INTERNATIONAL STANDARD

**ISO/IEC 9075-4**

Sixth edition
2016-12-15

# 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 stockés persistants (SQL/PSM)*



Reference number
ISO/IEC 9075-4:2016(E)



ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

# Contents

Page

 

 

# Tables

## Table

**Page**

# 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. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: www.iso.org/iso/foreword.html.

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, SC 32, *Data management and interchange*.

This sixth edition of ISO/IEC 9075-4 cancels and replaces the fifth edition (ISO/IEC 9075-4:2011), which has been technically revised.

A list of all parts in the ISO/IEC 9075 series, published under the general title *Information technology — Database lan-guages — SQL*, can be found on the ISO website.

NOTE The individual parts of multi-part standards are not necessarily published together. New editions of one or more parts can be published without publication of new editions of other parts.

# 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 9, “Additional common elements”, defines additional common elements used in the definition of persistent stored modules.

9) Clause 10, “Schema definition and manipulation”, defines the schema definition and manipulation statements associated with the definition of persistent stored modules.

10) Clause 11, “Access control”, defines facilities for controlling access to SQL-data.

11) Clause 12, “SQL-client modules”, defines the facilities for using persistent stored modules.

12) Clause 13, “Data manipulation”, defines data manipulation operations associated with persistent stored modules.

13) Clause 14, “Additional data manipulation rules”, defines additional rules for data manipulation.

14) Clause 15, “Control statements”, defines the control statements used with persistent stored modules.

15) Clause 16, “Dynamic SQL”, defines the facilities for executing SQL-statements dynamically in the context of persistent stored modules.

16) Clause 17, “Embedded SQL”, defines the host language embeddings.

17) Clause 18, “Diagnostics management”, defines enhancements to the facilities used with persistent stored modules.

18) Clause 19, “Information Schema”, defines the Information and Definition Schema objects associated with persistent stored modules.

19) Clause 20, “Definition Schema”, defines base tables on which the viewed tables containing schema information depend.

20) Clause 21, “Status codes”, defines SQLSTATE values related to persistent stored modules.



 

21) Clause 22, “Conformance”, defines the criteria for conformance to this part of ISO/IEC 9075.

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

23) 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.

24) 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.

25) Annex D, “Deprecated features”, is an informative Annex. It lists features that the responsible Technical Committee intend will not appear in a future revised version of this part of ISO/IEC 9075.

26) Annex E, “Incompatibilities with ISO/IEC 9075:2011”, is an informative Annex. It lists incompatibilities with the previous version of this part of ISO/IEC 9075.

27) 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 an identifier and a short descriptive name. This taxonomy is used to specify conformance.

28) Annex G, “Defect reports not addressed in this edition of this part of ISO/IEC 9075”, is an informative Annex. It describes the Defect Reports that were known at the time of publication of this part of this International Standard. Each of these problems is a problem carried forward from the previous edition of ISO/IEC 9075. No new problems have been created in the drafting of this edition of this International Standard.

In the text of this part of ISO/IEC 9075, Clauses and Annexes begin new odd-numbered pages, and in Clause 5, “Lexical elements”, through Clause 22, “Conformance”, Subclauses begin new pages. Any resulting blank space is not significant.

*(Blank page)*



 

# 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 standing SQL-server 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.

*(Blank page)*

# 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 ISO and IEC standards

[ISO9075-1] ISO/IEC 9075-1:2016, *Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework)*.

[ISO9075-2] ISO/IEC 9075-2:2016, *Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation)*.

[ISO9075-11] ISO/IEC 9075-11:2016, *Information technology — Database languages — SQL — Part 11: Information and Definition Schemas (SQL/Schemata)*.

*(Blank page)*

# 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.

*(Blank page)*

# 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 SQL-server module 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>.

— The creation timestamp.

## 4.2 Tables

*This Subclause modifies Subclause 4.15, "Tables", in ISO/IEC 9075-2.*

### 4.2.1 Base tables

*This Subclause modifies Subclause 4.15.2, "Base tables", in ISO/IEC 9075-2.*

#### 4.2.1.1 Temporary tables

*This Subclause modifies Subclause 4.15.2.3, "Temporary tables", in ISO/IEC 9075-2.*

Insert this paragraph An SQL-server module declared local temporary table is a declared local temporary table defined in an <SQL-server module definition>. An SQL-server module declared local temporary table is effectively materialized the first time any <module routine> in the <SQL-server module definition> that contains the <temporary table declaration> is executed, and it persists for that SQL-session. Every SQL-server module in every SQL-session that references an SQL-server module declared local temporary table causes a distinct instance of that declared local temporary table (i.e., a multiset of rows that is visible only to that SQL-server module during that SQL-session) to be materialized. That is, the multiset of rows that is referenced by the <table name> of an SQL-server module declared local temporary table cannot be shared between SQL-sessions, nor between SQL-server modules that execute during an SQL-session.

### 4.2.2 Unique identification of tables

*This Subclause modifies Subclause 4.15.5, "Unique identification of tables", in ISO/IEC 9075-2.*

— Append after 4th list item The <table name> of an SQL-server module declared local temporary table, together with an SQL-session identifier and an SQL-server module name, uniquely identifies a multiset of rows.

## 4.3 SQL-schemas

*This Subclause modifies Subclause 4.26, "SQL-schemas", in ISO/IEC 9075-2.*

Insert into 1st paragraph

— An SQL-server module descriptor.

## 4.4 SQL-invoked routines

*This Subclause modifies Subclause 4.33, "SQL-invoked routines", in ISO/IEC 9075-2.*

### 4.4.1 Overview of SQL-invoked routines

*This Subclause modifies Subclause 4.33.1, "Overview of SQL-invoked routines", in ISO/IEC 9075-2.*

Replace 3rd paragraph An SQL-invoked routine is either a component of an SQL-server module or an element of an SQL-schema. An SQL-invoked routine that is defined in an <SQL-server module definition> is called an *SQL-module-level routine*. An SQL-invoked routine that is not defined in an <SQL-server module definition> is an element of an SQL-schema and is called a *schema-level routine*.

### 4.4.2 Execution of conventional SQL-invoked routines

*This Subclause modifies Subclause 4.33.3, "Execution of conventional SQL-invoked routines", in ISO/IEC 9075-2.*

Replace 2nd 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.

### 4.4.3 Routine descriptors

*This Subclause modifies Subclause 4.33.5, "Routine descriptors", in ISO/IEC 9075-2.*

Insert into 1st 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.

## 4.5 SQL-paths

*This Subclause modifies Subclause 4.34, "SQL-paths", in ISO/IEC 9075-2.*

Replace 2nd 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 4.35, "Host parameters", in ISO/IEC 9075-2.*

### 4.6.1 Status parameters

*This Subclause modifies Subclause 4.35.2, "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 4.36, "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 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> or a handler execution. A handler cannot be the most appropriate hander for an exception condition that occurs during the execution of that handler. 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 [ISO9075-1]. 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.

— Execute the handler action.

— Pop the diagnostics area stack.

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

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 15.2, "<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.

## 4.9 Cursors

*This Subclause modifies Subclause 4.38, "Cursors", in ISO/IEC 9075-2.*

### 4.9.1 General description of cursors

*This Subclause modifies Subclause 4.38.1, "General description of cursors", in ISO/IEC 9075-2.*

Insert after 2nd paragraph A cursor specified by a <declare cursor> contained in an <SQL-client module definition> without an intervening <SQL schema statement> is a *standing SQL-client cursor*. A cursor specified by a <declare cursor> contained in a <compound statement> is a *standing SQL-server cursor*.

Replace the 3rd paragraph A *declared cursor* is either a standing cursor, a declared dynamic cursor, or a received cursor. A declared cursor has a <cursor name>. A <declare cursor>, <dynamic declare cursor>, or <allocate received cursor statement> is either immediately contained in the <module contents> of an <SQL-client module definition>, or contained in the <local cursor declaration list> of a <compound statement>. The scope of a <cursor name> is the innermost <SQL-client module definition> *M* that contains it, with the exception of any <SQL schema statement>s contained in *M*, or the <local handler declaration list> *LHDL* of a <compound statement> *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*.

Replace 2nd list item of the 5th paragraph The provenance of the cursor:

— If the cursor is a declared cursor or a local extended dynamic cursor, then an indication of an SQL-client module, or a <compound statement>.

— If the cursor is a PTF dynamic cursor, then a <routine invocation> that invokes a polymorphic table function.

— Otherwise, an SQL-session identifier.

Insert this paragraph For every <declare cursor> in a <compound statement>, a standing SQL-server cursor is effectively created each time the <compound statement> is executed and destroyed when that execution completes.

NOTE 2 — Destroying an open with-return cursor does not simultaneously destroy that cursor's result set.



 

## 4.10 SQL-statements

*This Subclause modifies Subclause 4.39, "SQL-statements", in ISO/IEC 9075-2.*

### 4.10.1 Classes of SQL-statements

*This Subclause modifies Subclause 4.39.1, "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 4.39.2, "SQL-statements classified by function", in ISO/IEC 9075-2.*

#### 4.10.2.1 SQL-schema statements

*This Subclause modifies Subclause 4.39.2.1, "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 4.39.2.6, "SQL-control statements", in ISO/IEC 9075-2.*

Insert this paragraph 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 4.39.2.8, "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 SQL-statements and transaction states

*This Subclause modifies Subclause 4.39.4, "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>



 

### 4.10.4 SQL-statement atomicity and statement execution contexts

*This Subclause modifies Subclause 4.39.5, "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>.

### 4.10.5 Embeddable SQL-statements

*This Subclause modifies Subclause 4.39.6, "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 3 — 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>.

### 4.10.6 Preparable and immediately executable SQL-statements

*This Subclause modifies Subclause 4.39.7, "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>

### 4.10.7 Directly executable SQL-statements

*This Subclause modifies Subclause 4.39.8, "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

### 4.10.8 Iterated SQL-statements

The following are the iterated SQL-statements:

— <loop statement>

— <while statement>

— <repeat statement>

— <for statement>

### 4.10.9 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 standing SQL-server cursors can be declared. See Subclause 15.1, "<compound statement>".

## 4.11 Basic security model

*This Subclause modifies Subclause 4.40, "Basic security model", in ISO/IEC 9075-2.*



 

### 4.11.1 Privileges

*This Subclause modifies Subclause 4.40.2, "Privileges", in ISO/IEC 9075-2.*

Insert into 1st paragraph

— SQL-server module

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 object identification included in an execute privilege descriptor may also identify the SQL-server module described by the descriptor.

Insert this paragraph If the object identified by an execute privilege descriptor is an SQL-invoked routine *R*, then *R* shall be a schema-level routine.

NOTE 4 — "schema-level routine" is defined in Subclause 11.60, "<SQL-invoked routine>", in [ISO9075-2].

## 4.12 SQL-sessions

*This Subclause modifies Subclause 4.43, "SQL-sessions", in ISO/IEC 9075-2.*

### 4.12.1 General description of SQL-sessions

*This Subclause modifies Subclause 4.43.1, "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.

*(Blank page)*

# 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 5)a) If *LSQ* is "MODULE", then *TN* shall be contained in exactly one of the following:

   a) An <SQL-client module definition> that contains, without an intervening <SQL schema statement>, a <temporary table declaration> *TT* whose <table name> has a <qualified identifier> equivalent to *QI*.

   b) An <SQL-server module definition> that contains a <temporary table declaration> *TT* whose <table name> has a <qualified identifier> equivalent to *QI*.

4) Replace SR 7) Let *CN* be a <cursor name>. At least one of the following shall be true:

   a) *CN* is contained, without an intervening <SQL schema statement>, in an <SQL-client module definition> whose <module contents> contains a <declare cursor> or <dynamic declare cursor> whose <cursor name> is *CN*.

   b) *CN* is contained, without an intervening <SQL schema statement>, in a <compound statement> whose <local cursor declaration list> contains a <declare cursor> whose <cursor name> is *CN*.

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 an 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 an 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>.

6) Replace SR 13)d)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.

7) Insert before SR 13)d)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.

8) Replace SR 28) If a <descriptor name> contains an <extended descriptor name> that identifies an <SQL parameter reference> or an <SQL variable reference>, respectively, and does not specify a <scope option>, then that <SQL parameter reference> or <SQL variable reference>, respectively, is used to supply the value for the <descriptor name>.

NOTE 5 — The previous rule disambiguates between an <extended descriptor name> that is an <SQL parameter reference> or an <SQL variable reference>, respectively, and a <non-extended descriptor name> that is an <identifier> and gives precedence to the <SQL parameter reference> or <SQL variable reference>, respectively.

## 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.*

*(Blank page)*

# 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

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

2) Replace GR 15) A <simple target specification> specifies a target that is a host parameter, an output SQL parameter, a column of a new transition variable, a host variable, or an SQL variable, according to whether the <simple target specification> is a <host parameter name>, an <SQL parameter reference>, a <column reference>, an <embedded variable name>, or an <SQL variable reference>, respectively.

NOTE 6 — A <simple target specification> can never be assigned the null value.

## 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 the lead text 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* or a period *P* 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 column lists include a column whose <column name> is equivalent to $I_1$, or in the scope of one or more range variables whose associated period lists include a period whose period 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 <SQL variable name> is equivalent to $I_1$. Let *IS* be the innermost such scope. Let the phrase *possible scope tags* denote those range variables, <routine name>s, and <beginning label>s whose scope is *IS*. The number of possible scope tags shall be 1 (one). Let *IPST* be that possible scope tag.

4) Insert after SR 8)a)ii)2) 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 *SV*, 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 the Note associated with SR 10)

NOTE 7 — Replace Note 173 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 either a <nonparenthesized value expression primary> that is a <column reference> or an <unsigned value specification> that is a <general value specification> that is either an <SQL parameter reference> or an <SQL variable reference>. The identifiers $I_{BL+1}$, ..., $I_N$ are parsed using the Format and Syntax Rules of Subclause 6.15, "<field reference>", and Subclause 6.17, "<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 Format and Syntax Rules of <modified field reference> and <mutator reference> in Subclause 15.5, "<assignment statement>".

9) Insert after SR 15) 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 6.14, "<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

*None.*



 

# 7 Query expressions

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

## 7.1 <table reference>

*This Subclause modifies Subclause 7.6, "<table reference>", in ISO/IEC 9075-2.*

### Function

Reference a table.

### Format

```
No additional Format items.
```

### Syntax Rules

1) Replace SR 8)d)iii) *QSTPS* shall not contain a <column reference>, an <SQL parameter reference>, or an <SQL variable reference>.

### Access Rules

*No additional Access Rules.*

### General Rules

*No additional General Rules.*

### Conformance Rules

*No additional Conformance Rules.*

## 7.2 <query specification>

*This Subclause modifies Subclause 7.16, "<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 8)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 8)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* 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 8)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 13)o)ii) If $DCOL_k$ is a single SQL variable reference, then $COLN_k$ is the <SQL variable name> of the SQL variable designated by the SQL variable reference.

5) Insert after SR 19)c) If the *i*-th <derived column> in the <select list> does not specify an <as clause> and the <value expression> of that <derived column> is a single SQL variable reference, then the <column name> of the *i*-th column of the result is the <SQL variable name> of the SQL variable designated by the SQL variable reference.

6) Insert after SR 21)b)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>.

*(Blank page)*



 

# 8 Additional common rules

*This Clause modifies Clause 9, "Additional common rules", in ISO/IEC 9075-2.*

## 8.1 Handler execution

### Subclause Signature

```
"Handler execution" [General Rules] (
  Parameter: "MOST APPROPRIATE HANDLER"
)
```

### Function

Execute the most appropriate handler for an exception condition.

### Syntax Rules

*None.*

### Access Rules

*None.*

### General Rules

NOTE 8 — Except where explicitly specified, the GRs of this Subclause are not terminated when an exception condition is raised.

1) Case:

   a) If this Subclause has been invoked as a "subroutine Subclause" from another Subclause, then: Let *MAH* be the *MOST APPROPRIATE HANDLER* in an application of the General Rules of this Subclause. *MAH* is the *most appropriate handler*.

   b) Otherwise, let *MAH* be the *most appropriate handler* for the condition *C*, the raising of which caused the invocation of this Subclause.

2) Let *HD* be the <handler declaration> that effectively created *MAH*.

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

4) Let *HT* be the <handler type> specified in *MAH*.

5) Let *HA* be the <handler action> simply contained in *MAH*.

6) *MAH* is effectively removed from the list of potential most appropriate handlers.

NOTE 9 — This prevents the handler currently being executed from being chosen as the most appropriate handler for an exception that is raised during the execution of this handler.

7) If *MAH* is activated in an atomic execution context and the condition raised is a *transaction rollback* with any subcondition, then the following <resignal statement> is effectively executed:

```
RESIGNAL
```

NOTE 10 — If a condition results in an implicit rollback (See Subclause 4.41.5, "Implicit rollbacks", in [ISO9075-2]) in an atomic execution context, the transaction has been effectively rolled back by the time the Handler body is executed. If any transaction initiating statement is executed following this, it would require a transaction to be initiated in an atomic execution context, which is not valid. Therefore, the condition is effectively resignaled to an outer non-atomic execution context.

8) If *HT* specifies CONTINUE, then:

a) The General Rules of Subclause 23.2, "Pushing and popping the diagnostics area stack", in [ISO9075-2], are applied with PUSH as *OPERATION* and the diagnostics area stack as *STACK*.

b) The General Rules of Subclause 15.1, "<compound statement>", are applied with *HA* as *COMPOUND STATEMENT*.

c) Case:

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

1) The General Rules of Subclause 23.2, "Pushing and popping the diagnostics area stack", in [ISO9075-2], are applied with PUSH as *OPERATION* and the diagnostics area stack as *STACK*.

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

```
RESIGNAL
```

ii) Otherwise:

1) The General Rules of Subclause 23.2, "Pushing and popping the diagnostics area stack", in [ISO9075-2], are applied with POP as *OPERATION* and the diagnostics area stack as *STACK*.

2) *MAH* is effectively added back to the list of potential most appropriate handlers.

3) *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.

9) If *HT* specifies EXIT, then:

a) The General Rules of Subclause 23.2, "Pushing and popping the diagnostics area stack", in [ISO9075-2], are applied with PUSH as *OPERATION* and the diagnostics area stack as *STACK*.

b) The General Rules of Subclause 15.1, "<compound statement>", are applied with *HA* as *COMPOUND STATEMENT*.

c) For every open standing SQL-server cursor *CR* declared in the <local cursor declaration list> of *CS*, the General Rules of Subclause 15.4, "Effect of closing a cursor", in [ISO9075-2], are applied with CR as *CURSOR* and SAVE as *DISPOSITION*.



 

d) Case:

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

1) The General Rules of Subclause 23.2, "Pushing and popping the diagnostics area stack", in [ISO9075-2], are applied with PUSH as *OPERATION* and the diagnostics area stack as *STACK*.

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

```
RESIGNAL
```

ii) Otherwise:

1) The General Rules of Subclause 23.2, "Pushing and popping the diagnostics area stack", in [ISO9075-2], are applied with POP as *OPERATION* and the diagnostics area stack as *STACK*.

2) *MAH* is effectively added back to the list of potential most appropriate handlers.

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

10) If *HT* specifies UNDO, then:

a) The General Rules of Subclause 23.2, "Pushing and popping the diagnostics area stack", in [ISO9075-2], are applied with PUSH as *OPERATION* and the diagnostics area stack as *STACK*.

b) 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.

c) For every open standing SQL-server cursor *CR* declared in the <local cursor declaration list> of *CS*, the General Rules of Subclause 15.4, "Effect of closing a cursor", in [ISO9075-2], are applied with CR as *CURSOR* and SAVE as *DISPOSITION*.

d) The General Rules of Subclause 15.1, "<compound statement>", are applied with *HA* as *COMPOUND STATEMENT*.

e) Case:

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

1) The General Rules of Subclause 23.2, "Pushing and popping the diagnostics area stack", in [ISO9075-2], are applied with PUSH as *OPERATION* and the diagnostics area stack as *STACK*.

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

```
RESIGNAL
```

ii) Otherwise:

1) The General Rules of Subclause 23.2, "Pushing and popping the diagnostics area stack", in [ISO9075-2], are applied with POP as *OPERATION* and the diagnostics area stack as *STACK*.

2) *MAH* is effectively added back to the list of potential most appropriate handlers.

3) *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

*None.*



 

# 9 Additional common elements

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

## 9.1 <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 6) 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)c) 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)c) 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) Replace SR 9)h)iii)5) If $XA_i$ is an <SQL variable reference>, an <SQL parameter reference>, a <column reference>, or a <target array element specification>, then the Syntax Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2, are applied with $P_i$ as *TARGET* and $XA_i$ as *VALUE*.

NOTE 11 — 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)f)i)4) The identities of declared local temporary tables that are defined in <SQL-server module definition>s.

2) Insert after GR 8)a) If *R* is included in an SQL-server module *S*, then the method and time of binding of *P* to *S* is implementation-defined.

3) Replace the lead text 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 12 — The <column reference> can only be a new transition variable column reference.

## Conformance Rules

*No additional Conformance Rules.*



 

## 9.2 <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 code and the last three have the form of a standard-defined subclass code.

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

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

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.*

*(Blank page)*



 

# 10 Schema definition and manipulation

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

## 10.1 <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.*

## 10.2 <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.*

## 10.3 <table definition>

*This Subclause modifies Subclause 11.3, "<table definition>", in ISO/IEC 9075-2.*

### Function

Define a persistent base table, a created local temporary table, or a global temporary table.

### Format

```
No additional Format items.
```

### Syntax Rules

1) Replace SR 4) The <table contents source> shall not contain a <host parameter specification>, an <SQL parameter reference>, a <dynamic parameter specification>, an <embedded variable specification>, or an <SQL variable reference>.

### Access Rules

*No additional Access Rules.*

### General Rules

*No additional General Rules.*

### Conformance Rules

*No additional Conformance Rules.*



 

## 10.4 <column definition>

*This Subclause modifies Subclause 11.4, "<column definition>", in ISO/IEC 9075-2.*

### Function

Define a column of a base table.

### Format

```
No additional Format items.
```

### Syntax Rules

1) Replace SR 1) The <column definition> shall not contain a <host parameter specification>, an <SQL parameter reference>, a <dynamic parameter specification>, an <embedded variable specification>, or an <SQL variable reference>.

### Access Rules

*No additional Access Rules.*

### General Rules

*No additional General Rules.*

### Conformance Rules

*No additional Conformance Rules.*

## 10.5 <default clause>

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

### Function

Specify the default for a column, domain, attribute, 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

*No additional General Rules.*

### Conformance Rules

*No additional Conformance Rules.*



 

## 10.6 <check constraint definition>

*This Subclause modifies Subclause 11.9, "<check constraint definition>", in ISO/IEC 9075-2.*

### Function

Specify a condition for the SQL-data.

### Format

```
No additional Format items.
```

### Syntax Rules

1) Replace SR 1) The <search condition> shall not contain a <host parameter specification>, an <SQL parameter reference>, a <dynamic parameter specification>, an <embedded variable specification>, an <SQL variable reference>, or a <column reference> that references a system-time period start column or a system-time period end column.

### Access Rules

*No additional Access Rules.*

### General Rules

*No additional General Rules.*

### Conformance Rules

*No additional Conformance Rules.*

## 10.7 <drop column scope clause>

*This Subclause modifies Subclause 11.18, "<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

*No additional Access Rules.*

### 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.*



 

## 10.8 <drop column definition>

*This Subclause modifies Subclause 11.23, "<drop column definition>", in ISO/IEC 9075-2.*

### Function

Destroy a column of a base table.

### Format

```
No additional Format items.
```

### Syntax Rules

1) Insert after SR 7)e) 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.*

## 10.9 <drop table constraint definition>

*This Subclause modifies Subclause 11.26, "<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> or any SQL-invoked routine that is dependent on *TC*.

   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 for every *R*:

      ```
      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.*



 

## 10.10 <drop table statement>

*This Subclause modifies Subclause 11.31, "<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.*

## 10.11 <view definition>

*This Subclause modifies Subclause 11.32, "<view definition>", in ISO/IEC 9075-2.*

### Function

Define a viewed table.

### Format

```
No additional Format items.
```

### Syntax Rules

1) Replace SR 2) *QE* shall not contain a <host parameter specification>, an <SQL parameter reference>, a <dynamic parameter specification>, an <embedded variable specification>, or an <SQL variable reference>.

2) Insert after SR 21)t)vi)5) 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.*



 

## 10.12 <drop view statement>

*This Subclause modifies Subclause 11.33, "<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.*

## 10.13 <drop domain statement>

*This Subclause modifies Subclause 11.40, "<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.*



 

## 10.14 <drop character set statement>

*This Subclause modifies Subclause 11.42, "<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.*

# 10.15 <drop collation statement>

*This Subclause modifies Subclause 11.44, "<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.*



 

## 10.16 <drop transliteration statement>

*This Subclause modifies Subclause 11.46, "<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 *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.*

## 10.17 <assertion definition>

*This Subclause modifies Subclause 11.47, "<assertion definition>", in ISO/IEC 9075-2.*

### Function

Specify an integrity constraint.

### Format

```
No additional Format items.
```

### Syntax Rules

1) Modify SR 5) Add ", <SQL variable name>" to the list in the Syntax Rule.

2) Insert after SR 5)

NOTE 13 — <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.*



 

## 10.18 <drop assertion statement>

*This Subclause modifies Subclause 11.48, "<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.*

# 10.19 <trigger definition>

*This Subclause modifies Subclause 11.49, "<trigger definition>", in ISO/IEC 9075-2.*

## Function

Defined triggered SQL-statements.

## Format

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

NOTE 14 — The preceding production defining <triggered SQL statement> completely supersedes the definition in [ISO9075-2].

## 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 AR 3) 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 *A* shall include SELECT on *T*.

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

## General Rules

*No additional General Rules.*

## Conformance Rules

*No additional Conformance Rules.*



 

# 10.20 <drop user-defined ordering statement>

*This Subclause modifies Subclause 11.66, "<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 whose <SQL routine body> contains an operand of an equality operation, a grouping operation, or ordering operation whose declared type is some user-defined type *T1* whose comparison type is *UDT*.

   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 an operand of an equality operation, a grouping operation, or an ordering operation whose declared type is some user-defined type *T1* whose comparison type is *UDT* 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.*



 

## 10.21 <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 15 — "Schema-level routine" is defined in Subclause 11.60, "<SQL-invoked routine>", in [ISO9075-2].

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 16 — 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>.

i) The CURRENT_TIMESTAMP as the value of the creation timestamp.

## Conformance Rules

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

# 10.22 <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 <parameter default> of any SQL parameter of any routine descriptor not included in the module descriptor of *M*.

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

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

   e) Any trigger descriptor.

   f) 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>.

# 10.23 <drop data type statement>

*This Subclause modifies Subclause 11.59, "<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.*



 

## 10.24 <SQL-invoked routine>

*This Subclause modifies Subclause 11.60, "<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 9)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 9)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 9)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 or implicit <schema name> of the containing <SQL-server module definition>. Let *S* be the SQL-schema identified by *SN*.

4) Insert after SR 23)h) An <SQL routine body> contained in a schema-level routine shall not immediately contain an <SQL procedure statement> that simply contains a <table reference> that identifies an SQL-server module declared local temporary table.

### Access Rules

*No additional Access Rules.*

## General Rules

1) Insert after GR 3)w) 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.*



 

## 10.25 <drop routine statement>

*This Subclause modifies Subclause 11.62, "<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 4)c)v) 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.*

## 10.26 <drop user-defined cast statement>

*This Subclause modifies Subclause 11.64, "<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.*



 

# 11 Access control

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

## 11.1 <grant statement>

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

### Function

Define privileges and role authorizations.

### 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*, and those privileges are grantable, then for every privilege descriptor with an <action> of EXECUTE, a <grantor> of "_SYSTEM", <object name> 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 17 — 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.*



 

## 11.2 <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 <SQL-server module name>
```

### Syntax Rules

1) Replace SR 6) 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> may 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 an <object name> that contains MODULE.

## 11.3 <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

*No additional Syntax Rules.*

### Access Rules

*No additional Access Rules.*

### General Rules

1) Insert after GR 15)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>, <method invocation>, <static method invocation>, or <method reference> that is contained in the *QE*.

2) Insert after GR 17)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>, <method invocation>, <static method invocation>, or <method reference> that is contained in any applicable <search condition> of *TC*.

3) Insert after GR 18)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>, <method invocation>, <static method invocation>, or <method reference> that is contained in any applicable <search condition> of *AX*.

4) Insert after GR 19)g) 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>, <method invocation>, <static method invocation>, or <method reference> that is contained in any <search condition> of *TR*.

5) Insert after GR 19)h) 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>, <method invocation>, <static method invocation>, or <method reference> that is contained in the <triggered SQL statement> of *TR*.

6) Insert after GR 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>, <method invocation>, <static method invocation>, or <method reference> that is contained in any <search condition> of *DC*.



 

7) Insert after GR 22)d) *CD* has a generation expression *GE* and the revoke destruction action would result in *A1* no longer having in its applicable privileges EXECUTE privilege on any SQL-server module that includes one or more SQL-invoked routines that are among the subject routines of any <routine invocation>, <method invocation>, <static method invocation>, or <method reference> that is contained in *GE*.

8) Insert after GR 29)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 <SQL routine body> of *RD* or in the <parameter default> of any SQL parameter of *RD*.

9) Insert after GR 32) 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 in its applicable privileges any 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 <SQL 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 <SQL routine body> of any SQL-invoked routine with an SQL 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 <SQL routine body> of any SQL-invoked routine with an SQL 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: positioned>, or a <merge statement> contained in the <SQL routine body> of any SQL-invoked routine with an SQL 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 an SQL 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 an SQL 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 an SQL 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> or a <merge statement>contained in the <SQL routine body> of any SQL-invoked routine with an SQL 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 an SQL security characteristic of DEFINER included in *SSM*.

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

l) DELETE privilege on the table identified by the <target table> contained in a <merge statement> that contains a <merge delete specification> and that is contained in the <SQL routine body> of any SQL-invoked routine with an SQL security characteristic of DEFINER included in *SSM*.

m) USAGE privilege on every domain, every collation, every character set, every transliteration, and every sequence generator whose name is contained in the <SQL routine body> of any SQL-invoked routine with an SQL security characteristic of DEFINER included in *SSM*.

n) USAGE privilege on each user-defined type *UDT* such that a declared type of any SQL parameter, returns data type, or result cast included in any SQL-invoked routine with an SQL security characteristic of DEFINER included in *SSM* that is usage-dependent on *UDT*.

o) USAGE privilege on every user-defined type *UDT* such that there is a <data type> contained in the <SQL routine body> of any SQL-invoked routine with an SQL security characteristic of DEFINER included in *SSM* that is usage-dependent on *UDT*.

p) 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 an SQL 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*.

q) SELECT privilege on any column identified by a <column reference> contained in a <scalar subquery> that is equivalent to a <dereference operation> contained in any of the following:

i) A <query expression> simply contained in a <cursor specification>, an <insert statement>, or a <merge statement> contained in the <SQL routine body> of any SQL-invoked routine with an SQL security characteristic of DEFINER included in *SSM*.

ii) A <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 an SQL security characteristic of DEFINER included in *SSM*.



 

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

iv) A <value expression> contained in an <update source> or an <assigned row> contained in the <SQL routine body> of any SQL-invoked routine with an SQL 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 <query expression> simply contained in a <cursor specification>, an <insert statement>, or a <merge statement> contained in the <SQL routine body> of any SQL-invoked routine with an SQL security characteristic of DEFINER included in *SSM*.

s) 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 an SQL security characteristic of DEFINER included in *SSM*.

t) 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 an SQL security characteristic of DEFINER included in *SSM*.

u) 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 an SQL security characteristic of DEFINER included in *SSM*.

v) SELECT privilege on the scoped table of any <reference resolution> that is contained in any of the following:

i) A <query expression> simply contained in a <cursor specification>, an <insert statement>, or a <merge statement> contained in the <SQL routine body> of any SQL-invoked routine with an SQL security characteristic of DEFINER included in *SSM*.

ii) A <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 an SQL security characteristic of DEFINER included in *SSM*.

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

iv) A <value expression> contained in an <update source> or an <assigned row> contained in the <SQL routine body> of any SQL-invoked routine with an SQL security characteristic of DEFINER included in *SSM*.

w) 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 an SQL security characteristic of DEFINER included in *SSM*.

10) Insert after GR 48) 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.*



 

# 12 SQL-client modules

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

## 12.1 <externally-invoked procedure>

*This Subclause modifies Subclause 13.3, "<externally-invoked procedure>", in ISO/IEC 9075-2.*

### Function

Define an externally-invoked procedure.

### Syntax Rules

1) Insert into SR 10)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.*

## 12.2 <SQL procedure statement>

*This Subclause modifies Subclause 13.4, "<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) Insert this SR An <SQL connection statement> shall not be contained in an <SQL control statement> or an <SQL-server module definition>.

2) Insert after SR 3)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, SYSTEM_USER, CURRENT_CATALOG, CURRENT_SCHEMA, or CURRENT_PATH.

3) Insert after SR 3)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, SYSTEM_USER, CURRENT_CATALOG, CURRENT_SCHEMA, or CURRENT_PATH.



 

## Access Rules

*No additional Access Rules.*

## General Rules

1) Replace GR 6) If *S* is not an <SQL diagnostics statement> or a <compound statement>, then the first diagnostics area is emptied.

2) Insert before GR 21) Condition handling mode becomes in effect in the SQL-session.

3) Insert before GR 21) If *S* did not execute successfully, then the General Rules of Subclause 8.1, "Handler execution", are applied with no handler as *MOST APPROPRIATE HANDLER*.

NOTE 18 — The General Rules of Subclause 8.1, "Handler execution" determine which handler will be invoked for the current condition.

4) Insert after GR 23) Condition handling mode ceases to be in effect in the SQL-session.

## Conformance Rules

*No additional Conformance Rules.*

*(Blank page)*



 

# 13 Data manipulation

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

## 13.1 <declare cursor>

*This Subclause modifies Subclause 14.1, "<declare cursor>", in ISO/IEC 9075-2.*

### Function

Declare a standing cursor.

### Format

```
No additional Format items.
```

### Syntax Rules

1) Replace the lead text of SR 1) If a <declare cursor> is contained in an <SQL-client module definition> *M* without an intervening <SQL schema statement>, then:

2) Insert this SR If a <declare cursor> *DC* is contained in the <local cursor declaration list> *LCDL* of a <compound statement> *CS*, then:

   a) The <cursor name> shall not be equivalent to the <cursor name> of any other <declare cursor>, <dynamic declare cursor>, or <allocate received cursor statement> contained in *LCDL*.

   b) The scope of the <cursor name> is *CS*.

### Access Rules

*No additional Access Rules.*

### General Rules

1) Replace GR 1)b) The provenance of the cursor is

   Case:

   a) If the <declare cursor> is contained in a <compound statement>, then the innermost <compound statement> that contains *DC*.

b) Otherwise, an indication of the SQL-client module whose <SQL-client module definition> contains the <declare cursor>.

## Conformance Rules

*No additional Conformance Rules.*



 

## 13.2 <open statement>

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

### Function

Open a standing 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.

### Access Rules

*No additional Access Rules.*

### General Rules

*No additional General Rules.*

### Conformance Rules

*No additional Conformance Rules.*

## 13.3 <fetch statement>

*This Subclause modifies Subclause 14.5, "<fetch statement>", in ISO/IEC 9075-2.*

### Function

Position a standing cursor on a specified row of the standing cursor's result set 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.

2) Replace SR 9)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 9075-2, are applied with *TS* as *TARGET* and an arbitrary value of the row type of *T* as *VALUE*.

3) Replace the lead text of SR 9)b)iii) For each <target specification> $TS1_i$, 1 (one) **≤** *i* **≤** *NTS*, that is either an <SQL variable reference>, an <SQL parameter reference>, or a <target array element specification>,

### Access Rules

*No additional Access Rules.*

### General Rules

1) Replace GR 5)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 9075-2, are applied with *TS* as *TARGET* and the current row as *VALUE*.

2) Replace the lead text of GR 5)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.*



 

## 13.4 <close statement>

*This Subclause modifies Subclause 14.6, "<close statement>", in ISO/IEC 9075-2.*

### Function

Close a standing cursor.

### Format

```
No additional Format items.
```

### Syntax Rules

1) Replace SR 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.

### Access Rules

*No additional Access Rules.*

### General Rules

*No additional General Rules.*

### Conformance Rules

*No additional Conformance Rules.*

## 13.5 <select statement: single row>

*This Subclause modifies Subclause 14.7, "<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 lead text of SR 3)b)ii) For *i* varying from 1 (one) to *NOE*, let $TS1_i$ 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 lead text 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.*



 

## 13.6 <delete statement: positioned>

*This Subclause modifies Subclause 14.8, "<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 *DSP* be the <delete statement: positioned> and let *CN* be the <cursor name> immediately contained in *DSP*. *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.

### Access Rules

*No additional Access Rules.*

### General Rules

*No additional General Rules.*

### Conformance Rules

*No additional Conformance Rules.*

## 13.7 <update statement: positioned>

*This Subclause modifies Subclause 14.13, "<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 *USP* be the <update statement: positioned> and let *CN* be the <cursor name> immediately contained in *USP*. *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.

### Access Rules

*No additional Access Rules.*

### General Rules

*No additional General Rules.*

### Conformance Rules

*No additional Conformance Rules.*



 

## 13.8 <temporary table declaration>

*This Subclause modifies Subclause 14.16, "<temporary table declaration>", in ISO/IEC 9075-2.*

### Function

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>, in a <direct SQL statement>, 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 the <qualified identifier> of *TN* shall not be equivalent to the <qualified identifier> of the <table name> of any other <temporary table declaration> contained without an intervening <SQL-server module definition> in *M*.

   b) Otherwise, the <qualified identifier> of *TN* shall not be equivalent to the <qualified identifier> of the <table name> of any other <temporary table declaration> contained in the containing <SQL-server module definition>.

### Access Rules

*No additional Access Rules.*

### General Rules

1) 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.*



 

# 14 Additional data manipulation rules

*This Clause modifies Clause 15, "Additional data manipulation rules", in ISO/IEC 9075-2.*

## 14.1 Effect of opening a cursor

*This Subclause modifies Subclause 15.1, "Effect of opening a cursor", in ISO/IEC 9075-2.*

### Function

Specify the effect of opening a cursor that is not a received cursor.

### Syntax Rules

*None.*

### Access Rules

*None.*

### General Rules

1) Replace GR 5)a)i) Each <embedded variable specification>, <host parameter specification>, <SQL parameter reference>, <dynamic parameter specification>, and <SQL variable reference> is replaced by a <literal> denoting the value resulting from evaluating the <embedded variable specification>, <host parameter specification>, <SQL parameter reference>, <dynamic parameter specification>, and <SQL variable reference>, respectively, with all such evaluations effectively done at the same instance in time.

*(Blank page)*



 

# 15 Control statements

*This Clause is modified by Clause 15, "Control statements", in ISO/IEC 9075-14.*
*This Clause modifies Clause 16, "Control statements", in ISO/IEC 9075-2.*

## 15.1 <compound statement>

*This Subclause is modified by Subclause 11.27, "<compound statement>", in ISO/IEC 9075-10.*
*This Subclause is modified by Subclause 15.1, "<compound statement>", in ISO/IEC 9075-14.*

### Subclause Signature

```
"<compound statement>" [General Rules] (
  Parameter: "COMPOUND STATEMENT"
)
```

### Function

Specify a statement that groups other statements together.

### Format

```
10 14 <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* 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 standing SQL-server 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 standing SQL-servers 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

NOTE 19 — Except where explicitly specified, the General Rules of this Subclause are not terminated when an exception condition is raised.

1) Case:

   a) If this Subclause has been invoked as a "subroutine Subclause" from another Subclause, then: Let *CS* be the *COMPOUND STATEMENT* in an application of the General Rules of this Subclause.

   b) Otherwise, let *CS* be the <compound statement>.

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

3) The SQL variables, standing SQL-server 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.

4) 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 13.4, "<SQL procedure statement>", in ISO/IEC 9075-2, are applied with $S_i$ as *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:

```
RESIGNAL
```

ii) If there are unhandled exception conditions at the completion of the execution of a handler (if any), then the execution of GR 4) is terminated immediately and evaluation continues with GR 5).

5) For every open cursor *CR* that is declared in the <local cursor declaration list> of *CS*, the General Rules of Subclause 15.4, "Effect of closing a cursor", in [ISO9075-2], are applied with CR as *CURSOR* and SAVE as *DISPOSITION*.

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

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

NOTE 20 — Destroying a savepoint level destroys all existing savepoints that are established at that level.

8) 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>.



 

## 15.2 <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 name> 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 codes corresponding to categories X, W, and N, respectively, in Subclause 24.1, "SQLSTATE", in [ISO9075-2].

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*.

## Conformance Rules

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



 

## 15.3 <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>.

## 15.4 <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) If <SQL variable declaration> contains <default clause> *DC*, then let *DV* be the <default option> contained in *DC*. Otherwise let *DV* be <null specification>. Let *SV* be the variable defined by the <SQL variable declaration>. The following SQL-statement is effectively executed:

   ```
   SET SV = DV
   ```

### Conformance Rules

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



 

## 15.5 <assignment statement>

*This Subclause is modified by Subclause 11.16, "<assignment statement>", in ISO/IEC 9075-10.*
*This Subclause is modified by Subclause 15.2, "<assignment statement>", in ISO/IEC 9075-14.*

### 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>

14 <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* ≤ *ATN*, 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 <target specification> immediately contained in a <modified field target> or a <mutated target specification> that is a <column reference> shall be a new transition variable column reference.

   NOTE 21 — "New transition variable column reference" is defined in Subclause 6.6, "<identifier chain>", in [ISO9075-2].

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 specification>, 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 22 — 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 with *F* as *TARGET* and *AS* as *VALUE*.



 

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 with <assignment target> as *TARGET* and <assignment source> as *VALUE*.

9) [10] 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 with <assignment target> as *TARGET* and <assignment source> as *VALUE*.

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", in ISO/IEC 9075-2, are applied with an arbitrary site whose declared type is the declared type of the <target specification> as *TARGET* and <assignment source> as *VALUE*.

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) [14] 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 General Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2, are applied with *T* as *TARGET* and <assignment source> as *VALUE*.

2) [10] 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 General Rules of Subclause 9.1, "Retrieval assignment", in ISO/IEC 9075-2, are applied with *T* as *TARGET* and <assignment source> as *VALUE*.

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 General Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2, are applied with *TV.C* as *TARGET* and <assignment source> as *VALUE*.

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) [14] Otherwise, the General Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2, are applied with *FT* as *TARGET* and <assignment source> as *VALUE*.

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

Case:

a) If the value of <target array reference> *TAR*, 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 *TAR*.

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

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

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

v) Case:

1) If *I* is greater than zero and less than or equal to *M*, then the value of *TAR* 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 *TAR*.

B) [14] The General Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2, are applied with *I*-th element of *A* as *TARGET* and <assignment source> as *VALUE*.

2) If *I* is greater than *M* and less than or equal to *N*, then the value of *TAR* 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 *TAR*.

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

C) [14] The General Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2, are applied with *I*-th element of *A* as *TARGET* and <assignment source> as *VALUE*.

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>.



 

## 15.6 <case statement>

### Function

Provide conditional execution based on the truth value 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 case 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
```

j) The Conformance Rules of the Subclauses of Clause 8, "Predicates", in [ISO9075-2], are applied to the result of this syntactic transformation.

NOTE 23 — The specific Subclauses of Clause 8, "Predicates", in [ISO9075-2], are determined by the predicates that are created as a result of the syntactic transformation.

## 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 13.4, "<SQL procedure statement>", in ISO/IEC 9075-2, are applied with $S_i$ as *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 predicand>.

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

## 15.7 <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>
  ISEC1
  [ <if statement elseif clause>... ]
  [ <if statement else clause> ]
END IF
```

is equivalent to

```
IF <search condition>
  <if statement then clause>
  ELSE
    IF SC1
      THEN SL1
      [ <if statement elseif clause>... ]
      [ <if statement else clause> ]
    END IF
END IF
```

where *SC1* is the <search condition> simply contained in <if statement elseif clause> *ISEC1* and *SL1* is the <SQL statement list> simply contained in *ISEC1*.



 

## 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 24 — "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 13.4, "<SQL procedure statement>", in ISO/IEC 9075-2, are applied with $S_i$ as *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>.

## 15.8 <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 25 — 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>.



 

## 15.9 <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 standing SQL-server cursor *CR* that is declared in the <local cursor declaration list> of *CS*, the General Rules of Subclause 15.4, "Effect of closing a cursor", in [ISO9075-2], are applied with CR as *CURSOR* and SAVE as *DISPOSITION*.

   b) The variables, standing SQL-server 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>.

# 15.10 <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* 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 13.4, "<SQL procedure statement>", in ISO/IEC 9075-2, are applied with *CCS* as *EXECUTING STATEMENT*

   NOTE 26 — 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 [ISO9075-1], and Subclause 15.9, "<leave statement>", respectively. Some actions taken by a condition handler might also cause execution of *LS* to be terminated; see Subclause 15.2, "<handler declaration>".



 

## Conformance Rules

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

## 15.11 <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* 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 13.4, "<SQL procedure statement>", in ISO/IEC 9075-2, are applied with *CCS* as *EXECUTING STATEMENT* and the execution of *WS* is repeated.

      NOTE 27 — 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 [ISO9075-1], and Subclause 15.9, "<leave statement>",



 

respectively. Some actions taken by a condition handler might also cause execution of *WS* to be terminated; see Subclause 15.2, "<handler declaration>".

## Conformance Rules

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

# 15.12 <repeat statement>

## Function

Repeat the execution of a statement until a specified condition becomes *True*.

## 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* 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 13.4, "<SQL procedure statement>", in ISO/IEC 9075-2, are applied with *CCS* as *EXECUTING STATEMENT* and then <search condition> is evaluated.

NOTE 28 — 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 [ISO9075-1], and Subclause 15.9, "<leave statement>", respectively. Some actions taken by a condition handler might also cause execution of *RS* to be terminated; see Subclause 15.2, "<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>.

# 15.13 <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* 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 V1 DT1;
DECLARE V2 DT2;
.
.
.
DECLARE Vn DTn;
DECLARE AT_END BOOLEAN DEFAULT FALSE;
DECLARE NOT_FOUND CONDITION FOR SQLSTATE '02000';
DECLARE OPEN_ERROR CONDITION;
DECLARE CN CS CURSOR FOR FCS
DECLARE EXIT HANDLER FOR OPEN_ERROR
   RESIGNAL;
BEGIN NOT ATOMIC
   DECLARE EXIT HANDLER FOR SQLEXCEPTION
     RESIGNAL OPEN_ERROR;
   OPEN CN;
END;
BL:LOOP
   BEGIN NOT ATOMIC
      DECLARE CONTINUE HANDLER FOR NOT_FOUND
      SET AT_END = TRUE;
      DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
         BEGIN NOT ATOMIC
            SET AT_END = TRUE;
            RESIGNAL;
         END;
      FETCH CN INTO V1, V2, ..., Vn;
    END;
    IF AT_END THEN
      LEAVE BL;
    END IF;
    SLL
  END LOOP BL;
  CLOSE CN;
```

Case:

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

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

b) Otherwise, *FS* is equivalent to:

```
BEGIN NOT ATOMIC
  COMMON_CODE
END
```

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>.



 

# 16 Dynamic SQL

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

## 16.1 <prepare statement>

*This Subclause modifies Subclause 20.7, "<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 text of GR 5) The syntactic substitutions specified in Subclause 15.5, "<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 5)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 <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 5)a)xxxi) If *DP* is the <assignment target> simply contained in a <singleton variable assignment> *SVA* of an <assignment statement>, 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 5)a)xxxi) If *DP* is the <value expression> simply contained in an <assignment source> in a <singleton variable assignment> *SVA* of an <assignment statement> 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 5)a)xxxi) If *DP* is a <case operand> or <when operand> simply contained in a <simple case statement> *CS*, or if *DP* represents the value of a subfield *SF* of the declared type of such a <case operand> or <when operand>, then:

a) Let *SDT* be the set union of the following sets of declared types:

i) The set consisting of the declared type of the <case operand>.

ii) The set consisting of the declared type of any <when operand> of *CS* that is a <row value predicand>.

iii) The set consisting of the declared types of any <row value predicand> simply contained in any <comparison predicate part 2>, <between predicate part 2>, <character like predicate part 2>, <octet like predicate part 2>, <similar predicate part 2>, <regex like predicate part 2>, <overlaps predicate part 2>, <distinct predicate part 2>, or <member predicate part 2> that is simply contained in *CS*.

iv) The set consisting of the declared row type of a <table subquery> simply contained in an <in predicate part 2>, <match predicate part 2>, or <quantified comparison predicate part 2> simply contained in *CS*.

v) The set consisting of the declared types of the <row value expression>s simply contained in an <in value list> simply contained in an <in predicate part 2> simply contained in *CS*.

NOTE 29 — The following “part 2” predicates do not have any value expressions with declared type information: <null predicate part 2>, <normalized predicate part 2>, <set predicate part 2>, <type predicate part 2>.

b) The Syntax Rules of Subclause 9.5, “Result of data type combinations”, in ISO/IEC 9075-2, are applied with *SDT* as *DTSET*; let *RT* be the *RESTYPE* returned from the application of those Syntax Rules.

c) Case:

i) If *DP* is a <case operand> or <when operand> simply contained in *CS*, then *DT* is *RT*.

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



 

## Conformance Rules

*No additional Conformance Rules.*

## 16.2 <input using clause>

*This Subclause modifies Subclause 20.11, "<input using clause>", in ISO/IEC 9075-2.*

### Function

Supply input values for an <SQL dynamic statement>.

### Format

```
No additional Format items.
```

### Syntax Rules

1) Replace SR 1) The <general value specification> immediately contained in <using argument> shall be either a <host parameter specification>, an <SQL parameter reference>, an <SQL variable reference>, or an <embedded variable specification>.

### Access Rules

*No additional Access Rules.*

### General Rules

*No additional General Rules.*

### Conformance Rules

*No additional Conformance Rules.*



 

## 16.3 <output using clause>

*This Subclause modifies Subclause 20.12, "<output using clause>", in ISO/IEC 9075-2.*

### Function

Supply output variables for an <SQL dynamic statement>.

### Format

```
No additional Format items.
```

### Syntax Rules

1) Replace SR 1) The <target specification> immediately contained in <into argument> shall be either a <host parameter specification>, an <SQL parameter reference>, an <SQL variable reference>, or an <embedded variable specification>.

### Access Rules

*No additional Access Rules.*

### General Rules

*No additional General Rules.*

### Conformance Rules

*No additional Conformance Rules.*

*(Blank page)*



 

# 17 Embedded SQL

*This Clause modifies Clause 21, "Embedded SQL", in ISO/IEC 9075-2.*

## 17.1 <embedded SQL host program>

*This Subclause modifies Subclause 21.1, "<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 Given an <identifier> *I* and a <host identifier> *HI*, it is implementation-defined whether *I* and *HI* are equivalent.

2) Insert this SR An <SQL variable name> contained in an <SQL variable declaration> that is contained in an <embedded SQL host program> shall not be equivalent to any <host identifier> that is contained in an <embedded SQL statement> in an <embedded SQL host program>.

3) Insert this SR If a <handler declaration> is 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 22)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 22)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 23)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 21.2, "<embedded exception declaration>", in [ISO9075-2].

## Access Rules

*No additional Access Rules.*

## General Rules

*No additional General Rules.*

## Conformance Rules

*No additional Conformance Rules.*



 

# 18 Diagnostics management

*This Clause modifies Clause 23, "Diagnostics management", in ISO/IEC 9075-2.*

## 18.1 <get diagnostics statement>

*This Subclause modifies Subclause 23.1, "<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 [ISO9075-2]
  | CONDITION_IDENTIFIER
```

### Syntax Rules

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

2) *Table 1, "<condition information item name>s for use with <get diagnostics statement>", modifies Table 36, "<condition information item name>s for use with <get diagnostics statement>", in [ISO9075-2].*

**Table 1 — <condition information item name>s for use with <get diagnostics statement>**

| <identifier> | Declared Type |
|---|---|
| *All alternatives from [ISO9075-2]* | |
| CONDITION_IDENTIFIER | variable-length character string with implementation-defined maximum length |

## 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 37, "SQL-statement codes", in [ISO9075-2].*

**Table 2 — SQL-statement codes**

| SQL-statement | Identifier | Code |
|---|---|---|
| *All alternatives from [ISO9075-2]* | | |
| <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 |
| <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 [ISO9075-2], but instead are set as specified in Subclause 18.2, "<signal statement>", and Subclause 18.3, "<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>.

## 18.2 <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, "<condition information item name>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 30 — "considered to be defined for" is defined in Subclause 15.3, "<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 23.2, "Pushing and popping the diagnostics area stack", in [ISO9075-2], 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>.

## 18.3 <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 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.

### 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 23.2, "Pushing and popping the diagnostics area stack", in [ISO9075-2], 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 <signal value> is specified, then:

   a) The statement information item NUMBER in *SA* is incremented by 1 (one).

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



 

c) Let *n* be the maximum number of condition areas for diagnostics areas. For *i* ranging from *n*–1 to 1 (one), copy the contents of the i-th condition area into the *i*+1-th condition area.

NOTE 31 — After evaluation of the preceding General Rule, the first and second condition areas in *DA* have identical contents; *CA* still identifies the first condition area.

d) The statement information item COMMAND_FUNCTION in *SA* is set to 'RESIGNAL'. The statement information item DYNAMIC_FUNCTION in *SA* is set to a zero-length string. The condition information item CONDITION_IDENTIFIER in *CA* is set to *CN*.

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 32 — "considered to be defined" is defined in Subclause 15.3, "<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.

5) 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.

6) Case:

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

i) Let *S* be that value.

ii) If a handler *H* is the most appropriate handler for *S*, then the General Rules of Subclause 8.1, "Handler execution", are applied with *H* as *MOST APPROPRIATE HANDLER*.

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

NOTE 33 — 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 the General Rules of Subclause 8.1, "Handler execution", are applied with *H* as *MOST APPROPRIATE HANDLER*.

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>.



 

# 19 Information Schema

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

## 19.1 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 MCU.MODULE_CATALOG, MCU.MODULE_SCHEMA, MCU.MODULE_NAME,
           MCU.TABLE_CATALOG, MCU.TABLE_SCHEMA, MCU.TABLE_NAME, MCU.COLUMN_NAME
    FROM DEFINITION_SCHEMA.MODULE_COLUMN_USAGE AS MCU
      JOIN
         DEFINITION_SCHEMA.SCHEMATA AS S
      ON ( ( MCU.TABLE_CATALOG, MCU.TABLE_SCHEMA ) =
           ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
    WHERE ( S.SCHEMA_OWNER = CURRENT_USER
          OR
            S.SCHEMA_OWNER IN
            ( SELECT ER.ROLE_NAME
              FROM ENABLED_ROLES AS ER ) )
       AND
          MCU.MODULE_CATALOG =
          ( SELECT ISCN.CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME AS ISCN );
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.

## 19.2 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 IN
                ( SELECT ROLE_NAME
                  FROM ENABLED_ROLES )
            OR
              GRANTOR = CURRENT_USER
            OR
              GRANTOR IN
                ( SELECT ROLE_NAME
                  FROM ENABLED_ROLES ) )
          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_PRIVILEGES.



 

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

## 19.4 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, CREATED
    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.

## 19.5 PARAMETERS view

*This Subclause modifies Subclause 5.37, "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 clause of the VIEW definition with:

```
WHERE ( ( ( R.MODULE_CATALOG, R.MODULE_SCHEMA, R.MODULE_NAME ) IS NULL
        AND
          ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME ) IN
          ( SELECT RP.SPECIFIC_CATALOG, RP.SPECIFIC_SCHEMA, RP.SPECIFIC_NAME
            FROM DEFINITION_SCHEMA.ROUTINE_PRIVILEGES AS RP
            WHERE ( RP.GRANTEE IN
                    ( 'PUBLIC', CURRENT_USER )
                 OR
                    RP.GRANTEE IN
                    ( SELECT ER.ROLE_NAME
                      FROM ENABLED_ROLES AS ER ) ) ) )
      OR
        ( ( R.MODULE_CATALOG, R.MODULE_SCHEMA, R.MODULE_NAME ) IS NOT NULL
        AND
          ( R.MODULE_CATALOG, R.MODULE_SCHEMA, R.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 ER.ROLE_NAME
                      FROM ENABLED_ROLES AS ER ) ) ) ) )
  AND SPECIFIC_CATALOG
    = ( SELECT ISCN.CATALOG_NAME
        FROM INFORMATION_SCHEMA_CATALOG_NAME AS ISCN );
```

### Conformance Rules

*No additional Conformance Rules.*