INTERNATIONAL STANDARD

ISO/IEC 9075-4

Second edition
1999-12-01

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 9075-4:1999(E)

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 734 10 79
E-mail copyright@iso.ch
Web www.iso.ch

Printed in Switzerland

Contents

Page

TABLES

Tables **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. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75% of the national bodies casting a vote.

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

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

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

— *Part 1: Framework (SQL/Framework)*

— *Part 2: Foundation (SQL/Foundation)*

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

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

— *Part 5: Host Language Bindings (SQL/Bindings)*

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

Introduction

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

24) Annex D, "Deprecated features", is an informative Annex. It lists features that the responsible Technical Committee intends will not appear in a future revised version of ISO/IEC 9075.

25) Annex E, "Incompatibilities with ISO/IEC 9075:1992", is an informative Annex. It lists the incompatibilities between this edition of this part of ISO/IEC 9075 and ISO/IEC 9075-4:1996.

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

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

Information technology — Database languages — SQL —

Part 4:
Persistent Stored Modules (SQL/PSM)

1 Scope

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

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

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

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

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

— The specification of statements to signal and resignal conditions.

— The declaration of local cursors.

— The declaration of local variables.

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

NOTE 1 – The context for this part of ISO/IEC 9075 is described by the Reference Model of Data Management (ISO/IEC 10032:1993).

2 Normative references

The following standards contain provisions that, through reference in this text, constitute provisions of this part of ISO/IEC 9075. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 9075 are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO/IEC 8652:1995, *Information technology — Programming languages — Ada.*

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

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

ISO/IEC 9075-3:1996, *Information technology — Database languages — SQL — Part 3: Call-Level Interface (SQL/CLI).*

ISO/IEC 9075-5, *Information technology — Database languages — SQL — Part 5: Host Language Bindings (SQL/Bindings).*

3 Definitions, notations, and conventions

3.1 Definitions

3.1.1 Definitions provided in Part 4

Insert this paragraph For the purposes of this part of ISO/IEC 9075, the definitions given in ISO/IEC 9075-1 and ISO/IEC 9075-2 apply.

3.2 Notations

Insert this paragraph The syntax notation used in this part of ISO/IEC 9075 is an extended version of BNF ("Backus Normal Form" or "Backus Naur Form"). This version of BNF is fully described in Subclause 6.1, "Notation", of ISO/IEC 9075-1.

3.3 Conventions

Insert this paragraph Except as otherwise specified in this part of ISO/IEC 9075, the conventions used in this part of ISO/IEC 9075 are identical to those described in ISO/IEC 9075-1 and ISO/IEC 9075-2.

3.3.1 Use of terms

3.3.1.1 Exceptions

Modified paragraph The phrase "an exception condition is raised:", followed by the name of a condition, is used in General Rules and elsewhere to indicate that:

— The execution of a statement is unsuccessful.

— The application of General Rules, other than those of Subclause 10.4, "<routine invocation>", in ISO/IEC 9075-2, Subclause 11.2, "<SQL procedure statement>", Subclause 17.1, "<direct SQL statement>", in ISO/IEC 9075-5, Subclause 13.1, "<compound statement>", and Subclause 13.2, "<handler declaration>", may be terminated.

— Diagnostic information is to be made available.

— Execution of the statement is to have no effect on SQL-data or schemas.

Insert this paragraph The phrase "*C* is re-raised by *S*" is used in General Rules and elsewhere to indicate that *C*, a condition raised by an SQL-statement executed during execution of *S*, is raised again by *S*.

3.3.1.2 Other terms

Insert this paragraph An SQL-statement *S1* may be said to be executed as a *direct result of executing an SQL-statement* if *S1* is the SQL-statement contained in an <externally-invoked procedure> or <SQL-invoked routine> that has been executed.

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

Insert this paragraph The phrase "The scope of a <handler declaration> contained in a/an *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.

3.3.2 Relationships to other parts of ISO/IEC 9075

3.3.2.1 Clause, Subclause, and Table relationships

Table 1—Clause, Subclause, and Table relationships

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Clause 1, "Scope"	Clause 1, "Scope"	ISO/IEC 9075-2
Clause 2, "Normative references"	Clause 2, "Normative references"	ISO/IEC 9075-2
Clause 3, "Definitions, notations, and conventions"	Clause 3, "Definitions, notations, and conventions"	ISO/IEC 9075-2
Subclause 3.1, "Definitions"	Subclause 3.1, "Definitions"	ISO/IEC 9075-2
Subclause 3.1.1, "Definitions provided in Part 4"	*(none)*	*(none)*
Subclause 3.3, "Conventions"	Subclause 3.3, "Conventions"	ISO/IEC 9075-2
Subclause 3.3.1, "Use of terms"	Subclause 3.3.1, "Use of terms"	ISO/IEC 9075-2
Subclause 3.3.1.1, "Exceptions"	Subclause 6.2.3.1, "Exceptions"	ISO/IEC 9075-1
Subclause 3.3.1.2, "Other terms"	Subclause 6.2.3.7, "Other terms"	ISO/IEC 9075-1
Subclause 3.3.2, "Relationships to other parts of ISO/IEC 9075"	*(none)*	*(none)*
Subclause 3.3.2.1, "Clause, Subclause, and Table relationships"	*(none)*	*(none)*
Subclause 3.4, "Object identifier for Database Language SQL"	Subclause 6.3, "Object identifier for Database Language SQL"	ISO/IEC 9075-1
Clause 4, "Concepts"	Clause 4, "Concepts"	ISO/IEC 9075-2
Subclause 4.1, "SQL-server modules"	*(none)*	*(none)*

Table 1—Clause, Subclause, and Table relationships (Cont.)

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Subclause 4.2, “SQL-invoked routines”	Subclause 4.23, "SQL-invoked routines"	ISO/IEC 9075-2
Subclause 4.3, “SQL-paths”	Subclause 4.25, "SQL-paths"	ISO/IEC 9075-2
Subclause 4.4, “Tables”	Subclause 4.16, "Tables"	ISO/IEC 9075-2
Subclause 4.5, “SQL-schemas”	Subclause 4.20, "SQL-schemas"	ISO/IEC 9075-2
Subclause 4.6, “Host parameters”	Subclause 4.26, "Host parameters"	ISO/IEC 9075-2
Subclause 4.6.1, “Status parameters”	Subclause 4.26.1, "Status parameters"	ISO/IEC 9075-2
Subclause 4.7, “Diagnostics area”	Subclause 4.27, "Diagnostics area"	ISO/IEC 9075-2
Subclause 4.8, “Cursors”	Subclause 4.29, "Cursors"	ISO/IEC 9075-2
Subclause 4.9, “Condition handling”	*(none)*	*(none)*
Subclause 4.10, “SQL-statements”	Subclause 4.30, "SQL-statements"	ISO/IEC 9075-2
Subclause 4.10.1, “SQL-statements classified by function”	Subclause 4.30.2, "SQL-statements classified by function"	ISO/IEC 9075-2
Subclause 4.10.2, “Embeddable SQL-statements”	Subclause 4.6.4, "Embeddable SQL-statements"	ISO/IEC 9075-5
Subclause 4.10.3, “Directly executable SQL-statements”	Subclause 4.6.6, "Directly executable SQL-statements"	ISO/IEC 9075-5
Subclause 4.10.4, “Iterated SQL-statements”	*(none)*	*(none)*
Subclause 4.10.5, “SQL-statements and transaction states”	Subclause 4.30.3, "SQL-statements and transaction states"	ISO/IEC 9075-2
Subclause 4.10.6, “Compound statements”	*(none)*	*(none)*
Subclause 4.10.7, “SQL-statement atomicity”	*(none)*	*(none)*
Subclause 4.11, “Basic security model”	Subclause 4.31, "Basic security model"	ISO/IEC 9075-2
Subclause 4.11.1, “Privileges”	Subclause 4.31, "Basic security model"	ISO/IEC 9075-2
Clause 5, “Lexical elements”	Clause 5, "Lexical elements"	ISO/IEC 9075-2
Subclause 5.1, “<token> and <separator>”	Subclause 5.2, "<token> and <separator>"	ISO/IEC 9075-2
Subclause 5.2, “Names and identifiers”	Subclause 5.4, "Names and identifiers"	ISO/IEC 9075-2
Clause 6, “Scalar expressions”	Clause 6, "Scalar expressions"	ISO/IEC 9075-2

Table 1—Clause, Subclause, and Table relationships (Cont.)

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Subclause 6.1, "<value specification> and <target specification>"	Subclause 6.3, "<value specification> and <target specification>"	ISO/IEC 9075-2
Subclause 6.2, "<identifier chain>"	Subclause 6.5, "<identifier chain>"	ISO/IEC 9075-2
Subclause 6.3, "<SQL variable reference>"	*(none)*	*(none)*
Clause 7, "Query expressions"	Clause 7, "Query expressions"	ISO/IEC 9075-2
Subclause 7.1, "<query specification>"	Subclause 7.11, "<query specification>"	ISO/IEC 9075-2
Clause 8, "Additional common elements"	Clause 10, "Additional common elements"	ISO/IEC 9075-2
Subclause 8.1, "<routine invocation>"	Subclause 10.4, "<routine invocation>"	ISO/IEC 9075-2
Subclause 8.2, "<privileges>"	Subclause 10.5, "<privileges>"	ISO/IEC 9075-2
Subclause 8.3, "<sqlstate value>"	*(none)*	*(none)*
Clause 9, "Schema definition and manipulation"	Clause 11, "Schema definition and manipulation"	ISO/IEC 9075-2
Subclause 9.1, "<schema definition>"	Subclause 11.1, "<schema definition>"	ISO/IEC 9075-2
Subclause 9.2, "<drop schema statement>"	Subclause 11.2, "<drop schema statement>"	ISO/IEC 9075-2
Subclause 9.3, "<default clause>"	Subclause 11.5, "<default clause>"	ISO/IEC 9075-2
Subclause 9.4, "<drop column scope clause>"	Subclause 11.16, "<drop column scope clause>"	ISO/IEC 9075-2
Subclause 9.5, "<drop column definition>"	Subclause 11.17, "<drop column definition>"	ISO/IEC 9075-2
Subclause 9.6, "<drop table constraint definition>"	Subclause 11.19, "<drop table constraint definition>"	ISO/IEC 9075-2
Subclause 9.7, "<drop table statement>"	Subclause 11.20, "<drop table statement>"	ISO/IEC 9075-2
Subclause 9.8, "<view definition>"	Subclause 11.21, "<view definition>"	ISO/IEC 9075-2
Subclause 9.9, "<drop view statement>"	Subclause 11.22, "<drop view statement>"	ISO/IEC 9075-2
Subclause 9.10, "<drop domain statement>"	Subclause 11.29, "<drop domain statement>"	ISO/IEC 9075-2
Subclause 9.11, "<drop character set statement>"	Subclause 11.31, "<drop character set statement>"	ISO/IEC 9075-2
Subclause 9.12, "<drop collation statement>"	Subclause 11.33, "<drop collation statement>"	ISO/IEC 9075-2

Table 1—Clause, Subclause, and Table relationships (Cont.)

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Subclause 9.13, "<drop translation statement>"	Subclause 11.35, "<drop translation statement>"	ISO/IEC 9075-2
Subclause 9.14, "<assertion definition>"	Subclause 11.36, "<assertion definition>"	ISO/IEC 9075-2
Subclause 9.15, "<drop assertion statement>"	Subclause 11.37, "<drop assertion statement>"	ISO/IEC 9075-2
Subclause 9.16, "<trigger definition>"	Subclause 11.38, "<trigger definition>"	ISO/IEC 9075-2
Subclause 9.17, "<drop user-defined ordering statement>"	Subclause 11.55, "<drop user-defined ordering statement>"	ISO/IEC 9075-2
Subclause 9.18, "<SQL-server module definition>"	*(none)*	*(none)*
Subclause 9.19, "<drop module statement>"	*(none)*	*(none)*
Subclause 9.21, "<SQL-invoked routine>"	Subclause 11.49, "<SQL-invoked routine>"	ISO/IEC 9075-2
Subclause 9.22, "<drop routine statement>"	Subclause 11.51, "<drop routine statement>"	ISO/IEC 9075-2
Clause 10, "Access control"	Clause 12, "Access control"	ISO/IEC 9075-2
Subclause 10.1, "<grant statement>"	Subclause 12.1, "<grant statement>"	ISO/IEC 9075-2
Subclause 10.2, "<revoke statement>"	Subclause 12.6, "<revoke statement>"	ISO/IEC 9075-2
Clause 11, "SQL-client modules"	Subclause 13.1, "<SQL-client module definition>"	ISO/IEC 9075-2
Subclause 11.1, "Calls to an <externally-invoked procedure>"	Subclause 13.4, "Calls to an <externally-invoked procedure>"	ISO/IEC 9075-2
Subclause 11.2, "<SQL procedure statement>"	Subclause 13.5, "<SQL procedure statement>"	ISO/IEC 9075-2
Clause 12, "Data manipulation"	Clause 14, "Data manipulation"	ISO/IEC 9075-2
Subclause 12.1, "<open statement>"	Subclause 14.2, "<open statement>"	ISO/IEC 9075-2
Subclause 12.2, "<fetch statement>"	Subclause 14.3, "<fetch statement>"	ISO/IEC 9075-2
Subclause 12.3, "<close statement>"	Subclause 14.4, "<close statement>"	ISO/IEC 9075-2
Subclause 12.4, "<select statement: single row>"	Subclause 14.5, "<select statement: single row>"	ISO/IEC 9075-2
Subclause 12.5, "<delete statement: positioned>"	Subclause 14.6, "<delete statement: positioned>"	ISO/IEC 9075-2
Subclause 12.6, "<update statement: positioned>"	Subclause 14.9, "<update statement: positioned>"	ISO/IEC 9075-2

Table 1—Clause, Subclause, and Table relationships (Cont.)

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Subclause 12.7, "<temporary table declaration>"	Subclause 14.11, "<temporary table declaration>"	ISO/IEC 9075-2
Clause 13, "Control statements"	*(none)*	*(none)*
Subclause 13.1, "<compound statement>"	*(none)*	*(none)*
Subclause 13.2, "<handler declaration>"	*(none)*	*(none)*
Subclause 13.3, "<condition declaration>"	*(none)*	*(none)*
Subclause 13.4, "<SQL variable declaration>"	*(none)*	*(none)*
Subclause 13.5, "<assignment statement>"	*(none)*	*(none)*
Subclause 13.6, "<case statement>"	*(none)*	*(none)*
Subclause 13.7, "<if statement>"	*(none)*	*(none)*
Subclause 13.8, "<iterate statement>"	*(none)*	*(none)*
Subclause 13.9, "<leave statement>"	*(none)*	*(none)*
Subclause 13.10, "<loop statement>"	*(none)*	*(none)*
Subclause 13.11, "<while statement>"	*(none)*	*(none)*
Subclause 13.12, "<repeat statement>"	*(none)*	*(none)*
Subclause 13.13, "<for statement>"	*(none)*	*(none)*
Clause 14, "Dynamic SQL"	Clause 15, "Dynamic SQL"	ISO/IEC 9075-5
Subclause 14.1, "<prepare statement>"	Subclause 15.6, "<prepare statement>"	ISO/IEC 9075-5
Clause 15, "Embedded SQL"	Clause 16, "Embedded SQL"	ISO/IEC 9075-5
Subclause 15.1, "<embedded SQL host program>"	Subclause 16.1, "<embedded SQL host program>"	ISO/IEC 9075-5
Clause 16, "Diagnostics management"	Clause 19, "Diagnostics management"	ISO/IEC 9075-2
Subclause 16.1, "<get diagnostics statement>"	Subclause 19.1, "<get diagnostics statement>"	ISO/IEC 9075-2
Subclause 16.2, "<signal statement>"	*(none)*	*(none)*
Subclause 16.3, "<resignal statement>"	*(none)*	*(none)*

Table 1—Clause, Subclause, and Table relationships (Cont.)

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Clause 17, "Information Schema"	Clause 20, "Information Schema"	ISO/IEC 9075-2
Subclause 17.1, "MODULE_COLUMN_USAGE view"	*(none)*	*(none)*
Subclause 17.2, "MODULE_PRIVILEGES view"	*(none)*	*(none)*
Subclause 17.3, "MODULE_TABLE_USAGE view"	*(none)*	*(none)*
Subclause 17.4, "MODULES view"	*(none)*	*(none)*
Clause 18, "Definition Schema"	Clause 21, "Definition Schema"	ISO/IEC 9075-2
Subclause 18.1, "MODULE_COLUMN_USAGE base table"	*(none)*	*(none)*
Subclause 18.2, "MODULE_PRIVILEGES base table"	*(none)*	*(none)*
Subclause 18.3, "MODULE_TABLE_USAGE base table"	*(none)*	*(none)*
Subclause 18.4, "MODULES base table"	*(none)*	*(none)*
Clause 19, "Status codes"	Clause 22, "Status codes"	ISO/IEC 9075-2
Subclause 19.1, "SQLSTATE"	Subclause 22.1, "SQLSTATE"	ISO/IEC 9075-2
Clause 20, "Conformance"	Clause 8, "Conformance"	ISO/IEC 9075-1
Subclause 20.1, "Claims of conformance"	Subclause 8.1.5, "Claims of conformance"	ISO/IEC 9075-1
Annex A, "SQL Conformance Summary"	Appendix A, "SQL Conformance Summary"	ISO/IEC 9075-2
Annex B, "Implementation-defined elements"	Appendix B, "Implementation-defined elements"	ISO/IEC 9075-2
Annex C, "Implementation-dependent elements"	Appendix C, "Implementation-dependent elements"	ISO/IEC 9075-2
Annex D, "Deprecated features"	Appendix D, "Deprecated features"	ISO/IEC 9075-2
Annex E, "Incompatibilities with ISO/IEC 9075:1992"	Appendix E, "Incompatibilities with ISO/IEC 9075:1992 and ISO/IEC 9075-4:1996"	ISO/IEC 9075-2
Annex F, "SQL Feature Taxonomy"	Appendix F, "SQL feature and package taxonomy"	ISO/IEC 9075-2
Table 1, "Clause, Subclause, and Table relationships"	*(none)*	*(none)*
Table 2, "<identifier>s for use with <get diagnostics statement>"	Table 25, "<identifier>s for use with <get diagnostics statement>"	ISO/IEC 9075-2

Table 1—Clause, Subclause, and Table relationships (Cont.)

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Table 3, "SQL-statement codes for use in the diagnostics area"	Table 26, "SQL-statement codes"	ISO/IEC 9075-2
Table 4, "SQLSTATE class and subclass values"	Table 27, "SQLSTATE class and subclass values"	ISO/IEC 9075-2
Table 5, "SQL/PSM feature taxonomy for features outside Core SQL"	Table 13, "SQL/Bindings feature taxonomy for features outside Core SQL"	ISO/IEC 9075-5

3.4 Object identifier for Database Language SQL

The object identifier for Database Language SQL is defined in ISO/IEC 9075-1 in Subclause 6.3, "Object identifier for Database Language SQL", with the following additions:

Format

```
<Part 4 yes> ::= <Part 4 conformance> <Part 4 module>

<Part 4 conformance> ::= 4 | sqlpsm1999 <left paren> 4 <right paren>

<Part 4 module> ::=  <Part 4 module yes> | <Part 4 module no>

<Part 4 module yes> ::= 1 | moduleyes <left paren> 1 <right paren>

<Part 4 module no> ::= 0 | moduleno <left paren> 0 <right paren>
```

Syntax Rules

1) Specification of <Part 4 yes> implies that conformance to ISO/IEC 9075-4 is claimed.

2) Specification of <Part 4 module yes> implies that conformance to Feature P01, "Stored modules", is claimed.

3) Specification of <Part 4 module no> implies that conformance to Feature P01, "Stored modules", is not claimed.

4 Concepts

4.1 SQL-server modules

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

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

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

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

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

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

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

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

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

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

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

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

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

4.2 SQL-invoked routines

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

Replace 22nd paragraph An SQL-invoked routine has a *routine SQL-path*, which is inherited from its containing SQL-server module or schema, the current SQL-session, or the containing SQL-client module.

Insert in the 23rd 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.

4.3 SQL-paths

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

4.4 Tables

Replace 16th paragraph A declared local temporary table may be declared in an SQL-client module or in an SQL-server module.

Insert after 16th paragraph A declared local temporary table that is declared in an SQL-server module is a named table defined by a <temporary table declaration> that is effectively materialized the first time any <module routine> in the <SQL-server module definition> that contains the <temporary table declaration> is executed.

Insert after 16th paragraph A declared local temporary table is accessible only by <module routine>s in the <SQL-server module definition> that contains the <temporary table declaration>. The effective <schema name> of the <schema qualified name> of the declared local temporary table may be thought of as the implementation-dependent SQL-session identifier associated with the SQL-session and the name of the <SQL-server module definition> that contains the <temporary table declaration>.

4.5 SQL-schemas

Replace 2nd paragraph In this part of ISO/IEC 9075, the term "schema" is used only in the sense of SQL-schema. Each component descriptor is either a domain descriptor, a base table descriptor, a view descriptor, a constraint descriptor, a privilege descriptor, a character set descriptor, a collation descriptor, a translation descriptor, a trigger descriptor, a user-defined type descriptor, an SQL-server module descriptor, or an SQL-invoked routine descriptor. The persistent objects described by

the descriptors are said to be *owned by* or to have been *created by* the <authorization identifier> of the schema.

4.6 Host parameters

4.6.1 Status parameters

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

Insert this paragraph An implementation shall place information about a completion or exception condition that causes a handler to be activated into the diagnostics area prior to activating the handler. If other conditions are raised, then it is implementation-defined whether the implementation places information about them into the diagnostics area.

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

4.8 Cursors

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

4.9 Condition handling

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

<handler declaration>s specify the handling of exception and completion conditions. <handler declaration>s can be specified in <compound statement>s. The scope of a <handler declaration> specified in a <compound statement> is that <compound statement> excluding every <SQL schema statement> contained in that <compound statement>.

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.

4.9 Condition handling

A condition represents an error or informational state caused by execution of an <SQL procedure statement>. Conditions are raised to provide information in the 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 SQLSTATE value. If a <condition declaration> does not specify an SQLSTATE value, it declares a *user-defined exception condition*. <condition name>s can be used in <handler declaration>s, <signal statement>s, and <resignal statement>s.

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

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

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

If there is no most appropriate handler and the condition is a completion condition, then execution is resumed as specified in Subclause 6.2.3.1, "Exceptions", in ISO/IEC 9075-1. This type of completion condition is called an *unhandled completion condition*.

A handler type specifies CONTINUE, EXIT, or UNDO.

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

— Execute the handler action.

— Return control to the SQL-statement following the executing statement that raised the condition. If the SQL-statement that raised the condition is an <SQL control statement>, then "the following SQL-statements" do not include the <SQL procedure statement>s contained within the <SQL control statement>.

NOTE 2 — For example, if the executing statement that raised the condition is an <if statement>, control is returned to the SQL-statement *following* the <if statement>.

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

— Execute the handler action.

— 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:

— 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.

— Return control to the end of the <compound statement> for which the handler was created.

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

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

4.10 SQL-statements

4.10.1 SQL-statements classified by function

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

— SQL-control declarations

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

— <SQL-server module definition>

— <alter module statement>

— <drop module statement>

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>

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

— <condition declaration>

— <handler declaration>

— <SQL variable declaration>

4.10 SQL-statements

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

— <signal statement>

— <resignal statement>

4.10.2 Embeddable SQL-statements

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>.

4.10.3 Directly executable SQL-statements

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

— All SQL-control statements

4.10.4 Iterated SQL-statements

The following are the iterated SQL-statements:

— <loop statement>

— <while statement>

— <repeat statement>

— <for statement>

4.10.5 SQL-statements and transaction states

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>

Insert this paragraph If the initiation of an SQL-transaction occurs in an atomic execution context, and an SQL-transaction has already been completed in this atomic execution context, then an exception condition is raised: *invalid transaction initiation*.

4.10.6 Compound statements

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

4.10.7 SQL-statement atomicity

Augment 1st paragraph The execution of <compound statement>s that specify ATOMIC is atomic.

4.11 Basic security model

4.11.1 Privileges

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

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

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

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

NOTE 4 – "schema-level routine" is defined in Subclause 11.49, "<SQL-invoked routine>", in ISO/IEC 9075-2.

5 Lexical elements

5.1 <token> and <separator>

Function

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

Format

```
<non-reserved word> ::=
        !! All alternatives from ISO/IEC 9075-2
      | !! All alternatives from ISO/IEC 9075-5
      | CONDITION_IDENTIFIER

<reserved word> ::=
        !! All alternatives from ISO/IEC 9075-2
      | !! All alternatives from ISO/IEC 9075-5

      | CONDITION

      | DO

      | ELSEIF | EXIT

      | HANDLER

      | IF | ITERATE

      | LEAVE | LOOP

      | REDO | REPEAT | RESIGNAL

      | SIGNAL

      | UNDO | UNTIL

      | WHILE
```


Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

5.2 Names and identifiers

Function

Specify names.

Format

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

<SQL variable name> ::=
      <identifier>

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

Syntax Rules

1) Replace ISO/IEC 9075-5 SR1) 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 SR4)a) 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 ISO/IEC 9075-5 SR2) If the <schema qualified name> is contained, without an intervening <schema definition> or <SQL-server module definition>, in a <preparable statement> that is prepared in the current SQL-session by an <execute immediate statement> or a <prepare statement> or in a <direct SQL statement> that is invoked directly, then the default <unqualified schema name> for the SQL-session is implicit.

4) Insert before SR11)a) If the <schema qualified name> is contained in an <SQL-server module definition> without an intervening <schema definition>, then the <default schema name> that is specified or implicit in the <SQL-server module definition> is implicit.

5) Replace SR8) Case:

   a) If <user-defined type name> *UDTN* with a <qualified identifier> QI is simply contained in <user-defined type>, then

      Case:

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

ii) Otherwise:

1) Case:

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

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

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

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

2) Let *N* be the number of <schema name>s in *DP*. Let S_i, 1 (one) $\leq i \leq N$, 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) $\leq i \leq N$, 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) Case:

a) If the set of subject types contains exactly one user-defined type *UDT*, then *UDT* is the user-defined type identified by *UDTN*.

b) Otherwise, 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) Otherwise,

Case:

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

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

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

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

Access Rules

No additional Access Rules.

General Rules

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

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

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

6 Scalar expressions

6.1 <value specification> and <target specification>

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
       | !! All alternatives from ISO/IEC 9075-5
       | <SQL variable reference>

<simple value specification> ::=
         !! All alternatives from ISO/IEC 9075-2
       | !! All alternatives from ISO/IEC 9075-5
       | <SQL variable reference>

<target specification> ::=
         !! All alternatives from ISO/IEC 9075-2
       | !! All alternatives from ISO/IEC 9075-5
       | <SQL variable reference>

<simple target specification> ::=
         !! All alternatives from ISO/IEC 9075-2
       | !! All alternatives from ISO/IEC 9075-5
       | <SQL variable reference>
```

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

6.2 <identifier chain>

Function

Disambiguate a <period>-separated chain of identifiers.

Format

No additional Format items.

Syntax Rules

1) Replace introductory paragraph of SR7) For at most one j between 1 and M, PIC_J is called the *basis* of IC, and j is called the *basis length* of IC. The *referent* of the basis is a column C of a table, an SQL parameter SP, or an SQL variable SV. The basis, basis length, basis scope and basis referent of IC are determined as follows:

2) Replace SR7)a) If $N = 1$, then IC shall be contained within the scope of one or more exposed <table or query name>s or <correlation name>s whose associated tables include a column whose <identifier> is equivalent to I_1 or within the scope of a <routine name> whose associated <SQL parameter declaration list> includes an SQL parameter whose <identifier> is equivalent to I_1 or within the scope of one or more <beginning label>s whose associated <local declaration list> includes an SQL variable whose <identifier> is equivalent to I_1. Let the phrase *possible scope tags* denote those exposed <table name>s, <correlation name>s, and <routine name>s.

3) Insert after SR7)a)i)2) If the innermost possible qualifier is a <beginning label>, then let SV be the SQL variable whose <identifier> is equivalent to I_1. PIC_1 is the basis of IC, the basis length is 1 (one), the basis scope is the scope of SP, and the basis referent is SV.

4) Insert before SR7)b)ii) If IC is contained within the scope of a <beginning label> whose associated <local declaration list> includes an SQL variable SV whose <identifier> 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.

5) Replace SR9) If $BL < N$, then let TIC be the <value expression primary>:

(PIC_{BL}) <period> I_{BL+1} <period> . . . <period> I_N

The Syntax Rules of Subclause 6.23, "<value expression>", are applied to TIC, yielding a column reference, an SQL parameter reference, or an SQL variable reference, and ($N-BL$) <field reference>s, <method invocation>s, <modified field reference>s, and/or <mutator reference>s.

NOTE 5 – In this transformation, (PIC_{BL}) is interpreted as a <value expression primary> of the form <left paren> <value expression> <right paren>. PIC_{BL} is a <value expression> that is a <value expression primary> that is an <unsigned value specification> that is either a <column reference> or an <SQL parameter reference>. The identifiers I_{BL+1}, . . . , I_N are parsed using the Syntax Rules of <field reference> and <method invocation>. Alternatively, on the left-hand side of an <assignment statement>, (PIC_{BL}) is interpreted as "<left paren> <target specification> <right paren>", and the identifiers I_{BL+1}, . . . , I_N are parsed using the Syntax Rules of <modified field reference> and <mutator reference>.

6) Insert after SR12) 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*.

6.3 <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.

7 Query expressions

7.1 <query specification>

Function

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

Format

No additional Format items.

Syntax Rules

1) Insert after SR7)f)i) If *IC* is contained within the scope of a <beginning label> whose associated <local declaration list> includes an SQL variable *SV* whose <identifier> is equivalent to I_1, then PIC_1 is a candidate basis of *IC* and the scope of PIC_1 is the scope of *SP*.

2) Insert after SR2) in Part 5 An SQL variable.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

8 Additional common elements

8.1 <routine invocation>

Function

Invoke an SQL-invoked routine.

Format

No additional Format items.

Syntax Rules

1) Replace SR5) 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 of the <authorization identifier> that owns the containing schema include EXECUTE on *M*.

      ii) Otherwise, the applicable privileges of 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*.

   NOTE 6 – "applicable privileges" and "current privileges" are defined in Subclause 10.5, "<privileges>", in ISO/IEC 9075-2.

2) Insert before SR7)b)i)1)C)I) If *RI* is contained in an <SQL-server module definition>, then let *DP* be the SQL-path of that <SQL-server module definition>.

3) Replace SR7)b)i)1)C)I) If *RI* is contained in a <schema definition> without an intervening <SQL-server module definition>, then let *DP* be the SQL-path of that <schema definition>.

4) Insert before SR8)b)i)1)C)I) If *RI* is contained in an <SQL-server module definition>, then let *DP* be the SQL-path of that <SQL-server module definition>.

5) Replace SR8)b)i)1)C)I) If *RI* is contained in a <schema definition> without an intervening <SQL-server module definition>, then let *DP* be the SQL-path of that <schema definition>.

6) Replace SR8)c)i)4)B) If A_i is an <SQL variable name> or the <SQL parameter name> of an SQL parameter of an SQL-invoked routine, then P_i shall be assignable to A_i, according to the Syntax Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2, with A_i and P_i as *TARGET* and *VALUE*, respectively.

Access Rules

No additional Access Rules.

General Rules

1) Insert after GR6)c)i)

NOTE 7 – The identities of declared local temporary tables that are defined in <SQL-server module>s are not removed.

2) Replace SR10)b)i) If TS_i is an <SQL variable name> or a <host parameter specification>, then CPV_i is assigned to TS_i according to the rules of Subclause 9.1, "Retrieval assignment", in ISO/IEC 9075-2.

3) Replace SR10)b)i) If TS_i is an <SQL variable name> or the <SQL parameter name> of an SQL parameter of an SQL-invoked routine, then CPV_i is assigned to TS_i according to the rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2.

8.2 <privileges>

Function

Specify privileges.

Format

```
<object name> ::=
        !! All alternatives from ISO/IEC 9075-2
      | !! All alternatives from ISO/IEC 9075-5
      | MODULE <module name>
```

Syntax Rules

1) Replace SR7) If the object identified by <object name> of the <grant statement> or <revoke statement> is an SQL-invoked routine or an SQL-server module, then <privileges> shall specify EXECUTE; otherwise, EXECUTE shall not be specified.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

8.3 <sqlstate value>

Function

Specify an SQLSTATE value.

Format

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

Syntax Rules

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

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

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

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

4) *V* shall comprise either:

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

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

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

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

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

Access Rules

None.

General Rules

None.

9 Schema definition and manipulation

9.1 <schema definition>

Function

Define a schema.

Format

```
<schema element> ::=
        !! All alternatives from ISO/IEC 9075-2
      | !! All alternatives from ISO/IEC 9075-5
      | <SQL-server module definition>
```

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

9.2 <drop schema statement>

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 GR8) 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 GR11) 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 GR11) 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
```

9.3 <default clause>

Function

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

Format

No additional Format items.

Syntax Rules

1) Replace SR1) The subject data type of a <default clause> is the data type specified in the descriptor identified by the containing <column definition>, <domain definition>, <attribute definition>, <alter column definition>, or <alter domain statement>, or that defined by the <data type> specified in the containing <SQL variable declaration>.

Access Rules

No additional Access Rules.

General Rules

None.

9.4 <drop column scope clause>

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 SR3)d) The module descriptor of any SQL-server module.

Access Rules

None.

General Rules

1) Replace GR1) For every SQL-invoked routine *R* whose routine descriptor includes a <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 GR4) 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
   ```

9.5 <drop column definition>

Function

Destroy a column.

Format

No additional Format items.

Syntax Rules

1) Replace SR5) If RESTRICT is specified, then *C* shall not be referenced in any of the following:

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

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

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

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

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

   NOTE 8 – A <drop column definition> that does not specify CASCADE will fail if there are any references to that column resulting from the use of CORRESPONDING, NATURAL, SELECT * (except where contained in an exists predicate>), or REFERENCES without a <reference column list> in its <referenced table and columns>.

   NOTE 9 – If CASCADE is specified, then any such dependent object will be dropped by the execution of the <revoke statement> specified in the General Rules of this Subclause.

   NOTE 10 – *CN* may be contained in an implicit trigger column list of a trigger descriptor.

Access Rules

No additional Access Rules.

General Rules

1) Replace GR4) Let *R* be any SQL-invoked routine whose routine descriptor contains the <column name> of *C* 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
```

2) Insert after GR4) Let *SSM* be any SQL-server module whose module descriptor includes the <column name> of *C* 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
```

9.6 <drop table constraint definition>

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 GR2) Let *R* be any SQL-invoked routine whose routine descriptor contains the <constraint name> of *TC* in the <SQL routine body>.

   Case:

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

      ```
      DROP MODULE MN CASCADE
      ```

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

      ```
      DROP SPECIFIC ROUTINE SN CASCADE
      ```

9.7 <drop table statement>

Function

Destroy a table.

Format

No additional Format items.

Syntax Rules

1) Insert after SR6) 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

1) Replace GR5) Let *R* be any SQL-invoked routine whose routine descriptor contains the <table name> of *T* in the <SQL routine body>.

   Case:

   a) If *R* 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
      ```

2) Insert after GR5) Let *SSM* be any SQL-server module whose module descriptor includes the <table name> of *T* 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
   ```

9.8 <view definition>

Function

Define a viewed table.

Format

No additional Format items.

Syntax Rules

1) Insert after SR2)

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

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

9.9 <drop view statement>

Function

Destroy a view.

Format

No additional Format items.

Syntax Rules

1) Insert after SR4) 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

1) Replace GR4) Let *R* be any SQL-invoked routine whose routine descriptor contains the <table name> of *V* 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
      ```

2) Insert after GR4) Let *SSM* be any SQL-server module whose module descriptor includes the <table name> of *V* 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
   ```

9.10 <drop domain statement>

Function

Destroy a domain.

Format

No additional Format items.

Syntax Rules

1) Insert after SR2) 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

1) Insert after GR1) Let *SSM* be any SQL-server module whose module descriptor includes the <column name> of *C* 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
```

9.11 <drop character set statement>

Function

Destroy a character set.

Format

No additional Format items.

Syntax Rules

1) Insert after SR4) *C* shall not be referenced in the module descriptor of any SQL-server module.

Access Rules

No additional Access Rules.

General Rules

1) Replace GR2) Let *R* be any SQL-invoked routine whose routine descriptor contains the <character set name> of *C* 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
```

2) Insert after GR2) Let *SSM* be any SQL-server module whose module descriptor includes the <character set name> of *C* 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
```

9.12 <drop collation statement>

Function

Destroy a collating sequence.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

1) Replace GR5) Let *R* be any SQL-invoked routine whose routine descriptor contains the <collation name> of *C* in the <SQL routine body> or the <SQL parameter declaration>s.

   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 GR5) Let *SSM* be any SQL-server module whose module descriptor includes the <collation name> of *C* 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
   ```

9.13 <drop translation statement>

Function

Destroy a character translation.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

1) Replace GR2) Let *R* be any SQL-invoked routine whose routine descriptor contains the <translation name> of *T* in the <SQL routine body>.

Case:

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

```
DROP MODULE MN CASCADE
```

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

```
DROP SPECIFIC ROUTINE SN CASCADE
```

9.14 <assertion definition>

Function

Specify an integrity constraint by means of an assertion and specify the initial default time for checking the assertion.

Format

No additional Format items.

Syntax Rules

1) Insert after SR4)

NOTE 12 – <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.

9.15 <drop assertion statement>

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 GR1) 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
```

9.16 <trigger definition>

Function

Defined triggered SQL-statements.

Format

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

NOTE 13 – The preceding production defining <triggered SQL statement> completely supersedes the definition in ISO/IEC 9075-2.

Syntax Rules

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

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

9.17 <drop user-defined ordering statement>

Function

Destroy a user-defined ordering method.

Format

No additional Format items.

Syntax Rules

1) Insert after SR5)d) The module descriptor of any SQL-server module.

Access Rules

No additional Access Rules.

General Rules

1) Replace GR1) Let *R* be any SQL-invoked routine that contains *P* in its <SQL routine body>.

   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 GR6) Let *SSM* be any SQL-server module whose module descriptor contains *P* and let *MN* be the <SQL-server module name> of *SSM*. The following <drop module statement> is effectively executed without further Access Rule checking:

   DROP MODULE *MN* CASCADE

9.18 <SQL-server module definition>

Function

Define an SQL-server module.

Format

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

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

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

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

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

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

Syntax Rules

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

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

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

   NOTE 14 – "Schema-level routine" is defined in Subclause 11.49, "<SQL-invoked routine>", in ISO/IEC 9075-2.

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

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

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

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

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

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

Access Rules

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

General Rules

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

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

   NOTE 15 – 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>.

9.19 <drop module statement>

Function

Destroy an SQL-server module.

Format

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

Syntax Rules

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

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

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

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

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

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

   d) Any trigger descriptor.

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

Access Rules

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

General Rules

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

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

2) The descriptor of *M* is destroyed.

9.20 <drop data type statement>

Function

Destroy a user-defined type.

Format

No additional Format items.

Syntax Rules

1) Insert after SR4)f)v) The module descriptor of any SQL-server module.

2) Insert after SR4)h)i)4) The module descriptor of any SQL-server module.

3) Insert after SR4)h)iv)4) The module descriptor of any SQL-server module.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

9.21 <SQL-invoked routine>

Function

Define an SQL-invoked routine.

Format

```
<SQL-invoked routine> ::=
         !! All alternatives from ISO/IEC 9075-2
       | !! All alternatives from ISO/IEC 9075-5
       | <module routine>

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

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

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

Syntax Rules

1) Replace SR5)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 SR5)m) 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 SR5)m) If <SQL-invoked routine> is contained in an <SQL-server module definition> and if *RN* contains a <schema name> *SN*, then *SN* shall be equivalent to the specified <schema name> of the containing <SQL-server module definition>. Let *S* be the SQL-schema identified by *SN*.

Access Rules

No additional Access Rules.

General Rules

1) Insert after GR3)x) 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.

9.22 <drop routine statement>

Function

Destroy an SQL-invoked routine.

Format

No additional Format items.

Syntax Rules

1) Insert after SR4)b)iv) The module descriptor of any SQL-server module.

2) Insert after SR5)a)iv) The module descriptor of any SQL-server module.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

9.23 <drop user-defined cast statement>

Function

Destroy a user-defined cast.

Format

No additional Format items.

Syntax Rules

1) Insert after SR7)d) The module descriptor of any SQL-server module.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

10 Access control

10.1 <grant statement>

Function

Define privileges.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

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

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

NOTE 16 – The privileges necessary include the EXECUTE privilege on every subject routine of every <routine invocation> contained in those <SQL procedure statement>s.

10.2 <revoke statement>

Function

Destroy privileges.

Format

No additional Format items.

Syntax Rules

1) Insert after SR20)d) EXECUTE privilege on every SQL-server module that includes one or more SQL-invoked routines that are among the subject routines of a <routine invocation> that is generally contained in the <query expression> of *V*.

2) Insert after SR22)c) EXECUTE privilege on every SQL-server modules that includes one or more SQL-invoked routines that are among the subject routines of a <routine invocation> that is generally contained in any <search condition> of *TC*.

3) Insert after SR23)c) EXECUTE privilege on every SQL-server module that includes one or more SQL-invoked routines that are among the subject routines of a <routine invocation> that is generally contained in any <search condition> of *AX*.

4) Insert after SR25)c) EXECUTE privilege on every SQL-server module that includes one or more SQL-invoked routines that are among the subject routines of a <routine invocation> that is generally contained in any <search condition> of *DC*.

5) Insert after SR35)a) EXECUTE privilege on every SQL-server module that includes one or more SQL-invoked routines that are among the subject routines of a <routine invocation> that is contained in the <routine body> of *RD*.

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

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

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

   c) SELECT privileges on every <table reference> contained in a <query expression> simply contained in a <cursor specification> or an <insert statement> contained in the <routine body> of any SQL-invoked routine included in *SSM*.

   d) SELECT privileges on every <table reference> contained in a <table expression> or <select list> immediately contained in a <select statement: single row> contained in the <routine body> of any SQL-invoked routine included in *SSM*.

e) SELECT privileges on every <table reference> and <column reference> contained in a <search condition> contained in a <delete statement: positioned> or an <update statement: searched> contained in the <routine body> of any SQL-invoked routine included in *SSM*.

f) SELECT privileges on every <table reference> and <column reference> contained in a <value expression> simply contained in a <row value constructor> immediately contained in a <set clause> contained in the <SQL routine body> of any SQL-invoked routine included in *SSM*.

g) INSERT privileges on every column

Case:

i) Named in the <insert column list> of an <insert statement> contained in the <routine body> of any SQL-invoked routine 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 included in *SSM*.

h) UPDATE privileges on every column whose name is contained in an <object column> contained in either an <update statement: positioned> or an <update statement: searched> contained in the <SQL routine body> of any SQL-invoked routine included in *SSM*.

i) DELETE privileges 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 included in *SSM*.

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

k) 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 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*.

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

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

n) SELECT privilege WITH HIERARCHY OPTION on at least one supertable of the scoped table of any <reference resolution> that is contained in any <search condition> contained in a <delete statement: searched> or an <update statement: searched> contained in the <SQL routine body> of *RD*.

o) SELECT privilege WITH HIERARCHY OPTION on at least one supertable of the scoped table of any <reference resolution> that is contained in any <value expression> simply contained in a <row value expression> immediately contained in a <set clause> contained in the <SQL routine body> of *RD*.

Access Rules

No additional Access Rules.

General Rules

1) Insert after GR17) 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
```

11 SQL-client modules

11.1 Calls to an <externally-invoked procedure>

Function

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

Syntax Rules

1) Insert into SR2)e)

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

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

11.2 <SQL procedure statement>

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
       | !! All alternatives from ISO/IEC 9075-5
       | <SQL-server module definition>

<SQL schema manipulation statement> ::=
         !! All alternatives from ISO/IEC 9075-2
       | !! All alternatives from ISO/IEC 9075-5
       | <drop module statement>

<SQL control statement> ::=
         !! All alternatives from ISO/IEC 9075-2
       | !! All alternatives from ISO/IEC 9075-5
       | <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
       | !! All alternatives from ISO/IEC 9075-5
       | <signal statement>
       | <resignal statement>
```

Syntax Rules

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

2) Insert after SR 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, or SYSTEM_USER.

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, or SYSTEM_USER.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

1) Without Feature P01, "Stored modules", an <SQL procedure statement> shall not be an <SQL server module definition> or a <drop module statement>.

12 Data manipulation

12.1 <open statement>

Function

Open a cursor.

Format

No additional Format items.

Syntax Rules

1) Replace SR1) Let *CN* be the <cursor name> in the <open statement>. *CN* shall be contained within the scope of one or more <cursor name>s that are equivalent to *CN*. If there is more than one such <cursor name>, then the one with the innermost scope is specified. Let *CR* be the cursor specified by *CN*.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

12.2 <fetch statement>

Function

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

Format

No additional Format items.

Syntax Rules

1) Replace SR3) Let *CN* be the <cursor name> in the <fetch statement>. *CN* shall be contained within the scope of one or more <cursor name>s that are equivalent to *CN*. If there is more than one such <cursor name>, then the one with the innermost scope is specified. Let *CR* be the cursor specified by *CN*. Let *T* be the table defined by the <cursor specification> of *CR*. Let *DC* be the <declare cursor> denoted by *CN*.

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

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

Access Rules

No additional Access Rules.

General Rules

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

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

12.3 <close statement>

Function

Close a cursor.

Format

No additional Format items.

Syntax Rules

1) Replace SR1) Let *CN* be the <cursor name> in the <close statement>. *CN* shall be contained within the scope of one or more <cursor name>s that are equivalent to *CN*. If there is more than one such <cursor name>, then the one with the innermost scope is specified. Let *CR* be the cursor specified by *CN*.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

12.4 <select statement: single row>

Function

Retrieve values from a specified row of a table.

Format

No additional Format items.

Syntax Rules

1) Insert after SR4) For each <target specification> *TS* that is an <SQL variable name>, the Syntax Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2, apply to *TS* and the corresponding element of the <select list>, as *TARGET* and *VALUE*, respectively.

Access Rules

No additional Access Rules.

General Rules

1) Insert after GR5) For each <target specification> *TS* that is an <SQL variable name>, the corresponding value in the row of *Q* is assigned to *TS* according to the General Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2, as *VALUE* and *TARGET*, respectively. The assignment of values to targets in the <select target list> is in an implementation-dependent order.

12.5 <delete statement: positioned>

Function

Delete a row of a table.

Format

No additional Format items.

Syntax Rules

1) Replace SR1) Let *CN* be the <cursor name> in the <delete statement: positioned>. *CN* shall be contained within the scope of one or more <cursor name>s that are equivalent to *CN*. If there is more than one such <cursor name>, then the one with the innermost scope is specified. Let *CR* be the cursor specified by *CN*.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

12.6 <update statement: positioned>

Function

Update a row of a table.

Format

No additional Format items.

Syntax Rules

1) Replace SR1) Let *CN* be the <cursor name> in the <update statement: positioned>. *CN* shall be contained within the scope of one or more <cursor name>s that are equivalent to *CN*. If there is more than one such <cursor name>, then the one with the innermost scope is specified. Let *CR* be the cursor specified by *CN*.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

12.7 <temporary table declaration>

Function

Replace 1st paragraph Declare a declared local temporary table that will be effectively materialized the first time that any <externally-invoked procedure> in the <SQL-client module definition> that contains, without an intervening <SQL-server module definition>, the <temporary table declaration> is executed or <SQL-invoked routine> in the <SQL-server module definition> that contains the <temporary table declaration> is executed. The scope of the declared local temporary table is all the <externally-invoked procedure>s of that <SQL-client module definition> or <SQL-invoked routine>s of that <SQL-server module definition> executed within the same SQL-session.

Format

No additional Format items.

Syntax Rules

1) Replace SR2) Case:

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

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

Access Rules

No additional Access Rules.

General Rules

1) Replace GR1) Case:

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

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

2) Replace GR3) 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*.

13 Control statements

13.1 <compound statement>

Function

Specify a statement that groups other statements together.

Format

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

<beginning label> ::= <statement label>

<ending label> ::= <statement label>

<statement label> ::= <identifier>

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

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

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

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

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

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

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

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

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

Syntax Rules

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

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

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

4) The scope of the <beginning label> is *CS* excluding every <SQL schema statement> contained in *CS* and every <local handler declaration list> contained in *CS*. <beginning label> shall not be equivalent to any other <beginning label>s contained in *CS* excluding every <SQL schema statement> that is contained in *CS* without an intervening <SQL schema statement> or <handler declaration>.

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 *CN* 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 *CN*.

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

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

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

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

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

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

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

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

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

Access Rules

None.

General Rules

1) If *CS* specifies ATOMIC, then an *atomic execution context* is active during the execution of *CS*.

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

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

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

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

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

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

         ```
         RESIGNAL
         ```

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

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

            ```
            CLOSE CR
            ```

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

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

   ```
   CLOSE CR
   ```

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

6) If *CS* specifies ATOMIC, then all savepoints established during the execution of *CS* are destroyed.

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

13.2 <handler declaration>

Function

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

Format

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

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

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

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

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

Syntax Rules

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

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

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

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

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

6) SQLEXCEPTION, SQLWARNING, and NOT FOUND correspond to SQLSTATE class values corresponding to categories X, W, and N, respectively, in Subclause 22.1, "SQLSTATE", in ISO/IEC 9075-2.

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

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

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

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

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

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

Access Rules

None.

General Rules

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

2) Let *CS* be the <compound statement> simply containing *HD*. Let *CC* be the <compound statement> from which *H* was activated.

3) When *H* is activated,

   Case:

   a) If *HD* specifies CONTINUE, then:

      i) *HA* is executed.

      ii) If there is an unhandled condition other than *successful completion* at the completion of *HA*, then the following <resignal statement> is effectively executed:

```
RESIGNAL
```

      Otherwise, *HA* completes with completion condition *successful completion* and control is returned to the SQL-statement following the executing SQL-statement that raised the condition in *CC*. If the SQL-statement that raised the condition is an <SQL control

statement>, then "the following SQL-statements" do not include the <SQL procedure statement>s contained within the <SQL control statement>.

NOTE 17 – For example, if the executing statement that raised the condition is an <if statement>, control is returned to the SQL-statement *following* the <if statement>.

b) If *HD* specifies EXIT, then:

i) *HA* is executed.

ii) If there is an unhandled condition other than *successful completion* at the completion of *HA*, then the following <resignal statement> is effectively executed:

```
RESIGNAL
```

Otherwise, *HA* completes with completion condition *successful completion* and control is returned to the end of *CS*.

c) If *HD* specifies UNDO, then:

i) 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.

ii) *HA* is executed.

iii) If there is an unhandled condition other than *successful completion* at the completion of *HA*, then the following <resignal statement> is effectively executed:

```
RESIGNAL
```

Otherwise, *HA* completes with completion condition *successful completion* and control is returned to the end of *CS*.

13.3 <condition declaration>

Function

Declare a condition name and an optional corresponding SQLSTATE value.

Format

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

Syntax Rules

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

2) No other <condition declaration> with the same scope as *CD* shall contain the same <sqlstate value> as *CD*.

Access Rules

None.

General Rules

1) <condition name> is *considered to be defined* for the SQLSTATE value specified by <sqlstate value>.

13.4 <SQL variable declaration>

Function

Declare one or more variables.

Format

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

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

Syntax Rules

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

Access Rules

None.

General Rules

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

   SET *SV* = *DV*

13.5 <assignment statement>

Function

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

Format

```
<assignment statement> ::=
       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) A <column reference> immediately contained in a <modified field target> or a <mutated target specification> shall be a new transition variable column reference.

   NOTE 18 – "New transition variable column reference" is defined in Subclause 6.5, "<identifier chain>", in ISO/IEC 9075-2.

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

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

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

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

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

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

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

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

5) 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>.

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

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

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

Access Rules

None.

General Rules

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

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

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

4) If <assignment target> is a <modified field reference> *FR*, then let *T* be the <target specification> simply contained in *FR*. Let F_i be a field identified by each <field name> simply contained in *FR*. Let *FT* be the field identified by the <field name> that is simply contained in <assignment target> and that is not simply contained in <modified field target>.

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

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

13.6 <case statement>

Function

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

Format

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

<simple case statement> ::=
      CASE <simple case operand 1>
        <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 <simple case operand 2>
        THEN <SQL statement list>

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

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



<simple case operand 1> ::= <value expression>

<simple case operand 2> ::= <value expression>
```

Syntax Rules

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

   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) The declared type of each <simple case operand 2> *SCO2* shall be comparable with the declared type of *SCO1*.

   c) The <simple case statement> is equivalent to a <searched case statement> in which each <searched case statement when clause> specifies a <search condition> of the form:

      SCO1 = *SCO2*

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 to *N*:

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

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

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

13.7 <if statement>

Function

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

Format

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

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

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

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

Syntax Rules

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

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

is equivalent to

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

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

Access Rules

None.

General Rules

1) Case:

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

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

      NOTE 20 – "Otherwise" means that the <search condition> immediately contained in the <if statement> evaluates to *false* or to *unknown*.

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

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

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

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

13.8 <iterate statement>

Function

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

Format

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

Syntax Rules

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

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

Access Rules

None.

General Rules

1) The execution of *SSL* is terminated.

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

13.9 <leave statement>

Function

Continue execution by leaving a labeled statement.

Format

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

Syntax Rules

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

Access Rules

None.

General Rules

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

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

      CLOSE *CR*

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

2) The execution of *S* is terminated.

13.10 <loop statement>

Function

Repeat the execution of a statement.

Format

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

Syntax Rules

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

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

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

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

Access Rules

None.

General Rules

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

   ```
   BEGIN NOT ATOMIC SSL END
   ```

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

   NOTE 22 – The occurrence of an exception condition or the execution of a <leave statement> may also cause execution of *LS* to be terminated; see Subclause 6.2.3.1, "Exceptions", in ISO/IEC 9075-1, and Subclause 13.9, "<leave statement>", respectively. Some actions taken by a condition handler might also cause execution of *LS* to be terminated; see Subclause 13.2, "<handler declaration>".

13.11 <while statement>

Function

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

Format

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

Syntax Rules

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

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

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

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

Access Rules

None.

General Rules

1) The <search condition> is evaluated.

2) Case:

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

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

      ```
      BEGIN NOT ATOMIC SSL END
      ```

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

      NOTE 23 – The occurrence of an exception condition or the execution of a <leave statement> may also cause execution of *WS* to be terminated; see Subclause 6.2.3.1, "Exceptions", in ISO/IEC 9075-1, and Subclause 13.9, "<leave statement>", respectively. Some actions taken by a condition handler might also cause execution of *WS* to be terminated; see Subclause 13.2, "<handler declaration>".

13.12 <repeat statement>

Function

Repeat the execution of a statement.

Format

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

Syntax Rules

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

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

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

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

Access Rules

None.

General Rules

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

   BEGIN NOT ATOMIC *SSL* END

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

   NOTE 24 – The occurrence of an exception condition or the execution of a <leave statement> may also cause execution of *RS* to be terminated; see Subclause 6.2.3.1, "Exceptions", in ISO/IEC 9075-1, and Subclause 13.9, "<leave statement>", respectively. Some actions taken by a condition handler might also cause execution of *RS* to be terminated; see Subclause 13.2, "<handler declaration>".

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

13.13 <for statement>

Function

Execute a statement for each row of a table.

Format

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

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

Syntax Rules

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

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

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

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

5) Let *QE* be the <query expression> of *FCS*. Each column of the table specified by *QE* shall have a <column name> that is not equivalent to every 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>.

The <for statement> is equivalent to:

```
BL: BEGIN NOT ATOMIC

  FLVN: BEGIN NOT ATOMIC
    DECLARE CN CS CURSOR FOR
        SELECT ROW ( Q.V1, Q.V2, ... , Q.Vn )
        FROM ( FCS ) AS Q
    DECLARE FLVN ROW ( V1 DT1, V2 DT2, ..., Vn DTn );
    DECLARE AT_END BOOLEAN DEFAULT FALSE;
    DECLARE NOT_FOUND CONDITION FOR SQLSTATE '02000';

    BEGIN NOT ATOMIC
      DECLARE CONTINUE HANDLER FOR NOT_FOUND
          SET AT_END = TRUE;

      OPEN CN;
      FETCH CN INTO FLVN;
      WHILE NOT AT_END DO
        SLL;
        BEGIN NOT ATOMIC
          FETCH CN INTO FLVN;
        END;
      END WHILE;
      CLOSE CN;
    END;
  END FLVN;
END BL
```

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

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

9) *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.

14 Dynamic SQL

14.1 <prepare statement>

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) Insert after GR6)a)xxvii) If *DP* is the <assignment target> simply contained in an <assignment statement> *AS*, 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 *AS*.

2) Insert after GR6)a)xxvii) If *DP* is the <value expression> simply contained in an <assignment source> in an <assignment statement> *AS* 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 *AS*.

   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*.

3) Insert after GR6)a)xxvii) If *DP* is a <value expression> simply contained in a <simple case operand 1> or a <simple case operand 2> of a <simple case statement> *CS*, or if *DP* represents the value of a subfield *SF* of such a <value expression>, then let *RT* be the result of applying the Syntax Rules of Subclause 9.3, "Data types of results of aggregations", in ISO/IEC 9075-5 to the <value

expression>s simply contained in the <simple case operand 1> and all <simple case operand 2>s simply contained in *CS*.

Case:

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

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

15 Embedded SQL

15.1 <embedded SQL host program>

Function

Specify an <embedded SQL host program>.

Format

No additional Format items.

Syntax Rules

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

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

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

4) Replace SR15) An <embedded exception declaration> that is contained in an <embedded SQL host program> shall precede in the text of that <embedded SQL host program> any SQL-statement or <handler declaration> that references the <exception name> contained in the <embedded exception declaration>.

5) Insert before SR22)j) *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*.

6) Insert before SR22)k) *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*.

7) Replace SR23)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 16.2, "<embedded exception declaration>".

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

16 Diagnostics management

16.1 <get diagnostics statement>

Function

Get exception or completion condition information from the diagnostics area.

Format

```
<condition information item name> ::=
        !! All alternatives from ISO/IEC 9075-2
      | !! All alternatives from ISO/IEC 9075-5
      | CONDITION_IDENTIFIER
```

Syntax Rules

Table 2—<identifier>s for use with <get diagnostics statement>

<identifier>	Data Type
<statement information item name>s	
All alternatives from ISO/IEC 9075-2 *All alternatives from ISO/IEC 9075-5*	
<condition information item name>s	
All alternatives from ISO/IEC 9075-2 *All alternatives from ISO/IEC 9075-5* CONDITION_IDENTIFIER	character varying (*L*)

Access Rules

No additional Access Rules.

General Rules

Table 3—SQL-statement codes for use in the diagnostics area

SQL-statement	Identifier	Code
All alternatives from ISO/IEC 9075-2		
All alternatives from ISO/IEC 9075-5		
<alter module statement>	ALTER MODULE	95
<assignment statement>	ASSIGNMENT	5
<case statement>	CASE	86
<compound statement>	BEGIN END	12
<drop module statement>	DROP MODULE	28
<for statement>	FOR	46
<handler declaration>	HANDLER	87
<if statement>	IF	88
<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
<SQL variable declaration>	DECLARE VARIABLE	96
<temporary table declaration>	TEMPORARY TABLE	93
<while statement>	WHILE	97

1) Insert before GR3)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.

16.2 <signal statement>

Function

Signal an exception condition.

Format

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

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

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

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

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

Syntax Rules

1) Case:

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

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

      ii) *CN* shall be contained within the scope of one or more <condition name>s whose associated <condition declaration> includes a condition whose <identifier> is *CN*. If there is more than one such <condition name>, then the one with the innermost scope is specified. Let *C* be that condition.

   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 not specify CONDITION_NUMBER, RETURNED_SQLSTATE, MESSAGE_LENGTH, or MESSAGE_OCTET_LENGTH. No other 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 2, "<identifier>s for use with <get diagnostics statement>".

Access Rules

None.

General Rules

1) Let *N* be the value of the statement information field NUMBER in the diagnostics area before the execution of the <signal statement>. The existing exception information areas 1 through *N* in the diagnostics area are cleared. The value of the statement information field NUMBER in the diagnostics area is set to 1 and the MORE field is set to 'N'.

   The statement information field COMMAND_FUNCTION is set to 'SIGNAL' and the DYNAMIC_FUNCTION field is set to a zero-length string. In the first exception information area in the diagnostics area, the field CONDITION_IDENTIFIER is set to contain *CN*. If *C* has an associated SQLSTATE value, then the exception information field RETURNED_SQLSTATE is set to that value.

2) The information fields CLASS_ORIGIN, SUBCLASS_ORIGIN, CONSTRAINT_CATALOG, CONSTRAINT_SCHEMA, CONSTRAINT_NAME, CATALOG_NAME, SCHEMA_NAME, TABLE_NAME, COLUMN_NAME, CURSOR_NAME, and MESSAGE_TEXT in the diagnostics area are set to a zero-length string. The information fields MESSAGE_LENGTH and MESSAGE_OCTET_LENGTH are set to 0 (zero).

3) 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
```

16.3 <resignal statement>

Function

Resignal an exception condition.

Format

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

Syntax Rules

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

2) If <signal value> is specified, then

   Case:

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

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

      ii) *CN* shall be contained within the scope of one or more <condition name>s whose associated <condition declaration> includes a condition whose <identifier> is *C*N. If there is more than one such <condition name>, then the one with the innermost scope is specified. Let *C* be that condition.

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

   a) In the first condition area in the diagnostics area, the information field identified by the <signal information name> is set to contain the value of the <simple value specification>.

   b) If the <signal information name> specifies MESSAGE_TEXT, then the information fields MESSAGE_LENGTH and MESSAGE_OCTET_LENGTH in the diagnostics area are set to contain the length and the length in octets of the value of the <simple value specification>, respectively.

2) Case:

a) If the first condition in the diagnostics area has no RETURN_SQLSTATE value and the value of the CONDITION_IDENTIFIER is a zero-length string, then an exception condition is raised: *resignal when handler not active*.

b) Otherwise, let *N* be the value of the statement information field NUMBER in the diagnostics area before the execution of *RS*.

Case:

i) If <signal value> is not specified, then the diagnostics area remains unchanged.

ii) If <signal value> is specified, then the statement information field NUMBER in the diagnostics area is incremented. All existing condition areas are stacked such that the *i*-th condition area is placed at the position of the *i*+1-st condition area in the diagnostics area. If the maximum number of condition areas for the diagnostics area is exceeded, then the value of the statement information field NUMBER contains the number of exception or completion conditions of the SQL-statement that raised the condition plus those raised by *RS*, and the value of the statement information field MORE is 'Y'.

In the first condition area in the diagnostics area, the statement information field COMMAND_FUNCTION is set to 'RESIGNAL', the DYNAMIC_FUNCTION field is set to a zero-length string, and CONDITION_IDENTIFIER is set to contain *CN*. If *C* has an associated SQLSTATE value, then the condition information field RETURNED_SQLSTATE is set to that value.

3) Case:

a) If the first condition in the diagnostics area has a RETURNED_SQLSTATE value, then:

i) Let *S* be that value.

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

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

NOTE 25 – 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 field of the first condition in the diagnostics area.

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

iii) If no handler is activated, 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*.

17 Information Schema

17.1 MODULE_COLUMN_USAGE view

Function

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

Definition

```
CREATE VIEW MODULE_COLUMN_USAGE AS
    SELECT ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME,
           TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME
    FROM DEFINITION_SCHEMA.MODULE_COLUMN_USAGE
      JOIN
         DEFINITION_SCHEMA.SCHEMATA S
      ON ( ( TABLE_CATALOG, TABLE_SCHEMA ) =
           ( S.CATALOG_NAME, S.SCHEMA_NAME ) )

    WHERE ( SCHEMA_OWNER = CURRENT_USER
          OR
            SCHEMA_OWNER IN
            ( SELECT ROLE_NAME
              FROM ENABLED_ROLES ) )
       AND
          MODULE_CATALOG =
          ( SELECT CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME );

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

17.2 MODULE_PRIVILEGES view

Function

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

Definition

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

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

17.3 MODULE_TABLE_USAGE view

Function

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

Definition

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

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

17.4 MODULES view

Function

Identify the SQL-server modules in this catalog that are accessible to a given user.

Definition

```
CREATE VIEW MODULES AS
    SELECT MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME,
           DEFAULT_CHARACTER_SET_CATALOG, DEFAULT_CHARACTER_SET_SCHEMA,
           DEFAULT_CHARACTER_SET_NAME,
           DEFAULT_SCHEMA_CATALOG, DEFAULT_SCHEMA_NAME,

           CASE
             WHEN EXISTS (
               SELECT *
               FROM DEFINITION_SCHEMA.SCHEMATA AS S
               WHERE ( MODULE_CATALOG, MODULE_SCHEMA ) =
                     ( S.CATALOG_NAME, S.SCHEMA_NAME )
                 AND
                     ( SCHEMA_OWNER IN
                       ( 'PUBLIC', CURRENT_USER )
                     OR
                     ( SCHEMA_OWNER IN
                       ( SELECT ROLE_NAME
                         FROM ENABLED_ROLES ) ) )
             THEN MODULE_DEFINITION
             ELSE NULL
           END AS MODULE_DEFINITION,
           MODULE_AUTHORIZATION, SQL_PATH, MODULE_CREATED, MODULE_LAST_ALTERED
    FROM DEFINITION_SCHEMA.MODULES
    WHERE ( MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME ) IN
            ( SELECT MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME
              FROM DEFINITION_SCHEMA.MODULE_PRIVILEGES
              WHERE ( SCHEMA_OWNER IN
                      ( 'PUBLIC', CURRENT_USER )
                    OR
                      SCHEMA_OWNER IN
                    ( SELECT ROLE_NAME
                      FROM ENABLED_ROLES ) ) )
      AND
          MODULE_CATALOG =
          ( SELECT CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME );

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

17.5 ROLE_MODULE_GRANTS view

Function

Identify the privileges on SQL-server modules defined in this catalog that are available to or granted by the currently enabled roles.

Definition

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

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

Conformance Rules

1) Without Feature T322, "Extended Roles", conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_MODULE_GRANTS.

17.6 Short name views

Function

Provide alternative views that use only identifiers that do not require Feature F391, "Long identifiers".

Definition

```
CREATE VIEW MODULE_COL_USAGE
          ( ROUTINE_CATALOG,    ROUTINE_SCHEMA,    ROUTINE_NAME,
            TABLE_CATALOG,      TABLE_SCHEMA,      TABLE_NAME,
            COLUMN_NAME) AS
    SELECT ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME,
           TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
           COLUMN_NAME
    FROM INFORMATION_SCHEMA.MODULE_COLUMN_USAGE;

GRANT SELECT ON TABLE MODULE_COL_USAGE
    TO PUBLIC WITH GRANT OPTION;

CREATE VIEW MODULES_S
          ( MODULE_CATALOG,      MODULE_SCHEMA,     MODULE_NAME,
            DEF_CHAR_SET_CAT,    DEF_CHAR_SET_SCH,  DEF_CHAR_SET_NAME,
            DEF_SCHEMA_CATALOG,  DEFAULT_SCHEMA,    MODULE_DEFINITION,
            MODULE_AUTH,         SQL_PATH,          CREATED,
            ALTERED) AS
    SELECT MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME,
           DEFAULT_CHARACTER_SET_CATALOG, DEFAULT_CHARACTER_SET_SCHEMA,
           DEFAULT_CHARACTER_SET_NAME,
           DEFAULT_SCHEMA_CATALOG, DEFAULT_SCHEMA, MODULE_DEFINITION,
           MODULE_AUTHORIZATION, SQL_PATH, CREATED,
           LAST_ALTERED
    FROM INFORMATION_SCHEMA.MODULES;

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