Contents

Foreword ................................................................. vii
Introduction ............................................................. xvii

1 Scope .................................................................. 1

2 Normative references .............................................. 3

3 Definitions, notations, and conventions ...................... 5
  3.1 Definitions ............................................................. 5
  3.1.1 Definitions provided in Part 4 ......................... 5
  3.2 Notations .............................................................. 5
  3.3 Conventions .......................................................... 5
  3.3.1 Use of terms ......................................................... 5
  3.3.1.1 Exceptions ....................................................... 5
  3.3.1.2 Other terms ....................................................... 6
  3.3.2 Relationships to other parts of ISO/IEC 9075 ........... 6
  3.3.2.1 Clause, Subclause, and Table relationships .......... 6
  3.4 Object identifier for Database Language SQL .......... 12

4 Concepts ................................................................13
  4.1 SQL-server modules ................................................ 13
  4.2 SQL-invoked routines ............................................ 14
  4.3 SQL-paths ............................................................ 14
  4.4 Tables ............................................................... 14
  4.5 SQL-schemas ........................................................ 14
  4.6 Host parameters ................................................... 15
  4.6.1 Status parameters ............................................... 15
  4.7 Diagnostics area .................................................... 15
  4.8 Cursors .............................................................. 15
  4.9 Condition handling ............................................... 15
  4.10 SQL-statements .................................................... 17
    4.10.1 SQL-statements classified by function ............... 17
    4.10.2 Embeddable SQL-statements ......................... 18
    4.10.3 Directly executable SQL-statements ............... 18
    4.10.4 Iterated SQL-statements ............................... 18
    4.10.5 SQL-statements and transaction states ........... 18

ii Persistent Stored Modules (SQL/PSM)
4.10.6 Compound statements ..................................................... 19
4.10.7 SQL-statement atomicity .................................................. 19
4.11 Basic security model ..................................................... 19
4.11.1 Privileges .............................................................. 19

5 Lexical elements ......................................................... 21
5.1 <token> and <separator> .................................................. 21
5.2 Names and identifiers .................................................... 23

6 Scalar expressions ....................................................... 27
6.1 <value specification> and <target specification> ...................... 27
6.2 <identifier chain> ....................................................... 28
6.3 <SQL variable reference> ................................................ 30

7 Query expressions ....................................................... 31
7.1 <query specification> ..................................................... 31

8 Additional common elements .............................................. 33
8.1 <routine invocation> ..................................................... 33
8.2 <privileges> ............................................................ 35
8.3 <sqlstate value> ....................................................... 36

9 Schema definition and manipulation ........................................ 37
9.1 <schema definition> ...................................................... 37
9.2 <drop schema statement> ................................................ 38
9.3 <default clause> ......................................................... 39
9.4 <drop column scope clause> .............................................. 40
9.5 <drop column definition> ................................................ 41
9.6 <drop table constraint definition> ...................................... 43
9.7 <drop table statement> ................................................... 44
9.8 <view definition> ....................................................... 45
9.9 <drop view statement> ................................................... 46
9.10 <drop domain statement> ................................................ 47
9.11 <drop character set statement> ........................................ 48
9.12 <drop collation statement> ............................................. 49
9.13 <drop translation statement> ........................................... 50
9.14 <assertion definition> ................................................... 51
9.15 <drop assertion statement> ............................................. 52
9.16 <trigger definition> ...................................................... 53
9.17 <drop user-defined ordering statement> ............................. 54
9.18 <SQL-server module definition> ....................................... 55
9.19 <drop module statement> ............................................. 58
9.20 <drop data type statement> .......................................... 59
9.21 <SQL-invoked routine> ................................................. 60
9.22 <drop routine statement> ............................................. 62
9.23 <drop user-defined cast statement> .................................. 63
# TABLES

<table>
<thead>
<tr>
<th>Tables</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clause, Subclause, and Table relationships</td>
<td>6</td>
</tr>
<tr>
<td>&lt;identifier&gt;s for use with &lt;get diagnostics statement&gt;</td>
<td>109</td>
</tr>
<tr>
<td>SQL-statement codes for use in the diagnostics area</td>
<td>110</td>
</tr>
<tr>
<td>SQLSTATE class and subclass values</td>
<td>129</td>
</tr>
<tr>
<td>SQL/PSM feature taxonomy for features outside Core SQL</td>
<td>143</td>
</tr>
</tbody>
</table>
Foreword

(This foreword is not a part of American National Standard ANSI/ISO/IEC 9075-4:1999.)


This American National Standard specifies the ability to persistently store SQL modules in a database for later use by applications.

ANSI/ISO/IEC 9075 consists of the following parts, under the general title Information Systems — Database Language — 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)

This American National Standard contains six Informative Annexes that are not considered part of the Standard:


Requests for interpretation, suggestions for improvement or addenda, or defect reports are welcome. They should be sent to the National Committee for Information Technology Standards (NCITS), 1250 Eye Street, NW, Suite 200, Washington, DC 20005.

This Standard was processed and approved for submittal to ANSI by NCITS. Committee approval of this Standard does not necessarily imply that all committee members voted for approval. At the time that it approved this Standard, NCITS had the following members:

<table>
<thead>
<tr>
<th>NCITS Chairman</th>
<th>NCITS Vice Chair</th>
<th>NCITS Secretary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ms. Karen Higgenbottom</td>
<td>Mr. Dave Michael</td>
<td>Ms. Monica Vago</td>
</tr>
</tbody>
</table>

*Non-Response **Abstain
PRODUCERS=13

Apple Computer Inc.
Mr. David Michael [P]
M/S 301-4F
1 INFINITE LOOP
CUPERTINO CA 95014
+1.408.862.5451
Fax: +1.408.974.2691
deek@apple.com

Mr. Jerry Kellenbenz [A]
M/S 301-4F
1 INFINITE LOOP
CUPERTINO CA 95014
+1.408.974.7341
Fax: +1.408.974.2691
jerryk@taurus.apple.com

Bull HN Information Systems Inc.
Mr. Randall Kilmartin [P]
M/S B58
13430 N. BLACK CANYON HIGHWAY
PHOENIX AZ 85029
+1.602.862.4905
Fax: +1.602.862.3285
randy.kilmartin@bull.com

Compaq Computer Corporation
Mr. Scott Jameson [P]
M/S AKO2-3/D10
50 NAGOG PARK
ACTON MA 01720
+1.508.264.7468
Fax: +1.508.264.7656
scott.jameson@digital.com

Mr. Stephen Heil [A]
M/S 110605
20555 SH249
HOUSTON TX 77269-2000
+1.281.518.0781
Fax: +1.281.518.3519
stephen.heil@compaq.com

Hewlett-Packard Company
Ms. Karen Higginbottom [P]
M/S 43UC
19420 HOMESTEAD ROAD
CUPERTINO CA 95014
+1.408.447.3274
Fax: +1.408.447.2247
higginbottom@cup.hp.com
Ms. Wendy Fong [A]
M/S 47U-10
19447 PRUNERIDGE AVENUE
CUPERTINO CA 95014
+1.408.447.4463
Fax: +1.408.447.3995
Wendy_Fong@HP-Cupertino-notes2.om.com

Hitachi American Ltd.
Mr. John Neumann [P]
43533 GOLDEN MEADOW CIRCLE
ASHBURN VA 22011
+1.703.729.4858
Fax: +1.703.729.0304
openstrat@aol.com

Mr. Hal Miyamoto [A]
MS 730
2000 SIERRA POINT PKWY
BRISBANE CA 94005-1835
+1.650.244.7218
Fax: +1.650.244.7250
miyamoth@halsp.hitachi.com

IBM Corporation
Ronald F. Silletti [P]
Program Director of Standards
Intellectual Property & Licensing
IBM Corporation
NORTH CASTLE DRIVE
ARMONK, NY 10504
+1.914.765.4373
Fax: +1.914.765.4420
silletti@us.ibm.com

Mr. Joel Urman [A]
IBM Corporation
M/S NC113, Room 1B111
NORTH CASTLE DRIVE
ARMONK, NY 10504-1785
+1.914.765.4392
Fax: +1.914.765.4420
joelu@us.ibm.com

Lucent Technologies Inc.
Mr. Herbert Bertine [P]
ROOM 4K-316
101 CRAWFORDS CORNER RD
HOLMDEL NJ 07733-3030
+1.908.949.4022
Fax: +1.908.949.1196
hbertine@lucent.com
x Persistent Stored Modules (SQL/PSM)
Sybase, Inc.
Mr. Billy Ho [P]
1650 65th STREET
EMERYVILLE CA 94608
+1.510.922.4416
billy.ho@sybase.com

US Department of Defense/DISA
Mr. Russ Richards [P]
10701 PARKRIDGE BLVD
RESTON VA 20191-4357
+1.703.735.3552
Fax: +1.703.735.3257
richarl@ncr.disa.mil

Dr. Doris Bernardini [A]
PO BOX 2309
RESTON VA 20195-0309
+1.703.735.3566
Fax: +1.703.735.3257
bernardd@ncr.disa.mil

US Department of Energy
Mr. Bruce White [P]
MA-43 GERMANTOWN BUILDING
19901 GERMANTOWN ROAD
GERMANTOWN MD 20874-1290
+1.301.903.6977
Fax: +1.301.903.4101
bruce.white@hq.doe.gov

Ms. Carol Blackston [A]
MA-43 GERMANTOWN BUILDING
19901 GERMANTOWN ROAD
GERMANTOWN MD 20874-1290
+1.301.903.4294
Fax: +1.301.903.4101
carol.blackston@hq.doe.gov

GENERAL INTEREST=2
Institute for Certification of Computer Professionals
Mr. Kenneth M. Zemrowski [P]
C/O ISN/TRW SUITE C-65
1280 MARYLAND AVENUE SW
WASHINGTON DC 20024
+1.202.479.0085 X225
Fax: +1.202.479.4187
kenneth.zemrowski@trw.com

Mr. Tom Kurihara [A]
TKSTDS & ASSOCIATES, INC.
5713 6TH STREET, NORTH
ARLINGTON, VA 22205-1013
+1.703.516.9650
Fax: +1.703.516.4688
American National Standard ANSI/ISO/IEC 9075-1:1999 was prepared by Technical Committee Group NCITS H2, Database, working under the auspices of Accredited National Standards Committee NCITS (National Committee for Information Technology Standards). Technical Committee H2 on Database, which developed this Standard, had the following members:

Donald R. Deutsch, Chair

Bruce M. Horowitz, Vice-Chair

Krishna Kulkarni, International Representative

Barry D. Vickers, Treasurer

Michael M. Gorman, Secretary

Jim Melton, Editor

Chuck Campbell
Keith Hare
R. Michael Lefler
Janet Prichard
Paul Scarponcini
Fred Zemke

Andrew Eisenberg
Doug Inkster
Michael Pantaleo
William E. Raab
Herb Sutter

Chris Farrar
Tom Julien
Frank Pellow
Bruce K. Rosen
Bie Tie
Others holding Technical Committee H2 membership while the committee was developing this standard were the following:

Dean A. Anderson
John Barney
David Beech
Amelia Carlson
Judy Dillman
Donna Fisher
Leonard J. Gallagher
Tom Harwood
Phil Jones
William J. Koster
Fran Lynch
M. Reza Monajjemi
Kee Ong
Richard Pedigo
Tom Perry
Michael Pizzo
Ed Reynolds
Chander Sarna
Scott Schnier
Phil Shaw
Yang Tsouya
Andrew Wade
Robert Zeek

Mark Ashworth
Daniel Barton
Richard Boyne
Joe Celko
T. N. Doraiswamy
Jeffrey Fried
Kyle W. Geiger
Wei Hong
Bill Kelley
Vince Kowalski
Nelson Mattos
Santanu Mukhopadhyay
Emmanuel Onuegbu
Ed Peeler
Sherry Petersen
Mahesh Rao
Jeff Richey
Robert Saunders
Christine Semeniuk
Maurice L. Smith
Michael Ubell
Ken Wilner
Hans Zeller

James Barnette
Aime Bayle
Andras Budinszky
Arthur Culbertson
Shel Finkelstein
Barry Fritchman
Jim Graham
Ken Jacobs
William Kent
Melissa LoBiando
Jeff Mischkinsky
Phil Nelson
Dipak Patel
Paul Perkovic
William Phillips
George Raudabaugh
John Sadd
David Schnepfer
Larry Settle
Madhukar Thakur
Murali Venkatrao
David Yach
Introduction

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

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

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

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

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

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

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

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

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

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


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

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

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

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

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

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

18) Clause 18, “Definition Schema”, defines base tables on which the viewed tables containing schema information depend.
19) Clause 19, “Status codes”, defines SQLSTATE values related to persistent stored modules.

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


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

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


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

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

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

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

---
- The specification of statements to direct the flow of control.
- The assignment of the result of expressions to variables and parameters.
- The specification of condition handlers that allow SQL-invoked routines to deal with various conditions that arise during their execution.
- The specification of statements to signal and resignal conditions.
- The declaration of local cursors.
- The declaration of local variables.
---

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

NOTE 1 – The context for this part of ISO/IEC 9075 is described by the Reference Model of Data Management (ISO/IEC 10032:1993).
2 Persistent Stored Modules (SQL/PSM)
2 Normative references

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


3 Definitions, notations, and conventions

3.1 Definitions

3.1.1 Definitions provided in Part 4

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

3.2 Notations

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

3.3 Conventions

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

3.3.1 Use of terms

3.3.1.1 Exceptions

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.


— Diagnostic information is to be made available.

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

The phrase “C is re-raised by S” is used in General Rules and elsewhere to indicate that C, a condition raised by an SQL-statement executed during execution of S, is raised again by S.
3.3 Conventions

3.3.1.2 Other terms

An SQL-statement \( S_1 \) may be said to be executed as a direct result of executing an SQL-statement if \( S_1 \) is the SQL-statement contained in an <externally-invoked procedure> or <SQL-invoked routine> that has been executed.

An SQL-statement \( S_1 \) may be said to be executed as a direct result of executing an <SQL control statement> \( S_2 \) if \( S_2 \) contains \( S_1 \).

The phrase “The scope of a <handler declaration> contained in a/an \( Y \) is that \( Y \), excluding every <SQL schema statement> contained in that \( Y \)” means that the scope of the <handler declaration> does not extend to SQL-statements contained in such an <SQL schema statement>; it does, however, extend to the <SQL schema statement> itself.

3.3.2 Relationships to other parts of ISO/IEC 9075

3.3.2.1 Clause, Subclause, and Table relationships

<table>
<thead>
<tr>
<th>Clause, Subclause, or Table in this part of ISO/IEC 9075</th>
<th>Corresponding Clause, Subclause, or Table from another part</th>
<th>Part containing correspondence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subclause 3.1.1, “Definitions provided in Part 4”</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Subclause 3.3.1, “Use of terms”</td>
<td>Subclause 3.3.1, &quot;Use of terms&quot;</td>
<td>ISO/IEC 9075-2</td>
</tr>
<tr>
<td>Subclause 3.3.1.1, “Exceptions”</td>
<td>Subclause 6.2.3.1, &quot;Exceptions&quot;</td>
<td>ISO/IEC 9075-1</td>
</tr>
<tr>
<td>Subclause 3.3.1.2, “Other terms”</td>
<td>Subclause 6.2.3.7, &quot;Other terms&quot;</td>
<td>ISO/IEC 9075-1</td>
</tr>
<tr>
<td>Subclause 3.3.2, “Relationships to other parts of ISO/IEC 9075”</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Subclause 3.3.2.1, “Clause, Subclause, and Table relationships”</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Subclause 4.1, “SQL-server modules”</td>
<td>(none)</td>
<td>(none)</td>
</tr>
</tbody>
</table>

6 Persistent Stored Modules (SQL/PSM)
Table 1—Clause, Subclause, and Table relationships (Cont.)

<table>
<thead>
<tr>
<th>Clause, Subclause, or Table in this part of ISO/IEC 9075</th>
<th>Corresponding Clause, Subclause, or Table from another part</th>
<th>Part containing correspondence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subclause 4.6, &quot;Host parameters&quot;</td>
<td>Subclause 4.26, &quot;Host parameters&quot;</td>
<td>ISO/IEC 9075-2</td>
</tr>
<tr>
<td>Subclause 4.7, &quot;Diagnostics area&quot;</td>
<td>Subclause 4.27, &quot;Diagnostics area&quot;</td>
<td>ISO/IEC 9075-2</td>
</tr>
<tr>
<td>Subclause 4.9, &quot;Condition handling&quot;</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Subclause 4.10.3, &quot;Directly executable SQL-statements&quot;</td>
<td>Subclause 4.6.6, &quot;Directly executable SQL-statements&quot;</td>
<td>ISO/IEC 9075-5</td>
</tr>
<tr>
<td>Subclause 4.10.4, &quot;Iterated SQL-statements&quot;</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Subclause 4.10.6, &quot;Compound statements&quot;</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Subclause 4.10.7, &quot;SQL-statement atomicity&quot;</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Subclause 4.11, &quot;Basic security model&quot;</td>
<td>Subclause 4.31, &quot;Basic security model&quot;</td>
<td>ISO/IEC 9075-2</td>
</tr>
<tr>
<td>Clause 5, &quot;Lexical elements&quot;</td>
<td>Clause 5, &quot;Lexical elements&quot;</td>
<td>ISO/IEC 9075-2</td>
</tr>
<tr>
<td>Subclause 5.1, &quot;&lt;token&gt; and &lt;separator&gt;&quot;</td>
<td>Subclause 5.2, &quot;&lt;token&gt; and &lt;separator&gt;&quot;</td>
<td>ISO/IEC 9075-2</td>
</tr>
<tr>
<td>Subclause 5.2, &quot;Names and identifiers&quot;</td>
<td>Subclause 5.4, &quot;Names and identifiers&quot;</td>
<td>ISO/IEC 9075-2</td>
</tr>
<tr>
<td>Clause 6, &quot;Scalar expressions&quot;</td>
<td>Clause 6, &quot;Scalar expressions&quot;</td>
<td>ISO/IEC 9075-2</td>
</tr>
</tbody>
</table>
### Table 1—Clause, Subclause, and Table relationships (Cont.)

<table>
<thead>
<tr>
<th>Clause, Subclause, or Table in this part of ISO/IEC 9075</th>
<th>Corresponding Clause, Subclause, or Table from another part</th>
<th>Part containing correspondence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subclause 6.3, &quot;&lt;SQL variable reference&gt;&quot;</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Clause 7, &quot;Query expressions&quot;</td>
<td>Clause 7, &quot;Query expressions&quot;</td>
<td>ISO/IEC 9075-2</td>
</tr>
<tr>
<td>Subclause 7.1, &quot;&lt;query specification&gt;&quot;</td>
<td>Subclause 7.11, &quot;&lt;query specification&gt;&quot;</td>
<td>ISO/IEC 9075-2</td>
</tr>
<tr>
<td>Subclause 8.3, &quot;&lt;sqlstate value&gt;&quot;</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Subclause 9.8, &quot;&lt;view definition&gt;&quot;</td>
<td>Subclause 11.21, &quot;&lt;view definition&gt;&quot;</td>
<td>ISO/IEC 9075-2</td>
</tr>
</tbody>
</table>
Table 1—Clause, Subclause, and Table relationships (Cont.)

<table>
<thead>
<tr>
<th>Clause, Subclause, or Table in this part of ISO/IEC 9075</th>
<th>Corresponding Clause, Subclause, or Table from another part</th>
<th>Part containing correspondence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subclause 9.17, &quot;&lt;drop user-defined ordering statement&gt;&quot;</td>
<td>Subclause 11.55, &quot;&lt;drop user-defined ordering statement&gt;&quot;</td>
<td>ISO/IEC 9075-2</td>
</tr>
<tr>
<td>Subclause 9.18, &quot;&lt;SQL-server module definition&gt;&quot;</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Subclause 9.19, &quot;&lt;drop module statement&gt;&quot;</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Subclause 10.1, &quot;&lt;grant statement&gt;&quot;</td>
<td>Subclause 12.6, &quot;&lt;revoke statement&gt;&quot;</td>
<td>ISO/IEC 9075-2</td>
</tr>
<tr>
<td>Subclause 11.1, &quot;Calls to an &lt;externally-invoked procedure&gt;&quot;</td>
<td>Subclause 13.4, &quot;Calls to an &lt;externally-invoked procedure&gt;&quot;</td>
<td>ISO/IEC 9075-2</td>
</tr>
<tr>
<td>Subclause 12.6, &quot;&lt;update statement: positioned&gt;&quot;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 1—Clause, Subclause, and Table relationships (Cont.)

<table>
<thead>
<tr>
<th>Clause, Subclause, or Table in this part of ISO/IEC 9075</th>
<th>Corresponding Clause, Subclause, or Table from another part</th>
<th>Part containing correspondence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clause 13, “Control statements”</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Subclause 13.1, &quot;&lt;compound statement&gt;&quot;</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Subclause 13.2, &quot;&lt;handler declaration&gt;&quot;</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Subclause 13.3, &quot;&lt;condition declaration&gt;&quot;</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Subclause 13.4, &quot;&lt;SQL variable declaration&gt;&quot;</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Subclause 13.5, &quot;&lt;assignment statement&gt;&quot;</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Subclause 13.6, &quot;&lt;case statement&gt;&quot;</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Subclause 13.7, &quot;&lt;df statement&gt;&quot;</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Subclause 13.8, &quot;&lt;iterate statement&gt;&quot;</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Subclause 13.9, &quot;&lt;leave statement&gt;&quot;</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Subclause 13.10, &quot;&lt;loop statement&gt;&quot;</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Subclause 13.11, &quot;&lt;while statement&gt;&quot;</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Subclause 13.12, &quot;&lt;repeat statement&gt;&quot;</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Subclause 13.13, &quot;&lt;for statement&gt;&quot;</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Subclause 15.1, &quot;&lt;embedded SQL host programs&gt;&quot;</td>
<td>Subclause 16.1, &quot;&lt;LB&gt;LB&gt;embedded SQL host program&gt;&quot;</td>
<td>ISO/IEC 9075-5</td>
</tr>
<tr>
<td>Subclause 16.2, &quot;&lt;signal statement&gt;&quot;</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Subclause 16.3, &quot;&lt;resignal statement&gt;&quot;</td>
<td>(none)</td>
<td>(none)</td>
</tr>
</tbody>
</table>
### Table 1—Clause, Subclause, and Table relationships (Cont.)

<table>
<thead>
<tr>
<th>Clause, Subclause, or Table in this part of ISO/IEC 9075</th>
<th>Corresponding Clause, Subclause, or Table from another part</th>
<th>Part containing correspondence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clause 17, &quot;Information Schema&quot;</td>
<td>Clause 20, &quot;Information Schema&quot;</td>
<td>ISO/IEC 9075-2</td>
</tr>
<tr>
<td>Subclause 17.1, &quot;MODULE_COLUMN_USAGE view&quot;</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Subclause 17.2, &quot;MODULE_PRIVILEGES view&quot;</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Subclause 17.3, &quot;MODULE_TABLE_USAGE view&quot;</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Subclause 17.4, &quot;MODULES view&quot;</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Clause 18, &quot;Definition Schema&quot;</td>
<td>Clause 21, &quot;Definition Schema&quot;</td>
<td>ISO/IEC 9075-2</td>
</tr>
<tr>
<td>Subclause 18.1, &quot;MODULE_COLUMN_USAGE base table&quot;</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Subclause 18.2, &quot;MODULE_PRIVILEGES base table&quot;</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Subclause 18.3, &quot;MODULE_TABLE_USAGE base table&quot;</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Subclause 18.4, &quot;MODULES base table&quot;</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Clause 19, &quot;Status codes&quot;</td>
<td>Clause 22, &quot;Status codes&quot;</td>
<td>ISO/IEC 9075-2</td>
</tr>
<tr>
<td>Clause 20, &quot;Conformance&quot;</td>
<td>Clause 8, &quot;Conformance&quot;</td>
<td>ISO/IEC 9075-1</td>
</tr>
<tr>
<td>Table 1, &quot;Clause, Subclause, and Table relationships&quot;</td>
<td>(none)</td>
<td>(none)</td>
</tr>
<tr>
<td>Table 2, &quot;&lt;identifier&gt;s for use with &lt;get diagnostics statement&gt;&quot;</td>
<td>Table 25, &quot;&lt;identifier&gt;s for use with &lt;get diagnostics statement&gt;&quot;</td>
<td>ISO/IEC 9075-2</td>
</tr>
</tbody>
</table>
3.4 Object identifier for Database Language SQL

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

**Format**

\[
\text{<Part 4 yes> ::= <Part 4 conformance> <Part 4 module>}
\]

\[
\text{<Part 4 conformance> ::= 4 | sqlpsm1999 <left paren> 4 <right paren>}
\]

\[
\text{<Part 4 module> ::= <Part 4 module yes> | <Part 4 module no>}
\]

\[
\text{<Part 4 module yes> ::= 1 | moduleyes <left paren> 1 <right paren>}
\]

\[
\text{<Part 4 module no> ::= 0 | moduleno <left paren> 0 <right paren>}
\]

**Syntax Rules**

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

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

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

4.1 SQL-server modules

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

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

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

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

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

— The SQL-server module name of the SQL-server module.
— The descriptor of the character set in which the SQL-server module is represented.
— The default schema name used for implicit qualification of unqualified names in the SQL-server module.
— The SQL-server module authorization identifier of the SQL-server module.
— The list of schema names contained in the <SQL-server module path specification>.
— The table descriptor of every local temporary table declared in the SQL-server module.
— The descriptor of every SQL-invoked routine contained in the SQL-server module.
— The text of the <SQL-server module definition>.
4.2 SQL-invoked routines

An SQL-invoked routine is either a component of an SQL-server module definition or an element of an SQL-schema. An SQL-invoked routine that is an element of an SQL-schema is called a schema-level routine.

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

If the SQL-invoked routine is not a schema-level routine, then the SQL-server module name of the SQL-server module that includes the SQL-invoked routine and the schema name of the schema that includes the SQL-server module.

4.3 SQL-paths

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

4.4 Tables

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

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.

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

4.5 SQL-schemas

In this part of ISO/IEC 9075, the term “schema” is used only in the sense of SQL-schema. Each component descriptor is either a domain descriptor, a base table descriptor, a view descriptor, a constraint descriptor, a privilege descriptor, a character set descriptor, a collation descriptor, a translation descriptor, a trigger descriptor, a user-defined type descriptor, an SQL-server module descriptor, or an SQL-invoked routine descriptor. The persistent objects described by
the descriptors are said to be owned by or to have been created by the <authorization identifier> of the schema.

4.6 Host parameters

4.6.1 Status parameters

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.

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

4.7 Diagnostics area

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.

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

4.8 Cursors

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

4.9 Condition handling

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

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

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

A general <handler declaration> is one that is associated with the <condition value>s SQLEXCEPTION, SQLWARNING, or NOT FOUND. All other <handler declaration>s are specific <handler declaration>s.
A condition represents an error or informational state caused by execution of an <SQL procedure statement>. Conditions are raised to provide information in the diagnostics area about the execution of an <SQL procedure statement>.

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

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

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

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

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

A handler type specifies CONTINUE, EXIT, or UNDO.

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

- Execute the handler action.
- Return control to the SQL-statement following the executing statement that raised the condition. If the SQL-statement that raised the condition is an <SQL control statement>, then “the following SQL-statements” do not include the <SQL procedure statement>s contained within the <SQL control statement>.
  
  NOTE 2 – For example, if the executing statement that raised the condition is an <if statement>, control is returned to the SQL-statement following the <if statement>.

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

- Execute the handler action.
- Implicitly LEAVE the <compound statement> for which the handler was created, with no active exception condition.

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

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

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

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

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

4.10 SQL-statements

4.10.1 SQL-statements classified by function

The following are additional main classes of SQL-statements:

— SQL-control declarations

The following are additional SQL-schema statements:

— <SQL-server module definition>
— <alter module statement>
— <drop module statement>

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>

The following are the SQL-control declarations:

— <condition declaration>
— <handler declaration>
— <SQL variable declaration>
4.10 SQL-statements

The following are additional SQL-diagnostics statements:

— <signal statement>
— <resignal statement>

4.10.2 Embeddable SQL-statements

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

— All SQL-control statements

NOTE 3 – SQL-control declarations contained in (for example) <compound statement>s are permitted, even when the containing SQL-statement is embedded in an <embedded SQL host program>.

4.10.3 Directly executable SQL-statements

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

— All SQL-control statements

4.10.4 Iterated SQL-statements

The following are the iterated SQL-statements:

— <loop statement>
— <while statement>
— <repeat statement>
— <for statement>

4.10.5 SQL-statements and transaction states

The following additional SQL-statement is a transaction-initiating SQL-statement:

• <for statement>

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

• <iterate statement>
• <leave statement>
4.10 SQL-statements

The following additional SQL-statements are possibly transaction-initiating SQL-statements:

- SQL-control statements other than:
  - <for statement>
  - <iterate statement>
  - <leave statement>

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

4.10.6 Compound statements

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

4.10.7 SQL-statement atomicity

The execution of <compound statement>s that specify ATOMIC is atomic.

4.11 Basic security model

4.11.1 Privileges

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

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

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

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.

5 Lexical elements

5.1 <token> and <separator>

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

Format

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

<reserved word> ::= !! All alternatives from ISO/IEC 9075-2
| !! All alternatives from ISO/IEC 9075-5
| CONDITION
| DO
| ELSEIF | EXIT
| HANDLER
| IF | ITERATE
| LEAVE | LOOP
| REDO | REPEAT | RESIGNAL
| SIGNAL
| UNDO | UNTIL
| WHILE

Syntax Rules
No additional Syntax Rules.

Access Rules
No additional Access Rules.
5.1 <token> and <separator>

**General Rules**

No additional General Rules.
5.2 Names and identifiers

Function
Specify names.

Format

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

<SQL variable name> ::= <identifier>

<condition name> ::= <identifier>

Syntax Rules

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

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

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

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

5) Case:
   a) If <user-defined type name> UDTN with a <qualified identifier> QI is simply contained in <user-defined type>, then
      Case:
      i) If UDTN contains a <schema name> SN, then the schema identified by SN shall contain the descriptor of a user-defined type UDT such that the <qualified identifier> of UDT is equivalent to QI. UDT is the user-defined type identified by UDTN.
5.2 Names and identifiers

ii) Otherwise:

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

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

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

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

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

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

4) Case:

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

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

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

b) Otherwise,

   Case:

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

   ii) If UDTN is contained in an <SQL-server module definition> without an intervening <schema definition>, then the implicit <schema name> of UDTN is the <schema name> that is specified or implicit in <SQL-server module definition>.
iii) If UDTN is contained in a <schema definition> that is not contained in an <SQL-client module definition>, then the implicit <schema name> of UDTN is the <schema name> that is specified or implicit in <schema definition>.

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

**Access Rules**

No additional Access Rules.

**General Rules**

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

2) An <SQL variable name> identifies an SQL variable.

3) A <condition name> identifies an exception condition or a completion condition and optionally a corresponding SQLSTATE value.
6 Scalar expressions

6.1 <value specification> and <target specification>

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

Format

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

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

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

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

Syntax Rules
No additional Syntax Rules.

Access Rules
No additional Access Rules.

General Rules
No additional General Rules.
6.2 <identifier chain>

Function
Disambiguate a <period>-separated chain of identifiers.

Format

No additional Format items.

Syntax Rules

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

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

3) If the innermost possible qualifier is a beginning label, then let SV be the SQL variable whose identifier is equivalent to $I_1$. PIC$_1$ is the basis of IC, the basis length is 1 (one), the basis scope is the scope of SP, and the basis referent is SV.

4) If IC is contained within the scope of a beginning label whose associated local declaration list includes an SQL variable whose identifier is equivalent to $I_1$, then PIC$_1$ is a candidate basis of IC, the scope of PIC$_1$ is the scope of SV, and the referent of PIC$_1$ is SV.

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

$$ (PIC_{BL}) \text{ <period> } I_{BL+1} \text{ <period> } \ldots \text{ <period> } I_N $$

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

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

6) A basic identifier chain whose basis referent is an SQL variable is an SQL variable reference.
Access Rules

None.

General Rules

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

Function
Reference an SQL variable.

Format

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

Syntax Rules

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

Access Rules

None.

General Rules

None.
7 Query expressions

7.1 <query specification>

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

Format
No additional Format items.

Syntax Rules
1) If IC is contained within the scope of a <beginning label> whose associated <local declaration list> includes an SQL variable SV whose <identifier> is equivalent to I, then PIC is a candidate basis of IC and the scope of PIC is the scope of SP.

2) An SQL variable.

Access Rules
No additional Access Rules.

General Rules
No additional General Rules.
8 Additional common elements

8.1 <routine invocation>

Function
Invoke an SQL-invoked routine.

Format
No additional Format items.

Syntax Rules
1) An SQL-invoked routine R is an executable routine if and only if R is a possibly candidate routine and
   Case:
   a) If RI is contained in an <SQL schema statement>, then
      Case:
      i) If RI is contained in an <SQL-server module definition> M, then the applicable privileges of the <authorization identifier> that owns the containing schema include EXECUTE on M.
      ii) Otherwise, the applicable privileges of the <authorization identifier> that owns the containing schema include EXECUTE on R.
   b) Otherwise,
      Case:
      i) If RI is contained in an <SQL-server module definition> M, then the current privileges include EXECUTE on M.
      ii) Otherwise, the current privileges include EXECUTE on R.

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

2) If RI is contained in an <SQL-server module definition>, then let DP be the SQL-path of that <SQL-server module definition>.

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

4) If RI is contained in an <SQL-server module definition>, then let DP be the SQL-path of that <SQL-server module definition>.
8.1 <routine invocation>

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

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

Access Rules

No additional Access Rules.

General Rules

1) Insert after GR6(c))

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

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

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

Function
Specify privileges.

Format

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

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

Access Rules
No additional Access Rules.

General Rules
No additional General Rules.
8.3 <sqlstate value>

Function
Specify an SQLSTATE value.

Format

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

Syntax Rules
1) Let \( L \) be the <character string literal> contained in <sqlstate value>.
2) The implicit or explicit character set of \( L \) shall be the implementation-defined character set in which SQLSTATE parameter values are returned.
3) Let \( V \) be the character string that is the value of
   \[ \text{TRIM (BOTH ' ' FROM } L \text{)} \]
4) \( V \) shall comprise either:
   a) Five characters of which the first two have the form of a standard-defined class value and the last three have the form of a standard-defined subclass value.
   b) Five characters of which the first two have the form of a standard-defined class value and the last three have the form of an implementation-defined subclass value.
   c) Five characters of which the first two have the form of an implementation-defined class value and the last three have the form of either a standard-defined subclass value or an implementation-defined subclass value.
5) \( V \) shall not be the SQLSTATE value for the condition successful completion.
6) The SQLSTATE value defined by the <sqlstate value> is \( V \).

Access Rules
None.

General Rules
None.
9 Schema definition and manipulation

9.1 <schema definition>

Function
Define a schema.

Format

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

Syntax Rules
No additional Syntax Rules.

Access Rules
No additional Access Rules.

General Rules
No additional General Rules.
9.2 <drop schema statement>

Function
Destroy a schema.

Format
No additional Format items.

Syntax Rules
1) If RESTRICT is specified, then S shall not include any SQL-server modules.

Access Rules
No additional Access Rules.

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

   DROP MODULE MN CASCADE
9.3 `<default clause>`

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

**Format**

No additional Format items.

**Syntax Rules**

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

**Access Rules**

No additional Access Rules.

**General Rules**

None.
9.4 <drop column scope clause>

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

Format
No additional Format items.

Syntax Rules
1) The module descriptor of any SQL-server module.

Access Rules
None.

General Rules
1) For every SQL-invoked routine R whose routine descriptor includes a <SQL routine body> that contains an impacted dereference operation,
   Case:
   a) If R is included in an SQL-server module M, then let MN be the <SQL-server module name> of M. The following <drop module statement> is effectively executed without further Access Rule checking:
      DROP MODULE MN CASCADE
   b) Otherwise, let SN be the <specific name> of R. The following <drop routine statement> is effectively executed for every R without further Access Rule checking:
      DROP SPECIFIC ROUTINE SN CASCADE

2) Let SSM be any SQL-server module whose module descriptor includes an impacted dereference operation, and let MN be the <SQL-server module name> of SSM. The following <drop module statement> is effectively executed without further Access Rule checking:
   DROP MODULE MN CASCADE
9.5 <drop column definition>

**Function**
Destroy a column.

**Format**
No additional Format items.

**Syntax Rules**
1) If RESTRICT is specified, then C shall not be referenced in any of the following:
   a) The <query expression> of any view descriptor.
   b) The <search condition> of any constraint descriptor other than a table constraint descriptor that contains references to no other column and that is included in the table descriptor of T.
   c) The <SQL routine body> of any routine descriptor.
   d) Either an explicit trigger column list or a triggered action column set of any trigger descriptor.
   e) The module descriptor of any SQL-server module.

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

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

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

**Access Rules**
No additional Access Rules.

**General Rules**
1) Let R be any SQL-invoked routine whose routine descriptor contains the <column name> of C in the <SQL routine body>.
   
   **Case:**
   a) If R is included in an SQL-server module M, then let MN be the <SQL-server module name> of M. The following <drop module statement> is effectively executed without further Access Rule checking:

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

```
DROP SPECIFIC ROUTINE SN CASCADE
```

2) Insert after GR4] Let SSM be any SQL-server module whose module descriptor includes the <column name> of C and let MN be the <SQL-server module name> of SSM. The following <drop module statement> is effectively executed without further Access Rule checking:

```
DROP MODULE MN CASCADE
```
9.6 *<drop table constraint definition>*

**Function**
Destroy a constraint on a table.

**Format**

*No additional Format items.*

**Syntax Rules**

*No additional Syntax Rules.*

**Access Rules**

*No additional Access Rules.*

**General Rules**

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

Case:

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

   `DROP MODULE MN CASCADE`

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

   `DROP SPECIFIC ROUTINE SN CASCADE`
9.7 `<drop table statement>`

**Function**
Destroy a table.

**Format**

No additional Format items.

**Syntax Rules**

1) Insert after SR6) If RESTRICT is specified, then T shall not be referenced in the module descriptor of any SQL-server module.

**Access Rules**

No additional Access Rules.

**General Rules**

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

   Case:
   a) If R an SQL-server module M, then let MN be the `<SQL-server module name>` of M. The following `<drop module statement>` is effectively executed without further Access Rule checking:
      
      DROP MODULE MN CASCADE
   
   b) Otherwise, let SN be the `<specific name>` of R. The following `<drop routine statement>` is effectively executed without further Access Rule checking:
      
      DROP SPECIFIC ROUTINE SN CASCADE

2) Insert after GR5) Let SSM be any SQL-server module whose module descriptor includes the `<table name>` of T and let MN be the `<SQL-server module name>` of SSM. The following `<drop module statement>` is effectively executed without further Access Rule checking:

      DROP MODULE MN CASCADE
9.8 <view definition>

Function
Define a viewed table.

Format
No additional Format items.

Syntax Rules
1) Insert after SR2)
   NOTE 11 – <SQL variable name> is also excluded because of the scoping rules for <SQL variable name>.

Access Rules
No additional Access Rules.

General Rules
No additional General Rules.
9.9  <drop view statement>

Function
Destroy a view.

Format
No additional Format items.

Syntax Rules
1) [Insert after SR4)] If RESTRICT is specified, then V shall not be referenced in the module descriptor of any SQL-server module.

Access Rules
No additional Access Rules.

General Rules
1) [Replace GR4)] Let R be any SQL-invoked routine whose routine descriptor contains the <table name> of V in the <SQL routine body>.
   Case:
   a) If R is included in an SQL-server module M, then let MN be the <SQL-server module name> of M. The following <drop module statement> is effectively executed without further Access Rule checking:

   DROP MODULE MN CASCADE

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

   DROP SPECIFIC ROUTINE SN CASCADE

2) [Insert after GR4)] Let SSM be any SQL-server module whose module descriptor includes the <table name> of V and let MN be the <SQL-server module name> of SSM. The following <drop module statement> is effectively executed without further Access Rule checking:

   DROP MODULE MN CASCADE
9.10 <drop domain statement>

Function
Destroy a domain.

Format
No additional Format items.

Syntax Rules
1) 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) Let SSM be any SQL-server module whose module descriptor includes the <column name> of C and let MN be the <SQL-server module name> of SSM. The following <drop module statement> is effectively executed without further Access Rule checking:

DROP MODULE MN CASCADE
9.11 <drop character set statement> 

Function
Destroy a character set.

Format
No additional Format items.

Syntax Rules
1) C shall not be referenced in the module descriptor of any SQL-server module.

Access Rules
No additional Access Rules.

General Rules
1) 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) Let SSM be any SQL-server module whose module descriptor includes the <character set name> of C and let MN be the <SQL-server module name> of SSM. The following <drop module statement> is effectively executed without further Access Rule checking:
   DROP MODULE MN CASCADE
9.12 <drop collation statement>

Function
Destroy a collating sequence.

Format
No additional Format items.

Syntax Rules
No additional Syntax Rules.

Access Rules
No additional Access Rules.

General Rules
1) Let $R$ be any SQL-invoked routine whose routine descriptor contains the \texttt{<collation name>} of $C$ in the \texttt{<SQL routine body>} or the \texttt{<SQL parameter declaration>}s.
   
   Case:
   
   a) If $R$ is included in an SQL-server module $M$ with no intervening \texttt{<schema definition>}, then let $MN$ be the \texttt{<SQL-server module name>} of $M$. The following \texttt{<drop module statement>} is effectively executed without further Access Rule checking:
   
   \begin{verbatim}
   DROP MODULE MN CASCADE
   \end{verbatim}

   b) Otherwise, let $SN$ be the \texttt{<specific name>} of $R$. The following \texttt{<drop routine statement>} is effectively executed without further Access Rule checking:
   
   \begin{verbatim}
   DROP SPECIFIC ROUTINE SN CASCADE
   \end{verbatim}

2) Let $SSM$ be any SQL-server module whose module descriptor includes the \texttt{<collation name>} of $C$ and let $MN$ be the \texttt{<SQL-server module name>} of $SSM$. The following \texttt{<drop module statement>} is effectively executed without further Access Rule checking:
   
   \begin{verbatim}
   DROP MODULE MN CASCADE
   \end{verbatim}
9.13 <drop translation statement>

Function
Destroy a character translation.

Format
No additional Format items.

Syntax Rules
No additional Syntax Rules.

Access Rules
No additional Access Rules.

General Rules
1) 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:

\[
\text{DROP MODULE } MN \text{ CASCADE}
\]

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

\[
\text{DROP SPECIFIC ROUTINE } SN \text{ CASCADE}
\]
9.14 <assertion definition>

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

Format
No additional Format items.

Syntax Rules
1) Insert after SR4)
   NOTE 12 - <SQL variable name> is also excluded because of the scoping rules for <SQL variable name>.

Access Rules
No additional Access Rules.

General Rules
No additional General Rules.
9.15  <drop assertion statement>

**Function**

Destroy an assertion.

**Format**

No additional Format items.

**Syntax Rules**

No additional Syntax Rules.

**Access Rules**

No additional Access Rules.

**General Rules**

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

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

      DROP MODULE MN CASCADE

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

      DROP SPECIFIC ROUTINE SN CASCADE
9.16 <trigger definition>

Function
Defined triggered SQL-statements.

Format

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

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

Syntax Rules

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

Access Rules
No additional Access Rules.

General Rules
No additional General Rules.
9.17 <drop user-defined ordering statement>

Function
Destroy a user-defined ordering method.

Format
No additional Format items.

Syntax Rules
1) The module descriptor of any SQL-server module.

Access Rules
No additional Access Rules.

General Rules
1) Let R be any SQL-invoked routine that contains P in its <SQL routine body>.
   a) If R is included in an SQL-server module M with no intervening <schema definition>, then let MN be the <SQL-server module name> of M. The following <drop module statement> is effectively executed without further Access Rule checking:

   DROP MODULE MN CASCADE

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

   DROP SPECIFIC ROUTINE SN CASCADE

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

   DROP MODULE MN CASCADE
9.18 <SQL-server module definition>

Function
Define an SQL-server module.

Format

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

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

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

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

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

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

Syntax Rules

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

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

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

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>.
9.18 <SQL-server module definition>

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

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

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

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

Access Rules

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

General Rules

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

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

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

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

   a) The SQL-server module name specified by the <SQL-server module name>.
   b) The descriptor of the character set specified by the <SQL-server module character set specification>.
   c) The default schema name specified by the <SQL-server module schema clause>.
   d) The SQL-server module authorization identifier that corresponds to the authorization identifier that owns the schema identified by the explicit or implicit <schema name> of the <SQL-server module name>.
   e) The list of schema names contained in the <SQL-server module path specification>.
   f) The descriptor of every local temporary table declared in the SQL-server module.
g) The descriptor of every SQL-invoked routine contained in the SQL-server module.

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

Function
Destroy an SQL-server module.

Format

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

Syntax Rules

1) Let MN be the <SQL-server module name> and let M be the SQL-server module identified by MN.
2) M shall be an SQL-server module.
3) If RESTRICT is specified, then the descriptor of M shall not include the descriptor of an SQL-invoked routine that is included in the subject routines of a <routine invocation> that is contained in any of the following:
   a) The <SQL routine body> of any routine descriptor not included in the module descriptor of M.
   b) The <query expression> of any view descriptor.
   c) The <search condition> of any constraint descriptor or assertion descriptor.
   d) Any trigger descriptor.
   e) The module descriptor of any SQL-server module other than M.

Access Rules

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

General Rules

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

      REVOKE EXECUTE ON MODULE MN FROM A CASCADE

2) The descriptor of M is destroyed.
9.20  <drop data type statement>

**Function**
Destroy a user-defined type.

**Format**

No additional Format items.

**Syntax Rules**
1) The module descriptor of any SQL-server module.
2) The module descriptor of any SQL-server module.
3) The module descriptor of any SQL-server module.

**Access Rules**

No additional Access Rules.

**General Rules**

No additional General Rules.
9.21 <SQL-invoked routine>

Function
Define an SQL-invoked routine.

Format

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

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

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

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

Syntax Rules

1) 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.

2) 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.

Access Rules

No additional Access Rules.
General Rules

1) Insert after GR3(x) If the SQL-invoked routine is a schema-level routine, then the schema name of the schema that includes the SQL-invoked routine; otherwise, the SQL-server module name of the SQL-server module that includes the SQL-invoked routine and the schema name of the schema that includes that SQL-server module.
9.22  <drop routine statement>

**Function**
Destroy an SQL-invoked routine.

**Format**
No additional Format items.

**Syntax Rules**
1) [Insert after SR4(b)iv] The module descriptor of any SQL-server module.
2) [Insert after SR5(a)iv] The module descriptor of any SQL-server module.

**Access Rules**
No additional Access Rules.

**General Rules**
No additional General Rules.
9.23  <drop user-defined cast statement>

**Function**

Destroy a user-defined cast.

**Format**

No additional Format items.

**Syntax Rules**

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

**Access Rules**

No additional Access Rules.

**General Rules**

No additional General Rules.
10 Access control

10.1 <grant statement>

Function
Define privileges.

Format
No additional Format items.

Syntax Rules
No additional Syntax Rules.

Access Rules
No additional Access Rules.

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

   GRANT EXECUTE ON M1 TO G WITH GRANT OPTION.

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

**Function**
Destroy privileges.

**Format**

No additional Format items.

**Syntax Rules**

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

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

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

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

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

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

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

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

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

   d) SELECT privileges on every <table reference> contained in a <table expression> or <select list> immediately contained in a <select statement: single row> contained in the <routine body> of any SQL-invoked routine included in SSM.
Access control 67

10.2 <revoke statement>

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

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

g) INSERT privileges on every column
   Case:
   i) Named in the <insert column list> of an <insert statement> contained in the <routine body> of any SQL-invoked routine included in SSM.
   ii) Of the table identified by the <table name> immediately contained in an <insert statement> that does not contain an <insert column list> and that is contained in the <SQL routine body> of any SQL-invoked routine included in SSM.

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

i) DELETE privileges on every table whose name is contained in a <table name> immediately contained in either a <delete statement: positioned> or a <delete statement: searched> contained in the <SQL routine body> of any SQL-invoked routine included in SSM.

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

k) The table/method privilege on every table T1 and every method M such that there is a <method reference> MR contained in the <SQL routine body> of any SQL-invoked routine included in SSM such that T1 is in the scope of the <value expression primary> of MR and M is the subject routine of MR.

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

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

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

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

No additional Access Rules.

General Rules

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

```
DROP MODULE MN CASCADE
```
11 SQL-client modules

11.1 Calls to an <externally-invoked procedure>

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

Syntax Rules
1) Insert into SR2(e)
   CASE_NOT_FOUND_FOR_CASE_STATEMENT_NO_SUBCLASS:
   \begin{verbatim}
   constant SQLSTATE_TYPE :="20000";
   \end{verbatim}
   DATA_EXCEPTION_NULL_VALUE_IN_FIELD_REFERENCE:
   \begin{verbatim}
   constant SQLSTATE_TYPE :="22006";
   \end{verbatim}
   INVALID_SQLSTATE_VALUE_NO_SUBCLASS:
   RESIGNAL_WHEN_HANDLER_NOT_ACTIVE_NO_SUBCLASS:
   \begin{verbatim}
   constant SQLSTATE_TYPE :="0K000";
   \end{verbatim}
   WARNING_RESIGNAL_STATEMENT_WITH_NO_ACTIVE_EXCEPTION:
   UNHANDLED_USER_DEFINED_EXCEPTION_NO_SUBCLASS:
   \begin{verbatim}
   constant SQLSTATE_TYPE :="45000";
   \end{verbatim}

Access Rules
No additional Access Rules.

General Rules
No additional General Rules.
11.2 <SQL procedure statement>

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

Format

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

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

<SQL control statement> ::=  
   !! All alternatives from ISO/IEC 9075-2  
   !! All alternatives from ISO/IEC 9075-5  
   <assignment statement>  
   <compound statement>  
   <case statement>  
   <if statement>  
   <iterate statement>  
   <leave statement>  
   <loop statement>  
   <while statement>  
   <repeat statement>  
   <for statement>

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

Syntax Rules

1) 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) S is a <compound statement> and S contains an <SQL variable declaration> that specifies a <default option> that contains a <datetime value function>, CURRENT_USER, CURRENT_ROLE, SESSION_USER, or SYSTEM_USER.

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

Access Rules

No additional Access Rules.
General Rules

No additional General Rules.

Conformance Rules

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

12.1 <open statement>

Function
Open a cursor.

Format

No additional Format items.

Syntax Rules

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

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.
12.2 <fetch statement>

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

Format
No additional Format items.

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

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

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

Access Rules
No additional Access Rules.

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

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

Function
Close a cursor.

Format
No additional Format items.

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

Access Rules
No additional Access Rules.

General Rules
No additional General Rules.
12.4 <select statement: single row>

Function
Retrieve values from a specified row of a table.

Format
No additional Format items.

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

Function
Delete a row of a table.

Format

No additional Format items.

Syntax Rules

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

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.
12.6  <update statement: positioned>

**Function**
Update a row of a table.

**Format**

*No additional Format items.*

**Syntax Rules**

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

**Access Rules**

*No additional Access Rules.*

**General Rules**

*No additional General Rules.*
12.7 <temporary table declaration>

**Function**

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

2) 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
12.7 <temporary table declaration>

U.T. Within the SQL-client module, any reference to MODULE.T that is not contained in an
<SQL schema statement> is equivalent to a reference to U.T.

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

13.1 <compound statement>

Function
Specify a statement that groups other statements together.

Format

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

<beginning label> ::= <statement label>
<ending label> ::= <statement label>
<statement label> ::= <identifier>

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

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

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

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

<SQL statement list> ::= <terminated SQL statement>...
<terminated SQL statement> ::= <SQL procedure statement> <semicolon>
13.1 <compound statement>

Syntax Rules

1) Let CS be the <compound statement>.

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

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

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

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

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

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

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

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

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

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

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

13) The scope of an <SQL variable name> of an <SQL variable declaration> simply contained in a <local declaration list> 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 list> 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> 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.
13.1 <compound statement>

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

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

Access Rules

None.

General Rules

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

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

3) Let N be the number of <SQL procedure statement>s contained in the <SQL statement list> that is immediately contained in CS without an intervening <SQL control statement>. For i ranging from 1 (one) to N:
   a) Let S_i be the i-th such <SQL procedure statement>.
   b) The General Rules of Subclause 13.5, "<SQL procedure statement>", are evaluated with S_i as the executing statement.
   c) If the execution of S_i terminates with exception conditions or completion conditions other than successful completion, then:
      i) The following <resignal statement> is effectively executed without further Syntax Rule checking:
         
         RESIGNAL
      ii) If there are unhandled exception conditions or completion conditions other than successful completion at the completion of the execution of a handler (if any), then the execution of CS is terminated immediately.

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

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

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

5) The SQL variables, cursors, and handlers specified in <local declaration list>, the <local cursor declaration list>, and the <local handler declaration list> of CS are destroyed.
6) If CS specifies ATOMIC, then all savepoints established during the execution of CS are destroyed.

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

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

Format

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

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

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

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

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

Syntax Rules

1) Let HD be the <handler declaration>.

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

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

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

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

6) SQLEXCEPTION, SQLWARNING, and NOT FOUND correspond to SQLSTATE class values corresponding to categories X, W, and N, respectively, in Subclause 22.1, “SQLSTATE”, in ISO/IEC 9075-2.
13.2 <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.

3) When H is activated,

   Case:

   a) If HD specifies CONTINUE, then:

      i) HA is executed.

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

             RESIGNAL

             Otherwise, HA completes with completion condition successful completion and control is returned to the SQL-statement following the executing SQL-statement that raised the condition in CC. If the SQL-statement that raised the condition is an <SQL control
statement>, then “the following SQL-statements” do not include the <SQL procedure
statement>s contained within the <SQL control statement>.

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

b) If HD specifies EXIT, then:
   i) HA is executed.
   ii) If there is an unhandled condition other than successful completion at the completion of
       HA, then the following <resignal statement> is effectively executed:

       RESIGNAL

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

c) If HD specifies UNDO, then:
   i) All changes made to SQL-data or schemas by the execution of SQL-statements contained
      in the <SQL statement list> of CS and any <SQL procedure statement>s triggered by
      the execution of any such statements are canceled.
   ii) HA is executed.
   iii) If there is an unhandled condition other than successful completion at the completion of
        HA, then the following <resignal statement> is effectively executed:

        RESIGNAL

        Otherwise, HA completes with completion condition successful completion and control is
        returned to the end of CS.
13.3  `<condition declaration>`

**Function**
Declare a condition name and an optional corresponding SQLSTATE value.

**Format**

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

**Syntax Rules**

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

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

**Access Rules**

None.

**General Rules**

1) `<condition name>` is considered to be defined for the SQLSTATE value specified by `<sqlstate value>`.
13.4 `<SQL variable declaration>`

**Function**
Declare one or more variables.

**Format**

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

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

**Syntax Rules**

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

**Access Rules**

None.

**General Rules**

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

```
SET SV = DV
```
13.5 <assignment statement>

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

Format

<assignment statement> ::= 
  SET <assignment target> <equals operator> <assignment source>

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

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

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

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

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

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

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

Syntax Rules

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

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

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

   SET TS = TS.FN (AS)
13.5 <assignment statement>

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

4) If <assignment target> is a <modified field reference> FR, then
   a) Let F be the field identified by <field name> simply contained in <assignment target> and not simply contained in <modified field target>.
   b) Let AS be the <assignment source>.
   c) The Syntax Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2 are applied to F and AS as TARGET and VALUE, respectively.

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

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

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

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

Access Rules

None.

General Rules

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

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

3) If <assignment target> is a <target specification> that is a new transition variable column reference, then let C be the column identified by the <column reference> and let R be the row that is to be replaced by that transition variable. For each transition variable TV that is a replacement for a subrow of R or for a superrow of R in a table in which C is a column, the value of <assignment source> is assigned to TV.C according to the General Rules of Subclause 9.2, "Store assignment", in ISO/IEC 9075-2, with <assignment source> and TV.C as VALUE and TARGET, respectively.
4) If <assignment target> is a <modified field reference> FR, then let T be the <target specification> simply contained in FR. Let F₁ 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 F₁ is the null value, then an exception condition is raised: data exception — null value in field reference.

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

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

Format

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

<simple case statement> ::=  
    CASE <simple case operand 1> 
    <simple case statement when clause>... 
    [ <case statement else clause> ] 
    END CASE

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

<simple case statement when clause> ::=  
    WHEN <simple case operand 2> 
    THEN <SQL statement list>

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

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

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

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

Syntax Rules

1) If a <case statement> specifies a <simple case statement>, then let SC01 be the <simple case operand 1>:
   a) SC01 shall not generally contain a <routine invocation> whose subject routines include an SQL-invoked routine that is possibly non-deterministic or that possibly modifies SQL-data.
   b) The declared type of each <simple case operand 2> SC02 shall be comparable with the declared type of SC01.
   c) The <simple case statement> is equivalent to a <searched case statement> in which each <searched case statement when clause> specifies a <search condition> of the form:

                 SC01 = SC02
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 Si be the i-th such <SQL procedure statement>.

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

   c) If the execution of Si terminates with an unhandled exception condition, then the execution of the <case statement> is terminated with that condition.
13.7 `<if statement>`

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

**Format**

```plaintext
<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:

```plaintext
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

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

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

None.

General Rules

1) Case:
   a) If the <search condition> immediately contained in the <if statement> evaluates to true, then let SL be the <SQL statement list> immediately contained in the <if statement then clause>.
   b) Otherwise, if an <if statement else clause> is specified, then let SL be the <SQL statement list> immediately contained in the <if statement else clause>. 
      NOTE 20 - “Otherwise” means that the <search condition> immediately contained in the <if statement> evaluates to false or to unknown.

2) Let N be the number of <SQL procedure statement>s simply contained in SL without an intervening <SQL control statement>. For i ranging from 1 to N:
   a) Let $S_i$ be the i-th such <SQL procedure statement>.
   b) The General Rules of Subclause 13.5, "<SQL procedure statement>", in ISO/IEC 9075-2, are evaluated with $S_i$ as the executing statement.
   c) If the execution of $S_i$ terminates with an unhandled exception condition, then the execution of the <if statement> is terminated and the condition remains active.
13.8 <iterate statement>

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

Format

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

Syntax Rules

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

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

Access Rules

None.

General Rules

1) The execution of SSL is terminated.
   
   NOTE 21 – If the iteration condition for IS is true or if IS does not have an iteration condition, then the next iteration of SSL commences immediately. If the iteration condition for IS is false, then there is no next iteration of SSL.
13.9  \textit{<leave statement>}

\textbf{Function}

Continue execution by leaving a labeled statement.

\textbf{Format}

\begin{verbatim}
<leave statement> ::= LEAVE <statement label>
\end{verbatim}

\textbf{Syntax Rules}

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

\textbf{Access Rules}

None.

\textbf{General Rules}

1) For every \textit{<compound statement>} \textit{CS} that is contained in \textit{S} and that contains the \textit{<leave statement>}:

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

       \begin{verbatim}
       CLOSE CR
       \end{verbatim}

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

2) The execution of \textit{S} is terminated.
13.10 <loop statement>

Function
Repeat the execution of a statement.

Format

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

Syntax Rules
1) Let LS be the <loop statement>.
2) If LS is contained in another <SQL control statement> and LS does not specify a <beginning label>, then an implementation-dependent <beginning label> that is not equivalent to any other <statement label> contained in the outermost containing <SQL control statement> is implicit.
3) If <ending label> is specified, then a <beginning label> shall be specified that is equivalent to <ending label>.
4) The scope of the <beginning label> is LS excluding every <SQL schema statement> contained in LS. <beginning label> shall not be equivalent to any other <beginning label> contained in LS excluding every <SQL schema statement> contained in LS.

Access Rules
None.

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

BEGIN NOT ATOMIC SSL END

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

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

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

Format

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

Syntax Rules
1) Let WS be the <while statement>.

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

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

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

Access Rules

None.

General Rules
1) The <search condition> is evaluated.

2) Case:
   a) If the <search condition> evaluates to false or unknown, then execution of WS is terminated.
   b) Let SSL be the <SQL statement list> and let CCS be the <compound statement>

BEGIN NOT ATOMIC SSL END

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

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

Function
Repeat the execution of a statement.

Format

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

Syntax Rules

1) Let RS be the <repeat statement>.

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

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

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

Access Rules
None.

General Rules

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

BEGIN NOT ATOMIC SSL END

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

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

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

Function
Execute a statement for each row of a table.

Format

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

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

Syntax Rules

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

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

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

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

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

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

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

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

   c) Let CS be the explicit or implicit <cursor sensitivity>. 

102 Persistent Stored Modules (SQL/PSM)
The `<for statement>` is equivalent to:

```sql
BL: BEGIN NOT ATOMIC

FLVN: BEGIN NOT ATOMIC
    DECLARE CN CS CURSOR FOR
    SELECT ROW ( Q.V1, Q.V2, ... , Q.Vn )
    FROM ( FCS ) A S
    DECLARE FLVN ROW ( V1 DT1, V2 DT2, ..., Vn DTn );
    DECLARE AT_END BOOLEAN DEFAULT FALSE;
    DECLARE NOT_FOUND CONDITION FOR SQLSTATE '02000';
    BEGIN NOT ATOMIC
    DECLARE CONTINUE HANDLER FOR NOT_FOUND
        SET AT_END = TRUE;
    OPEN CN;
    FETCH CN INTO FLVN;
    WHILE NOT AT_END DO
        SLL;
        BEGIN NOT ATOMIC
            FETCH CN INTO FLVN;
        END;
        END WHILE;
    CLOSE CN;
    END FLVN;
    END BL
```

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

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

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

### Access Rules

None.

### General Rules

None.
14 Dynamic SQL

14.1 <prepare statement>

Function
Prepare a statement for execution.

Format
No additional Format items.

Syntax Rules
No additional Syntax Rules.

Access Rules
No additional Access Rules.

General Rules
1) If DP is the <assignment target> simply contained in an <assignment statement> AS, then
   Case:
   a) If the <assignment source> immediately contains a <null specification>, then DT is unde-
      fined.
   b) Otherwise, DT is the declared type of the <value expression> simply contained in the
      <assignment source> of AS.

2) If DP is the <value expression> simply contained in an <assignment source> in an <assignment statement> AS or if DP represents the value of a subfield SF of the
decided type of such a <value expression>, then let RT be the declared type of the <assignment target> simply contained in AS.
   Case:
   a) If DP is the <value expression> simply contained in the <assignment source>, then DT is
      RT.
   b) Otherwise, DT is the declared type of the subfield of RT that corresponds to SF.

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

Case:

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

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

15.1 <embedded SQL host program>

Function
Specify an <embedded SQL host program>.

Format
No additional Format items.

Syntax Rules

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

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

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

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

5) M contains one <SQL variable declaration> for each <SQL variable declaration> contained in H. Each <SQL variable declaration> of M is a copy of the corresponding <SQL variable declaration> of H.

6) M contains one <handler declaration> for each <handler declaration> contained in H. Each <handler declaration> of M is a copy of the corresponding <handler declaration> of H.

7) Each <embedded SQL statement> that contains a <declare cursor>, a <dynamic declare cursor>, an <SQL variable declaration>, an <SQL-invoked routine>, or a <temporary table declaration> has been deleted, and every <embedded SQL statement> that contains an <embedded exception declaration> has been replaced with statements of the host language that will have the effect specified by the General Rules of Subclause 16.2, "<LB>LB>embedded exception declaration>".
Access Rules

No additional Access Rules.

General Rules

No additional General Rules.
16  Diagnostics management

16.1  <get diagnostics statement>

Function
Get exception or completion condition information from the diagnostics area.

Format

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

Syntax Rules

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

<table>
<thead>
<tr>
<th>&lt;identifier&gt;</th>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;statement information item name&gt;s</td>
<td></td>
</tr>
<tr>
<td>All alternatives from ISO/IEC 9075-2</td>
<td></td>
</tr>
<tr>
<td>All alternatives from ISO/IEC 9075-5</td>
<td></td>
</tr>
<tr>
<td>&lt;condition information item name&gt;s</td>
<td></td>
</tr>
<tr>
<td>All alternatives from ISO/IEC 9075-2</td>
<td></td>
</tr>
<tr>
<td>All alternatives from ISO/IEC 9075-5</td>
<td></td>
</tr>
<tr>
<td>CONDITION_IDENTIFIER</td>
<td>character varying (L)</td>
</tr>
</tbody>
</table>

Access Rules

No additional Access Rules.
16.1 <get diagnostics statement>

General Rules

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

<table>
<thead>
<tr>
<th>SQL-statement</th>
<th>Identifier</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>All alternatives from ISO/IEC 9075-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>All alternatives from ISO/IEC 9075-5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt;alter module statement&gt;</td>
<td>ALTER MODULE</td>
<td>95</td>
</tr>
<tr>
<td>&lt;assignment statement&gt;</td>
<td>ASSIGNMENT</td>
<td>5</td>
</tr>
<tr>
<td>&lt;case statement&gt;</td>
<td>CASE</td>
<td>86</td>
</tr>
<tr>
<td>&lt;compound statement&gt;</td>
<td>BEGIN END</td>
<td>12</td>
</tr>
<tr>
<td>&lt;drop module statement&gt;</td>
<td>DROP MODULE</td>
<td>28</td>
</tr>
<tr>
<td>&lt;for statement&gt;</td>
<td>FOR</td>
<td>46</td>
</tr>
<tr>
<td>&lt;handler declaration&gt;</td>
<td>HANDLER</td>
<td>87</td>
</tr>
<tr>
<td>&lt;if statement&gt;</td>
<td>IF</td>
<td>88</td>
</tr>
<tr>
<td>&lt;leave statement&gt;</td>
<td>LEAVE</td>
<td>89</td>
</tr>
<tr>
<td>&lt;loop statement&gt;</td>
<td>LOOP</td>
<td>90</td>
</tr>
<tr>
<td>&lt;resignal statement&gt;</td>
<td>RESIGNAL</td>
<td>91</td>
</tr>
<tr>
<td>&lt;repeat statement&gt;</td>
<td>REPEAT</td>
<td>95</td>
</tr>
<tr>
<td>&lt;signal statement&gt;</td>
<td>SIGNAL</td>
<td>92</td>
</tr>
<tr>
<td>&lt;SQL-server module definition&gt;</td>
<td>CREATE MODULE</td>
<td>51</td>
</tr>
<tr>
<td>&lt;SQL variable declaration&gt;</td>
<td>DECLARE VARIABLE</td>
<td>96</td>
</tr>
<tr>
<td>&lt;temporary table declaration&gt;</td>
<td>TEMPORARY TABLE</td>
<td>93</td>
</tr>
<tr>
<td>&lt;while statement&gt;</td>
<td>WHILE</td>
<td>97</td>
</tr>
</tbody>
</table>

1) If the value of the RETURNED_SQLSTATE corresponds to unhandled user-defined exception, then the value of CONDITION_IDENTIFIER is the <condition name> of the user-defined exception.
16.2  <signal statement>

Function
Signal an exception condition.

Format
<signal statement> ::= 
  SIGNAL <signal value>
  [ <set signal information> ]
<signal value> ::= 
  <condition name>
  | <sqlstate value>
<set signal information> ::= 
  SET <signal information item list>
<signal information item list> ::= 
  <signal information item> [ { <comma> <signal information item> }... ]
<signal information item> ::= 
  <condition information item name> <equals operator> <simple value specification>

Syntax Rules
1) Case:
   a) If <signal value> immediately contains <condition name>, then:
      i) Let CN be the <condition name> contained in the <signal statement>.
      ii) CN shall be contained within the scope of one or more <condition name>s whose associated <condition declaration> includes a condition whose <identifier> is CN. If there is more than one such <condition name>, then the one with the innermost scope is specified. Let C be that condition.
   b) Otherwise, let C be the SQLSTATE value defined by <sqlstate value> and let CN be a zero-length string.

2) <condition information item name> shall not specify CONDITION_NUMBER, RETURNED_SQLSTATE, MESSAGE_LENGTH, or MESSAGE_OCTET_LENGTH. No other alternative for <condition information item name> shall be specified more than once in <set signal information>.

3) The data type of a <condition information item name> contained in <signal information item> shall be the data type specified in Table 2, "<identifier>s for use with <get diagnostics statement>".
Access Rules

None.

General Rules

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

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

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

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

    RESIGNAL SSI

112 Persistent Stored Modules (SQL/PSM)
16.3 <resignal statement>

Function
Resignal an exception condition.

Format

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

Syntax Rules
1) Let RS be the <resignal statement>.
2) If <signal value> is specified, then
   Case:
   a) If <signal value> immediately contains <condition name>, then:
      i) Let CN be the <condition name> contained in RS.
      ii) CN shall be contained within the scope of one or more <condition name>s whose asso-
          ciated <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) If <set signal information> is specified, then for each <signal information item> in <set signal
   information>:
   a) In the first condition area in the diagnostics area, the information field identified by the
      <signal information name> is set to contain the value of the <simple value specification>.
   b) If the <signal information name> specifies MESSAGE_TEXT, then the information fields
      MESSAGE_LENGTH and MESSAGE_OCTET_LENGTH in the diagnostics area are set to
      contain the length and the length in octets of the value of the <simple value specification>,
      respectively.
2) Case:
   a) If the first condition in the diagnostics area has no RETURN_SQLSTATE value and the value of the CONDITION_IDENTIFIER is a zero-length string, then an exception condition is raised: resignal when handler not active.
   
b) Otherwise, let N be the value of the statement information field NUMBER in the diagnostics area before the execution of RS.
      Case:
      i) If <signal value> is not specified, then the diagnostics area remains unchanged.
      ii) If <signal value> is specified, then the statement information field NUMBER in the diagnostics area is incremented. All existing condition areas are stacked such that the i-th condition area is placed at the position of the i+1-st condition area in the diagnostics area. If the maximum number of condition areas for the diagnostics area is exceeded, then the value of the statement information field NUMBER contains the number of exception or completion conditions of the SQL-statement that raised the condition plus those raised by RS, and the value of the statement information field MORE is 'Y'.

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

3) Case:
   a) If the first condition in the diagnostics area has a RETURNED_SQLSTATE value, then:
      i) Let S be that value.
      ii) If a handler H is the most appropriate handler for S, then S is activated.
      iii) If no handler is activated and S identifies an SQLSTATE value associated with an exception condition, then this is an unhandled exception condition and the <SQL procedure statement> that resulted in execution of RS is terminated with this exception condition.

      NOTE 25 – If S identifies an SQLSTATE value associated with a completion condition, then this is an unhandled completion condition and processing continues without altering the flow of control.

   b) Otherwise:
      i) Let E be the value of the CONDITION_IDENTIFIER field of the first condition in the diagnostics area.
      ii) If a handler H is the most appropriate handler for E, then S is activated.
      iii) If no handler is activated, then this is an unhandled exception condition and the <SQL procedure statement> that resulted in execution of RS is terminated with the exception condition unhandled user-defined exception.
17 Information Schema

17.1 MODULE_COLUMN_USAGE view

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

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

GRANT SELECT ON TABLE MODULE_COLUMN_USAGE
TO PUBLIC WITH GRANT OPTION;
17.2 MODULE_PRIVILEGES view

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

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

GRANT SELECT ON TABLE MODULE_PRIVILEGES
    TO PUBLIC WITH GRANT OPTION;
17.3 MODULE_TABLE_USAGE view

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

Definition
CREATE VIEW MODULE_TABLE_USAGE AS
    SELECT MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME,
    TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME
    FROM DEFINITION_SCHEMA.MODULE_TABLE_USAGE
    JOIN
    DEFINITION_SCHEMA.SCHEMA S
    ON ( ( TABLE_CATALOG, TABLE_SCHEMA ) =
    ( S.CATALOG_NAME, S.SCHEMA_NAME ) )
    WHERE ( SCHEMA_OWNER = CURRENT_USER
    OR
    SCHEMA_OWNER IN
    ( SELECT ROLE_NAME
    FROM ENABLED_ROLES ) )
    AND
    MODULE_CATALOG =
    ( SELECT CATALOG_NAME
    FROM INFORMATION_SCHEMA_CATALOG_NAME )
    GRANT SELECT ON TABLE MODULE_TABLE_USAGE
    TO PUBLIC WITH GRANT OPTION;
17.4 MODULES view

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

**Definition**

```
CREATE VIEW MODULES AS
    SELECT MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME,
           DEFAULT_CHARACTER_SET_CATALOG, DEFAULT_CHARACTER_SET_SCHEMA,
           DEFAULT_CHARACTER_SET_NAME,
           DEFAULT_SCHEMA_CATALOG, DEFAULT_SCHEMA_NAME,
           CASE
               WHEN EXISTS ( 
                   SELECT * FROM DEFINITION_SCHEMA.SCHEMATA AS S 
                   WHERE ( MODULE_CATALOG, MODULE_SCHEMA ) = 
                   ( S.CATALOG_NAME, S.SCHEMA_NAME ) 
                   AND
                   ( SCHEMA_OWNER IN 
                     ( "PUBLIC", CURRENT_USER ) 
                   OR
                   ( SCHEMA_OWNER IN 
                     ( SELECT ROLE_NAME 
                         FROM ENABLED_ROLES ) ) ) 
               THEN MODULE_DEFINITION
               ELSE NULL
               END AS MODULE_DEFINITION,
           MODULE_AUTHORIZATION, SQL_PATH, MODULE_CREATED, MODULE_LAST_ALTERED
    FROM DEFINITION_SCHEMA.MODULES
    WHERE ( MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME ) IN
        ( SELECT MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME
          FROM DEFINITION_SCHEMA.MODULE_PRIVILEGES
          WHERE ( SCHEMA_OWNER IN 
            ( "PUBLIC", CURRENT_USER ) 
          OR
          SCHEMA_OWNER IN 
            ( SELECT ROLE_NAME 
              FROM ENABLED_ROLES ) ) )

    AND
    MODULE_CATALOG = 
        ( SELECT CATALOG_NAME 
          FROM INFORMATION_SCHEMA_CATALOG_NAME )

GRANT SELECT ON TABLE MODULES
    TO PUBLIC WITH GRANT OPTION;
```
17.5 ROLE_MODULE_GRANTS view

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

Definition
CREATE VIEW ROLE_MODULE_GRANTS AS
  SELECT GRANTOR, GRANTEE, MODULE_CATALOG,
         MODULE_SCHEMA, MODULE_NAME, PRIVILEGE_TYPE,
         IS_GRANTABLE
  FROM DEFINITION_SCHEMA.MODULE_PRIVILEGES
  WHERE ( GRANTEE IN
      ( SELECT ROLE_NAME
          FROM ENABLED_ROLES
      )
    OR
    GRANTOR IN
    ( SELECT ROLE_NAME
      FROM ENABLED_ROLES
    )
  )
  AND
  MODULE_CATALOG =
  ( SELECT CATALOG_NAME
    FROM INFORMATION_SCHEMA_CATALOG_NAME
  );
GRANT SELECT ON TABLE ROLE_MODULE_GRANTS
TO PUBLIC WITH GRANT OPTION;

Conformance Rules
1) Without Feature T322, “Extended Roles”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROLE_MODULE_GRANTS.
17.6 Short name views

Function
Provide alternative views that use only identifiers that do not require Feature F391, “Long identifiers”.

Definition
CREATE VIEW MODULE_COL_USAGE
     ( ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME,
      TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
      COLUMN_NAME) AS
    SELECT ROUTINE_CATALOG, ROUTINE_SCHEMA, ROUTINE_NAME,
      TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
      COLUMN_NAME
    FROM INFORMATION_SCHEMA.MODULE_COLUMN_USAGE;
GRANT SELECT ON TABLE MODULE_COL_USAGE
TO PUBLIC WITH GRANT OPTION;

CREATE VIEW MODULES_S
     ( MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME,
      DEF_CHAR_SET_CAT, DEF_CHAR_SET_SCH, DEF_CHAR_SET_NAME,
      DEF_SCHEMA_CATALOG, DEFAULT_SCHEMA, MODULE_DEFINITION,
      MODULE_AUTH, SQL_PATH, CREATED,
      ALTERED) AS
    SELECT MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME,
      DEFAULT_CHARACTER_SET_CATALOG, DEFAULT_CHARACTER_SET_SCHEMA,
      DEFAULT_CHARACTER_SET_NAME,
      DEFAULT_SCHEMA_CATALOG, DEFAULT_SCHEMA, MODULE_DEFINITION,
      MODULE_AUTHORIZATION, SQL_PATH, CREATED,
      LAST_ALTERED
    FROM INFORMATION_SCHEMA.MODULES;
GRANT SELECT ON TABLE MODULES_S
TO PUBLIC WITH GRANT OPTION;
18 Definition Schema

18.1 MODULE_COLUMN_USAGE base table

Function
The MODULE_COLUMN_USAGE table has one row for each column of a table that is explicitly or implicitly identified in the <query expression> of the view being described.

Definition
CREATE TABLE MODULE_COLUMN_USAGE (  
  MODULE_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER,  
  MODULE_SCHEMA INFORMATION_SCHEMA.SQL_IDENTIFIER,  
  MODULE_NAME INFORMATION_SCHEMA.SQL_IDENTIFIER,  
  TABLE_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER,  
  TABLE_SCHEMA INFORMATION_SCHEMA.SQL_IDENTIFIER,  
  TABLE_NAME INFORMATION_SCHEMA.SQL_IDENTIFIER,  
  COLUMN_NAME INFORMATION_SCHEMA.SQL_IDENTIFIER,  
  CONSTRAINT MODULE_COLUMN_USAGE_PRIMARY_KEY  
    PRIMARY KEY ( MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME, 
                  TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME ),

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

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

  CONSTRAINT MODULE_COLUMN_USAGE_CHECK_MODULE_TABLE_USAGE  
    CHECK ( MODULE_CATALOG <> 
            ANY ( SELECT MODULE_CATALOG 
                  FROM SCHEMATA )  
            OR  
            ( MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME, 
              TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME ) IN  
            ( SELECT MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME 
              FROM MODULE_TABLE_USAGE ) ) )

)

Description
1) The values of MODULE_CATALOG, MODULE_SCHEMA, and MODULE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the SQL-server module being described.
2) The values of TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, and COLUMN_NAME are the catalog name, unqualified schema name, qualified identifier, and identifier respectively, of a column that is referenced in the SQL-server module being described.
18.2 MODULE_PRIVILEGES base table

Function
The MODULE_PRIVILEGES table has one row for each execute privilege descriptor on an SQL-server module. It effectively contains a representation of the execute privilege descriptors.

Definition
CREATE TABLE MODULE_PRIVILEGES (
    GRANTOR INFORMATION_SCHEMA.SQL_IDENTIFIER,
    GRANTEE INFORMATION_SCHEMA.SQL_IDENTIFIER,
    MODULE_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER,
    MODULE_SCHEMA INFORMATION_SCHEMA.SQL_IDENTIFIER,
    MODULE_NAME INFORMATION_SCHEMA.SQL_IDENTIFIER,
    PRIVILEGE_TYPE INFORMATION_SCHEMA.CHARACTER_DATA
     CONSTRAINT MODULE_PRIVILEGES_TYPE_CHECK
     CHECK ( PRIVILEGE_TYPE = 'EXECUTE' ),
    IS_GRANTABLE INFORMATION_SCHEMA.CHARACTER_DATA
     CONSTRAINT MODULE_PRIVILEGES_GRANTABLE_NOT_NULL
     NOT NULL
     CONSTRAINT MODULE_PRIVILEGES_GRANTABLE_CHECK
     CHECK ( IS_GRANTABLE
     IN ( 'YES', 'NO' ) ),
     CONSTRAINT MODULE_PRIVILEGES_PRIMARY_KEY
     PRIMARY KEY ( GRANTOR, GRANTEE, MODULE_CATALOG, MODULE_SCHEMA,
                    MODULE_NAME ),
     CONSTRAINT MODULE_PRIVILEGES_FOREIGN_KEY_TABLES
     FOREIGN KEY ( MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME )
     REFERENCES MODULES,
     CONSTRAINT MODULE_PRIVILEGE_GRANTOR_CHECK
     CHECK ( GRANTOR IN
     ( SELECT ROLE_NAME
     FROM ROLES )
     OR
     GRANTOR IN
     ( SELECT USER_NAME
     FROM USERS ) ),
     CONSTRAINT MODULE_PRIVILEGE_GRANTEE_CHECK
     CHECK ( GRANTEE IN
     ( SELECT ROLE_NAME
     FROM ROLES )
     OR
     GRANTEE IN
     ( SELECT USER_NAME
     FROM USERS ) )
)

Description
1) The value of GRANTOR is the <authorization identifier> of the user or role who granted execute privileges, on the SQL-server module identified by MODULE_CATALOG, MODULE_SCHEMA, and MODULE_NAME, to the user or role identified by the value of GRANTEE for the privilege being described.

2) The value of GRANTEE is the <authorization identifier> of some user or role, or "PUBLIC" to indicate all users, to whom the privilege being described is granted.
3) The values of MODULE_CATALOG, MODULE_SCHEMA, and MODULE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the SQL-server module on which the privilege being described has been granted.

4) The values of PRIVILEGE_TYPE have the following meanings:

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

5) The values of IS_GRANTABLE have the following meanings:

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

   NO       The privilege being described was not granted WITH GRANT OPTION and is thus not grantable.
## 18.3 MODULE_TABLE_USAGE base table

### Function

The MODULE_TABLE_USAGE table has one row for each table table identified by a `<table name>` simply contained in a `<table reference>` that is contained in the `<query expression>` of a view.

### Definition

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

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

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

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

### Description

1) The values of MODULE_CATALOG, MODULE_SCHEMA, and MODULE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the SQL-server module being described.

2) The values of TABLE_CATALOG, TABLE_SCHEMA, and TABLE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of a table that is referenced in the SQL-server module being described.
18.4 MODULES base table

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

Definition
CREATE TABLE MODULES (  
    MODULE_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    MODULE_SCHEMA INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    MODULE_NAME INFORMATION_SCHEMA.SQL_IDENTIFIER,  
    DEFAULT_CHARACTER_SET_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER  
        CONSTRAINT MODULES_DEFAULT_CHARACTER_SET_CATALOG_NOT_NULL  
        NOT NULL,  
    DEFAULT_CHARACTER_SET_SCHEMA INFORMATION_SCHEMA.SQL_IDENTIFIER  
        CONSTRAINT MODULES_DEFAULT_CHARACTER_SET_SCHEMA_NOT_NULL  
        NOT NULL,  
    DEFAULT_CHARACTER_SET_NAME INFORMATION_SCHEMA.SQL_IDENTIFIER  
        CONSTRAINT MODULES_DEFAULT_CHARACTER_SET_NAME_NOT_NULL  
        NOT NULL,  
    DEFAULT_SCHEMA_CATALOG INFORMATION_SCHEMA.SQL_IDENTIFIER  
        CONSTRAINT MODULES_DEFAULT_SCHEMA_CATALOG_NOT_NULL  
        NOT NULL,  
    DEFAULT_SCHEMA_NAME INFORMATION_SCHEMA.SQL_IDENTIFIER  
        CONSTRAINT MODULES_DEFAULT_SCHEMA_NAME_NOT_NULL  
        NOT NULL,  
    MODULE_DEFINITION INFORMATION_SCHEMA.CHARACTER_DATA,  
    MODULE.Authorization INFORMATION_SCHEMA.SQL_IDENTIFIER  
        CONSTRAINT AUTHORIZATION_FOREIGN_KEY_USERS REFERENCES USERS,  
    SQL_PATH INFORMATION_SCHEMA.CHARACTER_DATA,  
    CREATED INFORMATION_SCHEMA.TIME_STAMP,  
    LAST_ALTERED INFORMATION_SCHEMA.TIME_STAMP,  
    CONSTRAINT MODULES_PRIMARY_KEY  
        PRIMARY KEY ( MODULE_CATALOG, MODULE_SCHEMA, MODULE_NAME ),  
    CONSTRAINT MODULES_FOREIGN_KEY_SCHEMATA  
        FOREIGN KEY ( MODULE_CATALOG, MODULE_SCHEMA )  
        REFERENCES SCHEMATA,  
    CONSTRAINT MODULES_FOREIGN_KEY_CHARACTER_SETS  
        FOREIGN KEY ( DEFAULT_CHARACTER_SET_CATALOG, DEFAULT_CHARACTER_SET_SCHEMA,  
                      DEFAULT_CHARACTER_SET_NAME )  
        REFERENCES CHARACTER_SETS,  
    CONSTRAINT MODULES_FOREIGN_KEY_DEFAULT_SCHEMA_SCHEMATA  
        FOREIGN KEY ( DEFAULT_SCHEMA_CATALOG, DEFAULT_SCHEMA_NAME )  
        REFERENCES SCHEMATA  
)  

Description
1) The values of MODULE_CATALOG, MODULE_SCHEMA, and MODULE_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the module name of the SQL-server module being described.

2) The values of DEFAULT_CHARACTER_SET_CATALOG, DEFAULT_CHARACTER_SET_SCHEMA, and DEFAULT_CHARACTER_SET_NAME are the catalog name, unqualified schema name, and qualified identifier, respectively, of the character set identified by the implicit or explicit <SQL-server module character set specification>.
3) The values of DEFAULT_SCHEMA_CATALOG, and DEFAULT_SCHEMA_NAME are the catalog name, and unqualified schema name, respectively, of the schema identified by the implicit or explicit <SQL-server module schema clause>.

4) Case:
   a) If the character representation of the <SQL-server module definition> that defined the SQL-server module being described can be represented without truncation, then the value of MODULE_DEFINITION is that character representation.
   b) Otherwise, the value of MODULE_DEFINITION is the null value.

   NOTE 26 – Any implicit <column reference>s that were contained in the <SQL-server module definition> are replaced by explicit <column reference>s in MODULE_DEFINITION.

5) Case:
   a) If AUTHORIZATION was specified in <module authorization clause> in the SQL-server module being described, then the value of MODULE_AUTHORIZATION is <module authorization identifier>.
   b) Otherwise, the value of MODULE_AUTHORIZATION is the null value.

6) Case:
   a) If <SQL-server module path specification> was specified in the <SQL-server module definition> that defined the SQL-server module described by this row and the character representation of the <SQL-server module path specification> can be represented without truncation, then the value of SQL_PATH is that character representation.
   b) Otherwise, the value of SQL_PATH is the null value.

7) The value of CREATED is the value of CURRENT_TIMESTAMP at the time when the SQL-server module being described was created.

8) The value of LAST_ALTERED is the value of CURRENT_TIMESTAMP at the time that the SQL-server module being described was last altered. This value is identical to the value of CREATED for SQL-server modules that have never been altered.
19 Status codes

19.1 SQLSTATE

<table>
<thead>
<tr>
<th>Category</th>
<th>Condition</th>
<th>Class</th>
<th>Subcondition</th>
<th>Subclass</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>case not found for case statement</td>
<td>20</td>
<td>(no subclass)</td>
<td>000</td>
</tr>
<tr>
<td>X</td>
<td>data exception</td>
<td>22</td>
<td>(no subclass)</td>
<td>000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>null value in field reference</td>
<td>006</td>
</tr>
<tr>
<td>X</td>
<td>resignal when handler not active</td>
<td>0K</td>
<td>(no subclass)</td>
<td>000</td>
</tr>
<tr>
<td>W</td>
<td>warning</td>
<td>01</td>
<td>(no subclass)</td>
<td>000</td>
</tr>
<tr>
<td>X</td>
<td>unhandled user-defined exception</td>
<td>45</td>
<td>(no subclass)</td>
<td>000</td>
</tr>
</tbody>
</table>
20 Conformance

20.1 Claims of conformance

In addition to the requirements of ISO/IEC 9075-1, Subclause 8.1.5, “Claims of conformance”, a claim of conformance to this part of ISO/IEC 9075 shall state whether or not Feature P01, “Stored modules”, is supported.
Annex A
(informative)

SQL Conformance Summary

The contents of this Annex summarizes all Conformance Rules, ordered by Feature ID and by Subclause.

1) Specifications for Feature P01, “Stored modules”:
   a) Subclause 11.2, “<SQL procedure statement>”:
      i) Without Feature P01, “Stored modules”, an <SQL procedure statement> shall not be an <SQL server module definition> or a <drop module statement>. 
Annex B
(informative)

Implementation-defined elements

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

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 4.7, “Diagnostics area“:
   a) An SQL-implementation places information about a completion or exception condition that causes a handler to be activated into the diagnostics area prior to activating the handler. If other conditions are raised, then it is implementation-defined whether the implementation places information about them into the diagnostics area.

2) Subclause 8.3, “<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.

3) Subclause 9.18, “<SQL-server module definition>“:
   a) If <SQL-server module path specification> is not specified, then an <SQL-server module path specification> containing an implementation-defined <schema name list> that includes the explicit or implicit <schema name> of the <SQL-server module name> is implicit.
Annex C
(informative)

Implementation-dependent elements

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

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 4.4, "Tables":
   a) The effective <schema name> of the <schema qualified name> of the declared local temporary table may be thought of as the implementation-dependent SQL-session identifier associated with the SQL-session and the name of the <SQL-server module definition> that contains the <temporary table declaration>.

2) Subclause 9.18, "<SQL-server module definition>":
   a) If the SQL-server module is actually represented in a character set other than the character set identified by the explicit or implicit <SQL-server module character set specification>, then the effects are implementation-dependent.

3) Subclause 12.4, "<select statement: single row>":
   a) The order of assignment of values to targets in the <select target list> is implementation-dependent.

4) Subclause 12.7, "<temporary table declaration>":
   a) If a <temporary table declaration> is contained in an <SQL-client module definition> without an intervening <SQL-server module definition>, then the implementation-dependent <schema name> is effectively derived from the implementation-dependent SQL-session identifier associated with the SQL-session and an implementation-dependent name associated with the SQL-client module that contains the <temporary table declaration>. Otherwise, the implementation-dependent <schema name> is effectively derived from the implementation-dependent SQL-session identifier associated with the SQL-session and the name associated of the <SQL-server module definition> that contains the <temporary table declaration>.

5) Subclause 13.1, "<compound statement>":
   a) The implicit <beginning label> of a <compound statement> with no explicit <beginning label> is implementation-dependent.
b) The variables, cursors, and handlers specified in the <local declaration list>, the <local
cursor declaration list>, and the <local handler declaration list> of a <compound statement>
are created in an implementation-dependent order.

6) Subclause 13.10, “<loop statement>“:
   a) The implicit <beginning label> of a <loop statement> with no explicit <beginning label> is
      implementation-dependent.

7) Subclause 13.11, “<while statement>“:
   a) The implicit <beginning label> of a <while statement> with no explicit <beginning label> is
      implementation-dependent.

8) Subclause 13.12, “<repeat statement>“:
   a) The implicit <beginning label> of a <repeat statement> with no explicit <beginning label> is
      implementation-dependent.

9) Subclause 13.13, “<for statement>“:
   a) The implicit <beginning label> of a <for statement> with no explicit <beginning label> is
      implementation-dependent.

   b) The <cursor name> used in the transformation of a <for statement> into a <while state-
      ment> is implementation-dependent, as are the <condition name> and the <SQL variable
      name> used in the <while statement> for getting diagnostics information.
Annex D
(informative)

Deprecated features

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

No additional deprecated items.
Annex E
(informative)

Incompatibilities with ISO/IEC 9075:1992

This edition of this part of ISO/IEC 9075 introduces some incompatibilities with the earlier version of Database Language SQL as specified in ISO/IEC 9075:1992. Unless specified in this Annex, features and capabilities of Database Language SQL are compatible with the earlier version of ISO/IEC 9075.

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

- CONDITION
- DO
- ELSEIF
- EXIT
- HANDLER
- IF
- ITERATE
- LEAVE
- LOOP
- REDO
- REPEAT
- RESIGNAL
- SIGNAL
- UNDO
- UNTIL
- WHILE
2) The definition of “possibly nullable” with regard to <routine invocation> has been tightened. If all subject routines of a <routine invocation> specify PARAMETER STYLE GENERAL, then the <routine invocation> is known not nullable. In ISO/IEC 9075-4:1996, all <routine invocation>s were regarded as possibly nullable.

3) Some of the normative material previously specified in ISO/IEC 9075-4:1996 has been moved to ISO/IEC 9075-2 and ISO/IEC 9075-5. Although this will change any statement of conformance to this and other parts of ISO/IEC 9075, no incompatibilities other than those listed above have been introduced by the reorganization of this normative material.
Annex F
(informative)

SQL Feature Taxonomy

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

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

The 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.

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>
<thead>
<tr>
<th>Feature ID</th>
<th>Feature Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>P001 Stored modules</td>
</tr>
<tr>
<td>2</td>
<td>P001-01 &lt;SQL-server module definition&gt;</td>
</tr>
<tr>
<td>3</td>
<td>P001-02 &lt;drop module statement&gt;</td>
</tr>
<tr>
<td>4</td>
<td>P002 Computational completeness</td>
</tr>
<tr>
<td>5</td>
<td>P002-01 &lt;compound statement&gt;</td>
</tr>
<tr>
<td>6</td>
<td>P002-02 &lt;handler declaration&gt;</td>
</tr>
<tr>
<td>7</td>
<td>P002-03 &lt;condition declaration&gt;</td>
</tr>
</tbody>
</table>
Table 5—SQL/PSM feature taxonomy for features outside Core SQL (Cont.)

<table>
<thead>
<tr>
<th>Feature ID</th>
<th>Feature Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>P002-04 &lt;SQL variable declaration&gt;</td>
</tr>
<tr>
<td>9</td>
<td>P002-05 &lt;assignment statement&gt;</td>
</tr>
<tr>
<td>10</td>
<td>P002-06 &lt;case statement&gt;</td>
</tr>
<tr>
<td>11</td>
<td>P002-07 &lt;if statement&gt;</td>
</tr>
<tr>
<td>12</td>
<td>P002-08 &lt;iterate statement&gt;</td>
</tr>
<tr>
<td>13</td>
<td>P002-09 &lt;leave statement&gt;</td>
</tr>
<tr>
<td>14</td>
<td>P002-10 &lt;loop statement&gt;</td>
</tr>
<tr>
<td>15</td>
<td>P002-11 &lt;repeat statement&gt;</td>
</tr>
<tr>
<td>16</td>
<td>P002-12 &lt;while statement&gt;</td>
</tr>
<tr>
<td>17</td>
<td>P002-13 &lt;for statement&gt;</td>
</tr>
<tr>
<td>18</td>
<td>P002-14 &lt;signal statement&gt;</td>
</tr>
<tr>
<td>19</td>
<td>P002-15 &lt;resignal statement&gt;</td>
</tr>
<tr>
<td>20</td>
<td>P002-16 &lt;control statement&gt; as the SQL-statement of an externally-invoked procedure</td>
</tr>
<tr>
<td><strong>P003</strong></td>
<td><strong>Information Schema views</strong></td>
</tr>
<tr>
<td>22</td>
<td>P003-01 MODULES view</td>
</tr>
<tr>
<td>23</td>
<td>P003-02 MODULE_TABLE_USAGE view</td>
</tr>
<tr>
<td>24</td>
<td>P003-03 MODULE_COLUMN_USAGE view</td>
</tr>
<tr>
<td>25</td>
<td>P003-04 MODULE_PRIVILEGES view</td>
</tr>
</tbody>
</table>
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 • 66, 68
<action> • 19
activated • 15, 16, 86, 114, 135
active • 15, 16, 83, 96, 114, 129
active completion condition • 15
active condition • 15
active exception condition • 15, 16
Ada • 3
ALTER • 110, 118, 120, 126, 127
<alter column definition> • 39
<alter domain statement> • 39
<alter module statement> • 17, 110
AND • 115, 116, 117, 118, 119
ANY • 121
applicable privileges • 33, 65
ARE • 55
AS • 102, 103, 115, 116, 117, 118, 119, 120
assertion • 9, 51, 52, 58
<assertion definition> • 51
assignable • 34
assignment • 1, 10, 17, 28, 70, 76, 90, 91, 92, 105, 110, 137, 144
ASSIGNMENT • 110
<assignment source> • 90, 91, 92, 105
<assignment statement> • 17, 28, 70, 90, 105, 110, 144
<assignment target> • 90, 91, 92, 105
associated with • 14, 16, 19, 79, 85, 86, 89, 114, 137, 143
AT • 21, 101, 141
atomic • 7, 19, 83
ATOMIC • 19, 53, 81, 82, 83, 84, 99, 100, 101, 103
atomic execution context • 19, 83
attribute • 39
<attribute definition> • 39
ATTRIBUTES • 6, 109, 110, 129, 143
AUTHORIZATION • 118, 120, 126, 127
<authorization identifier> • 15, 19, 33, 56, 58, 123

— B —
based on • 3, 93, 95
base table • 14, 40, 80, 121, 123, 125, 126
<basic identifier chain> • 28, 30
basis • 28, 31
basis length • 28
BEGIN • 81, 99, 100, 101, 103, 110
<beginning label> • 28, 31, 81, 82, 97, 98, 99, 100, 101, 102, 137, 138
BOOLEAN • 103

candidate basis • 28, 31
CASCADE • 38, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 52, 54, 58, 68
CASE • 93
case not found for case statement • 94, 129
<case statement> • 17, 70, 93, 94, 110, 144
<case statement else clause> • 93, 94
catalog • 55, 115, 116, 117, 118, 119, 121, 122, 124, 125, 126, 127
<catalog name> • 55
CATALOG_NAME • 112, 115, 116, 117, 118, 119, 121, 125
character • 13, 14, 36, 48, 50, 53, 56, 57, 59, 109, 110, 126, 127, 135, 137
character repertoire • 13
character set • 13, 14, 36, 48, 50, 53, 56, 57, 59, 109, 110, 126, 127, 135, 137
<character set name> • 48
<character set specification> • 55, 56
<character string literal> • 36, 135
CHARACTER_DATA • 123, 126
CHARACTER_SETS • 126
CHARACTER_SET_CATALOG • 118, 120, 126
CHARACTER_SET_NAME • 118, 120, 126
CHARACTER_SET_SCHEMA • 118, 120, 126
CHAR_SET_CAT • 120
CHAR_SET_NAME • 120
CHECK • 121, 123, 125
CHECK_REFERENCES • 121, 125
CLASS • 69, 112
CLASS_ORIGIN • 112
— H —

handle • 1, 5, 6, 10, 15, 16, 17, 81, 82, 83, 85, 86, 87, 94, 96, 98, 99, 100, 101, 107, 110, 114, 129, 135, 138, 143
HANDLER • 21, 69, 85, 103, 110, 141
<handler action> • 16, 17, 85, 86
<handler declaration> • 6, 15, 16, 17, 81, 82, 83, 85, 86, 107, 110, 143
<handler type> • 85
HIERARCHY • 67
host language • 107
<host parameter specification> • 34, 91
<host variable definition> • 107

— I —

identified • 13, 19, 23, 24, 35, 39, 55, 56, 58, 60, 67, 70, 82, 91, 92, 113, 121, 123, 125, 126, 127, 135, 137
dentifier • 12, 13, 14, 15, 19, 23, 24, 28, 30, 31, 33, 56, 58, 65, 79, 81, 102, 109, 111, 113, 120, 121, 122, 123, 124, 125, 126, 127, 137
<identifiers> • 23, 28, 31, 81, 102, 109, 111, 113
<identifier chains> • 28
identify • 19, 24, 135, 137
IF • 21, 95, 141
<if statement> • 16, 17, 70, 87, 95, 96, 110, 144
<if statement else clause> • 95, 96
<if statement elseif clause> • 95
<if statement then clause> • 95, 96
immediate • 14, 23, 24, 56, 66, 67, 82, 83, 90, 94, 96, 97, 105, 107, 111, 113
immediately contain • 56, 66, 67, 82, 83, 90, 96, 105, 107, 111, 113
impacted • 40
impacted dereference operation • 40
implementation-defined • 15, 36, 55, 135
implementation-dependent • 14, 56, 76, 79, 82, 83, 99, 100, 101, 137, 138
Implicit • 16
<implicitly typed value specification> • 90
IN • 115, 116, 117, 118, 119, 121, 123, 125
include • 1, 13, 14, 16, 19, 24, 28, 31, 33, 38, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 52, 54, 55, 56, 58, 61, 65, 66, 67, 82, 93, 111, 113, 135, 143
Information Schema • 1, 115, 144
INFORMATION_SCHEMA • 115, 116, 117, 118, 119, 120, 121, 123, 125, 126
INFORMATION_SCHEMA_CATALOG_NAME • 115, 116, 117, 118, 119, 120, 121, 123, 125, 126
inherirt • 14
innermost • 28, 73, 74, 75, 77, 78, 85, 111, 113
INSERT • 67
<insert column list> • 67
<insert statement> • 66, 67
INTO • 103
invalid • 19
invalid transaction initiation • 19
IS_GRANTABLE • 116, 119, 123, 124
ITERATE • 21, 97, 141

— J —

JOIN • 115, 117

— K —

KEY • 109, 121, 123, 125, 126
known not nullable • 142

— L —

LANGUAGE • 60
/language clause> • 60
LEAVE • 16, 21, 98, 110, 141
<leave statement> • 17, 18, 19, 70, 96, 99, 100, 101, 103, 110, 144
<left paren> • 12, 28, 90
LENGTH • 111, 112, 113
/local cursor declaration list> • 81, 82, 83, 98, 138
/local declaration> • 81, 82
/local declaration list> • 81, 82, 83, 84, 98, 138
/local handler declaration list> • 81, 82, 83, 98, 138
/local or schema qualified name> • 23
local temporary table • 13, 14, 34, 56, 79, 137
LOOP • 21, 99, 110, 141
<loop statement> • 17, 18, 70, 99, 110, 138, 144

— M —

MESSAGE_LENGTH • 111, 112, 113
MESSAGE_OCTET_LENGTH • 111, 112, 113
MESSAGE_TEXT • 112, 113
method • 15, 28, 54, 67, 90
<method invocation> • 28
<method name> • 90
<method reference> • 67
modified • 28, 90, 91, 92
<modified field reference> • 28, 90, 91, 92
<modified field target> • 90, 91, 92
MODULE • 118, 120, 123, 125
MODULE_AUTHORIZATION • 118, 120, 126, 127
MODULE_CATALOG • 115, 116, 117, 118, 119, 120, 121, 123, 125, 126
MODULE_COLUMN_USAGE • 115, 116, 117, 119, 120, 121, 123, 124, 125, 126, 127, 144
<module authorization clause> • 127
<module authorization identifier> • 127
<module function> • 60
<module name> • 35
<module procedure> • 60
<module routine> • 14, 60
MODULES • 118, 120, 121, 123, 125, 126, 144
MODULE_AUTHORIZATION • 118, 120, 126, 127
MODULE_CATALOG • 115, 116, 117, 118, 119, 120, 121, 123, 124, 125, 126
MODULE_COLUMN_USAGE • 115, 120, 121, 144
MODULE_CREATED • 118
MODULE_DEFINITION • 118, 120, 126, 127
MODULE_LAST_ALTERED • 118
MODULE_NAME • 116, 117, 118, 119, 120, 121, 123, 124, 125, 126
MODULE_PRIVILEGES • 116, 118, 119, 123, 144
MODULE_SCHEMA • 116, 117, 118, 119, 120, 121, 123, 124, 125, 126
MODULE_TABLE_USAGE • 117, 121, 125, 144
MORE • 112, 114
most appropriate • 16, 86, 114
most appropriate handler • 16, 86, 114
<mutated target> • 90
<mutated target specification> • 90
<mutator reference> • 28, 90
— N —
Name • 23, 67, 143
NAME • 55, 112, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127
NAMES • 55
NATURAL • 41
new transition variable column reference • 90
<non-reserved word> • 21
no subclass • 129
NOT • 81, 82, 85, 86, 99, 100, 101, 103, 123, 126
null • 31, 92, 105, 127, 129, 142
NULL • 123, 126
<null specification> • 105
null value • 92, 127, 129
null value in field reference • 92, 129
— O —
object • 6, 12, 13, 15, 35, 41, 65, 67
object column • 67
<object column> • 67
<object name> • 35
OCTET_LENGTH • 111, 112, 113
ON • 58, 115, 116, 117, 118, 119, 120
OPEN • 103
<open statement> • 73, 103
OPTION • 65, 67, 115, 116, 117, 118, 119, 120, 124
OR • 115, 116, 117, 118, 119, 121, 123, 125
order • 54, 76, 83, 133, 137, 138
outermost • 16, 82, 99, 100, 101, 102
owned by • 15, 65, 115, 117
— P —
package • 143
parameter • 1, 7, 15, 27, 28, 34, 36, 49, 74, 90, 91, 102, 135
Part 1 • 3
Part 2 • 3
Part 3 • 3
Part 4 • 5, 12, 109
<Part 4 conformance> • 12
<Part 4 module> • 12
<Part 4 module no> • 12
<Part 4 module yes> • 12
<Part yes> • 12
Part 5 • 3, 31
part of • 1, 3, 5, 6, 7, 8, 9, 10, 11, 12, 14, 131, 135, 137, 139, 141, 143
<path specification> • 55
period • 28, 90
<purpose> • 28, 90
permitted • 18, 135
persistent • 1, 13, 15, 80
persistent base table • 80
possible scope tags • 28
possibly candidate routine • 33
possibly modifies SQL-data • 93
possibly nullable • 142
precede • 24, 107
precedes • 24
<preparable statement> • 14, 23, 24
Prepare • 105
prepared • 14, 23, 24
<prepare statement> • 14, 23, 24, 105
PRIMARY • 121, 123, 125, 126
privilege • 7, 11, 13, 14, 19, 33, 35, 36, 65, 66, 67, 116, 119, 123, 124
PRIVILEGE • 116, 118, 119, 123, 124, 144
privilege descriptor • 14, 19, 56, 65, 123
PRIVILEGES • 116, 118, 119, 123, 144
<privileges> • 35, 65
PRIVILEGE_TYPE • 116, 119, 123, 124
PUBLIC • 115, 116, 117, 118, 119, 120, 123
— Q —
<qualified identifier> • 23, 24
query • 28, 31, 41, 58, 66, 67, 102, 121, 125
<query expression> • 41, 58, 66, 67, 102, 121, 125
<query specification> • 31
— R —
recursive • 91, 95
REDO • 16, 21, 83, 141
REF • 40
<reference column list> • 41
referenced table • 41
<referenced table and columns> • 41
<reference resolution> • 67
REFERENCES • 41, 121, 123, 125, 126
referent • 28
REPEAT • 21, 101, 141
<repeat statement> • 17, 18, 70, 101, 110, 138, 144
repertoire • 13
represented • 13, 56, 127, 137
<reserved word> • 21, 141
RESIGNAL • 21, 69, 83, 86, 87, 110, 112, 113, 144
<resignal statement> • 15, 16, 17, 18, 70, 83, 86, 87, 110, 112, 113, 144
resignal when handler not active • 114, 129
RESTRICT • 38, 41, 44, 46, 47, 58
RETRIEVED_SQLSTATE • 110, 111, 112, 114
REVOKE • 58
revoke destruction action • 66
<revoke statement> • 35, 41, 58, 66
Persistent Stored Modules (SQL/PSM)

150

SCOPE • 1, 6, 8, 15, 16, 19, 28, 31, 40, 67, 73, 74, 75, 77, 78, 79, 82, 83, 85, 86, 88, 99, 100, 101, 111, 113
Scope • 1
scope clause • 40
scoped table • 67
search condition • 41, 58, 66, 67, 93, 94, 95, 96, 100, 101
<search condition> • 41, 58, 66, 67, 93, 94, 95, 96, 100, 101
<searched case statement> • 93
<searched case statement when clause> • 93, 94
SELECT • 41, 66, 67, 103, 115, 116, 117, 118, 119, 120, 121, 123, 125
<select lists> • 66, 67, 76
<select statement: single row> • 66, 67, 76
<select target list> • 76, 137
<semicolon> • 55, 81
sensitivity • 102
<separator> • 21
sequence • 19, 49
SESSION • 70
SESSION_USER • 70
SET • 89, 90, 103, 111
<set clause> • 67
set of subject types • 24
SETS • 126
<set signal information> • 111, 112, 113
shall • 15, 23, 24, 28, 31, 40, 41, 43, 44, 46, 47, 48, 49, 50, 52, 54, 58, 60, 71, 73, 74, 75, 77, 78, 79, 82, 83, 85, 88, 90, 91, 93, 97, 98, 99, 100, 101, 102, 103, 107, 111, 113, 119, 131, 133, 135
SIGNAL • 21, 111, 141
<signal information item> • 111, 113
<signal information item list> • 111
<signal statement> • 15, 16, 18, 70, 110, 111, 112, 144
<signal value> • 111, 113, 114
SIMPLE • 3, 28, 36, 38, 40, 41, 42, 43, 44, 46, 47, 48, 49, 50, 52, 54, 58, 65, 68, 83, 86, 87, 89, 90, 93, 98, 99, 100, 101, 112
<simple case operand 1> • 93, 105, 106
<simple case operand 2> • 93, 105, 106
<simple case statement> • 93, 105
<simple case statement when clause> • 93
<simple target specification> • 27
<simple value specification> • 27, 111, 113
simply contain • 23, 66, 67, 82, 83, 90, 91, 92, 94, 97, 105, 106, 125
simply contained in • 23, 66, 67, 82, 83, 90, 91, 92, 94, 97, 105, 106, 125
simply containing • 86
source • 90, 91, 92, 105
SPECIFIC • 38, 42, 43, 44, 46, 48, 49, 50, 52, 54
specific <handler declaration> • 15, 86
specific name • 38, 40, 42, 43, 44, 46, 48, 49, 50, 52, 54
<specific name> • 38, 40, 42, 43, 44, 46, 48, 49, 50, 52
specified by • 13, 14, 55, 56, 73, 74, 75, 77, 78, 79, 82, 83, 85, 86, 88, 99, 100, 101, 111, 113
specify • 14, 15, 16, 19, 35, 41, 51, 53, 82, 99, 100, 101, 112, 114, 143
SQL • 60
SQL-agent • 69
SQL-client • 14, 16, 18, 24, 25, 56, 69, 79, 80, 102, 137
SQL-client module • 14, 18, 24, 25, 56, 69, 79, 80, 102, 137
<SQL-client module definition> • 24, 25, 56, 79, 102, 137
temporary table • 13, 14, 34, 56, 79, 83, 107, 110, 137, 138
<temporary table declaration> • 13, 14, 55, 79, 107, 110, 137
<terminated local cursor declaration> • 81, 82
<terminated local handler declaration> • 81
<terminated SQL statement> • 81
THEN • 93, 95, 96, 118
TIME_STAMP • 126
TO • 65, 115, 116, 117, 118, 119, 120
<token> • 21
transaction • 7, 18, 19
transaction-initiating • 18, 19
transaction state • 18
translation • 14, 50, 67
<translation name> • 50
trigger • 9, 14, 16, 41, 53, 58, 87
<trigger definition> • 53
triggered action • 41
<triggered SQL statement> • 53
TRIM • 36
TRUE • 103
type • 14, 15, 16, 23, 24, 39, 40, 59, 67, 74, 85, 89, 90, 91, 93, 102, 105, 106, 111
type • 109
types • 24, 102

— U —
undefined • 105
UNDO • 16, 21, 83, 85, 87, 141
unhandled completion condition • 16, 114
unhandled exception condition • 16, 83, 94, 96, 114
unhandled user-defined exception • 110, 114, 129
<unqualified schema name> • 23, 24
<unsigned value specification> • 28
UNTIL • 21, 101, 141
UPDATE • 67
(update statement: positioned) • 67, 78
(update statement: searched) • 67
USAGE • 67, 115, 117, 120, 121, 125, 144
USER • 69, 70, 115, 116, 117, 118, 123, 126
user-defined • 14, 16, 23, 24, 39, 59, 63, 67, 90, 110, 114, 129
user-defined type • 14, 23, 24, 59, 67, 90
<user-defined type> • 23
<user-defined type name> • 23
user identifier • 65
USERS • 123, 126
USER_NAME • 123

— V —
valid • 3, 19
<value expression> • 28, 67, 90, 91, 93, 105, 106
<value expression primary> • 28, 67
<value specification> • 27
VARIABLE • 110
view • 1, 8, 11, 14, 25, 41, 45, 46, 56, 58, 66, 83, 84, 87, 98, 99, 100, 101, 109, 115, 116, 117, 118, 119, 120, 121, 125, 144
VIEW • 115, 116, 117, 118, 119, 120
view definition • 45
&view definition> • 45
viewed table • 45
visible • 16

— W —
warning • 129
WHEN • 93
WHERE • 115, 116, 117, 118, 119
WHILE • 21, 100, 110, 141
<while statement> • 17, 18, 70, 100, 110, 138, 144
WITH • 65, 67, 115, 116, 117, 118, 119, 120, 124
without an intervening • 23, 24, 60, 79, 82, 83, 94, 96, 97, 98, 103, 137

— Y —
YES • 123, 124, 143