

INTERNATIONAL STANDARD

ISO/IEC
9075-4

First edition
1996-12-15

Information technology — Database languages — SQL —

Part 4: Persistent Stored Modules (SQL/PSM)

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



Reference number
ISO/IEC 9075-4:1996(E)

AVA Documents
Document Sharing Hub

Contents

	Page
Foreword	ix
Introduction	xi
1 Scope	1
2 Normative references	3
3 Definitions, notations, and conventions	5
3.1 Definitions	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 Syntactic containment	6
3.3.1.3 Rule evaluation order	6
3.3.2 Relationships to ISO/IEC 9075:1992	6
3.3.2.1 New and modified Clauses, Subclauses, and Annexes	6
3.3.2.2 New and modified Format items	7
3.3.2.3 New and modified paragraphs and rules	7
3.3.2.4 New and modified tables	8
3.3.2.5 Clause, Subclause, and Table relationships	8
3.4 Object identifier for Database Language SQL	15
4 Concepts	17
4.1 SQL-server Modules	17
4.2 SQL-invoked routines	18
4.3 SQL-paths	21
4.3.1 Type conversions and mixing of data types	21
4.4 Tables	22
4.5 Integrity constraints	22
4.6 SQL-schemas	22
4.7 Parameters	23
4.7.1 Status parameters	23
4.8 Diagnostics area	23
4.9 Cursors	23
4.10 Condition handling	23

© ISO/IEC 1996

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

ISO/IEC Copyright Office • Case postale 56 • CH-1211 Genève 20 • Switzerland

Printed in Switzerland

4.11	SQL-statements	25
4.11.1	SQL-statements classified by function	25
4.11.2	Embeddable SQL-statements	26
4.11.3	Preparable and immediately executable SQL-statements	26
4.11.4	Directly executable SQL-statements	26
4.11.5	SQL-statements and transaction states	26
4.11.6	Compound statements	27
4.11.7	SQL-statement atomicity	27
4.12	Privileges	27
4.13	SQL-sessions	28
5	Lexical elements	29
5.1	<token> and <separator>	29
5.2	Names and identifiers	31
6	Scalar expressions	33
6.1	<value specification> and <target specification>	33
6.2	<column reference>	35
6.3	<item reference>	36
6.4	<datetime value function>	38
6.5	<case expression>	39
6.6	<value expression>	40
7	Query expressions	41
7.1	<table value constructor>	41
7.2	<table expression>	42
7.3	<query specification>	43
7.4	<query expression>	44
7.5	<scalar subquery>, <row subquery>, and <table subquery>	45
8	Data assignment rules and routine determination	47
8.1	Subject routine determination	47
8.2	Type precedence list determination	48
9	Additional common elements	51
9.1	<routine invocation>	51
9.2	<privileges>	64
9.3	<specific routine designator>	65
9.4	<sqlstate value>	67
9.5	<language clause>	68
9.6	<path specification>	69
10	Schema definition and manipulation	71
10.1	<schema definition>	71
10.2	<drop schema statement>	73
10.3	<default clause>	75
10.4	<check constraint definition>	77
10.5	<drop column definition>	78
10.6	<drop table constraint definition>	79
10.7	<drop table statement>	80

10.8	<view definition>	81
10.9	<drop view statement>	82
10.10	<drop domain statement>	83
10.11	<drop character set statement>	84
10.12	<drop collation statement>	85
10.13	<drop translation statement>	86
10.14	<assertion definition>	87
10.15	<drop assertion statement>	88
10.16	<SQL-server module definition>	89
10.17	<drop module statement>	92
10.18	<SQL-invoked routine>	93
10.19	<drop routine statement>	102
10.20	<grant statement>	103
10.21	<revoke statement>	105
11	Modules	111
11.1	<module>	111
11.2	<procedure>	112
11.3	Calls to a <procedure>	113
11.4	<SQL procedure statement>	114
11.5	Data type correspondences	121
12	Data manipulation	129
12.1	<declare cursor>	129
12.2	<open statement>	130
12.3	<fetch statement>	131
12.4	<close statement>	132
12.5	<select statement: single row>	133
12.6	<delete statement: positioned>	134
12.7	<delete statement: searched>	135
12.8	<update statement: positioned>	136
12.9	<update statement: searched>	137
12.10	<temporary table declaration>	138
13	Control statements	141
13.1	<call statement>	141
13.2	<return statement>	142
13.3	<compound statement>	143
13.4	<handler declaration>	146
13.5	<condition declaration>	149
13.6	<SQL variable declaration>	150
13.7	<assignment statement>	151
13.8	<case statement>	152
13.9	<if statement>	154
13.10	<leave statement>	156
13.11	<loop statement>	157
13.12	<while statement>	158
13.13	<repeat statement>	159
13.14	<for statement>	160

14 Transaction management	163
14.1 <commit statement>	163
14.2 <rollback statement>	164
15 Session management	165
15.1 <set path statement>	165
16 Dynamic SQL	167
16.1 Description of SQL item descriptor areas	167
16.2 <prepare statement>	169
16.3 <using clause>	170
17 Diagnostics management	175
17.1 <get diagnostics statement>	175
17.2 <signal statement>	178
17.3 <resignal statement>	179
18 Embedded SQL	181
18.1 <embedded SQL host program>	181
19 Information Schema and Definition Schema	183
19.1 Information Schema	183
19.1.1 SCHEMATA view	183
19.1.2 DOMAINS view	184
19.1.3 COLUMNS view	185
19.1.4 MODULES view	187
19.1.5 ROUTINES view	188
19.1.6 PARAMETERS view	189
19.1.7 MODULE_TABLE_USAGE view	190
19.1.8 MODULE_COLUMN_USAGE view	191
19.1.9 MODULE_PRIVILEGES view	192
19.1.10 ROUTINE_PRIVILEGES view	193
19.1.11 ROUTINE_TABLE_USAGE view	194
19.1.12 ROUTINE_COLUMN_USAGE view	195
19.1.13 Definition of SQL built-in functions	196
19.2 Definition Schema	201
19.2.1 SCHEMATA base table	201
19.2.2 DATA_TYPE_DESCRIPTOR base table	202
19.2.3 MODULES base table	206
19.2.4 ROUTINES base table	208
19.2.5 PARAMETERS base table	211
19.2.6 MODULE_TABLE_USAGE base table	213
19.2.7 MODULE_COLUMN_USAGE base table	214
19.2.8 MODULE_PRIVILEGES base table	215
19.2.9 ROUTINE_PRIVILEGES base table	217
19.2.10 ROUTINE_TABLE_USAGE base table	219
19.2.11 ROUTINE_COLUMN_USAGE base table	220
20 Status codes	221
20.1 SQLSTATE	221

21 Conformance 223

21.1 Claims of conformance 223

21.2 Extensions and options 223

21.2.1 Information Schema requirements 223

21.2.2 Schema manipulation requirements 224

21.3 Flagger requirements 224

Annex A Implementation-defined elements 225

Annex B Implementation-dependent elements 227

Annex C Deprecated features 229

Annex D Incompatibilities with ISO/IEC 9075:1992 231

Index 233

TABLES

Table	Page
1	Clause, Subclause, and Table relationships 8
2	Standard programming languages 68
3	Data type correspondences for Ada 121
4	Data type correspondences for C 122
5	Data type correspondences for COBOL 123
6	Data type correspondences for Fortran 124
7	Data type correspondences for MUMPS 125
8	Data type correspondences for Pascal 126
9	Data type correspondences for PL/I 127
10	Data types of <key word>s used in SQL item descriptor areas 167
11	Codes used for input/output SQL parameter modes in Dynamic SQL 167
12	<identifier>s for use with <get diagnostics statement> 175
13	SQL-statement character codes for use in the diagnostics area 176
14	SQLSTATE class and subclass values 221

Foreword

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

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

International Standard ISO/IEC 9075-4, was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC21, *Open systems interconnection, data management, and open distributed processing*.

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

- Part 3: Call-Level Interface (SQL/CLI)
- Part 4: Persistent Stored Modules (SQL/PSM)

Parts 1, 2, and 5 are currently published as ISO/IEC 9075:1992.

Part 3 is currently published as ISO/IEC 9075-3:1995.

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

Introduction

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

- 1) Clause 1, “Scope”, specifies the scope of this part of ISO/IEC 9075.
- 2) Clause 2, “Normative references”, identifies additional standards that, through reference in this part of ISO/IEC 9075, constitute provisions of this part of ISO/IEC 9075.
- 3) Clause 3, “Definitions, notations, and conventions”, defines the notations and conventions used in this part of ISO/IEC 9075.
- 4) Clause 4, “Concepts”, presents concepts used in the definition of persistent stored modules.
- 5) Clause 5, “Lexical elements”, defines a number of lexical elements used in the definition of persistent stored modules.
- 6) Clause 6, “Scalar expressions”, defines a number of scalar expressions used in the definition of persistent stored modules.
- 7) Clause 7, “Query expressions”, defines the elements of the language that produce rows and tables of data as used in persistent stored modules.
- 8) Clause 8, “Data assignment rules and routine determination”, defines the data assignment rules used in the definition of persistent stored modules.
- 9) Clause 9, “Additional common elements”, defines additional common elements used in the definition of persistent stored modules.
- 10) Clause 10, “Schema definition and manipulation”, defines the schema definition and manipulation statements associated with the definition of persistent stored modules.
- 11) Clause 11, “Modules”, defines the facilities for using persistent stored modules.
- 12) Clause 12, “Data manipulation”, defines data manipulation operations associated with persistent stored modules.
- 13) Clause 13, “Control statements”, defines the control statements used with persistent stored modules.
- 14) Clause 14, “Transaction management”, defines the SQL-transaction management statements associated with persistent stored modules.
- 15) Clause 15, “Session management”, defines the SQL-session management statements associated with persistent stored modules.
- 16) Clause 16, “Dynamic SQL”, defines the facilities for executing SQL-statements dynamically in the context of persistent stored modules.
- 17) Clause 17, “Diagnostics management”, defines enhancements to the facilities used with persistent stored modules.

- 18) Clause 18, “Embedded SQL”, defines host language embeddings related to persistent stored modules.
- 19) Clause 19, “Information Schema and Definition Schema”, defines the Information and Definition Schema objects associated with persistent stored modules.
- 20) Clause 20, “Status codes”, defines SQLSTATE values related to persistent stored modules.
- 21) Clause 21, “Conformance”, defines the criteria for conformance to this part of ISO/IEC 9075.
- 22) Annex A, “Implementation-defined elements”, is an informative Annex. It lists those features for which the body of this part of the standard states that the syntax or meaning or effect on the database is partly or wholly implementation-defined, and describes the defining information that an implementer shall provide in each case.
- 23) Annex B, “Implementation-dependent elements”, is an informative Annex. It lists those features for which the body of this part of the standard states that the syntax or meaning or effect on the database is partly or wholly implementation-dependent.
- 24) Annex C, “Deprecated features”, is an informative Annex. It lists features that the responsible Technical Committee intends will not appear in a future revised version of ISO/IEC 9075.
- 25) Annex D, “Incompatibilities with ISO/IEC 9075:1992”, is an informative Annex. It lists the incompatibilities between this edition of ISO/IEC 9075 and ISO/IEC 9075:1992.

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 21, “Conformance”, Subclauses begin a new page. Any resulting blank space is not significant.

Information technology — Database languages — SQL —

Part 4:

Persistent Stored Modules (SQL/PSM)

1 Scope

This part of International Standard ISO/IEC 9075 specifies the syntax and semantics of a database language for declaring and maintaining persistent database language routines either in SQL-server modules or as standalone schema-level routines, and invoking them from programs written in a standard programming language.

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

- The specification of statements to direct the flow of control.
- The assignment of the result of expressions to variables and parameters.
- The specification of condition handlers that allow SQL-invoked routines to deal with various conditions that arise during their execution.
- The specification of statements to signal and resignal conditions.
- The ability to set an SQL-path for controlling the determination of the subject routine to be invoked.
- The declaration of local cursors.
- The declaration of local variables.

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

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

2 Normative references

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

ISO/IEC 1539:1991, *Information technology — Programming languages — FORTRAN*.

ISO 1989:1985, *Programming languages — COBOL*.

ISO 6160:1979, *Programming languages — PL/I*.

ISO\IEC 7185:1990, *Information technology — Programming languages — Pascal*.

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

NOTE 2 – ISO 8652:1987 has been superseded by a new edition (ISO/IEC 8652:1995). However, when this part of ISO/IEC 9075 was under development, the previous edition was valid and this part of ISO/IEC 9075 is therefore based on that edition, which is listed below.

ISO 8652:1987, *Programming languages — Ada*.

ISO/IEC 9075:1992, *Information technology — Database languages — SQL*.

ISO/IEC 9899:1990, *Programming languages — C*.

ISO/IEC 10206:1991, *Information technology — Programming languages — Extended Pascal*.

ISO/IEC 11756:1992, *Information technology — Programming languages — MUMPS*.

3 Definitions, notations, and conventions

3.1 Definitions

Insert this paragraph For the purposes of this part of ISO/IEC 9075, the definitions given in ISO/IEC 9075:1992 and the following definitions apply.

- a) **external routine**: An SQL-invoked routine whose routine body is an external body reference that identifies a program written in a standard programming language other than SQL.
- b) **signature (of an SQL-invoked routine)**: The name of the SQL-invoked routine, the position and data types of each of its SQL parameters, and an indication of whether it is an SQL-invoked function or an SQL-invoked procedure.
- c) **SQL routine**: An SQL-invoked routine whose routine body is written in SQL.
- d) **SQL-invoked routine**: A routine that is allowed to be invoked only from within SQL.

3.2 Notations

Insert this paragraph All notations in ISO/IEC 9075:1992 apply to this part of ISO/IEC 9075.

Insert this paragraph The syntax notation used in this part of ISO/IEC 9075 is an extended version of BNF (“Backus Normal Form” or “Backus Naur Form”).

Insert this paragraph This version of BNF is fully described in ISO/IEC 9075:1992.

3.3 Conventions

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

3.3.1 Use of terms

3.3.1.1 Exceptions

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

- The execution of a statement is unsuccessful.
- The application of General Rules, other than those of Subclause 12.3, “<procedure>”, in ISO/IEC 9075:1992, Subclause 9.1, “<routine invocation>”, Subclause 11.4, “<SQL procedure statement>”, Subclause 20.1, “<direct SQL statement>”, in ISO/IEC 9075:1992, Subclause 13.3, “<compound statement>”, and Subclause 13.4, “<handler declaration>”, may be terminated.
- Diagnostic information is to be made available.

3.3 Conventions

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

Insert this paragraph The effect on <target specification>s and SQL descriptor areas of an SQL-statement that terminates with an exception condition, unless explicitly defined by this International Standard, is implementation-dependent.

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

3.3.1.2 Syntactic containment

Insert this paragraph If <A> contains a <table name> that identifies a view that is defined by a <view definition> V, then <A> is said to *generally contain* the <query expression> contained in V. If <A> contains a <routine invocation> RI, then <A> is said to *generally contain* the <routine body>s of all <SQL-invoked routine>s that are included in the set of subject routines of RI. If <A> contains , then <A> generally contains . If <A> generally contains and generally contains <C>, then <A> generally contains <C>.

3.3.1.3 Rule evaluation order

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

3.3.2 Relationships to ISO/IEC 9075:1992

This part of ISO/IEC 9075 depends on ISO/IEC 9075:1992 and its Technical Corrigenda. This part of ISO/IEC 9075 is to be used as though it were merged with the text of ISO/IEC 9075:1992. This Subclause describes the conventions used to specify the merger. The merger described also accounts for the Technical Corrigenda that have been published to correct ISO/IEC 9075:1992. This accommodation is typically indicated by the presence of a phrase like “in the Technical Corrigenda” or “in the TC”.

3.3.2.1 New and modified Clauses, Subclauses, and Annexes

Clauses, Subclauses, and Annexes (other than Clause 1, “Scope” and Clause 2, “Normative references”) in this part of ISO/IEC 9075 that have names identical to Clauses, Subclauses, or Annexes in ISO/IEC 9075:1992 supplement the Clause, Subclause, or Annex, respectively, in ISO/IEC 9075:1992, typically by replacing paragraphs, Format items, or Rules or by providing new paragraphs, Format items, or Rules. However, Clauses, Subclauses, and Annexes in this part of ISO/IEC 9075 that have names identical to Clauses, Subclauses, and Annexes in ISO/IEC 9075:1992 do not necessarily have the same *number* or *letter* as the corresponding Clause, Subclause, or Annex in ISO/IEC 9075:1992. Any differences in Clause or Subclause number or in Annex letter is not significant. Table 1, “Clause, Subclause, and Table relationships”, identifies the relationships between Clauses, Subclauses, and Annexes in this part of ISO/IEC 9075 and the corresponding Clauses, Subclauses, and Annexes in ISO/IEC 9075:1992.

Clauses, Subclauses, and Annexes in this part of ISO/IEC 9075 that have names that are not identical to Clauses, Subclauses, or Annexes in ISO/IEC 9075:1992 provide language specification particular to this part of ISO/IEC 9075. Subclauses that are subsidiary to (“contained in”) Clauses or Subclauses identified as new are inherently new and are not further identified.

The Clauses, Subclauses, and Annexes in this part of ISO/IEC 9075 appear in the order in which they are intended to appear in the merged document. Absent other explicit instructions regarding its placement, any new Clause, Subclause, or Annex is to be positioned as follows: Locate the prior Clause, Subclause, or Annex in this part of ISO/IEC 9075 whose name is identical to the name of a corresponding Clause, Subclause, or Annex that appears in ISO/IEC 9075:1992. The new Clause, Subclause, or Annex shall immediately follow that Clause, Subclause, or Annex. If there are multiple new Clauses, Subclauses, or Annexes with no intervening Clause, Subclause, or Annex that modifies an existing Clause, Subclause, or Annex, then those new Clauses, Subclauses, or Annexes appear in order, following the prior Clause, Subclause, or Annex whose name was matched.

3.3.2.2 New and modified Format items

In modified Subclauses, Format items that define a BNF nonterminal symbol (that is, the BNF nonterminal symbol appears on the left-hand side of the `::=` mark) sometimes modify a Format item whose definition appears in ISO/IEC 9075:1992, sometimes replace a Format item whose definition appears in ISO/IEC 9075:1992, and sometimes define a new Format item that does not have a definition at all in ISO/IEC 9075:1992. Those Format items in this part of ISO/IEC 9075 that modify a Format item whose definition appears in ISO/IEC 9075:1992, are identified by the existence of a “Format comment” such as:

```
<modified item> ::=
    !! All alternatives from ISO/IEC 9075:1992
    | <new alternative>
```

By contrast, Format items that completely replace Format items in ISO/IEC 9075:1992 have BNF nonterminal symbols identical to BNF nonterminal symbols of Format items in ISO/IEC 9075:1992, but do not state that they include any alternatives from ISO/IEC 9075:1992.

New Format items that have no correspondence to any Format item in ISO/IEC 9075:1992 are not specially identified in this part of ISO/IEC 9075.

In new Subclauses, all Format items are also new and require no specific marking.

3.3.2.3 New and modified paragraphs and rules

In modified Subclauses, each paragraph or Rule is marked to indicate whether it is a modification of a paragraph or Rule in ISO/IEC 9075:1992, or is a new paragraph or Rule in this part of ISO/IEC 9075.

Modifications of paragraphs or Rules in ISO/IEC 9075:1992, are identified by the inclusion of an indication such as `Replace the 5th paragraph`, meaning that the fifth paragraph of the corresponding Subclause in ISO/IEC 9075:1992, is to be replaced by the paragraph to which the indication is attached, or `Replace SR6)b)ii)`, meaning that Syntax Rule 6)b)ii) of the corresponding Subclause in ISO/IEC 9075:1992, is to be replaced by the rule to which the indication is attached. In some cases, an indication such as `Augment SR3)` is used. When this appears, it means that the referenced Rule is extended or enhanced by the Rule to which the indication is attached. In most instances, the augmentation is the addition of a new alternative meant to support new syntax.

New paragraphs or Rules in this part of ISO/IEC 9075 are indicated by the inclusion of an indication such as Insert before 2nd paragraph, meaning that the paragraph to which the indication is attached is to be read as though it were inserted preceding the second paragraph of the corresponding Subclause in ISO/IEC 9075:1992. Insert before GR4 should be interpreted to mean that the rule to which the indication is attached is to be read as though it were inserted preceding General Rule 4) of the corresponding Subclause in ISO/IEC 9075:1992. When an indication does not indicate a specific insertion point, such as Insert this paragraph or Insert this GR, then it may be read as implicitly specifying that the new text is to be appended at the end of the appropriate section (the General Rules, for example) of the corresponding Subclause in ISO/IEC 9075:1992.

In such instructions, “SR” is used to mean “Syntax Rule”, “AR” is used to mean “Access Rule”, and “GR” is used to mean “General Rule”. “Desc.” is used to mean “Description” and “Func.” is used to mean “Function”.

All paragraphs, Format items, and Rules in new Clauses or Subclauses are also new and therefore do not require further identification.

3.3.2.4 New and modified tables

Similarly, tables in this part of ISO/IEC 9075 that have names that are identical to tables in ISO/IEC 9075:1992 supplement the table in ISO/IEC 9075:1992, typically by adding or replacing one or more table entries. Tables in this part of ISO/IEC 9075 that have names that are not identical to the names of tables in ISO/IEC 9075:1992 are new tables specified by this part of ISO/IEC 9075. Table 1, “Clause, Subclause, and Table relationships”, identifies the relationships between tables in this part of ISO/IEC 9075 and the corresponding tables in ISO/IEC 9075:1992.

The rows in modified tables are generally new rows to be effectively inserted into the corresponding table in ISO/IEC 9075:1992, though in rare cases rows already in tables in ISO/IEC 9075:1992 are effectively replaced by rows in the table in this part of ISO/IEC 9075. It is always obvious when such an effective replacement is required, based on equality of values in the first column of the table.

3.3.2.5 Clause, Subclause, and Table relationships

Table 1—Clause, Subclause, and Table relationships

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from ISO/IEC 9075:1992
Clause 1, “Scope”	Clause 1, “Scope”
Clause 2, “Normative references”	Clause 2, “Normative references”
Clause 3, “Definitions, notations, and conventions”	Clause 3, “Definitions, notations, and conventions”
Subclause 3.1, “Definitions”	Subclause 3.1, “Definitions”
Subclause 3.2, “Notations”	Subclause 3.2, “Notation”
Subclause 3.3, “Conventions”	Subclause 3.3, “Conventions”
Subclause 3.3.1, “Use of terms”	Subclause 3.3.4, “Use of terms”
Subclause 3.3.1.1, “Exceptions”	Subclause 3.3.4.1, “Exceptions”

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

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from ISO/IEC 9075:1992
Subclause 3.3.1.2, "Syntactic containment"	Subclause 3.3.4.2, "Syntactic containment"
Subclause 3.3.1.3, "Rule evaluation order"	Subclause 3.3.4.4, "Rule evaluation order"
Subclause 3.3.2, "Relationships to ISO/IEC 9075:1992"	(none)
Subclause 3.3.2.1, "New and modified Clauses, Subclauses, and Annexes"	(none)
Subclause 3.3.2.2, "New and modified Format items"	(none)
Subclause 3.3.2.3, "New and modified paragraphs and rules"	(none)
Subclause 3.3.2.4, "New and modified tables"	(none)
Subclause 3.3.2.5, "Clause, Subclause, and Table relationships"	(none)
Subclause 3.4, "Object identifier for Database Language SQL"	Subclause 3.4, "Object identifier for Database Language SQL"
Clause 4, "Concepts"	Clause 4, "Concepts"
Subclause 4.1, "SQL-server Modules"	(none)
Subclause 4.2, "SQL-invoked routines"	(none)
Subclause 4.3, "SQL-paths"	(none)
Subclause 4.3.1, "Type conversions and mixing of data types"	Subclause 4.6, "Type conversions and mixing of data types"
Subclause 4.4, "Tables"	Subclause 4.9, "Tables"
Subclause 4.5, "Integrity constraints"	Subclause 4.10, "Integrity constraints"
Subclause 4.6, "SQL-schemas"	Subclause 4.11, "SQL-schemas"
Subclause 4.7, "Parameters"	Subclause 4.18, "Parameters"
Subclause 4.7.1, "Status parameters"	Subclause 4.18.1, "Status parameters"
Subclause 4.8, "Diagnostics area"	Subclause 4.19, "Diagnostics area"
Subclause 4.9, "Cursors"	Subclause 4.21, "Cursors"
Subclause 4.10, "Condition handling"	(none)
Subclause 4.11, "SQL-statements"	Subclause 4.22, "SQL-statements"
Subclause 4.11.1, "SQL-statements classified by function"	Subclause 4.22.2, "SQL-statements classified by function"
Subclause 4.11.2, "Embeddable SQL-statements"	Subclause 4.22.3, "Embeddable SQL-statements"
Subclause 4.11.3, "Preparable and immediately executable SQL-statements"	Subclause 4.22.4, "Preparable and immediately executable SQL-statements"
Subclause 4.11.4, "Directly executable SQL-statements"	Subclause 4.22.5, "Directly executable SQL-statements"

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

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from ISO/IEC 9075:1992
Subclause 4.11.5, "SQL-statements and transaction states"	Subclause 4.22.6, "SQL-statements and transaction states"
Subclause 4.11.6, "Compound statements"	(none)
Subclause 4.11.7, "SQL-statement atomicity"	(none)
Subclause 4.12, "Privileges"	Subclause 4.26, "Privileges"
Subclause 4.13, "SQL-sessions"	Subclause 4.30, "SQL-sessions"
Clause 5, "Lexical elements"	Clause 5, "Lexical elements"
Subclause 5.1, "<token> and <separator>"	Subclause 5.2, "<token> and <separator>"
Subclause 5.2, "Names and identifiers"	Subclause 5.4, "Names and identifiers"
Clause 6, "Scalar expressions"	Clause 6, "Scalar expressions"
Subclause 6.1, "<value specification> and <target specification>"	Subclause 6.2, "<value specification> and <target specification>"
Subclause 6.2, "<column reference>"	Subclause 6.4, "<column reference>"
Subclause 6.3, "<item reference>"	(none)
Subclause 6.4, "<datetime value function>"	Subclause 6.8, "<datetime value function>"
Subclause 6.5, "<case expression>"	Subclause 6.9, "<case expression>"
Subclause 6.6, "<value expression>"	Subclause 6.11, "<value expression>"
Clause 7, "Query expressions"	Clause 7, "Query expressions"
Subclause 7.1, "<table value constructor>"	Subclause 7.2, "<table value constructor>"
Subclause 7.2, "<table expression>"	Subclause 7.3, "<table expression>"
Subclause 7.3, "<query specification>"	Subclause 7.9, "<query specification>"
Subclause 7.4, "<query expression>"	Subclause 7.10, "<query expression>"
Subclause 7.5, "<scalar subquery>, <row subquery>, and <table subquery>"	Subclause 7.11, "<scalar subquery>, <row subquery>, and <table subquery>"
Clause 8, "Data assignment rules and routine determination"	Clause 9, "Data assignment rules"
Subclause 8.1, "Subject routine determination"	(none)
Subclause 8.2, "Type precedence list determination"	(none)
Clause 9, "Additional common elements"	Clause 10, "Additional common elements"
Subclause 9.1, "<routine invocation>"	(none)
Subclause 9.2, "<privileges>"	Subclause 10.3, "<privileges>"
Subclause 9.3, "<specific routine designator>"	(none)
Subclause 9.4, "<sqlstate value>"	(none)
Subclause 9.5, "<language clause>"	Subclause 10.2, "<language clause>"

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

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from ISO/IEC 9075:1992
Subclause 9.6, "<path specification>"	(none)
Clause 10, "Schema definition and manipulation"	Clause 11, "Schema definition and manipulation"
Subclause 10.1, "<schema definition>"	Subclause 11.1, "<schema definition>"
Subclause 10.2, "<drop schema statement>"	Subclause 11.2, "<drop schema statement>"
Subclause 10.3, "<default clause>"	Subclause 11.5, "<default clause>"
Subclause 10.4, "<check constraint definition>"	Subclause 11.9, "<check constraint definition>"
Subclause 10.5, "<drop column definition>"	Subclause 11.15, "<drop column definition>"
Subclause 10.6, "<drop table constraint definition>"	Subclause 11.17, "<drop table constraint definition>"
Subclause 10.7, "<drop table statement>"	Subclause 11.18, "<drop table statement>"
Subclause 10.8, "<view definition>"	Subclause 11.19, "<view definition>"
Subclause 10.9, "<drop view statement>"	Subclause 11.20, "<drop view statement>"
Subclause 10.10, "<drop domain statement>"	Subclause 11.27, "<drop domain statement>"
Subclause 10.11, "<drop character set statement>"	Subclause 11.29, "<drop character set statement>"
Subclause 10.12, "<drop collation statement>"	Subclause 11.31, "<drop collation statement>"
Subclause 10.13, "<drop translation statement>"	Subclause 11.33, "<drop translation statement>"
Subclause 10.14, "<assertion definition>"	Subclause 11.34, "<assertion definition>"
Subclause 10.15, "<drop assertion statement>"	Subclause 11.35, "<drop assertion statement>"
Subclause 10.16, "<SQL-server module definition>"	(none)
Subclause 10.17, "<drop module statement>"	(none)
Subclause 10.18, "<SQL-invoked routine>"	(none)
Subclause 10.19, "<drop routine statement>"	(none)
Subclause 10.20, "<grant statement>"	Subclause 11.36, "<grant statement>"
Subclause 10.21, "<revoke statement>"	Subclause 11.37, "<revoke statement>"
Clause 11, "Modules"	Clause 12, "Module"
Subclause 11.1, "<module>"	Subclause 12.1, "<module>"
Subclause 11.2, "<procedure>"	Subclause 12.3, "<procedure>"
Subclause 11.3, "Calls to a <procedure>"	Subclause 12.4, "Calls to a <procedure>"
Subclause 11.4, "<SQL procedure statement>"	Subclause 12.5, "<SQL procedure statement>"
Subclause 11.5, "Data type correspondences"	(none)
Clause 12, "Data manipulation"	Clause 13, "Data manipulation"
Subclause 12.1, "<declare cursor>"	Subclause 13.1, "<declare cursor>"

3.3 Conventions

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

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from ISO/IEC 9075:1992
Subclause 12.2, "<open statement>"	Subclause 13.2, "<open statement>"
Subclause 12.3, "<fetch statement>"	Subclause 13.3, "<fetch statement>"
Subclause 12.4, "<close statement>"	Subclause 13.4, "<close statement>"
Subclause 12.5, "<select statement: single row>"	Subclause 13.5, "<select statement: single row>"
Subclause 12.6, "<delete statement: positioned>"	Subclause 13.6, "<delete statement: positioned>"
Subclause 12.7, "<delete statement: searched>"	Subclause 13.7, "<delete statement: searched>"
Subclause 12.8, "<update statement: positioned>"	Subclause 13.9, "<update statement: positioned>"
Subclause 12.9, "<update statement: searched>"	Subclause 13.10, "<update statement: searched>"
Subclause 12.10, "<temporary table declaration>"	Subclause 13.11, "<temporary table declaration>"
Clause 13, "Control statements"	(none)
Subclause 13.1, "<call statement>"	(none)
Subclause 13.2, "<return statement>"	(none)
Subclause 13.3, "<compound statement>"	(none)
Subclause 13.4, "<handler declaration>"	(none)
Subclause 13.5, "<condition declaration>"	(none)
Subclause 13.6, "<SQL variable declaration>"	(none)
Subclause 13.7, "<assignment statement>"	(none)
Subclause 13.8, "<case statement>"	(none)
Subclause 13.9, "<if statement>"	(none)
Subclause 13.10, "<leave statement>"	(none)
Subclause 13.11, "<loop statement>"	(none)
Subclause 13.12, "<while statement>"	(none)
Subclause 13.13, "<repeat statement>"	(none)
Subclause 13.14, "<for statement>"	(none)
Clause 14, "Transaction management"	Clause 14, "Transaction management"
Subclause 14.1, "<commit statement>"	Clause 14.3, "<commit statement>"
Subclause 14.2, "<rollback statement>"	Clause 14.4, "<rollback statement>"
Clause 15, "Session management"	Clause 16, "Session management"
Subclause 15.1, "<set path statement>"	(none)
Clause 16, "Dynamic SQL"	Clause 17, "Dynamic SQL"
Subclause 16.1, "Description of SQL item descriptor areas"	Subclause 17.1, "Description of SQL item descriptor areas"
Subclause 16.2, "<prepare statement>"	Subclause 17.6, "<prepare statement>"

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

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from ISO/IEC 9075:1992
Subclause 16.3, "<using clause>"	Subclause 17.9, "<using clause>"
Clause 17, "Diagnostics management"	Clause 18, "Diagnostics management"
Subclause 17.1, "<get diagnostics statement>"	Subclause 18.1, "<get diagnostics statement>"
Subclause 17.2, "<signal statement>"	(none)
Subclause 17.3, "<resignal statement>"	(none)
Clause 18, "Embedded SQL"	Clause 19, "Embedded SQL"
Subclause 18.1, "<embedded SQL host program>"	Subclause 19.1, "<embedded SQL host program>"
Clause 19, "Information Schema and Definition Schema"	Clause 21, "Information Schema and Definition Schema"
Subclause 19.1, "Information Schema"	Subclause 21.2, "Information Schema"
Subclause 19.1.1, "SCHEMATA view"	Subclause 21.2.4, "SCHEMATA view"
Subclause 19.1.2, "DOMAINS view"	Subclause 21.2.5, "DOMAINS view"
Subclause 19.1.3, "COLUMNS view"	Subclause 21.2.9, "COLUMNS view"
Subclause 19.1.4, "MODULES view"	(none)
Subclause 19.1.5, "ROUTINES view"	(none)
Subclause 19.1.6, "PARAMETERS view"	(none)
Subclause 19.1.7, "MODULE_TABLE_USAGE view"	(none)
Subclause 19.1.8, "MODULE_COLUMN_USAGE view"	(none)
Subclause 19.1.9, "MODULE_PRIVILEGES view"	(none)
Subclause 19.1.10, "ROUTINE_PRIVILEGES view"	(none)
Subclause 19.1.11, "ROUTINE_TABLE_USAGE view"	(none)
Subclause 19.1.12, "ROUTINE_COLUMN_USAGE view"	(none)
Subclause 19.1.13, "Definition of SQL built-in functions"	(none)
Subclause 19.2, "Definition Schema"	Subclause 21.3, "Definition Schema"
Subclause 19.2.1, "SCHEMATA base table"	Subclause 21.3.4, "SCHEMATA base table"
Subclause 19.2.2, "DATA_TYPE_DESCRIPTOR base table"	Subclause 21.3.5, "DATA_TYPE_DESCRIPTOR base table"
Subclause 19.2.3, "MODULES base table"	(none)
Subclause 19.2.4, "ROUTINES base table"	(none)
Subclause 19.2.5, "PARAMETERS base table"	(none)

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

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from ISO/IEC 9075:1992
Subclause 19.2.6, "MODULE_TABLE_USAGE base table"	(none)
Subclause 19.2.7, "MODULE_COLUMN_USAGE base table"	(none)
Subclause 19.2.8, "MODULE_PRIVILEGES base table"	(none)
Subclause 19.2.9, "ROUTINE_PRIVILEGES base table"	(none)
Subclause 19.2.10, "ROUTINE_TABLE_USAGE base table"	(none)
Subclause 19.2.11, "ROUTINE_COLUMN_USAGE base table"	(none)
Clause 20, "Status codes"	Clause 22, "Status codes"
Subclause 20.1, "SQLSTATE"	Subclause 22.1, "SQLSTATE"
Clause 21, "Conformance"	Clause 23, "Conformance"
Subclause 21.1, "Claims of conformance"	Subclause 23.2, "Claims of conformance"
Subclause 21.2, "Extensions and options"	Subclause 23.3, "Extensions and options"
Subclause 21.2.1, "Information Schema requirements"	(none)
Subclause 21.2.2, "Schema manipulation requirements"	(none)
Subclause 21.3, "Flagger requirements"	Subclause 23.4, "Flagger requirements"
Annex A, "Implementation-defined elements"	Annex B, "Implementation-defined elements"
Annex B, "Implementation-dependent elements"	Annex C, "Implementation-dependent elements"
Annex C, "Deprecated features"	Annex D, "Deprecated features"
Annex D, "Incompatibilities with ISO/IEC 9075:1992"	Annex E, "Incompatibilities with ISO/IEC 9075:1989"
Table 1, "Clause, Subclause, and Table relationships"	(none)
Table 2, "Standard programming languages"	Table 16, "Standard programming languages"
Table 3, "Data type correspondences for Ada"	(none)
Table 4, "Data type correspondences for C"	(none)
Table 5, "Data type correspondences for COBOL"	(none)
Table 6, "Data type correspondences for Fortran"	(none)
Table 7, "Data type correspondences for MUMPS"	(none)
Table 8, "Data type correspondences for Pascal"	(none)
Table 9, "Data type correspondences for PL/I"	(none)

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

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from ISO/IEC 9075:1992
Table 10, "Data types of <key word>s used in SQL item descriptor areas"	Table 17, "Data types of <key word>s used in SQL item descriptor areas"
Table 11, "Codes used for input/output SQL parameter modes in Dynamic SQL"	(none)
Table 12, "<identifier>s for use with <get diagnostics statement>"	Table 21, "<identifier>s for use with <get diagnostics statement>"
Table 13, "SQL-statement character codes for use in the diagnostics area"	Table 22, "SQL-statement character codes for use in the diagnostics area"
Table 14, "SQLSTATE class and subclass values"	Table 23, "SQLSTATE class and subclass values"

3.4 Object identifier for Database Language SQL

Function

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

Format

```
<SQL object identifier> ::=
    <SQL provenance> <SQL variant>
    | <SQL provenance> <SQL variant> <SQL parts>

<SQL parts> ::=
    4 | sqlpsml1996 <left paren> 4 <right paren>
```

Syntax Rules

- 1) Insert this SR <SQL parts> shall not be specified if the <SQL edition> is specified as <1987> or <1989>.

4 Concepts

4.1 SQL-server Modules

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

An <SQL-server module definition> contains an <SQL-server module name>, an optional <SQL-server module character set specification>, an optional <SQL-server module schema clause>, an optional <SQL-server module path specification>, zero or more declared local temporary tables specified by <temporary table declaration>s, and one or more <SQL-invoked routine>s. The <SQL-server module name> of an SQL-server module is a <qualified name>. The character set specified by the <SQL-server module character set specification> identifies the character repertoire used for expressing the names of schema objects used in the <SQL-server module definition>. The <default schema name> specified by the <SQL-server module schema clause> identifies the schema name used for implicit qualification of unqualified names appearing in the <SQL-server module definition>. The SQL-invoked routines of an SQL-server module are invoked only from SQL-statements.

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

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

- The SQL-server module name of the SQL-server module.
- The descriptor of the character set used for representing the SQL-server module's <identifier>s and <character string literal>s.
- The default schema name used for implicit qualification of unqualified names in the SQL-server module.
- The SQL-server module authorization identifier of the SQL-server module.
- The list of schema names contained in the <SQL-server module path specification>.
- The table descriptor of every local temporary table declared in the SQL-server module.
- The descriptor of every SQL-invoked routine contained in the SQL-server module.
- The text of the <SQL-server module definition>.

4.2 SQL-invoked routines

An *SQL-invoked routine* is an SQL-invoked procedure or an SQL-invoked function. An SQL-invoked routine comprises at least a <routine name>, a sequence of <SQL parameter declaration>s, and a <routine body>.

An SQL-invoked routine may be an element of an SQL-schema or of an SQL-server module. An SQL-invoked routine that is an element of an SQL-schema is called a *schema-level routine*.

An *SQL-invoked procedure* is an SQL-invoked routine that is invoked from an SQL <call statement>. An SQL-invoked procedure may have input SQL parameters, output SQL parameters, and SQL parameters that are both input SQL parameters and output SQL parameters. The format of an SQL-invoked procedure is specified by <SQL-invoked procedure> (see Subclause 10.18, “<SQL-invoked routine>”).

An *SQL-invoked function* is an SQL-invoked routine that is invoked as part of an SQL <value expression>. An SQL-invoked function has only input SQL parameters and returns its results as the value of its invocation. The format of an SQL-invoked function is specified by <SQL-invoked function> (see Subclause 10.18, “<SQL-invoked routine>”).

An SQL-invoked routine can be an *SQL routine* or an *external routine*. An SQL routine is an SQL-invoked routine whose <language clause> specifies SQL. The <routine body> of an SQL routine is an <SQL procedure statement>; the <SQL procedure statement> forming the <routine body> can be any SQL-statement, including an <SQL control statement>, but excluding an <SQL schema statement>, <SQL connection statement>, <SQL dynamic statement>, or <SQL transaction statement>.

An external routine is one whose <language clause> does not specify SQL. The <routine body> of an external routine is an <external body reference> whose <external routine name> identifies a program written in some standard programming language other than SQL.

An SQL-invoked routine is uniquely identified by a <specific name>, called the *specific name* of the SQL-invoked routine.

An invocation of an SQL-invoked routine specifies the <routine name> of the SQL-invoked routine and supplies a sequence of SQL argument values corresponding to the <SQL parameter declaration>s of the SQL-invoked routine. A *subject routine* is the SQL-invoked routine that may be invoked by a <routine invocation>. Such an invocation assigns the SQL arguments to the SQL parameters of one of the subject routines and causes the <routine body> of that subject routine to be executed. If the SQL-invoked routine is an SQL routine, then the <routine body> is an <SQL procedure statement> that is executed according to the General Rules of <SQL procedure statement>. If the SQL-invoked routine is an external routine, then the <routine body> identifies a program written in some standard programming language other than SQL that is executed according to the rules of that standard programming language.

The <routine body> of an SQL-invoked routine is always executed under the same SQL-session from which the SQL-invoked routine was invoked. Before the execution of the <routine body>, a new context for the current SQL-session is created and the values of the current context preserved. When the execution of the <routine body> completes the original context of the current SQL-session is restored.

If the SQL-invoked routine is an external routine, then an effective SQL parameter list is constructed before the execution of the <routine body>. The effective SQL parameter list has different entries depending on the parameter passing style of the SQL-invoked routine. The value of each entry in the effective SQL parameter list is set according to the General Rules of Subclause 9.1, “<routine invocation>”, and passed to the program identified by the <routine body> according to the rules of Subclause 11.5, “Data type correspondences”. After the execution of that program, if the

parameter passing style of the SQL-invoked routine is SQL, then the SQL implementation obtains the values for output parameters (if any), the value (if any) returned from the program, the value of the SQLSTATE, and the value of the message text (if any) from the values assigned by the program to the effective SQL parameter list. If the parameter passing style of the SQL-invoked routine is GENERAL, then such values are obtained in an implementation-defined manner.

Different SQL-invoked routines can have equivalent <routine name>s. No two SQL-invoked functions in the same schema are allowed to have the same signature. No two SQL-invoked procedures in the same schema are allowed to have the same name and the same number of parameters. *Subject routine determination* is the process for choosing the actual subject routine to be invoked for a given invocation of an SQL-invoked routine with a given <routine name>. Subject routine determination for SQL-invoked functions considers the data types of *all* of the arguments to the invocation of the SQL-invoked function in order from left to right. Where there is not an exact match between the data types of the arguments and the data types of the parameters, type precedence lists are used to determine the closest match. See Subclause 8.1, "Subject routine determination".

If a <routine invocation> is contained in a <query expression> of a view, a check constraint or an assertion, or in an <SQL-invoked routine>, then the subject routines for that invocation are determined at the time the view is created, the check constraint is defined, the assertion is created, or the SQL-invoked routine is created. Thus, the same SQL-invoked routines are invoked whenever the view is used, the check constraint or assertion is evaluated, or the SQL-invoked routine is invoked.

All <item reference>s in the <routine body> of an SQL routine are resolved to identify a particular column, SQL parameter, or SQL variable at the time that the SQL routine is created. Thus, the same columns, SQL parameters, and SQL variables are referenced whenever the SQL routine is invoked.

An SQL-invoked routine is either *deterministic* or *possibly non-deterministic*. An SQL-invoked function that is deterministic always returns the same return value for a given list of SQL argument values. An SQL-invoked procedure that is deterministic always returns the same values in its output and inout SQL parameters for a given list of SQL argument values. An SQL-invoked routine is possibly non-deterministic if, during invocation of that SQL-invoked routine, an implementation might, at two different times when the state of the SQL-data is the same, produce unequal results due to General Rules that specify implementation-dependent behavior.

An external routine either *does not possibly contain SQL* or *possibly contains SQL*. Only an external routine that possibly contains SQL may execute SQL-statements during its invocation.

An SQL-invoked routine may or may not *possibly read SQL-data*. Only an SQL-invoked routine that possibly reads SQL-data may read SQL-data during its invocation.

An SQL-invoked routine may or may not *possibly modify SQL-data*. Only an SQL-invoked routine that possibly modifies SQL-data may modify SQL-data during its invocation.

An SQL-invoked routine has a *routine authorization identifier*, which is (directly or indirectly) the authorization identifier of the owner of the schema that contains the SQL-invoked routine at the time that the SQL-invoked routine is created. An SQL-invoked routine that is an external routine also has an *external routine authorization identifier*, which is the <module authorization identifier>, if any, of the <module> contained in the external program identified by the <routine body> of the external routine. If that <module> does not specify a <module authorization identifier>, then the *external routine authorization identifier* is an implementation-defined authorization identifier.

For SQL routines, the routine authorization identifier acts as the current <authorization identifier> for privilege determination for the SQL objects, if any, contained in the SQL routine during the execution of that SQL routine. For external routines, the external routine authorization identifier

acts as the current <authorization identifier> for privilege determination for the SQL objects, if any, contained in the external routine during the execution of that external 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 module.

An SQL-invoked routine that is an external routine also has an *external routine SQL-path*, which is derived from the <module path specification>, if any, of the <module> contained in the external program identified by the routine body of the external routine. If that <module> does not specify a <module path specification>, then the *external routine SQL-path* is an implementation-defined SQL-path. For both SQL and external routines, the SQL-path of the current SQL-session is used to determine the search order for the subject routines of a <routine invocation> whose <routine name> does not contain a <schema name> if the <routine invocation> is contained in a <preparable statement> or in a <direct SQL statement>. SQL routines use the routine SQL-path to determine the search order for the subject routines of a <routine invocation> whose <routine name> does not contain a <schema name> if the <routine invocation> is not contained in a <preparable statement> or in a <direct SQL statement>. External routines use the external routine SQL-path to determine the search order for the subject routines of a <routine invocation> whose <routine name> does not contain a <schema name> if the <routine invocation> is not contained in a <preparable statement> or in a <direct SQL statement>.

An SQL-invoked routine is described by a routine descriptor. A routine descriptor contains:

- The routine name of the SQL-invoked routine.
- The <specific name> of the SQL-invoked routine.
- The routine authorization identifier of the SQL-invoked routine.
- The routine SQL-path of the SQL-invoked routine.
- The name of the language in which the body of the SQL-invoked routine is written.
- For each of the SQL-invoked routine's SQL parameters, the <SQL parameter name>, if it is specified, the <data type>, the ordinal position, and an indication of whether the SQL parameter is an input SQL parameter, an output SQL parameter, or both an input SQL parameter and an output SQL parameter.
- An indication of whether the SQL-invoked routine is an SQL-invoked function or an SQL-invoked procedure.
- An indication of whether the SQL-invoked routine is a schema-level routine.
- An indication of whether the SQL-invoked routine is deterministic or possibly non-deterministic.
- Indications of whether the SQL-invoked routine possibly modifies SQL-data, possibly reads SQL-data, possibly contains SQL, and does not possibly contain SQL.
- If the SQL-invoked routine is an SQL-invoked function, then the <returns data type> of the SQL-invoked function.
- If the SQL-invoked routine is an SQL routine, then the SQL routine body of the SQL-invoked routine.

- If the SQL-invoked routine is an external routine, then:
 - The <external routine name> of the external routine.
 - The <parameter style> of the external routine.
 - If the external routine specifies a <result cast>, then an indication that it specifies a <result cast> and the <data type> specified in the <result cast>.
 - The external routine authorization identifier of the external routine.
 - The external routine SQL-path of the external routine.
 - The effective SQL parameter list of the external routine.
- If the SQL-invoked routine is a schema-level routine, then the <schema name> of the schema that includes the SQL-invoked routine.
- 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

An SQL-path is a list of one or more <schema name>s that determines the search order for the subject routine of a <routine invocation> whose <routine name> does not contain a <schema name>. If the specification of an SQL-path does not specify a schema whose schema name is INFORMATION_SCHEMA, then INFORMATION_SCHEMA is assumed to precede all of the specified schema names.

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

4.3.1 Type conversions and mixing of data types

insert this paragraph Two data types are said to be *compatible* if they are mutually assignable and their descriptors include the same data type name.

NOTE 3 – The data types CHARACTER(*n*) CHARACTER SET *some-character-set* and CHARACTER(*n*) CHARACTER SET *different-character-set* have descriptors that include the same data type name (CHARACTER), but are not mutually assignable; therefore, they are not compatible.

4.4 Tables

4.4 Tables

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

Insert after 16th paragraph A declared local temporary table that is declared in a module is a named table defined by a <temporary table declaration> that is effectively materialized the first time any <procedure> in the <module> that contains the <temporary table declaration> is executed. A declared local temporary table is accessible only by <procedure>s in the <module> that contains the <temporary table declaration>. The effective <schema name> of the <qualified name> of the declared local temporary table may be thought of as the implementation-dependent SQL-session identifier associated with the SQL-session and a unique implementation-dependent name associated with the <module> that contains the <temporary table declaration>.

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

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

Insert after 16th paragraph All references to a declared local temporary table are prefixed by "MODULE".

4.5 Integrity constraints

Replace 2nd paragraph A <query expression> or <query specification> is *possibly non-deterministic* if, when evaluating that <query expression> or <query specification>, an implementation might, at two different times when the state of the SQL-data is the same, produce unequal results due to General Rules that specify implementation-dependent behavior.

Replace 3rd paragraph No integrity constraint shall be defined using a possibly non-deterministic <query specification> or <query expression>, or a <routine invocation> whose subject routines include an SQL-invoked routine that is a possibly non-deterministic routine.

4.6 SQL-schemas

Replace 2nd paragraph In this International standard, the term "schema" is used only in the sense of SQL-schema. Each component descriptor is either a domain descriptor, a base table descriptor, a view descriptor, a constraint descriptor, a privilege descriptor, a character set descriptor, a collation descriptor, a translation descriptor, 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.

An SQL-schema additionally includes:

- The SQL-path of the SQL-schema.

4.7 Parameters

4.7.1 Status parameters

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

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

4.8 Diagnostics area

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

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

4.9 Cursors

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

4.10 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 SQLEXCEPTION, SQLWARNING, or NOT FOUND SQLSTATE class. All other <handler declaration>s are *specific <handler declaration>s*.

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

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

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

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

If there is no most appropriate handler and the condition is an exception condition, then the SQL-statement raising the exception condition is terminated with that exception condition. This type of exception condition is called an *unhandled exception condition*. Unhandled exception conditions are examined at the next visible scope for handling. If an exception condition remains unhandled at the outer <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 3.3.4.1, "Exceptions", in ISO/IEC 9075:1992. This type of completion condition is called an *unhandled completion condition*.

A handler type specifies CONTINUE, EXIT, or UNDO.

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

- Execute the handler action.
- Return control to the <compound statement> from which it was invoked and execute all other SQL-statements following the one that raised the condition.

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

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

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

- Roll back all of the changes to SQL-data or to schemas by the execution of every SQL-statement contained in the SQL-statement list of the <compound statement> at the scope of the handler and cancel any <SQL procedure statement>s triggered by the execution of such statements.
- Execute the handler action.
- Return control to the end of the <compound statement> for which the handler was created.

If a <handler action> completes with a completion condition: *successful completion*, then it was able to resolve the condition, and execution resumes as specified in Subclause 13.4, “<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.

4.11 SQL-statements

4.11.1 SQL-statements classified by function

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

- SQL-control statements
- SQL-control declarations

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

- <SQL-server module definition>
- <drop module statement>
- <schema routine>
- <drop routine statement>

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

- <call statement>
- <return statement>
- <compound statement>
- <case statement>
- <if statement>
- <leave statement>
- <loop statement>
- <while statement>
- <repeat statement>
- <for statement>
- <assignment statement>

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

- <condition declaration>
- <handler declaration>

— <SQL variable declaration>

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

— <set path statement>

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

— <signal statement>

— <resignal statement>

4.11.2 Embeddable SQL-statements

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

— All SQL-control statements

4.11.3 Preparable and immediately executable SQL-statements

Insert this paragraph The following are additional SQL-statements that are preparable and immediately executable:

— <call statement>

4.11.4 Directly executable SQL-statements

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

— All SQL-control statements

4.11.5 SQL-statements and transaction states

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

- <for statement>

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

- <leave statement>

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

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

- <leave statement>

Insert this paragraph If an <SQL control statement> causes the evaluation of a <subquery> and there is no current SQL-transaction, then an SQL-transaction is initiated before evaluation of the <subquery>.

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

4.11.6 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 13.3, “<compound statement>”.

4.11.7 SQL-statement atomicity

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

An atomic execution context is said to be *active* during the execution of an atomic SQL-statement or evaluation of a <subquery>. Within one atomic execution context, another atomic execution context may become active.

An SQL-transaction cannot be explicitly terminated within an atomic execution context. If the execution of an atomic SQL-statement is unsuccessful, then the changes to SQL-data or schemas made by the SQL-statement are canceled.

4.12 Privileges

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

Insert this paragraph An additional <action> that can be specified is:

— EXECUTE

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

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

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

NOTE 4 – “schema-level routine” is defined in Subclause 10.18, “<SQL-invoked routine>”.

4.13 SQL-sessions

Insert this paragraph An SQL-session has an SQL-path that is used to effectively qualify unqualified <routine name>s that are immediately contained in <routine invocation>s that are contained in <preparable statement>s when those statements are prepared in the current SQL-session by either an <execute immediate statement> or a <prepare statement>, or are contained in <direct SQL statement>s when those statements are invoked directly. The SQL-path is initially set to an implementation-defined value, but can subsequently be changed by the successful execution of a <set path statement>.

Insert this paragraph The text defining the SQL-path can be referenced by using the <general value specification> CURRENT_PATH.

Insert this paragraph At any time during an SQL-session, *containing SQL* is said to be *permitted* or *not permitted*. Similarly, *reading SQL-data* is said to be *permitted* or *not permitted* and *modifying SQL-data* is either *permitted* or *not permitted*.

Insert this paragraph The SQL-session context additionally contains:

- The text defining the current default SQL-path.
- A *routine execution context*, comprising:
 - An indication as to whether or not an SQL-invoked routine is active.
 - An indication as to whether or not containing SQL is permitted.
 - An indication as to whether or not reading SQL-data is permitted.
 - An indication as to whether or not modifying SQL-data is permitted.
 - An identification of the SQL-invoked routine that is active.
 - The routine SQL-path derived from the routine SQL-path if the SQL-invoked routine that is active is an SQL routine and from the external routine SQL-path if the SQL-invoked routine that is active is an external routine.

Insert this paragraph An SQL-invoked routine is *active* as soon as an SQL-statement executed by an SQL-agent causes invocation of an SQL-invoked routine and ceases to be active when execution of that invocation is complete.

Insert this paragraph When an SQL-agent causes the invocation of an SQL-invoked routine, a new context for the current SQL-session is created and the values of the current context are preserved. When the execution of that SQL-invoked routine completes, the original context of the current SQL-session is restored and some SQL-session attributes are reset.

Insert this paragraph When the routine execution context of the SQL-session indicates that an SQL-invoked routine is active, then the routine SQL-path included in the routine execution context of the SQL-session is used to effectively qualify unqualified <routine name>s that are immediately contained in <routine invocation>s that are not contained in a <preparable statement> or in a <direct SQL statement>.

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 9075:1992  
  
    | ATOMIC  
  
    | CONDITION_IDENTIFIER  
  
    | GENERAL  
  
    | MODIFIES  
  
    | PARAMETER_NAME  
  
    | READS | ROUTINE_CATALOG | ROUTINE_NAME | ROUTINE_SCHEMA  
  
    | SPECIFIC_NAME | STYLE  
  
<reserved word> ::=  
    !! All alternatives from ISO/IEC 9075:1992  
  
    | CALL | CONDITION | CONTAINS | CURRENT_PATH  
  
    | DETERMINISTIC | DO  
  
    | ELSEIF | EXIT  
  
    | FUNCTION  
  
    | HANDLER  
  
    | IF | INOUT  
  
    | LEAVE | LOOP  
  
    | OUT  
  
    | PARAMETER | PATH  
  
    | REPEAT | RESIGNAL | RETURN | RETURNS | ROUTINE  
  
    | SIGNAL | SPECIFIC | SQLEXCEPTION | SQLWARNING
```

| UNDO | UNTIL
| WHILE

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

5.2 Names and identifiers

Function

Specify names.

Format

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

<SQL variable name> ::=
    <identifier>

<condition name> ::=
    <identifier>

<routine name> ::=
    <qualified name>

<specific name> ::=
    <qualified name>

<external routine name> ::=
    <identifier>
    | <character string literal>

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

Syntax Rules

- 1) **Replace SR1)** If a <character set specification> is not specified in an <identifier>, then the set of characters contained in the <identifier> shall be wholly contained in either <SQL language character> or the character repertoire indicated by:
Case:
 - a) If the <identifier> is contained, without an intervening <schema definition> or <SQL-server module definition>, in a <preparable statement> that is prepared in the current SQL-session by an <execute immediate statement> or a <prepare statement> or in a <direct SQL statement> that is invoked directly, then the default character set name for the SQL-session,
 - b) If the <identifier> is contained in an <SQL-server module definition>, then the <SQL-server module character set specification>,
 - c) If the <identifier> is contained in a <schema definition>, then the <schema character set specification>,
 - d) If the <identifier> is contained in a <module>, then the <module character set specification>.
- 2) **Replace SR8)** If a <qualified name> does not contain a <schema name>, then

Case:

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

Access Rules

No additional Access Rules.

General Rules

- 1) Replace GR10) A <parameter name> identifies a host parameter.
- 2) Insert after GR10) An <SQL parameter name> identifies an SQL parameter.
- 3) Insert this GR) An <SQL-server module name> identifies an SQL-server module.
- 4) Insert this GR) An <SQL variable name> identifies an SQL variable.
- 5) Insert this GR) A <condition name> identifies an exception condition or a completion condition and optionally a corresponding SQLSTATE value.
- 6) Insert this GR) A <routine name> identifies one or more SQL-invoked routines.
- 7) Insert this GR) A <specific name> identifies an SQL-invoked routine.
- 8) Insert this GR) An <external routine name> identifies a program written in a standard programming language other than SQL.

6 Scalar expressions

6.1 <value specification> and <target specification>

Function

Specify one or more values, parameters, or variables.

Format

```
<general value specification> ::=  
    !! All alternatives from ISO/IEC 9075:1992  
    | CURRENT_PATH  
    | <routine invocation>  
    | <SQL parameter or SQL variable reference>  
  
<simple value specification> ::=  
    !! All alternatives from ISO/IEC 9075:1992  
    | <SQL parameter or SQL variable reference>  
  
<target specification> ::=  
    !! All alternatives from ISO/IEC 9075:1992  
    | <SQL parameter or SQL variable reference>  
  
<simple target specification> ::=  
    !! All alternatives from ISO/IEC 9075:1992  
    | <SQL parameter or SQL variable reference>
```

Syntax Rules

- 1) Insert this SR If a <routine invocation> is specified, then the subject SQL-invoked routine shall be an SQL-invoked function.
- 2) Insert this SR The data type of CURRENT_PATH is character string. Whether the character string is fixed-length or variable-length, and its length if it is fixed-length or its maximum length if it is variable-length, are implementation-defined. The character set of the character string is SQL_TEXT.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert this GR The value specified by CURRENT_PATH is the value of the SQL-path of the current SQL-session.

6.1 <value specification> and <target specification>

- 2) Insert this GR The value specified by CURRENT_PATH is a <schema name list> where <catalog name>s are <delimited identifier>s and the <unqualified schema name>s are <delimited identifier>s. Each <schema name> is separated from the preceding <schema name> by a <comma> with no intervening <space>s.
- 3) Replace GR12 If a <simple value specification> evaluates to the null value, then an exception condition is raised: *data exception — null value not allowed*.

6.2 <column reference>

Function

Reference a column

Format

```
<column reference> ::=  
    <item reference>
```

Syntax Rules

- 1) Replace SR1 A <column reference> *CR* shall be an <item reference> that is a column reference. Let *CN* be the <column name> contained in *CR*.
- 2) Replace SR2 If *CR* contains a <qualifier>, let *Q* be that <qualifier>.
- 3) Replace SR3) through SR7) If *CR* does not contain a <qualifier>, then let *Q* be the implicit qualifier determined by the Syntax Rules of Subclause 6.3, "<item reference>", in this part of ISO/IEC 9075.

Access Rules

No additional Access Rules.

General Rules

- 1) Delete all GRs

6.3 <item reference>

6.3 <item reference>

Function

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

Format

```

<item reference> ::=
    [ <item qualifier> <period> ] <item name>

<item qualifier> ::=
    <qualifier>

<item name> ::=
    <column name>
    | <SQL parameter name>
    | <SQL variable name>

<SQL parameter or SQL variable reference> ::=
    <item reference>

```

Syntax Rules

- 1) Let *IR* be an <item reference> and let *IN* be the <item name> contained in *IR*.
- 2) If *IR* contains an <item qualifier> *IQ*, then *IR* shall appear within the scope of one or more exposed <table name>s or <correlation name>s that are equivalent to *IQ*. If there is more than one such exposed <table name> or <correlation name>, then the one with the innermost scope is specified.
 - a) Let *V* be the table identified by *IQ*.
 - b) *V* shall include a column whose <column name> is equivalent to *IN*.
 - c) If *V* is the table identified by a <table reference> in a <joined table> *JT*, then *IN* shall not be a common column name in *JT*.

NOTE 5 – “Common column name” is defined in Subclause 7.5, “<joined table>”, in ISO/IEC 9075:1992.
- 3) If *IR* does not contain an <item qualifier>, then *IR* shall be contained within the scope of one or more exposed <table name>s or <correlation name>s whose associated tables include a column whose <identifier> is equivalent to *IN* or within the scope of one or more <routine name>s or <beginning label>s whose associated <SQL parameter list> or <local declaration list> includes an SQL parameter or SQL variable whose <identifier> is equivalent to *IN*. Let the phrase *possible qualifiers* denote those exposed <table name>s, <correlation name>s, <routine name>s, and <statement label>s.
 - a) Case:
 - i) If the innermost scope contains exactly one possible qualifier, then the qualifier *IQ* equivalent to that unique exposed <table name>, <correlation name>, <routine name>, or <beginning label> is implicit.

ii) If there is more than one possible qualifier within the innermost scope, then:

- 1) Each possible qualifier shall be a <table name> or a <correlation name> of a <table reference> that is directly contained in a <joined table> *JT*.
- 2) *IN* shall be a common column name in *JT*.
NOTE 6 – “Common column name” is defined in Subclause 7.5, “<joined table>”, in ISO/IEC 9075:1992.
- 3) The implicit qualifier *IQ* is implementation-dependent. The scope of *IQ* is that which *IQ* would have had if *JT* had been replaced by the <table reference>:

(*JT*) AS *IQ*

b) Let *V* be the table, SQL parameter list, or <local declaration list> associated with *IQ*.

4) Case:

a) If *IQ* is a <table name> or <correlation name>, then *IR* is a *column reference*. *IN* shall uniquely identify a column of the table *V*. Let *R* be that column.

b) If *IQ* is the <routine name> of an SQL-invoked routine, then

Case:

- i) If *IN* is an SQL variable of *V*, then *IR* is an *SQL variable reference*. Let *R* be that SQL variable.
- ii) Otherwise, *IR* is an *SQL parameter reference*. *IN* is an SQL parameter of *V*. Let *R* be that SQL parameter.

c) If *IQ* is a <beginning label>, then *IR* is an *SQL variable reference*. *IN* is an SQL variable of *V*. Let *R* be that SQL variable.

5) The data type of *IR* is the data type of *R*.

6) If the data type of *IR* is character string, then *IR* has the *Implicit* coercibility attribute and its collating sequence is the default collating sequence for the column, SQL parameter, or SQL variable *R*.

7) An <SQL parameter or SQL variable reference> shall be an <item reference> that is an SQL parameter reference or an SQL variable reference.

Access Rules

None.

General Rules

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

6.4 <datetime value function>

Function

Specify a function yielding a value of type datetime.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

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

6.5 <case expression>

Function

Specify a conditional value.

Format

No additional Format items.

Syntax Rules

- 1) Insert this SR If <case specification> specifies <simple case>, then the <case operand> 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.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

6.6 <value expression>

Function

Specify a value.

Format

```
<value expression primary> ::=  
    !! All alternatives from ISO/IEC 9075:1992  
    | <SQL parameter or SQL variable reference>
```

Syntax Rules

- 1) Replace SR2 If the data type of a <value expression primary> is character string, then the collating sequence and coercibility attribute of the <value expression primary> are the collating sequence and coercibility attribute of the <unsigned value specification>, <column reference>, <SQL parameter or SQL variable reference>, <set function specification>, <scalar subquery>, <case expression>, <value expression>, or <cast specification> immediately contained in the <value expression primary>.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.