condition.

Format

```
<resignal statement> ::=
    RESIGNAL [ <signal value> ] [ <set signal information> ]
```

Syntax Rules

1) Let *RS* be the <resignal statement>.

2) If <signal value> is specified, then

   Case:

   a) If <signal value> immediately contains <condition name>, then:

      i) Let *CN* be the <condition name> contained in *RS*.

      ii) The <resignal statement> shall be contained within the scope a <condition name> equivalent to *CN*. Let *CNS* be the innermost such scope.

   b) Otherwise, let *C* be the SQLSTATE value defined by <sqlstate value> and let *CN* be a zero-length string.

Access Rules

None.

General Rules

1) If condition handling mode is not in effect, then an exception condition is raised: *resignal when handler not active*.

2) The General Rules of Subclause 22.2, "Pushing and popping the diagnostics area stack", in ISO/IEC 9075-2, are applied with "POP" as *OPERATION* and the diagnostics area stack as *STACK*.

3) Let *DA* be the first diagnostics area, let *SA* be the statement area in *DA*, and let *CA* be the first condition area in *DA*.

4) If <set signal information> is specified, then for each <signal information item> in <set signal information>:

   a) The condition information item in *CA* identified by the <condition information item name> is set to the value of the <simple value specification>.

b) If the <condition information item name> specifies MESSAGE_TEXT, then the condition information items MESSAGE_LENGTH and MESSAGE_OCTET_LENGTH in *CA* are set to contain the length in characters and the length in octets of the value of the <simple value specification>, respectively.

5) If <signal value> is specified, then:

a) The statement information item NUMBER in *SA* is incremented.

b) If the maximum number of condition areas for diagnostics areas is exceeded, then the statement information item MORE in *SA* is set to 'Y' and the last condition area in *DA* is made vacant.

c) All occupied condition areas in *DA* are stacked such that the *i*-th condition area is placed at the position of the *i*+1-st condition area in *DA*.

d) The statement information items COMMAND_FUNCTION, DYNAMIC_FUNCTION, and CONDITION_IDENTIFIER in *SA* are set to 'RESIGNAL', a zero-length string, and *CN*, respectively.

Case:

i) If *CN* is considered to be defined for an SQLSTATE value, then the condition information item RETURNED_SQLSTATE in *CA* is set to the SQLSTATE value for which *CN* is considered to be defined within *CNS*.

NOTE 26 — "considered to be defined" is defined in Subclause 13.3, "<condition declaration>".

ii) If <sqlstate value> is specified, then the condition information item RETURNED_SQLSTATE in *CA* is set to <sqlstate value>.

iii) Otherwise, the condition information item RETURNED_SQLSTATE in *CA* is set to a zero-length string.

6) Case:

a) If the condition information item RETURNED_SQLSTATE in *CA* is not the empty string, then:

i) Let *S* be that value.

ii) If a handler *H* is the most appropriate handler for *S*, then *H* is activated.

iii) If no handler is activated and *S* identifies an SQLSTATE value associated with an exception condition, then this is an *unhandled exception condition* and the <SQL procedure statement> that resulted in execution of *RS* is terminated with this exception condition.

NOTE 27 — If *S* identifies an SQLSTATE value associated with a completion condition, then this is an *unhandled completion condition* and processing continues without altering the flow of control.,

b) Otherwise:

i) Let *E* be the value of the CONDITION_IDENTIFIER item in *CA*.

ii) If a handler *H* is the most appropriate handler for *E*, then *H* is activated.

iii) If there is no appropriate handler for *E*, then this is an *unhandled exception condition* and the <SQL procedure statement> that resulted in execution of *RS* is terminated with the exception condition *unhandled user-defined exception*.

Conformance Rules

1) Without Feature P002, “Computational completeness”, conforming SQL language shall not contain a <resignal statement>.

17 Information Schema

This Clause modifies Clause 5, "Information Schema", in ISO/IEC 9075-11.

17.1 MODULE_COLUMN_USAGE view

Function

Identify the columns owned by a given user on which SQL-server modules defined in this catalog are dependent.

Definition

```
CREATE VIEW MODULE_COLUMN_USAGE AS
    SELECT ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME,
           TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME
    FROM DEFINITION_SCHEMA.MODULE_COLUMN_USAGE
      JOIN
         DEFINITION_SCHEMA.SCHEMATA S
      ON ( ( TABLE_CATALOG, TABLE_SCHEMA ) =
           ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
    WHERE ( SCHEMA_OWNER = CURRENT_USER
          OR
            SCHEMA_OWNER IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES ) )
       AND
          MODULE_CATALOG =
          ( SELECT CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE MODULE_COLUMN_USAGE
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMATION_SCHEMA . MODULE_COLUMN_USAGE.

2) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMATION_SCHEMA . MODULE_COLUMN_USAGE.

3) Without Feature P003, "Information Schema views", conforming SQL language shall not reference INFORMATION_SCHEMA . MODULE_COLUMN_USAGE.

17.2 MODULE_PRIVILEGES view

Function

Identify the privileges on SQL-server modules defined in this catalog that are available to or granted by a given user.

Definition

```
CREATE VIEW MODULE_PRIVILEGES AS
    SELECT
      GRANTOR, GRANTEE, MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME,
      PRIVILEGE_TYPE, IS_GRANTABLE
    FROM DEFINITION_SCHEMA.MODULE_PRIVILEGES
    WHERE ( GRANTEE IN
            ( 'PUBLIC', CURRENT_USER )
          OR
            GRANTEE = CURRENT_USER )
        AND
          MODULE_CATALOG =
          ( SELECT CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE MODULE_PRIVILEGES
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F231, “Privilege tables”, conforming SQL language shall not reference INFORMATION_SCHEMA . MODULE_PRIVILEGES.

2) Without Feature P003, “Information Schema views”, conforming SQL language shall not reference INFORMATION_SCHEMA . MODULE_PRIVILETES.

17.3 MODULE_TABLE_USAGE view

Function

Identify the tables owned by a given user on which SQL-server modules defined in this catalog are dependent.

Definition

```
CREATE VIEW MODULE_TABLE_USAGE AS
    SELECT MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME,
           TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
    FROM DEFINITION_SCHEMA.MODULE_TABLE_USAGE
      JOIN
         DEFINITION_SCHEMA.SCHEMATA S
      ON ( ( TABLE_CATALOG, TABLE_SCHEMA ) =
           ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
    WHERE ( SCHEMA_OWNER = CURRENT_USER
          OR
            SCHEMA_OWNER IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES ) )
       AND
          MODULE_CATALOG =
          ( SELECT CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE MODULE_TABLE_USAGE
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F341, “Usage tables”, conforming SQL language shall not reference INFORMATION_SCHEMA . MODULE_TABLE_USAGE.

2) Without Feature P003, “Information Schema views”, conforming SQL language shall not reference INFORMATION_SCHEMA . MODULE_TABLE_USAGE.

17.4 MODULES view

Function

Identify the SQL-server modules in this catalog that are accessible to a given user.

Definition

```
CREATE VIEW MODULES AS
    SELECT MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME,
           DEFAULT_CHARACTER_SET_CATALOG, DEFAULT_CHARACTER_SET_SCHEMA,
           DEFAULT_CHARACTER_SET_NAME,
           DEFAULT_SCHEMA_CATALOG, DEFAULT_SCHEMA_NAME,
           CASE
             WHEN EXISTS (
               SELECT *
               FROM DEFINITION_SCHEMA.SCHEMATA AS S
               WHERE ( MODULE_CATALOG, MODULE_SCHEMA ) =
                     ( S.CATALOG_NAME, S.SCHEMA_NAME )
                 AND
                     ( S.SCHEMA_OWNER = CURRENT_USER
                   OR
                     S.SCHEMA_OWNER IN
                     ( SELECT ROLE_NAME
                       FROM ENABLED_ROLES ) ) )
             THEN MODULE_DEFINITION
             ELSE NULL
           END AS MODULE_DEFINITION,
           MODULE_AUTHORIZATION, SQL_PATH, MODULE_CREATED, MODULE_LAST_ALTERED
    FROM DEFINITION_SCHEMA.MODULES
    WHERE ( MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME ) IN
            ( SELECT MP.MODULE_CATALOG, MP.MODULE_SCHEMA, MP.MODULE_NAME
              FROM DEFINITION_SCHEMA.MODULE_PRIVILEGES AS MP
              WHERE ( MP.GRANTEE IN
                      ( 'PUBLIC', CURRENT_USER )
                  OR
                    MP.GRANTEE IN
                    ( SELECT ROLE_NAME
                      FROM ENABLED_ROLES ) ) )
      AND
          MODULE_CATALOG =
          ( SELECT CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE MODULES
    TO PUBLIC WITH GRANT OPTION;
```


Conformance Rules

1) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA . MODULES.

2) Without Feature P003, “Information Schema views”, conforming SQL language shall not reference INFORMATION_SCHEMA . MODULES.

3) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION_SCHEMA.MODULES.MODULE_CREATED.

4) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION_SCHEMA.MODULES.MODULE_LAST_ALTERED.

17.5 PARAMETERS view

This Subclause modifies Subclause 5.34, "PARAMETERS view", in ISO/IEC 9075-11.

Function

Identify the SQL parameters of SQL-invoked routines defined in this catalog that are accessible to a given user or role.

Definition

Replace the outermost <where condition> of the <view definition> with the following:

```
WHERE ( ( ( MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME ) IS NULL
        AND
          ( P1.SPECIFIC_CATALOG, P1.SPECIFIC_SCHEMA, P1.SPECIFIC_NAME ) IN
          ( SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME
            FROM DEFINITION_SCHEMA.ROUTINE_PRIVILEGES
            WHERE ( GRANTEE IN
                    ( 'PUBLIC', CURRENT_USER )
                  OR
                    GRANTEE IN
                    ( SELECT ROLE_NAME
                      FROM ENABLED_ROLES ) ) ) )
      OR
        ( ( MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME ) IS NOT NULL
        AND
          ( MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME ) IN
          ( SELECT MP.MODULE_CATALOG, MP.MODULE_SCHEMA, MP.MODULE_NAME
            FROM DEFINITION_SCHEMA.MODULE_PRIVILEGES AS MP
            WHERE ( MP.GRANTEE IN
                    ( 'PUBLIC', CURRENT_USER )
                  OR
                    MP.GRANTEE IN
                    ( SELECT ROLE_NAME
                      FROM ENABLED_ROLES ) ) ) ) )
  AND P1.SPECIFIC_CATALOG
    = ( SELECT CATALOG_NAME
        FROM INFORMATION_SCHEMA_CATALOG_NAME );
```

Conformance Rules

No additional Conformance Rules.

17.6 ROLE_MODULE_GRANTS view

Function

Identify the privileges on SQL-server modules defined in this catalog that are available to or granted by the currently enabled roles.

Definition

```
CREATE VIEW ROLE_MODULE_GRANTS AS
    SELECT GRANTOR, GRANTEE, MODULE_CATALOG,
           MODULE_SCHEMA, MODULE_NAME, PRIVILEGE_TYPE,
           IS_GRANTABLE
    FROM DEFINITION_SCHEMA.MODULE_PRIVILEGES
    WHERE ( GRANTEE IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES )
          OR
            GRANTOR IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES ) )
       AND
          MODULE_CATALOG =
          ( SELECT CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE ROLE_MODULE_GRANTS
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature P003, “Information Schema views”, conforming SQL language shall not reference INFORMATION_SCHEMA . ROLE_MODULE_GRANTS.

2) Without Feature T322, “Extended Roles”, conforming SQL language shall not reference INFORMATION_SCHEMA . ROLE_MODULE_GRANTS.

17.7 ROUTINES view

This Subclause modifies Subclause 5.48, "ROUTINES view", in ISO/IEC 9075-11.

Function

Identify the SQL-invoked routines in this catalog that are accessible to a given user or role.

Definition

Replace the outermost <where condition> of the <view definition> with the following:

```
WHERE ( ( ( MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME) IS NULL
          AND
          ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME ) IN
            ( SELECT SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME
              FROM DEFINITION_SCHEMA.ROUTINE_PRIVILEGES
              WHERE ( GRANTEE IN
                      ( 'PUBLIC', CURRENT_USER )
                    OR
                      GRANTEE IN
                      ( SELECT ROLE_NAME
                        FROM ENABLED_ROLES ) ) ) )
      OR
        ( ( MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME) IS NOT NULL
          AND
          ( MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME ) IN
            ( SELECT MP.MODULE_CATALOG, MP.MODULE_SCHEMA, MP.MODULE_NAME
              FROM DEFINITION_SCHEMA.MODULE_PRIVILEGES AS MP
              WHERE ( MP.GRANTEE IN
                      ( 'PUBLIC', CURRENT_USER )
                    OR
                      MP.GRANTEE IN
                      ( SELECT ROLE_NAME
                        FROM ENABLED_ROLES ) ) ) ) )
  AND SPECIFIC_CATALOG
    = ( SELECT CATALOG_NAME
        FROM INFORMATION_SCHEMA_CATALOG_NAME );
```

Conformance Rules

No additional Conformance Rules.

17.8 Short name views

This Subclause modifies Subclause 5.77, "Short name views", in ISO/IEC 9075-2.

Function

Provide alternative views that use only identifiers that do not require Feature F391, "Long identifiers".

Definition

```
CREATE VIEW MODULE_COL_USAGE
          ( ROUTINE_CATALOG,    ROUTINE_SCHEMA,    ROUTINE_NAME,
            TABLE_CATALOG,      TABLE_SCHEMA,      TABLE_NAME,
            COLUMN_NAME) AS
    SELECT ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME,
           TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
           COLUMN_NAME
    FROM INFORMATION_SCHEMA.MODULE_COLUMN_USAGE;

GRANT SELECT ON TABLE MODULE_COL_USAGE
    TO PUBLIC WITH GRANT OPTION;


CREATE VIEW MODULES_S
          ( MODULE_CATALOG,       MODULE_SCHEMA,       MODULE_NAME,
            DEF_CHAR_SET_CAT,     DEF_CHAR_SET_SCH,    DEF_CHAR_SET_NAME,
            DEF_SCHEMA_CATALOG,   DEFAULT_SCHEMA,      MODULE_DEFINITION,
            MODULE_AUTH,          SQL_PATH,            CREATED,
            ALTERED) AS
    SELECT MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME,
           DEFAULT_CHARACTER_SET_CATALOG, DEFAULT_CHARACTER_SET_SCHEMA,
           DEFAULT_CHARACTER_SET_NAME,
           DEFAULT_SCHEMA_CATALOG, DEFAULT_SCHEMA, MODULE_DEFINITION,
           MODULE_AUTHORIZATION, SQL_PATH, MODULE_CREATED,
           MODULE_LAST_ALTERED
    FROM INFORMATION_SCHEMA.MODULES;

GRANT SELECT ON TABLE MODULES_S
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMATION_SCHEMA . MODULE_COL_USAGE.

2) Without Feature P003, "Information Schema views", conforming SQL language shall not reference INFORMATION_SCHEMA . MOD_COL_USAGE.

3) Without Feature P003, “Information Schema views”, conforming SQL language shall not reference INFORMATION_SCHEMA . MODULES_S.

4) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION_SCHEMA . MODULES_S . CREATED.

5) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION_SCHEMA . MODULES_s . ALTERED.

18 Definition Schema

This Clause modifies Clause 6, "Definition Schema", in ISO/IEC 9075-11.

18.1 MODULE_COLUMN_USAGE base table

Function

The MODULE_COLUMN_USAGE table has one row for each column of a table that is explicitly or implicitly identified in the <query expression> of the view being described.

Definition

```
CREATE TABLE MODULE_COLUMN_USAGE (
    MODULE_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    MODULE_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    MODULE_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_CATALOG       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    COLUMN_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CONSTRAINT MODULE_COLUMN_USAGE_PRIMARY_KEY
      PRIMARY KEY ( MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME,
                    TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME ),
    CONSTRAINT MODULE_COLUMN_USAGE_CHECK_REFERENCES_COLUMNS
      CHECK ( TABLE_CATALOG <>
              ANY ( SELECT CATALOG_NAME
                    FROM SCHEMATA )
           OR
             ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME ) IN
             ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME
               FROM COLUMNS ) ),
    CONSTRAINT MODULE_COLUMN_USAGE_FOREIGN_KEY_MODULES
      FOREIGN KEY ( MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME )
        REFERENCES MODULES,
    CONSTRAINT MODULE_COLUMN_USAGE_CHECK_MODULE_TABLE_USAGE
      CHECK ( MODULE_CATALOG <>
              ANY ( SELECT MODULE_CATALOG
                                  FROM SCHEMATA )
            OR
              ( MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME,
                TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) IN
                ( SELECT MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME
                         TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
                  FROM MODULE_TABLE_USAGE ) )
```

)

Description

1) The values of MODULE_CATALOG, MODULE_SCHEMA, and MODULE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the SQL-server module being described.

2) The values of TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, and COLUMN_NAME are the catalog name, unqualified schema name, qualified identifier, and identifier respectively, of a column that is referenced in the SQL-server module being described.

18.2 MODULE_PRIVILEGES base table

Function

The MODULE_PRIVILEGES table has one row for each execute privilege descriptor on an SQL-server module. It effectively contains a representation of the execute privilege descriptors.

Definition

```
CREATE TABLE MODULE_PRIVILEGES (
      GRANTOR                INFORMATION_SCHEMA.SQL_IDENTIFIER,
      GRANTEE                INFORMATION_SCHEMA.SQL_IDENTIFIER,
      MODULE_CATALOG         INFORMATION_SCHEMA.SQL_IDENTIFIER,
      MODULE_SCHEMA          INFORMATION_SCHEMA.SQL_IDENTIFIER,
      MODULE_NAME            INFORMATION_SCHEMA.SQL_IDENTIFIER,
      PRIVILEGE_TYPE         INFORMATION_SCHEMA.CHARACTER_DATA
        CONSTRAINT MODULE_PRIVILEGES_TYPE_CHECK
          CHECK ( PRIVILEGE_TYPE = 'EXECUTE' ),
      IS_GRANTABLE           INFORMATION_SCHEMA.CHARACTER_DATA
        CONSTRAINT MODULE_PRIVILEGES_GRANTABLE_NOT_NULL
            NOT NULL
        CONSTRAINT MODULE_PRIVILEGES_GRANTABLE_CHECK
          CHECK ( IS_GRANTABLE
                  IN ( 'YES', 'NO' ) ),
      CONSTRAINT MODULE_PRIVILEGES_PRIMARY_KEY
        PRIMARY KEY ( GRANTOR, GRANTEE, MODULE_CATALOG, MODULE_SCHEMA,
                      MODULE_NAME ),
      CONSTRAINT MODULE_PRIVILEGES_FOREIGN_KEY_TABLES
        FOREIGN KEY ( MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME )
          REFERENCES MODULES,
    CONSTRAINT MODULE_PRIVILEGE_GRANTOR_CHECK
      CHECK ( GRANTOR IN
              ( SELECT ROLE_NAME
                FROM ROLES )
            OR
              GRANTOR IN
              ( SELECT USER_NAME
                FROM USERS ) ),
    CONSTRAINT MODULE_PRIVILEGE_GRANTEE_CHECK
      CHECK ( GRANTEE IN
              ( SELECT ROLE_NAME
                FROM ROLES )
            OR
              GRANTEE IN
              ( SELECT USER_NAME
                FROM USERS ) )
    )
```

Description

1) The value of GRANTOR is the <authorization identifier> of the user or role who granted execute privileges, on the SQL-server module identified by MODULE_CATALOG, MODULE_SCHEMA, and MODULE_NAME, to the user or role identified by the value of GRANTEE for the privilege being described.

2) The value of GRANTEE is the <authorization identifier> of some user or role, or "PUBLIC" to indicate all users, to whom the privilege being described is granted.

3) The values of MODULE_CATALOG, MODULE_SCHEMA, and MODULE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the SQL-server module on which the privilege being described has been granted.

4) The values of PRIVILEGE_TYPE have the following meanings:

EXECUTE	The user has EXECUTE privilege on the SQL-server module identified by MODULE_CATALOG, MODULE_SCHEMA, and MODULE_NAME.

5) The values of IS_GRANTABLE have the following meanings:

YES	The privilege being described was granted WITH GRANT OPTION and is thus grantable.
NO	The privilege being described was not granted WITH GRANT OPTION and is thus not grantable.

18.3 MODULE_TABLE_USAGE base table

Function

The MODULE_TABLE_USAGE table has one row for each table identified by a <table name> simply contained in a <table reference> that is contained in the <query expression> of a view.

Definition

```
CREATE TABLE MODULE_TABLE_USAGE (
    MODULE_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    MODULE_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    MODULE_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_CATALOG       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_SCHEMA        INFORMATION_SCHEMA.SQL_IDENTIFIER,
    TABLE_NAME          INFORMATION_SCHEMA.SQL_IDENTIFIER,
    CONSTRAINT MODULE_TABLE_USAGE_PRIMARY_KEY
      PRIMARY KEY ( MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME,
                    TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ),
    CONSTRAINT MODULE_TABLE_USAGE_CHECK_REFERENCES_TABLES
      CHECK ( TABLE_CATALOG <>
              ANY ( SELECT CATALOG_NAME
                    FROM SCHEMATA )
            OR
              ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) IN
                ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
                  FROM TABLES ) ),
    CONSTRAINT MODULE_TABLE_USAGE_FOREIGN_KEY_MODULES
      FOREIGN KEY ( MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME )
       REFERENCES MODULES
    )
```

Description

1) The values of MODULE_CATALOG, MODULE_SCHEMA, and MODULE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the SQL-server module being described.

2) The values of TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of a table that is referenced in the SQL-server module being described.

18.4 MODULES base table

Function

The MODULES base table has one row for each SQL-server module.

Definition

```
CREATE TABLE MODULES (
    MODULE_CATALOG              INFORMATION_SCHEMA.SQL_IDENTIFIER,
    MODULE_SCHEMA               INFORMATION_SCHEMA.SQL_IDENTIFIER,
    MODULE_NAME                 INFORMATION_SCHEMA.SQL_IDENTIFIER,
    DEFAULT_CHARACTER_SET_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT MODULES_DEFAULT_CHARACTER_SET_CATALOG_NOT_NULL
        NOT NULL,
    DEFAULT_CHARACTER_SET_SCHEMA  INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT MODULES_DEFAULT_CHARACTER_SET_SCHEMA_NOT_NULL
        NOT NULL,
    DEFAULT_CHARACTER_SET_NAME    INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT MODULES_DEFAULT_CHARACTER_SET_NAME_NOT_NULL
        NOT NULL,
    DEFAULT_SCHEMA_CATALOG        INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT MODULES_DEFAULT_SCHEMA_CATALOG_NOT_NULL
        NOT NULL,
    DEFAULT_SCHEMA_NAME           INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT MODULES_DEFAULT_SCHEMA_NAME_NOT_NULL
        NOT NULL,
    MODULE_DEFINITION             INFORMATION_SCHEMA.CHARACTER_DATA,
    MODULE_AUTHORIZATION          INFORMATION_SCHEMA.SQL_IDENTIFIER
      CONSTRAINT AUTHORIZATION_FOREIGN_KEY_USERS REFERENCES USERS,
    SQL_PATH                      INFORMATION_SCHEMA.CHARACTER_DATA,
    CREATED                       INFORMATION_SCHEMA.TIME_STAMP,
    LAST_ALTERED                  INFORMATION_SCHEMA.TIME_STAMP,
    CONSTRAINT MODULES_PRIMARY_KEY
      PRIMARY KEY ( MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME ),
    CONSTRAINT MODULES_FOREIGN_KEY_SCHEMATA
      FOREIGN KEY ( MODULE_CATALOG, MODULE_SCHEMA )
        REFERENCES SCHEMATA,
    CONSTRAINT MODULES_FOREIGN_KEY_CHARACTER_SETS
      FOREIGN KEY ( DEFAULT_CHARACTER_SET_CATALOG, DEFAULT_CHARACTER_SET_SCHEMA,
                    DEFAULT_CHARACTER_SET_NAME )
        REFERENCES CHARACTER_SETS,
    CONSTRAINT MODULES_FOREIGN_KEY_DEFAULT_SCHEMA_SCHEMATA
      FOREIGN KEY ( DEFAULT_SCHEMA_CATALOG, DEFAULT_SCHEMA_NAME )
        REFERENCES SCHEMATA
    )
```


Description

1) The values of MODULE_CATALOG, MODULE_SCHEMA, and MODULE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the module name of the SQL-server module being described.

2) The values of DEFAULT_CHARACTER_SET_CATALOG, DEFAULT_CHARACTER_SET_SCHEMA, and DEFAULT_CHARACTER_SET_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the character set identified by the implicit or explicit <SQL-server module character set specification>.

3) The values of DEFAULT_SCHEMA_CATALOG, and DEFAULT_SCHEMA_NAME are the catalog name, and unqualified schema name, respectively, of the schema identified by the implicit or explicit <SQL-server module schema clause>.

4) Case:

   a) If the character representation of the <SQL-server module definition> that defined the SQL-server module being described can be represented without truncation, then the value of MODULE_DEFINITION is that character representation.

   b) Otherwise, the value of MODULE_DEFINITION is the null value.

   NOTE 28 — Any implicit <column reference>s that were contained in the <SQL-server module definition> are replaced by explicit <column reference>s in MODULE_DEFINITION.

5) Case:

   a) If AUTHORIZATION was specified in <module authorization clause> in the SQL-server module being described, then the value of MODULE_AUTHORIZATION is <module authorization identifier>.

   b) Otherwise, the value of MODULE_AUTHORIZATION is the null value.

6) Case:

   a) If <SQL-server module path specification> was specified in the <SQL-server module definition> that defined the SQL-server module described by this row and the character representation of the <SQL-server module path specification> can be represented without truncation, then the value of SQL_PATH is that character representation.

   b) Otherwise, the value of SQL_PATH is the null value.

7) The value of CREATED is the value of CURRENT_TIMESTAMP at the time when the SQL-server module being described was created.

8) The value of LAST_ALTERED is the value of CURRENT_TIMESTAMP at the time that the SQL-server module being described was last altered. This value is identical to the value of CREATED for SQL-server modules that have never been altered.

18.5 ROUTINES base table

This Subclause modifies Subclause 6.40, "ROUTINES base table", in ISO/IEC 9075-11.

Function

The ROUTINES base table has one row for each SQL-invoked routine.

Definition

Insert this constraint immediately before constraint ROUTINES_FOREIGN_KEY_USER_DEFINED_TYPES

```
CONSTRAINT ROUTINES_FOREIGN_KEY_MODULES
  FOREIGN KEY ( MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME )
  REFERENCES MODULES
    MATCH FULL,
```

Description

1) Replace Desc. 3) Case:

   a) If the SQL-invoked routine being described was defined in an SQL-server module, then the values of MODULE_CATALOG, MODULE_SCHEMA, and MODULE_NAME are the fully qualified module name of this SQL-server module.

   b) Otherwise, the values of MODULE_CATALOG, MODULE_SCHEMA, and MODULE_NAME are the null value.

19 Status codes

This Clause modifies Clause 23, "Status codes", in ISO/IEC 9075-2.

19.1 SQLSTATE

This Subclause modifies Subclause 23.1, "SQLSTATE", in ISO/IEC 9075-2.

Table 3, "SQLSTATE class and subclass values", modifies Table 32, "SQLSTATE class and subclass values", in ISO/IEC 9075-2.

Table 3 — SQLSTATE class and subclass values

Category	Condition	Class	Subcondition	Subclass
	All alternatives from ISO/IEC 9075-2			
X	*case not found for case statement*	20	(*no subclass*)	000
X	*data exception*	22	(*no subclass*)	000
			null value in field reference	02A
X	*diagnostics exception*	0Z	(*no subclass*)	000
			stacked diagnostics accessed without active handler	002
X	*resignal when handler not active*	0K	(*no subclass*)	000
W	*warning*	01	(*no subclass*)	000
X	*unhandled user-defined exception*	45	(*no subclass*)	000

This page intentionally left blank.

20 Conformance

20.1 Claims of conformance to SQL/PSM

In addition to the requirements of ISO/IEC 9075-1, Clause 8, "Conformance", a claim of conformance to this part of ISO/IEC 9075 shall:

1) Claim conformance to Feature P002, "Computational completeness".

20.2 Additional conformance requirements for SQL/PSM

There are no additional conformance requirements for this part of ISO/IEC 9075.

20.3 Implied feature relationships of SQL/PSM

Table 4 — Implied feature relationships of SQL/PSM

Feature ID	Feature Name	Implied Feature ID	Implied Feature Name
P001	Stored modules	P002	Computational completeness
P003	Information Schema views	P001	Stored modules
P004	Extended CASE statement	P002	Computational completeness
P005	Qualified SQL variable references	P002	Computational completeness
P006	Multiple assignment	P002	Computational completeness
P007	Enhanced diagnostics management	P002	Computational completeness

This page intentionally left blank.

Annex A

(informative)

SQL Conformance Summary

This Annex modifies Annex A, "SQL Conformance Summary", in ISO/IEC 9075-2.

The contents of this Annex summarizes all Conformance Rules, ordered by Feature ID and by Subclause.

1) Specifications for Feature F231, "Privilege tables":
   a) Subclause 17.2, "MODULE_PRIVILEGES view":
      i) Without Feature F231, "Privilege tables", conforming SQL language shall not reference INFORMATION_SCHEMA . MODULE_PRIVILEGES.
2) Specifications for Feature F341, "Usage tables":
   a) Subclause 17.1, "MODULE_COLUMN_USAGE view":
      i) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMATION_SCHEMA . MODULE_COLUMN_USAGE.
   b) Subclause 17.3, "MODULE_TABLE_USAGE view":
      i) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMATION_SCHEMA . MODULE_TABLE_USAGE.
   c) Subclause 17.8, "Short name views":
      i) Without Feature F341, "Usage tables", conforming SQL language shall not reference INFORMATION_SCHEMA . MODULE_COL_USAGE.
3) Specifications for Feature F391, "Long identifiers":
   a) Subclause 17.1, "MODULE_COLUMN_USAGE view":
      i) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMATION_SCHEMA . MODULE_COLUMN_USAGE.
   b) Subclause 17.4, "MODULES view":
      i) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMATION_SCHEMA . MODULES.
4) Specifications for Feature P001, "Stored modules":
   a) Subclause 9.18, "<SQL-server module definition>":
      i) Without Feature P001, "Stored modules", conforming SQL language shall not contain an <SQL-server module definition>.

b) Subclause 9.19, "<drop module statement>":

i) Without Feature P001, "Stored modules", conforming SQL language shall not contain a <drop module statement>.

c) Subclause 10.2, "<privileges>":

i) Without Feature P001, "Stored modules", conforming SQL language shall not contain a <privilege> of MODULE.

5) Specifications for Feature P002, "Computational completeness":

a) Subclause 13.1, "<compound statement>":

i) Without Feature P002, "Computational completeness", conforming SQL language shall not contain a <compound statement>.

b) Subclause 13.2, "<handler declaration>":

i) Without Feature P002, "Computational completeness", conforming SQL language shall not contain a <handler declaration>.

c) Subclause 13.3, "<condition declaration>":

i) Without Feature P002, "Computational completeness", conforming SQL language shall not contain a <condition declaration>.

d) Subclause 13.4, "<SQL variable declaration>":

i) Without Feature P002, "Computational completeness", conforming SQL language shall not contain a <SQL variable declaration>.

e) Subclause 13.5, "<assignment statement>":

i) Without Feature P002, "Computational completeness", conforming SQL language shall not contain a <assignment statement>.

f) Subclause 13.6, "<case statement>":

i) Without Feature P002, "Computational completeness", conforming SQL language shall not contain a <case statement>.

g) Subclause 13.7, "<if statement>":

i) Without Feature P002, "Computational completeness", conforming SQL language shall not contain a <if statement>.

h) Subclause 13.8, "<iterate statement>":

i) Without Feature P002, "Computational completeness", conforming SQL language shall not contain a <iterate statement>.

i) Subclause 13.9, "<leave statement>":

i) Without Feature P002, "Computational completeness", conforming SQL language shall not contain a <leave statement>.

j) Subclause 13.10, "<loop statement>":

i) Without Feature P002, "Computational completeness", conforming SQL language shall not contain a <loop statement>.

k) Subclause 13.11, "<while statement>":

i) Without Feature P002, "Computational completeness", conforming SQL language shall not contain a <while statement>.

l) Subclause 13.12, "<repeat statement>":

i) Without Feature P002, "Computational completeness", conforming SQL language shall not contain a <repeat statement>.

m) Subclause 13.13, "<for statement>":

i) Without Feature P002, "Computational completeness", conforming SQL language shall not contain a <for statement>.

n) Subclause 16.2, "<signal statement>":

i) Without Feature P002, "Computational completeness", conforming SQL language shall not contain a <signal statement>.

o) Subclause 16.3, "<resignal statement>":

i) Without Feature P002, "Computational completeness", conforming SQL language shall not contain a <resignal statement>.

6) Specifications for Feature P003, "Information Schema views":

a) Subclause 17.1, "MODULE_COLUMN_USAGE view":

i) Without Feature P003, "Information Schema views", conforming SQL language shall not reference INFORMATION_SCHEMA . MODULE_COLUMN_USAGE.

b) Subclause 17.2, "MODULE_PRIVILEGES view":

i) Without Feature P003, "Information Schema views", conforming SQL language shall not reference INFORMATION_SCHEMA . MODULE_PRIVILETES.

c) Subclause 17.3, "MODULE_TABLE_USAGE view":

i) Without Feature P003, "Information Schema views", conforming SQL language shall not reference INFORMATION_SCHEMA . MODULE_TABLE_USAGE.

d) Subclause 17.4, "MODULES view":

i) Without Feature P003, "Information Schema views", conforming SQL language shall not reference INFORMATION_SCHEMA . MODULES.

e) Subclause 17.6, "ROLE_MODULE_GRANTS view":

i) Without Feature P003, "Information Schema views", conforming SQL language shall not reference INFORMATION_SCHEMA . ROLE_MODULE_GRANTS.

f) Subclause 17.8, “Short name views”:

i) Without Feature P003, “Information Schema views”, conforming SQL language shall not reference INFORMATION_SCHEMA . MOD_COL_USAGE.

ii) Without Feature P003, “Information Schema views”, conforming SQL language shall not reference INFORMATION_SCHEMA . MODULES_S.

7) Specifications for Feature P004, “Extended CASE statement”:

a) Subclause 13.6, “<case statement>”:

i) Without Feature P004, “Extended CASE statement”, in conforming SQL language, both <case operand> immediately contained in a <simple case statement> and a <when operand> immediately contained in a <when operand list> immediately contained in a <simple case statement when clause> shall be a <row value predicand> that is a <row value constructor predicand> that is a single <common value expression> or <boolean value predicand>.

8) Specifications for Feature P005, “Qualified SQL variable references”:

a) Subclause 6.2, “<identifier chain>”:

i) Without Feature P005, “Qualified SQL variable references”, conforming SQL language shall not contain an SQL variable reference whose first <identifier> is the <beginning label> of a <compound statement>.

b) Subclause 7.1, “<query specification>”:

i) Without Feature P005, “Qualified SQL variable references”, conforming SQL language shall not contain an <asterisked identifier chain> whose first <identifier> is the <beginning label> of a <compound statement>.

9) Specifications for Feature P006, “Multiple assignment”:

a) Subclause 13.5, “<assignment statement>”:

i) Without Feature P006, “Multiple assignment”, conforming SQL language shall not contain a <multiple variable assignment>.

10) Specifications for Feature P007, “Enhanced diagnostics management”:

a) Subclause 16.1, “<get diagnostics statement>”:

i) Without Feature P007, “Enhanced diagnostics management”, conforming SQL language shall not contain a <which area>.

11) Specifications for Feature P008, “Comma-separated predicates in simple CASE statement”:

a) Subclause 13.6, “<case statement>”:

i) Without Feature P008, “Comma-separated predicates in simple CASE statement”, in conforming SQL language, a <when operand list> immediately contained in a <when operand list> immediately contained in a <simple case statement when clause> shall simply contain exactly one <when operand>.

12) Specifications for Feature T011, “Timestamp in Information Schema”:

a) Subclause 17.4, “MODULES view”:

i) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION_SCHEMA.MODULES.MODULE_CREATED.

ii) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION_SCHEMA.MODULES.MODULE_LAST_ALTERED.

b) Subclause 17.8, “Short name views”:

i) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION_SCHEMA . MODULES_S . CREATED.

ii) Without Feature T011, “Timestamp in Information Schema”, conforming SQL language shall not reference INFORMATION_SCHEMA . MODULES_s . ALTERED.

13) Specifications for Feature T051, “Row types”:

a) Subclause 13.5, “<assignment statement>”:

i) Without Feature T051, “Row types”, conforming SQL language shall not contain a <modified field reference>.

14) Specifications for Feature T322, “Extended Roles”:

a) Subclause 17.6, “ROLE_MODULE_GRANTS view”:

i) Without Feature T322, “Extended Roles”, conforming SQL language shall not reference INFORMATION_SCHEMA . ROLE_MODULE_GRANTS.

This page intentionally left blank.

Annex B

(informative)

Implementation-defined elements

This Annex modifies Annex B, "Implementation-defined elements", in ISO/IEC 9075-2.

This Annex references those features that are identified in the body of this part of ISO/IEC 9075 as implementation-defined.

1) Subclause 4.7, "Diagnostics area":

   a) An SQL-implementation places information about a completion or exception condition that causes a handler to be activated into the diagnostics area prior to activating the handler. If other conditions are raised, then it is implementation-defined whether the implementation places information about them into the diagnostics area.

2) Subclause 8.2, "<sqlstate value>":

   a) The implicit or explicit character set of the <character string literal> contained in <sqlstate value> shall be the implementation-defined character set in which SQLSTATE parameter values are returned.

   b) The value of the <character string literal> contained in <sqlstate value> may be composed of a standard SQLSTATE Class value for which an implementation-defined Subclass value is permitted and three characters with the form of an implementation-defined Subclass value.

   c) The value of the <character string literal> contained in <sqlstate value> may be composed of five characters of which the first two have the form of an implementation-defined Class value.

3) Subclause 9.18, "<SQL-server module definition>":

   a) If <SQL-server module path specification> is not specified, then an <SQL-server module path specification> containing an implementation-defined <schema name list> that includes the explicit or implicit <schema name> of the <SQL-server module name> is implicit.

This page intentionally left blank.

Annex C

(informative)

Implementation-dependent elements

This Annex modifies Annex C, "Implementation-dependent elements", in ISO/IEC 9075-2.

This Annex references those places where this part of ISO/IEC 9075 states explicitly that the actions of a conforming implementation are implementation-dependent.

1) Subclause 4.2, "Tables":

   a) The effective <schema name> of the <schema qualified name> of the declared local temporary table may be thought of as the implementation-dependent SQL-session identifier associated with the SQL-session and the name of the <SQL-server module definition> that contains the <temporary table declaration>.

2) Subclause 9.18, "<SQL-server module definition>":

   a) If the SQL-server module is actually represented in a character set other than the character set identified by the explicit or implicit <SQL-server module character set specification>, then the effects are implementation-dependent.

3) Subclause 12.4, "<select statement: single row>":

   a) The order of assignment of values to targets in the <select target list> is implementation-dependent.

4) Subclause 12.7, "<temporary table declaration>":

   a) If a <temporary table declaration> is contained in an <SQL-client module definition> without an intervening <SQL-server module definition>, then the implementation-dependent <schema name> is effectively derived from the implementation-dependent SQL-session identifier associated with the SQL-session and an implementation-dependent name associated with the SQL-client module that contains the <temporary table declaration>. Otherwise, the implementation-dependent <schema name> is effectively derived from the implementation-dependent SQL-session identifier associated with the SQL-session and the name associated of the <SQL-server module definition> that contains the <temporary table declaration>.

5) Subclause 13.1, "<compound statement>":

   a) The implicit <beginning label> of a <compound statement> with no explicit <beginning label> is implementation-dependent.

   b) The variables, cursors, and handlers specified in the <local declaration list>, the <local cursor declaration list>, and the <local handler declaration list> of a <compound statement> are created in an implementation-dependent order.

6) Subclause 13.10, "<loop statement>":

a) The implicit <beginning label> of a <loop statement> with no explicit <beginning label> is implementation-dependent.

7) Subclause 13.11, "<while statement>":

a) The implicit <beginning label> of a <while statement> with no explicit <beginning label> is implementation-dependent.

8) Subclause 13.12, "<repeat statement>":

a) The implicit <beginning label> of a <repeat statement> with no explicit <beginning label> is implementation-dependent.

9) Subclause 13.13, "<for statement>":

a) The implicit <beginning label> of a <for statement> with no explicit <beginning label> is implementation-dependent.

b) The <cursor name> used in the transformation of a <for statement> into a <while statement> is implementation-dependent, as are the <condition name> and the <SQL variable name> used in the <while statement> for getting diagnostics information.

Annex D

(informative)

Incompatibilities with ISO/IEC 9075:1999

This Annex modifies Annex E, "Incompatibilities with ISO/IEC 9075:1999", in ISO/IEC 9075-2.

This edition of this part of ISO/IEC 9075 introduces some incompatibilities with the earlier version of Database Language SQL as specified in ISO/IEC 9075:1999. Unless specified in this Annex, features and capabilities of Database Language SQL are compatible with the earlier version of ISO/IEC 9075.

1) In ISO/IEC 9075-4:1999, it was not permitted to use a <statement label> to qualify an <SQL variable reference>. This gives rise to an incompatibility with ISO/IEC 9075-4:200n in the case where an SQL variable's name is the same as the <beginning label> of the innermost <compound statement> in which it is declared, the declared type of that variable is such that it has components that can be referenced using "dot notation" and one of those components has the same name as one of the other SQL variables declared in that same <compound statement>.

   For example, if <compound statement> labeled CS simply contains the declaration of a variable named V and another variable named CS of type ROW(V INTEGER), then "CS.V" can be a reference to either the variable V or the field V of the variable CS and is thus a syntax error. In ISO/IEC 9075-4:1999, it unambiguously references the field.

2) Insert before list element LE 17) The definition of "possibly nullable" with regard to <routine invocation> has been tightened. If all subject routines of a <routine invocation> specify PARAMETER STYLE GENERAL, then the <routine invocation> is known not nullable. In ISO/IEC 9075-4:1996, all <routine invocation>s were regarded as possibly nullable.

3) Insert before list element LE 17) Some of the normative material previously specified in ISO/IEC 9075-4:1996 has been moved to ISO/IEC 9075-2. Although this will change any statement of conformance to this and other parts of ISO/IEC 9075, no incompatibilities other than those listed above have been introduced by the reorganization of this normative material.

This page intentionally left blank.

Annex E

(informative)

Defect reports not addressed in this edition of this part of ISO/IEC 9075

Each entry in this Annex describes a reported defect in the previous edition of this part of ISO/IEC 9075 that remains in this edition.

1) **Subclause 10.3, "<revoke statement>"**

   This subclause extends Subclause 12.7, "<revoke statement>", in ISO/IEC 9075-2, but fails to make any extension to SR 24) of that Subclause to take into account the fact that the <triggered action> of a trigger might be an SQL-control statement that includes (for example) a <scalar subquery> that references some schema object. Thus, the Syntax and General Rules of an SQL-procedure statement that would cause destruction of such a schema object do not necessarily take into account that some trigger might cease to be valid as a result of that destruction.

2) **Subclause 13.1, "<compound statement>"**:

   General Rule 3)c)ii)1) makes it possible for successful execution of a <compound statement> to leave open a result set cursor that is declared locally in that <compound statement>. In the particular case where the <compound statement> is contained in the <SQL procedure statement> of an <externally-invoked procedure>, it is not clear whether such a result set can be accessed by subsequently executing an <allocate cursor statement>, nor is it clear when such a result set is destroyed.

This page intentionally left blank.

Annex F

(informative)

SQL feature taxonomy

This Annex describes a taxonomy of features defined in this part of ISO/IEC 9075.

Table 5, "Feature taxonomy for optional features", contains a taxonomy of the optional features of the SQL language that are specified in this part of ISO/IEC 9075. In this table, the first column contains a counter that may be used to quickly locate rows of the table; these values otherwise have no use and are not stable — that is, they are subject to change in future editions of or even Technical Corrigenda to ISO/IEC 9075 without notice.

The column "Feature ID" column of this table specifies the formal identification of each feature and each sub-feature contained in the table.

The "Feature Name" column of this table contains a brief description of the feature or subfeature associated with the Feature ID value.

Table 5 — Feature taxonomy for optional features

	Feature ID	Feature Name
1	**P001**	**Stored modules**
2	P001-01	<SQL-server module definition>
3	P001-02	<drop module statement>
4	**P002**	**Computational completeness**
5	P002-01	<compound statement>
6	P002-02	<handler declaration>
7	P002-03	<condition declaration>
8	P002-04	<SQL variable declaration>
9	P002-05	<assignment statement>
10	P002-06	<case statement>
11	P002-07	<if statement>

	Feature ID	Feature Name
12	P002-08	<iterate statement>
13	P002-09	<leave statement>
14	P002-10	<loop statement>
15	P002-11	<repeat statement>
16	P002-12	<while statement>
17	P002-13	<for statement>
18	P002-14	<signal statement>
19	P002-15	<resignal statement>
20	P002-16	<control statement>s as the SQL-statement of an externally-invoked procedure
21	**P003**	**Information Schema views**
22	P003-01	MODULES view
23	P003-02	MODULE_TABLE_USAGE view
24	P003-03	MODULE_COLUMN_USAGE view
25	P003-04	MODULE_PRIVILEGES view
26	**P004**	**Extended CASE statement**
27	**P005**	**Qualified SQL variable references**
28	**P006**	**Multiple assignment**
29	**P007**	**Enhanced diagnostics management**
30	**P008**	**Comma-separated predicates in a CASE statement**

Table 5, "Feature taxonomy for optional features", does not provide definitions of the features; the definition of those features is found in the Conformance Rules that are further summarized in Annex A, "SQL Conformance Summary".

Index

Index entries appearing in **boldface** indicate the page where the word, phrase, or BNF nonterminal was defined; index entries appearing in *italics* indicate a page where the BNF nonterminal was used in a Format; and index entries appearing in roman type indicate a page where the word, phrase, or BNF nonterminal was used in a heading, Function, Syntax Rule, Access Rule, General Rule, Leveling Rule, Table, or other descriptive text.

— E —

— F —

— G —

— H —

— I —

— J —

— K —

— L —

— M —

— N —

— O —

— P —

— R —

— S —

— T —

— U —

— V —

— W —

1 Possible problems with SQL/PSM

I observe some possible problems with Persistent Stored Modules as defined in this document. These are noted below. Further contributions to this list are welcome. Deletions from the list (resulting from change proposals that correct the problems or from research indicating that the problems do not, in fact, exist) are even more welcome. Other comments may appear in the same list.

I have assigned "fixed" numbers to each possible problem. These numbers will not change from printing to printing, but will instead develop "gaps" between numbers as problems are solved.

Possible problems related to Persistent Stored Modules

Significant Possible Problems:

PSM-999 In the body of the Working Draft, I have occasionally highlighted a point that requires urgent attention thus:

****Editor's Note****
Text of the problem.

These items are indexed under "**Editor's Note**".

PSM-149 The following Possible Problem has been noted:

Severity: Major Technical
Reference: P04, SQL/PSM, Subclause 10.3, "<revoke statement>"
Note at: None.
Source: CD1-2000 comments USA-P04-005
Possible Problem:

Because PSM expands the possibilities of <SQL procedure statement>, the capabilities for the <triggered action> of a trigger are much increased. Consequently the rules regarding dependencies of a trigger on a privilege or schema object must be extended in PSM. For example, in 9075-2 (SQL/Foundation), Subclause 12.7, "<revoke statement>", SR 24) subrules g) through j) deal with when SELECT privilege is required to define a trigger. None of these rules cover the possibility of a <scalar subquery> in a <case statement>. Likewise the rules for SELECT WITH HIERARCHY OPTION are inadequate.

The commenter does not believe that the solution is to run around trying to find every case that is not currently covered. Instead, the commenter believes that we need a general mechanism that constructs a dependency graph relating arbitrary schema objects and privileges, so that as features and parts are added, each new feature or part need only specify its contribution to the dependency graph algorithm. For example, dependencies on privileges can be declared in the Access Rules, so that whenever an Access Rule is used, a dependency is automatically created. That way <revoke statement> would not need to duplicate information that is already implicit in the Access Rules. Similarly, dependencies on schema objects can be generated in the rules of <table reference>, <column reference>, etc. Then <revoke statement> and the drop statements would not need to generate dependencies, they could simply assume that they are defined.
Proposed Solution:

None provided with comment.

Editor's Notes for WG3:HBA-005 = H2-2003-308

Minor Problems and Wordsmithing Candidates:

Language Opportunities

PSM-088 In the course of discussing DBL:MCI-060, Steve Cannan noted the following Language Opportunity:

> Need some syntax to do an ALTER VIEW or similar to "rebind" subject routines, * column references, etc. for all objects that contain statically-bound references of any sort.

PSM-095 In the course of discussing DBL:MCI-132 ballot comments, Ed Dee noted the following Language Opportunity:

> FOR statements terminate (with a closed cursor exception) if the statement list of the <for statement> list contains a COMMIT or ROLLBACK. Further, no statement contained in the <for statement> can set any transaction attributes.
>
> It is desirable that an application programmer be able to initiate or terminate transactions within a <for statement>.

PSM-106 DBL:MCI-161, point 2.5, item 5, noted the following Language Opportunity:

> In Subclause 8.1, "<routine invocation>", the prohibitions on SQL-transaction staements and SQL-connection statements in SQL-invoked routines might be lifted, if a way can be found to make sure that SQL-invoked routines end SQL-sessions and SQL-transactions that they start, don't end SQL-transactions and SQL-sessions that they didn't start, and don't switch SQL-connections without restoring the SQL-connection with which they started.

PSM-107 Discussion of DBL:MCI-161, point 2.5, item 5, noted the following Language Opportunity:

> In Subclause 8.1, "<routine invocation>", the prohibitions on SQL-transaction statements and SQL-connection statements in SQL-invoked routines might be lifted by changing "SQL-connection statement" to "SQL-connection statement and the implementation does not support the execution of that SQL-statement in an invoked SQL-routine that is a procedure" in each of the two rules that make this prohibition, and making an appropriate entry in Appendix B, "Implementation-defined elements", saying something like "It is implementation-defined whether or not an SQL-implementation supports the execution of SQL-transaction statements and/or SQL-connection statements in an invoked SQL-routine; if it does so, then the effects are implementation-defined."

PSM-124 DBL:MCI-040/X3H2-96-169:UK-017 noted the following Language Opportunity:

> No way of obtaining the associated sqlstate of a condition name. We think the <condition name> feature is a nice idea, but we suspect it will generate a requirement, akin to the observation in the preceding comment, for a built-in function to return the associated sqlstate value of a given condition name.
>
> Furthermore, it might even be required to hold condition names in variables or arguments, in which case they have to become character strings. We would be happy to hold this feature over for SQL3, in the interests of simplification and early progression of PSM2 and to give time for the requirements to be fully thought through and appropriately addressed in the language.

PSM-140 The following Language Opportunity has been noted:

> Severity: Language Opportunity
> Reference: P02, SQL/Foundation, No specific location
> Note at: None specified
> Source: DBL:LGW-081/X3H2-97-???
> Language Opportunity:

Editor's Notes for WG3:HBA-005 = H2-2003-308

Is it possible in SQL3 to relax the specification of string data types such as <character string-type> and <bit string type> so that the declared length of these types (with appropriate usage restrictions) can be specified at execution time rather than at compile time? Can I declare avariable in an outer block of a compound statement and then use that variable as the <length> of a bit string variable declaration in an inner block?

Index

Index entries appearing in **boldface** indicate the page where the word, phrase, or BNF nonterminal was defined; index entries appearing in *italics* indicate a page where the BNF nonterminal was used in a Format; and index entries appearing in roman type indicate a page where the word, phrase, or BNF nonterminal was used in a heading, Function, Syntax Rule, Access Rule, General Rule, Leveling Rule, Table, or other descriptive text.

— P —