

Australian Standard™

**Information technology—Database
languages—SQL**

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



This Australian Standard was prepared by Committee IT-027, Data Management and Interchange. It was approved on behalf of the Council of Standards Australia on 16 May 2005. This Standard was published on 16 June 2005.

The following are represented on Committee IT-027:

Association of Superannuation Funds of Australia
Australia Post
Australia Bankers Association
Australian Bureau of Statistics
Australian Computer Society
Australian Customs Service
Australian Electoral Commission
Australian Electric and Electronic Manufacturers Association
Australian Industry Group
Australian Information Industry Association
Australian Institute of Health and Welfare
Australian Taxation Office
Centrelink
Department of Defence
Department of Immigration, Multicultural and Indigenous Affairs
Health Insurance Commission
Data Management Association Australia

Keeping Standards up-to-date

Standards are living documents which reflect progress in science, technology and systems. To maintain their currency, all Standards are periodically reviewed, and new editions are published. Between editions, amendments may be issued. Standards may also be withdrawn. It is important that readers assure themselves they are using a current Standard, which should include any amendments which may have been published since the Standard was purchased.

Detailed information about Standards can be found by visiting the Standards Web Shop at www.standards.com.au and looking up the relevant Standard in the on-line catalogue.

Alternatively, the printed Catalogue provides information current at 1 January each year, and the monthly magazine, *The Global Standard*, has a full listing of revisions and amendments published each month.

Australian Standards™ and other products and services developed by Standards Australia are published and distributed under contract by SAI Global, which operates the Standards Web Shop.

We also welcome suggestions for improvement in our Standards, and especially encourage readers to notify us immediately of any apparent inaccuracies or ambiguities. Contact us via email at mail@standards.org.au, or write to the Chief Executive, Standards Australia, GPO Box 5420, Sydney, NSW 2001.

This Standard was issued in draft form for comment as DR 05145.

Australian Standard™

Information technology—Database languages—SQL

Part 4: Persistent Stored Modules (SQL/PSM)

Originated as AS 3968.4—1998.
Revised and redesignated as AS ISO/IEC 9075.4—2005.

COPYRIGHT

© Standards Australia

All rights are reserved. No part of this work may be reproduced or copied in any form or by any means, electronic or mechanical, including photocopying, without the written permission of the publisher.

Published by Standards Australia GPO Box 5420, Sydney, NSW 2001, Australia

ISBN 0 7337 6766 4

PREFACE

This Standard was prepared by the Standards Australia Committee IT-027, Data Management and Interchange to supersede AS 3968.4—1998.

This Standard is identical with, and has been reproduced from ISO/IEC 9075-4:2003, *Information technology—Database languages—SQL—Part 4: Persistent Stored Modules (SQL/PSM)*.

The objective of this Standard is to provide database designers, administrators and developers with syntax and semantics of a database language for declaring and maintaining persistent database language routines in SQL-server modules.

This Standard is Part 4 of AS 9075, *Information technology—Database languages—SQL*, which is published in parts as follows:

- Part 1: Framework (SQL/Framework)
- Part 2: Foundation (SQL/Foundation)
- Part 3: Call-Level Interface (SQL/CLI)
- Part 4: Persistent Stored Modules (SQL/PSM) (this Standard)
- Part 5: Host Language Bindings (SQL/Bindings)
- Part 9: Management of External Data (SQL/MED)
- Part 10: Object Language Bindings (SQL/OLB)
- Part 11: Information and Definition Schemas (SQL/Schemata)
- Part 13: SQL Routines and Types Using the Java™ Programming Language (SQL/JRT)
- Part 14: XML-Related Specifications (SQL/XML)

As this Standard is reproduced from an international standard, the following applies:

- (a) Its number appears on the cover and title page while the international standard number appears only on the cover.
- (b) In the source text ‘this part of ISO/IEC 9075’ should read ‘this Australian Standard’.
- (c) A full point substitutes for a comma when referring to a decimal marker.

References to International Standards should be replaced by references to Australian or Australian/New Zealand Standards, as follows:

<i>Reference to International Standard</i>		<i>Australian Standard</i>	
ISO/IEC		AS	
9075	Information technology—Database languages—SQL	9075	Information technology—Database languages—SQL
9075-1	Part 1: Framework (SQL/Framework)	9075.1	Part 1: Framework (SQL/Framework)
9075-2	Part 2: Foundation (SQL/Foundation)	9075.2	Part 2: Foundation (SQL/Foundation)
9075-11	Part 11: Information and Definition Schemas (SQL/Schemas)	9075.11	Part 11: Information and Definition Schemas (SQL/Schemas)

CONTENTS

Page

1	Scope.....	1
2	Normative references.....	3
2.1	JTC1 standards.....	3
3	Definitions, notations, and conventions.....	5
3.1	Conventions.....	5
3.1.1	Use of terms.....	5
3.1.1.1	Other terms.....	5
4	Concepts.....	7
4.1	SQL-server modules.....	7
4.2	Tables.....	8
4.2.1	Types of tables.....	8
4.3	SQL-invoked routines.....	8
4.3.1	Overview of SQL-invoked routines.....	8
4.3.2	Execution of SQL-invoked routines.....	8
4.4	SQL-schemas.....	9
4.5	SQL-paths.....	9
4.6	Host parameters.....	9
4.6.1	Status parameters.....	9
4.7	Diagnostics area.....	10
4.8	Cursors.....	10
4.8.1	General description of cursors.....	10
4.9	Condition handling.....	10
4.10	SQL-statements.....	12
4.10.1	Classes of SQL-statements.....	12
4.10.2	SQL-statements classified by function.....	12
4.10.2.1	SQL-schema statements.....	12
4.10.2.2	SQL-control statements.....	13
4.10.2.3	SQL-control declarations.....	13
4.10.2.4	SQL-diagnostics statements.....	13
4.10.3	Embeddable SQL-statements.....	14
4.10.4	Preparable and immediately executable SQL-statements.....	14
4.10.5	Directly executable SQL-statements.....	14
4.10.6	Iterated SQL-statements.....	15
4.10.7	SQL-statements and transaction states.....	15

4.10.8	Compound statements.	15
4.10.9	SQL-statement atomicity and statement execution contexts.	16
4.11	Basic security model.	16
4.11.1	Privileges.	16
4.12	SQL-sessions.	17
4.12.1	General description of SQL-sessions.	17
5	Lexical elements.	19
5.1	<token> and <separator>.	19
5.2	Names and identifiers.	21
6	Scalar expressions.	25
6.1	<value specification> and <target specification>.	25
6.2	<identifier chain>.	27
6.3	<next value expression>.	29
6.4	<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	<sqlstate value>.	35
9	Schema definition and manipulation.	37
9.1	<schema definition>.	37
9.2	<drop schema statement>.	38
9.3	<default clause>.	40
9.4	<drop column scope clause>.	41
9.5	<drop column definition>.	43
9.6	<drop table constraint definition>.	44
9.7	<drop table statement>.	45
9.8	<view definition>.	46
9.9	<drop view statement>.	47
9.10	<drop domain statement>.	48
9.11	<drop character set statement>.	49
9.12	<drop collation statement>.	50
9.13	<drop transliteration statement>.	51
9.14	<assertion definition>.	52
9.15	<drop assertion statement>.	53
9.16	<trigger definition>.	54
9.17	<drop user-defined ordering statement>.	55
9.18	<SQL-server module definition>.	57
9.19	<drop module statement>.	60
9.20	<drop data type statement>.	61
9.21	<SQL-invoked routine>.	62

9.22	<drop routine statement>.....	64
9.23	<drop user-defined cast statement>.....	65
10	Access control.....	67
10.1	<grant statement>.....	67
10.2	<privileges>.....	68
10.3	<revoke statement>.....	69
11	SQL-client modules.....	73
11.1	Calls to an <externally-invoked procedure>.....	73
11.2	<SQL procedure statement>.....	74
12	Data manipulation.....	77
12.1	<open statement>.....	77
12.2	<fetch statement>.....	78
12.3	<close statement>.....	80
12.4	<select statement: single row>.....	81
12.5	<delete statement: positioned>.....	82
12.6	<update statement: positioned>.....	83
12.7	<temporary table declaration>.....	84
13	Control statements.....	87
13.1	<compound statement>.....	87
13.2	<handler declaration>.....	91
13.3	<condition declaration>.....	95
13.4	<SQL variable declaration>.....	96
13.5	<assignment statement>.....	97
13.6	<case statement>.....	102
13.7	<if statement>.....	105
13.8	<iterate statement>.....	107
13.9	<leave statement>.....	108
13.10	<loop statement>.....	109
13.11	<while statement>.....	111
13.12	<repeat statement>.....	113
13.13	<for statement>.....	115
14	Dynamic SQL.....	119
14.1	<prepare statement>.....	119
15	Embedded SQL.....	121
15.1	<embedded SQL host program>.....	121
16	Diagnostics management.....	123
16.1	<get diagnostics statement>.....	123
16.2	<signal statement>.....	126
16.3	<resignal statement>.....	128
17	Information Schema.....	131

17.1	MODULE_COLUMN_USAGE view.....	131
17.2	MODULE_PRIVILEGES view.....	132
17.3	MODULE_TABLE_USAGE view.....	133
17.4	MODULES view.....	134
17.5	PARAMETERS view.....	136
17.6	ROLE_MODULE_GRANTS view.....	137
17.7	ROUTINES view.....	138
17.8	Short name views.....	139
18	Definition Schema.....	141
18.1	MODULE_COLUMN_USAGE base table.....	141
18.2	MODULE_PRIVILEGES base table.....	143
18.3	MODULE_TABLE_USAGE base table.....	145
18.4	MODULES base table.....	146
18.5	ROUTINES base table.....	148
19	Status codes.....	149
19.1	SQLSTATE.....	149
20	Conformance.....	151
20.1	Claims of conformance to SQL/PSM.....	151
20.2	Additional conformance requirements for SQL/PSM.....	151
20.3	Implied feature relationships of SQL/PSM.....	151
Annex A	SQL Conformance Summary.....	153
Annex B	Implementation-defined elements.....	159
Annex C	Implementation-dependent elements.....	161
Annex D	Incompatibilities with ISO/IEC 9075:1999.....	163
Annex E	Defect reports not addressed in this edition of this part of ISO/IEC 9075.....	165
Annex F	SQL feature taxonomy.....	167
Index.....		169

TABLES

Page

1	<identifier>s for use with <get diagnostics statement>.....	123
2	SQL-statement codes.....	124
3	SQLSTATE class and subclass values.....	149
4	Implied feature relationships of SQL/PSM.....	151
5	Feature taxonomy for optional features.....	167

INTRODUCTION

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

- 1) [Clause 1, “Scope”](#), specifies the scope of this part of ISO/IEC 9075.
- 2) [Clause 2, “Normative references”](#), identifies additional standards that, through reference in this part of ISO/IEC 9075, constitute provisions of this part of ISO/IEC 9075.
- 3) [Clause 3, “Definitions, notations, and conventions”](#), defines the notations and conventions used in this part of ISO/IEC 9075.
- 4) [Clause 4, “Concepts”](#), presents concepts used in the definition of persistent stored modules.
- 5) [Clause 5, “Lexical elements”](#), defines a number of lexical elements used in the definition of persistent stored modules.
- 6) [Clause 6, “Scalar expressions”](#), defines a number of scalar expressions used in the definition of persistent stored modules.
- 7) [Clause 7, “Query expressions”](#), defines the elements of the language that produce rows and tables of data as used in persistent stored modules.
- 8) [Clause 8, “Additional common elements”](#), defines additional common elements used in the definition of persistent stored modules.
- 9) [Clause 9, “Schema definition and manipulation”](#), defines the schema definition and manipulation statements associated with the definition of persistent stored modules.
- 10) [Clause 10, “Access control”](#), defines facilities for controlling access to SQL-data.
- 11) [Clause 11, “SQL-client modules”](#), defines the facilities for using persistent stored modules.
- 12) [Clause 12, “Data manipulation”](#), defines data manipulation operations associated with persistent stored modules.
- 13) [Clause 13, “Control statements”](#), defines the control statements used with persistent stored modules.
- 14) [Clause 14, “Dynamic SQL”](#), defines the facilities for executing SQL-statements dynamically in the context of persistent stored modules.
- 15) [Clause 15, “Embedded SQL”](#), defines the host language embeddings.
- 16) [Clause 16, “Diagnostics management”](#), defines enhancements to the facilities used with persistent stored modules.
- 17) [Clause 17, “Information Schema”](#), defines the Information and Definition Schema objects associated with persistent stored modules.
- 18) [Clause 18, “Definition Schema”](#), defines base tables on which the viewed tables containing schema information depend.
- 19) [Clause 19, “Status codes”](#), defines SQLSTATE values related to persistent stored modules.

- 20) [Clause 20, “Conformance”](#), defines the criteria for conformance to this part of ISO/IEC 9075.
- 21) [Annex A, “SQL Conformance Summary”](#), is an informative Annex. It summarizes the conformance requirements of the SQL language.
- 22) [Annex B, “Implementation-defined elements”](#), is an informative Annex. It lists those features for which the body of this part of ISO/IEC 9075 states that the syntax, the meaning, the returned results, the effect on SQL-data and/or schemas, or any other behavior is partly or wholly implementation-defined.
- 23) [Annex C, “Implementation-dependent elements”](#), is an informative Annex. It lists those features for which the body of this part of ISO/IEC 9075 states that the syntax, the meaning, the returned results, the effect on SQL-data and/or schemas, or any other behavior is partly or wholly implementation-dependent.
- 24) [Annex D, “Incompatibilities with ISO/IEC 9075:1999”](#), is an informative Annex. It lists the incompatibilities between this edition of this part of ISO/IEC 9075 and ISO/IEC 9075-4:1996.
- 25) [Annex E, “Defect reports not addressed in this edition of this part of ISO/IEC 9075”](#), is an informative Annex. It describes reported defects in the previous edition of this part of ISO/IEC 9075 that remain in this edition.
- 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 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.

X

NOTES

AUSTRALIAN STANDARD

Information technology — Database languages — SQL —**Part 4:****Persistent Stored Modules (SQL/PSM)****1 Scope**

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

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

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

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

This page intentionally left blank.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

2.1 JTC1 standards

[Framework] ISO/IEC 9075-1:2003, *Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework)*.

[Foundation] ISO/IEC 9075-2:2003, *Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation)*.

[Schemata] ISO/IEC 9075-11:2003, *Information technology — Database languages — SQL — Part 11: Information and Definition Schemas (SQL/Schemata)*.

This page intentionally left blank.

3 Definitions, notations, and conventions

This Clause modifies Clause 3, “Definitions, notations, and conventions”, in ISO/IEC 9075-2.

3.1 Conventions

This Subclause modifies Subclause 3.3, “Conventions”, in ISO/IEC 9075-2.

3.1.1 Use of terms

This Subclause modifies Subclause 3.3.1, “Use of terms”, in ISO/IEC 9075-2.

3.1.1.1 Other terms

This Subclause modifies Subclause 3.3.1.1, “Other terms”, in ISO/IEC 9075-2.

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

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

This page intentionally left blank.

4 Concepts

This Clause modifies *Clause 4, “Concepts”, in ISO/IEC 9075-2.*

4.1 SQL-server modules

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

An <SQL-server module definition> contains an <SQL-server module name>, an optional <SQL-server module character set specification>, an optional <SQL-server module schema clause>, an optional <SQL-server module path specification>, zero or more 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 Tables

This Subclause modifies Subclause 4.14, “Tables”, in ISO/IEC 9075-2.

4.2.1 Types of tables

This Subclause modifies Subclause 4.14.2, “Types of tables”, in ISO/IEC 9075-2.

Insert this paragraph A declared local temporary table may be declared in an SQL-server module.

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

4.3 SQL-invoked routines

This Subclause modifies Subclause 4.27, “SQL-invoked routines”, in ISO/IEC 9075-2.

4.3.1 Overview of SQL-invoked routines

This Subclause modifies Subclause 4.27.1, “Overview of SQL-invoked routines”, in ISO/IEC 9075-2.

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

4.3.2 Execution of SQL-invoked routines

This Subclause modifies Subclause 4.27.3, “Execution of SQL-invoked routines”, in ISO/IEC 9075-2.

Replace 5th paragraph An SQL-invoked routine has a *routine SQL-path*, which is inherited from its containing SQL-server module or schema, the current SQL-session, or the containing SQL-client module.

Insert in the 7th paragraph — If the SQL-invoked routine is not a schema-level routine, then the <SQL-server module name> of the SQL-server module that includes the SQL-invoked routine and the <schema name> of the schema that includes the SQL-server module.

4.4 SQL-schemas

This Subclause modifies Subclause 4.20, “SQL-schemas”, in ISO/IEC 9075-2.

Insert into the 2nd paragraph

— An SQL-server module descriptor.

4.5 SQL-paths

This Subclause modifies Subclause 4.28, “SQL-paths”, in ISO/IEC 9075-2.

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

4.6 Host parameters

This Subclause modifies Subclause 4.29, “Host parameters”, in ISO/IEC 9075-2.

4.6.1 Status parameters

This Subclause modifies Subclause 4.29.2, “Status parameters”, in ISO/IEC 9075-2.

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

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

4.7 Diagnostics area

This Subclause modifies Subclause 4.30, “Diagnostics area”, in ISO/IEC 9075-2.

Insert this paragraph Information about a completion or exception condition is placed into one or more condition areas of the first diagnostics area before any handler is activated. The diagnostics area stack is then pushed so that the handler can access that information even while its own execution is causing the first diagnostics area to be modified.

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

4.8 Cursors

This Subclause modifies Subclause 4.32, “Cursors”, in ISO/IEC 9075-2.

4.8.1 General description of cursors

This Subclause modifies Subclause 4.32.1, “General description of cursors”, in ISO/IEC 9075-2.

Insert this paragraph For every <declare cursor> in a <compound statement>, a cursor is effectively created each time the <compound statement> is executed and, unless the cursor is an open result set cursor, 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 are optionally specified in <compound statement>s. The scope of a <handler declaration> specified in <compound statement> CS is CS, excluding every <SQL schema statement> contained in CS.

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

A *general* <handler declaration> is one that is associated with the <condition value>s SQL EXCEPTION, SQL WARNING, or NOT FOUND. All other <handler declaration>s are *specific* <handler declaration>s.

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

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

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

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

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

If there is no most appropriate handler and the condition is a completion condition, then execution is resumed as specified in [Subclause 6.3.3.7, “Exceptions”](#), in 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:

- Push the diagnostics area stack.
- Execute the handler action.
- Pop the diagnostics area stack.
- Cause the SQL-session to continue as it would have done if execution of the innermost executing statement that raised the condition had completed.

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

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

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

- Push the diagnostics area stack.
- Roll back all of the changes to SQL-data or to schemas by the execution of every SQL-statement contained in the SQL-statement list of the <compound statement> at the scope of the handler and cancel any <SQL procedure statement>s triggered by the execution of such statements.

- Pop the diagnostics area stack.
- Execute the handler action.
- Cause the SQL-session to continue as it would have done if execution of the <compound statement> for which the handler was created had completed.

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

This Subclause modifies Subclause 4.33, “SQL-statements”, in ISO/IEC 9075-2.

4.10.1 Classes of SQL-statements

This Subclause modifies Subclause 4.33.1, “Classes of SQL-statements”, in ISO/IEC 9075-2.

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

- SQL-control declarations

4.10.2 SQL-statements classified by function

This Subclause modifies Subclause 4.33.2, “SQL-statements classified by function”, in ISO/IEC 9075-2.

4.10.2.1 SQL-schema statements

This Subclause modifies Subclause 4.33.2.1, “SQL-schema statements”, in ISO/IEC 9075-2.

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

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

4.10.2.2 SQL-control statements

This Subclause modifies Subclause 4.33.2.6, “SQL-control statements”, in ISO/IEC 9075-2.

The following are additional SQL-control statements:

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

4.10.2.3 SQL-control declarations

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

- <condition declaration>
- <handler declaration>
- <SQL variable declaration>

4.10.2.4 SQL-diagnostics statements

This Subclause modifies Subclause 4.33.2.8, “SQL-diagnostics statements”, in ISO/IEC 9075-2.

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

- <signal statement>
- <resignal statement>

4.10.3 Embeddable SQL-statements

This Subclause modifies Subclause 4.33.6, “Embeddable SQL-statements”, in ISO/IEC 9075-2.

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

— All SQL-control statements

NOTE 1 — 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>.

4.10.4 Preparable and immediately executable SQL-statements

This Subclause modifies Subclause 4.33.7, “Preparable and immediately executable SQL-statements”, in ISO/IEC 9075-2.

Insert this paragraph Consequently, the following SQL-control statements are not preparable:

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

Insert this paragraph Consequently, the following SQL-control declarations are not preparable:

- <condition declaration>
- <handler declaration>
- <SQL variable declaration>

4.10.5 Directly executable SQL-statements

This Subclause modifies Subclause 4.33.8, “Directly executable SQL-statements”, in ISO/IEC 9075-2.

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

- All SQL-control statements

4.10.6 Iterated SQL-statements

The following are the iterated SQL-statements:

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

4.10.7 SQL-statements and transaction states

This Subclause modifies Subclause 4.33.4, “SQL-statements and transaction states”, in ISO/IEC 9075-2.

Insert this paragraph The following additional SQL-statement is a transaction-initiating SQL-statement:

- <for statement>

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

- <iterate statement>
- <leave statement>

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

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

4.10.8 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>”.

4.10.9 SQL-statement atomicity and statement execution contexts

This Subclause modifies Subclause 4.33.5, “SQL-statement atomicity and statement execution contexts”, in ISO/IEC 9075-2.

Add to the list of non-atomic SQL-statements

- <assignment statement>.
- <case statement>.
- <compound statement>, unless BEGIN ATOMIC is specified.
- <for statement>.
- <if statement>.
- <loop statement>.
- <repeat statement>.
- <while statement>.

4.11 Basic security model

This Subclause modifies Subclause 4.34, “Basic security model”, in ISO/IEC 9075-2.

4.11.1 Privileges

This Subclause modifies Subclause 4.34.2, “Privileges”, in ISO/IEC 9075-2.

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

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

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

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

NOTE 2 — “schema-level routine” is defined in Subclause 11.50, “<SQL-invoked routine>”, in ISO/IEC 9075-2.

4.12 SQL-sessions

This Subclause modifies Subclause 4.37, “SQL-sessions”, in ISO/IEC 9075-2.

4.12.1 General description of SQL-sessions

This Subclause modifies Subclause 4.37.1, “General description of SQL-sessions”, in ISO/IEC 9075-2.

Insert this paragraph Certain operations during an SQL-session SS are possible only when SS is in *condition handling mode*. This mode becomes in effect when execution of an SQL-statement has completed to the extent that all diagnostics information pertaining to that execution is recorded in the first diagnostics area. Condition handling mode ceases to be in effect when execution of the next SQL-statement begins.

This page intentionally left blank.

5 Lexical elements

This Clause modifies *Clause 5, “Lexical elements”*, in ISO/IEC 9075-2.

5.1 <token> and <separator>

This Subclause modifies *Subclause 5.2, “<token> and <separator>”*, in ISO/IEC 9075-2.

Function

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

Format

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

    | EXIT

    | STACKED

    | UNDO

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

    | ELSEIF

    | HANDLER

    | IF | ITERATE

    | LEAVE | LOOP

    | REPEAT | RESIGNAL

    | SIGNAL

    | UNTIL

    | WHILE
```

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

5.2 Names and identifiers

This Subclause modifies *Subclause 5.4, “Names and identifiers”*, in ISO/IEC 9075-2.

Function

Specify names.

Format

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

<SQL variable name> ::= <identifier>

<condition name> ::= <identifier>

Syntax Rules

- 1) **Replace SR 4)a)** If the <local or schema qualified name> is contained, without an intervening <schema definition> or <SQL-server module definition>, in a <preparable statement> that is prepared in the current SQL-session by an <execute immediate statement> or a <prepare statement> or in a <direct SQL statement> that is invoked directly, then the default <unqualified schema name> for the SQL-session is implicit.
- 2) **Insert before SR 4)b)** If the <local or schema qualified name> is contained in an <SQL-server module definition> without an intervening <schema definition>, then the <default schema name> that is specified or implicit in the <SQL-server module definition> is implicit.
- 3) **Replace SR 13)b)i)** If the <schema qualified name> is contained, without an intervening <schema definition> or <SQL-server module definition>, in a <preparable statement> that is prepared in the current SQL-session by an <execute immediate statement> or a <prepare statement> or in a <direct SQL statement> that is invoked directly, then the default <unqualified schema name> for the SQL-session is implicit.
- 4) **Insert before SR 13)b)ii)** If the <schema qualified name> is contained in an <SQL-server module definition> without an intervening <schema definition>, then the <default schema name> that is specified or implicit in the <SQL-server module definition> is implicit.
- 5) **Replace SR 8)** If <user-defined type name> *UDTN* with a <qualified identifier> *QI* is specified, then

Case:

- a) If *UDTN* is simply contained in <path-resolved user-defined type name>, then

Case:

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

- A) If *UDTN* is contained, without an intervening <schema definition> or <SQL-server module definition>, in a <preparable statement> that is prepared in the current SQL-session by an <execute immediate statement> or by a <prepare statement> or in a <direct SQL statement> that is invoked directly, then let *DP* be the SQL-path of the current SQL-session.
 - B) If *UDTN* is contained in an <SQL-server module definition> without in intervening <schema definition>, then let *DP* be the SQL-path of that <SQL-server module definition>.
 - C) If *UDTN* is contained in a <schema definition> that is not contained in an <SQL-client module definition>, then let *DP* be the SQL-path of that <schema definition>.
 - D) Otherwise, *UDTN* is contained in an <SQL-client module definition>; let *DP* be the SQL-path of that <SQL-client module definition>.
- 2) Let *N* be the number of <schema name>s in *DP*. Let *S_i*, 1 (one) ≤ *i* ≤ *N*, be the *i*-th <schema name> in *DP*.
 - 3) Let the *set of subject types* be the set containing every user-defined type *T* in the schema identified by some *S_i*, 1 (one) ≤ *i* ≤ *N*, such that the <qualified identifier> of *T* is equivalent to *QI*. There shall be at least one type in the set of subject types.
 - 4) Let *UDT* be the user-defined type contained in the set of subject types such that there is no other type *UDT2* for which the <schema name> of the schema that includes the user-defined type descriptor of *UDT2* precedes in *DP* the <schema name> identifying the schema that includes the user-defined type descriptor of *UDT*. *UDTN* identifies *UDT*.
 - 5) The implicit <schema name> of *UDTN* is the <schema name> of the schema that includes the user-defined type descriptor of *UDT*.
- b) If *UDTN* is simply contained in <schema-resolved user-defined type name>, then

Case:

 - i) If *UDTN* is contained, without an intervening <schema definition> or <SQL-server module definition>, in a <preparable statement> that is prepared in the current SQL-session by an <execute immediate statement> or by a <prepare statement> or in a <direct SQL statement> that is invoked directly, then the implicit <schema name> of *UDTN* is the default <unqualified schema name> of the current SQL-session.
 - ii) If *UDTN* is contained in an <SQL-server module definition> without in intervening <schema definition>, then the implicit <schema name> of *UDTN* is the <schema name> that is specified or implicit in <SQL-server module definition>.
 - iii) If *UDTN* is contained in a <schema definition> that is not contained in an <SQL-client module definition>, then the implicit <schema name> of *UDTN* is the <schema name> that is specified or implicit in <schema definition>.
 - iv) Otherwise, *UDTN* is contained in an <SQL-client module definition>; the implicit <schema name> of *UDTN* is the <schema name> that is specified or implicit in <SQL-client module definition>.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert this GR An <SQL-server module name> identifies an SQL-server module.
- 2) Insert this GR An <SQL variable name> identifies an SQL variable.
- 3) Insert this GR A <condition name> identifies an exception condition or a completion condition and optionally a corresponding SQLSTATE value.

Conformance Rules

No additional Conformance Rules.

This page intentionally left blank.

6 Scalar expressions

This Clause modifies Clause 6, “Scalar expressions”, in ISO/IEC 9075-2.

6.1 <value specification> and <target specification>

This Subclause modifies Subclause 6.4, “<value specification> and <target specification>”, in ISO/IEC 9075-2.

Function

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

Format

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

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

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

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

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

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

6.2 <identifier chain>

This Subclause modifies Subclause 6.6, “<identifier chain>”, in ISO/IEC 9075-2.

Function

Disambiguate a <period>-separated chain of identifiers.

Format

No additional Format items.

Syntax Rules

- 1) Insert before SR 8) An SQL variable V is said to be *refinable* if the declared type of V is a row type or a structured type.
- 2) Replace introductory paragraph of SR 8) For at most one j between 1 (one) and M , PIC_j is called the *basis* of IC , and j is called the *basis length* of IC . The *referent* of the basis is a column C of a table, an SQL parameter SP , or an SQL variable SV . The basis, basis length, basis scope and basis referent of IC are determined as follows:
- 3) Replace SR 8)a)ii) Otherwise, IC shall be contained within the scope of one or more range variables whose associated tables include a column whose <column name> is equivalent to I_1 or within the scope of a <routine name> whose associated <SQL parameter declaration list> includes an SQL parameter whose <SQL parameter name> is equivalent to I_1 or within the scope of one or more <beginning label>s whose associated <local declaration list> includes an SQL variable whose <identifier> is equivalent to I_1 . Let the phrase *possible scope tags* denote those range variables, <routine name>s, and <beginning label>s.
- 4) Insert after SR 8)a)ii)1)B) If $IPST$ is a <beginning label>, then let SV be the SQL variable whose <SQL variable name> is equivalent to I_1 . PIC_1 is the basis of IC , the basis length is 1 (one), the basis scope is the scope of SP , and the basis referent is SV .
- 5) Insert before SR 8)b)iv) If IC is contained in the scope of a <beginning label> whose associated <local declaration list> includes an SQL variable SV whose <SQL variable name> is equivalent to I_1 , then PIC_1 is a candidate basis of IC , the scope of PIC_1 is the scope of SV , and the referent of PIC_1 is SV .
- 6) Insert before SR 8)b)iv) If $N = 2$ and PIC_1 is equivalent to a <beginning label> BL whose scope contains IC and whose associated <local declaration list> includes an SQL variable SV whose <SQL variable name> is equivalent to I_2 , then PIC_2 is a candidate basis of IC , the scope of PIC_2 is the scope of SV , and the referent of PIC_2 is SV .
- 7) Insert before SR 8)b)iv) If $N > 2$ and PIC_1 is equivalent to a <beginning label> BL whose scope contains IC and whose associated <local declaration list> includes a refinable SQL variable SV whose <SQL variable name> is equivalent to I_2 , then PIC_2 is a candidate basis of IC , the scope of PIC_2 is the scope of SV , and the referent of PIC_2 is SV .

- 8) **Replace SR 10)** If $BL < N$, then let TIC be the <value expression primary>:

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

The Syntax Rules of Subclause 6.25, “<value expression>”, are applied to TIC , yielding a column reference, an SQL parameter reference, or an SQL variable reference, and $(N-BL)$ <field reference>s, <method invocation>s, <modified field reference>s, and/or <mutator reference>s.

NOTE 3 — 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>.

- 9) **Insert after SR 13)** A <basic identifier chain> whose basis referent is an SQL variable is an *SQL variable reference*.

Access Rules

None.

General Rules

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

Conformance Rules

- 1) Without Feature P005, “Qualified SQL variable references”, conforming SQL language shall not contain an SQL variable reference whose first <identifier> is the <beginning label> of a <compound statement>.

6.3 <next value expression>

This Subclause modifies Subclause 6.13, “<next value expression>”, in ISO/IEC 9075-2.

Function

Return the next value of a sequence generator.

Format

No additional Format items.

Syntax Rules

- 1) Insert after SR 1)e An <assignment statement>.
- 2) Insert this SR <next value expression> shall not be contained without an intervening <SQL statement list> in a <case statement>, an <if statement>, a <loop statement>, a <while statement>, or a <repeat statement>.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

6.4 <SQL variable reference>

Function

Reference an SQL variable.

Format

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

Syntax Rules

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

Access Rules

None.

General Rules

None.

Conformance Rules

No additional Conformance Rules.

7 Query expressions

This Clause modifies Clause 7, “Query expressions”, in ISO/IEC 9075-2.

7.1 <query specification>

This Subclause modifies Subclause 7.12, “<query specification>”, in ISO/IEC 9075-2.

Function

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

Format

No additional Format items.

Syntax Rules

- 1) Insert after SR 7)f)i If IC is contained in the scope of a <beginning label> whose associated <local declaration list> includes an SQL variable SV whose <SQL variable name> is equivalent to I_1 , then PIC_1 is a candidate basis of IC and the scope of PIC_1 is the scope of SV .
- 2) Insert after SR 7)f)i If $N = 2$ and PIC_1 is equivalent to a <beginning label> BL whose scope contains IC and whose associated <local declaration list> includes an SQL variable SV of row type whose <SQL variable name> is equivalent to I_2 , then PIC_2 is a candidate basis of IC , the scope of PIC_2 is the scope of SV , and the referent of PIC_2 is SV .
- 3) Insert after SR 7)f)i If $N > 2$ and PIC_1 is equivalent to a <beginning label> BL whose scope contains IC and whose associated <local declaration list> includes a refinable SQL variable SV whose <SQL variable name> is equivalent to I_2 , then PIC_2 is a candidate basis of IC , the scope of PIC_2 is the scope of SV , and the referent of PIC_2 is SV .
- 4) Insert after SR 18)a)iv An SQL variable.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

- 1) Without Feature P005, “Qualified SQL variable references”, conforming SQL language shall not contain an <asterisked identifier chain> whose first <identifier> is the <beginning label> of a <compound statement>.

8 Additional common elements

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

8.1 <routine invocation>

This Subclause modifies *Subclause 10.4, “<routine invocation>”*, in ISO/IEC 9075-2.

Function

Invoke an SQL-invoked routine.

Format

No additional Format items.

Syntax Rules

- 1) Replace SR 5) An SQL-invoked routine *R* is an *executable routine* if and only if *R* is a possibly candidate routine and

Case:

 - a) If *RI* is contained in an <SQL schema statement>, then

Case:

 - i) If *RI* is contained in an <SQL-server module definition> *M*, then the applicable privileges for the <authorization identifier> that owns the containing schema include EXECUTE on *M*.
 - ii) Otherwise, the applicable privileges for the <authorization identifier> that owns the containing schema include EXECUTE on *R*.
 - b) Otherwise,

Case:

 - i) If *RI* is contained in an <SQL-server module definition> *M*, then the current privileges include EXECUTE on *M*.
 - ii) Otherwise, the current privileges include EXECUTE on *R*.
- 2) Insert before SR 7)b)i)1)C)I) If *RI* is contained in an <SQL-server module definition>, then let *DP* be the SQL-path of that <SQL-server module definition>.

- 3) Replace SR 7)b)i)1)C)I) If RI is contained in a <schema definition> without an intervening <SQL-server module definition>, then let DP be the SQL-path of that <schema definition>.
- 4) Insert before SR 8)b)i)1)C)I) If RI is contained in an <SQL-server module definition>, then let DP be the SQL-path of that <SQL-server module definition>.
- 5) Replace SR 8)b)i)1)C)I) If RI is contained in a <schema definition> without an intervening <SQL-server module definition>, then let DP be the SQL-path of that <schema definition>.
- 6) Replace SR 8)c)i)4)C) If A_i is either an <SQL variable reference>, an <SQL parameter reference>, a <column reference>, or a <target array element specification>, 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.

NOTE 4 — The <column reference> can only be a new transition variable column reference.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert after GR 5)d)i)

NOTE 5 — The identities of declared local temporary tables that are defined in <SQL-server module>s are not removed.

- 2) Replace the introductory paragraph of GR 9)b)ii) If TS_i is either an <SQL variable reference>, an <SQL parameter reference>, a <column reference>, or a <target array element specification>, then

NOTE 6 — The <column reference> can only be a new transition variable column reference.

Conformance Rules

No additional Conformance Rules.

8.2 <sqlstate value>

Function

Specify an SQLSTATE value.

Format

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

Syntax Rules

- 1) Let L be the <character string literal> contained in <sqlstate value>.
- 2) The implicit or explicit character set of L shall be the implementation-defined character set in which SQLSTATE parameter values are returned.
- 3) Let V be the character string that is the value of
`TRIM ( BOTH ' ' FROM L )`
- 4) V shall comprise either:
 - a) Five characters of which the first two have the form of a standard-defined class 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.

Conformance Rules

None.

This page intentionally left blank.

9 Schema definition and manipulation

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

9.1 <schema definition>

This Subclause modifies Subclause 11.1, “<schema definition>”, in ISO/IEC 9075-2.

Function

Define a schema.

Format

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

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

9.2 <drop schema statement>

This Subclause modifies *Subclause 11.2*, “<drop schema statement>”, in ISO/IEC 9075-2.

Function

Destroy a schema.

Format

No additional Format items.

Syntax Rules

- 1) Insert this SR If RESTRICT is specified, then S shall not include any SQL-server modules.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert before GR 8 For every SQL-server module *M* contained in *S*, let *MN* be the <SQL-server module name> of *M*. For every *M*, the following <drop module statement> is effectively executed:

DROP MODULE *MN* CASCADE

- 2) Replace GR 11 Let *R* be any SQL-invoked routine whose routine descriptor contains the <schema name> of *S* in the <SQL routine body>.

Case:

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

DROP MODULE *MN* CASCADE

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

DROP SPECIFIC ROUTINE *SN* CASCADE

- 3) Insert after GR 11 Let *SSM* be any SQL-server module whose module descriptor includes the <schema name> of *S* and let *MN* be the <SQL-server module name> of *SSM*. The following <drop module statement> is effectively executed without further Access Rule checking:

DROP MODULE *MN* CASCADE