
**Information technology — Database
languages — SQL —**

**Part 9:
Management of External Data (SQL/MED)**

*Technologies de l'information — Langages de base de données — SQL —
Partie 9: Gestion des données externes (SQL/MED)*

PDF disclaimer

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

Adobe is a trademark of Adobe Systems Incorporated.

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

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

© ISO/IEC 2001

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 either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.ch
Web www.iso.ch

Printed in Switzerland

Contents

Page

Foreword	xiv
Introduction	xv
1 Scope	1
2 Normative references	3
2.1 ISO/IEC JTC 1 standards	3
2.2 Publicly-available specifications	4
3 Definitions, notations, and conventions	5
3.1 Definitions	5
3.1.1 Definitions provided in Part 9	5
3.1.2 Definitions taken from XML	6
3.2 Notations	6
3.3 Conventions	6
3.3.1 Relationships to other parts of ISO/IEC 9075	6
3.3.1.1 Clause, Subclause, and Table relationships	6
3.4 Object identifier for Database Language SQL	22
4 Concepts	27
4.1 SQL-environments and their components	27
4.1.1 SQL-environments	27
4.1.2 User mapping concepts	27
4.1.3 Catalogs and schemas	27
4.1.3.1 Catalogs	27
4.1.4 Foreign servers and descriptors	27
4.1.5 Foreign-data wrappers and descriptors	28
4.2 Data types	28
4.3 Foreign servers	28
4.4 Foreign-data wrappers	29
4.5 User mappings	30
4.6 Generic options	30
4.7 Capabilities and options information	31
4.8 Datalinks	31
4.8.1 Operations involving datalinks	34
4.8.1.1 Operators that operate on datalinks	34

ISO/IEC 9075-9:2001 (E)

4.8.1.2	Other operators involving datalinks	35
4.9	Type conversion and mixing of data types	35
4.10	Columns, fields, and attributes	35
4.11	Tables	35
4.11.1	Types of tables	36
4.12	Functional dependencies	36
4.12.1	Known functional dependencies in a foreign table	36
4.13	SQL-schemas	36
4.14	SQL-statements	36
4.14.1	SQL-statements classified by function	36
4.15	SQL-sessions	37
4.16	Privileges	38
4.17	SQL-transactions	38
4.18	Foreign-data wrapper interface	38
4.18.1	Handles	39
4.18.2	Foreign server sessions	40
4.18.3	Foreign-data wrapper interface routines	40
4.18.3.1	Handle routines	41
4.18.3.2	Initialization routines	43
4.18.3.3	Access routines	44
4.18.3.4	Termination routines	44
4.18.3.5	Decomposition and pass-through modes	45
4.18.3.6	Sequence of actions during the execution of requests involving foreign tables and foreign servers	45
4.18.4	Return codes	57
4.18.5	Foreign-data wrapper diagnostics areas	58
4.18.6	Null pointers	59
4.18.7	Foreign-data wrapper descriptor areas	60
4.19	Introduction to SQL/CLI	63
5	Lexical elements	65
5.1	<token> and <separator>	65
5.2	Names and identifiers	67
6	Scalar expressions	69
6.1	<data type>	69
6.2	<column reference>	72
6.3	<set function specification>	73
6.4	<string value function>	74
6.5	<datalink value function>	77
6.6	<cast specification>	79
6.7	<value expression>	81
6.8	<datalink value expression>	82

7	Query expressions	83
7.1	<table reference>	83
7.2	<joined table>	89
7.3	<group by clause>	90
7.4	<query specification>	91
7.5	<query expression>	92
8	Predicates	93
8.1	<unique predicate>	93
9	URLs	95
9.1	URL format	95
10	Data assignment rules and routine determination	99
10.1	Retrieval assignment	99
10.2	Store assignment	100
10.3	Data types of results of aggregations	101
10.4	Type precedence list determination	102
10.5	Determination of identical values	103
11	Additional common elements	105
11.1	<privileges>	105
11.2	<generic options>	106
11.3	<alter generic options>	107
12	Schema definition and manipulation	109
12.1	<schema definition>	109
12.2	<drop schema statement>	110
12.3	<table definition>	111
12.4	<column definition>	112
12.5	<unique constraint definition>	113
12.6	<drop column definition>	114
12.7	<domain definition>	115
12.8	<SQL-invoked routine>	116
12.9	<user-defined type definition>	117
12.10	<user-defined cast definition>	118
12.11	<user-defined ordering definition>	119
12.12	<foreign table definition>	120
12.13	<alter foreign table statement>	123
12.14	<add basic column definition>	125
12.15	<alter basic column definition>	127
12.16	<drop basic column definition>	128
12.17	<drop foreign table statement>	130

13	Catalog manipulation	133
13.1	<foreign server definition>	133
13.2	<alter foreign server statement>	135
13.3	<drop foreign server statement>	136
13.4	<foreign-data wrapper definition>	137
13.5	<alter foreign-data wrapper statement>	139
13.6	<drop foreign-data wrapper statement>	140
13.7	<import foreign schema statement>	141
14	Access control	143
14.1	<revoke statement>	143
14.2	<user mapping definition>	144
14.3	<alter user mapping statement>	146
14.4	<drop user mapping statement>	147
15	SQL-client modules	149
15.1	<SQL-client module definition>	149
15.2	<externally-invoked procedure>	150
15.3	Calls to an <externally-invoked procedure>	151
15.4	<SQL procedure statement>	153
15.5	Data type correspondences	154
16	Data manipulation	157
16.1	<declare cursor>	157
16.2	Effect of deleting rows from base tables	158
16.3	Effect of inserting tables into base tables	159
16.4	Effect of replacing rows in base tables	160
17	Session management	163
17.1	<set passthrough statement>	163
18	Dynamic SQL	165
18.1	Description of SQL descriptor areas	165
18.2	<prepare statement>	166
18.3	<deallocate prepared statement>	168
18.4	<describe statement>	169
18.5	<input using clause>	171
18.6	<output using clause>	175
18.7	<execute statement>	178
18.8	<dynamic declare cursor>	179
18.9	<allocate cursor statement>	180
18.10	<dynamic open statement>	181
18.11	<dynamic fetch statement>	182
18.12	<dynamic close statement>	183

19 Embedded SQL	185
19.1 <embedded SQL Ada program>	185
19.2 <embedded SQL C program>	186
19.3 <embedded SQL COBOL program>	187
19.4 <embedded SQL Fortran program>	188
19.5 <embedded SQL MUMPS program>	189
19.6 <embedded SQL Pascal program>	190
19.7 <embedded SQL PL/I program>	191
20 Call-Level Interface specifications	193
20.1 <CLI routine>	193
20.2 Implicit DESCRIBE USING clause	193
20.2.1 CLI-specific status codes	194
20.2.2 Description of CLI item descriptor areas	194
20.2.3 Other tables associated with CLI	195
20.3 SQL/CLI data type correspondences	197
21 SQL/CLI routines	199
21.1 BuildDataLink	199
21.2 GetDataLinkAttr	200
21.3 GetInfo	202
22 SQL/MED common specifications	203
22.1 Description of foreign-data wrapper item descriptor areas	203
22.2 Implicit cursor	207
22.3 Implicit DESCRIBE INPUT USING clause	209
22.4 Implicit DESCRIBE OUTPUT USING clause	212
22.5 Implicit EXECUTE USING and OPEN USING clauses	216
22.6 Implicit FETCH USING clause	219
22.7 Character string retrieval	223
22.8 Binary large object string retrieval	224
22.9 Tables used with SQL/MED	225
23 Foreign-data wrapper interface routines	235
23.1 <foreign-data wrapper interface routine>	235
23.2 <foreign-data wrapper interface routine> invocation	239
23.3 Foreign-data wrapper interface wrapper routines	241
23.3.1 AllocWrapperEnv	241
23.3.2 Close	243
23.3.3 ConnectServer	244
23.3.4 FreeExecutionHandle	246
23.3.5 FreeFSConnection	248
23.3.6 FreeReplyHandle	249
23.3.7 FreeWrapperEnv	250
23.3.8 GetNumReplySelectElems	251
23.3.9 GetNumReplyTableRefs	252
23.3.10 GetOpts	253

ISO/IEC 9075-9:2001 (E)

23.3.11 GetReplySelectElem	255
23.3.12 GetReplyTableRef	256
23.3.13 GetSPDHandle	257
23.3.14 GetSRDHandle	258
23.3.15 GetStatistics	259
23.3.16 GetWPDHandle	261
23.3.17 GetWRDHandle	262
23.3.18 InitRequest	263
23.3.19 Iterate	267
23.3.20 Open	269
23.3.21 ReOpen	273
23.3.22 TransmitRequest	274
23.4 Foreign-data wrapper interface SQL-server routines	277
23.4.1 AllocDescriptor	277
23.4.2 FreeDescriptor	278
23.4.3 GetAuthorizationId	279
23.4.4 GetDescriptor	280
23.4.5 GetNumSelectElems	282
23.4.6 GetNumServerOpts	283
23.4.7 GetNumTableColOpts	284
23.4.8 GetNumTableOpts	286
23.4.9 GetNumTableRefElems	287
23.4.10 GetNumUserOpts	288
23.4.11 GetNumWrapperOpts	289
23.4.12 GetSelectElem	290
23.4.13 GetSelectElemType	291
23.4.14 GetServerName	292
23.4.15 GetServerOpt	293
23.4.16 GetServerOptByName	295
23.4.17 GetServerType	297
23.4.18 GetServerVersion	298
23.4.19 GetSQLString	299
23.4.20 GetTableColOpt	300
23.4.21 GetTableColOptByName	302
23.4.22 GetTableOpt	304
23.4.23 GetTableOptByName	306
23.4.24 GetTableRefElem	308
23.4.25 GetTableRefElemType	309
23.4.26 GetTableRefTableName	310
23.4.27 GetTableServerName	310
23.4.28 GetTRDHandle	312
23.4.29 GetUserOpt	313
23.4.30 GetUserOptByName	315
23.4.31 GetValExprColName	317
23.4.32 GetWrapperLibraryName	318

23.4.33	GetWrapperName	319
23.4.34	GetWrapperOpt	320
23.4.35	GetWrapperOptByName	322
23.4.36	SetDescriptor	324
23.5	Foreign-data wrapper interface general routines	329
23.5.1	GetDiagnostics	329
24	Diagnostics management	333
24.1	<get diagnostics statement>	333
25	Information Schema	335
25.1	ATTRIBUTES view	335
25.2	COLUMN_OPTIONS view	337
25.3	COLUMNS view	338
25.4	FOREIGN_DATA_WRAPPER_OPTIONS view	341
25.5	FOREIGN_DATA_WRAPPERS view	342
25.6	FOREIGN_SERVER_OPTIONS view	343
25.7	FOREIGN_SERVERS view	344
25.8	FOREIGN_TABLE_OPTIONS view	345
25.9	FOREIGN_TABLES view	346
25.10	USER_MAP_OPTIONS view	347
25.11	USER_MAPPINGS view	348
25.12	Short name views	349
26	Definition Schema	353
26.1	COLUMN_OPTIONS base table	353
26.2	DATA_TYPE_DESCRIPTOR base table	354
26.3	FOREIGN_DATA_WRAPPER_OPTIONS base table	361
26.4	FOREIGN_DATA_WRAPPERS base table	362
26.5	FOREIGN_SERVER_OPTIONS base table	363
26.6	FOREIGN_SERVERS base table	364
26.7	FOREIGN_TABLE_OPTIONS base table	365
26.8	FOREIGN_TABLES base table	366
26.9	SQL_SIZING base table	367
26.10	TABLES base table	368
26.11	USAGE_PRIVILEGES base table	369
26.12	USER_MAPPING_OPTIONS base table	370
26.13	USER_MAPPINGS base table	371
27	Status codes	373
27.1	SQLSTATE	373
28	Conformance	375
28.1	SQL-server conformance to SQL/MED	375
28.2	Foreign-data-wrapper conformance to SQL/MED	376
28.3	Claims of conformance by SQL-servers	376
28.4	Claims of conformance by foreign-data wrappers	377

ISO/IEC 9075-9:2001 (E)

28.5	Extensions and options	377
Annex A	SQL Conformance Summary	379
Annex B	Implementation-defined elements	385
Annex C	Implementation-dependent elements	393
Annex D	Deprecated features	397
Annex E	Incompatibilities with ISO/IEC 9075:1992	399
Annex F	Typical header files	401
F.1	C Header File SQLCLI.H	401
F.2	COBOL Library Item SQLCLI	402
Annex G	SQL feature and package taxonomy	403
Annex H	SQL/MED model	405
Index		411

FIGURES

Figure		Page
1	SQL/MED interfaces	405
2	SQL/MED information flow	407

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

TABLES

Tables	Page
1 Clause, Subclause, and Table relationships	6
2 Valid datalink file control options	34
3 Sequence of actions during foreign server request executions	46
4 Fields used in foreign-data wrapper diagnostics areas	59
5 Fields in foreign-data wrapper descriptor areas	61
6 Data type correspondences for Ada	154
7 Data type correspondences for C	154
8 Data type correspondences for COBOL	154
9 Data type correspondences for Fortran	155
10 Data type correspondences for MUMPS	155
11 Data type correspondences for Pascal	155
12 Data type correspondences for PL/I	155
13 Codes used for SQL data types in Dynamic SQL	165
14 Abbreviated SQL/CLI generic names	193
15 SQLSTATE class and subclass values for SQL/CLI-specific conditions	194
16 Codes used for implementation data types in SQL/CLI	194
17 Codes used for application data types in SQL/CLI	195
18 Codes used to identify SQL/CLI routines	195
19 Codes and data types for implementation information	195
20 Codes used for datalink attributes	195
21 Data types of attributes	196
22 SQL/CLI data type correspondences for Ada	197
23 SQL/CLI data type correspondences for C	197
24 SQL/CLI data type correspondences for COBOL	197
25 SQL/CLI data type correspondences for Fortran	197
26 SQL/CLI data type correspondences for MUMPS	197
27 SQL/CLI data type correspondences for Pascal	198
28 SQL/CLI data type correspondences for PL/I	198
29 Codes used for <table reference> types	225
30 Codes used for <value expression> types	225
31 Codes used for foreign-data wrapper diagnostic fields	225
32 Codes used for foreign-data wrapper descriptor fields	225
33 Codes used for foreign-data wrapper handle types	227
34 Ability to retrieve foreign-data wrapper descriptor fields	228
35 Ability to set foreign-data wrapper descriptor fields	230
36 Foreign-data wrapper descriptor field default values	231
37 Codes used for the format of the character string transmitted by GetSQLString()	233
38 SQL-statement codes	333

39	SQLSTATE class and subclass values	373
40	Implied feature relationships	375
41	SQL/MED feature taxonomy for features outside Core SQL	403
42	Legend for SQL/MED interfaces	406
43	Legend for SQL/MED information flow	408

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

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.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

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.

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 9075 may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 9075-9 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange*.

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

- *Part 1: Framework (SQL/Framework)*
- *Part 2: Foundation (SQL/Foundation)*
- *Part 3: Call-Level Interface (SQL/CLI)*
- *Part 4: Persistent Stored Modules (SQL/PSM)*
- *Part 5: Host Language Bindings (SQL/Bindings)*
- *Part 9: Management of External Data (SQL/MED)*
- *Part 10: Object Language Bindings (SQL/OLB)*

Annexes A, B, C, D, E, F, G and H 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 related to this part of ISO/IEC 9075.
- 5) Clause 5, "Lexical elements", defines the lexical elements of the language specified in this part of ISO/IEC 9075.
- 6) Clause 6, "Scalar expressions", defines the elements of the language that produce scalar values.
- 7) Clause 7, "Query expressions", defines the elements of the language that produce rows and tables of data.
- 8) Clause 8, "Predicates", defines the predicates of the language.
- 9) Clause 9, "URLs", specifies the format of URLs used in this part of ISO/IEC 9075.
- 10) Clause 10, "Data assignment rules and routine determination", specifies the rules for assignments that retrieves data from or store data into SQL-data, and formation rules for set operations.
- 11) Clause 11, "Additional common elements", defines additional common elements used in the definition of foreign tables, foreign servers, and foreign-data wrappers.
- 12) Clause 12, "Schema definition and manipulation", defines facilities related to foreign tables and datalink type support for creating and managing a schema.
- 13) Clause 13, "Catalog manipulation", defines facilities for creating, altering, and dropping foreign servers and foreign-data wrappers.
- 14) Clause 14, "Access control", defines facilities for controlling access to SQL-data.
- 15) Clause 15, "SQL-client modules", defines SQL-client modules and externally-invoked procedures.
- 16) Clause 16, "Data manipulation", defines the data manipulation statements.
- 17) Clause 17, "Session management", defines the SQL-session management statements.
- 18) Clause 18, "Dynamic SQL", defines the dynamic SQL statements.
- 19) Clause 19, "Embedded SQL", defines the embedded SQL statements.

ISO/IEC 9075-9:2001 (E)

- 20) Clause 20, "Call-Level Interface specifications", defines facilities for using SQL through a Call-Level Interface.
- 21) Clause 21, "SQL/CLI routines", defines each of the routines that comprise the Call-Level Interface.
- 22) Clause 22, "SQL/MED common specifications", specifies common facilities used by SQL/MED.
- 23) Clause 23, "Foreign-data wrapper interface routines", specifies the interaction between an SQL-server and a foreign-data wrapper.
- 24) Clause 24, "Diagnostics management", defines the diagnostics management facilities.
- 25) Clause 25, "Information Schema", defines viewed tables that contain schema information.
- 26) Clause 26, "Definition Schema", defines base tables on which the viewed tables containing schema information depend.
- 27) Clause 27, "Status codes", defines values that identify the status of the execution of SQL-statements and the mechanisms by which those values are returned.
- 28) Clause 28, "Conformance", specifies the way in which conformance to this part of ISO/IEC 9075 may be claimed.
- 29) Annex A, "SQL Conformance Summary", is an informative Annex. It summarizes the conformance requirements of the SQL language.
- 30) Annex B, "Implementation-defined elements", is an informative Annex. It lists those features for which 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.
- 31) Annex C, "Implementation-dependent elements", is an informative Annex. It lists those features for which 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.
- 32) Annex D, "Deprecated features", is an informative Annex. It lists features that the responsible Technical Committee intend will not appear in a future revised version of this part of ISO/IEC 9075.
- 33) Annex E, "Incompatibilities with ISO/IEC 9075:1992", is an informative Annex. It lists incompatibilities with the previous version of ISO/IEC 9075.
- 34) Annex F, "Typical header files", is an informative Annex. It provides examples of typical definition files for application programs using the SQL Call-Level Interface.
- 35) Annex G, "SQL feature and package taxonomy", is an informative Annex. It identifies features and packages of the SQL language specified in this part of ISO/IEC 9075 by an identifier and a short descriptive name. This taxonomy is used to specify conformance to the packages specified in this part of ISO/IEC 9075. The feature taxonomy may be used to develop other profiles involving the SQL language.

- 36) Annex H, “SQL/MED model”, is an informative Annex. It uses annotated diagrams to illustrate the more important concepts of the model of SQL/MED, including the relationships between the SQL-server, foreign-data wrappers, and foreign servers.

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

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2007

Information technology — Database languages — SQL —

Part 9: Management of External Data (SQL/MED)

1 Scope

This part of ISO/IEC 9075 defines extensions to Database Language SQL to support management of external data through the use of foreign-data wrappers and datalink types.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

(Blank page)

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

2 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 9075. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, 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 normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

2.1 ISO/IEC JTC 1 standards

- ISO/IEC 1539-1:1997, *Information technology — Programming languages — Fortran — Part 1: Base language*.
- ISO 1989:1985, *Programming languages — COBOL*.
(Endorsement of ANSI X3.23-1985).
- ISO 6160:1979, *Programming languages — PL/I*.
(Endorsement of ANSI X3.53-1976).
- ISO/IEC 7185:1990, *Information technology — Programming languages — Pascal*.
- ISO/IEC 8652:1995, *Information technology — Programming languages — Ada*.
- ISO/IEC 9075-1:1999, *Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework)*.
- ISO/IEC 9075-2:1999, *Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation)*.
- ISO/IEC 9075-3:1999, *Information technology — Database languages — SQL — Part 3: Call-Level Interface (SQL/CLI)*.
- ISO/IEC 9075-4:1999, *Information technology — Database languages — SQL — Part 4: Persistent Stored Modules (SQL/PSM)*.
- ISO/IEC 9075-5:1999, *Information technology — Database languages — SQL — Part 5: Host Language Bindings (SQL/Bindings)*.

ISO/IEC 9075-9:2001 (E)
2.1 ISO/IEC JTC 1 standards

- 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 — M*.

2.2 Publicly-available specifications

- RFC 1738: *Uniform Resource Locators (URL)*, T. Berners-Lee, L. Masinter, M. McCahill, December, 1994.
- RFC 1808, *Relative Uniform Resource Locators*, R. Fielding, June, 1995.
- RFC 2368, *The mailto URL scheme*, R. Hoffman, L. Masinter, J. Zawinski, July, 1998.
- <http://www.w2.org/TR/1998/REC-xml-19980210>, *Extensible Markup Language (XML) 1.0*, Tim Bray, Jean Paoli, and C. M. Sperberg-McQueen (editors), corrected by <http://www.w3.org/XML/xml-19980210-errata>

3 Definitions, notations, and conventions

3.1 Definitions

3.1.1 Definitions provided in Part 9

Insert this paragraph For the purposes of this part of ISO/IEC 9075, the definitions given in ISO/IEC 9075-1, ISO/IEC 9075-2, ISO/IEC 9075-3, ISO/IEC 9075-4, and ISO/IEC 9075-5 and the following definitions apply.

- a) **access token:** An encrypted value returned under certain conditions by an SQL-server in combination with the File Reference of a datalink value.
- b) **datalink:** A value, of data type DATALINK, referencing some file that is not part of the SQL-environment. The file is assumed to be managed by some external file manager.
- c) **datalinker:** An implementation-dependent component for controlling access and referential integrity to external files.
- d) **external data:** Data that is not managed by an SQL-server involved in an SQL-session, but that is nevertheless accessible to that SQL-session.
- e) **foreign-data wrapper:** A named collection of routines, invocable by the SQL-server, supporting the programming interface specified for such routines in this part of ISO/IEC 9075.
- f) **foreign server:** A named server, external to the SQL-environment, but known to the SQL-server, that manages external data.
- g) **foreign table:** A named table whose rows are supplied when needed by some foreign server. The mechanism by which these rows are supplied is provided by a foreign-data wrapper. The data constituting a foreign table is not part of the SQL-environment.
- h) **integrity option:** Specifies the level of integrity of the link between a datalink and the file that it references.
- i) **link control:** A property of a column of data type DATALINK, specifying the extent to which the links between datalinks in that column and the files they reference are to be monitored (in various specific manners).
- j) **read permission option:** A link control option specifying how permission to read external files referenced by certain datalinks is determined.
- k) **recovery option:** A link control option specifying whether or not point in time recovery is required for the files referenced by certain datalinks.
- l) **SQL/MED-implementation:** An SQL-implementation that processes SQL-statements that are possibly extended by the language defined in this part of ISO/IEC 9075. A *conforming SQL/MED-implementation* is an SQL/MED-implementation that satisfied the requirements for SQL/MED-implementations as defined in Clause 28, "Conformance".

ISO/IEC 9075-9:2001 (E)

3.1 Definitions

- m) **unlink option**: A link control option specifying the action to be taken when certain sites occupied by datalinks are updated or deleted.
- n) **user mapping**: An implementation-defined mapping of an authorization identifier to an equivalent concept maintained by a foreign server.
- o) **write permission option**: A link control option specifying how permission to write files referenced by certain datalinks is determined.

3.1.2 Definitions taken from XML

This part of ISO/IEC 9075 makes use of the following terms defined in the W3C Recommendation for XML:

- a) **Valid XML document**
- b) **XML document**
- c) **XML document type declaration** (also known as "DTD")

3.2 Notations

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

3.3 Conventions

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

3.3.1 Relationships to other parts of ISO/IEC 9075

3.3.1.1 Clause, Subclause, and Table relationships

Table 1—Clause, Subclause, and Table relationships

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Clause 1, "Scope"	Clause 1, "Scope"	ISO/IEC 9075-2
Clause 2, "Normative references"	Clause 2, "Normative references"	ISO/IEC 9075-2
Subclause 2.1, "ISO/IEC JTC 1 standards"	Clause 2, "Normative references"	ISO/IEC 9075-2

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

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Subclause 2.2, "Publicly-available specifications"	<i>none</i>	<i>none</i>
Clause 3, "Definitions, notations, and conventions"	Clause 3, "Definitions, notations, and conventions"	ISO/IEC 9075-2
Subclause 3.1, "Definitions"	Subclause 3.1, "Definitions"	ISO/IEC 9075-2
Subclause 3.1.1, "Definitions provided in Part 9"	<i>none</i>	<i>none</i>
Subclause 3.1.2, "Definitions taken from XML"	<i>none</i>	<i>none</i>
Subclause 3.2, "Notations"	Subclause 6.1, "Notation"	ISO/IEC 9075-1
Subclause 3.3, "Conventions"	Subclause 3.3, "Conventions"	ISO/IEC 9075-1
Subclause 3.3.1, "Relationships to other parts of ISO/IEC 9075"	<i>none</i>	<i>none</i>
Subclause 3.3.1.1, "Clause, Subclause, and Table relationships"	<i>none</i>	<i>none</i>
Subclause 3.4, "Object identifier for Database Language SQL"	Subclause 6.3, "Object identifier for Database Language SQL"	ISO/IEC 9075-1
Clause 4, "Concepts"	Clause 4, "Concepts"	ISO/IEC 9075-2
Subclause 4.1, "SQL-environments and their components"	Subclause 4.2, "SQL-environments and their components"	ISO/IEC 9075-1
Subclause 4.1.1, "SQL-environments"	Subclause 4.2.1, "SQL-environments"	ISO/IEC 9075-1
Subclause 4.1.2, "User mapping concepts"	<i>none</i>	<i>none</i>
Subclause 4.1.3, "Catalogs and schemas"	Subclause 4.2.6, "Catalogs and schemas"	ISO/IEC 9075-1
Subclause 4.1.3.1, "Catalogs"	Subclause 4.2.6.1, "Catalogs"	ISO/IEC 9075-1
Subclause 4.1.4, "Foreign servers and descriptors"	<i>none</i>	<i>none</i>
Subclause 4.1.5, "Foreign-data wrappers and descriptors"	<i>none</i>	<i>none</i>
Subclause 4.2, "Data types"	Subclause 4.1, "Data types"	ISO/IEC 9075-2
Subclause 4.3, "Foreign servers"	<i>none</i>	<i>none</i>
Subclause 4.4, "Foreign-data wrappers"	<i>none</i>	<i>none</i>
Subclause 4.5, "User mappings"	<i>none</i>	<i>none</i>
Subclause 4.6, "Generic options"	<i>none</i>	<i>none</i>

ISO/IEC 9075-9:2001 (E)
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 another part	Part containing correspondence
Subclause 4.7, "Capabilities and options information"	<i>none</i>	<i>none</i>
Subclause 4.8, "Datalinks"	<i>none</i>	<i>none</i>
Subclause 4.8.1, "Operations involving datalinks"	<i>none</i>	<i>none</i>
Subclause 4.8.1.1, "Operators that operate on datalinks"	<i>none</i>	<i>none</i>
Subclause 4.8.1.2, "Other operators involving datalinks"	<i>none</i>	<i>none</i>
Subclause 4.9, "Type conversion and mixing of data types"	Subclause 4.12, "Type conversions and mixing of data types"	ISO/IEC 9075-2
Subclause 4.10, "Columns, fields, and attributes"	Subclause 4.15, "Columns, fields, and attributes"	ISO/IEC 9075-2
Subclause 4.11, "Tables"	Subclause 4.16, "Tables"	ISO/IEC 9075-2
Subclause 4.11.1, "Types of tables"	Subclause 4.16.1, "Types of tables"	ISO/IEC 9075-2
Subclause 4.12, "Functional dependencies"	Subclause 4.18, "Functional dependencies"	ISO/IEC 9075-2
Subclause 4.12.1, "Known functional dependencies in a foreign table"	<i>none</i>	<i>none</i>
Subclause 4.13, "SQL-schemas"	Subclause 4.20, "SQL-schemas"	ISO/IEC 9075-2
Subclause 4.14, "SQL-statements"	4.30, "SQL-statements"	ISO/IEC 9075-2
Subclause 4.14.1, "SQL-statements classified by function"	Subclause 4.30.2, "SQL-statements classified by function"	ISO/IEC 9075-2
Subclause 4.15, "SQL-sessions"	Subclause 4.34, "SQL-sessions"	ISO/IEC 9075-3
Subclause 4.16, "Privileges"	Subclause 4.31.2, "Privileges"	ISO/IEC 9075-2
Subclause 4.17, "SQL-transactions"	Subclause 4.32, "SQL-transactions"	ISO/IEC 9075-2
Subclause 4.18, "Foreign-data wrapper interface"	<i>none</i>	<i>none</i>
Subclause 4.18.1, "Handles"	<i>none</i>	<i>none</i>
Subclause 4.18.2, "Foreign server sessions"	<i>none</i>	<i>none</i>
Subclause 4.18.3, "Foreign-data wrapper interface routines"	<i>none</i>	<i>none</i>
Subclause 4.18.3.1, "Handle routines"	<i>none</i>	<i>none</i>
Subclause 4.18.3.2, "Initialization routines"	<i>none</i>	<i>none</i>

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

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Subclause 4.18.3.3, "Access routines"	<i>none</i>	<i>none</i>
Subclause 4.18.3.4, "Termination routines"	<i>none</i>	<i>none</i>
Subclause 4.18.3.5, "Decomposition and pass-through modes"	<i>none</i>	<i>none</i>
Subclause 4.18.3.6, "Sequence of actions during the execution of requests involving foreign tables and foreign servers"	<i>none</i>	<i>none</i>
Subclause 4.18.4, "Return codes"	<i>none</i>	<i>none</i>
Subclause 4.18.5, "Foreign-data wrapper diagnostics areas"	<i>none</i>	<i>none</i>
Subclause 4.18.6, "Null pointers"	<i>none</i>	<i>none</i>
Subclause 4.18.7, "Foreign-data wrapper descriptor areas"	<i>none</i>	<i>none</i>
Subclause 4.19, "Introduction to SQL/CLI"	Subclause 4.1, "Introduction to SQL/CLI"	ISO/IEC 9075-3
Clause 5, "Lexical elements"	Clause 4, "Lexical elements"	ISO/IEC 9075-2
Subclause 5.1, "<token> and <separator>"	Subclause 5.2, "<token> and <separator>"	ISO/IEC 9075-2
Subclause 5.2, "Names and identifiers"	Subclause 5.4, "Names and identifiers"	ISO/IEC 9075-2
Clause 6, "Scalar expressions"	Clause 6, "Scalar expressions"	ISO/IEC 9075-2
Subclause 6.1, "<data type>"	Subclause 6.1, "<data type>"	ISO/IEC 9075-2
Subclause 6.2, "<column reference>"	Subclause 6.6, "<column reference>"	ISO/IEC 9075-2
Subclause 6.3, "<set function specification>"	Subclause 6.16, "<set function specification>"	ISO/IEC 9075-2
Subclause 6.4, "<string value function>"	Subclause 6.18, "<string value function>"	ISO/IEC 9075-2
Subclause 6.5, "<datalink value function>"	<i>none</i>	<i>none</i>
Subclause 6.6, "<cast specification>"	Subclause 6.22, "<cast specification>"	ISO/IEC 9075-2
Subclause 6.7, "<value expression>"	Subclause 6.23, "<value expression>"	ISO/IEC 9075-2
Subclause 6.8, "<datalink value expression>"	<i>none</i>	<i>none</i>

ISO/IEC 9075-9:2001 (E)
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 another part	Part containing correspondence
Clause 7, "Query expressions"	Clause 7, "Query expressions"	ISO/IEC 9075-2
Subclause 7.1, "<table reference>"	Subclause 7.6, "<table reference>"	ISO/IEC 9075-2
Subclause 7.2, "<joined table>"	Subclause 7.7, "<joined table>"	ISO/IEC 9075-2
Subclause 7.3, "<group by clause>"	Subclause 7.8, "<group by clause>"	ISO/IEC 9075-2
Subclause 7.4, "<query specification>"	Subclause 7.11, "<query specification>"	ISO/IEC 9075-2
Subclause 7.5, "<query expression>"	Subclause 7.12, "<query expression>"	ISO/IEC 9075-2
Clause 8, "Predicates"	Clause 8, "Predicates"	ISO/IEC 9075-2
Subclause 8.1, "<unique predicate>"	Subclause 8.18, "<unique predicate>"	ISO/IEC 9075-2
Clause 9, "URLs"	<i>none</i>	<i>none</i>
Subclause 9.1, "URL format"	<i>none</i>	<i>none</i>
Clause 10, "Data assignment rules and routine determination"	Clause 9, "Data assignment rules and routine determination"	ISO/IEC 9075-2
Subclause 10.1, "Retrieval assignment"	Subclause 9.1, "Retrieval assignment"	ISO/IEC 9075-2
Subclause 10.2, "Store assignment"	Subclause 9.2, "Store assignment"	ISO/IEC 9075-2
Subclause 10.3, "Data types of results of aggregations"	Subclause 9.3, "Data types of results of aggregations"	ISO/IEC 9075-2
Subclause 10.4, "Type precedence list determination"	Subclause 9.5, "Type precedence list determination"	ISO/IEC 9075-2
Subclause 10.5, "Determination of identical values"	Subclause 9.0, "Determination of identical values"	ISO/IEC 9075-2/Cor.1
Clause 11, "Additional common elements"	Clause 10, "Additional common elements"	ISO/IEC 9075-2
Subclause 11.1, "<privileges>"	Subclause 10.5, "<privileges>"	ISO/IEC 9075-2
Subclause 11.2, "<generic options>"	<i>none</i>	<i>none</i>
Subclause 11.3, "<alter generic options>"	<i>none</i>	<i>none</i>
Clause 12, "Schema definition and manipulation"	Clause 11, "Schema definition and manipulation"	ISO/IEC 9075-2
Subclause 12.1, "<schema definition>"	Subclause 11.1, "<schema definition>"	ISO/IEC 9075-2
Subclause 12.2, "<drop schema statement>"	Subclause 11.2, "<drop schema statement>"	ISO/IEC 9075-2

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

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Subclause 12.3, "<table definition>"	Subclause 11.3, "<table definition>"	ISO/IEC 9075-2
Subclause 12.4, "<column definition>"	Subclause 11.4, "<column definition>"	ISO/IEC 9075-2
Subclause 12.5, "<unique constraint definition>"	Subclause 11.7, "<unique constraint definition>"	ISO/IEC 9075-2
Subclause 12.6, "<drop column definition>"	Subclause 11.17, "<drop column definition>"	ISO/IEC 9075-2
Subclause 12.7, "<domain definition>"	Subclause 11.23, "<domain definition>"	ISO/IEC 9075-2
Subclause 12.8, "<SQL-invoked routine>"	Subclause 11.49, "<SQL-invoked routine>"	ISO/IEC 9075-2
Subclause 12.9, "<user-defined type definition>"	Subclause 11.40, "<user-defined type definition>"	ISO/IEC 9075-2
Subclause 12.10, "<user-defined cast definition>"	Subclause 11.52, "<user-defined cast definition>"	ISO/IEC 9075-2
Subclause 12.11, "<user-defined ordering definition>"	Subclause 11.54, "<user-defined ordering definition>"	ISO/IEC 9075-2
Subclause 12.12, "<foreign table definition>"	<i>none</i>	<i>none</i>
Subclause 12.13, "<alter foreign table statement>"	<i>none</i>	<i>none</i>
Subclause 12.14, "<add basic column definition>"	<i>none</i>	<i>none</i>
Subclause 12.15, "<alter basic column definition>"	<i>none</i>	<i>none</i>
Subclause 12.16, "<drop basic column definition>"	<i>none</i>	<i>none</i>
Subclause 12.17, "<drop foreign table statement>"	<i>none</i>	<i>none</i>
Clause 13, "Catalog manipulation"	<i>none</i>	<i>none</i>
Subclause 13.1, "<foreign server definition>"	<i>none</i>	<i>none</i>
Subclause 13.2, "<alter foreign server statement>"	<i>none</i>	<i>none</i>
Subclause 13.3, "<drop foreign server statement>"	<i>none</i>	<i>none</i>
Subclause 13.4, "<foreign-data wrapper definition>"	<i>none</i>	<i>none</i>

ISO/IEC 9075-9:2001 (E)
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 another part	Part containing correspondence
Subclause 13.5, "<alter foreign-data wrapper statement>"	<i>none</i>	<i>none</i>
Subclause 13.6, "<drop foreign-data wrapper statement>"	<i>none</i>	<i>none</i>
Subclause 13.7, "<import foreign schema statement>"	<i>none</i>	<i>none</i>
Clause 14, "Access control"	Clause 12, "Access control"	ISO/IEC 9075-2
Subclause 14.1, "<revoke statement>"	Subclause 12.1, "<revoke statement>"	ISO/IEC 9075-2
Subclause 14.2, "<user mapping definition>"	<i>none</i>	<i>none</i>
Subclause 14.3, "<alter user mapping statement>"	<i>none</i>	<i>none</i>
Subclause 14.4, "<drop user mapping statement>"	<i>none</i>	<i>none</i>
Clause 15, "SQL-client modules"	Clause 13, "SQL-client modules"	ISO/IEC 9075-2
Subclause 15.1, "<SQL-client module definition>"	Subclause 13.1, "<SQL-client module definition>"	ISO/IEC 9075-2
Subclause 15.2, "<externally-invoked procedure>"	Subclause 13.3, "<externally-invoked procedure>"	ISO/IEC 9075-2
Subclause 15.3, "Calls to an <externally-invoked procedure>"	Subclause 13.4, "Calls to an <externally-invoked procedure>"	ISO/IEC 9075-2
Subclause 15.4, "<SQL procedure statement>"	Subclause 13.5, "<SQL procedure statement>"	ISO/IEC 9075-2
Subclause 15.5, "Data type correspondences"	Subclause 13.6, "Data type correspondences"	ISO/IEC 9075-2
Clause 16, "Data manipulation"	Clause 14, "Data manipulation"	ISO/IEC 9075-2
Subclause 16.1, "<declare cursor>"	Subclause 14.1, "<declare cursor>"	ISO/IEC 9075-2
Subclause 16.2, "Effect of deleting rows from base tables"	Subclause 14.14, "Effect of deleting rows from base tables"	ISO/IEC 9075-2
Subclause 16.3, "Effect of inserting tables into base tables"	Subclause 14.17, "Effect of inserting tables into base tables"	ISO/IEC 9075-2
Subclause 16.4, "Effect of replacing rows in base tables"	Subclause 14.20, "Effect of replacing rows in base tables"	ISO/IEC 9075-2
Clause 17, "Session management"	Clause 18, "Session management"	ISO/IEC 9075-2
Subclause 17.1, "<set passthrough statement>"	<i>none</i>	<i>none</i>
Clause 18, "Dynamic SQL"	Clause 15, "Dynamic SQL"	ISO/IEC 9075-5

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

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Subclause 18.1, "Description of SQL descriptor areas"	Subclause 15.1, "Description of SQL descriptor areas"	ISO/IEC 9075-5
Subclause 18.2, "<prepare statement>"	Subclause 15.6, "<prepare statement>"	ISO/IEC 9075-2
Subclause 18.3, "<deallocate prepared statement>"	Subclause 15.7, "<deallocate prepared statement>"	ISO/IEC 9075-2
Subclause 18.4, "<describe statement>"	Subclause 15.8, "<describe statement>"	ISO/IEC 9075-5
Subclause 18.5, "<input using clause>"	Subclause 15.9, "<input using clause>"	ISO/IEC 9075-5
Subclause 18.6, "<output using clause>"	Subclause 15.10, "<output using clause>"	ISO/IEC 9075-5
Subclause 18.7, "<execute statement>"	Subclause 15.11, "<execute statement>"	ISO/IEC 9075-5
Subclause 18.8, "<dynamic declare cursor>"	Subclause 15.13, "<dynamic declare cursor>"	ISO/IEC 9075-5
Subclause 18.9, "<allocate cursor statement>"	Subclause 15.14, "<allocate cursor statement>"	ISO/IEC 9075-5
Subclause 18.10, "<dynamic open statement>"	Subclause 15.15, "<dynamic open statement>"	ISO/IEC 9075-5
Subclause 18.11, "<dynamic fetch statement>"	Subclause 15.16, "<dynamic fetch statement>"	ISO/IEC 9075-5
Subclause 18.12, "<dynamic close statement>"	Subclause 15.18, "<dynamic close statement>"	ISO/IEC 9075-5
Clause 19, "Embedded SQL"	Clause 16, "Embedded SQL"	ISO/IEC 9075-5
Subclause 19.1, "<embedded SQL Ada program>"	Subclause 16.3, "<embedded SQL Ada program>"	ISO/IEC 9075-5
Subclause 19.2, "<embedded SQL C program>"	Subclause 16.4, "<embedded SQL C program>"	ISO/IEC 9075-5
Subclause 19.3, "<embedded SQL COBOL program>"	Subclause 16.5, "<embedded SQL COBOL program>"	ISO/IEC 9075-5
Subclause 19.4, "<embedded SQL Fortran program>"	Subclause 16.6, "<embedded SQL Fortran program>"	ISO/IEC 9075-5
Subclause 19.5, "<embedded SQL MUMPS program>"	Subclause 16.7, "<embedded SQL MUMPS program>"	ISO/IEC 9075-5
Subclause 19.6, "<embedded SQL Pascal program>"	Subclause 16.8, "<embedded SQL Pascal program>"	ISO/IEC 9075-5
Subclause 19.7, "<embedded SQL PL/I program>"	Subclause 16.9, "<embedded SQL PL/I program>"	ISO/IEC 9075-5

ISO/IEC 9075-9:2001 (E)
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 another part	Part containing correspondence
Clause 20, "Call-Level Interface specifications"	Clause 5, "Call-Level Interface specifications"	ISO/IEC 9075-3
Subclause 20.1, "<CLI routine>"	Subclause 5.1, "<CLI routine>"	ISO/IEC 9075-3
Subclause 20.2, "Implicit DESCRIBE USING clause"	Subclause 5.5, "Implicit DESCRIBE USING clause"	ISO/IEC 9075-3
Subclause 20.2.1, "CLI-specific status codes"	Subclause 5.12, "CLI-specific status codes"	ISO/IEC 9075-3
Subclause 20.2.2, "Description of CLI item descriptor areas"	Subclause 5.13, "Description of CLI item descriptor areas"	ISO/IEC 9075-3
Subclause 20.2.3, "Other tables associated with CLI"	Subclause 5.14, "Other tables associated with CLI"	ISO/IEC 9075-3
Subclause 20.3, "SQL/CLI data type correspondences"	Subclause 5.15, "SQL/CLI data type correspondences"	ISO/IEC 9075-3
Clause 21, "SQL/CLI routines"	Clause 6, "SQL/CLI routines"	ISO/IEC 9075-3
Subclause 21.1, "BuildDataLink"	<i>none</i>	<i>none</i>
Subclause 21.2, "GetDataLinkAttr"	<i>none</i>	<i>none</i>
Subclause 21.3, "GetInfo"	Subclause 6.38, "GetInfo"	ISO/IEC 9075-3
Clause 22, "SQL/MED common specifications"	<i>none</i>	<i>none</i>
Subclause 22.1, "Description of foreign-data wrapper item descriptor areas"	<i>none</i>	<i>none</i>
Subclause 22.2, "Implicit cursor"	<i>none</i>	<i>none</i>
Subclause 22.3, "Implicit DESCRIBE INPUT USING clause"	<i>none</i>	<i>none</i>
Subclause 22.4, "Implicit DESCRIBE OUTPUT USING clause"	<i>none</i>	<i>none</i>
Subclause 22.5, "Implicit EXECUTE USING and OPEN USING clauses"	<i>none</i>	<i>none</i>
Subclause 22.6, "Implicit FETCH USING clause"	<i>none</i>	<i>none</i>
Subclause 22.7, "Character string retrieval"	<i>none</i>	<i>none</i>
Subclause 22.8, "Binary large object string retrieval"	<i>none</i>	<i>none</i>
Subclause 22.9, "Tables used with SQL/MED"	<i>none</i>	<i>none</i>

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

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Clause 23, “Foreign-data wrapper interface routines”	<i>none</i>	<i>none</i>
Subclause 23.1, “<foreign-data wrapper interface routine>”	<i>none</i>	<i>none</i>
Subclause 23.2, “<foreign-data wrapper interface routine> invocation”	<i>none</i>	<i>none</i>
Subclause 23.3, “Foreign-data wrapper interface wrapper routines”	<i>none</i>	<i>none</i>
Subclause 23.3.1, “AllocWrapperEnv”	<i>none</i>	<i>none</i>
Subclause 23.3.2, “Close”	<i>none</i>	<i>none</i>
Subclause 23.3.3, “ConnectServer”	<i>none</i>	<i>none</i>
Subclause 23.3.4, “FreeExecutionHandle”	<i>none</i>	<i>none</i>
Subclause 23.3.5, “FreeFSConnection”	<i>none</i>	<i>none</i>
Subclause 23.3.6, “FreeReplyHandle”	<i>none</i>	<i>none</i>
Subclause 23.3.7, “FreeWrapperEnv”	<i>none</i>	<i>none</i>
Subclause 23.3.8, “GetNumReplySelectElems”	<i>none</i>	<i>none</i>
Subclause 23.3.9, “GetNumReplyTableRefs”	<i>none</i>	<i>none</i>
Subclause 23.3.10, “GetOpts”	<i>none</i>	<i>none</i>
Subclause 23.3.11, “GetReplySelectElem”	<i>none</i>	<i>none</i>
Subclause 23.3.12, “GetReplyTableRef”	<i>none</i>	<i>none</i>
Subclause 23.3.13, “GetSPDHandle”	<i>none</i>	<i>none</i>
Subclause 23.3.14, “GetSRDHandle”	<i>none</i>	<i>none</i>
Subclause 23.3.15, “GetStatistics”	<i>none</i>	<i>none</i>
Subclause 23.3.16, “GetWPDHandle”	<i>none</i>	<i>none</i>
Subclause 23.3.17, “GetWRDHandle”	<i>none</i>	<i>none</i>
Subclause 23.3.18, “InitRequest”	<i>none</i>	<i>none</i>

ISO/IEC 9075-9:2001 (E)
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 another part	Part containing correspondence
Subclause 23.3.19, "Iterate"	<i>none</i>	<i>none</i>
Subclause 23.3.20, "Open"	<i>none</i>	<i>none</i>
Subclause 23.3.21, "ReOpen"	<i>none</i>	<i>none</i>
Subclause 23.3.22, "TransmitRequest"	<i>none</i>	<i>none</i>
Subclause 23.4, "Foreign-data wrapper interface SQL-server routines"	<i>none</i>	<i>none</i>
Subclause 23.4.1, "AllocDescriptor"	<i>none</i>	<i>none</i>
Subclause 23.4.2, "FreeDescriptor"	<i>none</i>	<i>none</i>
Subclause 23.4.3, "GetAuthorizationId"	<i>none</i>	<i>none</i>
Subclause 23.4.4, "GetDescriptor"	<i>none</i>	<i>none</i>
Subclause 23.4.5, "GetNumSelectElems"	<i>none</i>	<i>none</i>
Subclause 23.4.6, "GetNumServerOpts"	<i>none</i>	<i>none</i>
Subclause 23.4.7, "GetNumTableColOpts"	<i>none</i>	<i>none</i>
Subclause 23.4.8, "GetNumTableOpts"	<i>none</i>	<i>none</i>
Subclause 23.4.9, "GetNumTableRefElems"	<i>none</i>	<i>none</i>
Subclause 23.4.10, "GetNumUserOpts"	<i>none</i>	<i>none</i>
Subclause 23.4.11, "GetNumWrapperOpts"	<i>none</i>	<i>none</i>
Subclause 23.4.12, "GetSelectElem"	<i>none</i>	<i>none</i>
Subclause 23.4.13, "GetSelectElemType"	<i>none</i>	<i>none</i>
Subclause 23.4.14, "GetServerName"	<i>none</i>	<i>none</i>
Subclause 23.4.15, "GetServerOpt"	<i>none</i>	<i>none</i>
Subclause 23.4.16, "GetServerOptByName"	<i>none</i>	<i>none</i>
Subclause 23.4.17, "GetServerType"	<i>none</i>	<i>none</i>
Subclause 23.4.18, "GetServerVersion"	<i>none</i>	<i>none</i>

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

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Subclause 23.4.19, "GetSQLString"	<i>none</i>	<i>none</i>
Subclause 23.4.20, "GetTable-ColOpt"	<i>none</i>	<i>none</i>
Subclause 23.4.21, "GetTable-ColOptByName"	<i>none</i>	<i>none</i>
Subclause 23.4.22, "GetTableOpt"	<i>none</i>	<i>none</i>
Subclause 23.4.23, "GetTableOptByName"	<i>none</i>	<i>none</i>
Subclause 23.4.24, "GetTableRef-Elem"	<i>none</i>	<i>none</i>
Subclause 23.4.25, "GetTableRef-ElemType"	<i>none</i>	<i>none</i>
Subclause 23.4.26, "GetTableRef-TableName"	<i>none</i>	<i>none</i>
Subclause 23.4.27, "GetTableServer-Name"	<i>none</i>	<i>none</i>
Subclause 23.4.28, "GetTRDHandle"	<i>none</i>	<i>none</i>
Subclause 23.4.29, "GetUserOpt"	<i>none</i>	<i>none</i>
Subclause 23.4.30, "GetUserOptByName"	<i>none</i>	<i>none</i>
Subclause 23.4.31, "GetValExprCol-Name"	<i>none</i>	<i>none</i>
Subclause 23.4.32, "GetWrapperLi-braryName"	<i>none</i>	<i>none</i>
Subclause 23.4.33, "GetWrapper-Name"	<i>none</i>	<i>none</i>
Subclause 23.4.34, "Get Wrapper-Opt"	<i>none</i>	<i>none</i>
Subclause 23.4.35, "GetWrapperOpt-ByName"	<i>none</i>	<i>none</i>
Subclause 23.4.36, "SetDescriptor"	<i>none</i>	<i>none</i>
Subclause 23.5, "Foreign-data wrap-per interface general routines"	<i>none</i>	<i>none</i>
Subclause 23.5.1, "GetDiagnostics"	<i>none</i>	<i>none</i>
Clause 24, "Diagnostics manage-ment"	Clause 19, "Diagnostics manage-ment"	ISO/IEC 9075-2
Subclause 24.1, "<get diagnostics statement>"	Subclause 19.1, "<get diagnostics statement>"	ISO/IEC 9075-2

ISO/IEC 9075-9:2001 (E)
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 another part	Part containing correspondence
Clause 25, "Information Schema"	Clause 20, "Information Schema"	ISO/IEC 9075-2
Subclause 25.1, "ATTRIBUTES view"	Subclause 20.11, "ATTRIBUTES view"	ISO/IEC 9075-2
Subclause 25.2, "COLUMN_OPTIONS view"	<i>none</i>	<i>none</i>
Subclause 25.3, "COLUMNS view"	Subclause 20.18, "COLUMNS view"	ISO/IEC 9075-2
Subclause 25.4, "FOREIGN_DATA_WRAPPER_OPTIONS view"	<i>none</i>	<i>none</i>
Subclause 25.5, "FOREIGN_DATA_WRAPPERS view"	<i>none</i>	<i>none</i>
Subclause 25.6, "FOREIGN_SERVER_OPTIONS view"	<i>none</i>	<i>none</i>
Subclause 25.7, "FOREIGN_SERVERS view"	<i>none</i>	<i>none</i>
Subclause 25.8, "FOREIGN_TABLE_OPTIONS view"	<i>none</i>	<i>none</i>
Subclause 25.9, "FOREIGN_TABLES view"	<i>none</i>	<i>none</i>
Subclause 25.10, "USER_MAP_OPTIONS view"	<i>none</i>	<i>none</i>
Subclause 25.11, "USER_MAPPINGS view"	<i>none</i>	<i>none</i>
Subclause 25.12, "Short name views"	Subclause 20.69, "Short name views"	ISO/IEC 9075-2
Clause 26, "Definition Schema"	Clause 21, "Definition Schema"	ISO/IEC 9075-2
Subclause 26.1, "COLUMN_OPTIONS base table"	<i>none</i>	<i>none</i>
Subclause 26.2, "DATA_TYPE_DESCRIPTOR base table"	Subclause 21.15, "DATA_TYPE_DESCRIPTOR base table"	ISO/IEC 9075-2
Subclause 26.3, "FOREIGN_DATA_WRAPPER_OPTIONS base table"	<i>none</i>	<i>none</i>
Subclause 26.4, "FOREIGN_DATA_WRAPPERS base table"	<i>none</i>	<i>none</i>
Subclause 26.5, "FOREIGN_SERVER_OPTIONS base table"	<i>none</i>	<i>none</i>
Subclause 26.6, "FOREIGN_SERVERS base table"	<i>none</i>	<i>none</i>
Subclause 26.7, "FOREIGN_TABLE_OPTIONS base table"	<i>none</i>	<i>none</i>

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

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Subclause 26.8, "FOREIGN_TABLES base table"	<i>none</i>	<i>none</i>
Subclause 26.9, "SQL_SIZING base table"	Subclause 7.2, "SQL_SIZING base table"	ISO/IEC 9075-3
Subclause 26.10, "TABLES base table"	Subclause 21.43, "TABLES base table"	ISO/IEC 9075-2
Subclause 26.11, "USAGE_PRIVILEGES base table"	Subclause 21.50, "USAGE_PRIVILEGES base table"	ISO/IEC 9075-2
Subclause 26.12, "USER_MAPPING_OPTIONS base table"	<i>none</i>	<i>none</i>
Subclause 26.13, "USER_MAPPINGS base table"	<i>none</i>	<i>none</i>
Clause 27, "Status codes"	Clause 22, "Status codes"	ISO/IEC 9075-2
Subclause 27.1, "SQLSTATE"	Subclause 22.1, "SQLSTATE"	ISO/IEC 9075-2
Clause 28, "Conformance"	Clause 8, "Conformance"	ISO/IEC 9075-1
Subclause 28.1, "SQL-server conformance to SQL/MED"	<i>none</i>	<i>none</i>
Subclause 28.2, "Foreign-data-wrapper conformance to SQL/MED"	<i>none</i>	<i>none</i>
Subclause 28.3, "Claims of conformance by SQL-servers"	<i>none</i>	<i>none</i>
Subclause 28.4, "Claims of conformance by foreign-data wrappers"	<i>none</i>	<i>none</i>
Subclause 28.5, "Extensions and options"	<i>none</i>	<i>none</i>
Annex A, "SQL Conformance Summary"	Annex A, "SQL Conformance Summary"	ISO/IEC 9075-2
Annex B, "Implementation-defined elements"	Annex B, "Implementation-defined elements"	ISO/IEC 9075-2
Annex C, "Implementation-dependent elements"	Annex C, "Implementation-dependent elements"	ISO/IEC 9075-2
Annex D, "Deprecated features"	Annex D, "Deprecated features"	ISO/IEC 9075-2
Annex E, "Incompatibilities with ISO/IEC 9075:1992"	Annex E, "Incompatibilities with ISO/IEC 9075:1999"	ISO/IEC 9075-2
Annex F, "Typical header files"	Annex A, "Typical header files"	ISO/IEC 9075-3
Subclause F.1, "C Header File SQLCLI.H"	Subclause A.1, "C header file SQL-CLI.H"	ISO/IEC 9075-3

ISO/IEC 9075-9:2001 (E)
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 another part	Part containing correspondence
Subclause F.2, "COBOL Library Item SQLCLI"	Subclause A.2, "COBOL library item SQLCLI"	ISO/IEC 9075-3
Annex G, "SQL feature and package taxonomy"	Annex F, "SQL feature and package taxonomy"	ISO/IEC 9075-2
Annex H, "SQL/MED model"	<i>none</i>	<i>none</i>
Figure 1, "SQL/MED interfaces"	<i>none</i>	<i>none</i>
Figure 2, "SQL/MED information flow"	<i>none</i>	<i>none</i>
Table 1, "Clause, Subclause, and Table relationships"	<i>none</i>	<i>none</i>
Table 2, "Valid datalink file control options"	<i>none</i>	<i>none</i>
Table 3, "Sequence of actions during foreign server request executions"	<i>none</i>	<i>none</i>
Table 4, "Fields used in foreign-data wrapper diagnostics areas"	<i>none</i>	<i>none</i>
Table 5, "Fields in foreign-data wrapper descriptor areas"	<i>none</i>	<i>none</i>
Table 6, "Data type correspondences for Ada"	Table 18, "Data type correspondences for Ada"	ISO/IEC 9075-2
Table 7, "Data type correspondences for C"	Table 19, "Data type correspondences for C"	ISO/IEC 9075-2
Table 8, "Data type correspondences for COBOL"	Table 20, "Data type correspondences for COBOL"	ISO/IEC 9075-2
Table 9, "Data type correspondences for Fortran"	Table 21, "Data type correspondences for Fortran"	ISO/IEC 9075-2
Table 10, "Data type correspondences for MUMPS"	Table 22, "Data type correspondences for MUMPS"	ISO/IEC 9075-2
Table 11, "Data type correspondences for Pascal"	Table 23, "Data type correspondences for Pascal"	ISO/IEC 9075-2
Table 12, "Data type correspondences for PL/I"	Table 24, "Data type correspondences for PL/I"	ISO/IEC 9075-2
Table 13, "Codes used for SQL data types in Dynamic SQL"	Table 4, "Codes used for SQL data types in Dynamic SQL"	ISO/IEC 9075-5
Table 14, "Abbreviated SQL/CLI generic names"	Table 4, "Abbreviated SQL/CLI generic names"	ISO/IEC 9075-3
Table 15, "SQLSTATE class and subclass values for SQL/CLI-specific conditions"	Table 5, "SQLSTATE class and subclass values for SQL/CLI-specific conditions"	ISO/IEC 9075-3

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

Clause, Subclause, or Table in this part of ISO/IEC 9075	Corresponding Clause, Subclause, or Table from another part	Part containing correspondence
Table 16, "Codes used for implementation data types in SQL/CLI"	Table 7, "Codes for implementation data types in SQL/CLI"	ISO/IEC 9075-3
Table 17, "Codes used for application data types in SQL/CLI"	Table 8, "Codes for application data types in SQL/CLI"	ISO/IEC 9075-3
Table 18, "Codes used to identify SQL/CLI routines"	Table 27, "Codes used to identify SQL/CLI routines"	ISO/IEC 9075-3
Table 19, "Codes and data types for implementation information"	Table 28, "Codes and data types for implementation information"	ISO/IEC 9075-3
Table 20, "Codes used for datalink attributes"	<i>none</i>	<i>none</i>
Table 21, "Data types of attributes"	Table 19, "Data types of attributes"	ISO/IEC 9075-3
Table 22, "SQL/CLI data type correspondences for Ada"	Table 44, "SQL/CLI data type correspondences for Ada"	ISO/IEC 9075-3
Table 23, "SQL/CLI data type correspondences for C"	Table 45, "SQL/CLI data type correspondences for C"	ISO/IEC 9075-3
Table 24, "SQL/CLI data type correspondences for COBOL"	Table 46, "SQL/CLI data type correspondences for COBOL"	ISO/IEC 9075-3
Table 25, "SQL/CLI data type correspondences for Fortran"	Table 47, "SQL/CLI data type correspondences for Fortran"	ISO/IEC 9075-3
Table 26, "SQL/CLI data type correspondences for MUMPS"	Table 48, "SQL/CLI data type correspondences for MUMPS"	ISO/IEC 9075-3
Table 27, "SQL/CLI data type correspondences for Pascal"	Table 49, "SQL/CLI data type correspondences for Pascal"	ISO/IEC 9075-3
Table 28, "SQL/CLI data type correspondences for PL/I"	Table 50, "SQL/CLI data type correspondences for PL/I"	ISO/IEC 9075-3
Table 29, "Codes used for <table reference> types"	<i>none</i>	<i>none</i>
Table 30, "Codes used for <value expression> types"	<i>none</i>	<i>none</i>
Table 31, "Codes used for foreign-data wrapper diagnostic fields"	<i>none</i>	<i>none</i>
Table 32, "Codes used for foreign-data wrapper descriptor fields"	<i>none</i>	<i>none</i>
Table 33, "Codes used for foreign-data wrapper handle types"	<i>none</i>	<i>none</i>
Table 34, "Ability to retrieve foreign-data wrapper descriptor fields"	<i>none</i>	<i>none</i>
Table 35, "Ability to set foreign-data wrapper descriptor fields"	<i>none</i>	<i>none</i>

ISO/IEC 9075-9:2001 (E)
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 another part	Part containing correspondence
Table 36, "Foreign-data wrapper descriptor field default values"	<i>none</i>	<i>none</i>
Table 37, "Codes used for the format of the character string transmitted by GetSQLString()"	<i>none</i>	<i>none</i>
Table 38, "SQL-statement codes"	Table 26, "SQL-statement codes"	ISO/IEC 9075-2
Table 39, "SQLSTATE class and subclass values"	Table 27, "SQLState class and subclass values"	ISO/IEC 9075-2
Table 40, "Implied feature relationships"	Table 30, "Implied feature relationships"	ISO/IEC 9075-2
Table 41, "SQL/MED feature taxonomy for features outside Core SQL"	<i>none</i>	<i>none</i>
Table 42, "Legend for SQL/MED interfaces"	<i>none</i>	<i>none</i>
Table 43, "Legend for SQL/MED information flow"	<i>none</i>	<i>none</i>

3.4 Object identifier for Database Language SQL

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

Format

```

<Part 9 yes> ::=
    <Part 9 conformance> <Part 9 datalink> <Part 9 wrapper>

<Part 9 conformance> ::= 9 | sqlmed2000 <left paren> 9 <right paren>

<Part 9 datalink> ::=
    <Part 9 datalink no>
    | <Part 9 datalink yes>
      <Part 9 datalink CLI> <Part 9 datalink language>

<Part 9 datalink CLI> ::=
    <Part 9 datalink CLI yes>
    | <Part 9 datalink CLI no>

<Part 9 datalink CLI yes> ::=
    1 | datalinkCLIyes <left paren> 1 <right paren>

<Part 9 datalink CLI no> ::=
    0 | datalinkCLIno <left paren> 0 <right paren>

<Part 9 datalink language> ::=
    <Part 9 datalink language yes>

```


3.4 Object identifier for Database Language SQL

| <Part 9 datalink language no>

<Part 9 datalink language yes> ::=
1 | datalinklanguageyes <left paren> 1 <right paren>

<Part 9 datalink language no> ::=
0 | datalinklanguageeno <left paren> 0 <right paren>

<Part 9 datalink yes> ::=
1 | datalinkyes <left paren> 1 <right paren>

<Part 9 datalink no> ::=
0 | datalinkno <left paren> 0 <right paren>

<Part 9 wrapper> ::=
 <Part 9 wrapper no>
 | <Part 9 wrapper SQLserver yes>
 <Part 9 server schemaimport> <Part 9 server getstring>
 | <Part 9 wrapper FDW yes>
 <Part 9 FDW transmit> <Part 9 FDW SQLaware> <Part 9 FDW options>

<Part 9 wrapper SQLserver yes> ::=
1 | sqlserveryes <left paren> 1 <right paren>

<Part 9 server schemaimport> ::=
 <Part 9 server schemaimport yes>
 | <Part 9 server schemaimport no>

<Part 9 server schemaimport yes> ::=
1 | sqlserverschemaimportyes <left paren> 1 <right paren>

<Part 9 server schemaimport no> ::=
0 | sqlserverschemaimportno <left paren> 0 <right paren>

<Part 9 server getstring> ::=
 <Part 9 server getstring yes>
 | <Part 9 server getstring no>

<Part 9 server getstring yes> ::=
1 | sqlservergetstringyes <left paren> 1 <right paren>

<Part 9 server getstring no> ::=
0 | sqlservergetstringno <left paren> 0 <right paren>

<Part 9 wrapper FDW yes> ::=
2 | fdwyes <left paren> 2 <right paren>

<Part 9 FDW transmit> ::=
 <Part 9 FDW transmit yes>
 | <Part 9 FDW transmit no>

<Part 9 FDW transmit yes> ::=
1 | sqlfdwtransmityes <left paren> 1 <right paren>

<Part 9 FDW transmit no> ::=
0 | sqlfdwtransmitno <left paren> 0 <right paren>

<Part 9 FDW sqlaware> ::=
 <Part 9 FDW sqlaware yes>
 | <Part 9 FDW sqlaware no>

ISO/IEC 9075-9:2001 (E)

3.4 Object identifier for Database Language SQL

<Part 9 FDW sqlaware yes> ::=
1 | sqlfdwsqlawareyes <left paren> 1 <right paren>

<Part 9 FDW sqlaware no> ::=
0 | sqlfdwsqlawareno <left paren> 0 <right paren>

<Part 9 FDW options> ::=
<Part 9 FDW options yes>
| <Part 9 FDW options no>

<Part 9 FDW options yes> ::=
1 | sqlfdwoptionsyes <left paren> 1 <right paren>

<Part 9 FDW options no> ::=
0 | sqlfdwoptionsno <left paren> 0 <right paren>

<Part 9 wrapper no> ::=
0 | wrapperno <left paren> 0 <right paren>

Syntax Rules

- 1) Specification of <Part 9 yes> implies that conformance to ISO/IEC 9075-9 is claimed.
- 2) Specification of <Part 9 datalink yes> implies that conformance to Feature M001, "Datalinks", is claimed.
- 3) Specification of <Part 9 datalink no> implies that conformance to Feature M001, "Datalinks", is not claimed.
- 4) Specification of <Part 9 wrapper SQLserver yes> implies that conformance to Feature M004, "Foreign data support", is claimed by an SQL-server.
- 5) Specification of <Part 9 server schemaimport yes> implies that conformance to Feature M005, "Foreign schema support", is claimed by an SQL-server.
- 6) Specification of <Part 9 server schemaimport no> implies that conformance to Feature M005, "Foreign schema support", is not claimed by an SQL-server.
- 7) Specification of <Part 9 server getstring yes> implies that conformance to Feature M006, "Get-SQLString routine", is claimed by an SQL-server.
- 8) Specification of <Part 9 server getstring no> implies that conformance to Feature M006, "Get-SQLString routine", is not claimed by an SQL-server.
- 9) Specification of <Part 9 wrapper FDW yes> implies that conformance to Feature M004, "Foreign data support", is claimed by a foreign-data wrapper.
- 10) Specification of <Part 9 FDW transmit yes> implies that conformance to Feature M007, "TransmitRequest", is claimed by an SQL-server.
- 11) Specification of <Part 9 FDW transmit no> implies that conformance to Feature M007, "TransmitRequest", is not claimed by an SQL-server.
- 12) Specification of <Part 9 FDW sqlaware yes> implies that conformance to Feature M008, "SQL-awareness", is claimed by an SQL-server.

3.4 Object identifier for Database Language SQL

- 13) Specification of <Part 9 FDW sqlaware no> implies that conformance to Feature M008, “SQL-awareness”, is not claimed by an SQL-server.
- 14) Specification of <Part 9 FDW options yes> implies that conformance to Feature M009, “GetOpts and GetStatistics routines”, is claimed by an SQL-server.
- 15) Specification of <Part 9 FDW options no> implies that conformance to Feature M009, “GetOpts and GetStatistics routines”, is not claimed by an SQL-server.
- 16) Specification of <Part 9 wrapper no> implies that conformance to Feature M004, “Foreign data support”, is not claimed.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

(Blank page)

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

4 Concepts

4.1 SQL-environments and their components

4.1.1 SQL-environments

Replace the first paragraph An SQL-environment comprises:

- One SQL-agent.
- One SQL-implementation.
- Zero or more SQL-client modules, containing externally-invoked procedures available to the SQL-agent.
- Zero or more authorization identifiers.
- Zero or more user mappings.
- Zero or more catalogs, each of which contains one or more SQL-schemas, zero or more foreign server descriptors, and zero or more foreign-data wrapper descriptors.
- The sites, principally base tables, that contain SQL-data, as described by the contents of the schemas. This data may be thought of as “the database”, but the term is not used in ISO/IEC 9075, because it has different meanings in the general context.

4.1.2 User mapping concepts

A *user mapping* pairs an authorization identifier with a foreign server descriptor.

4.1.3 Catalogs and schemas

4.1.3.1 Catalogs

Replace first paragraph A *catalog* is a named collection of SQL-schemas, foreign server descriptors, and foreign-data wrapper descriptors in an SQL-environment. The mechanisms for creating and destroying catalogs are implementation-defined.

4.1.4 Foreign servers and descriptors

A *foreign server* is a processor that is not part of the SQL-implementation. A foreign server is described by a foreign server descriptors. A foreign server manages data that is not part of the SQL-environment. An SQL-server and an SQL-client can use a foreign server descriptor, which is a catalog element, to communicate with a foreign server. The data managed by a foreign server can be accessed by an SQL-server or an SQL-client through foreign tables, which are SQL-schema elements.

4.1.5 Foreign-data wrappers and descriptors

A *foreign-data wrapper* provides the mechanism by which an SQL-server can access the data that is managed by a foreign server. A foreign-data wrapper is described by a foreign-data wrapper descriptor.

4.2 Data types

Insert after 7th paragraph SQL defines a predefined data type named by the following <key word>: DATALINK.

Insert after 8th paragraph For reference purposes, the data type DATALINK is referred to as a (or the) *datalink type*.

Insert this paragraph A type *T* is DATALINK-ordered if *T* is *D*-ordered, where *D* is the set of datalink types.

4.3 Foreign servers

A *foreign server* is a named server, external to the SQL-environment but known to the SQL-server, that manages external data. Such external data is manifested as SQL-data by use of a mechanism called a *foreign-data wrapper* (see Subclause 4.4, "Foreign-data wrappers").

A *foreign server descriptor* is a catalog element, identified by a *foreign server name* and created by invoking a <foreign server definition>. A foreign server descriptor consists of:

- A foreign server name, identifying the foreign server locally to the SQL-server.
- The <authorization identifier> of the owner of the resulting foreign server descriptor.
- The name of the foreign-data wrapper.
- A generic options descriptor.
- Optionally, the foreign server type.
- Optionally, the foreign server version.

The possible values of server type and server version, and their meanings, are implementation-defined.

The <authorization identifier> associated with a foreign server identifies the role of managing the capabilities and resources at that foreign server; however, that role is implementation-defined.

A foreign server descriptor is said to be *owned by* or to have been *created by* the current authorization identifier for the SQL-session when the <foreign server definition> was invoked.

A foreign server descriptor can be modified by an <alter foreign server statement> and destroyed by a <drop foreign server statement>.

A foreign server can be an *SQL-aware foreign server* or a *non-SQL-aware foreign server*. An SQL-aware foreign server is a foreign server that has the ability to process a subset of statements conforming to ISO/IEC 9075, particularly the statements comprising Feature E051, in a standard-conforming manner. A non-SQL-aware foreign server is a foreign server that has no ability to process SQL language. If the foreign-data wrapper associated with a non-SQL-aware foreign server

provides some (limited or conforming) ability to process SQL language, then the effect is that the foreign server can be treated as though it is an SQL-aware foreign server.

NOTE 1 – Some SQL-aware foreign servers may be, in fact, SQL-servers. However, because they are not in the same SQL-environment as the SQL-server responding to an SQL-client, they are managed only through foreign-data wrappers and are treated as foreign servers. Such foreign servers may concurrently respond to SQL-clients of their own; this does not change the relationships specified in this part of ISO/IEC 9075.

Some foreign servers, especially SQL-aware foreign servers, admit the concept of a schema and the concept of a table that are similar to SQL-schemas and to base tables, respectively. Such servers may (and SQL-aware foreign servers do) maintain schema information about those entities, such as the Information Schema and Definition Schema specified in ISO/IEC 9075-2.

If a foreign server maintains schema information about entities analogous to SQL-schemas and base tables, then execution of an <import foreign schema statement> retrieves information about the tables (either all or only some, as specified in the <import foreign schema statement>) associated with the named SQL-schema analog and performs one or more effective <foreign table definition> statement executions.

If a foreign server does not maintain such information or does not admit the concept of a schema, then foreign tables managed by that server must be specified by means of explicit <foreign table definition>s.

This International Standard does not specify the manner in which the SQL-server and the foreign-data wrapper interact to cause information about foreign tables to be retrieved by execution of an <import foreign schema statement>. In particular, no foreign-data wrapper interface routines are specified to support such interaction. Such interaction is implementation-dependent.

4.4 Foreign-data wrappers

A foreign-data wrapper is the mechanism by which the SQL-server accesses external data managed by foreign servers. Every foreign server is accessed through exactly one foreign-data wrapper, but one foreign-data wrapper can be used to access several different foreign servers. A foreign-data wrapper is made up of foreign-data wrapper interface routines, a set of routines written in a standard programming language. Foreign-data wrapper interface routines are used to access every foreign server whose descriptor includes the name of that foreign-data wrapper. It is possible for a foreign-data wrapper to exist that is not used to access any foreign server.

A *foreign-data wrapper descriptor* is a catalog element, identified by a *foreign-data wrapper name* and created by invoking a <foreign-data wrapper definition>. A <foreign-data wrapper definition> specifies the foreign-data wrapper name, a library name that identifies a library containing the foreign-data wrapper interface routines, and the name of the language in which the foreign-data wrapper interface routines are written.

A *foreign-data wrapper descriptor* consists of:

- A foreign-data wrapper name.
- The authorization identifier of the owner of the foreign-data wrapper descriptor.
- Name of the language in which the foreign-data wrapper interface routines are written.
- A generic options descriptor.
- A library name.

A foreign-data wrapper descriptor can be modified by an `<alter foreign-data wrapper statement>` and destroyed by a `<drop foreign-data wrapper statement>`.

4.5 User mappings

A user mapping is an SQL-environment element, pairing an authorization identifier *U* or the special identifier PUBLIC, denoting all `<authorization identifier>`s in the SQL-environment, with a foreign server *FS*. It defines how to map *U* to an equivalent concept known to *FS* when a foreign table whose source is *FS* is to be accessed during an SQL-session with current authorization identifier *U*. The mapping is specified by generic options defined by the foreign-data wrapper.

A user mapping is defined by invoking an `<user mapping definition>`. Invocation of an `<user mapping definition>` results in the creation of a user mapping descriptor in a catalog. A user mapping descriptor consists of:

- An authorization identifier.
- A foreign server name, identifying a foreign server descriptor.
- A generic options descriptor.

A user mapping descriptor can be modified by an `<alter user mapping statement>` and destroyed by a `<drop user mapping statement>`.

4.6 Generic options

Several of the objects used in connection with external data support the specification of *generic options*. These objects are foreign-data wrappers, foreign servers, foreign tables, columns of foreign tables, and user mappings. A generic option is an option name paired with an optional option value. Both the option name and the permissible ranges of option values of a generic option are defined by the foreign-data wrappers. A set of generic options is described by a generic options descriptor. A generic options descriptor is included in the descriptor of the object to which it pertains. The generic options are stored in the SQL-server for the foreign-data wrapper to retrieve when the foreign-data wrapper needs this information.

Generic options may be specified in either `<foreign-data wrapper definition>`, `<foreign server definition>`, `<foreign table definition>`, `<user mapping definition>`, `<alter foreign-data wrapper statement>`, `<alter foreign server statement>`, `<alter foreign table statement>`, or `<alter user mapping statement>`.

Generic options are specific to the object for which they are defined. For example, the generic options for a foreign table are most likely different from the generic options for a foreign server, in both option names and option values. Furthermore, generic options are highly dependent on the foreign-data wrapper that is used to access the external data. For example, the generic options for a foreign server that uses a foreign-data wrapper *A* might be totally different from the generic options specified for another foreign server that uses a foreign-data wrapper *B*. Even the fact that the option names of two generic options for two different foreign-data wrappers might be the same does not necessarily mean that the semantics and therefore the permissible range of option values are the same.

Since an SQL-server cannot anticipate the different kinds of foreign-data wrappers with which it is likely to deal, no generic option can ever be determined by the SQL-server or by this part of ISO/IEC 9075. Only a foreign-data wrapper can specify generic options for that foreign-data wrapper, or for a foreign server, a foreign table, a column of a foreign table or a user mapping for which it is used.

A *generic options descriptor* is either an empty list or a list consisting of one or more option names, each option name being paired with at most one option value.

4.7 Capabilities and options information

The SQL-server needs information from the foreign-data wrapper about the capabilities of the foreign-data wrapper itself, about the foreign server accessed through the foreign-data wrapper, and about certain schema elements (foreign tables and their columns, user mappings) managed by the foreign server. The SQL-server also needs information about options supported by the foreign-data wrapper, the foreign server, and certain schema elements. The SQL-server invokes the `GetOpts( )` routine to request the capabilities and other information from a foreign-data wrapper.

The specific capabilities and other information of a foreign-data wrapper, a foreign server, or any schema element managed by a foreign server that are reported to the SQL-server in response to an invocation of `GetOpts( )` are partly specified in this part of ISO/IEC 9075 and partly implementation-defined. In general, each capability or other piece of information that is reported corresponds to a generic option associated with the object being queried by the invocation.

The capabilities and other information is returned in a buffer whose contents may comprise an XML document or that may be returned in a format defined by the foreign-data wrapper. If the contents comprise an XML document, then it shall be a valid XML document, the format of which is specified by a Document Type Declaration (DTD) that is either internal to the XML document or external (requiring that it be available to the SQL-server in an implementation-defined manner).

NOTE 2 – This edition of this part of ISO/IEC 9075 specifies the use of a DTD. Future editions may specify the use of an XML Schema, either as an alternative to a DTD or instead of a DTD.

4.8 Datalinks

A datalink is a value of the DATALINK data type. A datalink references some file that is not part of the SQL-environment. The file is assumed to be managed by some external file manager. A datalink is conceptually represented by:

- File Reference: A character string forming a reference to an external file.
- SQL-Mediated Access Indication: A boolean value, where *True*, in data link *DL* indicates that the referenced file, being linked to the SQL-environment, is accessible only by use of the specially provided operations (see below) on *DL*.

The File Reference of a datalink is accessible by invoking built-in scalar functions defined in this part of ISO/IEC 9075. The character set of the File Reference, referred to as the *datalink character set* is implementation-defined.

The purpose of datalinks is to provide a mechanism to synchronize the integrity control, recovery, and access control of the files and the SQL-data associated with them. This part of ISO/IEC 9075 standardizes the way that an SQL-server is made aware of datalink values and how applications retrieve information about the files identified by datalink values. The mechanisms that enable integrity control, recovery, and access control for the files represented by the datalink values are implementation-dependent. These mechanisms are collectively called the *datalinker*.

A file is *linked* to the SQL-environment whenever execution of an SQL-data change statement causes a value *DL1* that references that file to appear in some datalink column whose descriptor includes the link control FILE LINK CONTROL. If the read permission option included in the column descriptor is DB, then access to the referenced file is said to be *SQL-mediated*. This is

ISO/IEC 9075-9:2001 (E)

4.8 Datalinks

indicated by setting the SQL-Mediated Access Indication of *DL1* to *True* , and *DL1* is said to be an *SQL-mediated datalink*. If the read permission option included in the column descriptor is not DB, then the Linked Indication of *DL1* is set to *False* .

Execution of an SQL-data change statement that causes a value *DL2* to appear in a datalink column defined with the link control NO LINK CONTROL does not cause any file to be linked to the SQL-environment.

A linked file cannot be renamed or deleted by any agency outside of the SQL-environment. A datalink value always references just one file. A file is *unlinked* from the SQL-environment whenever execution of an SQL-data change statement causes a datalink that references that file to be removed from some datalink column whose descriptor includes the link control FILE LINK CONTROL. The actions that occur when a datalink is removed from a column depend on the link control options that are specified in the column descriptor of that column. The file might be deleted, or the datalinker might return control of the file to the external data manager.

With the function provided by datalinks and the datalinker, it is possible to specify that access to the files should be mediated by the SQL-server rather than by the external data manager. When access to the files is mediated by an SQL-server, any request to access a file must operate on an SQL-mediated datalink to obtain a character string with which to reference the file, using one of the built-in functions provided for that purpose. This character string is constructed by combining the File Reference of a datalink value with an encrypted value called an *access token*. The generation of the access token and the method of combining it with the File Reference is implementation-dependent. When the application uses the returned character string value to access a file, the datalinker checks to see if the access token is *valid*. If it is valid, then the application is allowed to access the file pointed to by the File Reference. Every attempt by an application to access, without a valid access token, a file referenced by an SQL-mediated datalink is unsuccessful. The time at which a valid access token ceases to be valid is implementation-defined.

A datalink data type is described by a *datalink data type descriptor*. A datalink data type descriptor consists of the name DATALINK and the set of *link control options*:

- The link control (NO LINK CONTROL or FILE LINK CONTROL).
- The integrity control option (ALL, SELECTIVE, or NONE).
- The read permission option (FS or DB).
- The write permission option (FS or BLOCKED).
- The recovery option (NO or YES).
- The unlink option (RESTORE, DELETE, or NONE).

The meanings of the various link control options are:

- NO LINK CONTROL: Although every File Reference must conform to the Format and Syntax Rules of Subclause 9.1, "URL format", it is permitted for there to be no file referenced by that File Reference. This option implies that the integrity control option is NONE, the read permission option is FS, the write permission option is FS, the recovery option is NO and the unlink option is NONE, and no explicit syntax to specify these options is permitted.
- FILE LINK CONTROL: Every File Reference must reference an existing file. Further file control depends on the link control options.

- INTEGRITY ALL: Files referenced by File References cannot be deleted or renamed, except possibly through the use of operations on the column in question, invoked as part of some SQL-session.
- INTEGRITY SELECTIVE: Files referenced by File References can be deleted or renamed using operators provided by the file manager, unless a datalinker is installed in connection with the file manager.
- INTEGRITY NONE: Files referenced by File References can only be deleted or renamed using operators provided by the file manager. This option is not available if FILE LINK CONTROL is specified.
- READ PERMISSION FS: Permission to read files referenced by datalinks is determined by the file manager.
- READ PERMISSION DB: Permission to read files referenced by datalinks is determined by the SQL-implementation.
- WRITE PERMISSION FS: Permission to write files referenced by datalinks is determined by the file manager.
- WRITE PERMISSION BLOCKED: Write access to files referenced by datalinks is not available. Updates can, however, arise indirectly through the use of some implementation-defined mechanism.
- RECOVERY YES: Enables *point in time recovery* of files referenced by datalinks.
NOTE 3 – “point in time recovery” is an implementation-defined mechanism that provides for recovery that is coordinated between the SQL-server and the files of external file manager referenced by datalinks.
- RECOVERY NO: Point in time recovery of files referenced by datalinks is disabled.
- ON UNLINK RESTORE: When a file referenced by a datalink is unlinked, the external file manager attempts to reinstate the ownership and permissions that existed when that file was linked.
- ON UNLINK DELETE: A file referenced by a datalink is deleted when it is unlinked.
- ON UNLINK NONE: When a file referenced by a datalink is unlinked, there is no change in the ownership and permissions occasioned by that unlinking.

Table 2, “Valid datalink file control options”, specifies what combinations of datalink file control options are allowed.

The default value of a site whose declared type is DATALINK is the null value. Datalinks are subject to certain restrictions. As a consequence of these restrictions, neither datalinks nor expressions whose declared type is DATALINK-ordered can appear in (among other places):

- <comparison predicate>.
- <general set function>.
- <group by clause>.
- <order by clause>.
- <unique constraint definition>.

ISO/IEC 9075-9:2001 (E)

4.8 Datalinks

- <referential constraint definition>.
- <select list> of a <query specification> that has a <set quantifier> of DISTINCT.
- <select list> of an operand of UNION, INTERSECT, and EXCEPT.
- Columns used for matching when forming a <joined table>.

Table 2—Valid datalink file control options

Integrity	Read permission	Write permission	Recovery	Unlink
ALL	FS	FS	NO	NONE
ALL	FS	BLOCKED	NO	RESTORE
ALL	FS	BLOCKED	YES	RESTORE
ALL	DB	BLOCKED	NO	RESTORE
ALL	DB	BLOCKED	NO	DELETE
ALL	DB	BLOCKED	YES	RESTORE
ALL	DB	BLOCKED	YES	DELETE
SELECTIVE	FS	FS	NO	NONE

The implementation-defined *maximum datalink length* determines the amount of space, in octets, that is allocated for:

- A host variable of data type DATALINK.
- An argument of declared type DATALINK to an invocation of an external routine.
- The value returned by an invocation of an external function whose result type is DATALINK.

The maximum datalink length constrains the values of expressions whose declared type is DATALINK such that every such value can be assigned to a host variable, substituted for a parameter to an external routine, or returned by an invocation of an external function.

4.8.1 Operations involving datalinks

4.8.1.1 Operators that operate on datalinks

<url complete expression> returns the File Reference of a given datalink, possibly combined with an access token.

<url path expression> returns the path, including any access token, of the File Reference of a given datalink.

<url path only expression> returns the path, excluding any access token, of the File Reference of a given datalink.

<url scheme expression> returns the scheme of the File Reference of a given datalink.

<url server expression> returns the host of the File Reference of a given datalink.

NOTE 4 – “host”, “scheme”, and “path” are defined in Subclause 6.5, “<datalink value function>”.

4.8.1.2 Other operators involving datalinks

<datalink value constructor>, given a File Reference, returns the corresponding datalink.

4.9 Type conversion and mixing of data types

New paragraph Datalinks are not comparable. A datalink is assignable only to sites of data type DATALINK.

4.10 Columns, fields, and attributes

Insert following 7th paragraph An *immediate component* of a column value *C* is any of the following:

- *C*.
- An attribute of a structured type *ST*, where *ST* is the declared type of *C*.
- A field of a row of type *RT*, where *RT* is the declared type of *C*.
- An element of a collection of type *CT*, where *CT* is the declared type of *C*.

A *component* of a column value *C* is any immediate component of *C* or an immediate component a component *C1* of *C*.

4.11 Tables

Replaces 3rd paragraph A table is either a base table, a derived table, or a foreign table. A base table is either a persistent base table, a global temporary table, a created local temporary table, or a declared local temporary table.

Replaces 10th paragraph A table is described by a table descriptor. A table descriptor is either a base table descriptor, a view descriptor, a derived table descriptor (for a derived table that is not a view), or a foreign table descriptor.

Insert this paragraph The data constituting a foreign table is not part of the SQL-environment. Instead, its rows are supplied when needed by some foreign server, known as the *source* of the foreign table. The mechanism by which these rows are supplied is provided by a *foreign-data wrapper* (see Subclause 4.4, "Foreign-data wrappers"). A foreign table descriptor describes a foreign table. In addition to the components of every table descriptor, a foreign table descriptor includes:

- The name of the foreign table.
- A foreign server name, identifying the descriptor of the foreign server that is the source of the foreign table.
- A generic options descriptor.
- An indication of whether the foreign table is updatable or not.

NOTE 5 – This part of ISO/IEC 9075 currently restricts foreign tables such that they are neither insertable-into nor updatable. Future versions of this part of ISO/IEC 9075 may relax these restrictions.

4.11.1 Types of tables

- Replace 1st bullet following 5th paragraph If *TORQN* identifies a base table or a foreign table, then *TORQN* has no generally underlying tables.

4.12 Functional dependencies

Replace 1st paragraph This Subclause defines *functional dependency* and specifies a minimal set of rules that a conforming implementation must follow to determine functional dependencies and candidate keys in base tables, foreign tables, and <query expression>s.

4.12.1 Known functional dependencies in a foreign table

There are no rules in this part of ISO/IEC 9075 to determine known functional dependencies in a foreign table. However, implementation-defined rules may determine known functional dependencies, if any, in a foreign table.

4.13 SQL-schemas

Replace 5th paragraph Base tables, foreign tables, and views are identified by <table name>s. A <table name> consists of a <schema name> and an <identifier>. The <schema name> identifies the schema in which a persistent base table, foreign table, or view identified by the <table name> is defined. Base tables, foreign tables, and views defined in different schemas can have <identifier>s that are equal according to the General Rules of Subclause 8.2, "<comparison predicate>", in ISO/IEC 9075-2.

4.14 SQL-statements

4.14.1 SQL-statements classified by function

New paragraph The following are additional SQL-schema statements:

- <import foreign schema statement>
- <foreign table definition>
- <alter foreign table statement>
- <drop foreign table statement>
- <foreign server definition>
- <alter foreign server statement>
- <drop foreign server statement>
- <foreign-data wrapper definition>
- <alter foreign-data wrapper statement>

- <drop foreign-data wrapper statement>
- <user mapping definition>
- <alter user mapping statement>
- <drop user mapping statement>

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

- <set passthrough statement>

4.15 SQL-sessions

Insert this paragraph At any time during an SQL-session, the SQL-server may obtain a WrapperEnvHandle for a foreign-data wrapper and an FSConnectionHandle for a foreign server.

Insert this paragraph The SQL-session context also comprises:

- Zero or more {foreign-data wrapper name : WrapperEnvHandle} pairs.
- Zero or more {foreign server name : FSConnectionHandle} pairs.
- A pass-through flag.
- A pass-through foreign-server name, if any.
- Zero or more {<statement name> : ExecutionHandle} pairs.

Insert this paragraph At the end of every SQL-session, every FSConnection handle that is contained in the SQL-session context is freed.

Insert this paragraph At the end of every SQL-session, every WrapperEnv handle that is contained in the SQL-session context is freed.

Insert this paragraph An SQL-session has a pass-through flag that is initially set to False when the SQL-session is started. The successful execution of a <set passthrough statement> that contains a <foreign server name> changes the pass-through flag to True. An SQL-session whose pass-through flag is True additionally has a pass-through foreign server name. Every time a <set passthrough statement> is executed, all {<SQL statement name> : ExecutionHandle} pairs are removed from the current SQL-session context. Every time a <set passthrough statement> that contains a <foreign server name> FSN is successfully executed, the pass-through foreign server name included the current SQL-session context is set to FSN. Every time a <prepare statement> is executed after a <set passthrough statement> that contains a <foreign server name> has been executed successfully, an {<SQL statement name> : ExecutionHandle} pair is made part of the current SQL-session context. Every time a <deallocate prepared statement> is successfully executed after a <set passthrough statement> that identifies a <foreign server name> has been executed, the corresponding {<SQL statement name> : ExecutionHandle} pair is removed from the current SQL-session context. Every time a <set passthrough statement> that specifies 'OFF' is executed, the pass-through flag is set to False and the pass-through foreign server name is deleted from the current SQL-session context.

4.16 Privileges

Replace 1st paragraph A privilege authorizes a given category of <action> to be performed on a specified base table, foreign table, view, column, domain, character set, collation, translation, foreign-data wrapper, foreign server, user-defined type, trigger, or SQL-invoked routine by a specified <authorization identifier>.

NOTE 6 – Privileges granted on foreign tables are not privileges to use the data constituting foreign tables, but privileges to use the definitions of the foreign tables. The privileges to access the data constituting the foreign tables are enforced by the foreign server, based on the user mapping. Consequently, a request by an SQL-client to access external data may raise exceptions.

- **Replace 1st bullet of 2nd paragraph** The identification of the base table, foreign table, view, column, domain, character set, collation, translation, foreign-data wrapper, foreign server, user-defined type, table/method pair, trigger, or SQL-invoked routine module that the descriptor describes.

Replace 8th paragraph A privilege descriptor with an <action> of USAGE is called a *usage privilege descriptor* and identifies the existence of a privilege on the domain, user-defined type, character set, collation, foreign-data wrapper, foreign server, or translation identified by the privilege descriptor.

4.17 SQL-transactions

Replace 5th paragraph An SQL-transaction has an *access mode* that is either *read-only* or *read-write*. The access mode may be explicitly set by a <set transaction statement> before the start of an SQL-transaction or by the use of a <start transaction statement> to start an SQL-transaction; otherwise, it is implicitly set to the default access mode for the SQL-session before each SQL-transaction begins. If no <set session characteristics statement> has set the default access mode for the SQL-session, then the default access mode for the SQL-session is *read-write*. The term *read-only* applies only to viewed tables, foreign tables, and persistent base tables.

4.18 Foreign-data wrapper interface

A foreign-data wrapper interface consists of the signatures of the routines that make up a given foreign-data wrapper. These routines serve the following purposes:

- Allocate and deallocate resources.
- Control connections to foreign servers.
- Receive data from the SQL-server about the SQL statement to be executed at the foreign server.
- Send data from the foreign server to the SQL-server about the SQL statement that the foreign server is willing to execute.
- Initiate and terminate the execution of SQL statements by the foreign server.

4.18.1 Handles

A *handle* is a value of INTEGER data type that identifies an allocated resource that provides session state information about a foreign server, a foreign-data wrapper, or a foreign server session of interest to connected components of that session. Handles presented as arguments to invocations of foreign-data wrapper interface routines enable the invoker to give or obtain the information they reference. The handle of a particular resource is allocated by the keeper of the state information — either the foreign-data wrapper or the SQL-server — to enable the SQL-server or the foreign-data wrapper, respectively, to access that state information. Although the declared type for a handle is INTEGER, its value has no meaning in any other context and should not be used as a numeric operand or modified in any way.

The following are the handles specified in the foreign-data wrapper interface, presented in approximately the order in which they are materialized in Table 3, “Sequence of actions during foreign server request executions”. The operations that cause their creation and destruction are given in Table 3, “Sequence of actions during foreign server request executions”.

- **WrapperEnv handle:** This handle is allocated by a foreign-data wrapper to reference information during the interaction with the SQL-server. It identifies an *allocated FDW-environment* and is allocated via a call from the SQL-server to the `AllocWrapperEnv( )` routine. This handle must be allocated before any requests are made to a foreign-data wrapper. It remains valid until the SQL-server invokes `FreeWrapperEnv(wh)`, where *wh* is the handle in question.
- **Server handle:** This handle is allocated by the SQL-server to reference a foreign server. A foreign-data wrapper uses this handle to obtain information about a foreign server to which it needs to connect. Routines associated with a server handle allow information to be obtained about such things as the server name, server type, server version, *etc.* This handle is allocated implicitly and presented by the SQL-server to a foreign-data wrapper by invoking `ConnectServer( )`.
- **FSConnection handle:** This handle is allocated by a foreign-data wrapper to reference information about a foreign server session. It is allocated via a call to the `ConnectServer( )` routine. This handle must be allocated before any SQL statements to be executed during that foreign server session are presented to a foreign-data wrapper.
- **Request handle:** This handle is allocated by the SQL-server to reference an SQL-statement that is to be executed by a foreign server. A request handle may reference a simple statement, such as `SELECT * FROM T`, or it may reference a complex statement that includes predicates, joins, ordering, *etc.* A request handle is presented to the SQL-server by a foreign-data wrapper interface routine to retrieve (for example) the names of foreign tables referenced in the *from* clause, the names of column references in the *select* list, *etc.* This handle is allocated implicitly.
- **Table Reference handle:** This handle is allocated by the SQL-server to reference a <table reference> contained in the <from clause> of a <query specification>. This handle is allocated implicitly.
- **Value Expression handle:** This handle is allocated by the SQL-server to reference a <value expression> simply contained in the <select list> of a <query specification>. This handle is allocated implicitly.

NOTE 7 — The <value expression> referenced by a Value Expression handle is constrained to be a <column reference>.

4.18 Foreign-data wrapper interface

- **Reply handle:** This handle is allocated by a foreign-data wrapper to reference the subset of SQL-statements it is capable of executing. This handle is allocated via a call during a foreign server session to the `InitRequest( )` routine and remains valid in that foreign server session until it is the argument to an invocation of `FreeReplyHandle( )`.
 - **Execution handle:** This handle is allocated by a foreign-data wrapper. In decomposition mode, it is used to reference the information the foreign-data wrapper needs to process the SQL-statement reference by the corresponding reply handle and the information associated with the data resulting from the processing of the SQL-statement. This handle is allocated via a call to the `InitRequest( )` routine, which sets the associated `PASSTHROUGH` flag to *False*. In pass-through mode, this handle is used to reference the information that the foreign-data wrapper needs to process the statement that is sent to the foreign server. This handle is allocated via a call to the `TransmitRequest( )` routine, which sets the associated `PASSTHROUGH` flag to *True*.
- NOTE 8 – “decomposition mode” and “pass-through mode” are defined in Subclause 4.18.3.5, “Decomposition and pass-through modes”.
- **Wrapper handle:** This handle is allocated implicitly by the SQL-server to reference the information about a foreign-data wrapper.
 - **User handle:** This handle is allocated implicitly by the SQL-server to reference information about the user on whose behalf a connection to a foreign server is being made.
 - **Descriptor handle:** This handle is allocated by either the SQL-server or a foreign-data wrapper to reference a foreign-data wrapper descriptor area.

4.18.2 Foreign server sessions

A *foreign server session* is the sequence of operations performed by a foreign-data wrapper on a particular `FSConnection` handle during the existence of that handle.

A foreign server session on `FSConnection` handle *FSCH* begins with the invocation of `ConnectServer( )` that brings *FSCH* into existence and ends with the invocation of `FreeFSConnection(FSCH)`.

4.18.3 Foreign-data wrapper interface routines

The terms *foreign-data wrapper interface SQL-server routine* and *foreign-data wrapper interface wrapper routine* are used to distinguish routines provided by the SQL-server from routines provided by a foreign-data wrapper, respectively. A foreign-data wrapper interface routine that is both an SQL-server routine and a wrapper routine is referred to as a *foreign-data wrapper interface general routine*.

The foreign-data wrapper interface routines of a given implementation are either all functions or all procedures, the choice being implementation-defined. They are functions if their return codes are values returned by their invocations and they are procedures if their return codes are instead assigned to an output parameter named `ReturnCode`. The specific terms *foreign-data wrapper interface function* and *foreign-data wrapper interface procedure* are used when it is necessary to distinguish between the two kinds.

4.18.3.1 Handle routines

- **GetServerName:** This routine returns the name of a foreign server given a server handle.
- **GetServerType:** This routine returns the type of a foreign server given a server handle.
- **GetServerVersion:** This routine returns the version of a foreign server given a server handle.
- **GetNumServerOpts:** This routine returns the number of generic options associated with a foreign server given a server handle.
- **GetServerOpt:** This routine returns the generic option name and its value given a server handle and the position of the option in the options list.
- **GetServerOptByName:** This routine returns the generic option value given a server handle and the name of the option.
- **GetNumTableRefElems:** This routine returns the number of <table reference>s in the <from clause> of a query given a request handle.
- **GetTableRefElem:** This routine returns the table reference handle of a <table reference> in the <from clause> of a query given a request handle and the position of the <table reference> in the <from clause>.
- **GetTableRefElemType:** This routine returns the “type” of a <table reference> given the table reference handle. The only possible return value is TABLE_NAME.
- **GetTableRefTableName:** This routine returns the table name given a table reference handle.
- **GetNumSelectElems:** This routine returns the number of <value expression>s in the <select list> of a <query specification> given a request handle.
- **GetSelectElem:** This routine returns the value expression handle of a <value expression> in the <select list> of a <query specification> given a request handle and the position of the <value expression> in the <select list>.
- **GetSelectElemType:** This routine returns the “type” of a <value expression> given the value expression handle. The only possible return value is COLUMN_NAME.
- **GetValExprColumnName:** This routine returns the name of the column, given a value expression handle.
- **GetNumReplyTableRefs:** This routine returns the number of table references from the original request that the foreign-data wrapper is capable of accessing, given a reply handle.
- **GetReplyTableRef:** This routine returns the number of the table reference in the original request that the foreign-data wrapper is capable of accessing, given a reply handle and a number that ranges from 1 (one) to the value returned by the GetNumReplyTableRefs() routine.
- **GetNumReplySelectElems:** This routine returns the number of select list elements from the original request that the foreign-data wrapper is capable of accessing, given a reply handle.

4.18 Foreign-data wrapper interface

- **GetReplySelectElem:** This routine returns the number of the select list element in the original request that the foreign-data wrapper is capable of accessing, given a reply handle and a number that ranges from 1 (one) to the value returned by the `GetNumReplySelectElems()` routine.
- **GetAuthorizationId:** This routine returns the authorization identifier associated with a user mapping, given a user handle.
- **GetTableColOpt:** This routine returns the generic option name and its value, given a table reference handle, column name and the position of the option in the options list.
- **GetTableColOptByName:** This routine returns the generic option value, given a table reference handle, a column name and the name of the option.
- **GetTableOpt:** This routine returns the generic option name and its value, given a table reference handle and the position of the option in the options list.
- **GetTableOptByName:** This routine returns the generic option value, given a table reference handle and the name of the option.
- **GetTableServerName:** This routine returns the name of the foreign server associated with a foreign table, given a table reference handle.
- **GetNumTableColOpts:** This routine returns the number of generic options associated with a column of a foreign table, given a table reference handle and a column name.
- **GetNumTableOpts:** This routine returns the number of generic options associated with a foreign table, given a table reference handle.
- **GetNumUserOpts:** This routine returns the number of generic options associated with a user mapping, given a user handle.
- **GetNumWrapperOpts:** This routine returns the number of generic options associated with a foreign-data wrapper, given a wrapper handle.
- **GetUserOpt:** This routine returns the generic option name and its value, given a user handle and the position of the option in the options list.
- **GetUserOptByName:** This routine returns the generic option value, given a user handle and the name of the option.
- **GetWrapperLibraryName:** This routine returns the name of the library associated with a foreign-data wrapper, given a wrapper handle.
- **GetWrapperName:** This routine returns the name of a foreign-data wrapper, given a wrapper handle.
- **GetWrapperOpt:** This routine returns the generic option name and its value, given a wrapper handle and the position of the option in the options list.
- **GetWrapperOptByName:** This routine returns the generic option value, given a wrapper handle and the name of the option.
- **GetDescriptor:** This routine, given a descriptor handle and the identification of a descriptor area field, retrieves the value of the specified field from a descriptor area.

- **SetDescriptor:** This routine, given a descriptor handle, the identification of a descriptor area field, and a new value to be assigned to that field, sets the value of the specified field of a descriptor area.
- **GetSPDHandle:** This routine returns the SPDHandle given an ExecutionHandle.
- **GetSRDHandle:** This routine returns the SRDHandle given an ExecutionHandle.
- **GetTRDHandle:** This routine returns the TRDHandle given an TableReferenceHandle.
- **GetWPDHandle:** This routine returns the WPDHandle given an ExecutionHandle.
- **GetWRDHandle:** This routine returns the WRDHandle given an ExecutionHandle.
- **GetSQLString:** This routine returns a character string representation of the query that is associated with the request handle.

4.18.3.2 Initialization routines

- **AllocDescriptor:** This routine is used by the foreign-data wrapper to request that the SQL-server allocate a foreign-data wrapper descriptor area for use in exchanging information about values required to execute a statement or values expected to be returned from an execution of a statement.
- **AllocWrapperEnv:** This routine is used by the SQL-server to allow the foreign-data wrapper to perform any initialization steps and allocate and initialize any necessary global data structures. It has a single input parameter, a WrapperHandle, that describes the information about the foreign-data wrapper maintained by the SQL-server, and a single output parameter, a WrapperEnv handle, which is a handle to the foreign-data wrapper's newly initialized global data structures.
- **ConnectServer:** This routine is used by the SQL-server to request access to a foreign server, and allows the foreign-data wrapper associated with that foreign server to establish a connection (if necessary) and set up any required state information. ConnectServer has three input parameters: a previously allocated WrapperEnv handle, a server handle that describes the foreign server for which the SQL-server is requesting a connection, and a UserHandle that describes the user mapping maintained by the SQL-server. The routine has one output parameter, the newly allocated FSConnection handle for the foreign server.
- **GetOpts:** This routine is used by the SQL-server to request that the foreign-data wrapper return information about the capabilities and other aspects of the foreign-data wrapper, the foreign server, some foreign table at the foreign server, or some foreign column of some foreign table at the foreign server. This routine is invoked by the SQL-server, and executed by the foreign-data wrapper, whenever the SQL-server requires information about options supported by the foreign-data wrapper and the foreign server. It is thus invoked under implementation-dependent circumstances.
- **InitRequest:** This routine is used by the SQL-server to cause an SQL-statement to be prepared. The routine has two input parameters: a previously allocated FSConnection handle, and a request handle that describes the data retrieval request. The routine has two output parameters: a reply handle that describes how much of the request the foreign-data wrapper is willing to handle, and an execution handle to the state information the foreign-data wrapper needs to process the request and the data rows that will be returned.

4.18.3.3 Access routines

- **Open:** This routine is used by the SQL-server to allow the foreign-data wrapper to allocate any resources necessary to perform the operations represented by the ExecutionHandle (and described by a ReplyHandle previously returned by the InitRequest() routine). The routine has one input parameter: a previously allocated ExecutionHandle.
- **Iterate:** This routine is used by the SQL-server to iteratively retrieve data from a foreign-data wrapper. The routine has one input parameter, a previously allocated ExecutionHandle. As a result of this call, the foreign-data wrapper will associate the row with the ExecutionHandle. The SQL-server may invoke this routine until all data is returned.
- **ReOpen:** This routine may be used by the SQL-server to allow a foreign-data wrapper to re-initialize any resources necessary to re-execute the operations represented by the ExecutionHandle (and described by a ReplyHandle previously given in the InitRequest() routine). This routine allows a SQL-server to re-execute the operations associated with an ExecutionHandle multiple times, for example, if the work to be done by the foreign-data wrapper represents the inner node of a join being processed by the SQL-server. The routine has one input parameter: a previously allocated ExecutionHandle.
- **Close:** This routine is used by the SQL-server to allow a foreign-data wrapper to free any resources that had been allocated to perform the operations represented by the ExecutionHandle. The SQL-server invokes this routine after it is done processing a statement that initiated the communication with the foreign-data wrapper. The routine has one input parameter: a previously allocated ExecutionHandle.
- **GetStatistics:** This routine is used by the SQL-server to request statistics, if any, related to the SQL-statement previously sent to the foreign-data wrapper. Such statistics are entirely implementation-defined in nature. This routine is invoked by the SQL-server, and executed by the foreign-data wrapper, whenever the SQL-server requires statistics that may be provided by the foreign-data wrapper and the foreign server. It is thus invoked under implementation-dependent circumstances.
- **TransmitRequest:** This routine is used by the SQL-server to transmit a statement in the native language of the foreign server to the foreign server in pass-through mode. The foreign server analyzes the transmitted statement and returns information about that statement to the SQL-server. This information includes: Whether the statement requires one or more input values in order to be executed; and whether the statement returns one or more result values upon execution. This information is associated with the descriptors attached to the execution handle that is the output parameter of this routine. This routine has three input parameters: a previously allocated FSConnection handle, a string containing the statement, and an integer indicating the string length.

4.18.3.4 Termination routines

- **FreeReplyHandle:** This routine is used by the SQL-server to allow the foreign-data wrapper to free the resources associated with a ReplyHandle after the SQL-server has determined that it no longer needs the information encapsulated by the ReplyHandle. This routine has one input parameter, a previously allocated ReplyHandle.

- **FreeExecutionHandle:** This routine is used by the SQL-server to allow the foreign-data wrapper to free the resources associated with an ExecutionHandle after the SQL-server has determined that it no longer needs the information encapsulated by the ExecutionHandle. This routine has one input parameter, a previously allocated ExecutionHandle.
- **FreeFSConnection:** This routine is used by the SQL-server to terminate a connection to a foreign server. It allows the foreign-data wrapper to disconnect from the foreign server and to free any resources associated with the connection, such as the FSConnection handle. It has one input parameter: a previously allocated FSConnection handle.
- **FreeWrapperEnv:** This routine is used by the SQL-server to terminate communication with a foreign-data wrapper. It allows the foreign-data wrapper to free any global resources it had allocated, such as the WrapperEnv handle. The routine has one input parameter, a previously allocated WrapperEnv handle.
- **FreeDescriptor:** This routine is used by the foreign-data wrapper to request that the SQL-server deallocate a foreign-data wrapper descriptor area and to free the memory and other resources used by that descriptor.

4.18.3.5 Decomposition and pass-through modes

Depending on whether the pass-through flag in the current SQL-session context is set to *True* or *False*, the SQL-server is said to be either in *pass-through mode* or *decomposition mode*. When the SQL-server is in decomposition mode, the SQL-server analyzes the SQL-client request and invokes the `InitRequest()` routine to communicate that request to the foreign-data wrapper. When the SQL-server is in pass-through mode, the SQL-server does not analyze the SQL-client request and invokes the `TransmitRequest()` routine to communicate that request to the foreign-data wrapper.

4.18.3.6 Sequence of actions during the execution of requests involving foreign tables and foreign servers

For decomposition mode, Table 3, "Sequence of actions during foreign server request executions", shows the sequence of actions as described by the General Rules of Subclause 7.1, "<table reference>", when a foreign table is identified by the <table name> simply contained in the <table reference>.

For pass-through mode, Table 3, "Sequence of actions during foreign server request executions", shows the sequence of actions that is likely to occur when an SQL-client requests the preparation and execution of statements using dynamic SQL.

Table 3—Sequence of actions during foreign server request executions

Step	SQL-server		Flow	Foreign-data wrapper	
	Decomposition	Pass-through		Decomposition	Pass-through
1	Receives a query from SQL-client that involves data from a foreign table in a <table reference>. Determines relationship of foreign table → foreign server → foreign-data wrapper.	Receives from the SQL-client a statement to be executed in pass-through mode that involves one foreign server. Determines relationship of the foreign server to the foreign-data wrapper.			
2	Creates a WrapperHandle and associates information about the foreign-data wrapper with that handle.				
3	Invokes the AllocWrapperEnv (WrapperHandle, WrapperEnvHandle) routine to initialize the foreign data wrapper.		⇒		
4			⇐	Invokes the Get... (WrapperHandle, ...) routines in the SQL-server to retrieve information about the foreign-data wrapper that the SQL-server has stored in its Information Schema.	
NOTE 9 – This information is provided in the <foreign-data wrapper definition>.					
5	Executes the Get... (WrapperHandle, ...) routines as requested from the foreign-data wrapper.		⇒		
6				Allocates global data structures associated with the wrapper and associates them with the WrapperEnvHandle and performs any initialization required.	
7	Frees the WrapperHandle .				
NOTE 10 – If the current SQL-session context already includes a {foreign-data wrapper name : WrapperEnvHandle} pair that could be used, then steps 2 through 7 are optional.					
8	Creates a ServerHandle and associates information about the foreign server with that handle.				
9	Creates a UserHandle and associates information about the current user with that handle.				

Table 3—Sequence of actions during foreign server request executions (Cont.)

Step	SQL-server		Flow	Foreign-data wrapper	
	Decomposition	Pass-through		Decomposition	Pass-through
10	Invokes the ConnectServer (WrapperEnvHandle, ServerHandle, UserHandle, FSConnectionHandle) routine in the foreign-data wrapper to establish a connection to the foreign server that contains some of the data required to process the SQL-server client's query.		⇒	Invokes the Get... (ServerHandle, ...) routines in the SQL-Server to retrieve all information about the foreign server that the SQL-server has stored in its Information Schema.	
11			⇐		
NOTE 11 – This information is provided in the <foreign server definition>.					
12	Executes the Get... (ServerHandle, ...) routines as requested from the foreign data wrapper.		⇒	Invokes the Get... (UserHandle, ...) routines in the SQL-Server to retrieve all information about the user that the SQL-server has stored in its Information Schema.	
13			⇐		
NOTE 12 – This information is provided in the <user mapping definition>.					
14	Executes the Get... (UserHandle, ...) routines as requested from the foreign data wrapper.		⇒	Allocates global data structures associated with the foreign server and associates them with a FSConnectionHandle , establishes a connection to the foreign server, and performs any initialization required.	
15					
16	Frees the ServerHandle .				
17	Frees the UserHandle .				
NOTE 13 – If the current SQL-session context already includes a {foreign server name : FSConnectionHandle} pair that could be used, then steps 8 through 17 are optional.					

ISO/IEC 9075-9:2001 (E)
4.18 Foreign-data wrapper interface

Table 3—Sequence of actions during foreign server request executions (Cont.)

Step	SQL-server		Flow	Foreign-data wrapper	
	Decomposition	Pass-through		Decomposition	Pass-through
18	Creates a Re-questHandle and associates with it the information about the part of query that could be handled by the foreign-data wrapper.				
19	Creates as many TableReference-Handles as are needed and associates with each of them the information about a particular <table reference>.				
20	Creates a Table Reference Descriptor (TRD) for the foreign table in the <table reference>, set all the fields with details about each of the columns and associate its handle to the corresponding TableReference-Handle.				
21	Creates as many ValueExpressionHandles as are needed and associates with each of them the information about a particular <value expression> in the <select list>.				

Table 3—Sequence of actions during foreign server request executions (Cont.)

Step	SQL-server		Flow	Foreign-data wrapper	
	Decomposition	Pass-through		Decomposition	Pass-through
22	Invokes the InitRequest (FSCon- nectionHandle, RequestHandle, ReplyHandle, ExecutionHan- dle) routine in the foreign-data wrapper to find out how much of the request the foreign server can actually process.	Invokes the Trans- mitRequest (FSConnec- tionHandle, RequestString, StringLength, Ex- ecutionHandle) routine to allow the foreign-data wrap- per and the foreign server to analyze the statement.	⇒		
23			⇐	Invokes the Get... (RequestHandle, ...), GetTRDHandle (TableRef- erenceHandle), GetDescriptor (TRD), Get... (TableReference- Handle, ...), and Get... (ValueEx- pressionHandle, ...) routines in the SQL-server to examine the SQL- server's request and to determine how much of the request the foreign- data wrapper can handle.	Executes the Transmit- Request () routine.

NOTE 14 – The foreign-data wrapper could invoke the **GetSQLString (RequestHandle, StringFormat, SQLString, BufferLength, StringLength)** routine in the SQL-server to examine the SQL-server's request and to determine how much of the request the foreign-data wrapper can handle, instead of the **Get... (RequestHandle, ...), Get... (TableReferenceHandle, ...), Get... (ValueExpressionHandle, ...), GetTRDHandle (TableReferenceHandle), and GetDescriptor (TRD)** routines.

ISO/IEC 9075-9:2001 (E)
4.18 Foreign-data wrapper interface

Table 3—Sequence of actions during foreign server request executions (Cont.)

Step	SQL-server		Flow	Foreign-data wrapper	
	Decomposition	Pass-through		Decomposition	Pass-through
24	Executes the Get... (RequestHandle , ...), GetTRDHandle (TableReferenceHandle), GetDescriptor (TRD), Get... (TableReferenceHandle , ...), and Get... (ValueExpressionHandle , ...) routines as requested by the foreign-data wrapper.		⇒		
NOTE 15 – If the foreign-data wrapper invoked the GetSQLString (RequestHandle , StringFormat , SQLString , BufferLength , StringLength) routine, then this routine is executed instead of the Get... (RequestHandle , ...), Get... (TableReferenceHandle , ...), Get... (ValueExpressionHandle , ...), GetTRDHandle (TableReferenceHandle), and GetDescriptor (TRD) routines.					
25				Creates a ReplyHandle and associates with it the information about the part of query that could actually be handled by the foreign-data wrapper.	
26			⇐	Creates an ExecutionHandle and associates with it the information about the actual execution plan.	

ISO/IEC 9075-9:2001 (E)
4.18 Foreign-data wrapper interface

Table 3—Sequence of actions during foreign server request executions (Cont.)

Step	SQL-server		Flow	Foreign-data wrapper	
	Decomposition	Pass-through		Decomposition	Pass-through
27			⇐	Invokes the AllocDescriptor () routine to create a descriptor (SRD) and associates it with the Execution-Handle. Initializes the descriptor with default values wherever applicable.	Invokes the AllocDescriptor () routine to create two descriptors (WRD and SRD) to describe the columns of the result table and associates both of them with the ExecutionHandle . Initializes both the descriptors with default values wherever applicable. Sets the fields in the WRD to correspond to the result columns associated with the ExecutionHandle .
28	Executes the AllocDescriptor () routine as requested by the foreign-data wrapper.		⇒		

ISO/IEC 9075-9:2001 (E)
4.18 Foreign-data wrapper interface

Table 3—Sequence of actions during foreign server request executions (Cont.)

Step	SQL-server		Flow	Foreign-data wrapper	
	Decomposition	Pass-through		Decomposition	Pass-through
29			⇐		Invokes the AllocDescriptor () routine to create two descriptors, WPD and SPD, to describe <dynamic parameter specification>s. Associates both of them with the ExecutionHandle . Initializes both the descriptors with default values wherever applicable. Sets the fields in the WPD to correspond to the dynamic parameters associated with the ExecutionHandle .
30		Executes the AllocDescriptor () routine as requested by the foreign-data wrapper.	⇒		
31	Frees the RequestHandle .				
32	Frees each of the TableReferenceHandles .				
33	Frees each of the ValueExpressionHandles .				

Table 3—Sequence of actions during foreign server request executions (Cont.)

Step	SQL-server		Flow	Foreign-data wrapper	
	Decomposition	Pass-through		Decomposition	Pass-through
34	Invokes the Get...(ReplyHandle, ...) routines in the foreign-data wrapper to incorporate the work that a wrapper can do into the execution plan for the SQL-server client's query.		⇒		
35			⇐	Executes the Get...(ReplyHandle, ...) routines as requested by the SQL-server.	
36	Invokes the FreeReplyHandle (ReplyHandle) routine in the foreign-data wrapper to indicate that the ReplyHandle is no longer required.		⇒		
37				Frees resources associated with ReplyHandle .	
38	Invokes the GetSRDHandle (ExecutionHandle) routine to get the SRD.	Invokes the GetWRDHandle (ExecutionHandle) and GetSRDHandle (ExecutionHandle) routines to get the WRD and the SRD, respectively.	⇒		
39			⇐	Executes the GetSRDHandle (ExecutionHandle) routine as requested by the SQL-server.	Executes the GetWRDHandle (ExecutionHandle) and GetSRDHandle (ExecutionHandle) routines as requested by the SQL-server.

Table 3—Sequence of actions during foreign server request executions (Cont.)

Step	SQL-server		Flow	Foreign-data wrapper	
	Decomposition	Pass-through		Decomposition	Pass-through
40		Invokes the Get-Descriptor (WRD) routine multiple times to get all the information associated with WRD.			
41	Invokes the SetDescriptor (SRD) routine multiple times to populate appropriate fields in SRD.				
NOTE 16 – In pass-through mode, SetDescriptor (SRD) will only be invoked if results are returned by the foreign-data server.					
42		Invokes the GetW-PDHandle (ExecutionHandle) and GetSPDHandle (Execution-Handle) routines to get the WPD and the SPD, respectively.	⇒		
43			⇒		Executes the GetW-PDHandle (Execution-Handle) and GetSPDHandle (ExecutionHandle) routines as requested by the SQL-server.
44		Invokes GetDe-scriptor (WPD) multiple times to obtain the information associated with WPD.			

Table 3—Sequence of actions during foreign server request executions (Cont.)

Step	SQL-server		Flow	Foreign-data wrapper	
	Decomposition	Pass-through		Decomposition	Pass-through
45		If there are any dynamic parameters present, invokes SetDescriptor (SPD) multiple times to populate appropriate fields in SPD.			
46	Invokes the Open (ExecutionHandle) routine in the foreign-data wrapper to initiate the execution of the request in the foreign-data wrapper .		⇒		
47			⇐	Executes the Open (ExecutionHandle) routine as requested by the SQL-server.	
48	Invokes the Iterate (ExecutionHandle) routine in the foreign-data wrapper to retrieve the row.		⇒		
NOTE 17 – In pass-through mode, the Iterate () routine is only invoked if the foreign-data wrapper returns a set of rows.					
49			⇐	Performs the work needed to retrieve the next row from the foreign server and associates it with the ExecutionHandle .	
50	Repeats steps 48 through 49 until all data is retrieved.		⇔		
NOTE 18 – In pass-through mode, steps 48 through 49 are repeated only if the foreign-data wrapper returns a set of rows.					
51	Optional: If the work performed by the wrapper needs to be repeated, the SQL-server may choose to invoke ReOpen (ExecutionHandle) routine in the foreign-data wrapper to allow the wrapper to prepare to re-execute the query.		⇒		

ISO/IEC 9075-9:2001 (E)
4.18 Foreign-data wrapper interface

Table 3—Sequence of actions during foreign server request executions (Cont.)

Step	SQL-server		Flow	Foreign-data wrapper	
	Decomposition	Pass-through		Decomposition	Pass-through
NOTE 19 – The invocation of the ReOpen () routine is applicable only in decomposition mode.					
52				Optional: Executes the ReOpen (ExecutionHandle) routine as requested by the SQL-server to performs the steps necessary to reuse the resources allocated in the Open () call in order to re-execute. In the worst case, it may need to redo everything done in the Open () call. In the average case, it may only need to reset counters, cursors, etc.	
53	Completes the work necessary to answer the SQL-client's query. As a result, invokes Close (ExecutionHandle) routine in the foreign-data wrapper.	Completes the work necessary to answer the SQL-client's statement in pass-through mode. Possibly invokes the Close (ExecutionHandle) routine in the foreign-data wrapper.	⇒		
54				Executes the Close (ExecutionHandle) routine as requested by the SQL-server.	
55	Invokes FreeExecutionHandle (ExecutionHandle) routine in the foreign-data wrapper.		⇒		
56				Frees resources associated with ExecutionHandle .	

Table 3—Sequence of actions during foreign server request executions (Cont.)

Step	SQL-server		Flow	Foreign-data wrapper	
	Decomposition	Pass-through		Decomposition	Pass-through
57			⇐	Invokes the FreeDescriptor () routine with the SRDHandle as the input argument.	Invokes the FreeDescriptor () routine four times with the SRDHandle , SPDHandle , WRDHandle , and WPDHandle , respectively, as the input arguments.
58	Executes the FreeDescriptor () routine as requested by the foreign-data wrapper.		⇒		
59	Invokes the FreeFSConnection (FSConnectionHandle) routine in the foreign-data wrapper.				
60				Frees resources associated with FSConnectionHandle .	
NOTE 20 – Steps 59 and 60 are optional, if the SQL-server wants to reuse the FSConnectionHandle.					
61	Invokes FreeWrapperEnv (WrapperEnvHandle) routine in the foreign-data wrapper.		⇒		
62				Frees resources associated with WrapperEnvHandle .	
NOTE 21 – Steps 61 and 62 are optional, if the SQL-server wants to reuse the WrapperEnvHandle.					

4.18.4 Return codes

The execution of a foreign-data wrapper interface routine causes one or more conditions to be raised. The status of the execution is indicated by a code that is returned either as the result of invoking a foreign-data wrapper interface function or as the value of the ReturnCode argument resulting from invoking a foreign-data wrapper interface procedure. The values and meanings of the return codes are as follows. If more than one return code is possible, then the one appearing later in the list is the one returned.

NOTE 22 – Foreign-data wrapper functions and foreign-data wrapper procedures are defined in Sub-clause 23.1, “<foreign-data wrapper interface routine>”.

- A value of 0 (zero) indicates **Success**. The foreign-data wrapper interface routine executed successfully.
- A value of 100 indicates **No data found**. The foreign-data wrapper interface routine executed successfully but a completion condition was raised: *no data*.

ISO/IEC 9075-9:2001 (E)

4.18 Foreign-data wrapper interface

- A value of -1 indicates **Error**. The foreign-data wrapper interface routine did not execute successfully. An exception condition other than *FDW-specific condition — invalid handle* was raised.
- A value of -2 indicates **Invalid handle**. The foreign-data wrapper interface routine did not execute successfully because an exception condition was raised: *FDW-specific condition — invalid handle*.

If the foreign-data wrapper interface routine did not execute successfully, then the values of all output arguments are implementation-dependent unless explicitly defined by this part of ISO/IEC 9075.

In addition to providing the return code, for all foreign-data wrapper interface routines other than `GetDiagnostics()`, the implementation records information about completion conditions and about exception conditions raised other than *FDW-specific condition — invalid handle* in the diagnostics area associated with the resource being utilized.

The resource being utilized by a routine is the resource identified by its input handle. In case of routines that have multiple input handles, the resource being utilized is deemed to be the one identified by the handle that comes first in the parameter list, with one exception: in the case of `AllocWrapperEnv()` routine, diagnostics are returned on the output parameter, `WrapperEnvHandle`, provided an allocated FDW-environment is successfully created; otherwise, no diagnostics are returned.

4.18.5 Foreign-data wrapper diagnostics areas

Each diagnostics area consists of header fields that contain general information relating to the routine that was executed and zero or more status records containing information about individual conditions that occurred during the execution of the foreign-data wrapper interface routine. A condition that causes a status record to be generated is referred to as a status condition.

At the beginning of the execution of any foreign-data wrapper interface routine other than `GetDiagnostics()`, the diagnostics area for the resource being utilized is emptied. If the execution of such a routine does not result in the exception condition being raised: *FDW-specific condition — invalid handle*, then:

- Header information is generated in the diagnostics area.
- If the routine's return code indicates **Success**, then no status records are generated.
- If the routine's return code indicates **Success with information** or **Error**, then one or more status records are generated.
- If the routine's return code indicates **No data found**, then no status record is generated corresponding to SQLSTATE value '02000' but there may be status records generated corresponding to SQLSTATE value '02nnn', where 'nnn' is an implementation-defined subclass value.

Status records in the diagnostics area are placed in an order that is implementation-dependent except that:

- For the purpose of choosing the first status record, status records corresponding to transaction rollback have precedence over status records corresponding to other exceptions, which in turn have precedence over status records corresponding to the completion condition *no data*, which in turn have precedence over status records corresponding to the completion condition *warning*.

ISO/IEC 9075-9:2001 (E)
4.18 Foreign-data wrapper interface

- Apart from any status records corresponding to an implementation-specified *no data*, any status record corresponding to an implementation-specified condition that duplicates, in whole or in part, a condition defined in this part of ISO/IEC 9075 shall not be the first status record.

The GetDiagnostics() routine retrieves information from a diagnostics area. The SQL-server or foreign-data wrapper identifies which diagnostics area is to be accessed by providing the handle of the relevant resource as an input argument. The GetDiagnostics() routine returns a result code but does not modify the identified diagnostics area.

A foreign-data wrapper diagnostics area consists of the fields specified in Table 4, “Fields used in foreign-data wrapper diagnostics areas”.

Table 4—Fields used in foreign-data wrapper diagnostics areas

Field	Data type
Header fields	
MORE	INTEGER
NUMBER	INTEGER
RETURNCODE	SMALLINT
Implementation-defined header field	Implementation-defined
Fields in status records	
CLASS_ORIGIN	CHARACTER VARYING (L1)
MESSAGE_LENGTH	INTEGER
MESSAGE_OCTET_LENGTH	INTEGER
MESSAGE_TEXT	CHARACTER VARYING (L1)
NATIVE_CODE	INTEGER
SQLSTATE	CHARACTER (5)
SUBCLASS_ORIGIN	CHARACTER VARYING (L1)
Implementation-defined status field	Implementation-defined
Where L1 is an implementation-defined integer not less than 254.	

All diagnostics area fields in other parts of ISO/IEC 9075 that are not included in this table are not applicable to foreign-data wrapper interface routines.

4.18.6 Null pointers

If the standard programming language of the caller of a routine supports pointers, then the caller may provide a zero-valued pointer, referred to as a *null pointer*, in the following circumstances:

- In lieu of an output argument that is to receive the length of a returned character string. This indicates that the caller wishes to prohibit the return of this information.
- In lieu of other output arguments where specifically allowed by this part of ISO/IEC 9075. This indicates that the caller wishes to prohibit the return of this information.

ISO/IEC 9075-9:2001 (E)

4.18 Foreign-data wrapper interface

- In lieu of input arguments where specifically allowed by this part of ISO/IEC 9075. The semantics of such a specification depend on the context.

If the caller provides a null pointer in any other circumstances, then an exception condition is raised: *FDW-specific condition — invalid use of null pointer.*

4.18.7 Foreign-data wrapper descriptor areas

A foreign-data wrapper descriptor area provides an interface for a description of values required for the execution of an SQL-statement in decomposition mode or the execution of a statement in pass-through mode by a foreign-data wrapper and for a description of values resulting from such an execution.

Each foreign-data wrapper descriptor area comprises header fields and zero or more foreign-data wrapper item descriptor areas. The header and item descriptor area fields are specified in Table 5, “Fields in foreign-data wrapper descriptor areas”. The header fields include a COUNT field that indicates the number of item descriptor areas.

Some host languages are able to access host variables whose addresses are stored in an item descriptor field named DATA_POINTER in a foreign-data wrapper descriptor area. Such languages are called *pointer-supporting languages* and include Ada, C, Pascal, and PL/I. Languages that cannot access variables whose addresses are stored in the DATA_POINTER field of a foreign-data wrapper descriptor are called *non-pointer-supporting languages*. Such languages include COBOL, Fortran, and MUMPS.

The GetDescriptor() routine enables information to be retrieved from any foreign-data wrapper descriptor area. The SetDescriptor() routine enables information to be set by the SQL-server in the SRD and SPD, and by the foreign-data wrapper in any foreign-data wrapper descriptor area except a TRD.

The following foreign-data wrapper descriptor areas are either implicitly or explicitly allocated and deallocated:

- Table Reference Descriptor (TRD): This descriptor is allocated automatically by an SQL-server to describe a foreign table referenced in an SQL-statement, and is associated with a TableReferenceHandle created by an SQL-server. The foreign-data wrapper can obtain the handle of a TRD by invoking the GetTRDHandle() routine. It can then retrieve the information in the associated TRD descriptor by invoking the GetDescriptor() routine.
- Wrapper Row Descriptor (WRD): This descriptor is allocated by the SQL-server if a foreign-data wrapper requests its allocation. It is used to describe the result of a statement to be executed by that foreign-data wrapper in pass-through mode, and is associated with an ExecutionHandle. The foreign-data wrapper uses the SetDescriptor() routine to set information in the SRD. The SQL-server can obtain the handle to a WRD by invoking the GetWRDHandle() routine. It can then retrieve the information in that WRD by invoking the GetDescriptor() routine.
- Server Row Descriptor (SRD): This descriptor is allocated by the SQL-server if a foreign-data wrapper requests its allocation. It is used by the SQL-server to specify the type and location of data to be provided by the foreign-data wrapper. SRD is also associated with an ExecutionHandle. The SQL-server can obtain the handle to a SRD by invoking the GetSRDHandle() routine. It can then set the information in that SRD by invoking the SetDescriptor() routine.

ISO/IEC 9075-9:2001 (E)
4.18 Foreign-data wrapper interface

- **Wrapper Parameter Descriptor (WPD):** This descriptor is allocated by the SQL-server if a foreign-data wrapper requests its allocation. It is used to describe the input values required for the execution of a statement by that foreign-data wrapper in pass-through mode, and is associated with an ExecutionHandle. The foreign-data wrapper uses the SetDescriptor() routine to set information in the WPD. The SQL-server can obtain the handle to a WPD by invoking the GetWPDHandle() routine. It can then retrieve the information in that WPD by invoking the GetDescriptor() routine.
- **Server Parameter Descriptor (SPD):** This descriptor is allocated by the SQL-server if a foreign-data wrapper requests its allocation. It is used by the SQL-server to specify the type and location of input values to be provided by the SQL-server. The SPD is also associated with an ExecutionHandle. The SQL-server can obtain the handle to an SPD by invoking the GetSPDHandle() routine. It can then set the information in that SPD by invoking the SetDescriptor() routine.

Table 5—Fields in foreign-data wrapper descriptor areas

Field	Data Type
Header fields	
COUNT	SMALLINT
DYNAMIC_FUNCTION	CHARACTER VARYING(L ¹)
DYNAMIC_FUNCTION_CODE	INTEGER
KEY_TYPE	SMALLINT
TOP_LEVEL_COUNT	SMALLINT
Implementation-defined foreign-data wrapper descriptor header field	Implementation-defined
Fields in item descriptor areas	
CARDINALITY	INTEGER
CHARACTER_SET_CATALOG	CHARACTER VARYING(L ¹)
CHARACTER_SET_NAME	CHARACTER VARYING(L ¹)
CHARACTER_SET_SCHEMA	CHARACTER VARYING(L ¹)
COLLATION_CATALOG	CHARACTER VARYING(L ¹)
COLLATION_NAME	CHARACTER VARYING(L ¹)
COLLATION_SCHEMA	CHARACTER VARYING(L ¹)
CURRENT_TRANSFORM_GROUP	CHARACTER VARYING(L1 ¹)
DATA	ANY
DATA_POINTER	host variable address
DATETIME_INTERVAL_CODE	SMALLINT
DATETIME_INTERVAL_PRECISION	SMALLINT
DEGREE	INTEGER
¹ Where L is an implementation-defined integer not less than 128, and L1 is the implementation-defined maximum length for the <general value specification> CURRENT_TRANSFORM_GROUP_FOR_TYPE.	

(Continued on next page)

ISO/IEC 9075-9:2001 (E)
4.18 Foreign-data wrapper interface

Table 5—Fields in foreign-data wrapper descriptor areas (Cont.)

Field	Data Type
Fields in item descriptor areas	
INDICATOR	INTEGER
KEY_MEMBER	SMALLINT
LENGTH	INTEGER
LEVEL	INTEGER
NAME	CHARACTER VARYING(L^1)
NULLABLE	SMALLINT
OCTET_LENGTH	INTEGER
PARAMETER_MODE	SMALLINT
PARAMETER_ORDINAL_POSITION	SMALLINT
PARAMETER_SPECIFIC_CATALOG	CHARACTER VARYING(L^1)
PARAMETER_SPECIFIC_NAME	CHARACTER VARYING(L^1)
PARAMETER_SPECIFIC_SCHEMA	CHARACTER VARYING(L^1)
PRECISION	SMALLINT
RETURNED_CARDINALITY	INTEGER
RETURNED_OCTET_LENGTH	INTEGER
SCALE	SMALLINT
SCOPE_CATALOG	CHARACTER VARYING(L^1)
SCOPE_NAME	CHARACTER VARYING(L^1)
SCOPE_SCHEMA	CHARACTER VARYING(L^1)
SPECIFIC_TYPE_CATALOG	CHARACTER VARYING(L^1)
SPECIFIC_TYPE_NAME	CHARACTER VARYING(L^1)
SPECIFIC_TYPE_SCHEMA	CHARACTER VARYING(L^1)
TYPE	SMALLINT
UNNAMED	SMALLINT
USER_DEFINED_TYPE_CATALOG	CHARACTER VARYING(L^1)
USER_DEFINED_TYPE_NAME	CHARACTER VARYING(L^1)
USER_DEFINED_TYPE_SCHEMA	CHARACTER VARYING(L^1)
Implementation-defined foreign-data wrapper descriptor item field	Implementation-defined

¹Where L is an implementation-defined integer not less than 128, and $L1$ is the implementation-defined maximum length for the <general value specification> CURRENT_TRANSFORM_GROUP_FOR_TYPE.

4.19 Introduction to SQL/CLI

Insert this paragraph The BuildDataLink() routine can be used to build a datalink value. The GetDataLinkAttr() routine can be used to extract the attributes of a datalink value.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

(Blank page)

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

5 Lexical elements

5.1 <token> and <separator>

Function

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

Format

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

    | AUTHID

    | BLOCKED

    | CONTROL

    | DB | DLURLCOMPLETE | DLURLPATH | DLURLPATHONLY | DLURLSCHEME
    | DLURLSERVER | DLVALUE

    | FILE | FS

    | INTEGRITY

    | LIBRARY | LINK

    | MAPPING

    | PASSTHROUGH | PASSWORD | PERMISSION

    | RECOVERY | RESTORE

    | SELECTIVE | SERVER

    | UNLINK

    | VERSION

    | WRAPPER

    | YES

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

    | DATALINK

    | IMPORT
```

ISO/IEC 9075-9:2001 (E)
5.1 <token> and <separator>

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

5.2 Names and identifiers

Function

Specify names.

Format

```
<foreign server name> ::=  
    [ <catalog name> <period> ] <unqualified foreign server name>  
  
<foreign-data wrapper name> ::=  
    [ <catalog name> <period> ] <unqualified foreign-data wrapper name>  
  
<unqualified foreign server name> ::= <qualified identifier>  
  
<unqualified foreign-data wrapper name> ::= <qualified identifier>  
  
<option name> ::= <identifier body>
```

Syntax Rules

- 1) Insert this SR If a <foreign server name> does not contain a <catalog name>, then the <catalog name> that is specified or implicit for the SQL-client module is implicit.
- 2) Insert this SR If a <foreign-data wrapper name> does not contain a <catalog name>, then the <catalog name> that is specified or implicit for the SQL-client module is implicit.
- 3) Insert this SR In an <option name>, the number of <identifier part>s shall be less than 128.
- 4) Insert this SR The case-normal form of the <identifier body> of an <option name> is used for purposes such as and including determination of option name equivalence, representation in the Definition and Information Schemas, and representation in the diagnostics areas.
- 5) Insert this SR Two <option name>s are equivalent if the case-normal forms of their <identifier body>s, considered as the repetition of a <character string literal> that specifies a <character set specification> of SQL_IDENTIFIER and a collation *IDC* that is sensitive to case, compare equal according to the comparison rules in Subclause 8.2, "<comparison predicate>", in ISO/IEC 9075-2.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert this GR A <foreign server name> identifies a foreign server.
- 2) Insert this GR A <foreign-data wrapper name> identifies a foreign-data wrapper.

Conformance Rules

No additional Conformance Rules.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

6 Scalar expressions

6.1 <data type>

Function

Specify a data type.

Format

```

<predefined type> ::=
    !! All alternatives from ISO/IEC 9075-2
    | !! All alternatives from ISO/IEC 9075-5
    | <datalink type>

<datalink type> ::=
    DATALINK [ <datalink control definition> ]

<datalink control definition> ::=
    NO LINK CONTROL
    | FILE LINK CONTROL <datalink file control option>

<datalink file control option> ::=
    <integrity option> <read permission> <write permission>
    <recovery option> [ <unlink option> ]

<integrity option> ::=
    INTEGRITY ALL
    | INTEGRITY SELECTIVE

<read permission> ::=
    READ PERMISSION FS
    | READ PERMISSION DB

<write permission> ::=
    WRITE PERMISSION FS
    | WRITE PERMISSION BLOCKED

<recovery option> ::=
    RECOVERY NO
    | RECOVERY YES

<unlink option> ::=
    ON UNLINK RESTORE
    | ON UNLINK DELETE

```

Syntax Rules

- 1) Insert this SR DATALINK specifies the datalink type.
- 2) Insert this SR If <datalink control definition> is specified, then <data type> shall specify DATALINK.
- 3) Insert this SR If <data type> specifies DATALINK and <datalink control definition> is not specified, then NO LINK CONTROL is implicit.
- 4) Insert this SR If FILE LINK CONTROL is specified, then:
 - a) If INTEGRITY SELECTIVE is specified, then READ PERMISSION FS, WRITE PERMISSION FS, and RECOVERY NO shall be specified.
 - b) If READ PERMISSION DB is specified, then WRITE PERMISSION BLOCKED shall be specified.
 - c) If WRITE PERMISSION BLOCKED is specified, then INTEGRITY ALL and <unlink option> shall be specified.
 - d) If WRITE PERMISSION FS is specified, then READ PERMISSION FS and RECOVERY NO shall be specified and <unlink option> shall not be specified.
 - e) If RECOVERY YES is specified, then WRITE PERMISSION BLOCKED shall be specified.
 - f) If UNLINK DELETE is specified, then READ PERMISSION DB shall be specified.

NOTE 23 – Valid combinations of <datalink file control option> resulting from this Syntax Rule are shown in Table 2, “Valid datalink file control options”.
- 5) If <datalink control definition> is specified, then <data type> shall not be contained in an <SQL variable declaration>.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert this GR If <data type> is a <datalink type>, then a datalink type descriptor *DTD* is created. The link control options of *DTD* are:
 - a) The link control, according to whether NO LINK CONTROL or FILE LINK CONTROL is specified.
 - b) If FILE LINK CONTROL is specified, then:
 - i) The integrity control option, according to whether INTEGRITY ALL or INTEGRITY SELECTIVE is specified.
 - ii) The read permission option, according to whether READ PERMISSION FS or READ PERMISSION DB is specified.
 - iii) The write permission option, according to whether WRITE PERMISSION FS or WRITE PERMISSION BLOCKED is specified.

- iv) The recovery option, according to whether RECOVERY NO or RECOVERY YES is specified.
 - v) The unlink option, according to whether ON UNLINK RESTORE or ON UNLINK DELETE is specified.
- c) If NO LINK CONTROL is specified, then:
- i) The integrity control option is NONE.
 - ii) The read permission option is FS.
 - iii) The write permission option is FS.
 - iv) The recovery option is NO.
 - v) The unlink option is NONE.

Conformance Rules

- 1) Without Feature M001, "Datalinks", conforming SQL language shall not specify <datalink type>.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

6.2 <column reference>

Function

Reference a column.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

- 1) Replace AR1 If *CR* is a <column reference> whose qualifying table is a base table, a foreign table, or a viewed table and that is contained in any of:
- A <query expression> simply contained in a <cursor specification>, a <view definition> or an <insert statement>.
 - A <sort specification list> contained in a <cursor specification>.
 - A <table expression> immediately contained in a <select statement: single row>.
 - A <search condition> immediately contained in a <trigger definition>, a <delete statement: searched> or an <update statement: searched>.
 - A <select list> immediately contained in a <select statement: single row>.
 - A <value expression> simply contained in a <row value expression> immediately contained in a <set clause>.

then let *C* be the column referenced by *CR*.

Case:

- a) If <column reference> is contained in an <SQL schema statement>, then the applicable privileges of the <authorization identifier> that owns the containing schema shall include SELECT for *C*.

- b) Otherwise, the current privileges shall include SELECT on *C*.

NOTE 24 – “applicable privileges” and “current privileges” are defined in Subclause 11.1, “<privileges>”.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

6.3 <set function specification>

Function

Specify a value derived by the application of a function to an argument.

Format

No additional Format items.

Syntax Rules

- 1) Insert after SR6.1 in the TC If the <set function specification> specifies a <general set function> whose <set quantifier> is DISTINCT, then *DT* shall not be DATALINK-ordered.
- 2) Insert after SR8) If the <set function specification> specifies a <set function type> that is MAX or MIN, then *DT* shall not be DATALINK-ordered.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

6.4 <string value function>

Function

Specify a function yielding a value of type character string or bit string.

Format

```
<string value function> ::=
    !! All alternatives from ISO/IEC 9075-2
    | !! All alternatives from ISO/IEC 9075-5
    | <url complete expression>
    | <url path expression>
    | <url path only expression>
    | <url scheme expression>
    | <url server expression>

<url complete expression> ::=
    DLURLCOMPLETE <left paren> <datalink value expression> <right paren>

<url path expression> ::=
    DLURLPATH <left paren> <datalink value expression> <right paren>

<url path only expression> ::=
    DLURLPATHONLY <left paren> <datalink value expression> <right paren>

<url scheme expression> ::=
    DLURLSCHEME <left paren> <datalink value expression> <right paren>

<url server expression> ::=
    DLURLSERVER <left paren> <datalink value expression> <right paren>
```

Syntax Rules

- 1) Replace SR1 The declared type of <string value function> is the declared type of the immediately contained <character value function>, <blob value function>, <bit value function>, <url complete expression>, <url path expression>, <url path only expression>, or <url server expression>. If <string value function> is <character value function>, then the coercibility and collating sequence of <string value function> are the coercibility and collating sequence, respectively, of the simply contained <character value function>.
- 2) Insert this SR Let *DLCS* be the <character set name> of the datalink character set.
NOTE 25 – “datalink character set” is defined in Subclause 4.8, “Datalinks”.
- 3) Insert this SR Let *DVE* be the <datalink value expression>.
- 4) Insert this SR The declared type of <url complete expression> is variable-length character string with character set *DLCS* and an implementation-defined maximum length. The collating sequence and coercibility characteristics are determined as specified for monadic operators in Table 1, “Collating coercibility rules for monadic operators”, in ISO/IEC 9075-2, where the File Reference of *DVE* plays the role of the monadic operand.
NOTE 26 – The character set name, collating sequence, and coercibility characteristic of the File Reference of a <datalink value expression> are defined in Subclause 6.5, “<datalink value function>”.

- 5) Insert this SR The declared type of <url path expression> is variable-length character string with character set *DLCS* and an implementation-defined maximum length. The collating sequence and coercibility characteristics are determined as specified for monadic operators in Table 1, "Collating coercibility rules for monadic operators", in ISO/IEC 9075-2, where the File Reference of *DVE* plays the role of the monadic operand.
- 6) Insert this SR The declared type of <url path only expression> is variable-length character string with character set *DLCS* and an implementation-defined maximum length. The collating sequence and coercibility characteristics are determined as specified for monadic operators in Table 1, "Collating coercibility rules for monadic operators", in ISO/IEC 9075-2, where the File Reference of *DVE* plays the role of the monadic operand.
- 7) Insert this SR The declared type of <url scheme expression> is variable-length character string with character set *DLCS* and an implementation-defined maximum length. The collating sequence and coercibility characteristics are determined as specified for monadic operators in Table 1, "Collating coercibility rules for monadic operators", in ISO/IEC 9075-2, where the File Reference of *DVE* plays the role of the monadic operand.
- 8) Insert this SR The declared type of <url server expression> is variable-length character string with character set *DLCS* and an implementation-defined maximum length. The collating sequence and coercibility characteristics are determined as specified for monadic operators in Table 1, "Collating coercibility rules for monadic operators", in ISO/IEC 9075-2, where the File Reference of *DVE* plays the role of the monadic operand.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert this GR Let *DVE* be the <datalink value expression> simply contained in <string value function>. Let *DV* be the result of *DVE*. If *DV* is the null value, then the result of the <string value function> is the null value.
- 2) Insert this GR If <url complete expression> is specified, then
Case:
 - a) If *DV* is SQL-mediated, then the result is the File Reference of *DV* combined with an access token in an implementation-dependent manner.
 - b) Otherwise, the result is the File Reference of *DV*.
- 3) Insert this GR If <url path expression> is specified, then
Case:
 - a) If the File Reference of *DV* contains an <http url> that contains an <hpath> *HP*, then
Case:
 - i) If *DV* is SQL-mediated, then the result is *HP* combined with an access token in an implementation-dependent manner.
 - ii) Otherwise, the result is *HP*.

ISO/IEC 9075-9:2001 (E)
6.4 <string value function>

- b) If the File Reference of *DV* contains a <file url>, that contains the <fpath> *FP*, then
Case:
- i) If *DV* is SQL-mediated, then the result is *FP* combined with an access token in an implementation-dependent manner.
 - ii) Otherwise, the result is *FP*.
- c) Otherwise, a zero-length character string.
- 4) Insert this GR If <url path only expression> is specified, then the result is
Case:
- a) If the File Reference of *DV* contains an <http url> that contains an <hpath> *HP*, then *HP*, excluding any access token.
 - b) If the File Reference of *DV* contains a <file url>, then the <fpath> contained in that <file url>.
 - c) Otherwise, a zero-length character string.
- 5) Insert this GR If <url scheme expression> is specified, then the result is
Case:
- a) If the File Reference of *DV* contains an <http url>, then the <http> contained in that <http url>.
 - b) If the File Reference of *DV* contains a <file url>, then the <file> contained in that <file url>.
 - c) Otherwise, a zero-length character string.
- 6) Insert this GR If <url server expression> is specified, then the result is
Case:
- a) If the File Reference of *DV* contains an <http url>, then the <host> contained in that <http url>.
 - b) If the File Reference of *DV* contains a <file url>, then the <host> contained in that <file url>.
 - c) Otherwise, a zero-length character string.

Conformance Rules

No additional Conformance Rules.

6.5 <datalink value function>

Function

Specify a function yielding a datalink value.

Format

```
<datalink value function> ::=  
    <datalink value constructor>  
  
<datalink value constructor> ::=  
    DLVALUE <left paren> <data location> <right paren>  
  
<data location> ::= <character value expression>
```

Syntax Rules

- 1) The declared type of a <datalink value constructor> *DVC* is DATALINK.
- 2) The declared type of a <datalink value function> is the declared type of its <datalink value constructor>.
- 3) The character set name of the declared type of <data location> shall be equivalent to the character set name of the datalink character set.
NOTE 27 – “datalink character set” is defined in Subclause 4.8, “Datalinks”.
- 4) The character set name, collating sequence, and coercibility characteristic of the File Reference of the result of valuating a the <datalink value constructor> are the character set name, collating sequence, and coercibility characteristic, respectively, of the <data location>.

Access Rules

No additional Access Rules.

General Rules

- 1) Let *DLOC* be the result of evaluating <data location>.
 - a) If *DLOC* is the null value, then the result of *DVC* is the null value.
 - b) If *DLOC* conforms neither to the Format of Subclause 9.1, “URL format”, nor to an implementation-defined format, then an exception condition is raised: *data exception — invalid data specified for datalink*.
 - c) If the number of octets occupied by the implementation-defined representation of the results of *DVC* exceeds the maximum datalink length, then an exception condition is raised: *data exception — datalink value exceeds maximum length*.
NOTE 28 – The term “maximum datalink length” is defined in Subclause 4.8, “Datalinks”.

ISO/IEC 9075-9:2001 (E)
6.5 <datalink value function>

d) Otherwise, the result of *DVC* is the datalink value *DL* such that:

i) The File Reference of *DL* is *DLOC*.

Case:

1) If *DLOC* conforms to the Format of Subclause 9.1, "URL format", then

Case:

A) If *DLOC* contains an <http url>, then the <http>, <host>, and <hpath> contained in the <http url> are the *scheme* of *DL*, the *host* of *DL*, and the *path* of *DL*, respectively.

B) If *DLOC* contains a <file url>, then the <file>, <host>, and <fpath> contained in the <file url> are the *scheme* of *DL*, the *host* of *DL*, and the *path* of *DL*, respectively.

2) Otherwise, the *scheme* of *DL*, the *host* of *DL*, and the *path* of *DL* are implementation-defined.

ii) The SQL-Mediated Access Indication of *DL* is False.

2) The result of a <datalink value function> *DVF* is the result of the <datalink value constructor> contained in *DVF*.

Conformance Rules

1) Without Feature M001, "Datalinks", conforming SQL language shall not specify <datalink value function>.

6.6 <cast specification>

Function

Specify a data conversion.

Format

No additional Format items.

Syntax Rules

- 1) Insert before SR2) *TD* shall not contain a <datalink control definition>.
- 2) Augments SR6) If the <cast operand> is a <value expression>, then the valid combinations of *TD* and *SD* in a <cast specification> are given by the following table.

<data type> <i>SD</i> of <value expression>	<data type> of <i>TD</i>																		
	EN	AN	VC	FC	VB	FB	D	T	TS	YM	DT	BO	UDT	CL	BL	RT	CT	RW	DL
EN	Y	Y	Y	Y	N	N	N	N	N	M	M	N	M	Y	N	M	N	N	N
AN	Y	Y	Y	Y	N	N	N	N	N	N	N	N	M	Y	N	M	N	N	N
C	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	M	Y	N	M	N	N	N
B	N	N	Y	Y	Y	Y	N	N	N	N	N	N	M	Y	N	M	N	N	N
D	N	N	Y	Y	N	N	Y	N	Y	N	N	N	M	Y	N	M	N	N	N
T	N	N	Y	Y	N	N	N	Y	Y	N	N	N	M	Y	N	M	N	N	N
TS	N	N	Y	Y	N	N	Y	Y	Y	N	N	N	M	Y	N	M	N	N	N
YM	M	N	Y	Y	N	N	N	N	N	Y	N	N	M	Y	N	M	N	N	N
DT	M	N	Y	Y	N	N	N	N	N	N	Y	N	M	Y	N	M	N	N	N
BO	N	N	Y	Y	N	N	N	N	N	N	N	Y	M	Y	N	M	N	N	N
UDT	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	N	N	N
BL	N	N	N	N	N	N	N	N	N	N	N	N	M	N	Y	M	N	N	N
RT	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	N	N	N
CT	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	M	N	N
RW	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	M	N	N
DL	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y

Where:

EN = Exact Numeric
 AN = Approximate Numeric
 C = Character (Fixed- or Variable-length, or character large object)
 FC = Fixed-length Character
 VC = Variable-length Character
 CL = Character Large Object
 B = Bit String (Fixed- or Variable-length)
 FB = Fixed-length Bit String
 VB = Variable-length Bit String
 D = Date
 T = Time
 TS = Timestamp
 YM = Year-Month Interval
 DT = Day-Time Interval
 BO = Boolean

ISO/IEC 9075-9:2001 (E)

6.6 <cast specification>

UDT = User-Defined Type
BL = Binary Large Object
RT = Reference type
CT = Collection type
RW = Row type
DL = Datalink

Access Rules

No additional Access Rules.

General Rules

- 1) Insert this GR If *TD* and *SD* are datalink types, then *TV* is *SV*.

Conformance Rules

No additional Conformance Rules.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

6.7 <value expression>

Function

Specify a value.

Format

```
<value expression> ::=  
    !! All alternatives from ISO/IEC 9075-2  
    | !! All alternatives from ISO/IEC 9075-5  
    | <datalink value expression>
```

Syntax Rules

- 1) Replaces SR1 The declared type of a <value expression> is the declared type of the <numeric value expression>, <string value expression>, <datetime value expression>, <interval value expression>, <boolean value expression>, <user-defined type value expression>, <row value expression>, <collection value expression>, <reference value expression>, or <datalink value expression>, respectively.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

6.8 <datalink value expression>

Function

Specify a datalink value.

Format

```
<datalink value expression> ::=  
    <datalink value function>  
    | <value expression primary>
```

Syntax Rules

- 1) The declared type of <value expression primary> shall be DATALINK.

Access Rules

None.

General Rules

- 1) Case:
 - a) If <datalink value function> *DVF* is specified, then the result of the <datalink value expression> is the result of *DVF*.
 - b) If <value expression primary> *VEP* is specified, then the result of the <datalink value expression> is the result of *VEP*.

Conformance Rules

- 1) Without Feature M001, "Datalinks", conforming SQL language shall not specify <datalink value expression>.

7 Query expressions

7.1 <table reference>

Function

Reference a table.

Format

No additional Format items

Syntax Rules

No additional Syntax Rules.

Access Rules

- 1) Replace AR1 If <table reference> simply contains a <table or query name> that is a <table name>, then:
 - a) Let *T* be the table identified by the <table name> immediately contained in the <table or query name> simply contained in <table reference>.
 - b) If *T* is a base table, a foreign table, or a viewed table and the <table reference> is contained in any of:
 - A <query expression> simply contained in a <cursor specification>, a <view definition>, or an <insert statement>.
 - A <table expression> or <select list> immediately contained in a <select statement: single row>.
 - A <search condition> immediately contained in a <delete statement: searched> or an <update statement: searched>.
 - A <value expression> simply contained in a <row value expression> immediately contained in a <set clause>.

then

Case:

- i) If <table reference> is contained in an <SQL schema statement>, then the applicable privileges of the <authorization identifier> that owns the containing schema shall include SELECT on at least one column of *T*.

- ii) Otherwise, the current privileges shall include SELECT on at least one column of *T*.

NOTE 29 – “applicable privileges” and “current privileges” are defined in Subclause 11.1, “<privileges>”.

ISO/IEC 9075-9:2001 (E)

7.1 <table reference>

- c) If the <table reference> is contained in a <query expression> simply contained in a <view definition> then the applicable privileges of the <authorization identifier> that owns the view shall include SELECT for at least one column of *T*.

- d) If *TR* simply contains <only spec> and *TR* identifies a typed table, then

Case:

- i) If <table reference> is contained in a <schema definition>, then the applicable privileges of the <authorization identifier> that owns the containing schema shall include SELECT WITH HIERARCHY OPTION on at least one supertable of *T*.
- ii) Otherwise, the current privileges shall include SELECT WITH HIERARCHY OPTION on at least one supertable of *T*.

NOTE 30 – “applicable privileges” and *current privileges* are defined in Subclause 11.1, “<privileges>”.

General Rules

- 1) Replace GR2) If the <table reference> simply contains a <table name> *TN*, then

Case:

- a) If *TN* identifies a view or a base table *T*, then

Case:

- i) If ONLY is specified, then the <table reference> references the table that consists of every row in *T*, except those rows that have a subrow in a proper subtable of *T*.
- ii) Otherwise, the <table reference> references the table that consists of every row of *T*.

- b) If *TN* identifies a foreign table *FTN*, then the table referenced by the <table reference> is effectively determined as follows:

- i) Let *FSN* be the name of the foreign server included in the table descriptor of the foreign table identified by *FTN*. Let *WN* be the name of the foreign-data wrapper included in the foreign server descriptor of the foreign server identified by *FSN*. Let *WR* be the foreign-data wrapper identified by *WN*. Let *WRLN* be the name of the library identified in the foreign-data wrapper descriptor of *WR*.

- ii) Case:

- 1) If the current SQL-session context includes a {foreign-data wrapper name : WrapperEnvHandle} pair whose foreign-data wrapper name is equivalent to *WN*, then let *WEH* be the WrapperEnvHandle associated with *WN*.

- 2) Otherwise:

- A) Let *WH* be the WrapperHandle allocated for the foreign-data wrapper identified by *WN*. The resource identified by *WH* is referred to as an *allocated foreign-data wrapper description*.
- B) Let *WEH* be the WrapperEnvHandle returned by the invocation of AllocWrapperEnv() routine in the library identified by *WRLN*, with *WH* as the argument).
- C) The {*WN* : *WEH*} pair is included in the current SQL-session context.

- D) *WH* is deallocated and all its resources are freed.
- iii) Case:
- 1) If the current SQL-session context includes a {foreign server name : *FSConnectionHandle*} pair whose foreign server name is equivalent to *FSN*, then let *FSCH* be the *FSConnectionHandle* associated with *FSN*.
 - 2) Otherwise:
 - A) Let *SH* be the *ServerHandle* allocated for the foreign server identified by *FSN*. The resource identified by *SH* is referred to as an *allocated foreign server description*.
 - B) If there is a user mapping identified by the current authorization identifier, then let *UH* be the *UserHandle* allocated for that user mapping; otherwise, let *UH* be the *UserHandle* allocated for the user mapping identified by *PUBLIC*. The resource identified by *UH* is referred to as an *allocated user mapping description*.
 - C) Let *FSCH* be the *FSConnectionHandle* returned by the invocation of the *ConnectServer*() routine in the library identified by *WRLN* with *WEH*, *SH*, and *UH* as the arguments.
 - D) The {*FSN* : *FSCH*} pair is included in the current SQL-session context.
 - E) *SH* is deallocated and all its resources are freed.
 - F) *UH* is deallocated and all its resources are freed.
 - iv) Let *RQH* be the *RequestHandle* allocated for a <query specification> of the form “SELECT * FROM *FTN*”.
 - v) Let *TRH* be the *TableReferenceHandle* allocated for a <table reference> of the form “*FTN*”.
 - vi) Let *N* be the number of columns of the foreign table *FT* identified by *FTN*. Let *CN_i*, 1 (one) ≤ *i* ≤ *N*, be the column name of *i*-th column of *FT*. Let *VEH_i*, 1 (one) ≤ *i* ≤ *N* be the *i*-th *ValueExpressionHandle* allocated for a <column reference> of the form “*CN_i*”.
 - vii) A table reference descriptor *TRD* is automatically allocated. Each of the fields in *TRD* that have non-blank entries in Table 36, “Foreign-data wrapper descriptor field default values”, is set to the specified default value. All other fields in *TRD* are initially undefined.
 - viii) The General Rules of Subclause 22.4, “Implicit DESCRIBE OUTPUT USING clause”, are applied with “SELECT * FROM *FTN*” and *TRD* as *SOURCE* and *DESCRIPTOR*, respectively.
 - ix) Let *TRDH* be the *TableReferenceDescriptorHandle* allocated for *TRD*. *TRDH* is associated with *TRH*.
 - x) Let *RPH* and *EXH* be the *ReplyHandle* and *ExecutionHandle*, respectively, returned by the invocation of the *InitRequest*() routine in the library identified by *WRLN* with *FSCH* and *RQH* as input arguments.
 - xi) *TRD* is associated with *EXH*.

- xii) Let $NRTR$ be the NumberOfTableReferences that would be returned by an invocation of the `GetNumReplyTableRefs( )` routine with RPH as the ReplyHandle parameter.
- xiii) Let TRN_i , $1 \text{ (one)} \leq i \leq NRTR$, be the TableReferenceNumber that would be returned by an invocation of the `GetReplyTableRef( )` routine with RPH as the ReplyHandle parameter and i as the Index parameter.
- xiv) Let $NSLE$ be the NumberOfSelectListElements that would be returned by an invocation of the `GetNumReplySelectElems( )` routine with RPH as the ReplyHandle parameter.
- xv) Let $SELN_i$, $1 \text{ (one)} \leq i \leq NSLE$, be the SelectListElementNumber that would be returned by an invocation of the `GetReplySelectElem( )` routine with RPH as the ReplyHandle parameter and i as the Index parameter.
- xvi) The `FreeReplyHandle( )` routine in the library identified by $WRLN$ is invoked with RPH as the argument.
- xvii) Let NC be the value of the COUNT descriptor field that would be returned by invocation of the `GetDescriptor( )` routine with TRD as the DescriptorHandle parameter, 0 (zero) as the RecordNumber parameter, and the code for COUNT from Table 32, 'Codes used for foreign-data wrapper descriptor fields', as the FieldIdentifier parameter.
- xviii) Let DT_j be the effective data type of the j -th column, for $1 \text{ (one)} \leq j \leq NC$, as represented by the values of the TYPE, LENGTH, OCTET_LENGTH, PRECISION, SCALE, DATETIME_INTERVAL_CODE, DATETIME_INTERVAL_PRECISION, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME, SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME fields that would be returned by separate invocations of the `GetDescriptor( )` routine with TRD as the DescriptorHandle parameter, j as the RecordNumber parameter, and the code for the fields TYPE, LENGTH, OCTET_LENGTH, PRECISION, SCALE, DATETIME_INTERVAL_CODE, DATETIME_INTERVAL_PRECISION, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME, SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME from Table 32, 'Codes used for foreign-data wrapper descriptor fields', as the FieldIdentifier parameter.
- xix) Let SRD be the SRDHandle that would be returned by an invocation of the `GetSRDHandle( )` routine with EXH as the ExecutionHandle parameter.
- xx) Let TDT_j be the effective data type of the j -th <target specification>, for $1 \text{ (one)} \leq j \leq NC$, as represented by the values of the TYPE, LENGTH, OCTET_LENGTH, PRECISION, SCALE, DATETIME_INTERVAL_CODE, DATETIME_INTERVAL_PRECISION, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME, SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME fields that would be set by separate invocations of the `SetDescriptor( )` routine with SRD as the DescriptorHandle parameter, j as the RecordNumber parameter, and the code for the fields TYPE, LENGTH, OCTET_LENGTH, PRECISION, SCALE, DATETIME_INTERVAL_CODE, DATETIME_INTERVAL_PRECISION, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME, SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME from Table 32, 'Codes used for foreign-data wrapper descriptor fields', as the FieldIdentifier

parameter. TYPE either indicates ROW or is one of the code values in Table 17, "Codes used for application data types in SQL/CLI".

xxi) For every DT_j and TDT_j , $1 \text{ (one)} \leq j \leq NC$:

1) If DT_j is an array data type and TDT_j is not an array locator data type, then an exception condition is raised: *FDW-specific condition — invalid data type descriptors*.

2) If DT_j is a row data type, then

Case:

A) If TDT_j is not a row data type, then an exception condition is raised: *FDW-specific condition — invalid data type descriptors*.

B) If TDT_j is a row data type and DT_j and TDT_j do not conform to the Syntax Rules of Subclause 10.14, "Data type identity", in ISO/IEC 9075-2, then an exception condition is raised: *FDW-specific condition — invalid data type descriptors*.

3) If DT_j and TDT_j are predefined data types, then let HL be the standard programming language in which the invoking SQL-server is written. Let *operative data type correspondence table* be the data type correspondence table for HL as specified in Subclause 20.3, "SQL/CLI data type correspondences". Refer to the two columns of the operative data type correspondence table as the "SQL data type column" and the "host data type column".

Case:

A) If the row that contains the SQL data type corresponding to DT_j in the SQL data type column of the operative data type correspondence table contains "None" in the host data type column, and TDT_j is not a character string data type, then an exception condition is raised: *FDW-specific condition — invalid data type descriptors*.

B) Otherwise, if DT_j and TDT_j do not conform to the Syntax Rules of Subclause 10.14, "Data type identity", in ISO/IEC 9075-2, then an exception condition is raised: *FDW-specific condition — invalid data type descriptors*.

4) If DT_j is a user-defined type, then an exception condition is raised: *FDW-specific condition — invalid data type descriptors*.

xxii) VEH_i , $1 \text{ (one)} \leq i \leq N$, is deallocated and all its resources are freed.

xxiii) TRH is deallocated and all its resources are freed.

xxiv) RQH is deallocated and all its resources are freed.

xxv) The `Open( )` routine in the library identified by $WRLN$ is invoked with EXH as the argument.

xxvi) The <table reference> references the table that consists of every row returned by the repeated invocation of the `Iterate( )` routine in the library identified by $WRLN$ with $EXCH$ as the argument until the return code indicates **No data found**.

xxvii) The `Close( )` routine in the library identified by $WRLN$ is invoked with EXH as the argument.

ISO/IEC 9075-9:2001 (E)

7.1 <table reference>

- xxviii) The FreeExecutionHandle() routine in the library identified by *WRLN* is invoked with *EXH* as the argument.

Conformance Rules

No additional Conformance Rules.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

7.2 <joined table>

Function

Specify a table derived from a Cartesian product, inner or outer join, or union join.

Format

No additional Format items.

Syntax Rules

- 1) Insert after SR7)c) The declared type of neither C_1 nor C_2 shall be DATALINK-ordered.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

7.3 <group by clause>

Function

Specify a grouped table derived by the application of the <group by clause> to the result of the previously specified clause.

Format

No additional Format items.

Syntax Rules

- 1) Insert after SR5) The declared type of a grouping column shall not be DATALINK-ordered.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

7.4 <query specification>

Function

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

Format

No additional Format items.

Syntax Rules

- 1) Insert after SR5) If a <set quantifier> DISTINCT is specified, then no column of *T* shall have a declared type that is DATALINK-ordered.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

7.5 <query expression>

Function

Specify a table.

Format

No additional Format items.

Syntax Rules

- 1) Insert after SR11)c) If the set operator is UNION DISTINCT, EXCEPT DISTINCT, EXCEPT ALL, INTERSECT DISTINCT, or INTERSECT ALL, then the declared type of no column of *T1* and the declared type of no column of *T2* shall be DATALINK-ordered.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

8 Predicates

8.1 <unique predicate>

Function

Specify a test for the absence of duplicate rows.

Format

No additional Format items.

Syntax Rules

- 1) No column of the result of the <table subquery> shall have a declared type that is DATALINK-ordered.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

(Blank page)

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

9 URLs

9.1 URL format

Function

Specify the precise format of a URL within a datalink. The specification is a direct translation of the format of HTTP and FILE URLs specified in RFC 1738, as modified by RDC 1808, except that "localhost" has been omitted from the format of FILE URL. RFC 1738, RFC 1808, and RFC 2368 specify other URL schemes; URLs formatted according to those other schemes are not supported within datalinks.

Format

```

<url> ::=
    <http url>
    | <file url>

<http url> ::= <http> <colon> <solidus> <solidus> <host port> [ <solidus> <hpath> ]

<http> ::= { h | H } { t | T } { t | T } { p | P }

<host port> ::= <host> [ <colon> <port> ]

<host> ::=
    <host name>
    | <host number>

<host name> ::= [ { <domain label> <period> }... ] <top label>

<domain label> ::=
    <letter or digit>
    | <letter or digit> <label tail>

<letter or digit> ::=
    <simple Latin letter>
    | <digit>

<label tail> ::= [ { <letter or digit> | <minus sign> }... ] <letter or digit>

<top label> ::=
    <simple Latin letter>
    | <simple Latin letter> <label tail>

<host number> ::= <digits> <period> <digits> <period> <digits> <period> <digits>

<digits> ::= <digit>...

<port> ::= <digits>

<hpath> ::= <hsegment> [ { <solidus> <hsegment> }... ]

<hsegment> ::= [ <hsegment character>... ]

```

ISO/IEC 9075-9:2001 (E)

9.1 URL format

```
<hsegment character> ::=
    <uchar>
    | <colon>
    | <commercial at>
    | <ampersand>
    | <equals operator>

<uchar> ::=
    <unreserved>
    | <escape>

<unreserved> ::=
    <simple Latin letter>
    | <digit>
    | <safe>
    | <extra>

<safe> ::=
    <dollar sign>
    | <minus sign>
    | <underscore>
    | <period>
    | <plus sign>

<extra> ::=
    <exclamation point>
    | <asterisk>
    | <quote>
    | <left paren>
    | <right paren>
    | <comma>

<escape> ::= <percent> <hexit> <hexit>

<file url> ::= <file> <colon> <solidus> <solidus> <host> <solidus> <fpath>

<file> ::= { f | F } { i | I } { l | L } { e | E }

<fpath> ::= <fsegment> [ { <solidus> <fsegment> }... ]

<fsegment> ::= [ <fsegment character>... ]

<fsegment character> ::=
    <uchar>
    | <question mark>
    | <colon>
    | <commercial at>
    | <ampersand>
    | <equals operator>

<commercial at> ::= @

<dollar sign> ::= $

<exclamation point> ::= !
```

Syntax Rules

- 1) In an SQL-environment, a <url> shall reference the same file, regardless of which component in the SQL-environment is interpreting the <url>.

Access Rules

None.

General Rules

None.

Conformance Rules

None.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

(Blank page)

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

10 Data assignment rules and routine determination

10.1 Retrieval assignment

Function

Specify rules for assignments to targets that do not support null values or that support null values with indicator parameters (*e.g.*, assigning SQL-data to host parameters or host variables).

Syntax Rules

- 1) Insert this SR If the declared type of T is DATALINK, then the declared type of V shall be DATALINK.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert after GR4u) If the declared type of T is DATALINK, then the value of T is set to V .

Conformance Rules

No additional Conformance Rules.

10.2 Store assignment

Function

Specify rules for assignments where the target permits null without the use of indicator parameters or indicator variables, such as storing SQL-data or setting the value of SQL parameters.

Syntax Rules

- 1) Insert this SR If the declared type of T is DATALINK, then the declared type of V shall be DATALINK.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert after GR2)b)xxi) If the declared type of T is DATALINK, then the value of T is set to V .

Conformance Rules

No additional Conformance Rules.

10.3 Data types of results of aggregations

Function

Specify the result data type of the result of an aggregation over values of compatible data types, such as <case expression>s, <collection value expression>s, or a column in the result of a <query expression>.

Syntax Rules

- 1) Insert after SR3)h) If any data type in *DTS* is DATALINK, then each data type in *DTS* shall be DATALINK and the result data type is DATALINK.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

10.4 Type precedence list determination

Function

Determine the type precedence list of a given type.

Syntax Rules

- 1) Insert this SR If *DT* is datalink, then *TPL* is
 DATALINK *DT*

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

10.5 Determination of identical values

Function

Determine whether two instances of values are identical, that is to say, are occurrences of the same value.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert after GR2)a) If *V1* and *V2* are datalinks, then *V1* is identical to *V2* if and only if the File Reference of *V1* is identical to the File Reference of *V2* and the SQL-Mediated Access Indication of *V1* is identical to the SQL-Mediated Access Indication of *V2*.

Conformance Rules

No more Conformance Rules.

(Blank page)

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

11 Additional common elements

11.1 <privileges>

Function

Specify privileges.

Format

```
<object name> ::=
    !! All alternatives from ISO/IEC 9075-2
    | FOREIGN DATA WRAPPER <foreign-data wrapper name>
    | FOREIGN SERVER <foreign server name>
```

Syntax Rules

- 1) Augment SR3 Add <foreign server name> and <foreign-data wrapper name> to the list of <object name>s that shall require the specification of USAGE.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

11.2 <generic options>

Function

Specify a list of options identified by keywords.

Format

```
<generic options> ::=
    OPTIONS <left paren> <generic option list> <right paren>

<generic option list> ::=
    <generic option> [ { <comma> <generic option> }... ]

<generic option> ::= <option name> [ <option value> ]

<option value> ::= <character string literal>
```

Syntax Rules

- 1) Let *GOPL* be the <generic option list>.
- 2) No two <generic option>s immediately contained in *GOPL* shall have the same <option name>.
NOTE 31 – The permissible values of <option name> and <option value> are defined by the foreign-data wrapper that deals with the object for which these generic options are being specified.

Access Rules

None.

General Rules

- 1) A generic options descriptor *GOPD* is created as follows. Let *n* be the number of <generic option>s contained in <generic option list> *GOPL*. For *i* ranging from 1 (one) to *n*, the *i*-th <option name> included in *GOPD* is the *i*-th <option name> contained in *GOPL* and the *i*-th option value included in *GOPD* is the *i*-th <option value> contained in *GOPL*, if any.

Conformance Rules

None.

11.3 <alter generic options>

Function

Change the contents of a generic options descriptor

Format

```
<alter generic options> ::=  
    OPTIONS <left paren> <alter generic option list> <right paren>  
  
<alter generic option list> ::=  
    <alter generic option> [ { <comma> <alter generic option> }... ]  
  
<alter generic option> ::=  
    [ <alter operation> ] <option name> [ <option value> ]  
  
<alter operation> ::=  
    ADD  
    | SET  
    | DROP
```

Syntax Rules

- 1) Let *GOPD* be the applicable generic options descriptor. Let *AGOPL* be the <alter generic option list>.
- 2) Let *m* be the number of <alter generic option>s immediately contained in *AGOPL*. For *j* ranging from 1 (one) to *m*:
 - a) Let *AGOP_j* be the *j*-th <alter generic option> immediately contained in *AGOPL*.
 - b) For each *AGOP_j*, if <alter operation> is omitted, then *ADD* is implicit.
 - c) Let *AOP_j* and *OPN_j* be the <alter operation> and <option name>, respectively, specified or implied by *AGOP_j*.

Case:

- i) If *AOP_j* is *ADD*, then:
 - 1) <option value> shall be specified and *GOPD* shall not include an <option name> that is equivalent to *OPN_j*.
 - 2) *AGOPL* shall not immediately contain any other <alter generic option> that immediately contains an <alter operation> that specifies or implies *ADD*, and an <option name> that is equivalent to *OPN_j*.
- ii) If *AOP_j* is *SET*, then <option value> shall be specified and *GOPD* shall include an <option name> that is equivalent to *OPN_j*.
- iii) Otherwise, <option value> shall not be specified and *GOPD* shall include an <option name> that is equivalent to *OPN_j*.

ISO/IEC 9075-9:2001 (E)
11.3 <alter generic options>

Access Rules

None.

General Rules

- 1) For each <alter generic option> *AGOP* contained in *AGOL*, let *AOP* and *OPN* be the <alter operation> and <option name>, respectively, specified or implied by *AGOP* and let *OPV* be the result of <option value> contained in *AGOP*.

Case:

- a) If *AOP* is ADD, then let *n* be the number of <option name>s included in *GOPD*. *OPN* is added as the *n*+1-th <option name> included in *GOPD* and *OPV* is added as the *n*+1-th <option value> included in *GOPD*.
- b) If *AOP* is SET, then let *i* be the ordinal position of *OPN* in *GOPD*. The *i*-th <option value> in *GOPD* is replaced by *OPV*.
- c) If *AOP* is DROP, then let *i* be the ordinal position of *OPN* in *GOPD*. The *i*-th <option name> and the *i*-th <option value> are removed from *GOPD*. The ordinal positions of all <option name>s and <option value>s having an ordinal position greater than *i* are reduced by 1 (one).

Conformance Rules

None.

12 Schema definition and manipulation

12.1 <schema definition>

Function

Define a schema.

Format

```
<schema element> ::=  
    !! All alternatives from ISO/IEC 9075-2  
    | !! All alternatives from ISO/IEC 9075-5  
    | <foreign table 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.

12.2 <drop schema statement>

Function

Destroy a schema.

Format

No additional Format items.

Syntax Rules

- 1) Replace SR3 If RESTRICT is specified, then *S* shall not contain any persistent base tables, global temporary tables, created local temporary tables, foreign tables, views, domains, assertions, character sets, collations, translations, triggers, user-defined types, SQL-invoked routines, or roles, and the <schema name> of *S* shall not be generally contained in the SQL routine body of any routine descriptor.

NOTE 32 – If CASCADE is specified, then such objects will be dropped by the effective execution of the SQL schema manipulation statements specified in the General Rules of this Subclause.

Access Rules

No additional Access Rules.

General Rules

- 1) Replace GR1 Let *T* be the <table name> included in the descriptor of any base table, foreign table, or temporary table included in *S*.

Case:

- a) If *T* is a base table or temporary table, then the following <drop table statement> is effectively executed:

DROP TABLE *T* CASCADE

- b) Otherwise, the following <drop foreign table statement> is effectively executed:

DROP FOREIGN TABLE *T* CASCADE

Conformance Rules

No additional Conformance Rules.

12.3 <table definition>

Function

Define a persistent base table, a created local temporary table, or a global temporary table.

Format

```
<column option list> ::=  !! All alternatives from ISO/IEC 9075-2  
|  !! All alternatives from ISO/IEC 9075-5  
[ <datalink control definition> ]
```

Syntax Rules

- 1) Insert after SR9)d If *CO* specifies <datalink control definition> *DCS*, then let *CURITIBA* be the <column name> contained in *RCD* followed in turn by the <data type> or <domain name> contained in *RCD*, the <default clause> (if any) contained in *RCD*, every <column constraint definition> contained in *RCD*, and *DCS*. *RCD* is replaced by *CURITIBA*.
- 2) Replace SR17 A <column option list> shall immediately contain either a <scope clause> or a <default clause>, or at least one <column constraint definition>, or a <collate clause>, or a <datalink control definition>.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert after GR3 For each <column options> *CO*, if *CO* contains a <datalink control definition> *DCD*, then let *CD* be the column descriptor identified by the <column name> specified in *CO*. The link control options specified in *DCD* are included in the datalink data type descriptor that is included in *CD*.

Conformance Rules

No additional Conformance Rules.

12.4 <column definition>

Function

Define a column of a base table.

Format

No additional Format items.

Syntax Rules

- 1) Insert after SR9.1) in the TC If <data type> is a <reference type> that is DATALINK-ordered, then REFERENCES ARE CHECKED shall not be specified.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

12.5 <unique constraint definition>

Function

Specify a uniqueness constraint for a table.

Format

No additional Format items.

Syntax Rules

- 1) Insert after SR1 The declared type of no column identified by any <column name> in the <unique column list> shall be DATALINK-ordered.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

12.6 <drop column definition>

Function

Destroy a column of a base table.

Format

No additional Format items

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert after GR4) Let *CC* be a component of *C*. If the declared type of *CC* is DATALINK or some distinct type with a source data type of DATALINK, whose descriptor includes a <datalink control definition> that specifies FILE LINK CONTROL, then, for every non-null value *DLV* in *CC*, let *EF* be the external file referenced by *DLV*. If *EF* is linked, then *EF* is unlinked.

NOTE 33 – The effect of unlinking depends on the unlink control option, RESTORE or DELETE, included in the descriptor of *CC*, as specified in Subclause 4.10, “Columns, fields, and attributes”.

Conformance Rules

No additional Conformance Rules.

12.7 <domain definition>

Function

Define a domain.

Format

No additional Format items.

Syntax Rules

- 1) Insert before SR1 <data type> shall not contain a <datalink control definition>.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

12.8 <SQL-invoked routine>

Function

Define an SQL-invoked routine.

Format

No additional Format items.

Syntax Rules

- 1) Insert before SR1 Neither <returns type> nor <parameter type> shall contain a <datalink control definition>.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

12.9 <user-defined type definition>

Function

Define a user-defined type.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

- 1) Replace GR 1)c)) If *SDT* is not a large object type or a datalink type, then the following SQL-statement is executed without further Access Rule checking:

```
CREATE ORDERING FOR UDTN
ORDER FULL BY
  MAP WITH METHOD FNSDT()
FOR UDTN
```

Conformance Rules

No additional Conformance Rules.

12.10 <user-defined cast definition>

Function

Define a user-defined cast.

Format

No additional Format items.

Syntax Rules

- 1) Insert before SR1 Neither <source data type> nor <target data type> shall contain a <datalink control definition>.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

12.11 <user-defined ordering definition>

Function

Define a user-defined ordering for a user-defined type.

Format

No additional Format items.

Syntax Rules

- 1) Insert after SR 6)a)i.2) in the TC The declared type of no attribute of *UDT* shall be DATALINK-ordered.
- 2) Insert after SR 8)c) in the TC The result data type of *F* shall not be a datalink type.

Access Rules

No additional Syntax Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

12.12 <foreign table definition>

Function

Define a foreign table.

Format

```
<foreign table definition> ::=  
    CREATE FOREIGN TABLE <table name>  
        [ <left paren> <basic column definition list> <right paren> ]  
        SERVER <foreign server name> [ <table generic options> ]  
  
<table generic options> ::= <generic options>  
  
<basic column definition list> ::=  
    <basic column definition> [ <comma> <basic column definition>... ]  
  
<basic column definition> ::=  
    <column name> <data type> [ <column generic options> ]  
  
<column generic options> ::= <generic options>
```

Syntax Rules

- 1) If <foreign table definition> is contained in a <schema definition>, and if the <table name> contains a <schema name>, then that <schema name> shall be equivalent to the specified or implicit <schema name> of the containing <schema definition>.
- 2) Let *TN* be the <table name>. Let *S* be the schema identified by the explicit or implicit schema name of *TN*. *S* shall not include a table descriptor whose table name is equivalent to *TN*.
- 3) If <basic column definition list> is specified, then let *n* be the cardinality of the <basic column definition list>. For all *i*, $1 \text{ (one)} \leq i \leq n$:
 - a) For all *j*, $1 \text{ (one)} \leq j \leq n$, if the <column name> contained in the *i*-th <basic column definition> is equivalent to the <column name> contained in the *j*-th <basic column definition>, then $i=j$.
 - b) If the <data type> contained in the *i*-th <basic column definition> specifies a <character string type> and does not specify a <character set specification>, then the <character set specification> specified or implicit in the <schema character set specification> of the <schema definition> that created the schema *S* is implicit.
- 4) Let *FSN* be the <foreign server name>.
- 5) The catalog identified by the explicit or implicit catalog name of *FSN* shall include a foreign server descriptor whose foreign server name is equivalent to *FSN*.
- 6) If the <foreign table definition> is contained in a <schema definition> *SD*, then let *A* be the explicit or implicit <authorization identifier> of *SD*. Otherwise, let *A* be the <authorization identifier> that owns the schema identified by the implicit or explicit <schema name> of *TN*.

Access Rules

- 1) If <foreign table definition> is contained in an SQL-client module, then the enabled authorization identifiers shall include *A*.
- 2) If <basic column definition list> is specified, then for each <data type> *DT* simply contained in <basic column definition list>, if *DT* is one of the following:
 - a) A user-defined type *U*.
 - b) A reference type whose referenced type is a user-defined type *U*.
 - c) An array type whose element type is a user-defined type *U*.
 - d) An array type whose element type is a reference type whose referenced type is a user-defined type *U*.
 - e) A row type with a field that has a declared type that is:
 - i) A user-defined type *U*.
 - ii) A reference type whose referenced type is a user-defined type *U*.
 - iii) An array type whose element type is a user-defined type *U*.
 - iv) An array type whose element type is a reference type whose referenced type is a user-defined type *U*.then the applicable privileges of *A* shall include USAGE on *U*.
- 3) The applicable privileges shall include the USAGE privilege on the foreign-server identified by <foreign-server name>.
- 4) Additional privileges, if any, necessary to execute <foreign table definition> are implementation-defined.

General Rules

- 1) A foreign table descriptor *FTD* is created in *S*. *FTD* includes:
 - a) The table name *TN*.
 - b) The foreign server name *FSN*.
 - c) If <table generic options> *TGO* is specified, then the generic options descriptor created by *TGO*; otherwise, an empty generic options descriptor.
 - d) Case:
 - i) If <basic column definition list> *BCDL* is specified, then *n* column descriptors. For each <basic column definition> *BCD_i*, $1 \text{ (one)} \leq i \leq n$, the corresponding *i*-th column descriptor includes:
 - 1) The <column name> contained in *BCD_i*.
 - 2) An indication that the column name is not an implementation-dependent name.
 - 3) The data type descriptor of the <data type> *DT* simply contained in *BCD_i*.

ISO/IEC 9075-9:2001 (E)

12.12 <foreign table definition>

- 4) The ordinal position, i .
 - 5) The implementation-defined nullability characteristic.
 - 6) The implementation-defined <default option>.
 - 7) If <column generic options> CGO is specified, then the generic options descriptor created by CGO ; otherwise, an empty generic options descriptor.
- ii) Otherwise, the column descriptors included in FTD are implementation-defined.
- e) An indication that the table is not referenceable.
 - f) An empty list of direct supertable names.
 - g) An empty list of direct subtable names.
 - h) An indication that the table is not insertable-into.
 - i) An indication that the table is not updatable.

NOTE 34 – This part of ISO/IEC 9075 currently restricts foreign tables such that they are neither insertable-into nor updatable. Future versions of this part of ISO/IEC 9075 may relax these restrictions.

- 2) Let T be the table described by FTD . Let m be the number of column descriptors CD_i , $1 \text{ (one)} \leq i \leq m$, included in FTD . The row type of T consists of m fields F_i such that, for all i , $1 \text{ (one)} \leq i \leq m$, the field name of F_i is the column name included in CD_i and the declared type of F_i is the data type described by the data type descriptor included in CD_i .
- 3) A set of privilege descriptors is created that define the privilege SELECT on T and SELECT for every column of T . These privileges are grantable. The grantor for each of these privilege descriptors is set to the special grantor value “_SYSTEM”. The grantee is A.

Conformance Rules

- 1) Without Feature M004, “Foreign data support”, conforming SQL language shall not specify <foreign table definition>.

12.13 <alter foreign table statement>

Function

Change the definition of a foreign table.

Format

```
<alter foreign table statement> ::=  
    ALTER FOREIGN TABLE <table name> <alter foreign table action>
```

```
<alter foreign table action> ::=  
    <add basic column definition>  
    | <alter basic column definition>  
    | <drop basic column definition>  
    | <alter generic options>
```

Syntax Rules

- 1) The schema *S* identified by the explicit or implicit schema name of the <table name> *TN* shall include a foreign table descriptor *FTD* whose table name is equivalent to *TN*. *FTD* is the descriptor of the foreign table being altered.
- 2) Let *A* be the <authorization identifier> that owns the schema identified by the <schema name> of the table identified by *TN*.
- 3) If <alter generic options> *AGO* is specified, then the Syntax Rules of Subclause 11.3, "<alter generic options>", are applied to *AGO* with the generic options descriptor included in *FTD* as the applicable generic options descriptor.

Access Rules

- 1) The enabled authorization identifiers shall include *A*.

General Rules

- 1) *FTD* is modified as specified by <alter foreign table action>.
- 2) If <alter generic options> *AGO* is specified, then the General Rules of Subclause 11.3, "<alter generic options>", are applied to *AGO* with the generic options descriptor included in *FTD* as the applicable generic options descriptor.
- 3) If <alter generic options> is specified, any effect on *FTD*, apart from that on its generic options descriptor, is implementation-defined.
- 4) Let *T* be the table described by *FTD*. Let *m* be the number of column descriptors *CD_i*, 1 (one) ≤ *i* ≤ *m*, included in *FTD*. The row type of *T* consists of *m* fields *F_i* such that, for all *i*, 1 (one) ≤ *i* ≤ *m*, the field name of *F_i* is the column name included in *CD_i* and the declared type of *F_i* is the data type described by the data type descriptor included in *CD_i*.

Conformance Rules

- 1) Without Feature M004, “Foreign data support”, conforming SQL language shall not specify <alter foreign table statement>.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

12.14 <add basic column definition>

Function

Add a column to a foreign table.

Format

```
<add basic column definition> ::=  
    ADD [ COLUMN ] <basic column definition>
```

Syntax Rules

- 1) Let *FTD* be the descriptor of the foreign table being altered.
- 2) *FTD* shall not include a column descriptor whose column name is equivalent to the <column name> *CN* specified in the <basic column definition> *BCD*.
- 3) Let *A* be the <authorization identifier> that owns the schema that includes *FTD*.

Access Rules

- 1) Let *DT* be the <data type> simply contained in *BCD*. If *DT* is one of the following:
 - a) A user-defined type *U*.
 - b) A reference type whose referenced type is a user-defined type *U*.
 - c) An array type whose element type is a user-defined type *U*.
 - d) An array type whose element type is a reference type whose referenced type is a user-defined type *U*.
 - e) A row type with a field that has a declared type that is:
 - i) A user-defined type *U*.
 - ii) A reference type whose referenced type is a user-defined type *U*.
 - iii) An array type whose element type is a user-defined type *U*.
 - iv) An array type whose element type is a reference type whose referenced type is a user-defined type *U*.

then the applicable privileges of *A* shall include USAGE on *U*.

General Rules

- 1) Let *n* be the number of column descriptors included in *FTD*.
- 2) The degree of the table being altered by the containing <alter foreign table statement> is increased by 1 (one).
- 3) A column descriptor *CD* is added to *FTD*. *CD* includes:
 - a) The <column name> *CN* contained in *BCD*.

ISO/IEC 9075-9:2001 (E)

12.14 <add basic column definition>

- b) An indication that the column name is not an implementation-dependent name.
 - c) The data type descriptor of the <data type> *DT* simply contained in *BCD*.
 - d) The ordinal position, $n+1$.
 - e) The implementation-defined nullability characteristic.
 - f) The implementation-defined <default option>.
 - g) If <column generic options> *CGO* is specified, then the generic options descriptor created by *CGO*; otherwise, an empty generic options descriptor.
- 4) For every table privilege descriptor that specifies *T* and a privilege of SELECT, a new column privilege descriptor is created that specifies *T*, the same action, grantor, and grantee, and the same grantability, and specifies *CN*.

Conformance Rules

None.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

12.15 <alter basic column definition>

Function

Change the definition of a column of a foreign table.

Format

```
<alter basic column definition> ::=  
    ALTER [ COLUMN ] <column name> <alter basic column action>  
  
<alter basic column action> ::=  
    <alter generic options>
```

Syntax Rules

- 1) Let *FTD* be the descriptor of the foreign table identified in the containing <alter table statement>.
- 2) *FTD* shall include a column descriptor *CD* whose column name is equivalent to <column name>.
- 3) Let *C* be the column described by *CD*.

Access Rules

None.

General Rules

- 1) *CD* is modified as specified by <alter basic column action>.
- 2) If <alter generic options> is specified, any effect on *CD*, apart from that on its generic options descriptor, is implementation-defined.

Conformance Rules

None.

12.16 <drop basic column definition>

Function

Destroy a column of a foreign table.

Format

```
<drop basic column definition> ::=  
    DROP [ COLUMN ] <column name> <drop behavior>
```

Syntax Rules

- 1) Let *FTD* be the descriptor of the foreign table being altered.
- 2) *FTD* shall include a column descriptor *CD* whose column name is equivalent to the <column name> *CN*.
- 3) *FTD* shall include at least two column descriptors.
- 4) Let *C* be the column described by *CD*.
- 5) If RESTRICT is specified, then *C* shall not be referenced in any of the following:
 - a) The <query expression> of any view descriptor.
 - b) The <search condition> of any constraint descriptor.
 - c) The <SQL routine body> of any routine descriptor.
 - d) Either an explicit trigger column list or a triggered action column set of any trigger descriptor.

NOTE 35 – A <drop basic column definition> that does not specify CASCADE will fail if there are any references to that column resulting from the use of CORRESPONDING, NATURAL, or SELECT * (except where contained in an exists predicate).

NOTE 36 – If CASCADE is specified, then any such dependent object will be dropped by the execution of the <revoke statement> specified in the General Rules of this Subclause.

NOTE 37 – *CN* may be contained in an implicit trigger column list of a trigger descriptor.

Access Rules

None.

General Rules

- 1) Let *TR* be the trigger name of any trigger descriptor having an explicit trigger column list or a triggered action column set that contains *CN*. The following <drop trigger statement> is effectively executed without further Access Rule checking:

```
DROP TRIGGER TR
```

12.16 <drop basic column definition>

- 2) Let *A* be the <authorization identifier> that owns *T*. The following <revoke statement> is effectively executed with a current authorization identifier of “_SYSTEM” and without further Access Rule checking:

REVOKE SELECT(*CN*) ON TABLE *TN* FROM *A* CASCADE

- 3) Let *R* be any SQL-invoked routine whose routine descriptor contains *CN* in the <SQL routine body>. Let *SN* be the <specific name> of *R*. The following <drop routine statement> is effectively executed for every *R* without further Access Rule checking:

DROP SPECIFIC ROUTINE *SN* CASCADE

- 4) *CD* is destroyed and the ordinal position of every column descriptor following *CD* in *FTD* is reduced by 1 (one).
- 5) The degree of the table described by *FTD* is reduced by 1 (one).

Conformance Rules

None.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

12.17 <drop foreign table statement>

Function

Destroy a foreign table.

Format

```
<drop foreign table statement> ::=  
    DROP FOREIGN TABLE <table name> <drop behavior>
```

Syntax Rules

- 1) The schema *S* identified by the explicit or implicit schema name of the <table name> *TN* shall include a foreign table descriptor *FTD* whose table name is equivalent to *TN*. Let *T* be the table described by *FTD*.
- 2) If RESTRICT is specified, then *T* shall not be referenced in any of the following:
 - a) The <query expression> of any view descriptor.
 - b) The <SQL routine body> of any SQL-invoked routine descriptor.
 - c) The trigger action of any trigger descriptor.

NOTE 38 – If CASCADE is specified, then such referenced objects will be dropped by the execution of the <revoke statement> specified in the General Rules of this Subclause.

Access Rules

- 1) The enabled authorization identifiers shall include the <authorization identifier> that owns *S*.

General Rules

- 1) Every row of *T* is effectively deleted at the end of the SQL-statement, prior to the checking of any integrity constraints.

NOTE 39 – This deletion creates neither a new trigger execution context nor the definition of a new state change in the current trigger execution context.
- 2) The following <revoke statement> is effectively executed with a current authorization identifier of “_SYSTEM” and without further Access Rule checking:

 REVOKE ALL PRIVILEGES ON *TN* FROM A CASCADE
- 3) Let *R* be any SQL-invoked routine whose routine descriptor contains *TN* in the <SQL routine body>. 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
- 4) *FTD* is destroyed.

Conformance Rules

- 1) Without Feature M004, “Foreign data support”, conforming SQL language shall not specify <drop foreign table statement>.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

(Blank page)

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

13 Catalog manipulation

13.1 <foreign server definition>

Function

Define a foreign server.

Format

```
<foreign server definition> ::=
    CREATE SERVER <foreign server name>
    [ TYPE <server type> ]
    [ VERSION <server version> ]
    [ AUTHORIZATION <authorization identifier> ]
    FOREIGN DATA WRAPPER <foreign-data wrapper name>
    [ <generic options> ]
```

<server type> ::= !! See the Syntax Rules

<server version> ::= !! See the Syntax Rules

Syntax Rules

- 1) Let *FSN* be the <foreign server name>. Let *C1* be the catalog identified by the explicit or implicit catalog name of *FSN*. *C1* shall not include a foreign server descriptor whose foreign server name is equivalent to *FSN*.
- 2) Let *WN* be the <foreign-data wrapper name>. Let *C2* be the catalog identified by the explicit or implicit catalog name of *WN*. *C2* shall include a foreign-data wrapper descriptor whose foreign-data wrapper name is *WN*.
- 3) If AUTHORIZATION <authorization identifier> is not specified, then
Case:
 - a) If the <foreign server definition> is contained in an SQL-client module that has a <module authorization identifier> specified, then an <authorization identifier> equal to that <module authorization identifier> is implicit for the <foreign server definition>.
 - b) Otherwise, an <authorization identifier> equal to the SQL-session user identifier is implicit.
- 4) The permissible Format and values for <server type> and <server version> are implementation-defined.

ISO/IEC 9075-9:2001 (E)
13.1 <foreign server definition>

Access Rules

- 1) The applicable privileges shall include the USAGE privilege on the foreign-data wrapper identified by <foreign-data wrapper name>.
- 2) Additional privileges, if any, necessary to execute <foreign server definition> are implementation-defined.

General Rules

- 1) A foreign server descriptor *FSD* is created. *FSD* includes:
 - a) The foreign server name *FSN*.
 - b) The foreign-data wrapper name *WN*.
 - c) The <server type>, if specified.
 - d) The <server version>, if specified.
 - e) The implicit or explicit <authorization identifier>.
 - f) If <generic options> *GO* is specified, then the generic options descriptor created by *GO*; otherwise, an empty generic options descriptor.
- 2) A privilege descriptor is created that defines the USAGE privilege on this foreign server to the <authorization identifier> of the <foreign server definition>. The grantor of the privilege descriptor is set to the special grantor value "SYSTEM". This privilege is grantable.

Conformance Rules

- 1) Without Feature M004, "Foreign data support", conforming SQL language shall not specify <foreign server definition>.

13.2 <alter foreign server statement>

Function

Change the definition of a foreign server.

Format

```
<alter foreign server statement> ::=  
    ALTER SERVER <foreign server name>  
    [ <new version> ]  
    [ <alter generic options> ]  
  
<new version> ::= VERSION <server version>
```

Syntax Rules

- 1) If <new version> is not specified, then <alter generic options> shall be specified.
- 2) If <alter generic options> is not specified, then <new version> shall be specified.
- 3) Let *FSN* be the <foreign server name>. Let *C* be the catalog identified by the explicit or implicit catalog name of *FSN*. *C* shall include a foreign server descriptor *FSD* whose foreign server name is equivalent to *FSN*.
- 4) If <alter generic options> *AGO* is specified, then the Syntax Rules of Subclause 11.3, "<alter generic options>", are applied to *AGO* with the generic options descriptor included in *FSD* as the applicable generic options descriptor.
- 5) Let *A* be the authorization identifier that owns the foreign server descriptor identified by *FSN*.

Access Rules

- 1) The enabled authorization identifiers shall include *A*.

General Rules

- 1) If <new version> *NV* is specified, then the <server version> included in *FSD* is the <server version> specified in *NV*.
- 2) If <alter generic options> *AGO* is specified, then the General Rules of Subclause 11.3, "<alter generic options>", are applied to *AGO* with the generic options descriptor included in *FSD* as the applicable generic options descriptor.

Conformance Rules

- 1) Without Feature M004, "Foreign data support", conforming SQL language shall not specify <alter foreign server statement>.

13.3 <drop foreign server statement>

Function

Destroy a foreign server descriptor.

Format

```
<drop foreign server statement> ::=  
    DROP SERVER <foreign server name> <drop behavior>
```

Syntax Rules

- 1) Let *FSN* be the <foreign server name>. Let *C* be the catalog identified by the explicit or implicit catalog name of *FSN*. *C* shall include a foreign server descriptor *S* whose foreign server name is equivalent to *FSN*.
- 2) If <drop behavior> specifies RESTRICT, then *S* shall not be referenced by any foreign table descriptor or by any user mapping descriptor.
- 3) Let *A* be the authorization identifier that owns the foreign server descriptor identified by *FSN*.

Access Rules

- 1) The enabled authorization identifiers shall include *A*.

General Rules

- 1) Let *UM* be any user mapping descriptor that includes a foreign server name that is equivalent to *SN*. Let *AI* be the authorization identifier included in *UM*. The following <drop user mapping statement> is effectively executed without further Access Rule checking:

```
DROP USER MAPPING FOR AI SERVER SN
```

- 2) The following <revoke statement> is effectively executed with a current authorization identifier of “_SYSTEM” and without further Access Rule checking:

```
REVOKE ALL PRIVILEGES ON SN FROM A CASCADE
```

- 3) The descriptor *S* is destroyed.

Conformance Rules

- 1) Without Feature M004, “Foreign data support”, conforming SQL language shall not specify <drop foreign server statement>.

13.4 <foreign-data wrapper definition>

Function

Define a foreign-data wrapper

Format

```
<foreign-data wrapper definition> ::=  
    CREATE FOREIGN DATA WRAPPER <foreign-data wrapper name>  
    [ AUTHORIZATION <authorization identifier> ]  
    [ <library name specification> ]  
    <language clause>  
    [ <generic options> ]
```

```
<library name specification> ::= LIBRARY <library name>
```

```
<library name> ::= <character string literal>
```

Syntax Rules

- 1) Let *WN* be the <foreign-data wrapper name>. Let *C* be the catalog identified by the explicit or implicit catalog name of *WN*. *C* shall not include a foreign-data wrapper descriptor whose foreign-data wrapper name is equivalent to *WN*.
- 2) If <library name specification> is not specified, then a <library name specification> with an implementation-dependent <library name> is implicit.
- 3) If AUTHORIZATION <authorization identifier> is not specified, then
Case:
 - a) If the <foreign-data wrapper definition> is contained in an SQL-client module that has a <module authorization identifier> specified, then an <authorization identifier> equal to that <module authorization identifier> is implicit for the <foreign-data wrapper definition>.
 - b) Otherwise, an <authorization identifier> equal to the SQL-session user identifier is implicit.

Access Rules

- 1) The privileges necessary to execute <foreign-data wrapper definition> are implementation-defined.

General Rules

- 1) A foreign-data wrapper descriptor *WD* is created. *WD* includes:
 - a) The foreign-data wrapper name *WN*.
 - b) The implicit or explicit <authorization identifier>.
 - c) The implicit or explicit <library name>.
 - d) The name of the language specified in <language clause>.

ISO/IEC 9075-9:2001 (E)

13.4 <foreign-data wrapper definition>

- e) If <generic options> GO is specified, then the generic options descriptor created by GO; otherwise, an empty generic options descriptor.
- 2) A privilege descriptor is created that defines the USAGE privilege on this foreign-data wrapper to the <authorization identifier> of the <foreign-data wrapper definition>. The grantor of the privilege descriptor is set to the special grantor value “_SYSTEM”. This privilege is grantable.

Conformance Rules

- 1) Without Feature M004, “Foreign data support”, conforming SQL language shall not specify <foreign-data wrapper definition>.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

13.5 <alter foreign-data wrapper statement>

Function

Change the definition of a foreign-data wrapper.

Format

```
<alter foreign-data wrapper statement> ::=
    ALTER FOREIGN DATA WRAPPER <foreign-data wrapper name>
    [ <library name specification> ]
    [ <alter generic options> ]
```

Syntax Rules

- 1) Let *WN* be the <foreign-data wrapper name>. Let *C* be the catalog identified by the explicit or implicit catalog name of *FSN*. *C* shall include a foreign-data wrapper descriptor *W* whose foreign-data wrapper name is equivalent to *WN*.
- 2) If <library name specification> is not specified, then <alter generic options> shall be specified.
- 3) If <alter generic options> is not specified, then <library name specification> shall be specified.
- 4) If <alter generic options> *AGO* is specified, then the Syntax Rules of Subclause 11.3, "<alter generic options>", are applied to *AGO* with the generic options descriptor included in *W* as the applicable generic options descriptor.
- 5) Let *A* be the authorization identifier that owns the foreign-data wrapper descriptor identified by *WN*.

Access Rules

- 1) The enabled authorization identifiers shall include *A*.

General Rules

- 1) If <library name specification> is specified, then the <library name> is included in *W*, replacing any existing <library name>.
- 2) If <alter generic options> *AGO* is specified, then the General Rules of Subclause 11.3, "<alter generic options>", are applied to *AGO* with the generic options descriptor included in *W* as the applicable generic options descriptor.

Conformance Rules

- 1) Without Feature M004, "Foreign data support", conforming SQL language shall not specify <alter foreign-data wrapper statement>.

13.6 <drop foreign-data wrapper statement>

Function

Destroy a foreign-data wrapper.

Format

```
<drop foreign-data wrapper statement> ::=  
    DROP FOREIGN DATA WRAPPER <foreign-data wrapper name> <drop behavior>
```

Syntax Rules

- 1) Let *WN* be the <foreign-data wrapper name>. Let *C* be the catalog identified by the explicit or implicit catalog name of *FSN*. *C* shall include a foreign-data wrapper descriptor *W* whose foreign-data wrapper name is equivalent to *WN*.
- 2) If <drop behavior> specifies RESTRICT, then *W* shall not be referenced by the foreign server name included in any foreign server descriptor.
- 3) Let *A* be the authorization identifier that owns the foreign-data wrapper descriptor identified by *WN*.

Access Rules

- 1) The enabled authorization identifiers shall include *A*.

General Rules

- 1) The following <revoke statement> is effectively executed with a current authorization identifier of “_SYSTEM” and without further Access Rule checking:

```
REVOKE ALL PRIVILEGES ON WN FROM A CASCADE
```

- 2) The descriptor of *W* is destroyed.

Conformance Rules

- 1) Without Feature M004, “Foreign data support”, conforming SQL language shall not specify <drop foreign-data wrapper statement>.

13.7 <import foreign schema statement>

Function

Acquire information about some or all foreign tables associated with a schema managed by a foreign server.

Format

```
<import foreign schema statement> ::=
    IMPORT FOREIGN SCHEMA <foreign schema name>
    [ <import qualifications> ]
    FROM SERVER <foreign server name>
    INTO <local schema name>
```

```
<import qualifications> ::=
    LIMIT TO <left paren> <table name list> <right paren>
    | EXCEPT <left paren> <table name list> <right paren>
```

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

```
<foreign schema name> ::= <schema name>
```

```
<local schema name> ::= <schema name>
```

Syntax Rules

- 1) Let *FSN* be <foreign schema name>.
- 2) For every <table name> *TN* contained in <table name list>:
 - a) If *TN* specifies a <schema name> *SN*, then *SN* shall be equivalent to *FSN*.
 - b) Otherwise, a <schema name> that is equivalent to *FSN* is implicit.
- 3) There shall be an SQL-schema identified by <local schema name> *LSN*.

Access Rules

None.

General Rules

- 1) If the foreign server *FSVR* identified by <foreign server name> *FSVRN* does not maintain information analogous to schemas, or if the foreign-data wrapper by which the SQL-server accesses *FSVR* does not support schema importation, then an exception condition is raised: *FDW-specific condition — no schemas*.
- 2) If *FSVR* does not maintain information about a schema *FS* whose name is equivalent to *FSN*, then an exception condition is raised: *FDW-specific condition — schema not found*.

ISO/IEC 9075-9:2001 (E)

13.7 <import foreign schema statement>

3) Case:

- a) If <import qualifications> is not specified, then let *ITNL* be a <table name list> that contains the <table name> of every table associated with *FS*.
 - b) If <import qualifications> specifies LIMIT TO, then let *ITNL* be the explicit <table name list>.
 - c) If <import qualifications> specifies EXCEPT, then let *ITNL* be a <table name list> that contains the <table name> of every table associated with *FS* except the tables whose names are specified in the explicit <table name list>.
- 4) For every <table name> *FTN* contained in *ITNL*, if *FS* does not include a descriptor of a table whose <table name> is equivalent to *FTN*, then an exception condition is raised: *FDW-specific condition — table not found*.
- 5) For every <table name> *FTN* contained in *ITNL*:
- a) Let *n* be the number of columns whose descriptors are included in the table identified by *FTN*.
 - b) Let BCD_i , $1 \text{ (one)} \leq i \leq n$, be a <basic column definition> that contains a <column name> equivalent to the name of the *i*-th column *COL* of the table identified by *FTN*, a <data type> corresponding to the data type of *COL*, and implementation-defined <column generic options>.
 - c) Let *FTD* be a <foreign table definition> that contains *FTN*, every BCD_i , $1 \text{ (one)} \leq i \leq n$, in sequence, separated by <comma>s, *FSVRN*, and implementation-defined <table generic options>.
 - d) *FTD* is effectively executed.

Conformance Rules

- 1) Without Feature M004, “Foreign data support”, and Feature M005, “Foreign schema support”, conforming SQL language shall not specify <import foreign schema statement>.

14 Access control

14.1 <revoke statement>

Function

Destroy privileges and role authorizations.

Format

No additional Format items.

Syntax Rules

- 1) Insert after SR21 Let *T* be any table descriptor included in *S1*. *T* is said to be *abandoned* if the revoke destruction action would result in *A1* no longer having USAGE privilege on the foreign server associated with the table described by *T*.
- 2) Insert after SR36 Let *FS* be any foreign server descriptor. *FS* is said to be *abandoned* if the revoke destruction action would result in *A1* no longer having USAGE privilege on the foreign-data wrapper associated with the foreign server described by *FS*.
- 3) Augment SR37 Add abandoned foreign server descriptors to the list of objects that shall not exist.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert this GR For every abandoned foreign server descriptor *FS*, let *FSN* be the <foreign server name> of *FS*. The following <drop foreign server statement> is effectively executed without further Access Rule checking:

DROP SERVER *FSN* CASCADE

Conformance Rules

No additional Conformance Rules.

14.2 <user mapping definition>

Function

Define the mapping of an authorization identifier to a foreign server.

Format

```
<user mapping definition> ::=  
    CREATE USER MAPPING FOR <specific or generic authorization identifier>  
    SERVER <foreign server name>  
    [ <generic options> ]  
  
<specific or generic authorization identifier> ::=  
    <authorization identifier>  
    | USER  
    | CURRENT_USER  
    | PUBLIC
```

Syntax Rules

- 1) Let *FSN* be the <foreign server name>. If <authorization identifier> is specified, then let *U* be the <authorization identifier>; if PUBLIC is specified, then let *U* be PUBLIC; otherwise, let *U* be the current authorization identifier.
- 2) The SQL-environment shall not include an user mapping descriptor whose authorization identifier is *U* and whose foreign server name is equivalent to *FSN*.
- 3) The catalog identified by the explicit or implicit catalog name of *FSN* shall include a foreign server descriptor whose foreign server name is equivalent to *FSN*.

Access Rules

- 1) The applicable privileges shall include the USAGE privilege on the foreign server identified by *FSN*.
- 2) Additional privileges, if any, necessary to execute <user mapping definition> are implementation-defined.

General Rules

- 1) A user mapping descriptor *UMD* is created. *UMD* includes:
 - a) Case:
 - i) If <specific or generic authorization identifier> specifies PUBLIC, then PUBLIC.
 - ii) Otherwise, the authorization identifier *U*.
 - b) The foreign server name *FSN*.
 - c) If <generic options> *GO* is specified, then the generic options descriptor created by *GO*; otherwise, an empty generic options descriptor.

Conformance Rules

- 1) Without Feature M004, “Foreign data support”, conforming SQL language shall not specify <user mapping definition>.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

14.3 <alter user mapping statement>

Function

Change the definition of an user mapping.

Format

```
<alter user mapping statement> ::=  
    ALTER USER MAPPING <specific or generic authorization identifier>  
    SERVER <foreign server name>  
    <alter generic options>
```

Syntax Rules

- 1) Let *FSN* be the <foreign server name> and let *AGO* be the <alter generic options>. If <authorization identifier> is specified, then let *U* be the <authorization identifier>; if PUBLIC is specified, then let *U* be PUBLIC; otherwise, let *U* be the current authorization identifier.
- 2) The SQL-environment shall include a user mapping descriptor *UMD* whose authorization identifier is *U* and whose foreign server name is equivalent to *FSN*.
- 3) The Syntax Rules of Subclause 11.3, "<alter generic options>", are applied to *AGO* with the generic options descriptor included in *UMD* as the applicable generic options descriptor.

Access Rules

- 1) The privileges necessary to execute <alter user mapping statement> are implementation-defined.

General Rules

- 1) The General Rules of Subclause 11.3, "<alter generic options>", are applied to *AGO* with the generic options descriptor included in *AM* as the applicable generic options descriptor.

Conformance Rules

- 1) Without Feature M004, "Foreign data support", conforming SQL language shall not specify <alter user mapping statement>.

14.4 <drop user mapping statement>

Function

Destroy an user mapping.

Format

```
<drop user mapping statement> ::=  
    DROP USER MAPPING FOR <specific or generic authorization identifier>  
    SERVER <foreign server name>
```

Syntax Rules

- 1) Let *FSN* be the <foreign server name>. If <authorization identifier> is specified, then let *U* be the <authorization identifier>; if PUBLIC is specified, then let *U* be PUBLIC; otherwise, let *U* be the current authorization identifier.
- 2) The SQL-environment shall include a user mapping descriptor *UMD* whose authorization identifier is *U* and whose foreign server name is equivalent to *FSN*.

Access Rules

- 1) The privileges necessary to execute <drop user mapping statement> are implementation-defined.

General Rules

- 1) *UMD* is destroyed.

Conformance Rules

- 1) Without Feature M004, "Foreign data support", conforming SQL language shall not specify <drop user mapping statement>.

(Blank page)

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

15 SQL-client modules

15.1 <SQL-client module definition>

Function

Define an SQL-client module.

Format

No additional Format items

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert after GR5)a) If the SQL-session context of any of the SQL-sessions associated with the SQL-agent include {foreign server name : FSConnectionHandle} pairs, then for each such pair:
 - i) Let *CH* be the FSConnectionHandle.
 - ii) The FreeFSConnection() routine is invoked with *CH* as the argument.
- 2) Insert after GR5)a) If the SQL-session context of any of the SQL-sessions associated with the SQL-agent include {foreign-data wrapper name : WrapperEnvHandle} pairs, then for each such pair:
 - i) Let *EH* be the WrapperEnvHandle.
 - ii) The FreeWrapperEnv() routine is invoked with *EH* as the argument.

Conformance Rules

No additional Conformance Rules.

15.2 <externally-invoked procedure>

Function

Define an externally-invoked procedure.

Format

No additional Format items.

Syntax Rules

- 1) Insert before SR1 <host parameter data type> shall not contain a <datalink control definition>.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

15.3 Calls to an <externally-invoked procedure>

Function

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

Syntax Rules

1) Insert into SR2)e)

```

DATA_EXCEPTION_DATAINK_VALUE_EXCEEDS_MAXIMUM_LENGTH:
    constant SQLSTATE_TYPE := "2201D";

DATA_EXCEPTION_INVALID_DATA_SPECIFIED_FOR_DATAINK:
    constant SQLSTATE_TYPE := "22017";
DATA_EXCEPTION_NULL_ARGUMENT_PASSED_TO_DATAINK_CONSTRUCTOR:
    constant SQLSTATE_TYPE := "2201A";
DATAINK_EXCEPTION_NO_SUBCLASS:
    constant SQLSTATE_TYPE := "HW000";
DATAINK_EXCEPTION_EXTERNAL_FILE_NOT_LINKED:
    constant SQLSTATE_TYPE := "HW001";
DATAINK_EXCEPTION_EXTERNAL_FILE_ALREADY_LINKED:
    constant SQLSTATE_TYPE := "HW002";
DATAINK_EXCEPTION_REFERENCED_FILE_DOES_NOT_EXIST:
    constant SQLSTATE_TYPE := "HW003";
FDW_SPECIFIC_CONDITION_NO_SUBCLASS:
    constant SQLSTATE_TYPE := "HV000";
FDW_SPECIFIC_CONDITION_COLUMN_NAME_NOT_FOUND:
    constant SQLSTATE_TYPE := "HV005";
FDW_SPECIFIC_CONDITION_DYNAMIC_PARAMETER_VALUE_NEEDED:
    constant SQLSTATE_TYPE := "HV002";
FDW_SPECIFIC_CONDITION_FUNCTION_SEQUENCE_ERROR:
    constant SQLSTATE_TYPE := "HV010";
FDW_SPECIFIC_CONDITION_INCONSISTENT_DESCRIPTOR_INFORMATION:
    constant SQLSTATE_TYPE := "HV021";
FDW_SPECIFIC_CONDITION_INVALID_ATTRIBUTE_VALUE:
    constant SQLSTATE_TYPE := "HV024";
FDW_SPECIFIC_CONDITION_INVALID_COLUMN_NAME:
    constant SQLSTATE_TYPE := "HV007";
FDW_SPECIFIC_CONDITION_INVALID_COLUMN_NUMBER:
    constant SQLSTATE_TYPE := "HV008";
FDW_SPECIFIC_CONDITION_INVALID_DATA_TYPE:
    constant SQLSTATE_TYPE := "HV004";
FDW_SPECIFIC_CONDITION_INVALID_DATA_TYPE_DESCRIPTOR:
    constant SQLSTATE_TYPE := "HV006";
FDW_SPECIFIC_CONDITION_INVALID_DESCRIPTOR_FIELD_IDENTIFIER:
    constant SQLSTATE_TYPE := "HV091";
FDW_SPECIFIC_CONDITION_INVALID_HANDLE:
    constant SQLSTATE_TYPE := "HV00B";
FDW_SPECIFIC_CONDITION_INVALID_OPTION_INDEX:
    constant SQLSTATE_TYPE := "HV00C";
FDW_SPECIFIC_CONDITION_INVALID_OPTION_NAME:
    constant SQLSTATE_TYPE := "HV00D";
FDW_SPECIFIC_CONDITION_INVALID_STRING_FORMAT:
    constant SQLSTATE_TYPE := "HV00A";
FDW_SPECIFIC_CONDITION_INVALID_STRING_LENGTH_OR_BUFFER_LENGTH:
    constant SQLSTATE_TYPE := "HV090";
FDW_SPECIFIC_CONDITION_INVALID_USE_OF_NULL_POINTER:
    constant SQLSTATE_TYPE := "HV009";
FDW_SPECIFIC_CONDITION_LIMIT_ON_NUMBER_OF_HANDLES_EXCEEDED:
    constant SQLSTATE_TYPE := "HV014";
FDW_SPECIFIC_CONDITION_MEMORY_ALLOCATION_ERROR:
    constant SQLSTATE_TYPE := "HV001";

```

ISO/IEC 9075-9:2001 (E)

15.3 Calls to an <externally-invoked procedure>

```
FDW_SPECIFIC_CONDITION_NO_SCHEMAS:
    constant SQLSTATE_TYPE := "HV00P";
FDW_SPECIFIC_CONDITION_OPTION_NAME_NOT_FOUND:
    constant SQLSTATE_TYPE := "HV00J";
FDW_SPECIFIC_CONDITION_REPLY_HANDLE:
    constant SQLSTATE_TYPE := "HY00K";
FDW_SPECIFIC_CONDITION_SCHEMA_NOT_FOUND:
    constant SQLSTATE_TYPE := "HV00Q";
FDW_SPECIFIC_CONDITION_TABLE_NOT_FOUND:
    constant SQLSTATE_TYPE := "HV00R";
FDW_SPECIFIC_CONDITION_UNABLE_TO_CREATE_EXECUTION:
    constant SQLSTATE_TYPE := "HV00L";
FDW_SPECIFIC_CONDITION_UNABLE_TO_CREATE_REPLY:
    constant SQLSTATE_TYPE := "HV00M";
FDW_SPECIFIC_CONDITION_UNABLE_TO_ESTABLISH_CONNECTION:
    constant SQLSTATE_TYPE := "HV00N";
INVALID_FOREIGN_SERVER_SPECIFICATION_NO_SUBCLASS:
    constant SQLSTATE_TYPE := "0X000";
PASSTHROUGH_SPECIFIC_CONDITION_NO_SUBCLASS:
    constant SQLSTATE_TYPE := "0Y000";
PASSTHROUGH_SPECIFIC_CONDITION_INVALID_CURSOR_OPTION:
    constant SQLSTATE_TYPE := "0Y001";
PASSTHROUGH_SPECIFIC_CONDITION_INVALID_CURSOR_ALLOCATION:
    constant SQLSTATE_TYPE := "0Y002";
```

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

15.4 <SQL procedure statement>

Function

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

Format

```
<SQL schema definition statement> ::=  
    !! All alternatives from ISO/IEC 9075-2  
    | !! All alternatives from ISO/IEC 9075-5  
    | <foreign table definition>  
    | <foreign server definition>  
    | <foreign-data wrapper definition>  
    | <user mapping definition>  
  
<SQL schema manipulation statement> ::=  
    !! All alternatives from ISO/IEC 9075-2  
    | !! All alternatives from ISO/IEC 9075-5  
    | <alter foreign table statement>  
    | <drop foreign table statement>  
    | <alter foreign server statement>  
    | <drop foreign server statement>  
    | <alter foreign-data wrapper statement>  
    | <drop foreign-data wrapper statement>  
    | <alter user mapping statement>  
    | <drop user mapping statement>  
  
<SQL session statement> ::=  
    !! All alternatives from ISO/IEC 9075-2  
    | <set passthrough statement>
```

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

15.5 Data type correspondences

Function

Specify the data type correspondences for SQL data types and host language types.

Tables

Table 6—Data type correspondences for Ada

SQL Data Type	Ada Data Type
<i>All alternatives from ISO/IEC 9075-2</i> <i>All alternatives from ISO/IEC 9075-5</i> DATALINK	SQL_STANDARD.CHAR, with P'LENGTH of LD^1
¹ The length LD of the character data type corresponding with SQL data type DATALINK is the smallest integer not less than the quotient of the division N/B , where N is the maximum datalink length and B is the implementation-defined number of octets contained in a character of the host language. (The term “maximum datalink length” is defined in Subclause 4.8, “Datalinks”.)	

Table 7—Data type correspondences for C

SQL Data Type	C Data Type
<i>All alternatives from ISO/IEC 9075-2</i> <i>All alternatives from ISO/IEC 9075-5</i> DATALINK	char, with length LD^5
⁵ The length LD of the character data type corresponding with SQL data type DATALINK is the smallest integer not less than the quotient of the division N/B , where N is the maximum datalink length and B is the implementation-defined number of octets contained in a character of the host language. (The term “maximum datalink length” is defined in Subclause 4.8, “Datalinks”.)	

Table 8—Data type correspondences for COBOL

SQL Data Type	COBOL Data Type
<i>All alternatives from ISO/IEC 9075-2</i> <i>All alternatives from ISO/IEC 9075-5</i> DATALINK	alphanumeric, with length LD^4
⁴ The length LD of the character data type corresponding with SQL data type DATALINK is the smallest integer not less than the quotient of the division N/B , where N is the maximum datalink length and B is the implementation-defined number of octets contained in a character of the host language. (The term “maximum datalink length” is defined in Subclause 4.8, “Datalinks”.)	

Table 9—Data type correspondences for Fortran

SQL Data Type	Fortran Data Type
<i>All alternatives from ISO/IEC 9075-2</i> <i>All alternatives from ISO/IEC 9075-5</i> DATALINK	CHARACTER with length LD^4
⁴ The length LD of the character data type corresponding with SQL data type DATALINK is the smallest integer not less than the quotient of the division N/B , where N is the maximum datalink length and B is the implementation-defined number of octets contained in a character of the host language. (The term “maximum datalink length” is defined in Subclause 4.8, “Datalinks”).	

Table 10—Data type correspondences for MUMPS

SQL Data Type	MUMPS Data Type
<i>All alternatives from ISO/IEC 9075-2</i> <i>All alternatives from ISO/IEC 9075-5</i> DATALINK	character

Table 11—Data type correspondences for Pascal

SQL Data Type	Pascal Data Type
<i>All alternatives from ISO/IEC 9075-2</i> <i>All alternatives from ISO/IEC 9075-5</i> DATALINK	PACKED ARRAY[1.. LD^2] OF CHAR
² The length LD of the character data type corresponding with SQL data type DATALINK is the smallest integer not less than the quotient of the division N/B , where N is the maximum datalink length and B is the implementation-defined number of octets contained in a character of the host language. (The term “maximum datalink length” is defined in Subclause 4.8, “Datalinks”).	

Table 12—Data type correspondences for PL/I

SQL Data Type	PL/I Data Type
<i>All alternatives from ISO/IEC 9075-2</i> <i>All alternatives from ISO/IEC 9075-5</i>	

(Continued on next page)

ISO/IEC 9075-9:2001 (E)
15.5 Data type correspondences

Table 12—Data type correspondences for PL/I (Cont.)

SQL Data Type	PL/I Data Type
DATALINK	CHARACTER VARYING(LD^2)
² The length LD of the character data type corresponding with SQL data type DATALINK is the smallest integer not less than the quotient of the division N/B , where N is the maximum datalink length and B is the implementation-defined number of octets contained in a character of the host language. (The term “maximum datalink length” is defined in Subclause 4.8, “Datalinks”.)	

Conformance Rules

No additional Conformance Rules.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

16 Data manipulation

16.1 <declare cursor>

Function

Define a cursor.

Format

No additional Format items.

Syntax Rules

- 1) Insert after SR18)b) *DT* shall not be DATALINK-ordered.

Access Rules

No additional Access Rules.

General Rules

No additional Access Rules.

Conformance Rules

No additional Access Rules.

16.2 Effect of deleting rows from base tables

Function

Specify the effect of deleting rows from one or more base tables.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert after GR6) For each row *R* that is marked for deletion from *T*, for each component *DLC* of *T* whose declared type is DATALINK or some distinct type with a source data type of DATALINK and whose descriptor includes a link control option that specifies FILE LINK CONTROL, let *DLCV* be the value of *DLC* in *R*.
NOTE 40 – “component” is defined in Subclause 4.10, “Columns, fields, and attributes”.
- 2) Insert after GR6) If *DLCV* is not the null value, then let *EF* be the external file referenced by *DLCV*.
Case:
 - a) If *EF* is not linked and the integrity control option included in the descriptor of *DLC* specifies INTEGRITY ALL, then an exception condition is raised: *datalink exception — external file not linked*.
 - b) If *EF* is linked, then *EF* is unlinked.
NOTE 41 – The effect of unlinking depends on the unlink control option, RESTORE or DELETE, included in the column descriptor of *C*, as specified in Subclause 4.10, “Columns, fields, and attributes”.

Conformance Rules

No additional Conformance Rules.

16.3 Effect of inserting tables into base tables

Function

Specify the effect of inserting each of one or more given tables into its associated base table.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert this GR For each row *R* inserted into *T*, for each component *DLC* of *T* whose declared type is DATALINK or some distinct type with a source data type of DATALINK and whose descriptor includes a link control option that specifies FILE LINK CONTROL and includes a link control option that is INTEGRITY ALL or INTEGRITY SELECTIVE, let *DLCV* be the value of *DLC* in *R*.

NOTE 42 – “component” is defined in Subclause 4.10, “Columns, fields, and attributes”.

If *DLCV* is not the null value, then let *EF* be the external file referenced by *DLCV*.

a) Case:

- i) If *EF* is linked, then an exception condition is raised: *datalink exception — external file already linked*.
- ii) If INTEGRITY ALL is specified, then *EF* is linked according to the <datalink file control option> READ PERMISSION and WRITE PERMISSION of *DLC*.
- iii) If INTEGRITY SELECTIVE is specified, then *EF* may be linked in an implementation-defined manner according to the <datalink file control option> READ PERMISSION and WRITE PERMISSION of *DLC*.

b) Case:

- i) If the read permission option included in the descriptor of *DLC* is DB, then the SQL-Mediated Access Indication of *DLCV* is set to True .
- ii) Otherwise, the SQL-Mediated Access Indication of *DLCV* is set to False .

- 2) If *DLCV* does not reference a file, then an exception condition is raised: *datalink exception — referenced file does not exist*

Conformance Rules

No additional Conformance Rules.

16.4 Effect of replacing rows in base tables

Function

Specify the effect of replacing some of the rows in one or more base tables.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert after GR7) For each replaced row *R*, for each component *DLC* of *R* whose declared type is DATALINK or some distinct type with a source data type of DATALINK and whose descriptor includes a link control option that specifies FILE LINK CONTROL and includes a <datalink file control option> that is INTEGRITY ALL or INTEGRITY SELECTIVE, let *DLCV1* be the existing value of *DLC* in *R* and let *DLCV2* be the corresponding datalink in the new transition variable.

NOTE 43 – “component” is defined in Subclause 4.10, “Columns, fields, and attributes”.

- a) If *DLCV1* is not the null value, then let *EF1* be the external file referenced by *DLCV1*.

Case:

- i) If *EF1* is not linked and the integrity control option included in the descriptor of *DLC* specifies INTEGRITY ALL, then an exception condition is raised: *datalink exception — external file not linked*.
- ii) If *EF1* is linked, *EF1* is unlinked.

NOTE 44 – The effect of unlinking depends on the unlink control option, RESTORE or DELETE, included in the column descriptor of *C*, as specified in Subclause 4.10, “Columns, fields, and attributes”.

- b) If *DLCV2* is not the null value, then let *EF2* be the external file referenced by *DLCV2*.

i) Case:

- 1) If *EF2* is linked, then an exception condition is raised: *datalink exception — external file already linked*.
- 2) If INTEGRITY ALL is specified, then *EF2* is linked according to the <datalink file control option> of READ PERMISSION and WRITE PERMISSION of *DLC*.
- 3) If INTEGRITY SELECTIVE is specified, then *EF2* may be linked in an implementation-defined manner according to the <datalink file control option> of READ PERMISSION and WRITE PERMISSION of *DLC*.

16.4 Effect of replacing rows in base tables

ii) Case:

- 1) If the read permission option included in the descriptor of *DLC* is DB, then the SQL-Mediated Access Indication of *DLCV2* is set to True .
 - 2) Otherwise, the SQL-Mediated Access Indication of *DLCV2* is set to False .
- 2) If *DLCV2* does not reference a file, then an exception condition is raised: *datalink exception — referenced file does not exist*.

Conformance Rules

No additional Conformance Rules.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

(Blank page)

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

17 Session management

17.1 <set passthrough statement>

Function

Set the pass-through flag to True or False for the current SQL-session context.

Format

```
<set passthrough statement> ::=
    SET PASSTHROUGH <passthrough specification>

<passthrough specification> ::=
    <value specification>
    | OFF
```

Syntax Rules

The declared type of the <value specification> shall be an SQL character data type.

Access Rules

None.

General Rules

- 1) If there is a pass-through foreign server name included in the current SQL-session context, then let *FSN* be that pass-through foreign server name, let *WN* be the name of the foreign-data wrapper included in the foreign server descriptor of the foreign server identified by *FSN*, let *WR* be the foreign-data wrapper identified by *WN*, and let *WRLN* be the name of the library identified in the foreign-data wrapper descriptor of *WR*.
- 2) For each execution handle *EXH_i* that is part of an {<SQL statement name> : ExecutionHandle} pair that is present in the current SQL-session context, the FreeExecutionHandle() routine in the library identified by *WRLN* is invoked with *EXH_i* as the argument.
- 3) All {<SQL statement name> : ExecutionHandle} pairs present in the current SQL-session context are removed from the current SQL-session context.
- 4) Case:
 - a) If <value specification> is specified, then:
 - i) Let *S* be <value specification> and let *V* be the character string that is the value of
`TRIM ( BOTH ' ' FROM S )`
 - ii) If *V* does not conform to the Format and Syntax Rules of a <foreign server name>, then an exception condition is raised: *invalid foreign server specification*.

ISO/IEC 9075-9:2001 (E)

17.1 <set passthrough statement>

- iii) If a foreign server descriptor that includes *V* as the foreign server name exists, then let *FS* be that foreign server. Otherwise, an exception condition is raised: *invalid foreign server specification*.
 - iv) If the current privileges do not include USAGE privilege on *FS*, then an exception condition is raised: *invalid foreign server specification*.
 - v) The pass-through flag of the current SQL-session context is set to True .
 - vi) The pass-through foreign server name included in the current SQL-session context is set to the foreign server name of *FS*.
- b) Otherwise:
- i) The pass-through flag of the current SQL-session context is set to False .
 - ii) The pass-through foreign server name included in the current SQL-session context is deleted.

Conformance Rules

- 1) Without Feature M004, “Foreign data support”, conforming SQL language shall not contain any <set passthrough statement>.

18 Dynamic SQL

18.1 Description of SQL descriptor areas

Function

Specify the identifiers, data types, and codes used in SQL item descriptor areas.

Syntax Rules

- 1) Insert before SR6)q) TYPE indicates DATALINK.
- 2) Insert before SR7)v) TYPE indicates DATALINK and *T* is specified by DATALINK.

Access Rules

No additional Access Rules.

General Rules

- 1) Replace GR1) Table 13, “Codes used for SQL data types in Dynamic SQL”, specifies the codes associated with the SQL data types.

Table 13—Codes used for SQL data types in Dynamic SQL

Data Type	Code
<i>All alternatives from ISO/IEC 9075-5</i>	<i>All alternatives from ISO/IEC 9075-5</i>
DATALINK	70

Conformance Rules

- 1) Without Feature M003, “Datalinks via SQL language”, TYPE shall not indicate DATALINK.

18.2 <prepare statement>

Function

Prepare a statement for execution.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert before GR1 If the pass-through flag of the current SQL-session context is True and if <SQL statement variable> conforms to the Format and Syntax Rules of a <preparable statement> other than <set passthrough statement>, then:
 - a) Let *FSN* be the pass-through foreign server name included in the current SQL-session context. Let *WN* be the name of the foreign-data wrapper included in the foreign server descriptor of the foreign server identified by *FSN*. Let *WR* be the foreign-data wrapper identified by *WN*. Let *WRLN* be the name of the library identified in the foreign-data wrapper descriptor of *WR*.
 - b) Case:
 - i) If the current SQL-session context includes a {foreign-data wrapper name : WrapperEnvHandle} pair whose foreign-data wrapper name is equivalent to *WN*, then let *WEH* be the WrapperEnvHandle associated with *WN*.
 - ii) Otherwise:
 - 1) Let *WH* be the WrapperHandle allocated for the foreign-data wrapper identified by *WN*. The resource identified by *WH* is referred to as an *allocated foreign-data wrapper description*.
 - 2) Let *WEH* be the WrapperEnvHandle returned by the invocation of the AllocWrapperEnv() routine in the library identified by *WRLN*, with *WH* as the argument.
 - 3) The { *WN* : *WEH* } pair is included in the current SQL-session context.
 - 4) *WH* is deallocated and all its resources are freed.

c) Case:

- i) If the current SQL-session context includes a {foreign server name : FSConnectionHandle} pair whose foreign server name is equivalent to *FSN*, then let *FSCH* be the FSConnectionHandle associated with *FSN*.
- ii) Otherwise:
 - 1) Let *SH* be the ServerHandle allocated for the foreign server identified by *FSN*. The resource identified by *SH* is referred to as an *allocated foreign server description*.
 - 2) If there is a user mapping identified by the current authorization identifier, then let *UH* be the UserHandle allocated for that user mapping; otherwise, let *UH* be the UserHandle allocated for the user mapping identified by PUBLIC. The resource identified by *UH* is referred to as an *allocated user mapping description*.
 - 3) Let *FSCH* be the FSConnectionHandle returned by the invocation of the ConnectServer() routine in the library identified by *WRLN* with *WEH*, *SH*, and *UH* as the arguments.
 - 4) The {*FSN* : *FSCH*} pair is included in the current SQL-session context.
 - 5) *SH* is deallocated and all its resources are freed.
 - 6) *UH* is deallocated and all its resources are freed.
- d) Let *STV* be the contents of <SQL statement variable>. Let *STVL* be the length of *STV*. Let *EXH* be the ExecutionHandle returned by the invocation of the TransmitRequest() routine in the library identified by *WRLN* with *FSCH*, *STV*, and *STVL* as arguments.
- e) If the current SQL-session context includes an {SQL statement name : ExecutionHandle} pair whose SQL statement name is equivalent to <SQL statement name>, then let *OEXH* be the ExecutionHandle associated with <SQL statement name>.
 - i) The FreeExecutionHandle() routine in the library identified by *WRLN* is invoked with *OEXH* as the argument.
 - ii) The {SQL statement name : ExecutionHandle} pair whose SQL statement name is equivalent to <SQL statement name> is removed from the current SQL-session context.
- f) The {<SQL statement name> : *EXH*} pair is included in the current SQL-session context.
- g) No further General Rules of this Subclause are applied.

Conformance Rules

No additional Conformance Rules.

18.3 <deallocate prepared statement>

Function

Deallocate SQL-statements that have been prepared with a <prepare statement>.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert before GR1) If the pass-through flag of the current SQL-session context is *True* , and the current SQL-session context includes an {SQL statement name : ExecutionHandle} pair whose SQL statement name is equivalent to <SQL statement name>, then:
 - a) Let *FSN* be the pass-through foreign server name included in the current SQL-session context. Let *WN* be the name of the foreign-data wrapper included in the foreign server descriptor of the foreign server identified by *FSN*. Let *WR* be the foreign-data wrapper identified by *WN*. Let *WRLN* be the name of the library identified in the foreign-data wrapper descriptor of *WR*. Let *EXH* be the ExecutionHandle associated with <SQL statement name>.
 - b) The FreeExecutionHandle() routine in the library identified by *WRLN* is invoked with *EXH* as the argument.
 - c) The {SQL statement name : ExecutionHandle} pair whose SQL statement name is equivalent to <SQL statement name> is removed from the current SQL-session context.
 - d) No further General Rules of this Subclause are applied.

Conformance Rules

No additional Conformance Rules.

18.4 <describe statement>

Function

Obtain information about the <select list> columns or <dynamic parameter specification>s contained in a prepared statement or about the columns of the result set associated with a cursor.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert before GR1) If the pass-through flag of the current SQL-session context is *True* , and the current SQL-session context includes an {SQL statement name : ExecutionHandle} pair whose SQL statement name is equivalent to <SQL statement name>, then:
 - a) Let *EXH* be the ExecutionHandle associated with <SQL statement name>. Let *WPD* and *WRD* be the wrapper parameter descriptor and server parameter descriptor, respectively, associated with the WPDHandle and WRDHandle, respectively, that would be returned be the invocation of the GetWPDHandle() and GetWRDHandle() routines with *EXH* as the ExecutionHandle parameter.
 - b) An SQL system descriptor area shall have been allocated and not yet deallocated whose name is the value of the <descriptor name>'s <simple value specification> and whose scope is that specified by the <scope option>. Otherwise, an exception condition is raised: *invalid SQL descriptor name*.
 - c) Let *DA* be the descriptor area identified by the <descriptor name>. Let *N* be the <occurrences> specified when *DA* was allocated.
 - d) *DA* is set as follows:
 - i) If the statement being executed is a <describe output statement>, then:
 - 1) Let *TD* be the value of the COUNT field in *WRD*.
 - 2) If *TD* is greater than *N*, then a completion condition is raised: *warning — insufficient item descriptor areas*.
 - 3) All header fields are set to the values of the header fields of *WRD* with the same name.

ISO/IEC 9075-9:2001 (E)
18.4 <describe statement>

- 4) If TD is 0 (zero) or TD is greater than N , then no item descriptor areas are set. Otherwise, the first TD item descriptor areas are set with values from the corresponding item descriptor areas and, optionally, subordinate descriptors from WRD .
- ii) If the statement being executed is a <describe input statement>, then:
 - 1) Let TD be the value of the COUNT field in WPD .
 - 2) If TD is greater than N , then a completion condition is raised: *warning — insufficient item descriptor areas*.
 - 3) All header fields are set to the values of the header fields of WPD with the same name.
 - 4) If TD is 0 (zero) or TD is greater than N , then no item descriptor areas are set. Otherwise, the first TD item descriptor areas are set with values from the corresponding item descriptor areas and, optionally, subordinate descriptors from WPD .
- e) No further General Rules of this Subclause are applied.
- 2) Insert after GR8)d)xi) If TYPE indicates DATALINK, then LENGTH and OCTET_LENGTH are set to the maximum datalink length.
NOTE 45 – The term “maximum datalink length” is defined in Subclause 4.8, “Datalinks”.

Conformance Rules

- 1) Without Feature M003, “Datalinks via SQL language”, TYPE shall not indicate DATALINK.

18.5 <input using clause>

Function

Supply input values for an <SQL dynamic statement>.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert after GR1) If the pass-through flag of the current SQL-session context is True , then:
 - a) Let *HL1* be the standard host language in which the SQL-server is written and let *HL2* be the standard host language in which the foreign-data wrapper is written.
 - b) Case:
 - i) If an <input using clause> is used in a <dynamic open statement>, then let *SN* be the <statement name> of the associated <dynamic declare cursor>.
 - ii) Otherwise, let *SN* be the <SQL statement name> of the <execute statement>.
 - c) Let *EXH* be the ExecutionHandle associated with *SN*. Let *WPD* and *SPD* be the wrapper parameter descriptor and server parameter descriptor, respectively, associated with the *WPDHandle* and *SPDHandle*, respectively, that would be returned by the invocation of the *GetWPDHandle()* and *GetSPDHandle()* routines with *EXH* as the ExecutionHandle parameter.
 - d) Let *D* be the value of COUNT in *WPD*.
 - e) If <using arguments> is specified and the number of <using argument>s is not *D*, then an exception condition is raised: *dynamic SQL error — using clause does not match dynamic parameter specifications*.
 - f) If <using input descriptor> is specified, then:
 - i) Let *DA* be the descriptor area identified by <descriptor name>.
 - ii) Let *N* be the value of COUNT in *DA*.
 - iii) If *N* is greater than the value of <occurrences> specified when the SQL descriptor area identified by <descriptor name> was allocated or is less than zero, then an exception condition is raised: *dynamic SQL error — invalid descriptor count*.

ISO/IEC 9075-9:2001 (E)
18.5 <input using clause>

- iv) If the first N item descriptor areas in DA are not valid as specified in Subclause 18.1, “Description of SQL descriptor areas”, then an exception condition is raised: *dynamic SQL error — using clause does not match dynamic parameter specifications*.
- v) In the first N item descriptor areas in DA :
 - 1) If the number of item descriptor areas in which the value of $LEVEL$ is 0 (zero) is not D , then an exception condition is raised: *dynamic SQL error — using clause does not match dynamic parameter specifications*.
 - 2) If the value of $INDICATOR$ is not negative, $TYPE$ does not indicate ROW , and the item descriptor area is not subordinate to an item descriptor area whose $INDICATOR$ value is negative or whose $TYPE$ field indicates $ARRAY$ or $ARRAY LOCATOR$, and if the value of $DATA$ is not a valid value of the data type represented by the item descriptor area, then an exception condition is raised: *dynamic SQL error — using clause does not match dynamic parameter specifications*.
- g) For $1 \text{ (one)} \leq i \leq D$:
 - i) Let TDT be the effective declared type of the i -th input <dynamic parameter specification> defined by the type representation of the corresponding item descriptor area and its subordinate descriptor areas in WPD .
 - ii) Case:
 - 1) If <using input descriptor> is specified, then:
 - A) Let IDA be the i -th item descriptor area in DA whose $LEVEL$ value is 0 (zero).
 - B) Let SDT be the effective declared type represented by IDA .
 - C) Let SV be the associated value of IDA , which is defined to be
Case:
 - I) If the value of $INDICATOR$ is negative, then SV is the null value.
 - II) Otherwise,
Case:
 - 1) If $TYPE$ indicates ROW , then SV is a row whose type is SDT and whose field values are the associated values of the immediately subordinate descriptor areas of IDA .
 - 2) Otherwise, SV is the value of $DATA$ with data type SDT .
 - 2) If <using arguments> is specified, then let SDT and SV be the declared type and value, respectively, of the i -th <using argument>.
 - iii) Case:
 - 1) If SDT is a locator type, then:
 - A) If SV is not the null value, then let the value TV_i of the i -th dynamic parameter be the value of SV .
 - B) Otherwise, let the value TV_i of the i -th dynamic parameter be the null value.

- 2) If *SDT* and *TDI* are predefined data types, then

Case:

- A) If the <cast specification>

CAST (*SV* AS *TDI*)

does not conform to the Syntax Rules of Subclause 6.22, "<cast specification>", in ISO/IEC 9075-2, and there is an implementation-defined conversion from type *SDT* to type *TDI*, then that implementation-defined conversion is effectively performed, converting *SV* to type *TDI*, and the result is the value TV_i of the *i*-th input dynamic parameter.

- B) Otherwise:

- I) If the <cast specification>

CAST (*SV* AS *TDI*)

does not conform to the Syntax Rules of Subclause 6.22, "<cast specification>", in ISO/IEC 9075-2, then an exception condition is raised: *dynamic SQL error — restricted data type attribute violation*.

- II) If the <cast specification>

CAST (*SV* AS *TDI*)

does not conform to the General Rules of Subclause 6.22, "<cast specification>", in ISO/IEC 9075-2, then an exception condition is raised in accordance with the General Rules of Subclause 6.22, "<cast specification>", in ISO/IEC 9075-2.

- III) The <cast specification>

CAST (*SV* AS *TDI*)

is effectively performed and the result is the value TV_i of the *i*-th input dynamic parameter.

- iv) Case:

- 1) If <using input descriptor> is specified, then all fields, except *DATA* and *DATA_POINTER*, in the *i*-th item descriptor area of *SPD*, that can be set according to Table 35, "Ability to set foreign-data wrapper descriptor fields", are set with values from the corresponding fields of the item descriptor area and, optionally, subordinate descriptors of *DA*.
- 2) If <using arguments> is specified, then all fields, except *DATA* and *DATA_POINTER*, in the *i*-th item descriptor area of *SPD*, that can be set according to Table 35, "Ability to set foreign-data wrapper descriptor fields", are set to implementation-dependent values.

- v) Case:

- 1) If *HL1* and *HL2* are both pointer-supporting languages, then the *DATA_POINTER* field in the *i*-th item descriptor area of *SPD* is set to the address of the buffer that contains the value TV_i .
- 2) Otherwise, the *DATA* field in the *i*-th item descriptor area of *SPD* is set to TV_i .

ISO/IEC 9075-9:2001 (E)
18.5 <input using clause>

- h) All header fields in *SPD*, that can be set according to Table 35, “Ability to set foreign-data wrapper descriptor fields”, are set to the values of the header fields of *WPD* with equivalent names.
- i) No further General Rules of this Subclause are applied.

Conformance Rules

No additional Conformance Rules.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

18.6 <output using clause>

Function

Supply output variables for an <SQL dynamic statement>.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert after GR1) If the pass-through flag of the current SQL-session context is True , then:
 - a) Let *HL1* be the standard host language in which the SQL-server is written and let *HL2* be the standard host language in which the foreign-data wrapper is written.
 - b) Case:
 - i) If an <output using clause> is used in a <dynamic fetch statement>, then let *SN* be the <statement name> of the associated <dynamic declare cursor>.
 - ii) Otherwise, let *SN* be the <SQL statement name> of the <execute statement>.
 - c) Let *EXH* be the ExecutionHandle associated with *SN*. Let *WRD* and *SRD* be the wrapper parameter descriptor and server parameter descriptor, respectively, associated with the *WRDHandle* and *SRDHandle*, respectively, that would be returned by the invocation of the *GetWRDHandle*() and *GetSRDHandle*() routines with *EXH* as the ExecutionHandle parameter.
 - d) Let *D* be the value of COUNT in *WRD*.
 - e) If <into arguments> is specified and the number of <into argument>s is not *D*, then an exception condition is raised: *dynamic SQL error — using clause does not match target specifications*.
 - f) If <into descriptor> is specified, then:
 - i) Let *DA* be the descriptor area identified by <descriptor name>.
 - ii) Let *N* be the value of COUNT in *DA*.
 - iii) If *N* is greater than the value of <occurrences> specified when the SQL descriptor area identified by <descriptor name> was allocated or is less than zero, then an exception condition is raised: *dynamic SQL error — invalid descriptor count*.

ISO/IEC 9075-9:2001 (E)
18.6 <output using clause>

- iv) If the first N item descriptor areas in DA are not valid as specified in Subclause 18.1, “Description of SQL descriptor areas”, then an exception condition is raised: *dynamic SQL error — using clause does not match target specifications*.
- v) In the first N item descriptor areas in DA , if the number of item descriptor areas in which the value of $LEVEL$ is 0 (zero) is not D , then an exception condition is raised: *dynamic SQL error — using clause does not match target specifications*.
- g) For $1 \text{ (one)} \leq i \leq D$:
 - i) Let SDT be the effective declared type of i -th descriptor area whose $LEVEL$ value is 0 (zero) and its subordinate descriptor areas in WRD .
 Case:
 - 1) If $HL1$ and $HL2$ are both pointer-supporting languages, then let SV be the value of the buffer addressed by the $DATA_POINTER$ field in the corresponding item descriptor area of SRD , with data type SDT .
 - 2) Otherwise, let SV be the value of the $DATA$ field in the corresponding item descriptor area of SRD , with data type SDT .
 - ii) Case:
 - 1) If <into descriptor> is specified, then:
 - A) Let IDA be the i -th item descriptor area in DA whose $LEVEL$ value is 0 (zero).
 - B) Let TDT be the declared type represented by IDA .
 - 2) If <into arguments> is specified, then let TDT be the data type of the i -th <into argument>.
 - iii) If the <output using clause> is used in a <dynamic fetch statement>, then let $LTDT$ be the data type on the most recently executed <dynamic fetch statement>, if any, for the cursor CR . It is implementation-defined whether or not an exception condition is raised: *dynamic SQL error — restricted data type attribute violation* if any of the following are true:
 - 1) $LTDT$ and TDT both identify a binary large object type and only one of $LTDT$ and TDT is a binary large object locator.
 - 2) $LTDT$ and TDT both identify a character large object type and only one of $LTDT$ and TDT is a character large object locator.
 - 3) $LTDT$ and TDT both identify an array type and only one of $LTDT$ and TDT is an array locator.
 - 4) $LTDT$ and TDT both identify a user-defined type and only one of $LTDT$ and TDT is a user-defined type locator.
 - iv) Case:
 - 1) If TDT is a locator type, then:
 - A) If SV is not the null value, then a locator L that uniquely identifies SV is generated and is the value TV_i of the i -th <target specification>.

B) Otherwise, the value TV_i of the i -th <target specification> is the null value.

2) If SDT and TDT are predefined data types, then

Case:

A) If the <cast specification>

CAST (SV AS TDT)

does not conform to the Syntax Rules of Subclause 6.22, "<cast specification>", in ISO/IEC 9075-2, and there is an implementation-defined conversion from type SDT to type TDT , then that implementation-defined conversion is effectively performed, converting SV to type TDT , and the result is the value TV_i of the i -th <target specification>.

B) Otherwise:

I) If the <cast specification>

CAST (SV AS TDT)

does not conform to the Syntax Rules of Subclause 6.22, "<cast specification>", in ISO/IEC 9075-2, then an exception condition is raised: *dynamic SQL error — restricted data type attribute violation*.

II) If the <cast specification>

CAST (SV AS TDT)

does not conform to the General Rules of Subclause 6.22, "<cast specification>", in ISO/IEC 9075-2, then an exception condition is raised in accordance with the General Rules of Subclause 6.22, "<cast specification>", in ISO/IEC 9075-2.

III) The <cast specification>

CAST (SV AS TDT)

is effectively performed and the result is the value TV_i of the i -th <target specification>.

v) Case:

1) If <into descriptor> is specified, then all fields in IDA are set with values from the corresponding fields of the item descriptor area and, optionally, subordinate descriptors of SRD .

2) If <into arguments> is specified, then the Rules in Subclause 10.1, "Retrieval assignment", are applied to TV_i and the i -th <into argument> as $VALUE$ and $TARGET$, respectively.

h) No further General Rules of this Subclause are applied.

Conformance Rules

No additional Conformance Rules.

18.7 <execute statement>

Function

Associate input SQL parameters and output targets with a prepared statement and execute the statement.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert before GR1 If the pass-through flag of the current SQL-session context is True , and the current SQL-session context includes an {SQL statement name : ExecutionHandle} pair whose SQL statement name is equivalent to <SQL statement name>, then:
 - a) Let *FSN* be the pass-through foreign server name included in the current SQL-session context. Let *WN* be the name of the foreign-data wrapper included in the foreign server descriptor of the foreign server identified by *FSN*. Let *WR* be the foreign-data wrapper identified by *WN*. Let *WRLN* be the name of the library identified in the foreign-data wrapper descriptor of *WR*. Let *EXH* be the ExecutionHandle associated with <SQL statement name>.
 - b) If a <parameter using clause> is specified, then the General Rules specified in Subclause 18.5, "<input using clause>", for a <parameter using clause> in an <execute statement> are applied.
 - c) The Open() routine in the library identified by *WRLN* is invoked with *EXH* as argument.
 - d) If a <result using clause> is specified, then the General Rules specified in Subclause 18.6, "<output using clause>", for a <result using clause> in an <execute statement> are applied.
 - e) No further General Rules of this Subclause are applied.

Conformance Rules

No additional Conformance Rules.

18.8 <dynamic declare cursor>

Function

Declare a cursor to be associated with a <statement name>, which may in turn be associated with a <cursor specification>.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert before GR1 If the pass-through flag of the current SQL-session context is True , then:
 - a) If <cursor sensitivity> is specified, then an exception condition is raised: *pass-through specific condition — invalid cursor option*.
 - b) If <cursor scrollability> is specified, then an exception condition is raised: *pass-through specific condition — invalid cursor option*.
 - c) If <cursor holdability> is specified, then an exception condition is raised: *pass-through specific condition — invalid cursor option*.
 - d) If <cursor returnability> is specified, then an exception condition is raised: *pass-through specific condition — invalid cursor option*.
 - e) No further General Rules of this Subclause are applied.

Conformance Rules

No additional Conformance Rules.

18.9 <allocate cursor statement>

Function

Define a cursor based on a prepared statement for a <cursor specification> or assign a cursor to the ordered set of result sets returned from an SQL-invoked procedure.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert before GR1 If the pass-through flag of the current SQL-session context is True , then an exception condition is raised: *pass-through specific condition — invalid cursor allocation*.

Conformance Rules

No additional Conformance Rules.

18.10 <dynamic open statement>

Function

Associate input dynamic parameters with a <cursor specification> and open the cursor.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert before GR1 If the pass-through flag of the current SQL-session context is *True* , and the current SQL-session context includes an {SQL statement name : ExecutionHandle} pair whose SQL statement name is equivalent to <SQL statement name>, then:
 - a) Let *FSN* be the name of the pass-through foreign server included in the current SQL-session context. Let *WN* be the name of the foreign-data wrapper included in the foreign server descriptor of the foreign server identified by *FSN*. Let *WR* be the foreign-data wrapper identified by *WN*. Let *WRLN* be the name of the library identified in the foreign-data wrapper descriptor of *WR*. Let *EXH* be the ExecutionHandle associated with <SQL statement name>.
 - b) If an <input using clause> is specified, then the General Rules specified in Subclause 18.5, "<input using clause>", for <dynamic open statement> are applied.
 - c) The Open() routine in the library identified by *WRLN* is invoked with *EXH* as argument.
 - d) No further General Rules of this Subclause are applied.

Conformance Rules

No additional Conformance Rules.

18.11 <dynamic fetch statement>

Function

Fetch a row for a cursor declared with a <dynamic declare cursor>.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert before GR1 If the pass-through flag of the current SQL-session context is *True* , and the current SQL-session context includes a {SQL statement name : ExecutionHandle} pair whose SQL statement name is equivalent to <SQL statement name>, then:
 - a) Let *FSN* be the pass-through foreign server name included in the current SQL-session context. Let *WN* be the name of the foreign-data wrapper included in the foreign server descriptor of the foreign server identified by *FSN*. Let *WR* be the foreign-data wrapper identified by *WN*. Let *WRLN* be the name of the library identified in the foreign-data wrapper descriptor of *WR*. Let *EXH* be the ExecutionHandle associated with <SQL statement name>.
 - b) The Iterate() routine in the library identified by *WRLN* is invoked with *EXH* as argument.
 - c) The General Rules of Subclause 18.6, "<output using clause>", are applied to the <dynamic fetch statement> and the current row of *CR* as the retrieved row.
 - d) No further General Rules of this Subclause are applied.

Conformance Rules

No additional Conformance Rules.

18.12 <dynamic close statement>

Function

Close a cursor.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

- 1) Insert before GR1 If the pass-through flag of the current SQL-session context is *True* , and the current SQL-session context includes an {SQL statement name : ExecutionHandle} pair whose SQL statement name is equivalent to <SQL statement name>, then:
 - a) Let *FSN* be the pass-through foreign server name included in the current SQL-session context. Let *WN* be the name of the foreign-data wrapper included in the foreign server descriptor of the foreign server identified by *FSN*. Let *WR* be the foreign-data wrapper identified by *WN*. Let *WRLN* be the name of the library identified in the foreign-data wrapper descriptor of *WR*. Let *EXH* be the ExecutionHandle associated with <SQL statement name>.
 - b) The Close() routine in the library identified by *WRLN* is invoked with *EXH* as argument.
 - c) No further General Rules of this Subclause are applied.

Conformance Rules

No additional Conformance Rules.

(Blank page)

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

19 Embedded SQL

19.1 <embedded SQL Ada program>

Function

Specify an <embedded SQL Ada program>.

Format

```
<Ada derived type specification> ::=
    !! All alternatives from ISO/IEC 9075-5
    | <Ada DATALINK variable>
```

```
<Ada DATALINK variable> ::=
    SQL TYPE IS <datalink type>
```

Syntax Rules

- 1) Insert after SR5)i) The syntax

SQL TYPE IS <datalink type>

shall be replaced by

Interfaces.SQL.CHAR(1..*MDL*)

where *MDL* is the maximum datalink length, in any <Ada DATALINK variable>.

NOTE 46 – The term “maximum datalink length” is defined in Subclause 4.8, “Datalinks”.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

- 1) Without Feature M003, “Datalinks via SQL language”, conforming SQL language shall not specify <Ada DATALINK variable>.

19.2 <embedded SQL C program>

Function

Specify an <embedded SQL C program>.

Format

```
<C derived variable> ::=  
    !! All alternatives from ISO/IEC 9075-5  
    | <C DATALINK variable>  
  
<C DATALINK variable> ::=  
    SQL TYPE IS <datalink type>  
    <C host identifier> [ <C initial value> ]  
    [ { <comma> <C host identifier> [ <C initial value> ] }... ]
```

Syntax Rules

- 1) Insert after SR6)n) The syntax

SQL TYPE IS <datalink type>

shall be replaced by

char[*MDL*]

where *MDL* is the maximum datalink length, in any <C DATALINK variable>.

NOTE 47 – The term “maximum datalink length” is defined in Subclause 4.8, “Datalinks”.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

- 1) Without Feature M003, “Datalinks via SQL language”, conforming SQL language shall not specify <C DATALINK variable>.

19.3 <embedded SQL COBOL program>

Function

Specify an <embedded SQL COBOL program>.

Format

```
<COBOL derived type specification> ::=  
    !! All alternatives from ISO/IEC 9075-5  
    | <COBOL DATALINK variable>
```

```
<COBOL DATALINK variable> ::=  
    [ USAGE [ IS ] ]  
    SQL TYPE IS <datalink type>
```

Syntax Rules

- 1) Insert after SR6)k) The syntax

SQL TYPE IS <datalink type>

shall be replaced by

PIC X(*MDL*).

where *MDL* is the maximum datalink length, in any <COBOL DATALINK variable>.

NOTE 48 – The term “maximum datalink length” is defined in Subclause 4.8, “Datalinks”.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

- 1) Without Feature M003, “Datalinks via SQL language”, conforming SQL language shall not specify <COBOL DATALINK variable>.

19.4 <embedded SQL Fortran program>

Function

Specify an <embedded SQL Fortran program>.

Format

```
<Fortran derived type specification> ::=  
    !! All alternatives from ISO/IEC 9075-5  
    | <Fortran DATALINK variable>
```

```
<Fortran DATALINK variable> ::=  
    SQL TYPE IS <datalink type>
```

Syntax Rules

- 1) Insert after SR6)k) The syntax

SQL TYPE IS <datalink type>

for a given <Fortran host identifier> *fhi* shall be replaced by

CHARACTER *fhi* * *MDL*

where *MDL* is the maximum datalink length, in any <Fortran DATALINK variable>.

NOTE 49 – The term “maximum datalink length” is defined in Subclause 4.8, “Datalinks”.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

- 1) Without Feature M003, “Datalinks via SQL language”, conforming SQL language shall not specify <Fortran DATALINK variable>.

19.5 <embedded SQL MUMPS program>

Function

Specify an <embedded SQL MUMPS program>.

Format

```
<MUMPS derived type specification> ::=  
    !! All alternatives from ISO/IEC 9075-5  
    | <MUMPS DATALINK variable>
```

```
<MUMPS DATALINK variable> ::=  
    SQL TYPE IS <datalink type>
```

Syntax Rules

- 1) Insert after SR9)h) The syntax

SQL TYPE IS <datalink type>

for a given <MUMPS host identifier> *mhi* shall be replaced by

VARCHAR *mhi* *MDL*

where *MDL* is the maximum datalink length, in any <MUMPS DATALINK variable>.

NOTE 50 – The term “maximum datalink length” is defined in Subclause 4.8, “Datalinks”.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

- 1) Without Feature M003, “Datalinks via SQL language”, conforming SQL language shall not specify <MUMPS DATALINK variable>.

19.6 <embedded SQL Pascal program>

Function

Specify an <embedded SQL Pascal program>.

Format

```
<Pascal derived type specification> ::=  
    !! All alternatives from ISO/IEC 9075-5  
    | <Pascal DATALINK variable>
```

```
<Pascal DATALINK variable> ::=  
    SQL TYPE IS <datalink type>
```

Syntax Rules

- 1) Insert after SR5)) The syntax

SQL TYPE IS <datalink type>

shall be replaced by

PACKED ARRAY [1..MDL] OF CHAR

where MDL is the maximum datalink length, in any <Pascal DATALINK variable>.

NOTE 51 – The term “maximum datalink length” is defined in Subclause 4.8, “Datalinks”.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

- 1) Without Feature M003, “Datalinks via SQL language”, conforming SQL language shall not specify <Pascal DATALINK variable>.

19.7 <embedded SQL PL/I program>

Function

Specify an <embedded SQL PL/I program>.

Format

```
<PL/I derived type specification> ::=  
    !! All alternatives from ISO/IEC 9075-5  
    | <PL/I DATALINK variable>
```

```
<PL/I DATALINK variable> ::=  
    SQL TYPE IS <datalink type>
```

Syntax Rules

- 1) Insert after SR5)i) The syntax
SQL TYPE IS <datalink type>

shall be replaced by

CHARACTER(*MDL*)

where *MDL* is the maximum datalink length, in any <PL/I DATALINK variable>.

NOTE 52 – The term “maximum datalink length” is defined in Subclause 4.8, “Datalinks”.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

- 1) Without Feature M003, “Datalinks via SQL language”, conforming SQL language shall not specify <PL/I DATALINK variable>.

(Blank page)

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

20 Call-Level Interface specifications

20.1 <CLI routine>

Format

Describe a generic SQL/CLI routine.

Format

```
<CLI routine> ::=
    !! All alternatives from ISO/IEC 9075-3
    | BuildDataLink
    | GetDataLinkAttr
```

Syntax Rules

Table 14—Abbreviated SQL/CLI generic names

Generic Name	Abbreviation
	<i>All alternatives from ISO/IEC 9075-3</i>
BuildDataLink	BDL
GetDataLinkAttr	GDL

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

20.2 Implicit DESCRIBE USING clause

Function

Specify the rules for an implicit DESCRIBE USING clause.

General Rules

- 1) Insert after GR5)c)iv)11) If TYPE indicates DATALINK, then LENGTH and OCTET_LENGTH are set to the maximum possible length in octets of the datalink.

ISO/IEC 9075-9:2001 (E)

20.2 Implicit DESCRIBE USING clause

- 2) Insert after GR8)d)vi)11) If TYPE indicates DATALINK, then LENGTH and OCTET_LENGTH are set to the maximum possible length in octets of the datalink.

20.2.1 CLI-specific status codes

Table 15—SQLSTATE class and subclass values for SQL/CLI-specific conditions

Category	Condition	Class	Subcondition	Subclass
X	<i>All alternatives from ISO/IEC 9075-3</i> CLI-specific condition	HY	invalid datalink value	093

20.2.2 Description of CLI item descriptor areas

Function

Specify the identifiers, data types and codes for fields used in CLI item descriptor areas.

Syntax Rules

- 1) Insert after SR5)c)xiv) TYPE indicates DATALINK.
- 2) Replaces SR7)c)iv) TYPE indicates DEFAULT, CHARACTER, CHARACTER LARGE OBJECT, BINARY LARGE OBJECT, CHARACTER LARGE OBJECT LOCATOR, BINARY LARGE OBJECT LOCATOR, USER-DEFINED TYPE LOCATOR, DATALINK, REF, INTEGER, SMALLINT, REAL, or DOUBLE PRECISION.
- 3) Insert after SR12)c)x) TYPE indicates DATALINK and one of the following is true:
- 1) NULL is true.
 - 2) DEFERRED is true.
- 4) Replace SR13)c)iv) TYPE indicates DEFAULT, CHARACTER, CHARACTER LARGE OBJECT, BINARY LARGE OBJECT, CHARACTER LARGE OBJECT LOCATOR, BINARY LARGE OBJECT LOCATOR, USER-DEFINED TYPE LOCATOR, DATALINK, INTEGER, SMALLINT, REAL, or DOUBLE PRECISION.

General Rules

Table 16—Codes used for implementation data types in SQL/CLI

Data Type	Code
DATALINK	<i>All alternatives from ISO/IEC 9075-3</i> 70

Table 17—Codes used for application data types in SQL/CLI

Data Type	Code
DATALINK	<i>All alternatives from ISO/IEC 9075-3</i> 70

20.2.3 Other tables associated with CLI

Table 18—Codes used to identify SQL/CLI routines

Generic Name	Code
BuildDataLink	<i>All alternatives from ISO/IEC 9075-3</i> 1029
GetDataLinkAttr	1034

Table 19—Codes and data types for implementation information

Information Type	Code	Data Type
<i>All alternatives from ISO/IEC 9075-3</i> MAXIMUM DATALINK LENGTH	20004	INTEGER

Table 20—Codes used for datalink attributes

Attribute	Code
URL COMPLETE	3
URL PATH	4
URL PATH ONLY	5
URL SCHEME	6
URL SERVER	7
Implementation-defined datalink attribute	<0

ISO/IEC 9075-9:2001 (E)
20.2 Implicit DESCRIBE USING clause

Table 21—Data types of attributes

Attribute	Data type	Values
	<i>All alternatives from ISO/IEC 9075-3</i>	
Datalink attributes		
URL COMPLETE	CHARACTER VARYING(L) ¹	Datalink complete URL
URL PATH	CHARACTER VARYING(L) ¹	Datalink URL path
URL PATH ONLY	CHARACTER VARYING(L) ¹	Datalink URL path only
URL SCHEME	CHARACTER VARYING(L) ¹	Datalink URL schema
URL SERVER	CHARACTER VARYING(L) ¹	Datalink URL server
Implementation-defined datalink attribute	Implementation-defined data type	Implementation-defined value
¹ Where <i>L</i> is an implementation-defined integer not less than the maximum datalink length. (The term “maximum datalink length” is defined in Subclause 4.8, “Datalinks”.)		

Conformance Rules

- 1) Without Feature M002, “Datalinks via SQL/CLI”, conforming SQL language shall not specify BuildDataLink().
- 2) Without Feature M002, “Datalinks via SQL/CLI”, conforming SQL language shall not specify GetDataLinkAttr().

20.3 SQL/CLI data type correspondences

Function

Replaces first paragraph Specify the SQL/CLI data type correspondences for SQL data types and host language types associated with the required parameter mechanisms, as shown in Table 3, "Supported calling conventions of SQL/CLI routines by language", in ISO/IEC 9075-3.

Tables

Table 22—SQL/CLI data type correspondences for Ada

SQL Data Type	Ada Data Type
All alternatives from ISO/IEC 9075-3 DATALINK	All alternatives from ISO/IEC 9075-3 SQL_STANDARD.CHAR, with P'Length of LD ¹
¹ The length LD of the character data type corresponding with SQL data type DATALINK is implementation-defined.	

Table 23—SQL/CLI data type correspondences for C

SQL Data Type	C Data Type
All alternatives from ISO/IEC 9075-3 DATALINK	All alternatives from ISO/IEC 9075-3 char, with length LD ³
³ The length LD of the character data type corresponding with SQL data type DATALINK is implementation-defined.	

Table 24—SQL/CLI data type correspondences for COBOL

SQL Data Type	COBOL Data Type
All alternatives from ISO/IEC 9075-3 DATALINK	All alternatives from ISO/IEC 9075-3 alphanumeric, with length LD ³
³ The length LD of the character data type corresponding with SQL data type DATALINK is implementation-defined.	

Table 25—SQL/CLI data type correspondences for Fortran

SQL Data Type	Fortran Data Type
All alternatives from ISO/IEC 9075-3 DATALINK	All alternatives from ISO/IEC 9075-3 CHARACTER with length LD ²
² The length LD of the character data type corresponding with SQL data type DATALINK is implementation-defined.	

Table 26—SQL/CLI data type correspondences for MUMPS

SQL Data Type	MUMPS Data Type
All alternatives from ISO/IEC 9075-3	All alternatives from ISO/IEC 9075-3

Table 26—SQL/CLI data type correspondences for MUMPS (Cont.)

SQL Data Type	MUMPS Data Type
DATALINK	character

Table 27—SQL/CLI data type correspondences for Pascal

SQL Data Type	Pascal Data Type
<i>All alternatives from ISO/IEC 9075-3</i>	<i>All alternatives from ISO/IEC 9075-3</i>
DATALINK	PACKED ARRAY[1.. <i>LD</i> ²] OF CHAR
² The length <i>LD</i> of the character data type corresponding with SQL data type DATALINK is implementation-defined.	

Table 28—SQL/CLI data type correspondences for PL/I

SQL Data Type	PL/I Data Type
<i>All alternatives from ISO/IEC 9075-3</i>	<i>All alternatives from ISO/IEC 9075-3</i>
DATALINK	CHARACTER VARYING(<i>LD</i>), where <i>LD</i> is implementation-defined

21 SQL/CLI routines

21.1 BuildDataLink

Function

Build a datalink value.

Definition

```
BuildDataLink (
    StatementHandle      IN  INTEGER,
    DataLocation         IN  CHARACTER(L1),
    DataLocationLength   IN  INTEGER,
    DataLink             OUT CHARACTER(L2),
    BufferLength          IN  INTEGER,
    StringLength         OUT INTEGER )
RETURNS SMALLINT
```

where *L1* has a maximum value equal to the implementation-defined maximum length of a variable-length character string and *L2* has a maximum value equal to the implementation-defined maximum length of a datalink.

General Rules

- 1) Let *SH* be the value of *StatementHandle*.
NOTE 53 – *SH* is used only if *BuildDataLink* issues a completion or exception condition.
- 2) Let *DL* be the datalink value whose File Reference is *DataLocation*.
NOTE 54 – *File Reference* is defined in Subclause 4.8, "Datalinks".
- 3) Let *DLL* be the length in octets of *DL*.
- 4) If *DLL* is greater than the maximum datalink length, then an exception condition is raised:
CLI-specific condition — invalid datalink value.
NOTE 55 – The term "maximum datalink length" is defined in Subclause 4.8, "Datalinks".
- 5) Apply the General Rules of Subclause 5.9, "Character string retrieval", in ISO/IEC 9075-3 with *DataLink*, *DL*, *BufferLength*, and *StringLength* as *TARGET*, *VALUE*, *OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.

Conformance Rules

- 1) Without Feature M002, "Datalinks via SQL/CLI", conforming SQL language shall not specify *BuildDataLink*().

21.2 GetDataLinkAttr

Function

Retrieve the value of a datalink attribute.

Definition

```
GetDataLinkAttr (
    StatementHandle      IN  INTEGER,
    Attribute             IN  SMALLINT,
    DataLink             IN  CHARACTER(L),
    DataLinkLength       IN  INTEGER,
    Value                OUT  ANY,
    BufferLength          IN  INTEGER,
    StringLength         OUT  INTEGER )
RETURNS SMALLINT
```

where L has a maximum value equal to the implementation-defined maximum length of a datalink.

General Rules

- 1) Let SH be the value of StatementHandle.
NOTE 56 – SH is used only if GetDataLinkAttr issues a completion or exception condition.
- 2) Let A be the value of Attribute.
- 3) If A is not one of the code values in Table 20, “Codes used for datalink attributes”, then an exception condition is raised: *CLI-specific condition — invalid attribute identifier*.
- 4) Let DLL be the value of DataLinkLength.
- 5) Case:
 - a) If DLL is not negative, then let L be DLL .
 - b) Otherwise, an exception condition is raised: *CLI-specific condition — invalid string length or buffer length*.
- 6) Case:
 - a) If L is zero, then an exception condition is raised: *CLI-specific condition — invalid string length or buffer length*.
 - b) Otherwise, let DL be the first L octets of DataLink.
- 7) Case:
 - a) If L is zero, then an exception condition is raised: *CLI-specific condition — invalid string length or buffer length*.

- b) Otherwise:
- i) Let N be the number of whole characters in the first L octets of DataLink and let NO be the number of octets occupied by those N characters. If $NO \neq L$, then an exception condition is raised: *invalid cursor name*.
 - ii) Otherwise, let DL be the first L octets of Datalink.
- 8) Let ML be the implementation-defined maximum length in characters of a datalink value.
- 9) If DL is not a valid datalink value, then an exception condition is raised: *CLI-specific condition — invalid datalink value*.
- 10) Let BL be the value of BufferLength.
- 11) If A specifies an implementation-defined datalink attribute, then
- Case:
- a) If the data type for the datalink attribute is specified as INTEGER in Table 21, "Data types of attributes", then Value is set to the value of the implementation-defined datalink attribute and no further General Rules of this Subclause are applied.
 - b) Otherwise:
 - i) Let AV be the value of the implementation-defined datalink attribute.
 - ii) The General Rules of Subclause 5.9, "Character string retrieval", in ISO/IEC 9075-3 are applied with Value, AV , BL , and StringLength as *TARGET*, *VALUE*, *OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.
- 12) Case:
- a) If A indicates URL COMPLETE, then AV is set to the value of the File Reference of DL .
 - b) If A indicates URL PATH, then AV is set to the value of the path of DL , possibly combined with an access token under the General Rules of Subclause 6.4, "<string value function>".
 - c) If A indicates URL PATH ONLY, then AV is set to the value of the path of DL .
 - d) If A indicates URL SCHEME, then AV is set to the value of the scheme of DL .
 - e) If A indicates URL SERVER, then AV is set to the value of the host of DL .
- NOTE 57 – "host", "scheme", and "path" are defined in Subclause 6.5, "<datalink value function>".
- 13) The General Rules of Subclause 5.9, "Character string retrieval", in ISO/IEC 9075-3 are applied with Value, AV , BL , and StringLength as *TARGET*, *VALUE*, *OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.

Conformance Rules

- 1) Without Feature M002, "Datalinks via SQL/CLI", conforming SQL language shall not specify GetDataLinkAttr().

21.3 GetInfo

Function

Get information about the implementation.

Definition

No additional Definition items.

General Rules

- 1)

Insert into the dashed list in GR10)a)
--

— MAXIMUM DATALINK LENGTH

Conformance Rules

- 1) Without Feature M002, “Datalinks via SQL/CLI”, the value of InfoType shall not indicate MAXIMUM DATALINK LENGTH.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

22 SQL/MED common specifications

22.1 Description of foreign-data wrapper item descriptor areas

Function

Specify the identifiers, data types and codes for fields used in foreign-data wrapper item descriptor areas.

Syntax Rules

- 1) A foreign-data wrapper item descriptor area consists of the fields specified in Table 5, "Fields in foreign-data wrapper descriptor areas".
- 2) Let *HL1* be the standard host language in which the SQL-server is written and let *HL2* be the standard host language in which the foreign-data wrapper is written.
- 3) Given a foreign-data wrapper item descriptor area *IDA* in which the value of LEVEL is some value *N*, the immediately subordinate descriptor areas of *IDA* are those foreign-data wrapper item descriptor areas in which the value of LEVEL is *N+1* and whose position in the foreign-data wrapper descriptor area follows that of *IDA* and precedes that of any foreign-data wrapper item descriptor area in which the value of LEVEL is less than *N+1*. The subordinate descriptor areas of *IDA* are those foreign-data wrapper item descriptor areas that are immediately subordinate descriptor areas of *IDA* or that are subordinate descriptor areas of a foreign-data wrapper item descriptor area that is immediately subordinate to *IDA*.
- 4) Given a data type *DT* and its descriptor *DE*, the immediately subordinate descriptors of *DE* are defined to be
Case:
 - a) If *DT* is ROW, then the field descriptors of the fields of *DT*. The *i*-th immediately subordinate descriptor is the descriptor of the *i*-th field of *DT*.
 - b) If *DT* is ARRAY, then the descriptor of the associated element type of *DT*. The subordinate descriptors of *DE* are those descriptors that are immediately subordinate descriptors of *DE* or that are subordinate descriptors of a descriptor that is immediately subordinate to *DE*.
- 5) Given a descriptor *DE*, let *SDE_j* represent its *j*-th immediately subordinate descriptor. There is an implied ordering of the subordinate descriptors of *DE*, such that:
 - a) *SDE₁* is in the first ordinal position.
 - b) The ordinal position of *SDE_{j+1}* is *K+NS+1*, where *K* is the ordinal position of *SDE_j* and *NS* is the number of subordinate descriptors of *SDE_j*. The implicitly ordered subordinate descriptors of *SDE_j* occupy contiguous ordinal positions starting at position *K+1*.
- 6) Let *HL* be the standard programming language of the invoking SQL-server. Let *operative data type correspondence table* be the data type correspondence table for *HL* as specified in Subclause 20.3, "SQL/CLI data type correspondences". Refer to the two columns of the operative data type correspondence table as the *SQL data type column* and the *host data type column*.

ISO/IEC 9075-9:2001 (E)

22.1 Description of foreign-data wrapper item descriptor areas

- 7) A foreign-data wrapper item descriptor area *IDA* in a foreign-data wrapper descriptor area that is a server row descriptor or a server parameter descriptor is *consistent* if and only if all of the following are true:
- a) TYPE indicates ROW or is one of the code values in Table 17, "Codes used for application data types in SQL/CLI".
 - b) Exactly one of the following is true:
 - i) TYPE indicates NUMERIC, and PRECISION and SCALE are valid precision and scale values for the NUMERIC data type.
 - ii) TYPE indicates DECIMAL, and PRECISION and SCALE are valid precision and scale values for the DECIMAL data type.
 - iii) TYPE indicates FLOAT, and PRECISION is a valid precision value for the FLOAT data type.
 - iv) TYPE indicates DEFAULT, CHARACTER, CHARACTER LARGE OBJECT, BINARY LARGE OBJECT, CHARACTER LARGE OBJECT LOCATOR, BINARY LARGE OBJECT LOCATOR, USER-DEFINED TYPE LOCATOR, REF, BOOLEAN, DATALINK, INTEGER, SMALLINT, REAL, or DOUBLE PRECISION.
 - v) TYPE indicates ROW and, where *N* is the value of the DEGREE field, there are exactly *N* immediately subordinate descriptor areas of *IDA*, and those item descriptor areas are valid.
 - vi) TYPE indicates ARRAY LOCATOR, there is exactly 1 (one) immediately subordinate descriptor area of *IDA*, and that item descriptor area is valid.
 - vii) TYPE indicates an implementation-defined data type.
- 8) Let *IDA* be a foreign-data wrapper item descriptor area in a server parameter descriptor.
- 9) If the value of INDICATOR is the appropriate 'Code' for SQL NULL DATA in Table 26, "Miscellaneous codes used in CLI", in ISO/IEC 9075-3, then NULL is true for *IDA*. Otherwise, NULL is false for *IDA*.
- 10) *IDA* is *valid* if and only if:
- a) TYPE is one of the code values in Table 17, "Codes used for application data types in SQL/CLI", or TYPE indicates ROW.
 - b) If LEVEL is 0 (zero) for *IDA*, then let *TLC* be the value of TOP_LEVEL_COUNT in the server parameter descriptor associated with *IDA*. *IDA* shall be one of exactly *TLC* item descriptor areas in the server parameter descriptor.
 - c) One of the following is true:

Case:

 - i) TYPE indicates NUMERIC, and PRECISION and SCALE are valid precision and scale values for the NUMERIC data type.
 - ii) TYPE indicates DECIMAL, and PRECISION and SCALE are valid precision and scale values for the DECIMAL data type.

22.1 Description of foreign-data wrapper item descriptor areas

- iii) TYPE indicates FLOAT, and PRECISION is a valid precision value for the FLOAT data type.
- iv) TYPE indicates INTEGER, SMALLINT, REAL, or DOUBLE PRECISION.
- v) TYPE indicates CHARACTER, CHARACTER LARGE OBJECT, or BINARY LARGE OBJECT, and the value V of OCTET_LENGTH is greater than zero, and

Case:

- 1) If *HL1* and *HL2* are both pointer-supporting languages, then the number of characters wholly contained in the first V octets of the host variable addressed by DATA_POINTER is a valid length value for a CHARACTER, CHARACTER LARGE OBJECT, or BINARY LARGE OBJECT data type, as indicated by TYPE.
- 2) Otherwise, the number of characters wholly contained in the first V octets of DATA is a valid length value for a CHARACTER, CHARACTER LARGE OBJECT, or BINARY LARGE OBJECT data type, as indicated by TYPE.

- vi) TYPE indicates REF.
 - vii) TYPE indicates CHARACTER LARGE OBJECT LOCATOR, BINARY LARGE OBJECT LOCATOR, or USER-DEFINED TYPE LOCATOR.
 - viii) TYPE indicates ROW and, where *N* is the value of the DEGREE field, there are exactly *N* immediately subordinate foreign-data wrapper descriptor areas of *IDA*, and those foreign-data wrapper item descriptor areas are valid.
 - ix) TYPE indicates ARRAY LOCATOR, there is exactly 1 (one) immediately subordinate descriptor area of *IDA*, and that subordinate descriptor area is valid.
 - x) TYPE indicates DATALINK.
 - xi) TYPE indicates an implementation-defined data type.
- d) Case:
- i) If *HL1* and *HL2* are both pointer-supporting languages, then one of the following is true:
 - 1) DATA_POINTER is zero and NULL is true.
 - 2) DATA_POINTER is not zero and the value of the host variable addressed by DATA_POINTER is a valid value of the data type indicated by TYPE.
 - ii) Otherwise, DATA is a valid value of the data type indicated by TYPE.
- 11) A foreign-data wrapper item descriptor area *IDA* in a server row descriptor is valid if and only if:
- a) TYPE is one of the code values in Table 17, "Codes used for application data types in SQL/CLI", or TYPE indicates ROW.
 - b) If LEVEL is 0 (zero) for *IDA*, then let *TLC* be the value of TOP_LEVEL_COUNT in the server row descriptor associated with *IDA*. *IDA* shall be one of exactly *TLC* foreign-data wrapper item descriptor areas in the server row descriptor.

ISO/IEC 9075-9:2001 (E)

22.1 Description of foreign-data wrapper item descriptor areas

c) One of the following is true:

Case:

- i) TYPE indicates NUMERIC, and PRECISION and SCALE are valid precision and scale values for the NUMERIC data type.
- ii) TYPE indicates DECIMAL, and PRECISION and SCALE are valid precision and scale values for the DECIMAL data type.
- iii) TYPE indicates FLOAT, and PRECISION is a valid precision value for the FLOAT data type.
- iv) TYPE indicates CHARACTER, CHARACTER LARGE OBJECT, BINARY LARGE OBJECT, CHARACTER LARGE OBJECT LOCATOR, BINARY LARGE OBJECT LOCATOR, USER-DEFINED TYPE LOCATOR, REF, BOOLEAN, DATA LINK, INTEGER, SMALLINT, REAL, or DOUBLE PRECISION.
- v) TYPE indicates ROW and, where N is the value of the DEGREE field, there are exactly N immediately subordinate descriptor areas of *IDA*, and those subordinate descriptor areas are valid.
- vi) TYPE indicates ARRAY LOCATOR, there is exactly 1 (one) immediately subordinate descriptor area of *IDA*, and that subordinate descriptor area is valid.
- vii) TYPE indicates an implementation-defined data type.

General Rules

None.

Conformance Rules

None.

22.2 Implicit cursor

Function

Specify the rules for an implicit DECLARE CURSOR and OPEN statement.

General Rules

- 1) Let *AE* be an *ALLOCATED FDW-EXECUTION* specified in an application of this Subclause.
- 2) If there is no cursor effectively associated with *AE*, then a cursor is effectively associated with *AE* and the cursor name associated with *AE* becomes the name of the cursor.
- 3) Let *CT* be 'NO SCROLL'.
- 4) Let *CS* be 'ASENSITIVE'.
- 5) Let *CH* be 'WITHOUT HOLD'.
- 6) Let *SS* be the SQL-statement that is effectively associated with *AE*.
- 7) Let *CN* be the name of the cursor effectively associated with *AE* and let *CR* be the following <declare cursor>:

DECLARE *CN CS CT CURSOR CH FOR SS*

- 8) Cursor *CN* is effectively opened in the following steps:
 - a) A copy of *SS* is effectively created in which:
 - i) Each <dynamic parameter specification> is replaced by the value of the corresponding dynamic parameter.
 - ii) Each <value specification> generally contained in *SS* that is *CURRENT_USER*, *CURRENT_ROLE*, *CURRENT_PATH*, *SESSION_USER*, or *SYSTEM_USER* is effectively replaced by the value resulting from evaluation of *CURRENT_USER*, *CURRENT_ROLE*, *CURRENT_PATH*, *SESSION_USER*, or *SYSTEM_USER*, respectively, with all such evaluations effectively done at the same instant in time.
 - iii) Each <datetime value function> generally contained in *SS* is effectively replaced by the value resulting from evaluation of that <datetime value function>, with all such evaluations effectively done at the same instant in time.
 - b) Let *T* be the table specified by the copy of *SS*.
 - c) A table descriptor for *T* is effectively created.
 - d) The General Rules of Subclause 14.1, "<declare cursor>", in ISO/IEC 9075-2, are effectively applied to *CR*.
 - e) Cursor *CN* is effectively placed in the open state and its position is before the first row of *T*.

ISO/IEC 9075-9:2001 (E)
22.2 Implicit cursor

Conformance Rules

None.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

22.3 Implicit DESCRIBE INPUT USING clause

Function

Populate a specified descriptor area with information about the input values required to execute a foreign-data request.

General Rules

- 1) Let *S* and *DESC* be a *SOURCE* and a *DESCRIPTOR* specified in the rules of this Subclause.
- 2) Let *HL* be the standard programming language of the invoking SQL-server.
- 3) The value of *DYNAMIC_FUNCTION* and *DYNAMIC_FUNCTION_CODE* in *DESC* are respectively a character string representation of the foreign-data request and a numeric code that identifies the foreign-data request, and are set to the value of the 'Identifier' and 'Code' columns, respectively, of the row in Table 38, "SQL-statement codes", that identifies in the 'SQL-statement' column the foreign-data request.
- 4) A descriptor for the <dynamic parameter specification>s for the foreign-data request is stored in *DESC* as follows:
 - a) Let *D* be the number of <dynamic parameter specification>s in *S*. Let *NS_i*, 1 (one) ≤ *i* ≤ *D*, be the number of subordinate descriptors of the descriptor for the *i*-th input dynamic parameter.
 - b) *TOP_LEVEL_COUNT* is set to *D*. If *D* is 0 (zero), then let *TD* be 0 (zero); otherwise, let *TD* be $D + \sum_{i=1}^D (NS_i)$. *COUNT* is set to *TD*.
NOTE 58 – The *KEY_TYPE* field is not relevant in this case.
 - c) If *TD* is zero, then no item descriptor areas are set. Otherwise, the first *TD* item descriptor areas are set so that the *i*-th item descriptor area contains a descriptor of the *j*-th <dynamic parameter specification> such that:
 - i) The descriptor for the first such <dynamic parameter specification> is assigned to the first descriptor area.
 - ii) The descriptor for the *j*+1-th <dynamic parameter specification> is assigned to the *i*+*NS_j*+1-th item descriptor area.
 - iii) The implicitly ordered subordinate descriptors for the *j*-th <dynamic parameter specification>, if any, are assigned to contiguous item descriptor areas starting at the *i*+1-th item descriptor area.
 - d) The descriptor of a <dynamic parameter specification> consists of values for *LEVEL*, *TYPE*, *NULLABLE*, *NAME*, *UNNAMED*, *PARAMETER_MODE*, *PARAMETER_ORDINAL_POSITION*, *PARAMETER_SPECIFIC_CATALOG*, *PARAMETER_SPECIFIC_SCHEMA*, *PARAMETER_SPECIFIC_NAME*, and other fields depending on the value of *TYPE* as described below. Those fields and fields that are not applicable for a particular value of *TYPE* are set to implementation-dependent values. The *DATA*, *DATA_POINTER*, *INDICATOR*, *OCTET_LENGTH*, *RETURNED_CARDINALITY*, and *KEY_MEMBER* fields are not relevant in this case.
 - i) If the item descriptor area is set to a descriptor that is immediately subordinate to another whose *LEVEL* value is some value *k*, then *LEVEL* is set to *k*+1; otherwise,

LEVEL is set to 0 (zero).

- ii) TYPE is set to a code as shown in Table 7, "Codes used for implementation data types in SQL/CLI", in ISO/IEC 9075-3, indicating the data type of the <dynamic parameter specification> or subordinate descriptor.

- iii) NULLABLE is set to 1 (one).

NOTE 59 – This indicates that the <dynamic parameter specification> can have the null value.

- iv) KEY_MEMBER is set to 0 (zero).

- v) UNNAMED is set to 1 (one) and NAME is set to an implementation-dependent value.

- vi) Case:

- 1) If TYPE indicates a <character string type>, then: LENGTH is set to the length or maximum length in characters of the character string and OCTET_LENGTH is set to the maximum possible length in octets of the character string; CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, and CHARACTER_SET_NAME are set to the <character set name> of the character string's character set; COLLATION_CATALOG, COLLATION_SCHEMA, and COLLATION_NAME are set to the <collation name> of the character string's collation.
- 2) If TYPE indicates a <bit string type>, then LENGTH is set to the length or maximum length in bits of the bit string and OCTET_LENGTH is set to the maximum possible length in octets of the bit string.
- 3) If TYPE indicates a <binary string type>, then LENGTH is set to the maximum length in octets of the binary string.
- 4) If TYPE indicates an <exact numeric type>, then PRECISION and SCALE are set to the precision and scale of the exact numeric.
- 5) If TYPE indicates an <approximate numeric type>, then PRECISION is set to the precision of the approximate numeric.
- 6) If TYPE indicates a <datetime type>, then LENGTH is set to the length in positions of the datetime type, DATETIME_INTERVAL_CODE is set to a code as specified in Table 9, "Codes associated with datetime data types in SQL/CLI", in ISO/IEC 9075-3, to indicate the specific datetime data type and PRECISION is set to the <time precision> or <timestamp precision> if either is applicable.
- 7) If TYPE indicates INTERVAL, then LENGTH is set to the length in positions of the interval type, DATETIME_INTERVAL_CODE is set to a code as specified in Table 10, "Codes associated with <interval qualifier> in SQL/CLI", in ISO/IEC 9075-3, to indicate the specific <interval qualifier>, DATETIME_INTERVAL_PRECISION is set to the <interval leading field precision>, and PRECISION is set to the <interval fractional seconds precision>, if applicable.
- 8) If TYPE indicates REF, then LENGTH and OCTET_LENGTH are set to the length in octets of the <reference type>, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are set to the <user-defined type name> of the <reference type>, and SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME are set to the qualified name of the referenceable base table.

22.3 Implicit DESCRIBE INPUT USING clause

- 9) If TYPE indicates USER-DEFINED TYPE, then USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are set to the <user-defined type name> of the user-defined type. SPECIFIC_TYPE_CATALOG, SPECIFIC_TYPE_SCHEMA, and SPECIFIC_TYPE_NAME are set to the <user-defined type name> of the user-defined type and CURRENT_TRANSFORM_GROUP is set to the CURRENT_TRANSFORM_GROUP_FOR_TYPE <user-defined type name>.
- 10) If TYPE indicates ROW, then DEGREE is set to the degree of the row type.
- 11) If TYPE indicates ARRAY, then CARDINALITY is set to the cardinality of the array type.
- 12) If TYPE indicates DATALINK, then LENGTH and OCTET_LENGTH are set to the maximum possible length in octets of the datalink.

Conformance Rules

None.

22.4 Implicit DESCRIBE OUTPUT USING clause

Function

Populate a specified descriptor area with information about the values returned by an execution of a foreign-data request.

General Rules

- 1) Let *S* and *DESC* be a *SOURCE* and a *DESCRIPTOR* specified in the rules of this Subclause.
- 2) Let *HL* be the standard programming language of the invoking SQL-server.
- 3) The value of *DYNAMIC_FUNCTION* and *DYNAMIC_FUNCTION_CODE* in *DESC* are respectively a character string representation of the foreign-data request and a numeric code that identifies the foreign-data request, and are set to the value of the 'Identifier' and 'Code' columns, respectively, of the row in Table 38, "SQL-statement codes", that identifies in the 'SQL-statement' column the foreign-data request.
- 4) A representation of the column descriptors of the <select list> columns for the foreign-data request is stored in *DESC* as follows:
 - a) Case:
 - i) If there is a select source associated with *DESC*, then:
 - 1) Let *TBL* be the table defined by *S* and let *D* be the degree of *TBL*. Let NS_i , $1 \text{ (one)} \leq i \leq D$, be the number of subordinate descriptors of the descriptor for the *i*-th column of *T*.
 - 2) *TOP_LEVEL_COUNT* is set to *D*. If *D* is 0 (zero), then let *TD* be 0 (zero); otherwise, let *TD* be $D + \sum_{i=1}^D (NS_i)$. *COUNT* is set to *TD*.
 - 3) Case:
 - A) If some subset of *SL* is the primary key of *TBL*, then *KEY_TYPE* is set to 1 (one).
 - B) If some subset of *SL* is the preferred key of *TBL*, then *KEY_TYPE* is set to 2.
 - C) Otherwise, *KEY_TYPE* is set to 0 (zero).
 - ii) Otherwise:
 - 1) Let *D* be 0 (zero). Let *TD* be 0 (zero).
 - 2) *KEY_TYPE* is set to 0 (zero).
 - b) If *TD* is zero, then no item descriptor areas are set. Otherwise, the first *TD* item descriptor areas are set so that the *i*-th item descriptor area contains a descriptor of the *j*-th column such that:
 - i) The descriptor for the first such column is assigned to the first descriptor area.
 - ii) The descriptor for the *j*+1-th column is assigned to the *i*+*NS_j*+1-th item descriptor area.

22.4 Implicit DESCRIBE OUTPUT USING clause

- iii) The implicitly ordered subordinate descriptors for the j -th column, if any, are assigned to contiguous item descriptor areas starting at the $i+1$ -th item descriptor area.
- c) The descriptor of a column consists of values for LEVEL, TYPE, NULLABLE, NAME, UNNAMED, KEY_MEMBER, and other fields depending on the value of TYPE as described below. Those fields and fields that are not applicable for a particular value of TYPE are set to implementation-dependent values. The DATA, DATA_POINTER, INDICATOR, and OCTET_LENGTH fields are not relevant in this case.
 - i) If the item descriptor area is set to a descriptor that is immediately subordinate to another whose LEVEL value is some value k , then LEVEL is set to $k+1$; otherwise, LEVEL is set to 0 (zero).
 - ii) TYPE is set to a code as shown in Table 7, "Codes used for implementation data types in SQL/CLI", in ISO/IEC 9075-3, indicating the data type of the column or subordinate descriptor.
 - iii) Case:
 - 1) If the value of LEVEL is 0 (zero), then:
 - A) If the resulting column is possibly nullable, then NULLABLE is set to 1 (one); otherwise NULLABLE is set to 0 (zero).
 - B) If the column name is implementation-dependent, then NAME is set to the implementation-dependent name of the column and UNNAMED is set to 1 (one); otherwise, NAME is set to the <derived column> name for the column and UNNAMED is set to 0 (zero).
 - C) Case:
 - I) If a <select list> column C is a member of a primary or preferred key of TBL , then KEY_MEMBER is set to 1 (one).
 - II) Otherwise, KEY_MEMBER is set to 0 (zero).
 - 2) Otherwise:
 - A) NULLABLE is set to 1 (one).
 - B) Case:
 - I) If the item descriptor area describes a field of a row, then Case:
 - 1) If the name of the field is implementation-dependent, then NAME is set to the implementation-dependent name of the field and UNNAMED is set to 1 (one).
 - 2) Otherwise, NAME is set to the name of the field and UNNAMED is set to 0 (zero).
 - II) Otherwise, UNNAMED is set to 1 (one) and NAME is set to an implementation-dependent value.
 - C) KEY_MEMBER is set to 0 (zero).

iv) Case:

- 1) If TYPE indicates a <character string type>, then: LENGTH is set to the length or maximum length in characters of the character string and OCTET_LENGTH is set to the maximum possible length in octets of the character string; CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, and CHARACTER_SET_NAME are set to the <character set name> of the character string's character set; COLLATION_CATALOG, COLLATION_SCHEMA, and COLLATION_NAME are set to the <collation name> of the character string's collation.
- 2) If TYPE indicates a <bit string type>, then LENGTH is set to the length or maximum length in bits of the bit string and OCTET_LENGTH is set to the maximum possible length in octets of the bit string.
- 3) If TYPE indicates a <binary string type>, then LENGTH is set to the maximum length in octets of the binary string.
- 4) If TYPE indicates an <exact numeric type>, then PRECISION and SCALE are set to the precision and scale of the exact numeric.
- 5) If TYPE indicates an <approximate numeric type>, then PRECISION is set to the precision of the approximate numeric.
- 6) If TYPE indicates a <datetime type>, then LENGTH is set to the length in positions of the datetime type, DATETIME_INTERVAL_CODE is set to a code as specified in Table 9, "Codes associated with datetime data types in SQL/CLI", in ISO/IEC 9075-3, to indicate the specific datetime data type and PRECISION is set to the <time precision> or <timestamp precision> if either is applicable.
- 7) If TYPE indicates INTERVAL, then LENGTH is set to the length in positions of the interval type, DATETIME_INTERVAL_CODE is set to a code as specified in Table 10, "Codes associated with <interval qualifier> in SQL/CLI", in ISO/IEC 9075-3, to indicate the specific <interval qualifier>, DATETIME_INTERVAL_PRECISION is set to the <interval leading field precision>, and PRECISION is set to the <interval fractional seconds precision>, if applicable.
- 8) If TYPE indicates REF, then LENGTH and OCTET_LENGTH are set to the length in octets of the <reference type>, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are set to the <user-defined type name> of the <reference type>, and SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME are set to the qualified name of the referenceable base table.
- 9) If TYPE indicates USER-DEFINED TYPE, then USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, and USER_DEFINED_TYPE_NAME are set to the <user-defined type name> of the user-defined type. SPECIFIC_TYPE_CATALOG, SPECIFIC_TYPE_SCHEMA, and SPECIFIC_TYPE_NAME are set to the <user-defined type name> of the user-defined type and CURRENT_TRANSFORM_GROUP is set to the CURRENT_TRANSFORM_GROUP_FOR_TYPE <user-defined type name>.
- 10) If TYPE indicates ROW, then DEGREE is set to the degree of the row type.
- 11) If TYPE indicates ARRAY, then CARDINALITY is set to the cardinality of the array type.

22.4 Implicit DESCRIBE OUTPUT USING clause

- 12) If TYPE indicates DATALINK, then LENGTH and OCTET_LENGTH are set to the maximum possible length in octets of the datalink.

Conformance Rules

None.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

22.5 Implicit EXECUTE USING and OPEN USING clauses

Function

Specify the rules for an implicit EXECUTE USING clause and an implicit OPEN USING clause.

General Rules

- 1) Let *T* and *AE* be a *TYPE* and *ALLOCATED FWD-EXECUTION* specified in the rules of this Subclause.
- 2) Let *HL1* be the standard host language in which the SQL-server is written and let *HL2* be the standard host language in which the foreign-data wrapper is written.
- 3) Let *WPD* and *SPD* be the wrapper parameter descriptor and server parameter descriptor, respectively, for *AE*.
- 4) *WPD* and *SPD* describe the <dynamic parameter specification>s and <dynamic parameter specification> values, respectively, for the statement being executed. Let *NSPD* be the value of COUNT for *SPD* and let *NWPD* be the value of COUNT for *WPD*.
 - a) If *NSPD* is less than zero, then an exception condition is raised: *dynamic SQL error — invalid descriptor count*.
 - b) Let *AD* be the minimum of *NSPD* and *NWPD*.
 - c) For each of the first *AD* item descriptor areas of *SPD*, if *TYPE* indicates DEFAULT, then:
 - i) Let *TP*, *P*, and *SC* be the values of the *TYPE*, *PRECISION*, and *SCALE* fields, respectively, for the corresponding item descriptor area of *WPD*.
 - ii) The data type, precision, and scale of the described <dynamic parameter specification> value (or part thereof, if the item descriptor area is a subordinate descriptor) are set to *TP*, *P*, and *SC*, respectively, for the purposes of this invocation only.
 - d) If the first *AD* item descriptor areas of *SPD* are not valid as specified in Subclause 22.1, "Description of foreign-data wrapper item descriptor areas", then an exception condition is raised: *dynamic SQL error — using clause does not match dynamic parameter specifications*.
 - e) For the first *AD* item descriptor areas in *SPD*:
 - i) If the number of item descriptor areas in which the value of *LEVEL* is 0 (zero) is not *NWPD*, then an exception condition is raised: *dynamic SQL error — using clause does not match dynamic parameter specifications*.
 - ii) If all of the following are true, then an exception condition is raised: *dynamic SQL error — using clause does not match dynamic parameter specifications*.
 - 1) The value of *INDICATOR* is not negative.
 - 2) Either of the following is true:
 - A) *TYPE* does not indicate ROW and the item descriptor area is not subordinate to an item descriptor area for which the value of *INDICATOR* is not negative.

22.5 Implicit EXECUTE USING and OPEN USING clauses

B) TYPE indicates ARRAY or ARRAY LOCATOR.

C) Case:

- I) If *HL1* and *HL2* are both pointer-supporting languages, then the value of the host variable addressed by *DATA_POINTER* is not a valid value of the data type represented by the item descriptor area.
- II) Otherwise, the value of *DATA* is not a valid value of the data type represented by the item descriptor area.

f) Let *IDA* be the *i*-th item descriptor area of *SPD* whose *LEVEL* value is 0 (zero). Let *SDT* be the data type represented by *IDA*. The associated value of *IDA* denoted by *SV*, is defined as follows.

Case:

- i) If *NULL* is true for *IDA*, then *SV* is the null value.
- ii) If *TYPE* indicates *ROW*, then *SV* is a row whose type is *SDT* and whose field values are the associated values of the immediately subordinate descriptor areas of *IDA*.
- iii) Otherwise:
 - 1) Case:
 - A) If *HL1* and *HL2* are both pointer-supporting languages, then let *V* be the value of the host variable addressed by *DATA_POINTER*.
 - B) Otherwise, let *V* be the value of *DATA*.
 - 2) Case:
 - A) If *TYPE* indicates *CHARACTER*, then let *Q* be the value of *OCTET_LENGTH* and let *L* be the number of characters wholly contained in the first *Q* octets of *V*.
 - B) Otherwise, let *L* be zero.
 - 3) Let *SV* be *V* with effective data type *SDT*, as represented by the length value *L* and by the values of the *TYPE*, *PRECISION*, and *SCALE* fields.
- g) Let *TDT* be the effective data type of the *i*-th parameter as represented by the values of the *TYPE*, *LENGTH*, *PRECISION*, *SCALE*, *DATETIME_INTERVAL_CODE*, *DATETIME_INTERVAL_PRECISION*, *CHARACTER_SET_CATALOG*, *CHARACTER_SET_SCHEMA*, *CHARACTER_SET_NAME*, *USER_DEFINED_TYPE_CATALOG*, *USER_DEFINED_TYPE_SCHEMA*, *USER_DEFINED_TYPE_NAME*, *SCOPE_CATALOG*, *SCOPE_SCHEMA*, and *SCOPE_NAME* fields in the *i*-th item descriptor area of *WPD* for which the *LEVEL* value is 0 (zero), and all its subordinate descriptor areas.
- h) If *SDT* is an array locator data type, then let *TV* be the value *SV*.
- i) If *SDT* and *TDT* are predefined data types, then

Case:

 - i) If *SDT* and *TDT* are binary string data types or bit string data types, then the <cast specification>

ISO/IEC 9075-9:2001 (E)

22.5 Implicit EXECUTE USING and OPEN USING clauses

CAST (*SV* AS *TDT*)

is effectively performed and the result is the value *TV* is the *i*-th parameter.

- ii) If *SDT* and *TDT* are numeric data types, then the <cast specification>

CAST (*SV* AS *TDT*)

is effectively performed and the result is the value *TV* is the *i*-th parameter.

- iii) Otherwise, the <cast specification>

CAST (*SV* AS *TDT*)

is effectively performed and the result is the value *TV* is the *i*-th parameter.

Conformance Rules

None.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

22.6 Implicit FETCH USING clause

Function

Specify the rules for an implicit FETCH USING clause.

General Rules

- 1) Let *OE* be an *OPENED FDW-EXECUTION* specified in the rules of this Subclause.
- 2) Let *HL1* be the standard host language in which the SQL-server is written and let *HL2* be the standard host language in which the foreign-data wrapper is written.
- 3) Case:
 - a) If the *PASSTHROUGH* flag associated with *OE* is True , then let *RD* be the wrapper row descriptor associated with *OE*.
 - b) Otherwise, let *RD* be the table reference descriptor associated with *OE*.
- 4) Let *SRD* be the server row descriptor associated with *OE*.
- 5) *RD* and *SRD* describe the <select list> columns and <target specification>s, respectively, for the column values that are to be retrieved.
 - a) Let *AD* be the value of the *COUNT* field of *SRD*. If *AD* is less than zero, then an exception condition is raised: *dynamic SQL error — invalid descriptor count*.
 - b) Case:
 - i) If *HL1* and *HL2* are both pointer-supporting languages, then for each item descriptor area in *SRD* whose *LEVEL* is 0 (zero) in the first *AD* item descriptor areas of *SRD*, and for all of their subordinate descriptor areas, refer to a <target specification> whose corresponding item descriptor areas have a non-zero *DATA_POINTER* as a *bound target* and refer to the corresponding <select list> column as a *bound column*.
 - ii) Otherwise, for each item descriptor area in *SRD* whose *LEVEL* is 0 (zero) in the first *AD* item descriptor areas of *SRD*, and for all of their subordinate descriptor areas, refer to a <target specification> as a *bound target* and refer to the corresponding <select list> column as a *bound column*.
 - c) If any item descriptor area corresponding to a bound target in the first *AD* item descriptor areas of *SRD* is not valid as specified in Subclause 22.1, "Description of foreign-data wrapper item descriptor areas", then an exception condition is raised: *dynamic SQL error — using clause does not match target specifications*.
 - d) Let *SDT* be the effective data type of the *i*-th bound column as represented by the values of the *TYPE*, *LENGTH*, *PRECISION*, *SCALE*, *DATETIME_INTERVAL_CODE*, *DATETIME_INTERVAL_PRECISION*, *CHARACTER_SET_CATALOG*, *CHARACTER_SET_SCHEMA*, *CHARACTER_SET_NAME*, *USER_DEFINED_TYPE_CATALOG*, *USER_DEFINED_TYPE_SCHEMA*, *USER_DEFINED_TYPE_NAME*, *SCOPE_CATALOG*, *SCOPE_SCHEMA*, and *SCOPE_NAME* fields in the *i*-th item descriptor area of *RD* whose *LEVEL* is 0 (zero) and all of its subordinate descriptor areas.

22.6 Implicit FETCH USING clause

- e) Let *TYPE*, *OL*, *D*, *DP*, *IP*, and *LP* be the values of the *TYPE*, *OCTET_LENGTH*, *DATA*, *DATA_POINTER*, *INDICATOR*, and *OCTET_LENGTH* fields, respectively, in the item descriptor area of *SRD* corresponding to the *i*-th bound target (or part thereof, if the item descriptor area is a subordinate descriptor).
- f) Let *SV* be the value of the <select list> column, with data type *SDT*.
- g) Case:
 - i) If *TYPE* indicates *CHARACTER*, then:
 - 1) Let *UT* be the code value corresponding to *CHARACTER VARYING* as specified in Table 7, "Codes used for implementation data types in SQL/CLI", in ISO/IEC 9075-3.
 - 2) Let *LV* be the implementation-defined maximum length for a *CHARACTER VARYING* data type.
 - ii) Otherwise, let *UT* be *TYPE* and let *LV* be 0 (zero).
- h) Let *TDT* be the effective data type of the *i*-th bound target as represented by the type *UT*, the length value *LV*, and the values of the *PRECISION*, *SCALE*, *CHARACTER_SET_CATALOG*, *CHARACTER_SET_SCHEMA*, *CHARACTER_SET_NAME*, *USER_DEFINED_TYPE_CATALOG*, *USER_DEFINED_TYPE_SCHEMA*, *USER_DEFINED_TYPE_NAME*, *SCOPE_CATALOG*, *SCOPE_SCHEMA*, and *SCOPE_NAME* fields in the item descriptor area of *SRD* whose *LEVEL* is 0 (zero) and all of its subordinate descriptor areas.
- i) If *TDT* is an array locator data type, then:
 - i) If *SV* is not the null value, then a locator *L* that uniquely identifies *SV* is generated and the value *TV* of the *i*-th bound target is set to an implementation-dependent four-octet value that represents *L*.
 - ii) Otherwise, the value *TV* of the *i*-th bound target is the null value.
- j) If *SDT* and *TDT* are predefined data types, then

Case:

 - i) If *SDT* and *TDT* are binary string data types or bit string data types, then the <cast specification>


```
CAST ( SV AS TDT )
```

 is effectively performed and the result is the value *TV* is the *i*-th parameter.
 - ii) If *SDT* and *TDT* are numeric data types, then the <cast specification>


```
CAST ( SV AS TDT )
```

 is effectively performed and the result is the value *TV* is the *i*-th parameter.
 - iii) Otherwise, the <form-of-use conversion>


```
CONVERT ( CAST ( SV AS CHARACTER VARYING (M) ) USING UTF16 )
```

 is effectively performed, where *M* is the implementation-defined maximum length of a variable-length character string, and the result is the value *TV* is the *i*-th parameter.
- k) Let *IDA* be the top-level item descriptor area corresponding to the *i*-th bound column.

l) Case:

i) If TYPE indicates ROW, then

Case:

- 1) If *TV* is the null value, then the value of *IP* for *IDA* and that in all subordinate descriptor areas of *IDA* that are not subordinate to an item descriptor area whose TYPE indicates ARRAY or ARRAY_LOCATOR, is set to the appropriate 'Code' for SQL NULL DATA in Table 26, "Miscellaneous codes used in SQL/CLI", in ISO/IEC 9075-3, and the value of the host variable addressed by *DP* and the values of *D* and *LP* are implementation-dependent.
- 2) Otherwise, the *i*-th subordinate descriptor area of *IDA* is set to reflect the value of the *i*-th field of *TV* by applying General Rule , "5)l)" to the *i*-th subordinate descriptor area of *IDA* as *IDA*, the value of *i*-th field of *TV* as *TV*, the value of the *i*-th field of *SV* as *SV*, and the data type of the *i*-th field of *SV* as *SDT*.

ii) Otherwise,

Case:

- 1) If *TV* is the null value, then the value of *IP* is set to the appropriate 'Code' for SQL NULL DATA in Table 26, "Miscellaneous codes used in SQL/CLI", in ISO/IEC 9075-3, and the value of the host variable addressed by *DP* and the value of *D* and the value of *LP* are implementation-dependent.

2) Otherwise:

A) The value of *IP* is set to 0 (zero).

B) Case:

I) If TYPE indicates CHARACTER or CHARACTER LARGE OBJECT, then:

- 1) If *TV* is a zero-length character string, then it is implementation-defined whether or not an exception condition is raised: *data exception — zero-length character string*.

2) Case:

- a) If *HL1* and *HL2* are both pointer-supporting languages, then the General Rules of Subclause 22.7, "Character string retrieval", are applied with *DP*, *TV*, *OL*, and *LP* as *TARGET*, *VALUE*, *TARGET OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.
- b) Otherwise, the General Rules of Subclause 22.7, "Character string retrieval", are applied with *D*, *TV*, *OL*, and *LP*, as *TARGET*, *VALUE*, *TARGET OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.

II) If TYPE indicates BINARY LARGE OBJECT, then

Case:

- 1) If *HL1* and *HL2* are both pointer-supporting languages, then the General Rules of Subclause 22.8, “Binary large object string retrieval”, are applied with *DP*, *TV*, *OL*, and *LP* as *TARGET*, *VALUE*, *TARGET OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.
 - 2) Otherwise, the General Rules of Subclause 22.8, “Binary large object string retrieval”, are applied with *D*, *TV*, *OL*, and *LP* as *TARGET*, *VALUE*, *TARGET OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.
- III) If *TYPE* indicates *ARRAY* or *ARRAY LOCATOR*, then the value of *RETURNED_CARDINALITY* is set to the cardinality of *TV*.
- IV) Otherwise,

Case:

- 1) If *HL1* and *HL2* are both pointer-supporting languages, then the value of the host variable addressed by *DP* is set to *TV* and the value of *LP* is implementation-dependent.
- 2) Otherwise, the value of *D* is set to *TV* and the value of *LP* is implementation-dependent.

Conformance Rules

None.

22.7 Character string retrieval

Function

Specify the rules for retrieving character string values.

General Rules

- 1) Let *T*, *V*, *TL*, and *RL* be a *TARGET*, *VALUE*, *TARGET OCTET LENGTH*, and *RETURNED OCTET LENGTH* specified in an application of this Subclause.
- 2) If *TL* is not greater than zero, then an exception condition is raised: *FDW-specific condition — invalid string length or buffer length*.
- 3) Let *L* be the length in octets of *V*.
- 4) If *RL* is not a null pointer, then *RL* is set to *L*.
- 5) Case:
 - a) If *L* is not greater than *TL*, then the first *L* octets of *T* are set to *V* and the values of the remaining octets of *T* are implementation-dependent.
 - b) Otherwise, *T* is set to the first *TL* octets of *V* and a completion condition is raised: *warning — string data, right truncation*.

22.8 Binary large object string retrieval

Function

Specify the rules for retrieving BINARY LARGE OBJECT string values.

General Rules

- 1) Let *T*, *V*, *TL*, and *RL* be a *TARGET*, *VALUE*, *TARGET OCTET LENGTH*, and *RETURNED OCTET LENGTH* specified in an application of this Subclause.
- 2) If *TL* is not greater than zero, then an exception condition is raised: *FDW-specific condition — invalid string length or buffer length*.
- 3) Let *L* be the length in octets of *V*.
- 4) If *RL* is not a null pointer, then *RL* is set to *L*.
- 5) Case:
 - a) If *L* is not greater than *TL*, then the first *L* octets of *T* are set to *V* and the values of the remaining octets of *T* are implementation-dependent.
 - b) Otherwise, *T* is set to the first *TL* octets of *V* and a completion condition is raised: *warning — string data, right truncation*.

22.9 Tables used with SQL/MED

The tables contained in this Subclause are used to specify the codes used by the various foreign-data wrapper interface routines.

Table 29—Codes used for <table reference> types

<table reference> type	Code
TABLE_NAME	1

Table 30—Codes used for <value expression> types

<value expression> type	Code
COLUMN_NAME	1

Table 31—Codes used for foreign-data wrapper diagnostic fields

Field	Code	Type
CLASS_ORIGIN	1	Status
MESSAGE_LENGTH	2	Status
MESSAGE_OCTET_LENGTH	3	Status
MESSAGE_TEXT	4	Status
MORE	5	Header
NATIVE_CODE	6	Status
NUMBER	7	Header
RETURNCODE	8	Header
SQLSTATE	9	Status
SUBCLASS_ORIGIN	10	Status
Implementation-defined diagnostics header field	< 0	Header
Implementation-defined diagnostics status field	< 0	Status

Table 32—Codes used for foreign-data wrapper descriptor fields

Field	Code	SQL Item Descriptor Name	Type
CARDINALITY	1040	CARDINALITY	Item
CHARACTER_SET_CATALOG	1018	CHARACTER_SET_CATALOG	Item
CHARACTER_SET_NAME	1020	CHARACTER_SET_NAME	Item

ISO/IEC 9075-9:2001 (E)
22.9 Tables used with SQL/MED

Table 32—Codes used for foreign-data wrapper descriptor fields (Cont.)

Field	Code	SQL Item Descriptor Name	Type
CHARACTER_SET_SCHEMA	1019	CHARACTER_SET_SCHEMA	Item
COLLATION_CATALOG	1015	COLLATION_CATALOG	Item
COLLATION_NAME	1017	COLLATION_NAME	Item
COLLATION_SCHEMA	1016	COLLATION_SCHEMA	Item
COUNT	1001	COUNT	Header
CURRENT_TRANSFORM_GROUP	1039	(Not applicable)	Item
DATA	1050	DATA	Item
DATA_POINTER	1010	DATA	Item
DATETIME_INTERVAL_CODE	1007	DATETIME_INTERVAL_CODE	Item
DATETIME_INTERVAL_PRECISION	26	DATETIME_INTERVAL_PRECISION	Item
DEGREE	1041	DEGREE	Item
DYNAMIC_FUNCTION	1031	DYNAMIC_FUNCTION	Header
DYNAMIC_FUNCTION_CODE	1032	DYNAMIC_FUNCTION_CODE	Header
INDICATOR	1051	INDICATOR	Item
KEY_MEMBER	1030	KEY_MEMBER	Item
KEY_TYPE	1029	KEY_TYPE	Header
LENGTH	1003	LENGTH	Item
LEVEL	1042	LEVEL	Item
NAME	1011	NAME	Item
NULLABLE	1008	NULLABLE	Item
OCTET_LENGTH	1013	OCTET_LENGTH	Item
PARAMETER_MODE	1021	PARAMETER_MODE	Item
PARAMETER_ORDINAL_POSITION	1022	PARAMETER_ORDINAL_POSITION	Item
PARAMETER_SPECIFIC_CATALOG	1023	PARAMETER_SPECIFIC_CATALOG	Item
PARAMETER_SPECIFIC_NAME	1025	PARAMETER_SPECIFIC_NAME	Item
PARAMETER_SPECIFIC_SCHEMA	1024	PARAMETER_SPECIFIC_SCHEMA	Item
PRECISION	1005	PRECISION	Item
RETURNED_CARDINALITY	1052	RETURNED_CARDINALITY	Item
RETURNED_OCTET_LENGTH	1053	Both OCTET_LENGTH (input) and RETURNED_OCTET_LENGTH (out- put)	Item
SCALE	1006	SCALE	Item
SCOPE_CATALOG	1033	SCOPE_CATALOG	Item
SCOPE_NAME	1034	SCOPE_NAME	Item

Table 32—Codes used for foreign-data wrapper descriptor fields (Cont.)

Field	Code	SQL Item Descriptor Name	Type
SCOPE_SCHEMA	1035	SCOPE_SCHEMA	Item
SPECIFIC_TYPE_CATALOG	1036	(Not applicable)	Item
SPECIFIC_TYPE_NAME	1038	(Not applicable)	Item
SPECIFIC_TYPE_SCHEMA	1037	(Not applicable)	Item
TOP_LEVEL_COUNT	1044	TOP_LEVEL_COUNT	Header
TYPE	1002	TYPE	Item
UNNAMED	1012	UNNAMED	Item
USER_DEFINED_TYPE_CATALOG	1026	USER_DEFINED_TYPE_CATALOG	Item
USER_DEFINED_TYPE_NAME	1028	USER_DEFINED_TYPE_NAME	Item
USER_DEFINED_TYPE_SCHEMA	1027	USER_DEFINED_TYPE_SCHEMA	Item
Implementation-defined foreign-data wrapper descriptor header field	0 (zero) through 999, or ≥ 1200 , excluding values defined in this table	Implementation-defined foreign-data wrapper descriptor header field	Header
Implementation-defined foreign-data wrapper descriptor item field	0 (zero) through 999, or ≥ 1200 , excluding values defined in this table	Implementation-defined foreign-data wrapper descriptor item field	Item

Table 33—Codes used for foreign-data wrapper handle types

Handle type	Code
ExecutionHandle	1
FSConnectionHandle	2

ISO/IEC 9075-9:2001 (E)
22.9 Tables used with SQL/MED

Table 33—Codes used for foreign-data wrapper handle types (Cont.)

Handle type	Code
ReplyHandle	3
RequestHandle	4
DataHandle	5
ServerHandle	6
TableReferenceHandle	7
UserHandle	8
ValueExpressionHandle	9
WrapperHandle	10
WrapperEnvHandle	11
DescriptorHandle	12

Table 34—Ability to retrieve foreign-data wrapper descriptor fields

Field	May be retrieved			
	SRD	WRD or TRD	SPD	WPD
CARDINALITY	No		No	
CHARACTER_SET_CATALOG				
CHARACTER_SET_NAME				
CHARACTER_SET_SCHEMA				
COLLATION_CATALOG				
COLLATION_NAME				
COLLATION_SCHEMA				
COUNT				
CURRENT_TRANSFORM_GROUP				
DATA		No		No
DATA_POINTER		No		No
DATETIME_INTERVAL_CODE				
DATETIME_INTERVAL_PRECISION				
DEGREE	No		No	
DYNAMIC_FUNCTION	No		No	
DYNAMIC_FUNCTION_CODE	No		No	
INDICATOR		No		No
KEY_MEMBER	No		No	No

Table 34—Ability to retrieve foreign-data wrapper descriptor fields (Cont.)

Field	May be retrieved			
	SRD	WRD or TRD	SPD	WPD
KEY_TYPE	No		No	No
LENGTH				
LEVEL				
NAME				
NULLABLE				
OCTET_LENGTH				
PARAMETER_MODE	No		No	
PARAMETER_ORDINAL_POSITION	No		No	
PARAMETER_SPECIFIC_CATALOG	No		No	
PARAMETER_SPECIFIC_NAME	No		No	
PARAMETER_SPECIFIC_SCHEMA	No		No	
PRECISION				
RETURNED_CARDINALITY		No		No
RETURNED_OCTET_LENGTH		No		No
SCALE				
SCOPE_CATALOG				
SCOPE_NAME				
SCOPE_SCHEMA				
SPECIFIC_TYPE_CATALOG				
SPECIFIC_TYPE_NAME				
SPECIFIC_TYPE_SCHEMA				
TOP_LEVEL_COUNT				
TYPE				
UNNAMED				
USER_DEFINED_TYPE_CATALOG				
USER_DEFINED_TYPE_NAME				
USER_DEFINED_TYPE_SCHEMA				
Implementation-defined foreign-data wrapper descriptor header field	ID	ID	ID	ID
Implementation-defined foreign-data wrapper descriptor item field	ID	ID	ID	ID

ISO/IEC 9075-9:2001 (E)
22.9 Tables used with SQL/MED

Table 35—Ability to set foreign-data wrapper descriptor fields

Field	May be set			
	SRD	WRD or TRD	SPD	WPD
CARDINALITY	No	No	No	
CHARACTER_SET_CATALOG		No		
CHARACTER_SET_NAME		No		
CHARACTER_SET_SCHEMA		No		
COLLATION_CATALOG		No		
COLLATION_NAME		No		
COLLATION_SCHEMA		No		
COUNT		No		
CURRENT_TRANSFORM_GROUP	No	No	No	No
DATA		No		
DATA_POINTER		No		
DATETIME_INTERVAL_CODE		No		
DATETIME_INTERVAL_PRECISION		No		
DEGREE	No	No	No	
DYNAMIC_FUNCTION	No	No	No	No
DYNAMIC_FUNCTION_CODE	No	No	No	No
INDICATOR		No		No
KEY_MEMBER	No	No	No	No
KEY_TYPE	No	No	No	No
LENGTH		No		
LEVEL		No		
NAME		No		
NULLABLE		No		
OCTET_LENGTH		No		
PARAMETER_MODE	No	No	No	
PARAMETER_ORDINAL_POSITION	No	No	No	
PARAMETER_SPECIFIC_CATALOG	No	No	No	
PARAMETER_SPECIFIC_NAME	No	No	No	
PARAMETER_SPECIFIC_SCHEMA	No	No	No	
PRECISION		No		
RETURNED_CARDINALITY		No		No

Table 35—Ability to set foreign-data wrapper descriptor fields (Cont.)

Field	May be set			
	SRD	WRD or TRD	SPD	WPD
RETURNED_OCTET_LENGTH		No		No
SCALE		No		
SCOPE_CATALOG		No		
SCOPE_NAME		No		
SCOPE_SCHEMA		No		
SPECIFIC_TYPE_CATALOG	No	No	No	No
SPECIFIC_TYPE_NAME	No	No	No	No
SPECIFIC_TYPE_SCHEMA	No	No	No	No
TOP_LEVEL_COUNT		No		
TYPE		No		
UNNAMED		No		
USER_DEFINED_TYPE_CATALOG		No		
USER_DEFINED_TYPE_NAME		No		
USER_DEFINED_TYPE_SCHEMA		No		
Implementation-defined foreign-data wrapper descriptor header field	ID	ID	ID	ID
Implementation-defined foreign-data wrapper descriptor item field	ID	ID	ID	ID

Table 36—Foreign-data wrapper descriptor field default values

Field	Default values			
	SRD	WRD or TRD	APD	WPD
CARDINALITY				
CHARACTER_SET_CATALOG				
CHARACTER_SET_NAME				
CHARACTER_SET_SCHEMA				
COLLATION_CATALOG				
COLLATION_NAME				
COLLATION_SCHEMA				
COUNT	0 (zero)		0 (zero)	

ISO/IEC 9075-9:2001 (E)
22.9 Tables used with SQL/MED

Table 36—Foreign-data wrapper descriptor field default values (Cont.)

Field	Default values			
	SRD	WRD or TRD	APD	WPD
CURRENT_TRANSFORM_GROUP				
DATA				
DATA_POINTER	Null		Null	
DATETIME_INTERVAL_CODE				
DATETIME_INTERVAL_PRECISION				
DEGREE				
DYNAMIC_FUNCTION				
DYNAMIC_FUNCTION_CODE				
INDICATOR				
KEY_MEMBER				
KEY_TYPE				
LENGTH				
LEVEL	0 (zero)			
NAME				
NULLABLE				
OCTET_LENGTH				
PARAMETER_MODE				
PARAMETER_ORDINAL_POSITION				
PARAMETER_SPECIFIC_CATALOG				
PARAMETER_SPECIFIC_NAME				
PARAMETER_SPECIFIC_SCHEMA				
PRECISION				
RETURNED_CARDINALITY				
RETURNED_OCTET_LENGTH				
SCALE				
SCOPE_CATALOG				
SCOPE_NAME				
SCOPE_SCHEMA				

Table 36—Foreign-data wrapper descriptor field default values (Cont.)

Field	Default values			
	SRD	WRD or TRD	APD	WPD
SPECIFIC_TYPE_CATALOG	0 (zero)		0 (zero)	
SPECIFIC_TYPE_NAME				
SPECIFIC_TYPE_SCHEMA				
TOP_LEVEL_COUNT				
TYPE				
UNNAMED				
USER_DEFINED_TYPE_CATALOG	ID	ID	ID	ID
USER_DEFINED_TYPE_NAME				
USER_DEFINED_TYPE_SCHEMA				
Implementation-defined foreign-data wrapper descriptor header field				
Implementation-defined foreign-data wrapper descriptor item field				
	ID	ID	ID	ID

Table 37—Codes used for the format of the character string transmitted by GetSQLString()

Format	Code
SQL-string format	1
Implementation-defined formats	x ¹
¹ An implementation-defined negative number different from the value associated with any other format.	

(Blank page)

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

23 Foreign-data wrapper interface routines

23.1 <foreign-data wrapper interface routine>

Function

Describe a generic foreign-data wrapper interface routine.

Format

```
<foreign-data wrapper interface routine> ::=
    <foreign-data wrapper interface routine prefix>
    <foreign-data wrapper interface routine generic>
```

```
<foreign-data wrapper interface routine prefix> ::= MED
```

```
<foreign-data wrapper interface routine generic> ::=
    <foreign-data wrapper interface routine name>
    <foreign-data wrapper parameter list>
    [ <foreign-data wrapper returns clause> ]
```

```
<foreign-data wrapper interface routine name> ::=
```

```
    AllocDescriptor
    | AllocWrapperEnv
    | Close
    | ConnectServer
    | FreeDescriptor
    | FreeExecutionHandle
    | FreeFSConnection
    | FreeReplyHandle
    | FreeWrapperEnv
    | GetAuthorizationId
    | GetDescriptor
    | GetDiagnostics
    | GetTableColOpt
    | GetTableColOptByName
    | GetTableOpt
    | GetTableOptByName
    | GetTableServerName
    | GetNumTableColOpts
    | GetNumTableOpts
    | GetNumReplySelectElems
    | GetNumReplyTableRefs
    | GetNumSelectElems
    | GetNumServerOpts
    | GetNumTableRefElems
    | GetNumUserOpts
    | GetNumWrapperOpts
    | GetOpts
    | GetReplySelectElem
    | GetReplyTableRef
    | GetSelectElem
    | GetSelectElemType
    | GetServerName
    | GetServerOpt
    | GetServerOptByName
```

ISO/IEC 9075-9:2001 (E)

23.1 <foreign-data wrapper interface routine>

- | GetServerType
- | GetServerVersion
- | GetSPDHandle
- | GetSQLString
- | GetSRDHandle
- | GetStatistics
- | GetTableRefElem
- | GetTableRefElemType
- | GetTableRefTableName
- | GetTRDHandle
- | GetUserOpt
- | GetUserOptByName
- | GetValExprColName
- | GetWPDHandle
- | GetWRDHandle
- | GetWrapperLibraryName
- | GetWrapperName
- | GetWrapperOpt
- | GetWrapperOptByName
- | InitRequest
- | Iterate
- | Open
- | ReOpen
- | SetDescriptor
- | TransmitRequest

<foreign-data wrapper parameter list> ::=
 <left paren> <foreign-data wrapper parameter declaration>
 [{ <comma> <foreign-data wrapper parameter declaration> }...] <right paren>

<foreign-data wrapper parameter declaration> ::=
 <foreign-data wrapper parameter name>
 <foreign-data wrapper parameter mode>
 <foreign-data wrapper parameter data type>

<foreign-data wrapper parameter name> ::=
 !! See the individual foreign-data wrapper interface routine definitions

<foreign-data wrapper parameter mode> ::=
 IN
 | OUT
 | INOUT

<foreign-data wrapper parameter data type> ::=
 INTEGER
 | SMALLINT
 | ANY
 | CHARACTER <left paren> <length> <right paren>

<foreign-data wrapper returns clause> ::= RETURNS SMALLINT

Syntax Rules

- 1) A <foreign-data wrapper interface routine> defines a predefined routine written in a standard programming language that is invoked by a compilation unit of the same standard programming language. Let *HL* be that standard programming language. *HL* shall be one of Ada, C, COBOL, Fortran, MUMPS, Pascal, or PL/I.

23.1 <foreign-data wrapper interface routine>

- 2) A <foreign-data wrapper interface routine> that contains a <foreign-data wrapper returns clause> is called a *foreign-data wrapper interface function*. A <foreign-data wrapper interface routine> that does not contain a <foreign-data wrapper returns clause> is called a *foreign-data wrapper interface procedure*.
- 3) For each foreign-data wrapper interface function *WF*, there is a corresponding foreign-data wrapper interface procedure *WP*, with the same <foreign-data wrapper interface routine name>. The <foreign-data wrapper parameter list> for *WP* is the same as the <foreign-data wrapper parameter list> for *WF* but with the following additional <foreign-data wrapper parameter declaration>:

ReturnCode OUT SMALLINT

- 4) *HL* shall support either the invocation of *WF* or the invocation of *WP*. It is implementation-defined which is supported.
- 5) Case:
 - a) If <foreign-data wrapper parameter mode> is IN, then the parameter is an *input parameter*. The value of an input argument is established when a foreign-data wrapper interface routine is invoked.
 - b) If <foreign-data wrapper parameter mode> is OUT, then the parameter is an *output parameter*. The value of an output argument is established when a foreign-data wrapper interface routine is invoked.
 - c) If <foreign-data wrapper parameter mode> is INOUT, then the parameter is both an input parameter and an output parameter.
- 6) The value of an output parameter is an address. It is either a non-pointer host variable passed by reference or a pointer host variable passed by value.
- 7) There shall be no <separator> between the <foreign-data wrapper interface routine prefix> and the <foreign-data wrapper interface routine generic> in a <foreign-data wrapper interface routine name>.
- 8) Let *WR* be a <foreign-data wrapper interface routine> and let *RN* be its <foreign-data wrapper interface routine name>. Let *RNU* be the value of UPPER(*RN*).

Case:

- a) If *HL* supports case sensitive routine names, then the name used for the invocation of *WR* shall be *RN*.
 - b) If *HL* does not support <simple Latin lower case letter>s, then the name used for the invocation of *WR* shall be *RNU*.
 - c) If *HL* does not support case sensitive routine names, then the name used for the invocation of *WR* shall be *RN* or *RNU*.
- 9) Let *operative data type correspondence table* be the data type correspondence table for *HL* as specified in Subclause 20.3, "SQL/CLI data type correspondences". Refer to the two columns of the operative data type correspondence table as the "SQL data type column" and the "host data type column".

ISO/IEC 9075-9:2001 (E)

23.1 <foreign-data wrapper interface routine>

- 10) Let TI , TS , TC , and TV be the types listed in the host data type column for the rows that contains INTEGER, SMALLINT, CHARACTER(L) and CHARACTER VARYING(L), respectively, in the SQL data type column.
- a) If TS is "None", then let $TS = TI$.
 - b) If TC is "None", then let $TC = TV$.
 - c) For each parameter P ,
Case:
 - i) If the foreign-data wrapper parameter data type is INTEGER, then the type of the corresponding argument shall be TI .
 - ii) If the foreign-data wrapper parameter data type is SMALLINT, then the type of the corresponding argument shall be TS .
 - iii) If the foreign-data wrapper parameter data type is CHARACTER(L), then the type of the corresponding argument shall be TC .
 - iv) If the foreign-data wrapper parameter data type is ANY, then
Case:
 - 1) If HL is C, then the type of the corresponding argument shall be "void *".
 - 2) Otherwise, the type of the corresponding argument shall be any type (other than 'None') listed in the host data type column.
 - d) If the foreign-data wrapper interface routine is a foreign-data wrapper interface function, then the type of the returned value is TS .

Access Rules

None.

General Rules

- 1) The rules for invocation of the <foreign-data wrapper interface routine> are specified in Sub-clause 23.2, "<foreign-data wrapper interface routine> invocation".

Conformance Rules

None.

23.2 <foreign-data wrapper interface routine> invocation

Function

Specify the rules for invocation of a <foreign-data wrapper interface routine>.

Syntax Rules

- 1) Let *HL* be the standard programming language of *CP*, the caller of a <foreign-data wrapper interface routine>.
- 2) A foreign-data wrapper interface function or foreign-data wrapper interface procedure is invoked by the *HL* mechanism for invoking functions or procedures, respectively.
- 3) Let *RN* be the <foreign-data wrapper interface routine name> of the <foreign-data wrapper interface routine> invoked by *CP*. The number of arguments provided in the invocation shall be the same as the number of <foreign-data wrapper parameter declaration>s for *RN*.
- 4) Let *DA* be the data type of the *i*-th argument in the invocation and let *DP* be the <foreign-data wrapper parameter data type> of the *i*-th <foreign-data wrapper parameter declaration> of *RN*. *DA* shall be the *HL* equivalent of *DP* as specified by the rules of Subclause 23.1, "<foreign-data wrapper interface routine>".
- 5) Each argument to a <foreign-data wrapper interface routine> that is of type CHARACTER(*n*) shall be passed by reference, according to the mechanisms of *HL*.

Case:

- a) Of *HL* is C, then each argument to a <foreign-data wrapper interface routine> that is not of type CHARACTER(*n*) shall be passed by value.
- b) Otherwise, each argument to a <foreign-data wrapper interface routine> that is not of type CHARACTER(*n*) shall be passed by reference.

Access Rules

None.

General Rules

- 1) If the value of any input argument provided by *CP* falls outside the set of allowed values of the data type of the parameter, or if the value of any output argument resulting from the execution of the <foreign-data wrapper interface routine> falls outside the set of values supported by *CP* for that parameter, then the effect is implementation-defined.
- 2) When the <foreign-data wrapper interface routine> is called by *CP*:
 - a) The values of all input arguments to *RN* are established.
 - b) *RN* is invoked.

ISO/IEC 9075-9:2001 (E)

23.2 <foreign-data wrapper interface routine> invocation

3) Case:

- a) If the <foreign-data wrapper interface routine> is a foreign-data wrapper interface function, then:
 - i) The values of all output arguments are established.
 - ii) Let *RC* be the return value.
- b) If the <foreign-data wrapper interface routine> is a foreign-data wrapper interface procedure, then:
 - i) The values of all output arguments are established except for the argument associated with the *ReturnCode* parameter.
 - ii) Let *RC* be the argument associated with the *ReturnCode* parameter.

4) Case:

- a) If *RN* executed successfully, then:
 - i) Either a completion condition is raised: *successful completion*, or a completion condition is raised: *no data*.
 - ii) Case:
 - 1) If a completion condition is raised: *successful completion*, then *RC* is set to indicate **Success**.
 - 2) If a completion condition is raised: *no data*, then *RC* is set to indicate **No data found**.
- b) If *RN* did not execute successfully, then:
 - i) All changes made to SQL-data or schemas by the execution of *RN* are canceled.
 - ii) One or more exception conditions are raised as determined by the General Rules of this and other Subclauses of this part of ISO/IEC 9075 or by implementation-defined rules.
 - iii) Case:
 - 1) If an exception condition is raised: *FDW-specific condition — invalid handle*, then *RC* is set to indicate **Invalid handle**.
 - 2) Otherwise, *RC* is set to indicate **Error**.

Conformance Rules

None.

23.3 Foreign-data wrapper interface wrapper routines

23.3.1 AllocWrapperEnv

Function

Allocate a foreign-data wrapper environment and assign a handle to it.

Definition

```
AllocWrapperEnv (
    WrapperHandle      IN      INTEGER,
    WrapperEnvHandle   OUT     INTEGER )
RETURNS SMALLINT
```

General Rules

- 1) Let *WH* be the value of WrapperHandle.
- 2) If *WH* does not identify an allocated foreign-wrapper description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) If the maximum number of foreign-data wrapper environments that can be allocated at one time has already been reached, then an exception condition is raised: *FDW-specific condition — limit on number of handles exceeded*. A skeleton FDW-environment is allocated and is assigned a unique value that is returned in WrapperEnvHandle.
- 4) Case:
 - a) If the memory requirements to manage an foreign-data wrapper environment cannot be satisfied, then WrapperEnvHandle is set to zero and an exception condition is raised: *FDW-specific condition — memory allocation error*.
NOTE 60 – No diagnostic information is generated in this case, as there is no valid WrapperEnvHandle that can be used to obtain diagnostics information.
 - b) If the resources to manage an foreign-data wrapper environment cannot be allocated for implementation-defined reasons, then an implementation-defined exception condition is raised. A skeleton FDW-environment is allocated and is assigned a unique value that is returned in WrapperEnvHandle.
 - c) Otherwise, the resources to manage an foreign-data wrapper environment are allocated and are referred to as an *allocated FDW-environment*. The allocated FDW-environment is assigned a unique value that is returned in WrapperEnvHandle.
- 5) If an exception condition is raised in any of the routines invoked in any of the following General Rules, then the diagnostics records returned by the invoked routines are transferred to the diagnostics area associated with the WrapperEnvHandle and further processing of this routine is terminated as if the exception condition had been raised in this routine.
- 6) Let *WN* be the WrapperName that would be returned by an invocation of the GetWrapperName() routine with *WH* as the WrapperHandle parameter.
- 7) Let *WL* be the WrapperLibraryName that would be returned by an invocation of the GetWrapperLibraryName() routine with *WH* as the WrapperHandle parameter.

ISO/IEC 9075-9:2001 (E)

23.3 Foreign-data wrapper interface wrapper routines

- 8) It is implementation-dependent what use the AllocWrapperEnv() routine makes of the values of WN and WL.

Conformance Rules

None.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

23.3.2 Close

Function

Close an FDW-execution.

Definition

```
Close (
    ExecutionHandle      IN      INTEGER )
    RETURNS SMALLINT
```

General Rules

- 1) Let *EH* be the value of ExecutionHandle.
- 2) If *EH* does not identify an opened FDW-execution, then an exception condition is raised: *FDW-specific condition — function sequence error*.
- 3) Let *E* be the opened FDW-execution identified by *EH*.
- 4) Case:
 - a) If there is no open cursor associated with *E*, then an exception condition is raised: *invalid cursor state*.
 - b) Otherwise:
 - i) The open cursor associated with *E* is placed in the closed state and its copy of the select source is destroyed.
 - ii) Any fetched row associated with *E* is removed from association with *E*.
- 5) *EH* is reset to be an allocated FDW-execution.

Conformance Rules

None.

23.3.3 ConnectServer

Function

Establish a connection to a foreign server and assign a handle to it.

Definition

```
ConnectServer (
    WrapperEnvHandle    IN    INTEGER,
    ServerHandle        IN    INTEGER,
    UserHandle          IN    INTEGER,
    FSConnectionHandle  OUT   INTEGER )
RETURNS SMALLINT
```

General Rules

- 1) If WrapperEnvHandle does not identify an allocated FDW-environment, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 2) Let *SH* be the value of ServerHandle.
- 3) If *SH* does not identify an allocated foreign server description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 4) Let *UH* be the value of UserHandle.
- 5) If *UH* does not identify an allocated user mapping description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 6) If an exception condition is raised in any of the routines invoked in any of the following General Rules, then the diagnostics records returned by the invoked routines are transferred to the foreign-data wrapper diagnostics area associated with the WrapperEnvHandle and further processing of this routine is terminated as if the exception condition had been raised in this routine.
- 7) Let *UN* be the AuthorizationId that would be returned by an invocation of the GetAuthorizationId() routine with *UH* as the UserHandle parameter.
- 8) Let *SN* be the ServerName that would be returned by an invocation of the GetServerName() routine with *SH* as the ServerHandle parameter.
- 9) Let *ST* be the ServerType that would be returned by an invocation of the GetServerType() routine with *SH* as the ServerHandle parameter.
- 10) Let *SV* be the ServerVersion that would be returned by an invocation of the GetServerVersion() routine with *SH* as the ServerHandle parameter.
- 11) Let *E* be the FDW-environment identified by WrapperEnvHandle.
- 12) The foreign-data wrapper diagnostics area associated with *E* is emptied.
- 13) If the maximum number of FS-connections that can be allocated at one time has already been reached, then FSConnectionHandle is set to zero and an exception condition is raised: *FDW-specific condition — limit on number of handles exceeded*.

23.3 Foreign-data wrapper interface wrapper routines

14) Case:

- a) If the memory requirements to manage an FS-connection cannot be satisfied, then FSConnectionHandle is set to zero and an exception condition is raised: *FDW-specific condition — memory allocation error*.
- b) If the resources to manage an FS-connection cannot be allocated for implementation-defined reasons, then FSConnectionHandle is set to zero and an implementation-defined exception condition is raised.
- c) Otherwise, the resources to manage an FS-connection are allocated and are referred to as an allocated FS-connection. The allocated FS-connection is assigned a unique value that is returned in FSConnectionHandle.

15) Case:

- a) If a connection to *FS* cannot be made, then an exception condition is raised: *FDW-specific condition — unable to establish connection*.
- b) Otherwise, the connection to *FS* is established.

16) It is implementation-dependent what use the foreign-data wrapper makes of the values of *UN*, *SN*, *ST*, and *SV*.

Conformance Rules

None.

23.3.4 FreeExecutionHandle

Function

Deallocate an FDW-execution.

Definition

```
FreeExecutionHandle (
    ExecutionHandle IN      INTEGER )
    RETURNS SMALLINT
```

General Rules

- 1) Let *EH* be the value of ExecutionHandle.
- 2) If *EH* does not identify an allocated FDW-execution, then an exception condition is raised:
FDW-specific condition — invalid handle.
- 3) Let *E* be the FDW-execution identified by *EH*.
- 4) The foreign-data wrapper diagnostics area associated with *E* is emptied.
- 5) If there is an open cursor associated with *E*, then:
 - a) The open cursor associated with *E* is placed in the closed state and its copy of the select source is destroyed.
 - b) Any fetched row associated with *E* is removed from association with *E*.
- 6) Case:
 - a) If the PASSTHROUGH flag associated with *EH* is *False* , then:
 - i) Let *SRD* be the server row descriptor associated with *E* and let *SRDHandle* be the descriptor handle that identifies *SRD*.
 - ii) The FreeDescriptor() routine is invoked with *SRDHandle* as the DescriptorHandle parameter.
 - b) Otherwise:
 - i) Let *SRD* be the server row descriptor associated with *E* and let *SRDHandle* be the descriptor handle that identifies *SRD*. The FreeDescriptor() routine is invoked with *SRDHandle* as the DescriptorHandle parameter.
 - ii) Let *SPD* be the server parameter descriptor associated with *E* and let *SPDHandle* be the descriptor handle that identifies *SPD*. The FreeDescriptor() routine is invoked with *SPDHandle* as the DescriptorHandle parameter.
 - iii) Let *WRD* be the wrapper row descriptor associated with *E* and let *WRDHandle* be the descriptor handle that identifies *WRD*. The FreeDescriptor() routine is invoked with *WRDHandle* as the DescriptorHandle parameter.

23.3 Foreign-data wrapper interface wrapper routines

- iv) Let *WPD* be the wrapper parameter descriptor associated with *E* and let *WPDHandle* be the descriptor handle that identifies *WPD*. The `FreeDescriptor( )` routine is invoked with *WPDHandle* as the `DescriptorHandle` parameter.

- 7) *E* is deallocated and all its resources are freed.

Conformance Rules

None.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

23.3.5 FreeFSConnection

Function

Deallocate a FS-connection.

Definition

FreeFSConnection (
 FSConnectionHandle IN INTEGER)
 RETURNS SMALLINT

General Rules

- 1) Let *FSCH* be the value of FSConnectionHandle.
- 2) If *FSCH* does not identify an allocated FS-connection, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Let *C* be the allocated FS-connection identified by *FSCH*.
- 4) The foreign-data wrapper diagnostics area associated with *C* is emptied.
- 5) If an allocated reply description or FDW-execution is associated with *C*, then an exception condition is raised: *FDW-specific condition — function sequence error*.
- 6) *C* is deallocated and all its resources are freed.

Conformance Rules

None.

23.3 Foreign-data wrapper interface wrapper routines

23.3.6 FreeReplyHandle

Function

Deallocate an FDW-reply.

Definition

```
FreeReplyHandle (
    ReplyHandle      IN      INTEGER )
    RETURNS SMALLINT
```

General Rules

- 1) Let *RH* be the value of ReplyHandle.
- 2) If *RH* does not identify an allocated reply description, then an exception condition is raised:
FDW-specific condition — invalid handle.
- 3) Let *R* be the FDW-reply identified by *RH*.
- 4) The foreign-data wrapper diagnostics area associated with *R* is emptied.
- 5) *R* is deallocated and all its resources are freed.

Conformance Rules

None.

23.3.7 FreeWrapperEnv

Function

Deallocate a FDW-environment.

Definition

FreeWrapperEnv (
 WrapperEnvHandle IN INTEGER)
 RETURNS SMALLINT

General Rules

- 1) Let *WEH* be the value of WrapperEnvHandle.
- 2) If *WEH* does not identify an allocated FDW-environment, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Let *E* be the allocated FDW-environment identified by *WEH*.
- 4) The foreign-data wrapper diagnostics area associated with *E* is emptied.
- 5) If an allocated FS-connection is associated with *E*, then an exception condition is raised: *FDW-specific condition — function sequence error*.
- 6) *E* is deallocated and all its resources are freed.

Conformance Rules

None.

23.3 Foreign-data wrapper interface wrapper routines

23.3.8 GetNumReplySelectElems

Function

Get the number of <value expressions>s in the <select list> of a query that the foreign-data wrapper is capable of handling.

Definition

```
GetNumReplySelectElems (
    ReplyHandle          IN      INTEGER,
    NumberOfSelectListElements OUT  SMALLINT )
    RETURNS SMALLINT
```

General Rules

- 1) Let *RH* be the value of ReplyHandle.
- 2) If *RH* does not identify an allocated reply description, then an exception condition is raised:
FDW-specific condition — invalid handle.
- 3) Let *Q* be the query associated with *RH*.
- 4) Let *N* be the number of <value expression> elements simply contained in the <select list> of *Q* that the foreign-data wrapper is capable of handling.
- 5) NumberOfSelectListElements is set to *N*.

Conformance Rules

None.

23.3.9 GetNumReplyTableRefs

Function

Get the number of <table reference>s in the <from clause> of a query that can be processed by the foreign-data wrapper.

Definition

```
GetNumReplyTableRefs (
    ReplyHandle          IN      INTEGER,
    NumberOfTableReferences OUT   SMALLINT )
    RETURNS SMALLINT
```

General Rules

- 1) Let *RH* be the value of ReplyHandle.
- 2) If *RH* does not identify an allocated reply description, then an exception condition is raised:
FDW-specific condition — invalid handle.
- 3) Let *Q* be the query associated with *RH*.
- 4) Let *N* be the number of <table reference> elements simply contained in the <from clause> of *Q* that the foreign-data wrapper is capable of handling.
- 5) NumberOfTableReferences is set to *N*.

Conformance Rules

None.

23.3 Foreign-data wrapper interface wrapper routines

23.3.10 GetOpts

Function

Request the foreign-data wrapper to supply information about the capabilities and other information of the requested object.

Definition

```

GetOpts (
    InputHandle      IN      INTEGER,
    HandleType       IN      SMALLINT,
    ReturnFormat     OUT     INTEGER,
    Options          OUT     CHARACTER VARYING(L2),
    BufferLength      IN      INTEGER,
    StringLength     OUT     INTEGER )
RETURNS SMALLINT

```

where: *L1* is the implementation-defined maximum length of a variable-length character string, and *L2* is determined by the value of *StringLength* and has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

General Rules

- 1) Let *IH* be the value of *InputHandle* and let *HT* be the value of *HandleType*.
- 2) If *HT* is not one of the code values in Table 33, “Codes used for foreign-data wrapper handle types”, then an exception condition is raised: *FDW-specific exception — invalid handle*.
- 3) If *IH* does not identify a handle of the type indicated by *HT*, then an exception condition is raised: *FDW-specific exception — invalid handle*.
- 4) Case:
 - a) If *HT* indicates WRAPPER HANDLE, then

Case:

 - i) If the foreign-data wrapper *FDW* described by *IH* cannot return a report of its capabilities and other information, then a completion condition is raised: *no data*.
 - ii) Otherwise, a description *CD* of the capabilities of *FDW* is created.
 - b) If *HT* indicates SERVER HANDLE, then

Case:

 - i) If the foreign server *FS* described by *IH* cannot return a report of its capabilities and other information, then a completion condition is raised: *no data*.
 - ii) Otherwise, a description *CD* of the capabilities of *FS* is created. If *CD* is an XML document, then it shall be a valid XML document according to the following DTD:

ISO/IEC 9075-9:2001 (E)

23.3 Foreign-data wrapper interface wrapper routines

```
<?xml version="1.0" encoding="charencoding" ?>
<!-- SQL/MED GetOpts Document -->
<!-- UTF-8 and UTF-16 are the only required encodings -->
<!ELEMENT SQLMEDGenericOptions (SQLMEDGenericOption)+ >
  <!ELEMENT SQLMEDGenericOption (#PCDATA)>
    <!-- ATTLIST SQLMEDGenericOption SQLMEDOptionName CDATA #REQUIRED -->
    <!-- ATTLIST SQLMEDGenericOption
      SQLMEDOptionType (INTEGER | CHARACTER) #REQUIRED -->
```

where *charencoding* is either UTF-8 or UTF-16.

NOTE 61 – The CDATA values of the SQLMEDOptionName attribute and the PCDATA text of the SQLMEDGenericOption tag are implementation-defined.

NOTE 62 – The DTD can be internal to the XML document or it can be an external DTD referenced by a URI as specified in the XML specification. The way in which the foreign-data wrapper knows the URI to specify in the XML document is implementation-defined.

- 5) The General Rules of Subclause 22.7, “Character string retrieval”, are applied to Options, *CD*, *LO*, and *StringLength* as *TARGET*, *VALUE*, *TARGET OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.
- 6) Case:
 - a) If *CD* is an XML document, then the value of *ReturnFormat* is set to one (1).
 - b) If *CD* is in a format defined by the foreign-data wrapper, then the value of *ReturnFormat* is set to a value defined by the foreign-data wrapper that corresponds to that format.

NOTE 63 – All negative values are reserved for use by foreign-data wrappers. All non-negative values are reserved for use by this International Standard.

Conformance Rules

None.

23.3.11 GetReplySelectElem

Function

Get the number of a <value expression> element from the <select list> of a query that the foreign-data wrapper is capable of handling.

Definition

```
GetReplySelectElem (
    ReplyHandle          IN      INTEGER,
    Index                IN      SMALLINT,
    SelectListElementNumber OUT   SMALLINT )
    RETURNS SMALLINT
```

General Rules

- 1) Let *RH* be the value of ReplyHandle.
- 2) If *RH* does not identify an allocated reply description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Let *I* be the value of Index.
- 4) If *I* is less than 1 (one), then an exception condition is raised: *FDW-specific condition — invalid option index*.
- 5) Let *Q* be the query associated with *RH*.
- 6) Let *N* be the number of <value expression> elements simply contained in the <select list> of *Q* that the foreign-data wrapper is capable of handling.
- 7) If *I* is greater than *N*, then a completion condition is raised: *no data*, and no further rules of this Subclause are applied.
- 8) Information from the *I*-th <value expression> that the foreign-data wrapper is capable is handling is retrieved.
 - a) Let *TRN* be the number of the <value expression> element simply contained in the <select list> of *Q*.
 - b) SelectListElementNumber is set to *TRN*.

Conformance Rules

None.

ISO/IEC 9075-9:2001 (E)
23.3 Foreign-data wrapper interface wrapper routines

23.3.12 GetReplyTableRef

Function

Get the number of a <table reference> element from the <from clause> of a query that the foreign-data wrapper is capable of handling.

Definition

```
GetReplyTableRef (
    ReplyHandle          IN      INTEGER,
    Index                IN      SMALLINT,
    TableReferenceNumber OUT     SMALLINT )
RETURNS SMALLINT
```

General Rules

- 1) Let *RH* be the value of ReplyHandle.
- 2) If *RH* does not identify an allocated reply description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Let *I* be the value of Index.
- 4) If *I* is less than 1 (one), then an exception condition is raised: *FDW-specific condition — invalid option index*.
- 5) Let *Q* be the query associated with *RH*.
- 6) Let *N* be the number of <table reference> elements simply contained in the <from clause> of *Q* that the foreign-data wrapper is capable of handling.
- 7) If *I* is greater than *N*, then a completion condition is raised: *no data*, and no further rules of this Subclause are applied.
- 8) Information from the *I*-th <table reference> that the foreign-data wrapper is capable of handling is retrieved.
 - a) Let *TRN* be the number of the <table reference> element simply contained in the <from clause> of *Q*.
 - b) TableReferenceNumber is set to *TRN*.

Conformance Rules

None.

23.3 Foreign-data wrapper interface wrapper routines

23.3.13 GetSPDHandle

Function

Get the descriptor handle of the server parameter descriptor associated with an ExecutionHandle.

Definition

```
GetSPDHandle (
    ExecutionHandle    IN    INTEGER,
    SPDHandle          OUT   INTEGER )
    RETURNS SMALLINT
```

General Rules

- 1) Let *EH* be the value of ExecutionHandle.
- 2) If *EH* does not identify an allocated FDW-execution, then an exception condition is raised:
FDW-specific condition — invalid handle.
- 3) Let *SPDH* be the descriptor handle of the server parameter descriptor associated with *EH*.
- 4) SPDHandle is set to *SPDH*.

Conformance Rules

None.

23.3.14 GetSRDHandle

Function

Get the descriptor handle of the server row descriptor associated with an ExecutionHandle.

Definition

```
GetSRDHandle (
    ExecutionHandle    IN    INTEGER,
    SRDHandle          OUT   INTEGER )
RETURNS SMALLINT
```

General Rules

- 1) Let *EH* be the value of ExecutionHandle.
- 2) If *EH* does not identify an allocated FDW-execution, then an exception condition is raised:
FDW-specific condition — invalid handle.
- 3) Let *SRDH* be the descriptor handle of the server row descriptor associated with *EH*.
- 4) SRDHandle is set to *SRDH*.

Conformance Rules

None.

23.3.15 GetStatistics

Function

Retrieve implementation-defined statistics associated with a specified SQL-statement.

Definition

```
GetStatistics (
    ExecutionHandle    IN    INTEGER,
    ReturnFormat       OUT   INTEGER,
    Statistics         OUT   CHARACTER VARYING(L),
    BufferLength       IN    INTEGER,
    StringLength       OUT   INTEGER )
RETURNS SMALLINT
```

where: *L* is determined by the value of *StringLength* and has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

General Rules

- 1) Let *EH* be the value of *ExecutionHandle*.
- 2) If *EH* does not identify an allocated FDW-execution, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Case:
 - a) If the foreign-data wrapper is able to report statistics associated with the foreign request associated with *EH*, then a report of those statistics is created. If the report is in the form of an XML document, then it shall be a valid XML document according to the following DTD.

```
<?xml version="1.0" encoding="charencoding" ?>
<!-- SQL/MED GetStatistics Document -->
<!-- UTF-8 and UTF-16 are the only required encodings -->
<!ELEMENT SQLMEDStatisticsSet (SQLMEDStatistics)+ >
  <!ELEMENT SQLMEDStatistics (#PCDATA)>
    <!ATTLIST SQLMEDStatistics SQLMEDStatisticName CDATA #REQUIRED>
    <!ATTLIST SQLMEDStatistics
      SQLMEDStatisticType (INTEGER | CHARACTER) #REQUIRED>
```

where *charencoding* is either UTF-8 or UTF-16.

NOTE 64 – The CDATA values of the *SQLMEDStatisticName* attribute and the *PCDATA* text of the *SQLMEDStatistics* tag are implementation-defined.

NOTE 65 – The DTD can be internal to the XML document or it can be an external DTD referenced by a URI as specified in the XML specification. The way in which the foreign-data wrapper knows the URI to specify in the XML document is implementation-defined.

- b) Otherwise, a completion condition is raised: *no data*.
- 4) The General Rules of Subclause 22.7, “Character string retrieval”, are applied to *Statistics*, *SI*, *LOS*, and *StringLength* as *TARGET*, *VALUE*, *TARGET OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.
- 5) Case:
 - a) If *SI* is an XML document, then the value of *ReturnFormat* is set to one (1).

ISO/IEC 9075-9:2001 (E)

23.3 Foreign-data wrapper interface wrapper routines

- b) If *SI* is in a format defined by the foreign-data wrapper, then the value of *ReturnFormat* is set to a value defined by the foreign-data wrapper that corresponds to that format.

NOTE 66 – All negative values are reserved for use by foreign-data wrappers. All non-negative values are reserved for use by this International Standard.

Conformance Rules

None.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

23.3 Foreign-data wrapper interface wrapper routines

23.3.16 GetWPDHandle

Function

Get the descriptor handle of the wrapper parameter descriptor associated with an ExecutionHandle.

Definition

```
GetWPDHandle (
    ExecutionHandle    IN    INTEGER,
    WPDHandle         OUT   INTEGER )
    RETURNS SMALLINT
```

General Rules

- 1) Let *EH* be the value of ExecutionHandle.
- 2) If *EH* does not identify an allocated FDW-execution, then an exception condition is raised:
FDW-specific condition — invalid handle.
- 3) Let *WPDH* be the descriptor handle of the wrapper parameter descriptor associated with *EH*.
- 4) WPDHandle is set to *WPDH*.

Conformance Rules

None.

23.3.17 GetWRDHandle

Function

Get the descriptor handle of the wrapper row descriptor associated with an ExecutionHandle.

Definition

```
GetWRDHandle (
    ExecutionHandle    IN    INTEGER,
    WRDHandle          OUT   INTEGER )
    RETURNS SMALLINT
```

General Rules

- 1) Let *EH* be the value of ExecutionHandle.
- 2) If *EH* does not identify an allocated FDW-execution, then an exception condition is raised:
FDW-specific condition — invalid handle.
- 3) Let *WRDH* be the descriptor handle of the wrapper row descriptor associated with *EH*.
- 4) WRDHandle is set to *WRDH*.

Conformance Rules

None.

23.3.18 InitRequest

Function

Determine whether a foreign-data wrapper can execute an SQL-statement.

Definition

```
InitRequest (
    FSConnectionHandle    IN    INTEGER,
    RequestHandle         IN    INTEGER,
    ReplyHandle           OUT   INTEGER,
    ExecutionHandle       OUT   INTEGER )
RETURNS SMALLINT
```

General Rules

- 1) Let *FSCH* be the value of *FSConnectionHandle*.
- 2) If *FSCH* does not identify an allocated *FSConnection*, then an exception condition is raised:
FDW-specific condition — invalid handle.
- 3) If an exception condition is raised in any of the routines invoked in any of the following General Rules, then the diagnostics records returned by the invoked routines are transferred to the foreign-data wrapper diagnostics area associated with the *FSConnectionHandle* and further processing of this routine is terminated as if the exception condition had been raised in this routine.
- 4) Let *RH* be the value of *RequestHandle*.
- 5) Let *IG* be the indication of whether *GetSQLString()* will be invoked or not. It is foreign-data wrapper implementation-dependent whether *IG* is True or False .
NOTE 67 – The only possible values for *IG* are True and False .
- 6) Case:
 - a) If *IG* is True , then let *SS* be the *SQLString* value returned by an invocation of *GetSQLString()* with *RH* as the *RequestHandle* parameter.
 - b) Otherwise:
 - i) Let *NTRE* be the *NumberOfTableReferenceElements* that would be returned by an invocation of the *GetNumTableRefElems()* routine with *RH* as the *RequestHandle* parameter.
 - ii) Let *TRH_i* be the *TableReferenceHandle* that would be returned by invocation of the *GetTableRefElem()* routine with *RH* as the *RequestHandle* parameter and *i* as the *TableReferenceElementNumber* parameter for $1 \text{ (one)} \leq i \leq NTRE$.
 - iii) Let *TRDH_i* be the *TableReferenceDescriptorHandle* that would be returned by invocation of the *GetTRDHandle()* routine with *TRH_i* as the *TableReferenceHandle* parameter.

23.3 Foreign-data wrapper interface wrapper routines

- iv) Let NC_i be the value of the COUNT descriptor field that would be returned by invocation of the `GetDescriptor()` routine with $TRDH_i$ as the DescriptorHandle parameter, 0 (zero) as the RecordNumber parameter, and the code for COUNT from Table 32, "Codes used for foreign-data wrapper descriptor fields", as the FieldIdentifier parameter.
 - v) Let DT_{ij} be the effective data type of the j -th column, for $1 \text{ (one)} \leq j \leq NC_i$, as represented by the values of the TYPE, LENGTH, OCTET_LENGTH, PRECISION, SCALE, DATETIME_INTERVAL_CODE, DATETIME_INTERVAL_PRECISION, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME, SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME fields that would be returned by separate invocations of the `GetDescriptor()` routine with $TRDH_i$ as the DescriptorHandle parameter, j as the RecordNumber parameter, and the code for the fields TYPE, LENGTH, OCTET_LENGTH, PRECISION, SCALE, DATETIME_INTERVAL_CODE, DATETIME_INTERVAL_PRECISION, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME, SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME from Table 32, "Codes used for foreign-data wrapper descriptor fields", as the FieldIdentifier parameter.
 - vi) Let TRT_i be the TableReferenceType that would be returned by invocation of the `GetTableRefElemType()` routine with TRH_i as the TableReferenceHandle parameter.
 - vii) Let TN_i be the TableName that would be returned by invocation of the `GetTableRefTableName()` routine with TRH_i as the TableReferenceHandle parameter.
 - viii) Let $NSLE$ be the NumberOfSelectListElements that would be returned by an invocation of the `GetNumSelectElems()` routine with RH as the RequestHandle parameter.
 - ix) Let VEH_k be the ValueExpressionHandle that would be returned by invocation of the `GetSelectElem()` routine with RH as the RequestHandle parameter, and k as the SelectListElementNumber parameter for $1 \text{ (one)} \leq k \leq NSLE$.
 - x) Let VET_k be the ValueExpressionType that would be returned by invocation of the `GetSelectElemType()` routine with VEH_k as the ValueExpressionHandle parameter.
 - xi) Let CN_k be the ColumnName that would be returned by invocation of the `GetValExprColName()` routine with VEH_k as the ValueExpressionHandle parameter.
- 7) If the maximum number of FDW-replies that can be allocated at one time has already been reached, then ReplyHandle is set to zero and an exception condition is raised: *FDW-specific condition — limit on number of handles exceeded*.
- 8) Case:
- a) If the memory requirements to manage an FDW-reply cannot be satisfied, then ReplyHandle is set to zero and an exception condition is raised: *FDW-specific condition — memory allocation error*.
 - b) If the resources to manage an FDW-reply cannot be allocated for implementation-defined reasons, then ReplyHandle is set to zero and an implementation-defined exception condition is raised.

23.3 Foreign-data wrapper interface wrapper routines

- c) Otherwise, the resources to manage an FDW-reply are allocated and are referred to as an *allocated reply description*. The allocated reply description is assigned a unique value that is returned in ReplyHandle.

9) Case:

- a) If *IG* is *False* and, for any reason, the foreign-data wrapper cannot create an FDW-reply that corresponds to *RH* as described by *NTRE*, (*TRH_i*, *TRDH_i*, *NC_i*, *TRT_i*, and *TN_i*, for $1 \text{ (one)} \leq i \leq \text{NTRE}$), (*DT_{ij}*, for $1 \text{ (one)} \leq i \leq \text{NTRE}$ and $1 \text{ (one)} \leq j \leq \text{NC}_i$), *NSLE*, and (*VEH_k*, *VET_k*, and *CN_k*, for $1 \text{ (one)} \leq k \leq \text{NSLE}$), then an exception condition is raised: *FDW-specific condition — unable to create reply*.

NOTE 68 – One reason for raising this exception could be an Access Rule violation at the foreign server.

- b) If *IG* is *True* and, for any reason, the foreign-data wrapper cannot create an FDW-reply that corresponds to *RH* as described by *SS*, then an exception condition is raised: *FDW-specific condition — unable to create reply*.

- c) Otherwise, the FDW-reply corresponding to *RH* is created.

- 10) If the maximum number of FDW-executions that can be allocated at one time has already been reached, then ExecutionHandle is set to zero and an exception condition is raised: *FDW-specific condition — limit on number of handles exceeded*.

11) Case:

- a) If the memory requirements to manage an FDW-execution cannot be satisfied, then ExecutionHandle is set to zero and an exception condition is raised: *FDW-specific condition — memory allocation error*.
- b) If the resources to manage an FDW-execution cannot be allocated for implementation-defined reasons, then ExecutionHandle is set to zero and an implementation-defined exception condition is raised.
- c) Otherwise, the resources to manage an FDW-execution are allocated and are referred to as an *allocated execution description*. The allocated execution description is assigned a unique value that is returned in ExecutionHandle.

12) Case:

- a) If *IG* is *False* and the foreign-data wrapper cannot create an FDW-execution that corresponds to *RH* as described by *NTRE*, (*TRH_i*, *TRDH_i*, *NC_i*, *TRT_i*, and *TN_i*, for $1 \text{ (one)} \leq i \leq \text{NTRE}$), (*DT_{ij}*, for $1 \text{ (one)} \leq i \leq \text{NTRE}$ and $1 \text{ (one)} \leq j \leq \text{NC}_i$), *NSLE*, and (*VEH_k*, *VET_k*, and *CN_k*, for $1 \text{ (one)} \leq k \leq \text{NSLE}$), then an exception condition is raised: *FDW-specific condition — unable to create execution*.
- b) If *IG* is *False* and the foreign-data wrapper cannot create an FDW-execution that corresponds to *RH* as described by *SS*, then an exception condition is raised: *FDW-specific condition — unable to create execution*.
- c) Otherwise, the FDW-execution corresponding to *RH* is created.

- 13) The PASSTHROUGH flag associated with the allocated FDW-execution is set to *False*.

ISO/IEC 9075-9:2001 (E)

23.3 Foreign-data wrapper interface wrapper routines

- 14) Let *NIDA* be the number of item descriptor areas that must be set up for the server row descriptor. Let *SRD* be the server row descriptor identified by the *DescriptorHandle* that is returned by an invocation of the *AllocDescriptor()* routine with *NIDA* as the *MaxDetailAreas* parameter. *SRD* is associated with the allocated FDW-execution.

For this descriptor area, fields with non-blank entries in Table 36, “Foreign-data wrapper descriptor field default values”, are set to the specified default values by the invocation of the *SetDescriptor()* routine with *SRD* as the *DescriptorHandle* parameter and *r* as the *Record-Number* parameter, $1 \text{ (one)} \leq r \leq NIDA$, and the code for the fields with non-blank entries in Table 36, “Foreign-data wrapper descriptor field default values”, from Table 32, “Codes used for foreign-data wrapper descriptor fields”, as the *FieldIdentifier* parameter. All other fields in the item descriptor areas of *SRD* are initially undefined.

Conformance Rules

None.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

23.3 Foreign-data wrapper interface wrapper routines

23.3.19 Iterate

Function

Retrieve the next row from a FDW-execution.

Definition

```
Iterate (
    ExecutionHandle    IN    INTEGER )
    RETURNS SMALLINT
```

General Rules

- 1) Let *EH* be the value of ExecutionHandle.
- 2) If *EH* does not identify an opened FDW-execution, then an exception condition is raised: *FDW-specific condition — function sequence error*.
- 3) Let *HL1* be the standard host language in which the SQL-server is written and let *HL2* be the standard host language in which the foreign-data wrapper is written.
- 4) Let *S* be the opened FDW-execution identified by ExecutionHandle.
- 5) Let *CR* be the open cursor effectively associated with *S* and let *T* be the table effectively associated with the open cursor.
- 6) Let *SRD* be the server row descriptor for *S* and let *N* be the value of the TOP_LEVEL_COUNT field of *SRD*.
- 7) Case:
 - a) If *HL1* and *HL2* are both pointer-supporting languages, then for each item descriptor area in *SRD* whose LEVEL is 0 (zero) in the first AD item descriptor areas of *SRD*, and for all of their subordinate descriptor areas, refer to a <target specification> whose corresponding item descriptor area has a non-zero value of DATA_POINTER as a *bound target* and refer to the corresponding <select list> column as a *bound column*.
 - b) Otherwise, for each item descriptor area in *SRD* whose LEVEL is 0 (zero) in the first AD item descriptor areas of *SRD*, and for all of their subordinate descriptor areas, refer to a <target specification> as a *bound target* and refer to the corresponding <select list> column as a *bound column*.
- 8) Let *IDA* be the item descriptor area of *SRD* corresponding to the *i*-th bound target and let *TT* be the value of the TYPE field of *IDA*.
- 9) If *TT* indicates DEFAULT, then:
 - a) Case:
 - i) If the PASSTHROUGH flag associated with *EH* is True , then let *RD* be the wrapper row descriptor associated with *S*.
 - ii) Otherwise, let *RD* be the table reference descriptor associated with *S*.

ISO/IEC 9075-9:2001 (E)

23.3 Foreign-data wrapper interface wrapper routines

- b) Let *CT*, *P*, and *SC* be the values of the TYPE, PRECISION, and SCALE fields, respectively, for the item descriptor area of *RD* corresponding to the *i*-th bound column.
 - c) The data type, precision, and scale of the <target specification> described by *IDA* are effectively set to *CT*, *P*, and *SC*, respectively, for the purposes of this invocation of *Iterate()* only.
- 10) If *T* is empty, or if *CR* is positioned after the end of the result set, then:
- a) *CR* is positioned after the last row of *T*.
 - b) No values are assigned to bound targets.
 - c) A completion condition is raised: *no data* and no further rules of this Subclause are applied.
- 11) Case:
- a) If the position of *CR* is before a row *NR*, then *CR* is positioned on row *NR*.
 - b) If the position of *CR* is on a row *OR* other than the last row, then *CR* is positioned on the row immediately after *OR*. Let *NR* be the row immediately after *OR*.
- 12) *NR* becomes the current row of *CR*.
- 13) Case:
- a) If an exception condition is raised during derivation of any <derived column> associated with *NR*, then there is no fetched row associated with *S*, but *NR* remains the current row of *CR*.
 - b) Otherwise:
 - i) *NR* becomes the fetched row associated with *S*.
 - ii) Let *SS* be the select source associated with *S*.
 - iii) The General Rules of Subclause 22.6, “Implicit FETCH USING clause”, are applied with *S* as *OPENED FDW-EXECUTION*.
 - iv) If an exception condition is raised during the derivation of any target value, then the values of all the bound targets are implementation-dependent and *CR* remains positioned on the current row.

Conformance Rules

None.

23.3.20 Open

Function

Open an FDW-execution.

Definition

Open (
 ExecutionHandle IN INTEGER)
 RETURNS SMALLINT

General Rules

- 1) Let *EH* be the value of ExecutionHandle.
- 2) If *EH* does not identify an allocated FDW-execution, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) If *EH* identifies an opened FDW-execution, then an exception condition is raised: *FDW-specific condition — function sequence error*.
- 4) Let *S* be the allocated FDW-execution identified by ExecutionHandle.
- 5) If the PASSTHROUGH flag associated with *EH* is True, then:
 - a) Let *NCR* be the value of the COUNT descriptor field that would be returned by invocation of the GetDescriptor() routine with *WRD* as the DescriptorHandle parameter, 0 (zero) as the RecordNumber parameter, and the code for COUNT from Table 32, “Codes used for foreign-data wrapper descriptor fields”, as the FieldIdentifier parameter.
 - b) Let *DT_j* be the effective data type of the *j*-th column, for $1 \text{ (one)} \leq j \leq NCR$, as represented by the values of the TYPE, LENGTH, OCTET_LENGTH, PRECISION, SCALE, DATETIME_INTERVAL_CODE, DATETIME_INTERVAL_PRECISION, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME, SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME fields that would be returned by separate invocations of the GetDescriptor() routine with *WRD* as the DescriptorHandle parameter, *j* as the RecordNumber parameter, and the code for the fields TYPE, LENGTH, OCTET_LENGTH, PRECISION, SCALE, DATETIME_INTERVAL_CODE, DATETIME_INTERVAL_PRECISION, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME, SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME from Table 32, “Codes used for foreign-data wrapper descriptor fields”, as the FieldIdentifier parameter. TYPE is one of the code values in Table 16, “Codes used for implementation data types in SQL/CLI”.
 - c) Let *TDT_j* be the effective data type of the *j*-th <target specification>, for $1 \text{ (one)} \leq j \leq NCR$, as represented by the values of the TYPE, LENGTH, OCTET_LENGTH, PRECISION, SCALE, DATETIME_INTERVAL_CODE, DATETIME_INTERVAL_PRECISION, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME, SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME fields that would be set by separate invocations of the SetDescriptor() routine with *SRD* as the

23.3 Foreign-data wrapper interface wrapper routines

DescriptorHandle parameter, j as the RecordNumber parameter, and the code for the fields TYPE, LENGTH, OCTET_LENGTH, PRECISION, SCALE, DATETIME_INTERVAL_CODE, DATETIME_INTERVAL_PRECISION, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME, SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME from Table 32, "Codes used for foreign-data wrapper descriptor fields", as the FieldIdentifier parameter. TYPE either indicates ROW or is one of the code values in Table 17, "Codes used for application data types in SQL/CLI".

- d) For every DT_j and TDT_j , $1 \text{ (one)} \leq j \leq NCR$:
 - i) If DT_j is an array data type and TDT_j is not an array locator data type, then an exception condition is raised: *FDW-specific condition — invalid data type descriptors*.
 - ii) If DT_j is a row data type, then

Case:

 - 1) If TDT_j is not a row data type, then an exception condition is raised: *FDW-specific condition — invalid data type descriptors*.
 - 2) If TDT_j is a row data type and DT_j and TDT_j do not conform to the Syntax Rules of Subclause 10.14, "Data type identity", in ISO/IEC 9075-2, then an exception condition is raised: *FDW-specific condition — invalid data type descriptors*.
 - iii) If DT_j and TDT_j are predefined data types, then let HL be the standard programming language in which the invoking SQL-server is written. Let *operative data type correspondence table* be the data type correspondence table for HL as specified in Subclause 20.3, "SQL/CLI data type correspondences". Refer to the two columns of the operative data type correspondence table as the "SQL data type column" and the "host data type column".

Case:

 - 1) If the row that contains the SQL data type corresponding to DT_j in the SQL data type column of the operative data type correspondence table contains "None" in the host data type column, and TDT_j is not a character string data type, then an exception condition is raised: *FDW-specific condition — invalid data type descriptors*.
 - 2) Otherwise, if DT_j and TDT_j do not conform to the Syntax Rules of Subclause 10.14, "Data type identity", in ISO/IEC 9075-2, then an exception condition is raised: *FDW-specific condition — invalid data type descriptors*.
 - iv) If DT_j is a user-defined type, then an exception condition is raised: *FDW-specific condition — invalid data type descriptors*.
- e) Let NCP be the value of the COUNT descriptor field that would be returned by invocation of the GetDescriptor() routine with WPD as the DescriptorHandle parameter, 0 (zero) as the RecordNumber parameter, and the code for COUNT from Table 32, "Codes used for foreign-data wrapper descriptor fields", as the FieldIdentifier parameter.

23.3 Foreign-data wrapper interface wrapper routines

- f) Let PDT_j be the effective data type of the j -th parameter, for $1 \text{ (one)} \leq j \leq NCP$, as represented by the values of the TYPE, LENGTH, OCTET_LENGTH, PRECISION, SCALE, DATETIME_INTERVAL_CODE, DATETIME_INTERVAL_PRECISION, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME, SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME fields that would be returned by separate invocations of the `GetDescriptor()` routine with WPD as the `DescriptorHandle` parameter, j as the `RecordNumber` parameter, and the code for the fields TYPE, LENGTH, OCTET_LENGTH, PRECISION, SCALE, DATETIME_INTERVAL_CODE, DATETIME_INTERVAL_PRECISION, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME, SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME from Table 32, "Codes used for foreign-data wrapper descriptor fields", as the `FieldIdentifier` parameter. TYPE is one of the code values in Table 16, "Codes used for implementation data types in SQL/CLI".
- g) Let $PTDT_j$ be the effective data type of the j -th <target specification>, for $1 \text{ (one)} \leq j \leq NCP$, as represented by the values of the TYPE, LENGTH, OCTET_LENGTH, PRECISION, SCALE, DATETIME_INTERVAL_CODE, DATETIME_INTERVAL_PRECISION, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME, SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME fields that would be returned by separate invocations of the `GetDescriptor()` routine with SPD as the `DescriptorHandle` parameter, j as the `RecordNumber` parameter, and the code for the fields TYPE, LENGTH, OCTET_LENGTH, PRECISION, SCALE, DATETIME_INTERVAL_CODE, DATETIME_INTERVAL_PRECISION, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME, SCOPE_CATALOG, SCOPE_SCHEMA, and SCOPE_NAME from Table 32, "Codes used for foreign-data wrapper descriptor fields", as the `FieldIdentifier` parameter. TYPE either indicates ROW or is one of the code values in Table 17, "Codes used for application data types in SQL/CLI".
- h) For every PDT_j and $PTDT_j$, $1 \text{ (one)} \leq j \leq NCP$:
- If PDT_j is an array data type and $PTDT_j$ is not an array locator data type, then an exception condition is raised: *FDW-specific condition — invalid data type descriptors*.
 - If PDT_j is a row data type, then

Case:

 - If $PTDT_j$ is not a row data type, then an exception condition is raised: *FDW-specific condition — invalid data type descriptors*.
 - If $PTDT_j$ is a row data type and PDT_j and $PTDT_j$ do not conform to the Syntax Rules of Subclause 10.14, "Data type identity", in ISO/IEC 9075-2, then an exception condition is raised: *FDW-specific condition — invalid data type descriptors*.
 - If PDT_j and $PTDT_j$ are predefined data types, then let HL be the standard programming language in which the invoking SQL-server is written. Let *operative data type correspondence table* be the data type correspondence table for HL as specified in Subclause 20.3, "SQL/CLI data type correspondences". Refer to the two columns of the operative data type correspondence table as the "SQL data type column" and the "host data type column".

ISO/IEC 9075-9:2001 (E)

23.3 Foreign-data wrapper interface wrapper routines

Case:

- 1) If the row that contains the SQL data type corresponding to PDT_j in the SQL data type column of the operative data type correspondence table contains "None" in the host data type column, and $PTDT_j$ is not a character string data type, then an exception condition is raised: *FDW-specific condition — invalid data type descriptors*.
- 2) Otherwise, if PDT_j and $PTDT_j$ do not conform to the Syntax Rules of Subclause 10.14, "Data type identity", in ISO/IEC 9075-2, then an exception condition is raised: *FDW-specific condition — invalid data type descriptors*.
- iv) If PDT_j is a user-defined type, then an exception condition is raised: *FDW-specific condition — invalid data type descriptors*.

6) Case:

- a) If the foreign-data request associated with EH returns a set of rows, then:
 - i) The General Rules of Subclause 22.5, "Implicit EXECUTE USING and OPEN USING clauses", are applied to 'OPEN' and S as $TYPE$ and $ALLOCATED\ FDW-EXECUTION$, respectively.
 - ii) The General Rules of Subclause 22.2, "Implicit cursor", are applied to S as $ALLOCATED\ FDW-EXECUTION$.
- b) Otherwise, the General Rules of Subclause 22.5, "Implicit EXECUTE USING and OPEN USING clauses", are applied to 'EXECUTE' and S , as $TYPE$ and $ALLOCATED\ FDW-EXECUTION$, respectively.

7) EH is said to be an *opened FDW-execution*.

Conformance Rules

None.

23.3 Foreign-data wrapper interface wrapper routines

23.3.21 ReOpen

Function

Reopen an FDW-execution.

Definition

ReOpen (
 ExecutionHandle IN INTEGER)
 RETURNS SMALLINT

General Rules

- 1) Let *EH* be the value of ExecutionHandle.
- 2) If *EH* does not identify an allocated FDW-execution, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Let *ODFW* be the opened FDW-execution identified by *EH*.
- 4) Let *CN* be the name of the cursor associated with *ODFW*.
- 5) Cursor *CN* is re-opened in the following steps:
 - a) Let *T* be the table specified by the select source associated with *AS*.
 - b) Cursor *CN* is positioned before the first row of *T*.

Conformance Rules

None.

23.3.22 TransmitRequest**Function**

Supply a statement to be analyzed by the foreign server.

Definition

```
TransmitRequest (
    FSConnectionHandle  IN      INTEGER,
    RequestString        IN      CHARACTER VARYING (L),
    StringLength        IN      INTEGER,
    ExecutionHandle      OUT     INTEGER )
RETURNS SMALLINT
```

where: *L* is determined by the value of *StringLength* and has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

General Rules

- 1) Let *FSCH* be the value of *FSConnectionHandle*.
- 2) If *FSCH* does not identify an allocated FS-connection, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Let *C* be the allocated FS-connection identified by *FSCH*.
- 4) The foreign-data wrapper diagnostics area associated with *C* is emptied.
- 5) Let *FR* be the foreign request associated with the *RequestString*.
- 6) If the maximum number of FDW-executions that can be allocated at one time has already been reached, then *ExecutionHandle* is set to zero and an exception condition is raised: *FDW-specific condition — limit on number of handles exceeded*.
- 7) Case:
 - a) If the memory requirements to manage an FDW-execution cannot be satisfied, then *ReplyHandle* is set to zero and an exception condition is raised: *FDW-specific condition — memory allocation error*.
 - b) If the resources to manage an FDW-execution cannot be allocated for implementation-defined reasons, then *ReplyHandle* is set to zero and an implementation-defined exception condition is raised.
 - c) Otherwise, the resources to manage an FDW-execution are allocated and are referred to as an *allocated FDW-execution*. The allocated FDW-execution is assigned a unique value *RHV* that is returned in *ExecutionHandle*.
- 8) Case:
 - a) If the foreign-data wrapper cannot create an FDW-execution that corresponds to *FR*, then an exception condition is raised: *FDW-specific condition — unable to create reply*.
 - b) Otherwise, the FDW-execution corresponding to *FR* is created.

23.3 Foreign-data wrapper interface wrapper routines

- 9) The PASSTHROUGH flag associated with the allocated FDW-execution is set to True .
- 10) Let *SRDItemDescriptorAreas* be the number of item descriptor areas that need to be set up for the server row descriptor. Let *SRD* be the server row descriptor identified by the DescriptorHandle that is returned by the invocation of the AllocDescriptor() routine with *SRDItemDescriptorAreas* as the MaxDetailAreas parameter. *SRD* is associated with the allocated FDW-execution.

For this descriptor, fields with non-blank entries in Table 36, "Foreign-data wrapper descriptor field default values", are set to the specified default values by the invocation of the SetDescriptor() routine with *SRD* as the DescriptorHandle parameter and *r* as the Record-Number parameter, $1 \text{ (one)} \leq r \leq \textit{SRDItemDescriptorAreas}$, and the code for the fields with non-blank entries in Table 36, "Foreign-data wrapper descriptor field default values", from Table 32, "Codes used for foreign-data wrapper descriptor fields", as the FieldIdentifier parameter. All other fields in the item descriptor areas of *SRD* are initially undefined.

- 11) Let *SPDItemDescriptorAreas* be the number of item descriptor areas that need to be set up for the server parameter descriptor. Let *SPD* be the server parameter descriptor identified by the DescriptorHandle that is returned by the invocation of the AllocDescriptor() routine with *SPDItemDescriptorAreas* as the MaxDetailAreas parameter. *SPD* is associated with the allocated FDW-execution.

For this descriptor, fields with non-blank entries in Table 36, "Foreign-data wrapper descriptor field default values", are set to the specified default values by the invocation of the SetDescriptor() routine with *SPD* as the DescriptorHandle parameter and *r* as the Record-Number parameter, $1 \text{ (one)} \leq r \leq \textit{SPDItemDescriptorAreas}$, and the code for the fields with non-blank entries in Table 36, "Foreign-data wrapper descriptor field default values", from Table 32, "Codes used for foreign-data wrapper descriptor fields", as the FieldIdentifier parameter. All other fields in the item descriptor areas of *SPD* are initially undefined.

- 12) Let *WRDItemDescriptorAreas* be the number of item descriptor areas that need to be set up for the wrapper row descriptor. Let *WRD* be the wrapper row descriptor identified by the DescriptorHandle that is returned by the invocation of the AllocDescriptor() routine with *WRDItemDescriptorAreas* as the MaxDetailAreas parameter. *WRD* is associated with the allocated FDW-execution.

For this descriptor, fields with non-blank entries in Table 36, "Foreign-data wrapper descriptor field default values", are set to the specified default values by the invocation of the SetDescriptor() routine with *WRD* as the DescriptorHandle parameter and *r* as the Record-Number parameter, $1 \text{ (one)} \leq r \leq \textit{WRDItemDescriptorAreas}$, and the code for the fields with non-blank entries in Table 36, "Foreign-data wrapper descriptor field default values", from Table 32, "Codes used for foreign-data wrapper descriptor fields", as the FieldIdentifier parameter. All other fields in the item descriptor areas of *WRD* are initially undefined.

- 13) Let *WPDItemDescriptorAreas* be the number of item descriptor areas that need to be set up for the wrapper parameter descriptor. Let *WPD* be the wrapper parameter descriptor identified by the DescriptorHandle that is returned by the invocation of the AllocDescriptor() routine with *WPDItemDescriptorAreas* as the MaxDetailAreas parameter. *WPD* is associated with the allocated FDW-execution.

For this descriptor, fields with non-blank entries in Table 36, "Foreign-data wrapper descriptor field default values", are set to the specified default values by the invocation of the SetDescriptor() routine with *WPD* as the DescriptorHandle parameter and *r* as the Record-Number parameter, $1 \text{ (one)} \leq r \leq \textit{WPDItemDescriptorAreas}$, and the code for the fields with

ISO/IEC 9075-9:2001 (E)

23.3 Foreign-data wrapper interface wrapper routines

non-blank entries in Table 36, “Foreign-data wrapper descriptor field default values”, from Table 32, “Codes used for foreign-data wrapper descriptor fields”, as the *FieldIdentifier* parameter. All other fields in the item descriptor areas of *WPD* are initially undefined.

- 14) The General Rules of Subclause 22.4, “Implicit DESCRIBE OUTPUT USING clause”, are applied with *RequestString* and *WRD* as *SOURCE* and *DESCRIPTOR*, respectively.
- 15) The General Rules of Subclause 22.3, “Implicit DESCRIBE INPUT USING clause”, are applied with *RequestString* and *WPD* as *SOURCE* and *DESCRIPTOR*, respectively.

Conformance Rules

None.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

23.4 Foreign-data wrapper interface SQL-server routines

23.4.1 AllocDescriptor

Function

Allocate a foreign-data wrapper descriptor area and assign a handle to it.

Definition

```
AllocDescriptor (
    MaxDetailAreas      IN      SMALLINT,
    DescriptorHandle     OUT     INTEGER )
RETURNS SMALLINT
```

General Rules

- 1) Let *MDA* be the value of *MaxDetailAreas*.
- 2) If the maximum number of foreign-data wrapper descriptor areas that can be allocated at one time has already been reached, then *DescriptorHandle* is set to 0 (zero) and an exception condition is raised: *FDW-specific condition — limit on number of handles exceeded*.
- 3) Case:
 - a) If the memory requirements to manage a foreign-data wrapper descriptor area having *MDA* item descriptor areas cannot be satisfied, then *DescriptorHandle* is set to 0 (zero) and an exception condition is raised: *FDW-specific condition — memory allocation error*.
 - b) If the resources to manage a foreign-data wrapper descriptor area cannot be allocated for implementation-defined reasons, then *DescriptorHandle* is set to 0 (zero) and an implementation-defined exception condition is raised.
 - c) Otherwise, the resources to manage a foreign-data wrapper descriptor area are allocated and are referred to as an *allocated foreign-data wrapper descriptor area*. The allocated foreign-data wrapper descriptor area is assigned a unique value that is returned in *DescriptorHandle*.

Conformance Rules

None.

23.4.2 FreeDescriptor

Function

Release resources associated with a foreign-data wrapper descriptor area.

Definition

```
FreeDescriptor (
    DescriptorHandle    IN      INTEGER )
    RETURNS SMALLINT
```

General Rules

- 1) If *DH* does not identify an allocated foreign-data wrapper descriptor area, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 2) Let *D* be the allocated foreign-data wrapper descriptor area identified by *DH*.
- 3) *D* is deallocated and all its resources are freed.

Conformance Rules

None.

23.4 Foreign-data wrapper interface SQL-server routines

23.4.3 GetAuthorizationId

Function

Get the authorization identifier associated with a user mapping.

Definition

```
GetAuthorizationId (
    UserHandle      IN      INTEGER,
    AuthorizationId OUT     CHARACTER(L),
    BufferLength     IN      SMALLINT,
    StringLength    OUT     SMALLINT )
RETURNS SMALLINT
```

where *L* has a maximum value equal to the implementation-defined length of an <identifier>.

General Rules

- 1) Let *UH* be the value of UserHandle.
- 2) If *UH* does not identify an allocated user mapping description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Let *AID* be the authorization identifier associated with *UH*.
- 4) Let *BL* be the value of BufferLength.
- 5) The General Rules of Subclause 22.7, "Character string retrieval", are applied to AuthorizationId, *AID*, *BL*, StringLength as *TARGET*, *VALUE*, *TARGET OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.

Conformance Rules

None.

23.4.4 GetDescriptor

Function

Get a field from a foreign-data wrapper descriptor area.

Definition

```
GetDescriptor (
    DescriptorHandle IN      INTEGER,
    RecordNumber    IN      SMALLINT,
    FieldIdentifier  IN      SMALLINT,
    Value           OUT      ANY,
    BufferLength     IN      INTEGER,
    StringLength    OUT      INTEGER )
RETURNS SMALLINT
```

General Rules

- 1) Let *D* be the allocated foreign-data wrapper descriptor area identified by *DescriptorHandle* and let *N* be the value of the *COUNT* field of *D*.
- 2) Let *FI* be the value of *FieldIdentifier*.
- 3) If *FI* is not one of the code values in Table 32, "Codes used for foreign-data wrapper descriptor fields", then an exception condition is raised: *FDW-specific condition — invalid descriptor field identifier*.
- 4) Let *RN* be the value of *RecordNumber*.
- 5) Let *TYPE* be the value of the *Type* column in the row of Table 32, "Codes used for foreign-data wrapper descriptor fields", that contains *FI*.
- 6) If *TYPE* is 'ITEM', then:
 - a) If *RN* is less than 1 (one), then an exception condition is raised: *dynamic SQL error — invalid descriptor index*.
 - b) If *RN* is greater than *N*, then a completion condition is raised: *no data*.
- 7) Let *MBR* be the value of the *May Be Retrieved* column in the row of Table 34, "Ability to retrieve foreign-data wrapper descriptor fields", that contains *FI* and the column that contains the descriptor type *D*.
- 8) If *MBR* is 'No', then an exception condition is raised: *FDW-specific condition — invalid descriptor field identifier*.
- 9) If *FI* indicates a descriptor field whose value is the initially undefined value created when the descriptor was created, then an exception condition is raised: *FDW-specific condition — invalid descriptor field identifier*.
- 10) Let *IDA* be the foreign-data wrapper item descriptor area of *D* specified by *RN*.
- 11) If *TYPE* is 'HEADER', then header information from the descriptor area *D* is retrieved as follows.

23.4 Foreign-data wrapper interface SQL-server routines

Case:

- a) If *FI* indicates COUNT, then the value retrieved is *N*.
- b) If *FI* indicates an implementation-defined descriptor header field, then the value retrieved is the value of the implementation-defined descriptor header field identified by *FI*.
- c) Otherwise, if *FI* indicates a descriptor header field defined in Table 32, "Codes used for foreign-data wrapper descriptor fields", then the value retrieved is the value of the descriptor header field identified by *FI*.

- 12) If *TYPE* is 'ITEM', then item information from the descriptor area *D* is retrieved as follows:

Case:

- a) If *FI* indicates an implementation-defined descriptor item field, then the value retrieved is the value of the implementation-defined descriptor item field of *IDA* identified by *FI*.
- b) Otherwise, if *FI* indicates a descriptor item field defined in Table 32, "Codes used for foreign-data wrapper descriptor fields", then the value retrieved is the value of the descriptor header field identified by *FI*.

- 13) Let *V* be the value retrieved.

- 14) If *FI* indicates a descriptor field whose row in Table 5, "Fields in foreign-data wrapper descriptor areas", contains a Data Type that is not CHARACTER VARYING, then Value is set to *V* and no further rules of this Subclause are applied.

- 15) Let *BL* be the value of BufferLength.

- 16) If *FI* indicates a descriptor field whose row in Table 5, "Fields in foreign-data wrapper descriptor areas", contains a Data Type that is CHARACTER VARYING, then the General Rules of Subclause 22.7, "Character string retrieval", are applied with Value, *V*, *BL*, and StringLength as *TARGET*, *VALUE*, *TARGET OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.

Conformance Rules

None.

23.4.5 GetNumSelectElems

Function

Get the number of <value expressions>s in the <select list> of a query.

Definition

```
GetNumSelectElems (
    RequestHandle          IN      INTEGER,
    NumberOfSelectListElements OUT  SMALLINT )
    RETURNS SMALLINT
```

General Rules

- 1) Let *RH* be the value of RequestHandle.
- 2) If *RH* does not identify an allocated request description, then an exception condition is raised:
FDW-specific condition — invalid handle.
- 3) Let *Q* be the query associated with *RH*.
- 4) Let *N* be the number of <value expression> elements simply contained in the <select list> of *Q*.
- 5) NumberOfSelectListElements is set to *N*.

Conformance Rules

None.

23.4 Foreign-data wrapper interface SQL-server routines

23.4.6 GetNumServerOpts

Function

Get the number of generic options associated with the foreign server.

Definition

```
GetNumServerOpts (
    ServerHandle      IN      INTEGER,
    OptionCount       OUT     INTEGER )
RETURNS SMALLINT
```

General Rules

- 1) Let *SH* be the value of *ServerHandle*.
- 2) If *SH* does not identify an allocated foreign server description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) *OptionCount* is set to the number of generic options associated with *SH*.

Conformance Rules

None.

23.4.7 GetNumTableColOpts**Function**

Get the number of generic options of a column of a foreign table.

Definition

GetNumTableColOpts (		
TableReferenceHandle	IN	INTEGER,
ColumnName	IN	CHARACTER(L),
NameLength	IN	SMALLINT,
OptionCount	OUT	INTEGER)
RETURNS SMALLINT		

where L and has a maximum value equal to the implementation-defined length of a variable-length character string.

General Rules

- 1) Let TRH be the value of TableReferenceHandle.
- 2) If TRH does not identify an allocated table reference description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Let NL be the value of NameLength.
- 4) Case:
 - a) If NL is not negative, then let L be NL .
 - b) Otherwise, an exception condition is raised: *FDW-specific condition — invalid string length or buffer length*.
- 5) Case:
 - a) If L is zero, then an exception condition is raised: *FDW-specific condition — invalid string length or buffer length*.
 - b) Otherwise:
 - i) Let N be the number of whole characters in the first L octets of ColumnName and let NO be the number of octets occupied by those N characters. If $NO \neq L$, then an exception condition is raised: *FDW-specific condition — invalid column name*.
 - ii) Otherwise, let CN be the first L octets of ColumnName and let TCN be the value of

TRIM (BOTH ' ' FROM CN)
- 6) Let NC be the number of columns of the foreign table referenced by TRH .
- 7) Case:
 - a) If TCN is not equivalent to the name of a column of the foreign table referenced by TRH , then an exception condition is raised: *FDW-specific condition — column name not found*.

23.4 Foreign-data wrapper interface SQL-server routines

b) Otherwise:

- i) Let *ON* be the number of generic options of the column of the foreign table referenced by *TRH* whose name is *TCN*.
- ii) OptionCount is set to *ON*.

Conformance Rules

None.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

23.4.8 GetNumTableOpts

Function

Get the number of generic options of a foreign table.

Definition

```
GetNumTableOpts (
    TableReferenceHandle      IN      INTEGER,
    OptionCount               OUT     INTEGER )
    RETURNS SMALLINT
```

General Rules

- 1) Let *TRH* be the value of TableReferenceHandle.
- 2) If *TRH* does not identify an allocated table reference description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Let *N* be the number of generic options of the foreign table referenced by *TRH*.
- 4) OptionCount is set to *N*.

Conformance Rules

None.

23.4.9 GetNumTableRefElems**Function**

Get the number of <table reference>s contained in the <from clause> of a query.

Definition

```
GetNumTableRefElems (
    RequestHandle          IN      INTEGER,
    NumberOfTableReferences OUT    SMALLINT )
    RETURNS SMALLINT
```

General Rules

- 1) Let *RH* be the value of RequestHandle.
- 2) If *RH* does not identify an allocated request description, then an exception condition is raised:
FDW-specific condition — invalid handle.
- 3) Let *Q* be the query associated with *RH*.
- 4) Let *N* be the number of <table reference> elements simply contained in the <from clause> of *Q*.
- 5) NumberOfTableReferenceElements is set to *N*.

Conformance Rules

None.

23.4.10 GetNumUserOpts

Function

Get the number of generic options of a user mapping.

Definition

```
GetNumUserOpts (
    UserHandle          IN    INTEGER,
    OptionCount         OUT   INTEGER )
    RETURNS SMALLINT
```

General Rules

- 1) Let *UH* be the value of UserHandle.
- 2) If *UH* does not identify an allocated user mapping description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Let *N* be the number of generic options of the user mapping described by *UH*.
- 4) OptionCount is set to *N*.

Conformance Rules

None.

23.4 Foreign-data wrapper interface SQL-server routines

23.4.11 GetNumWrapperOpts

Function

Get the number of generic options of a foreign-data wrapper.

Definition

```

GetNumWrapperOpts (
    WrapperHandle      IN      INTEGER,
    OptionCount        OUT    INTEGER )
    RETURNS SMALLINT
  
```

General Rules

- 1) Let *WH* be the value of WrapperHandle.
- 2) If *WH* does not identify an allocated foreign-data wrapper description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Let *N* be the number of generic options of the foreign-data wrapper described by *WH*.
- 4) OptionCount is set to *N*.

Conformance Rules

None.

23.4.12 GetSelectElem

Function

Get a <value expression> element from the <select list> of a query.

Definition

```
GetSelectElem (
    RequestHandle          IN      INTEGER,
    SelectListElementNumber IN      SMALLINT,
    ValueExpressionHandle  OUT     INTEGER )
RETURNS SMALLINT
```

General Rules

- 1) Let *RH* be the value of RequestHandle.
- 2) If *RH* does not identify an allocated request description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Let *SLEN* be the value of SelectListElementNumber.
- 4) If *SLEN* is less than 1 (one), then an exception condition is raised: *FDW-specific condition — invalid option index*.
- 5) Let *Q* be the query associated with *RH*.
- 6) Let *N* be the number of <value expression> elements simply contained in the <select list> of *Q*.
- 7) If *SLEN* is greater than *N*, then a completion condition is raised: *no data*, and no further rules of this Subclause are applied.
- 8) Information from the *SLEN*-th <value expression> associated with *Q* is retrieved.
 - a) Let *VEH* be the value expression handle associated with the <value expression>.
 - b) ValueExpressionHandle is set to *VEH*.

Conformance Rules

None.

23.4.13 GetSelectElemType**Function**

Get the type of a <value expression>

Definition

```
GetSelectElemType (
    ValueExpressionHandle    IN    INTEGER,
    ValueExpressionType      OUT   SMALLINT )
    RETURNS SMALLINT
```

General Rules

- 1) Let *VEH* be the value of ValueExpressionHandle.
- 2) If *VEH* does not identify an allocated <value expression> description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Let *VET* be the type of the <value expression> associated with *VEH*.
- 4) ValueExpressionType is set to *VET*.

Conformance Rules

None.

ISO/IEC 9075-9:2001 (E)
23.4 Foreign-data wrapper interface SQL-server routines

23.4.14 GetServerName

Function

Get the name of the foreign server.

Definition

```
GetServerName (
    ServerHandle      IN      INTEGER,
    ServerName        OUT     CHARACTER(L),
    BufferLength       IN      SMALLINT,
    StringLength      OUT     SMALLINT )
RETURNS SMALLINT
```

where L has a maximum value equal to $(2n+1)$, where n is the implementation-defined length of an <identifier>.

NOTE 69 – The length $(2n+1)$ supports the syntax of <server name>, which is "<identifier><period><identifier>".

General Rules

- 1) Let SH be the value of ServerHandle.
- 2) If SH does not identify an allocated foreign server description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Let SN be the server name associated with SH .
- 4) Let BL be the value of BufferLength.
- 5) The General Rules of Subclause 22.7, "Character string retrieval", are applied to ServerName, SN , BL , and StringLength as *TARGET*, *VALUE*, *TARGET OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.

Conformance Rules

None.

23.4.15 GetServerOpt

Function

Get the value of a generic option associated with a foreign server.

Definition

```
GetServerOpt (
    ServerHandle      IN      INTEGER,
    OptionNumber      IN      INTEGER,
    OptionName        OUT     CHARACTER(L1),
    BufferLength1      IN      SMALLINT,
    StringLength1     OUT     SMALLINT,
    OptionValue       OUT     CHARACTER(L2),
    BufferLength2      IN      SMALLINT,
    StringLength2     OUT     SMALLINT )
RETURNS SMALLINT
```

where each of *L1* and *L2* has a maximum value of equal to the implementation-defined length of a variable-length character string.

General Rules

- 1) Let *SH* be the value of *ServerHandle*.
- 2) If *SH* does not identify an allocated foreign server description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Let *ON* be the value of *OptionNumber*.
- 4) Let *N* be the number of generic options associated with *SH*.
- 5) If *ON* is less than 1 (one), then an exception condition is raised: *FDW-specific condition — invalid option index*.
- 6) If *ON* is greater than *N*, then a completion condition is raised: *no data*, and no further rules of this Subclause are applied.
- 7) Information from the *ON*-th generic option associated with the *SH* is retrieved.
 - a) Let *NAME* be the name of the generic option.
 - b) The General Rules of Subclause 22.7, “Character string retrieval”, are applied to *Option-Name*, *NAME*, *BufferLength1*, and *StringLength1* as *TARGET*, *VALUE*, *TARGET OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.
 - c) Let *OPTIONVALUE* be the value of the generic option.
 - d) The General Rules of Subclause 22.7, “Character string retrieval”, are applied to *Option-Value*, *OPTIONVALUE*, *BufferLength2*, and *StringLength2* as *TARGET*, *VALUE*, *TARGET OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.

ISO/IEC 9075-9:2001 (E)

23.4 Foreign-data wrapper interface SQL-server routines

Conformance Rules

None.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

23.4.16 GetServerOptByName**Function**

Get the value of a generic option associated with a foreign server.

Definition

```

GetServerOptByName (
    ServerHandle      IN      INTEGER,
    OptionName        IN      CHARACTER(L1),
    BufferLength1      IN      SMALLINT,
    OptionValue       OUT     CHARACTER(L2),
    BufferLength2      IN      SMALLINT,
    StringLength2     OUT     SMALLINT )
RETURNS SMALLINT

```

where each of *L1* and *L2* has a maximum value of equal to the implementation-defined length of a variable-length character string.

General Rules

- 1) Let *SH* be the value of *ServerHandle*.
- 2) If *SH* does not identify an allocated foreign server description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Let *BL* be the value of *BufferLength1*.
- 4) Case:
 - a) If *BL* is not negative, then let *L* be *BL*.
 - b) Otherwise, an exception condition is raised: *FDW-specific condition — invalid string length or buffer length*.
- 5) Case:
 - a) If *L* is zero, then an exception condition is raised: *FDW-specific condition — invalid string length or buffer length*.
 - b) Otherwise:
 - i) Let *N* be the number of whole characters in the first *L* octets of *OptionName* and let *NO* be the number of octets occupied by those *N* characters. If *NO* ≠ *L*, then an exception condition is raised: *FDW-specific condition — invalid option name*.
 - ii) Otherwise, let *ON* be the first *L* octets of *OptionName* and let *TON* be the value of:

TRIM (BOTH ' ' FROM ON)
- 6) Let *N* be the number of generic options associated with *SH*.

ISO/IEC 9075-9:2001 (E)

23.4 Foreign-data wrapper interface SQL-server routines

7) Case:

- a) If *TON* is equivalent to the name of a generic option associated with *SH*, then:
 - i) Let *OPTIONVALUE* be the value of the generic option associated with *SH* whose name is equivalent to *TON*.
 - ii) The General Rules of Subclause 22.7, “Character string retrieval”, are applied to *OptionValue*, *OPTIONVALUE*, *BufferLength2*, and *StringLength2* as *TARGET*, *VALUE*, *TARGET OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.
- b) Otherwise, an exception condition is raised: *FDW-specific condition — option name not found*.

Conformance Rules

None.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

23.4 Foreign-data wrapper interface SQL-server routines

23.4.17 GetServerType

Function

Get the type of a foreign server.

Definition

```
GetServerType (
    ServerHandle      IN      INTEGER,
    ServerType        OUT     CHARACTER(L),
    BufferLength       IN      SMALLINT,
    StringLength      OUT     SMALLINT )
RETURNS SMALLINT
```

where *L* has a maximum value of equal to the implementation-defined length of a variable-length character string.

General Rules

- 1) Let *SH* be the value of *ServerHandle*.
- 2) If *SH* does not identify an allocated foreign server description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Let *ST* be the server type associated with *SH*.
- 4) Let *BL* be the value of *BufferLength*.
- 5) The General Rules of Subclause 22.7, “Character string retrieval”, are applied to *ServerType*, *ST*, *BL*, and *StringLength* as *TARGET*, *VALUE*, *TARGET OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.

Conformance Rules

None.

23.4.18 GetServerVersion

Function

Get the version of a foreign server.

Definition

```
GetServerVersion (
    ServerHandle      IN      INTEGER,
    ServerVersion     OUT     CHARACTER(L),
    BufferLength       IN      SMALLINT,
    StringLength      OUT     SMALLINT )
RETURNS SMALLINT
```

where *L* has a maximum value of equal to the implementation-defined length of a variable-length character string.

General Rules

- 1) Let *SH* be the value of *ServerHandle*.
- 2) If *SH* does not identify an allocated foreign server description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Let *SV* be the server version associated with *SH*.
- 4) Let *BL* be the value of *BufferLength*.
- 5) The General Rules of Subclause 22.7, “Character string retrieval”, are applied to *ServerVersion*, *SV*, *BL*, and *StringLength* as *TARGET*, *VALUE*, *TARGET OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.

Conformance Rules

None.

23.4 Foreign-data wrapper interface SQL-server routines

23.4.19 GetSQLString

Function

Get a character string representation of the query that is associated with the request handle.

Definition

```
GetSQLString (
    RequestHandle      IN      INTEGER,
    StringFormat       IN      INTEGER,
    SQLString          OUT     CHARACTER VARYING(L),
    BufferLength        IN      INTEGER,
    StringLength       OUT     INTEGER )
RETURNS SMALLINT
```

where: *L* is determined by the value of *StringLength* and has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

General Rules

- 1) Let *RH* be the value of *RequestHandle*.
- 2) If *RH* does not identify an allocated request description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Let *Q* be the query associated with *RH*.
- 4) Let *SF* be the value of *StringFormat*.
- 5) If *SF* does not identify a value that is equal to any value in Table 37, "Codes used for the format of the character string transmitted by *GetSQLString()*", then an exception condition is raised: *FDW-specific condition — invalid string format*.
- 6) Case:
 - a) If *SF* identifies an implementation-defined value, then let *QueryString* be the implementation-defined character string representation of *Q*.
 - b) Otherwise, let *QueryString* be a character string conforming to the Format and Syntax Rules of Subclause 15.4, "<SQL procedure statement>".
- 7) The General Rules of Subclause 22.7, "Character string retrieval", are applied to *SQLString*, *QueryString*, *BufferLength*, and *StringLength* as *TARGET*, *VALUE*, *TARGET OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.

Conformance Rules

None.

23.4.20 GetTableColOpt**Function**

Get the name and value of a generic option associated with a column of a foreign table, given an option number.

Definition

```
GetTableColOpt (
    TableReferenceHandle IN    INTEGER,
    ColumnName           IN    CHARACTER(L),
    NameLength           IN    SMALLINT,
    OptionNumber         IN    INTEGER,
    OptionName           OUT   CHARACTER(L1),
    BufferLength1         IN    SMALLINT,
    StringLength1        OUT   SMALLINT,
    OptionValue          OUT   CHARACTER(L2),
    BufferLength2         IN    SMALLINT,
    StringLength2        OUT   SMALLINT )
RETURNS SMALLINT
```

where each of L , $L1$, and $L2$ has a maximum value equal to the implementation-defined length of a variable-length character string.

General Rules

- 1) Let TRH be the value of TableReferenceHandle.
- 2) If TRH does not identify an allocated table reference description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Let NL be the value of NameLength.
- 4) Case:
 - a) If NL is not negative, then let L be NL .
 - b) Otherwise, an exception condition is raised: *FDW-specific condition — invalid string length or buffer length*.
- 5) Case:
 - a) If L is zero, then an exception condition is raised: *FDW-specific condition — invalid string length or buffer length*.
 - b) Otherwise:
 - i) Let N be the number of whole characters in the first L octets of ColumnName and let NO be the number of octets occupied by those N characters. If $NO \neq L$, then an exception condition is raised: *FDW-specific condition — invalid column name*.
 - ii) Otherwise, let CN be the first L octets of ColumnName and let TCN be the value of

TRIM (BOTH ' ' FROM CN)

23.4 Foreign-data wrapper interface SQL-server routines

- 6) Let *NC* be the number of columns of the foreign table referenced by *TRH*.
- 7) Case:
- a) If *TCN* is not equivalent to the name of a column of the foreign table referenced by *TRH*, then an exception condition is raised: *FDW-specific condition — column name not found*.
 - b) Otherwise:
 - i) Let *ON* be the value of OptionNumber.
 - ii) Let *N* be the number of generic options associated with the column identified by *TCN*.
 - iii) If *ON* is less than 1 (one), then an exception condition is raised: *FDW-specific condition — invalid option index*.
 - iv) If *ON* is greater than *N*, then a completion condition is raised: *no data* and no further rules of this Subclause are applied.
 - v) Information from the *ON*-th generic option associated with the column identified by *TCN* is retrieved.
 - 1) Let *NAME* be the name of the generic option.
 - 2) The General Rules of Subclause 22.7, “Character string retrieval”, are applied to OptionName, *NAME*, BufferLength1, and StringLength1 as *TARGET*, *VALUE*, *TARGET OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.
 - 3) Let *VALUE* be the value of the generic option.
 - 4) The General Rules of Subclause 22.7, “Character string retrieval”, are applied to OptionValue, *VALUE*, BufferLength2, and StringLength2 as *TARGET*, *VALUE*, *TARGET OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.

Conformance Rules

None.

23.4.21 GetTableColOptByName**Function**

Get the value of a generic option associated with a column of a foreign table, given the option name.

Definition

```
GetTableColOptByName (
    TableReferenceHandle IN    INTEGER,
    ColumnName           IN    CHARACTER(L),
    NameLength           IN    SMALLINT,
    OptionName           IN    CHARACTER(L1),
    BufferLength1         IN    SMALLINT,
    OptionValue          OUT   CHARACTER(L2),
    BufferLength2         IN    SMALLINT,
    StringLength2        OUT   SMALLINT )
RETURNS SMALLINT
```

where each of L , $L1$, and $L2$ has a maximum value equal to the implementation-defined length of a variable-length character string.

General Rules

- 1) Let TRH be the value of TableReferenceHandle.
- 2) If TRH does not identify an allocated table reference description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Let NL be the value of NameLength.
- 4) Case:
 - a) If NL is not negative, then let L be NL .
 - b) Otherwise, an exception condition is raised: *FDW-specific condition — invalid string length or buffer length*.
- 5) Case:
 - a) If L is zero, then an exception condition is raised: *FDW-specific condition — invalid string length or buffer length*.
 - b) Otherwise:
 - i) Let N be the number of whole characters in the first L octets of ColumnName and let NO be the number of octets occupied by those N characters. If $NO \neq L$, then an exception condition is raised: *FDW-specific condition — invalid column name*.
 - ii) Otherwise, let CN be the first L octets of ColumnName and let TCN be the value of

TRIM (BOTH ' ' FROM CN)

- 6) Let NC be the number of columns of the foreign table referenced by TRH .

23.4 Foreign-data wrapper interface SQL-server routines

7) Case:

a) If *TCN* is not equivalent to the name of a column of the foreign table referenced by *TRH*, then an exception condition is raised: *FDW-specific condition — column name not found*.

b) Otherwise:

i) Let *BL* be the value of *BufferLength1*.

ii) Case:

1) If *BL* is not negative, then let *BL1* be *BL*.

2) Otherwise, an exception condition is raised: *FDW-specific condition — invalid string length or buffer length*.

iii) Case:

1) If *BL1* is zero, then an exception condition is raised: *FDW-specific condition — invalid string length or buffer length*.

2) Otherwise:

A) Let *BN* be the number of whole characters in the first *BL1* octets of *OptionName* and let *BNO* be the number of octets occupied by those *BN* characters. If *BNO* $\neq$ *BL1*, then an exception condition is raised: *FDW-specific condition — invalid option name*.

B) Otherwise, let *ON* be the first *BL1* octets of *OptionName* and let *TON* be the value of

TRIM (BOTH FROM *ON*)

iv) Let *N* be the number of generic options associated with *TRH*.

v) Case:

1) If *TON* is not equivalent to the name of a generic option associated with *TRH*, then an exception condition is raised: *FDW-specific condition — option name not found*.

2) Otherwise:

A) Let *VALUE* be the value of the generic option associated with *TRH* whose name is *TON*.

B) The General Rules of Subclause 22.7, "Character string retrieval", are applied to *OptionValue*, *VALUE*, *BufferLength2*, and *StringLength2* as *TARGET*, *VALUE*, *TARGET OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.

Conformance Rules

None.

23.4.22 GetTableOpt

Function

Get the name and value of a generic option associated with a foreign table, given an option number.

Definition

```
GetTableOpt (
    TableReferenceHandle IN    INTEGER,
    OptionNumber         IN    INTEGER,
    OptionName           OUT   CHARACTER(L1),
    BufferLength1         IN    SMALLINT,
    StringLength1        OUT   SMALLINT,
    OptionValue          OUT   CHARACTER(L2),
    BufferLength2         IN    SMALLINT,
    StringLength2        OUT   SMALLINT )
RETURNS SMALLINT
```

where each of *L1* and *L2* has a maximum value equal to the implementation-defined length of a variable-length character string.

General Rules

- 1) Let *TRH* be the value of TableReferenceHandle.
- 2) If *TRH* does not identify an allocated table reference description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Let *ON* be the value of OptionNumber.
- 4) Let *N* be the number of generic options associated with *TRH*.
- 5) If *ON* is less than 1 (one), then an exception condition is raised: *FDW-specific condition — invalid option index*.
- 6) If *ON* is greater than *N*, then a completion condition is raised: *no data* and no further rules of this Subclause are applied.
- 7) Information from the *ON*-th generic option associated with *TRH* is retrieved.
 - a) Let *NAME* be the name of the generic option.
 - b) The General Rules of Subclause 22.7, “Character string retrieval”, are applied to Option-Name, *NAME*, BufferLength1, and StringLength1 as *TARGET*, *VALUE*, *TARGET OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.
 - c) Let *VALUE* be the value of the generic option.
 - d) The General Rules of Subclause 22.7, “Character string retrieval”, are applied to Option-Value, *VALUE*, BufferLength2, and StringLength2 as *TARGET*, *VALUE*, *TARGET OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.

Conformance Rules

None.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

23.4.23 GetTableOptByName

Function

Get the value of a generic option associated with a foreign table, given the option name.

Definition

```
GetTableOptByName (
    TableReferenceHandle IN      INTEGER,
    OptionName           IN      CHARACTER(L1),
    BufferLength1         IN      SMALLINT,
    OptionValue          OUT     CHARACTER(L2),
    BufferLength2         IN      SMALLINT,
    StringLength2        OUT     SMALLINT )
RETURNS SMALLINT
```

where each of *L1* and *L2* has a maximum value equal to the implementation-defined length of a variable-length character string.

General Rules

- 1) Let *TRH* be the value of TableReferenceHandle.
- 2) If *TRH* does not identify an allocated table reference description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Let *NL* be the value of BufferLength1.
- 4) Case:
 - a) If *NL* is not negative, then let *L* be *NL*.
 - b) Otherwise, an exception condition is raised: *FDW-specific condition — invalid string length or buffer length*.
- 5) Case:
 - a) If *L* is zero, then an exception condition is raised: *FDW-specific condition — invalid string length or buffer length*.
 - b) Otherwise:
 - i) Let *N* be the number of whole characters in the first *L* octets of OptionName and let *NO* be the number of octets occupied by those *N* characters. If *NO* ≠ *L*, then an exception condition is raised: *FDW-specific condition — invalid option name*.
 - ii) Otherwise, let *ON* be the first *L* octets of OptionName and let *TON* be the value of

TRIM (BOTH ' ' FROM *ON*)

- 6) Let *N* be the number of generic options associated with *TRH*.

23.4 Foreign-data wrapper interface SQL-server routines

7) Case:

- a) If *TON* is not equivalent to the name of a generic option associated with *TRH*, then an exception condition is raised: *FDW-specific condition — option name not found*.
- b) Otherwise:
 - i) Let *VALUE* be the value of the generic option associated with *TRH* whose name is *TON*.
 - ii) The General Rules of Subclause 22.7, “Character string retrieval”, are applied to *OptionValue*, *VALUE*, *BufferLength2*, and *StringLength2* as *TARGET*, *VALUE*, *TARGET OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.

Conformance Rules

None.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

23.4.24 GetTableRefElem**Function**

Get a <table reference> element from the <from clause> of a query.

Definition

```
GetTableRefElem (
    RequestHandle           IN      INTEGER,
    TableReferenceElementNumber IN    INTEGER,
    TableReferenceHandle    OUT    INTEGER )
    RETURNS SMALLINT
```

General Rules

- 1) Let *RH* be the value of RequestHandle.
- 2) If *RH* does not identify an allocated request description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Let *TREN* be the value of TableReferenceElementNumber.
- 4) If *TREN* is less than 1 (one), then an exception condition is raised: *FDW-specific condition — invalid option index*.
- 5) Let *Q* be the query associated with *RH*.
- 6) Let *N* be the number of <table reference> elements in the <from clause> of *Q*.
- 7) If *TREN* is greater than *N*, then a completion condition is raised: *no data*, and no further rules of this Subclause are applied.
- 8) Information from the *TREN*-th <table reference> associated with *Q* is retrieved.
 - a) Let *TRH* be the table reference handle associated with the <table reference>.
 - b) TableReferenceHandle is set to *TRH*.

Conformance Rules

None.

23.4.25 GetTableRefElemType**Function**

Get the type of a <table reference>.

Definition

```
GetTableRefElemType (
    TableReferenceHandle    IN    INTEGER,
    TableReferenceType      OUT   SMALLINT )
    RETURNS SMALLINT
```

General Rules

- 1) Let *TRH* be the value of TableReferenceHandle.
- 2) If *TRH* does not identify an allocated table reference description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Let *TRT* be the type of the <table reference> associated with *TRH*.
- 4) TableReferenceType is set to *TRT*.

Conformance Rules

None.

23.4.26 GetTableRefTableName**Function**

Get the table name of a TABLE_NAME <table reference>.

Definition

```
GetTableRefTableName (
    TableReferenceHandle    IN    INTEGER,
    TableName               OUT   CHARACTER(L),
    BufferLength             IN    SMALLINT,
    StringLength            OUT   SMALLINT )
    RETURNS SMALLINT
```

where *L* has a maximum value of equal to the implementation-defined length of a variable-length character string.

General Rules

- 1) Let *TRH* be the value of TableReferenceHandle.
- 2) If *TRH* does not identify an allocated table reference description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) If *TRH* does not identify a <table reference> with a type of TABLE_NAME, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 4) Let *NAME* be the value of the table name of the <table reference>.
- 5) Let *BL* be the value of BufferLength.
- 6) The General Rules of Subclause 22.7, “Character string retrieval”, are applied to TableName, NAME, BL, and StringLength as TARGET, VALUE, TARGET OCTET LENGTH, and RETURNED OCTET LENGTH, respectively.

Conformance Rules

None.

23.4.27 GetTableServerName**Function**

Get the name of the foreign server associated with a foreign table.

Definition

```
GetTableServerName (
    TableReferenceHandle IN    INTEGER,
    ServerName           OUT   CHARACTER(L),
    BufferLength          IN    SMALLINT,
    StringLength         OUT   SMALLINT )
    RETURNS SMALLINT
```


23.4 Foreign-data wrapper interface SQL-server routines

where *L* has a maximum value equal to the implementation-defined length of an <identifier>.

General Rules

- 1) Let *TRH* be the value of TableReferenceHandle.
- 2) If *TRH* does not identify an allocated table reference description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Let *SN* be the server name associated with the foreign table identified by *TRH*.
- 4) Let *BL* be the value of BufferLength.
- 5) The General Rules of Subclause 22.7, “Character string retrieval”, are applied to *ServerName*, *SN*, *BL*, and *StringLength* as *TARGET*, *VALUE*, *TARGET OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.

Conformance Rules

None.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

23.4.28 GetTRDHandle

Function

Get the descriptor handle of the table reference descriptor associated with a TableReferenceHandle.

Definition

```
GetTRDHandle (
    TableReferenceHandle IN    INTEGER,
    TRDHandle           OUT   INTEGER )
    RETURNS SMALLINT
```

General Rules

- 1) Let *TRH* be the value of TableReferenceHandle.
- 2) If *TRH* does not identify an allocated table reference description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Let *TRDH* be the descriptor handle of the table reference descriptor associated with *TRH*.
- 4) TRDHandle is set to *TRDH*.

Conformance Rules

None.

23.4 Foreign-data wrapper interface SQL-server routines

23.4.29 GetUserOpt

Function

Get the name and value of a generic option associated with a user mapping, given an option number.

Definition

```

GetUserOpt (
    UserHandle          IN      INTEGER,
    OptionNumber        IN      INTEGER,
    OptionName          OUT     CHARACTER(L1),
    BufferLength1        IN      SMALLINT,
    StringLength1       OUT     SMALLINT,
    OptionValue         OUT     CHARACTER(L2),
    BufferLength2        IN      SMALLINT,
    StringLength2       OUT     SMALLINT )
RETURNS SMALLINT

```

where each of *L1* and *L2* has a maximum value equal to the implementation-defined length of a variable-length character string.

General Rules

- 1) Let *UH* be the value of *UserHandle*.
- 2) If *UH* does not identify an allocated user mapping description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Let *ON* be the value of *OptionNumber*.
- 4) Let *N* be the number of generic options associated with *UH*.
- 5) If *ON* is less than 1 (one), then an exception condition is raised: *FDW-specific condition — invalid option index*.
- 6) If *ON* is greater than *N*, then a completion condition is raised: *no data* and no further rules of this Subclause are applied.
- 7) Information from the *ON*-th generic option associated with *UH* is retrieved.
 - a) Let *NAME* be the name of the generic option.
 - b) The General Rules of Subclause 22.7, “Character string retrieval”, are applied to *OptionName*, *NAME*, *BufferLength1*, and *StringLength1* as *TARGET*, *VALUE*, *TARGET OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.
 - c) Let *OPTIONVALUE* be the value of the generic option.
 - d) The General Rules of Subclause 22.7, “Character string retrieval”, are applied to *OptionValue*, *OPTIONVALUE*, *BufferLength2*, and *StringLength2* as *TARGET*, *VALUE*, *TARGET OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.

ISO/IEC 9075-9:2001 (E)

23.4 Foreign-data wrapper interface SQL-server routines

Conformance Rules

None.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

23.4.30 GetUserOptByName**Function**

Get the value of a generic option associated with a user mapping, given the option name.

Definition

```

GetUserOptByName (
    UserHandle          IN      INTEGER,
    OptionName          IN      CHARACTER(L1),
    BufferLength1        IN      SMALLINT,
    OptionValue         OUT     CHARACTER(L2),
    BufferLength2        IN      SMALLINT,
    StringLength2       OUT     SMALLINT )
RETURNS SMALLINT

```

where each of *L1* and *L2* has a maximum value equal to the implementation-defined length of a variable-length character string.

General Rules

- 1) Let *UH* be the value of *UserHandle*.
- 2) If *UH* does not identify an allocated user mapping description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Let *NL* be the value of *BufferLength1*.
- 4) Case:
 - a) If *NL* is not negative, then let *L* be *NL*.
 - b) Otherwise, an exception condition is raised: *FDW-specific condition — invalid string length or buffer length*.
- 5) Case:
 - a) If *L* is zero, then an exception condition is raised: *FDW-specific condition — invalid string length or buffer length*.
 - b) Otherwise:
 - i) Let *N* be the number of whole characters in the first *L* octets of *OptionName* and let *NO* be the number of octets occupied by those *N* characters. If *NO* ≠ *L*, then an exception condition is raised: *FDW-specific condition — invalid option name*.
 - ii) Otherwise, let *ON* be the first *L* octets of *OptionName* and let *TON* be the value of

TRIM (BOTH ' ' FROM *ON*)

- 6) Let *N* be the number of generic options associated with *UH*.

ISO/IEC 9075-9:2001 (E)

23.4 Foreign-data wrapper interface SQL-server routines

7) Case:

- a) If *TON* is equivalent to the name of a generic option associated with *UH*, then:
 - i) Let *OPTIONVALUE* be the value of the generic option associated with *UH* whose name is equivalent to *TON*.
 - ii) The General Rules of Subclause 22.7, “Character string retrieval”, are applied to *OptionValue*, *OPTIONVALUE*, *BufferLength2*, and *StringLength2* as *TARGET*, *VALUE*, *TARGET OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.
- b) Otherwise, an exception condition is raised: *FDW-specific condition — option name not found*.

Conformance Rules

None.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

23.4.31 GetValExprColName**Function**

Get the column name of a COLUMN_NAME <value expression>

Definition

```

GetValExprColName (
    ValueExpressionHandle    IN    INTEGER,
    ColumnName               OUT   CHARACTER(L),
    BufferLength              IN    SMALLINT,
    StringLength             OUT   SMALLINT )
RETURNS SMALLINT

```

where *L* has a maximum value of equal to the implementation-defined length of a variable-length character string.

General Rules

- 1) Let *VEH* be the value of ValueExpressionHandle.
- 2) If *VEH* does not identify an allocated value expression description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) If *VEH* does not identify a <value expression> with a type of COLUMN_NAME, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 4) Let *NAME* be the value of the column name of the <value expression>.
- 5) Let *BL* be the value of BufferLength.
- 6) The General Rules of Subclause 22.7, “Character string retrieval”, are applied to Column-Name, *NAME*, *BL*, and StringLength as *TARGET*, *VALUE*, *TARGET OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.

Conformance Rules

None.

23.4.32 GetWrapperLibraryName

Function

Get the library name associated with a foreign-data wrapper.

Definition

```
GetWrapperLibraryName (
    WrapperHandle      IN      INTEGER,
    WrapperLibraryName OUT     CHARACTER(L),
    BufferLength        IN      SMALLINT,
    StringLength       OUT     SMALLINT )
RETURNS SMALLINT
```

where *L* has a maximum value equal to the implementation-defined maximum length of a variable-length character string.

General Rules

- 1) Let *WH* be the value of *WrapperHandle*.
- 2) If *WH* does not identify an allocated foreign-data wrapper description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Let *WL* be the name of the library included in the foreign-data wrapper descriptor of the foreign-data wrapper associated with *WH*.
- 4) The General Rules of Subclause 22.7, “Character string retrieval”, are applied to *WrapperLibraryName*, *WL*, *BufferLength*, and *StringLength* as *TARGET*, *VALUE*, *TARGET OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.

Conformance Rules

None.

23.4 Foreign-data wrapper interface SQL-server routines

23.4.33 GetWrapperName

Function

Get the name of a foreign-data wrapper.

Definition

```
GetWrapperName (
    WrapperHandle      IN      INTEGER,
    WrapperName        OUT     CHARACTER(L),
    BufferLength        IN      SMALLINT,
    StringLength        OUT     SMALLINT )
RETURNS SMALLINT
```

where L has a maximum value equal to $(2n+1)$, where n is the implementation-defined length of an <identifier>.

NOTE 70 – The length $(2n+1)$ supports the syntax of <server name>, which is “<identifier><period><identifier>”.

General Rules

- 1) Let WH be the value of WrapperHandle.
- 2) If WH does not identify an allocated foreign-data wrapper description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Let WN be the foreign-data wrapper name included in the foreign-data wrapper descriptor of the foreign-data wrapper associated with WH .
- 4) The General Rules of Subclause 22.7, “Character string retrieval”, are applied to WrapperName, WN , BufferLength, and StringLength as *TARGET*, *VALUE*, *TARGET OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.

Conformance Rules

None.

23.4.34 GetWrapperOpt**Function**

Get the name and value of a generic option associated with a foreign-data wrapper, given an option number.

Definition

```
GetWrapperOpt (
    WrapperHandle      IN      INTEGER,
    OptionNumber       IN      INTEGER,
    OptionName         OUT     CHARACTER(L1),
    BufferLength1       IN      SMALLINT,
    StringLength1      OUT     SMALLINT,
    OptionValue        OUT     CHARACTER(L2),
    BufferLength2       IN      SMALLINT,
    StringLength2      OUT     SMALLINT )
RETURNS SMALLINT
```

where each of *L1* and *L2* has a maximum value equal to the implementation-defined length of a variable-length character string.

General Rules

- 1) Let *WH* be the value of *WrapperHandle*.
- 2) If *WH* does not identify an allocated foreign-data wrapper description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Let *ON* be the value of *OptionNumber*.
- 4) Let *N* be the number of generic options associated with *WH*.
- 5) If *ON* is less than 1 (one), then an exception condition is raised: *FDW-specific condition — invalid option index*.
- 6) If *ON* is greater than *N*, then a completion condition is raised: *no data* and no further rules of this Subclause are applied.
- 7) Information from the *ON*-th generic option associated with *WH* is retrieved.
 - a) Let *NAME* be the name of the generic option.
 - b) The General Rules of Subclause 22.7, “Character string retrieval”, are applied to *Option-Name*, *NAME*, *BufferLength1*, and *StringLength1* as *TARGET*, *VALUE*, *TARGET OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.
 - c) Let *OPTIONVALUE* be the value of the generic option.
 - d) The General Rules of Subclause 22.7, “Character string retrieval”, are applied to *Option-Value*, *OPTIONVALUE*, *BufferLength2*, and *StringLength2* as *TARGET*, *VALUE*, *TARGET OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.

Conformance Rules

None.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

23.4.35 GetWrapperOptByName

Function

Get the value of a generic option associated with a foreign-data wrapper, given the option name.

Definition

```
GetWrapperOptByName (
    WrapperHandle      IN      INTEGER,
    OptionName         IN      CHARACTER(L1),
    BufferLength1       IN      SMALLINT,
    OptionValue        OUT     CHARACTER(L2),
    BufferLength2       IN      SMALLINT,
    StringLength2      OUT     SMALLINT )
RETURNS SMALLINT
```

where each of *L1* and *L2* has a maximum value equal to the implementation-defined length of a variable-length character string.

General Rules

- 1) Let *WH* be the value of *WrapperHandle*.
- 2) If *WH* does not identify an allocated foreign-data wrapper description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Let *NL* be the value of *BufferLength1*.
- 4) Case:
 - a) If *NL* is not negative, then let *L* be *NL*.
 - b) Otherwise, an exception condition is raised: *FDW-specific condition — invalid string length or buffer length*.
- 5) Case:
 - a) If *L* is zero, then an exception condition is raised: *FDW-specific condition — invalid string length or buffer length*.
 - b) Otherwise:
 - i) Let *N* be the number of whole characters in the first *L* octets of *OptionName* and let *NO* be the number of octets occupied by those *N* characters. If *NO* ≠ *L*, then an exception condition is raised: *FDW-specific condition — invalid option name*.
 - ii) Otherwise, let *ON* be the first *L* octets of *OptionName* and let *TON* be the value of

TRIM (BOTH ' ' FROM *ON*)

- 6) Let *N* be the number of generic options associated with *WH*.

23.4 Foreign-data wrapper interface SQL-server routines

7) Case:

- a) If *TON* is equivalent to the name of a generic option associated with *WH*, then:
 - i) Let *OPTIONVALUE* be the value of the generic option associated with *WH* whose name is equivalent to *TON*.
 - ii) The General Rules of Subclause 22.7, “Character string retrieval”, are applied to *OptionValue*, *OPTIONVALUE*, *BufferLength2*, and *StringLength2* as *TARGET*, *VALUE*, *TARGET OCTET LENGTH*, and *RETURNED OCTET LENGTH*, respectively.
- b) Otherwise, an exception condition is raised: *FDW-specific condition — option name not found*.

Conformance Rules

None.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

23.4.36 SetDescriptor**Function**

Set a field in a foreign-data wrapper descriptor area.

Definition

```
SetDescriptor (
    DescriptorHandle    IN    INTEGER,
    RecordNumber       IN    SMALLINT,
    FieldIdentifier     IN    SMALLINT,
    Value              IN    ANY,
    BufferLength        IN    INTEGER )
RETURNS SMALLINT
```

General Rules

- 1) Let *D* be the allocated foreign-data wrapper descriptor area identified by *DescriptorHandle* and let *N* be the value of the *COUNT* field of *D*.
- 2) Let *HL1* be the standard host language in which the SQL-server is written and let *HL2* be the standard host language in which the foreign-data wrapper is written.
- 3) Let *FI* be the value of *FieldIdentifier*.
- 4) If *FI* is not one of the code values in Table 32, "Codes used for foreign-data wrapper descriptor fields", then an exception condition is raised: *FDW-specific condition — invalid descriptor field identifier*.
- 5) Let *DT* be the type of the descriptor *D*.
- 6) Let *MBS* be the value of the *May Be Set* column in the row of Table 35, "Ability to set foreign-data wrapper descriptor fields", that contains *FI* in the column that contains the descriptor type *DT*.
- 7) If *MBS* is 'No', then an exception condition is raised: *FDW-specific condition — invalid descriptor field identifier*.
- 8) Let *RN* be the value of *RecordNumber*.
- 9) Let *TYPE* be the value of the *Type* column in the row of Table 32, "Codes used for foreign-data wrapper descriptor fields", that contains *FI*.
- 10) If *TYPE* is 'ITEM' and *RN* is less than 1 (one), then an exception condition is raised: *dynamic SQL error — invalid descriptor index*.
- 11) Let *IDA* be the item descriptor area of *D* specified by *RN*.
- 12) If an exception condition is raised in any of the following General Rules, then all fields of *IDA* for which specific values were provided in the invocation of *SetDescField* are set to implementation-dependent values and the value of *COUNT* for *D* is unchanged.
- 13) Information is set in *D*:

23.4 Foreign-data wrapper interface SQL-server routines

Case:

- a) If *FI* indicates COUNT, then

Case:

- i) If the memory requirements to manage the foreign-data wrapper descriptor area cannot be satisfied, then an exception condition is raised: *FDW-specific condition — memory allocation error*.
 - ii) Otherwise, the count of the number of foreign-data wrapper item descriptor areas is set to the value of Value.
- b) If *FI* indicates OCTET_LENGTH, then the value of the OCTET_LENGTH field of *IDA* is set to the value of Value.
- c) If *FI* indicates DATA_POINTER, then the value of the DATA_POINTER field of *IDA* is set to the address of Value. If Value is a null pointer, then the address is set to 0 (zero).
- d) If *FI* indicates DATA, then the value of the DATA field of *IDA* is set to the value of Value.
- e) If *FI* indicates INDICATOR, then the value of the INDICATOR field of *IDA* is set to the value of Value.
- f) If *FI* indicates RETURNED_CARDINALITY, then the value of the RETURNED_CARDINALITY field of *IDA* is set to the value of Value.
- g) If *FI* indicates CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, or CHARACTER_SET_NAME, then:
- i) Let *BL* be the value of BufferLength.
 - ii) Case:
 - 1) If *BL* is not negative, then let *L* be *BL*.
 - 2) Otherwise, an exception condition is raised: *FDW-specific condition — invalid string length or buffer length*.
 - iii) Case:
 - 1) If *L* is zero, then an exception condition is raised: *FDW-specific condition — invalid string length or buffer length*.
 - 2) Otherwise, let *FV* be the first *L* octets of Value and let *TFV* be the value of

TRIM (BOTH ' ' FROM *FV*)

- iv) Let *ML* be the maximum length in characters allowed for an <identifier> as specified in the Syntax Rules of Subclause 5.4, "Names and identifiers", in ISO/IEC 9075-2, and let *TFVL* be the length in characters of *TFV*.
- v) Case:
 - 1) If *TFVL* is greater than *ML*, then *FV* is set to the first *ML* characters of *TFV* and a completion condition is raised: *warning — string data, right truncation*.

23.4 Foreign-data wrapper interface SQL-server routines

2) Otherwise, *FV* is set to *TFV*.

vi) Case:

- 1) If *FI* indicates CHARACTER_SET_CATALOG and *FV* does not conform to the Format and Syntax Rules of an <identifier>, then an exception condition is raised: *invalid catalog name*.
- 2) If *FI* indicates CHARACTER_SET_SCHEMA and *FV* does not conform to the Format and Syntax Rules of an <identifier>, then an exception condition is raised: *invalid schema name*.
- 3) If *FI* indicates CHARACTER_SET_NAME and *FV* does not conform to the Format and Syntax Rules of an <identifier>, then an exception condition is raised: *invalid character set name*.

vii) The value of the field of *IDA* identified by *FI* is set to the value of *FV*.

h) Otherwise, the value of the field of *IDA* identified by *FI* is set to the value of *Value*.

14) If *FI* indicates LEVEL, then:

- a) If *RI* is 1 (one) and *Value* is not 0 (zero), then an exception condition is raised: *dynamic SQL error — invalid LEVEL value*.
- b) If *RI* is greater than 1 (one), then let *PIDA* be *IDA*'s immediately preceding foreign-data wrapper item descriptor area and let *K* be its LEVEL value.
 - i) If *Value* is *K*+1 and TYPE in *PIDA* does not indicate ROW, ARRAY, or ARRAY LOCATOR, then an exception condition is raised: *dynamic SQL error — invalid LEVEL value*.
 - ii) If *Value* is greater than *K*+1, then an exception condition is raised: *dynamic SQL error — invalid LEVEL value*.
 - iii) If *Value* is less than *K*+1, then let *OIDA_i* be the *i*-th foreign-data wrapper item descriptor area to which *PIDA* is subordinate and whose TYPE field indicates ROW. Let *NS_i* be the number of immediately subordinate descriptor areas of *OIDA_i* between *OIDA_i* and *IDA*, and let *D_i* be the value of DEGREE of *OIDA_i*.
 - 1) For each *OIDA_i* whose LEVEL value is greater than *V*, if *D_i* is not equal to *NS_i*, then an exception condition is raised: *dynamic SQL error — invalid LEVEL value*.
 - 2) If *K* is not 0 (zero), then let *OIDA_j* be the *OIDA_j* whose LEVEL value is *K*. If there exists no such *OIDA_j* or *D_j* is not greater than *NS_j*, then an exception condition is raised: *dynamic SQL error — invalid LEVEL value*.

c) The value of LEVEL in *IDA* is set to *Value*.

15) If TYPE is 'ITEM' and *RN* is greater than *N*, then the COUNT field of *D* is set to *RN*.

16) Case:

- a) If *HL1* and *HL2* are both pointer-supporting languages, and if *FI* indicates TYPE, LENGTH, OCTET_LENGTH, PRECISION, SCALE, DATETIME_INTERVAL_CODE, DATETIME_INTERVAL_PRECISION, PARAMETER_MODE, PARAMETER_ORDINAL_POSITION,

23.4 Foreign-data wrapper interface SQL-server routines

PARAMETER_SPECIFIC_CATALOG, PARAMETER_SPECIFIC_SCHEMA, PARAMETER_SPECIFIC_NAME, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME, SCOPE_CATALOG, SCOPE_SCHEMA, or SCOPE_NAME, then the DATA_POINTER field of *IDA* is set to 0 (zero).

- b) Otherwise, if *FI* indicates TYPE, LENGTH, OCTET_LENGTH, PRECISION, SCALE, DATETIME_INTERVAL_CODE, DATETIME_INTERVAL_PRECISION, PARAMETER_MODE, PARAMETER_ORDINAL_POSITION, PARAMETER_SPECIFIC_CATALOG, PARAMETER_SPECIFIC_SCHEMA, PARAMETER_SPECIFIC_NAME, CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME, USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA, USER_DEFINED_TYPE_NAME, SCOPE_CATALOG, SCOPE_SCHEMA, or SCOPE_NAME, then the value of the DATA field of *IDA* is set to 0 (zero).
- 17) If *FI* indicates DATA or if *FI* indicates DATA_POINTER, and Value is not a null pointer, and *IDA* is not consistent as specified in Subclause 22.1, "Description of foreign-data wrapper item descriptor areas", then an exception condition is raised: *FDW-specific condition — inconsistent descriptor information*.
- 18) Let *V* be the value of Value.
- 19) If *FI* indicates TYPE, then:
- a) All the other fields of *IDA* are set to implementation-dependent values.
 - b) Case:
 - i) If *V* indicates CHARACTER, CHARACTER VARYING or CHARACTER LARGE OBJECT then the CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, and CHARACTER_SET_NAME fields of *IDA* are set to the values for the default character set name for the SQL-session and the LENGTH field of *IDA* is set to the maximum possible length in characters of the indicated data type.
 - ii) If *V* indicates BIT or BIT VARYING, then the LENGTH field of *IDA* is set to the maximum possible length in bits of the indicated data type.
 - iii) If *V* indicates BINARY LARGE OBJECT, then the LENGTH field of *IDA* is set to the maximum possible length in octets of the indicated data type.
 - iv) If *V* indicates a <datetime type>, then the PRECISION field of *IDA* is set to 0 (zero).
 - v) If *V* indicates INTERVAL, then the DATETIME_INTERVAL_PRECISION field of *IDA* is set to 2.
 - vi) If *V* indicates NUMERIC or DECIMAL, then the SCALE field of *IDA* is set to 0 (zero) and the PRECISION field of *IDA* is set to the implementation-defined default value for the precision of the NUMERIC or DECIMAL data types, respectively.
 - vii) If *V* indicates SMALLINT or INTEGER, then the SCALE field of *IDA* is set to 0 (zero) and the PRECISION field of *IDA* is set to the implementation-defined value for the precision of the SMALLINT or INTEGER data types, respectively.
 - viii) If *V* indicates FLOAT, then the PRECISION field of *IDA* is set to the implementation-defined default value for the precision of the FLOAT data type.

ISO/IEC 9075-9:2001 (E)

23.4 Foreign-data wrapper interface SQL-server routines

- ix) If *V* indicates REAL or DOUBLE PRECISION, then the PRECISION field of *IDA* is set to the implementation-defined value for the precision of the REAL or DOUBLE PRECISION data types, respectively.
 - x) If *V* indicates an implementation-defined data type, then an implementation-defined set of fields of *IDA* are set to implementation-defined default values.
 - xi) Otherwise, an exception condition is raised: *FDW-specific condition — invalid data type*.
- 20) If *FI* indicates DATETIME_INTERVAL_CODE and the TYPE field of *IDA* indicates a <datetime type>, then:
- a) All the fields of *IDA* other than DATETIME_INTERVAL_CODE and TYPE are set to implementation-dependent values.
 - b) Case:
 - i) If *V* indicates DATE, TIME, or TIME WITH TIME ZONE, then the PRECISION field of *IDA* is set to 0 (zero).
 - ii) If *V* indicates TIMESTAMP or TIMESTAMP WITH TIME ZONE, then the PRECISION field of *IDA* is set to 6.
- 21) If *FI* indicates DATETIME_INTERVAL_CODE and the TYPE field of *IDA* indicates INTERVAL, then the DATETIME_INTERVAL_PRECISION field of *IDA* is set to 2 and:
- a) If *V* indicates DAY TO SECOND, HOUR TO SECOND, MINUTE TO SECOND, or SECOND, then the PRECISION field of *IDA* is set to 6.
 - b) Otherwise, the PRECISION field of *IDA* is set to 0 (zero).

Conformance Rules

None.

23.5 Foreign-data wrapper interface general routines

23.5.1 GetDiagnostics

Function

Get information from a foreign-data wrapper diagnostics area.

Definition

```
GetDiagnostics (
    HandleType      IN      SMALLINT,
    Handle          IN      INTEGER,
    RecordNumber    IN      SMALLINT,
    DiagIdentifier   IN      SMALLINT,
    DiagInfo        OUT     ANY,
    BufferLength     IN      SMALLINT,
    StringLength    OUT     SMALLINT )
RETURNS SMALLINT
```

General Rules

- 1) Let *HT* be the value of *HandleType*.
- 2) If *HT* is not one of the code values in Table 33, “Codes used for foreign-data wrapper handle types”, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 3) Case:
 - a) If *HT* indicates EXECUTION HANDLE and *Handle* does not identify an allocated execution description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
 - b) If *HT* indicates FSCONNECTION HANDLE and *Handle* does not identify an allocated FS-connection, then an exception condition is raised: *FDW-specific condition — invalid handle*.
 - c) If *HT* indicates REPLY HANDLE and *Handle* does not identify an allocated reply description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
 - d) If *HT* indicates REQUEST HANDLE and *Handle* does not identify an allocated request description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
 - e) If *HT* indicates DATA HANDLE and *Handle* does not identify an allocated data row description, then an exception condition is raised: *FDW-specific condition—invalid handle*.
 - f) If *HT* indicates SERVER HANDLE and *Handle* does not identify an allocated foreign-server description, then an exception condition is raised: *FDW-specific condition — invalid handle*.

23.5 Foreign-data wrapper interface general routines

- g) If *HT* indicates **TABLEREFERENCE HANDLE** and *Handle* does not identify an allocated table reference description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
 - h) If *HT* indicates **USER HANDLE** and *Handle* does not identify an allocated user mapping description, then an exception condition is raised: *FDW-specific condition—invalid handle*.
 - i) If *HT* indicates **VALUEEXPRESSION HANDLE** and *Handle* does not identify an allocated value expression description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
 - j) If *HT* indicates **WRAPPER HANDLE** and *Handle* does not identify an allocated foreign-data wrapper description, then an exception condition is raised: *FDW-specific condition — invalid handle*.
 - k) If *HT* indicates **WRAPPERENV HANDLE** and *Handle* does not identify an allocated FDW-environment, then an exception condition is raised: *FDW-specific condition — invalid handle*.
- 4) Let *DI* be the value of *DiagIdentifier*.
 - 5) If *DI* is not one of the code values in Table 31, “Codes used for foreign-data wrapper diagnostic fields”, then an exception condition is raised: *FDW-specific condition — invalid attribute value*.
 - 6) Let *TYPE* be the value of the *Type* column in the row that contains *DI* in Table 31, “Codes used for foreign-data wrapper diagnostic fields”
 - 7) Let *RN* be the value of *RecordNumber*.
 - 8) Let *R* be the most recently executed foreign-data wrapper interface routine, other than *GetDiagnostics()*, for which *Handle* was passed as the value of an input handle and let *N* be the number of status records generated by the execution of *R*.
NOTE 71 – The *GetDiagnostics()* routine may cause exception or completion conditions to be raised, but it does not cause diagnostic information to be generated.
 - 9) If *TYPE* is ‘STATUS’, then:
 - a) If *RN* is less than 1 (one), then an exception condition is raised: *invalid condition number*.
 - b) If *RN* is greater than *N*, then a completion condition is raised: *no data*, and no further rules of this Subclause are applied.
 - 10) If *TYPE* is ‘HEADER’, then header information from the diagnostics area associated with the resource identified by *Handle* is retrieved.
 - a) If *DI* indicates **NUMBER**, then the value retrieved is *N*.
 - b) If *DI* indicates **RETURNCODE**, then the value retrieved is the code indicating the basic result of the execution of *R*. Subclause 4.18.4, “Return codes”, specifies the code values and their meanings.
NOTE 72 – The value retrieved will never indicate **Invalid handle** or **Data needed**, since no diagnostic information is generated if this is the basic result of the execution of *R*.
 - c) If *DI* indicates **MORE**, then the value retrieved is

ISO/IEC 9075-9:2001 (E)

23.5 Foreign-data wrapper interface general routines

Case:

- i) If more conditions were raised during execution of *R* than have been stored in the diagnostics area, then 1 (one).
 - ii) If all the conditions that were raised during execution of *R* have been stored in the diagnostics area, then 0 (zero).
 - d) If *DI* indicates an implementation-defined diagnostics header field, then the value retrieved is the value of the implementation-defined diagnostics header field.
- 11) If *TYPE* is 'STATUS', then information from the *RN*-th status record in the foreign-data wrapper diagnostics area associated with the resource identified by *Handle* is retrieved.
- a) If *DI* indicates *SQLSTATE*, then the value retrieved is the *SQLSTATE* value corresponding to the status condition.
 - b) If *DI* indicates *NATIVE_CODE*, then the value retrieved is the implementation-defined native error code corresponding to the status condition.
 - c) If *DI* indicates *MESSAGE_TEXT*, then the value retrieved is an implementation-defined character string.
NOTE 73 – An implementation may provide <space>s or a zero-length string or a character string that describes the status condition.
 - d) If *DI* indicates *MESSAGE_LENGTH*, then the value retrieved is the length in characters of the character string value of *MESSAGE_TEXT* corresponding to the status condition.
 - e) If *DI* indicates *MESSAGE_OCTET_LENGTH*, then the value retrieved is the length in octets of the character string value of *MESSAGE_TEXT* corresponding to the status condition.
 - f) If *DI* indicates *CLASS_ORIGIN*, then the value retrieved is the identification of the naming authority that defined the class value of the *SQLSTATE* value corresponding to the status condition. That value shall be 'ISO 9075' if the class value is fully defined in Subclause 27.1, "SQLSTATE", and shall be an implementation-defined character string other than 'ISO 9075' for any implementation-defined class value.
 - g) If *DI* indicates *SUBCLASS_ORIGIN*, then the value retrieved is the identification of the naming authority that defined the subclass value of the *SQLSTATE* value corresponding to the status condition. That value shall be 'ISO 9075' if the class value is fully defined in Subclause 27.1, "SQLSTATE", and shall be an implementation-defined character string other than 'ISO 9075' for any implementation-defined subclass value.
 - h) If *DI* indicates an implementation-defined diagnostics status field, then the value retrieved is the value of the implementation-defined diagnostics status field.
- 12) Let *V* be the value retrieved.
- 13) If *DI* indicates a diagnostics field whose row in Table 4, "Fields used in foreign-data wrapper diagnostics areas", contains a Data Type that is neither CHARACTER nor CHARACTER VARYING, then *DiagInfo* is set to *V* and no further rules of this Subclause are applied.
- 14) Let *BL* be the value of *BufferLength*.

ISO/IEC 9075-9:2001 (E)

23.5 Foreign-data wrapper interface general routines

- 15) If BL is not greater than zero, then an exception condition is raised: *FDW-specific condition — invalid string length or buffer length*.
- 16) Let L be the length in octets of V .
- 17) If `StringLength` is not a null pointer, then `StringLength` is set to L .
- 18) Case:
 - a) If L is not greater than BL , then the first L octets of `DiagInfo` are set to V and the values of the remaining octets of `DiagInfo` are implementation-dependent.
 - b) Otherwise, `DiagInfo` is set to the first BL octets of V .

Conformance Rules

None.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

24 Diagnostics management

24.1 <get diagnostics statement>

Function

Get exception or completion condition information from the diagnostics area.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

Table 38—SQL-statement codes

SQL-statement	Identifier	Code
All alternatives from ISO/IEC 9075-2		
All alternatives from ISO/IEC 9075-5		
<alter foreign-data wrapper statement>	ALTER FOREIGN DATA WRAPPER	120
<alter foreign server statement>	ALTER SERVER	108
<alter foreign table statement>	ALTER FOREIGN TABLE	104
<alter user mapping statement>	ALTER USER MAPPING	123
<drop foreign-data wrapper statement>	DROP FOREIGN DATA WRAPPER	121
<drop foreign server statement>	DROP SERVER	110
<drop foreign table statement>	DROP FOREIGN TABLE	105
<drop user mapping statement>	DROP USER MAPPING	124
<foreign-data wrapper definition>	CREATE FOREIGN DATA WRAPPER	119
<foreign server definition>	CREATE SERVER	107
<foreign table definition>	CREATE FOREIGN TABLE	103
<import foreign schema statement>	IMPORT FOREIGN SCHEMA	125

ISO/IEC 9075-9:2001 (E)
24.1 <get diagnostics statement>

Table 38—SQL-statement codes (Cont.)

SQL-statement	Identifier	Code
<set passthrough statement>	SET PASSTHROUGH	126
<user mapping definition>	CREATE USER MAPPING	122
Statements that are defined by the foreign-data wrapper	A character string value defined by the foreign-data wrapper different from the value associated with any other SQL-statement	x ¹
¹ An implementation-defined negative number different from the value associated with any other SQL-statement.		

Conformance Rules

No additional Conformance Rules.

25 Information Schema

25.1 ATTRIBUTES view

Function

Identify the attributes of user-defined types defined in this catalog that are accessible to a given user.

Definition

```
CREATE VIEW ATTRIBUTES AS
SELECT DISTINCT
    UDT_CATALOG, UDT_SCHEMA, UDT_NAME, A.ATTRIBUTE_NAME, ORDINAL_POSITION,
    CASE
        WHEN EXISTS
            ( SELECT *
              FROM DEFINITION_SCHEMA.SCHEMATA AS S
              WHERE ( UDT_CATALOG, UDT_SCHEMA )
                    = ( S.CATALOG_NAME, S.SCHEMA_NAME )
                AND
                  ( SCHEMA_OWNER IN
                    ( 'PUBLIC', CURRENT_USER )
                  OR
                    SCHEMA_OWNER IN
                    ( SELECT ROLE_NAME
                      FROM ENABLED_ROLES ) ) )
            THEN ATTRIBUTE_DEFAULT
          ELSE NULL
        END AS ATTRIBUTE_DEFAULT,
    IS_NULLABLE, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH,
    C1.CHARACTER_SET_CATALOG, C1.CHARACTER_SET_SCHEMA, C1.CHARACTER_SET_NAME,
    D1.COLLATION_CATALOG, D1.COLLATION_SCHEMA, D1.COLLATION_NAME,
    NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE,
    DATETIME_PRECISION, INTERVAL_TYPE, INTERVAL_PRECISION,
    D1.USER_DEFINED_TYPE_CATALOG AS ATTRIBUTE_UDT_CATALOG,
    D1.USER_DEFINED_TYPE_SCHEMA AS ATTRIBUTE_UDT_SCHEMA,
    D1.USER_DEFINED_TYPE_NAME AS ATTRIBUTE_UDT_NAME,
    D1.SCOPE_CATALOG, D1.SCOPE_SCHEMA, D1.SCOPE_NAME,
    MAXIMUM_CARDINALITY, A.DTD_IDENTIFIER, CHECK_REFERENCES,
    IS_DERIVED_REFERENCE_ATTRIBUTE,
    D1.DATALINK_LINK_CONTROL, D1.DATALINK_INTEGRITY, D1.DATALINK_READ_PERMISSION,
    D1.DATALINK_WRITE_PERMISSION, D1.DATALINK_RECOVERY, D1.DATALINK_UNLINK
FROM ( DEFINITION_SCHEMA.ATTRIBUTES AS A
      LEFT JOIN
        ( DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D1
          LEFT JOIN
            DEFINITION_SCHEMA.COLLATIONS AS C1
            ON ( ( C1.COLLATION_CATALOG, C1.COLLATION_SCHEMA, C1.COLLATION_NAME )
                = ( D1.COLLATION_CATALOG, D1.COLLATION_SCHEMA, D1.COLLATION_NAME ) ) )
        ON ( ( A.UDT_CATALOG, A.UDT_SCHEMA, A.UDT_NAME,
              'USER-DEFINED TYPE', A.DTD_IDENTIFIER )
            = ( D1.OBJECT_CATALOG, D1.OBJECT_SCHEMA, D1.OBJECT_NAME,
              D1.OBJECT_TYPE, D1.DTD_IDENTIFIER ) ) )
WHERE ( A.UDT_CATALOG, A.UDT_SCHEMA, A.UDT_NAME ) IN
      ( SELECT UDT_CATALOG, UDT_SCHEMA, UDT_NAME
        FROM DEFINITION_SCHEMA.USER_DEFINED_TYPE_PRIVILEGES
```

ISO/IEC 9075-9:2001 (E)
25.1 ATTRIBUTES view

```
        WHERE ( SCHEMA_OWNER IN
                ( 'PUBLIC', CURRENT_USER )
              OR
                SCHEMA_OWNER IN
                ( SELECT ROLE_NAME
                  FROM ENABLED_ROLES ) ) )
    AND
    A.UDT_CATALOG
    = ( SELECT CATALOG_NAME
        FROM INFORMATION_SCHEMA_CATALOG_NAME );

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

Conformance Rules

No additional Conformance Rules.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001

25.2 COLUMN_OPTIONS view

Function

Identify the generic options specified for columns that are defined in this catalog.

Definition

```
CREATE VIEW COLUMN_OPTIONS AS
  SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME,
         COLUMN_NAME, OPTION_NAME, OPTION_VALUE
  FROM DEFINITION_SCHEMA.COLUMN_OPTIONS CO
 WHERE ( CO.TABLE_CATALOG, CO.TABLE_SCHEMA, CO.TABLE_NAME, CO.COLUMN_NAME )
        IN ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME
              FROM DEFINITION_SCHEMA.COLUMN_PRIVILEGES
              WHERE ( SCHEMA_OWNER IN ( 'PUBLIC', CURRENT_USER )
                     OR
                     SCHEMA_OWNER IN ( SELECT ROLE_NAME
                                         FROM ENABLED_ROLES ) ) ) )
        AND
        CO.TABLE_CATALOG
        = ( SELECT CATALOG_NAME
            FROM INFORMATION_SCHEMA.CATALOG_NAME ) ;
```

Conformance Rules

- 1) Without Feature M004, "Foreign data support", conforming SQL language shall not reference the COLUMN_OPTIONS view.

25.3 COLUMNS view

Function

Identify the columns of tables defined in this catalog that are accessible to a given user.

Definition

```
CREATE VIEW COLUMNS AS
  SELECT DISTINCT
    TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, C.COLUMN_NAME, ORDINAL_POSITION,
    CASE
      WHEN EXISTS
        ( SELECT *
          FROM DEFINITION_SCHEMA.SCHEMATA AS S
          WHERE ( TABLE_CATALOG, TABLE_SCHEMA )
              = ( S.CATALOG_NAME, S.SCHEMA_NAME )
            AND
              SCHEMA_OWNER IN
                ( 'PUBLIC', CURRENT_USER )
            OR
              SCHEMA_OWNER IN
                ( SELECT ROLE_NAME
                  FROM ENABLED_ROLES ) )
        THEN COLUMN_DEFAULT
      ELSE NULL
    END AS COLUMN_DEFAULT,
    IS_NULLABLE,
    COALESCE (D1.DATA_TYPE, D2.DATA_TYPE) AS DATA_TYPE,
    COALESCE (D1.CHARACTER_MAXIMUM_LENGTH, D2.CHARACTER_MAXIMUM_LENGTH)
      AS CHARACTER_MAXIMUM_LENGTH,
    COALESCE (D1.CHARACTER_OCTET_LENGTH, D2.CHARACTER_OCTET_LENGTH)
      AS CHARACTER_OCTET_LENGTH,
    COALESCE (D1.NUMERIC_PRECISION, D2.NUMERIC_PRECISION)
      AS NUMERIC_PRECISION,
    COALESCE (D1.NUMERIC_PRECISION_RADIX, D2.NUMERIC_PRECISION_RADIX)
      AS NUMERIC_PRECISION_RADIX,
    COALESCE (D1.NUMERIC_SCALE, D2.NUMERIC_SCALE)
      AS NUMERIC_SCALE,
    COALESCE (D1.DATETIME_PRECISION, D2.DATETIME_PRECISION)
      AS DATETIME_PRECISION,
    COALESCE (D1.INTERVAL_TYPE, D2.INTERVAL_TYPE)
      AS INTERVAL_TYPE,
    COALESCE (D1.INTERVAL_PRECISION, D2.INTERVAL_PRECISION)
      AS INTERVAL_PRECISION,
    COALESCE (C1.CHARACTER_SET_CATALOG, C2.CHARACTER_SET_CATALOG)
      AS CHARACTER_SET_CATALOG,
    COALESCE (C1.CHARACTER_SET_SCHEMA, C2.CHARACTER_SET_SCHEMA)
      AS CHARACTER_SET_SCHEMA,
    COALESCE (C1.CHARACTER_SET_NAME, C2.CHARACTER_SET_NAME)
      AS CHARACTER_SET_NAME,
    COALESCE (D1.COLLATION_CATALOG, D2.COLLATION_CATALOG)
      AS COLLATION_CATALOG,
    COALESCE (D1.COLLATION_SCHEMA, D2.COLLATION_SCHEMA)
      AS COLLATION_SCHEMA,
    COALESCE (D1.COLLATION_NAME, D2.COLLATION_NAME)
      AS COLLATION_NAME,
    DOMAIN_CATALOG, DOMAIN_SCHEMA, DOMAIN_NAME,
    COALESCE (D1.USER_DEFINED_TYPE_CATALOG, D2.USER_DEFINED_TYPE_CATALOG)
      AS USER_DEFINED_TYPE_CATALOG,
    COALESCE (D1.USER_DEFINED_TYPE_SCHEMA, D2.USER_DEFINED_TYPE_SCHEMA)
      AS USER_DEFINED_TYPE_SCHEMA,
    COALESCE (D1.USER_DEFINED_TYPE_NAME, D2.USER_DEFINED_TYPE_NAME)
```

```

        AS USER_DEFINED_TYPE_NAME,
    COALESCE (D1.SCOPE_CATALOG, D2.SCOPE_CATALOG)
        AS SCOPE_CATALOG,
    COALESCE (D1.SCOPE_SCHEMA, D2.SCOPE_SCHEMA)
        AS SCOPE_SCHEMA,
    COALESCE (D1.SCOPE_NAME, D2.SCOPE_NAME)
        AS SCOPE_NAME,
    COALESCE (D1.MAXIMUM_CARDINALITY, D2.MAXIMUM_CARDINALITY)
        AS MAXIMUM_CARDINALITY,
    COALESCE (D1.DTD_IDENTIFIER, D2.DTD_IDENTIFIER)
        AS DTD_IDENTIFIER,
    IS_SELF_REFERENCING,

    DATALINK_LINK_CONTROL, DATALINK_INTEGRITY, DATALINK_READ_PERMISSION,
    DATALINK_WRITE_PERMISSION, DATALINK_RECOVERY, DATALINK_UNLINK
FROM ( DEFINITION_SCHEMA.COLUMNS AS C
    LEFT JOIN
        ( DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D1
            LEFT JOIN
                DEFINITION_SCHEMA.COLLATIONS AS C1
            ON
                ( ( C1.COLLATION_CATALOG, C1.COLLATION_SCHEMA, C1.COLLATION_NAME )
                  = ( D1.COLLATION_CATALOG, D1.COLLATION_SCHEMA, D1.COLLATION_NAME ) ) )
        ON
            ( ( C.TABLE_CATALOG, C.TABLE_SCHEMA, C.TABLE_NAME,
              'TABLE', C.DTD_IDENTIFIER )
              = ( D1.OBJECT_CATALOG, D1.OBJECT_SCHEMA, D1.OBJECT_NAME,
                D1.OBJECT_TYPE, D1.DTD_IDENTIFIER ) ) )
    LEFT JOIN
        ( DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D2
            LEFT JOIN
                DEFINITION_SCHEMA.COLLATIONS AS C2
            ON
                ( ( C2.COLLATION_CATALOG, C2.COLLATION_SCHEMA, C2.COLLATION_NAME )
                  = ( D2.COLLATION_CATALOG, D2.COLLATION_SCHEMA, D2.COLLATION_NAME ) ) )
        ON
            ( ( C.DOMAIN_CATALOG, C.DOMAIN_SCHEMA, C.DOMAIN_NAME,
              'DOMAIN', C.DTD_IDENTIFIER )
              = ( D2.OBJECT_CATALOG, D2.OBJECT_SCHEMA, D2.OBJECT_NAME,
                D2.OBJECT_TYPE, D2.DTD_IDENTIFIER ) ) )
WHERE ( C.TABLE_CATALOG, C.TABLE_SCHEMA, C.TABLE_NAME, C.COLUMN_NAME )
    IN
        ( SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME
          FROM DEFINITION_SCHEMA.COLUMN_PRIVILEGES
          WHERE ( SCHEMA_OWNER IN
                  ( 'PUBLIC', CURRENT_USER )
                OR
                  SCHEMA_OWNER IN
                  ( SELECT ROLE_NAME
                    FROM ENABLED_ROLES ) ) ) )
AND
    C.TABLE_CATALOG
    = ( SELECT CATALOG_NAME
      FROM INFORMATION_SCHEMA.CATALOG_NAME ) ;

GRANT SELECT ON TABLE COLUMNS
TO PUBLIC WITH GRANT OPTION ;

```

Conformance Rules

- 1) Without Feature M001, “Datalinks”, conforming SQL language shall not reference the columns DATALINK_CONTROL, DATALINK_INTEGRITY, DATALINK_READ_PERMISSION, DATALINK_WRITE_PERMISSION, DATALINK_RECOVERY, or DATALINK_UNLINK.
- 2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference the COLUMNS view.

IECNORM.COM : Click to view the full PDF of ISO/IEC 9075-9:2001