

# **SC32 N00003**

**ISO Final Committee Draft (FCD)**

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

**«Part 4»**

**October 1997**

<b>Contents</b>	<b>Page</b>
Foreword .....	ix
Introduction .....	xi
<b>1 Scope</b> .....	<b>1</b>
<b>2 Normative references</b> .....	<b>3</b>
<b>3 Definitions, notations, and conventions</b> .....	<b>5</b>
3.1 Definitions .....	5
3.1.1 Definitions provided in Part 4 .....	5
3.2 Conventions .....	5
3.2.1 Use of terms .....	5
3.2.1.1 Exceptions .....	5
3.2.1.2 Rule evaluation order .....	6
3.2.1.3 Other terms .....	6
3.2.2 Relationships to other parts of ISO/IEC 9075 .....	6
3.2.2.1 New and modified Clauses, Subclauses, and Annexes .....	6
3.2.2.2 Clause, Subclause, and Table relationships .....	6
3.3 Object identifier for Database Language SQL .....	12
<b>4 Concepts</b> .....	<b>15</b>
4.1 SQL-server Modules .....	15
4.2 SQL-invoked routines .....	16
4.3 Tables .....	16
4.4 SQL-schemas .....	16
4.5 Host parameters .....	16
4.5.1 Status parameters .....	17
4.6 Diagnostics area .....	17
4.7 Cursors .....	17
4.8 Condition handling .....	17
4.9 SQL-statements .....	19
4.9.1 SQL-statements classified by function .....	19
4.9.2 Embeddable SQL-statements .....	20
4.9.3 Directly executable SQL-statements .....	20
4.9.4 Iterated SQL-statement> .....	20
4.9.5 SQL-statements and transaction states .....	20
4.9.6 Compound statements .....	21
4.9.7 SQL-statement atomicity .....	21
4.10 Privileges and roles .....	21

<b>5</b>	<b>Lexical elements</b> .....	23
5.1	<token> and <separator> .....	23
5.2	Names and identifiers .....	25
<b>6</b>	<b>Scalar expressions</b> .....	27
6.1	<value specification> and <target specification> .....	27
6.2	<item reference> .....	28
6.3	<SQL variable reference> .....	30
6.4	<datetime value function> .....	31
<b>7</b>	<b>Query expressions</b> .....	33
7.1	<query specification> .....	33
<b>8</b>	<b>Additional common elements</b> .....	35
8.1	<routine invocation> .....	35
8.2	<privileges> .....	37
8.3	<sqlstate value> .....	38
<b>9</b>	<b>Schema definition and manipulation</b> .....	39
9.1	<schema definition> .....	39
9.2	<drop schema statement> .....	40
9.3	<default clause> .....	41
9.4	<drop column definition> .....	43
9.5	<drop table constraint definition> .....	44
9.6	<drop table statement> .....	45
9.7	<view definition> .....	46
9.8	<drop view statement> .....	47
9.9	<drop domain statement> .....	48
9.10	<drop character set statement> .....	49
9.11	<drop collation statement> .....	50
9.12	<drop translation statement> .....	51
9.13	<assertion definition> .....	52
9.14	<drop assertion statement> .....	53
9.15	<SQL-server module definition> .....	54
9.16	<drop module statement> .....	57
9.17	<SQL-invoked routine> .....	58
9.18	<drop routine statement> .....	60
9.19	<grant statement> .....	61
9.20	<revoke statement> .....	62
<b>10</b>	<b>SQL-client modules</b> .....	67
10.1	Calls to an <externally-invoked procedure> .....	68
10.2	<SQL procedure statement> .....	69

<b>11</b>	<b>Data manipulation</b>	71
11.1	<open statement>	71
11.2	<fetch statement>	72
11.3	<close statement>	73
11.4	<select statement: single row>	74
11.5	<delete statement: positioned>	75
11.6	<delete statement: searched>	76
11.7	<update statement: positioned>	77
11.8	<update statement: searched>	78
11.9	<temporary table declaration>	79
<b>12</b>	<b>Control statements</b>	81
12.1	<compound statement>	81
12.2	<handler declaration>	85
12.3	<condition declaration>	88
12.4	<SQL variable declaration>	89
12.5	<assignment statement>	90
12.6	<case statement>	92
12.7	<if statement>	94
12.8	<iterate statement>	96
12.9	<leave statement>	97
12.10	<loop statement>	98
12.11	<while statement>	99
12.12	<repeat statement>	100
12.13	<for statement>	101
<b>13</b>	<b>Diagnostics management</b>	105
13.1	<get diagnostics statement>	105
13.2	<signal statement>	107
13.3	<resignal statement>	108
<b>14</b>	<b>Information Schema and Definition Schema</b>	111
14.1	Information Schema	111
14.1.1	MODULES view	111
14.1.2	MODULE_TABLE_USAGE view	112
14.1.3	MODULE_COLUMN_USAGE view	113
14.1.4	MODULE_PRIVILEGES view	114
14.2	Definition Schema	115
14.2.1	MODULES base table	115
14.2.2	MODULE_TABLE_USAGE base table	117
14.2.3	MODULE_COLUMN_USAGE base table	118
14.2.4	MODULE_PRIVILEGES base table	119
<b>15</b>	<b>Status codes</b>	121
15.1	SQLSTATE	121

<b>16</b>	<b>Conformance</b>	123
16.1	Claims of conformance	123
16.2	Extensions and options	123
16.2.1	Information Schema requirements	123
16.2.2	Schema manipulation requirements	124
16.3	Flagger requirements	124
<b>Annex A</b>	<b>Implementation-defined elements</b>	125
<b>Annex B</b>	<b>Implementation-dependent elements</b>	127
<b>Annex C</b>	<b>Deprecated features</b>	129
<b>Annex D</b>	<b>Incompatibilities with ISO/IEC 9075:1992</b>	131
<b>Annex E</b>	<b>SQL Feature Taxonomy</b>	133
<b>Index</b>		Index1

**TABLES**

<b>Table</b>		<b>Page</b>
1	Clause, Subclause, and Table relationships . . . . .	6
2	<identifier>s for use with <get diagnostics statement> . . . . .	105
3	SQL-statement codes for use in the diagnostics area . . . . .	106
4	SQLSTATE class and subclass values . . . . .	121
5	SQL/PSM feature taxonomy . . . . .	133

## 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 SC21, *Open systems interconnection, data management, and open distributed processing*.

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)
- Part 6: XA Specialization (SQL/Transaction)
- Part 7: Temporal (SQL/Temporal)

Annexes A, B, C, and D of ISO/IEC 9075-4 are for information only.



## Introduction

The organization of ISO/IEC 9075-4 is as follows:

- 1) Clause 1, “Scope”, specifies the scope of ISO/IEC 9075-4.
- 2) Clause 2, “Normative references”, identifies additional standards that, through reference in this part of ISO/IEC 9075, constitute provisions of ISO/IEC 9075-4.
- 3) Clause 3, “Definitions, notations, and conventions”, defines the notations and conventions used in ISO/IEC 9075-4.
- 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, “SQL-client modules”, defines the facilities for using persistent stored modules.
- 11) Clause 11, “Data manipulation”, defines data manipulation operations associated with persistent stored modules.
- 12) Clause 12, “Control statements”, defines the control statements used with persistent stored modules.
- 13) Clause 13, “Diagnostics management”, defines enhancements to the facilities used with persistent stored modules.
- 14) Clause 14, “Information Schema and Definition Schema”, defines the Information and Definition Schema objects associated with persistent stored modules.
- 15) Clause 15, “Status codes”, defines SQLSTATE values related to persistent stored modules.
- 16) Clause 16, “Conformance”, defines the criteria for conformance to ISO/IEC 9075.
- 17) Annex A, “Implementation-defined elements”, is an informative Annex. It lists those features for which the body of this part of the standard states that the syntax or meaning or effect on the database is partly or wholly implementation-defined, and describes the defining information that an implementer shall provide in each case.

- 18) Annex B, “Implementation-dependent elements”, is an informative Annex. It lists those features for which the body of this part of the standard states that the syntax or meaning or effect on the database is partly or wholly implementation-dependent.
- 19) Annex C, “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.
- 20) Annex D, “Incompatibilities with ISO/IEC 9075:1992”, is an informative Annex. It lists the incompatibilities between this edition of ISO/IEC 9075-4 and ISO/IEC 9075:1996.
- 21) Annex E, “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 ISO/IEC 9075-4, Clauses begin a new odd-numbered page, and in Clause 5, “Lexical elements”, through Clause 16, “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

International Standard ISO/IEC 9075-4 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 <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 ability to set an SQL-path for controlling the determination of the subject routine to be invoked.
- 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 ISO/IEC 9075-4 is described by the Reference Model of Data Management (ISO/IEC 10032:1993).

**ISO/IEC JTC1/SC21 N11138 = DBL:CWB-003**

## 2 Normative references

The following standards contain provisions that, through reference in this text, constitute provisions of ISO/IEC 9075-4. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on ISO/IEC 9075-4 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 9075-1:199x, *Information Technology — Database languages — SQL — Part 1: Framework (SQL/Framework)*.

ISO/IEC 9075-2:199x, *Information Technology — Database Languages — SQL — Part 2: Foundation (SQL/Foundation)*.

ISO/IEC 9075-3:199x, *Information Technology — Database Languages — SQL — Part 3: Call-level interface (SQL/CLI)*.

ISO/IEC 9075-5:199x, *Information Technology — Database Languages — SQL — Part 5: Host language bindings (SQL/Bindings)*.

ISO/IEC 9075-5:199x, *Information Technology — Database Languages — SQL — Part 6: XA Specialization (SQL/Transaction)*.

ISO/IEC 9075-5:199x, *Information Technology — Database Languages — SQL — Part 7: Temporal (SQL/Temporal)*.

**ISO/IEC JTC1/SC21 N11138 = DBL:CWB-003**

## 3 Definitions, notations, and conventions

### 3.1 Definitions

#### 3.1.1 Definitions provided in Part 4

Insert this paragraph For the purposes of ISO/IEC 9075-4, the definitions given in ISO/IEC 9075-1 and ISO/IEC 9075-2 and the following definitions apply.

- 1 definition deleted.
- a) **SQL routine:** An SQL-invoked routine whose routine body is written in SQL.
- 1 Subclause deleted.

### 3.2 Conventions

Insert this paragraph Except as otherwise specified in ISO/IEC 9075-4, the conventions used in ISO/IEC 9075-4 are identical to those described in ISO/IEC 9075-1 and ISO/IEC 9075-2.

#### 3.2.1 Use of terms

##### 3.2.1.1 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 10.2, “<SQL procedure statement>”, Subclause 15.1, “<direct SQL statement>”, in ISO/IEC 9075-5, Subclause 12.1, “<compound statement>”, and Subclause 12.2, “<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.

- 1 paragraph deleted.

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*.

- 1 Subclause deleted.

**3.2 Conventions**

**3.2.1.2 Rule evaluation order**

Insert this paragraph An invocation of an SQL-invoked function is *inessential* if the SQL-function is deterministic and does not possibly modify SQL-data; otherwise, it is implementation-defined whether or not it is inessential.

**3.2.1.3 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*.

**3.2.2 Relationships to other parts of ISO/IEC 9075**

- 1 paragraph deleted.

**3.2.2.1 New and modified Clauses, Subclauses, and Annexes**

Clauses, Subclauses, and Annexes (other than Clause 1, “Scope” and Clause 2, “Normative references”) in ISO/IEC 9075-4 that have names identical to Clauses, Subclauses, or Annexes in ISO/IEC 9075-2 and/or ISO/IEC 9075-5 supplement the Clause, Subclause, or Annex, respectively, in ISO/IEC 9075-2 and/or ISO/IEC 9075-5, typically by replacing paragraphs, Format items, or Rules or by providing new paragraphs, Format items, or Rules. However, Clauses, Subclauses, and Annexes in ISO/IEC 9075-4 that have names identical to Clauses, Subclauses, and Annexes in ISO/IEC 9075-2 and/or ISO/IEC 9075-5 do not necessarily have the same *number* or *letter* as the corresponding Clause, Subclause, or Annex in ISO/IEC 9075-2 and/or ISO/IEC 9075-5. Any differences in Clause or Subclause number or in Annex letter is not significant. Table 1, “Clause, Subclause, and Table relationships”, identifies the relationships between Clauses, Subclauses, and Annexes in ISO/IEC 9075-4 and the corresponding Clauses, Subclauses, and Annexes in other parts of ISO/IEC 9075.

Clauses, Subclauses, and Annexes in ISOIEC 9075-4 that have names that are identical to Clauses, Subclauses, and Annexes in both ISO/IEC 9075-2 and ISO/IEC 9075-5 are to be understood as follows: First, perform the required modifications of the Clause, Subclause, or Annex in ISO/IEC 9075-5 on ISO/IEC 9075-2, then perform the required modifications from ISO/IEC 9075-4 on the result of those first modifications.

**3.2.2.2 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

**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 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"	<i>(none)</i>	<i>(none)</i>
Subclause 3.2, "Conventions"	Subclause 3.3, "Conventions"	ISO/IEC 9075-2
Subclause 3.2.1, "Use of terms"	Subclause 3.3.1, "Use of terms"	ISO/IEC 9075-2
Subclause 3.2.1.1, "Exceptions"	Subclause 6.2.3.1, "Exceptions"	ISO/IEC 9075-1
Subclause 3.2.1.2, "Rule evaluation order"	Subclause 6.2.3.4, "Rule evaluation order"	ISO/IEC 9075-1
Subclause 3.2.1.3, "Other terms"	Subclause 6.2.3.7, "Other terms"	ISO/IEC 9075-1
Subclause 3.2.2, "Relationships to other parts of ISO/IEC 9075"	<i>(none)</i>	<i>(none)</i>
Subclause 3.2.2.1, "New and modified Clauses, Subclauses, and Annexes"	<i>(none)</i>	<i>(none)</i>
Subclause 3.2.2.2, "Clause, Subclause, and Table relationships"	<i>(none)</i>	<i>(none)</i>
Subclause 3.3, "Object identifier for Database Language SQL"	Subclause 3.4, "Object identifier for Database Language SQL"	ISO/IEC 9075-2
Clause 4, "Concepts"	Clause 4, "Concepts"	ISO/IEC 9075-2
Subclause 4.1, "SQL-server Modules"	<i>(none)</i>	<i>(none)</i>
Subclause 4.2, "SQL-invoked routines"	<i>(none)</i>	<i>(none)</i>
Subclause 4.3, "Tables"	Subclause 4.18, "Tables"	ISO/IEC 9075-2
Subclause 4.4, "SQL-schemas"	Subclause 4.23, "SQL-schemas"	ISO/IEC 9075-2
Subclause 4.5, "Host parameters"	Subclause 4.29, "Host parameters"	ISO/IEC 9075-2
Subclause 4.5.1, "Status parameters"	Subclause 4.29.1, "Status parameters"	ISO/IEC 9075-2
Subclause 4.6, "Diagnostics area"	Subclause 4.30, "Diagnostics area"	ISO/IEC 9075-2
Subclause 4.7, "Cursors"	Subclause 4.32, "Cursors"	ISO/IEC 9075-2
Subclause 4.8, "Condition handling"	<i>(none)</i>	<i>(none)</i>
Subclause 4.9, "SQL-statements"	Subclause 4.33, "SQL-statements"	ISO/IEC 9075-2
Subclause 4.9.1, "SQL-statements classified by function"	Subclause 4.33.2, "SQL-statements classified by function"	ISO/IEC 9075-2

3.2 Conventions

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 4.9.2, "Embeddable SQL-statements"	Subclause 4.6.4, "Embeddable SQL-statements"	ISO/IEC 9075-5
Subclause 4.9.3, "Directly executable SQL-statements"	Subclause 4.6.6, "Directly executable SQL-statements"	ISO/IEC 9075-5
Subclause 4.9.4, "Iterated SQL-statement>"	<i>(none)</i>	<i>(none)</i>
Subclause 4.9.5, "SQL-statements and transaction states"	Subclause 4.33.3, "SQL-statements and transaction states"	ISO/IEC 9075-2
Subclause 4.9.6, "Compound statements"	<i>(none)</i>	<i>(none)</i>
Subclause 4.9.7, "SQL-statement atomicity"	<i>(none)</i>	<i>(none)</i>
Subclause 4.10, "Privileges and roles"	Subclause 4.34, "Privileges and roles"	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
Subclause 6.1, "<value specification> and <target specification>"	Subclause 6.2, "<value specification> and <target specification>"	ISO/IEC 9075-2
Subclause 6.2, "<item reference>"	Subclause 6.5, "<item reference>"	ISO/IEC 9075-2
Subclause 6.3, "<SQL variable reference>"	<i>(none)</i>	<i>(none)</i>
Subclause 6.4, "<datetime value function>"	Subclause 6.15, "<datetime value function>"	ISO/IEC 9075-2
Clause 7, "Query expressions"	Clause 7, "Query expressions"	ISO/IEC 9075-2
Subclause 7.1, "<query specification>"	Subclause 7.10, "<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>"	<i>(none)</i>	<i>(none)</i>
Clause 9, "Schema definition and manipulation"	Clause 11, "Schema definition and manipulation"	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 9.1, "<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 11.7, "<default clause>"	ISO/IEC 9075-2
Subclause 9.4, "<drop column definition>"	Subclause 11.17, "<drop column definition>"	ISO/IEC 9075-2
Subclause 9.5, "<drop table constraint definition>"	Subclause 11.19, "<drop table constraint definition>"	ISO/IEC 9075-2
Subclause 9.6, "<drop table statement>"	Subclause 11.20, "<drop table statement>"	ISO/IEC 9075-2
Subclause 9.7, "<view definition>"	Subclause 11.21, "<view definition>"	ISO/IEC 9075-2
Subclause 9.8, "<drop view statement>"	Subclause 11.22, "<drop view statement>"	ISO/IEC 9075-2
Subclause 9.9, "<drop domain statement>"	Subclause 11.29, "<drop domain statement>"	ISO/IEC 9075-2
Subclause 9.10, "<drop character set statement>"	Subclause 11.31, "<drop character set statement>"	ISO/IEC 9075-2
Subclause 9.11, "<drop collation statement>"	Subclause 11.33, "<drop collation statement>"	ISO/IEC 9075-2
Subclause 9.12, "<drop translation statement>"	Subclause 11.35, "<drop translation statement>"	ISO/IEC 9075-2
Subclause 9.13, "<assertion definition>"	Subclause 11.36, "<assertion definition>"	ISO/IEC 9075-2
Subclause 9.14, "<drop assertion statement>"	Subclause 11.37, "<drop assertion statement>"	ISO/IEC 9075-2
Subclause 9.15, "<SQL-server module definition>"	(none)	(none)
Subclause 9.16, "<drop module statement>"	(none)	(none)
Subclause 9.17, "<SQL-invoked routine>"	Subclause 11.42, "<SQL-invoked routine>"	ISO/IEC 9075-2
Subclause 9.18, "<drop routine statement>"	Subclause 11.43, "<drop routine statement>"	ISO/IEC 9075-2
Subclause 9.19, "<grant statement>"	Subclause 11.46, "<grant statement>"	ISO/IEC 9075-2
Subclause 9.20, "<revoke statement>"	Subclause 11.51, "<revoke statement>"	ISO/IEC 9075-2

## 3.2 Conventions

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 10, "SQL-client modules"	Subclause 12.1, "<SQL-client module definition>"	ISO/IEC 9075-2
Subclause 10.1, "Calls to an <externally-invoked procedure>"	Subclause 12.4, "Calls to an <externally-invoked procedure>"	ISO/IEC 9075-2
Subclause 10.2, "<SQL procedure statement>"	Subclause 12.6, "<SQL procedure statement>"	ISO/IEC 9075-2
Clause 11, "Data manipulation"	Clause 13, "Data manipulation"	ISO/IEC 9075-2
Subclause 11.1, "<open statement>"	Subclause 13.2, "<open statement>"	ISO/IEC 9075-2
Subclause 11.2, "<fetch statement>"	Subclause 13.3, "<fetch statement>"	ISO/IEC 9075-2
Subclause 11.3, "<close statement>"	Subclause 13.4, "<close statement>"	ISO/IEC 9075-2
Subclause 11.4, "<select statement: single row>"	Subclause 13.5, "<select statement: single row>"	ISO/IEC 9075-2
Subclause 11.5, "<delete statement: positioned>"	Subclause 13.6, "<delete statement: positioned>"	ISO/IEC 9075-2
Subclause 11.6, "<delete statement: searched>"	Subclause 13.7, "<delete statement: searched>"	ISO/IEC 9075-2
Subclause 11.7, "<update statement: positioned>"	Subclause 13.9, "<update statement: positioned>"	ISO/IEC 9075-2
Subclause 11.8, "<update statement: searched>"	Subclause 13.10, "<update statement: searched>"	ISO/IEC 9075-2
Subclause 11.9, "<temporary table declaration>"	Subclause 13.11, "<temporary table declaration>"	ISO/IEC 9075-2
Clause 12, "Control statements"	<i>(none)</i>	<i>(none)</i>
Subclause 12.1, "<compound statement>"	<i>(none)</i>	<i>(none)</i>
Subclause 12.2, "<handler declaration>"	<i>(none)</i>	<i>(none)</i>
Subclause 12.3, "<condition declaration>"	<i>(none)</i>	<i>(none)</i>
Subclause 12.4, "<SQL variable declaration>"	<i>(none)</i>	<i>(none)</i>
Subclause 12.5, "<assignment statement>"	<i>(none)</i>	<i>(none)</i>
Subclause 12.6, "<case statement>"	<i>(none)</i>	<i>(none)</i>
Subclause 12.7, "<if statement>"	<i>(none)</i>	<i>(none)</i>
Subclause 12.8, "<iterate statement>"	<i>(none)</i>	<i>(none)</i>
Subclause 12.9, "<leave statement>"	<i>(none)</i>	<i>(none)</i>

**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 12.10, "<loop statement>"	<i>(none)</i>	<i>(none)</i>
Subclause 12.11, "<while statement>"	<i>(none)</i>	<i>(none)</i>
Subclause 12.12, "<repeat statement>"	<i>(none)</i>	<i>(none)</i>
Subclause 12.13, "<for statement>"	<i>(none)</i>	<i>(none)</i>
Clause 13, "Diagnostics management"	Clause 18, "Diagnostics management"	ISO/IEC 9075-2
Subclause 13.1, "<get diagnostics statement>"	Subclause 18.1, "<get diagnostics statement>"	ISO/IEC 9075-2
Subclause 13.2, "<signal statement>"	<i>(none)</i>	<i>(none)</i>
Subclause 13.3, "<resignal statement>"	<i>(none)</i>	<i>(none)</i>
Clause 14, "Information Schema and Definition Schema"	Clause 19, "Information Schema, Definition Schema, and Documentation Schema"	ISO/IEC 9075-2
Subclause 14.1, "Information Schema"	Subclause 19.2, "Information Schema"	ISO/IEC 9075-2
Subclause 14.1.1, "MODULES view"	<i>(none)</i>	<i>(none)</i>
Subclause 14.1.2, "MODULE_TABLE_USAGE view"	<i>(none)</i>	<i>(none)</i>
Subclause 14.1.3, "MODULE_COLUMN_USAGE view"	<i>(none)</i>	<i>(none)</i>
Subclause 14.1.4, "MODULE_PRIVILEGES view"	<i>(none)</i>	<i>(none)</i>
Subclause 14.2, "Definition Schema"	Subclause 19.3, "Definition Schema"	ISO/IEC 9075-2
Subclause 14.2.1, "MODULES base table"	<i>(none)</i>	<i>(none)</i>
Subclause 14.2.2, "MODULE_TABLE_USAGE base table"	<i>(none)</i>	<i>(none)</i>
Subclause 14.2.3, "MODULE_COLUMN_USAGE base table"	<i>(none)</i>	<i>(none)</i>
Subclause 14.2.4, "MODULE_PRIVILEGES base table"	<i>(none)</i>	<i>(none)</i>
Clause 15, "Status codes"	Clause 20, "Status codes"	ISO/IEC 9075-2
Subclause 15.1, "SQLSTATE"	Subclause 20.1, "SQLSTATE"	ISO/IEC 9075-2
Clause 16, "Conformance"	Clause 8, "Conformance"	ISO/IEC 9075-1

3.2 Conventions

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 16.1, "Claims of conformance"	Subclause 8.1.4, "Claims of conformance"	ISO/IEC 9075-1
Subclause 16.2, "Extensions and options"	<i>(none)</i>	<i>(none)</i>
Subclause 16.2.1, "Information Schema requirements"	<i>(none)</i>	<i>(none)</i>
Subclause 16.2.2, "Schema manipulation requirements"	<i>(none)</i>	<i>(none)</i>
Subclause 16.3, "Flagger requirements"	<i>(none)</i>	<i>(none)</i>
Annex A, "Implementation-defined elements"	Appendix B, "Implementation-defined elements"	ISO/IEC 9075-2
Annex B, "Implementation-dependent elements"	Appendix C, "Implementation-dependent elements"	ISO/IEC 9075-2
Annex C, "Deprecated features"	Appendix D, "Deprecated features"	ISO/IEC 9075-2
Annex D, "Incompatibilities with ISO/IEC 9075:1992"	Appendix E, "Incompatibilities with X3.135-1992 and ISO/IEC 9075:1992"	ISO/IEC 9075-2
Annex E, "SQL Feature Taxonomy"	Appendix F, "SQL Feature Taxonomy"	ISO/IEC 9075-2
Table 1, "Clause, Subclause, and Table relationships"	<i>(none)</i>	<i>(none)</i>
Table 2, "<identifier>s for use with <get diagnostics statement>"	Table 24, "<identifier>s for use with <get diagnostics statement>"	ISO/IEC 9075-2
Table 3, "SQL-statement codes for use in the diagnostics area"	Table 25, "SQL-statement codes for use in the diagnostics area"	ISO/IEC 9075-2
Table 4, "SQLSTATE class and subclass values"	Table 26, "SQLSTATE class and subclass values"	ISO/IEC 9075-2

### 3.3 Object identifier for Database Language SQL

#### Function

The object identifier for Database Language SQL identifies the characteristics of an SQL-implementation to other entities in an open systems environment.

#### Format

```
<Part 4 yes> ::=
    4 | sqlpsm199x <left paren> 4 <right paren>
```

## Syntax Rules

None.

|

**ISO/IEC JTC1/SC21 N11138 = DBL:CWB-003**

## 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 <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 used for representing the SQL-server module's <identifier>s and <character string literal>s.
- 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

### 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.

**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.

- 2 Subclause2 deleted.

### 4.3 Tables

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

- 1 paragraph deleted.

**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 <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>.

- 1 paragraph deleted.
- 1 Subclause deleted.

### 4.4 SQL-schemas

**Replace 2nd paragraph** In this International standard, 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, an abstract data 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.

- 1 paragraph deleted.

### 4.5 Host parameters

### 4.5.1 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>.

## 4.6 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>.

## 4.7 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.

## 4.8 Condition handling

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

<handler declaration>s specify the handling of exception and completion conditions. <handler declaration>s can be specified in 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 SQLEXCEPTION, SQLWARNING, or NOT FOUND SQLSTATE class. All other <handler declaration>s are *specific* <handler declaration>s.

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

## 4.8 Condition handling

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 outer <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 <compound statement> from which it was invoked and execute all other SQL-statements following the one that raised the condition.

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 12.2, "<handler declaration>".

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

## 4.9 SQL-statements

### 4.9.1 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>

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

- <signal statement>
- <resignal statement>

## 4.9 SQL-statements

### 4.9.2 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

- 1 Subclause deleted.

### 4.9.3 Directly executable SQL-statements

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

— All SQL-control statements

### 4.9.4 Iterated SQL-statement>

The following are the iterated SQL-statements:

- <loop statement>
- <while statement>
- <repeat statement>
- <for statement>

### 4.9.5 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:

- <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>

- 1 paragraph deleted.

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*.

#### 4.9.6 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 12.1, “<compound statement>”.

#### 4.9.7 SQL-statement atomicity

Replace 1st paragraph The execution of all SQL-statements other than SQL-control statements is atomic. The execution of <compound statement>s that specify ATOMIC is atomic. Such an SQL-statement is called an *atomic SQL-statement*.

- 2 paragraphs deleted.

### 4.10 Privileges and roles

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>.

- 1 paragraph deleted.

Insert this paragraph A privilege descriptor with an action of EXECUTE is called an *execute privilege descriptor* and identifies the existence of a privilege on the SQL-server module identified by the privilege descriptor.

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

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

NOTE 2 – “schema-level routine” is defined in Subclause 9.17, “<SQL-invoked routine>”.

**ISO/IEC JTC1/SC21 N11138 = DBL:CWB-003**

## 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 part 2 of ISO/IEC 9075
    | !! All alternatives from part 5 of ISO/IEC 9075
```

- 1 alternatives deleted.

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

    | 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.

**ISO/IEC JTC1/SC21 N11138 = DBL:CWB-003**

**5.1 <token> and <separator>**

## **General Rules**

No additional General Rules.

## 5.2 Names and identifiers

### Function

Specify names.

### Format

- 1 production deleted.

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

- 2 productions deleted.

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

```
<SQL variable name> ::=  
    <identifier>
```

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

- 2 productions deleted.

### Syntax Rules

- 1) Replace SR1 If a <character set specification> is not specified in an <identifier>, then the set of characters contained in the <identifier> shall be wholly contained in either <SQL language character> or the character repertoire indicated by:

Case:

- a) If the <identifier> 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 character set name for the SQL-session,
  - b) If the <identifier> is contained in an <SQL-server module definition> without an intervening <schema definition>, then the <SQL-server module character set specification>,
  - c) If the <identifier> is contained in a <schema definition> that is not contained in an <SQL-client module>, then the <schema character set specification>,
  - d) If the <identifier> is contained in an <SQL-client module definition>, then the <module character set specification>.
- 2) Replace SR13 If a <qualified name> does not contain a <schema name>, then

## 5.2 Names and identifiers

Case:

- a) If the <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.
- b) If the <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.
- c) If the <qualified name> is contained in a <schema definition>, then the <schema name> that is specified or implicit in the <schema definition> is implicit.
- d) Otherwise, the <schema name> that is specified or implicit for the <module> is implicit.

- 1 Rule deleted.

### Access Rules

No additional Access Rules.

### General Rules

- 2 Rules deleted.

- 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.

- 1 Rule deleted.

## 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 part 2 of ISO/IEC 9075
    | !! All alternatives from part 5 of ISO/IEC 9075
    | <SQL variable reference>

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

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

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

```

#### Syntax Rules

No additional Syntax Rules.

#### Access Rules

No additional Access Rules.

#### General Rules

No additional General Rules.

## 6.2 <item reference>

### Function

Reference a column, an SQL parameter, or an SQL variable.

### Format

```
<item name> ::=
    !! All alternatives from part 2 of ISO/IEC 9075
    | !! All alternatives from part 5 of ISO/IEC 9075
    | <SQL variable name>
```

### Syntax Rules

- 1) Replace SR2 If *IR* contains an <item qualifier> *IQ*, then *IR* shall appear within the scope of one or more exposed <table or query name>s or <correlation name>s that are equivalent to *IQ*. If there is more than one such exposed <table or query name> or <correlation name>, then the one with the innermost scope is specified. Let *V* be the collection identified by *IQ*.

- a) *V* shall include a column whose <column name> is equivalent to *IN*.
- b) If *V* is the table identified by a <table reference> in a <joined table> *JT*, then *IN* shall not be a common column name in *JT*.

NOTE 3 – “Common column name” is defined in Subclause 7.6, “<joined table>”, in ISO/IEC 9075-2.

- 2) Replace SR3 If *IR* does not contain an <item qualifier>, then *IR* 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 *IN* or within the scope of one or more <routine name>s or <beginning label>s whose associated <SQL parameter list> or <local declaration list> includes an SQL parameter or SQL variable whose <identifier> is equivalent to *IN*. Let the phrase *possible qualifiers* denote those exposed <table or query name>s, <correlation name>s, <routine name>s, and <beginning label>s.

- a) Case:

- i) If the innermost scope contains exactly one possible qualifier, then

Case:

- 1) If the possible qualifier with innermost scope is a <table or query name> or a <correlation name>, then the qualifier *IQ* equivalent to that unique exposed <table or query name> or <correlation name> is implicit. Let *V* be the table identified by *IQ*.
- 2) If the possible qualifier with innermost scope is a <routine name>, then let *V* be the SQL parameter list associated with this possible qualifier.
- 3) If the possible qualifier with innermost scope is a <beginning label>, then let *V* be the <local declaration list> associated with this possible qualifier.

- ii) If there is more than one possible qualifier within the innermost scope, then:
- 1) Each possible qualifier shall be a <table or query name> or a <correlation name> of a <table reference> that is directly contained in a <joined table> *JT*.
  - 2) *IN* shall be a common column name in *JT*.  
NOTE 4 – “Common column name” is defined in Subclause 7.6, “<joined table>”, in ISO/IEC 9075-2.
  - 3) The implicit qualifier *IQ* is implementation-dependent. The scope of *IQ* is that which *IQ* would have had if *JT* had been replaced by the <table reference>:

(*JT*) AS *IQ*

- 4) Let *V* be the table identified by *IQ*.

3) Replace SR4 Case:

- a) If *V* is a table, then *IR* is a *column reference*. *IN* shall uniquely identify a column of *V*. Let *R* be that column.
  - i) *R* is an *underlying column* of *IR*. If *R* is a <derived column>, then every underlying column of *R* is an underlying column of *IR*.  
NOTE 5 – The underlying columns of a <derived column> are defined in Subclause 7.10, “<query specification>”, in ISO/IEC 9075-2.
  - ii) If *IR* is contained in a <table expression> *TE* or a <quantified predicate> *TE* and the scope of *IQ* is some <SQL procedure statement>, <trigger definition>, <quantified predicate>, or <table reference> that contains *TE*, then *IR* is an *outer reference* to the table associated with *IQ*.
- b) If *V* is an SQL parameter list, then *IR* is an *SQL variable reference*. Let *R* be the SQL parameter of *V* identified by *IN*.
- c) If *V* is a <local declaration list>, then *IR* is an *SQL variable reference*. Let *R* be the SQL variable of *V* identified by *IN*.

## Access Rules

None.

## General Rules

- 1) Replace GR1 Depending on whether *IR* is a column reference, SQL parameter reference, or SQL variable reference, *IR* references column *IN* in a given row of *V*, SQL parameter *IN* of a given invocation of the SQL-invoked routine that contains *V*, or SQL variable *IN* of a given execution of the <compound statement> whose <local declaration list> is *V*.

## 6.3 <SQL variable reference>

### Function

Reference an SQL variable.

### Format

```
<SQL variable reference> ::=  
    <item reference>
```

### Syntax Rules

- 1) An <SQL variable reference> shall be an <item reference> that is an SQL variable reference.

### Access Rules

None.

### General Rules

None.

## 6.4 <datetime value function>

### Function

Specify a function yielding a value of type datetime.

### Format

*No additional Format items.*

### Syntax Rules

No additional Syntax Rules.

### Access Rules

No additional Access Rules.

### General Rules

- 1) Replace GR3 If an <SQL procedure statement> *S* that is not generally contained in a <triggered action> contains, without an intervening <SQL procedure statement>, one or more <value expression>s that generally contain, without an intervening <routine invocation> whose subject routines do not include an SQL function, one or more <datetime value function>s, then all such <datetime value function>s are effectively evaluated simultaneously. The time of evaluation of a <datetime value function> during the execution of *S* is implementation-dependent.

- 1 Subclause deleted.

**ISO/IEC JTC1/SC21 N11138 = DBL:CWB-003**

## 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) Replace SR1) in Part 5 A column of *TQS* is *possibly nullable* if and only if it contains a column reference for a column *C* that is possibly nullable, an <indicator parameter>, an indicator variable, a dynamic parameter specification, an SQL parameter, an SQL variable, a <routine invocation> whose subject routines include an SQL-invoked routine that either is an SQL routine or is an external routine that specifies or implies PARAMETER STYLE SQL, a <subquery>, CAST (NULL AS *X*) (*X* represents a <data type> or a <domain name>), SYSTEM\_USER, or a <set function specification> that does not contain COUNT.

#### Access Rules

No additional Access Rules.

#### General Rules

No additional General Rules.

**ISO/IEC JTC1/SC21 N11138 = DBL:CWB-003**

## 8 Additional common elements

### 8.1 <routine invocation>

#### Function

Invoke an SQL-invoked routine.

#### Format

*No additional Format items.*

#### Syntax Rules

Replace SR3) 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 user privileges include EXECUTE on *M*.
- ii) Otherwise, the user privileges include EXECUTE on *R*.

- b) Otherwise,

Case:

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

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

- 1) Insert before SR5)b)i)1)B)l) If *RI* is contained in an <SQL-server module definition>, then let *DP* be the SQL-path of that <SQL-server module definition>.
- 2) Replace SR5)b)i)1)B)l) 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>.
- 3) Insert before SR6)c)i)1)B)l) If *RI* is contained in an <SQL-server module definition>, then let *DP* be the SQL-path of that <SQL-server module definition>.
- 4) Replace SR6)c)i)1)B)l) 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>.

### **Access Rules**

No additional Access Rules.

### **General Rules**

No additional General Rules.

## 8.2 <privileges>

### Function

Specify privileges.

### Format

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

### Syntax Rules

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

NOTE 7 – “schema-level routine” is defined in Subclause 9.17, “<SQL-invoked routine>”.

### Access Rules

No additional Access Rules.

### General Rules

No additional General Rules.

### 8.3 <sqlstate value>

## 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 that is the value of 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) The value of  $V$  shall comprise either:
  - a) A standard SQLSTATE Class and Subclass value;
  - b) A standard SQLSTATE Class value for which in implementation-defined Subclass value is permitted and three characters with the form of an implementation-defined Subclass value;  
or
  - c) Five characters of which the first two have the form of an implementation-defined Class value.
- 5) The value of  $V$  shall not be the SQLSTATE value for the condition *successful completion*.
- 6) The SQLSTATE value defined by the <sqlstate value> is the value of  $V$ .

### 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 part 2 of ISO/IEC 9075  
    | !! All alternatives from part 5 of ISO/IEC 9075  
    | <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 contains 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>, <abstract data type definition>, <alter column definition>, or <alter domain statement>, or that defined by the <data type> specified in the containing <SQL variable declaration>.

**\*\*Editor's Note\*\***

Len Gallagher notes that a length of 128 characters is sufficient for CURRENT\_USER, SESSION\_USER, and SYSTEM\_USER because they are single identifiers, which are limited to 128 characters in all levels of SQL. However, CURRENT\_PATH is a *list* of <schema name>s, each of which could be 257 characters in length (128 for the <catalog name>, 1 for the <period>, and 128 for the <unqualified schema name>). Therefore, it is inappropriate for 128 to be the floor for CURRENT\_PATH; some (much?) larger floor is required.

- 1 Rule deleted.

### Access Rules

No additional Access Rules.

### General Rules

- 1 Note and 1 Rule deleted.

- 1) Insert before GR5 The default value of an SQL variable is  
Case:
  - a) If the <SQL variable declaration> contains a <default clause>, then the value specified by that <default clause>.
  - b) Otherwise, the null value.
- 2) Insert before GR5 When a default value is required for an SQL variable, the default value for the variable is derived from the <default option> as follows:
  - a) If the <default option> specifies NULL, then the null value.
  - b) If the <default option> contains a <literal>, then  
Case:
    - i) If the subject data type is numeric, then the numeric value of the <literal>.

**9.3 <default clause>**

- ii) If the subject data type is character string with variable length, then the value of the <literal>.
- iii) If the subject data type is character string with fixed length, then the value of the <literal>, extended as necessary on the right with <space>s to the length in characters of the subject data type.
- iv) If the subject data type is bit string with variable length, then the value of the <literal>.
- v) If the subject data type is datetime or interval, then the value of the <literal>.
- c) If the <default option> specifies CURRENT\_USER, SESSION\_USER, SYSTEM\_USER, or CURRENT\_PATH, then  
Case:
  - i) If the subject data type is character string with variable length, then the value obtained by an evaluation of CURRENT\_USER, SESSION\_USER, SYSTEM\_USER, or CURRENT\_PATH, respectively, at the time that the default value is required.
  - ii) If the subject data type is character string with fixed length, then the value obtained by an evaluation of CURRENT\_USER, SESSION\_USER, SYSTEM\_USER, or CURRENT\_PATH, respectively, at the time that the default value is required, extended as necessary on the right with <space>s to the length in characters of the subject data type.
- d) If the <default option> contains a <datetime value function>, then the value of an evaluation of the <datetime value function> at the time that the default value is required.

- 1 subrule deleted.

## 9.4 <drop column definition>

### Function

Destroy a column.

### Format

*No additional Format items.*

### Syntax Rules

- 1) Replace SR4 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.

### 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
```

## 9.5 <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

## 9.6 <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 GR3 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 GR3 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

## 9.7 <view definition>

### Function

Define a viewed table.

### Format

*No additional Format items.*

### Syntax Rules

1) Insert after SR2)

NOTE 8 – <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.

## 9.8 <drop view statement>

### Function

Destroy a view.

### Format

*No additional Format items.*

### Syntax Rules

- 1) Insert after SR3 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 GR2 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 GR2 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
```

## 9.9 <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 Rule deleted.

- 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

## 9.10 <drop character set statement>

### Function

Destroy a character set.

### Format

*No additional Format items.*

### Syntax Rules

- 1) Insert after SR3 *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

## 9.11 <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
```

## 9.12 <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 GR3) 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
```

## 9.13 <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 9 – <SQL variable name> is also excluded because of the scoping rules for <SQL variable naem>.

### Access Rules

No additional Access Rules.

### General Rules

No additional General Rules.

## 9.14 <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

## 9.15 <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 10 – “Schema-level routine” is defined in Subclause 9.17, “<SQL-invoked routine>”.
- 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.

### Access Rules

- 1) If an <SQL-server module definition> is contained in a <module> with no intervening <schema definition>, then the current <authorization identifier> shall be equivalent to 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 11 – 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 default 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.

**9.15 <SQL-server module definition>**

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

## 9.16 <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.
- 3 subrules deleted.
    - e) The module descriptor of any SQL-server module other than *M*.

### Access Rules

- 1) The current <authorization identifier> shall be equivalent to 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.

## 9.17 <SQL-invoked routine>

### Function

Define an SQL-invoked routine.

### Format

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

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

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

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

<module iterative routine> ::=
    [ DECLARE ] <iterative routine>
```

### Syntax Rules

- 1) Replace SR3 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 SR10 If <SQL-invoked routine> is contained in a <schema definition> without an intervening <SQL-server module definition> and if *RN* contains a <schema name>, then that <schema name> shall be equivalent to the specified <schema name> of the containing <schema definition>.
  - 3) Insert after SR10 If <SQL-invoked routine> is contained in an <SQL-server module definition> and if *RN* contains a <schema name>, then that <schema name> shall be equivalent to the specified <schema name> of the containing <SQL-server module definition>.
- 3 Rules deleted.

### Access Rules

No additional Access Rules.

## General Rules

- 1) Replace GR3u An indication of whether the routine is a schema-level routine.
- 2) Insert after GR3u 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.

## 9.18 <drop routine statement>

### Function

Destroy an SQL-invoked routine.

### Format

*No additional Format items.*

### Syntax Rules

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

### Access Rules

No additional Access Rules.

### General Rules

No additional General Rules.

## 9.19 <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 Rule deleted.

- 1) Insert this GR For every <grantee> *G* and for every SQL-server module *MI* owned by *G*, if the user 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 *MI* WITH GRANT OPTION, then for every privilege descriptor with a <privileges> EXECUTE, a <grantor> of “\_SYSTEM”, <object> of *MI*, and <grantee> *G* that is not grantable, the following <grant statement> is executed with a current <authorization identifier> of “\_SYSTEM” and without further Access Rule checking:

GRANT EXECUTE ON *MI* TO *G* WITH GRANT OPTION.

NOTE 12 – The privileges necessary include the EXECUTE privilege on every subject routine of every <routine invocation> contained in those <SQL procedure statement>s.

## 9.20 <revoke statement>

### Function

Destroy privileges.

### Format

*No additional Format items.*

### Syntax Rules

- 1) Replace SR12) Let *V* be any view descriptor included in *S1*. *V* is said to be *abandoned* if the destruction of all abandoned privilege descriptors and, if GRANT OPTION FOR is not specified, all identified privilege descriptors would result in *A1* no longer having any of the following:
  - a) SELECT privileges on one or more tables contained in the <query expression> of *V*.
  - b) USAGE privilege on one or more domains, user-defined types, collations, character sets, or translations whose names are contained in the <query expression> of *V*.
  - c) EXECUTE privilege on one or more SQL-invoked routines that are one of the subject SQL-invoked routines of a <routine invocation> that is contained in the <query expression> of *V*.
  - d) EXECUTE privilege on one or more SQL-server modules that include 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) Replace SR13) Let *TC* be any table constraint descriptor included in *S1*. *TC* is said to be *abandoned* if the destruction of all abandoned privilege descriptors and, if GRANT OPTION FOR is not specified, all identified privilege descriptors, would result in *A1* no longer having any of the following:
  - a) REFERENCES privilege on one or more columns referenced in any <search condition> of *TC*.
  - b) USAGE privilege on one or more domains, user-defined types, collations, character sets, or translations whose names are contained in any <search condition> of *TC*.
  - c) EXECUTE privilege on one or more SQL-invoked routines that are among the subject routines of a <routine invocation> that is contained in any <search condition> of *TC*.
  - d) EXECUTE privilege on one or more SQL-server modules that include 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) Replace SR14) Let *AX* be any assertion descriptor included in *S1*. *AX* is said to be *abandoned* if the destruction of all abandoned privilege descriptors and, if GRANT OPTION FOR is not specified, all identified privilege descriptors would result in *A1* no longer having any of the following:
  - a) REFERENCES privilege on one or more referenced columns of *AX*.

- b) USAGE privilege on one or more domains, user-defined types, collations, character sets, or translations whose names are contained in any <search condition> of *AX*.
  - c) EXECUTE privilege on one or more SQL-invoked routines that are among the subject routines of a <routine invocation> that is contained in any <search condition> of *AX*.
  - d) EXECUTE privilege on one or more SQL-server modules that include 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) Replace SR16) Let *DC* be any domain constraint descriptor included in *S1*. *DC* is said to be *abandoned* if the destruction of all abandoned privilege descriptors and, if GRANT OPTION FOR is not specified, all identified privilege descriptors would result in *A1* no longer having any of the following:
- a) REFERENCES privilege on one or more referenced columns of *DC*.
  - b) USAGE privilege on one or more domains, user-defined types, collations, character sets, or translations whose names are contained in any <search condition> of *DC*.
  - c) EXECUTE privilege on one or more SQL-invoked routines that are among the subject routines of a <routine invocation> that is contained in any <search condition> of *DC*.
  - d) EXECUTE privilege on one or more SQL-server modules that include 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) Replace SR26) Let *RD* be any routine descriptor included in *S1*. *RD* is said to be *abandoned* if the destruction of all abandoned privilege descriptors and, if GRANT OPTION FOR is not specified, all identified privilege descriptors would result in *A1* no longer satisfying one or more of the following criteria:
- a) Having EXECUTE privilege on 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*.
  - b) Having EXECUTE privilege on one or more SQL-server modules that include 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*.
  - c) Having SELECT privilege on each <table reference> contained in a <query expression> simply contained in a <cursor specification> or an <insert statement> contained in the <SQL routine body> of *RD*.
  - d) Having SELECT privilege on each <table reference> contained in a <table expression> or <select list> immediately contained in a <select statement: single row> contained in the <SQL routine body> of *RD*.
  - e) Having SELECT privilege on each <table reference> and <column reference> contained in a <search condition> contained in a <delete statement: searched> or an <update statement: searched> contained in the <SQL routine body> of *RD*.
  - f) Having SELECT privilege on each <table reference> and <column reference> contained in a <value expression> immediately contained in an <update source> contained in the <SQL routine body> of *RD*.
  - g) Having INSERT privileges on each column

9.20 <revoke statement>

Case:

- i) Named in the <insert column list> of an <insert statement> contained in the <SQL routine body> of *RD*.
  - ii) Of the table identified by the <table name> immediately contained in an <insert statement> that does not contain an <insert column list> and that is contained in the <SQL routine body> of *RD*.
  - h) Having UPDATE privileges on each 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 *RD*.
  - i) Having DELETE privileges on each table whose name is contained in a <table name> contained in either a <delete statement: positioned> or a <delete statement: searched> contained in the <SQL routine body> of *RD*.
  - j) Having USAGE privilege on each domain, user-defined type collation, character set, and translation whose name is contained in the <SQL routine body> of *RD*.
- 6) Insert this SR Let *SSM* be any SQL-server module descriptor of an SQL-server module included in *SI*. *SSM* is said to be *abandoned* if the destruction of all abandoned privilege descriptors and, if GRANT OPTION FOR is not specified, all identified privilege descriptors would result in *A1* no longer satisfying one or more of the following criteria:
- a) Having EXECUTE privilege on one or more schema-level routines that are among the subject routines of a <routine invocation> that is generally contained in the <routine body> of any SQL-invoked routine included in *SSM*.
  - b) Having EXECUTE privilege on one or more SQL-server modules that include one or more SQL-invoked routines that are among the subject routines of a <routine invocation> that is generally contained in the <routine body> of any SQL-invoked routine included in *SSM*.
  - c) Having SELECT privileges on each <table reference> contained in a <query expression> simply contained in a <cursor specification> or an <insert statement> generally contained in the <routine body> of any SQL-invoked routine included in *SSM*.
  - d) Having SELECT privileges on each <table reference> contained in a <table expression> or <select list> immediately contained in a <select statement: single row> generally contained in the <routine body> of any SQL-invoked routine included in *SSM*.
  - e) Having SELECT privileges on each <table reference> and <column reference> contained in a <search condition> contained in a <delete statement: positioned> or an <update statement: searched> generally contained in the <routine body> of any SQL-invoked routine included in *SSM*.
  - f) Having SELECT privileges on each <table reference> and <column reference> contained in a <value expression> immediately contained in an <update source> generally contained in the <routine body> of any SQL-invoked routine included in *SSM*.
  - g) Having INSERT privileges on each column

Case:

- i) Named in the <insert column list> of an <insert statement> generally 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 generally contained in the <routine body> of any SQL-invoked routine included in *SSM*.
- h) Having UPDATE privileges on each column whose name is contained in an <object column> contained in either an <update statement: positioned> or an <update statement: searched> generally contained in the <routine body> of any SQL-invoked routine included in *SSM*.
- i) Having DELETE privileges on each table whose name is contained in a <table name> immediately contained in either a <delete statement: positioned> or a <delete statement: searched> generally contained in the <routine body> of any SQL-invoked routine included in *SSM*.
- j) Having USAGE privilege on each domain, user-defined type collation, character set, and translation whose name is contained in the <routine body> of any SQL-invoked routine included in *SSM*.

## 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
```





## 10 SQL-client modules

- 1 Subclause deleted.

## 10.1 Calls to an <externally-invoked procedure>

### Function

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

### Syntax Rules

1) Insert into SR1)c

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

### Access Rules

No additional Access Rules.

### General Rules

No additional General Rules.

## 10.2 <SQL procedure statement>

### Function

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

### Format

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

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

<SQL control statement> ::=
    !! All alternatives from part 2 of ISO/IEC 9075
    | !! All alternatives from part 5 of ISO/IEC 9075
    | <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 part 2 of ISO/IEC 9075
    | !! All alternatives from part 5 of ISO/IEC 9075
    | <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, SESSION\_USER, 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, SESSION\_USER, SYSTEM\_USER.
- 4) Replace SR 7) An <SQL schema statement> shall not be generally contained in an <SQL-invoked routine> or in an <SQL-server module definition>.

### **Access Rules**

No additional Access Rules.

### **General Rules**

No additional General Rules.

## 11 Data manipulation

- 1 Subclause deleted.

### 11.1 <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.

## 11.2 <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 SR2 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*.

### Access Rules

No additional Access Rules.

### General Rules

No additional General Rules.

## 11.3 <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.

## 11.4 <select statement: single row>

### Function

Retrieve values from a specified row of a table.

### Format

*No additional Format items.*

### Syntax Rules

- 1) Insert after SR2)a)iii) If *TS* is an <SQL variable name>, then the Syntax Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2, shall apply to *TS* and a <row value constructor> whose *i*-th <row value constructor element> is the *i*-th element of the <select list>, as *TARGET* and *VALUE*, respectively.
- 2) Insert after SR2)b)iii) 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 GR4)a)ii) If the <target specification> *TS* is an <SQL variable name>, then the first 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.
- 2) Insert after GR4)b)i)2) If the <select target list> contains a single <target specification> *TS* that is an <SQL variable name> and the number of elements in the <select list> is greater than 1 (one), then 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.
- 3) Insert after GR4)b)ii)2) 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.

## 11.5 <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.

## 11.6 <delete statement: searched>

### Function

Delete rows of a table.

### Format

*No additional Format items.*

### Syntax Rules

- 1) Insert this SR The <search condition> shall not generally contain a <routine invocation> whose subject routines include an SQL-invoked routine that possibly modifies SQL-data.

### Access Rules

No additional Access Rules.

### General Rules

No additional General Rules.

## 11.7 <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.

## 11.8 <update statement: searched>

### Function

Update rows of a table.

### Format

*No additional Format items.*

### Syntax Rules

- 1) Insert this SR The <search condition> shall not generally contain a <routine invocation> whose subject routines include an SQL-invoked routine that possibly modifies SQL-data.

### Access Rules

No additional Access Rules.

### General Rules

No additional General Rules.

## 11.9 <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>.

11.9 <temporary table declaration>

2) Replace GR2 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*.

## 12 Control statements

### 12.1 <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>

```

12.1 <compound statement>

**Syntax Rules**

- 1) Let *CS* be the <compound statement>.
- 2) If *CS* is contained in another <compound statement> and *CS* does not specify a <beginning label>, then an implementation-dependent <beginning label> 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*. <beginning label> shall not be equivalent to any other <beginning label>s contained in *CS* excluding every <SQL schema statement> contained in *CS*.
- 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 REDO or 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 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 12.6, "<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 statement is effectively executed:  
CLOSE *CR*
- 5) The 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.

**ISO/IEC JTC1/SC21 N11138 = DBL:CWB-003**

**12.1 <compound statement>**

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

## 12.2 <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 a <handler declaration> 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 20.1, "SQLSTATE", in ISO/IEC 9075-2.

## 12.2 <handler declaration>

- 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.

**\*\*Editor's Note\*\***

The SQL3 and SQL4 case of module-level exception declarations is not addressed in the previous General Rule.

- 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 one that raised the condition in *CC*.

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*.

## 12.3 <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>.

## 12.4 <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

None.

### 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
```

## 12.5 <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>
    | <null specification>

<modified field reference> ::=
    <modified field target> <double greater than operator> <field name>

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

<mutator reference> ::=
    <mutated target> <double greater than operator> <function name>

<mutated target> ::=
    <target specification>
    | <mutator reference>

```

### Syntax Rules

- 1) An <assignment target> that is a <modified field reference> shall not contain a column reference.
- 2) The data type of the <target specification> simply contained in a <mutator reference> *MR* shall be an abstract data type.
- 3) If <assignment target> immediately contains a <mutator reference>, then let *TS* be the <mutated target>, let *FN* be the <function name>, and let *AS* be the <assignment source>. The <assignment statement> is equivalent to:

$$\text{SET } TS = FN(TS, AS)$$

NOTE 13 – 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 an <SQL variable name> or the <SQL parameter name> of a parameter of an SQL-invoked routine 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.

### Access Rules

None.

### General Rules

- 1) If <assignment target> is a <target specification> that is the <SQL variable name> of an SQL variable  $T$  or the <SQL parameter name> of 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 <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.

## 12.6 <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 data type of each <simple case operand 2> *SCO2* shall be comparable with the data type of *SCO1*.
  - c) The <simple case statement> is equivalent to a <searched case statement> in which each <searched 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 12.6, "<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 terminated with that condition.

## 12.7 <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 14 – “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 12.6, "<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.

## 12.8 <iterate statement>

### Function

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

### Format

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

### Syntax Rules

- 1) The <statement label> shall be the <beginning label> *BL* of some iterated SQL-statement *IS* and <iterate statement> shall be contained in the scope of *BL*.
- 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 15 – If the iteration condition for *IS* is true , then the next iteration of *SSL* commences immediately.

## 12.9 <leave statement>

### Function

Continue execution by leaving 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.

12.10 <loop statement>

## 12.10 <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) If <ending label> is specified, then a <beginning label> shall be specified that is equivalent to <ending label>.
- 2) Let *LS* be the <loop statement>. 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 12.6, "<SQL procedure statement>", of ISO/IEC 9075-2, are evaluated repeatedly with *CCS* as the *executing statement*.

NOTE 16 – 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 12.9, "<leave statement>", in ISO/IEC 9075-4, respectively. Some actions taken by a condition handler might also cause execution of *LS* to be terminated; see Subclause 12.2, "<handler declaration>".

## 12.11 <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) If <ending label> is specified, then a <beginning label> shall be specified that is equivalent to <ending label>.
- 2) Let *WS* be the <while statement>. 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 12.6, "<SQL procedure statement>", of ISO/IEC 9075-2, are evaluated with *CCS* as the *executing statement* and the execution of *WS* is repeated.

NOTE 17 – 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 12.9, "<leave statement>", in ISO/IEC 9075-4, respectively. Some actions taken by a condition handler might also cause execution of *LS* to be terminated; see Subclause 12.2, "<handler declaration>".

## 12.12 <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) If <ending label> is specified, then a <beginning label> shall be specified that is equivalent to <ending label>.
- 2) Let *RS* be the <repeat statement>. 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 12.6, "<SQL procedure statement>", of ISO/IEC 9075-2, are evaluated with *CCS* as the *executing statement* and then <search condition> is evaluated.

NOTE 18 – 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 12.9, "<leave statement>", in ISO/IEC 9075-4, respectively. Some actions taken by a condition handler might also cause execution of *LS* to be terminated; see Subclause 12.2, "<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.

## 12.13 <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 <ending label> is specified, then a <beginning label> shall be specified that is equivalent to <ending label>.
- 3) If <cursor name> is specified, then let *CN* be that <cursor name>. Otherwise, let *CN* be an implementation-dependent <cursor name> that is different from any other <cursor name> in the outermost containing <SQL-client module definition> or <SQL-invoked routine>.
- 4) Let *QE* be the <query expression> of *FCS*. Each column of the table specified by *QE* shall have a <column name> that is different from 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 data types of the respective columns.
- 5) Let *BL*, *FLVN*, and *SLL* be the <beginning label>, <for loop variable name>, and <SQL statement list> of *FS*, respectively.
  - a) If *BL* is not specified, then let *BL* be an implementation-dependent <statement label> that is different from any other <statement label> contained in the outermost containing <SQL control statement>.
  - b) Let *AT\_END* be an implementation-dependent <SQL variable name> that is different from 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>.
  - c) Let *NOT\_FOUND* be an implementation-dependent <condition name> that is different from any other <condition name> contained in the outermost containing <SQL-server module definition>, <SQL-invoked routine>, or <compound statement>.
  - d) Let *CS* be the explicit or implicit <cursor sensitivity>.

**12.13 <for statement>**

The <for statement> is equivalent to:

```
BL: BEGIN

    FLVN: BEGIN
        DECLARE CN CS CURSOR FOR
            SELECT ROW Q 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
        DECLARE CONTINUE HANDLER FOR NOT_FOUND
            SET AT_END = TRUE;

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

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

- 1 Subclause deleted.

**ISO/IEC JTC1/SC21 N11138 = DBL:CWB-003**

## 13 Diagnostics management

### 13.1 <get diagnostics statement>

#### Function

Get exception or completion condition information from the diagnostics area.

#### Format

```
<condition information item name> ::=
    !! All alternatives from part 2 of ISO/IEC 9075
    | !! All alternatives from part 5 of ISO/IEC 9075
    | CONDITION_IDENTIFIER
    | ROUTINE_CATALOG
    | ROUTINE_SCHEMA
    | ROUTINE_NAME
    | SPECIFIC_NAME
    | PARAMETER_NAME
```

#### Syntax Rules

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

<identifier>	Data Type
<b>&lt;statement information item name&gt;s</b>	
<i>All alternatives from part 2 of ISO/IEC 9075</i> <i>All alternatives from part 5 of ISO/IEC 9075</i>	
<b>&lt;condition information item name&gt;s</b>	
<i>All alternatives from part 2 of ISO/IEC 9075</i> <i>All alternatives from part 5 of ISO/IEC 9075</i> CONDITION_IDENTIFIER	character varying (L)

• 5 table rows deleted.

#### Access Rules

No additional Access Rules.

13.1 <get diagnostics statement>

General Rules

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

SQL-statement	Identifier	Code
<i>All alternatives from part 2 of ISO/IEC 9075</i>		
<i>All alternatives from part 5 of ISO/IEC 9075</i>		
<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 COMMAND\_FUNCTION or DYNAMIC\_FUNCTION identifies a <signal statement> or <resignal statement>, then the values of CLASS\_ORIGIN, SUBCLASS\_ORIGIN, CONSTRAINT\_CATALOG, CONSTRAINT\_SCHEMA, CONSTRAINT\_NAME, CATALOG\_NAME, SCHEMA\_NAME, TABLE\_NAME, COLUMN\_NAME, CURSOR\_NAME, MESSAGE\_TEXT, MESSAGE\_LENGTH, and MESSAGE\_OCTET\_LENGTH are not set as specified in ISO/IEC 9075-2, but instead are set as specified in Subclause 13.3, “<resignal statement>”.
- 2) 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.

## 13.2 <signal statement>

### Function

Signal an exception condition.

### Format

```
<signal statement> ::=  
    SIGNAL <signal value>
```

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

### 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.

### 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 following <resignal statement> is effectively executed without further Syntax Rule checking:

```
RESIGNAL SSI
```

## 13.3 <resignal statement>

### Function

Resignal an exception condition.

### Format

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

### 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 *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.

### Access Rules

None.

### General Rules

- 1) 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.

2) 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 this <resignal statement> is terminated with this exception condition.

NOTE 19 – 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 this <resignal statement> is terminated with the exception condition *unhandled user-defined exception*.

**ISO/IEC JTC1/SC21 N11138 = DBL:CWB-003**

## 14 Information Schema and Definition Schema

### 14.1 Information Schema

- 1 Subclause deleted.

#### 14.1.1 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,
    CASE WHEN ( MODULE_CATALOG, MODULE_SCHEMA, CURRENT_USER )
      IN ( SELECT CATALOG_NAME, SCHEMA_NAME, SCHEMA_OWNER
          FROM DEFINITION_SCHEMA.SCHEMATA )
      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 GRANTEE IN ( 'PUBLIC', CURRENT_USER )
        )
  AND MODULE_CATALOG
     = ( SELECT CATALOG_NAME FROM INFORMATION_SCHEMA_CATALOG_NAME )
```

**14.1 Information Schema**

**14.1.2 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
    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
AND MODULE_CATALOG = ( SELECT CATALOG_NAME
                        FROM INFORMATION_SCHEMA_CATALOG_NAME )
```

### 14.1.3 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
AND MODULE_CATALOG = ( SELECT CATALOG_NAME
                        FROM INFORMATION_SCHEMA.CATALOG_NAME )
```

## 14.1 Information Schema

### 14.1.4 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 )
```

## 14.2 Definition Schema

- 1 Subclause deleted.

### 14.2.1 MODULES base table

#### Function

The MODULES base table has one row for each SQL-server module.

#### Definition

```
CREATE TABLE MODULES
(
  MODULE_CATALOG          INFORMATION_SCHEMA.SQL_IDENTIFIER,
  MODULE_SCHEMA           INFORMATION_SCHEMA.SQL_IDENTIFIER,
  MODULE_NAME              INFORMATION_SCHEMA.SQL_IDENTIFIER,
  DEFAULT_CHARACTER_SET_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT MODULES_DEFAULT_CHARACTER_SET_CATALOG_NOT_NULL NOT NULL,
  DEFAULT_CHARACTER_SET_SCHEMA INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT MODULES_DEFAULT_CHARACTER_SET_SCHEMA_NOT_NULL NOT NULL,
  DEFAULT_CHARACTER_SET_NAME INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT MODULES_DEFAULT_CHARACTER_SET_NAME_NOT_NULL NOT NULL,
  DEFAULT_SCHEMA_CATALOG  INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT MODULES_DEFAULT_SCHEMA_CATALOG_NOT_NULL NOT NULL,
  DEFAULT_SCHEMA_NAME     INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT MODULES_DEFAULT_SCHEMA_NAME_NOT_NULL NOT NULL,

  MODULE_DEFINITION       INFORMATION_SCHEMA.CHARACTER_DATA,
  MODULE_AUTHORIZATION    INFORMATION_SCHEMA.SQL_IDENTIFIER
  CONSTRAINT AUTHORIZATION_FOREIGN_KEY_USERS REFERENCES USERS,
  SQL_PATH                INFORMATION_SCHEMA.CHARACTER_DATA,
  MODULE_CREATED          INFORMATION_SCHEMA.TIME_STAMP,
  MODULE_LAST_ALTERED    INFORMATION_SCHEMA.TIME_STAMP,

  CONSTRAINT MODULES_PRIMARY_KEY
    PRIMARY KEY ( MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME ),

  CONSTRAINT MODULES_FOREIGN_KEY_SCHEMATA
    FOREIGN KEY ( MODULE_CATALOG, MODULE_SCHEMA )
    REFERENCES SCHEMATA,

  CONSTRAINT MODULES_FOREIGN_KEY_CHARACTER_SETS
    FOREIGN KEY ( DEFAULT_CHARACTER_SET_CATALOG, DEFAULT_CHARACTER_SET_SCHEMA,
                 DEFAULT_CHARACTER_SET_NAME ) REFERENCES CHARACTER_SETS ),

  CONSTRAINT MODULES_FOREIGN_KEY_DEFAULT_SCHEMA_SCHEMATA
    FOREIGN KEY ( DEFAULT_SCHEMA_CATALOG, DEFAULT_SCHEMA_NAME )
    REFERENCES SCHEMATA )
)
```

## 14.2 Definition Schema

### Description

- 1) The values of `MODULE_CATALOG`, `MODULE_SCHEMA`, and `MODULE_NAME` are the catalog name, unqualified schema name, and qualified identifier of the module name of the SQL-server module being described.
- 2) The values of `DEFAULT_CHARACTER_SET_CATALOG`, `DEFAULT_CHARACTER_SET_SCHEMA`, and `DEFAULT_CHARACTER_SET_NAME` are the catalog name, unqualified schema name, and qualified identifier, respectively, of the character set identified by the implicit or explicit `<SQL-server module character set specification>`.
- 3) The values of `DEFAULT_SCHEMA_CATALOG`, and `DEFAULT_SCHEMA_NAME` are the catalog name, and unqualified schema name, respectively, of the schema identified by the implicit or explicit `<SQL-server module schema clause>`.
- 4) Case:
  - a) If the character representation of the `<SQL-server module definition>` that defined the SQL-server module being described can be represented without truncation, then the value of `MODULE_DEFINITION` is that character representation.
  - b) Otherwise, the value of `MODULE_DEFINITION` is the null value.

NOTE 20 – Any implicit `<column reference>`s that were contained in the `<SQL-server module definition>` are replaced by explicit `<column reference>`s in `MODULE_DEFINITION`.
- 5) Case:
  - a) If `AUTHORIZATION` was specified in `<module authorization clause>` in the SQL-server module being described, then the value of `MODULE_AUTHORIZATION` is `<module authorization identifier>`.
  - b) Otherwise, the value of `MODULE_AUTHORIZATION` is the null value.
- 6) Case:
  - a) If `<SQL-server module path specification>` was specified in the `<SQL-server module definition>` that defined the SQL-server module described by this row and the character representation of the `<SQL-server module path specification>` can be represented without truncation, then the value of `SQL_PATH` is that character representation.
  - b) Otherwise, the value of `SQL_PATH` is the null value.
- 7) The value of `MODULE_CREATED` is the value of `CURRENT_TIMESTAMP` at the time when the SQL-server module being described was created.
- 8) The value of `MODULE_LAST_ALTERED` is the value of `CURRENT_TIMESTAMP` at the time that the SQL-server module being described was last altered. This value is identical to the value of `MODULE_CREATED` for SQL-server modules that have never been altered.

### 14.2.2 MODULE\_TABLE\_USAGE base table

#### Function

The MODULE\_TABLE\_USAGE table has one row for each table referenced in an SQL-server module.

#### Definition

```
CREATE TABLE MODULE_TABLE_USAGE
(
  MODULE_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
  MODULE_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
  MODULE_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,

  CONSTRAINT MODULE_TABLE_USAGE_PRIMARY_KEY
    PRIMARY KEY ( MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME,
                 TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ),

  CONSTRAINT MODULE_TABLE_USAGE_CHECK_REFERENCES_TABLES
    CHECK ( TABLE_CATALOG <> ANY ( SELECT CATALOG_NAME FROM SCHEMATA )
    OR
    ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) IN
    ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME FROM TABLES ) ),

  CONSTRAINT MODULE_TABLE_USAGE_FOREIGN_KEY_MODULES
    FOREIGN KEY ( MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME )
    REFERENCES MODULES
)
)
```

#### Description

- 1) The values of MODULE\_CATALOG, MODULE\_SCHEMA, and MODULE\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the SQL-server module being described.
- 2) The values of TABLE\_CATALOG, TABLE\_SCHEMA, and TABLE\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of a table that is referenced in the SQL-server module being described.

## 14.2 Definition Schema

### 14.2.3 MODULE\_COLUMN\_USAGE base table

#### Function

The MODULE\_COLUMN\_USAGE table has one row for each column referenced in an SQL-server module.

#### Definition

```
CREATE TABLE MODULE_COLUMN_USAGE
(
  MODULE_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
  MODULE_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
  MODULE_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_CATALOG      INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_SCHEMA       INFORMATION_SCHEMA.SQL_IDENTIFIER,
  TABLE_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,
  COLUMN_NAME         INFORMATION_SCHEMA.SQL_IDENTIFIER,

  CONSTRAINT MODULE_COLUMN_USAGE_PRIMARY_KEY
    PRIMARY KEY ( MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME,
                 TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME ),

  CONSTRAINT MODULE_COLUMN_USAGE_CHECK_REFERENCES_COLUMNS
    CHECK ( TABLE_CATALOG <> ANY ( SELECT CATALOG_NAME FROM SCHEMATA )
    OR
    ( TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME ) IN
    ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMNS FROM COLUMNS ) ),

  CONSTRAINT MODULE_COLUMN_USAGE_FOREIGN_KEY_MODULES
    FOREIGN KEY ( MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME )
    REFERENCES MODULES
)
```

#### Description

The values of MODULE\_CATALOG, MODULE\_SCHEMA, and MODULE\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the SQL-server module being described.

- 1) The values of TABLE\_CATALOG, TABLE\_SCHEMA, TABLE\_NAME, and COLUMN\_NAME are the catalog name, unqualified schema name, qualified identifier, and identifier respectively, of a column that is referenced in the SQL-server module being described.

## 14.2.4 MODULE\_PRIVILEGES base table

### Function

The MODULE\_PRIVILEGES table has one row for each module privilege descriptor. It effectively contains a representation of the module privilege descriptors.

### Definition

```
CREATE TABLE MODULE_PRIVILEGES
(
  GRANTOR          INFORMATION_SCHEMA.SQL_IDENTIFIER,
  GRANTEE          INFORMATION_SCHEMA.SQL_IDENTIFIER,
  MODULE_CATALOG  INFORMATION_SCHEMA.SQL_IDENTIFIER,
  MODULE_SCHEMA   INFORMATION_SCHEMA.SQL_IDENTIFIER,
  MODULE_NAME     INFORMATION_SCHEMA.SQL_IDENTIFIER,
  PRIVILEGE_TYPE  INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT MODULE_PRIVILEGES_TYPE_CHECK
  CHECK ( PRIVILEGE_TYPE = 'EXECUTE' ),
  IS_GRANTABLE    INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT MODULE_PRIVILEGES_GRANTABLE_NOT_NULL NOT NULL
  CONSTRAINT MODULE_PRIVILEGES_GRANTABLE_CHECK
  CHECK ( IS_GRANTABLE IN ( 'YES', 'NO' ) ),

  CONSTRAINT MODULE_PRIVILEGES_PRIMARY_KEY
  PRIMARY KEY ( GRANTOR, GRANTEE, MODULE_CATALOG, MODULE_SCHEMA,
               MODULE_NAME, PRIVILEGE_TYPE ),

  CONSTRAINT MODULE_PRIVILEGES_FOREIGN_KEY_TABLES
  FOREIGN KEY ( MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME )
  REFERENCES MODULES,

  CONSTRAINT MODULE_PRIVILEGES_GRANTOR_FOREIGN_KEY_USERS
  FOREIGN KEY ( GRANTOR )
  REFERENCES USERS,

  CONSTRAINT MODULE_PRIVILEGES_GRANTEE_FOREIGN_KEY_USERS
  FOREIGN KEY ( GRANTEE )
  REFERENCES USERS
)
```

### Description

- 1) The value of GRANTOR is the <authorization identifier> of the user who granted module privileges, on the SQL-server module identified by MODULE\_CATALOG, MODULE\_SCHEMA, and MODULE\_NAME, to the user identified by the value of GRANTEE for the module privilege being described.
- 2) The value of GRANTEE is the <authorization identifier> of some user, or "PUBLIC" to indicate all users, to whom the module privilege being described is granted.
- 3) The values of MODULE\_CATALOG, MODULE\_SCHEMA, and MODULE\_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the SQL-server module on which the privilege being described has been granted.

**14.2 Definition Schema**

- 4) The values of PRIVILEGE\_TYPE have the following meanings:

EXECUTE	The user has EXECUTE privilege on the SQL-server module identified by MODULE_CATALOG, MODULE_SCHEMA, and MODULE_NAME.
---------	---

- 5) The values of IS\_GRANTABLE have the following meanings:

YES	The privilege being described was granted WITH GRANT OPTION and is thus grantable.
-----	--

NO	The privilege being described was not granted WITH GRANT OPTION and is thus not grantable.
----	--

## 15 Status codes

### 15.1 SQLSTATE

Table 4—SQLSTATE class and subclass values

Category	Condition	Class	Subcondition	Subclass
	<i>All alternatives from part 2 of ISO/IEC 9075</i>			
	<i>All alternatives from part 5 of ISO/IEC 9075</i>			
X	case not found for case statement	20	(no subclass)	000
X	data exception	22	(no subclass)	000
			null value in field reference	006
X	invalid transaction initiation	0B	(no subclass)	000
X	resignal when handler not active	0K	(no subclass)	000
			function executed no return statement	005
			modifying SQL-data not permitted	002
			prohibited SQL-statement attempted	003
			reading SQL-data not permitted	004
X	invalid SQLSTATE value	0H	(no subclass)	000
W	warning	01	(no subclass)	000
			resignal statement with no active exception	00C
X	unhandled user-defined exception	45	(no subclass)	000

• 1 table row deleted.

**ISO/IEC JTC1/SC21 N11138 = DBL:CWB-003**

## 16 Conformance

### 16.1 Claims of conformance

Insert this paragraph Claims of conformance to ISO/IEC 9075-4 shall state:

- 1) A conformance claim to ISO/IEC 9075, in Clause 21, "Conformance".
- 2) Whether or not <SQL-server module definition> is supported.
- 3) The definitions for all elements and actions that ISO/IEC 9075-4 specifies as implementation-defined.

### 16.2 Extensions and options

Insert this paragraph A conforming implementation may provide additional facilities or options not specified by ISO/IEC 9075-4. This may imply an implementation-defined extension of the list of reserved words (<reserved word>) and thereby may prevent proper processing of some programs that otherwise meet the requirements of ISO/IEC 9075.

Insert this paragraph An implementation remains conforming even if it provides user options to process nonconforming SQL language or to process conforming SQL language in a nonconforming manner.

#### 16.2.1 Information Schema requirements

An implementation claiming conformance to ISO/IEC 9075-4 is required to provide an accessible INFORMATION\_SCHEMA comprising the following views defined in Subclause 14.1, "Information Schema", of ISO/IEC 9075-4.

- 1 list element deleted.
  - MODULES
- 2 list elements deleted.
  - MODULE\_PRIVILEGES
- 3 list elements deleted.
- 1 Note deleted.

## 16.2 Extensions and options

### 16.2.2 Schema manipulation requirements

An implementation claiming conformance to ISO/IEC 9075-4 is required to support the following <SQL schema definition statement>s as if they were also <SQL procedure statement>s.

- <grant statement>
- <schema routine>

If conformance is claimed for <SQL-server module definition>, then an implementation claiming conformance to ISO/IEC 9075-4 is required to support <SQL-server module definition> as if it were also an <SQL procedure statement>.

## 16.3 Flagger requirements

Insert this paragraph An SQL/PSM Flagger is an implementation-provided facility that is able to identify SQL/PSM language extensions that may be provided by an SQL/PSM conforming implementation. It is a user-invocable enhancement to the SQL Flagger described in <REFERENCE>(frame\_flagger\FULL), of ISO/IEC 9075-1, and it has the same motivation, purpose, and limitations. An SQL Flagger shall be invocable with the SQL/PSM Flagger enhancement either “on” or “off”. When the SQL/PSM Flagger is turned “off” the SQL Flagger will identify all SQL/PSM language as an SQL extension; when it is turned “on” the SQL Flagger will not flag conforming SQL/PSM language as an SQL extension. The SQL/PSM Flagger has the same “extent of checking” and “level of flagging” requirements as does the SQL Flagger; it does not have any additional “level of flagging” requirements because SQL/PSM is not leveled.

Insert this paragraph SQL/PSM implementations shall provide an SQL/PSM Flagger, as an enhancement to the SQL Flagger, that when turned “on” properly identifies SQL/PSM language extensions under the same “extent of checking” and “level of flagging” options as specified for the SQL Flagger.

## Annex A (informative)

### Implementation-defined elements

Insert this paragraph This Annex references those features that are identified in the body of ISO/IEC 9075-4 as implementation-defined.

Insert this paragraph The term *implementation-defined* is used to identify characteristics that may differ between implementations, but that shall be defined for each particular implementation.

- 1) Subclause 8.3, “<sqlstate value>”:
  - a) The implicit or explicit character set of the <character string literal> contained in <sqlstate value> shall be the implementation-defined character set in which SQLSTATE parameter values are returned.
  - b) The value of the <character string literal> contained in <sqlstate value> may be composed of a standard SQLSTATE Class value for which an implementation-defined Subclass value is permitted and three characters with the form of an implementation-defined Subclass value.
  - c) The value of the <character string literal> contained in <sqlstate value> may be composed of five characters of which the first two have the form of an implementation-defined Class value.
  
- 2) Subclause 9.17, “<SQL-invoked routine>”:
  - 2 list elements deleted.
    - a) If the <SQL-invoked routine> is an external routine, then the method and time of binding of that external routine to the schema or <SQL-server module definition> containing the <SQL-invoked routine> is implementation-defined.
  - 7 list elements deleted.
  
- 3) Subclause 10.2, “<SQL procedure statement>”: The Ada package SQL\_STANDARD requires a number of implementation-defined character types and integer values.
  
- 4) Table 2, “<identifier>s for use with <get diagnostics statement>”: The length of the diagnostics area field EXCEPTION\_IDENTIFIER is implementation-defined, but not less than 128 characters.
  
- 5) Subclause 16.2, “Extensions and options”: The list of <reserved words> may be augmented by an implementation-defined extension if an implementation provides additional facilities or options.

**ISO/IEC JTC1/SC21 N11138 = DBL:CWB-003**

## Annex B (informative)

### Implementation-dependent elements

Insert this paragraph This Annex references those places where ISO/IEC 9075-4 states explicitly that the actions of a conforming implementation are implementation-dependent.

Insert this paragraph The term *implementation-dependent* is used to identify characteristics that may differ between implementations, but that are not necessarily specified for any particular implementation.

- 1) Subclause 6.2, “<item reference>”: The implicit qualifier for an unqualified <item reference> where there is more than one possible qualifier with the most local scope is implementation-dependent.
- 2) Subclause 6.4, “<datetime value function>”: The time of evaluation of the <datetime value function> during the execution of an SQL-statement is implementation-dependent.
- 3) Subclause 12.1, “<compound statement>”:
  - a) The implicit <beginning label> of a <compound statement> with no explicit <beginning label> is implementation-dependent.
  - b) The variables and temporary tables specified in the <local declaration list> of *CS* are created in an implementation-dependent order.
- 4) Subclause 12.4, “<SQL variable declaration>”: When a variable associated with an <SQL variable declaration> is created and the <SQL variable declaration> does not contain a <default clause>, then the initial value of the variable is implementation-dependent.
- 5) Subclause 12.13, “<for statement>”: The <cursor name> used in the transformation of a <for statement> into a <while statement> is implementation-dependent, as are the <condition name> and the <SQL variable name> used in the <while statement> for getting diagnostics information.



**Annex C**  
(informative)

**Deprecated features**

It is intended that the following features will be removed at a later date from a revised version of ISO/IEC 9075-4:

No additional deprecated items.

**ISO/IEC JTC1/SC21 N11138 = DBL:CWB-003**

## Annex D (informative)

### Incompatibilities with ISO/IEC 9075:1992

ISO/IEC 9075-4:199 $x$  introduces some incompatibilities with the earlier version of Database Language SQL as specified in ISO/IEC 9075:1992. Unless specified in this Annex, features and capabilities of Database Language SQL are compatible with the earlier version of ISO/IEC 9075.

1) Augment list element 5) A number of additional <reserved word>s have been added to the language. These <reserved word>s are:

- 1 list element deleted.

- CONDITION
- DO
- ELSEIF
- EXIT

- 1 list element deleted.

- HANDLER
- IF
- ITERATE
- LEAVE
- LOOP
- REDO
- REPEAT
- RESIGNAL
- SIGNAL

- 2 list elements deleted.

- UNDO
- UNTIL

— WHILE

- 2) Insert after list element 5) The definition of “possibly nullable” with regard to <routine invocation> has been tightened. If all subject routines of a <routine invocation> specify PARAMETER STYLE GENERAL, then the <routine invocation> is known not nullable. In ISO/IEC 9075-4:1996, all <routine invocation>s were regarded as possibly nullable.

## Annex E (informative)

### SQL Feature Taxonomy

This Annex describes a taxonomy of features of the SQL language.

Table 5, “SQL/PSM feature taxonomy”, contains a taxonomy of the features of the SQL language that are specified in ISO/IEC 9075-4. In this table, the first column contains a counter that may be used to quickly locate rows of the table; these values otherwise have no use and are not stable — that is, they are subject to change in future editions of or even Technical Corrigenda to ISO/IEC 9075 without notice.

The second column, “Feature ID”, specifies the formal identification of each feature and each subfeature contained in the table. The Feature ID is stable and can be depended on to remain constant. A Feature ID value comprises either a letter and three digits or a letter, three digits, a hyphen, and one or two additional digits. Feature ID values containing a hyphen and additional digits indicate “subfeatures” that help to define complete features, which are in turn indicated by Feature ID values without a hyphen. Only entire features are used to specify the contents of Core SQL and various packages.

The “Feature Description” column contains a brief description of the feature or subfeature associated with the Feature ID value.

**\*\*Editor’s Note\*\***

The Feature Description column must be enhanced during the FCD ballot period to identify specific BNF non-terminal symbols and specific Syntax and/or General Rules.

The final column, named “Core SQL?”, provides the definition of the minimal conformance possibility for ISO/IEC 9075, called Core SQL. Features that are included in the definition of Core SQL contain the value “YES” in this column; their subfeatures contain the value “(yes)” for consistency. Features and subfeatures that are not part of Core SQL contain a dash (“—”) in this column.

**Table 5—SQL/PSM feature taxonomy**

	<b>Feature ID</b>	<b>Feature Description</b>	<b>Core SQL?</b>
1	P01	Stored modules (<SQL-server module definition>)	—
2	P01-1	<SQL-server module definition>	—
3	P01-2	<drop module statement>	—
4	P02	Computational completeness	—

Table 5—SQL/PSM feature taxonomy (Cont.)

	Feature ID	Feature Description	Core SQL?
5	P02-1	<compound statement>	—
6	P02-2	<handler declaration>	—
7	P02-3	<condition declaration>	—
8	P02-4	<SQL variable declaration>	—
9	P02-5	<assignment statement>	—
10	P02-6	<case statement>	—
11	P02-7	<if statement>	—
12	P02-8	<iterate statement>	—
13	P02-9	<leave statement>	—
14	P02-10	<loop statement>	—
15	P02-11	<repeat statement>	—
16	P02-12	<while statement>	—
17	P02-13	<for statement>	—
18	P02-14	<signal statement>	—
19	P02-15	<resignal statement>	—
20	P02-16	<control statement>s as the SQL-statement of an externally-invoked procedure	—
21	P03	Information Schema views	—
22	P03-1	MODULES view	—
23	P03-2	MODULE_TABLE_USAGE view	—
24	P03-3	MODULE_COLUMN_USAGE view	—
25	P03-4	MODULE_PRIVILEGES view	—

# Index

Index entries appearing in **boldface** indicate the page where the word, phrase, or BNF nonterminal was defined; index entries appearing in *italics* indicate a page where the BNF nonterminal was used in a Format; and index entries appearing in roman type indicate a page where the word, phrase, or BNF nonterminal was used in a heading, Function, Syntax Rule, Access Rule, General Rule, Leveling Rule, Table, or other descriptive text.

## — A —

abandoned • 62, 63, 64, 65  
abstract data type • 16, 41, 90  
<action> • 21  
activated • 17, 18, 86, 109  
active • 17, 18, 83, 95, 108, 121  
active completion condition • 17  
active condition • 17  
active exception condition • 17, 18  
Ada • 3, 125  
ALTER • 106, 111, 115, 116  
<alter column definition> • 41  
<alter domain statement> • 41  
<alter module statement> • 106  
AND • 111, 112, 113, 114  
applicable • 35  
applicable privileges • 35  
ARE • 54  
AS • 29, 101, 102, 111, 112, 113, 114  
assertion • 9, 52, 53, 57, 62  
<assertion definition> • 52  
assignment • 1, 10, 19, 69, 74, 90, 91, 106, 134  
<assignment source> • **90**, 91  
<assignment statement> • 19, 69, **90**, 106, 134  
<assignment target> • **90**, 91  
AT • 23, 100, 131  
atomic • 8, 21, 83  
ATOMIC • 21, 81, 82, 83, 98, 99, 100  
atomic execution context • 21, 83  
atomic SQL-statement • 21  
attribute • 41  
<attribute definition> • 41  
ATTRIBUTES • 6, 105, 106, 121  
AUTHORIZATION • 111, 115, 116  
authorization identifier • 15, 16, 21, 55, 57, 61, 116, 119  
<authorization identifier> • 15, 16, 21, 55, 57, 61, 119

## — B —

based • 3, 92, 94  
base table • 16, 80, 115, 117, 118, 119  
BEGIN • 81, 98, 99, 100, 102, 106  
<beginning label> • 28, **81**, 82, 96, 97, 98, 99, 100, 101, 127  
BOOLEAN • 102

## — C —

candidate routine • 35  
CASE • 92  
case not found for case statement • 93, 121  
<case statement> • 19, 69, **92**, 93, 106, 134  
<case statement else clause> • **92**, 93  
CAST • 33  
catalog • 41, 54, 111, 112, 113, 114, 116, 117, 118, 119  
<catalog name> • 41, 54  
CATALOG\_NAME • 106, 111, 112, 113, 114, 117, 118  
character • 12, 15, 16, 25, 38, 41, 42, 49, 51, 54, 55, 62, 63, 64, 65, 105, 106, 116, 125, 127  
character repertoire • 15, 25  
character set • 15, 16, 25, 38, 49, 55, 62, 63, 64, 65, 116, 125  
character set descriptor • 16  
<character set name> • 49  
<character set specification> • 25, 54, 55  
<character string literal> • 15, 38, 125  
character type • 125  
CHARACTER\_DATA • 115, 119  
CHARACTER\_SETS • 115  
CHARACTER\_SET\_CATALOG • 111, 115, 116  
CHARACTER\_SET\_NAME • 111, 115, 116  
CHARACTER\_SET\_SCHEMA • 111, 115, 116  
CHECK • 117, 118, 119  
CLASS • 68, 106  
CLASS\_ORIGIN • 106  
CLOSE • 83, 97, 102  
<close statement> • 73, 102  
collating sequence • 50  
collation • 9, 16, 50, 62, 63, 64, 65  
collation descriptor • 16

## ISO/IEC JTC1/SC21 N11138 = DBL:CWB-003

<collation name> • 50  
collection • 28  
<colon> • 81, 98, 99, 100, 101  
column • 11, 28, 29, 33, 41, 43, 48, 62, 63, 64, 65, 90, 101, 113, 116, 118  
<column definition> • 41  
column list • 43, 64, 65  
<column name> • 28, 43, 48, 101  
column reference • 29, 33, 63, 64, 90, 116, 118  
<column reference> • 63, 64, 116  
COLUMNS • 118  
COLUMN\_NAME • 106, 113, 118  
<comma> • 85, 89  
COMMAND\_FUNCTION • 106, 107, 109  
<commit statement> • 82  
common column name • 28, 29  
comparable • 92  
compatible • 131  
component • 16  
<compound statement> • 17, 18, 19, 21, 29, 69, 81, 82, 83, 86, 97, 98, 99, 100, 101, 106, 127, 134  
CONDITION • 23, 88, 102, 131  
<condition declaration> • 18, 19, 81, 82, 84, 85, 88, 107, 108, 134  
<condition information item name> • 105  
<condition name> • 18, 25, 26, 82, 84, 85, 86, 88, 101, 106, 107, 108, 127  
conditions • 1, 17, 18, 83, 85, 86, 109  
<condition value> • 85, 86  
<condition value list> • 85, 86  
CONDITION\_IDENTIFIER • 105, 106, 107, 108, 109  
conformance • 12, 123, 124, 125  
Conformance • 123  
considered to be defined • 84, 88  
<constraint name> • 44, 53  
CONSTRAINT\_CATALOG • 106  
CONSTRAINT\_NAME • 106  
CONSTRAINT\_SCHEMA • 106  
contain • 1, 3, 6, 15, 16, 17, 18, 21, 25, 26, 28, 29, 31, 33, 35, 38, 40, 41, 42, 43, 44, 45, 47, 49, 50, 51, 53, 54, 55, 57, 58, 61, 62, 63, 64, 65, 69, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 82, 83, 84, 85, 86, 87, 88, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 107, 108, 109, 116, 119, 125, 127  
contained in • 6, 15, 17, 18, 21, 25, 26, 29, 31, 35, 38, 40, 54, 55, 57, 58, 61, 62, 63, 64, 65, 69, 79, 80, 82, 83, 85, 87, 90, 91, 93, 94, 95, 96, 97, 98, 99, 100, 101, 107, 108, 116, 125  
containing • 6, 18, 41, 54, 55, 58, 86, 101, 125  
contains • 6, 15, 16, 28, 29, 31, 33, 40, 41, 42, 43, 44, 45, 47, 49, 50, 51, 53, 54, 58, 69, 74, 79, 83, 85, 90, 91, 93, 97, 107, 108, 109, 119  
CONTINUE • 18, 85, 86, 102  
<correlation name> • 28, 29  
COUNT • 33  
CREATE • 54, 106, 111, 112, 113, 114, 115, 116, 117, 118, 119  
created by • 16  
current SQL-session • 25, 26

CURRENT\_PATH • 41, 42  
CURRENT\_TIMESTAMP • 116  
CURRENT\_USER • 41, 42, 69, 111, 112, 113, 114  
<cursor name> • 71, 72, 73, 75, 77, 82, 101, 127  
<cursor sensitivity> • 101  
<cursor specification> • 63, 64, 72, 101  
CURSOR\_NAME • 106

### — D —

Data • 1, 3, 12, 71, 105, 131  
database • 1  
data exception • 91, 121  
data type • 16, 33, 41, 42, 90, 92, 101  
<data type> • 33, 41, 89  
data type descriptor • 16  
date • 1, 8, 10, 26, 28, 29, 31, 35, 42, 63, 64, 65, 69, 77, 78, 83, 84, 87, 97, 99, 100, 127, 129  
<datetime value function> • 31, 42, 69, 127  
DECLARE • 58, 85, 88, 89, 102, 106  
<declare cursor> • 17, 81, 82  
declared local name • 82  
declared local temporary table • 16, 79  
DEFAULT • 102, 111, 115, 116  
<default clause> • 41, 89, 127  
<default option> • 41, 42, 69  
<default schema name> • 15, 26, 54, 55  
DEFAULT\_CHARACTER\_SET\_CATALOG • 111, 115, 116  
DEFAULT\_CHARACTER\_SET\_NAME • 111, 115, 116  
DEFAULT\_CHARACTER\_SET\_SCHEMA • 111, 115, 116  
Definition Schema • 111, 115  
DEFINITION\_SCHEMA • 111, 112, 113, 114  
DELETE • 64, 65  
<delete statement: positioned> • 64, 65, 75  
<delete statement: searched> • 63, 64, 65, 76  
depend • 16, 29, 31, 74, 79, 82, 83, 101, 112, 113, 127  
dependent • 16, 29, 31, 74, 79, 82, 83, 101, 112, 113, 127  
deprecated • 129  
<derived column> • 29  
Description • 116, 117, 118, 119, 133  
descriptor • 15, 16, 21, 40, 41, 43, 44, 45, 47, 48, 49, 50, 51, 53, 54, 55, 57, 60, 61, 62, 63, 64, 65, 69, 119  
determines • 19  
deterministic • 6, 92  
diagnostics area • 17, 105, 106, 107, 108, 109, 125  
directly • 20, 25, 26, 29, 94  
directly contain • 94  
direct result of executing an <SQL control statement> • 6  
direct result of executing an SQL-statement • 6  
<direct SQL statement> • 18, 25, 26  
DO • 23, 99, 101, 102, 131  
domain • 16, 33, 41, 48, 62, 63, 64, 65, 69  
domain constraint • 63  
domain constraint descriptor • 63  
<domain definition> • 41

domain descriptor • 69  
 <domain name> • 33, 69  
 <double greater than operator> • 90  
 DROP • 40, 43, 44, 45, 47, 48, 49, 50, 51, 53, 57, 65, 106  
 <drop assertion statement> • 53  
 <drop behavior> • 57  
 <drop character set statement> • 49  
 <drop collation statement> • 50  
 <drop column definition> • 43  
 <drop domain statement> • 48  
 <drop module statement> • 15, 19, 40, 43, 44, 45, 47, 48, 49, 50, 51, 53, 57, 65, 69, 106, 133  
 <drop routine statement> • 40, 43, 44, 45, 47, 49, 50, 51, 53, 60  
 <drop schema statement> • 15, 40  
 <drop table constraint definition> • 44  
 <drop table statement> • 45  
 <drop translation statement> • 51  
 <drop view statement> • 47  
 DYNAMIC • 106, 107, 109  
 DYNAMIC\_FUNCTION • 106, 107, 109

— E —

\*\*Editor's Note\*\* • 41, 86, 133  
 effective • 16, 17, 31, 40, 43, 44, 45, 47, 48, 49, 50, 51, 53, 55, 57, 65, 79, 80, 83, 86, 87, 89, 97, 107, 119  
 effectively • 16, 17, 31, 40, 43, 44, 45, 47, 48, 49, 50, 51, 53, 55, 57, 65, 79, 80, 83, 86, 87, 97, 107, 119  
 element • 16, 23, 35, 39, 74, 123, 125, 127, 131, 132  
 ELSE • 94  
 ELSEIF • 23, 94, 131  
 embedded • 20, 91  
 <embedded SQL host program> • 20  
 <embedded variable name> • 91  
 END • 54, 81, 92, 94, 98, 99, 100, 101, 102  
 <ending label> • 81, 82, 98, 99, 100, 101  
 environment • 12  
 <equals operator> • 90  
 equivalent • 28, 54, 55, 57, 58, 71, 72, 73, 75, 77, 79, 80, 82, 90, 92, 94, 98, 99, 100, 101, 102  
 exception • 5, 7, 17, 18, 19, 21, 26, 83, 85, 86, 91, 93, 95, 98, 99, 100, 105, 106, 107, 108, 109, 121  
 EXCEPTION • 17, 68, 85, 86, 125  
 executable routine • 35  
 Execute • 18, 101  
 EXECUTE • 21, 35, 37, 55, 57, 61, 62, 63, 64, 119, 120  
 <execute immediate statement> • 25, 26  
 execute privilege descriptor • 21  
 executing statement • 83, 93, 95, 98, 99, 100  
 EXIT • 18, 23, 85, 87, 131  
 explicit • 6, 7, 17, 18, 38, 43, 54, 55, 101, 116, 125, 127  
 exposed • 28  
 <externally-invoked procedure> • 6, 18, 20, 68, 79  
 external routine • 125

— F —

FALSE • 102  
 FETCH • 102  
 <fetch statement> • 72, 102  
 <field name> • 90, 91  
 FOR • 85, 88, 101, 102  
 FOREIGN • 115, 117, 118, 119  
 <for loop variable name> • 101  
 <for statement> • 19, 20, 69, 101, 102, 106, 127, 134  
 FOUND • 17, 68, 85, 86, 101, 102  
 FROM • 38, 57, 102, 111, 112, 113, 114, 117, 118  
 function • 6, 7, 19, 31, 33, 42, 58, 69, 90, 121, 127  
 FUNCTION • 106, 107, 109  
 function executed no return statement • 121

— G —

GENERAL • 132  
 general <handler declaration> • 17, 86  
 generally contain • 31, 62, 63, 64, 65, 69, 76, 78, 92  
 <general value specification> • 27  
 <get diagnostics statement> • 105  
 GRANTEE • 111, 114, 119  
 <grantee> • 61  
 GRANTOR • 114, 119  
 <grant statement> • 37, 61, 124

— H —

handle • 1, 5, 10, 17, 18, 19, 21, 81, 82, 83, 85, 86, 87, 93, 95, 97, 98, 99, 100, 106, 108, 109, 121, 134  
 HANDLER • 23, 68, 85, 102, 106, 131  
 <handler action> • 18, 19, 85, 86  
 <handler declaration> • 17, 18, 19, 81, 82, 83, 85, 86, 106, 134  
 <handler type> • 85  
 host parameter • 27, 90, 91  
 <host parameter specification> • 91

— I —

<identifier> • 15, 25, 28, 81, 101, 105, 107, 108  
 IF • 23, 94, 131  
 <if statement> • 19, 69, 94, 95, 106, 134  
 <if statement else clause> • 94, 95  
 <if statement elseif clause> • 94  
 <if statement then clause> • 94, 95  
 immediate • 25, 26, 55, 63, 64, 65, 82, 83, 90, 93, 95, 96, 107, 108  
 immediately contain • 55, 63, 64, 65, 82, 83, 90, 95, 107, 108  
 implementation • 6, 12, 16, 17, 29, 31, 38, 54, 74, 79, 82, 83, 101, 123, 124, 125, 127  
 implementation-defined • 6, 17, 38, 54, 123, 125  
 implementation-dependent • 16, 29, 31, 74, 79, 82, 83, 101, 127  
 implicit • 15, 18, 19, 26, 28, 29, 38, 54, 55, 58, 82, 101, 116, 125, 127  
 Implicit • 18  
 IN • 111, 114, 117, 118

## ISO/IEC JTC1/SC21 N11138 = DBL:CWB-003

include • 1, 15, 16, 21, 28, 31, 33, 35, 40, 43, 44, 45, 47, 48, 49, 50, 51, 53, 54, 55, 57, 59, 61, 62, 63, 64, 65, 76, 78, 92, 107, 108  
indicator • 33  
indicator parameter • 33  
<indicator parameter> • 33  
inessential • 6  
Information Schema • 1, 111, 123, 134  
INFORMATION\_SCHEMA • 111, 112, 113, 114, 115, 117, 118, 119, 123  
innermost • 28, 29, 71, 72, 73, 75, 77, 85, 107, 108  
INSERT • 63, 64  
<insert column list> • 64, 65  
<insert statement> • 63, 64, 65  
integrity constraint • 52  
interface • 3  
intervening • 25, 26, 31, 35, 50, 51, 55, 58, 79, 80, 83, 93, 95, 97, 102  
INTO • 102  
invalid transaction initiation • 21, 121  
IS\_GRANTABLE • 114, 119, 120  
<item name> • 28  
<item qualifier> • 28  
<item reference> • 28, 30, 127  
ITERATE • 23, 96, 131  
<iterate statement> • 19, 20, 69, 96, 134  
<iterative routine> • 58

### — J —

JOIN • 112, 113  
<joined table> • 28, 29

### — K —

KEY • 105, 115, 117, 118, 119  
known not nullable • 132

### — L —

LANGUAGE • 58  
<language clause> • 58  
LAST • 111, 115, 116  
LEAVE • 18, 23, 97, 106, 131  
<leave statement> • 19, 20, 21, 69, 97, 98, 99, 100, 102, 106, 134  
<left paren> • 12  
LENGTH • 106  
<literal> • 41, 42  
<local cursor declaration list> • 81, 82, 83, 97  
<local declaration> • 81, 82  
<local declaration list> • 28, 29, 81, 82, 83, 84, 97, 127  
<local handler declaration list> • 81, 82, 83, 97  
local temporary table • 15, 16, 55, 79  
LOOP • 23, 98, 106, 131  
<loop statement> • 19, 20, 69, 98, 106, 134

### — M —

MESSAGE\_LENGTH • 106  
MESSAGE\_OCTET\_LENGTH • 106  
MESSAGE\_TEXT • 106  
<modified field reference> • 90, 91

<modified field target> • 90, 91  
modifying SQL-data • 121  
modifying SQL-data not permitted • 121  
module • 1, 7, 11, 15, 16, 19, 20, 21, 25, 26, 35, 37, 39, 40, 43, 44, 45, 47, 48, 49, 50, 51, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 67, 69, 79, 80, 85, 86, 101, 106, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 123, 124, 125, 133  
MODULE • 37, 40, 43, 44, 45, 47, 48, 49, 50, 51, 53, 54, 57, 65, 80, 106, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 123, 134  
<module> • 26, 55  
<module authorization clause> • 116  
<module authorization identifier> • 116  
<module character set specification> • 25  
module descriptor • 15, 16, 40, 43, 45, 47, 48, 49, 50, 54, 55, 57, 60, 64, 65  
<module function> • 58  
<module iterative routine> • 58  
<module name> • 37  
<module procedure> • 58  
<module routine> • 16, 58  
MODULES • 111, 115, 117, 118, 119, 123, 134  
MODULE\_AUTHORIZATION • 111, 115, 116  
MODULE\_CATALOG • 111, 112, 113, 114, 115, 116, 117, 118, 119, 120  
MODULE\_DEFINITION • 111, 115, 116  
MODULE\_NAME • 111, 112, 114, 115, 116, 117, 118, 119, 120  
MODULE\_PRIVILEGES • 111, 114, 119, 123, 134  
MODULE\_SCHEMA • 111, 112, 114, 115, 116, 117, 118, 119, 120  
MORE • 107, 109  
most appropriate handler • 18, 86, 109  
<mutated target> • 90  
<mutator reference> • 90

### — N —

Name • 25, 64  
NAMES • 54  
non-deterministic • 92  
<non-reserved word> • 23  
no subclass • 121  
NOT • 81, 82, 85, 86, 98, 99, 100, 102, 115, 119  
not permitted • 121  
null • 33, 41, 90, 91, 116, 121, 132  
NULL • 33, 41, 115, 119  
<null specification> • 90  
null value • 41, 91, 116, 121

### — O —

object • 7, 12, 13, 15, 16, 37, 61, 64, 65  
<object column> • 64, 65  
object identifier • 12  
<object name> • 37  
OCTET\_LENGTH • 106  
ON • 57, 112, 113  
OPEN • 102  
<open statement> • 71, 102

operator • 90  
 OPTION • 61, 62, 63, 64, 120  
 OR • 114, 117, 118  
 outermost • 101  
 outer reference • 29  
 owned by • 16, 61, 112, 113

## — P —

parameter • 1, 7, 16, 17, 27, 28, 29, 33, 38, 50, 90, 91, 101, 125  
 PARAMETER\_NAME • 105  
 Part 1 • 3  
 Part 2 • 3  
 Part 3 • 3  
 Part 4 • 5, 12, 105  
 <Part 4 yes> • 12  
 Part 5 • 3, 33  
 Part 6 • 3  
 Part 7 • 3  
 part of • 6, 7, 8, 9, 10, 11, 12  
 <path specification> • 54  
 period • 41  
 <period> • 41  
 permitted • 38, 121, 125  
 persist • 1, 15, 16, 80  
 persistent • 1, 15, 16, 80  
 persistent base table • 80  
 possible qualifiers • 28  
 possibly candidate routine • 35  
 possibly modifies SQL-data • 76, 78, 92  
 possibly modify SQL-data • 6  
 possibly nullable • 33, 132  
 <preparable statement> • 25, 26  
 <prepare statement> • 25, 26  
 PRIMARY • 115, 117, 118, 119  
 privilege • 8, 11, 15, 16, 21, 35, 37, 55, 61, 62, 63, 64, 65, 114, 119, 120  
 privilege descriptor • 16, 21, 55, 61, 62, 63, 64, 119  
 PRIVILEGES • 111, 114, 119, 123, 134  
 <privileges> • 37, 61  
 PRIVILEGE\_TYPE • 114, 119, 120  
 procedure • 1, 6, 17, 18, 20, 29, 31, 55, 58, 61, 68, 69, 79, 81, 83, 85, 87, 93, 95, 97, 109, 124, 134  
 prohibited SQL-statement attempted • 121  
 PUBLIC • 111, 114, 119

## — Q —

<qualified name> • 15, 16, 25, 26  
 <quantified predicate> • 29  
 query • 28, 29, 33, 43, 57, 62, 63, 64, 101  
 <query expression> • 43, 57, 62, 63, 64, 101  
 <query specification> • 33

## — R —

reading SQL-data • 121  
 reading SQL-data not permitted • 121  
 recursive • 90, 94  
 REDO • 18, 23, 83, 131  
 referenced columns • 62, 63

REFERENCES • 62, 63, 115, 117, 118, 119  
 REPEAT • 23, 100, 131  
 <repeat statement> • 19, 20, 69, 100, 106, 134  
 repertoire • 15, 25  
 representation • 116, 119  
 requires • 125  
 reserved • 23, 123, 125, 131  
 <reserved word> • 23, 123, 131  
 RESIGNAL • 23, 68, 83, 86, 87, 106, 107, 108, 109, 131  
 <resignal statement> • 17, 18, 19, 69, 83, 86, 87, 106, 107, 108, 109, 134  
 resignal when handler not active • 108, 121  
 RESTRICT • 40, 43, 45, 47, 48, 57  
 RETURNED\_SQLSTATE • 106, 107, 109  
 REVOKE • 57  
 <revoke statement> • 37, 57, 62  
 <right paren> • 12  
 role • 21  
 <rollback statement> • 82  
 ROUTINE • 40, 43, 44, 45, 47, 49, 50, 51, 53, 105, 113  
 <routine body> • 55, 61, 63, 64, 65  
 <routine invocation> • 31, 33, 35, 55, 57, 61, 62, 63, 64, 76, 78, 92, 132  
 <routine name> • 28, 55  
 ROUTINE\_CATALOG • 105, 113  
 ROUTINE\_NAME • 105, 113  
 ROUTINE\_SCHEMA • 105, 113  
 row • 29, 63, 64, 72, 74, 75, 76, 77, 78, 101, 105, 115, 116, 117, 118, 119, 121  
 ROW • 102  
 <row value constructor> • 74  
 <row value constructor element> • 74

## — S —

<savepoint clause> • 82  
 schema • 1, 5, 7, 9, 12, 15, 16, 17, 18, 19, 21, 25, 26, 35, 37, 39, 40, 41, 50, 51, 54, 55, 57, 58, 59, 64, 69, 79, 80, 82, 87, 97, 98, 99, 100, 102, 116, 117, 118, 119, 123, 124, 125  
 SCHEMA • 54, 105, 106, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 123  
 <schema character set specification> • 25, 55  
 <schema definition> • 25, 26, 35, 39, 50, 51, 54, 55, 58  
 <schema element> • 39  
 schema-level routine • 16, 21, 37, 54, 59, 64  
 <schema name> • 16, 25, 26, 40, 41, 54, 55, 57, 58, 79  
 <schema name list> • 54, 55  
 <schema qualified name> • 25  
 schema routine • 124  
 <schema routine> • 124  
 SCHEMATA • 111, 112, 113, 115, 117, 118  
 SCHEMA\_NAME • 106, 111, 112, 113, 115, 116  
 SCHEMA\_OWNER • 111, 112, 113  
 scope • 1, 6, 17, 18, 21, 28, 29, 71, 72, 73, 75, 77, 79, 82, 85, 86, 88, 96, 98, 99, 100, 107, 108, 127

## ISO/IEC JTC1/SC21 N11138 = DBL:CWB-003

search condition • 43, 57, 62, 63, 64, 76, 78, 92, 93, 94, 95, 99, 100  
<search condition> • 43, 57, 62, 63, 64, 76, 78, 92, 93, 94, 95, 99, 100  
<searched case statement> • **92**  
<searched case statement when clause> • **92**, 93  
SELECT • 62, 63, 64, 102, 111, 112, 113, 114, 117, 118  
<select list> • 63, 64, 74  
<select statement: single row> • 63, 64, 74  
<select target list> • 74  
<semicolon> • 54, 81  
sensitivity • 101  
<separator> • 23  
sequence • 21, 50  
SESSION • 41, 42, 69  
SESSION\_USER • 41, 42, 69  
SET • 89, 90, 102  
set function • 33  
<set function specification> • 33  
SETS • 115  
shall • 17, 25, 28, 29, 30, 37, 38, 40, 43, 45, 47, 48, 49, 54, 55, 57, 58, 69, 71, 72, 73, 74, 75, 76, 77, 78, 79, 82, 83, 85, 88, 90, 91, 92, 96, 97, 98, 99, 100, 101, 102, 107, 108, 123, 124, 125  
SIGNAL • 23, 107, 131  
<signal statement> • 17, 18, 19, 69, 106, **107**, 134  
<signal value> • **107**, 108, 109  
significant • 6  
<simple case operand 1> • **92**  
<simple case operand 2> • **92**  
<simple case statement> • **92**  
<simple case statement when clause> • **92**  
<simple target specification> • **27**  
<simple value specification> • **27**  
simply contain • 63, 64, 82, 83, 86, 90, 91, 93, 95, 96  
simply contained in • 63, 64, 82, 90, 91, 93, 95, 96  
simply containing • 86  
simply contains • 83, 91, 93  
source • 63, 64, 90, 91  
<space> • 42  
SPECIFIC • 40, 43, 44, 45, 47, 49, 50, 51, 53, 105  
specific <handler declaration> • 17, 86  
specific name • 40, 43, 44, 45, 47, 49, 50, 51, 53  
<specific name> • 40, 43, 44, 45, 47, 49, 50, 51, 53  
SPECIFIC\_NAME • 105  
specified by • 15, 41, 54, 55, 71, 72, 73, 75, 77, 85, 86, 88, 101, 123  
specifies • 1, 18, 33, 41, 42, 69, 82, 83, 85, 86, 87, 92, 102, 123  
specify • 17, 18, 21, 37, 52, 82, 132  
SQL • 5, 33, 58  
SQL-agent • 68  
SQL-client • 16, 18, 20, 25, 67, 79, 80, 101  
SQL-client module • 16, 20, 25, 67, 79, 80, 101  
<SQL-client module definition> • 25, 79, 80, 101  
<SQL connection statement> • 69  
<SQL control statement> • 6, **69**, 83, 93, 95, 101  
SQL-data • 5, 6, 18, 76, 78, 87, 92, 121  
<SQL diagnostics statement> • **69**  
SQLEXCEPTION • 17, 85, 86  
SQL-implementation • 12  
SQL-invoked function • 6, 58  
<SQL-invoked function> • 58  
SQL-invoked procedure • 58  
<SQL-invoked procedure> • 58  
SQL-invoked routine • 1, 5, 6, 15, 16, 21, 29, 33, 35, 40, 43, 44, 45, 47, 49, 50, 51, 53, 54, 55, 57, 58, 59, 60, 61, 62, 63, 64, 65, 69, 76, 78, 79, 91, 92, 101, 102, 125  
<SQL-invoked routine> • 1, 6, 15, 54, 55, **58**, 69, 79, 101, 102, 125  
<SQL language character> • 25  
SQL parameter • 27, 28, 29, 33, 50, 90, 91, 101  
<SQL parameter declaration> • 50  
SQL parameter name • 91, 101  
<SQL parameter name> • 91, 101  
SQL parameter reference • 29  
SQL-path • 1, 35, 55  
<SQL procedure statement> • 17, 18, 20, 29, 31, 55, 61, 69, 81, 83, 85, 87, 93, 95, 97, 109, 124  
SQL routine • 5, 33, 40, 43, 44, 45, 47, 49, 50, 51, 53, 57, 63, 64  
<SQL routine body> • 40, 43, 44, 45, 47, 49, 50, 51, 53, 57, 63, 64  
SQL-schema • 16, 19, 97  
<SQL schema definition statement> • **69**, 124  
<SQL schema manipulation statement> • **69**  
<SQL schema statement> • 17, 35, 69, 80, 82, 98, 99, 100, 102  
SQL-server • 1, 15, 16, 18, 19, 21, 25, 26, 35, 37, 39, 40, 43, 44, 45, 47, 48, 49, 50, 51, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 69, 79, 80, 101, 106, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 123, 124, 125, 133  
SQL-server module • 1, 15, 16, 19, 21, 25, 26, 35, 37, 39, 40, 43, 44, 45, 47, 48, 49, 50, 51, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 69, 79, 80, 101, 106, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 123, 124, 125, 133  
SQL-server module authorization identifier • 15, 55  
<SQL-server module character set specification> • 15, 25, **54**, 55, 116  
<SQL-server module contents> • **54**  
<SQL-server module definition> • 15, 16, 19, 25, 26, 35, 39, **54**, 55, 56, 58, 69, 79, 80, 101, 106, 116, 123, 124, 125, 133  
<SQL-server module name> • 15, 16, **25**, 26, 40, 43, 44, 45, 47, 48, 49, 50, 51, 53, 54, 55, 57, 65  
<SQL-server module path specification> • 15, **54**, 55, 116  
<SQL-server module schema clause> • 15, **54**, 55, 116  
SQL-session • 16, 25, 26, 79  
SQLSTATE • 17, 18, 26, 38, 68, 85, 86, 88, 102, 106, 107, 108, 109, 121, 125  
SQL-statement • 5, 6, 15, 18, 19, 20, 21, 69, 83, 86, 87, 89, 96, 106, 109, 121, 127, 134

<SQL statement list> • 81, 82, 83, 87, 92, 93, 94, 95, 96, 98, 99, 100, 101  
 <sqlstate value> • 38, 85, 86, 88, 107, 108, 125  
 SQL-transaction • 21  
 <SQL variable declaration> • 19, 41, 69, 81, 82, 89, 106, 127, 134  
 <SQL variable name> • 25, 26, 28, 46, 52, 74, 82, 89, 91, 101, 127  
 <SQL variable name list> • 89  
 SQL variable reference • 29, 30  
 <SQL variable reference> • 27, 30  
 SQLWARNING • 17, 85, 86  
 SQL\_IDENTIFIER • 115, 117, 118, 119  
 state • 1, 5, 6, 7, 8, 10, 11, 12, 15, 17, 18, 19, 20, 21, 25, 26, 29, 31, 35, 37, 38, 40, 41, 43, 44, 45, 47, 48, 49, 50, 51, 53, 55, 57, 60, 61, 62, 63, 64, 65, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 80, 81, 82, 83, 85, 86, 87, 88, 89, 90, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 105, 106, 107, 108, 109, 121, 123, 124, 125, 127, 133, 134  
 STATE • 17, 18, 26, 38, 68, 85, 86, 88, 102, 106, 107, 108, 109, 121, 125  
 <statement information item name> • 105  
 <statement label> • 81, 96, 97, 101  
 STYLE • 33, 132  
 SUBCLASS\_ORIGIN • 106  
 subject routine • 1, 31, 33, 57, 62, 63, 64, 76, 78, 92, 132  
 <subquery> • 33  
 successful completion • 18, 19, 38, 83, 86, 87  
 \_SYSTEM • 55, 57, 61  
 SYSTEM\_USER • 33, 41, 42, 69

— T —

table • 1, 6, 7, 11, 12, 15, 16, 20, 28, 29, 33, 35, 43, 44, 45, 46, 47, 54, 55, 61, 62, 63, 64, 65, 72, 74, 75, 76, 77, 78, 79, 80, 83, 101, 105, 106, 112, 115, 116, 117, 118, 119, 120, 121, 127  
 TABLE • 106, 115, 117, 118, 119  
 table constraint • 43, 44, 62  
 table constraint descriptor • 43, 62  
 table descriptor • 15, 16, 43  
 <table expression> • 29, 33, 63, 64  
 <table name> • 45, 47, 64, 65, 79  
 <table or query name> • 28, 29  
 <table reference> • 28, 29, 63, 64  
 TABLES • 117, 119  
 TABLE\_CATALOG • 112, 113, 117, 118  
 TABLE\_NAME • 106, 112, 113, 117, 118  
 TABLE\_SCHEMA • 112, 113, 117, 118  
 target • 27, 74, 90, 91  
 <target specification> • 27, 74, 90, 91  
 TEMPORARY • 106  
 temporary table • 15, 16, 55, 79, 80, 83, 106, 127  
 <temporary table declaration> • 15, 16, 54, 79, 80, 106  
 <terminated local cursor declaration> • 81, 82  
 <terminated local declaration> • 81  
 <terminated local handler declaration> • 81

<terminated SQL statement> • 81  
 THEN • 92, 94, 111  
 TO • 61  
 <token> • 23  
 transaction • 8, 20, 21, 121  
 transaction-initiating • 20  
 transaction state • 20  
 translation • 16, 51, 62, 63, 64, 65  
 translation descriptor • 16  
 <translation name> • 51  
 trigger • 16, 18, 29, 31, 43, 57, 87  
 <trigger definition> • 29  
 trigger descriptor • 16, 43, 57  
 <triggered action> • 31  
 TRIM • 38  
 TRUE • 102  
 Type • 105

— U —

underlying • 29  
 UNDO • 18, 23, 83, 85, 87, 131  
 unhandled completion condition • 18, 109  
 unhandled exception condition • 18, 83, 93, 95, 109  
 unhandled user-defined exception • 106, 109, 121  
 <unqualified schema name> • 26, 41  
 UNTIL • 23, 100, 131  
 UPDATE • 64, 65  
 <update statement: positioned> • 64, 65, 77  
 <update statement: searched> • 63, 64, 65, 78  
 USAGE • 62, 63, 64, 65, 112, 113, 117, 118, 134  
 USER • 33, 41, 42, 68, 69, 111, 112, 113, 114, 115, 119  
 user-defined • 18, 62, 63, 64, 65, 106, 109, 121  
 user-defined type • 62, 63, 64, 65  
 USERS • 115, 119

— V —

valid • 3, 21, 121  
 <value expression> • 31, 63, 64, 90, 91, 92  
 <value specification> • 27  
 variable • 1, 19, 21, 25, 26, 27, 28, 29, 30, 33, 41, 42, 46, 52, 69, 74, 81, 82, 83, 89, 90, 91, 97, 101, 106, 127, 134  
 VARIABLE • 106  
 view • 1, 9, 11, 16, 26, 28, 29, 43, 46, 47, 55, 57, 62, 83, 84, 87, 97, 98, 99, 100, 105, 111, 112, 113, 114, 123, 134  
 VIEW • 111, 112, 113, 114  
 view definition • 46  
 <view definition> • 46  
 view descriptor • 16, 43, 57, 62  
 viewed table • 46  
 visible • 18

— W —

warning • 121  
 WHEN • 92  
 WHERE • 111, 112, 113, 114  
 WHILE • 23, 99, 102, 106, 132  
 <while statement> • 19, 20, 69, 99, 106, 127, 134

**ISO/IEC JTC1/SC21 N11138 = DBL:CWB-003**

WITH • 61, 120  
without an intervening • 25, 26, 31, 58, 79, 80, 83,  
93, 95, 97, 102

— **Y** —

YES • 119, 120

*Note 1, Jim Melton, 16-10-97 14:57:04*  
SC32 N00003