INTERNATIONAL STANDARD

ISO/IEC 9075-14

Third edition
2008-07-15

Information technology — Database languages — SQL —

Part 14: XML-Related Specifications (SQL/XML)

Technologies de l'information — Langages de base de données — SQL —

Partie 14: Spécifications relatives au XML (SQL/XML)

Reference number
ISO/IEC 9075-14:2008(E)

© ISO/IEC 2008

--`,,```,,,,````-`-,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

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.



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2008

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.org
Web www.iso.org

Published in Switzerland

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

© ISO/IEC 2008 – All rights reserved

Not for Resale

Contents

Page

Foreword. xi
Introduction. xii
1 Scope. 1
2 Normative references. 3
2.1 ISO and IEC standards. 3
2.2 Other international standards. 3
3 Definitions, notations and conventions. 7
3.1 Definitions. 7
3.1.1 Definitions taken from XML. 7
3.1.2 Definitions taken from XML Schema. 7
3.1.3 Definitions provided in Part 14. 7
3.2 Notation. 11
4 Concepts. 13
4.1 Data types. 13
4.1.1 Naming of predefined types. 13
4.1.2 Data type terminology. 13
4.2 XML. 13
4.2.1 Introduction. 13
4.2.2 XML types. 14
4.2.3 Characteristics of XML values. 15
4.2.4 XML comparison and assignment. 15
4.2.5 Operations involving XML values. 16
4.2.6 Registered XML Schemas. 18
4.3 Data conversions. 19
4.4 Data analysis operations (involving tables). 20
4.4.1 Aggregate functions. 20
4.5 SQL-invoked routines. 20
4.5.1 Routine descriptors. 20
4.6 SQL-statements. 21
4.6.1 SQL-statements classified by function. 21
4.6.1.1 SQL-session statements. 21
4.7 Basic security model. 21
4.7.1 Privileges. 21
4.8 SQL-sessions. 22
4.8.1 SQL-session properties. 22

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4.9 XML namespaces. . . . 22
4.10 Overview of mappings. . . . 23
4.10.1 Mapping SQL character sets to Unicode. . . . 23
4.10.2 Mapping Unicode to SQL character sets. . . . 24
4.10.3 Mapping SQL <identifier>s to XML. . . . 24
4.10.4 Mapping XML Names to SQL. . . . 24
4.10.5 Mapping SQL data types to XML. . . . 24
4.10.6 Mapping values of SQL data types to XML. . . . 26
4.10.7 Mapping XQuery atomic values to SQL values. . . . 27
4.10.8 Visibility of columns, tables, and schemas in mappings from SQL to XML. . . . 28
4.10.9 Mapping an SQL table to XML. . . . 28
4.10.10 Mapping an SQL schema to XML. . . . 29
4.10.11 Mapping an SQL catalog to XML. . . . 30

5 Lexical elements. . . . 31
5.1 <token> and <separator>. . . . 31
5.2 Names and identifiers. . . . 33

6 Scalar expressions. . . . 35
6.1 <data type>. . . . 35
6.2 <field definition>. . . . 38
6.3 <value expression primary>. . . . 39
6.4 <case expression>. . . . 40
6.5 <cast specification>. . . . 41
6.6 <XML cast specification>. . . . 44
6.7 <value expression>. . . . 51
6.8 <string value function>. . . . 53
6.9 <XML value expression>. . . . 58
6.10 <XML value function>. . . . 59
6.11 <XML comment>. . . . 60
6.12 <XML concatenation>. . . . 62
6.13 <XML document>. . . . 64
6.14 <XML element>. . . . 66
6.15 <XML forest>. . . . 70
6.16 <XML parse>. . . . 73
6.17 <XML PI>. . . . 75
6.18 <XML query>. . . . 78
6.19 <XML text>. . . . 84
6.20 <XML validate>. . . . 86

7 Query expressions. . . . 91
7.1 <table reference>. . . . 91
7.2 <query expression>. . . . 96

8 Predicates. . . . 99
8.1 <predicate>. . . . 99
8.2 <XML content predicate>. . . . 100

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.3 <XML document predicate>. . . . 102
8.4 <XML exists predicate>. . . . 104
8.5 <XML valid predicate>. . . . 105
9 Mappings. . . . 111
9.1 Mapping SQL <identifier>s to XML Names. . . . 111
9.2 Mapping a multi-part SQL name to an XML Name. . . . 114
9.3 Mapping XML Names to SQL <identifier>s. . . . 115
9.4 Mapping an SQL data type to an XML Name. . . . 117
9.5 Mapping SQL data types to XML Schema data types. . . . 121
9.6 Mapping an SQL data type to a named XML Schema data type. . . . 140
9.7 Mapping a collection of SQL data types to XML Schema data types. . . . 142
9.8 Mapping values of SQL data types to values of XML Schema data types. . . . 144
9.9 Mapping an SQL table to XML Schema data types. . . . 150
9.10 Mapping an SQL table to an XML element or a sequence of XML elements. . . . 154
9.11 Mapping an SQL table to XML and an XML Schema document. . . . 157
9.12 Mapping an SQL schema to XML Schema data types. . . . 162
9.13 Mapping an SQL schema to an XML element. . . . 164
9.14 Mapping an SQL schema to an XML document and an XML Schema document. . . . 166
9.15 Mapping an SQL catalog to XML Schema data types. . . . 171
9.16 Mapping an SQL catalog to an XML element. . . . 173
9.17 Mapping an SQL catalog to an XML document and an XML Schema document. . . . 175
10 Additional common rules. . . . 181
10.1 Retrieval assignment. . . . 181
10.2 Store assignment. . . . 183
10.3 Result of data type combinations. . . . 185
10.4 Type precedence list determination. . . . 188
10.5 Type name determination. . . . 189
10.6 Determination of identical values. . . . 190
10.7 Determination of equivalent XML values. . . . 191
10.8 Equality operations. . . . 194
10.9 Grouping operations. . . . 195
10.10 Multiset element grouping operations. . . . 196
10.11 Ordering operations. . . . 197
10.12 Determination of namespace URI. . . . 198
10.13 Construction of an XML element. . . . 200
10.14 Concatenation of two XML values. . . . 203
10.15 Serialization of an XML value. . . . 204
10.16 Parsing a string as an XML value. . . . 208
10.17 Removing XQuery document nodes from an XQuery sequence. . . . 211
10.18 Constructing a copy of an XML value. . . . 212
10.19 Constructing an unvalidated XQuery document node. . . . 213
10.20 Creation of an XQuery expression context. . . . 214
10.21 Determination of an XQuery formal type notation. . . . 216

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10.22 Validating an XQuery document or element node. . . . 219
11 Additional common elements. . . . 221
11.1 <routine invocation>. . . . 221
11.2 <aggregate function>. . . . 224
11.3 <XML lexically scoped options>. . . . 227
11.4 <XML returning clause>. . . . 229
11.5 <XML passing mechanism>. . . . 230
11.6 <XML valid according to clause>. . . . 231
12 Schema definition and manipulation. . . . 235
12.1 <column definition>. . . . 235
12.2 <check constraint definition>. . . . 237
12.3 <view definition>. . . . 238
12.4 <assertion definition>. . . . 240
12.5 <user-defined type definition>. . . . 241
12.6 <attribute definition>. . . . 242
12.7 <SQL-invoked routine>. . . . 243
12.8 <user-defined cast definition>. . . . 247
13 SQL-client modules. . . . 249
13.1 Calls to an <externally-invoked procedure>. . . . 249
13.2 <SQL procedure statement>. . . . 251
13.3 Data type correspondences. . . . 252
14 Data manipulation. . . . 255
14.1 <fetch statement>. . . . 255
14.2 <select statement: single row>. . . . 257
14.3 <delete statement: searched>. . . . 259
14.4 <insert statement>. . . . 260
14.5 <merge statement>. . . . 261
14.6 <update statement: positioned>. . . . 262
14.7 <update statement: searched>. . . . 263
15 Control statements. . . . 265
15.1 <compound statement>. . . . 265
15.2 <assignment statement>. . . . 267
16 Session management. . . . 269
16.1 <set XML option statement>. . . . 269
17 Dynamic SQL. . . . 271
17.1 Description of SQL descriptor areas. . . . 271
17.2 <input using clause>. . . . 272
17.3 <output using clause>. . . . 273
17.4 <prepare statement>. . . . 275
18 Embedded SQL. . . . 277
18.1 <embedded SQL host program>. . . . 277

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

18.2 <embedded SQL Ada program>. . . . 283
18.3 <embedded SQL C program>. . . . 286
18.4 <embedded SQL COBOL program>. . . . 291
18.5 <embedded SQL Fortran program>. . . . 294
18.6 <embedded SQL MUMPS program>. . . . 297
18.7 <embedded SQL Pascal program>. . . . 300
18.8 <embedded SQL PL/I program>. . . . 303
19 Diagnostics management. . . . 307
19.1 <get diagnostics statement>. . . . 307
20 Information Schema. . . . 309
20.1 NCNAME domain. . . . 309
20.2 URI domain. . . . 310
20.3 ATTRIBUTES view. . . . 311
20.4 COLUMNS view. . . . 312
20.5 DOMAINS view. . . . 313
20.6 ELEMENT_TYPES view. . . . 314
20.7 FIELDS view. . . . 315
20.8 METHOD_SPECIFICATION_PARAMETERS view. . . . 316
20.9 METHOD_SPECIFICATIONS view. . . . 317
20.10 PARAMETERS view. . . . 318
20.11 ROUTINES view. . . . 319
20.12 XML_SCHEMA_ELEMENTS view. . . . 321
20.13 XML_SCHEMA_NAMESPACES view. . . . 322
20.14 XML_SCHEMAS view. . . . 323
20.15 Short name views. . . . 324
21 Definition Schema. . . . 329
21.1 DATA_TYPE_DESCRIPTOR base table. . . . 329
21.2 PARAMETERS base table. . . . 332
21.3 ROUTINES base table. . . . 334
21.4 USAGE_PRIVILEGES base table. . . . 336
21.5 XML_SCHEMA_ELEMENTS base table. . . . 337
21.6 XML_SCHEMA_NAMESPACES base table. . . . 338
21.7 XML_SCHEMAS base table. . . . 339
22 The SQL/XML XML Schema. . . . 341
22.1 The SQL/XML XML Schema. . . . 341
23 Status codes. . . . 345
23.1 SQLSTATE. . . . 345
24 Conformance. . . . 347
24.1 Claims of conformance to SQL/XML. . . . 347
24.2 Additional conformance requirements for SQL/XML. . . . 348
24.3 Implied feature relationships of SQL/XML. . . . 349
Annex A SQL Conformance Summary (informative)**. . . . 359**

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex B **Implementation-defined elements** (informative). 395
Annex C **Implementation-dependent elements** (informative). 405
Annex D **Deprecated features** (informative). 407
Annex E **Incompatibilities with ISO/IEC 9075-14:2006** (informative). 409
Annex F **SQL feature taxonomy** (informative). 411
Annex G **Defect reports not addressed in this edition of this part of ISO/IEC 9075** (informative). . 419
Index. 421

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Tables

Table

Page

1 Permanently registered XML Schemas. . . . 19
2 XML namespace prefixes and their URIs. . . . 22
3 Constraining facets of XML Schema integer types. . . . 127
4 XQuery node properties. . . . 192
5 Data type correspondences for Ada. . . . 252
6 Data type correspondences for C. . . . 252
7 Data type correspondences for COBOL. . . . 252
8 Data type correspondences for Fortran. . . . 253
9 Data type correspondences for M. . . . 253
10 Data type correspondences for Pascal. . . . 253
11 Data type correspondences for PL/I. . . . 253
12 Codes used for SQL data types in Dynamic SQL. . . . 271
13 SQL-statement codes. . . . 307
14 SQLSTATE class and subclass values. . . . 345
15 Implied feature relationships of SQL/XML. . . . 349
16 Feature taxonomy for optional features. . . . 411

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

(Blank page)

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

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 2.

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 International Standard may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 9075-14 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange*.

This third edition of ISO/IEC 9075-14 cancels and replaces the second edition (ISO/IEC 9075-14:2006), which has been technically revised. It also incorporates Technical Corrigendum ISO/IEC 9075-14:2006/Cor.1:2007.

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 9: Management of External Data (SQL/MED)

— Part 10: Object Language Bindings (SQL/OLB)

— Part 11: Information and Definition Schemas (SQL/Schemata)

— Part 13: SQL Routines and Types Using the Java™ Programming Language (SQL/JRT)

— Part 14: XML-Related Specifications (SQL/XML)

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Introduction

The organization of this part of ISO/IEC 9075-14 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.

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, "Mappings", defines the ways in which certain SQL information can be mapped into XML and certain XML information can be mapped into SQL.

10) Clause 10, "Additional common rules", specifies the rules for assignments that retrieve data from or store data into SQL-data, and formation rules for set operations.

11) Clause 11, "Additional common elements", defines additional language elements that are used in various parts of the language.

12) Clause 12, "Schema definition and manipulation", defines facilities for creating and managing a schema.

13) Clause 13, "SQL-client modules", defines SQL-client modules and externally-invoked procedures.

14) Clause 14, "Data manipulation", defines the data manipulation statements.

15) Clause 15, "Control statements", defines the SQL-control statements.

16) Clause 16, "Session management", defines the SQL-session management statements.

17) Clause 17, "Dynamic SQL", defines the SQL dynamic statements.

18) Clause 18, "Embedded SQL", defines the host language embeddings.

19) Clause 19, "Diagnostics management", defines the diagnostics management facilities.

20) Clause 20, "Information Schema", defines viewed tables that contain schema information.

21) Clause 21, "Definition Schema", defines base tables on which the viewed tables containing schema information depend.

22) Clause 22, "The SQL/XML XML Schema", defines the content of an XML namespace that is used when SQL and XML are utilized together.

23) Clause 23, "Status codes", defines values that identify the status of the execution of SQL-statements and the mechanisms by which those values are returned.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

24) Clause 24, “Conformance”, specifies the way in which conformance to this part of ISO/IEC 9075 may be claimed.

25) Annex A, “SQL Conformance Summary”, is an informative Annex. It summarizes the conformance requirements of the SQL language.

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

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

28) 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.

29) Annex E, “Incompatibilities with ISO/IEC 9075-14:2006”, is an informative Annex. It lists incompatibilities with the previous version of this part of ISO/IEC 9075.

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

31) Annex G, “Defect reports not addressed in this edition of this part of ISO/IEC 9075”, is an informative Annex. It describes the Defect Reports that were known at the time of publication of this part of this International Standard. Each of these problems is a problem carried forward from the previous edition of ISO/IEC 9075. No new problems have been created in the drafting of this edition of this International Standard.

In the text of this part of ISO/IEC 9075, Clauses begin a new odd-numbered page. Any resulting blank space is not significant.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Information technology — Database languages — SQL —

Part 14:
XML-Related Specifications (SQL/XML)

1 Scope

This part of ISO/IEC 9075 defines ways in which Database Language SQL can be used in conjunction with XML.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

(Blank page)

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2 Normative references

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

2.1 ISO and IEC standards

[ISO9075-1] ISO/IEC 9075-1:2008, *Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework)*.

[ISO9075-2] ISO/IEC 9075-2:2008, *Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation)*.

[ISO9075-4] ISO/IEC 9075-4:2008, *Information technology — Database languages — SQL — Part 4: Persistent Stored Modules (SQL/PSM)*.

[ISO9075-11] ISO/IEC 9075-11:2008, *Information technology — Database languages — SQL — Part 11: Information and Definition Schemas (SQL/Schemata)*.

[ISO10646] ISO/IEC 10646:2003, *Information technology — Universal Multi-Octet Coded Character Set (UCS)*.

2.2 Other international standards

[CanonicalXML] *(Recommendation) Canonical XML Version 1.0*, 15 March, 2001
`http://www.w3.org/TR/xml-c14n`

[Infoset] *(Recommendation) XML Information Set (Second Edition)*, 4 February, 2004
`http://www.w3.org/TR/2004/REC-xml-infoset-20040204`

[Namespaces] is used to reference either [Namespaces 1.0] or [Namespaces 1.1] when there is no significant difference between the two for the purposes of a given citation.

[Namespaces 1.0] *(Recommendation) Namespaces in XML 1.0 (Second Edition)*, First published 14 January 1999, revised 16 August 2006
`http://www.w3.org/TR/2006/REC-xml-names-20060816`

[Namespaces 1.1] *(Recommendation) Namespaces in XML 1.1 (Second Edition)*, First published 4 February 2004, revised 16 August 2006
`http://www.w3.org/TR/2006/REC-xml-names-20060816`

[RFC3986] RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax*, T. Berners-Lee, R. Fielding, L. Masinter, January, 2005
`http://www.ietf.org/rfc/rfc3986.txt`

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

[RFC2732] RFC 2732, *Format for Literal IPv6 Addresses in URLs*, R. Hinden, B. Carpenter, L. Masinter, December, 1999
http://www.ietf.org/rfc/rfc2732.txt

[Schema1] *(Recommendation) XML Schema Part 1: Structures Second Edition*, First published 2 May 2001, revised 28 October 2004
http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/

[Schema2] *(Recommendation) XML Schema Part 2: Datatypes Second Edition*, First published 2 May 2001, revised 28 October 2004
http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/

[Serialization] *(Recommendation) XSLT 2.0 and XQuery 1.0 Serialization*, W3C Recommendation, 23 January 2007
http://www.w3.org/TR/2007/REC-xslt-xquery-serialization-20070123/

[Unicode] The Unicode Consortium, *The Unicode Standard*. (Information about the latest version of the Unicode standard cam be found by using the "Latest Unicode Version" link on the "Enumerated Versions of The Unicode Standard" page.)
http://www.unicode.org/versions/enumeratedversions.html

[Unicode15] Davis, Mark and Dürst, Martin, *Unicode Standard Annex #15: Unicode Normalization Forms, Version 4.1.0*, 2005-03-25, The Unicode Consortium
http://www.unicode.org/reports/tr15/tr15-25.html

[UniXML] *(Note) Unicode in XML and Other Markup Languages*, 16 May 2007
http://www.w3.org/TR/2007/NOTE-unicode-xml-20070516/

[XML] is used to reference either [XML 1.0] or [XML 1.1] when there is no significant difference between the two for the purposes of a given citation. [UniXML] provides rules for character usage in XML.

[XML 1.0] *(Recommendation) Extensible Markup Language (XML) Version 1.0 (Fourth Edition)*, First published 4 February 2004, revised 16 August 2006
http://www.w3.org/TR/2006/REC-xml-20060816

[XML 1.1] *(Recommendation) Extensible Markup Language (XML) Version 1.1 (Second Edition)*, First published 4 February 2004, revised 16 August 2006
http://www.w3.org/TR/2006/REC-xml11-20060816

[XPath] *(Recommendation) XML Path Language (XPath) Version 2.0*, 23 January 2007
http://www.w3.org/TR/2007/REC-xpath20-20070123

[XQuery] *(Recommendation) XQuery 1.0: an XML Query Language*, W3C Recommendation, 23 January 2007
http://www.w3.org/TR/2007/REC-xquery-20070123/

[XQueryDM] *(Recommendation) XQuery 1.0 and XPath 2.0 Data Model*, W3C Recommendation, 23 January 2007
http://www.w3.org/TR/2007/REC-xpath-datamodel-20070123/

[XQueryFO] *(Recommendation) XQuery 1.0 and XPath 2.0 Functions and Operators*, W3C Recommendation, 23 January 2007
http://www.w3.org/TR/2007/REC-xpath-functions-20070123/

[XQueryFS] *(Recommendation) XQuery 1.0 and XPath 2.0 Formal Semantics*, W3C Recommendation, 23 January 2007

© ISO/IEC 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

http://www.w3.org/TR/2007/REC-xquery-semantics-20070123/

[XQueryUpdate] *(Candidate Recommendation)* XQuery Update Facility 1.0, W3C Working Draft, 14 March 2008
http://www.w3.org/TR/xquery-update-10/

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

(Blank page)

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3 Definitions, notations and conventions

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

3.1 Definitions

This Subclause modifies Subclause 3.1, "Definitions", in ISO/IEC 9075-2.

3.1.1 Definitions taken from XML

For the purposes of this document, the definitions of the following terms given in [XML] apply:

3.1.1.1 DTD

3.1.1.2 well-formed

3.1.1.3 XML declaration

3.1.1.4 XML document

3.1.2 Definitions taken from XML Schema

For the purposes of this document, the definitions of the following terms given in [Schema1] and [Schema2] apply:

3.1.2.1 annotation

3.1.2.2 facet

3.1.2.3 value space

3.1.2.4 wildcard schema component

3.1.2.5 XML Schema

3.1.3 Definitions provided in Part 14

For the purposes of this document, the following definitions apply:

3.1.3.1 **all group content model**: The content model of an XML Schema complex type described with **`xs:all`** as defined in [Schema1].

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

3.1.3.2 **canonical XML Schema literal**: A canonical lexical representation for an XML Schema type ***XST***, as defined in [Schema2], based on [CanonicalXML]. There is a unique canonical XML Schema literal for each value in the value space of ***XST***.

3.1.3.3 **empty XQuery sequence**: An empty sequence, as defined in [XQueryDM].

3.1.3.4 **global element declaration schema component**: An element declaration schema component of a global element declaration, as defined in [Schema1].

3.1.3.5 **metadata**: Data about data. In this International Standard, metadata is included in table descriptors, column descriptors, and so forth, as defined in [ISO9075-2] and other parts of this International Standard.

3.1.3.6 **sequence content model**: The content model of an XML Schema complex type described with `xs:sequence` as defined in [Schema1].

3.1.3.7 **SQL value space**: The set of all values for a particular SQL <data type>.

3.1.3.8 **URI**: Uniform Resource Identifier as defined in [RFC2396], as updated by [RFC2732].

3.1.3.9 **valid XML character**: If Feature X211, "XML 1.1 support", is supported, then a valid XML 1.1 character; otherwise, a valid XML 1.0 character.

3.1.3.10 **valid XML 1.0 character**: A legal character as defined in [XML 1.0], rule [2], "Char".

3.1.3.11 **valid XML 1.1 character**: A legal character as defined in [XML 1.1], rule [2], "Char".

3.1.3.12 **XML attribute**: An attribute as defined by [XML].

3.1.3.13 **XML attribute information item**: An attribute information item, as defined in [Infoset].

3.1.3.14 **XML character information item**: A character information item, as defined in [Infoset].

3.1.3.15 **XML declaration**: An XMLDecl, as defined in [XML].

3.1.3.16 **XML document information item**: A document information item, as defined in [Infoset].

3.1.3.17 **XML element**: An element as defined by [XML].

3.1.3.18 **XML element information item**: An element information item, as defined in [Infoset].

3.1.3.19 **XML information item**: An information item, as defined in [Infoset].

3.1.3.20 **XML 1.0 Name**: A Name as defined by [XML 1.0].

3.1.3.21 **XML 1.1 Name**: A Name as defined by [XML 1.1].

3.1.3.22 **XML 1.0 NameChar**: A NameChar as defined by [XML 1.0].

3.1.3.23 **XML 1.1 NameChar**: A NameChar as defined by [XML 1.1].

3.1.3.24 **XML 1.1 NameStartChar**: A NameStartChar as defined by [XML 1.1].

3.1.3.25 **XML namespace**: An XML namespace as defined by [Namespaces].

3.1.3.26 **XML namespace prefix**: A namespace prefix as defined by [Namespaces].

3.1.3.27 **XML 1.0 NCName**: An NCName, as defined by rule [4] in [Namespaces 1.0].

3.1.3.28 **XML 1.1 NCName**: An NCName, as defined by rule [4] in [Namespaces 1.1].

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3.1.3.29 **XML 1.0 QName**: A QName, as defined by rule [6] of [Namespaces 1.0].

3.1.3.30 **XML 1.1 QName**: A QName, as defined by rule [6] of [Namespaces 1.1].

3.1.3.31 **XML 1.0 QName prefix**: A Prefix, as defined by rule [7] of [Namespaces 1.0].

3.1.3.32 **XML 1.1 QName prefix**: A Prefix, as defined by rule [7] of [Namespaces 1.1].

3.1.3.33 **XML 1.0 QName local part**: A LocalPart, as defined by rule [8] of [Namespaces 1.0].

3.1.3.34 **XML 1.1 QName local part**: A LocalPart, as defined by rule [8] of [Namespaces 1.1].

3.1.3.35 **XML Schema built-in data type**: A built-in datatype, as defined in [Schema2].

3.1.3.36 **XML Schema complex type**: A complex type defined by a complex type definition, as defined in [Schema1].

3.1.3.37 **XML Schema data type**: A datatype, as defined in [Schema2].

3.1.3.38 **XML Schema document**: A schema document, as defined in [Schema1].

3.1.3.39 **XML Schema primitive type**: A primitive type, as defined in [Schema2].

3.1.3.40 **XML Schema simple type**: A simple type defined by a simple type definition, as defined in [Schema2].

3.1.3.41 **XML Schema type**: A term used to collectively refer to XML Schema built-in data types, XML Schema complex types, XML Schema data types, and XML Schema simple types, when it is not necessary to distinguish among those terms.

3.1.3.42 **XML target namespace**: A target namespace as defined by [Schema1].

3.1.3.43 **XML target namespace URI**: The URI of an XML target namespace.

3.1.3.44 **XML text**: A character string that is a substring of a textual XML 1.0 content or a textual XML 1.1 content, as defined in Subclause 10.16, “Parsing a string as an XML value”.

3.1.3.45 **XML value space**: A value space, as defined by [Schema2].

3.1.3.46 **XQuery accessor**: An accessor, as defined in [XQueryDM].

3.1.3.47 **XQuery atomic type**: An atomic type, as defined in [XQueryDM].

3.1.3.48 **XQuery atomic value**: An atomic value, as defined in [XQueryDM].

3.1.3.49 **XQuery attribute node**: An attribute node, as defined in [XQueryDM].

3.1.3.50 **XQuery comment node**: A comment node, as defined in [XQueryDM].

3.1.3.51 **XQuery datetime normalized value**: The normalized value of a value of XML Schema types **`xs:dateTime`**, **`xs:date`**, **`xs:time`**, or any type derived from these types, as this term is used in [XQueryFO], and implicitly defined in [XQueryDM], section 3.3.3, “Storing **`xs:dateTime`**, **`xs:date`**, and **`xs:time`** values in the data model”.

3.1.3.52 **XQuery datetime timezone component**: The timezone component of a value of XML Schema types **`xs:dateTime`**, **`xs:date`**, **`xs:time`**, or any type derived from these types, as this term is used in [XQueryFO] and implicitly defined in [XQueryDM], section 3.3.3, “Storing **`xs:dateTime`**, **`xs:date`**, and **`xs:time`** values in the data model”.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3.1.3.53 XQuery document node: A document node, as defined in [XQueryDM].

3.1.3.54 XQuery dynamic context: A dynamic context, as defined in [XQuery].

3.1.3.55 XQuery element node: An element node, as defined in [XQueryDM].

3.1.3.56 XQuery error: A static error, type error or dynamic error, as defined in [XQuery].

3.1.3.57 XQuery evaluation with XML 1.0 lexical rules: The process of determining the value of an XQuery expression with XML 1.0 lexical rules, as defined in [XQuery].

3.1.3.58 XQuery evaluation with XML 1.1 lexical rules: The process of determining the value of an XQuery expression with XML 1.1 lexical rules, as defined in [XQuery].

3.1.3.59 XQuery expression: An XQuery expression with XML 1.0 lexical rules, or an XQuery expression with XML 1.1 lexical rules.

3.1.3.60 XQuery expression context: An expression context, consisting of a static context and a dynamic context, as defined in [XQuery].

3.1.3.61 XQuery expression with XML 1.0 lexical rules: An expression, as defined by rule [40], "Expr", in [XQuery], with rules [19], "NCName", and [21], "QName", interpreted to reference [Namespaces 1.0].

3.1.3.62 XQuery expression with XML 1.1 lexical rules: An expression, as defined by rule [40], "Expr", in [XQuery], with rules [19], "NCName", and [21], "QName", interpreted to reference [Namespaces 1.1].

3.1.3.63 XQuery formal type notation: A formal type notation, as defined by rule [24 (Formal)] "Type", in [XQueryFS].

3.1.3.64 XQuery item: An item, as defined in [XQueryDM].

3.1.3.65 XQuery namespace node: A namespace node, as defined in [XQueryDM].

3.1.3.66 XQuery node: A node, as defined in [XQueryDM].

3.1.3.67 XQuery node identity: Node identity, as defined in [XQueryDM].

3.1.3.68 XQuery node property: A property of a node, as defined in [XQueryDM].

3.1.3.69 XQuery processing instruction node: A processing instruction node, as defined in [XQueryDM].

3.1.3.70 XQuery sequence: A sequence, as defined in [XQueryDM].

3.1.3.71 XQuery serialization normalization: Normalization, as defined in [Serialization] and [Unicode15].

3.1.3.72 XQuery static context: A static context, as defined in [XQuery].

3.1.3.73 XQuery text node: A text node, as defined in [XQueryDM].

3.1.3.74 XQuery tree: A tree, as defined in [XQueryDM].

3.1.3.75 XQuery variable: A variable, as defined in [XQuery].

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3.2 Notation

This Subclause modifies Subclause 3.2, "Notation", in ISO/IEC 9075-2.

Insert this paragraph Many of the standards referenced in Subclause 2.2, "Other international standards", are produced and maintained by the World Wide Web Consortium, including [XML], [XPath], [Namespaces], [Schema1], [Schema2], [Infoset], [XQuery]. W3C calls these international standards "Recommendations", and places them on its web site (http://www.w3.org). Whenever this part of ISO/IEC 9075 mentions a value, type, or other object (broadly conceived) that is specified by a W3C Recommendation, then that object is indicated using bold monospace font (for example, **`<xs:element>`**).

Insert this paragraph Similarly, when a textual variable in a Rule denotes an object that is specified by a W3C Recommendation, then the variable is written in italicized bold monospace font (for example, ***FACETP***).

Insert this paragraph The following list enumerates many, but not necessarily all, of the kinds of objects that are specified by W3C Recommendations:

— XML text.

— XML values.

— XML namespaces and XML namespace prefixes.

— XML Schema types.

— XQuery types.

— XQuery nodes.

— XQuery node properties.

— XQuery expressions.

— XQuery functions.

— XQuery variables.

— XQuery formal type notations.

— XQuery static and dynamic contexts.

Insert this paragraph On the other hand, an object that has some feature or aspect that is not specified by W3C is not in bold. For example, W3C objects do not have the ability to be null. Since a value of XML type may be null, it is not represented in bold. As another example, a registered XML Schema has an SQL privilege associated with it. In addition, non-bold variables are used for SQL BNF non-terminals, even when they are used to identify W3C objects.

Insert this paragraph This part of this International Standard specifies an interface between SQL and objects specified by W3C. As such, some values may be both W3C objects and also SQL objects. Therefore no notational convention can be totally consistent. In particular, it is permitted to assign a bold value to a non-bold variable, or to assign a non-bold value to a bold variable (provided the non-bold value has been determined to have no SQL-only feature, such as being a null). It is also permitted to compare a bold value and a non-bold value.

Insert this paragraph Whenever XML text is presented, an implementation may substitute equivalent XML text, for example, through insertion or deletion of insignificant blanks or new lines.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Insert this paragraph In this part of this International Standard, <left bracket>/<right bracket> pairs occur in five contexts:

— As part of BNF productions, in which a pair brackets surrounds one or more non-terminal and/or terminal symbols that are, treated as a single group, optional.

— In code examples, in which the brackets are literal characters that represent themselves as part of the code.

— In ordinary text, in which a pair of brackets enclose a word or phrase that identifies a publication referenced in Clause 2, "Normative references".

— In ordinary text, in which a pair of brackets enclose a number that identifies the number of a BNF production specified in the publication whose reference appears nearby.

— In ordinary text, in which a pair of brackets enclose a word or phrase that identifies the name of a property defined in [Infoset].

Insert this paragraph In this part of this International Standard, <left brace>/<right brace> pairs occur in three contexts:

— As part of BNF productions, in which a pair braces surrounds one or more non-terminal and/or terminal symbols that are to be handled as a single group.

— In code examples, in which the braces are literal characters that represent themselves as part of the code.

— In ordinary text, in which the braces enclose a word or phrase that identifies the name of a property defined in [Schema1].

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4 Concepts

This Clause modifies Clause 4, "Concepts", in ISO/IEC 9075-2.

4.1 Data types

This Subclause modifies Subclause 4.1, "Data types", in ISO/IEC 9075-2.

4.1.1 Naming of predefined types

This Subclause modifies Subclause 4.1.2, "Naming of predefined types", in ISO/IEC 9075-2.

Insert after 1st paragraph SQL defines a predefined data type named by the following <key word>: XML.

Insert after 3rd paragraph The data types XML(DOCUMENT(UNTYPED)), XML(DOCUMENT(ANY)), XML(DOCUMENT(XMLSCHEMA)), XML(CONTENT(UNTYPED)), XML(CONTENT(ANY)), XML(CONTENT(XMLSCHEMA)), and XML(SEQUENCE) are referred to as the *XML types*. Values of XML types are called *XML values*.

4.1.2 Data type terminology

This Subclause modifies Subclause 4.1.4, "Data type terminology", in ISO/IEC 9075-2.

Augment the list in the 11th paragraph

— A type *T* is *XML-ordered* if *T* is *S*-ordered, where *S* is the set consisting of the XML types.

4.2 XML

4.2.1 Introduction

Fuller descriptions of XML concepts can be found in the XML documents included in Clause 2, "Normative references".

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4.2.2 XML types

An XML type is described by an XML type descriptor. An XML type descriptor contains:

— The name of the data type (XML).

— The primary XML type modifier described by the <primary XML type modifier> (DOCUMENT, CONTENT, or SEQUENCE).

— The secondary XML type modifier described by the <secondary XML type modifier> (UNTYPED, ANY, or XMLSCHEMA), if any.

— The registered XML Schema descriptor of the indicated registered XML Schema, if any.

— The XML namespace URI of the indicated XML namespace, if any.

— The XML NCName of the indicated global element declaration schema component, if any.

NOTE 1 — "Indicated registered XML Schema", "indicated XML namespace", and "indicated global element declaration schema component" are defined in Subclause 11.6, "<XML valid according to clause>".

An XML type whose primary XML type modifier is DOCUMENT and whose secondary XML type modifier is UNTYPED is called an XML(DOCUMENT(UNTYPED)) type.

An XML type whose primary XML type modifier is DOCUMENT and whose secondary XML type modifier is ANY is called an XML(DOCUMENT(ANY)) type.

An XML type whose primary XML type modifier is DOCUMENT and whose secondary XML type modifier is XMLSCHEMA is called an XML(DOCUMENT(XMLSCHEMA)) type.

NOTE 2 — The number of XML(DOCUMENT(XMLSCHEMA)) types is determined by the number of registered XML Schemas, the number of XML namespaces in those XML Schemas, and the number of global element declaration schema components in those XML namespaces.

An XML type whose primary XML type modifier is CONTENT and whose secondary XML type modifier is UNTYPED is called an XML(CONTENT(UNTYPED)) type.

An XML type whose primary XML type modifier is CONTENT and whose secondary XML type modifier is ANY is called an XML(CONTENT(ANY)) type.

An XML type whose primary XML type modifier is CONTENT and whose secondary XML type modifier is XMLSCHEMA is called an XML(CONTENT(XMLSCHEMA)) type.

NOTE 3 — The number of XML(CONTENT(XMLSCHEMA)) types is determined by the number of registered XML Schemas, the number of XML namespaces in those XML Schemas, and the number of global element declaration schema components in those XML Schemas.

An XML type whose primary XML type modifier is SEQUENCE is called an XML(SEQUENCE) type.

When determining the declared type of an <XML value expression>, an SQL-implementation is permitted to use the XML(DOCUMENT(ANY)) type in place of XML(DOCUMENT(UNTYPED)) and XML(DOCUMENT(XMLSCHEMA)) types, the XML(CONTENT(ANY)) type in place of XML(CONTENT(UNTYPED)), XML(CONTENT(XMLSCHEMA)), and XML(DOCUMENT(ANY)) types, and the XML(SEQUENCE) type in place of the XML(CONTENT(ANY)) type.

NOTE 4 — The above substitutions can be applied transitively. For instance, an SQL-implementation is permitted to use the XML(SEQUENCE) type in place of the XML(DOCUMENT(UNTYPED)) type.

© ISO/IEC 2008 – All rights reserved
--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4.2.3 Characteristics of XML values

An *XML value* is either the null value or an XQuery sequence.

Every XML value is a value of type XML(SEQUENCE).

Every XML value that is either the null value or an XQuery document node is a value of type XML(CONTENT(ANY)).

Every XML value that is either the null value or a non-null value of type XML(CONTENT(ANY)) that is an XQuery document node ***D*** such that, for every XQuery element node that is contained in the XQuery tree ***T*** rooted in ***D***, the **type-name** property is **`xs:untyped`** and the **nilled** property is **`false`**, and for every XQuery attribute node that is contained in ***T***, the type property is **`xs:untypedAtomic`** is a value of **type-name** XML(CONTENT(UNTYPED)).

Every XML value that is either the null value or a non-null value of type XML(CONTENT(ANY)) that is an XQuery document node ***D*** such that every XQuery element node that is contained in the XQuery tree ***T*** rooted in ***D*** is valid according to an XML Schema ***S***, or is valid according to an XML namespace ***N*** in an XML Schema ***S***, or is valid according to a global element declaration schema component ***E*** in an XML Schema ***S*** is a value of type XML(CONTENT(XMLSCHEMA)) whose type descriptor includes the registered XML Schema descriptor of ***S***, and, if ***N*** is specified, the XML namespace URI of ***N***, or if ***E*** is specified, the XML namespace URI of ***E*** and the XML NCName of ***E***.

Every XML value that is either the null value or a non-null value of type XML(CONTENT(ANY)) that is an XQuery document node whose **children** property has exactly one XQuery element node, zero or more XQuery comment nodes, and zero or more XQuery processing instruction nodes is a value of type XML(DOCUMENT(ANY)).

Every XML value that is either the null value or a non-null value of type XML(CONTENT(UNTYPED)) that is an XQuery document node whose **children** property has exactly one XQuery element node, zero or more XQuery comment nodes, and zero or more XQuery processing instruction nodes is a value of type XML(DOCUMENT(UNTYPED)).

Every XML value that is either the null value or a non-null value of type XML(DOCUMENT(ANY)) that is valid according to an XML Schema ***S***, or that is valid according to an XML namespace ***N*** in an XML Schema ***S***, or that is valid according to a global element declaration schema component ***E*** in an XML Schema ***S*** is a value of type XML(DOCUMENT(XMLSCHEMA)) whose type descriptor includes the registered XML Schema descriptor of ***S***, and, if ***N*** is specified, then the XML namespace URI of ***N***, or if ***E*** is specified, the XML namespace URI of ***E*** and the XML NCName of ***E***.

4.2.4 XML comparison and assignment

A value of an XML type *S* is assignable to a site of an XML type *T* if any of the following is true:

— *T* is either XML(DOCUMENT(UNTYPED)) or XML(CONTENT(UNTYPED)) and *S* is either XML(DOCUMENT(UNTYPED)) or XML(CONTENT(UNTYPED)).

— *T* is either XML(DOCUMENT(XMLSCHEMA)) or XML(CONTENT(XMLSCHEMA)) and *S* is either XML(DOCUMENT(XMLSCHEMA)) or XML(CONTENT(XMLSCHEMA)) such that the registered XML Schema descriptors included in the XML type descriptors of both *T* and *S* identify identical XML Schemas, the XML namespace URI included in the XML type descriptor of *T*, if any, is identical to the XML namespace URI included in the XML type descriptor of *S*, as defined by [Namespaces], and the XML

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

NCName of the global element declaration schema component included in the XML type descriptor of *T*, if any, is identical to the XML NCName of the global element declaration schema component included in the XML type descriptor of *S*.

NOTE 5 — The notion of identical XML Schemas is defined in Subclause 4.2.6, "Registered XML Schemas".

— *T* is either XML(DOCUMENT(ANY)), XML(CONTENT(ANY)), or XML(SEQUENCE) and *S* is an XML type.

Operations that involve assignment of XML values use the keywords BY REF to indicate that the assignment preserves XQuery node identity, and BY VALUE to indicate that the assignment loses XQuery node identity.

XML values are not comparable.

4.2.5 Operations involving XML values

<XML document> is an operator that returns an XML value, given another XML value. The new XML value consists of an XQuery document node that is constructed according to the rules of the computed document constructor in [XQuery].

<XML element> is an operator that returns an XML value given an XML element name, an optional list of XML namespaces, an optional list of XML attributes, and an optional list of values as the content of the new element. The value of <XML element content> can be any value that has a mapping to an XML value. The result, an XQuery element node, may optionally be placed as the sole child of an XQuery document node.

<XML forest> is an operator that returns an XML value, given an optional list of XML namespaces and a list of <forest element>s. An XQuery element node is produced from each <forest element>, using the column name or, if provided, the <forest element name> as the XML element name and the <forest element value> as the element content. The value of <forest element value> can be any value that has a mapping to an XML value. The result, an XQuery sequence, may optionally be placed in an XQuery document node.

<XML concatenation> is an operator that returns an XML value by concatenating a list of XML values, as defined in Subclause 6.12, "<XML concatenation>". The result, an XQuery sequence, may optionally be placed in an XQuery document node.

<XML comment> is an operator that returns an XML value, given a character string *CS*. The XML value consists of an XQuery comment node whose **content** property is *CS*, mapped to Unicode. This XQuery comment node may optionally be placed as the sole child of an XQuery document node.

<XML PI> is an operator that returns an XML value, given an <identifier> and an optional character string *CS*. The XML value consists of an XQuery processing node whose **target** property is the <identifier>, and whose **content** property is *CS*, trimmed of leading blanks and mapped to Unicode. This XQuery processing instruction node may optionally be placed as the sole child of an XQuery document node.

<XML text> is an operator that returns an XML value, given a character string *CS*. The XML value consists of an XQuery text node whose **content** property is *CS*, mapped to Unicode. The XQuery text node may optionally be placed as the sole child of an XQuery document node.

<XML query> is an operator that evaluates an XQuery expression, which may be parameterized with any number of input parameters. The result, an XQuery sequence, may optionally be placed in an XQuery document node.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

<XML table> is a kind of <derived table>, which may be used to query an XML value as a table. The result is specified by means of <XML table row pattern>, <XML table argument list>, and <XML table column definitions>. The result is computed in the following steps:

— The <XML table row pattern> is an XQuery expression that is evaluated, with <XML table argument list> as the (optional) input arguments. This produces an intermediate result, an XQuery sequence.

— Each XQuery item in the intermediate result is used to create a row of the final result.

— The columns of the result are specified by <XML table column definitions>, which is a collection of <XML table column definition>s modeled on a <table definition>. Thus, each result column has a <data type>, and, optionally, a <default clause>. Each <XML table column definition> also specifies, explicitly or implicitly, an <XML table column pattern> *XTCP*, which is an XQuery expression.

— To generate the value for a particular row and column of the final result, the column pattern (*i.e.*, the XQuery expression *XTCP*) is evaluated using a particular XQuery item of the intermediate result as the input parameter, and then cast to the <data type> of the result column. If *XTCP* evaluates to the empty XQuery sequence, then the column's <default clause>, if any, determines the column's default value.

<XML character string serialization> is an operator that returns a character string, given an XML value, using the algorithm in [Serialization]. Optionally, the input XML value may be checked to ensure that it is an XQuery document node whose **children** property has exactly one XQuery element node. Other options may be used to specify the version of XML text that is produced (*i.e.*, either a textual XML 1.0 content or a textual XML 1.1 content); and whether an XML declaration is produced or not.

<XML binary string serialization> is an operator that returns a binary string, given an XML value, using the algorithm in [Serialization]. <XML binary string serialization> supports the same options as <XML character string serialization>, plus the ability to specify the encoding (character set) of the result.

<XML parse> is an operator that parses a character string or binary string value, i.e., converts XML text to a set of XML information items, according to the rules of [Infoset], and then converts that set of XML information items into an XQuery document node, according to the rules of [XQueryDM].

<XML validate> is an operator that validates an XML value according to a registered XML Schema. Optionally, the registered XML Schema against which to validate may be chosen on the basis of the contents of the data value, or it may be specified directly in the <XML validate>. In the latter case, additional options allow the validation to be confined to a particular namespace of the designated registered XML Schema, or to a particular global element declaration schema component. Upon successful validation, <XML validate> returns a copy of the input XML value augmented with default values and type annotations. If validation fails, <XML validate> raises an exception.

<XML content predicate> is a predicate that determines if an XML value is an XQuery document node.

<XML document predicate> is a predicate that determines if an XML value is an XQuery document node whose **children** property contains exactly one XQuery element node, zero or more XQuery comment nodes, and zero or more XQuery processing instruction nodes.

<XML exists predicate> is a predicate that evaluates an XQuery expression and determines if the result is a null value, an empty XQuery sequence, or a non-empty XQuery sequence.

<XML valid predicate> is a predicate that determines if an XML value is valid according to a registered XML Schema. Optionally, the registered XML Schema to validate against may be chosen on the basis of the contents of the data value, or it may be specified directly in the <XML valid predicate>. In the latter case, additional options allow the validation to be confined to a particular namespace of the designated registered XML Schema, or to a particular global element declaration schema component.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4.2.6 Registered XML Schemas

A *registered XML Schema* is an XML Schema that has been made known to the SQL-server. The means by which an XML Schema is registered is implementation-defined.

A global element declaration schema component of a registered XML Schema is *non-deterministic* if it contains or references a wildcard schema component whose **{namespace constraint}** property is either **not** together with a namespace name, or **any**, and whose **{process contents}** property is either **strict** or **lax** (as defined in [Schema1] section 3.10.1, "The Wildcard Schema Component").

NOTE 6 — The elements of an XML Schema Document corresponding to such wildcard schema components are elements **<xs:any>** or **<xs:anyAttribute>** in which the **namespace** attribute is either missing or has the value "**##any**" or "**##other**", and the **processContents** attribute is either missing or has either the value "**strict**" or the value "**lax**".

An XML namespace *NS* contained in a registered XML Schema is *non-deterministic* if *NS* contains a global element declaration schema component that is non-deterministic.

A registered XML Schema is *non-deterministic* if it contains a non-deterministic XML namespace.

A registered XML Schema is described by a registered XML Schema descriptor. A registered XML Schema descriptor includes:

— The target namespace URI of the registered XML Schema.

— The schema location URI of the registered XML Schema.

— The <registered XML Schema name> of the registered XML Schema.

— An indication of whether the registered XML Schema is permanently registered.

— An indication of whether the registered XML Schema is non-deterministic.

— An unordered collection of the namespaces defined by the registered XML Schema (the target namespace is one of these namespaces).

— For each namespace defined by the registered XML Schema, an unordered collection of the global element declaration schema components in that namespace, with an indication for each global element declaration schema component whether that global element declaration schema component is non-deterministic.

NOTE 7 — Without Feature X161, "Advanced Information Schema for registered XML Schemas", information whether an XML Schema is deterministic, information about the collection of namespaces defined in that XML Schema, and, for each such namespace information about the global element declaration schema components in that namespace, is not available in the XML_SCHEMAS, XML_SCHEMA_NAMESPACES, and XML_SCHEMA_ELEMENTS views.

A registered XML Schema is identified by its <registered XML Schema name>.

Two registered XML Schemas are considered *identical* if both are identified by the same <registered XML Schema name>.

Certain XML Schemas, defined by either this part of ISO/IEC 9075 or by some normative reference, are always registered. These XML Schemas are enumerated in Table 1, "Permanently registered XML Schemas".

--`,,```,,,,````-`-`,,`,,`,`,,`---
© ISO/IEC 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 1 — Permanently registered XML Schemas

Common prefix (non-normative)	target namespace URI	Schema location URI	<registered XML Schema name>
`xs`	`http://www.w3.org/2001/XMLSchema`	implementation-defined	implementation-defined
`xsi`	`http://www.w3.org/2001/XMLSchema-instance`	implementation-defined	implementation-defined
`sqlxml`	`http://standards.iso.org/iso/9075/2003/sqlxml`	implementation-defined	implementation-defined

NOTE 8 — The "common prefix" column in the preceding table indicates the prefix(es) commonly associated with the target namespaces of these XML Schemas, but there is no requirement to refer to them by these prefixes.

If a <data type>, an <XML validate>, or an <XML valid predicate> that contains an <XML valid according to what> *XVACC* is contained in a <query expression> of a view, a check constraint, or an assertion, the <triggered action> of a trigger, or in an <SQL-invoked routine>, then the registered XML Schema that is referenced by *XVACC* is determined at the time the view is created, the check constraint is defined, the assertion is created, the trigger is created, or the SQL-invoked routine is created. The same registered XML Schema is referenced whenever the view is used, or the check constraint or assertion is evaluated, the trigger is executed, or the SQL-invoked routine is invoked.

4.3 Data conversions

This Subclause modifies Subclause 4.11, "Data conversions", in ISO/IEC 9075-2.

Insert before 3rd paragraph Data conversions between the predefined data types defined in [ISO9075-2] and the XML types can be specified by an <XML cast specification>.

A conversion from an XML type to a non-XML type is computed as follows:

1) XQuery document nodes are removed from the source value.

2) The result is converted to an XQuery sequence of XQuery atomic values ***AV*** using the XQuery function **`fn:data()`**.

3) An XML Schema data type ***XT*** is chosen to correspond to the target <data type>. For example, if the target <data type> is INTEGER then *XT* is **`xs:integer`** or an XML Schema data type derived from **`xs:integer`**.

4) ***AV*** is converted to ***XT*** using the rules of [XQuery], producing ***BV***.

5) ***BV*** is a value of an XML Schema data type, which is either an XML Schema primitive type, or derived from an XML Schema primitive type. For example, ***BV*** might be a value of XML Schema type **`xs:short`**, which is derived from **`xs:decimal`**. The value spaces of most XML Schema primitive types are identified

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

with an SQL value space. For example, the XML value space of **xs:decimal** is identified with the SQL value space of exact numeric values. If, say, ***BV*** has the value **9075.14** in the XML value space of **xs:decimal**, then ***BV*** is also regarded as being the value **9075.14** in the SQL value space of exact numeric values. These value space identifications are defined in Subclause 4.10.7, "Mapping XQuery atomic values to SQL values".

6) ***BV***, regarded now as a value of an SQL value space, is converted to the target <data type>, using a <cast specification>.

A conversion from a non-XML type to an XML type is computed as follows:

1) The source value is converted to a <character string literal> whose value is the result of applying the General Rules of Subclause 9.8, "Mapping values of SQL data types to values of XML Schema data types".

2) The <character string literal> is parsed using <XML parse>, producing an XQuery document node ***D***.

3) If the target <data type> is XML(CONTENT(UNTYPED)) or XML(CONTENT(ANY)), then ***D*** is the result.

4) If the target <data type> is XML(SEQUENCE), then ***D*** is converted to an XQuery atomic value by evaluating an XQuery cast expression. The XQuery type to convert to is chosen based on the declared type of the source value. For example, if the declared type of the source is exact numeric, then the XQuery type of the result is **xs:decimal**.

4.4 Data analysis operations (involving tables)

This Subclause modifies Subclause 4.15, "Data analysis operations (involving tables)", in ISO/IEC 9075-2.

4.4.1 Aggregate functions

This Subclause modifies Subclause 4.15.4, "Aggregate functions", in ISO/IEC 9075-2.

Augment the 7th paragraph

— If XMLAGG is specified, then an XML value formed from the <XML value expression> evaluated for each row that qualifies.

4.5 SQL-invoked routines

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

4.5.1 Routine descriptors

This Subclause modifies Subclause 4.27.4, "Routine descriptors", in ISO/IEC 9075-2.

Augment the routine descriptor of SQL routines

© ISO/IEC 2008 – All rights reserved
--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

— For every SQL parameter whose declared type is an XML type or a distinct type whose source type is an XML type, an indication of the <XML passing mechanism>.

— If the SQL routine is an SQL-invoked function, then an indication of the <XML passing mechanism> of the <returns clause>.

Augment the routine descriptor of external routines

— For every SQL parameter that has an associated string type, the routine descriptor includes the character string type descriptor of the associated string type.

— For every SQL parameter that has an associated XML option, the routine descriptor includes an indication of the associated XML option.

4.6 SQL-statements

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

4.6.1 SQL-statements classified by function

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

4.6.1.1 SQL-session statements

This Subclause modifies Subclause 4.33.2.7, "SQL-session statements", in ISO/IEC 9075-2.

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

— <set XML option statement>

4.7 Basic security model

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

4.7.1 Privileges

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

Augment the list following 1st paragraph

— registered XML Schema

Augment the list following 8th paragraph

© ISO/IEC 2008 – All rights reserved

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

— registered XML Schema

Append this paragraph USAGE privileges on registered XML Schemas are granted or revoked by implementation-defined means.

4.8 SQL-sessions

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

4.8.1 SQL-session properties

This Subclause modifies Subclause 4.37.3, "SQL-session properties", in ISO/IEC 9075-2.

Insert after 6th paragraph An SQL-session has an XML option that is used to identify the <document or content> option needed in the implicit invocation of XMLSERIALIZE and XMLPARSE operators during the execution of <preparable statement>s that are prepared in the current SQL-session by either an <execute immediate statement> or a <prepare statement>. The XML option is initially set to an implementation-defined value, but can subsequently be changed by the successful execution of a <set XML option statement>.

Augment the 12th paragraph

— The current XML option.

4.9 XML namespaces

This part of ISO/IEC 9075 references certain XML namespaces that are defined by the World-Wide Web Consortium or by this standard. Each XML namespace is referenced using an XML namespace prefix. The XML namespace prefixes and their definitions are shown in Table 2, "XML namespace prefixes and their URIs".

Table 2 — XML namespace prefixes and their URIs

XML namespace prefix	XML namespace URI
xs	**http://www.w3.org/2001/XMLSchema**
xsi	**http://www.w3.org/2001/XMLSchema-instance**
sqlxml	**http://standards.iso.org/iso/9075/2003/sqlxml**

A conforming implementation is not required to use the XML namespace prefixes **xs**, **xsi**, or **sqlxml** to reference these XML namespaces, but whatever XML namespace prefix it uses shall be associated with the proper URI.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

The XML namespace identified by the XML namespace prefix "**sqlxml**" is normatively defined in Clause 22, "The SQL/XML XML Schema".

A resource containing the XML Schema definition of the XML namespace identified by the XML namespace prefix "**sqlxml**" (that is, a file containing an XML Schema document) has been made available on the World Wide Web. The URI of that resource is:

http://standards.iso.org/iso/9075/2003/sqlxml.xsd

It is intended that the contents of that file be identical to the contents of Clause 22, "The SQL/XML XML Schema". This file has been created for the convenience of the implementors of this part of ISO/IEC 9075.

4.10 Overview of mappings

This International Standard defines mappings from SQL to XML, and from XML to SQL. The mappings from SQL to XML include:

— Mapping SQL character sets to Unicode.

— Mapping SQL <identifier>s to XML Names.

— Mapping SQL data types (as used in SQL-schemas to define SQL-schema objects such as columns) to XML Schema data types.

— Mapping values of SQL data types to values of XML Schema data types.

— Mapping an SQL table to an XML document and an XML Schema document.

— Mapping an SQL schema to an XML document and an XML Schema document.

— Mapping an SQL catalog to an XML document and an XML Schema document.

The mappings from XML to SQL include:

— Mapping Unicode to SQL character sets.

— Mapping XML Names to SQL <identifier>s.

4.10.1 Mapping SQL character sets to Unicode

For each character set *SQLCS* in the SQL-environment, there shall be a mapping *CSM* of strings of *SQLCS* to strings of Unicode, as defined in [Unicode]. In this part of this International Standard, "Unicode" refers to the character repertoire named "UCS". The mapping *CSM* is called *homomorphic* if for each nonnegative integer *N*, there exists a nonnegative integer *M* such that all strings of length *N* in *SQLCS* are mapped to strings of length *M* in Unicode. *CSM* is implementation-defined. However, if any Unicode code point is mapped to a character that is not a valid XML character, an exception condition is raised.

NOTE 9 — The entity references **<**, **&**, **>**, **'**, and **"**, as well as character references, as defined by [XML] are regarded as each representing a single character in XML, and do not pose an obstacle to defining homomorphic mappings.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

4.10.2 Mapping Unicode to SQL character sets

For each character set *SQLCS* in the SQL-environment, there shall be an implementation-defined mapping *CSM* of strings of Unicode to strings of *SQLCS*.

4.10.3 Mapping SQL <identifier>s to XML

Since not every SQL <identifier> is an acceptable XML Name, it is necessary to define a mapping of SQL <identifier>s to XML Names. This mapping is defined in Subclause 9.1, "Mapping SQL <identifier>s to XML Names". The basic idea of this mapping is that characters that are not valid in XML Names are converted to a sequence of hexadecimal digits derived from the Unicode encoding of the character, bracketed by an introductory underscore and lowercase `x` and a trailing underscore.

There are two variants of the mapping, known as *partially escaped* and *fully escaped*. The two differences are in the treatment of non-initial <colon> and the treatment of an <identifier> beginning with the letters `xml` in any combination of upper or lower case. The fully escaped variant maps a non-initial <colon> to **`_x003A_`**, whereas the partially escaped variant maps non-initial <colon> to **`:`**. Also, the fully escaped variant maps initial `xml` and `XML` to **`_x0078_ml`** and **`_x0058_ML`**, respectively, whereas the partially escaped does not.

NOTE 10 — This part of this International Standard specifies no syntax for invoking the partially escaped mapping specified in Subclause 9.1, "Mapping SQL <identifier>s to XML Names". This specification is intended to be used by applications and referenced by other standards.

4.10.4 Mapping XML Names to SQL

A single algorithm suffices to reverse both the partially escaped and the fully escaped variants of the mapping of SQL <identifier>s to XML Names. This algorithm is found in Subclause 9.3, "Mapping XML Names to SQL <identifier>s". The basic idea is to scan the XML Name from left to right, looking for escape sequences of the form **_x*NNNN*_** or **_x*NNNNNN*_** where ***N*** denotes a hexadecimal digit. Such sequences are converted to the character of SQL_TEXT that corresponds to the Unicode code point U+0000*NNNN* or U+00*NNNNNN*, respectively.

NOTE 11 — This part of this International Standard specifies no syntax for invoking the mapping specified in Subclause 9.3, "Mapping XML Names to SQL <identifier>s". This specification is intended to be used by applications and referenced by other standards. It is the responsibility of any such application or other standard to ensure that the correct number of arguments as well as a valid value for each argument are supplied for this mapping.

NOTE 12 — The sequence of mappings from SQL <identifier> to XML Name (using either the fully escaped mapping or the partially escaped mapping) to SQL <identifier> restores the original SQL <identifier> (assuming that every character in the source SQL-implementation's SQL <identifier> is a character of SQL_TEXT in the target SQL-implementation). However, the sequence of mappings from XML Name to SQL <identifier> to XML Name does not necessarily restore the XML Name. Also, more than one XML Name may be mapped to the same SQL <identifier>.

4.10.5 Mapping SQL data types to XML

For each SQL type or domain that is not unmappable, there is a corresponding XML Schema type. The mapping is fully specified in Subclause 9.5, "Mapping SQL data types to XML Schema data types". The following is a conceptual description of this mapping.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

In general, each SQL predefined type, distinct type, or domain *SQLT* is mapped to the XML Schema type ***XMLT*** that is the closest analog to *SQLT*. Since the value space of ***XMLT*** is frequently richer than the set of values that can be represented by *SQLT*, facets are used to restrict ***XMLT*** in order to capture the restrictions on *SQLT* as much as possible.

In addition, many of the distinctions in the SQL type system (for example, CHARACTER VARYING *versus* CHARACTER LARGE OBJECT) have no corresponding distinction in the XML Schema type system. In order to represent these distinctions, XML Schema annotations are defined. The content of the annotations is defined by this standard; however, whether such annotations are actually generated is implementation-defined. Elements from the XML namespace identified by the XML namespace prefix "**sqlxml**" are used to populate these annotations.

The SQL character string types are mapped to the XML Schema type **xs:string**. For the SQL type CHARACTER, if the mapping of the SQL character set to Unicode is homomorphic, then fixed length strings are mapped to fixed length strings, and the facet **xs:length** is used. Otherwise (*i.e.*, CHARACTER when the mapping is not homomorphic, as well as CHARACTER VARYING and CHARACTER LARGE OBJECT), the facet **xs:maxLength** is used. Annotations optionally indicate the precise SQL type (CHARACTER, CHARACTER VARYING, or CHARACTER LARGE OBJECT), the length or maximum length of the SQL type, the character set, and the default collation.

The SQL binary string types are mapped to either the XML Schema type **xs:hexBinary** or the XML Schema type **xs:base64Binary**. The **xs:maxLength** facet is set to the maximum length of the binary string in octets. Annotations optionally indicate the SQL type (BINARY LARGE OBJECT) and the maximum length in octets. For <XML element> and <XML forest>, the choice of whether to map to **xs:hexBinary** or **xs:base64Binary** is governed by the innermost <XML binary encoding> whose scope includes the <XML element> or <XML forest>; the default is implementation-defined. When mapping an SQL table, schema or catalog to XML, the choice is governed by a parameter, as specified in Subclause 9.11, "Mapping an SQL table to XML and an XML Schema document", Subclause 9.14, "Mapping an SQL schema to an XML document and an XML Schema document", and Subclause 9.17, "Mapping an SQL catalog to an XML document and an XML Schema document".

The exact numeric SQL types NUMERIC and DECIMAL are mapped to the XML Schema type **xs:decimal** using the facets **xs:precision** and **xs:scale**. It is implementation-defined whether the SQL types INTEGER, SMALLINT, and BIGINT are mapped to the XML Schema type **xs:integer** using the facets **xs:maxInclusive** and **xs:minInclusive** or to the closest XML Schema type that is a subtype of **xs:integer**, using the facets **xs:maxInclusive** and **xs:minInclusive** if the range of the SQL type does not exactly match the range of the XML Schema type to which it is mapped. Annotations optionally indicate the SQL type (NUMERIC, DECIMAL, INTEGER, SMALLINT, or BIGINT), precision of NUMERIC, user-specified precision of DECIMAL (which may be less than the actual precision), and scale of NUMERIC and DECIMAL.

The approximate numeric SQL types are mapped to either the XML Schema type **xs:float**, if the binary precision is less than or equal to 24 binary digits (bits) and the range of the binary exponent lies between -149 and 104, inclusive; otherwise, the XML Schema type **xs:double** is used. Annotations optionally indicate the SQL type (REAL, DOUBLE PRECISION, or FLOAT), the binary precision, the minimum and maximum values of the range of binary exponents, and, for FLOAT, the user-specified binary precision (which may be less than the actual precision).

The SQL type BOOLEAN is mapped to the XML Schema type **xs:boolean**. Optionally, an annotation indicates the SQL type (BOOLEAN).

The SQL type DATE is mapped to the XML Schema type **xs:date**. The **xs:pattern** facet is used to exclude the possibility of a time zone displacement. Optionally, an annotation indicates the SQL type, DATE.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

The SQL types TIME WITHOUT TIME ZONE and TIME WITH TIME ZONE are mapped to the XML Schema type **xs:time**. The **xs:pattern** facet is used to exclude the possibility of a time zone displacement, in the case of TIME WITHOUT TIME ZONE, or to require a time zone displacement, in the case of TIME WITH TIME ZONE. The pattern also reflects the fractional seconds precision of the SQL type. Annotations optionally indicate the SQL type (TIME or TIME WITH TIME ZONE) and the fractional seconds precision.

The SQL types TIMESTAMP WITHOUT TIME ZONE and TIMESTAMP WITH TIME ZONE are mapped to the XML Schema type **xs:dateTime**. The **xs:pattern** facet is used to exclude the possibility of a time zone displacement, in the case of TIMESTAMP WITHOUT TIME ZONE, or to require a time zone displacement, in the case of TIMESTAMP WITH TIME ZONE. The pattern also reflects the fractional seconds precision of the SQL type. Annotations optionally indicate the SQL type (TIMESTAMP or TIMESTAMP WITH TIME ZONE) and the fractional seconds precision.

The SQL interval types are mapped to the XML Query types **xs:yearMonthDuration** and **xs:dayTime-Duration**. The **xs:pattern** facet is used to require precisely the year, month, day, hour, minute and second fields indicated by the SQL type. The pattern also reflects the leading field precision and the fractional seconds precision (when applicable). Annotations optionally indicate the SQL type, leading field precision and (when applicable) the fractional seconds precision.

An SQL row type is mapped to an XML Schema complex type that consists of one element for each field of the SQL row type. For each field *F* of the SQL row type, the name of the corresponding XML element is obtained by mapping the field name of *F* using the fully escaped variant, and the XML Schema type of the element is obtained by mapping the field type of *F*.

An SQL domain is mapped to XML by mapping the domain's data type to XML and then optionally applying to the generated XML Schema type an annotation that identifies the name of the domain.

An SQL distinct type is mapped to an XML Schema simple type by mapping the source type of the distinct type. Optionally, an annotation specifying the name of the distinct type is applied to the generated XML Schema type.

An SQL collection type is mapped to an XML Schema complex type having a single XML element named **element** whose XML Schema type is obtained by mapping the element type of the SQL collection type. This XML element is defined using **minOccurs="0"**. For an SQL array type, **maxOccurs** is the maximum cardinality of the array, whereas for an SQL multiset type, **maxOccurs="unbounded"**.

An SQL XML type is mapped to an XML Schema complex type that allows mixed content and an unvalidated **any** section. Optionally, an annotation indicates the SQL type, XML.

4.10.6 Mapping values of SQL data types to XML

For each SQL type or domain *SQLT*, with the exception of structured types and reference types, there is also a mapping of values of type *SQLT* to the value space of the corresponding XML Schema type. The mappings of values are largely determined by the data type mappings. The precise rules for non-null values are found in Subclause 9.8, “Mapping values of SQL data types to values of XML Schema data types”. The mappings for values of predefined types are designed to exploit <cast specification> as much as possible. As for null values, there is generally a choice of whether to represent nulls using absence or **xsi:nil="true"**. However, for elements of a collection type, null values are always represented by **xsi:nil="true"**.

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4.10.7 Mapping XQuery atomic values to SQL values

As defined in [XQueryDM], an XQuery atomic type is either an XML Schema primitive type, or derived from an XML Schema primitive type by restriction (and not by union or list).

Let ***AV*** be an XQuery atomic value. Let ***AT*** be the XQuery atomic type of ***AV***. Let ***PT*** be given by

Case:

— If ***AT*** is an XML Schema primitive type, then ***AT***.

— If ***AT*** is **xs:yearMonthDuration**, or derived from **xs:yearMonthDuration**, then **xs:yearMonthDuration**.

— If ***AT*** is **xs:dayTimeDuration**, or derived from **xs:dayTimeDuration**, then **xs:dayTimeDuration**.

— Otherwise, the XML Schema primitive type from which ***AT*** is derived.

This part of ISO/IEC 9075 (notably, in Subclause 6.6, "<XML cast specification>") regards ***AV*** as being a value belonging to some category of SQL predefined type, as follows.

Case:

— If ***PT*** is **xs:string**, then ***AV*** is regarded as being a character string whose character repertoire is Unicode.

— If ***PT*** is **xs:hexBinary** or **xs:base64Binary**, then ***AV*** is regarded as being a binary string.

— If ***PT*** is **xs:decimal**, then ***AV*** is regarded as being an exact numeric value.

— If ***PT*** is **xs:float** or **xs:double**, then ***AV*** is regarded as being an approximate numeric value.

— If ***PT*** is **xs:time** and the XQuery datetime timezone component of ***AV*** is an empty XQuery sequence, then the XQuery datetime normalized value of ***AV*** is regarded as being a value of type TIME WITHOUT TIME ZONE.

— If ***PT*** is **xs:time** and the XQuery datetime timezone component of ***AV*** is not an empty XQuery sequence, then ***AV*** is regarded as being a value of type TIME WITH TIME ZONE, in which the XQuery datetime timezone component of ***AV*** is the timezone component, and the XQuery datetime normalized value is the UTC component.

— If ***PT*** is **xs:dateTime**, the XQuery datetime normalized value ***XDNV*** of ***AV*** is positive, and the XQuery datetime timezone component of ***AV*** is an empty XQuery sequence, then ***XDNV*** is regarded as being a value of type TIMESTAMP WITHOUT TIME ZONE. If ***XDNV*** would have a SECOND field greater than or equal to 59 in a minute of UTC that has exactly 59 seconds, then it is implementation-defined whether an implementation-defined value of type TIMESTAMP WITHOUT TIME ZONE is identified with ***XDNV***, or whether an exception condition is raised: *data exception — datetime field overflow*.

— If ***PT*** is **xs:dateTime**, the XQuery datetime normalized value of ***AV*** is positive, and the XQuery datetime timezone component of ***AV*** is not an empty XQuery sequence, then ***AV*** is regarded as being a value of type TIMESTAMP WITH TIME ZONE, in which the XQuery datetime timezone component of ***AV*** is the timezone component, and the XQuery datetime normalized value is the UTC component. If AV denotes a minute of UTC that has exactly 59 seconds and the SECOND field ***AV*** is greater than or equal to 59, then it is implementation-defined whether an implementation-defined value of type TIMESTAMP WITHOUT TIME ZONE is identified with ***AV***, or whether an exception condition is raised: *data exception — datetime field overflow*.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

— If ***PT*** is **xs:date**, the XQuery datetime normalized value of ***AV*** is positive, and the XQuery datetime timezone component of ***AV*** is an empty XQuery sequence, then the XQuery datetime normalized value of ***AV*** is regarded as being a value of type DATE.

— If ***PT*** is **xs:yearMonthDuration**, then ***AV*** is regarded as being a year-month interval.

— If ***PT*** is **xs:dayTimeDuration**, then ***AV*** is regarded as being a day-time interval.

— If ***PT*** is **xs:boolean**, then ***AV*** is regarded as being a value of type BOOLEAN.

4.10.8 Visibility of columns, tables, and schemas in mappings from SQL to XML

An *XML unmappable data type* is a data type that is one of the following: a structured type, a reference type, XML(SEQUENCE), or a type defined in some part of ISO/IEC 9075 other than [ISO9075-2] and this part. An *XML unmappable column* is a column that has a declared type that is one of the XML unmappable data types or has a declared type that is based on an XML unmappable data type.

A column *C* of table *T* is a *visible column* of *T* for authorization identifier *U* if the applicable privileges for *U* include the SELECT privilege on *C* and, if the declared type of *C* is a distinct type, the applicable privileges for *U* include EXECUTE on the user-defined cast function identified by the Syntax Rules of Subclause 6.5, "<cast specification>". A column *C* of table *T* is an *XML visible column* of *T* for authorization identifier *U* if *C* is a visible column of *T* for authorization identifier *U* and the declared type of *C* is not an XML unmappable data type.

A table *T* of schema *S* is a *visible table* of *S* for authorization identifier *U* if *T* is either a base table or a viewed table that contains a column *C* that is a visible column for *U*. A table *T* of schema *S* is an *XML visible table* of *S* for authorization identifier *U* if *T* is either a base table or a viewed table that contains a column *C* that is an XML visible column for *U*.

A schema *S* of catalog *C* is a *visible schema* of *C* for authorization identifier *U* if *S* contains a table *T* that is a visible table for *U*. A schema *S* of catalog *C* is an *XML visible schema* of *C* for authorization identifier *U* if *S* contains a table *T* that is an XML visible table for *U*.

4.10.9 Mapping an SQL table to XML

Subclause 9.11, "Mapping an SQL table to XML and an XML Schema document", defines a mapping of an SQL table to one or both of two documents: an XML Schema document that describes the structure of the mapped XML and either an XML document or a sequence of XML elements. Only base tables and viewed tables may be the source of this mapping.

NOTE 13 — This part of this International Standard specifies no syntax for invoking the mapping specified in Subclause 9.11, "Mapping an SQL table to XML and an XML Schema document". This specification is intended to be used by applications and referenced by other standards. It is the responsibility of any such application or other standard to ensure that the correct number of arguments as well as a valid value for each argument are supplied for this mapping.

Only the XML visible columns of this table for the user that invokes this mapping will be represented in the generated XML.

This mapping allows the invoker to specify:

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

— Whether to map the table to a sequence of XML elements where the name of each top-level element is derived from the table name and represents a row in the table, or to map the table to an XML document with a single root element whose name is derived from the table name and to map each row to an element named **<row>**.

— The XML target namespace URI of the XML Schema and data to be mapped (if the XML target namespace URI is specified as a zero-length string, then no XML namespace is added).

— Whether to map null values to absent elements, or whether to map them to elements that are marked with **xsi:nil="true"**.

— Whether to map the table into XML data, to map the table into an XML Schema document, or both.

Some of the XML Schema type definitions and element declarations may contain annotations to represent SQL metadata that is not directly relevant to XML. It is implementation-defined whether these annotations are generated.

The rules of Subclause 9.11, "Mapping an SQL table to XML and an XML Schema document", are supported by the rules of Subclause 9.9, "Mapping an SQL table to XML Schema data types", and Subclause 9.10, "Mapping an SQL table to an XML element or a sequence of XML elements".

4.10.10 Mapping an SQL schema to XML

Subclause 9.14, "Mapping an SQL schema to an XML document and an XML Schema document", defines a mapping between the tables of an SQL-schema and either or both of two documents: an XML document that represents the data in these tables, and an XML Schema document that describes the first document.

NOTE 14 — This part of this International Standard specifies no syntax for invoking the mapping specified in Subclause 9.14, "Mapping an SQL schema to an XML document and an XML Schema document". This specification is intended to be used by applications and referenced by other standards. It is the responsibility of any such application or other standard to ensure that the correct number of arguments as well as a valid value for each argument are supplied for this mapping.

Only the XML visible tables of the schema for the user that invokes this mapping will be represented in these two XML documents. Only the XML visible columns of these tables for the user that invokes this mapping will be represented in these two XML documents.

This mapping allows the invoker to specify:

— Whether to map each table to a sequence of XML elements where the name of each top-level element is derived from the table name and represents a row in the table, or to map each table to a single element whose name is derived from the table name and to map each row to an element named **<row>**.

— The XML target namespace URI of the XML Schema and data to be mapped (if the XML target namespace URI is specified as a zero-length string, then no XML namespace is added).

— Whether to map null values to absent elements, or whether to map them to elements that are marked with **xsi:nil="true"**.

— Whether to map the schema into XML data, to map the schema into an XML Schema document, or both.

Some of the XML Schema type definitions and element declarations may contain annotations to represent SQL metadata that is not directly relevant to XML. It is implementation-defined whether these annotations are generated.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

The SQL-schema mapping assumes an implementation-dependent *repeatable ordering* when iterating over the XML visible tables in the SQL-schema. This allows generating the correct alignment of the table data with the element declarations in situations where the generated XML Schema uses a sequence content model instead of an all group content model.

The rules of Subclause 9.14, "Mapping an SQL schema to an XML document and an XML Schema document", are supported by the rules of Subclause 9.12, "Mapping an SQL schema to XML Schema data types", and Subclause 9.13, "Mapping an SQL schema to an XML element".

4.10.11 Mapping an SQL catalog to XML

Subclause 9.17, "Mapping an SQL catalog to an XML document and an XML Schema document", defines a mapping between the tables of an SQL catalog and either or both of two documents: an XML document that represents the data in these tables, and an XML Schema document that describes the first document.

> NOTE 15 — This part of this International Standard specifies no syntax for invoking the mapping specified in Subclause 9.17, "Mapping an SQL catalog to an XML document and an XML Schema document". This specification is intended to be used by applications and referenced by other standards. It is the responsibility of any such application or other standard to ensure that the correct number of arguments as well as a valid value for each argument are supplied for this mapping.

Only the XML visible schemas of this catalog for the user that invokes this mapping will be represented in these two XML documents. Only the XML visible tables of these schemas for the user that invokes this mapping will be represented in these two XML documents. Only the XML visible columns of these tables for the user that invokes this mapping will be represented in these two XML documents.

This mapping allows the user that invokes this mapping to specify:

— Whether to map each table to a sequence of XML elements where the name of each top-level element is derived from the table name and represents a row in the table, or to map each table to a single element whose name is derived from the table name and each row is mapped to an element names **`<row>`**.

— The XML target namespace URI of the XML Schema and data to be mapped (if the XML target namespace URI is specified as a zero-length string, then no XML namespace is added).

— Whether to map null values to absent elements, or whether to map them to elements that are marked with **`xsi:nil="true"`**.

— Whether to map the catalog into XML data, to map the catalog into an XML Schema document, or both.

Some of the XML Schema type definitions and element declarations may contain annotations to represent SQL metadata that is not directly relevant to XML. It is implementation-defined whether these annotations are generated.

The rules of Subclause 9.17, "Mapping an SQL catalog to an XML document and an XML Schema document", are supported by the rules of Subclause 9.15, "Mapping an SQL catalog to XML Schema data types", and Subclause 9.16, "Mapping an SQL catalog to an XML element".

© ISO/IEC 2008 – All rights reserved
--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

5 Lexical elements

This Clause modifies Clause 5, "Lexical elements", in ISO/IEC 9075-2.

5.1 <token> and <separator>

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

Function

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

Format

```
<non-reserved word> ::=
    !! All alternatives from ISO/IEC 9075-2
  | ABSENT | ACCORDING
  | BASE64 | BOM
  | COLUMNS | CONTENT
  | DOCUMENT
  | EMPTY | ENCODING
  | HEX
  | ID | INDENT
  | LOCATION
  | NAMESPACE | NIL
  | PASSING | PATH | PRESERVE
  | RETURNING
  | SEQUENCE | STANDALONE | STRIP
  | UNTYPED | URI
  | VALID | VERSION
  | WHITESPACE
  | XMLSCHEMA | XMLDECLARATION
```

© ISO/IEC 2008 – All rights reserved

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
<reserved word> ::=

    !! All alternatives from ISO/IEC 9075-2

  | XML | XMLAGG | XMLATTRIBUTES | XMLBINARY | XMLCAST
  | XMLCOMMENT | XMLCONCAT | XMLDOCUMENT | XMLELEMENT | XMLEXISTS | XMLFOREST
  | XMLITERATE | XMLNAMESPACES | XMLPARSE | XMLPI
  | XMLQUERY | XMLSERIALIZE | XMLTABLE | XMLTEXT | XMLVALIDATE
```

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

5.2 Names and identifiers

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

Function

Specify names.

Format

```
<registered XML Schema name> ::=
  <schema qualified name>
```

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

1) Insert this GR A <registered XML Schema name> identifies a registered XML Schema.

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

(Blank page)

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6 Scalar expressions

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

6.1 <data type>

This Subclause modifies Subclause 6.1, "<data type>", in ISO/IEC 9075-2.

Function

Specify a data type.

Format

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

<XML type> ::=
  XML [ <left paren> <XML type modifier> <right paren> ]

<XML type modifier> ::=
  <primary XML type modifier>
      [ <left paren> <secondary XML type modifier> <right paren> ]

<primary XML type modifier> ::=
    DOCUMENT
  | CONTENT
  | SEQUENCE

<secondary XML type modifier> ::=
    ANY
  | UNTYPED
  | XMLSCHEMA <XML valid according to what> [ <XML valid element clause> ]
```

Syntax Rules

1) Insert this SR XML specifies an XML data type.

2) Insert this SR If an <XML type> does not specify <XML type modifier>, then it is implementation-defined whether SEQUENCE, CONTENT(ANY), or CONTENT(UNTYPED) is implicit.

3) Insert this SR Case:

   a) If <primary XML type modifier> specifies SEQUENCE, then <secondary XML type modifier> shall not be specified.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

b) Otherwise, if <secondary XML type modifier> is not specified, then it is implementation-defined whether UNTYPED or ANY is implicit.

4) Insert this SR If <secondary XML type modifier> specifies XMLSCHEMA, then let *RXS* be the indicated registered XML Schema, let *ENSURI* be the indicated XML namespace, if any, and let ***GEDSC*** be the indicated global element declaration schema component, if any.

NOTE 16 — Indicated registered XML Schema, indicated XML namespace, and indicated global element declaration schema component are defined in Subclause 11.6, "<XML valid according to clause>".

Access Rules

No additional Access Rules.

General Rules

1) Insert after GR 20) If <data type> is an <XML type>, then an XML type descriptor is created, including the following:

a) The name of the data type (XML).

b) The primary XML type modifier described by the <primary XML type modifier> (DOCUMENT, CONTENT, or SEQUENCE).

c) The secondary XML type modifier described by the <secondary XML type modifier> (UNTYPED, ANY, or XMLSCHEMA), if any.

d) The registered XML Schema descriptor of the indicated registered XML Schema, if any.

e) The XML namespace URI of the indicated XML namespace, if any.

f) The XML NCName of the indicated global element declaration schema component, if any.

Conformance Rules

1) Insert this CR Without Feature X010, "XML type", conforming SQL language shall not contain an <XML type>.

2) Insert this CR Without Feature X011, "Arrays of XML type", conforming SQL language shall not contain an <array type> that is based on a <data type> that is either an XML type or a distinct type whose source type is an XML type.

3) Insert this CR Without Feature X012, "Multisets of XML type", conforming SQL language shall not contain a <multiset type> that is based on a <data type> that is either an XML type or a distinct type whose source type is an XML type.

4) Insert this CR Without Feature X181, "XML(DOCUMENT(UNTYPED)) type", conforming SQL language shall not contain an <XML type> whose <primary XML type modifier> is DOCUMENT and <secondary XML type modifier> is UNTYPED.

5) Insert this CR Without Feature X182, "XML(DOCUMENT(ANY)) type", conforming SQL language shall not contain an <XML type> whose <primary XML type modifier> is DOCUMENT and <secondary XML type modifier> is ANY.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6) Insert this CR Without Feature X231, “XML(CONTENT(UNTYPED)) type”, conforming SQL language shall not contain an <XML type> whose <primary XML type modifier> is CONTENT and <secondary XML type modifier> is UNTYPED.

7) Insert this CR Without Feature X232, “XML(CONTENT(ANY)) type”, conforming SQL language shall not contain an <XML type> whose <primary XML type modifier> is CONTENT and <secondary XML type modifier> is ANY.

8) Insert this CR Without Feature X191, “XML(DOCUMENT(XMLSCHEMA)) type”, conforming SQL language shall not contain an <XML type> whose <primary XML type modifier> is DOCUMENT and <secondary XML type modifier> specifies XMLSCHEMA.

9) Insert this CR Without Feature X192, “XML(CONTENT(XMLSCHEMA)) type”, conforming SQL language shall not contain an <XML type> whose <primary XML type modifier> is CONTENT and <secondary XML type modifier> specifies XMLSCHEMA.

10) Insert this CR Without Feature X260, “XML type: ELEMENT clause”, conforming SQL language shall not contain an <XML type> that contains <XML valid element clause>.

11) Insert this CR Without Feature X261, “XML type: NAMESPACE without ELEMENT clause”, conforming SQL language shall not contain an <XML type> that contains an <XML valid element clause> that does not contain an <XML valid element name specification>.

12) Insert this CR Without Feature X263, “XML type: NO NAMESPACE with ELEMENT clause”, conforming SQL language shall not contain an <XML type> that contains an <XML valid element namespace specification> that contains NO NAMESPACE.

13) Insert this CR Without Feature X264, “XML type: schema location”, conforming SQL language shall not contain an <XML type> that contains <XML valid schema location>.

14) Insert this CR Without Feature X190, “XML(SEQUENCE) type”, conforming SQL language shall not contain an <XML type> whose <XML type modifier> is SEQUENCE.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.2 <field definition>

This Subclause modifies Subclause 6.2, "<field definition>", in ISO/IEC 9075-2.

Function

Define a field of a row type.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

1) Insert this CR Without Feature X015, "Fields of XML type", conforming SQL language shall not contain a <field definition> that contains a <data type> that is based on either an XML type or a distinct type whose source type is an XML type.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.3 <value expression primary>

This Subclause modifies Subclause 6.3, "<value expression primary>", in ISO/IEC 9075-2.

Function

Specify a value that is syntactically self-delimited.

Format

```
<nonparenthesized value expression primary> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <XML cast specification>
```

Syntax Rules

1) Replace SR 1) The declared type of a <value expression primary> is the declared type of the simply contained <value expression>, <unsigned value specification>, <column reference>, <set function specification>, <window function>, <scalar subquery>, <case expression>, <cast specification>, <XML cast specification>, <field reference>, <subtype treatment>, <method invocation>, <static method invocation>, <new specification>, <attribute or method reference>, <reference resolution>, <collection value constructor>, <array element reference>, <multiset element reference>, or <next value expression>, or the effective returns type of the simply contained <routine invocation>.

Access Rules

No additional Access Rules.

General Rules

1) Replace GR 1) The value of a <value expression primary> is the value of the simply contained <value expression>, <unsigned value specification>, <column reference>, <set function specification>, <window function>, <scalar subquery>, <case expression>, <cast specification>, <XML cast specification>, <field reference>, <subtype treatment>, <method invocation>, <static method invocation>, <new specification>, <attribute or method reference>, <reference resolution>, <collection value constructor>, <array element reference>, <multiset element reference>, <next value expression>, or <routine invocation>.

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.4 <case expression>

This Subclause modifies Subclause 6.11, "<case expression>", in ISO/IEC 9075-2.

Function

Specify a conditional value.

Format

```
<when operand> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <XML content predicate part 2>
  | <XML document predicate part 2>
  | <XML valid predicate part 2>
```

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.5 <cast specification>

This Subclause modifies Subclause 6.12, "<cast specification>", in ISO/IEC 9075-2.

Function

Specify a data conversion.

Format

```
<cast specification> ::=
  CAST <left paren> <cast operand> AS <cast target>
      [ <XML passing mechanism> ]
      <right paren>
```

Syntax Rules

1) Insert after SR 3) If <XML passing mechanism> is specified, then <cast operand> shall be a <value expression> and both *SD* and *TD* shall be XML types.

2) Insert after SR 3) If *SD* and *TD* are both XML types and <XML passing mechanism> is not specified, then it is implementation-defined whether BY REF or BY VALUE is implicit.

3) Replace SR 6) 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. "Y" indicates that the combination is syntactically valid without restriction; "M" indicates that the combination is valid subject to other Syntax Rules in this Subclause being satisfied; and "N" indicates that the combination is not valid:

SD \ *TD*	EN	AN	C	D	T	TS	YM	DT	BO	UDT	B	RT	CT	RW	XML
EN	Y	Y	Y	N	N	N	M	M	N	M	N	M	N	N	N
AN	Y	Y	Y	N	N	N	N	N	N	M	N	M	N	N	N
C	Y	Y	Y	Y	Y	Y	Y	Y	Y	M	N	M	N	N	N
D	N	N	Y	Y	N	Y	N	N	N	M	N	M	N	N	N
T	N	N	Y	N	Y	Y	N	N	N	M	N	M	N	N	N
TS	N	N	Y	Y	Y	Y	N	N	N	M	N	M	N	N	N
YM	M	N	Y	N	N	N	Y	N	N	M	N	M	N	N	N
DT	M	N	Y	N	N	N	N	Y	N	M	N	M	N	N	N
BO	N	N	Y	N	N	N	N	N	Y	M	N	M	N	N	N
UDT	M	M	M	M	M	M	M	M	M	M	M	M	N	N	N
B	N	N	N	N	N	N	N	N	N	M	Y	M	N	N	N
RT	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	M	N	N
RW	N	N	N	N	N	N	N	N	N	N	N	N	N	M	N
XML	N	N	N	N	N	N	N	N	N	N	N	N	N	N	M

Where:
EN = Exact Numeric
AN = Approximate Numeric
C = Character (Fixed- or Variable-Length, or Character Large Object)
D = Date
T = Time
TS = Timestamp

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
 YM  = Year-Month Interval
 DT  = Day-Time Interval
 BO  = Boolean
UDT  = User-Defined Type
 B   = Binary (Fixed- or Variable-Length or Binary Large Object)
 RT  = Reference type
 CT  = Collection type
 RW  = Row type
XML  = XML type
```

4) Insert before SR 15) If BY REF is specified or implied, then

Case:

a) If *TD* is either XML(DOCUMENT(UNTYPED)) or XML(CONTENT(UNTYPED)), then *SD* shall be either XML(DOCUMENT(UNTYPED)) or XML(CONTENT(UNTYPED)).

b) If *TD* is either XML(DOCUMENT(XMLSCHEMA)) or XML(CONTENT(XMLSCHEMA)), then

Case:

i) If the type descriptor of *TD* includes an XML namespace URI ***N*** and an XML NCName ***EN*** of a global element declaration schema component, then *SD* shall be either XML(DOCUMENT(XMLSCHEMA)) or XML(CONTENT(XMLSCHEMA)) such that the XML Schema identified by the registered XML Schema descriptor included in the type descriptor of *TD* is identical to the XML Schema identified by the registered XML Schema descriptor included in the type descriptor of *SD* and the type descriptor of *SD* includes an XML namespace URI that is identical to ***N***, as defined by [Namespaces], and an XML NCName that is equivalent to ***EN***.

ii) If the type descriptor of *TD* includes an XML namespace URI ***N***, then *SD* shall be either XML(DOCUMENT(XMLSCHEMA)) or XML(CONTENT(XMLSCHEMA)) such that the XML Schema identified by the registered XML Schema descriptor included in the type descriptor of *TD* is identical to the XML Schema identified by the registered XML Schema descriptor included in the type descriptor of *SD* and the type descriptor of *SD* includes an XML namespace URI that is identical to ***N***, as defined by [Namespaces].

iii) Otherwise, *SD* shall be either XML(DOCUMENT(XMLSCHEMA)) or XML(CONTENT(XMLSCHEMA)) such that the XML Schema identified by the registered XML Schema descriptor included in the type descriptor of *TD* is identical to the XML Schema identified by the registered XML Schema descriptor included in the type descriptor of *SD*.

Access Rules

No additional Access Rules.

General Rules

1) Insert before GR 3) If *SD* and *TD* are both XML types, then:

a) Case:

i) If *TD* is XML(DOCUMENT(UNTYPED)), XML(DOCUMENT(ANY)), or XML(DOCUMENT(XMLSCHEMA)), and *SV* is not an XQuery sequence of length 1 (one) whose sole XQuery item is an XQuery document node whose **children** property contains exactly one XQuery

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

element node, zero or more XQuery comment nodes, zero or more XQuery processing instruction nodes, then an exception condition is raised: *data exception — not an XML document*.

ii) If *TD* is XML(CONTENT(UNTYPED)), XML(CONTENT(ANY)), or XML(CONTENT(XMLSCHEMA)), and *SV* is not an XQuery sequence of length 1 (one) whose sole XQuery item is an XQuery document node, then an exception condition is raised: *data exception — not an XQuery document node*.

b) Case:

i) If BY VALUE is specified or implied, then

Case:

1) If *TD* is either XML(DOCUMENT(XMLSCHEMA)) or XML(CONTENT(XMLSCHEMA)), then let *S* be the <secondary XML type modifier> contained in <cast target>.

A) Case:

I) If *TD* is XML(DOCUMENT(XMLSCHEMA)), then let *DCS* be DOCUMENT.

II) Otherwise, let *DCS* be CONTENT.

B) *TV* is the result of

```
XMLVALIDATE( DCS VE ACCORDING TO S )
```

2) If *TD* is either XML(DOCUMENT(ANY)), XML(CONTENT(ANY)), or XML(SEQUENCE), then *TV* is the result of applying the General Rules of Subclause 10.18, "Constructing a copy of an XML value", with *SV* as *VALUE*.

3) Otherwise, *TV* is the result of applying the General Rules of Subclause 10.19, "Constructing an unvalidated XQuery document node", with *SV* as *XQuery document node*.

ii) Otherwise, *TV* is *SV*.

Conformance Rules

No additional Conformance Rules.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

6.6 <XML cast specification>

Function

Specify a data conversion whose source or target type is an XML type.

Format

```
<XML cast specification> ::=
  XMLCAST <left paren> <XML cast operand> AS
      <XML cast target> [ <XML passing mechanism> ] <right paren>

<XML cast operand> ::=
    <value expression>
  | <implicitly typed value specification>

<XML cast target> ::=
    <domain name>
  | <data type>
```

Syntax Rules

1) Case:

   a) If the <XML cast specification> is contained within the scope of an <XML binary encoding>, then let *XBE* be the <XML binary encoding> with innermost scope that contains the <XML cast specification>.

   b) Otherwise, let *XBE* be an implementation-defined <XML binary encoding>.

2) Case:

   a) If *XBE* specifies BASE64, then let *ENC* be an indication that binary strings are to be encoded in base64.

   b) Otherwise, let *ENC* be an indication that binary strings are to be encoded in hex.

3) Case:

   a) If <XML cast target> is <domain name>, then let *TD* be the data type of the specified domain.

   b) Otherwise, let *TD* be the data type identified by <data type>. <data type> shall not contain a <collate clause>.

4) At least one of the following shall be true:

   a) *TD* is an XML type.

   b) The <XML cast operand> is a <value expression> whose declared type is an XML type.

5) *TD* shall not be a collection type, a row type, a structured type or a reference type.

6) The declared type of the result of the <XML cast specification> is *TD*.

7) If *TD* is XML(DOCUMENT(UNTYPED)), XML(DOCUMENT(ANY)), XML(DOCUMENT(XMLSCHEMA)), or XML(CONTENT(XMLSCHEMA)), then the <XML cast operand> shall be either NULL or a <value expression> whose declared type is an XML type.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8) If the <XML cast operand> is a <value expression> *VE*, then let *SD* be the declared type of *VE*.

9) If <XML passing mechanism> is specified, then:

   a) <XML cast operand> shall be a <value expression>.

   b) *SD* and *TD* shall both be XML types.

10) If the <XML cast operand> is a <value expression> *VE*, then

    Case:

    a) If *VE* simply contains a <dynamic parameter specification> *DPS*, then the <XML cast specification> is equivalent to

       ```
       CAST ( DPS AS TD )
       ```

    b) Otherwise:

       i) *SD* shall not be a collection type, a row type, a structured type or a reference type.

       ii) If *SD* and *TD* are both XML types, then

           1) If <XML passing mechanism> is not specified, then it is implementation-defined whether BY REF or BY VALUE is implicit.

           2) Let *XPM* be the implicit or explicit <XML passing mechanism>.

           3) The <XML cast specification> is equivalent to

              ```
              CAST ( VE AS TD XPM )
              ```

11) If <XML cast operand> is an <implicitly typed value specification> *ITVS*, then the <XML cast specification> is equivalent to

    ```
    CAST ( ITVS AS TD )
    ```

12) If *TD* is character string type, then the declared type collation of the <XML cast specification> is the character set collation of the character set of *TD* and its collation derivation is *implicit*.

13) If <XML cast target> is <domain name>, then let *D* be the domain identified by the <domain name>. The schema identified by the explicit or implicit qualifier of the <domain name> shall include the descriptor of *D*.

14) Case:

    a) If the <XML cast target> is a <domain name>, then let *SQLT* be the <data type> of the identified domain.

    b) If the <XML cast target> identifies a distinct type, then let *SQLT* be the source type of the distinct type.

    c) Otherwise, let *SQLT* be <data type>.

15) Case:

    a) If *SQLT* is a character string type, then let *XT* be **`xs:string`**.

    b) If *SQLT* is a binary string type, then

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Case:

i) If *XBE* specifies BASE64, then let *XT* be **xs:base64Binary**.

ii) Otherwise, let *XT* be **xs:hexBinary**.

c) If *SQLT* is an exact numeric type, then

Case:

i) If the type designator of *SQLT* is DECIMAL or NUMERIC, then let *XT* be **xs:decimal**.

ii) If the type designator of *SQLT* is SMALLINT, INTEGER, or BIGINT, then let *XT* be **xs:integer**.

d) If *SQLT* is an approximate numeric type, then let *XT* be **xs:double**.

e) If *SQLT* is a datetime type, then

Case:

i) If the type designator of *SQLT* is DATE, then let *XT* be **xs:date**.

ii) If the type designator of *SQLT* is TIME WITH TIME ZONE, then let *XT* be **xs:time**.

iii) If the type designator of *SQLT* is TIME WITHOUT TIME ZONE, then let *XT* be **xs:time**.

iv) If the type designator of *SQLT* is TIMESTAMP WITH TIME ZONE, then let *XT* be **xs:dateTime**.

v) If the type designator of *SQLT* is TIMESTAMP WITHOUT TIME ZONE, then let *XT* be **xs:dateTime**.

f) Otherwise, let *XT* be the XML Schema type obtained by applying the General Rules of Subclause 9.5, "Mapping SQL data types to XML Schema data types", with *SQLT* as *SQLTYPE*, *ENC* as *ENCODING*, and "absent" as *NULLS*.

Access Rules

1) If *TD* is a distinct type, then let *STD* be the source type of *TD*, and let *AVE* be an arbitrary <value expression> of declared type *STD*. The Access Rules of Subclause 6.5, "<cast specification>", are applied to

```
CAST ( AVE AS TD )
```

2) If *SD* is a distinct type, then let *SSD* be the source type of *SD*. The Access Rules of Subclause 6.5, "<cast specification>", are applied to

```
CAST ( VE AS SSD )
```

3) If <XML cast target> is a <domain name>, then

Case:

a) If the <XML cast specification> is contained, without an intervening <SQL routine spec> that specifies SQL SECURITY INVOKER, then the applicable privileges of the <authorization identifier> that owns the containing SQL-schema shall include USAGE on the domain identified by the <domain name>.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

b) Otherwise, the current privileges shall include USAGE on the domain identified by the <domain name>.

General Rules

1) Let *V* be the value of the <XML cast operand> simply contained in the <XML cast specification>.

2) If *V* is the null value, then the result of the <XML cast specification> is the null value, and no further General Rules of this Subclause are applied.

3) If *TD* is an XML type, then:

   a) Let *XMLV* be a <character string literal> whose value is the result of evaluating the General Rules of Subclause 9.8, "Mapping values of SQL data types to values of XML Schema data types", to *V* as *the value of an SQL data type*, *ENC* as *ENCODING*, "absent" as *NULLS*, and *True* as *CHARMAPPING*.

   b) Let *TEMPV* be result of

```
XMLPARSE (CONTENT XMLV PRESERVE WHITESPACE)
```

   c) If *TD* is XML(CONTENT(UNTYPED)) or XML(CONTENT(ANY)), then let *TV* be *TEMPV*.

   d) If *TD* is XML(SEQUENCE), then:

      i) Let ***XST*** be the XML Schema type obtained by applying the General Rules of Subclause 9.5, "Mapping SQL data types to XML Schema data types", with *SD* as *SQLTYPE*, *ENC* as *ENCODING*, and "absent" as *NULLS*.

      ii) Case:

         1) If *SD* is a year-month interval type, then let ***XSBT*** be the XQuery simple type **`xs:yearMonthDuration`**.

         2) If *SD* is a day-time interval type, then let ***XSBT*** be the XQuery simple type **`xs:dayTimeDuration`**.

         3) If ***XST*** is an XML Schema built-in type, then let ***XSBT*** be *XST*.

         4) If ***XST*** an XML Schema atomic type, then let ***XSBT*** be the XML Schema built-in type from which ***XST*** is derived.

      iii) Let ***XSBTN*** be an XML 1.1 QName for ***XSBT***.

      iv) Let ***XSC*** be an XQuery static context created according to the Syntax Rules of Subclause 10.20, "Creation of an XQuery expression context", with the *BNF* non-terminal argument omitted. Let ***XDC*** be an XQuery dynamic context created according to the General Rules of Subclause 10.20, "Creation of an XQuery expression context".

      v) Let ***XSC*** and ***XDC*** be augmented with an XQuery variable **`$TEMP`**, whose XQuery formal type notation is "**`document { text ? }`**", and whose value is *TEMPV*.

      vi) Case:

         1) If the SQL-implementation supports Feature X211, "XML 1.1 support", then let *TV* be the result of the XQuery evaluation with XML 1.1 lexical rules, using ***XSC*** and ***XDC*** as the XQuery expression context, of the XQuery expression

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
$TEMP cast as XSBTN
```

If this XQuery evaluation raises an XQuery error, then an exception condition is raised: *XQuery error*.

2) Otherwise, let *TV* be the result of the XQuery evaluation with XML 1.0 lexical rules, using ***XSC*** and ***XDC*** as the XQuery expression context, of the XQuery expression

```
$TEMP cast as XSBTN
```

If this XQuery evaluation raises an XQuery error, then an exception condition is raised: *XQuery error*.

e) *TV* is the result of the <XML cast specification>.

4) If *SD* is an XML type and *TD* is not an XML type, then:

a) Let ***XV*** be the result of applying the General Rules of Subclause 10.17, "Removing XQuery document nodes from an XQuery sequence", with ***V*** as the *SEQUENCE*.

b) Let ***AV*** be the result of applying the XQuery function **`fn:data()`** to ***XV***.

c) If ***AV*** is the empty sequence, then the result is the null value and no further General Rules of this Subclause are applied.

d) Let ***XMLT*** be an XML 1.1 QName for ***XT***.

NOTE 17 — ***XMLT*** may be in one of the built-in namespaces denoted by the prefix **`xs`**, or it may be in an implementation-dependent namespace, not necessarily available to the user.

e) Let ***XSC*** be an XQuery static context created according to the Syntax Rules of Subclause 10.20, "Creation of an XQuery expression context", with the *BNF* non-terminal argument omitted.

f) Let ***XDC*** be an XQuery dynamic context created according to the General Rules of Subclause 10.20, "Creation of an XQuery expression context".

g) Let ***XSC*** and ***XDC*** be augmented with an XQuery variable **`$TEMP`** whose XQuery formal type notation is "**`xs:anyAtomicType`**" and whose value is ***AV***.

h) Let ***BV*** be the result of the XQuery evaluation with XML 1.1 lexical rules, using ***XSC*** and ***XDC*** as the XQuery expression context, of the XQuery expression

Case:

i) If *SQLT* is TIMESTAMP WITHOUT TIME ZONE, then

```
fn:adjust-dateTime-to-timezone(
  fn:adjust-dateTime-to-timezone( $TEMP cast as xs:dateTime,
                                  xs:dayTimeDuration("PT0H")
                                ), () ) cast as XMLT
```

ii) If *SQLT* is TIME WITHOUT TIME ZONE, then

```
fn:adjust-time-to-timezone(
  fn:adjust-time-to-timezone( $TEMP cast as xs:time,
                              xs:dayTimeDuration("PT0H")
                            ), () ) cast as XMLT
```


© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

iii) If *SQLT* is DATE, then

```
fn:adjust-date-to-timezone(
  fn:adjust-date-to-timezone( $TEMP cast as xs:date,
                              xs:dayTimeDuration("PT0H")
                            ), () ) cast as XML
```

iv) Otherwise,

```
$TEMP cast as XML
```

If this XQuery evaluation raises an XQuery error, then an exception condition is raised: *XQuery error*.

i) ***BV*** is an XQuery atomic value.

NOTE 18 — The following rules are based on the fact that the value space of XQuery atomic types is the same as the value space of corresponding SQL types. That is, it is possible to treat ***BV*** as a value of an SQL type. For example, if ***BV*** is of type **xs:integer**, then ***BV*** is a (mathematical) integer, and, as such, it may also be the value of an exact numeric type with scale 0 (zero). See Subclause 4.10.7, "Mapping XQuery atomic values to SQL values".

Case:

i) If ***XT*** is **xs:string** or derived from **xs:string**, then let *A* be an arbitrary <literal> of character string type whose character repertoire is Unicode and whose value is ***BV***.

ii) If ***XT*** is **xs:hexBinary** or derived from **xs:hexBinary**, then let *A* be an arbitrary <literal> of binary string type whose value is ***BV*** decoded as hex.

iii) If ***XT*** is **xs:base64Binary**, or derived from **xs:base64Binary**, then let *A* be an arbitrary <literal> of binary string type whose value is ***BV*** decoded as base64.

iv) If ***XT*** is **xs:decimal** or derived from **xs:decimal**, then let *A* be an arbitrary <literal> of exact numeric type whose value is ***BV***.

v) If ***XT*** is **xs:float** or **xs:double**, or derived from **xs:float** or **xs:double**, then:

1) If ***AV*** is `INF`, `-INF`, or `NaN`, then an exception condition is raised: *data exception — numeric value out of range*.

2) Let *A* be an arbitrary <literal> of approximate numeric type whose value is ***BV***.

vi) If ***XT*** is **xs:date** or derived from **xs:date**, then:

1) If the XQuery datetime normalized value component of ***AV*** is not positive, then an exception condition is raised: *data exception — invalid datetime format*.

2) Let *A* be an arbitrary <literal> of declared type *SQLT* whose value is ***BV*** .

vii) If ***XT*** is **xs:dateTime** or derived from **xs:dateTime**, then:

1) If the XQuery datetime normalized value component of ***AV*** is not positive, then an exception condition is raised: *data exception — invalid datetime format*.

2) If *SQLT* is TIMESTAMP WITHOUT TIME ZONE, then let *A* be an arbitrary <literal> of declared type TIMESTAMP WITHOUT TIME ZONE whose value is ***BV***.

3) If *SQLT* is TIMESTAMP WITH TIME ZONE, then:

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

A) If the XQuery datetime timezone component of ***AV*** is the XQuery empty sequence, then an exception condition is raised: *data exception — invalid datetime format*.

B) Let *A* be an arbitrary <literal> of declared type TIMESTAMP WITH TIME ZONE whose value is ***BV***.

viii) If ***XT*** is **xs:time** or derived from **xs:time**, then

Case:

1) If *SQLT* is TIME WITHOUT TIME ZONE, then let *A* be an arbitrary <literal> of declared type TIME WITHOUT TIME ZONE whose value is the XQuery datetime normalized value of ***BV***.

2) If *SQLT* is TIME WITH TIME ZONE, then:

A) If the XQuery datetime timezone component of ***AV*** is the XQuery empty sequence, then an exception condition is raised: *data exception — invalid datetime format*.

B) Let *A* be an arbitrary <literal> of declared type TIME WITH TIME ZONE whose value is ***BV*** (more precisely, whose UTC component is the XQuery datetime normalized value of ***BV*** and whose timezone component is the XQuery datetime timezone component of ***BV***).

ix) If ***XT*** is **xs:yearMonthDuration** or derived from **xs:yearMonthDuration**, then let *A* be an arbitrary <literal> of year-month interval type whose value is ***BV***.

x) If ***XT*** is **xs:dayTimeDuration** or derived from **xs:dayTimeDuration**, then let *A* be an arbitrary <literal> of day-time interval type whose value is ***BV***.

xi) If ***XT*** is **xs:boolean** or derived from **xs:boolean**, then let *A* be an arbitrary <literal> of declared type BOOLEAN whose value is ***BV***.

j) The result of the <XML cast specification> is the result of

```
CAST (A AS SQLT)
```

NOTE 19 — This may raise may raise a warning or an exception condition if the value of *A* does not conform to constraints of *SQLT*. It may also perform rounding or truncation, and so forth, in order to produce a value of type *SQLT*.

Conformance Rules

1) Without Feature X025, "XMLCast", conforming SQL language shall not contain an <XML cast specification>.

2) Without Feature X026, "XMLCast: SQL cast semantics", *XT* is unconditionally the XML Schema type obtained by applying the General Rules of Subclause 9.5, "Mapping SQL data types to XML Schema data types", with *SQLT* as *SQLTYPE*, *ENC* as *ENCODING*, and "absent" as *NULLS*.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.7 <value expression>

This Subclause modifies Subclause 6.25, "<value expression>", in ISO/IEC 9075-2.

Function

Specify a value.

Format

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

Syntax Rules

1) Replace SR 2) The declared type of a <common value expression> is the declared type of the <numeric value expression>, <string value expression>, <datetime value expression>, <interval value expression>, <user-defined type value expression>, <collection value expression>, <reference value expression>, or <XML value expression>, respectively.

2) Insert after SR 7)c)ii) A <cast specification> whose result type is an XML type and whose <cast operand> has a declared type that is an XML type and whose <XML passing mechanism> specifies or implies BY VALUE.

3) Insert after SR 7)n) An <aggregate function> that specifies <XML aggregate>.

4) Insert after SR 7)t) An <XML valid predicate> *XVP* that satisfies one of the following:

   a) *XVP* does not specify <XML valid according to clause>.

   b) *XVP* specifies an <XML valid according to clause> that identifies a non-deterministic registered XML Schema and *XVP* does not specify an <XML valid element name specification> of an <XML valid element namespace specification>.

   c) *XVP* specifies an <XML valid according to clause> that identifies a non-deterministic XML namespace and *XVP* does not specify an <XML valid element name specification>.

   d) *XVP* specifies an <XML valid element clause> that identifies a non-deterministic global element declaration schema component of a registered XML Schema.

   NOTE 20 — This implies that an <XML valid predicate> that is used in a <check constraint definition> or an <assertion definition> shall contain an <XML valid according to clause> that either identifies a deterministic registered XML Schema, or contains an <XML valid element namespace specification> that identifies a deterministic XML namespace of a registered XML Schema, or contains an <XML valid element name specification> that identifies a deterministic global element declaration schema component of a registered XML Schema.

5) Insert after SR 7)t) An <XML value function> other than <XML query> and <XML concatenation>.

6) Insert after SR 7)t) An <XML query> that does not conform to implementation-defined rules enabling the SQL-implementation to deduce that the result of the <XML query> is deterministic.

7) Insert after SR 7)t) An <XML concatenation> that does not implicitly or explicitly specify RETURNING SEQUENCE.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8) Insert after SR 7)t) An <XML character string serialization> or an <XML binary string serialization>.

9) Insert after SR 7)t) An <XML exists predicate> that does not conform to implementation-defined rules enabling the SQL-implementation to deduce that the result of the <XML exists predicate> is deterministic.

10) Insert after SR 7)t) An <XML cast specification> whose <XML cast target> is an XML type whose <XML cast operand> has a declared type that is an XML type, and whose <XML passing mechanism> specifies or implies BY VALUE.

11) Insert after SR 7)t) An <XML cast specification> whose <XML cast target> is either XML(CONTENT(ANY)) or XML(CONTENT(UNTYPED)) type and whose <XML cast operand> has a declared type that is not an XML type.

Access Rules

No additional Access Rules.

General Rules

1) Replace GR 2) The value of a <common value expression> is the value of the immediately contained <numeric value expression>, <string value expression>, <datetime value expression>, <interval value expression>, <user-defined type value expression>, <collection value expression>, <reference value expression>, or <XML value expression>.

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.8 <string value function>

This Subclause modifies Subclause 6.29, "<string value function>", in ISO/IEC 9075-2.

Function

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

Format

```
<character value function> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <XML character string serialization>

<XML character string serialization> ::=
  XMLSERIALIZE <left paren> [ <document or content> ]
      <XML value expression> AS <data type>
      [ <XML serialize bom> ]
      [ <XML serialize version> ]
      [ <XML declaration option> ]
      [ <XML serialize indent> ]
      <right paren>

<XML serialize version> ::=
  VERSION <character string literal>

<XML serialize bom> ::=
  WITH [ NO ] BOM

<XML declaration option> ::=
    INCLUDING XMLDECLARATION
  | EXCLUDING XMLDECLARATION

<XML serialize indent> ::=
  [ NO ] INDENT

<document or content> ::=
    DOCUMENT
  | CONTENT

<blob value function> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <XML binary string serialization>

<XML binary string serialization> ::=
  XMLSERIALIZE <left paren> [ <document or content> ]
      <XML value expression> AS <data type>
      [ ENCODING <XML encoding specification> ]
      [ <XML serialize bom> ]
      [ <XML serialize version> ]
      [ <XML declaration option> ]
      [ <XML serialize indent> ]
      <right paren>

<XML encoding specification> ::=
```

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
  <XML encoding name>

<XML encoding name> ::=
  <SQL language identifier>
```

Syntax Rules

1) Replace SR 2) The declared type of <character value function> is the declared type of the immediately contained <character substring function>, <regular expression substring function>, <regex substring function>, <fold>, <transcoding>, <character transliteration>, <regex transliteration>, <trim function>, <character overlay function>, <normalize function>, <specific type method>, or <XML character string serialization>.

2) Insert this SR If <XML character string serialization> is specified, then:

   a) If <document or content> is not specified, then a <document or content> that specifies CONTENT is implicit.

   b) The <data type> shall be a character string type. The declared type of <XML character string serialization> is <data type>.

   c) The <character string literal> immediately contained in <XML serialize version> shall be '1.0' or '1.1', or it shall identify some successor to [XML 1.0] and [XML 1.1].

   d) If <XML serialize version> is not specified, then an implementation-defined <XML serialize version> is implicit.

3) Replace SR 17) The declared type of <blob value function> is the declared type of the immediately contained <blob substring function>, <blob trim function>, <blob overlay function>, or <XML binary string serialization>.

4) Insert this SR If <XML binary string serialization> is specified, then:

   a) If <document or content> is not specified, then a <document or content> that specifies CONTENT is implicit.

   b) The <data type> shall be a binary string type.

   c) The declared type of <XML binary string serialization> is <data type>.

   d) If <XML encoding specification> is not specified, then an implementation-defined <XML encoding name> is implicit.

   e) The supported <XML encoding name>s are implementation-defined.

   f) The <character string literal> immediately contained in <XML serialize version> shall be '1.0' or '1.1', or it shall identify some successor to [XML 1.0] and [XML 1.1].

   g) If <XML serialize version> is not specified, then an implementation-defined <XML serialize version> is implicit.

Access Rules

No additional Access Rules.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

General Rules

1) Replace GR 2) The result of <character value function> is the result of the immediately contained <character substring function>, <regular expression substring function>, <regex substring function>, <fold>, <transcoding>, <character transliteration>, <regex transliteration>, <trim function>, <character overlay function>, <normalize function>, <specific type method>, or <XML character string serialization>.

2) Insert this GR If <XML character string serialization> is specified, then

   a) Let *DCS* be the explicit or implicit <document or content>.

   b) Let *XMLV* be the value of the <XML value expression>.

   c) Let *DT* be the <data type>.

   d) Let *CS* be the character set of *DT*.

   e) Case:

      i) If <XML serialize bom> is specified, then

         Case:

         1) If WITH BOM is specified, then let *B* be *True*.

         2) Otherwise, let *B* be *False*.

      ii) Otherwise, let *B* be *Unknown*.

   f) Let *VER* be the <string value expression> simply contained in the explicit or implicit <XML serialize version>.

   g) Case:

      i) If <XML declaration option> is specified and specifies INCLUDING XMLDECLARATION, then let *DECL* be *True*.

      ii) If <XML declaration option> is specified and specifies EXCLUDING XMLDECLARATION, then let *DECL* be *False*.

      iii) Otherwise, let *DECL* be *Unknown*.

   h) Case:

      i) If <XML serialize indent> is specified and does not contain NO, then let *IND* be *True*.

      ii) Otherwise, let *IND* be *False*.

   i) The result of <XML character string serialization> is the result determined by applying the General Rules of Subclause 10.15, "Serialization of an XML value", with *DCS* as *SYNTAX*, *XMLV* as *VALUE*, *DT* as *TYPE*, *B* as *BOMDIA*, *CS* as *ENCODING*, *VER* as *VERSION*, *DECL* as *XMLDECLARATION*, and *IND* as *INDENT*.

3) Replace GR 13) The result of <blob value function> is the result of the simply contained <blob substring function>, <blob trim function>, <blob overlay function>, or <XML binary string serialization>.

4) Insert this GR If <XML binary string serialization> is specified, then:

   a) Let *DCS* be the explicit or implicit <document or content>.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

b) Let *XMLV* be the value of the <XML value expression>.

c) Let *DT* be the <data type>.

d) Let *XEN* be the <XML encoding name>.

e) Case:

   i) If <XML serialize bom> is specified, then

      Case:

      1) If WITH BOM is specified, then let *B* be *True*.

      2) Otherwise, let *B* be *False*.

   ii) Otherwise, let *B* be *Unknown*.

f) Let *VER* be the <string value expression> simply contained in the explicit or implicit <XML serialize version>.

g) Case:

   i) If <XML declaration option> is specified and specifies INCLUDING XMLDECLARATION, then let *DECL* be *True*.

   ii) If <XML declaration option> is specified and specifies EXCLUDING XMLDECLARATION, then let *DECL* be *False*.

   iii) Otherwise, let *DECL* be *Unknown*.

h) Case:

   i) If <XML serialize indent> is specified and does not contain NO, then let *IND* be *True*.

   ii) Otherwise, let *IND* be *False*.

i) The result of <XML binary string serialization> is the result determined by applying the General Rules of Subclause 10.15, "Serialization of an XML value", with *DCS* as *SYNTAX*, *XMLV* as *VALUE*, *DT* as *TYPE*, *B* as *BOMDIA*, *XEN* as *ENCODING*, *VER* as *VERSION*, *DECL* as *XMLDECLARATION*, and *IND* as *INDENT*.

Conformance Rules

1) Insert this CR Without Feature X070, "XMLSerialize: Character string serialization and CONTENT option", conforming SQL language shall not contain an <XML character string serialization> that immediately contains a <document or content> that is CONTENT.

2) Insert this CR Without Feature X071, "XMLSerialize: Character string serialization and DOCUMENT option", conforming SQL language shall not contain an <XML character string serialization> that immediately contains a <document or content> that is DOCUMENT.

3) Insert this CR Without Feature X072, "XMLSerialize: character string serialization", conforming SQL language shall not contain an <XML character string serialization>.

© ISO/IEC 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4) Insert this CR Without Feature X073, “XMLSerialize: BLOB serialization and CONTENT option”, conforming SQL language shall not contain an <XML binary string serialization> that immediately contains a <document or content> that is CONTENT.

5) Insert this CR Without Feature X074, “XMLSerialize: BLOB serialization and DOCUMENT option”, conforming SQL language shall not contain an <XML binary string serialization> that immediately contains a <document or content> that is DOCUMENT.

6) Insert this CR Without Feature X075, “XMLSerialize: BLOB serialization”, conforming SQL language shall not contain an <XML binary string serialization>.

7) Insert this CR Without Feature X076, “XMLSerialize: VERSION”, in conforming SQL language, <XML character string serialization> shall not contain VERSION.

8) Insert this CR Without Feature X076, “XMLSerialize: VERSION”, in conforming SQL language, <XML binary string serialization> shall not contain VERSION.

9) Insert this CR Without Feature X077, “XMLSerialize: explicit ENCODING option”, conforming SQL language shall not contain an <XML binary string serialization> that contains ENCODING.

10) Insert this CR Without Feature X078, “XMLSerialize: explicit XML declaration”, in conforming SQL language, <XML character string serialization> shall not contain XMLDECLARATION.

11) Insert this CR Without Feature X078, “XMLSerialize: explicit XML declaration”, in conforming SQL language, <XML binary string serialization> shall not contain XMLDECLARATION.

12) Insert this CR Without Feature X068, “XMLSerialize: BOM”, in conforming SQL language, <XML serialize bom> shall not be specified.

13) Insert this CR Without Feature X069, “XMLSerialize: INDENT”, in conforming SQL language, <XML serialize indent> shall not be specified.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.9 <XML value expression>

Function

Specify an XML value.

Format

```
<XML value expression> ::=
  <XML primary>

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

Syntax Rules

1) The declared type of the <value expression primary> immediately contained in <XML primary> shall be an XML type.

2) The declared type of <XML value expression> is the declared type of the simply contained <value expression primary> or <XML value function>.

Access Rules

None.

General Rules

1) The value of <XML value expression> is the value of the simply contained <value expression primary> or <XML value function>.

Conformance Rules

1) Without Feature X010, "XML type", conforming SQL language shall not contain an <XML value expression>.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.10 <XML value function>

Function

Specify a function that yields a value of type XML.

Format

```
<XML value function> ::=
    <XML comment>
  | <XML concatenation>
  | <XML document>
  | <XML element>
  | <XML forest>
  | <XML parse>
  | <XML PI>
  | <XML query>
  | <XML text>
  | <XML validate>
```

Syntax Rules

1) The declared type of <XML value function> is the declared type of the simply contained <XML comment>, <XML concatenation>, <XML document>, <XML element>, <XML forest>, <XML parse>, <XML PI>, <XML query>, <XML text>, or <XML validate>.

Access Rules

None.

General Rules

1) The result of an <XML value function> is the XML value of the immediately contained <XML comment>, <XML concatenation>, <XML document>, <XML element>, <XML forest>, <XML parse>, <XML PI>, <XML query>, <XML text>, or <XML validate>.

Conformance Rules

1) Without Feature X010, "XML type", conforming SQL language shall not contain an <XML value function>.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.11 <XML comment>

Function

Generate an XML value with a single XQuery comment node, possibly as a child of an XQuery document node.

Format

```
<XML comment> ::=
  XMLCOMMENT <left paren> <character value expression>
      [ <XML returning clause> ] <right paren>
```

Syntax Rules

1) If <XML returning clause> is not specified, then it is implementation-defined whether RETURNING CONTENT or RETURNING SEQUENCE is implicit.

2) Let *SVE* be the <character value expression>.

3) Case:

   a) If RETURNING CONTENT is specified or implied, then <XML comment> is equivalent to

      ```
      XMLDOCUMENT ( XMLCOMMENT ( SVE RETURNING SEQUENCE ) RETURNING CONTENT )
      ```

   b) Otherwise, the declared type of <XML comment> is XML(SEQUENCE).

      NOTE 21 — If RETURNING CONTENT is specified or implied, then the declared type of <XML comment> is effectively established by the Syntax Rules of Subclause 6.13, "<XML document>".

Access Rules

None.

General Rules

1) Let *SV* be the value of *SVE*.

2) If *SV* is the null value, then the result of <XML comment> is the null value and no further General Rules of this Subclause are applied.

3) If the value of

   ```
   '<!--' || SV || '-->'
   ```

   does not conform to rule [15], "comment", of [XML], then an exception condition is raised: *data exception — invalid comment*.

4) Let ***XCII*** be an XQuery comment node with the following properties:

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

a) The value of the **content** property is the value of *SV*, converted to Unicode using the implementation-defined mapping from the character set of *SV* to Unicode.

b) The value of the **parent** property is set to **empty**.

5) The result of the <XML comment> is the XQuery sequence consisting of one node, ***XCII***.

Conformance Rules

1) Without Feature X036, "XMLComment", conforming SQL language shall not contain an <XML comment>.

2) Without Feature X241, "RETURNING CONTENT in XML publishing", in conforming SQL language, an <XML comment> shall not specify an <XML returning clause> that is RETURNING CONTENT.

3) Without Feature X242, "RETURNING SEQUENCE in XML publishing", in conforming SQL language, an <XML comment> shall not specify an <XML returning clause> that is RETURNING SEQUENCE.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.12 <XML concatenation>

Function

Concatenate a list of XML values.

Format

```
<XML concatenation> ::=
  XMLCONCAT <left paren> <XML value expression>
      { <comma> <XML value expression> }...
      [ <XML returning clause> ] <right paren>
```

Syntax Rules

1) If <XML returning clause> is not specified, then it is implementation-defined whether RETURNING CONTENT or RETURNING SEQUENCE is implicit.

2) Let *N* be the number of <XML value expression>s simply contained in <XML concatenation>. Let XVE_i, 1 (one) ≤ *i* ≤ *N*, be the *i*-th <XML value expression> and let T_i be the declared type of XVE_i.

3) Case:

   a) If RETURNING CONTENT is specified or implied, then the declared type of <XML concatenation> is

      Case:

      i) If, for all *i*, 1 (one) ≤ *i* ≤ *N*, T_i is either XML(DOCUMENT(UNTYPED)) or XML(CONTENT(UNTYPED)), then XML(CONTENT(UNTYPED)).

      ii) If there exists a registered XML Schema *S* and a global element declaration schema component ***E*** such that, for all *i*, 1 (one) ≤ *i* < *N*, T_i is XML(DOCUMENT(XMLSCHEMA)) or XML(CONTENT(XMLSCHEMA)) and the XML type descriptor of T_i includes the registered XML Schema descriptor of *S* and an XML namespace URI ***NS*** that is identical to the XML Namespace URI of ***E***, as defined by [Namespaces], and an XML NCName ***EN*** that is equivalent to the XML NCName of *E*, then XML(CONTENT(XMLSCHEMA)), whose XML type descriptor includes the registered XML Schema descriptor of *S* and the XML namespace URI ***NS*** and the XML NCName ***EN***.

      iii) If there exists a registered XML Schema *S* and an XML namespace URI ***NS*** such that, for all *i*, 1 (one) ≤ *i* < *N*, T_i is XML(DOCUMENT(XMLSCHEMA)) or XML(CONTENT(XMLSCHEMA)) and the XML type descriptor of T_i includes the registered XML Schema descriptor of *S* and an XML namespace URI that is identical to ***NS***, as defined by [Namespaces], then XML(CONTENT(XMLSCHEMA)), whose XML type descriptor includes the registered XML Schema descriptor of *S* and the XML namespace URI ***NS***.

      iv) If there exists a registered XML Schema *S* such that, for all *i*, 1 (one) ≤ *i* < *N*, T_i is XML(DOCUMENT(XMLSCHEMA)) or XML(CONTENT(XMLSCHEMA)) and the XML type descriptor of T_i includes the registered XML Schema descriptor of *S*, then XML(CON-

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

TENT(XMLSCHEMA)), whose XML type descriptor includes the registered XML Schema descriptor of *S*.

v) Otherwise, XML(CONTENT(ANY)).

b) Otherwise, the declared type of <XML concatenation> is XML(SEQUENCE).

4) If RETURNING CONTENT is specified or implied, then

a) Let *RT* be the declared type of <XML concatenation>.

b) <XML concatenation> is equivalent to

```
CAST(
  XMLDOCUMENT (
    XMLCONCAT ( XVE1, XVE2, ..., XVEN RETURNING SEQUENCE )
    RETURNING CONTENT )
  AS RT )
```

Access Rules

None.

General Rules

1) For all i, 1 (one) $\le i \le N$, let XV_i be the results of XVE_i.

2) Let R_0 be the null value.

3) For all i, 1 (one) $\le i \le N$, let R_i be the result of applying the General Rules of Subclause 10.14, "Concatenation of two XML values", with R_{i-1} and XV_i as *the two XML values*.

4) The result of <XML concatenation> is R_N.

Conformance Rules

1) Without Feature X020, "XMLConcat", conforming SQL language shall not contain an <XML concatenation>.

2) Without Feature X241, "RETURNING CONTENT in XML publishing", in conforming SQL language, an <XML concatenation> shall not specify an <XML returning clause> that is RETURNING CONTENT.

3) Without Feature X242, "RETURNING SEQUENCE in XML publishing", in conforming SQL language, an <XML concatenation> shall not specify an <XML returning clause> that is RETURNING SEQUENCE.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.13 <XML document>

Function

Generate an XML value with a single XQuery document node.

Format

```
<XML document> ::=
  XMLDOCUMENT <left paren> <XML value expression>
      [ <XML returning clause> ] <right paren>
```

Syntax Rules

1) If <XML returning clause> is not specified, then it is implementation-defined whether RETURNING CONTENT or RETURNING SEQUENCE is implicit.

2) Let *XVE* be the <XML value expression> simply contained in the <XML document>.

3) Case:

   a) If RETURNING CONTENT is specified or implied, then it is implementation-defined whether the declared type of <XML document> is XML(CONTENT(UNTYPED)) or XML(CONTENT(ANY)).

   b) Otherwise, the declared type of <XML document> is XML(SEQUENCE).

Access Rules

None.

General Rules

1) Let *XV* be the value of *XVE*.

2) If *XV* is the null value, then the result of <XML document> is the null value and no further General Rules of this Subclause are applied.

3) Let ***XVFT*** be the XQuery formal type notation determined by the Syntax Rules of Subclause 10.21, "Determination of an XQuery formal type notation", with *XV* as *SOURCE*, and *False* as *CONTEXT*.

4) Let ***XSC*** be an XQuery static context created according to the Syntax Rules of Subclause 10.20, "Creation of an XQuery expression context", with the BNF non-terminal argument omitted.

5) The construction mode component of ***XSC*** is set to **preserve**.

6) Let ***XDC*** be an XQuery dynamic context created according to the General Rules of Subclause 10.20, "Creation of an XQuery expression context".

7) Let ***XSC*** and ***XDC*** be augmented with an XQuery variable **$EXPR**, whose XQuery formal type notation is ***XVFT***, and whose value is *XV*.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8) Case:

   a) If Feature X211, "XML 1.1 support" is supported, then let *DN* be the result of an XQuery evaluation with XML 1.1 lexical rules, using ***XSC*** and ***XDC*** as the XQuery expression context of the XQuery expression

      ```
      document { $EXPR }
      ```

      If an XQuery error occurs during the evaluation, then an exception condition is raised: *XQuery error*.

   b) Otherwise, let *DN* be the result of an XQuery evaluation with XML 1.0 lexical rules, using ***XSC*** and ***XDC*** as the XQuery expression context of the XQuery expression

      ```
      document { $EXPR }
      ```

      If an XQuery error occurs during the evaluation, then an exception condition is raised: XQuery *error*.

9) Case:

   a) If the declared type of the <XML document> is XML(CONTENT(UNTYPED)), then the result of the <XML document> is the result of applying the General Rules of Subclause 10.19, "Constructing an unvalidated XQuery document node", with *DN* as *XQuery document node*.

   b) Otherwise, the result of the <XML document> is *DN*.

Conformance Rules

1) Without Feature X030, "XMLDocument", conforming SQL language shall not contain an <XML document>.

2) Without Feature X241, "RETURNING CONTENT in XML publishing", in conforming SQL language, an <XML document> shall not specify an <XML returning clause> that is RETURNING CONTENT.

3) Without Feature X242, "RETURNING SEQUENCE in XML publishing", in conforming SQL language, an <XML document> shall not specify an <XML returning clause> that is RETURNING SEQUENCE.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.14 <XML element>

Function

Generate an XML value with a single XQuery element node, possibly as a child of an XQuery document node.

Format

```
<XML element> ::=
  XMLELEMENT <left paren> NAME <XML element name>
      [ <comma> <XML namespace declaration> ] [ <comma> <XML attributes> ]
      [ { <comma> <XML element content> }...
      [ OPTION <XML content option> ] ]
      [ <XML returning clause> ] <right paren>

<XML element name> ::=
  <identifier>

<XML attributes> ::=
  XMLATTRIBUTES <left paren> <XML attribute list> <right paren>

<XML attribute list> ::=
  <XML attribute> [ { <comma> <XML attribute> }... ]

<XML attribute> ::=
  <XML attribute value> [ AS <XML attribute name> ]

<XML attribute value> ::=
  <value expression>

<XML attribute name> ::=
  <identifier>

<XML element content> ::=
  <value expression>

<XML content option> ::=
    NULL ON NULL
  | EMPTY ON NULL
  | ABSENT ON NULL
  | NIL ON NULL
  | NIL ON NO CONTENT
```

Syntax Rules

1) The scope of the <XML namespace declaration> is the <XML element>.

2) Let *n* be the number of occurrences of <XML attribute> in <XML attribute list>.

3) Let *i* range from 1 (one) to *n*.

   a) Let A_i be the *i*-th <XML attribute> contained in <XML attribute list>.

   b) Let AV_i be the <XML attribute value> of A_i.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

c) The declared type of AV_i shall be either a predefined type other than XML or a distinct type whose source type is not an XML type.

d) Case:

i) If A_i contains an <XML attribute name> AN_i, then let ***ANC***$_i$ be AN_i.

ii) Otherwise, AV_i shall be a <column reference>. Let CN_i be the <column name> of the column designated by the <column reference> that is AV_i. Let ***ANC***$_i$ be the result of the fully escaped mapping from an SQL <identifier> to an XML Name specified in Subclause 9.1, "Mapping SQL <identifier>s to XML Names", applied to CN_i as *IDENT*.

e) ***ANC***$_i$ shall be an XML 1.1 QName.

f) ***ANC***$_i$ shall not be equivalent to **'xmlns'**, and ***ANC***$_i$ shall not have an XML QName prefix that is equivalent to **'xmlns'**.

g) The Syntax Rules of Subclause 10.12, "Determination of namespace URI", are applied, with ***ANC***$_i$ as *QNAME* and with the <XML element> as *BNFTERM*, resulting in an XML namespace URI ***NSURI***$_i$.

4) There shall not be two <XML attribute>s A_i and A_j such that i does not equal j, ***NSURI***$_i$ is identical as defined in [Namespaces] to ***NSURI***$_j$, and the XML QName local part of ***ANC***$_i$ equals the XML QName local part of ***ANC***$_j$.

5) Let ***EN*** be the character representation of <XML element name>. ***EN*** shall be an XML 1.1 QName.

6) The Syntax Rules of Subclause 10.12, "Determination of namespace URI", are applied, with ***EN*** as *QNAME* and with the <XML element> as *BNFTERM*, resulting in an XML namespace URI ***ENSURI***.

7) For each <XML element content> *XEC*, the declared type of *XEC* shall be a predefined type, a collection type that is not based on an XML unmappable type, or a distinct type.

8) Case:

a) If <XML element content> is specified, then:

i) If <XML content option> is not specified, then EMPTY ON NULL is implicit.

ii) Let *OPT* be the explicit or implicit <XML content option>.

b) Otherwise, let *OPT* be the zero-length string.

9) If <XML element content> is specified, then there shall not be an <XML attribute> A_i such that ***NSURI***$_i$ is the XML namespace URI given in Table 2, "XML namespace prefixes and their URIs", corresponding to the XML namespace prefix **xsi**, and the XML QName local part of ANC_i is **nil**.

10) If <XML returning clause> is not specified, then it is implementation-defined whether RETURNING CONTENT or RETURNING SEQUENCE is implicit.

11) Let *XEN* be the <XML element name>.

12) If <XML namespace declaration> is specified, then let *XND* be "<comma> <XML namespace declaration>"; otherwise, let *XND* be a zero-length string.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

13) If <XML attributes> is specified, then let *XAT* be "<comma> <XML attributes>"; otherwise, let *XAT* be the zero-length string.

14) Let *k* be the number of <XML element content>s immediately contained in <XML element>. For each such <XML element content> in order of its position in <XML element> from left to right, let $XMLEC_l$, 1 (one) ≤ *l* ≤ *k*, be "<comma> <XML element content>".

15) Let *XCO* be the specified or implied <XML content option>.

16) Case:

    a) If RETURNING CONTENT is specified or implied, then <XML element> is equivalent to

```
XMLDOCUMENT (
  XMLELEMENT (
    NAME XEN XND XAT XMLEC1 XMLEC2 ... XMLECk XCO
    RETURNING SEQUENCE )
  RETURNING CONTENT )
```

    b) Otherwise, the declared type of <XML element> is XML(SEQUENCE).

    NOTE 22 — If RETURNING CONTENT is specified or implied, then the declared type of <XML element> is effectively established by the Syntax Rules of Subclause 6.13, "<XML document>".

Access Rules

1) If the declared type *DT* of the <value expression> *VE* immediately contained in an <XML attribute value> or an <XML element content> is a distinct type, then let *ST* be the source type of *DT*. The Access Rules of Subclause 6.5, "<cast specification>", are applied to:

```
CAST ( VE AS ST )
```

General Rules

1) Case:

    a) If the <XML element> is contained within the scope of an <XML binary encoding>, then let *XBE* be the <XML binary encoding> with innermost scope that contains the <XML element>.

    b) Otherwise, let *XBE* be an implementation-defined <XML binary encoding>.

2) Case:

    a) If *XBE* specifies BASE64, then let *ENC* be an indication that binary strings are to be encoded in base64.

    b) Otherwise, let *ENC* be an indication that binary strings are to be encoded in hex.

3) Let *i* range from 1 (one) to *n*. If the value of AV_i is not the null value, then:

    a) Let $\mathbf{CAV}_i$ be the result of applying the General Rules of Subclause 9.8, "Mapping values of SQL data types to values of XML Schema data types", to AV_i as *the value of an SQL data type*, *ENC* as *ENCODING*, "absent" as *NULLS*, and *False* as *CHARMAPPING*. $\mathbf{CAV}_i$ is a character string of Unicode characters.

--`,,```,,,,````-`-`,,`,,`,`,,`---
© ISO/IEC 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

b) Let ***AII***$_i$ be an XQuery attribute node having the following properties:

i) The **node-name** is an XQuery atomic value of XQuery atomic type **xs:QName**, whose local name is the XML 1.1 Qname local part of ***ANC***$_i$ and whose namespace URI is ***NSURI***$_i$.

ii) The **string-value** property is ***CAV***$_i$.

iii) The **type-name** property is **xs:untypedAtomic**.

iv) The **parent** property is initially unknown.

4) Let ***ATTRS*** be the XQuery sequence of XML attribute nodes ***AII***$_i$, 1 (one) ≤ i ≤ n, such that AV_i is not the null value.

5) Let N be the number of <XML element content>s. Let V_j be an enumeration of the values of the <XML element content>s, 1 (one) ≤ j ≤ N. Let *CON* be the list of values V_1, ... , V_N.

6) The result of <XML element> is the result of applying the General Rules of Subclause 10.13, "Construction of an XML element", with ***EN*** as the *QNAME*, ***ENSURI*** as the *NAMESPACE*, ***ATTRS*** as the *ATTRIBUTES*, *CON* as the *CONTENTS*, *OPT* as the *OPTION*, and *ENC* as the *ENCODING*.

Conformance Rules

1) Without Feature X031, "XMLElement", conforming SQL language shall not contain an <XML element>.

2) Without Feature X080, "Namespaces in XML publishing", in conforming SQL language, <XML element> shall not immediately contain <XML namespace declaration>.

3) Without Feature X170, "XML null handling options", conforming SQL language shall not specify <XML content option>.

4) Without Feature X171, "NIL ON NO CONTENT option", conforming SQL language shall not specify NIL ON NO CONTENT.

5) Without Feature X211, "XML 1.1 support", in conforming SQL language, an <XML element name> shall be an XML 1.0 QName.

6) Without Feature X211, "XML 1.1 support", in conforming SQL language, an <XML attribute name> shall be an XML 1.0 QName.

7) Without Feature X241, "RETURNING CONTENT in XML publishing", in conforming SQL language, an <XML element> shall not specify an <XML returning clause> that is RETURNING CONTENT.

8) Without Feature X242, "RETURNING SEQUENCE in XML publishing", in conforming SQL language, an <XML element> shall not specify an <XML returning clause> that is RETURNING SEQUENCE.

9) Without Feature X085, "Predefined namespace prefixes", conforming SQL language shall not contain an <XML element name> E that has an XML QName prefix that is not equivalent to an <XML namespace prefix> contained in one or more <XML namespace declaration>s that are the scope of the <XML element> that contains E.

10) Without Feature X085, "Predefined namespace prefixes", conforming SQL language shall not contain an explicit or implicit <XML attribute name> A that has an XML QName prefix other than **'xml'** that is not equivalent to an <XML namespace prefix> contained in one or more <XML namespace declaration>s that are the scope of the <XML element> that contains A.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.15 <XML forest>

Function

Generate an XML value with a sequence of XQuery element nodes, possibly as the children of an XQuery document node.

Format

```
<XML forest> ::=
  XMLFOREST <left paren> [ <XML namespace declaration> <comma> ]
      <forest element list>
      [ OPTION <XML content option> ]
      [ <XML returning clause> ]
      <right paren>

<forest element list> ::=
  <forest element> [ { <comma> <forest element> }... ]

<forest element> ::=
  <forest element value> [ AS <forest element name> ]

<forest element value> ::=
  <value expression>

<forest element name> ::=
  <identifier>
```

Syntax Rules

1) The scope of the <XML namespace declaration> is the <XML forest>.

2) Let *n* be the number of occurrences of <forest element> in <forest element list>. For each *i* between 1 (one) and *n*:

   a) Let F_i be the *i*-th <forest element> contained in <forest element list>.

   b) Let FV_i be the <forest element value> immediately contained in F_i. The declared type of FV_i shall be a predefined type, a collection type that is not based on an XML unmappable type, or a distinct type.

   c) Case:

      i) If F_i contains a <forest element name> FEN_i, then let $\mathbf{FNC}_i$ be FEN_i.

      ii) Otherwise, FV_i shall be a column reference. Let CN_i be the <column name> of the column designated by FV_i. Let $\mathbf{FNC}_i$ be the result of the fully escaped mapping from an SQL <identifier> to an XML Name specified in Subclause 9.1, "Mapping SQL <identifier>s to XML Names", applied to CN_i.

   d) $\mathbf{FNC}_i$ shall be an XML 1.1 QName.

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

e) The Syntax Rules of Subclause 10.12, "Determination of namespace URI", are applied, with ***FNC***$_i$ as *QNAME* and with the <XML forest> as *BNFTERM*, resulting in an XML namespace URI ***NSURI***$_i$.

3) If <XML content option> is not specified, then NULL ON NULL is implicit.

4) If <XML returning clause> is not specified, then it is implementation-defined whether RETURNING CONTENT or RETURNING SEQUENCE is implicit.

5) If <XML namespace declaration> is specified, then let *XND* be "<XML namespace declaration> <comma>"; otherwise, let *XND* be a zero-length string.

6) Let *FEL* be the <forest element list>.

7) Let *XCO* be the specified or implied <XML content option>.

8) Case:

a) If RETURNING CONTENT is specified or implied, then <XML forest> is equivalent to

```
XMLDOCUMENT (
  XMLFOREST (
    XND FEL XCO
    RETURNING SEQUENCE )
  RETURNING CONTENT )
```

b) Otherwise, the declared type of <XML forest> is XML(SEQUENCE).

NOTE 23 — If RETURNING CONTENT is specified or implied, then the declared type of <XML forest> is effectively established by the Syntax Rules of Subclause 6.13, "<XML document>".

Access Rules

1) If the declared type *DT* of the <value expression> *VE* immediately contained in a <forest element value> is a distinct type, then let *ST* be the source type of *DT*. The Access Rules of Subclause 6.5, "<cast specification>", are applied to:

```
CAST ( VE AS ST )
```

General Rules

1) Case:

a) If the <XML forest> is contained within the scope of an <XML binary encoding>, then let *XBE* be the <XML binary encoding> with innermost scope that contains the <XML forest>.

b) Otherwise, let *XBE* be an implementation-defined <XML binary encoding>.

2) Case:

a) If *XBE* specifies BASE64, then let *ENC* be an indication that binary strings are to be encoded in base64.

b) Otherwise, let *ENC* be an indication that binary strings are to be encoded in hex.

3) For each *i* between 1 (one) and *n*, let V_i be the value of the *i*-th <forest element> contained in <forest element list>.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4) For each *i* between 1 (one) and *n*, let CON_i be the list of values consisting of the single value V_i.

5) For each *i* between 1 (one) and *n*, let XV_i be the result of applying the General Rules of Subclause 10.13, "Construction of an XML element", with $\mathbf{FNC}_i$ as the *QNAME*, $\mathbf{NSURI}_i$ as the *NAMESPACE*, the empty XQuery sequence as the *ATTRIBUTES*, CON_i as the *CONTENTS*, the explicit or implicit <XML content option> as the *OPTION*, and *ENC* as the *ENCODING*.

6) Let R_1 be XV_1. For all *i*, $2 \le i \le n$, let R_i be the result of applying the General Rules of Subclause 10.14, "Concatenation of two XML values", with R_{i-1} and XV_i as *the two XML values*.

7) The result of <XML forest> is R_n.

Conformance Rules

1) Without Feature X032, "XMLForest", conforming SQL language shall not contain an <XML forest>.

2) Without Feature X080, "Namespaces in XML publishing", in conforming SQL language, <XML forest> shall not immediately contain <XML namespace declaration>.

3) Without Feature X211, "XML 1.1 support", in conforming SQL language, a <forest element name> shall be an XML 1.0 QName.

4) Without Feature X241, "RETURNING CONTENT in XML publishing", in conforming SQL language, an <XML forest> shall not specify an <XML returning clause> that is RETURNING CONTENT.

5) Without Feature X242, "RETURNING SEQUENCE in XML publishing", in conforming SQL language, an <XML forest> shall not specify an <XML returning clause> that is RETURNING SEQUENCE.

6) Without Feature X085, "Predefined namespace prefixes", conforming SQL language shall not contain an explicit or implicit <forest element name> *F* that has an XML QName prefix that is not equivalent to an <XML namespace prefix> contained in one or more <XML namespace declaration>s that are the scope of the <XML forest> that contains *F*.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.16 <XML parse>

Function

Perform a non-validating parse of a string to produce an XML value.

Format

```
<XML parse> ::=
  XMLPARSE <left paren> <document or content> <string value expression>
      <XML whitespace option> <right paren>

<XML whitespace option> ::=
  { PRESERVE | STRIP } WHITESPACE
```

Syntax Rules

1) Case:

   a) If <document or content> is DOCUMENT, then it is implementation-defined whether the declared type of <XML parse> is XML(DOCUMENT(UNTYPED)) or XML(DOCUMENT(ANY)).

   b) Otherwise, it is implementation-defined whether the declared type of <XML parse> is XML(CONTENT(UNTYPED)) or XML(CONTENT(ANY)).

Access Rules

None.

General Rules

1) Let *DC* be <document or content>.

2) Let *V* be the value of <string value expression>.

3) Let *WO* be the <XML whitespace option>.

4) Case:

   a) If *V* is the null value, then the result of <XML parse> is the null value.

   b) Otherwise, the result of <XML parse> is determined by applying the General Rules of Subclause 10.16, "Parsing a string as an XML value", with *DC* as *SYNTAX*, *V* as *TEXT*, and *WO* as *OPTION*.

Conformance Rules

1) Without Feature X060, "XMLParse: Character string input and CONTENT option", in conforming SQL language, the declared type of the <string value expression> immediately contained in <XML parse> shall not be a character string type and <XML parse> shall not immediately contain a <document or content> that is CONTENT.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2) Without Feature X061, "XMLParse: Character string input and DOCUMENT option", in conforming SQL language, the declared type of the <string value expression> immediately contained in <XML parse> shall not be a character string type and <XML parse> shall not immediately contain a <document or content> that is DOCUMENT.

3) Without Feature X065, "XMLParse: BLOB input and CONTENT option", in conforming SQL language, the declared type of the <string value expression> immediately contained in <XML parse> shall not be a binary string type and <XML parse> shall not immediately contain a <document or content> that is CONTENT.

4) Without Feature X066, "XMLParse: BLOB input and DOCUMENT option", in conforming SQL language, the declared type of the <string value expression> immediately contained in <XML parse> shall not be a binary string type and <XML parse> shall not immediately contain a <document or content> that is DOCUMENT.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.17 <XML PI>

Function

Generate an XML value with a single XQuery processing instruction node, possibly as a child of an XQuery document node.

Format

```
<XML PI> ::=
  XMLPI <left paren> NAME <XML PI target>
      [ <comma> <character value expression> ]
      [ <XML returning clause> ]
      <right paren>

<XML PI target> ::=
  <identifier>
```

Syntax Rules

1) Let *I* be the result of mapping the <identifier> contained in the <XML PI target> to Unicode.

2) *I* shall be an XML 1.1 NCName.

3) *I* shall not consist of three characters, the first being 'X' or 'x', the second being 'M' or 'm', and the third being 'L' or 'l'.

4) If <XML returning clause> is not specified, then it is implementation-defined whether RETURNING CONTENT or RETURNING SEQUENCE is implicit.

5) Let *XPT* be the <XML PI target>.

6) If <character value expression> is specified, then let *CSVE* be "<comma> <character value expression>"; otherwise, let *CSVE* be the zero-length string.

7) Case:

   a) If RETURNING CONTENT is specified or implied, then <XML PI> is equivalent to:

```
XMLDOCUMENT (
  XMLPI (
    NAME XPT CSVE
    RETURNING SEQUENCE )
  RETURNING CONTENT )
```

   b) Otherwise, the declared type of <XML PI> is XML(SEQUENCE).

NOTE 24 — If RETURNING CONTENT is specified or implied, then the declared type of <XML PI> is effectively established by the Syntax Rules of Subclause 6.13, "<XML document>".

Access Rules

None.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

General Rules

1) If the value of the <string value expression> is the null value, then the result of <XML PI> is the null value and no further General Rules of this Subclause are applied.

2) If <character value expression> is not specified, then let *SVE* be `''` (the <character string literal> for the zero-length string); otherwise, let *SVE* be the <string value expression>.

3) Let *USV* be the result of applying the General Rules of Subclause 9.8, "Mapping values of SQL data types to values of XML Schema data types", to *SVE* as *the value of an SQL data type*, an indication that binary strings are to be encoded in hex as *ENCODING*, "absent" as *NULLS*, and *False* as *CHARMAPPING*. *USV* is a character string of Unicode characters.

4) If the value of

   ```
   '<?' I || ' ' || USV || '?>'
   ```

   does not conform to rule [16], "PI", of [XML 1.1], then an exception condition is raised: *data value — invalid processing instruction*.

   NOTE 25 — If the implementation does not support XML 1.1, then the processing instruction target *I* is subject to a Conformance Rule limiting it to be an XML 1.0 Name. Under that assumption, if the proposed processing instruction conforms to rule [16] of [XML 1.1], then it also conforms to rule [16] of [XML 1.0].

5) Let *XVFT* be the XQuery formal type notation determined by the Syntax Rules of Subclause 10.21, "Determination of an XQuery formal type notation", with *USV* as *SOURCE*, and *False* as *CONTEXT*.

6) Let *XSC* be an XQuery static context created according to the Syntax Rules of Subclause 10.20, "Creation of an XQuery expression context", with the BNF non-terminal argument omitted.

7) Let *XDC* be an XQuery dynamic context created according to the General Rules of Subclause 10.20, "Creation of an XQuery expression context".

8) Let *XSC* and *XDC* be augmented with an XQuery variable **$EXPR**, whose XQuery formal type notation is ***XVFT***, and whose value is ***USV***.

9) Case:

   a) If Feature X211, "XML 1.1 support" is supported, then let ***PN*** be the result of an XQuery evaluation with XML 1.1 lexical rules, using ***XSC*** and ***XDC*** as the XQuery expression context of the XQuery expression

      ```
      processing-instruction I { $EXPR }
      ```

   b) Otherwise, let ***PN*** be the result of an XQuery evaluation with XML 1.0 lexical rules, using ***XSC*** and ***XDC*** as the XQuery expression context of the XQuery expression

      ```
      processing-instruction I { $EXPR }
      ```

10) The result of the <XML PI> is ***PN***.

Conformance Rules

1) Without Feature X037, "XMLPI", conforming SQL language shall not contain an <XML PI>.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2) Without Feature X211, “XML 1.1 support”, in conforming SQL language, an <identifier> contained in an <XML PI target>, when mapped to Unicode, shall be an XML 1.0 NCName.

NOTE 26 — The set of XML 1.0 NCNames is a proper subset of the set of XML 1.1 NCNames. That is, all XML 1.0 NCNames are also XML 1.1 NCNames, but not all XML 1.1 NCNames are also XML 1.0 NCNames.

3) Without Feature X241, “RETURNING CONTENT in XML publishing”, in conforming SQL language, an <XML PI> shall not specify an <XML returning clause> that is RETURNING CONTENT.

4) Without Feature X242, “RETURNING SEQUENCE in XML publishing”, in conforming SQL language, an <XML PI> shall not specify an <XML returning clause> that is RETURNING SEQUENCE.

© ISO/IEC 2008 – All rights reserved
--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.18 <XML query>

Function

Evaluate an XQuery expression.

Format

```
<XML query> ::=
  XMLQUERY <left paren>
      <XQuery expression>
      [ <XML query argument list> ]
      [ <XML returning clause>
      [ <XML query returning mechanism> ] ]
      <XML query empty handling option>
      <right paren>

<XQuery expression> ::=
  <character string literal>

<XML query argument list> ::=
  PASSING <XML query default passing mechanism>
      <XML query argument>
      [ { <comma> <XML query argument> }... ]

<XML query default passing mechanism> ::=
  <XML passing mechanism>

<XML query argument> ::=
    <XML query context item>
  | <XML query variable>

<XML query context item> ::=
  <value expression> [ <XML passing mechanism> ]

<XML query variable> ::=
  <value expression> AS <identifier>
      [ <XML passing mechanism> ]

<XML query returning mechanism> ::=
  <XML passing mechanism>

<XML query empty handling option> ::=
    NULL ON EMPTY
  | EMPTY ON EMPTY
```

Syntax Rules

1) Let *XMQ* be the <XML query>.

2) If *XMQ* contains <XML query argument list>, then let *DPM* be the <XML query default passing mechanism> immediately contained in the <XML query argument list>.

3) If <XML returning clause> is not specified, then it is implementation-defined whether RETURNING CONTENT or RETURNING SEQUENCE is implicit.

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4) Case:

   a) If the implicit or explicit <XML returning clause> is RETURNING CONTENT, then <XML query returning mechanism> shall not be specified. Let *XQRM* be BY VALUE.

   b) If *XMQ* contains <XML query returning mechanism>, then let *XQRM* be that <XML query returning mechanism>.

   c) Otherwise, *XMQ* shall contain <XML query argument list>. Let *XQRM* be *DPM*.

5) Let ***XSC*** be an XQuery static context defined by applying the Syntax Rules of Subclause 10.20, "Creation of an XQuery expression context", with the <XML query> as the *BNF*, modifying the in-scope variables component of ***XSC*** as follows:

   a) For each <XML query variable> *XQV*:

      i) The <identifier> *I* contained in *XQV* shall be an XML 1.1 NCName.

      ii) *I* shall not be equivalent to the name of any implementation-defined XQuery variable, or to the <identifier> of any other <XML query variable>.

      iii) Let *VVE* be the <value expression> simply contained in *XQV*.

      iv) The declared type of *VVE* shall be convertible to XML(SEQUENCE) according to the Syntax Rules of Subclause 6.6, "<XML cast specification>".

      v) If the declared type of *VVE* is not an XML type, then *XQV* shall not contain an <XML passing mechanism>.

      vi) If *XQV* contains an <XML passing mechanism>, then let *VPM* be that <XML passing mechanism>; otherwise, let *VPM* be *DPM*.

      vii) Let ***VFT*** be the XQuery formal type notation determined by the Syntax Rules of Subclause 10.21, "Determination of an XQuery formal type notation", with *VVE* as *SOURCE*, and *False* as *CONTEXT*.

      viii) The pair (*I*, ***VFT***) is placed in the in-scope variables component of ***XSC***.

   b) If *XMQ* contains <XML query context item>, then

      i) *XMQ* shall contain exactly one <XML query context item> *XQCI*.

      ii) Let *CVE* be the <value expression> simply contained in *XQCI*.

      iii) The declared type of *CVE* shall be convertible to XML(SEQUENCE) according to the Syntax Rules of Subclause 6.6, "<XML cast specification>".

      iv) If the declared type declared type of *CVE* is not an XML type, then *XQCI* shall not contain an <XML passing mechanism>.

      v) If *XQCI* contains an <XML passing mechanism>, then let *CPM* be that <XML passing mechanism>; otherwise, let *CPM* be *DPM*.

      vi) Let ***CFT*** be the XQuery formal type notation determined by the Syntax Rules of Subclause 10.21, "Determination of an XQuery formal type notation", with *CVE* as *SOURCE*, and *True* as *CONTEXT*.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

vii) Let **$*fs*:dot** be the XQuery variable in the fictitious namespace associated with the prefix ***fs*** as posited in [XQueryFS], section 3.1.2, "Dynamic context", corresponding to the context item in ***XSC***. The pair (***fs*:dot**, ***CFT***) is placed in the in-scope variables component of ***XSC***.

NOTE 27 — Initialization of the XQuery static context ***XSC*** can in many cases be overridden by the prolog of the <XQuery expression>.

6) <XQuery expression> shall conform to an implementation-defined subset of the normative rules of XQuery expressions that are found in [XQuery], [XQueryFO], [XQueryFS], and [XQueryUpdate], and that define the analysis phase, using the XQuery static context *XSC* and XML 1.1 lexical rules, without raising an XQuery static error, type error, or statically-detected dynamic error.

NOTE 28 — The subset of mandatory and optional features of XQuery supported by an implementation is implementation-defined. This includes:

— What subset of the BNF in [XQuery] and [XQueryUpdate] is supported.

— For the supported BNF, which of the normative rules in [XQuery] and [XQueryFS] are supported.

— Which of the functions in [XQueryFO] are supported.

— Which of the features described in the [XQuery] section titled "Optional Features" are supported, and to what extent.

NOTE 29 — XQuery permits an implementation to postpone some or all type checking to the evaluation phase (*i.e.*, until the General Rules of this Subclause are performed).

7) <XQuery expression> shall not contain an updating expression, except as part of a transform expression, as defined by [XQueryUpdate].

8) Let *XQE* be the <XQuery expression>.

9) If <XML query argument list> is specified, then let *XQAL* be that <XML query argument list>; otherwise, let *XQAL* be the zero-length string.

10) Let *XQEHO* be the <XML query empty handling option>.

11) Case:

a) If RETURNING CONTENT is specified or implied, then <XML query> is equivalent to

```
XMLDOCUMENT (
  XMLQUERY (
    XQE XQAL
    RETURNING SEQUENCE XQRM XQEHO )
  RETURNING CONTENT )
```

b) Otherwise, the declared type of <XML query> is XML(SEQUENCE).

NOTE 30 — If RETURNING CONTENT is specified, then the declared type of <XML query> is effectively established by the Syntax Rules of Subclause 6.13, "<XML document>".

Access Rules

None.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

General Rules

1) Let ***XDC*** be an XQuery dynamic context created by applying the General Rules of Subclause 10.20, "Creation of an XQuery expression context".

   a) Case:

      i) If there is no <XML query context item>, then there is no context item in ***XDC***.

      ii) Otherwise,

      Case:

      1) If the value of *CVE* is the null value, then the result of the <XML query> is the null value, and no further General Rules of this Subclause are applied.

      2) Otherwise:

         A) Case:

            I) If the declared type of *CVE* is an XML type, then let *CI* be

            Case:

            1) If *CPM* is BY REF, then the value of *CVE*.

            2) Otherwise, the result of applying the General Rules of Subclause 10.18, "Constructing a copy of an XML value", with the value of *CVE* as the *VALUE*.

            II) Otherwise, let *CI* be the result of the <XML cast specification>:

            ```
            XMLCAST (CVE AS XML(SEQUENCE))
            ```

         B) Case:

            I) If *CI* is the empty XQuery sequence, then there is no context item in ***XDC***.

            II) If *CI* is an XQuery sequence of length 1 (one), then the context item, context position, and context size of ***XDC*** are set by initializing **$*fs*:dot** to reference *CI*, **$*fs*:position** to 1 (one), and **$*fs*:last** to 1 (one).

            III) Otherwise, an exception condition is raised: *data exception — invalid XQuery context item*.

   b) For each <XML query variable> *XQV*:

      i) Let *VVE* be the <value expression> simply contained in *XQV*, and let *V* be the value of *VVE*.

      Case:

      1) If *V* is the null value, then let *VV* be the empty sequence.

      2) If *V* is a value of an XML type, then let *VV* be

      Case:

      A) If *VPM* is BY REF, then *V*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

B) Otherwise, the result of applying the General Rules of Subclause 10.18, "Constructing a copy of an XML value", with *V* as the *VALUE*.

3) Otherwise, let *VV* be the result of

```
XMLCAST ( VVE AS XML(SEQUENCE))
```

ii) The XQuery variable whose QName is equivalent to the <identifier> simply contained in *XQV* is set to *VV*.

2) Case:

a) If the implementation supports Feature X211, "XML 1.1 support", then the <XQuery expression> is evaluated as an XQuery expression with XML 1.1 lexical rules, using ***XSC*** and ***XDC*** as the XQuery expression context, yielding a value *X1* of an XML type.

b) Otherwise, the <XQuery expression> is evaluated as an XQuery expression with XML 1.0 lexical rules, using ***XSC*** and ***XDC*** as the XQuery expression context, yielding a value *X1* of an XML type.

3) If the result of the XQuery evaluation is an XQuery error, then an exception condition is raised: *XQuery error*.

4) Case:

a) If *X1* is an empty sequence and the <XML query empty handling option> is NULL ON EMPTY, then let *X2* be the null value.

b) Otherwise,

Case:

i) If *XQRM* is BY REF, then let *X2* be *X1*.

ii) Otherwise, let *X2* be the result of applying the General Rules of Subclause 10.18, "Constructing a copy of an XML value", with *X1* as *VALUE*.

5) *X2* is the result of the <XML query>.

Conformance Rules

1) Without Feature X200, "XMLQuery", conforming SQL language shall not contain an <XML query>.

2) Without Feature X201, "XMLQuery: RETURNING CONTENT", conforming SQL language shall not contain an <XML query> that contains RETURNING CONTENT.

3) Without Feature X202, "XMLQuery: RETURNING SEQUENCE", conforming SQL language shall not contain an <XML query> that contains RETURNING SEQUENCE.

4) Without Feature X203, "XMLQuery: passing a context item", in conforming SQL language, an <XML query> shall not contain an <XML query context item>.

5) Without Feature X204, "XMLQuery: initializing an XQuery variable", in conforming SQL language, an <XML query> shall not contain an <XML query variable>.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6) Without Feature X205, “XMLQuery: EMPTY ON EMPTY option”, conforming SQL language shall not contain an <XML query> that contains an <XML query empty handling option> that specifies EMPTY ON EMPTY.

7) Without Feature X206, “XMLQuery: NULL ON EMPTY option”, conforming SQL language shall not contain an <XML query> that contains an <XML query empty handling option> that specifies NULL ON EMPTY.

8) Without Feature X211, “XML 1.1 support”, in conforming SQL language, the value of the <XQuery expression> shall be an XQuery expression with XML 1.0 lexical rules.

9) Without Feature X211, “XML 1.1 support”, in conforming SQL language, the <identifier> contained in an <XML query variable> shall be an XML 1.0 NCName.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.19 <XML text>

Function

Generate an XML value with a single XQuery text node, possibly as a child of an XQuery document node.

Format

```
<XML text> ::=
  XMLTEXT <left paren> <character value expression>
      [ <XML returning clause> ] <right paren>
```

Syntax Rules

1) If <XML returning clause> is not specified, then it is implementation-defined whether RETURNING CONTENT or RETURNING SEQUENCE is implicit.

2) Let *SVE* be the <character value expression>.

3) Case:

   a) If RETURNING CONTENT is specified or implied, then <XML text> is equivalent to

```
XMLDOCUMENT (
  XMLTEXT ( SVE RETURNING SEQUENCE )
  RETURNING CONTENT )
```

   b) Otherwise, the declared type of <XML text> is XML(SEQUENCE).

NOTE 31 — If RETURNING CONTENT is specified or implied, then the declared type of <XML text> is effectively established by the Syntax Rules of Subclause 6.13, "<XML document>".

Access Rules

None.

General Rules

1) Let *SV* be the value of *SVE*.

2) If *SV* is the null value, then the result of <XML text> is the null value and no further General Rules of this Subclause are applied.

3) Case:

   a) If <XML text> is contained within the scope of an <XML binary encoding>, then let *XBE* be the <XML binary encoding> with innermost scope that contains the <XML text>.

   b) Otherwise, let *XBE* be an implementation-defined <XML binary encoding>.

4) Case:

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

a) If *XBE* specifies BASE64, then let *ENC* be an indication that binary strings are to be encoded in base64.

b) Otherwise, let *ENC* be an indication that binary strings are to be encoded in hex.

5) Let ***USV*** be the result of applying the General Rules of Subclause 9.8, "Mapping values of SQL data types to values of XML Schema data types", to *SV* as *the value of an SQL data type*, *ENC* as *ENCODING*, "absent" as *NULLS*, and *False* as *CHARMAPPING*. ***USV*** is a character string of Unicode characters.

6) Let ***XTN*** be an XQuery text node with the following properties:

a) The value of the **content** property is the value of ***USV***.

b) The value of the **parent** property is set to **empty**.

7) The result of <XML text> is the XQuery sequence consisting of one node, ***XTN***.

Conformance Rules

1) Without Feature X038, "XMLText", conforming SQL language shall not contain an <XML text>.

2) Without Feature X241, "RETURNING CONTENT in XML publishing", in conforming SQL language, an <XML text> shall not specify an <XML returning clause> that is RETURNING CONTENT.

3) Without Feature X242, "RETURNING SEQUENCE in XML publishing", in conforming SQL language, an <XML text> shall not specify an <XML returning clause> that is RETURNING SEQUENCE.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6.20 <XML validate>

Function

Validate an XML value.

Format

```
<XML validate> ::=
  XMLVALIDATE <left paren>
       <document or content or sequence>
       <XML value expression>
       [ <XML valid according to clause> ]
       <right paren>

<document or content or sequence> ::=
    <document or content>
  | SEQUENCE
```

Syntax Rules

1) Let *XVE* be the <XML value expression>. Let *DCS* be the <document or content or sequence>.

2) If <XML valid according to clause> is specified, then let *RXS* be the indicated registered XML Schema, let *ENSURI* be the indicated XML namespace, if any, and let *EN* be the XML NCName of the indicated global element declaration schema component, if any.

   NOTE 32 — Indicated registered XML Schema, indicated XML namespace, and indicated global element declaration schema component are defined in Subclause 11.6, "<XML valid according to clause>".

3) Let *XVT* be the declared type of *XVE*. The declared type of <XML validate> is

   Case:

   a) If *XVT* is XML(DOCUMENT(ANY)), XML(DOCUMENT(UNTYPED)), or XML(DOCUMENT(XMLSCHEMA)), then

      Case:

      i) If <XML valid according to clause> is specified, then

         Case:

         1) If <XML valid element name specification> is specified, then XML(DOCUMENT(XMLSCHEMA)) whose XML type descriptor includes the registered XML Schema descriptor of *RXS*, the XML namespace *ENSURI*, and XML NCName *EN*.

         2) If <XML valid element namespace specification> is specified, then XML(DOCUMENT(XMLSCHEMA)) whose XML type descriptor includes the registered XML Schema descriptor of *RXS* and the XML namespace *ENSURI*.

         3) Otherwise, XML(DOCUMENT(XMLSCHEMA)) whose XML type descriptor includes the registered XML Schema descriptor of *RXS*.

      ii) Otherwise, XML(DOCUMENT(ANY)).

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

b) If *XVT* is XML(CONTENT(ANY)), XML(CONTENT(UNTYPED)), or XML(CONTENT(XMLSCHEMA)), then

Case:

i) If <XML valid according to clause> is specified, then

Case:

1) If <XML valid element name specification> is specified, then XML(CONTENT(XMLSCHEMA)) whose XML type descriptor includes the registered XML Schema descriptor of *RXS*, the XML namespace *ENSURI*, and XML NCName *EN*.

2) If <XML valid element namespace specification> is specified, then XML(CONTENT(XMLSCHEMA)) whose XML type descriptor includes the registered XML Schema descriptor of *RXS* and the XML namespace *ENSURI*.

3) Otherwise, XML(CONTENT(XMLSCHEMA)) whose XML type descriptor includes the registered XML Schema descriptor of *RXS*.

ii) Otherwise, XML(CONTENT(ANY)).

c) If *XVT* is XML(SEQUENCE), then XML(SEQUENCE).

Access Rules

None.

General Rules

1) Let *V* be the value of *XVE*.

2) If *V* is the null value, then the result of the <XML validate> is the null value and no further General Rules of this Subclause are applied.

3) Case:

a) If *DCS* is DOCUMENT and *V* is not an XQuery sequence of length 1 (one) whose sole XQuery item is an XQuery document node whose **children** property contains exactly one XQuery element node, zero or more XQuery comment nodes, and zero or more XQuery processing instruction nodes, then an exception condition is raised: *data exception — invalid XML document*.

b) If *DCS* is CONTENT and *V* is not an XQuery sequence of length 1 (one) whose sole XQuery item is an XQuery document node, then an exception condition is raised: *data exception — invalid XML content*.

c) Otherwise, if any XQuery item contained in *V* is an XQuery atomic value, XQuery attribute node, XQuery namespace node, or an XQuery text node, then an exception condition is raised: *data exception — XQuery sequence cannot be validated*.

4) If *V* is an empty XQuery sequence, then the result of <XML validate> is an empty XQuery sequence and no further General Rules of this Subclause are applied.

5) Let *N* be the number of XQuery items in *V*. Let $\boldsymbol{I}_j$, 1 (one) $\leq j \leq N$, be an enumeration in order of the XQuery items in *V*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

6) For each j, 1 (one) $\le j \le N$,

Case:

a) If $\boldsymbol{I}_j$ is an XQuery document node ***D***, then:

i) Let M be the number of XQuery nodes in the **children** property of ***D***.

ii) Let $\boldsymbol{C}_k$, 1 (one) $\le k \le M$, be an enumeration in order of the XQuery nodes in the **children** property of ***D***.

iii) For each k, 1 (one) $\le k \le M$,

Case:

1) If $\boldsymbol{C}_k$ is not an XQuery element node, an XQuery comment node, or an XQuery processing instruction node, then an exception condition is raised: *data exception — XQuery document node cannot be validated*.

2) If $\boldsymbol{C}_k$ is an XQuery element node, then:

A) Case:

I) If <XML valid according to clause> is specified, then let XS_k be *RXS*.

II) Otherwise:

1) Let ***NS*** be the XML namespace of $\boldsymbol{C}_k$.

2) Let XS_k be a registered XML Schema for which the current user has USAGE privilege and whose target namespace is identical to ***NS***, as defined by [Namespaces], chosen according to a deterministic implementation-defined algorithm that is repeatable, in the sense that if the algorithm is re-evaluated with the same collection of registered XML Schemas that are accessible to the user and the same value *V*, then the same registered XML Schema will be chosen. If no XS_k is found, then an exception condition is raised: *data exception — no XML schema found*.

B) If <XML valid element clause> is specified, then let ***ENAME*** be the **node-name** property of $\boldsymbol{C}_k$.

Case:

I) If <XML valid element name specification> is specified, and either the XML NCName of ***ENAME*** is not equivalent to *EN*, or the XML namespace of ***ENAME*** is not identical to *ENSURI*, as defined by [Namespaces], then an exception condition is raised: *data exception — no XML element with the specified QName*.

II) Otherwise, if <XML valid element namespace specification> is specified, and the XML namespace of ***ENAME*** is not identical to *ENSURI*, as defined by [Namespaces], then an exception condition is raised: *data exception — no XML element with the specified namespace*.

C) Case:

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

I) If *DCS* is DOCUMENT, then the General Rules of Subclause 10.22, "Validating an XQuery document or element node", are applied with $\mathbf{I}_j$ as *ITEM* and XS_k as *SCHEMA*. Let *STAT* be the *STATUS* and **RES** be the *RESULT*, if any, returned by the General Rules of Subclause 10.22, "Validating an XQuery document or element node".

II) Otherwise, the General Rules of Subclause 10.22, "Validating an XQuery document or element node", are applied with $\mathbf{C}_k$ as *ITEM* and XS_k as *SCHEMA*. Let *STAT* be the *STATUS* and **RES** be the *RESULT*, if any, returned by the General Rules of Subclause 10.22, "Validating an XQuery document or element node".

D) If *STAT* is FAILURE, then an exception condition is raised: *data exception — validation failure*.

E) Case:

I) If *DCS* is DOCUMENT, then let $\mathbf{O}_k$ be the XQuery element node that is the **child** of **RES**.

II) Otherwise, let $\mathbf{O}_k$ be **RES**.

3) Otherwise, let $\mathbf{O}_k$ be the result of applying the General Rules of Subclause 10.18, "Constructing a copy of an XML value", with $\mathbf{C}_k$ as *VALUE*.

iv) Let $\mathbf{R}_j$ be an XQuery document node whose **children** property is replaced by an enumeration of $\mathbf{O}_k$, 1 (one) $\le k \le M$.

b) If $\mathbf{I}_j$ is an XQuery element node **E**, then

i) Case:

1) If <XML valid according to clause> is specified, then let *XS* be *RXS*.

2) Otherwise, let **NS** be the XML namespace of **E**. Let *XS* be a registered XML Schema for which the current user has USAGE privilege and whose target namespace is identical to **NS**, as defined by [Namespaces], chosen according to a deterministic implementation-defined algorithm that is repeatable, in the sense that if the algorithm is re-evaluated with the same collection of registered XML Schemas that are accessible to the user and the same value *V*, then the same registered XML Schema will be chosen. If no *XS* is found, then an exception condition is raised: *data exception — no XML schema found*.

ii) If <XML valid element clause> is specified, then let **ENAME** be the **node-name** property of **E**.

Case:

1) If <XML valid element name specification> is specified, and either the XML NCName of **ENAME** is not equivalent to *EN* or the XML namespace of **ENAME** is not identical to *ENSURI*, as defined by [Namespaces], then an exception condition is raised: *data exception — no XML element with the specified QName*.

2) Otherwise, if <XML valid element namespace specification> is specified and the XML namespace of **ENAME** is not identical to *ENSURI*, as defined by [Namespaces], then an exception condition is raised: *data exception — no XML element with the specified namespace*.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

iii) The General Rules of Subclause 10.22, "Validating an XQuery document or element node", are applied with $\boldsymbol{I}_j$ as *ITEM* and *XS* as *SCHEMA*. Let *STAT* be the *STATUS* and ***RES*** be the *RESULT*, if any, returned by the General Rules of Subclause 10.22, "Validating an XQuery document or element node".

iv) If *STAT* is FAILURE, then an exception condition is raised: *data exception — validation failure*.

v) Let $\boldsymbol{R}_j$ be ***RES***.

c) Otherwise, $\boldsymbol{I}_j$ is an XQuery comment node or an XQuery processing instruction node. Let $\boldsymbol{R}_j$ be the result of applying the General Rules of Subclause 10.18, "Constructing a copy of an XML value", with $\boldsymbol{I}_j$ as *VALUE*.

7) Let ***R*** be an XQuery sequence enumerated by $\boldsymbol{R}_j$, 1 (one) $\le j \le N$.

8) The result of <XML validate> is ***R***.

Conformance Rules

1) Without Feature X271, "XMLValidate: data-driven case", conforming SQL language shall not contain an <XML validate> that does not contain <XML valid according to clause>.

2) Without Feature X272, "XMLValidate: ACCORDING TO clause", conforming SQL language shall not contain an <XML validate> that contains <XML valid according to clause>.

3) Without Feature X273, "XMLValidate: ELEMENT clause", conforming SQL language shall not contain an <XML validate> that contains <XML valid element clause>.

4) Without Feature X284, "XMLValidate: NAMESPACE without ELEMENT clause", conforming SQL language shall not contain an <XML validate> that contains an <XML valid element clause> that does not contain an <XML valid element name specification>.

5) Without Feature X286, "XMLValidate: NO NAMESPACE with ELEMENT clause", conforming SQL language shall not contain an <XML validate> that contains an <XML valid element namespace specification> that contains NO NAMESPACE.

6) Without Feature X274, "XMLValidate: schema location", conforming SQL language shall not contain an <XML validate> that contains <XML valid schema location>.

7) Without Feature X281, "XMLValidate with DOCUMENT option", conforming SQL language shall not contain an <XML validate> that immediately contains a <document or content or sequence> that is DOCUMENT.

8) Without Feature X282, "XMLValidate with CONTENT option", conforming SQL language shall not contain an <XML validate> that immediately contains a <document or content or sequence> that is CONTENT.

9) Without Feature X283, "XMLValidate with SEQUENCE option", conforming SQL language shall not contain an <XML validate> that immediately contains a <document or content or sequence> that is SEQUENCE.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7 Query expressions

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

7.1 <table reference>

This Subclause modifies Subclause 7.6, "<table reference>", in ISO/IEC 9075-2.

Function

Reference a table.

Format

```
<table primary> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <XML iterate> [ AS ] <correlation name>
      <left paren> <derived column list> <right paren>
  | <XML table> [ AS ] <correlation name>
      [ <left paren> <derived column list> <right paren> ]

<XML iterate> ::=
  XMLITERATE <left paren> <XML value expression> <right paren>

<XML table> ::=
  XMLTABLE <left paren>
      [ <XML namespace declaration> <comma> ]
      <XML table row pattern>
      [ <XML table argument list> ]
      COLUMNS <XML table column definitions> <right paren>

<XML table row pattern> ::=
  <character string literal>

<XML table argument list> ::=
  PASSING <XML table argument passing mechanism>
      <XML query argument>
      [ { <comma> <XML query argument> }... ]

<XML table argument passing mechanism> ::=
  <XML passing mechanism>

<XML table column definitions> ::=
  <XML table column definition>
      [ { <comma> <XML table column definition> }... ]

<XML table column definition> ::=
    <XML table ordinality column definition>
```

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
  | <XML table regular column definition>

<XML table ordinality column definition> ::=
  <column name> FOR ORDINALITY

<XML table regular column definition> ::=
  <column name> <data type> [ <XML passing mechanism> ]
      [ <default clause> ]
      [ PATH <XML table column pattern> ]

<XML table column pattern> ::=
  <character string literal>
```

Syntax Rules

1) Insert after SR 25) An <XML table> is possibly non-deterministic if it does not conform to implementation-defined rules enabling the SQL-implementation to deduce that the result of the <XML table> is deterministic.

2) Insert after SR 25) A <table primary> is also possibly non-deterministic if it is an <XML table> that is possibly non-deterministic.

3) Insert this SR If <table primary> contains <XML iterate>, then the <derived column list> shall contain exactly two <column name>s *CN1* and *CN2*, which shall not be equivalent. The declared type of <XML iterate> is a row type consisting of two fields, the first of which has a <field name> that is equivalent to *CN1* and a data type that is XML(SEQUENCE), and the second of which has a <field name> that is equivalent to *CN2* and a data type that is exact numeric with scale 0 (zero).

4) Insert this SR If <XML table> is specified, then:

   a) Case:

      i) If <XML namespace declaration> *XND* is specified, then let *XNDC* be

         `WITH` *XND*

      ii) Otherwise, let *XNDC* be the zero-length string.

   b) Let *XTRP* be the <XML table row pattern>.

   c) Case:

      i) If <XML table argument list> is specified, then let *XTTPM* be the <XML table argument passing mechanism>, let *NA* be the number of <XML query argument>s, let XQA_i, 1 (one) ≤ *i* ≤ *NA*, be an enumeration of the <XML query argument>s, and let *XQAL* be

         `PASSING` *XTTPM* XQA_1`, ...,` XQA_{NA}

      ii) Otherwise, let *XTTPM* be BY REF, and let *XQAL* be the zero-length string.

   d) Let *NC* be the number of <XML table column definition>s. Let $XTCD_j$, 1 (one) ≤ j ≤ *NC*, be an enumeration of the <XML table column definition>s, in order from left to right. There shall be at most one $XTCD_j$ that is <XML table ordinality column definition>.

   e) For each *j* between 1 (one) and *NC*, inclusive, let CN_j be the <column name> contained in $XTCD_j$.

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Case:

i) If $XTCD_j$ is an <XML table ordinality column definition>, then let SLI_j be

```
I.N
```

ii) Otherwise:

1) Let DT_j be the <data type> contained in $XTCD_j$.

2) Case:

A) If DT_j is XML(SEQUENCE) then:

I) If <XML table argument list> is not specified, then $XTCD_j$ shall contain <XML passing mechanism>.

II) If $XTCD_j$ contains <XML passing mechanism>, then let XPM_j be this <XML passing mechanism>; otherwise, let XPM_j be *XTTPM*.

III) Let XRM_j be RETURNING SEQUENCE XPM_j.

B) Otherwise, <XML passing mechanism> shall not be specified. Let XRM_j be RETURNING CONTENT.

3) If $XTCD_j$ contains a <default option>, then let DEF_j be that <default option>; otherwise, let DEF_j be NULL.

4) If $XTCD_j$ contains an <XML table column pattern>, then let $PATH_j$ be that <XML table column pattern>; otherwise, let $PATH_j$ be a <character string literal> whose value is equivalent to the column name of the column identified by CN_j.

5) Let XQC_j be

```
XMLQUERY ( PATHj PASSING BY REF I.V XRMj EMPTY ON EMPTY )
```

6) Let XE_j be

```
XMLEXISTS ( PATHj PASSING BY REF I.V )
```

7) Case:

A) If DT_j is XML(SEQUENCE), then let CPM_j be XPM_j.

B) If DT_j is an XML type, then let CPM_j be BY VALUE.

C) Otherwise, let CPM_j be a zero-length string.

8) Let SLI_j be

```
CASE WHEN XEj
    THEN XMLCAST( XQCj AS DTj CPMj )
```

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
            ELSE DEFj
        END
```

f) Let *CORR* be the <correlation name>.

g) Case:

i) If <derived column list> is specified, then let *DCL* be that <derived column list> and let *DCLP* be

```
( DCL )
```

ii) Otherwise, let *DCLP* be the zero-length string.

h) The <XML table> is equivalent to

```
LATERAL
  ( XNDC
    SELECT SLI1 AS CN1, SLI2 AS CN2, ..., SLINC AS CNNC
    FROM XMLITERATE ( XMLQUERY ( XTRP XQAL
                      RETURNING SEQUENCE BY REF EMPTY ON EMPTY ) )
         AS I ( V, N )
  ) AS CORR DCLP
```

Access Rules

No additional Access Rules.

General Rules

1) Insert after GR 2) If a <table primary> *TP* simply contains an <XML iterate>, then let *V* be the value of the <XML value expression>.

Case:

a) If *V* is the null value or the empty XQuery sequence, then the result of *TP* is an empty table.

b) Otherwise, the result of *TP* is a table consisting of one row for each XQuery item in *V*. The value of the first column of each row is the corresponding XQuery item in *V*. The value of the second column is the sequential number of the XQuery item in *V*.

Conformance Rules

1) Insert this CR Conforming SQL language shall not contain <XML iterate>.

NOTE 33 — This Conformance Rule ensures that <XML iterate> exists purely as a specification device within this edition of this part of this International Standard and cannot be used by conforming SQL language. Conforming SQL language may achieve the same effect by use of <XML table>.

2) Insert this CR Without Feature X300, “XMLTable”, conforming SQL language shall not contain <XML table>.

3) Insert this CR Without Feature X301, “XMLTable: derived column list option”, in conforming SQL language, a <table primary> that is an <XML table> shall not contain a <derived column list>.

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4) Insert this CR Without Feature X302, "XMLTable: ordinality column option", in conforming SQL language, an <XML table> shall not contain an <XML table ordinality column definition>.

5) Insert this CR Without Feature X303, "XMLTable: column default option", in conforming SQL language, an <XML table regular column definition> shall not contain a <default clause>.

6) Insert this CR Without Feature X304, "XMLTable: passing a context item", in conforming SQL language, an <XML table argument list> shall not contain an <XML query context item>.

7) Insert this CR Without Feature X305, "XMLTable: initializing an XQuery variable", in conforming SQL language, an <XML table argument list> shall not contain an <XML query variable>.

8) Insert this CR Without Feature X086, "XML namespace declarations in XMLTable", in conforming SQL language, an <XML table> shall not contain an <XML namespace declaration>.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7.2 <query expression>

This Subclause modifies Subclause 7.13, "<query expression>", in ISO/IEC 9075-2.

Function

Specify a table.

Format

```
<with clause> ::=
  WITH [ <XML lexically scoped options> ] [ <comma> ] [ [ RECURSIVE ] <with list> ]
```

Syntax Rules

1) Insert this SR A <with clause> shall contain an <XML lexically scoped option> or a <with list> or both.

2) Insert this SR The scope of an <XML namespace declaration> contained in an <XML lexically scoped options> immediately contained in a <with clause> is the <query expression>.

3) Insert this SR The scope of an <XML binary encoding> contained in an <XML lexically scoped options> immediately contained in a <with clause> is the <query expression>.

4) Insert this SR A <with clause> shall immediately contain <comma> if and only if the <with clause> immediately contains both <XML lexically scoped options> and <with list>.

Access Rules

No additional Access Rules.

General Rules

1) Replace GR 3)b)ii)1) If the declared type of the *i*-th column of *T* is not an XML type, then let DTC_i be that declared type; otherwise, let $DDTC_i$ be the declared type of the *i*-th column of *T* and let DTC_i be

   $DDTC_i$ BY REF

Conformance Rules

1) Insert this CR Without Feature X081, "Query-level XML namespace declarations", in conforming SQL language, <with clause> shall not immediately contain an <XML lexically scoped options> that contains an <XML namespace declaration>.

2) Insert this CR Without Feature X131, "Query-level XMLBINARY clause", in conforming SQL language, a <with clause> shall not immediately contain an <XML lexically scoped options> that contains an <XML binary encoding>.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3) Insert this CR Without Feature X135, “XMLBINARY clause in subqueries”, in conforming SQL language, a <query expression> that is contained in another <query expression> shall not contain an <XML lexically scoped options> that contains an <XML binary encoding>.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

(Blank page)

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8 Predicates

This Clause modifies Clause 8, "Predicates", in ISO/IEC 9075-2.

8.1 <predicate>

This Subclause modifies Subclause 8.1, "<predicate>", in ISO/IEC 9075-2.

Function

Specify a condition that can be evaluated to give a boolean value.

Format

```
<predicate> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <XML content predicate>
  | <XML document predicate>
  | <XML exists predicate>
  | <XML valid predicate>
```

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

1) Replace GR 1) The result of a <predicate> is the truth value of the immediately contained <comparison predicate>, <between predicate>, <in predicate>, <like predicate>, <similar predicate>, <regex like predicate>, <null predicate>, <quantified comparison predicate>, <exists predicate>, <unique predicate>, <match predicate>, <overlaps predicate>, <distinct predicate>, <member predicate>, <submultiset predicate>, <set predicate>, <type predicate>, <XML content predicate>, <XML document predicate>, <XML exists predicate>, or <XML valid predicate>.

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.2 <XML content predicate>

Function

Determine whether an XML value is an XQuery document node.

Format

```
<XML content predicate> ::=
  <row value predicand> <XML content predicate part 2>

<XML content predicate part 2> ::=
  IS [ NOT ] CONTENT
```

Syntax Rules

1) The <row value predicand> shall be a <row value constructor predicand> that is a <common value expression> that is an <XML value expression> *XVE*.

2) The expression

   XVE IS NOT CONTENT

   is equivalent to

   NOT (*XVE* IS CONTENT)

Access Rules

None.

General Rules

1) Let *V* be the value of *XVE*.

2) The result of

   XVE IS CONTENT

   is determined as follows.

   Case:

   a) If *V* is the null value, then the result is *Unknown*.

   b) If *V* is an XQuery sequence of length 1 (one) whose sole XQuery item is an XQuery document node, then the result is *True*.

   c) Otherwise, the result is *False*.

© ISO/IEC 2008 – All rights reserved
--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Conformance Rules

1) Without Feature X091, “XML content predicate”, conforming SQL language shall not contain <XML content predicate>.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.3 <XML document predicate>

Function

Determine whether an XML value is an XQuery document node whose **children** property contains exactly one XQuery element node, zero or more XQuery comment nodes, and zero or more XQuery processing instruction nodes.

Format

```
<XML document predicate> ::=
  <row value predicand> <XML document predicate part 2>

<XML document predicate part 2> ::=
  IS [ NOT ] DOCUMENT
```

Syntax Rules

1) The <row value predicand> shall be a <row value constructor predicand> that is a <common value expression> that is an <XML value expression> *XVE*.

2) The expression

   XVE IS NOT DOCUMENT

   is equivalent to

   NOT (*XVE* IS DOCUMENT)

Access Rules

None.

General Rules

1) Let *V* be the value of *XVE*.

2) The result of

   XVE IS DOCUMENT

   is determined as follows.

   Case:

   a) If *V* is the null value, then the result is *Unknown*.

   b) If *V* is an XQuery sequence of length 1 (one) whose sole XQuery item is an XQuery document node whose **children** property contains exactly one XQuery element node, zero or more XQuery comment nodes, and zero or more XQuery processing instruction nodes, then the result is *True*.

© ISO/IEC 2008 – All rights reserved
--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

c) Otherwise, the result is *False*.

Conformance Rules

1) Without Feature X090, "XML document predicate", conforming SQL language shall not contain <XML document predicate>.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.4 <XML exists predicate>

Function

Specify a test for a non-empty XQuery sequence.

Format

```
<XML exists predicate> ::=
  XMLEXISTS <left paren> <XQuery expression>
      [ <XML query argument list> ]<right paren>
```

Syntax Rules

1) Let *XQE* be the <XQuery expression>.

2) If <XML query argument list> is specified, then let *XQP* be that <XML query argument list>; otherwise, let *XQP* be the zero-length string.

3) The Syntax Rules of Subclause 6.18, "<XML query>", are applied to

   `XMLQUERY ( XQE XQP RETURNING SEQUENCE EMPTY ON EMPTY )`

Access Rules

1) The Access Rules of Subclause 6.18, "<XML query>", are applied to

   `XMLQUERY ( XQE XQP RETURNING SEQUENCE EMPTY ON EMPTY )`

General Rules

1) Let *V* be the value of

   `XMLQUERY ( XQE XQP RETURNING SEQUENCE EMPTY ON EMPTY )`

2) The value of <XML exists predicate> is

   Case:

   a) If *V* is the null value, then *Unknown*.

   b) If *V* is an empty XQuery sequence, then *False*.

   c) Otherwise, *True*.

Conformance Rules

1) Without Feature X096, "XMLExists", conforming SQL language shall not contain <XML exists predicate>.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

8.5 <XML valid predicate>

Function

Specify a test to determine whether an XML value is valid according to a registered XML Schema.

Format

```
<XML valid predicate> ::=
  <row value predicand> <XML valid predicate part 2>

<XML valid predicate part 2> ::=
  IS [ NOT ] VALID
      <document or content or sequence>
      [ <XML valid according to clause> ]
```

Syntax Rules

1) The <row value predicand> shall be a <row value constructor predicand> that is a <common value expression> that is an <XML value expression> *XVE*.

2) Let *DCS* be the <document or content or sequence>. Let *XVACC* be the <XML valid according to clause>, if any; otherwise, let *XVACC* be the zero-length string.

3) If <XML valid predicate> immediately contains NOT, then the <XML valid predicate> is equivalent to

   ```
   NOT ( XVE IS VALID DCS XVACC )
   ```

4) If <XML valid according to clause> is specified, then let *RXS* be the indicated registered XML Schema, let *ENSURI* be the indicated XML namespace, if any, and let *EN* be the XML NCName of the indicated global element declaration schema component, if any.

   NOTE 34 — Indicated registered XML Schema, indicated XML namespace, and indicated global element declaration schema component are defined in Subclause 11.6, "<XML valid according to clause>".

Access Rules

None.

General Rules

1) Let *V* be the value of *XVE*.

2) If *V* is the null value, then the result of the <XML valid predicate> is *Unknown* and no further General Rules of this Subclause are applied.

3) If *DCS* is DOCUMENT and *V* is not an XQuery sequence of length 1 (one) whose sole XQuery item is an XQuery document node whose **children** property contains exactly one XQuery element node, zero or more XQuery comment nodes, and zero or more XQuery processing instruction nodes, then the result of the <XML valid predicate> is *False* and no further General Rules of this Subclause are applied.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4) If *DCS* is CONTENT and *V* is not an XQuery sequence of length 1 (one) whose sole XQuery item is an XQuery document node, then the result of the <XML valid predicate> is *False* and no further General Rules of this Subclause are applied.

5) If *DCS* is SEQUENCE, then

   Case:

   a) If *V* is an empty XQuery sequence, then the result of the <XML valid predicate> is *True* and no further General Rules of this Subclause are applied.

   b) Otherwise, if any XQuery item contained in *V* is an XQuery atomic value, XQuery attribute node, XQuery namespace node, or an XQuery text node, then the result of the <XML valid predicate> is *False* and no further General Rules of this Subclause are applied.

6) Let *N* be the number of XQuery items in *V*. Let $\boldsymbol{I}_j$, 1 (one) $\le j \le N$, be an enumeration in order of the XQuery items in *V*. Let $TV_{0\ (\text{zero})}$ be *True*.

7) For each each *j*, 1 (one) $\le j \le N$,

   Case:

   a) If $\boldsymbol{I}_j$ is an XQuery document node ***D***, then:

      i) Let *M* be the number of XQuery nodes in the **children** property of ***D***.

      ii) Let $\boldsymbol{C}_k$, 1 (one) $\le k \le M$, be an enumeration in order of the XQuery nodes in the **children** property of ***D***. Let $ITV_{0\ (\text{zero})}$ be *True*.

      iii) For each *k*, 1 (one) $\le k \le M$,

         Case:

         1) If $\boldsymbol{C}_k$ is not an XQuery element node, XQuery comment node, or an XQuery processing instruction node, then the result of the <XML valid predicate> is *False* and no further General Rules of this Subclause are applied.

         2) If $\boldsymbol{C}_k$ is an XQuery comment node or an XQuery processing instruction node, then let ITV_k be ITV_{k-1}.

         3) Otherwise, $\boldsymbol{C}_k$ is an XQuery element node.

            A) Case:

               I) If <valid according to clause> is specified, then let XS_k be *RXS* and let *FOUND* be *True*.

               II) Otherwise:

                  1) Let ***NS*** be the XML namespace of $\boldsymbol{C}_k$.

                  2) Let XS_k be a registered XML Schema for which the current user has USAGE privilege and whose target namespace is identical to ***NS***, as defined by [Namespaces], chosen according to a deterministic implementation-defined algorithm that is repeatable, in the sense that if the algorithm is re-evaluated with the same collection of registered XML Schemas that are accessible to

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

the user and the same value *V*, then the same registered XML Schema will be chosen. If there is no XS_k, then let *FOUND* be *False*; otherwise, let *FOUND* be *True*.

B) Case:

I) If *FOUND* is *False*, then let ITV_k be the result of

ITV_{k-1} AND *Unknown*

II) Otherwise:

1) If <XML valid element clause> is specified, then let ***ENAME*** be the **node-name** property of C_k.

Case:

a) If <XML valid element name specification> is specified, and if either the XML NCName of ***ENAME*** is not equivalent to *EN* or the XML namespace of ***ENAME*** is not identical to *ENSURI*, as defined by [Namespaces], then the result of the <XML valid predicate> is *False* and no further General Rules of this Subclause are applied.

b) Otherwise, if <XML valid element namespace specification> is specified, and the XML namespace of ***ENAME*** is not identical to *ENSURI*, as defined by [Namespaces], then the result of the <XML valid predicate> is *False* and no further General Rules of this Subclause are applied.

2) Case:

a) If *DCS* is DOCUMENT, then the General Rules of Subclause 10.22, "Validating an XQuery document or element node", are applied with $\boldsymbol{I}_j$ as *ITEM* and XS_k as *SCHEMA*. Let *STAT* be the *STATUS* returned by the General Rules of Subclause 10.22, "Validating an XQuery document or element node".

b) Otherwise, the General Rules of Subclause 10.22, "Validating an XQuery document or element node", are applied with $\boldsymbol{C}_k$ as *ITEM* and XS_k as *SCHEMA*. Let *STAT* be the *STATUS* returned by the General Rules of Subclause 10.22, "Validating an XQuery document or element node".

3) Case:

a) If *STAT* is FAILURE, then the result of the <XML valid predicate> is *False* and no further General Rules of this Subclause are applied.

b) Otherwise, let ITV_k be ITV_{k-1}.

iv) Let TV_j be the result of

TV_{j-1} AND ITV_M

b) If $\boldsymbol{I}_j$ is an XQuery element node ***E***, then:

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

i) Case:

1) If <XML valid according to clause> is specified, then let *XS* be *RXS*, and let *FOUND* be *True*.

2) Otherwise, let ***NS*** be the XML namespace of ***E***. Let *XS* be a registered XML Schema for which the current user has USAGE privilege and whose target namespace is identical to ***NS***, as defined by [Namespaces], chosen according to a deterministic implementation-defined algorithm that is repeatable, in the sense that if the algorithm is re-evaluated with the same collection of registered XML Schemas that are accessible to the user and the same value *V*, then the same registered XML Schema will be chosen. If there is no *XS*, then let *FOUND* be *False*; otherwise, let *FOUND* be *True*.

ii) Case:

1) If *FOUND* is *False*, then let TV_j be the result of

TV_{j-1} AND *Unknown*

2) Otherwise:

A) If <XML valid element clause> is specified, then let ***ENAME*** be the **node-name** property of ***E***.

Case:

I) If <XML valid element name specification> is specified, and if either the XML NCName of ***ENAME*** is not equivalent to *EN*, or the XML namespace of ***ENAME*** is not identical to *ENSURI*, as defined by [Namespaces], then the result of the <XML valid predicate> is *False* and no further General Rules of this Subclause are applied.

II) Otherwise, if <XML valid element namespace specification> is specified, and the XML namespace of ***ENAME*** is not identical to *ENSURI*, as defined by [Namespaces], then the result of the <XML valid predicate> is *False* and no further General Rules of this Subclause are applied.

B) The General Rules of Subclause 10.22, "Validating an XQuery document or element node", are applied with $\boldsymbol{I}_j$ as *ITEM* and *XS* as *SCHEMA*. Let *STAT* be the *STATUS* returned by the General Rules of Subclause 10.22, "Validating an XQuery document or element node".

C) Case:

I) If *STAT* is FAILURE, then the result of the <XML valid predicate> is *False* and no further General Rules of this Subclause are applied.

II) Otherwise, let TV_j be TV_{j-1}.

c) Otherwise, $\boldsymbol{I}_j$ is an XQuery comment node or an XQuery processing instruction node. Let TV_j be TV_{j-1}.

8) The result of <XML valid predicate> is TV_N.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Conformance Rules

1) Without Feature X141, "IS VALID predicate: data-driven case", conforming SQL language shall not contain an <XML valid predicate> that does not contain <XML valid according to clause>.

2) Without Feature X155, "IS VALID predicate: NAMESPACE without ELEMENT clause", conforming SQL language shall not contain an <XML valid predicate> that contains an <XML valid element clause> that does not contain an <XML valid element name specification>.

3) Without Feature X157, "IS VALID predicate: NO NAMESPACE with ELEMENT clause", conforming SQL language shall not contain an <XML valid predicate> that contains an <XML valid element namespace specification> that contains NO NAMESPACE.

4) Without Feature X142, "IS VALID predicate: ACCORDING TO clause", conforming SQL language shall not contain an <XML valid predicate> that contains <XML valid according to clause>.

5) Without Feature X143, "IS VALID predicate: ELEMENT clause", conforming SQL language shall not contain an <XML valid predicate> that contains <XML valid element clause>.

6) Without Feature X144, "IS VALID predicate: schema location", conforming SQL language shall not contain an <XML valid predicate> that contains <XML valid schema location>.

7) Without Feature X145, "IS VALID predicate outside check constraints", conforming SQL language shall not contain an <XML valid predicate> that is not directly contained in the <search condition> of a <check constraint definition>.

8) Without Feature X151, "IS VALID predicate: with DOCUMENT option", conforming SQL language shall not contain an <XML valid predicate> that immediately contains a <document or content or sequence> that is DOCUMENT.

9) Without Feature X152, "IS VALID predicate: with CONTENT option", conforming SQL language shall not contain an <XML valid predicate> that immediately contains a <document or content or sequence> that is CONTENT.

10) Without Feature X153, "IS VALID predicate: with SEQUENCE option", conforming SQL language shall not contain an <XML valid predicate> that immediately contains a <document or content or sequence> that is SEQUENCE.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

(Blank page)

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

9 Mappings

9.1 Mapping SQL <identifier>s to XML Names

Subclause Signature

```
"Mapping SQL <identifier>s to XML Names" [General Rules] (
Parameter: "IDENT"
)
```

Function

Define the mapping of SQL <identifier>s to XML Names.

Format

```
<uppercase hexit> ::=
  <digit> | A | B | C | D | E | F
```

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let *SQLI* be the *IDENT* in an application of this Subclause. *SQLI* is a sequence of characters of SQL_TEXT. Let *N* be the number of characters in *SQLI*. Let S_1, S_2, ..., S_N be the characters of *SQLI*, in order from left to right.

2) Let *EV* be the escape variant in an application of this Subclause. *EV* is either *partially escaped* or *fully escaped*.

3) Let *TM* be the implementation-defined mapping of the characters of SQL_TEXT to characters of Unicode.

NOTE 35 — Unicode scalar values in the ranges U+0000 through U+001F (inclusive), sometimes called the "C0 controls", and U+007F through U+009F (inclusive), sometimes called "delete" (U+007F) and the "C1 controls" (the remainder of that latter range) are not encoding of abstract characters in Unicode. Programs that conform to the Unicode Standard may treat these Unicode scalar values in exactly the same way as they treat the 7- and 8-bit equivalents in other protocols. Such usage

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

constitutes a higher-level protocol and is beyond the scope of the Unicode standard. These Unicode scalar values do not occur in XML Names, but may appear in other places in XML text.

4) For each *i* between 1 (one) and *N*, let T_i be the mapping of S_i to Unicode using *TM* and let X_i be the Unicode character string defined by the following rules.

Case:

a) If S_i has no mapping to Unicode (*i.e.*, $TM(S_i)$ is undefined), then X_i is implementation-defined.

b) If S_i is <colon>, then

Case:

i) If $i = 1$ (one), then let X_i be `_x003A_`.

ii) If *EV* is fully escaped, then let X_i be `_x003A_`.

iii) Otherwise, let X_i be T_i.

c) If $i \leq N-1$, S_i is <underscore>, and S_{i+1} is the lowercase letter "x", then let X_i be `_x005F_`.

d) If *EV* is fully escaped, $i = 1$ (one), $N \geq 3$, S_1 is either the uppercase letter "X" or the lowercase letter "x", S_2 is either the uppercase letter "M" or the lowercase letter "m", and S_3 is either the uppercase letter "L" or the lowercase letter "l", then

Case:

i) If S_1 is the lowercase letter "x", then let $\boldsymbol{X}_1$ be `_x0078_`.

ii) If S_1 is the uppercase letter "X", then let $\boldsymbol{X}_1$ be `_x0058_`.

e) If either of the following is true:

— The SQL-implementation supports Feature X211, "XML 1.1 support", and either T_i is not a valid XML 1.1 NameChar, or $i = 1$ (one) and T_1 is not a valid XML 1.1 NameStartChar

— The SQL-implementation does not support Feature X211, "XML 1.1 support", and either T_i is not a valid XML 1.0 NameChar, or $i = 1$ (one) and T_1 is not a valid XML 1.0 NameStartChar

then:

i) Let U_1, U_2, ..., U_8 be the eight <uppercase hexit>s such that T_i is U+$U_1U_2...U_8$ in the UCS-4 encoding.

ii) Case:

1) If $U_1 = 0$ (zero), $U_2 = 0$ (zero), $U_3 = 0$ (zero), and $U_4 = 0$ (zero), then let X_i be _x$U_5U_6U_7U_8$_.

NOTE 36 — This case implies that T_i has a UCS-2 encoding, which is U+$U_5U_6U_7U_8$.

2) Otherwise, let $\boldsymbol{X_i}$ be _x$U_3U_4U_5U_6U_7U_8$_.

f) Otherwise, let X_i be T_i.

NOTE 37 — That is, any character in *SQLI* that does not occasion a problem as a character in an XML 1.0 NCName or XML 1.1 NCName is simply copied into the result.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

5) Let ***XMLN*** be the XML Name that is the character string concatenation of X_1, X_2, ..., and X_N in order from left to right.

Conformance Rules

None.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

9.2 Mapping a multi-part SQL name to an XML Name

Function

Define the mapping of a sequence of SQL <identifier>s to an XML Name.

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let $SQLI_i$, 1 (one) ≤ i ≤ n be a sequence of n SQL <identifier>s provided for an application of this Subclause.

2) Let NP(*S*) be the mapping of a string *S* to a result string defined as follows:

   a) Let m be the number of characters in *S*. For each character S_j, 1 (one) ≤ j ≤ m, in *S*, let NPS_j be defined as follows.

      Case:

      i) If S_j is <period>, then NPS_j is `_x002E_`.

      ii) Otherwise, NPS_j is S_j.

   b) NP(*S*) is the concatenation of NPS_j, 1 (one) ≤ j ≤ m.

3) For each i between 1 (one) and n, the General Rules of Subclause 9.1, "Mapping SQL <identifier>s to XML Names", are applied with $SQLI_i$ as *IDENT*, using the fully escaped variant of the mapping, resulting in the XML name $\mathbf{XMLN}_i$.

4) Let ***XMLR*** be the result of:

   NP(***XMLN***$_1$) || '.' || NP(***XMLN***$_2$) || '.' || ... || NP(***XMLN***$_n$)

5) ***XMLR*** is the XML Name that is the result of this mapping.

Conformance Rules

None.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

9.3 Mapping XML Names to SQL <identifier>s

Function

Define the mapping of XML Names to SQL <identifier>s.

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let ***XMLN*** be an XML Name in an application of this Subclause. ***XMLN*** is a sequence of Unicode characters. Let *N* be the number of characters in ***XMLN***. Let X_1, X_2 , ..., X_N be the characters of ***XMLN*** in order from left to right.

2) Let the *N* Unicode character strings U_1, U_2, ..., U_N be defined as follows:

   If U_i, 1 (one) ≤ i ≤ N, has not yet been determined, then

   Case:

   a) If X_i = '_' (an <underscore>), and X_{i+1} = 'x', and each of X_{i+2}, X_{i+3}, X_{i+4}, and X_{i+5} are all <hexit>s, and X_{i+6} = '_', then

      Case:

      i) If the Unicode code point U+$X_{i+2}X_{i+3}X_{i+4}X_{i+5}$ is a Unicode assigned character *UC*, then let U_i be the character string of length 1 (one) whose character is *UC* and let U_{i+1}, U_{i+2}, U_{i+3}, U_{i+4}, U_{i+5}, and U_{i+6} be the zero-length string.

      ii) Otherwise, U_i, U_{i+1}, U_{i+2}, U_{i+3}, U_{i+4}, U_{i+5}, and U_{i+6} are implementation-defined.

   b) If X_i = '_' (an <underscore>), and X_{i+1} = 'x', and each of X_{i+2}, X_{i+3}, X_{i+4}, X_{i+5}, X_{i+6}, and X_{i+7}, are all <hexit>s, and X_{i+8} = '_', then

      Case:

      i) If the Unicode code point U+$X_{i+2}X_{i+3}X_{i+4}X_{i+5}X_{i+6}X_{i+7}$ is a Unicode assigned character *UC*, then let U_i be the character string of length 1 (one) whose character is *UC* and let U_{i+1}, U_{i+2}, U_{i+3}, U_{i+4}, U_{i+5}, U_{i+6}, U_{i+7}, and U_{i+8} be the zero-length string.

      ii) Otherwise, U_i, U_{i+1}, U_{i+2}, U_{i+3}, U_{i+4}, U_{i+5}, U_{i+6}, U_{i+7}, and U_{i+8} are implementation-defined.

   c) Otherwise, let U_i be the character string of length 1 (one) whose character is X_i.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3) Let *U* be the Unicode character string constructed by concatenating every U_i, 1 (one) ≤ *i* ≤ *N*, in order by *i*.

4) Let *SQLI* be the SQL_TEXT character string obtained by mapping the Unicode character string *U* to SQL_TEXT using the implementation-defined mapping of Unicode to SQL_TEXT. If *SQLI* can not be mapped to SQL_TEXT, then an exception condition is raised: *SQL/XML mapping error — unmappable XML Name*.

5) The SQL <identifier> that is the mapping of ***XMLN*** is the <delimited identifier> **"***SQLI***"**.

Conformance Rules

1) Without Feature X400, “Name and identifier mapping”, a conforming application shall not invoke this Subclause of this part of this International Standard.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

9.4 Mapping an SQL data type to an XML Name

Function

Define the mapping of an SQL data type or domain to an XML Name.

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let *D* be the SQL data type or the underlying data type of the domain provided for an application of this Subclause.

2) If *D* is a character string type, then:

   a) Let *SQLCS* be the character set of *D*.

   b) Let *N* be the length or maximum length of *D*.

   c) Let *CSM* be the implementation-defined mapping of strings of *SQLCS* to strings of Unicode. Let *MAXCSL* be the maximum length in characters of *CSM*(*S*), for all strings *S* of length *N* characters.

   d) Let ***MLIT*** be the canonical XML Schema literal of the XML Schema type **xs:integer** denoting *MAXCSL*.

   e) Let ***NLIT*** be the canonical XML Schema literal denoting *N* in the lexical representation of XML Schema type **xs:integer**.

   f) Case:

      i) If *CSM* is homomorphic, and *N* equals *MAXCSL*, then

         Case:

         1) If the type designator of *D* is CHARACTER, then let ***XMLN*** be the following:

            CHAR_*MLIT*

         2) If the type designator of *D* is CHARACTER VARYING, then let ***XMLN*** be the following:

            VARCHAR_*MLIT*

         3) If the type designator of *D* is CHARACTER LARGE OBJECT, then let ***XMLN*** be the following:

            CLOB_*MLIT*

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

ii) If *CSM* is homomorphic, and *N* does not equal *MAXCSL*, then

Case:

1) If the type designator of *D* is CHARACTER, then let ***XMLN*** be the following:

CHAR_*NLIT*_*MLIT*

2) If the type designator of *D* is CHARACTER VARYING, then let ***XMLN*** be the following:

VARCHAR_*NLIT*_*MLIT*

3) If the type designator of *D* is CHARACTER LARGE OBJECT, then let ***XMLN*** be the following:

CLOB_*NLIT*_*MLIT*

iii) Otherwise,

Case:

1) If the type designator of *D* is CHARACTER or CHARACTER VARYING, then let ***XMLN*** be the following:

VARCHAR_*NLIT*_*MLIT*

2) If the type designator of *D* is CHARACTER LARGE OBJECT, then let ***XMLN*** be the following:

CLOB_*NLIT*_*MLIT*

3) If the type designator of *D* is BINARY LARGE OBJECT, then:

a) Let *N* be the maximum length of *D*. Let ***XN*** be the canonical XML Schema literal denoting *N* in the lexical representation of XML Schema type **xs:integer**.

b) Let ***XMLN*** be the following:

BLOB_*XN*

4) If the type designator of *D* is NUMERIC, then:

a) Let *P* be the precision of *D*. Let ***XP*** be the canonical XML Schema literal denoting *P* in the lexical representation of XML Schema type **xs:integer**.

b) Let *S* be the scale of *D*. Let ***XS*** be the canonical XML Schema literal denoting *S* in the lexical representation of XML Schema type **xs:integer**.

c) Let ***XMLN*** be the following:

NUMERIC_*XP*_*XS*

5) If the type designator of *D* is DECIMAL, then:

a) Let *P* be the precision of *D*. Let ***XP*** be the canonical XML Schema literal denoting *P* in the lexical representation of XML Schema type **xs:integer**.

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

b) Let *S* be the scale of *D*. Let ***XS*** be the canonical XML Schema literal denoting *S* in the lexical representation of XML Schema type **xs:integer**.

c) Let ***XMLN*** be the following:

DECIMAL_*XP*_*XS*

6) If the type designator of *D* is INTEGER, then let ***XMLN*** be the following:

INTEGER

7) If the type designator of *D* is SMALLINT, then let ***XMLN*** be the following:

SMALLINT

8) If the type designator of *D* is BIGINT, then let ***XMLN*** be the following:

BIGINT

9) If the type designator of *D* is FLOAT, then:

a) Let *P* be the precision of *D*. Let ***XP*** be the canonical XML Schema literal denoting *P* in the lexical representation of XML Schema type **xs:integer**.

b) Let ***XMLN*** be the following:

FLOAT_***XP***

10) If the type designator of *D* is REAL, then let ***XMLN*** be the following:

REAL

11) If the type designator of *D* is DOUBLE PRECISION, then let ***XMLN*** be the following:

DOUBLE

12) If the type designator of *D* is BOOLEAN, then let ***XMLN*** be the following:

BOOLEAN

13) If the type designator of *D* is TIME WITHOUT TIME ZONE, then:

a) Let *TP* be the time precision of *D*. Let ***XTP*** be the canonical XML Schema literal denoting *TP* in the lexical representation of XML Schema type **xs:integer**.

b) Let ***XMLN*** be the following:

TIME_*XTP*

14) If the type designator of *D* is TIME WITH TIME ZONE, then:

a) Let *TP* be the time precision of *D*. Let ***XTP*** be the canonical XML Schema literal denoting *TP* in the lexical representation of XML Schema type **xs:integer**.

b) Let ***XMLN*** be the following:

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

TIME_WTZ_*XTP*

15) If the type designator of *D* is TIMESTAMP WITHOUT TIME ZONE, then:

a) Let *TSP* be the timestamp precision of *D*. Let ***XTSP*** be the canonical XML Schema literal denoting *TSP* in the lexical representation of XML Schema type **xs:integer**.

b) Let ***XMLN*** be the following:

TIMESTAMP_*XTSP*

16) If the type designator of *D* is TIMESTAMP WITH TIME ZONE, then:

a) Let *TSP* be the timestamp precision of *D*. Let ***XTSP*** be the canonical XML Schema literal denoting *TSP* in the lexical representation of XML Schema type **xs:integer**.

b) Let ***XMLN*** be the following:

TIMESTAMP_WTZ_*XTSP*

17) If the type designator of *D* is DATE, then let ***XMLN*** be the following:

DATE

18) If *D* is a domain, then let *C*, *S*, and *N* be the catalog name, schema name, and domain name of *D*, respectively. Let ***XMLN*** be the result of applying the mapping defined in Subclause 9.2, "Mapping a multi-part SQL name to an XML Name", to "Domain", *C*, *S*, and *N*.

19) If *D* is a row type, then let the XML Name ***IDI*** be an implementation-dependent identifier for the row type. Two row types that have different numbers of fields, different field names, or different declared types in corresponding fields shall have different values of ***IDI***. It is implementation-dependent whether the types of two sites of row type, having the same number of fields, and having corresponding fields of the same name and declared type, receive the same row type identifier. Let ***XMLN*** be **Row.*IDI***.

20) If *D* is a distinct type, then let *C*, *S*, and *N* be the catalog name, schema name, and type name of *D*, respectively. Let ***XMLN*** be the result of applying the mapping defined in Subclause 9.2, "Mapping a multi-part SQL name to an XML Name", to "UDT", *C*, *S*, and *N*.

21) If *D* is an array type, then let *ET* be the element type of *D* and let *M* be the maximum cardinality of *D*. Let ***XMLET*** be the result of applying this Subclause to *ET*. Let ***MLIT*** be the canonical XML Schema literal denoting *M* in the lexical representation of XML Schema type **xs:integer**. Let ***XMLN*** be **Array_*MLIT*.*XMLET***.

22) If *D* is a multiset type, then let *ET* be the element type of *D*. Let ***XMLET*** be the result of applying this Subclause to *ET*. Let ***XMLN*** be **Multiset.*XMLET***.

23) If *D* is an XML type, then let ***XMLN*** be **XML**.

24) ***XMLN*** is the XML Name that is result of this mapping.

Conformance Rules

None.

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

9.5 Mapping SQL data types to XML Schema data types

Function

Define the mapping of SQL data types and domains to XML Schema data types.

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let *SQLT* be *SQLTYPE*, the SQL data type or domain in an application of this Subclause.

2) Let *NULLS* be *NULLS*, the choice of whether to map null values to absent elements (absent) or to elements that are marked with **`xsi:nil="true"`** (nil).

   Case:

   a) If *NULLS* is absent, then let the XML text ***`XMLNULLS`*** be **`minOccurs="0"`**.

   b) If *NULLS* is nil, then let the XML text ***`XMLNULLS`*** be **`nillable="true"`**.

3) Let *ENCODING* be *ENCODING*, the choice of whether to encode binary strings in base64 or in hex.

4) Let *TM* be the implementation-defined mapping of character strings of SQL_TEXT to character strings of Unicode.

5) Let **`xs`** be the XML namespace prefix to be used to identify the XML Schema namespace as shown in Table 2, "XML namespace prefixes and their URIs".

6) Let **`sqlxml`** be the XML namespace prefix to be used to identify the XML namespace as shown in Table 2, "XML namespace prefixes and their URIs".

7) Let ***`XMLT`*** denote the representation of the XML Schema data type that is the mapping of *SQLT* into XML. ***`XMLT`*** is defined by the following rules.

8) Case:

   a) If *SQLT* is a character string type, then:

      i) Let *SQLCS* be the character set of *SQLT*. Let *SQLCSN* be the name of *SQLCS*. Let *N* be the length or maximum length of *SQLT*.

      ii) Let *CSM* be the implementation-defined mapping of strings of *SQLCS* to strings of Unicode. Let *MAXCSL* be the maximum length in characters of *CSM*(*S*), for all strings *S* of length *N* characters.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

iii) Let ***NLIT*** and ***MLIT*** be canonical XML Schema literals of the XML Schema type **xs:integer** denoting *N* and *MAXCSL*, respectively.

iv) Case:

1) If the type designator of *SQLT* is CHARACTER, then:

A) Case:

I) If *CSM* is homomorphic, then let ***FACET*** be the XML text:

<xs:length value="*MLIT*">

II) Otherwise, let ***FACET*** be the XML text:

<xs:maxLength value="*MLIT*">

B) It is implementation-defined whether the XML text ***ANNT*** is the zero-length string or given by:

name="CHAR"

C) It is implementation-defined whether the XML text ***ANNL*** is the zero-length string or given by:

length="*NLIT*"

2) If the type designator of *SQLT* is CHARACTER VARYING, then:

A) Let ***FACET*** be the XML text:

<xs:maxLength value="*MLIT*">

B) It is implementation-defined whether the XML text ***ANNT*** is the zero-length string or given by:

name="VARCHAR"

C) It is implementation-defined whether the XML text ***ANNL*** is the zero-length string or given by:

maxLength="*NLIT*"

3) If the type designator of *SQLT* is CHARACTER LARGE OBJECT, then:

A) Let ***FACET*** be the XML text:

<xs:maxLength value="*MLIT*">

B) It is implementation-defined whether the XML text ***ANNT*** is the zero-length string or given by:

name="CLOB"

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

C) It is implementation-defined whether the XML text ***ANNL*** is the zero-length string or given by:

```
maxLength="NLIT"
```

v) Let the XML text ***SQLCSNLIT*** be the result of mapping *SQLCSN* to Unicode using *TM*. It is implementation-defined whether the XML text ***ANNCS*** is the zero-length string or given by:

```
characterSetName="SQLCSNLIT"
```

vi) Let *SQLCON* be the name of the collation of *SQLT*. Let the XML text ***SQLCONLIT*** be the result of mapping *SQLCON* to Unicode using *TM*. It is implementation-defined whether the XML text ***ANNCO*** is the zero-length string or given by:

```
collation="SQLCONLIT"
```

vii) It is implementation-defined whether the XML text ***ANN*** is the zero-length string or given by:

```
<xs:annotation>
  <xs:appinfo>
    <sqlxml:sqltype kind="PREDEFINED"
                    ANNT ANNL ANNCS ANNCO/>
  </xs:appinfo>
</xs:annotation>
```

viii) ***XMLT*** is the XML Schema type defined by:

```
<xs:simpleType>
  ANN
  <xs:restriction base="xs:string">
    FACET
  </xs:restriction>
</xs:simpleType>
```

b) If *SQLT* is a binary string type, then:

i) Let *N* be the length or maximum length of *SQLT*. Let ***NLIT*** be an XML Schema literal denoting *N* in the lexical representation of the XML Schema type **xs:integer**.

ii) Case:

1) If *ENCODING* indicates that binary strings are to be encoded in hex, then let ***EN*** be the XML text **hexBinary**.

2) Otherwise, let ***EN*** be the XML text **base64Binary**.

iii) Let ***FACET*** be the XML text:

```
<xs:maxLength value="NLIT">
```

iv) It is implementation-defined whether the XML text ***ANNT*** is the zero-length string or given by:

```
name="BLOB"
```

v) It is implementation-defined whether the XML text ***ANNL*** is the zero-length string or given by:

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
maxLength="NLIT"
```

vi) It is implementation-defined whether the XML text ***ANN*** is the zero-length string or given by:

```
<xs:annotation>
  <xs:appinfo>
    <sqlxml:sqltype kind="PREDEFINED"
                    ANNT ANNL/>
  </xs:appinfo>
</xs:annotation>
```

vii) ***XMLT*** is the XML Schema type defined by:

```
<xs:simpleType>
  ANN
  <xs:restriction base="xs:EN">
    FACET
  </xs:restriction>
</xs:simpleType>
```

c) If the type designator of *SQLT* is NUMERIC or DECIMAL, then:

i) Let *P* be the precision of *SQLT*. Let ***PLIT*** be an XML Schema literal denoting *P* in the lexical representation of the XML Schema type **xs:integer**. Let ***FACETP*** be the XML text:

```
<xs:totalDigits value="PLIT"/>
```

ii) Let *S* be the scale of *SQLT*. Let ***SLIT*** be an XML Schema literal denoting *S* in the lexical representation of the XML Schema type **xs:integer**. Let ***FACETS*** be the XML text:

```
<xs:fractionDigits value="SLIT"/>
```

iii) Case:

1) If the type designator of *SQLT* is NUMERIC, then:

A) It is implementation-defined whether the XML text ***ANNT*** is the zero-length string or

```
name="NUMERIC"
```

B) It is implementation-defined whether the XML text ***ANNP*** is the zero-length string or:

```
precision="PLIT"
```

2) If the type designator of *SQLT* is DECIMAL, then:

A) It is implementation-defined whether the XML text ***ANNT*** is the zero-length string or:

```
name="DECIMAL"
```

B) Let *UP* be the value of the <precision> specified in the <data type> used to create the descriptor of *SQLT*. Let ***UPLIT*** be an XML Schema literal denoting *UP* in the lexical representation of the XML Schema type **xs:integer**. It is implementation-defined whether the XML text ***ANNP*** is the zero-length string or:

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
userPrecision="UPLIT"
```

NOTE 38 — *UP* may be less than *P*, as specified in SR 27) of Subclause 6.1, "<data type>", in [ISO9075-2].

iv) It is implementation-defined whether the XML text ***ANNS*** is the zero-length string or:

```
scale="SLIT"
```

v) It is implementation-defined whether the XML text ***ANN*** is the zero-length string or:

```
<xs:annotation>
  <xs:appinfo>
    <sqlxml:sqltype kind="PREDEFINED"
                    ANNT ANNP ANNS/>
  </xs:appinfo>
</xs:annotation>
```

vi) ***XMLT*** is the XML Schema type defined by:

```
<xs:simpleType>
  ANN
  <xs:restriction base="xs:decimal">
    FACETP
    FACETS
  </xs:restriction>
</xs:simpleType>
```

d) If the type designator of *SQLT* is INTEGER, SMALLINT, or BIGINT, then:

i) Let *MAX* be the maximum value representable by *SQLT*. Let ***MAXLIT*** be an XML Schema literal denoting *MAX* in the lexical representation of the XML Schema type **`xs:integer`**. Let ***FACETMAX*** be the XML text:

```
<xs:maxInclusive value="MAXLIT"/>
```

ii) Let *MIN* be the minimum value representable by *SQLT*. Let ***MINLIT*** be an XML Schema literal denoting *MIN* in the lexical representation of the XML Schema type **`xs:integer`**. Let ***FACETMIN*** be the XML text:

```
<xs:minInclusive value="MINLIT"/>
```

iii) Case:

1) If the type designator of *SQLT* is INTEGER, then it is implementation-defined whether the XML text ***ANN*** is the zero-length string or:

```
<xs:annotation>
  <xs:appinfo>
    <sqlxml:sqltype kind="PREDEFINED"
                    name="INTEGER"/>
  </xs:appinfo>
</xs:annotation>
```

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2) If the type designator of *SQLT* is SMALLINT, then it is implementation-defined whether the XML text ***ANN*** is the zero-length string or:

```
<xs:annotation>
  <xs:appinfo>
    <sqlxml:sqltype kind="PREDEFINED"
                    name="SMALLINT"/>
  </xs:appinfo>
</xs:annotation>
```

3) If the type designator of *SQLT* is BIGINT, then it is implementation-defined whether the XML text ***ANN*** is the zero-length string or:

```
<xs:annotation>
  <xs:appinfo>
    <sqlxml:sqltype kind="PREDEFINED"
                    name="BIGINT"/>
  </xs:appinfo>
</xs:annotation>
```

iv) It is implementation-defined whether *REST* is

```
<xs:restriction base="xs:integer">
  FACETMAX
  FACETMIN
</xs:restriction>
```

or determined by

1) Case:

A) If there is no row in Table 3, "Constraining facets of XML Schema integer types" such that *MAX* is less than or equal to the value in the "maxInclusive" column and *MIN* is greater than or equal to the value in the "minInclusive" column, then:

I) Let ***TYPE*** be `xs:integer`.

II) Let ***FMAX*** be *FACETMAX*.

III) Let ***FMIN*** be *FACETMIN*.

B) Otherwise:

I) Let ***TYPE*** be the contents of the "Type" column, in Table 3, "Constraining facets of XML Schema integer types", taken from the first row in the table for which *MAX* is less than or equal to the value in the "maxInclusive" column and *MIN* is greater than or equal to the value in the "minInclusive" column.

II) If *MAX* is equal to the value of the "maxInclusive", in the selected row of the table, then let ***FMAX*** be the zero-length string; otherwise, let ***FMAX*** be ***FACETMAX***.

III) If *MIN* is equal to the value of the "minInclusive", in the selected row of the table, then let ***FMIN*** be the zero-length string; otherwise, let ***FMIN*** be ***FACETMIN***.

2) Let *REST* be:

```
<xs:restriction base="TYPE">
```


© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
    FMAX
    FMIN
</xs:restriction>
```

Table 3 — Constraining facets of XML Schema integer types

Type	minInclusive	maxInclusive
xs:unsignedByte	0	255
xs:byte	-128	127
xs:unsigned-Short	0	2^{16}-1 (65,535)
xs:short	-2^{15} (-32,768)	2^{15}-1 (32,767)
xs:unsignedInt	0	2^{32}-1 (4,294,967,295)
xs:int	-2^{31} (-2,147,483,648)	2^{31}-1 (2,147,483,647)
xs:unsignedLong	0	2^{64}-1 (18,446,744,073,709,551,615)
xs:long	-2^{63} (-9,223,372,036,854,775,808)	2^{63}-1 (9,223,372,036,854,775,807)

v) ***XMLT*** is the XML Schema type defined by:

```
<xs:simpleType>
  ANN
  REST
</xs:simpleType>
```

e) If *SQLT* is approximate numeric, then:

i) Let *P* be the binary precision of *SQLT*, let *MINEXP* be the minimum binary exponent supported by *SQLT*, and let *MAXEXP* be the maximum binary exponent supported by *SQLT*.

ii) Case:

1) If *P* is less than or equal to 24 binary digits (bits), *MINEXP* is greater than or equal to -149, and *MAXEXP* is less than or equal to 104, then let the XML text ***TYPE*** be **float**.

2) Otherwise, let the XML text ***TYPE*** be **double**.

iii) Case:

1) If the type designator of *SQLT* is REAL, then the XML text ***ANNUP*** is the zero-length string, and it is implementation-defined whether the XML text ***ANNT*** is the zero-length string or:

```
name="REAL"
```

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2) If the type designator of *SQLT* is DOUBLE PRECISION, then the XML text ***ANNUP*** is the zero-length string, and it is implementation-defined whether the XML text ***ANNT*** is the zero-length string or:

```
name="DOUBLE PRECISION"
```

3) Otherwise:

A) It is implementation-defined whether the XML text ***ANNT*** is the zero-length string or:

```
name="FLOAT"
```

B) Let *UP* be the value of the <precision> specified in the <data type> used to create the descriptor of *SQLT*. Let ***UPLIT*** be an XML Schema literal denoting *UP* in the lexical representation of the XML Schema type **xs:integer**. It is implementation-defined whether the XML text ***ANNUP*** is the zero-length string or:

```
userPrecision="UPLIT"
```

NOTE 39 — *UP* may be less than *P*, as specified in SR 29) of Subclause 6.1, "<data type>", in [ISO9075-2].

iv) Let ***PLIT*** be an XML Schema literal denoting *P* in the lexical representation of the XML Schema type **xs:integer**. It is implementation-defined whether the XML text ***ANNP*** is the zero-length string or:

```
precision="PLIT"
```

v) Let ***MINLIT*** be an XML Schema literal denoting *MINEXP* in the lexical representation of the XML Schema type **xs:integer**. It is implementation-defined whether the XML text ***ANNMIN*** is the zero-length string or:

```
minExponent="MINLIT"
```

vi) Let ***MAXLIT*** be an XML Schema literal denoting *MAXEXP* in the lexical representation of the XML Schema type **xs:integer**. It is implementation-defined whether the XML text ***ANNMAX*** is the zero-length string or:

```
maxExponent="MAXLIT"
```

vii) It is implementation-defined whether the XML text ***ANN*** is the zero-length string or:

```
<xs:annotation>
  <xs:appinfo>
    <sqlxml:sqltype kind="PREDEFINED"
                    ANNT ANNP ANNUP ANNMAX ANNMIN/>
  </xs:appinfo>
</xs:annotation>
```

viii) It is implementation-defined whether ***XMLT*** is **xs:*TYPE*** or the XML Schema type defined by:

```
<xs:simpleType>
  ANN
  <xs:restriction base="xs:TYPE">
```


© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
    </xs:restriction>
  </xs:simpleType>
```

f) If the type designator of *SQLT* is BOOLEAN, then it is implementation-defined whether ***XMLT*** is **xs:boolean** or the XML Schema type defined by:

```
<xs:simpleType>
  <xs:annotation>
    <xs:appinfo>
      <sqlxml:sqltype kind="PREDEFINED"
                      name="BOOLEAN"/>
    </xs:appinfo>
  </xs:annotation>
  <xs:restriction base="xs:boolean"/>
</xs:simpleType>
```

g) If the type designator of *SQLT* is DATE, then:

i) It is implementation-defined whether the XML text ***ANN*** is the zero-length string or:

```
<xs:annotation>
  <xs:appinfo>
    <sqlxml:sqltype kind="PREDEFINED"
                    name="DATE"/>
  </xs:appinfo>
</xs:annotation>
```

ii) ***XMLT*** is the XML Schema type defined by:

```
<xs:simpleType>
  ANN
  <xs:restriction base="xs:date">
    <xs:pattern
      value="\p{Nd}{4}-\p{Nd}{2}-\p{Nd}{2}"/>
  </xs:restriction>
</xs:simpleType>
```

h) If *SQLT* is TIME WITHOUT TIME ZONE, then:

i) Let *S* be the <time fractional seconds precision> of *SQLT*. Let ***SLIT*** be an XML Schema literal denoting *S* in the lexical representation of XML Schema type **xs:integer**.

ii) Case:

1) If *S* is greater than 0 (zero), then let the XML text ***FACETP*** be:

```
<xs:pattern value=
    "\p{Nd}{2}:\p{Nd}{2}:\p{Nd}{2}.\p{Nd}{SLIT}"/>
```

2) Otherwise, let the XML text ***FACETP*** be:

```
<xs:pattern value=
    "\p{Nd}{2}:\p{Nd}{2}:\p{Nd}{2}"/>
```

iii) It is implementation-defined whether the XML text ***ANNT*** is the zero-length string or:

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
name="TIME"
```

iv) It is implementation-defined whether the XML text ***ANNS*** is the zero-length string or:

```
scale="SLIT"
```

v) It is implementation-defined whether the XML text ***ANN*** is the zero-length string or:

```
<xs:annotation>
  <xs:appinfo>
    <sqlxml:sqltype kind="PREDEFINED"
                    ANNT ANNS/>
  </xs:appinfo>
</xs:annotation>
```

vi) ***XMLT*** is the XML Schema type defined by:

```
<xs:simpleType>
  ANN
  <xs:restriction base="xs:time">
    FACETP
  </xs:restriction>
</xs:simpleType>
```

i) If *SQLT* is TIME WITH TIME ZONE, then:

i) Let *S* be the <time fractional seconds precision> of *SQLT*. Let ***SLIT*** be an XML Schema literal denoting *S* in the lexical representation of XML Schema type **xs:integer**.

ii) Let the XML text ***TZ*** be:

```
(+|-)\p{Nd}{2}:\p{Nd}{2}
```

iii) Case:

1) If *S* is greater than 0 (zero), then let the XML text ***FACETP*** be:

```
<xs:pattern value=
    "\p{Nd}{2}:\p{Nd}{2}:\p{Nd}{2}.\p{Nd}{SLIT}TZ"/>
```

2) Otherwise, let the XML text ***FACETP*** be:

```
<xs:pattern value=
    "\p{Nd}{2}:\p{Nd}{2}:\p{Nd}{2}TZ"/>
```

iv) It is implementation-defined whether the XML text ***ANNT*** is the zero-length string or:

```
name="TIME WITH TIME ZONE"
```

v) It is implementation-defined whether the XML text ***ANNS*** is the zero-length string or:

```
scale="SLIT"
```

vi) It is implementation-defined whether the XML text ***ANN*** is the zero-length string or:

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
<xs:annotation>
  <xs:appinfo>
    <sqlxml:sqltype kind="PREDEFINED"
                    ANNT ANNS/>
  </xs:appinfo>
</xs:annotation>
```

vii) ***XMLT*** is the XML Schema type defined by:

```
<xs:simpleType>
  ANN
  <xs:restriction base="xs:time">
    FACETP
  </xs:restriction>
</xs:simpleType>
```

j) If *SQLT* is TIMESTAMP WITHOUT TIME ZONE, then:

i) Let *S* be the <time fractional seconds precision> of *SQLT*. Let ***SLIT*** be an XML Schema literal denoting *S* in the lexical representation of XML Schema type **xs:integer**.

ii) Let the XML text ***DATETIME*** be:

```
\p{Nd}{4}-\p{Nd}{2}-\p{Nd}{2}T\p{Nd}{2}:\p{Nd}{2}:\p{Nd}{2}
```

iii) Case:

1) If *S* is greater than 0 (zero), then let the XML text ***FACETP*** be:

```
<xs:pattern value=
    "DATETIME.\p{Nd}{SLIT}"/>
```

2) Otherwise, let the XML text ***FACETP*** be:

```
<xs:pattern value=
    "DATETIME"/>
```

iv) It is implementation-defined whether the XML text ***ANNT*** is the zero-length string or:

```
name="TIMESTAMP"
```

v) It is implementation-defined whether the XML text ***ANNS*** is the zero-length string or:

```
scale="SLIT"
```

vi) It is implementation-defined whether the XML text ***ANN*** is the zero-length string or:

```
<xs:annotation>
  <xs:appinfo>
    <sqlxml:sqltype kind="PREDEFINED"
                    ANNT ANNS/>
  </xs:appinfo>
</xs:annotation>
```

vii) ***XMLT*** is the XML Schema type defined by:

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
<xs:simpleType>
  ANN
  <xs:restriction base="xs:dateTime">
    FACETP
  </xs:restriction>
</xs:simpleType>
```

k) If *SQLT* is TIMESTAMP WITH TIME ZONE, then:

i) Let *S* be the <time fractional seconds precision> of *SQLT*. Let ***SLIT*** be an XML Schema literal denoting *S* in the lexical representation of XML Schema type **xs:integer**.

ii) Let the XML text ***DATETIME*** be:

```
\p{Nd}{4}-\p{Nd}{2}-\p{Nd}{2}T\p{Nd}{2}:\p{Nd}{2}:\p{Nd}{2}
```

iii) Let the XML text ***TZ*** be:

```
(+|-)\p{Nd}{2}:\p{Nd}{2}
```

iv) Case:

1) If *S* is greater than 0 (zero), then let the XML text ***FACETP*** be:

```
<xs:pattern value=
    "DATETIME.\p{Nd}{SLIT}TZ"/>
```

2) Otherwise, let the XML text ***FACETP*** be:

```
<xs:pattern value="DATETIMETZ"/>
```

v) It is implementation-defined whether the XML text ***ANNT*** is the zero-length string or:

```
name="TIMESTAMP WITH TIME ZONE"
```

vi) It is implementation-defined whether the XML text ***ANNS*** is the zero-length string or:

```
scale="SLIT"
```

vii) It is implementation-defined whether the XML text ***ANN*** is the zero-length string or:

```
<xs:annotation>
  <xs:appinfo>
    <sqlxml:sqltype kind="PREDEFINED"
                    ANNT ANNS/>
  </xs:appinfo>
</xs:annotation>
```

viii) ***XMLT*** is the XML Schema type defined by:

```
<xs:simpleType>
  ANN
  <xs:restriction base="xs:dateTime">
    FACETP
```


© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
    </xs:restriction>
  </xs:simpleType>
```

l) If the type designator of *SQLT* is INTERVAL, then:

i) Let *P* be the <interval leading field precision> of *SQLT*. Let ***PLIT*** be an XML Schema literal for *P* in the XML Schema type **xs:integer**. It is implementation-defined whether the XML text ***ANNP*** is the zero-length string or:

```
leadingPrecision="PLIT"
```

ii) Case:

1) If the <end field> or <single datetime field> of *SQLT* specifies SECOND, then let *S* be the <interval fractional seconds precision> of *SQLT*, and let ***SLIT*** be an XML Schema literal for *S* in the XML Schema type **xs:integer**. Let the XML text ***SECS*** be:

```
\p{Nd}{2}.\p{Nd}{SLIT}S
```

It is implementation-defined whether the XML text ***ANNS*** is the zero-length string or:

```
scale="SLIT"
```

2) Otherwise, let the XML text ***ANNS*** be the zero-length string, and let the XML text ***SECS*** be:

```
\p{Nd}{2}S
```

iii) Case:

1) If *SQLT* is INTERVAL YEAR then:

A) Let the XML text ***FACETP*** be:

```
<xs:pattern value="-?P\p{Nd}{PLIT}Y"/>
```

B) It is implementation-defined whether the XML text ***ANNT*** is the zero-length string or:

```
name="INTERVAL YEAR"
```

2) If *SQLT* is INTERVAL YEAR TO MONTH then:

A) Let the XML text ***FACETP*** be:

```
<xs:pattern value="-?P\p{Nd}{PLIT}Y\p{Nd}{2}M"/>
```

B) It is implementation-defined whether the XML text ***ANNT*** is the zero-length string or:

```
name="INTERVAL YEAR TO MONTH"
```

3) If *SQLT* is INTERVAL MONTH then:

A) Let the XML text ***FACETP*** be:

```
<xs:pattern value="-?P\p{Nd}{PLIT}M"/>
```

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

B) It is implementation-defined whether the XML text ***ANNT*** is the zero-length string or:

```
name="INTERVAL MONTH"
```

4) If SQLT is INTERVAL DAY then:

A) Let the XML text ***FACETP*** be:

```
<xs:pattern value="-?P\p{Nd}{PLIT}D"/>
```

B) It is implementation-defined whether the XML text ***ANNT*** is the zero-length string or:

```
name="INTERVAL DAY"
```

5) If *SQLT* is INTERVAL DAY TO HOUR then:

A) Let the XML text ***FACETP*** be:

```
<xs:pattern value="-?P\p{Nd}{PLIT}DT\p{Nd}{2}H"/>
```

B) It is implementation-defined whether the XML text ***ANNT*** is the zero-length string or:

```
name="INTERVAL DAY TO HOUR"
```

6) If SQLT is INTERVAL DAY TO MINUTE then:

A) Let the XML text ***FACETP*** be:

```
<xs:pattern value=
  "-?P\p{Nd}{PLIT}DT\p{Nd}{2}H\p{Nd}{2}M"/>
```

B) It is implementation-defined whether the XML text ***ANNT*** is the zero-length string or:

```
name="INTERVAL DAY TO MINUTE"
```

7) If *SQLT* is INTERVAL DAY TO SECOND then:

A) Let the XML text ***FACETP*** be:

```
<xs:pattern value=
  "-?P\p{Nd}{PLIT}DT\p{Nd}{2}H\p{Nd}{2}MSECS"/>
```

B) It is implementation-defined whether the XML text ***ANNT*** is the zero-length string or:

```
name="INTERVAL DAY TO SECOND"
```

8) If SQLT is INTERVAL HOUR then:

A) Let the XML text ***FACETP*** be:

```
<xs:pattern value="-?PT\p{Nd}{PLIT}H"/>
```

B) It is implementation-defined whether the XML text ***ANNT*** is the zero-length string or:

```
name="INTERVAL HOUR"
```


© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

9) If *SQLT* is INTERVAL HOUR TO MINUTE then:

   A) Let the XML text ***FACETP*** be:

```
<xs:pattern value=
  "-?PT\p{Nd}{PLIT}H\p{Nd}{2}M"/>
```

   B) It is implementation-defined whether the XML text ***ANNT*** is the zero-length string or:

```
name="INTERVAL HOUR TO MINUTE"
```

10) If *SQLT* is INTERVAL HOUR TO SECOND then:

   A) Let the XML text ***FACETP*** be:

```
<xs:pattern value=
  "-?PT\p{Nd}{PLIT}H\p{Nd}{2}MSECS"/>
```

   B) It is implementation-defined whether the XML text ***ANNT*** is the zero-length string or:

```
name="INTERVAL HOUR TO SECOND"
```

11) If *SQLT* is INTERVAL MINUTE then:

   A) Let the XML text ***FACETP*** be:

```
<xs:pattern value="-?PT\p{Nd}{PLIT}M"/>
```

   B) It is implementation-defined whether the XML text ***ANNT*** is the zero-length string or:

```
name="INTERVAL MINUTE"
```

12) If *SQLT* is INTERVAL MINUTE TO SECOND then:

   A) Let the XML text ***FACETP*** be:

```
<xs:pattern value="-?PT\p{Nd}{PLIT}MSECS"/>
```

   B) It is implementation-defined whether the XML text ***ANNT*** is the zero-length string or:

```
name="INTERVAL MINUTE TO SECOND"
```

13) If *SQLT* is INTERVAL SECOND then:

   A) Let the XML text ***FACETP*** be:

```
<xs:pattern value="-?PTSECS"/>
```

   B) It is implementation-defined whether the XML text ***ANNT*** is the zero-length string or:

```
name="INTERVAL SECOND"
```

iv) It is implementation-defined whether the XML text ***ANN*** is the zero-length string or:

```
<xs:annotation>
```

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
  <xs:appinfo>
    <sqlxml:sqltype kind="PREDEFINED"
                    ANNT ANNP ANNS/>
  </xs:appinfo>
</xs:annotation>
```

v) Case:

1) If *SQLT* is a year-month duration, then let ***DTYPE*** be **xs:yearMonthDuration**.

2) If *SQLT* is a day-time duration, then let ***DTYPE*** be **xs:dayTimeDuration**.

vi) ***XMLT*** is the XML Schema type defined by:

```
<xs:simpleType>
  ANN
  <xs:restriction base="DTYPE">
    FACETP
  </xs:restriction>
</xs:simpleType>
```

m) If *SQLT* is a domain, then:

i) Let *DT* be the data type of *SQLT*.

ii) Let ***XMLN*** be the XML Name obtained by applying Subclause 9.4, "Mapping an SQL data type to an XML Name", to *DT*.

iii) Let *DC*, *DS*, and *DN* be the domain's catalog name, schema name, and domain name, respectively.

iv) Let *DCLIT*, *DSLIT*, and *DNLIT* be the result of mapping *DC*, *DS*, and *DN* to Unicode using *TM*.

v) It is implementation-defined whether the XML text ***ANN*** is the zero-length string or given by:

```
<xs:annotation>
  <xs:appinfo>
    <sqlxml:sqltype kind="DOMAIN"
                    catalogName="DCLIT" schemaName="DSLIT"
                    typeName="DNLIT" mappedType="XMLN"/>
  </xs:appinfo>
</xs:annotation>
```

vi) ***XMLT*** is the XML Schema type defined by:

```
<xs:simpleType>
  ANN
  <xs:restriction base="XMLN"/>
</xs:simpleType>
```

n) If *SQLT* is a row type, then:

i) Let *N* be the number of fields of *SQLT*. Let FT_i and FN_i be the declared type and name of the *i*-th field of *SQLT*, respectively, for *i* between 1 (one) and *N*.

ii) Let $\mathbf{XMLMT_i}$ be the result of applying the mapping in Subclause 9.4, "Mapping an SQL data type to an XML Name", to FT_i, for *i* between 1 (one) and *N*.

© ISO/IEC 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

iii) For each *i* between 1 (one) and *N*, the General Rules of Subclause 9.1, "Mapping SQL <identifier>s to XML Names", are applied with FN_i as *IDENT*, using the fully escaped variant of the mapping, resulting in $\boldsymbol{XMLFN_i}$.

iv) Let $\boldsymbol{FNLIT_i}$ be the result of mapping FN_i to Unicode using *TM*, for *i* between 1 (one) and *N*.

v) It is implementation-defined whether the XML text ***ANN*** is the zero-length string or given by:

```
<xs:annotation>
  <xs:appinfo>
    <sqlxml:sqltype kind="ROW">
      <sqlxml:field name="FNLIT1"
                    mappedType="XMLMT1"/>
      . . .
      <sqlxml:field name="FNLITN"
                    mappedType="XMLMTN"/>
    </sqlxml:sqltype>
  </xs:appinfo>
</xs:annotation>
```

vi) ***XMLT*** is the XML Schema type defined by:

```
<xs:complexType>
  ANN
  <xs:sequence>
    <xs:element name="XMLFN1" type="XMLMT1" XMLNULLS/>
    . . .
    <xs:element name="XMLFNN" type="XMLMTN" XMLNULLS/>
  </xs:sequence>
</xs:complexType>
```

o) If *SQLT* is a distinct type, then:

i) Let *ST* be the source type of *SQLT*.

ii) Let ***XMLN*** be the XML Name obtained by applying Subclause 9.4, "Mapping an SQL data type to an XML Name", to *ST*.

iii) Let *DTC*, *DTS*, and *DTN* be the catalog name, schema name, and type name, respectively, of *SQLT*.

iv) Let *DTCLIT*, *DTSLIT*, and *DTNLIT* be the result of mapping *DTC*, *DTS*, and *DTN* to Unicode using *TM*.

v) It is implementation-defined whether the XML text ***ANN*** is the zero-length string or given by:

```
<xs:annotation>
  <xs:appinfo>
    <sqlxml:sqltype kind="DISTINCT"
                    catalogName="DTCLIT" schemaName="DTSLIT"
                    typeName="DTNLIT" mappedType="XMLN"
                    final="true"/>
  </xs:appinfo>
</xs:annotation>
```

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

vi) ***XMLT*** is the XML Schema type defined by:

```
<xs:simpleType>
  ANN
  <xs:restriction base="XMLN"/>
</xs:simpleType>
```

p) If *SQLT* is an array type, then:

i) Let *ET* be the element type of *SQLT*, and let *M* be the maximum cardinality of *SQLT*.

ii) Let ***XMLN*** be the XML Name obtained by applying Subclause 9.4, "Mapping an SQL data type to an XML Name", to *ET*.

iii) Let ***MLIT*** be an XML Schema literal for *M* in the XML Schema type **xs:integer**.

iv) It is implementation-defined whether the XML text ***ANN*** is the zero-length string or given by:

```
<xs:annotation>
  <xs:appinfo>
    <sqlxml:sqltype kind="ARRAY"
                    maxElements="MLIT"
                    mappedElementType="XMLN"/>
  </xs:appinfo>
</xs:annotation>
```

v) ***XMLT*** is the XML Schema type defined by:

```
<xs:complexType>
  ANN
  <xs:sequence>
    <xs:element name="element" minOccurs="0"
                maxOccurs="MLIT" nillable="true"
                type="XMLN"/>
  </xs:sequence>
</xs:complexType>
```

q) If *SQLT* is a multiset type, then:

i) Let *ET* be the element type of *SQLT*.

ii) Let ***XMLN*** be the XML Name obtained by applying Subclause 9.4, "Mapping an SQL data type to an XML Name", to *ET*.

iii) It is implementation-defined whether the XML text ***ANN*** is the zero-length string or given by:

```
<xs:annotation>
  <xs:appinfo>
    <sqlxml:sqltype kind="MULTISET"
                    mappedElementType="XMLN"/>
  </xs:appinfo>
</xs:annotation>
```

iv) ***XMLT*** is the XML Schema type defined by:

```
<xs:complexType>
  ANN
```


© ISO/IEC 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
  <xs:sequence>
    <xs:element name="element" minOccurs="0"
                maxOccurs="unbounded" nillable="true"
                type="XMLN"/>
  </xs:sequence>
</xs:complexType>
```

r) If *SQLT* is an XML type other than XML(SEQUENCE), then:

i) It is implementation-defined whether the XML text ***ANN*** is the zero-length string or:

```
<xs:annotation>
  <xs:appinfo>
    <sqlxml:sqltype kind="PREDEFINED"
                    name="XML"/>
  </xs:appinfo>
</xs:annotation>
```

ii) ***XMLT*** is the XML Schema type defined by:

```
<xs:complexType mixed="true">
  ANN
  <xs:sequence>
    <xs:any name="element" minOccurs="0" maxOccurs="unbounded"
            processContents="skip"/>
  </xs:sequence>
</xs:complexType>
```

9) ***XMLT*** is the result of this mapping.

Conformance Rules

None.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

9.6 Mapping an SQL data type to a named XML Schema data type

Function

Define the mapping of an SQL data type or domain to an XML Schema data type.

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let *D* be the SQL data type or domain provided for an application of this subclause.

2) Let ***XMLN*** be the result of applying the mapping defined in Subclause 9.4, "Mapping an SQL data type to an XML Name", to *D*.

3) Let *NULLS* be the choice of whether to map null values to absent elements (absent) or to elements that are marked with **`xsi:nil="true"`** (nil).

4) Let *ENCODING* be the choice of whether to encode binary strings in base64 or in hex.

5) If *D* is a character string type, then:

   a) Let *SQLCS* be the character set of *D*.

   b) Let *N* be the length or maximum length of *D*.

   c) Let *CSM* be the implementation-defined mapping of strings of *SQLCS* to strings of Unicode. Let *MAXCSL* be the maximum length in characters of *CSM*(*S*), for all strings *S* of length *N* characters.

   d) Let ***MLIT*** be the canonical XML Schema literal of the XML Schema type **`xs:integer`** denoting *MAXCSL*.

   e) Case:

      i) If *CSM* is homomorphic, *N* equals *MAXCSL*, and the type designator of *D* is CHARACTER, then let ***SQLCDT*** be the following:

```
<xs:simpleType name="XMLN">
   <xs:restriction base="xs:string">
      <xs:length value="MLIT" />
   </xs:restriction>
</xs:simpleType>
```

      ii) Otherwise, let ***SQLCDT*** be the following:

```
<xs:simpleType name="XMLN">
```


© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
   <xs:restriction base="xs:string">
      <xs:maxLength value="MLIT" />
   </xs:restriction>
</xs:simpleType>
```

6) If *D* is a domain or a data type that is not a character string type, then:

   a) Let *ENC* be the choice of whether to encode binary strings in base64 or in hex, let *NC* be the choice of whether to map null values to absent elements or elements that are marked with **xsi:nil="true"**, and let ***XMLT*** be the XML Schema data type that is the result of applying the mapping defined in Subclause 9.5, "Mapping SQL data types to XML Schema data types", with *D* as *SQLTYPE*, *ENC* as *ENCODING*, and *NC* as *NULLS*.

   b) Case:

      i) If *D* is an XML type, then *XMLT* is of the form

```
<xs:complexType MIXED>
  XMLTC
</xs:complexType>
```

      where *XMLTC* is the string comprising the element content and *MIXED* is the string comprising the attribute of the element. Let *SQLDT* be the following:

```
<xs:complexType name="XMLN" MIXED>
  XMLTC
</xs:complexType>
```

      ii) If ***XMLT*** is of the form **<xs:complexType>*XMLTC*</xs:complexType>**, where ***XMLTC*** is the string comprising the element content, then let ***SQLCDT*** be the following:

```
<xs:complexType name="XMLN">
   XMLTC
</xs:complexType>
```

      iii) If ***XMLT*** is of the form **<xs:simpleType>*XMLTC*</xs:simpleType>**, where ***XMLTC*** is the string comprising the element content, then let ***SQLCDT*** be the following:

```
<xs:simpleType name="XMLN">
   XMLTC
</xs:simpleType>
```

      iv) Otherwise, let ***SQLCDT*** be the following:

```
<xs:simpleType name="XMLN">
   <xs:restriction base="XMLT" />
</xs:simpleType>
```

7) ***SQLCDT*** is the XML Schema data type that is the result of this mapping.

Conformance Rules

None.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

9.7 Mapping a collection of SQL data types to XML Schema data types

Function

Define the mapping of a collection of SQL data types and domains to XML Schema data types.

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let *NULLS* be the choice of whether to map null values to absent elements (absent), or whether to map them to elements that are marked with **`xsi:nil="true"`** (nil).

2) Let *ENCODING* be the choice of whether to encode binary strings in base64 or in hex.

3) Let *C* be the collection of SQL data types and domains provided for an application of this Subclause. *C* is augmented recursively as follows, until no more data types are added to *C*:

   a) If *DO* is a domain contained in *C* and the data type of *DO* is not contained in *C*, then the data type of *DO* is added to *C*.

   b) If *RT* is a row type contained in *C* and *F* is a field of *RT* whose declared type is not contained in *C*, then the declared type of *F* is added to *C*.

   c) If *DT* is a distinct type contained in *C* whose source type is not in *C*, then the source type of *DT* is added to *C*.

   d) If *CT* is a collection type contained in *C* whose element type is not in *C*, then the element type of *CT* is added to *C*.

4) Let *n* be the number of SQL data types and domains in *C*.

5) Let ***XMLD*** be the zero-length string. Let ***XMLTL*** be an empty list of XML Names.

6) For *i* ranging from 1 (one) to *n*:

   a) Let D_i be the *i*-th SQL data type or domain in *C*.

   b) Let $\boldsymbol{XMLN_i}$ be the result of applying the mapping defined in Subclause 9.4, "Mapping an SQL data type to an XML Name", to D_i.

   c) Let $\boldsymbol{XMLT_i}$ be the XML Schema data type that is the result of applying the mapping defined in Subclause 9.6, "Mapping an SQL data type to a named XML Schema data type", to D_i using *NULLS* as the choice of whether to map null values to absent elements or elements that are marked with

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

xsi:nil="true" and *ENCODING* as the choice of whether to encode binary strings in base64 or in hex.

d) Two XML Names are considered to be equivalent to each other if they have the same number of characters and the Unicode values of all corresponding characters are equal.

e) If ***XMLN*** $_i$ is not equivalent to the value of any XML Name in ***XMLTL***, then:

i) Let ***XMLD*** be:

XMLD || ***XMLT*** $_i$

ii) Append ***XMLN*** $_i$ to ***XMLTL***.

7) ***XMLD*** contains the XML Schema data types that are the result of this mapping.

Conformance Rules

None.

© ISO/IEC 2008 – All rights reserved

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

9.8 Mapping values of SQL data types to values of XML Schema data types

Function

Define the mapping of non-null values of SQL data types to XML.

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let *DV* be *the value of an SQL data type* in an application of this Subclause.

   Case:

   a) If *DV* is a value of a distinct type, then let *SQLT* be the source type of the distinct type, and let *SQLV* be the result of:

      ```
      CAST (DV AS SQLT)
      ```

   b) Otherwise, let *SQLT* be the most specific type of *DV* and let *SQLV* be *DV*.

2) Let *NULLS* be the choice of whether to map null values to absent elements (absent) or to elements that are marked with **`xsi:nil="true"`** (nil).

3) Let *ENCODING* be the choice of whether to encode binary strings in base64 or in hex.

4) Let *CHARMAPPING* be the choice of whether to replace certain characters ("&" (U+0026), "<" (U+003C), ">" (U+003E), and Carriage Return (U+000D)) with their character references (*True*) or not (*False*).

5) Let *NL* be the choice of whether to map null values to absent elements or to elements that are marked with **`xsi:nil="true"`**, let *ENC* be the choice of whether to encode binary strings in base64 or in hex, and let ***`XMLT`*** be the XML Schema data type that is the result of applying the mapping defined in Subclause 9.5, "Mapping SQL data types to XML Schema data types", to *SQLT* as *SQLTYPE*, *NL* as *NULLS* and *ENC* as *ENCODING*.

6) Let *M* be the implementation-defined maximum length of variable-length character strings.

7) If *SQLT* is not a binary string type, a character string type, a row type, a collection type, an interval type, or the XML type, then let *CV* be the result of

   ```
   CAST ( SQLV AS CHARACTER VARYING(M) )
   ```

8) Let *CSM* be the implementation-defined mapping of the default character set of CHARACTER VARYING to Unicode.

9) Case:

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

a) If *SQLT* is a character string type, then:

   i) Let *CS* be the character set of *SQLT*. Let ***XMLVRAW*** be the XML text that is the result of mapping *SQLV* to Unicode using the implementation-defined mapping of character strings of *CS* to Unicode.

   ii) Case:

      1) If the SQL-implementation supports Feature X211, "XML 1.1 support", then if any Unicode code point in ***XMLVRAW*** does not represent a valid XML 1.1 character, then an exception condition is raised: *SQL/XML mapping error — invalid XML character*.

      2) Otherwise, if any Unicode code point in ***XMLVRAW*** does not represent a valid XML 1.0 character, then an exception condition is raised: *SQL/XML mapping error — invalid XML character*.

   iii) Case:

      1) If *CHARMAPPING* is *True*, then let ***XMLV*** be ***XMLVRAW***, with each instance of "&" (U+0026) replaced by "&", each instance of "<" (U+003C) replaced by "<", each instance of ">" (U+003E) replaced by ">", and each instance of Carriage Return (U+000D) replaced by "
".

      2) Otherwise, let ***XMLV*** be ***XMLVRAW***.

b) If *SQLT* is a binary string type, then

   Case:

   i) If *ENCODING* indicates that binary strings are to be encoded in hex, then let ***XMLV*** be the hex encoding as defined by [Schema2] of *SQLV*.

   ii) Otherwise, let ***XMLV*** be the base64 encoding as defined by [Schema2] of *SQLV*.

c) If *SQLT* is a numeric type, then let ***XMLV*** be the result of mapping *CV* to Unicode using *CSM*.

d) If *SQLT* is a BOOLEAN, then let *TEMP* be the result of:

```
LOWER (CV)
```

   Let ***XMLV*** be the result of mapping *TEMP* to Unicode using *CSM*.

e) If *SQLT* is DATE, then let *TEMP* be the result of:

```
SUBSTRING (CV FROM 6 FOR 10)
```

   Let ***XMLV*** be the result of mapping *TEMP* to Unicode using *CSM*.

f) If *SQLT* specifies TIME, then:

   i) Let *P* be the <time fractional seconds precision> of *SQLT*.

   ii) If *P* is 0 (zero), then let *Q* be 0 (zero); otherwise, let *Q* be *P* + 1 (one).

   iii) If *SQLT* specifies WITH TIME ZONE, then let *Z* be 6; otherwise, let *Z* be 0 (zero).

   iv) Case:

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

1) If the SECOND field of *CV* is greater than or equal to 60, then it is implementation-defined whether an implementation-defined value is assigned to *TEMP*, or whether to raise an exception condition: *data exception — datetime field overflow*.

2) Otherwise, let *TEMP* be the result of:

```
SUBSTRING (CV FROM 6 FOR 8 + Q + Z)
```

v) Let ***XMLV*** be the result of mapping *TEMP* to Unicode using *CSM*.

g) If *SQLT* specifies TIMESTAMP, then:

i) Let *P* be the <timestamp fractional seconds precision> of *SQLT*.

ii) If *P* is 0 (zero), then let *Q* be 0 (zero); otherwise, let *Q* be *P* + 1 (one).

iii) If *SQLT* specifies WITH TIME ZONE, then let *Z* be 6; otherwise, let *Z* be 0 (zero).

iv) Case:

1) If the SECOND field of *CV* is greater than or equal to 60, then it is implementation-defined whether an implementation-defined value is assigned to *TEMP*, or whether to raise an exception condition: *data exception — datetime field overflow*.

2) Otherwise, let *TEMP* be the result of:

```
   SUBSTRING (CV FROM 11 FOR 10)
|| 'T'
|| SUBSTRING (CV FROM 22 FOR 8 + Q + Z)
```

v) Let ***XMLV*** be the result of mapping *TEMP* to Unicode using *CSM*.

h) If *SQLT* specifies INTERVAL, then:

i) If *SQLV* is negative, then let *SIGN* be `'-'` (a character string of length 1 (one) consisting of <minus sign>); otherwise, let *SIGN* be the zero-length string.

ii) Let *SQLVA* be ABS(*SQLV*).

iii) Let *CVA* be the result of:

```
CAST ( SQLVA AS CHARACTER VARYING(M) )
```

iv) Let *L* be the <interval leading field precision> of *SQLT*.

v) Let *P* be the <interval fractional seconds precision> of *SQLT*, if any.

vi) If *P* is 0 (zero), then let *Q* be 0 (zero); otherwise, let *Q* be *P* + 1 (one).

vii) Case:

1) If *SQLT* is INTERVAL YEAR, then let *TEMP* be the result of:

```
SIGN || 'P' || SUBSTRING (CVA FROM 10 FOR L) || 'Y'
```

2) If *SQLT* is INTERVAL YEAR TO MONTH, then let *TEMP* be the result of

```
SIGN || 'P'
```

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
|| SUBSTRING (CVA FROM 10 FOR L) || 'Y'
|| SUBSTRING (CVA FROM 11 + L FOR 2) || 'M'
```

3) If *SQLT* is INTERVAL MONTH, then let *TEMP* be the result of:

```
   SIGN || 'P'
|| SUBSTRING (CVA FROM 10 FOR L) || 'M'
```

4) If *SQLT* is INTERVAL DAY, then let *TEMP* be the result of:

```
   SIGN || 'P'
|| SUBSTRING (CVA FROM 10 FOR L) || 'D'
```

5) If *SQLT* is INTERVAL DAY TO HOUR, then let *TEMP* be the result of:

```
   SIGN || 'P'
|| SUBSTRING (CVA FROM 10 FOR L) || 'DT'
|| SUBSTRING (CVA FROM 11 + L FOR 2) || 'H'
```

6) If *SQLT* is INTERVAL DAY TO MINUTE, then let *TEMP* be the result of:

```
   SIGN || 'P'
|| SUBSTRING (CVA FROM 10 FOR L) || 'DT'
|| SUBSTRING (CVA FROM 11 + L FOR 2) || 'H'
|| SUBSTRING (CVA FROM 14 + L FOR 2) || 'M'
```

7) If *SQLT* is INTERVAL DAY TO SECOND, then let *TEMP* be the result of:

```
   SIGN || 'P'
|| SUBSTRING (CVA FROM 10 FOR L) || 'DT'
|| SUBSTRING (CVA FROM 11 + L FOR 2) || 'H'
|| SUBSTRING (CVA FROM 14 + L FOR 2) || 'M'
|| SUBSTRING (CVA FROM 17 + L FOR 2 + Q) || 'S'
```

8) If *SQLT* is INTERVAL HOUR, then let *TEMP* be the result of:

```
   SIGN || 'PT'
|| SUBSTRING (CVA FROM 10 FOR L) || 'H'
```

9) If *SQLT* is INTERVAL HOUR TO MINUTE, then let *TEMP* be the result of:

```
   SIGN || 'PT'
|| SUBSTRING (CVA FROM 10 FOR L) || 'H'
|| SUBSTRING (CVA FROM 11 + L FOR 2) || 'M'
```

10) If *SQLT* is INTERVAL HOUR TO SECOND, then let *TEMP* be the result of:

```
   SIGN || 'PT'
|| SUBSTRING (CVA FROM 10 FOR L) || 'H'
|| SUBSTRING (CVA FROM 11 + L FOR 2) || 'M'
|| SUBSTRING (CVA FROM 14 + L FOR 2 + Q) || 'S'
```

11) If *SQLT* is INTERVAL MINUTE, then let *TEMP* be the result of:

© ISO/IEC 2008 – All rights reserved
--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
   SIGN || 'PT'
|| SUBSTRING (CVA FROM 10 FOR L) || 'M'
```

12) If *SQLT* is INTERVAL MINUTE TO SECOND, then let *TEMP* be the result of:

```
   SIGN || 'PT'
|| SUBSTRING (CVA FROM 10 FOR L) || 'M'
|| SUBSTRING (CVA FROM 11 + L FOR 2 + Q) || 'S'
```

13) If *SQLT* is INTERVAL SECOND, then let *TEMP* be the result of:

```
   SIGN || 'PT'
|| SUBSTRING (CVA FROM 10 FOR L + Q) || 'S'
```

viii) Let ***XMLV*** be the result of mapping *TEMP* to Unicode using *CSM*.

i) If *SQLT* is a row type, then:

i) Let *N* be the number of fields of *SQLT*. For *i* between 1 (one) and *N*, let FT_i, FN_i, and FV_i be the declared type, name, and value of the *i*-th field, respectively.

ii) For each *i* between 1 (one) and *N*:

1) The General Rules of Subclause 9.1, "Mapping SQL <identifier>s to XML Names", are applied with FN_i as *IDENT*, using the fully escaped variant of the mapping, resulting in $\boldsymbol{XMLFN_i}$.

2) Case:

A) If FV_i is the null value and *NULLS* is absent, then let $\boldsymbol{XMLE_i}$ be the empty string.

B) If FV_i is the null value and *NULLS* is nil, then let $\boldsymbol{XMLE_i}$ be the XML element:

$\boldsymbol{<XMLFN_i\ xsi:nil="true"/>}$

C) Otherwise, let the XML text $\boldsymbol{XMLV_i}$ be the result of applying the mapping defined in this Subclause to FV_i. Let $\boldsymbol{XMLE_i}$ be the XML element:

$\boldsymbol{<XMLFN_i>XMLV_i</XMLFN_i>}$

iii) Let ***XMLV*** be:

$\boldsymbol{XMLE_1\ XMLE_2\ \ldots\ XMLE_N}$

j) If *SQLV* is an array value or a multiset value, then:

i) Let *N* be the number of elements in *SQLV*.

ii) Let *ET* be the element type of *SQLV*.

iii) For *i* between 1 (one) and *N*:

1) Let E_i be the value of the *i*-th element of *SQLV*. (If *SQLV* is a multiset value, then the ordering of the elements is implementation-dependent.)

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2) Case:

A) If E_i is the null value, then let the XML text ***XMLE*** $_i$ be **<element xsi:nil="true"/>**.

B) Otherwise, let ***X*** $_i$ be the result of applying this Subclause to E_i. Let the XML text ***XMLE*** $_i$ be:

<element>*X* $_i$**</element>**

iv) Let ***XMLV*** be:

XMLE $_1$ ***XMLE*** $_2$ ... ***XMLE*** $_N$

k) If *SQLT* is an XML type, then let ***XMLV*** be the serialized value of the XML value in *SQLV*:

```
XMLSERIALIZE (CONTENT SQLV AS CLOB)
```

10) ***XMLV*** is the result of this mapping.

Conformance Rules

None.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

9.9 Mapping an SQL table to XML Schema data types

Function

Define the mapping of an SQL table to XML Schema data types.

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let *T* be the table provided for an application of this mapping. Let *NULLS* be the choice of whether to map null values to absent elements (absent), or whether to map them to elements that are marked with **`xsi:nil="true"`** (nil). Let *TABLEFOREST* be the choice of whether to map the table to a sequence of XML elements (*True*) or to map the table to an XML document with a single root element (*False*). Let *U* be the authorization identifier that is invoking this mapping.

2) Let *TC*, *TS*, and *TN* be the <catalog name>, <unqualified schema name>, and <qualified identifier> of the <table name> of *T*, respectively.

3) The General Rules of Subclause 9.1, "Mapping SQL <identifier>s to XML Names", are applied with *TN* as *IDENT*, using the fully escaped variant of the mapping, resulting in ***XMLTN***.

4) Let *n* be the number of XML visible columns of *T* for *U*.

   NOTE 40 — "XML visible column" is defined in Subclause 4.10.8, "Visibility of columns, tables, and schemas in mappings from SQL to XML".

5) For *i* ranging from 1 (one) to *n*:

   a) Let C_i be the *i*-th XML visible column of *T* for *U* in order of its ordinal position within *T*.

   b) Let *CN* be the <column name> of C_i. Let *D* be the data type of C_i.

   c) Let ***XMLCN*** be the result of applying the mapping defined in Subclause 9.1, "Mapping SQL <identifier>s to XML Names", to *CN* using the fully escaped variant of the mapping.

   d) Let ***XMLCTN*** be the result of applying the mapping defined in Subclause 9.4, "Mapping an SQL data type to an XML Name", to *D*.

   e) Case:

      i) If C_i is known not nullable, then let ***XMLNULLS*** be the zero-length string.

      ii) Otherwise,

          Case:

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

1) If *NULLS* is absent, then let ***XMLNULLS*** be

```
minOccurs="0"
```

2) If *NULLS* is nil, then let ***XMLNULLS*** be

```
nillable="true"
```

f) Case:

i) If *D* is a character string type, then:

1) Let *CS* be the character set of *D*.

2) Let *CSC*, *CSS*, and *CSN* be the <catalog name>, <unqualified schema name>, and <SQL language identifier> of the <character set name> of *CS*, respectively.

3) Let ***XMLCSCN***, ***XMLCSSN***, and ***XMLCSN*** be the result of applying the mapping defined in Subclause 9.1, "Mapping SQL <identifier>s to XML Names", to *CSC*, *CSS*, and *CSN*, respectively, using the fully escaped variant of the mapping.

4) Let *CO* be the collation of *D*.

5) Let *COC*, *COS*, and *CON* be the <catalog name>, <unqualified schema name>, and <qualified identifier> of the <collation name> of *CO*, respectively.

6) Let ***XMLCOCN***, ***XMLCOSN***, and ***XMLCON*** be the result of applying the mapping defined in Subclause 9.1, "Mapping SQL <identifier>s to XML Names", to *COC*, *COS*, and *CON*, respectively, using the fully escaped variant of the mapping.

7) It is implementation-defined whether ***COLANN*** is the zero-length string or

```
<xs:annotation>
   <xs:appinfo>
      <sqlxml:sqlname
         type="CHARACTER SET"
         catalogName="XMLCSCN"
         schemaName="XMLCSSN"
         localName="XMLCSN" />
      <sqlxml:sqlname
         type="COLLATION"
         catalogName="XMLCOCN"
         schemaName="XMLCOSN"
         localName="XMLCON" />
   </xs:appinfo>
</xs:annotation>
```

ii) Otherwise, let ***COLANN*** be the zero-length string.

g) Let $\mathbf{XMLCE}_i$ be

```
<xs:element name="XMLCN" type="XMLCTN" XMLNULLS>
   COLANN
</xs:element>
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

6) Let ***XMLTYPEN*** be the result of applying the mapping defined in Subclause 9.2, "Mapping a multi-part SQL name to an XML Name", to "TableType", *TC*, *TS*, and *TN*.

7) Let ***XMLROWN*** be the result of applying the mapping defined in Subclause 9.2, "Mapping a multi-part SQL name to an XML Name", to "RowType", *TC*, *TS*, and *TN*.

8) Let ***XMLCN***, ***XMLSN***, and ***XMLTN*** be the result of applying the mapping defined in Subclause 9.1, "Mapping SQL <identifier>s to XML Names", to *TC*, *TS*, and *TN*, respectively, using the fully escaped variant of the mapping.

9) If *T* is a base table, then let ***TYPE*** be **BASE TABLE**; otherwise, let ***TYPE*** be **VIEWED TABLE**. It is implementation-dependent whether ***SQLANN*** is the zero-length string or

```
<xs:annotation>
   <xs:appinfo>
      <sqlxml:sqlname
          type="TYPE"
          catalogName="XMLCN"
          schemaName="XMLSN"
          localName="XMLTN" />
   </xs:appinfo>
</xs:annotation>
```

10) Case:

a) If *TABLEFOREST* is *False*, then let ***XMLTYPMAP*** be:

```
<xs:complexType name="XMLROWN">
   <xs:sequence>
      XMLCE1
      ...
      XMLCEn
   </xs:sequence>
</xs:complexType>
<xs:complexType name="XMLTYPEN">
   SQLANN
   <xs:sequence>
      <xs:element name="row"
         type="XMLROWN"
         minOccurs="0"
         maxOccurs="unbounded" />
   </xs:sequence>
</xs:complexType>
```

b) If *TABLEFOREST* is *True*, then let ***XMLTYPMAP*** be:

```
<xs:complexType name="XMLROWN">
   <xs:sequence>
      XMLCE1
      ...
      XMLCEn
   </xs:sequence>
</xs:complexType>
<xs:element name="XMLTN" type="XMLROWN" />
```

11) ***XMLTYPMAP*** contains the XML Schema data types that are the result of this mapping.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Conformance Rules

None.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

9.10 Mapping an SQL table to an XML element or a sequence of XML elements

Function

Define the mapping of an SQL table to XML.

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let *T* be the table provided for an application of this mapping. Let *NULLS* be the choice of whether to map null values to absent elements (absent) or to elements that are marked with **`xsi:nil="true"`** (nil). Let *TABLEFOREST* be the choice of whether to map the table to a sequence of XML elements (*True*) or to an XML document with a single root element (*False*). Let ***TARGETNS*** be the XML target namespace URI of the XML Schema and data to be mapped. If ***TARGETNS*** is the zero-length string, then no XML namespace is added. Let *U* be the authorization identifier that is invoking this mapping. Let *ENCODING* be the choice of whether to encode binary strings in base64 or in hex.

2) Let *TN* be the <qualified identifier> of the <table name> of *T*.

3) Let ***XMLTN*** be the result of applying the mapping defined in Subclause 9.1, "Mapping SQL <identifier>s to XML Names", to *TN* using the fully escaped variant of the mapping.

4) Let *n* be the number of rows of *T* and let *m* be the number of XML visible columns of *T* for *U*.

   NOTE 41 — "XML visible column" is defined in Subclause 4.10.8, "Visibility of columns, tables, and schemas in mappings from SQL to XML".

5) Let ***XSINS*** be the value of the XML namespace URI provided for the XML namespace prefix **`xsi`** in Table 2, "XML namespace prefixes and their URIs".

   Case:

   a) If *NULLS* is absent, then let ***XSI*** be the zero-length string.

   b) If *NULLS* is nil, let ***XSI*** be

      `xmlns:xsi="`*`XSINS`*`"`

6) For *i* ranging from 1 (one) to *n*:

   a) Let R_i be the *i*-th row of *T*, in the implementation-dependent repeatable ordering of *T*.

   b) For *j* ranging from 1 (one) to *m*:

      i) Let C_j be the *j*-th XML visible column of *T* for *U*.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ii) Let CN_j be the <column name> of C_j.

iii) Let ***XMLCN$_j$*** be the result of applying the mapping defined in Subclause 9.1, "Mapping SQL <identifier>s to XML Names", to CN_j using the fully escaped variant of the mapping.

iv) Let V_j be the value of C_j for R_i.

v) Case:

1) If V_j is the null value and *NULLS* is absent, then ***XMLC$_j$*** is the zero-length string.

2) If V_j is the null value and *NULLS* is nil, then ***XMLC$_j$*** is

```
<XMLCNj xsi:nil="true" />
```

3) Otherwise:

A) Let ***XMLV$_j$*** be the result of applying the mapping defined in Subclause 9.8, "Mapping values of SQL data types to values of XML Schema data types", to V_j as *the value of an SQL data type*, *ENCODING* as *ENCODING*, *NULLS* as *NULLS*, and *True* as *CHARMAPPING*.

B) ***XMLC$_j$*** is

```
<XMLCNj>XMLVj</XMLCNj>
```

c) Case:

i) If *TABLEFOREST* is *False*, then let ***XMLR$_i$*** be

```
<row>
   XMLC1
   ...
   XMLCm
</row>
```

ii) If *TABLEFOREST* is *True* and ***TARGETNS*** is the zero-length string, then let ***XMLR$_i$*** be

```
<XMLTN XSI>
   XMLC1
   ...
   XMLCm
</XMLTN>
```

iii) If *TABLEFOREST* is *True* and ***TARGETNS*** is not the zero-length string, then let ***XMLR$_i$*** be

```
<XMLTN XSI xmlns="TARGETNS">
   XMLC1
   ...
   XMLCm
</XMLTN>
```

7) Case:

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

a) If *TABLEFOREST* is *False* and ***TARGETNS*** is the zero-length string, then let ***XMLTE*** be:

```
<XMLTN>
   XMLR1
   ...
   XMLRn
</XMLTN>
```

b) If *TABLEFOREST* is *False* and ***TARGETNS*** is not the zero-length string, then let ***XMLTE*** be:

```
<XMLTN xmlns="TARGETNS">
   XMLR1
   ...
   XMLRn
</XMLTN>
```

c) If *TABLEFOREST* is *True*, then let ***XMLTE*** be:

```
   XMLR1
   ...
   XMLRn
```

8) ***XMLTE*** is the XML that is the result of the application of this Subclause.

Conformance Rules

None.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

9.11 Mapping an SQL table to XML and an XML Schema document

Function

Define the mapping of an SQL table to an XML Schema document that describes the structure of the mapped XML, and either an XML document or a sequence of XML elements.

Syntax Rules

1) Let *T* be the table provided for an application of this mapping. Let *NULLS* be the choice of whether to map null values to absent elements (absent) or to elements that are marked with **`xsi:nil="true"`** (nil). Let *TABLEFOREST* be the choice of whether to map the table to a sequence of XML elements (*True*) or to an XML document with a single XML element (*False*). Let ***TARGETNS*** be the XML target namespace URI of the XML Schema and data to be mapped. If ***TARGETNS*** is the zero-length string, then no XML namespace is added. Let *DATA* be the choice of whether the mapping results in an XML representation of the data of *T* (*True*) or not (*False*). Let *METADATA* be the choice of whether the mapping results in an XML Schema document that describes the XML representation of the data of *T* (*True*) or not (*False*). Let *U* be the authorization identifier that is invoking this mapping. Let *ENCODING* be the choice of whether binary strings are to be encoded in base64 or in hex.

2) At least one of *DATA* and *METADATA* shall be *True*.

Access Rules

None.

General Rules

1) Let *TC*, *TS*, and *TN* be the <catalog name>, <unqualified schema name>, and <qualified identifier> of the <table name> of *T*, respectively.

2) Let ***XMLTN*** be the result of applying the mapping defined in Subclause 9.1, "Mapping SQL <identifier>s to XML Names", to *TN* using the fully escaped variant of the mapping.

3) If any of the visible columns of *T* is an XML unmappable column, then a completion condition is raised: *warning — column cannot be mapped to XML*.

4) Let ***XMLTYPEN*** be the result of applying the mapping defined in Subclause 9.2, "Mapping a multi-part SQL name to an XML Name", to "TableType", *TC*, *TS*, and *TN*.

5) Let *CT* be the XML visible columns of *T* for *U*. Let ***XSCT*** be the result of applying the mapping defined in Subclause 9.7, "Mapping a collection of SQL data types to XML Schema data types", to the data types and domains of the columns of *CT* using *NULLS* as the choice of whether to map null values to absent elements or to elements that are marked with **`xsi:nil="true"`**, and *ENCODING* as the choice of whether binary strings are to be encoded using base64 or using hex.

   NOTE 42 — "XML visible column" is defined in Subclause 4.10.8, "Visibility of columns, tables, and schemas in mappings from SQL to XML".

6) Let ***XST*** be the result of applying the mapping defined in Subclause 9.9, "Mapping an SQL table to XML Schema data types", to *T* using *NULLS* as the choice of whether to map null values to absent elements or

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

elements that are marked with **xsi:nil="true"**, *TABLEFOREST* as the choice of how to map *T*, and *U* as the invoker of this mapping.

7) Case:

   a) If either ***XSCT*** or ***XST*** contains an XML 1.1 QName that is not an XML 1.0 QName, then let *VER* be 1.1.

   b) Otherwise, it is implementation-defined whether *VER* is 1.1 or 1.0.

8) The encoding to use for the result(s) of this mapping is implementation-defined. If this encoding is not UTF-8 or UTF-16, then let *ENCODING* be the name of the encoding, and let ***ENC*** be

   encoding = '*ENCODING*'

   Otherwise, let ***ENC*** be the zero-length string.

9) If *VER* is not **1.0** or ***ENC*** is not the zero-length string, then let ***XMLDECL*** be

   <?xml version='*VER*' *ENC* ?>

10) If *METADATA* is *True*, then:

   a) Let ***SQLXMLNS*** be the value of the XML namespace URI provided for the XML namespace prefix **sqlxml** in Table 2, "XML namespace prefixes and their URIs".

   b) Let ***XSDNS*** be the value of the XML namespace URI provided for the XML namespace prefix **xs** in Table 2, "XML namespace prefixes and their URIs".

   c) Let ***SLOCVAL*** be an implementation-defined URI that references the XML Schema document describing the XML namespace ***SQLXMLNS***. It is implementation-defined whether ***SLOC*** is the zero-length string or

      schemaLocation="*SLOCVAL*"

      Case:

      i) If neither ***XSCT*** nor ***XST*** uses the XML namespace that corresponds to the **sqlxml** XML namespace prefix, then let ***XSDIMP*** be the zero-length string.

      ii) If at least one of ***XSCT*** and ***XST*** uses the XML namespace that corresponds to the ***sqlxml*** XML namespace prefix, then let ***XSDIMP*** be

      <xs:import namespace="*SQLXMLNS*" *SLOC* />

   d) Case:

      i) If neither ***XSCT*** nor ***XST*** uses the XML namespace that corresponds to the **sqlxml** XML namespace prefix, then let ***XSDECL*** be the zero-length string.

      ii) If at least one of ***XSCT*** and ***XST*** uses the XML namespace that corresponds to the ***sqlxml*** XML namespace prefix, then let ***XSDECL*** be

      xmlns:sqlxml="*SQLXMLNS*"

   e) Case:

© ISO/IEC 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

i) If *TABLEFOREST* is *False* and ***TARGETNS*** is not the zero-length string, then ***XS*** has the following contents:

```
XMLDECL
<xs:schema
      xmlns:xs="XSDNS"
      XSDECL
      targetNamespace="TARGETNS"
      elementFormDefault="qualified">
   XSDIMP
   XSCT
   XST
   <xs:element name="XMLTN" type="XMLTYPEN" />
</xs:schema>
```

ii) If *TABLEFOREST* is *False* and ***TARGETNS*** is the zero-length string, then ***XS*** has the following content:

```
XMLDECL
<xs:schema
      xmlns:xs="XSDNS"
      XSDECL>
   XSDIMP
   XSCT
   XST
   <xs:element name="XMLTN" type="XMLTYPEN" />
</xs:schema>
```

iii) If *TABLEFOREST* is *True* and ***TARGETNS*** is not the zero-length string, then ***XS*** has the following contents:

```
XMLDECL
<xs:schema
      xmlns:xs="XSDNS"
      XSDECL
      targetNamespace="TARGETNS"
      elementFormDefault="qualified">
   XSDIMP
   XSCT
   XST
</xs:schema>
```

iv) If *TABLEFOREST* is *True* and ***TARGETNS*** is the zero-length string, then ***XS*** has the following content:

```
XMLDECL
<xs:schema
      xmlns:xs="XSDNS"
      XSDECL>
   XSDIMP
   XSCT
   XST
</xs:schema>
```

f) Let ***XSR*** be:

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
XMLSERIALIZE ( DOCUMENT
               XMLPARSE ( DOCUMENT 'XS' PRESERVE WHITESPACE )
               AS CLOB )
```

11) Case:

a) If *METADATA* is *True*, then let **XSL** be the URI that identifies **XSR**.

Case:

i) If **TARGETNS** is the zero-length string, then let **XSLA** be

xsi:noNamespaceSchemaLocation="XSL"

ii) Otherwise, let **XSLA** be

xsi:schemaLocation="TARGETNS XSL"

b) Otherwise, let **XSLA** be the zero-length string.

12) If *DATA* is *True*, then:

a) Let **XDROWS** be the result of applying the mapping defined in Subclause 9.10, "Mapping an SQL table to an XML element or a sequence of XML elements", to *T* using *NULLS* as the choice of whether to map null values to absent elements or elements that are marked with **xsi:nil="true"**, *TABLEFOREST* as the choice of how to map the table, **TARGETNS** indicating the XML target namespace, *U* as the invoker of this mapping, and *ENCODING* as the choice of whether binary strings are to be encoded using base64 or using hex.

b) Let **XSINS** be the value of the XML namespace URI provided for the XML namespace prefix **xsi** in Table 2, "XML namespace prefixes and their URIs".

c) Case:

i) If *TABLEFOREST* is *False* and **TARGETNS** is the zero-length string, then **XD** has the following contents:

```
XMLDECL
<XMLTN
   xmlns:xsi="XSINS"
   XSLA>
   XDROWS
</XMLTN>
```

ii) If *TABLEFOREST* is *False* and **TARGETNS** is not the zero-length string, then **XD** has the following contents:

```
XMLDECL
<XMLTN
   xmlns:xsi="XSINS"
   xmlns:xs="TARGETNS"
   XSLA>
   XDROWS
</XMLTN>
```


© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

iii) If *TABLEFOREST* is *True*, then ***XD*** is ***XDROWS***.

d) Case:

i) If *TABLEFOREST* is *True*, then let ***XDR*** be:

```
XMLSERIALIZE ( CONTENT
               XMLPARSE ( CONTENT 'XD' PRESERVE WHITESPACE )
               AS CLOB )
```

ii) If *TABLEFOREST* is *False*, then let *XDR* be:

```
XMLSERIALIZE ( DOCUMENT
               XMLPARSE ( DOCUMENT 'XD' PRESERVE WHITESPACE )
               AS CLOB )
```

13) If *DATA* is *True*, then the value of *XDR* is the XML representation of the data of *T*. If *METADATA* is *True*, then ***XSR*** is the XML Schema document that describes the XML representation of the data of *T*.

Conformance Rules

1) Without Feature X040, "Basic table mapping", a conforming application shall not invoke this Subclause of this part of this International Standard.

2) Without Feature X041, "Basic table mapping: nulls absent", a conforming application shall not invoke this Subclause of this part of this International Standard with *NULLS* set to indicate that nulls are mapped to elements that are marked to absent elements.

3) Without Feature X042, "Basic table mapping: null as nil", a conforming application shall not invoke this Subclause of this part of this International Standard with *NULLS* set to indicate that nulls are mapped to elements that are marked with **xsi:nil="true"**.

4) Without Feature X043, "Basic table mapping: table as forest", a conforming application shall not invoke this Subclause of this part of this International Standard with *TABLEFOREST* set to *True*.

5) Without Feature X044, "Basic table mapping: table as element", a conforming application shall not invoke this Subclause of this part of this International Standard with *TABLEFOREST* set to *False*.

6) Without Feature X045, "Basic table mapping: with target namespace", a conforming application shall not invoke this Subclause of this part of this International Standard with *TARGETNS* that is not a zero-length string.

7) Without Feature X046, "Basic table mapping: data mapping", a conforming application shall not invoke this Subclause of this part of this International Standard with *DATA* set to *True*.

8) Without Feature X047, "Basic table mapping: metadata mapping", a conforming application shall not invoke this Subclause of this part of this International Standard with *METADATA* set to *True*.

9) Without Feature X048, "Basic table mapping: base64 encoding of binary strings", a conforming application shall not invoke this Subclause of this part of this International Standard with *ENCODING* set to indicate that binary strings are to be encoded using base64.

10) Without Feature X049, "Basic table mapping: hex encoding of binary strings", a conforming application shall not invoke this Subclause of this part of this International Standard with *ENCODING* set to indicate that binary strings are to be encoded using hex.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

9.12 Mapping an SQL schema to XML Schema data types

Function

Define the mapping of an SQL-schema to XML Schema data types.

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let *S* be the schema provided for an application of this mapping. Let *TABLEFOREST* be the choice of whether to map the table to a sequence of XML elements (*True*) or to map the table to an XML document with a single root element (*False*). Let *U* be the authorization identifier that is invoking this mapping.

2) Let *SC*, and *SN* be the <catalog name> and <unqualified schema name> of the <schema name> of *S*, respectively.

3) Let ***XMLSN*** be the result of applying the mapping defined in Subclause 9.1, "Mapping SQL <identifier>s to XML Names", to *SN* using the fully escaped variant of the mapping.

4) Let *n* be the number of XML visible tables of *S* for *U*.

   NOTE 43 — "XML visible table" is defined in Subclause 4.10.8, "Visibility of columns, tables, and schemas in mappings from SQL to XML".

5) For *i* ranging from 1 (one) to *n*:

   a) Let T_i be the *i*-th XML visible table of *S* for *U* in the implementation-dependent repeatable ordering of *S*.

   b) Let *TN* be the <qualified identifier> of the <table name> of T_i.

   c) Let ***XMLTN*** be the result of applying the mapping defined in Subclause 9.1, "Mapping SQL <identifier>s to XML Names", to *TN* using the fully escaped variant of the mapping.

   d) Let ***XMLTTN*** be the result of applying the mapping defined in Subclause 9.2, "Mapping a multi-part SQL name to an XML Name", to "TableType", *SC*, *SN*, and *TN*.

   e) Let ***XMLROWN*** be the result of applying the mapping defined in Subclause 9.2, "Mapping a multi-part SQL name to an XML Name", to "RowType", *SC*, *SN*, and *TN*.

   f) Case:

      i) If *TABLEFOREST* is *False*, then let $\boldsymbol{XMLTE_i}$ be

```
<xs:element name="XMLTN" type="XMLTTN" />
```

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ii) If *TABLEFOREST* is *True*, then let ***XMLTE*$_i$** be

```
<xs:element name="XMLTN" type="XMLROWN"
    minOccurs="0" maxOccurs="unbounded" />
```

6) Let ***XMLTYPEN*** be the result of applying the mapping defined in Subclause 9.2, "Mapping a multi-part SQL name to an XML Name", to "SchemaType", *SC*, and *SN*.

7) Let ***XMLSC*** be the result of applying the mapping defined in Subclause 9.1, "Mapping SQL <identifier>s to XML Names", to *SC* using the fully escaped variant of the mapping.

8) It is implementation-defined whether ***SQLANN*** is the zero-length string or

```
<xs:annotation>
   <xs:appinfo>
      <sqlxml:sqlname
          type="SCHEMA"
          catalogName="XMLSC"
          schemaName="XMLSN" />
   </xs:appinfo>
</xs:annotation>
```

9) Case:

a) If *TABLEFOREST* is *False*, then let ***XMLSCHEMAT*** be:

```
<xs:complexType name="XMLTYPEN">
   SQLANN
   <xs:all>
      XMLTE1
      ...
      XMLTEn
   </xs:all>
</xs:complexType>
```

b) If *TABLEFOREST* is *True*, then let ***XMLSCHEMAT*** be:

```
<xs:complexType name="XMLTYPEN">
   SQLANN
   <xs:sequence>
      XMLTE1
      ...
      XMLTEn
   </xs:sequence>
</xs:complexType>
```

10) ***XMLSCHEMAT*** contains the XML Schema data types that are the result of this mapping.

Conformance Rules

None.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

9.13 Mapping an SQL schema to an XML element

Function

Define the mapping of an SQL-schema to an XML element.

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let *S* be the schema provided for an application of this mapping. Let *NULLS* be the choice of whether to map null values to absent elements (absent) or to elements that are marked with **`xsi:nil="true"`** (nil). Let *TABLEFOREST* be the choice of whether to map a table to a sequence of XML elements (*True*) or to a single XML element (*False*). Let ***TARGETNS*** be the XML target namespace URI of the XML Schema and data to be mapped. If ***TARGETNS*** is the zero-length string, then no XML namespace is added. Let *U* be the authorization identifier that is invoking this mapping. Let *ENCODING* be the choice of whether to encode binary strings in base64 or in hex.

2) Let *SN* be the <unqualified schema name> of *S*.

3) Let ***XMLSN*** be the result of applying the mapping defined in Subclause 9.1, "Mapping SQL <identifier>s to XML Names", to *SN* using the fully escaped variant of the mapping.

4) Let *n* be the number of XML visible tables of *S* for *U*.

   NOTE 44 — "XML visible table " is defined in Subclause 4.10.8, "Visibility of columns, tables, and schemas in mappings from SQL to XML".

5) For *i* ranging from 1 (one) to *n*:

   a) Let T_i be the *i*-th XML visible table of *S* for *U* in the implementation-dependent repeatable ordering of *S*.

   b) Let ***XMLT$_i$*** be the result of applying the mapping defined in Subclause 9.10, "Mapping an SQL table to an XML element or a sequence of XML elements", to T_i using *NULLS* as the choice of whether to map null values to absent elements or to elements that are marked with **`xsi:nil="true"`**, *TABLEFOREST* as the choice of how to map the table, ***TARGETNS*** set to the zero-length string for the application of Subclause 9.10, "Mapping an SQL table to an XML element or a sequence of XML elements", *U* as the invoker of this mapping, and *ENCODING* as the choice of whether to encode binary strings in base64 or in hex.

6) Case:

   a) If ***TARGETNS*** is the zero-length string, then let ***XMLSE*** be:

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
<XMLSN>
   XMLT1
   ...
   XMLTn
</XMLSN>
```

b) If ***TARGETNS*** is not the zero-length string, then let ***XMLSE*** be:

```
<XMLSN xmlns="TARGETNS">
   XMLT1
   ...
   XMLTn
</XMLSN>
```

7) ***XMLSE*** is the XML element that is the result of the application of this Subclause.

Conformance Rules

None.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

9.14 Mapping an SQL schema to an XML document and an XML Schema document

Function

Define the mapping of an SQL-schema to an XML document and an XML Schema document that describes this XML document.

Syntax Rules

1) Let *S* be the schema provided for an application of this mapping. Let *NULLS* be the choice of whether to map null values to absent elements (absent) or to elements that are marked with **`xsi:nil="true"`** (nil). Let *TABLEFOREST* be the choice of whether to map a table to a sequence of XML elements (*True*) or to a single XML element (*False*). Let ***TARGETNS*** be the XML target namespace URI of the XML Schema and data to be mapped. If ***TARGETNS*** is the zero-length string, then no XML namespace is added. Let *DATA* be the choice of whether the mapping results in an XML representation of the data of *S* (*True*) or not (*False*). Let *METADATA* be the choice of whether the mapping results in an XML Schema document that describes the XML representation of the data of *S* (*True*) or not (*False*). Let *U* be the authorization identifier that is invoking this mapping. Let *ENCODING* be the choice of whether binary strings are to be encoded using base64 or using hex.

2) At least one of *DATA* and *METADATA* shall be *True*.

Access Rules

None.

General Rules

1) Let *SC* and *SN* be the <catalog name> and <unqualified schema name> of *S*, respectively.

2) Let ***XMLSN*** be the result of applying the mapping defined in Subclause 9.1, "Mapping SQL <identifier>s to XML Names", to *SN* using the fully escaped variant of the mapping.

3) If any of the visible columns of the viewed and base tables contained in *S* for *U* is an XML unmappable column, then a completion condition is raised: *warning — column cannot be mapped to XML*.

   NOTE 45 — "visible column" is defined in Subclause 4.10.8, "Visibility of columns, tables, and schemas in mappings from SQL to XML".

4) Let *CT* be the XML visible columns of the viewed and base tables contained in *S* for *U*.

5) Let ***XSCT*** be the result of applying the mapping defined in Subclause 9.7, "Mapping a collection of SQL data types to XML Schema data types", to the data types and domains of the columns of *CT* using *NULLS* as the choice of whether to map null values to absent elements or elements that are marked with **`xsi:nil="true"`**, and *ENCODING* as the choice of whether binary strings are to be encoded using base64 or using hex.

6) Let ***XMLTYPEN*** be the result of applying the mapping defined in Subclause 9.2, "Mapping a multi-part SQL name to an XML Name", to "SchemaType", *SC*, and *SN*.

© ISO/IEC 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7) Let ***XST*** be the result of applying the mapping defined in Subclause 9.12, "Mapping an SQL schema to XML Schema data types", to *S* using *TABLEFOREST* as the choice of how to map *T* and *U* as the invoker of this mapping.

8) Case:

   a) If either ***XSCT*** or ***XST*** contains an XML 1.1 QName that is not an XML 1.0 QName, then let *VER* be 1.1.

   b) Otherwise, it is implementation-defined whether *VER* is 1.1 or 1.0.

9) The encoding to use for the result(s) of this mapping is implementation-defined. If this encoding is not UTF-8 or UTF-16, then let ***ENCODING*** be the name of the encoding, and let ***ENC*** be

```
encoding = 'ENCODING'
```

   Otherwise, let ***ENC*** be the zero-length string.

10) If *VER* is not 1.0 or ***ENC*** is not the zero-length string, then let ***XMLDECL*** be

```
<?xml version='VER' ENC ?>
```

11) If *METADATA* is *True*, then:

   a) Let ***SQLXMLNS*** be the value of the XML namespace URI provided for the XML namespace prefix **sqlxml** in Table 2, "XML namespace prefixes and their URIs".

   b) Let ***XSDNS*** be the value of the XML namespace URI provided for the XML namespace prefix **xs** in Table 2, "XML namespace prefixes and their URIs".

   c) Let ***SLOCVAL*** be an implementation-defined URI that references the XML Schema document describing the XML namespace ***SQLXMLNS***. It is implementation-defined whether ***SLOC*** is the zero-length string or

```
schemaLocation="SLOCVAL"
```

      Case:

      i) If neither ***XSCT*** nor ***XST*** uses the XML namespace that corresponds to the **sqlxml** XML namespace prefix, then let ***XSDIMP*** be the zero-length string.

      ii) If at least one of ***XSCT*** and ***XST*** uses the XML namespace that corresponds to the **sqlxml** XML namespace prefix, then let ***XSDIMP*** be

```
<xs:import
    namespace="SQLXMLNS" SLOC />
```

   d) Case:

      i) If neither ***XSCT*** nor ***XST*** uses the XML namespace that corresponds to the **sqlxml** XML namespace prefix, then let ***XSDECL*** be the zero-length string.

      ii) If at least one of ***XSCT*** and ***XST*** uses the XML namespace that corresponds to the **sqlxml** XML namespace prefix, then let ***XSDECL*** be

```
xmlns:sqlxml="SQLXMLNS"
```

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

e) Case:

i) If ***TARGETNS*** is the zero-length string, then ***XS*** has the following contents:

```
XMLDECL
<xs:schema
      xmlns:xs="XSDNS"
      XSDECL>
   XSDIMP
   XSCT
   XST
   <xs:element name="XMLSN" type="XMLTYPEN" />
</xs:schema>
```

ii) If ***TARGETNS*** is not the zero-length string, then ***XS*** has the following contents:

```
XMLDECL
<xs:schema
      xmlns:xs="XSDNS"
      XSDECL
      targetNamespace="TARGETNS"
      elementFormDefault="qualified">
   XSDIMP
   XSCT
   XST
   <xs:element name="XMLSN" type="XMLTYPEN" />
</xs:schema>
```

f) Let ***XSR*** be:

```
XMLSERIALIZE ( DOCUMENT
               XMLPARSE ( DOCUMENT 'XS' PRESERVE WHITESPACE )
               AS CLOB )
```

12) Case:

a) If *METADATA* is *True*, then let ***XSL*** be the URI that identifies ***XSR***.

Case:

i) If ***TARGETNS*** is the zero-length string, then let ***XSLA*** be

```
xsi:noNamespaceSchemaLocation="XSL"
```

ii) Otherwise, let *XSLA* be

```
xsi:schemaLocation="TARGETNS XSL"
```

b) Otherwise, let ***XSLA*** be the zero-length string.

13) If *DATA* is *True*, then:

a) Let ***XDSCHEMA*** be the result of applying the mapping defined in Subclause 9.13, "Mapping an SQL schema to an XML element", to *S* using *NULLS* as the choice of whether to map null values to absent elements or elements that are marked with **xsi:nil="true"**, *TABLEFOREST* as the choice of how

© ISO/IEC 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

to map the tables, ***TARGETNS*** indicating the XML target namespace, *U* as the invoker of this mapping, and *ENCODING* as the choice of whether binary strings are to be encoded using base64 or using hex.

b) Let ***XSINS*** be the value of the XML namespace URI provided for the XML namespace prefix **xsi** in Table 2, "XML namespace prefixes and their URIs".

c) Case:

i) If ***TARGETNS*** is the zero-length string, then ***XD*** has the following contents:

```
XMLDECL
<XMLSN
   xmlns:xsi="XSINS"
   XSLA>
   XDSCHEMA
</XMLSN>
```

ii) If ***TARGETNS*** is not the zero-length string, then ***XD*** has the following contents:

```
XMLDECL
<XMLSN
   xmlns:xsi="XSINS"
   xmlns="TARGETNS"
   XSLA>
   XDSCHEMA
</XMLSN>
```

d) Let *XDR* be:

```
XMLSERIALIZE ( DOCUMENT
               XMLPARSE ( DOCUMENT 'XD' PRESERVE WHITESPACE )
               AS CLOB )
```

14) If *DATA* is *True*, then the value of *XDR* is the XML representation of the data of *S*. If *METADATA* is *True*, then ***XSR*** is the XML Schema document that describes the XML representation of the data of *S*.

Conformance Rules

1) Without Feature X050, "Advanced table mapping", a conforming application shall not invoke this Subclause of this part of this International Standard.

2) Without Feature X051, "Advanced table mapping: nulls absent", a conforming application shall not invoke this Subclause of this part of this International Standard with *NULLS* set to indicate that nulls are mapped to elements that are marked to absent elements.

3) Without Feature X052, "Advanced table mapping: null as nil", a conforming application shall not invoke this Subclause of this part of this International Standard with *NULLS* set to indicate that nulls are mapped to elements that are marked with **xsi:nil="true"**.

4) Without Feature X053, "Advanced table mapping: table as forest", a conforming application shall not invoke this Subclause of this part of this International Standard with *TABLEFOREST* set to *True*.

5) Without Feature X054, "Advanced table mapping: table as element", a conforming application shall not invoke this Subclause of this part of this International Standard with *TABLEFOREST* set to *False*.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

6) Without Feature X055, "Advanced table mapping: with target namespace", a conforming application shall not invoke this Subclause of this part of this International Standard with *TARGETNS* that is not a zero-length string.

7) Without Feature X056, "Advanced table mapping: data mapping", a conforming application shall not invoke this Subclause of this part of this International Standard with *DATA* set to *True*.

8) Without Feature X057, "Advanced table mapping: metadata mapping", a conforming application shall not invoke this Subclause of this part of this International Standard with *METADATA* set to *True*.

9) Without Feature X058, "Advanced table mapping: base64 encoding of binary strings", a conforming application shall not invoke this Subclause of this part of this International Standard with *ENCODING* set to indicate that binary strings are to be encoded using base64.

10) Without Feature X059, "Advanced table mapping: hex encoding of binary strings", a conforming application shall not invoke this Subclause of this part of this International Standard with *ENCODING* set to indicate that binary strings are to be encoded using hex.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

9.15 Mapping an SQL catalog to XML Schema data types

Function

Define the mapping of an SQL catalog to XML Schema data types.

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let *C* be the catalog provided for an application of this mapping. Let *U* be the authorization identifier that is invoking this mapping.

2) Let *CN* be the <catalog name> of *C*.

3) Let ***XMLCN*** be the result of applying the mapping defined in Subclause 9.1, "Mapping SQL <identifier>s to XML Names", to *CN* using the fully escaped variant of the mapping.

4) Let *n* be the number of XML visible schemas of *C* for *U*.

   NOTE 46 — "XML visible schema" is defined in Subclause 4.10.8, "Visibility of columns, tables, and schemas in mappings from SQL to XML".

5) For *i* ranging from 1 (one) to *n*:

   a) Let S_i be the *i*-th XML visible schema of *C*.

   b) Let *SN* be the <unqualified schema name> of the <schema name> of S_i.

   c) Let ***XMLSN*** be the result of applying the mapping defined in Subclause 9.1, "Mapping SQL <identifier>s to XML Names", to *SN* using the fully escaped variant of the mapping.

   d) Let ***XMLSTN*** be the result of applying the mapping defined in Subclause 9.2, "Mapping a multi-part SQL name to an XML Name", to "SchemaType", *CN*, and *SN*.

   e) Let ***XMLSE$_i$*** be

      ```
      <xs:element name="XMLSN" type="XMLSTN" />
      ```

6) Let ***XMLTYPEN*** be the result of applying the mapping defined in Subclause 9.2, "Mapping a multi-part SQL name to an XML Name", to "CatalogType" and *CN*.

7) It is implementation-defined whether ***SQLANN*** is the zero-length string or

   ```
   <xs:annotation>
      <xs:appinfo>
   ```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
        <sqlxml:sqlname
            type="CATALOG"
            catalogName="XMLCN" />
      </xs:appinfo>
   </xs:annotation>
```

8) Let ***XMLCATT*** be:

```
<xs:complexType name="XMLTYPEN">
   SQLANN
   <xs:all>
      XMLSE1
      ...
      XMLSEn
   </xs:all>
</xs:complexType>
```

9) ***XMLCATT*** contains the XML Schema data types that are the result of this mapping.

Conformance Rules

None.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

9.16 Mapping an SQL catalog to an XML element

Function

Define the mapping of an SQL catalog to an XML element.

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let *C* be the catalog provided for an application of this mapping. Let *NULLS* be the choice of whether to map null values to absent elements (absent) or to elements that are marked with **`xsi:nil="true"`** (nil). Let *TABLEFOREST* be the choice of whether to map a table to a sequence of XML elements (*True*) or to a single XML element (*False*). Let ***TARGETNS*** be the XML target namespace URI of the XML Schema and data to be mapped. If ***TARGETNS*** is the zero-length string, then no XML namespace is added. Let *U* be the authorization identifier that is invoking this mapping. Let *ENCODING* be the choice of whether to encode binary strings in base64 or in hex.

2) Let *CN* be the <catalog name> of *C*.

3) Let ***XMLCN*** be the result of applying the mapping defined in Subclause 9.1, "Mapping SQL <identifier>s to XML Names", to *CN* using the fully escaped variant of the mapping.

4) Let *n* be the number of XML visible schemas of *C* for *U*.

   NOTE 47 — "XML visible schema" is defined in Subclause 4.10.8, "Visibility of columns, tables, and schemas in mappings from SQL to XML".

5) For *i* ranging from 1 (one) to *n*:

   a) Let S_i be the *i*-th XML visible schema of *C* for *U* in the implementation-dependent repeatable ordering of *C*.

   b) Let ***XMLS$_i$*** be the result of applying the mapping defined in Subclause 9.13, "Mapping an SQL schema to an XML element", to S_i using *NULLS* as the choice of whether to map null values to absent elements or to elements that are marked with **`xsi:nil="true"`**, *TABLEFOREST* as the choice of how to map a table, ***TARGETNS*** set to the zero-length string for the application of Subclause 9.13, "Mapping an SQL schema to an XML element", *U* as the invoker of this mapping, and *ENCODING* as the choice of whether to encode binary strings in base64 or in hex.

6) Case:

   a) If ***TARGETNS*** is the zero-length string, then let ***XMLCE*** be:

      <XMLCN>

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
   XMLS1
   ...
   XMLSn
</XMLCN>
```

b) If ***TARGETNS*** is not the zero-length string, then let ***XMLCE*** be:

```
<XMLCN xmlns="TARGETNS">
   XMLS1
   ...
   XMLSn
</XMLCN>
```

7) ***XMLCE*** is the XML element that is the result of the application of this Subclause.

Conformance Rules

None.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

9.17 Mapping an SQL catalog to an XML document and an XML Schema document

Function

Define the mapping of an SQL catalog to an XML document and an XML Schema document that describes this XML document.

Syntax Rules

1) Let *C* be the catalog provided for an application of this mapping. Let *NULLS* be the choice of whether to map null values to absent elements (absent) or to elements that are marked with **`xsi:nil="true"`** (nil). Let *TABLEFOREST* be the choice of whether to map a table to a sequence of XML elements (*True*) or to a single XML element (*False*). Let ***TARGETNS*** be the XML target namespace URI of the XML Schema and data to be mapped. If ***TARGETNS*** is the zero-length string, then no XML namespace is added. Let *DATA* be the choice of whether the mapping results in an XML representation of the data of *C* (*True*) or not (*False*). Let *METADATA* be the choice of whether the mapping results in an XML Schema document that describes the XML representation of the data of *C* (*True*) or not (*False*). Let *U* be the authorization identifier that is invoking this mapping. Let *ENCODING* be the choice of whether binary strings are to be encoded using base64 or using hex.

2) At least one of *DATA* and *METADATA* shall be *True*.

Access Rules

None.

General Rules

1) Let *CN* be the <catalog name> of *C*.

2) Let ***XMLCN*** be the result of applying the mapping defined in Subclause 9.1, "Mapping SQL <identifier>s to XML Names", to *CN* using the fully escaped variant of the mapping.

3) If any of the visible columns of the viewed and base tables contained in *C* for *U* is an XML unmappable column, then a completion condition is raised: *warning — column cannot be mapped to XML*.

NOTE 48 — "visible column" is defined in Subclause 4.10.8, "Visibility of columns, tables, and schemas in mappings from SQL to XML".

4) Let *CT* be the XML visible columns of the viewed and base tables contained in *C* for *U*.

5) Let ***XSCT*** be the result of applying the mapping defined in Subclause 9.7, "Mapping a collection of SQL data types to XML Schema data types", to the data types and domains of the columns of *CT* using *NULLS* as the choice of whether to map null values to absent elements or elements that are marked with **`xsi:nil="true"`**, and *ENCODING* as the choice of whether binary strings are to be encoded using base64 or using hex.

6) Let ***XMLTYPEN*** be the result of applying the mapping defined in Subclause 9.2, "Mapping a multi-part SQL name to an XML Name", to "CatalogType" and *CN*.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7) Let ***XST*** be the result of applying the mapping defined in Subclause 9.15, "Mapping an SQL catalog to XML Schema data types", to *C* using *U* as the invoker of this mapping.

8) Case:

   a) If either ***XSCT*** or ***XST*** contains an XML 1.1 QName that is not an XML 1.0 QName, then let *VER* be 1.1.

   b) Otherwise, it is implementation-defined whether *VER* is 1.1 or 1.0.

9) The encoding to use for the result(s) of this mapping is implementation-defined. If this encoding is not UTF-8 or UTF-16, then let ***ENCODING*** be the name of the encoding, and let ***ENC*** be

   ```
   encoding = 'ENCODING'
   ```

   Otherwise, let ***ENC*** be the zero-length string.

10) If *VER* is not 1.0 or ***ENC*** is not the zero-length string, then let ***XMLDECL*** be

    ```
    <?xml version='VER' ENC ?>
    ```

11) If *METADATA* is *True*, then:

    a) Let ***SQLXMLNS*** be the value of the XML namespace URI provided for the XML namespace prefix **sqlxml** in Table 2, "XML namespace prefixes and their URIs".

    b) Let ***XSDNS*** be the value of the XML namespace URI provided for the XML namespace prefix **xs** in Table 2, "XML namespace prefixes and their URIs".

    c) Let ***SLOCVAL*** be an implementation-defined URI that references the XML Schema document describing the XML namespace ***SQLXMLNS***. It is implementation-defined whether ***SLOC*** is the zero-length string or

       ```
       schemaLocation="SLOCVAL"
       ```

       Case:

       i) If neither ***XSCT*** nor ***XST*** uses the XML namespace that corresponds to the **sqlxml** XML namespace prefix, then let ***XSDIMP*** be the zero-length string.

       ii) If at least one of ***XSCT*** and ***XST*** uses the XML namespace that corresponds to the **sqlxml** XML namespace prefix, then let ***XSDIMP*** be

       ```
       <xs:import
           namespace="SQLXMLNS" SLOC />
       ```

    d) Case:

       i) If neither ***XSCT*** nor ***XST*** uses the XML namespace that corresponds to the **sqlxml** XML namespace prefix, then let ***XSDECL*** be the zero-length string.

       ii) If at least one of ***XSCT*** and ***XST*** uses the XML namespace that corresponds to the **sqlxml** XML namespace prefix, then let ***XSDECL*** be

       ```
       xmlns:sqlxml="SQLXMLNS"
       ```

    e) Case:

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

i) If ***TARGETNS*** is the zero-length string, then ***XS*** has the following contents:

```
XMLDECL
<xs:schema
      xmlns:xs="XSDNS"
      XSDECL>
   XSDIMP
   XSCT
   XST
   <xs:element name="XMLCN" type="XMLTYPEN" />
</xs:schema>
```

ii) If ***TARGETNS*** is not the zero-length string, then ***XS*** has the following contents:

```
XMLDECL
<xs:schema
      xmlns:xs="XSDNS"
      XSDECL
      targetNamespace="TARGETNS"
      elementFormDefault="qualified">
   XSDIMP
   XSCT
   XST
   <xs:element name="XMLCN" type="XMLTYPEN" />
</xs:schema>
```

f) Let ***XSR*** be:

```
XMLSERIALIZE ( DOCUMENT
               XMLPARSE ( DOCUMENT 'XS' PRESERVE WHITESPACE )
               AS CLOB )
```

12) Case:

a) If *METADATA* is *True*, then let ***XSL*** be the URI that identifies ***XSR***.

Case:

i) If ***TARGETNS*** is the zero-length string, then let ***XSLA*** be

```
xsi:noNamespaceSchemaLocation="XSL"
```

ii) Otherwise, let ***XSLA*** be

```
xsi:schemaLocation="TARGETNS XSL"
```

b) Otherwise, let ***XSLA*** be the zero-length string.

13) If *DATA* is *True*, then:

a) Let ***XDCATALOG*** be the result of applying the mapping defined in Subclause 9.16, "Mapping an SQL catalog to an XML element", to *C* using *NULLS* as the choice of whether to map null values to absent elements or to elements that are marked with **xsi:nil="true"**, *TABLEFOREST* as the choice of how to map tables, ***TARGETNS*** indicating the XML target namespace, *U* as the invoker of this mapping and *ENCODING* as the choice of whether binary strings are to be encoded using base64 or using hex.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

b) Let ***XSINS*** be the value of the XML namespace URI provided for the XML namespace prefix **xsi** in Table 2, "XML namespace prefixes and their URIs".

c) Case:

i) If ***TARGETNS*** is the zero-length string, then ***XD*** has the following contents:

```
XMLDECL
<XMLCN
   xmlns:xsi="XSINS"
   XSLA>
   XDCATALOG
</XMLCN>
```

ii) If ***TARGETNS*** is not the zero-length string, then ***XD*** has the following contents:

```
XMLDECL
<XMLCN
   xmlns:xsi="XSINS"
   xmlns="TARGETNS"
   XSLA>
   XDCATALOG
</XMLCN>
```

d) Let *XDR* be:

```
XMLSERIALIZE ( DOCUMENT
               XMLPARSE ( DOCUMENT 'XD' PRESERVE WHITESPACE )
               AS CLOB )
```

14) If *DATA* is *True*, then the value of *XDR* is the XML representation of the data of *C*. If *METADATA* is *True*, then ***XSR*** is the XML Schema document that describes the XML representation of the data of *C*.

Conformance Rules

1) Without Feature X050, "Advanced table mapping", a conforming application shall not invoke this Subclause of this part of this International Standard.

2) Without Feature X051, "Advanced table mapping: nulls absent", a conforming application shall not invoke this Subclause of this part of this International Standard with *NULLS* set to indicate that nulls are mapped to elements that are marked to absent elements.

3) Without Feature X052, "Advanced table mapping: null as nil", a conforming application shall not invoke this Subclause of this part of this International Standard with *NULLS* set to indicate that nulls are mapped to elements that are marked with **xsi:nil="true"**.

4) Without Feature X053, "Advanced table mapping: table as forest", a conforming application shall not invoke this Subclause of this part of this International Standard with *TABLEFOREST* set to *True*.

5) Without Feature X054, "Advanced table mapping: table as element", a conforming application shall not invoke this Subclause of this part of this International Standard with *TABLEFOREST* set to *False*.

6) Without Feature X055, "Advanced table mapping: with target namespace", a conforming application shall not invoke this Subclause of this part of this International Standard with *TARGETNS* that is not a zero-length string.

© ISO/IEC 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

7) Without Feature X056, “Advanced table mapping: data mapping”, a conforming application shall not invoke this Subclause of this part of this International Standard with *DATA* set to ***True***.

8) Without Feature X057, “Advanced table mapping: metadata mapping”, a conforming application shall not invoke this Subclause of this part of this International Standard with *METADATA* set to ***True***.

9) Without Feature X058, “Advanced table mapping: base64 encoding of binary strings”, a conforming application shall not invoke this Subclause of this part of this International Standard with *ENCODING* set to indicate that binary strings are to be encoded using base64.

10) Without Feature X059, “Advanced table mapping: hex encoding of binary strings”, a conforming application shall not invoke this Subclause of this part of this International Standard with *ENCODING* set to indicate that binary strings are to be encoded using hex.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

(Blank page)

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10 Additional common rules

This Clause modifies Clause 9, "Additional common rules", in ISO/IEC 9075-2.

10.1 Retrieval assignment

This Subclause modifies Subclause 9.1, "Retrieval assignment", in ISO/IEC 9075-2.

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 *TD* is an XML type, then

   Case:

   a) If *TD* is either XML(DOCUMENT(UNTYPED)) or XML(CONTENT(UNTYPED)), then *SD* shall be either XML(DOCUMENT(UNTYPED)) or XML(CONTENT(UNTYPED)).

   b) If *TD* is either XML(DOCUMENT(XMLSCHEMA)) or XML(CONTENT(XMLSCHEMA)), then

      Case:

      i) If the type descriptor of *TD* includes the the XML namespace URI *N* and the XML NCName *EN* of a global element declaration schema component, then *SD* shall be either XML(DOCUMENT(XMLSCHEMA)) or XML(CONTENT(XMLSCHEMA)), the XML Schema identified by the registered XML Schema descriptor included in the type descriptor of *TD* shall be identical to the XML Schema identified by the registered XML Schema descriptor included in the type descriptor of *SD*, and the type descriptor of *SD* shall include an XML namespace URI that is identical to *N*, as defined by [Namespaces] and an XML NCName that is equivalent to *EN*.

      ii) If the type descriptor of *TD* includes the the XML namespace URI *N*, then *SD* shall be either XML(DOCUMENT(XMLSCHEMA)) or XML(CONTENT(XMLSCHEMA)), the XML Schema identified by the registered XML Schema descriptor included in the type descriptor of *TD* shall be identical to the XML Schema identified by the registered XML Schema descriptor included in the type descriptor of *SD*, and the type descriptor of *SD* shall include an XML namespace URI that is identical to *N*, as defined by [Namespaces].

      iii) Otherwise, *SD* shall be either XML(DOCUMENT(XMLSCHEMA)) or XML(CONTENT(XMLSCHEMA)) and the XML Schema identified by the registered XML Schema descriptor included in the type descriptor of *TD* shall be identical to the XML Schema identified by the registered XML Schema descriptor included in the type descriptor of *SD*.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

c) Otherwise, *SD* shall be an XML type.

Access Rules

No additional Access Rules.

General Rules

1) Augment GR 6) If the declared type of *T* is an XML type, then:

   a) Case:

      i) If the declared type of *T* is XML(DOCUMENT(UNTYPED)), XML(DOCUMENT(ANY)), or XML(DOCUMENT(XMLSCHEMA)), and *V* is not an XQuery sequence of length 1 (one) whose sole XQuery item is an XQuery document node whose **children** property contains exactly one XQuery element node, zero or more XQuery comment nodes, and zero or more XQuery processing instruction nodes, then an exception condition is raised: *data exception — not an XML document*.

      ii) If the declared type of *T* is XML(CONTENT(UNTYPED)), XML(CONTENT(ANY)), or XML(CONTENT(XMLSCHEMA)), and *V* is not an XQuery sequence of length 1 (one) whose sole XQuery item is an XQuery document node, then an exception condition is raised: *data exception — not an XQuery document node*.

   b) The value of *T* is set to *V*.

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10.2 Store assignment

This Subclause modifies Subclause 9.2, "Store assignment", in ISO/IEC 9075-2.

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 *TD* is an XML type, then:

   a) Let *P* be the *PASSING* specified in an application of this Subclause, if any.

   b) Case:

      i) If *TD* is either XML(DOCUMENT(UNTYPED)) or XML(CONTENT(UNTYPED)), then *SD* shall be either XML(DOCUMENT(UNTYPED)) or XML(CONTENT(UNTYPED)).

      ii) If *TD* is either XML(DOCUMENT(XMLSCHEMA)) or XML(CONTENT(XMLSCHEMA)), then

      Case:

      1) If the type descriptor of *TD* includes the XML namespace URI *N* and the XML NCName *EN* of a global element declaration schema component, then *SD* shall be either XML(DOCUMENT(XMLSCHEMA)) or XML(CONTENT(XMLSCHEMA)), the XML Schema identified by the registered XML Schema descriptor included in the type descriptor of *TD* shall be identical to the XML Schema identified by the registered XML Schema descriptor included in the type descriptor of *SD*, and the type descriptor of *SD* shall include an XML namespace URI that is identical to *N*, as defined by [Namespaces], and an XML NCName that is equivalent to *EN*.

      2) If the type descriptor of *TD* includes the XML namespace URI *N*, then *SD* shall be either XML(DOCUMENT(XMLSCHEMA)) or XML(CONTENT(XMLSCHEMA)), the XML Schema identified by the registered XML Schema descriptor included in the type descriptor of *TD* shall be identical to the XML Schema identified by the registered XML Schema descriptor included in the type descriptor of *SD*, and the type descriptor of *SD* shall include an XML namespace URI that is identical to *N*, as defined by [Namespaces].

      3) Otherwise, *SD* shall be either XML(DOCUMENT(XMLSCHEMA)) or XML(CONTENT(XMLSCHEMA)) and the XML Schema identified by the registered XML Schema descriptor included in the type descriptor of *TD* shall be identical to the XML Schema identified by the registered XML Schema descriptor included in the type descriptor of *SD*.

      iii) Otherwise, *SD* shall be an XML type.

Access Rules

No additional Access Rules.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

General Rules

1) Augment GR 2)b) If the declared type of *T* is the XML type, then:

   a) Case:

      i) If the declared type of *T* is XML(DOCUMENT(UNTYPED)), XML(DOCUMENT(ANY)), or XML(DOCUMENT(XMLSCHEMA)), and *V* is not an XQuery sequence of length 1 (one) whose sole XQuery item is an XQuery document node whose **children** property contains exactly one XQuery element node, zero or more XQuery comment nodes, and zero or more XQuery processing instruction nodes, then an exception condition is raised: *data exception — not an XML document*.

      ii) If the declared type of *T* is XML(CONTENT(UNTYPED)), XML(CONTENT(ANY)), or XML(CONTENT(XMLSCHEMA)), and *V* is not an XQuery sequence of length 1 (one) whose sole XQuery item is an XQuery document node, then an exception condition is raised: *data exception — not an XQuery document node*.

   b) Case:

      i) If *T* is a <column reference> or a <target array element specification> whose <target array reference> is a <column reference>, or *P* is BY VALUE, then let *V1* be the result of applying the General Rules of Subclause 10.18, "Constructing a copy of an XML value", with *V* as *VALUE*.

      ii) Otherwise, let *V1* be *V*.

   c) The value of *T* is set to *V1*.

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10.3 Result of data type combinations

This Subclause modifies Subclause 9.3, "Result of data type combinations", in ISO/IEC 9075-2.

Function

Specify the result data type of the result of certain combinations of 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 SR 3)g) If any data type in *DTS* is an XML type, then:

   a) Each data type in *DTS* shall be an XML type.

   b) Case:

      i) If any of the data types in *DTS* is XML(SEQUENCE), then the result data type is XML(SEQUENCE).

      ii) Otherwise:

         1) Let *N* be the number of data types in *DTS*. Let T_i, 1 (one) ≤ *j* ≤ *N*, be the *i*-th data type in *DTS*.

         2) Case:

            A) If *N* = 1 (one), then the result data type is T_1.

            B) Otherwise,

               Case:

               I) If T_i, for all *i*, 1 (one) ≤ *i* ≤ *N*, is XML(DOCUMENT(UNTYPED)), then the result data type is XML(DOCUMENT(UNTYPED)).

               II) If T_i, for all *i*, 1 (one) ≤ *i* ≤ *N*, is either XML(DOCUMENT(UNTYPED)) or XML(CONTENT(UNTYPED)), then the result data type is XML(CONTENT(UNTYPED)).

               III) If there exists a registered XML Schema *S* and a global element declaration schema component ***E*** such that, for all *i*, 1 (one) ≤ *i* ≤ *N*, T_i is XML(DOCUMENT(XMLSCHEMA)) and the XML type descriptor of T_i includes the registered XML Schema descriptor of *S* and an XML namespace URI ***NS*** that is identical to the XML namespace URI of ***E***, as defined by [Namespaces], and an XML NCName ***EN*** that is equivalent to the XML NCName of ***E***, then the result data type is XML(DOCUMENT(XMLSCHEMA)) whose XML type descriptor includes the registered XML Schema descriptor of *S*, the XML namespace URI ***NS***, and the XML NCName ***EN***.

               IV) If there exists a registered XML Schema *S* and an XML namespace ***NS*** such that, for all *i*, 1 (one) ≤ *i* ≤ *N*, T_i is XML(DOCUMENT(XMLSCHEMA)) and the

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

XML type descriptor of T_i includes the registered XML Schema descriptor of *S* and an XML namespace URI that is identical to ***NS***, as defined by [Namespaces], then the result data type is XML(DOCUMENT(XMLSCHEMA)) whose XML type descriptor includes the registered XML Schema descriptor of *S* and the XML namespace URI ***NS***.

V) If there exists a registered XML Schema *S* such that, for all *i*, 1 (one) ≤ *i* ≤ *N*, T_i is XML(DOCUMENT(XMLSCHEMA)) and the XML type descriptor of T_i includes the registered XML Schema descriptor of *S*, then the result data type is XML(DOCUMENT(XMLSCHEMA)) whose XML type descriptor includes the registered XML Schema descriptor of *S*.

VI) If there exists a registered XML Schema *S* and a global element declaration schema component ***E*** such that, for all *i*, 1 (one) ≤ *i* ≤ *N*, T_i is either XML(DOCUMENT(XMLSCHEMA)) or XML(CONTENT(XMLSCHEMA)) and the XML type descriptor of T_i includes the registered XML Schema descriptor of *S* and an XML namespace URI ***NS*** that is identical to the XML namespace URI of ***E***, as defined by [Namespaces], and an XML NCName ***EN*** that is equivalent to the XML NCName of ***E***, then the result data type is XML(CONTENT(XMLSCHEMA)) whose XML type descriptor includes the registered XML Schema descriptor of *S*, the XML namespace URI ***NS***, and the XML NCName ***EN***.

VII) If there exists a registered XML Schema *S* and an XML namespace ***NS*** such that, for all *i*, 1 (one) ≤ *i* ≤ *N*, T_i is either XML(DOCUMENT(XMLSCHEMA)) or XML(CONTENT(XMLSCHEMA)) and the XML type descriptor of T_i includes the registered XML Schema descriptor of *S* and an XML namespace URI that is identical to ***NS***, as defined by [Namespaces], then the result data type is XML(CONTENT(XMLSCHEMA)) whose XML type descriptor includes the registered XML Schema descriptor of *S* and the XML namespace URI ***NS***.

VIII) If there exists a registered XML Schema *S* such that, for all *i*, 1 (one) ≤ *i* ≤ *N*, T_i is either XML(DOCUMENT(XMLSCHEMA)) or XML(CONTENT(XMLSCHEMA)) and the XML type descriptor of T_i includes the registered XML Schema descriptor of *S*, then the result data type is XML(CONTENT(XMLSCHEMA)) whose XML type descriptor includes the registered XML Schema descriptor of *S*.

IX) Otherwise, the result data type is XML(CONTENT(ANY)).

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10.4 Type precedence list determination

This Subclause modifies Subclause 9.5, "Type precedence list determination", in ISO/IEC 9075-2.

Function

Determine the type precedence list of a given type.

Syntax Rules

1) Insert this SR If *DT* is an XML type, then *TPL* is XML.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10.5 Type name determination

This Subclause modifies Subclause 9.7, "Type name determination", in ISO/IEC 9075-2.

Function

Determine an <identifier> given the name of a predefined data type.

Syntax Rules

1) Augment SR 2) If *DT* specifies XML, then let *FNSDT* be "XML".

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2008 – All rights reserved

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10.6 Determination of identical values

This Subclause modifies Subclause 9.8, "Determination of identical values", in ISO/IEC 9075-2.

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 before GR 2)d) If *V1* and *V2* are both of the XML type, then whether *V1* and *V2* are identical or not identical is defined as follows:

   a) Two XQuery sequences are identical if they have the same number of XQuery items and their corresponding XQuery items are identical.

   b) Two XQuery atomic values ***XQAV1*** and ***XQAV2*** are identical if they are labeled with the same XQuery atomic type ***XQAT*** and

      Case:

      i) If ***XQAT*** is **xs:duration**, or an XML Schema type derived from **xs:duration**, then let ***YM1*** and ***YM2*** be XQuery atomic values of type **xs:yearMonthDuration** whose year and month components equal the year and month components of ***XQAV1*** and ***XQAV2***, respectively, and let ***DT1*** and ***DT2*** be XQuery atomic values of type **xs:dayTimeDuration** whose day, hour, minute, and second components equal the day, hour, minute, and second components of ***XQAV1*** and ***XQAV2***, respectively. ***XQAV1*** and ***XQAV2*** are identical values if ***YM1*** equals ***YM2*** and ***DT1*** equals ***DT2***.

      ii) Otherwise, the result of comparing ***XQAV1*** and ***XQAV2*** using the XQuery "eq" operation is *True*.

   c) Two XQuery nodes ***XQN1*** and ***XQN2*** are defined to be identical if the XQuery node identity of ***XQN1*** is the same as the XQuery node identity of ***XQN2***.

      NOTE 49 — This definition is equivalent to the XQuery expression "***XQN1*** **is** ***XQN2***".

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10.7 Determination of equivalent XML values

Function

Determine whether two instances of XML values are equivalent.

NOTE 50 — This Subclause is implicitly invoked whenever the word *equivalent* is used with respect to XML values.

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let *V1* and *V2* be two XML values specified in an application of this Subclause.

2) Whether *V1* and *V2* are equivalent or not equivalent is defined as follows:

NOTE 51 — This definition involves the simultaneous recursive definition of equivalent XQuery items and identical properties of XQuery nodes, as well as equivalent XML values.

a) Two XQuery sequences are equivalent if they have the same number of XQuery items and their corresponding XQuery items are equivalent.

b) Two XQuery atomic values are equivalent if they are identical.

c) Two XQuery nodes ***XN1*** and ***XN2*** are defined to be equivalent if they satisfy the following conditions:

i) ***XN1*** and ***XN2*** are the same kind of XQuery node (both are an XQuery document node, an XQuery element node, an XQuery attribute node, an XQuery processing instruction node, an XQuery text node, an XQuery comment node, or an XQuery namespace node).

ii) If ***XN1*** is not a document node, then either of the following conditions holds:

1) The **parent** property of *XN1* and the **parent** property of ***XN2*** are both empty.

2) The XQuery node ***P1*** that is the **parent** property of ***XN1*** is equivalent to the XQuery node ***P2*** that is the **parent** property of ***XN2***, and if ***XN1*** and ***XN2*** are not XQuery attribute nodes or XQuery namespace nodes, then ***XN1*** and ***XN2*** have the same ordinal position in the **children** property of ***P1*** and ***P2***, respectively.

iii) Every significant property ***P*** of ***XN1*** is identical to the corresponding property ***P*** of ***XN2***. The significant and insignificant properties of XQuery nodes are shown in Table 4, "XQuery node properties".

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 4 — XQuery node properties

XQuery Node Type	Significant Properties	Insignificant Properties
document	children unparsed entities	base-uri document-uri
element	node-name parent type children attributes nilled	namespaces base-URI
attribute	node-name string value parent type	
processing instruction	target content parent	base-URI
text	content parent	
comment	content parent	
namespace	uri parent	prefix

d) A property ***P*** of an XQuery node ***XN1*** is identical to the same property ***P*** of another XQuery node ***XN2*** if

Case:

i) If ***P*** is the **children** property or the **parent** property, then the XQuery sequences that are returned by the XQuery accessor of property ***P*** applied to ***XN1*** and ***XN2*** are identical.

ii) If ***P*** is the **unparsed-entities** property or the **attributes** property, then the number of XQuery items returned by the XQuery accessor of property ***P*** applied to ***XN1*** equals the number of XQuery items returned by the XQuery accessor of property ***P*** applied to ***XN2***, and there exists a one-to-one mapping of the XQuery items returned by the XQuery accessor of property ***P*** applied to ***XN1*** to the XQuery items returned by the XQuery accessor of property ***P*** applied to ***XN1*** such that corresponding XQuery items are identical.

iii) If ***P*** is the **node-name** property, **type-name** property, **nilled** property, **string-value** property, **uri** property, **target** property, or **content** property, then the XQuery atomic value that is returned

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

by the XQuery accessor of property ***P*** applied to ***XN1*** is identical to the XQuery atomic value that is returned by the XQuery accessor of property ***P*** applied to ***XN2***.

NOTE 52 — The **content** and **uri** properties are accessed by the ***dm*:string-value** XQuery accessor, and the **target** property is accessed by the ***dm*:node-name** XQuery accessor. Otherwise, the name of the XQuery accessor is the same as the name of the property.

e) An XQuery atomic value and an XQuery node are not equivalent.

Conformance Rules

None.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10.8 Equality operations

This Subclause modifies Subclause 9.9, "Equality operations", in ISO/IEC 9075-2.

Function

Specify the prohibitions and restrictions by data type on operations that involve testing for equality.

Syntax Rules

1) Insert this SR The declared type of an operand of an equality operation shall not be XML-ordered.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10.9 Grouping operations

This Subclause modifies Subclause 9.10, "Grouping operations", in ISO/IEC 9075-2.

Function

Specify the prohibitions and restrictions by data type on operations that involve grouping of data.

Syntax Rules

1) Insert this SR The declared type of an operand of a grouping operation shall not be XML-ordered.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10.10 Multiset element grouping operations

This Subclause modifies Subclause 9.11, "Multiset element grouping operations", in ISO/IEC 9075-2.

Function

Specify the prohibitions and restrictions by data type on the declared element type of a multiset for operations that involve grouping the elements of a multiset.

Syntax Rules

1) Insert this SR The declared element type of a multiset operand of a multiset element grouping operation shall not be XML-ordered.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10.11 Ordering operations

This Subclause modifies Subclause 9.12, "Ordering operations", in ISO/IEC 9075-2.

Function

Specify the prohibitions and restrictions by data type on operations that involve ordering of data.

Syntax Rules

1) Insert this SR The declared type of an operand of an ordering operation shall not be XML-ordered.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10.12 Determination of namespace URI

Function

Determine the XML namespace URI of an XML QName.

Syntax Rules

1) Let ***QN*** be the *QNAME* in an application of this Subclause, and let *B* be the *BNFTERM* in an application of this Subclause.

2) Case:

   a) If ***QN*** has an XML QName prefix, then:

      i) Let ***P*** be the XML QName prefix of ***QN***.

      ii) Case:

         1) If ***P*** is equivalent to **'xml'**, then let *NSURI* be

            http://www.w3.org/XML/1998/namespace

         2) Otherwise, *B* shall be within the scope of one or more <XML namespace declaration>s that contain an <XML namespace prefix> equivalent to ***P*** or ***P*** shall be equivalent to one of **'sqlxml'**, **'xs'**, or **'xsi'**.

            Case:

            A) If *B* is within the scope of one or more <XML namespace declaration>s that contain an <XML namespace prefix> equivalent to ***P***, then:

               I) Let *XND* be the <XML namespace declaration> that contains an <XML namespace prefix> equivalent to ***P*** with innermost scope that includes *B*.

               II) Let *XNDI* be the <XML namespace declaration item> of *XND* that contains an <XML namespace prefix> equivalent to ***P***.

               III) Let *NSURI* be the <XML namespace URI> contained in *XNDI*.

            B) Otherwise,

               Case:

               I) If ***P*** is equivalent to **'sqlxml'**, then let *NSURI* be

                  http://standards.iso.org/iso/9075/2003/sqlxml

               II) If ***P*** is equivalent to **'xs'**, then let *NSURI* be

                  http://www.w3.org/2001/XMLSchema

               III) If ***P*** is equivalent to **'xsi'**, then let *NSURI* be

                  http://www.w3.org/2001/XMLSchema-instance

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

NOTE 53 — The XML namespace prefixes **fn**, **xs**, and **local**, which are predeclared in [XQuery], if present in an <XML element name> or an <XML attribute name>, need to be in the scope of one or more <XML namespace declaration>s.

b) Otherwise,

Case:

i) If *B* is contained within the scope of one or more <XML namespace declaration>s that contain an <XML default namespace declaration item>, then:

1) Let *XDNDI* be the <XML default namespace declaration item> with innermost scope that includes *B*.

2) Case:

A) If *XDNDI* contains an <XML namespace URI>, then let *NSURI* be that <XML namespace URI>.

B) Otherwise, let *NSURI* be the zero-length string.

ii) Otherwise, let *NSURI* be the zero-length string.

3) Let *CS* be the character set of the declared type of *NSURI*. Let *CSM* be the implementation-defined mapping of strings of *CS* to strings of Unicode. Let *R* be the result of mapping *NSURI* to a string of Unicode using *CSM*.

4) *R* is the result of the Syntax Rules of this Subclause.

Access Rules

None.

General Rules

None.

Conformance Rules

None.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10.13 Construction of an XML element

Function

Construct an XML value consisting of a single XQuery element node.

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let ***QN*** be the *QNAME* in an application of this Subclause (an XML 1.1 QName), let ***NSURI*** be the *NAMESPACE* in an application of this Subclause (an XML namespace URI), let ***ATTRS*** be the *ATTRIBUTES* in an application of this Subclause (a set of XQuery attribute nodes), let *CON* be the *CONTENTS* in an application of this Subclause (a list of values), let *OPT* be the *OPTION* in an application of this Subclause, and let *ENC* be the *ENCODING* in an application of this Subclause (an indication of whether binary strings are to be encoded in base64 or hex).

2) Let *N* be the number of values in *CON*. Let V_1, ... , V_N be an enumeration of the values in *CON*.

3) Case:

   a) If $N > 0$ (zero) and, for every *i* between 1 (one) and *N*, V_i is the null value, then let *ALLNULL* be *True*.

   b) Otherwise, let *ALLNULL* be *False*.

4) Case:

   a) If *ALLNULL* is *True* and *OPT* is NULL ON NULL, then the result is the null value.

   b) If *ALLNULL* is *True* and *OPT* is ABSENT ON NULL, then the result is an empty XQuery sequence.

   c) Otherwise,

      i) For each *j*, 1 (one) $\le j \le N$, such that V_j is not the null value:

         1) Case:

            A) If the most specific type of V_j is an XML type, then let $\mathbf{CEC}_j$ be a variable whose value is identical to V_j.

            B) Otherwise, let $\mathbf{XMLV}_j$ be the result of applying the General Rules of Subclause 9.8, "Mapping values of SQL data types to values of XML Schema data types", to V_j as *the value of an SQL data type*, *ENC* as *ENCODING*, "absent" as *NULLS*, and *True* as *CHARMAPPING*. $\mathbf{XMLV}_j$ is a character string of Unicode characters. Let $\mathbf{CEC}_j$ be the result of

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
XMLPARSE (CONTENT XMLV_j PRESERVE WHITESPACE)
```

2) Let ***S***$_j$ be the result of applying Subclause 10.17, "Removing XQuery document nodes from an XQuery sequence", with ***CEC***$_j$ as the SEQUENCE.

ii) Let *NILLED* be defined as follows.

Case:

1) If *OPT* is NIL ON NULL and *ALLNULL* is *True*, then *NILLED* is *True*.

2) If *OPT* is NIL ON NO CONTENT and there does not exist at least one *j*, 1 (one) ≤ *j* ≤ *N*, such that V_j is an XQuery element node or an XQuery text node, then *NILLED* is *True*.

3) Otherwise, *NILLED* is *False*.

iii) Case:

1) If *NILLED* is *True*, then:

A) Let ***XSIURI*** be the XML namespace URI given in Table 2, "XML namespace prefixes and their URIs", corresponding to the XML namespace prefix **xsi**.

B) Let ***NILAIII*** be an XQuery attribute node whose properties are set as follows:

I) The **node-name** property is an XQuery atomic value of XQuery atomic type **xs:QName**, whose local name is **nil** and whose namespace URI is ***XSIURI***.

II) The **string-value** property is **true**.

III) The **type** property is **xs:untypedAtomic**.

IV) The **parent** property is empty.

2) Otherwise, let ***NILAII*** be the empty XQuery sequence.

iv) Let ***S*** be the XQuery sequence formed by concatenating ***ATTRS***, ***NILAII***, and the ***S***$_j$ in order from *j* = 1 (one) through *j* = *N*.

v) Let ***XSC*** be an XQuery static context created by applying the Syntax Rules of Subclause 10.20, "Creation of an XQuery expression context", with the <XML element> or <XML forest> that invoked the current Subclause as the *BNF*. The validation mode of ***XSC*** is set to an implementation-defined value.

vi) Let ***XDC*** be an XQuery dynamic context created by applying the General Rules of Subclause 10.20, "Creation of an XQuery expression context".

vii) Let ***XSC*** and ***XDC*** be augmented with the following XQuery variables:

1) An XQuery variable **$QNAME** whose XQuery formal type notation is **xs:QName** and whose value is a QName whose local name is the XML 1.1 QName local part of ***QN*** and whose namespace URI is ***NSURI***.

2) An XQuery variable **$SEQ**, whose XQuery formal type notation is **(element of type xs:anyType | attribute of type xs:anySimpleType | text | comment | processing-instruction | xs:anyAtomicType | document**

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

{ (element of type xs:anyType | comment | processing-instruction | text) * }) * and whose value is ***S***.

NOTE 54 — If the implementation does not support Feature X211, "XML 1.1 support", then any Names, NCNames, and QNames found in either **$QNAME** or **$SEQ** will already be XML 1.0 Names, NCNames, and QNames.

viii) Case:

1) If the SQL-implementation supports Feature X211, "XML 1.1 support", then let ***EI*** be the result of an XQuery evaluation with XML 1.1 lexical rules, using ***XSC*** and ***XDC*** as the XQuery expression context, of the XQuery expression

```
element { $QNAME } { $SEQ }
```

If an XQuery error occurs during the evaluation, then an exception condition is raised: *XQuery error*.

2) Otherwise, let ***EI*** be the result of an XQuery evaluation with XML 1.0 lexical rules, using ***XSC*** and ***XDC*** as the XQuery expression context, of the XQuery expression

```
element { $QNAME } { $SEQ }
```

If an XQuery error occurs during the evaluation, then an exception condition is raised: *XQuery error*.

ix) The result is an XQuery element node that is equivalent to ***EI***.

Conformance Rules

None.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10.14 Concatenation of two XML values

Function

Specify the result of the concatenation of two XML values.

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let *X* and *Y* be *the two XML values*.

2) Case:

   a) If *X* is the null value, then let *C* be *Y*.

   b) If *Y* is the null value, then let *C* be *X*.

   c) Otherwise, let *C* be the XQuery sequence consisting of every XQuery item of *X*, in order, followed by every XQuery item of *Y*, in order.

3) *C* is the result of the concatenation of *X* and *Y*.

Conformance Rules

None.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10.15 Serialization of an XML value

Function

Specify the serialization of an XML value.

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let *V* be the XML value *VALUE* in an application of this Subclause. Let *DC* be the *SYNTAX* in an application of this Subclause (*i.e.*, either DOCUMENT or CONTENT). Let *DT* be the string type *TYPE* in an application of this Subclause. Let *CS* be the *ENCODING* in an application of this Subclause. Let *B* be the *BOMDIA* in an application of this Subclause (*i.e.*, either *True*, *False*, or *Unknown*). Let *VP* be the *VERSION* in an application of this Subclause. Let *DECL* be the *XMLDECLARATION* in an application of this Subclause (*i.e.*, either *True*, *False*, or *Unknown*). Let *IND* be the *INDENT* in an application of this Subclause (*i.e.*, either *True* or *False*).

2) If *V* is the null value, then the result is the null value and no further General Rules are applied.

3) If *DC* is DOCUMENT and *V* is not an XQuery sequence of length 1 (one) whose sole XQuery item is an XQuery document node whose **children** property contains exactly one XQuery element node, zero or more XQuery comment nodes, and zero or more XQuery processing instruction nodes, then an exception condition is raised: *data exception — not an XML document.*

4) Case:

   a) If *B* is *True*, then let ***BOM*** be **yes**.

   b) If *B* is *False*, then let ***BOM*** be **no**.

   c) Otherwise, *B* is *Unknown*, and

      Case:

      i) If *DT* is a binary string type, then

         Case:

         1) If *CS* is UTF16, then let ***BOM*** be **yes**.

         2) Otherwise, it is implementation-defined whether ***BOM*** is **yes** or **no**.

            NOTE 55 — The purpose of the previous rule is to ensure that a Byte Order Mark (BOM) can be prefixed to the serialized value when needed because of the specified encoding.

      ii) Otherwise, let ***BOM*** be **no**.

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

5) Let ***IND*** be "no".

6) Let ***METH*** be an XML 1.0 QName whose XML 1.0 QName prefix is the zero-length string and whose XML 1.0 QName local part is "**xml**".

7) It is implementation-defined whether ***SA*** is **yes**, **no**, or **none**.

8) Case:

   a) If *DECL* is *True*, then let ***OXD*** be **no**.

   b) If *DECL* is *False*, then let ***OXD*** be **yes**.

   c) Otherwise (*DECL* is *Unknown*),

      Case:

      i) If *DC* is DOCUMENT and any of the following is true:

         1) *CS* is neither UTF8 nor UTF16.

         2) ***SA*** is not **none**.

         3) *VP* is not "1.0".

         then let ***OXD*** be **no**.

      ii) If *DC* is DOCUMENT and all of the following are true:

         1) *CS* is either UTF8 or UTF16.

         2) ***SA*** is **none**.

         3) *VP* is "1.0".

         then it is implementation-defined whether ***OXD*** is **yes** or **no**.

      iii) Otherwise, let ***OXD*** be **yes**.

9) Case:

   a) If *VP* is "1.0", then let ***UN*** be **no**.

   b) Otherwise, it is implementation-defined whether ***UN*** is **yes** or **no**.

10) Let ***CSE***, ***DP***, ***DS***, ***EUA***, ***ICT***, ***MT***, ***NF***, and ***UCM*** be implementation-dependent values that are permitted values for the cdata-section-elements, doctype-public, doctype-system, escape-uri-attributes, include-content-type, media-type, normalization-form, and use-character-maps serialization parameters, respectively, as defined in Section 3, "Serialization Parameters", of [Serialization].

11) Case:

   a) If *DT* is a character string type, then:

      i) Let ***CV*** be the result of applying Section 5, "XML output method", of [Serialization] to *V* with the following values for the serialization parameters:

         1) ***BOM*** as the byte-order-mark.

         2) ***CSE*** as the cdata-section-elements.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3) ***DP*** as the doctype-public.

4) ***DS*** as the doctype-system.

5) ***CS*** as the encoding.

6) ***EUA*** as the escape-uri-attributes.

7) ***ICT*** as the include-content-type.

8) ***IND*** as the indent.

9) ***MT*** as the media-type.

10) ***METH*** as the method.

11) ***NF*** as the normalization-form.

12) ***OXD*** as the omit-xml-declaration.

13) ***SA*** as the standalone.

14) ***UN*** as the undeclare-prefixes.

15) ***UCM*** as the use-character-maps.

16) ***VP*** as the version.

If an XQuery serialization error occurs during the application of Section 5, "XML output method", of [Serialization] to *V*, then an exception condition is raised: *data exception — XQuery serialization error*.

ii) The serialization of *V* is the result of an application of the Rules of Subclause 10.2, "Store assignment", with *CV* as *VALUE* and a site of declared type *DT* as *TARGET*.

b) Otherwise (*DT* is a binary string type):

i) Let *CV* be a binary string equivalent to the result of applying Section 5, "XML output method", of [Serialization] to *V* with the following values for the serialization parameters:

1) ***BOM*** as the byte-order-mark.

2) ***CSE*** as the cdata-section-elements.

3) ***DP*** as the doctype-public.

4) ***DS*** as the doctype-system.

5) ***CS*** as the encoding.

6) ***EUA*** as the escape-uri-attributes.

7) ***ICT*** as the include-content-type.

8) ***IND*** as the indent.

9) ***MT*** as the media-type.

10) ***METH*** as the method.

11) ***NF*** as the normalization-form.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

12) ***OXD*** as the omit-xml-declaration.

13) ***SA*** as the standalone.

14) ***UN*** as the undeclare-prefixes.

15) ***UCM*** as the use-character-maps.

16) ***VP*** as the version.

If an XQuery serialization error occurs during the application of Section 5, "XML output method", of [Serialization] to *V*, then an exception condition is raised: *data exception — XQuery serialization error*.

ii) The serialization of *V* is the result of an application of the Rules of Subclause 10.2, "Store assignment", with *CV* as *VALUE* and a site of declared type *DT* as *TARGET*.

Conformance Rules

None.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10.16 Parsing a string as an XML value

Function

Parse a character string or a binary string to obtain an XML value.

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let *DC* be the *SYNTAX* (either DOCUMENT or CONTENT), let *V* be the value of the string *TEXT*, and let *WO* be the value of the *OPTION* (either PRESERVE WHITESPACE or STRIP WHITESPACE) in an application of this Subclause.

2) A string is called a *textual XML 1.0 document* if the string conforms to the definition of a well-formed XML document as defined in [XML 1.0], as modified by [Namespaces 1.0].

3) A string is called a *textual XML 1.1 document* if the string conforms to the definition of a well-formed XML document as defined in [XML 1.1], as modified by [Namespaces 1.1].

4) A string is called a *textual XML 1.0 content* if any of the following is true:

   a) The character string is a textual XML 1.0 document.

   b) The character string conforms to the definition of a well-formed external parsed entity as defined in [XML 1.0], as modified by [Namespaces 1.0].

5) A string is called a *textual XML 1.1 content* if any of the following is true:

   a) The character string is a textual XML 1.0 document.

   b) The character string conforms to the definition of a well-formed external parsed entity as defined in [XML 1.0], as modified by [Namespaces 1.0].

6) Case:

   a) If the SQL-implementation supports Feature X211, “XML 1.1 support”, then

      Case:

      i) If *DC* is DOCUMENT and *V* is neither a textual XML 1.0 document nor a textual XML 1.1 document, then an exception condition is raised: *data exception — invalid XML document*.

      ii) If *DC* is CONTENT and *V* is neither a textual XML 1.0 content nor a textual XML 1.1 content, then an exception condition is raised: *data exception — invalid XML content*.

   b) Otherwise,

© ISO/IEC 2008 – All rights reserved
--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Case:

i) If *DC* is DOCUMENT and *V* is not a textual XML 1.0 document, then an exception condition is raised: *data exception — invalid XML document*.

ii) If *DC* is CONTENT and *V* is not a textual XML 1.0 content, then an exception condition is raised: *data exception — invalid XML content*.

7) *V* is parsed and a collection ***C*** of XML information items is produced according to the rules of [Infoset], with the following modifications:

a) The generation of the XML document information item ***XRII*** is modified as follows:

i) If *DC* is CONTENT, then the **[children]** property of the XML document information item consists of the list of XML information items that would be produced using the rules of [Infoset] corresponding to the BNF nonterminal content defined in rule [43] of [XML 1.0] or of [XML 1.1].

NOTE 56 — It is not an error for the XML document information item to contain zero or more than one XML element information item in this case, or for it to contain XML character information items.

ii) The **[notations]** property is implementation-defined.

iii) The **[unparsed entities]** property is implementation-defined.

b) The **[base URI]** property of any XML information item is implementation-dependent.

c) References to entities that are defined in an internal DTD are replaced by their expanded form.

d) Default values defined by an internal DTD are applied.

e) Support for an external DTD is implementation-defined.

8) If *WO* is STRIP WHITESPACE, then:

a) An XML element information item ***EII*** contained in ***C*** is *potentially whitespace-strippable* if any of the following is true:

i) ***EII*** is contained in the **[children]** property of ***XRII*** and ***EII*** does not have an **[attributes]** property that contains an XML attribute information item for which all of the following are true:

1) The **[local name]** property is **space**.

2) The **[namespace name]** property is **http://www.w3.org/XML/1998/namespace**.

3) The **[normalized value]** property is **preserve**.

ii) ***EII*** has an **[attributes]** property that contains an XML attribute information item for which all of the following are true:

1) The **[local name]** property is **space**.

2) The **[namespace name]** property is **http://www.w3.org/XML/1998/namespace**.

3) The **[normalized value]** property is **default**.

iii) ***EII*** is contained in the **[children]** property of a potentially whitespace-strippable XML element information item and ***EII*** does not have an **[attributes]** property that contains an XML attribute information item for which all of the following are true:

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

1) The **[local name]** property is **space**.

2) The **[namespace name]** property is **http://www.w3.org/XML/1998/namespace**.

3) The **[normalized value]** property is **preserve**.

b) For every potentially whitespace-strippable XML element information item ***PWSEII*** contained in ***C***:

i) Let *N* be the cardinality of the **[children]** property of ***PWSEII***. Let $\boldsymbol{XII}_i$, 1 (one) ≤ *i* ≤ *N*, be the list of XML information items that is the **[children]** property of ***PWSEII***.

ii) For every *i* between 1 (one) and *N*, if $\boldsymbol{XII}_i$ is an XML character information item, then:

1) Let *j* be the least subscript less than or equal to *i* such that for all *q* between *j* and *i*, $\boldsymbol{XII}_q$ is an XML character information item.

2) Let *k* be the greatest subscript greater than or equal to *i* such that for all *q* between *i* and *k*, XII_q is an XML character information item.

NOTE 57 — Thus the list $\boldsymbol{XII}_j, \ldots, \boldsymbol{XII}_k$ is the maximal sublist of $\boldsymbol{XII}_1, \ldots, \boldsymbol{XII}_N$ containing $\boldsymbol{XII}_i$ and consisting entirely of XML character information items. Such a maximal list of XML character information items is commonly called a "text node".

3) If for all *q* between *j* and *k*, $\boldsymbol{XII}_q$ is an XML character information item whose [character code] property is a whitespace character, then $\boldsymbol{XII}_q$ is marked for removal from ***C***.

iii) For every *i* between 1 (one) and *N*, if $\boldsymbol{XII}_i$ is marked for removal from ***C***, then $\boldsymbol{XII}_i$ is removed from ***C***.

c) Let *N* be the cardinality of the **[children]** property of ***XRII***. Let $\boldsymbol{XII}_i$, 1 (one) ≤ *i* ≤ *N*, be the list of XML information items that is the **[children]** property of ***XRII***.

i) For every *i* between 1 (one) and *N*, if $\boldsymbol{XII}_i$ is an XML character information item, then:

1) Let *j* be the least subscript less than or equal to *i* such that for all *q* between *j* and *i*, $\boldsymbol{XII}_q$ is an XML character information item.

2) Let *k* be the greatest subscript greater than or equal to *i* such that for all *q* between *i* and *k*, $\boldsymbol{XII}_q$ is an XML character information item.

3) If for all *q* between *j* and *k*, $\boldsymbol{XII}_q$ is an XML character information item whose **[character code]** property is a whitespace character, then $\boldsymbol{XII}_q$ is marked for removal from ***C***.

ii) For every *i* between 1 (one) and *N*, if $\boldsymbol{XII}_i$ is marked for removal from ***C***, then $\boldsymbol{XII}_i$ is removed from ***C***.

9) ***C*** is converted to an XQuery document node ***D*** (and its children) according to the rules of [XQueryDM] section 6, "Nodes".

10) ***D*** is the result of parsing *V* as an XML value.

Conformance Rules

None.

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10.17 Removing XQuery document nodes from an XQuery sequence

Function

Replace each XQuery document node in a sequence by the sequence of children of the XQuery document node.

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let *S* be the *SEQUENCE* in an application of this Subclause.

2) Case:

   a) If *S* is the null value, then the result of this Subclause is the null value.

   b) Otherwise:

      i) Let *N* be the number of XQuery items in *S*.

      ii) Let $\boldsymbol{I}_j$, 1 (one) ≤ *j* ≤ *N*, be an enumeration in order of the XQuery items in *S*.

      iii) For each *j*, 1 (one) ≤ *j* ≤ *N*,

         Case:

         1) If $\boldsymbol{I}_j$ is an XQuery document node, then let $\boldsymbol{C}_j$ be the XQuery sequence that is the **children** property of $\boldsymbol{I}_j$.

         2) Otherwise, let $\boldsymbol{C}_j$ be $\boldsymbol{I}_j$.

      iv) Let ***R*** be the concatenation of the XQuery sequences $\boldsymbol{C}_j$ in order from *j* = 1 (one) through *j* = *N*.

      v) ***R*** is the result of this Subclause.

Conformance Rules

None.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

10.18 Constructing a copy of an XML value

Function

Construct a copy of an XML value.

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let *V* be the *VALUE* (a value of an XML type) in an application of this Subclause.

2) Let *N* be the number of XQuery items in *V*.

3) Let $\boldsymbol{I}_j$, 1 (one) ≤ *j* ≤ *N*, be an enumeration in order of the XQuery items in *V*.

4) For each *j*, 1 (one) ≤ *j* ≤ *N*, let $\boldsymbol{C}_j$ be defined as follows.

   Case:

   a) If $\boldsymbol{I}_j$ is an XQuery atomic value, then let $\boldsymbol{C}_j$ be $\boldsymbol{I}_j$.

   b) Otherwise, let $\boldsymbol{C}_j$ be a value of XML type that is equivalent to $\boldsymbol{I}_j$ except that the **parent** property (if any) of $\boldsymbol{C}_j$ is **empty** and the XQuery node identity of $\boldsymbol{C}_j$ is different from the XQuery node identity of $\boldsymbol{I}_j$.

5) Let $\boldsymbol{R}$ be the XQuery sequence enumerated by $\boldsymbol{C}_j$, 1 (one) ≤ *j* ≤ *N*.

6) $\boldsymbol{R}$ is the result of this Subclause.

Conformance Rules

None.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10.19 Constructing an unvalidated XQuery document node

Function

Convert an XQuery document node to a value of type XML(CONTENT(UNTYPED)).

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let ***XDN*** be the *XQuery document node* in an application of this Subclause.

2) Let ***R*** be an XQuery document node that is equivalent to ***XDN*** except:

   a) The XQuery node identity of ***R*** is different from the XQuery node identity of ***XDN***.

   b) The **type-name** property of every XQuery element node in the XQuery tree rooted by ***R*** is `xs:untyped`.

   c) The **nilled** property of every XQuery element node in the XQuery tree rooted by ***R*** is `false`.

   d) The **type-name** property of every XQuery attribute node in the XQuery tree rooted by ***R*** is `xs:untypedAtomic`.

3) ***R*** is the result of this Subclause.

Conformance Rules

None.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10.20 Creation of an XQuery expression context

Function

Create an XQuery expression context.

Syntax Rules

1) Let *BNF* be the BNF non-terminal (possibly omitted) specified in an invocation of this Subclause.

2) Let ***XSC*** be an XQuery static context, created as follows:

   a) ***XSC*** is initially created as defined by [XQuery] Appendix C.1, "Static context components", as specified in the table titled "Static context components" in the column headed "Default initial value".

   b) The statically known namespaces component of ***XSC*** is augmented with the XML namespace prefix/URI pair consisting of

      ```
      ('sqlxml', "http://standards.iso.org/iso/9075/2003/sqlxml")
      ```

   c) ***XSC*** is modified with implementation-defined values, as permitted by [XQuery] Appendix C.1, "static context components", in the column headed "Can be overwritten or augmented by implementation?".

   d) If *BNF* is specified, then:

      i) The statically known namespaces component of ***XSC*** is overwritten or augmented with certain namespaces, as follows.

         1) For each <XML namespace prefix> *XNP* contained in an <XML namespace declaration> whose scope contains *BNF*, let *XND* be the <XML namespace declaration> with innermost scope containing *BNF*, and let *XNURI* be the <XML namespace URI> of the <XML regular namespace declaration item> containing *XNP* in *XND*.

         2) The pair (*XNP*, *XNURI*) is placed in the statically known namespaces component of ***XSC***, either,

            Case:

            A) If *XNP* is previously defined, then overwriting the previously defined namespace.

            B) Otherwise, augmenting the statically known namespaces component of ***XSC***.

            NOTE 58 — The scope of <XML namespace declaration>s is defined in the various Subclauses that reference that nonterminal symbol, such as Subclause 6.14, "<XML element>", and Subclause 7.2, "<query expression>".

      ii) If there is an <XML namespace declaration> whose scope includes *BNF* and that contains an <XML default namespace declaration item>, then let *XND* be the innermost such <XML namespace declaration>.

         Case:

         1) If *XND* contains an <XML default namespace declaration item> of the form "DEFAULT <XML namespace URI>", then let *XNURI* be that <XML namespace URI>.

            Case:

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

A) If *XNURI* is the zero-length string, then the default element/type namespace component of ***XSC*** is set to empty.

B) Otherwise, the default element/type namespace component of ***XSC*** is set to *XNURI*.

2) If *XND* contains an <XML default namespace declaration item> of the the form "NO DEFAULT", then the default element/type namespace component of ***XSC*** is set to empty.

e) The in-scope schema definitions component of ***XSC*** (*i.e.*, the in-scope type definitions, the in-scope element declarations, and the in-scope attribute declarations) consists of an implementation-defined collection of registered XML Schemas for which the current user has USAGE privilege.

Access Rules

None.

General Rules

1) Let ***XDC*** be an XQuery dynamic context whose components are initially set as specified in [XQuery] Appendix C.2, "Dynamic context components", in the table titled "Dynamic context components", in the columns titled "Default initial value" and "Can be overwritten or augmented by implementation?".

2) ***XDC*** is the result of this Subclause.

Conformance Rules

None.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10.21 Determination of an XQuery formal type notation

Function

Determine the XQuery formal type notation of an XQuery variable in the XQuery static context.

Syntax Rules

1) Let *S* be the *SOURCE* and let *C* be the *CONTEXT* in an invocation of this Subclause.

2) Let *SD* be the declared type of *S*.

3) Case:

   a) If *C* is *True*, then let ***SUFFIX*** be the zero-length string.

   b) If *SD* is XML(SEQUENCE), then let ***SUFFIX*** be "*******"

   c) If *S* is a column reference that references a known not null column or a <value expression> for which the SQL-implementation can deduce via an implementation-defined rule that the value of *S* cannot be null, then let ***SUFFIX*** be the zero-length string.

   d) Otherwise, let ***SUFFIX*** be "***?***".

4) If *SD* is XML(DOCUMENT(XMLSCHEMA)) or XML(CONTENT(XMLSCHEMA)), then:

   a) Let ***XS*** be the XML Schema identified by the registered XML Schema descriptor included in the XML type descriptor of *SD*.

   b) Case:

      i) If the XML type descriptor of *SD* includes the XML namespace URI and the XML NCName of a global element declaration schema component ***GE***, then let ***ET*** be the XML QName of ***GE***.

      ii) If the XML type descriptor of *SD* includes the XML namespace URI ***NS***, then let *N* be the number of global element declaration schema components included in ***NS*** and let $\boldsymbol{ET}_i$, 1 (one) $\leq i \leq N$, be the XML QNames of those global element declaration schema components.

      iii) Otherwise, let *N* be the number of global element declaration schema components included in ***XS*** and let $\boldsymbol{ET}_i$, 1 (one) $\leq i \leq N$, be the XML QNames of those global element declaration schema components.

5) Let ***XFTN*** be the XQuery formal type notation determined as follows.

   Case:

   a) If *SD* is XML(DOCUMENT(UNTYPED)), then let ***XFTN*** be

```
document { (comment | processing-instruction)*,
                element * of type xs:untyped,
                (comment | processing-instruction)* } SUFFIX
```

   b) If *SD* is XML(DOCUMENT(ANY)),then let ***XFTN*** be

```
document { (comment | processing-instruction)*,
```


© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
                element * of type xs:anyType,
                (comment | processing-instruction)* } SUFFIX
```

c) If *SD* is XML(DOCUMENT(XMLSCHEMA)), then:

i) If the type descriptor of *SD* includes the XML namespace URI and the XML NCName of a global element declaration schema component, then let ***XFTN*** be

```
document { (comment | processing-instruction)*,
                element * of type ET,
                (comment | processing-instruction)* } SUFFIX
```

ii) Otherwise, let ***XFTN*** be

```
document { (comment | processing-instruction)*,
                (element * of type ET1 | element * of type ET2 | ... | element
 * of type ETN,
                (comment | processing-instruction)* } SUFFIX
```

d) If *SD* is XML(CONTENT(UNTYPED)), then let ***XFTN*** be

```
document { ( element * of type xs:untyped | comment
               | processing-instruction | text)* } SUFFIX
```

e) If *SD* is XML(CONTENT(ANY)), then let ***XFTN*** be

```
document { ( element * of type xs:anyType | comment
               | processing-instruction | text)* } SUFFIX
```

f) If *SD* is XML(CONTENT(XMLSCHEMA)), then:

i) If the type descriptor of *SD* includes the XML namespace URI and the XML NCName of a global element declaration schema component, then let ***XFTN*** be

```
document { ( element * of type ET |
                comment | processing-instruction)* } SUFFIX
```

ii) Otherwise, let ***XFTN*** be

```
document { ( element * of type ET1 | element * of type ET2 | ... | element *
of type ETN |
                comment | processing-instruction)* } SUFFIX
```

g) If *SD* is XML(SEQUENCE), then let ***XFTN*** be

```
( element * of type xs:anyType | attribute of type xs:anySimpleType |
  text | comment | processing-instruction | xs:anyAtomicType |
  document { ( element * of type xs:anyType | comment |
  processing-instruction | text )* } ) SUFFIX
```

h) If *SD* is an SQL predefined type, then let *ENC* be an indication that the binary strings are to be encoded in hex, and let ***XMLT*** be the result of applying the General Rules of Subclause 9.5, "Mapping SQL data types to XML Schema data types", with *SD* as *SQLTYPE*, *ENC* as *ENCODING*, and "absent" as *NULLS*. Let ***PT*** be the XML Schema primitive type from which ***XMLT*** is derived.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Case:

i) If *SD* is a year-month interval, then let ***XFTN*** be "**xs:yearMonthDuration *SUFFIX***".

ii) If *SD* is a day-time interval, then let ***XFTN*** be "**xs:dayTimeDuration *SUFFIX***".

iii) If *SD* is an exact numeric type with scale 0 (zero), then let ***XFTN*** be "**xs:integer *SUFFIX***".

iv) Otherwise, let ***XFTN*** be "***PT SUFFIX***".

6) ***XFTN***, or an implementation-defined XML Schema subtype of ***XFTN*** that describes *S*, is the result of this Subclause.

Access Rules

None.

General Rules

None.

Conformance Rules

None.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

10.22 Validating an XQuery document or element node

Function

Validate an XQuery document or an XQuery element node.

Syntax Rules

None.

Access Rules

None.

General Rules

1) Let ***V*** be the *ITEM* and *SXS* be the *SCHEMA* in an application of this Subclause.

2) Let ***XSC*** be an XQuery static context created according to the Syntax Rules of Subclause 10.20, "Creation of an XQuery expression context", with the BNF non-terminal argument omitted.

3) Let ***XDC*** be an XQuery dynamic context created according to the General Rules of Subclause 10.20, "Creation of an XQuery expression context".

4) *SXS* is included in the in-scope schema definitions component of ***XSC***.

5) ***XSC*** and ***XDC*** are augmented with an XQuery variable **$SEQ**, whose XQuery formal type notation is

```
(element of type xs:anyType | attribute of type xs:anySimpleType |
 text | comment | processing-instruction | xs:anyAtomicType |
 document-node { ( element of type xs:anyType | comment |
                   processing-instruction | text )* } )*
```

   and whose value is ***V***.

6) Case:

   a) If the implementation supports Feature X211, "XML 1.1 support", then let ***EI*** be the result of an XQuery evaluation with XML 1.1 lexical rules, using ***XSC*** and ***XDC*** as the XQuery expression context, of the XQuery expression

      ```
      validate strict { $SEQ }
      ```

      If an XQuery error occurs during the evaluation, then let *STAT* be FAILURE; otherwise, let *STAT* be SUCCESS.

   b) Otherwise, let ***EI*** be the result of an XQuery evaluation with XML 1.0 lexical rules, using ***XSC*** and ***XDC*** as the XQuery expression context, of the XQuery expression

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
validate strict { $SEQ }
```

If an XQuery error occurs during the evaluation, then let *STAT* be FAILURE; otherwise, let *STAT* be SUCCESS.

7) *STAT* is returned as *STATUS*. If *STAT* is SUCCESS, then ***EI*** is returned as the *RESULT*.

Conformance Rules

None.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

11 Additional common elements

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

11.1 <routine invocation>

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

Function

Invoke an SQL-invoked routine.

Format

```
No additional Format items.
```

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

1) Replace GR 3)a) If P_i is an input SQL parameter or both an input SQL parameter and an output SQL parameter, then

   Case:

   a) If T_i is an XML type, then let CPV_i be the value of P_i after applying the General Rules of Subclause 10.2, "Store assignment", with P_i as *TARGET*, V_i as *VALUE*, and the <XML passing mechanism> of P_i as *PASSING*.

   b) Otherwise, let CPV_i be the value of P_i after applying the General Rules of Subclause 10.2, "Store assignment", with P_i as *TARGET* and V_i as *VALUE*.

2) Insert after GR 4)b)i)1) If T_i is an XML type, then:

   a) Let XDT_i be the associated string type of P_i.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

b) Let XO_i be the associated XML option of P_i.

c) It is implementation-defined whether XDO_i is INCLUDING XMLDECLARATION or EXCLUDING XMLDECLARATION.

d) Case:

i) If the character set of XDT_i is UTF16, then let BOM_i be `U&'\FEFF'`.

ii) Otherwise, let BOM_i be the zero-length string of type XDT_i.

e) CPV_i is replaced by the result of

```
BOMi || XMLSERIALIZE (XOi CPVi AS XDTi XDOi)
```

3) Insert after GR 4)b)ii)1) If T_i is an XML type, then:

a) Let XDT_i be the associated string type of P_i.

b) CPV_i is replaced by an implementation-defined value of type XDT_i.

4) Insert before GR 8)h)i)4)B)III)2)b) If *CRT* is an XML type, then:

a) Let *XO* be the associated XML option of the result of *R*.

b) It is implementation-defined whether *XWO* is STRIP WHITESPACE or PRESERVE WHITESPACE.

c) Let *RDI* be the result of

```
XMLPARSE (XO ERDI XWO)
```

5) Insert before GR 8)i)iii)3)C) If T_i is an XML type, then:

a) Let XO_i be the associated XML option of P_i.

b) It is implementation-defined whether XWO_i is STRIP WHITESPACE or PRESERVE WHITESPACE.

c) CPV_i is set to the result of

```
XMLPARSE (XOi EVi XWOi)
```

6) Replace GR 9)a)iii) Let the result of the <routine invocation> be

Case:

a) If *ERDT* is an XML type, then the value of a target *T* of declared type *ERDT* after applying the General Rules of Subclause 10.2, "Store assignment", to *T* as *TARGET*, *RV* as *VALUE*, and the <XML passing mechanism> of the <returns clause> of *R* as *PASSING*.

b) Otherwise, the value of a target *T* of declared type *ERDT* after applying the General Rules of Subclause 10.2, "Store assignment", to *T* as *TARGET* and *RV* as *VALUE*.

7) Replace GR 9)b)ii)1)B)V)1)b) The *I*-th element of *A* is set to

Case:

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

a) If *EDT* is an XML type, then the value of CPV_i, denoted by *SV*, by applying the General Rules of Subclause 10.2, "Store assignment", to the *I*-th element of *A* as *TARGET*, *SV* as *VALUE*, and the <XML passing mechanism> of P_i as *PASSING*.

b) Otherwise, the value of CPV_i, denoted by *SV*, by applying the General Rules of Subclause 10.2, "Store assignment", to the *I*-th element of *A* as *TARGET* and *SV* as *VALUE*.

8) Replace GR 9)b)ii)1)B)V)2)c) The *I*-th element of *A* is set to

Case:

a) If *EDT* is an XML type, then the value of CPV_i, denoted by *SV*, by applying the General Rules of Subclause 10.2, "Store assignment", to the *I*-th element of *A* as *TARGET*, *SV* as *VALUE*, and the <XML passing mechanism> of P_i as *PASSING*.

b) Otherwise, the value of CPV_i, denoted by *SV*, by applying the General Rules of Subclause 10.2, "Store assignment", to the *I*-th element of *A* as *TARGET* and *SV* as *VALUE*.

9) Replace GR 9)b)ii)2) Otherwise,

Case:

a) If the declared type of P_i is an XML type, then the General Rules of Subclause 10.2, "Store assignment", are applied to TS_i as *TARGET*, CPV_i as *VALUE*, and the <XML passing mechanism> of P_i as *PASSING*.

b) Otherwise, the General Rules of Subclause 10.2, "Store assignment", are applied to TS_i as *TARGET* and CPV_i as *VALUE*.

Conformance Rules

No additional Conformance Rules.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

11.2 <aggregate function>

This Subclause modifies Subclause 10.9, "<aggregate function>", in ISO/IEC 9075-2.

Function

Specify a value computed from a collection of rows.

Format

```
<aggregate function> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <XML aggregate>

<XML aggregate> ::=
  XMLAGG <left paren> <XML value expression>
      [ ORDER BY <sort specification list> ]
      [ <XML returning clause> ]
      <right paren>
```

Syntax Rules

1) Insert this SR If <XML aggregate> is specified, then:

   a) If <XML returning clause> is not specified, then it is implementation-dependent whether RETURNING CONTENT or RETURNING SEQUENCE is implicit.

   b) Case:

      i) If RETURNING CONTENT is specified or implied, then

         Case:

         1) If the declared type of the <XML value expression> is XML(DOCUMENT(UNTYPED)) or XML(CONTENT(UNTYPED)), then the declared type of <XML aggregate> is XML(CONTENT(UNTYPED)).

         2) If the declared type of the <XML value expression> is XML(DOCUMENT(XMLSCHEMA)) or XML(CONTENT(XMLSCHEMA)), then let *XVT* be the declared type of <XML value expression>, let *XS* be the registered XML Schema descriptor included in the XML type descriptor of *XVT*, let *NS* be the XML Namespace URI included in the XML type descriptor of *XVT*, if any, and let *EN* be the XML NCName of a global element declaration schema component included in the XML type descriptor of *XVT*, if any. The declared type of <XML aggregate> is XML(CONTENT(XMLSCHEMA)) whose XML type descriptor includes *XS*, if *NS* is defined, then *NS*, and, if *EN* is defined, then *EN*.

         3) Otherwise, the declared type of <XML aggregate> is XML(CONTENT(ANY)).

      ii) Otherwise, the declared type of <XML aggregate> is XML(SEQUENCE).

   c) Let *XVE* be the <XML value expression>.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

d) If <sort specification list> is specified, then let *SSL* be "ORDER BY <sort specification list>"; otherwise, let *SSL* be the zero-length string.

e) If RETURNING CONTENT is specified or implied, then

i) Let *RT* be the declared type of <XML aggregate>.

ii) <XML aggregate> is equivalent to

```
CAST (
  XMLDOCUMENT (
    XMLAGG (
      XVE SSL
      RETURNING SEQUENCE )
    RETURNING CONTENT )
  AS RT )
```

Access Rules

No additional Access Rules.

General Rules

1) Insert after GR 1)b) If the most specific type of the result of *AF* is an XML type, then an exception condition is raised: *data exception — XML value overflow*.

2) Insert this GR If <XML aggregate> is specified, then:

a) If <sort specification list> is specified, then let *K* be the number of <sort key>s; otherwise, let *K* be 0 (zero).

b) Let *TXA* be the table of *K*+1 columns obtained by applying <XML value expression> to each row of *T1* to obtain the first column of *TXA*, and, for all *i* between 1 (one) and *K*, applying the <value expression> simply contained in the *i*-th <sort key> to each row of *T1* to obtain the (*i*+1)-th column of *TXA*.

c) Every row of *TXA* in which the value of the first column is the null value is removed from *TXA*.

d) Let *TXA* be ordered according to the values of the <sort key>s found in the second through (*K*+1)-th columns of *TXA*. If *K* is 0 (zero), then the ordering of *TXA* is implementation-dependent. Let *N* be the number of rows in *TXA*. Let R_i, 1 (one) $\le i \le N$, be the rows of *TXA* according to the ordering of *TXA*.

e) Case:

i) If *TXA* is empty, then the result of <XML aggregate> is the null value.

ii) Otherwise:

1) Let V_1 be the value of the first column of R_1.

2) Let V_i, $2 \le i \le N$, be the result of applying the General Rules of Subclause 10.14, "Concatenation of two XML values", V_{i-1} and the value of the first column of R_i as *the two XML values*.

3) The result is V_N.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Conformance Rules

1) Insert this CR Without Feature X034, “XMLAgg”, conforming SQL language shall not contain an <XML aggregate>.

2) Insert this CR Without Feature X035, “XMLAgg: ORDER BY option”, conforming SQL language shall not contain an <XML aggregate> that contains a <sort specification list>.

3) Insert this CR Without Feature X241, “RETURNING CONTENT in XML publishing”, in conforming SQL language, an <XML aggregate> shall not specify an <XML returning clause> that is RETURNING CONTENT.

4) Insert this CR Without Feature X242, “RETURNING SEQUENCE in XML publishing”, in conforming SQL language, an <XML aggregate> shall not specify an <XML returning clause> that is RETURNING SEQUENCE.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

11.3 <XML lexically scoped options>

Function

Declare one or more XML namespaces and the encoding to use for binary strings.

Format

```
<XML lexically scoped options> ::=
  <XML lexically scoped option> [ <comma> <XML lexically scoped option> ]

<XML lexically scoped option> ::=
    <XML namespace declaration>
  | <XML binary encoding>

<XML namespace declaration> ::=
  XMLNAMESPACES <left paren> <XML namespace declaration item>
      [ { <comma> <XML namespace declaration item> }... ] <right paren>

<XML namespace declaration item> ::=
    <XML regular namespace declaration item>
  | <XML default namespace declaration item>

<XML namespace prefix> ::=
  <identifier>

<XML namespace URI> ::=
  <character string literal>

<XML regular namespace declaration item> ::=
  <XML namespace URI> AS <XML namespace prefix>

<XML default namespace declaration item> ::=
    DEFAULT <XML namespace URI>
  | NO DEFAULT

<XML binary encoding> ::=
  XMLBINARY [ USING ] { BASE64 | HEX }
```

Syntax Rules

1) An <XML lexically scoped options> shall contain at most one <XML namespace declaration>, and at most one <XML binary encoding>.

2) <XML namespace declaration> shall contain at most one <XML default namespace declaration item>.

3) Each <XML namespace prefix> shall be an XML 1.1 NCName.

4) No two <XML namespace prefix>es shall be equivalent.

5) No <XML namespace prefix> shall be equivalent to **`xml`** or **`xmlns`**.

6) No <XML namespace URI> shall be identical, as defined in [Namespaces], to **`http://www.w3.org/2000/xmlns/`** or to **`http://www.w3.org/XML/1998/namespace`**.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

7) The value of an <XML namespace URI> contained in an <XML regular namespace declaration item> shall not be a zero-length string.

Access Rules

None.

General Rules

None.

Conformance Rules

1) Without Feature X211, "XML 1.1 support", in conforming SQL language, each <XML namespace prefix> shall be an XML 1.0 NCName.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

11.4 <XML returning clause>

Function

Specify whether the declared type of certain <XML value function>s is XML(SEQUENCE) or some other XML type.

Format

```
<XML returning clause> ::=
  RETURNING { CONTENT | SEQUENCE }
```

Syntax Rules

None.

Access Rules

None.

General Rules

None.

Conformance Rules

None.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

11.5 <XML passing mechanism>

Function

Specify the semantics for assigning or casting an XML value.

Format

```
<XML passing mechanism> ::=
    BY REF
  | BY VALUE
```

Syntax Rules

None.

Access Rules

None.

General Rules

None.

Conformance Rules

1) Without Feature X221, “XML passing mechanism BY VALUE”, conforming SQL language shall not contain an <XML passing mechanism> that is BY VALUE.

2) Without Feature X222, “XML passing mechanism BY REF”, conforming SQL language shall not contain an <XML passing mechanism> that is BY REF.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

11.6 <XML valid according to clause>

Function

Indicate a registered XML Schema, and (optionally) an XML namespace of that registered XML Schema, and (optionally) a global element declaration schema component of that registered XML Schema.

Format

```
<XML valid according to clause> ::=
  ACCORDING TO XMLSCHEMA <XML valid according to what>
      [ <XML valid element clause> ]

<XML valid according to what> ::=
    <XML valid according to URI>
  | <XML valid according to identifier>

<XML valid according to URI> ::=
    URI <XML valid target namespace URI> [ <XML valid schema location> ]
  | NO NAMESPACE [ <XML valid schema location> ]

<XML valid target namespace URI> ::=
  <XML URI>

<XML URI> ::=
  <character string literal>

<XML valid schema location> ::=
  LOCATION <XML valid schema location URI>

<XML valid schema location URI> ::=
  <XML URI>

<XML valid according to identifier> ::=
  ID <registered XML Schema name>

<XML valid element clause> ::=
    <XML valid element name specification>
  | <XML valid element namespace specification>
      [ <XML valid element name specification> ]

<XML valid element name specification> ::=
  ELEMENT <XML valid element name>

<XML valid element namespace specification> ::=
    NO NAMESPACE
  | NAMESPACE <XML valid element namespace URI>

<XML valid element namespace URI> ::=
  <XML URI>

<XML valid element name> ::=
  <identifier>
```

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Syntax Rules

1) If <XML valid according to identifier> is specified, then the <registered XML Schema name> shall identify a registered XML Schema *RXS*. Let *NSURI* be the target namespace URI of *RXS*.

2) If <XML valid according to URI> is specified, then:

   a) Case:

      i) If NO NAMESPACE is specified, then let *NSURI* be the zero-length string.

      ii) Otherwise, let *NSURI* be the <XML valid target namespace URI>.

   b) Case:

      i) If the SQL-implementation identifies registered XML Schemas by the combination of their target namespace URIs and schema location URIs, then

         Case:

         1) If <XML valid schema location> is specified, then there shall exist a registered XML Schema whose target namespace URI is identical to *NSURI*, as defined by [Namespaces], and whose schema location URI equals <XML valid schema location URI> *XVSL* according to the UCS_BASIC collation.

            Case:

            A) If there exists more than one such registered XML Schema, then let *XSDN* be a <registered XML Schema name> chosen using a deterministic implementation-defined algorithm that is repeatable, in the sense that if the algorithm is re-evaluated with the same collection of registered XML Schemas that are accessible to the user, the same <registered XML Schema name> will be chosen.

            B) Otherwise, let *XSD* be the registered XML Schema descriptor that includes *NSURI* as its target namespace URI and *XVSL* as its schema location URI. Let *XSDN* be the <registered XML Schema name> included in *XSD*.

         2) Otherwise, a <registered XML Schema name> *XSDN* is chosen using a deterministic implementation-defined algorithm that is repeatable, in the sense that if the algorithm is re-evaluated with the same collection of registered XML Schemas that are accessible to the user, the same <registered XML Schema name> will be chosen.

            NOTE 59 — This does not say that the algorithm must choose a <registered XML Schema name> that identifies a registered XML Schema. One possibility for the implementation-defined algorithm is to choose a <registered XML Schema name> that does not identify a registered XML Schema, resulting in a syntax error.

      ii) Otherwise, there shall exist a registered XML Schema whose target namespace URI is *NSURI*.

         Case:

         1) If there exists more than one such registered XML Schema, then let *XSDN* be a <registered XML Schema name> chosen using a deterministic implementation-defined algorithm that is repeatable, in the sense that if the algorithm is re-evaluated with the same collection of registered XML Schemas that are accessible to the user, the same <registered XML Schema name> will be chosen.

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2) Otherwise, let *XSD* be the registered XML Schema descriptor that includes *NSURI* as its target namespace URI. Let *XSDN* be the <registered XML Schema name> included in *XSD*.

c) *XSDN* shall identify a registered XML Schema *RXS*.

3) *RXS* is the *indicated registered XML schema*.

4) If <XML valid element clause> is specified, then:

a) Case:

i) If <XML valid element namespace specification> is specified, then

Case:

1) If NO NAMESPACE is specified, then let *ENSURI* be the zero-length string.

2) Otherwise, let *ENSURI* be <XML valid element namespace URI>.

ii) Otherwise, let *ENSURI* be *NSURI*.

b) It is implementation-defined whether *RXS* shall have a namespace, among its unordered collection of namespaces, whose namespace name is identical to *ENSURI*, as defined by [Namespaces].

c) *ENSURI* is the *indicated XML namespace*.

d) If <XML valid element name specification> is specified, then:

i) Let *EN* be the <XML valid element name>.

ii) Let *GEDSC* be the global element declaration schema component whose namespace URI is identical to *ENSURI*, as defined by [Namespaces], and whose XML NCName is *EN*. *GEDSC* is the *indicated global element declaration schema component*.

iii) It is implementation-defined whether *RXS* shall have a global element declaration schema component whose namespace URI is identical to *ENSURI*, as defined by [Namespaces], and whose XML NCName is *EN*.

Access Rules

1) Case:

a) If the <XML valid according to clause> is contained, without an intervening <SQL routine spec> that specifies SQL SECURITY INVOKER, in an <SQL schema statement>, then the applicable privileges of the <authorization identifier> that owns the containing schema shall include USAGE on *RXS*.

b) Otherwise, the current privileges shall include USAGE on *RXS*.

General Rules

1) If <XML valid element clause> is specified, then:

a) If *RXS* does not have a namespace, among its unordered collection of namespaces, whose namespace name is identical to *ENSURI*, as defined by [Namespaces], then an exception condition is raised: *data exception — element namespace not declared*.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

b) If <XML valid element name specification> is specified and *RXS* does not have a global element declaration schema component whose namespace is identical to *ENSURI*, as defined by [Namespaces], and whose XML NCName is *EN*, then an exception condition is raised: *data exception — global element not declared*.

Conformance Rules

None.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

12 Schema definition and manipulation

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

12.1 <column definition>

This Subclause modifies Subclause 11.4, "<column definition>", in ISO/IEC 9075-2.

Function

Define a column of a base table.

Format

```
<generation expression> ::=
  <left paren> [ WITH <XML lexically scoped options> ]
      <value expression> <right paren>
```

Syntax Rules

1) Insert this SR The scope of each <XML namespace declaration item> contained in the <XML lexically scoped options> is the <generation expression>.

2) Insert this SR The scope of an <XML binary encoding> contained in the <XML lexically scoped options> is the <generation expression>.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

1) Insert this CR Without Feature X083, "XML namespace declarations in DDL", in conforming SQL language, a <generation expression> shall not immediately contain an <XML lexically scoped options> that contains an <XML namespace declaration>.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2) Insert this CR Without Feature X133, “XMLBINARY clause in DDL”, in conforming SQL language, a <generation expression> shall not immediately contain an <XML lexically scoped options> that contains an <XML binary encoding>.

3) Insert this CR Without Feature X016, “Persistent XML values”, conforming SQL language shall not contain a <column definition> whose declared type is based on either an XML type or a distinct type whose source type is an XML type.

4) Insert this CR Without Feature X251, “Persistent XML values of XML(DOCUMENT(UNTYPED)) type”, conforming SQL language shall not contain a <column definition> whose declared type is based on either the XML(DOCUMENT(UNTYPED)) type or a distinct type whose source type is the XML(DOCUMENT(UNTYPED)) type.

5) Insert this CR Without Feature X252, “Persistent XML values of XML(DOCUMENT(ANY)) type”, conforming SQL language shall not contain a <column definition> whose declared type is based on either the XML(DOCUMENT(ANY)) type or a distinct type whose source type is the XML(DOCUMENT(ANY)) type.

6) Insert this CR Without Feature X256, “Persistent XML values of XML(DOCUMENT(XMLSCHEMA)) type”, conforming SQL language shall not contain a <column definition> whose declared type is based on either the XML(DOCUMENT(XMLSCHEMA)) type or a distinct type whose source type is the XML(DOCUMENT(XMLSCHEMA)) type.

7) Insert this CR Without Feature X253, “Persistent XML values of XML(CONTENT(UNTYPED)) type”, conforming SQL language shall not contain a <column definition> whose declared type is based on either the XML(CONTENT(UNTYPED)) type or a distinct type whose source type is the XML(CONTENT(UNTYPED)) type.

8) Insert this CR Without Feature X254, “Persistent XML values of XML(CONTENT(ANY)) type”, conforming SQL language shall not contain a <column definition> whose declared type is based on either the XML(CONTENT(ANY)) type or a distinct type whose source type is the XML(CONTENT(ANY)) type.

9) Insert this CR Without Feature X257, “Persistent XML values of XML(CONTENT(XMLSCHEMA)) type”, conforming SQL language shall not contain a <column definition> whose declared type is based on either the XML(CONTENT(XMLSCHEMA)) type or a distinct type whose source type is the XML(CONTENT(XMLSCHEMA)) type.

10) Insert this CR Without Feature X255, “Persistent XML values of XML(SEQUENCE) type”, conforming SQL language shall not contain a <column definition> whose declared type is based on either the XML(SEQUENCE) type or a distinct type whose source type is the XML(SEQUENCE) type.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

12.2 <check constraint definition>

This Subclause modifies Subclause 11.9, "<check constraint definition>", in ISO/IEC 9075-2.

Function

Specify a condition for the SQL-data.

Format

```
<check constraint definition> ::=
  CHECK <left paren> [ WITH <XML lexically scoped options> ]
      <search condition> <right paren>
```

Syntax Rules

1) Insert this SR The scope of each <XML namespace declaration item> contained in the <XML lexically scoped option> is the <check constraint definition>.

2) Insert this SR The scope of an <XML binary encoding> contained in the <XML lexically scoped options> is the <check constraint definition>.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

1) Insert this CR Without Feature X083, "XML namespace declarations in DDL", in conforming SQL language, a <check constraint definition> shall not immediately contain and <XML lexically scoped options> that contains an <XML namespace declaration>.

2) Insert this CR Without Feature X133, "XMLBINARY clause in DDL", in conforming SQL language, a <check constraint definition> shall not immediately contain an <XML lexically scoped options> that contains an <XML binary encoding>.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

12.3 <view definition>

This Subclause modifies Subclause 11.23, "<view definition>", in ISO/IEC 9075-2.

Function

Define a viewed table.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

1) Insert this CR Without Feature X016, "Persistent XML values", conforming SQL language shall not contain a <view definition> that defines a column whose declared type is based on either an XML type or a distinct type whose source type is an XML type.

2) Insert this CR Without Feature X251, "Persistent XML values of XML(DOCUMENT(UNTYPED)) type", conforming SQL language shall not contain a <view definition> that defines a column whose declared type is based on either the XML(DOCUMENT(UNTYPED)) type or a distinct type whose source type is the XML(DOCUMENT(UNTYPED)) type.

3) Insert this CR Without Feature X252, "Persistent XML values of XML(DOCUMENT(ANY)) type", conforming SQL language shall not contain a <view definition> that defines a column whose declared type is based on either the XML(DOCUMENT(ANY)) type or a distinct type whose source type is the XML(DOCUMENT(ANY)) type.

4) Insert this CR Without Feature X256, "Persistent XML values of XML(DOCUMENT(XMLSCHEMA)) type", conforming SQL language shall not contain a <view definition> that defines a column whose declared type is based on either the XML(DOCUMENT(XMLSCHEMA)) type or a distinct type whose source type is the XML(DOCUMENT(XMLSCHEMA)) type.

5) Insert this CR Without Feature X253, "Persistent XML values of XML(CONTENT(UNTYPED)) type", conforming SQL language shall not contain a <view definition> that defines a column whose declared type

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

is based on either the XML(CONTENT(UNTYPED)) type or a distinct type whose source type is the XML(CONTENT(UNTYPED)) type.

6) Insert this CR Without Feature X254, "Persistent XML values of XML(CONTENT(ANY)) type", conforming SQL language shall not contain a <view definition> that defines a column whose declared type is based on either the XML(CONTENT(ANY)) type or a distinct type whose source type is the XML(CONTENT(ANY)) type.

7) Insert this CR Without Feature X257, "Persistent XML values of XML(CONTENT(XMLSCHEMA)) type", conforming SQL language shall not contain a <view definition> that defines a column whose declared type is based on either the XML(CONTENT(XMLSCHEMA)) type or a distinct type whose source type is the XML(CONTENT(XMLSCHEMA)) type.

8) Insert this CR Without Feature X255, "Persistent XML values of XML(SEQUENCE) type", conforming SQL language shall not contain a <view definition> that defines a column whose declared type is based on either the XML(SEQUENCE) type or a distinct type whose source type is the XML(SEQUENCE) type.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

12.4 <assertion definition>

This Subclause modifies Subclause 11.38, "<assertion definition>", in ISO/IEC 9075-2.

Function

Specify an integrity constraint.

Format

```
<assertion definition> ::=
  CREATE ASSERTION <constraint name> CHECK
      <left paren> [ WITH <XML lexically scoped options> ] <search condition> <right paren>
```

Syntax Rules

1) Insert this SR The scope of each <XML namespace declaration item> contained in the <XML lexically scoped options> is the <assertion definition>.

2) Insert this SR The scope of an <XML binary encoding> contained in the <XML lexically scoped options> is the <assertion definition>.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

1) Insert this CR Without Feature X083, "XML namespace declarations in DDL", in conforming SQL language, an <assertion definition> shall not immediately contain an <XML lexically scoped options> that contains an <XML namespace declaration>.

2) Insert this CR Without Feature X133, "XMLBINARY clause in DDL", in conforming SQL language, an <assertion definition> shall not immediately contain an <XML lexically scoped options> that contains an <XML binary encoding>.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

12.5 <user-defined type definition>

This Subclause modifies Subclause 11.42, "<user-defined type definition>", in ISO/IEC 9075-2.

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

No additional General Rules.

Conformance Rules

1) Insert this CR Without Feature X013, "Distinct types of XML type", conforming SQL language shall not contain a <representation> that is a <predefined type> that is an XML type.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

12.6 <attribute definition>

This Subclause modifies Subclause 11.43, "<attribute definition>", in ISO/IEC 9075-2.

Function

Define an attribute of a structured type.

Format

No additional Format items.

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

1) Insert this CR Without Feature X014, "Attributes of XML type", conforming SQL language shall not contain an <attribute definition> that contains a <data type> that is based on either an XML type or a distinct type whose source type is an XML type.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

12.7 <SQL-invoked routine>

This Subclause modifies Subclause 11.51, "<SQL-invoked routine>", in ISO/IEC 9075-2.

Function

Define an SQL-invoked routine.

Format

```
<SQL parameter declaration> ::=
  [ <parameter mode> ]
      [ <SQL parameter name> ]
      <parameter type>
      [ RESULT ]
      [ <XML passing mechanism> ]

<parameter type> ::=
  <data type> [ <locator indication> ]
      [ <document or content> ] [ <string type option> ]

<returns clause> ::=
  RETURNS <returns type> [ <XML passing mechanism> ]

<returns data type> ::=
  <data type> [ <locator indication> ]
      [ <document or content> ] [ <string type option> ]

<string type option> ::=
  AS <character string type>
```

Syntax Rules

1) Insert before SR 6)y) Case:

   a) If the <data type> immediately contained in a <parameter type> or <returns data type> is an XML type and *R* is an external routine, then:

      i) <locator indication> shall not be specified.

      ii) <document or content> and <string type option> shall be specified.

   b) Otherwise, <document or content> and <string type option> shall not be specified.

2) Insert before SR 6)y) If the <data type> immediately contained in a <returns data type> that is immediately contained in a <returns type> *RT* is an XML type, then *RT* shall not contain a <result cast>.

3) Insert after SR 20)d)iii)1)A) If the <parameter type> T_i simply contained in the *i*-th <SQL parameter declaration> P_i is an XML type, then:

   a) Let *XDT* be the data type identified by the <character string type> contained in <string type option> immediately contained in T_i. *XDT* is the *associated string type* of P_i.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

b) Let *XO* be the <document or content> immediately contained in T_i. *XO* is the *associated XML option* of P_i.

c) The *i*-th effective SQL parameter list entry is the *i*-th <SQL parameter declaration> with the <parameter type> replaced by *XDT*.

4) Insert after SR 20)d)iii)2)A)II)1) If *RT* is an XML type, then:

a) Let *XDT* be the data type identified by the <character string type> contained in <string type option> immediately contained in *RT*. *XDT* is the *associated string type* of the result of *R*.

b) Let *XO* be the <document or content> immediately contained in *RT*. *XO* is the *associated XML option* of the result of *R*.

c) *PT* is *XDT*.

5) Insert before SR 20)d)iii)2)B)II)2) If $RFT_{i\text{-}PN}$ is an XML type, then:

a) Let *XDT* be the data type identified by the <character string type> contained in <string type option> immediately contained in $RFT_{i\text{-}PN}$. *XDT* is the *associated string type* of the result of *R*.

b) Let *XO* be the <document or content> immediately contained in $RFT_{i\text{-}PN}$. *XO* is the associated XML option of the result of *R*.

c) PT_i is *XDT*.

6) Insert after SR 20)d)iv)1)A) If the <parameter type> T_i simply contained in the *i*-th <SQL parameter declaration> is an XML type, then:

a) Let *XDT* be the data type identified by the <character string type> contained in <string type option> immediately contained in T_i. *XDT* is the *associated string type* of T_i.

b) Let *XO* be the <document or content> immediately contained in T_i. *XO* is the *associated XML option* of T_i.

c) The *i*-th effective SQL parameter list entry is the *i*-th <SQL parameter declaration> with the <parameter type> replaced by *XDT*.

7) Insert after SR 20)e)i) If the <parameter type> T_i simply contained in the *i*-th <SQL parameter declaration> is an XML type, then:

a) Let *XDT* be the data type identified by the <character string type> contained in <string type option> immediately contained in T_i. *XDT* is the *associated string type* of T_i.

b) Let *XO* be the <document or content> immediately contained in T_i. *XO* is the *associated XML option* of T_i.

c) The *i*-th effective SQL parameter list entry is the *i*-th <SQL parameter declaration> with the <parameter type> replaced by *XDT*.

8) Insert this SR Case:

a) If *R* is an external routine, then <SQL parameter declaration> and <returns clause> shall not contain an <XML passing mechanism>.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

b) Otherwise:

i) If the declared type of an <SQL parameter declaration> is an XML type or a distinct type whose source type is an XML type, and the <SQL parameter declaration> does not contain an <XML passing mechanism>, then it is implementation-defined whether BY REF or BY VALUE is implicit.

ii) If the declared type of an <SQL parameter declaration> is not an XML type or a distinct type whose source type is an XML type, then the <SQL parameter declaration> shall not contain an <XML passing mechanism>.

iii) If *R* is an SQL-invoked function, the declared type of the <returns type> is an XML type or a distinct type whose source type is an XML type, and the <returns clause> does not contain an <XML passing mechanism>, then it is implementation-defined whether BY REF or BY VALUE is implicit.

iv) If *R* is an SQL-invoked function and the declared type of the <returns type> is not an XML type or a distinct type whose source type is an XML type, then the <returns clause> shall not contain an <XML passing mechanism>.

Access Rules

No additional Access Rules.

General Rules

1) Insert after GR 3)q) For every SQL parameter that has an associated string type, the routine descriptor includes the character string type descriptor of the associated string type.

2) Insert after GR 3)q) For every SQL parameter that has an associated XML option, the routine descriptor includes an indication of the associated XML option.

3) Insert after GR 3)q) For every SQL parameter whose <SQL parameter declaration> contains an explicit or implicit <XML passing mechanism>, an indication of that <XML passing mechanism>.

4) Insert after GR 3)r) If *R* is an SQL function, then an indication of the explicit or implicit <XML passing mechanism> contained in the <returns clause>.

Conformance Rules

1) Insert this CR Without Feature X120, "XML parameters in SQL routines", conforming SQL language shall not contain an <SQL-invoked routine> that simply contains a <language clause> that contains SQL and that simply contains a <parameter type> or a <returns data type> that contains a <data type> that is based on either an XML type or a distinct type whose source type is an XML type.

2) Insert this CR Without Feature X121, "XML parameters in external routines", conforming SQL language shall not contain an <SQL-invoked routine> that simply contains a <language clause> that contains ADA, C, COBOL, FORTRAN, MUMPS, PASCAL, or PLI and that simply contains a <parameter type> or a <returns data type> that contains a <data type> that is based on either an XML type or a distinct type whose source type is an XML type.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3) Insert this CR Without Feature X110, “Host language support for XML: VARCHAR mapping”, conforming SQL language shall not contain a <string type option> that contains CHARACTER VARYING, CHAR VARYING, or VARCHAR.

4) Insert this CR Without Feature X111, “Host language support for XML: CLOB mapping”, conforming SQL language shall not contain a <string type option> that contains CHARACTER LARGE OBJECT, CHAR LARGE OBJECT, or CLOB.

5) Insert this CR Without Feature X100, “Host language support for XML: CONTENT option”, conforming SQL language shall not contain a <document or content> that is CONTENT.

6) Insert this CR Without Feature X101, “Host language support for XML: DOCUMENT option”, conforming SQL language shall not contain a <document or content> that is DOCUMENT.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

12.8 <user-defined cast definition>

This Subclause modifies Subclause 11.54, "<user-defined cast definition>", in ISO/IEC 9075-2.

Function

Define a user-defined cast.

Format

```
No additional Format items.
```

Syntax Rules

1) Insert this SR Neither *SDT* nor *TDT* shall be a distinct type whose source type is an XML type.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

(Blank page)

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

13 SQL-client modules

This Clause modifies Clause 13, "SQL-client modules", in ISO/IEC 9075-2.

13.1 Calls to an <externally-invoked procedure>

This Subclause modifies Subclause 13.4, "Calls to an <externally-invoked procedure>", in ISO/IEC 9075-2.

Function

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

Syntax Rules

1) Insert into SR 2)e)

```
DATA_EXCEPTION_NONIDENTICAL_NOTATIONS_WITH_THE_SAME_NAME:
  constant SQLSTATE_TYPE := "2200J";
DATA_EXCEPTION_NONIDENTICAL_UNPARSED_ENTITIES_WITH_THE_SAME_NAME:
  constant SQLSTATE_TYPE := "2200K";
DATA_EXCEPTION_NOT_AN_XML_DOCUMENT:
  constant SQLSTATE_TYPE := "2200L";
DATA_EXCEPTION_INVALID_XML_DOCUMENT:
  constant SQLSTATE_TYPE := "2200M";
DATA_EXCEPTION_INVALID_XML_CONTENT:
  constant SQLSTATE_TYPE := "2200N";
DATA_EXCEPTION_XML_VALUE_OVERFLOW:
  constant SQLSTATE_TYPE := "2200R";
DATA_EXCEPTION_INVALID_COMMENT:
  constant SQLSTATE_TYPE := "2200S";
DATA_EXCEPTION_INVALID_PROCESSING_INSTRUCTION:
  constant SQLSTATE_TYPE := "2200T";
DATA_EXCEPTION_NOT_AN_XQUERY_DOCUMENT_NODE:
  constant SQLSTATE_TYPE := "2200U";
DATA_EXCEPTION_INVALID_XQUERY_CONTEXT_ITEM:
  constant SQLSTATE_TYPE := "2200V";
DATA_EXCEPTION_XQUERY_SERIALIZATION_ERROR:
  constant SQLSTATE_TYPE := "2200W";
DATA_EXCEPTION_XQUERY_SEQUENCE_CANNOT_BE_VALIDATED:
  constant SQLSTATE_TYPE := "2201J";
DATA_EXCEPTION_XQUERY_DOCUMENT_NODE_CANNOT_BE_VALIDATED:
  constant SQLSTATE_TYPE := "2201K";
DATA_EXCEPTION_NO_XML_SCHEMA_FOUND:
  constant SQLSTATE_TYPE := "2201L";
DATA_EXCEPTION_ELEMENT_NAMESPACE_NOT_DECLARED:
  constant SQLSTATE_TYPE := "2201M";
```

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
DATA_EXCEPTION_GLOBAL_ELEMENT_NOT_DECLARED:
  constant SQLSTATE_TYPE := "2201N";
DATA_EXCEPTION_NO_XML_ELEMENT_WITH_SPECIFIED_QNAME:
  constant SQLSTATE_TYPE := "2201P";
DATA_EXCEPTION_NO_XML_ELEMENT_WITH_SPECIFIED_NAMESPACE:
  constant SQLSTATE_TYPE := "2201Q";
DATA_EXCEPTION_VALIDATION_FAILURE:
  constant SQLSTATE_TYPE := "2201R";
SQLXML_MAPPING_ERROR_NO_SUBCLASS:
  constant SQLSTATE_TYPE := "0N000";
SQLXML_MAPPING_ERROR_INVALID_XML_CHARACTER:
  constant SQLSTATE_TYPE := "0N002";
SQLXML_MAPPING_ERROR_UNMAPPABLE_XML_NAME:
  constant SQLSTATE_TYPE := "0N001";
XQUERY_ERROR_NO_SUBCLASS:
  constant SQLSTATE_TYPE := "10000";
WARNING_COLUMN_CANNOT_BE_MAPPED:
  constant SQLSTATE_TYPE := "01010";
```

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

13.2 <SQL procedure statement>

This Subclause modifies Subclause 13.5, "<SQL procedure statement>", in ISO/IEC 9075-2.

Function

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

Format

```
<SQL session statement> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <set XML option 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.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

13.3 Data type correspondences

This Subclause modifies Subclause 13.6, "Data type correspondences", in ISO/IEC 9075-2.

Function

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

Tables

Augment Table 16, "Data type correspondences for Ada"

Table 5 — Data type correspondences for Ada

SQL Data Type	Ada Data Type
All alternatives from ISO/IEC 9075-2	
XML	*None*

Augment Table 17, "Data type correspondences for C"

Table 6 — Data type correspondences for C

SQL Data Type	C Data Type
All alternatives from ISO/IEC 9075-2	
XML	*None*

Augment Table 18, "Data type correspondences for COBOL"

Table 7 — Data type correspondences for COBOL

SQL Data Type	COBOL Data Type
All alternatives from ISO/IEC 9075-2	
XML	*None*

Augment Table 19, "Data type correspondences for Fortran"

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Table 8 — Data type correspondences for Fortran

SQL Data Type	Fortran Data Type
All alternatives from ISO/IEC 9075-2	
XML	*None*

Augment Table 20, "Data type correspondences for M"

Table 9 — Data type correspondences for M

SQL Data Type	M Data Type
All alternatives from ISO/IEC 9075-2	
XML	*None*

Augment Table 21, "Data type correspondences for Pascal"

Table 10 — Data type correspondences for Pascal

SQL Data Type	Pascal Data Type
All alternatives from ISO/IEC 9075-2	
XML	*None*

Augment Table 22, "Data type correspondences for PL/I"

Table 11 — Data type correspondences for PL/I

SQL Data Type	PL/I Data Type
All alternatives from ISO/IEC 9075-2	
XML	*None*

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

(Blank page)

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

14 Data manipulation

This Clause modifies Clause 14, "Data manipulation", in ISO/IEC 9075-2.

14.1 <fetch statement>

This Subclause modifies Subclause 14.5, "<fetch statement>", in ISO/IEC 9075-2.

Function

Position a standing cursor on a specified row of the standing cursor's result set and retrieve values from that row.

Format

```
<fetch target list> ::=
  <target specification 1> [ { <comma> <target specification 1> }... ]

<target specification 1> ::=
  <target specification> [ <XML passing mechanism> ]
```

Syntax Rules

1) Insert this SR If a <target specification 1> *TS1* contains an <XML passing mechanism>, then:

   a) The declared type of the <target specification> *TS2* contained in *TS1* shall be an XML type.

   b) *TS2* shall not be a <host parameter specification>, an <embedded variable specification>, or a <dynamic parameter specification>.

2) Insert this SR If the declared type of the <target specification> *TS2* contained in a <target specification 1> *TS1* is an XML type, *TS2* is not a <host parameter specification>, an <embedded variable specification>, or a <dynamic parameter specification>, and *TS1* does not contain an <XML passing mechanism>, then it is implementation-defined whether an <XML passing mechanism> of BY REF or BY VALUE is implicit.

Access Rules

No additional Access Rules.

General Rules

1) Replace GR 5)a)i) If *TS* is an <SQL parameter reference>, then

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Case:

a) If the declared type of *TS* is an XML type, then the General Rules of Subclause 10.2, "Store assignment", are applied to *TS* as *TARGET*. the current row as *VALUE*, and the <XML passing mechanism> of *TS* as *PASSING*.

b) Otherwise, the General Rules of Subclause 10.2, "Store assignment", are applied to *TS* as *TARGET* and the current row as *VALUE*.

2) Replace GR 5)b)i)1)B)V)1)b) The *I*-th element of *A* is set to

Case:

a) If *EDT* is an XML type, the value of *SV*, denoted by SV_i, by applying the General Rules of Subclause 10.2, "Store assignment", to the *I*-th element of *A* as *TARGET*, SV_i as *VALUE*, and the <XML passing mechanism> of TV_i as *PASSING*.

b) Otherwise, the value of *SV*, denoted by SV_i, by applying the General Rules of Subclause 10.2, "Store assignment", to the *I*-th element of *A* as *TARGET* and SV_i as *VALUE*.

3) Replace GR 5)b)i)1)B)V)2)c) The *I*-th element of *A* is set to

Case:

a) If *EDT* is an XML type, the value of *SV*, denoted by SV_i, by applying the General Rules of Subclause 10.2, "Store assignment", to the *I*-th element of *A* as *TARGET*, SV_i as *VALUE*, and the <XML passing mechanism> of TV_i as *PASSING*.

b) Otherwise, the value of *SV*, denoted by SV_i, by applying the General Rules of Subclause 10.2, "Store assignment", to the *I*-th element of *A* as *TARGET* and SV_i as *VALUE*.

4) Replace GR 5)b)i)2) Otherwise,

Case:

a) If the declared type of TV_i is an XML type, then the General Rules of Subclause 10.2, "Store assignment", are applied to TV_i as *TARGET*, SV_i as *VALUE*, and the <XML passing mechanism> of TV_i as *PASSING*.

b) Otherwise, the General Rules of Subclause 10.2, "Store assignment", are applied to TV_i as *TARGET* and SV_i as *VALUE*.

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

14.2 <select statement: single row>

This Subclause modifies Subclause 14.7, "<select statement: single row>", in ISO/IEC 9075-2.

Function

Retrieve values from a specified row of a table.

Format

```
<select target list> ::=
  <target specification 1> [ { <comma> <target specification 1> }... ]
```

Syntax Rules

1) Insert this SR If a <target specification 1> *TS1* contains an <XML passing mechanism>, then:

   a) The declared type of the <target specification> *TS2* contained in *TS1* shall be an XML type.

   b) *TS2* shall not be a <host parameter specification>, an <embedded variable specification>, or a <dynamic parameter specification>.

2) Insert this SR If the declared type of the <target specification> *TS2* contained in a <target specification 1> *TS1* is an XML type, *TS2* is not a <host parameter specification>, an <embedded variable specification>, or a <dynamic parameter specification>, and *TS1* does not contain an <XML passing mechanism>, then it is implementation-defined whether an <XML passing mechanism> of BY REF or BY VALUE is implicit.

Access Rules

No additional Access Rules.

General Rules

1) Replace GR 4)a)i) If *TS* is an <SQL parameter reference>, then

   Case:

   a) If the declared type of *TS* is an XML type, then the General Rules of Subclause 10.2, "Store assignment", are applied to *TS* as *TARGET*, the current row as *VALUE*, and the <XML passing mechanism> of *TS* as *PASSING*.

   b) Otherwise, the General Rules of Subclause 10.2, "Store assignment", are applied to *TS* as *TARGET* and the current row as *VALUE*.

2) Replace GR 4)b)ii)1)B)V)1)b) The *I*-th element of *A* is set to

   Case:

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

a) If *EDT* is an XML type, the value of *SL*, denoted by SL_i, by applying the General Rules of Subclause 10.2, "Store assignment", to the *I*-th element of *A* as *TARGET*, SL_i as *VALUE*, and the <XML passing mechanism> of TS_i as *PASSING*.

b) Otherwise, the value of *SL*, denoted by SL_i, by applying the General Rules of Subclause 10.2, "Store assignment", to the *I*-th element of *A* as *TARGET* and SL_i as *VALUE*.

3) Replace GR 4)b)ii)1)B)V)2)c) The *I*-th element of *A* is set to

Case:

a) If *EDT* is an XML type, the value of *SL*, denoted by SL_i, by applying the General Rules of Subclause 10.2, "Store assignment", to the *I*-th element of *A* as *TARGET*, SL_i as *VALUE*, and the <XML passing mechanism> of TS_i as *PASSING*.

b) Otherwise, the value of *SL*, denoted by SL_i, by applying the General Rules of Subclause 10.2, "Store assignment", to the *I*-th element of *A* as *TARGET* and SL_i as *VALUE*.

4) Replace GR 4)b)ii)2) Otherwise,

Case:

a) If the declared type of TS_i is an XML type, then the General Rules of Subclause 10.2, "Store assignment", are applied to TS_i as *TARGET*, SL_i as *VALUE*, and the <XML passing mechanism> of TS_i as *PASSING*.

b) Otherwise, the General Rules of Subclause 10.2, "Store assignment", are applied to TS_i as *TARGET* and SL_i as *VALUE*.

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

14.3 <delete statement: searched>

This Subclause modifies Subclause 14.9, "<delete statement: searched>", in ISO/IEC 9075-2.

Function

Delete rows of a table.

Format

```
<delete statement: searched> ::=
  DELETE [ WITH <XML lexically scoped options> ] FROM <target table>
      [ WHERE <search condition> ]
```

Syntax Rules

1) Insert this SR The scope of each <XML namespace declaration item> contained in the <XML lexically scoped options> is the <delete statement: searched>.

2) Insert this SR The scope of an <XML binary encoding> contained in the <XML lexically scoped options> is the <delete statement: searched>.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

1) Insert this CR Without Feature X082, "XML namespace declarations in DML", in conforming SQL language, a <delete statement: searched> shall not immediately contain an <XML lexically scoped options> that contains an <XML namespace declaration>.

2) Insert this CR Without Feature X132, "XMLBINARY clause in DML", in conforming SQL language, a <delete statement: searched> shall not immediately contain an <XML lexically scoped options> that contains an <XML binary encoding>.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

14.4 <insert statement>

This Subclause modifies Subclause 14.11, "<insert statement>", in ISO/IEC 9075-2.

Function

Create new rows in a table.

Format

```
<insert statement> ::=
  INSERT [ WITH <XML lexically scoped options> ] INTO <insertion target>
      <insert columns and source>
```

Syntax Rules

1) Insert this SR The scope of each <XML namespace declaration item> contained in <XML lexically scoped options> is the <insert statement>.

2) Insert this SR The scope of an <XML binary encoding> contained in <XML lexically scoped options> is the <insert statement>.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

1) Insert this CR Without Feature X082, "XML namespace declarations in DML", in conforming SQL language, an <insert statement> shall not immediately contain an <XML lexically scoped options> that contains an <XML namespace declaration>.

2) Insert this CR Without Feature X132, "XMLBINARY clause in DML", in conforming SQL language, an <insert statement> shall not immediately contain an <XML lexically scoped options> that contains an <XML binary encoding>.

© ISO/IEC 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

14.5 <merge statement>

This Subclause modifies Subclause 14.12, "<merge statement>", in ISO/IEC 9075-2.

Function

Conditionally update rows of a table, or insert new rows into a table, or both.

Format

```
<merge statement> ::=
  MERGE [ WITH <XML lexically scoped options> ] INTO <target table>
      [ [ AS ] <correlation name> ]
      USING <table reference> ON <search condition>
      <merge operation specification>
```

Syntax Rules

1) Insert this SR The scope of each <XML namespace declaration item> contained in <XML lexically scoped options> is the <merge statement>.

2) Insert this SR The scope of an <XML binary encoding> contained in <XML lexically scoped options> is the <merge statement>.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

1) Insert this CR Without Feature X082, "XML namespace declarations in DML", in conforming SQL language, a <merge statement> shall not immediately contain an <XML lexically scoped options> that contains an <XML namespace declaration>.

2) Insert this CR Without Feature X132, "XMLBINARY clause in DML", in conforming SQL language, a <merge statement> shall not immediately contain an <XML lexically scoped options> that contains an <XML binary encoding>.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

14.6 <update statement: positioned>

This Subclause modifies Subclause 14.13, "<update statement: positioned>", in ISO/IEC 9075-2.

Function

Update a row of a table.

Format

```
<update statement: positioned> ::=
  UPDATE [ WITH <XML lexically scoped options> ] <target table>
      SET <set clause list> WHERE CURRENT OF <cursor name>
```

Syntax Rules

1) Insert this SR The scope of each <XML namespace declaration item> contained in <XML lexically scoped options> is the <update statement: positioned>.

2) Insert this SR The scope of an <XML binary encoding> contained in <XML lexically scoped options> is the <update statement: positioned>.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

1) Insert this CR Without Feature X082, "XML namespace declarations in DML", in conforming SQL language, an <update statement: positioned> shall not immediately contain an <XML lexically scoped options> that contains an <XML namespace declaration>.

2) Insert this CR Without Feature X132, "XMLBINARY clause in DML", in conforming SQL language, an <update statement: positioned> shall not immediately contain an <XML lexically scoped options> that contains an <XML binary encoding>.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

14.7 <update statement: searched>

This Subclause modifies Subclause 14.14, "<update statement: searched>", in ISO/IEC 9075-2.

Function

Update rows of a table.

Format

```
<update statement: searched> ::=
  UPDATE [ WITH <XML lexically scoped options> ] <target table>
      SET <set clause list> [ WHERE <search condition> ]
```

Syntax Rules

1) Insert this SR The scope of each <XML namespace declaration item> contained in <XML lexically scoped options> is the <update statement: searched>.

2) Insert this SR The scope of an <XML binary encoding> contained in <XML lexically scoped options> is the <update statement: searched>.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

1) Insert this CR Without Feature X082, "XML namespace declarations in DML", in conforming SQL language, an <update statement: searched> shall not immediately contain an <XML lexically scoped options> that contains an <XML namespace declaration>.

2) Insert this CR Without Feature X132, "XMLBINARY clause in DML", in conforming SQL language, an <update statement: searched> shall not immediately contain an <XML lexically scoped options> that contains an <XML binary encoding>.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

(Blank page)

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

15 Control statements

This Clause modifies Clause 16, "Control statements", in ISO/IEC 9075-2.

15.1 <compound statement>

This Subclause modifies Subclause 14.1, "<compound statement>", in ISO/IEC 9075-4.

Function

Specify a statement that groups other statements together.

Format

```
<compound statement> ::=
  [ <beginning label> <colon> ]
      BEGIN [ [ NOT ] ATOMIC ]
      [ DECLARE <XML lexically scoped options> <semicolon> ]
      [ <local declaration list> ]
      [ <local cursor declaration list> ]
      [ <local handler declaration list> ]
      [ <SQL statement list> ]
      END [ <ending label> ]
```

Syntax Rules

1) Insert this SR The scope of each <XML namespace declaration item> contained in <XML lexically scoped options> is the <compound statement>.

2) Insert this SR The scope of an <XML binary encoding> contained in <XML lexically scoped options> is the <compound statement>.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Conformance Rules

1) Insert this CR Without Feature X084, “XML namespace declarations in compound statements”, in conforming SQL language, a <compound statement> shall not immediately contain an <XML lexically scoped options> that contains an <XML namespace declaration>.

2) Insert this CR Without Feature X134, “XMLBINARY clause in compound statements”, in conforming SQL language, a <compound statement> shall not immediately contain an <XML lexically scoped options> that contains an <XML binary encoding>.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

15.2 <assignment statement>

This Subclause modifies Subclause 14.5, "<assignment statement>", in ISO/IEC 9075-4.

Function

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

Format

```
<singleton variable assignment> ::=
  SET <assignment target> <equals operator> <assignment source>
      [ <XML passing mechanism> ]
```

Syntax Rules

1) Insert this SR If a <single variable assignment> *SVA* contains an <XML passing mechanism>, then:

   a) The declared type of the <assignment target> *AT* contained in *SVA* shall be an XML type.

   b) *SVA* shall not be a <host parameter specification>, an <embedded variable specification>, or a <dynamic parameter specification>.

2) Insert this SR If the declared type of the <assignment target> *AT* contained in a <singleton variable assignment> *SVA* is an XML type, *AT* is not a <host parameter specification>, an <embedded variable specification>, or a <dynamic parameter specification>, and *SVA* does not contain an <XML passing mechanism>, then it is implementation-defined whether an <XML passing mechanism> of BY REF or BY VALUE is implicit.

Access Rules

No additional Access Rules.

General Rules

1) Replace GR 1) If <assignment target> is a <target specification> that is a <column reference> *T*, an <SQL variable reference> to an SQL variable *T*, or an <SQL parameter reference> to an SQL parameter *T* of an SQL-invoked routine, then

   Case:

   a) If the declared type of *T* is an XML type, then the General Rules of Subclause 10.2, "Store assignment", are applied with *T* as *TARGET*, the value of <assignment source> as *VALUE*, and the <XML passing mechanism> as *PASSING*.

   b) Otherwise, the General Rules of Subclause 10.2, "Store assignment", in [ISO9075-2], are applied with *T* as *TARGET* and the value of <assignment source> as *VALUE*.

2) Replace GR 4)b) Otherwise,

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Case:

a) If the declared type of *FT* is an XML type, then the General Rules of Subclause 10.2, "Store assignment", are applied to *FT* as *TARGET*, the value of <assignment source> as *VALUE*, and the <XML passing mechanism> as *PASSING*.

b) Otherwise, the General Rules of Subclause 10.2, "Store assignment", are applied to *FT* as *TARGET* and the value of <assignment source> as *VALUE*.

3) Replace GR 5)b)v)1)B) The *I*-th element of *A* is set to

Case:

a) If *EDT* is an XML type, then the value of the <assignment source>, denoted by *SV*, by applying the General Rules of Subclause 10.2, "Store assignment", to the *I*-th element of *A* as *TARGET*, *SV* as *VALUE*, and the <XML passing mechanism> as *PASSING*.

b) Otherwise, the value of the <assignment source>, denoted by *SV*, by applying the General Rules of Subclause 10.2, "Store assignment", to the *I*-th element of *A* as *TARGET* and *SV* as *VALUE*.

4) Replace GR 5)b)v)2)C) The *I*-th element of *A* is set to

Case:

a) If *EDT* is an XML type, then the value of the <assignment source>, denoted by *SV*, by applying the General Rules of Subclause 10.2, "Store assignment", to the *I*-th element of *A* as *TARGET*, *SV* as *VALUE*, and the <XML passing mechanism> as *PASSING*.

b) Otherwise, the value of the <assignment source>, denoted by *SV*, by applying the General Rules of Subclause 10.2, "Store assignment", to the *I*-th element of *A* as *TARGET* and *SV* as *VALUE*.

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

16 Session management

This Clause modifies Clause 19, "Session management", in ISO/IEC 9075-2.

16.1 <set XML option statement>

Function

Set the XML option of the current SQL-session.

Format

```
<set XML option statement> ::=
  SET XML OPTION <document or content>
```

Syntax Rules

None.

Access Rules

None.

General Rules

1) Case:
   a) If DOCUMENT is specified, then the XML option of the current SQL-session is set to DOCUMENT.
   b) Otherwise, the XML option of the current SQL-session is set to CONTENT.

Conformance Rules

1) Without Feature F761, "Session management", conforming SQL language shall not contain a <set XML option statement>.
2) Without Feature X100, "Host language support for XML: CONTENT option", conforming SQL language shall not contain a <set XML option statement> that contains CONTENT.
3) Without Feature X101, "Host language support for XML: DOCUMENT option", conforming SQL language shall not contain a <set XML option statement> that contains DOCUMENT.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

(Blank page)

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

17 Dynamic SQL

This Clause modifies Clause 20, "Dynamic SQL", in ISO/IEC 9075-2.

17.1 Description of SQL descriptor areas

This Subclause modifies Subclause 20.1, "Description of SQL descriptor areas", in ISO/IEC 9075-2.

Function

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

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

1) Replace GR 1) Table 12, "Codes used for SQL data types in Dynamic SQL", specifies the codes associated with the SQL data types.

   Augment Table 25, "Codes used for SQL data types in Dynamic SQL"

Table 12 — Codes used for SQL data types in Dynamic SQL

Data Type	Code
All alternatives from ISO/IEC 9075-2	*All alternatives from ISO/IEC 9075-2*
XML	137

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

17.2 <input using clause>

This Subclause modifies Subclause 20.10, "<input using clause>", in ISO/IEC 9075-2.

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 before GR 6)c)ii) If *SDT* is a character string type and *TDT* is an XML type, then:

   a) Let *XO* be the XML option of the current SQL-session.

   b) It is implementation-defined whether *XWO* is STRIP WHITESPACE or PRESERVE WHITESPACE.

   c) If the <XML parse>

      ```
      XMLPARSE (XO SV XWO)
      ```

      does not conform to the Syntax Rules of Subclause 6.16, "<XML parse>", then an exception condition is raised: *dynamic SQL error — restricted data type attribute violation*.

   d) The <XML parse>

      ```
      XMLPARSE (XO SV XWO)
      ```

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

      NOTE 60 — The effective performance of the <XML parse> includes the raising of exceptions as specified in the General Rules of that Subclause.

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

17.3 <output using clause>

This Subclause modifies Subclause 20.11, "<output using clause>", in ISO/IEC 9075-2.

Function

Supply output 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 GR 5)d)ii) If *SDT* is the XML type and *TDT* is a character string type, then:

   a) Let *XO* be the XML option of the current SQL-session.

   b) It is implementation-defined whether *XDO* is INCLUDING XMLDECLARATION or EXCLUDING XMLDECLARATION.

   c) Case:

      i) If the character set of *TDT* is UTF16, then let *BOM* be `U&'\FEFF'`.

      ii) Otherwise, let *BOM* be the zero-length string of type *TDT*.

   d) If the <string value function>

      ```
      XMLSERIALIZE (XO SV AS TDT XDO)
      ```

      does not conform to the Syntax Rules of Subclause 6.8, "<string value function>", then an exception condition is raised: *dynamic SQL error — restricted data type attribute violation*.

   e) The <string value function>

      ```
      BOM || XMLSERIALIZE (XO SV AS TDT XDO)
      ```

      is effectively performed and is the value *TV* of the *i*-th <target specification>.

      NOTE 61 — The effective performance of the <XML character string serialization> includes the raising of exceptions as specified in the General Rules of that Subclause.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

17.4 <prepare statement>

This Subclause modifies Subclause 20.6, "<prepare statement>", in ISO/IEC 9075-2.

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 after GR 5)a)xxxvi) If *DP* is an <XML value expression> simply contained in <XML character string serialization>, <XML binary string serialization>, <XML concatenation>, <XML document>, or <XML validate>, then *DT* is an implementation-defined XML type.

2) Insert after GR 5)a)xxxvi) If *DP* is an <row value predicand> simply contained in <XML content predicate>, <XML document predicate>, or <XML valid predicate>, then *DT* is an implementation-defined XML type.

3) Insert after GR 5)a)xxxvi) If *DP* is a <character value expression> simply contained in <XML comment>, <XML PI>, or <XML text>, then *DT* is CHARACTER VARYING (*ML*).

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

(Blank page)

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

18 Embedded SQL

This Clause modifies Clause 21, "Embedded SQL", in ISO/IEC 9075-2.

18.1 <embedded SQL host program>

This Subclause modifies Subclause 21.1, "<embedded SQL host program>", in ISO/IEC 9075-2.

Function

Specify an <embedded SQL host program>.

Format

No additional Format items.

Syntax Rules

1) Insert after SR 21)h)i)5) If *EVN* identifies an XML VARCHAR host variable, then:
   a) Let *L* be the length of *EVN*.
   b) Let *CS* be the character set of *EVN*.
   c) *PT* is

      ```
      VARCHAR(L) CHARACTER SET CS
      ```

2) Insert after SR 21)h)i)5) If *EVN* identifies an XML CLOB host variable, then:
   a) Let *L* be the length of *EVN*.
   b) Let *CS* be the character set of *EVN*.
   c) *PT* is

      ```
      CLOB(L) CHARACTER SET CS
      ```

3) Insert after SR 21)h)i)5) If *EVN* identifies an XML BLOB host variable, then:
   a) Let *L* be the length of *EVN*.
   b) *PT* is

      ```
      BLOB(L)
      ```

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4) Insert before SR 21)h)ii)2) If *EVN* identifies an XML VARCHAR host variable, an XML CLOB host variable, or an XML BLOB host variable, then:

   a) Let *XO* be the XML option of *EVN*.

   b) Let *XWO* be the XML whitespace option of *EVN*.

   c) *EVN* is replaced by

      ```
      XMLPARSE (XO EVN XWO)
      ```

5) Insert after SR 21)l)i)3)B)V) If HV_i identifies an XML VARCHAR host variable, then:

   a) Let *L* be the length of HV_i.

   b) Let *CS* be the character set of HV_i.

   c) PT_i is

      ```
      VARCHAR(L) CHARACTER SET CS
      ```

6) Insert after SR 21)l)i)3)B)V) If HV_i identifies an XML CLOB host variable, then:

   a) Let *L* be the length of HV_i.

   b) Let *CS* be the character set of HV_i.

   c) PT_i is

      ```
      CLOB(L) CHARACTER SET CS
      ```

7) Insert after SR 21)l)i)3)B)V) If HV_i identifies an XML BLOB host variable, then:

   a) Let *L* be the length of HV_i.

   b) PT_i is

      ```
      BLOB(L)
      ```

8) Insert before SR 21)l)i)7) For each HVN_i , 1 (one) ≤ i ≤ n, that identifies some HV_i that is either an XML VARCHAR host variable, an XML CLOB host variable, or an XML BLOB host variable, apply the Syntax Rules of Subclause 9.6, "Host parameter mode determination", in [ISO9075-2], with the PD_i corresponding to HVN_i and *ES* as <host parameter declaration> and <SQL procedure statement>, respectively, to determine whether the corresponding XP_i is an input host parameter, an output host parameter, or both an input host parameter and an output host parameter.

   a) Among XP_i, 1 (one) ≤ i ≤ n, let d be the number of input host parameters, e be the number of output host parameters, and f be the number of host parameters that are both input host parameters and output host parameters.

   b) Among XP_i, 1 (one) ≤ i ≤ n, let XPI_j, 1 (one) ≤ j ≤ d, be the input host parameters, let XPO_k, 1 (one) ≤ k ≤ e, be the output host parameters, and let $XPIO_l$, 1 (one) ≤ l ≤ f, be the host parameters that are both input host parameters and output host parameters.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

c) For 1 (one) ≤ *j* ≤ *d*:

i) Let *XPNI*$_j$ be the <host parameter name> of *XPI*$_j$.

ii) Let *XHVI*$_j$ be the host variable corresponding to *XPI*$_j$.

iii) Let *XDOCI*$_j$ be the XML option of *XHVI*$_j$.

iv) Let *XSPWI*$_j$ be the XML whitespace option of *XHVI*$_j$.

v) It is implementation-defined whether *XDOI*$_j$ is INCLUDING XMLDECLARATION or EXCLUDING XMLDECLARATION.

d) For 1 (one) ≤ *k* ≤ *e*:

i) Let *XPNO*$_k$ be the <host parameter name> of *XPO*$_k$.

ii) Let *XHVO*$_k$ be the host variable corresponding to *XPO*$_k$.

iii) Let *XDOCO*$_k$ be the XML option of *XHVO*$_k$.

iv) Let *XSPWO*$_k$ be the XML whitespace option of *XHVO*$_k$.

v) It is implementation-defined whether *XDOO*$_k$ is INCLUDING XMLDECLARATION or EXCLUDING XMLDECLARATION.

vi) Let *XLO*$_k$ be the length of *XHVO*$_k$.

vii) If *XHVOk* is an XML VARCHAR host variable or XML CLOB host variable, then let *XCSO*$_k$ be the character set of *XHVO*$_k$.

viii) Case:

1) If *XHVO*$_k$ is an XML VARCHAR host variable, then let *XTO*$_k$ be

```
VARCHAR(XLOk) CHARACTER SET XCSOk
```

2) If *XHVO*$_k$ is an XML BLOB host variable, then let *XTO*$_k$ be

```
BLOB(XLOk)
```

3) Otherwise, let *XTO*$_k$ be

```
CLOB(XLOk) CHARACTER SET XCSOk
```

ix) Case:

1) If *XCSO*$_k$ is UTF16, then let *OBOM*$_k$ be `U&'\FEFF'`.

2) Otherwise, let *OBOM*$_k$ be the zero-length string of type *XTO*$_k$.

e) For 1 (one) ≤ *l* ≤ *f*:

i) Let *XPNIO*$_l$ be the <host parameter name> of *XPIO*$_l$.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ii) Let $XHVIO_l$ be the host variable corresponding to $XPIO_l$.

iii) Let $XDOCIO_l$ be the XML option of $XHVIO_l$.

iv) Let $XSPWIO_l$ be the XML whitespace option of $XHVIO_l$.

v) It is implementation-defined whether $XDOIO_l$ is INCLUDING XMLDECLARATION or EXCLUDING XMLDECLARATION.

vi) Let $XLIO_l$ be the length of $XHVIO_l$.

vii) If $XHVIO_l$ is an XML VARCHAR host variable or XML CLOB host variable, then let $XCSIO_l$ be the character set of $XHVIO_l$.

viii) Case:

1) If $XHVIO_l$ is an XML VARCHAR host variable, then let $XTIO_l$ be

```
VARCHAR(XLIOl) CHARACTER SET XCSIOl
```

2) If $XHVIO_l$ is an XML BLOB host variable, then let $XTIO_l$ be

```
BLOB(XLIOl)
```

3) Otherwise, let $XTIO_l$ be

```
CLOB(XLIOl) CHARACTER SET XCSIOl
```

ix) Case:

1) If $XCSIO_l$ is UTF16, then let $IOBOM_l$ be `U&'\FEFF'`.

2) Otherwise, let $IOBOM_l$ be the zero-length string of type $XTIO_l$.

f) Let XVI_j, 1 (one) $\le j \le d$, XVO_k, 1 (one) $\le k \le e$, and $XVIO_l$, 1 (one) $\le l \le f$, be implementation-dependent <SQL variable name>s, each of which is not equivalent to any other <SQL variable name> contained in *ES*, to any <SQL parameter name> contained in *ES*, or to any <column name> contained in *ES*.

9) Insert before SR 21)l)i)7)B) If HV_i identifies an XML VARCHAR host variable, an XML CLOB host variable, or an XML BLOB variable, then

Case:

a) If P_i is an input host parameter, then let $PXNI_j$, 1(one) $\le j \le d$, be the <host parameter name> of the input host parameter that corresponds to P_i. HVN_i is replaced by XVI_j.

b) If P_i is an output host parameter, then let $PXNO_k$, 1 (one) $\le k \le e$, be the <host parameter name> of the output host parameter that corresponds to P_i. HVN_i is replaced by XVO_k.

c) Otherwise, let $PXNIO_l$, 1 (one) $\le l \le f$, be the <host parameter name> of the input host parameter and the output host parameter that corresponds to P_i. HVN_i is replaced by $XVIO_l$.

10) Replace SR 21)l)i)8) The <SQL procedure statement> of *PS* is:

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
BEGIN ATOMIC
    DECLARE SVI1 TUI1;
    ...
    DECLARE SVIa TUIa;
    DECLARE XVI1 XML;
    ...
    DECLARE XVId XML;
    DECLARE SVO1 TUO1;
    ...
    DECLARE SVOb TUOb;
    DECLARE XVO1 XML;
    ...
    DECLARE XVOe XML;
    DECLARE SVIO1 TUIO1;
    ...
    DECLARE SVIOc TUIOc;
    DECLARE XVIO1 XML;
    ...
    DECLARE XVIOf XML;
    SET SVI1 = TSIN1 (CAST (PNI1 AS TTI1));
    ...
    SET SVIa = TSINa (CAST (PNIa AS TTIa));
    SET XVI1 = XMLPARSE (XDOCI1 XPNI1 XSPWI1);
    ...
    SET XVId = XMLPARSE (XDOCId XPNId XSPWId);
    SET SVIO1 = TSION1 (CAST (PNIO1 AS TTIO1));
    ...
    SET SVIOc = TSIONc (CAST (PNIOc AS TTIOc));
    SET XVIO1 = XMLPARSE (XDOCIO1 XPNIO1 XSPWIO1);
    ...
    SET XVIOf = XMLPARSE (XDOCIOf XPNIOf XSPWIOf);
    NES;
    SET PNO1 = CAST ( FSON1 (SVO1) AS TSO1);
    ...
    SET PNOb = CAST ( FSONb (SVOb) AS TSOb);
    SET XPNO1 = OBOM1 || XMLSERIALIZE ( XDOCO1 XVO1 AS XTO1 XDOO1 );
    ...
    SET XPNOe = OBOMe || XMLSERIALIZE ( XDOCOe XVOe AS XTOe XDOOe );
    SET PNIO1 = CAST ( FSION1 (SVIO1) AS TSIO1);
    ...
    SET PNIOc = CAST ( FSIONc (SVIOc) AS TSIOc);
    SET XPNIO1 = IOBOM1 || XMLSERIALIZE ( XDOCIO1 XVIO1 AS XTIO1 XDOIO1 );
    ...
    SET XPNIOf = IOBOMf || XMLSERIALIZE ( XDOCIOf XVIOf AS XTIOf XDOIOf );
END;
```

Access Rules

No additional Access Rules.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

18.2 <embedded SQL Ada program>

This Subclause modifies Subclause 21.3, "<embedded SQL Ada program>", in ISO/IEC 9075-2.

Function

Specify an <embedded SQL Ada program>.

Format

```
<Ada derived type specification> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <Ada XML CLOB variable>
  | <Ada XML BLOB variable>

<Ada XML BLOB variable> ::=
  SQL TYPE IS XML [ <document or content> ]
      [ <XML whitespace option> ]
      AS BLOB <left paren> <large object length> <right paren>

<Ada XML CLOB variable> ::=
  SQL TYPE IS XML [ <document or content> ]
      [ <XML whitespace option> ]
      AS CLOB <left paren> <character large object length> <right paren>
      [ CHARACTER SET [ IS ] <character set specification> ]
```

Syntax Rules

1) Insert after SR 5)f) The syntax

```
SQL TYPE IS XML XO XWO AS CLOB (L)
    CHARACTER SET IS CS
```

for a given <Ada host identifier> *HVN* shall be replaced by

```
TYPE HVN IS RECORD
  HVN_RESERVED : Interfaces.SQL.INT ;
  HVN_LENGTH : Interfaces.SQL.INT ;
  HVN_DATA : Interfaces.SQL.CHAR (1..LL);
END RECORD;
```

in any <Ada XML CLOB variable>, where:

a) *NV* is the numeric value of *L* as specified in Subclause 6.1, "<data type>".

b) The value of *LL* is *NV***k*.

c) *XO* is either <document or content> as specified in Subclause 6.8, "<string value function>", or the zero-length string.

d) XWO is either <XML whitespace option> as specified in Subclause 6.16, "<XML parse>", or the zero-length string.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

e) *CS* is a <character set name> as specified in Subclause 5.2, "Names and identifiers".

HVN is an *XML CLOB host variable*. *L* is the *length of XML CLOB host variable*. *CS* is the *character set of XML CLOB host variable*. If *XO* is the zero-length string, then it is implementation-defined whether DOCUMENT or CONTENT is the *XML option of XML CLOB host variable*; otherwise *XO* is the *XML option of XML CLOB host variable*. If *XWO* is the zero-length string, then it is implementation-defined whether STRIP WHITESPACE or PRESERVE WHITESPACE is the *XML whitespace option of the XML CLOB host variable*; otherwise, *XWO* is the *XML whitespace option of the XML CLOB host variable*.

NOTE 62 — The length of the XML CLOB host variable contains an implicit or explicit <char length units>.

2) Insert after SR 5)f) The syntax

```
SQL TYPE IS XML XO XWO AS BLOB (L)
```

for a given <Ada host identifier> *HVN* shall be replaced by

```
TYPE HVN IS RECORD
  HVN_RESERVED : Interfaces.SQL.INT ;
  HVN_LENGTH : Interfaces.SQL.INT ;
  HVN_DATA : Interfaces.SQL.CHAR (1..L);
END RECORD;
```

in any <Ada XML BLOB variable>, where:

a) *L* is the numeric value of <large object length> as specified in Subclause 5.1, "<token> and <separator>".

b) *XO* is either <document or content> as specified in Subclause 6.8, "<string value function>", or the zero-length string.

c) XWO is either <XML whitespace option> as specified in Subclause 6.16, "<XML parse>", or the zero-length string.

HVN is an *XML BLOB host variable*. *L* is the *length of XML BLOB host variable*. If *XO* is the zero-length string, then it is implementation-defined whether DOCUMENT or CONTENT is the *XML option of XML BLOB host variable*; otherwise *XO* is the *XML option of XML BLOB host variable*. If *XWO* is the zero-length string, then it is implementation-defined whether STRIP WHITESPACE or PRESERVE WHITESPACE is the *XML whitespace option of the XML BLOB host variable*; otherwise, *XWO* is the *XML whitespace option of the XML BLOB host variable*.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

1) Insert this CR Without Feature X111, "Host language support for XML: CLOB mapping", conforming SQL language shall not contain an <Ada XML CLOB variable>.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2) Insert this CR Without Feature X100, “Host language support for XML: CONTENT option”, in conforming SQL language, <Ada XML CLOB variable> shall not immediately contain a <document or content> that is CONTENT.

3) Insert this CR Without Feature X101, “Host language support for XML: DOCUMENT option”, in conforming SQL language, <Ada XML CLOB variable> shall not immediately contain a <document or content> that is DOCUMENT.

4) Insert this CR Without Feature X112, “Host language support for XML: BLOB mapping”, conforming SQL language shall not contain an <Ada XML BLOB variable>.

5) Insert this CR Without Feature X100, “Host language support for XML: CONTENT option”, in conforming SQL language, <Ada XML BLOB variable> shall not immediately contain a <document or content> that is CONTENT.

6) Insert this CR Without Feature X101, “Host language support for XML: DOCUMENT option”, in conforming SQL language, <Ada XML BLOB variable> shall not immediately contain a <document or content> that is DOCUMENT.

7) Insert this CR Without Feature X113, “Host language support for XML: STRIP WHITESPACE option”, in conforming SQL language, <Ada XML CLOB variable> shall not immediately contain an <XML whitespace option> that is STRIP WHITESPACE.

8) Insert this CR Without Feature X113, “Host language support for XML: STRIP WHITESPACE option”, in conforming SQL language, <Ada XML BLOB variable> shall not immediately contain an <XML whitespace option> that is STRIP WHITESPACE.

9) Insert this CR Without Feature X114, “Host language support for XML: PRESERVE WHITESPACE option”, in conforming SQL language, <Ada XML CLOB variable> shall not immediately contain an <XML whitespace option> that is PRESERVE WHITESPACE.

10) Insert this CR Without Feature X114, “Host language support for XML: PRESERVE WHITESPACE option”, in conforming SQL language, <Ada XML BLOB variable> shall not immediately contain an <XML whitespace option> that is PRESERVE WHITESPACE.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

18.3 <embedded SQL C program>

This Subclause modifies Subclause 21.4, "<embedded SQL C program>", in ISO/IEC 9075-2.

Function

Specify an <embedded SQL C program>.

Format

```
<C derived variable> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <C XML VARCHAR variable>
  | <C XML CLOB variable>
  | <C XML BLOB variable>

<C XML VARCHAR variable> ::=
  SQL TYPE IS XML [ <document or content> ]
      [ <XML whitespace option> ]
      AS VARCHAR
      [ CHARACTER SET [ IS ] <character set specification> ]
      <C host identifier> <C array specification> [ <C initial value> ]
      [ { <comma> <C host identifier> <C array specification> [ <C initial value> ] }...
  ]

<C XML CLOB variable> ::=
  SQL TYPE IS XML [ <document or content> ]
      [ <XML whitespace option> ]
      AS CLOB <left paren> <character large object length> <right paren>
      [ CHARACTER SET [ IS ] <character set specification> ]
      <C host identifier> [ <C initial value> ]
      [ { <comma> <C host identifier> [ <C initial value> ] }... ]

<C XML BLOB variable> ::=
  SQL TYPE IS XML [ <document or content> ]
      [ <XML whitespace option> ]
      AS BLOB <left paren> <large object length> <right paren>
      <C host identifier> [ <C initial value> ]
      [ { <comma> <C host identifier> [ <C initial value> ] }... ]
```

Syntax Rules

1) Replace SR 5)a) Any optional CHARACTER SET specification shall be removed from a <C VARCHAR variable>, a <C character variable>, a <C CLOB variable>, a <C NCHAR variable>, <C NCHAR VARYING variable>, a <C NCLOB variable>, a <C XML VARCHAR variable>, or a <C XML CLOB variable>.

2) Insert after SR 5)f) The syntax

   ```
   SQL TYPE IS XML XO XWO AS VARCHAR
     CHARACTER SET IS CS hvn[L]
   ```

   for a given <C host identifier> *hvn* shall be replaced by:

© ISO/IEC 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
char hvn [LL]
```

in any <C XML VARCHAR variable>, where:

a) *NV* is the numeric value of *L* and the value of *LL* is *NV***k*.

b) *XO* is either <document or content> as specified in Subclause 6.8, "<string value function>", or the zero-length string.

c) *XWO* is either <XML whitespace option> as specified in Subclause 6.16, "<XML parse>", or the zero-length string.

d) *CS* is a <character set name> as specified in Subclause 5.2, "Names and identifiers".

hvn is an *XML VARCHAR host variable*. *L* is the *length of XML VARCHAR host variable*. *CS* is the *character set of XML VARCHAR host variable*. If *XO* is the zero-length string, then it is implementation-defined whether DOCUMENT or CONTENT is the XML option of XML VARCHAR host variable; otherwise *XO* is the *XML option of XML VARCHAR host variable*. If *XWO* is the zero-length string, then it is implementation-defined whether STRIP WHITESPACE or PRESERVE WHITESPACE is the *XML whitespace option of the XML VARCHAR host variable*; otherwise *XWO* is the *XML whitespace option of the XML VARCHAR host variable*.

NOTE 63 — The length of the XML VARCHAR host variable contains an implicit or explicit <char length units>.

3) Insert after SR 5)f) The syntax

```
SQL TYPE IS XML XO XWO AS CLOB(L)
  CHARACTER SET IS CS hvn
```

for a given <C host identifier> *hvn* shall be replaced by:

```
struct {
  long          hvn_reserved;
  unsigned long hvn_length;
  char          hvn_data[LL];
} hvn
```

in any <C XML CLOB variable>, where:

a) *NV* is the numeric value of *L* as specified in Subclause 6.1, "<data type>", and the value of *LL* is *NV***k*.

b) *XO* is either <document or content> as specified in Subclause 6.8, "<string value function>", or the zero-length string.

c) *XWO* is either <XML whitespace option> as specified in Subclause 6.16, "<XML parse>", or the zero-length string.

d) *CS* is a <character set name> as specified in Subclause 5.2, "Names and identifiers".

hvn is an *XML CLOB host variable*. *L* is the *length of XML CLOB host variable*. *CS* is the *character set of XML CLOB host variable*. If *XO* is the zero-length string, then it is implementation-defined whether DOCUMENT or CONTENT is the XML option of XML CLOB host variable; otherwise *XO* is the *XML option of XML CLOB host variable*. If *XWO* is the zero-length string, then it is implementation-defined whether STRIP WHITESPACE or PRESERVE WHITESPACE is the *XML whitespace option of the XML CLOB host variable*; otherwise *XWO* is the *XML whitespace option of the XML CLOB host variable*.

NOTE 64 — The length of the XML CLOB host variable contains an implicit or explicit <char length units>.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

4) Insert after SR 5)f) The syntax

```
SQL TYPE IS XML XO XWO AS BLOB(L)
```

for a given <C host identifier> *hvn* shall be replaced by:

```
struct {
  long          hvn_reserved;
  unsigned long hvn_length;
  char          hvn_data[L];
} hvn
```

in any <C XML BLOB variable>, where:

a) *L* is the numeric value of <large object length> as specified in Subclause 5.1, "<token> and <separator>".

b) *XO* is either <document or content> as specified in Subclause 6.8, "<string value function>", or the zero-length string.

c) *XWO* is either <XML whitespace option> as specified in Subclause 6.16, "<XML parse>", or the zero-length string.

d) *CS* is a <character set name> as specified in Subclause 5.2, "Names and identifiers".

hvn is an *XML BLOB host variable*. *L* is the *length of XML BLOB host variable*. *CS* is the *character set of XML BLOB host variable*. If *XO* is the zero-length string, then it is implementation-defined whether DOCUMENT or CONTENT is the XML option of XML BLOB host variable; otherwise *XO* is the *XML option of XML BLOB host variable*. If *XWO* is the zero-length string, then it is implementation-defined whether STRIP WHITESPACE or PRESERVE WHITESPACE is the *XML whitespace option of the XML BLOB host variable*; otherwise *XWO* is the *XML whitespace option of the XML BLOB host variable*.

5) Replace SR 9) In a <C variable definition>, the words "VARCHAR", "CHARACTER", "SET", "IS", "VARYING", "BLOB", "CLOB", "NCHAR", "NCLOB", "AS", "LOCATOR", "REF" and "XML" may be specified in any combination of upper-case and lower-case letters (see the Syntax Rules of Subclause 5.1, "<token> and <separator>").

6) Insert after SR 10) In a <C XML VARCHAR variable>, if a <character set specification> is specified, then the SQL data type of the character string resulting from the invocation of the XMLSERIALIZE operator on a value of XML type is VARCHAR whose character set is the same as the character set specified by the <character set specification>. If <character set specification> is not specified, then an implementation-defined <character set specification> is implicit.

7) Insert after SR 10) In a <C XML CLOB variable>, if a <character set specification> is specified, then the SQL data type of the character string resulting from the invocation of the XMLSERIALIZE operator on a value of XML type is CHARACTER LARGE OBJECT whose character set is the same as the character set specified by the <character set specification>. If <character set specification> is not specified, then an implementation-defined <character set specification> is implicit.

8) Insert after SR 10) Each <C host identifier> specified in a <C XML VARCHAR variable> describes a variable-length character string. The maximum length is specified by the <character length> of the <C array specification>, measured in the units specified by the explicit or implicit <char length units>. The value in the host variable is terminated by a null character and the position occupied by this null character is included in the maximum length of the host variable. The SQL data type of the character string resulting from the invocation of the XMLSERIALIZE operator on a value of XML type is CHARACTER VARYING whose maximum length is 1 (one) less than the <character length> of the <C array specification> and whose value does not include the terminating null character. The <length> shall be greater than 1 (one).

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

1) Insert this CR Without Feature X110, "Host language support for XML: VARCHAR mapping", conforming SQL language shall not contain an <C XML VARCHAR variable>.

2) Insert this CR Without Feature X111, "Host language support for XML: CLOB mapping", conforming SQL language shall not contain an <C XML CLOB variable>.

3) Insert this CR Without Feature X100, "Host language support for XML: CONTENT option", in conforming SQL language, neither <C XML VARCHAR variable> nor <C XML CLOB variable> shall immediately contain a <document or content> that is CONTENT.

4) Insert this CR Without Feature X101, "Host language support for XML: DOCUMENT option", in conforming SQL language, neither <C XML VARCHAR variable> nor <C XML CLOB variable> shall immediately contain a <document or content> that is DOCUMENT.

5) Insert this CR Without Feature X112, "Host language support for XML: BLOB mapping", conforming SQL language shall not contain a <C XML BLOB variable>.

6) Insert this CR Without Feature X100, "Host language support for XML: CONTENT option", in conforming SQL language, <C XML BLOB variable> shall not immediately contain a <document or content> that is CONTENT.

7) Insert this CR Without Feature X101, "Host language support for XML: DOCUMENT option", in conforming SQL language, <C XML BLOB variable> shall not immediately contain a <document or content> that is DOCUMENT.

8) Insert this CR Without Feature X113, "Host language support for XML: STRIP WHITESPACE option", in conforming SQL language, <C XML CLOB variable> shall not immediately contain an <XML whitespace option> that is STRIP WHITESPACE.

9) Insert this CR Without Feature X113, "Host language support for XML: STRIP WHITESPACE option", in conforming SQL language, <C XML BLOB variable> shall not immediately contain an <XML whitespace option> that is STRIP WHITESPACE.

10) Insert this CR Without Feature X113, "Host language support for XML: STRIP WHITESPACE option", in conforming SQL language, <C XML VARCHAR variable> shall not immediately contain an <XML whitespace option> that is STRIP WHITESPACE.

11) Insert this CR Without Feature X114, "Host language support for XML: PRESERVE WHITESPACE option", in conforming SQL language, <C XML CLOB variable> shall not immediately contain an <XML whitespace option> that is PRESERVE WHITESPACE.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

12) Insert this CR Without Feature X114, “Host language support for XML: PRESERVE WHITESPACE option”, in conforming SQL language, <C XML BLOB variable> shall not immediately contain an <XML whitespace option> that is PRESERVE WHITESPACE.

13) Insert this CR Without Feature X114, “Host language support for XML: PRESERVE WHITESPACE option”, in conforming SQL language, <C XML VARCHAR variable> shall not immediately contain an <XML whitespace option> that is PRESERVE WHITESPACE.

© ISO/IEC 2008 – All rights reserved

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

18.4 <embedded SQL COBOL program>

This Subclause modifies Subclause 21.5, "<embedded SQL COBOL program>", in ISO/IEC 9075-2.

Function

Specify an <embedded SQL COBOL program>.

Format

```
<COBOL derived type specification> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <COBOL XML CLOB variable>
  | <COBOL XML BLOB variable>

<COBOL XML BLOB variable> ::=
  [ USAGE [ IS ] ] SQL TYPE IS XML [ <document or content> ]
      [ <XML whitespace option> ]
      AS BLOB <left paren> <large object length> <right paren>

<COBOL XML CLOB variable> ::=
  [ USAGE [ IS ] ] SQL TYPE IS XML [ <document or content> ]
      [ <XML whitespace option> ]
      AS CLOB <left paren> <character large object length> <right paren>
      [ CHARACTER SET [ IS ] <character set specification> ]
```

Syntax Rules

1) Insert after SR 5)d) The syntax

```
SQL TYPE IS XML XO XWO AS CLOB ( L )
    CHARACTER SET IS CS
```

for a given <COBOL host identifier> *HVN* shall be replaced by:

```
49 HVN-RESERVED PIC S9(9) USAGE IS BINARY.
49 HVN-LENGTH PIC S9(9) USAGE IS BINARY.
49 HVN-DATA PIC X(LL).
```

in any <COBOL XML CLOB variable>, where:

a) *NV* is the numeric value of *L* as specified in Subclause 6.1, "<data type>" and the value of *LL* is *NV***k*.

b) *XO* is either <document or content> as specified in Subclause 6.8, "<string value function>", or the zero-length string.

c) *XWO* is either <XML whitespace option> as specified in Subclause 6.16, "<XML parse>", or the zero-length string.

d) *CS* is a <character set name> as specified in Subclause 5.2, "Names and identifiers".

HVN is an *XML CLOB host variable*. *L* is the *length of XML CLOB host variable*. *CS* is the *character set of XML CLOB host variable*. If *XO* is the zero-length string, then it is implementation-defined whether

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

DOCUMENT or CONTENT is the XML option of XML CLOB host variable; otherwise *XO* is the *XML option of XML CLOB host variable*. If *XWO* is the zero-length string, then it is implementation-defined whether STRIP WHITESPACE or PRESERVE WHITESPACE is the *XML whitespace option of the XML CLOB host variable*; otherwise *XWO* is the *XML whitespace option of the XML CLOB host variable*.

NOTE 65 — The length of the XML CLOB host variable contains an implicit or explicit <char length units>.

2) Insert after SR 5)d) The syntax

```
SQL TYPE IS XML XO XWO AS BLOB ( L )
```

for a given <COBOL host identifier> *HVN* shall be replaced by:

```
49 HVN-RESERVED PIC S9(9) USAGE IS BINARY.
49 HVN-LENGTH PIC S9(9) USAGE IS BINARY.
49 HVN-DATA PIC X(L).
```

in any <COBOL XML BLOB variable>, where:

a) *L* is the numeric value of <large object length> as specified in Subclause 5.1, "<token> and <separator>".

b) *XO* is either <document or content> as specified in Subclause 6.8, "<string value function>", or the zero-length string.

c) *XWO* is either <XML whitespace option> as specified in Subclause 6.16, "<XML parse>", or the zero-length string.

HVN is an *XML BLOB host variable*. *L* is the *length of XML BLOB host variable*. *CS* is the *character set of XML BLOB host variable*. If *XO* is the zero-length string, then it is implementation-defined whether DOCUMENT or CONTENT is the XML option of XML BLOB host variable; otherwise *XO* is the *XML option of XML BLOB host variable*. If *XWO* is the zero-length string, then it is implementation-defined whether STRIP WHITESPACE or PRESERVE WHITESPACE is the *XML whitespace option of the XML BLOB host variable*; otherwise *XWO* is the *XML whitespace option of the XML BLOB host variable*.

3) Insert after SR 8) A <COBOL XML CLOB variable> describes a character string resulting from the invocation of the XMLSERIALIZE operator on a value of XML type, whose equivalent SQL data type is CHARACTER LARGE OBJECT with the same length and character set specified by <character set specification>. If <character set specification> is not specified, then an implementation-defined <character set specification> is implicit.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

1) Insert this CR Without Feature X111, "Host language support for XML: CLOB mapping", conforming SQL language shall not contain an <COBOL XML CLOB variable>.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2) Insert this CR Without Feature X100, “Host language support for XML: CONTENT option”, in conforming SQL language, <COBOL XML CLOB variable> shall not immediately contain a <document or content> that is CONTENT.

3) Insert this CR Without Feature X101, “Host language support for XML: DOCUMENT option”, in conforming SQL language, <COBOL XML CLOB variable> shall not immediately contain a <document or content> that is DOCUMENT.

4) Insert this CR Without Feature X112, “Host language support for XML: BLOB mapping”, conforming SQL language shall not contain a <COBOL XML BLOB variable>.

5) Insert this CR Without Feature X100, “Host language support for XML: CONTENT option”, in conforming SQL language, <COBOL XML BLOB variable> shall not immediately contain a <document or content> that is CONTENT.

6) Insert this CR Without Feature X101, “Host language support for XML: DOCUMENT option”, in conforming SQL language, <COBOL XML BLOB variable> shall not immediately contain a <document or content> that is DOCUMENT.

7) Insert this CR Without Feature X113, “Host language support for XML: STRIP WHITESPACE option”, in conforming SQL language, <COBOL XML CLOB variable> shall not immediately contain an <XML whitespace option> that is STRIP WHITESPACE.

8) Insert this CR Without Feature X113, “Host language support for XML: STRIP WHITESPACE option”, in conforming SQL language, <COBOL XML BLOB variable> shall not immediately contain an <XML whitespace option> that is STRIP WHITESPACE.

9) Insert this CR Without Feature X114, “Host language support for XML: PRESERVE WHITESPACE option”, in conforming SQL language, <COBOL XML CLOB variable> shall not immediately contain an <XML whitespace option> that is PRESERVE WHITESPACE.

10) Insert this CR Without Feature X114, “Host language support for XML: PRESERVE WHITESPACE option”, in conforming SQL language, <COBOL XML BLOB variable> shall not immediately contain an <XML whitespace option> that is PRESERVE WHITESPACE.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

18.5 <embedded SQL Fortran program>

This Subclause modifies Subclause 21.6, "<embedded SQL Fortran program>", in ISO/IEC 9075-2.

Function

Specify an <embedded SQL Fortran program>.

Format

```
<Fortran derived type specification> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <Fortran XML CLOB variable>
  | <Fortran XML BLOB variable>

<Fortran XML BLOB variable> ::=
  SQL TYPE IS XML [ <document or content> ]
      [ <XML whitespace option> ]
      AS BLOB <left paren> <large object length> <right paren>

<Fortran XML CLOB variable> ::=
  SQL TYPE IS XML [ <document or content> ]
      [ <XML whitespace option> ]
      AS CLOB <left paren> <character large object length> <right paren>
      [ CHARACTER SET [ IS ] <character set specification> ]
```

Syntax Rules

1) Insert after SR 6)d) The syntax

```
SQL TYPE IS XML XO XWO AS CLOB ( L )
  CHARACTER SET IS CS
```

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

```
INTEGER   HVN_RESERVED
INTEGER   HVN_LENGTH
CHARACTER HVN_DATA [ * LL ]
```

in any <Fortran XML CLOB variable>, where:

a) *NV* is the numeric value of *L* as specified in Subclause 6.1, "<data type>" and the value of *LL* is *NV***k*.

b) *XO* is either <document or content> as specified in Subclause 6.8, "<string value function>", or the zero-length string.

c) *XWO* is either <XML whitespace option> as specified in Subclause 6.16, "<XML parse>", or the zero-length string.

d) *CS* is a <character set name> as specified in Subclause 5.2, "Names and identifiers".

HVN is an *XML CLOB host variable*. *L* is the *length of XML CLOB host variable*. *CS* is the *character set of XML CLOB host variable*. If *XO* is the zero-length string, then it is implementation-defined whether

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

DOCUMENT or CONTENT is the XML option of XML CLOB host variable; otherwise *XO* is the *XML option of XML CLOB host variable*. If *XWO* is the zero-length string, then it is implementation-defined whether STRIP WHITESPACE or PRESERVE WHITESPACE is the *XML whitespace option of the XML CLOB host variable*; otherwise *XWO* is the *XML whitespace option of the XML CLOB host variable*.

NOTE 66 — The length of the XML CLOB host variable contains an implicit or explicit <char length units>.

2) Insert after SR 6)d) The syntax

```
SQL TYPE IS XML XO XWO AS BLOB ( L )
```

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

```
INTEGER   HVN_RESERVED
INTEGER   HVN_LENGTH
CHARACTER HVN_DATA [ <asterisk> L ]
```

in any <Fortran XML BLOB variable>, where:

a) *L* is the numeric value of <large object length> as specified in Subclause 5.1, "<token> and <separator>".

b) *XO* is either <document or content> as specified in Subclause 6.8, "<string value function>", or the zero-length string.

c) *XWO* is either <XML whitespace option> as specified in Subclause 6.16, "<XML parse>", or the zero-length string.

HVN is an *XML BLOB host variable*. *L* is the *length of XML BLOB host variable*. *CS* is the *character set of XML BLOB host variable*. If *XO* is the zero-length string, then it is implementation-defined whether DOCUMENT or CONTENT is the XML option of XML BLOB host variable; otherwise *XO* is the *XML option of XML BLOB host variable*. If *XWO* is the zero-length string, then it is implementation-defined whether STRIP WHITESPACE or PRESERVE WHITESPACE is the *XML whitespace option of the XML BLOB host variable*; otherwise *XWO* is the *XML whitespace option of the XML BLOB host variable*.

3) Insert after SR 9) A <Fortran XML CLOB variable> describes a character string resulting from the invocation of the XMLSERIALIZE operator on a value of XML type, whose equivalent SQL data type is CHARACTER LARGE OBJECT with the same length and character set specified by <character set specification>. If <character set specification> is not specified, then an implementation-defined <character set specification> is implicit.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

1) Insert this CR Without Feature X111, "Host language support for XML: CLOB mapping", conforming SQL language shall not contain an <Fortran XML CLOB variable>.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

2) Insert this CR Without Feature X100, “Host language support for XML: CONTENT option”, in conforming SQL language, <Fortran XML CLOB variable> shall not immediately contain a <document or content> that is CONTENT.

3) Insert this CR Without Feature X101, “Host language support for XML: DOCUMENT option”, in conforming SQL language, <Fortran XML CLOB variable> shall not immediately contain a <document or content> that is DOCUMENT.

4) Insert this CR Without Feature X112, “Host language support for XML: BLOB mapping”, conforming SQL language shall not contain a <Fortran XML BLOB variable>.

5) Insert this CR Without Feature X100, “Host language support for XML: CONTENT option”, in conforming SQL language, <Fortran XML BLOB variable> shall not immediately contain a <document or content> that is CONTENT.

6) Insert this CR Without Feature X101, “Host language support for XML: DOCUMENT option”, in conforming SQL language, <Fortran XML BLOB variable> shall not immediately contain a <document or content> that is DOCUMENT.

7) Insert this CR Without Feature X113, “Host language support for XML: STRIP WHITESPACE option”, in conforming SQL language, <Fortran XML CLOB variable> shall not immediately contain an <XML whitespace option> that is STRIP WHITESPACE.

8) Insert this CR Without Feature X113, “Host language support for XML: STRIP WHITESPACE option”, in conforming SQL language, <Fortran XML BLOB variable> shall not immediately contain an <XML whitespace option> that is STRIP WHITESPACE.

9) Insert this CR Without Feature X114, “Host language support for XML: PRESERVE WHITESPACE option”, in conforming SQL language, <Fortran XML CLOB variable> shall not immediately contain an <XML whitespace option> that is PRESERVE WHITESPACE.

10) Insert this CR Without Feature X114, “Host language support for XML: PRESERVE WHITESPACE option”, in conforming SQL language, <Fortran XML BLOB variable> shall not immediately contain an <XML whitespace option> that is PRESERVE WHITESPACE.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

18.6 <embedded SQL MUMPS program>

This Subclause modifies Subclause 21.7, "<embedded SQL MUMPS program>", in ISO/IEC 9075-2.

Function

Specify an <embedded SQL MUMPS program>.

Format

```
<MUMPS derived type specification> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <MUMPS XML CLOB variable>
  | <MUMPS XML BLOB variable>

<MUMPS XML CLOB variable> ::=
  SQL TYPE IS XML [ <document or content> ]
      [ <XML whitespace option> ]
      AS CLOB <left paren> <large object length> <right paren>
      [ CHARACTER SET [ IS ] <character set specification> ]

<MUMPS XML BLOB variable> ::=
  SQL TYPE IS XML [ <document or content> ]
      [ <XML whitespace option> ]
      AS BLOB <left paren> <large object length> <right paren>
```

Syntax Rules

1) Insert after SR 9)b) The syntax

```
SQL TYPE IS XML XO XWO AS CLOB ( L )
  CHARACTER SET IS CS
```

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

```
INTEGER HVN_RESERVED
INTEGER HVN_LENGTH
VARCHAR HVN_DATA NV
```

in any <MUMPS XML CLOB variable>, where:

a) *NV* is the numeric value of *L* as specified in Subclause 6.1, "<data type>".

b) *XO* is either <document or content> as specified in Subclause 6.8, "<string value function>", or the zero-length string.

c) *XWO* is either <XML whitespace option> as specified in Subclause 6.16, "<XML parse>", or the zero-length string.

d) *CS* is a <character set name> as specified in Subclause 5.2, "Names and identifiers".

HVN is an *XML CLOB host variable*. *L* is the *length of XML CLOB host variable*. *CS* is the *character set of XML CLOB host variable*. If *XO* is the zero-length string, then it is implementation-defined whether

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

DOCUMENT or CONTENT is the XML option of XML CLOB host variable; otherwise *XO* is the *XML option of XML CLOB host variable*. If *XWO* is the zero-length string, then it is implementation-defined whether STRIP WHITESPACE or PRESERVE WHITESPACE is the *XML whitespace option of the XML CLOB host variable*; otherwise *XWO* is the *XML whitespace option of the XML CLOB host variable*.

NOTE 67 — The length of the XML CLOB host variable contains an implicit <char length units> that specifies CHARACTERS.

2) Insert after SR 9)b) The syntax

```
SQL TYPE IS XML XO XWO AS BLOB ( L )
```

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

```
INTEGER HVN_RESERVED
INTEGER HVN_LENGTH
VARCHAR HVN_DATA L
```

in any <MUMPS XML BLOB variable>, where:

a) *L* is the numeric value of <large object length> as specified in Subclause 5.1, "<token> and <separator>".

b) *XO* is either <document or content> as specified in Subclause 6.8, "<string value function>", or the zero-length string.

c) *XWO* is either <XML whitespace option> as specified in Subclause 6.16, "<XML parse>", or the zero-length string.

HVN is an *XML BLOB host variable*. *L* is the *length of XML BLOB host variable*. *CS* is the *character set of XML BLOB host variable*. If *XO* is the zero-length string, then it is implementation-defined whether DOCUMENT or CONTENT is the XML option of XML BLOB host variable; otherwise *XO* is the *XML option of XML BLOB host variable*. If *XWO* is the zero-length string, then it is implementation-defined whether STRIP WHITESPACE or PRESERVE WHITESPACE is the *XML whitespace option of the XML BLOB host variable*; otherwise *XWO* is the *XML whitespace option of the XML BLOB host variable*.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

1) Insert this CR Without Feature X111, "Host language support for XML: CLOB mapping", conforming SQL language shall not contain a <MUMPS XML CLOB variable>.

2) Insert this CR Without Feature X100, "Host language support for XML: CONTENT option", in conforming SQL language, <MUMPS XML CLOB variable> shall not immediately contain a <document or content> that is CONTENT.

© ISO/IEC 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

3) Insert this CR Without Feature X101, “Host language support for XML: DOCUMENT option”, in conforming SQL language, <MUMPS XML CLOB variable> shall not immediately contain a <document or content> that is DOCUMENT.

4) Insert this CR Without Feature X112, “Host language support for XML: BLOB mapping”, conforming SQL language shall not contain a <MUMPS XML BLOB variable>.

5) Insert this CR Without Feature X100, “Host language support for XML: CONTENT option”, in conforming SQL language, <MUMPS XML BLOB variable> shall not immediately contain a <document or content> that is CONTENT.

6) Insert this CR Without Feature X101, “Host language support for XML: DOCUMENT option”, in conforming SQL language, <MUMPS XML BLOB variable> shall not immediately contain a <document or content> that is DOCUMENT.

7) Insert this CR Without Feature X113, “Host language support for XML: STRIP WHITESPACE option”, in conforming SQL language, <MUMPS XML CLOB variable> shall not immediately contain an <XML whitespace option> that is STRIP WHITESPACE.

8) Insert this CR Without Feature X113, “Host language support for XML: STRIP WHITESPACE option”, in conforming SQL language, <MUMPS XML BLOB variable> shall not immediately contain an <XML whitespace option> that is STRIP WHITESPACE.

9) Insert this CR Without Feature X114, “Host language support for XML: PRESERVE WHITESPACE option”, in conforming SQL language, <MUMPS XML CLOB variable> shall not immediately contain an <XML whitespace option> that is PRESERVE WHITESPACE.

10) Insert this CR Without Feature X114, “Host language support for XML: PRESERVE WHITESPACE option”, in conforming SQL language, <MUMPS XML BLOB variable> shall not immediately contain an <XML whitespace option> that is PRESERVE WHITESPACE.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

18.7 <embedded SQL Pascal program>

This Subclause modifies Subclause 21.8, "<embedded SQL Pascal program>", in ISO/IEC 9075-2.

Function

Specify an <embedded SQL Pascal program>.

Format

```
<Pascal derived type specification> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <Pascal XML CLOB variable>
  | <Pascal XML BLOB variable>

<Pascal XML CLOB variable> ::=
  SQL TYPE IS XML [ <document or content> ]
      [ <XML whitespace option> ]
      AS CLOB <left paren> <character large object length> <right paren>
      [ CHARACTER SET [ IS ] <character set specification> ]

<Pascal XML BLOB variable> ::=
  SQL TYPE IS XML [ <document or content> ]
      [ <XML whitespace option> ]
      AS BLOB <left paren> <large object length> <right paren>
```

Syntax Rules

1) Replace SR 5)a) Any optional CHARACTER SET specification shall be removed from the PACKED ARRAY OF CHAR or CHAR alternatives of a <Pascal type specification>, a <Pascal CLOB variable>, and a <Pascal XML CLOB variable>.

2) Insert after SR 5)f) The syntax

   ```
   SQL TYPE IS XML XO XWO AS CLOB ( L )
     CHARACTER SET IS CS
   ```

   for a given <Pascal host identifier> *HVN* shall be replaced by

   ```
   VAR HVN = RECORD
     HVN_RESERVED : INTEGER ;
     HVN_LENGTH : INTEGER ;
     HVN_DATA : PACKED ARRAY [ 1..LL ] OF CHAR ;
   END ;
   ```

   in any <Pascal XML CLOB variable>, where:

   a) *NV* is the numeric value of *L* as specified in Subclause 6.1, "<data type>", and the value of *LL* is *NV***k*.

   b) *XO* is either <document or content> as specified in Subclause 6.8, "<string value function>", or the zero-length string.

© ISO/IEC 2008 – All rights reserved
--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

c) *XWO* is either <XML whitespace option> as specified in Subclause 6.16, "<XML parse>", or the zero-length string.

d) *CS* is a <character set name> as specified in Subclause 5.2, "Names and identifiers".

HVN is an *XML CLOB host variable*. *L* is the *length of XML CLOB host variable*. *CS* is the *character set of XML CLOB host variable*. If *XO* is the zero-length string, then it is implementation-defined whether DOCUMENT or CONTENT is the XML option of XML CLOB host variable; otherwise *XO* is the *XML option of XML CLOB host variable*. If *XWO* is the zero-length string, then it is implementation-defined whether STRIP WHITESPACE or PRESERVE WHITESPACE is the *XML whitespace option of the XML CLOB host variable*; otherwise *XWO* is the *XML whitespace option of the XML CLOB host variable*.

NOTE 68 — The length of the XML CLOB host variable contains an implicit or explicit <char length units>.

3) Insert after SR 5)f) The syntax

```
SQL TYPE IS XML XO XWO AS BLOB ( L )
```

for a given <Pascal host identifier> *HVN* shall be replaced by

```
VAR HVN = RECORD
  HVN_RESERVED : INTEGER ;
  HVN_LENGTH : INTEGER ;
  HVN_DATA : PACKED ARRAY [ 1..L ] OF CHAR ;
END ;
```

in any <Pascal XML BLOB variable>, where:

a) *L* is the numeric value of <large object length> as specified in Subclause 5.1, "<token> and <separator>".

b) *XO* is either <document or content> as specified in Subclause 6.8, "<string value function>", or the zero-length string.

c) *XWO* is either <XML whitespace option> as specified in Subclause 6.16, "<XML parse>", or the zero-length string.

HVN is an *XML BLOB host variable*. *L* is the *length of XML BLOB host variable*. *CS* is the *character set of XML BLOB host variable*. If *XO* is the zero-length string, then it is implementation-defined whether DOCUMENT or CONTENT is the XML option of XML BLOB host variable; otherwise *XO* is the *XML option of XML BLOB host variable*. If *XWO* is the zero-length string, then it is implementation-defined whether STRIP WHITESPACE or PRESERVE WHITESPACE is the *XML whitespace option of the XML BLOB host variable*; otherwise *XWO* is the *XML whitespace option of the XML BLOB host variable*.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Conformance Rules

1) Insert this CR Without Feature X111, “Host language support for XML: CLOB mapping”, conforming SQL language shall not contain an <Pascal XML CLOB variable>.

2) Insert this CR Without Feature X100, “Host language support for XML: CONTENT option”, in conforming SQL language, <Pascal XML CLOB variable> shall not immediately contain a <document or content> that is CONTENT.

3) Insert this CR Without Feature X101, “Host language support for XML: DOCUMENT option”, in conforming SQL language, <Pascal XML CLOB variable> shall not immediately contain a <document or content> that is DOCUMENT.

4) Insert this CR Without Feature X112, “Host language support for XML: BLOB mapping”, conforming SQL language shall not contain a <Pascal XML BLOB variable>.

5) Insert this CR Without Feature X100, “Host language support for XML: CONTENT option”, in conforming SQL language, <Pascal XML BLOB variable> shall not immediately contain a <document or content> that is CONTENT.

6) Insert this CR Without Feature X101, “Host language support for XML: DOCUMENT option”, in conforming SQL language, <Pascal XML BLOB variable> shall not immediately contain a <document or content> that is DOCUMENT.

7) Insert this CR Without Feature X113, “Host language support for XML: STRIP WHITESPACE option”, in conforming SQL language, <Pascal XML CLOB variable> shall not immediately contain an <XML whitespace option> that is STRIP WHITESPACE.

8) Insert this CR Without Feature X113, “Host language support for XML: STRIP WHITESPACE option”, in conforming SQL language, <Pascal XML BLOB variable> shall not immediately contain an <XML whitespace option> that is STRIP WHITESPACE.

9) Insert this CR Without Feature X114, “Host language support for XML: PRESERVE WHITESPACE option”, in conforming SQL language, <Pascal XML CLOB variable> shall not immediately contain an <XML whitespace option> that is PRESERVE WHITESPACE.

10) Insert this CR Without Feature X114, “Host language support for XML: PRESERVE WHITESPACE option”, in conforming SQL language, <Pascal XML BLOB variable> shall not immediately contain an <XML whitespace option> that is PRESERVE WHITESPACE.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

18.8 <embedded SQL PL/I program>

This Subclause modifies Subclause 21.9, "<embedded SQL PL/I program>", in ISO/IEC 9075-2.

Function

Specify an <embedded SQL PL/I program>.

Format

```
<PL/I derived type specification> ::=
    !! All alternatives from ISO/IEC 9075-2
  | <PL/I XML VARCHAR variable>
  | <PL/I XML CLOB variable>

<PL/I XML VARCHAR variable> ::=
  SQL TYPE IS XML [ <document or content> ]
      [ <XML whitespace option> ]
      AS VARCHAR <left paren> <character length> <right paren>
      [ CHARACTER SET [ IS ] <character set specification> ]

<PL/I XML CLOB variable> ::=
  SQL TYPE IS XML [ <document or content> ]
      [ <XML whitespace option> ]
      AS CLOB <left paren> <character large object length> <right paren>
      [ CHARACTER SET [ IS ] <character set specification> ]

<PL/I XML BLOB variable> ::=
  SQL TYPE IS XML [ <document or content> ]
      [ <XML whitespace option> ]
      AS BLOB <left paren> <large object length> <right paren>
```

Syntax Rules

1) Insert after SR 5)d) The syntax

```
SQL TYPE IS XML XO XWO AS VARCHAR (L)
  CHARACTER SET IS CS
```

for a given <PL/I host identifier> *HVN* shall be replaced by:

```
CHARACTER VARYING (LL)
```

in any <PL/I XML VARCHAR variable>, where:

a) *LL* is the numeric value of *L* and the value of *LL* is *NV***k*.

b) *XO* is either <document or content> as specified in Subclause 6.8, "<string value function>", or the zero-length string.

c) *XWO* is either <XML whitespace option> as specified in Subclause 6.16, "<XML parse>", or the zero-length string.

d) *CS* is a <character set name> as specified in Subclause 5.2, "Names and identifiers".

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

HVN is an *XML VARCHAR host variable*. *L* is the *length of XML VARCHAR host variable*. *CS* is the *character set of XML VARCHAR host variable*. If *XO* is the zero-length string, then it is implementation-defined whether DOCUMENT or CONTENT is the XML option of XML VARCHAR host variable; otherwise *XO* is the *XML option of XML VARCHAR host variable*. If *XWO* is the zero-length string, then it is implementation-defined whether STRIP WHITESPACE or PRESERVE WHITESPACE is the *XML whitespace option of the XML VARCHAR host variable*; otherwise *XWO* is the *XML whitespace option of the XML VARCHAR host variable*.

NOTE 69 — The length of the XML VARCHAR host variable contains an implicit or explicit <char length units>.

2) Insert after SR 5)d) The syntax

```
SQL TYPE IS XML XO XWO AS CLOB ( L )
  CHARACTER SET IS CS
```

for a given <PL/I host identifier> *HVN* shall be replaced by:

```
DCL 1 HVN
      2 HVN_RESERVED FIXED BINARY (31),
      2 HVN_LENGTH   FIXED BINARY (31),
      2 HVN_DATA     CHARACTER ( LL );
```

in any <PL/I XML CLOB variable>, where:

a) *NV* is the numeric value of *L* as specified in Subclause 6.1, "<data type>", and the value of *LL* is *NV***k*.

b) *XO* is either <document or content> as specified in Subclause 6.8, "<string value function>", or the zero-length string.

c) *XWO* is either <XML whitespace option> as specified in Subclause 6.16, "<XML parse>", or the zero-length string.

d) *CS* is a <character set name> as specified in Subclause 5.2, "Names and identifiers".

HVN is an *XML CLOB host variable*. *L* is the *length of XML CLOB host variable*. *CS* is the *character set of XML CLOB host variable*. If *XO* is the zero-length string, then it is implementation-defined whether DOCUMENT or CONTENT is the XML option of XML CLOB host variable; otherwise *XO* is the *XML option of XML CLOB host variable*. If *XWO* is the zero-length string, then it is implementation-defined whether STRIP WHITESPACE or PRESERVE WHITESPACE is the *XML whitespace option of the XML CLOB host variable*; otherwise *XWO* is the *XML whitespace option of the XML CLOB host variable*.

NOTE 70 — The length of the XML CLOB host variable contains an implicit or explicit <char length units>.

3) Insert after SR 5)d) The syntax

```
SQL TYPE IS XML XO XWO AS BLOB ( L )
```

for a given <PL/I host identifier> *HVN* shall be replaced by:

```
DCL 1 HVN
      2 HVN_RESERVED FIXED BINARY (31),
      2 HVN_LENGTH   FIXED BINARY (31),
      2 HVN_DATA     CHARACTER ( L );
```

in any <PL/I XML CLOB variable>, where:

a) *L* is the numeric value of <large object length> as specified in Subclause 5.1, "<token> and <separator>".

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

b) *XO* is either <document or content> as specified in Subclause 6.8, "<string value function>", or the zero-length string.

c) *XWO* is either <XML whitespace option> as specified in Subclause 6.16, "<XML parse>", or the zero-length string.

d) *CS* is a <character set name> as specified in Subclause 5.2, "Names and identifiers".

HVN is an *XML BLOB host variable*. *L* is the *length of XML BLOB host variable*. *CS* is the *character set of XML BLOB host variable*. If *XO* is the zero-length string, then it is implementation-defined whether DOCUMENT or CONTENT is the XML option of XML BLOB host variable; otherwise *XO* is the *XML option of XML BLOB host variable*. If *XWO* is the zero-length string, then it is implementation-defined whether STRIP WHITESPACE or PRESERVE WHITESPACE is the *XML whitespace option of the XML BLOB host variable*; otherwise *XWO* is the *XML whitespace option of the XML BLOB host variable*.

4) Insert after SR 9) In a <PL/I XML VARCHAR variable>, if a <character set specification> is specified, then the SQL data type of the character string resulting from the invocation of the XMLSERIALIZE operator on a value of XML type is VARCHAR whose character set is the same as the character set specified by the <character set specification>. If <character set specification> is not specified, then an implementation-defined <character set specification> is implicit.

5) Insert after SR 10) In a <PL/I XML CLOB variable>, if a <character set specification> is specified, then the SQL data type of the character string resulting from the invocation of the XMLSERIALIZE operator on a value of XML type is CHARACTER LARGE OBJECT whose character set is the same as the character set specified by the <character set specification>. If <character set specification> is not specified, then an implementation-defined <character set specification> is implicit.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

1) Insert this CR Without Feature X110, "Host language support for XML: VARCHAR mapping", conforming SQL language shall not contain an <PL/I XML VARCHAR variable>.

2) Insert this CR Without Feature X111, "Host language support for XML: CLOB mapping", conforming SQL language shall not contain an <PL/I XML CLOB variable>.

3) Insert this CR Without Feature X100, "Host language support for XML: CONTENT option", in conforming SQL language, neither <PL/I XML VARCHAR variable> nor <PL/I XML CLOB variable> shall immediately contain a <document or content> that is CONTENT.

4) Insert this CR Without Feature X101, "Host language support for XML: DOCUMENT option", in conforming SQL language, neither <PL/I XML VARCHAR variable> nor <PL/I XML CLOB variable> shall immediately contain a <document or content> that is DOCUMENT.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

5) Insert this CR Without Feature X112, “Host language support for XML: BLOB mapping”, conforming SQL language shall not contain a <PL/I XML BLOB variable>.

6) Insert this CR Without Feature X100, “Host language support for XML: CONTENT option”, in conforming SQL language, <PL/I XML BLOB variable> shall not immediately contain a <document or content> that is CONTENT.

7) Insert this CR Without Feature X101, “Host language support for XML: DOCUMENT option”, in conforming SQL language, <PL/I XML BLOB variable> shall not immediately contain a <document or content> that is DOCUMENT.

8) Insert this CR Without Feature X113, “Host language support for XML: STRIP WHITESPACE option”, in conforming SQL language, <PL/I XML CLOB variable> shall not immediately contain an <XML whitespace option> that is STRIP WHITESPACE.

9) Insert this CR Without Feature X113, “Host language support for XML: STRIP WHITESPACE option”, in conforming SQL language, <PL/I XML BLOB variable> shall not immediately contain an <XML whitespace option> that is STRIP WHITESPACE.

10) Insert this CR Without Feature X113, “Host language support for XML: STRIP WHITESPACE option”, in conforming SQL language, <PL/I XML VARCHAR variable> shall not immediately contain an <XML whitespace option> that is STRIP WHITESPACE.

11) Insert this CR Without Feature X114, “Host language support for XML: PRESERVE WHITESPACE option”, in conforming SQL language, <PL/I XML CLOB variable> shall not immediately contain an <XML whitespace option> that is PRESERVE WHITESPACE.

12) Insert this CR Without Feature X114, “Host language support for XML: PRESERVE WHITESPACE option”, in conforming SQL language, <PL/I XML BLOB variable> shall not immediately contain an <XML whitespace option> that is PRESERVE WHITESPACE.

13) Insert this CR Without Feature X114, “Host language support for XML: PRESERVE WHITESPACE option”, in conforming SQL language, <PL/I XML VARCHAR variable> shall not immediately contain an <XML whitespace option> that is PRESERVE WHITESPACE.

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

19 Diagnostics management

This Clause modifies Clause 23, "Diagnostics management", in ISO/IEC 9075-2.

19.1 <get diagnostics statement>

This Subclause modifies Subclause 23.1, "<get diagnostics statement>", in ISO/IEC 9075-2.

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

1) Augment 1st paragraph

Table 13 — SQL-statement codes

SQL-statement	Identifier	Code
All alternatives from ISO/IEC 9075-2		
<set XML option statement>	SET XML OPTION	138

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Conformance Rules

No additional Conformance Rules.

,,,,`-`-`,,`,,`,`,,`---
© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

20 Information Schema

This Clause modifies Clause 5, "Information Schema", in ISO/IEC 9075-11.

20.1 NCNAME domain

Function

Define a domain that contains an NCNAME.

Definition

```
CREATE DOMAIN NCNAME AS
  CHARACTER VARYING(L)
    CHARACTER SET CS;
GRANT USAGE ON NCNAME
    TO PUBLIC WITH GRANT OPTION;
```

Description

1) It is implementation-defined whether this domain specifies all variable-length character string values that conform to the rules for formation and representation of an XML 1.0 NCName or an XML 1.1 NCName.

   NOTE 71 — There is no way in SQL to specify a <domain constraint> that would be true for an XML 1.0 NCName or an XML 1.1 NCName and false for all other character string values.

2) *L* is the implementation-defined maximum length of an XML 1.0 NCName or XML 1.1 NCName.

3) *CS* is an implementation-defined character set whose character repertoire is UCS.

Conformance Rules

1) Without Feature F251, "Domain support", conforming SQL language shall not reference INFORMATION_SCHEMA.NCNAME.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

20.2 URI domain

Function

Define a domain that contains a URI.

Definition

```
CREATE DOMAIN URI AS
  CHARACTER VARYING(L)
    CHARACTER SET CS;
GRANT USAGE ON URI
    TO PUBLIC WITH GRANT OPTION;
```

Description

1) This domain specifies all variable-length character string values that conform to the rules for formation and representation of an <XML URI>.

2) *L* is the implementation-defined maximum length of an <XML URI>.

3) *CS* is an implementation-defined character set whose character repertoire is UCS.

Conformance Rules

1) Without Feature F251, “Domain support”, conforming SQL language shall not reference INFORMATION_SCHEMA.URI.

© ISO/IEC 2008 – All rights reserved
--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

20.3 ATTRIBUTES view

This Subclause modifies Subclause 5.11, "ATTRIBUTES view", in ISO/IEC 9075-11.

Function

Identify the attributes of user-defined types defined in this catalog that are accessible to a given user or role.

Definition

Add the following columns to the end of outermost select list of the view definition

```
,
D1.XML_PRIMARY_MODIFIER, D1.XML_SECONDARY_MODIFIER,
D1.XML_SCHEMA_CATALOG, D1.XML_SCHEMA_SCHEMA, D1.XML_SCHEMA_NAME,
D1.XML_SCHEMA_NAMESPACE, D1.XML_SCHEMA_ELEMENT
```

Conformance Rules

1) Insert this CR Without Feature X160, "Basic Information Schema for registered XML Schemas", conforming SQL language shall not reference INFORMATION_SCHEMA.ATTRIBUTES.XML_SCHEMA_CATALOG, INFORMATION_SCHEMA.ATTRIBUTES.XML_SCHEMA_SCHEMA, or INFORMATION_SCHEMA.ATTRIBUTES.XML_SCHEMA_NAME.

2) Insert this CR Without Feature X161, "Advanced Information Schema for registered XML Schemas", conforming SQL language shall not reference INFORMATION_SCHEMA.ATTRIBUTES.XML_SCHEMA_NAMESPACE or INFORMATION_SCHEMA.ATTRIBUTES.XML_SCHEMA_ELEMENT.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

20.4 COLUMNS view

This Subclause modifies Subclause 5.21, "COLUMNS view", in ISO/IEC 9075-11.

Function

Identify the columns of tables defined in this catalog that are accessible to a given user or role.

Definition

Add the following columns to the end of outermost select list of the view definition

```
,
COALESCE (D1.XML_PRIMARY_MODIFIER, D2.XML_PRIMARY_MODIFIER) AS
         XML_PRIMARY_MODIFIER,
COALESCE (D1.XML_SECONDARY_MODIFIER, D2.XML_SECONDARY_MODIFIER) AS
         XML_SECONDARY_MODIFIER,
COALESCE (D1.XML_SCHEMA_CATALOG, D2.XML_SCHEMA_CATALOG) AS
         XML_SCHEMA_CATALOG,
COALESCE (D1.XML_SCHEMA_SCHEMA, D2.XML_SCHEMA_SCHEMA) AS
         XML_SCHEMA_SCHEMA,
COALESCE (D1.XML_SCHEMA_NAME, D2.XML_SCHEMA_NAME) AS
         XML_SCHEMA_NAME,
COALESCE (D1.XML_SCHEMA_NAMESPACE, D2.XML_SCHEMA_NAMESPACE) AS
         XML_SCHEMA_NAMESPACE,
COALESCE (D1.XML_SCHEMA_ELEMENT, D2.XML_SCHEMA_ELEMENT) AS
         XML_SCHEMA_ELEMENT
```

Conformance Rules

1) Insert this CR Without Feature X160, "Basic Information Schema for registered XML Schemas", conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMNS.XML_SCHEMA_CATALOG, INFORMATION_SCHEMA.COLUMNS.XML_SCHEMA_SCHEMA, or INFORMATION_SCHEMA.COLUMNS.XML_SCHEMA_NAME.

2) Insert this CR Without Feature X161, "Advanced Information Schema for registered XML Schemas", conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMNS.XML_SCHEMA_NAMESPACE or INFORMATION_SCHEMA.COLUMNS.XML_SCHEMA_ELEMENT.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

20.5 DOMAINS view

This Subclause modifies Subclause 5.28, "DOMAINS view", in ISO/IEC 9075-11.

Function

Identify the domains defined in this catalog that are accessible to a given user or role.

Definition

Add the following columns to the end of outermost select list of the view definition

```
,
D2.XML_PRIMARY_MODIFIER, D2.XML_SECONDARY_MODIFIER,
D2.XML_SCHEMA_CATALOG, D2.XML_SCHEMA_SCHEMA, D2.XML_SCHEMA_NAME,
D2.XML_SCHEMA_NAMESPACE, D2.XML_SCHEMA_ELEMENT
```

Conformance Rules

1) Insert this CR Without Feature X160, "Basic Information Schema for registered XML Schemas", conforming SQL language shall not reference INFORMATION_SCHEMA.DOMAINS.XML_SCHEMA_CATALOG, INFORMATION_SCHEMA.DOMAINS.XML_SCHEMA_SCHEMA, or INFORMATION_SCHEMA.DOMAINS.XML_SCHEMA_NAME.

2) Insert this CR Without Feature X161, "Advanced Information Schema for registered XML Schemas", conforming SQL language shall not reference INFORMATION_SCHEMA.DOMAINS.XML_SCHEMA_NAMESPACE or INFORMATION_SCHEMA.DOMAINS.XML_SCHEMA_ELEMENT.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

20.6 ELEMENT_TYPES view

This Subclause modifies Subclause 5.29, "ELEMENT_TYPES view", in ISO/IEC 9075-11.

Function

Identify the collection element types defined in this catalog that are accessible to a given user or role.

Definition

Add the following columns to the end of outermost select list of the view definition

```
,
D.XML_PRIMARY_MODIFIER, D.XML_SECONDARY_MODIFIER,
D.XML_SCHEMA_CATALOG, D.XML_SCHEMA_SCHEMA, D.XML_SCHEMA_NAME,
D.XML_SCHEMA_NAMESPACE, D.XML_SCHEMA_ELEMENT
```

Conformance Rules

1) Insert this CR Without Feature X160, "Basic Information Schema for registered XML Schemas", conforming SQL language shall not reference INFORMATION_SCHEMA.ELEMENT_TYPES.XML_SCHEMA_CATALOG, INFORMATION_SCHEMA.ELEMENT_TYPES.XML_SCHEMA_SCHEMA, or INFORMATION_SCHEMA.ELEMENT_TYPES.XML_SCHEMA_NAME.

2) Insert this CR Without Feature X161, "Advanced Information Schema for registered XML Schemas", conforming SQL language shall not reference INFORMATION_SCHEMA.ELEMENT_TYPES.XML_SCHEMA_NAMESPACE or INFORMATION_SCHEMA.ELEMENT_TYPES.XML_SCHEMA_ELEMENT.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

20.7 FIELDS view

This Subclause modifies Subclause 5.31, "FIELDS view", in ISO/IEC 9075-11.

Function

Identify the field types defined in this catalog that are accessible to a given user or role.

Definition

Add the following columns to the end of outermost select list of the view definition

```
,
D.XML_PRIMARY_MODIFIER, D.XML_SECONDARY_MODIFIER,
D.XML_SCHEMA_CATALOG, D.XML_SCHEMA_SCHEMA, D.XML_SCHEMA_NAME,
D.XML_SCHEMA_NAMESPACE, D.XML_SCHEMA_ELEMENT
```

Conformance Rules

1) Insert this CR Without Feature X160, "Basic Information Schema for registered XML Schemas", conforming SQL language shall not reference INFORMATION_SCHEMA.FIELDS.XML_SCHEMA_CATALOG, INFORMATION_SCHEMA.FIELDS.XML_SCHEMA_SCHEMA, or INFORMATION_SCHEMA.FIELDS.XML_SCHEMA_NAME.

2) Insert this CR Without Feature X161, "Advanced Information Schema for registered XML Schemas", conforming SQL language shall not reference INFORMATION_SCHEMA.FIELDS.XML_SCHEMA_NAMESPACE or INFORMATION_SCHEMA.FIELDS.XML_SCHEMA_ELEMENT.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

20.8 METHOD_SPECIFICATION_PARAMETERS view

This Subclause modifies Subclause 5.33, "METHOD_SPECIFICATION_PARAMETERS view", in ISO/IEC 9075-11.

Function

Identify the SQL parameters of method specifications described in the METHOD_SPECIFICATIONS view that are accessible to a given user or role.

Definition

Add the following columns to the end of outermost select list of the view definition

```
,
D.XML_PRIMARY_MODIFIER, D.XML_SECONDARY_MODIFIER,
D.XML_SCHEMA_CATALOG, D.XML_SCHEMA_SCHEMA, D.XML_SCHEMA_NAME,
D.XML_SCHEMA_NAMESPACE, D.XML_SCHEMA_ELEMENT
```

Conformance Rules

1) Insert this CR Without Feature X160, "Basic Information Schema for registered XML Schemas", conforming SQL language shall not reference INFORMATION_SCHEMA.METHOD_SPECIFICATION_PARAMETERS.XML_SCHEMA_CATALOG, INFORMATION_SCHEMA.METHOD_SPECIFICATION_PARAMETERS.XML_SCHEMA_SCHEMA, or INFORMATION_SCHEMA.METHOD_SPECIFICATION_PARAMETERS.XML_SCHEMA_NAME.

2) Insert this CR Without Feature X161, "Advanced Information Schema for registered XML Schemas", conforming SQL language shall not reference INFORMATION_SCHEMA.METHOD_SPECIFICATION_PARAMETERS.XML_SCHEMA_NAMESPACE or INFORMATION_SCHEMA.METHOD_SPECIFICATION_PARAMETERS.XML_SCHEMA_ELEMENT.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

20.9 METHOD_SPECIFICATIONS view

This Subclause modifies Subclause 5.34, "METHOD_SPECIFICATIONS view", in ISO/IEC 9075-11.

Function

Identify the SQL-invoked methods in the catalog that are accessible to a given user or role.

Definition

Add the following columns to the end of outermost select list of the view definition

```
,
D.XML_PRIMARY_MODIFIER, D.XML_SECONDARY_MODIFIER,
D.XML_SCHEMA_CATALOG, D.XML_SCHEMA_SCHEMA, D.XML_SCHEMA_NAME,
D.XML_SCHEMA_NAMESPACE, D.XML_SCHEMA_ELEMENT,
DT.XML_PRIMARY_MODIFIER AS RESULT_CAST_XML_PRIMARY_MODIFIER,
DT.XML_SECONDARY_MODIFIER AS RESULT_CAST_XML_SECONDARY_MODIFIER,
DT.XML_SCHEMA_CATALOG AS RESULT_CAST_XML_SCHEMA_CATALOG,
DT.XML_SCHEMA_SCHEMA AS RESULT_CAST_XML_SCHEMA_SCHEMA,
DT.XML_SCHEMA_NAME AS RESULT_CAST_XML_SCHEMA_NAME,
DT.XML_SCHEMA_NAMESPACE AS RESULT_CAST_XML_SCHEMA_NAMESPACE,
DT.XML_SCHEMA_ELEMENT AS RESULT_CAST_XML_SCHEMA_ELEMENT
```

Conformance Rules

1) Insert this CR Without Feature X160, "Basic Information Schema for registered XML Schemas", conforming SQL language shall not reference INFORMATION_SCHEMA.METHOD_SPECIFICATIONS.XML_SCHEMA_CATALOG, INFORMATION_SCHEMA.METHOD_SPECIFICATIONS.XML_SCHEMA_SCHEMA, INFORMATION_SCHEMA.METHOD_SPECIFICATIONS.XML_SCHEMA_NAME, INFORMATION_SCHEMA.METHOD_SPECIFICATION_PARAMETERS.RESULT_CAST_XML_SCHEMA_CATALOG, INFORMATION_SCHEMA.METHOD_SPECIFICATION_PARAMETERS.RESULT_CAST_XML_SCHEMA_SCHEMA, or INFORMATION_SCHEMA.METHOD_SPECIFICATION_PARAMETERS.RESULT_CAST_XML_SCHEMA_NAME.

2) Insert this CR Without Feature X161, "Advanced Information Schema for registered XML Schemas", conforming SQL language shall not reference INFORMATION_SCHEMA.METHOD_SPECIFICATIONS.XML_SCHEMA_NAMESPACE, INFORMATION_SCHEMA.METHOD_SPECIFICATIONS.XML_SCHEMA_ELEMENT, INFORMATION_SCHEMA.METHOD_SPECIFICATIONS.RESULT_CAST_XML_SCHEMA_NAMESPACE, or INFORMATION_SCHEMA.METHOD_SPECIFICATIONS.RESULT_CAST_XML_SCHEMA_ELEMENT.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

20.10 PARAMETERS view

This Subclause modifies Subclause 5.35, "PARAMETERS view", in ISO/IEC 9075-11.

Function

Identify the SQL parameters of SQL-invoked routines defined in this catalog that are accessible to a given user or role.

Definition

Add the following columns to the <select list>

```
,
DTD.XML_PRIMARY_MODIFIER, DTD.XML_SECONDARY_MODIFIER,
DTD.XML_SCHEMA_CATALOG, DTD.XML_SCHEMA_SCHEMA, DTD.XML_SCHEMA_NAME,
DTD.XML_SCHEMA_NAMESPACE, DTD.XML_SCHEMA_ELEMENT,
DTD2.DATA_TYPE AS XML_CHAR_TYPE,
DTD2.CHARACTER_MAXIMUM_LENGTH AS XML_CHAR_MAX_LEN,
DTD2.CHARACTER_OCTET_LENGTH AS XML_CHAR_OCT_LEN,
DTD2.CHARACTER_SET_CATALOG AS XML_CHAR_SET_CAT,
DTD2.CHARACTER_SET_SCHEMA AS XML_CHAR_SET_SCH,
DTD2.CHARACTER_SET_NAME AS XML_CHAR_SET_NAME,
DTD2.COLLATION_CATALOG AS XML_CHAR_COLL_CAT,
DTD2.COLLATION_SCHEMA AS XML_CHAR_COLL_SCH,
DTD2.COLLATION_NAME AS XML_CHAR_COLL_NAME,
DTD2.DTD_IDENTIFIER AS XML_CHAR_DTD_ID,
P.XML_OPTION, P.XML_PASSING_MECHANISM
```

Augment the LEFT JOIN with the following additional LEFT JOIN

```
LEFT JOIN
  DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR DTD2
  ON ( P.SPECIFIC_CATALOG, P.SPECIFIC_SCHEMA, P.SPECIFIC_NAME,
       'ROUTINE', P.XML_STRING_DTD_IDENTIFIER )
   = ( DTD2.OBJECT_CATALOG, DTD2.OBJECT_SCHEMA, DTD2.OBJECT_NAME,
       DTD2.OBJECT_TYPE, DTD2.DTD_IDENTIFIER )
```

Conformance Rules

1) Insert this CR Without Feature X160, "Basic Information Schema for registered XML Schemas", conforming SQL language shall not reference reference INFORMATION_SCHEMA.PARAMETERS.XML_SCHEMA_CATALOG, INFORMATION_SCHEMA.PARAMETERS.XML_SCHEMA_SCHEMA, or INFORMATION_SCHEMA.PARAMETERS.XML_SCHEMA_NAME.

2) Insert this CR Without Feature X161, "Advanced Information Schema for registered XML Schemas", conforming SQL language shall not reference reference INFORMATION_SCHEMA.PARAMETERS.XML_SCHEMA_NAMESPACE or INFORMATION_SCHEMA.PARAMETERS.XML_SCHEMA_ELEMENT.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

20.11 ROUTINES view

This Subclause modifies Subclause 5.49, "ROUTINES view", in ISO/IEC 9075-11.

Function

Identify the SQL-invoked routines in this catalog that are accessible to a given user or role.

Definition

Add the following columns to the <select list> following D.DTD_IDENTIFIER

```
        D.XML_PRIMARY_MODIFIER, D.XML_SECONDARY_MODIFIER,
        D.XML_SCHEMA_CATALOG, D.XML_SCHEMA_SCHEMA, D.XML_SCHEMA_NAME,
        D.XML_SCHEMA_NAMESPACE, D.XML_SCHEMA_ELEMENT, D2.DATA_TYPE AS XML_CHAR_TYPE
```

Add the following columns to the <select list> following DT.DTD_IDENTIFIER AS RESULT_CAST_DTD_IDENTIFIER

```
        ,
        DT.XML_PRIMARY_MODIFIER AS RESULT_CAST_XML_PRIMARY_MODIFIER,
        DT.XML_SECONDARY_MODIFIER AS RESULT_CAST_XML_SECONDARY_MODIFIER,
        DT.XML_SCHEMA_CATALOG AS RESULT_CAST_XML_SCHEMA_CATALOG,
        DT.XML_SCHEMA_SCHEMA AS RESULT_CAST_XML_SCHEMA_SCHEMA,
        DT.XML_SCHEMA_NAME AS RESULT_CAST_XML_SCHEMA_NAME,
        DT.XML_SCHEMA_NAMESPACE AS RESULT_CAST_XML_SCHEMA_NAMESPACE,
        DT.XML_SCHEMA_ELEMENT AS RESULT_CAST_XML_SCHEMA_ELEMENT,
        D2.DATA_TYPE AS XML_CHAR_TYPE,
        D2.CHARACTER_MAXIMUM_LENGTH AS XML_CHAR_MAX_LEN,
        D2.CHARACTER_OCTET_LENGTH AS XML_CHAR_OCT_LEN,
        D2.CHARACTER_SET_CATALOG AS XML_CHAR_SET_CAT,
        D2.CHARACTER_SET_SCHEMA AS XML_CHAR_SET_SCH,
        D2.CHARACTER_SET_NAME AS XML_CHAR_SET_NAME,
        D2.COLLATION_CATALOG AS XML_CHAR_COLL_CAT,
        D2.COLLATION_SCHEMA AS XML_CHAR_COLL_SCH,
        D2.COLLATION_NAME AS XML_CHAR_COLL_NAME,
        D2.DTD_IDENTIFIER AS XML_CHAR_DTD_ID,
        R.XML_OPTION, R.XML_RETURN_MECHANISM
```

Replace the first line of the <from clause> with the following

```
    FROM ( ( ( DEFINITION_SCHEMA.ROUTINES AS R
```

Append the following JOIN to the <from clause>

```
      LEFT JOIN
           DEFINITION_SCHEMA.DATA_TYPE_DESCRIPTOR AS D2
        ON ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
             'ROUTINE', R.XML_STRING_DTD_IDENTIFIER )
         = ( D2.OBJECT_CATALOG, D2.OBJECT_SCHEMA, D2.OBJECT_NAME,
             D2.OBJECT_TYPE, D2.DTD_IDENTIFIER ) )
```

NOTE 72 — The ROUTINES view contains three sets of columns that each describe a data type. The set of columns that are prefixed with "XML_" describes the associated string type, if any, of the result of the <SQL-invoked routine>, if its declared type is XML. The set of columns that are prefixed with "RESULT_CAST_" describes the data type specified in the <result cast>, if any, contained

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

in the <SQL-invoked routine>. The third set of columns describes the data type specified in the <returns data type> contained in the <SQL-invoked routine>.

Conformance Rules

1) Insert this CR Without Feature X160, "Basic Information Schema for registered XML Schemas", conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINES.XML_SCHEMA_CATALOG, INFORMATION_SCHEMA.ROUTINES.XML_SCHEMA_SCHEMA, INFORMATION_SCHEMA.ROUTINES.XML_SCHEMA_NAME, NFORMATION_SCHEMA.ROUTINES.RESULT_CAST_XML_SCHEMA_CATALOG, INFORMATION_SCHEMA.ROUTINES.RESULT_CAST_XML_SCHEMA_SCHEMA, or INFORMATION_SCHEMA.ROUTINES.RESULT_CAST_XML_SCHEMA_NAME.

2) Insert this CR Without Feature X161, "Advanced Information Schema for registered XML Schemas", conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINES.XML_SCHEMA_NAMESPACE, INFORMATION_SCHEMA.ROUTINES.XML_SCHEMA_ELEMENT, INFORMATION_SCHEMA.ROUTINES.RESULT_CAST_XML_SCHEMA_NAMESPACE, or INFORMATION_SCHEMA.ROUTINES.RESULT_CAST_XML_SCHEMA_ELEMENT.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

20.12 XML_SCHEMA_ELEMENTS view

Function

Identify the global element declaration schema components defined in registered XML Schemas that are accessible to a given user or role.

Definition

```
CREATE VIEW XML_SCHEMA_ELEMENTS AS
  SELECT XS.XML_SCHEMA_TARGET_NAMESPACE, XS.XML_SCHEMA_LOCATION,
         XML_SCHEMA_CATALOG, XML_SCHEMA_SCHEMA, XML_SCHEMA_NAME,
         XE.XML_SCHEMA_NAMESPACE, XE.XML_SCHEMA_ELEMENT,
         XE.XML_SCHEMA_ELEMENT_IS_DETERMINISTIC
  FROM DEFINITION_SCHEMA.XML_SCHEMA_ELEMENTS AS XE
  JOIN DEFINITION_SCHEMA.XML_SCHEMAS AS XS
  USING ( XML_SCHEMA_CATALOG, XML_SCHEMA_SCHEMA, XML_SCHEMA_NAME )
  WHERE ( XML_SCHEMA_CATALOG, XML_SCHEMA_SCHEMA,
          XML_SCHEMA_NAME, 'XML SCHEMA' )
       IN ( SELECT UP.OBJECT_CATALOG, UP.OBJECT_SCHEMA,
                   UP.OBJECT_NAME, UP.OBJECT_TYPE
            FROM DEFINITION_SCHEMA.USAGE_PRIVILEGES AS UP
            WHERE ( UP.GRANTEE IN ( 'PUBLIC', CURRENT_USER )
                 OR
                    UP.GRANTEE IN ( SELECT ROLE_NAME
                                    FROM ENABLED_ROLES ) ) )
    AND XML_SCHEMA_CATALOG =
        ( SELECT CATALOG_NAME
          FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE XML_SCHEMA_ELEMENTS
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature X161, “Advanced Information Schema for registered XML Schemas”, conforming SQL language shall not reference INFORMATION_SCHEMA.XML_SCHEMA_ELEMENTS.

2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.XML_SCHEMA_ELEMENTS.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

20.13 XML_SCHEMA_NAMESPACES view

Function

Identify the namespaces that are defined in registered XML Schemas that are accessible to a given user or role.

Definition

```
CREATE VIEW XML_SCHEMA_NAMESPACES AS
    SELECT XS.XML_SCHEMA_TARGET_NAMESPACE, XS.XML_SCHEMA_LOCATION,
           XML_SCHEMA_CATALOG, XML_SCHEMA_SCHEMA, XML_SCHEMA_NAME,
           XSN.XML_SCHEMA_NAMESPACE,
           XSN. XML_SCHEMA_NAMESPACE_IS_DETERMINISTIC
    FROM DEFINITION_SCHEMA.XML_SCHEMA_NAMESPACES AS XSN
    JOIN DEFINITION_SCHEMA.XML_SCHEMAS AS XS
    USING ( XML_SCHEMA_CATALOG, XML_SCHEMA_SCHEMA, XML_SCHEMA_NAME )
    WHERE ( XML_SCHEMA_CATALOG, XML_SCHEMA_SCHEMA,
            XML_SCHEMA_NAME, 'XML SCHEMA' )
        IN ( SELECT UP.OBJECT_CATALOG, UP.OBJECT_SCHEMA,
                    UP.OBJECT_NAME, UP.OBJECT_TYPE
             FROM DEFINITION_SCHEMA.USAGE_PRIVILEGES AS UP
             WHERE ( UP.GRANTEE IN ( 'PUBLIC', CURRENT_USER )
                  OR
                     UP.GRANTEE IN ( SELECT ROLE_NAME
                                     FROM ENABLED_ROLES ) ) )
      AND XML_SCHEMA_CATALOG =
          ( SELECT CATALOG_NAME
            FROM INFORMATION_SCHEMA_CATALOG_NAME );

GRANT SELECT ON TABLE XML_SCHEMA_NAMESPACES
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature X161, “Advanced Information Schema for registered XML Schemas”, conforming SQL language shall not reference INFORMATION_SCHEMA.XML_SCHEMA_NAMESPACES.

2) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.XML_SCHEMA_NAMESPACES.

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

20.14 XML_SCHEMAS view

Function

Identify the registered XML Schemas that are accessible to a given user or role.

Definition

```
CREATE VIEW XML_SCHEMAS AS
    SELECT XML_SCHEMA_TARGET_NAMESPACE, XML_SCHEMA_LOCATION,
    XML_SCHEMA_CATALOG, XML_SCHEMA_SCHEMA, XML_SCHEMA_NAME,
    XML_SCHEMA_IS_DETERMINISTIC, XML_SCHEMA_IS_PERMANENT
FROM DEFINITION_SCHEMA.XML_SCHEMAS
WHERE ( XML_SCHEMA_CATALOG, XML_SCHEMA_SCHEMA,
        XML_SCHEMA_NAME, 'XML SCHEMA' )
     IN ( SELECT UP.OBJECT_CATALOG, UP.OBJECT_SCHEMA,
                 UP.OBJECT_NAME, UP.OBJECT_TYPE
            FROM DEFINITION_SCHEMA.USAGE_PRIVILEGES AS UP
            WHERE ( UP.GRANTEE IN ( 'PUBLIC', CURRENT_USER )
          OR
            UP.GRANTEE IN ( SELECT ROLE_NAME
                            FROM ENABLED_ROLES ) ) )
   AND XML_SCHEMA_CATALOG =
       ( SELECT CATALOG_NAME
         FROM INFORMATION_SCHEMA_CATALOG_NAME );
GRANT SELECT ON TABLE XML_SCHEMAS
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Without Feature X160, “Basic Information Schema for registered XML Schemas”, conforming SQL language shall not reference INFORMATION_SCHEMA.XML_SCHEMAS.

2) Without Feature X161, “Advanced Information Schema for registered XML Schemas”, conforming SQL language shall not reference INFORMATION_SCHEMA.XML_SCHEMAS.SCHEMA_IS_DETERMINISTIC.

3) Without Feature F391, “Long identifiers”, conforming SQL language shall not reference INFORMATION_SCHEMA.XML_SCHEMAS.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

20.15 Short name views

This Subclause modifies Subclause 5.77, "Short name views", in ISO/IEC 9075-11.

Function

Provide alternative views that use only identifiers that do not require Feature F391, "Long identifiers".

Definition

Append the following to the <view column list> of ATTRIBUTES_S view

```
,
XML_PRIMARY_MOD,     XML_SECONDARY_MOD,  XML_SCHEMA_CATALOG,
XML_SCHEMA_SCHEMA,   XML_SCHEMA_NAME,    XML_SCHEMA_NSPACE,
XML_SCHEMA_ELEMENT
```

Append the following to the <select list> of ATTRIBUTES_S view definition

```
,
XML_PRIMARY_MODIFIER, XML_SECONDARY_MODIFIER, XML_SCHEMA_CATALOG,
XML_SCHEMA_SCHEMA, XML_SCHEMA_NAME, XML_SCHEMA_NAMESPACE,
XML_SCHEMA_ELEMENT
```

Append the following to the <view column list> of COLUMN_S view

```
,
XML_PRIMARY_MOD,     XML_SECONDARY_MOD,  XML_SCHEMA_CATALOG,
XML_SCHEMA_SCHEMA,   XML_SCHEMA_NAME,    XML_SCHEMA_NSPACE,
XML_SCHEMA_ELEMENT
```

Append the following to the <select list> of COLUMN_S view definition

```
,
XML_PRIMARY_MODIFIER, XML_SECONDARY_MODIFIER, XML_SCHEMA_CATALOG,
XML_SCHEMA_SCHEMA, XML_SCHEMA_NAME, XML_SCHEMA_NAMESPACE,
XML_SCHEMA_ELEMENT
```

Append the following to the <view column list> of DOMAIN_S view

```
,
XML_PRIMARY_MOD,     XML_SECONDARY_MOD,  XML_SCHEMA_CATALOG,
XML_SCHEMA_SCHEMA,   XML_SCHEMA_NAME,    XML_SCHEMA_NSPACE,
XML_SCHEMA_ELEMENT
```

Append the following to the <select list> of DOMAIN_S view definition

```
,
XML_PRIMARY_MODIFIER, XML_SECONDARY_MODIFIER, XML_SCHEMA_CATALOG,
XML_SCHEMA_SCHEMA, XML_SCHEMA_NAME, XML_SCHEMA_NAMESPACE,
XML_SCHEMA_ELEMENT
```

Append the following to the <view column list> of ELEMENT_TYPES_S view

```
,
XML_PRIMARY_MOD,     XML_SECONDARY_MOD,  XML_SCHEMA_CATALOG,
```


© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
        XML_SCHEMA_SCHEMA,  XML_SCHEMA_NAME,    XML_SCHEMA_NSPACE,
        XML_SCHEMA_ELEMENT
```

Append the following to the <select list> of ELEMENT_TYPES_S view definition

```
        ,
        XML_PRIMARY_MODIFIER, XML_SECONDARY_MODIFIER, XML_SCHEMA_CATALOG,
        XML_SCHEMA_SCHEMA, XML_SCHEMA_NAME, XML_SCHEMA_NAMESPACE,
        XML_SCHEMA_ELEMENT
```

Append the following to the <view column list> of FIELDS_S view

```
        ,
        XML_PRIMARY_MOD,    XML_SECONDARY_MOD,  XML_SCHEMA_CATALOG,
        XML_SCHEMA_SCHEMA,  XML_SCHEMA_NAME,    XML_SCHEMA_NSPACE,
        XML_SCHEMA_ELEMENT
```

Append the following to the <select list> of FIELDS_S view definition

```
        ,
        XML_PRIMARY_MODIFIER, XML_SECONDARY_MODIFIER, XML_SCHEMA_CATALOG,
        XML_SCHEMA_SCHEMA, XML_SCHEMA_NAME, XML_SCHEMA_NAMESPACE,
        XML_SCHEMA_ELEMENT
```

Append the following to the <view column list> of METHOD_SPEC_PARAMS_S view

```
        ,
        XML_PRIMARY_MOD,    XML_SECONDARY_MOD,  XML_SCHEMA_CATALOG,
        XML_SCHEMA_SCHEMA,  XML_SCHEMA_NAME,    XML_SCHEMA_NSPACE,
        XML_SCHEMA_ELEMENT
```

Append the following to the <select list> of METHOD_SPEC_PARAMS_S view definition

```
        ,
        XML_PRIMARY_MODIFIER, XML_SECONDARY_MODIFIER, XML_SCHEMA_CATALOG,
        XML_SCHEMA_SCHEMA, XML_SCHEMA_NAME, XML_SCHEMA_NAMESPACE,
        XML_SCHEMA_ELEMENT
```

Append the following to the <view column list> of METHOD_SPECS_S view

```
        ,
        XML_PRIMARY_MOD,    XML_SECONDARY_MOD,  XML_SCHEMA_CATALOG,
        XML_SCHEMA_SCHEMA,  XML_SCHEMA_NAME,    XML_SCHEMA_NSPACE,
        XML_SCHEMA_ELEMENT, RC_XML_PRIMARY_MOD, RC_XML_SECOND_MOD,
        RC_XML_SCHEMA_CAT,  RC_XML_SCH_SCH,     RC_XML_SCH_NAME,
        RC_XML_SCH_NSPACE,  RC_XML_SCH_ELEMENT
```

Append the following to the <select list> of METHOD_SPECS_S view definition

```
        ,
        XML_PRIMARY_MODIFIER, XML_SECONDARY_MODIFIER, XML_SCHEMA_CATALOG,
        XML_SCHEMA_SCHEMA, XML_SCHEMA_NAME, XML_SCHEMA_NAMESPACE,
        XML_SCHEMA_ELEMENT, RESULT_CAST_XML_PRIMARY_MODIFIER,
        RESULT_CAST_XML_SECONDARY_MODIFIER,
        RESULT_CAST_XML_SCHEMA_CATALOG, RESULT_CAST_XML_SCHEMA_SCHEMA,
        RESULT_CAST_XML_SCHEMA_NAME, RESULT_CAST_XML_SCHEMA_NAMESPACE,
        RESULT_CAST_XML_SCHEMA_ELEMENT
```

Append the following to the <view column list> of PARAMETERS_S view

```
        ,     XML_PRIMARY_MOD,    XML_SECONDARY_MOD,
```


Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

```
        XML_SCHEMA_CATALOG, XML_SCHEMA_SCHEMA,  XML_SCHEMA_NAME,
        XML_SCHEMA_NSPACE,  XML_SCHEMA_ELEMENT, XML_CHAR_TYPE,
        XML_CHAR_MAX_LEN,   XML_CHAR_OCT_LEN,   XML_CHAR_SET_CAT,
        XML_CHAR_SET_SCH,   XML_CHAR_SET_NAME,  XML_CHAR_COLL_CAT,
        XML_CHAR_COLL_SCH,  XML_CHAR_COLL_NAME, XML_CHAR_DTD_ID,
        XML_OPTION,         XML_PASSING_MECH
```

Append the following to the <select list> of PARAMETERS_S view definition

```
        , XML_PRIMARY_MODIFIER, XML_SECONDARY_MODIFIER,
        XML_SCHEMA_CATALOG, XML_SCHEMA_SCHEMA, XML_SCHEMA_NAME,
        XML_SCHEMA_NAMESPACE, XML_SCHEMA_ELEMENT, XML_CHAR_TYPE,
        XML_CHAR_MAX_LEN, XML_CHAR_OCT_LEN, XML_CHAR_SET_CAT,
        XML_CHAR_SET_SCH, XML_CHAR_SET_NAME, XML_CHAR_COLL_CAT,
        XML_CHAR_COLL_SCH, XML_CHAR_COLL_NAME, XML_CHAR_DTD_ID,
        XML_OPTION, XML_PASSING_MECHANISM
```

Add the following columns to the <view column list> of the ROUTINES_S view, following DTD_IDENTIFIER

```
        XML_PRIMARY_MOD,    XML_SECOND_MOD,
        XML_SCHEMA_CAT,     XML_SCH_SCH,        XML_SCH_NAME,
        XML_SCH_NSPACE,     XML_SCH_ELEMENT,
```

Add the following columns to the <view column list> of the ROUTINES_S view, following RC_DTD_IDENTIFIER

```
        ,
        RC_XML_PRIMARY_MOD, RC_XML_SECOND_MOD,  RC_XML_SCHEMA_CAT,
        RC_XML_SCH_SCH,     RC_XML_SCH_NAME,    RC_XML_SCH_NSPACE,
        RC_XML_SCH_ELEMENT, XML_CHAR_TYPE,      XML_CHAR_MAX_LEN,
        XML_CHAR_OCT_LEN,   XML_CHAR_SET_CAT,   XML_CHAR_SET_SCH,
        XML_CHAR_SET_NAME,  XML_CHAR_COLL_CAT,  XML_CHAR_COLL_SCH,
        XML_CHAR_COLL_NAME, XML_CHAR_DTD_ID,    XML_OPTION,
        XML_RETURN_MECH
```

Add the following to the <select list> of ROUTINES_S view definition following DTD_IDENTIFIER

```
        XML_PRIMARY_MODIFIER, XML_SECONDARY_MODIFIER,
        XML_SCHEMA_CATALOG, XML_SCHEMA_SCHEMA, XML_SCHEMA_NAME,
        XML_SCHEMA_NAMESPACE, XML_SCHEMA_ELEMENT,
```

Add the following to the <select list> of ROUTINES_S view definition following RESULT_CAST_DTD_IDENTIFIER

```
        ,
        RESULT_CAST_XML_PRIMARY_MODIFIER,
        RESULT_CAST_XML_SECONDARY_MODIFIER,
        RESULT_CAST_XML_SCHEMA_CATALOG, RESULT_CAST_XML_SCHEMA_SCHEMA,
        RESULT_CAST_XML_SCHEMA_NAME, RESULT_CAST_XML_SCHEMA_NAMESPACE,
        RESULT_CAST_XML_SCHEMA_ELEMENT, XML_CHAR_TYPE, XML_CHAR_MAX_LEN,
        XML_CHAR_OCT_LEN, XML_CHAR_SET_CAT, XML_CHAR_SET_SCH,
        XML_CHAR_SET_NAME, XML_CHAR_COLL_CAT, XML_CHAR_COLL_SCH,
        XML_CHAR_COLL_NAME, XML_CHAR_DTD_ID, XML_OPTION,
        XML_RETURN_MECHANISM
```

Insert the following view definitions:

```
CREATE VIEW XML_SCH_ELEMENTS_S
         ( XML_SCH_TGT_NSPACE, XML_SCH_LOCATION,   XML_SCHEMA_CATALOG,
           XML_SCHEMA_SCHEMA,  XML_SCHEMA_NAME,    XML_SCH_NSPACE,
```


© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
            XML_SCHEMA_ELEMENT, XML_SCH_EL_IS_DET ) AS
        SELECT XML_SCHEMA_TARGET_NAMESPACE, XML_SCHEMA_LOCATION, XML_SCHEMA_CATALOG,
               XML_SCHEMA_SCHEMA, XML_SCHEMA_NAME, XML_SCHEMA_NAMESPACE,
               XML_SCHEMA_ELEMENT, XML_SCHEMA_ELEMENT_IS_DETERMINISTIC
        FROM INFORMATION_SCHEMA.XML_SCHEMA_ELEMENTS;

GRANT SELECT ON TABLE XML_SCH_ELEMENTS_S
    TO PUBLIC WITH GRANT OPTION;



CREATE VIEW XML_SCH_NSPACES_S
            ( XML_SCH_TGT_NSPACE, XML_SCH_LOCATION,   XML_SCHEMA_CATALOG,
              XML_SCHEMA_SCHEMA,  XML_SCHEMA_NAME,    XML_SCH_NSPACE,
              XML_SCH_NSP_IS_DET ) AS
        SELECT XML_SCHEMA_TARGET_NAMESPACE, XML_SCHEMA_LOCATION, XML_SCHEMA_CATALOG,
               XML_SCHEMA_SCHEMA, XML_SCHEMA_NAME, XML_SCHEMA_NAMESPACE,
               XML_SCHEMA_NAMESPACE_IS_DETERMINISTIC
        FROM INFORMATION_SCHEMA.XML_SCHEMA_NAMESPACES;

GRANT SELECT ON TABLE XML_SCH_NSPACES_S
    TO PUBLIC WITH GRANT OPTION;


CREATE VIEW XML_SCHEMAS_S
            ( XML_SCH_TGT_NSPACE, XML_SCH_LOCATION,   XML_SCHEMA_CATALOG,
              XML_SCHEMA_SCHEMA,  XML_SCHEMA_NAME,    XML_SCHEMA_IS_DET,
              XML_SCHEMA_IS_PERM ) AS
        SELECT XML_SCHEMA_TARGET_NAMESPACE, XML_SCHEMA_LOCATION, XML_SCHEMA_CATALOG,
               XML_SCHEMA_SCHEMA, XML_SCHEMA_NAME, XML_SCHEMA_IS_DETERMINISTIC,
               XML_SCHEMA_IS_PERMANENT
        FROM INFORMATION_SCHEMA.XML_SCHEMAS;

GRANT SELECT ON TABLE XML_SCHEMAS_S
    TO PUBLIC WITH GRANT OPTION;
```

Conformance Rules

1) Insert this CR Without Feature X160, “Basic Information Schema for registered XML Schemas”, conforming SQL language shall not reference INFORMATION_SCHEMA.XML_SCHEMAS_S.

2) Insert this CR Without Feature X161, “Advanced Information Schema for registered XML Schemas”, conforming SQL language shall not reference INFORMATION_SCHEMA.XML_SCH_ELEMENTS_S.

3) Insert this CR Without Feature X161, “Advanced Information Schema for registered XML Schemas”, conforming SQL language shall not reference INFORMATION_SCHEMA.XML_SCH_NSPACES_S.

4) Insert this CR Without Feature X161, “Advanced Information Schema for registered XML Schemas”, conforming SQL language shall not reference INFORMATION_SCHEMA.XML_SCHEMAS_S.XML_SCHEMA_IS_DET.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

(Blank page)

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

21 Definition Schema

This Clause modifies Clause 6, "Definition Schema", in ISO/IEC 9075-11.

21.1 DATA_TYPE_DESCRIPTOR base table

This Subclause modifies Subclause 6.21, "DATA_TYPE_DESCRIPTOR base table", in ISO/IEC 9075-11.

Function

The DATA_TYPE_DESCRIPTOR table has one row for each usage of a datatype as identified by ISO/IEC 9075. It effectively contains a representation of the data type descriptors.

Definition

Add the following column definitions

```
XML_PRIMARY_MODIFIER       INFORMATION_SCHEMA.CHARACTER_DATA,
XML_SECONDARY_MODIFIER     INFORMATION_SCHEMA.CHARACTER_DATA,
XML_SCHEMA_CATALOG         INFORMATION_SCHEMA.SQL_IDENTIFIER,
XML_SCHEMA_SCHEMA          INFORMATION_SCHEMA.SQL_IDENTIFIER,
XML_SCHEMA_NAME            INFORMATION_SCHEMA.SQL_IDENTIFIER,
XML_SCHEMA_NAMESPACE       INFORMATION_SCHEMA.URI,
XML_SCHEMA_ELEMENT         INFORMATION_SCHEMA.NCNAME,
```

Augment constraint DATA_TYPE_DESCRIPTOR_DATA_TYPE_CHECK_COMBINATIONS:

Add the following predicate to each OR clause excepting the final OR clause of the constraint:

```
AND
  ( XML_PRIMARY_MODIFIER, XML_SECONDARY_MODIFIER, XML_SCHEMA_CATALOG,
    XML_SCHEMA_SCHEMA, XML_SCHEMA_NAME, XML_SCHEMA_NAMESPACE,
    XML_SCHEMA_ELEMENT ) IS NULL
```

Add the following OR clause to the end of the constraint:

```
OR
  ( DATA_TYPE = 'XML'
  AND
    ( XML_PRIMARY_MODIFIER IN ( 'DOCUMENT', 'CONTENT', 'SEQUENCE' ) )
    AND
      ( XML_SECONDARY_MODIFIER IN ( 'UNTYPED', 'ANY', 'XMLSCHEMA' ) )
  AND
    XML_PRIMARY_MODIFIER IS NOT NULL
  AND
    ( XML_PRIMARY_MODIFIER NOT IN ( 'DOCUMENT', 'CONTENT' )
```

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
            OR
              XML_SECONDARY_MODIFIER IS NOT NULL )
          AND
              ( XML_SECONDARY_MODIFIER <> 'XMLSCHEMA'
              OR
                ( XML_SCHEMA_CATALOG, XML_SCHEMA_SCHEMA, XML_SCHEMA_NAME ) IS NULL )
          AND
            ( CHARACTER_SET_CATALOG, CHARACTER_SET_SCHEMA, CHARACTER_SET_NAME,
              CHARACTER_OCTET_LENGTH, CHARACTER_MAXIMUM_LENGTH,
              COLLATION_CATALOG, COLLATION_SCHEMA, COLLATION_NAME )
                IS NULL
          AND
            ( NUMERIC_PRECISION, NUMERIC_PRECISION_RADIX, NUMERIC_SCALE )
                IS NULL
          AND
            DATETIME_PRECISION IS NULL
          AND
            ( INTERVAL_TYPE, INTERVAL_PRECISION )
                IS NULL
          AND
            ( USER_DEFINED_TYPE_CATALOG, USER_DEFINED_TYPE_SCHEMA,
              USER_DEFINED_TYPE_NAME ) IS NULL
          AND
            ( SCOPE_CATALOG, SCOPE_SCHEMA, SCOPE_NAME ) IS NULL
          AND
            MAXIMUM_CARDINALITY IS NULL )
```

Add 'XML' to the IN list of the final OR clause of the constraint

Add the following check constraint at the end of the constraints:

```
,
CONSTRAINT
    DATA_TYPE_DESCRIPTOR_CHECK_REFERENCES_XML_SCHEMA_NAMESPACES
  CHECK ( XML_SCHEMA_CATALOG NOT IN
          ( SELECT CATALOG_NAME
            FROM SCHEMATA )
        OR
         ( XML_SCHEMA_CATALOG, XML_SCHEM_SCHEMA, XML_SCHEMA_NAME,
           XML_SCHEMA_NAMESPACE) IN
           ( SELECT XML_SCHEMA_CATALOG, XML_SCHEMA_SCHEMA,
                    XML_SCHEMA_NAME, XML_SCHEMA_NAMESPACE
           FROM XML_SCHEMA_NAMESPACES )
         )
```

Add the following check constraint at the end of the constraints:

```
,
CONSTRAINT
    DATA_TYPE_DESCRIPTOR_CHECK_REFERENCES_XML_SCHEMA_ELEMENTS
  CHECK ( XML_SCHEMA_CATALOG NOT IN
          ( SELECT CATALOG_NAME
            FROM SCHEMATA )
        OR
         ( XML_SCHEMA_CATALOG, XML_SCHEM_SCHEMA, XML_SCHEMA_NAME,
           XML_SCHEMA_NAMESPACE, XML_SCHEMA_ELEMENT) IN
           ( SELECT XML_SCHEMA_CATALOG, XML_SCHEMA_SCHEMA,
                    XML_SCHEMA_NAME, XML_SCHEMA_NAMESPACE,
```


© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
            XML_SCHEMA_ELEMENT
        FROM XML_SCHEMA_ELEMENTS )
    )
```

Description

1) Insert this Descr. Case:

   a) If DATA_TYPE is 'XML', then:

      i) The value of XML_PRIMARY_MODIFIER is 'DOCUMENT', 'CONTENT', or 'SEQUENCE'.

      ii) If the value of XML_PRIMARY_MODIFIER is either 'DOCUMENT' or 'CONTENT', then the value of XML_SECONDARY_MODIFIER is 'UNTYPED', 'ANY', or 'XMLSCHEMA'; otherwise, the value of XML_SECONDARY_MODIFIER is the null value.

      iii) Case:

         1) If the value of XML_SECONDARY_MODIFIER is 'XMLSCHEMA', then

            A) The values the values of XML_SCHEMA_CATALOG, XML_SCHEMA_SCHEMA, and XML_SCHEMA_NAME are the catalog name, the unqualified schema name, and the name of a registered XML Schema, respectively, identified by the registered XML Schema descriptor included in the XML type descriptor.

            B) The value of XML_SCHEMA_NAMESPACE is:

               Case:

               I) If the XML type descriptor contains an XML namespace URI *NS*, then *NS*.

               II) Otherwise, the null value.

            C) The value of XML_SCHEMA_ELEMENT is:

               Case:

               I) If the XML type descriptor contains an XML NCName *EN*, then *EN*.

               II) Otherwise, the null value.

         2) Otherwise, the values of XML_SCHEMA_CATALOG, XML_SCHEMA_SCHEMA, XML_SCHEMA_NAME, XML_SCHEMA_NAMESPACE, and XML_SCHEMA_ELEMENT are the null value.

   b) Otherwise, the values of XML_PRIMARY_MODIFIER, XML_SECONDARY_MODIFIER, XML_SCHEMA_CATALOG, XML_SCHEMA_SCHEMA, XML_SCHEMA_NAME, XML_SCHEMA_NAMESPACE and XML_SCHEMA_ELEMENT are the null value.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

21.2 PARAMETERS base table

This Subclause modifies Subclause 6.31, "PARAMETERS base table", in ISO/IEC 9075-11.

Function

The PARAMETERS table has one row for each SQL parameter of each SQL-invoked routine described in the ROUTINES base table.

Definition

Append the following <column definition>s

```
XML_STRING_DTD_IDENTIFIER  INFORMATION_SCHEMA.SQL_IDENTIFIER,
XML_OPTION                 INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT XML_OPTION_CHECK
    CHECK (XML_OPTION IN ( 'CONTENT', 'DOCUMENT' ) ),
XML_PASSING_MECHANISM      INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT XML_PASSING_MECHANISM_CHECK
    CHECK (XML_PASSING_MECHANISM IN
      ( 'BY REF', 'BY VALUE' ) ),
```

Append the following <table constraint definition>

```
,
CONSTRAINT PARAMETERS_CHECK_XML_DATA_TYPE
  CHECK (
    ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA,
      SPECIFIC_NAME, 'ROUTINE', XML_STRING_DTD_IDENTIFIER ) IN
      ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA,
               OBJECT_NAME, OBJECT_TYPE, DTD_IDENTIFIER
        FROM DATA_TYPE_DESCRIPTOR ) )
```

Description

1) Insert this Descr. SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME, and XML_STRING_DTD_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the associated string type of the parameter being described.

   NOTE 73 — The meaning of the term "DTD" used in "XML_STRING_DTD_IDENTIFIER" and "DTD_IDENTIFIER" is different from the meaning of the term "DTD" defined in Subclause 3.1.1, "Definitions taken from XML".

2) Insert this Descr. The value of XML_OPTION is the associated XML option of the parameter being described.

3) Insert this Descr. The values of XML_PASSING_MECHANISM have the following meanings:

null	The parameter has no <XML passing mechanism>.
BY REF	The <XML passing mechanism> of the parameter is BY REF.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

BY VALUE	The <XML passing mechanism> of the parameter is BY VALUE.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

21.3 ROUTINES base table

This Subclause modifies Subclause 6.40, "ROUTINES base table", in ISO/IEC 9075-11.

Function

The ROUTINES base table has one row for each SQL-invoked routine.

Definition

Append the following <column definition>s

```
XML_STRING_DTD_IDENTIFIER  INFORMATION_SCHEMA.SQL_IDENTIFIER,
XML_OPTION                 INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT XML_OPTION_CHECK
    CHECK (XML_OPTION IN ( 'CONTENT', 'DOCUMENT' ) ),
XML_RETURN_MECHANISM       INFORMATION_SCHEMA.CHARACTER_DATA
  CONSTRAINT XML_RETURN_MECHANISM_CHECK
    CHECK (XML_RETURN_MECHANISM IN
      ( 'BY REF', 'BY VALUE' ) ),
```

Append the following <table constraint definition>

```
,
CONSTRAINT ROUTINES_CHECK_XML_STRING_DTD_IDENTIFIER
  CHECK (
    ( SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME,
      'ROUTINE', XML_STRING_DTD_IDENTIFIER ) IN
      ( SELECT OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME,
               OBJECT_TYPE, DTD_IDENTIFIER
        FROM DATA_TYPE_DESCRIPTOR ) )
```

Description

1) Insert this Descr. SPECIFIC_CATALOG, SPECIFIC_SCHEMA, SPECIFIC_NAME, and XML_STRING_DTD_IDENTIFIER are the values of OBJECT_CATALOG, OBJECT_SCHEMA, OBJECT_NAME, and DTD_IDENTIFIER, respectively, of the row in DATA_TYPE_DESCRIPTOR that describes the associated string type of the result of SQL-invoked routine being described.

   NOTE 74 — The meaning of the term "DTD" used in "XML_STRING_DTD_IDENTIFIER" and "DTD_IDENTIFIER" is different from the meaning of the term "DTD" defined in Subclause 3.1.1, "Definitions taken from XML".

2) Insert this Descr. The value of XML_OPTION is the associated XML option of the result of SQL-invoked routine being described.

3) Insert this Descr. The values of XML_RETURN_MECHANISM have the following meanings:

null	The SQL-invoked routine does not have a <returns clause>, or its <returns clause> does not have an <XML passing mechanism>.
BY REF	The <XML passing mechanism> of the <returns clause> is BY REF.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

BY VALUE	The <XML passing mechanism> of the <returns clause> is BY VALUE.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

21.4 USAGE_PRIVILEGES base table

This Subclause modifies Subclause 6.59, "USAGE_PRIVILEGES base table", in ISO/IEC 9075-11.

Function

The USAGE_PRIVILEGES table has one row for each usage privilege descriptor. It effectively contains a representation of the usage privilege descriptors.

Definition

Augment the <in value list> in constraint USAGE_PRIVILEGES_OBJECT_TYPE_CHECK

```
, 'XML SCHEMA'
```

Augment the <table subquery> in constraint USAGE_PRIVILEGES_CHECK_REFERENCES_OBJECT

```
UNION
  SELECT XML_SCHEMA_CATALOG, XML_SCHEMA_SCHEMA, XML_SCHEMA_NAME,
         'XML SCHEMA'
  FROM XML_SCHEMAS
```

Description

1) Augment Desc. 3).

XML SCHEMA	The object to which the privilege applies is a registered XML Schema.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

21.5 XML_SCHEMA_ELEMENTS base table

Function

The XML_SCHEMA_ELEMENTS base table has one row for each global element declaration component of each registered XML Schema.

Definition

```
CREATE TABLE XML_SCHEMA_ELEMENTS (
    XML_SCHEMA_CATALOG                      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    XML_SCHEMA_SCHEMA                       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    XML_SCHEMA_NAME                         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    XML_SCHEMA_NAMESPACE                    INFORMATION_SCHEMA.URI,
    XML_SCHEMA_ELEMENT                      INFORMATION_SCHEMA.NCNAME,
    XML_SCHEMA_ELEMENT_IS_DETERMINISTIC     INFORMATION_SCHEMA.YES_OR_NO
      CONSTRAINT XML_SCHEMA_ELEMENT_IS_DETERMINISTIC_NOT_NULL
        NOT NULL,
    CONSTRAINT XML_SCHEMA_ELEMENTS_PRIMARY_KEY
      PRIMARY KEY ( XML_SCHEMA_CATALOG, XML_SCHEMA_SCHEMA,
                    XML_SCHEMA_NAME, XML_SCHEMA_NAMESPACE,
                    XML_SCHEMA_ELEMENT ),
    CONSTRAINT XML_SCHEMA_ELEMENTS_FOREIGN_KEY_XML_SCHEMA_NAMESPACES
      FOREIGN KEY ( XML_SCHEMA_CATALOG, XML_SCHEMA_SCHEMA,
                    XML_SCHEMA_NAME, XML_SCHEMA_NAMESPACE )
      REFERENCES XML_SCHEMA_NAMESPACES )
```

Description

1) The values of XML_SCHEMA_CATALOG and XML_SCHEMA_SCHEMA are the catalog name and the unqualified schema name, respectively, of the registered XML Schema containing the global element declaration schema component being described.

2) The value of XML_SCHEMA_NAME is the name of the registered XML Schema containing the global element declaration schema component being described.

3) The value of XML_SCHEMA_NAMESPACE is the namespace URI of the global element declaration schema component being described.

4) The value of XML_SCHEMA_ELEMENT is the XML QName local part of the name of the global element declaration schema component being described.

5) The values of XML_SCHEMA_ELEMENT_IS_DETERMINISTIC have the following meanings:

YES	The global element declaration schema component being described is not non-deterministic.
NO	The global element declaration schema component being described is non-deterministic.

NOTE 75 — The concept of a global element declaration schema component being non-deterministic is defined in Subclause 4.2.6, "Registered XML Schemas".

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

21.6 XML_SCHEMA_NAMESPACES base table

Function

The XML_SCHEMA_NAMESPACES base table has one row for each namespace of each registered XML Schema descriptor.

Definition

```
CREATE TABLE XML_SCHEMA_NAMESPACES (
    XML_SCHEMA_CATALOG                      INFORMATION_SCHEMA.SQL_IDENTIFIER,
    XML_SCHEMA_SCHEMA                       INFORMATION_SCHEMA.SQL_IDENTIFIER,
    XML_SCHEMA_NAME                         INFORMATION_SCHEMA.SQL_IDENTIFIER,
    XML_SCHEMA_NAMESPACE                    INFORMATION_SCHEMA.URI,
    XML_SCHEMA_NAMESPACE_IS_DETERMINISTIC   INFORMATION_SCHEMA.YES_OR_NO
    CONSTRAINT XML_SCHEMA_NAMESPACE_IS_DETERMINISTIC_NOT_NULL
      NOT NULL,

    CONSTRAINT XML_SCHEMA_NAMESPACES_PRIMARY_KEY
      PRIMARY KEY ( XML_SCHEMA_CATALOG, XML_SCHEMA_SCHEMA,
                    XML_SCHEMA_NAME, XML_SCHEMA_NAMESPACE ),

    CONSTRAINT XML_SCHEMA_NAMESPACES_FOREIGN_KEY_XML_SCHEMAS
      FOREIGN KEY ( XML_SCHEMA_CATALOG, XML_SCHEMA_SCHEMA,
                    XML_SCHEMA_NAME )
        REFERENCES XML_SCHEMAS )
```

Description

1) The values of XML_SCHEMA_CATALOG and XML_SCHEMA_SCHEMA are the catalog name and the unqualified schema name, respectively, of the registered XML Schema containing the namespace being described.

2) The value of XML_SCHEMA_NAME is the name of the registered XML Schema containing the namespace being described.

3) The value of XML_SCHEMA_NAMESPACE is the namespace URI of the namespace being described.

4) The values of XML_SCHEMA_NAMESPACE_IS_DETERMINISTIC have the following meanings:

YES	The namespace being described is not non-deterministic.
NO	The namespace being described is non-deterministic.

NOTE 76 — The concept of an XML namespace being non-deterministic is defined in Subclause 4.2.6, "Registered XML Schemas".

© ISO/IEC 2008 – All rights reserved
--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

21.7 XML_SCHEMAS base table

Function

The XML_SCHEMAS base table has one row for each registered XML Schema.

Definition

```
CREATE TABLE XML_SCHEMAS (
    XML_SCHEMA_TARGET_NAMESPACE      INFORMATION_SCHEMA.URI
      CONSTRAINT XML_SCHEMA_TARGET_NAMESPACE_NOT_NULL
        NOT NULL,
    XML_SCHEMA_LOCATION              INFORMATION_SCHEMA.URI
      CONSTRAINT XML_SCHEMA_LOCATION_NOT_NULL
        NOT NULL,
    XML_SCHEMA_CATALOG               INFORMATION_SCHEMA.SQL_IDENTIFIER,
    XML_SCHEMA_SCHEMA                INFORMATION_SCHEMA.SQL_IDENTIFIER,
    XML_SCHEMA_NAME                  INFORMATION_SCHEMA.SQL_IDENTIFIER,
    XML_SCHEMA_IS_DETERMINISTIC      INFORMATION_SCHEMA.YES_OR_NO
      CONSTRAINT XML_SCHEMA_IS_DETERMINISTIC_NOT_NULL
        NOT NULL,
    XML_SCHEMA_IS_PERMANENT          INFORMATION_SCHEMA.YES_OR_NO
      CONSTRAINT XML_SCHEMA_IS_PERMANENT_NOT_NULL
        NOT NULL,
    CONSTRAINT XML_SCHEMAS_PRIMARY_KEY
      PRIMARY KEY ( XML_SCHEMA_CATALOG, XML_SCHEMA_SCHEMA, XML_SCHEMA_NAME ),
    CONSTRAINT XML_SCHEMA_FOREIGN_KEY_XML_SCHEMA_NAMESPACES
      FOREIGN KEY ( XML_SCHEMA_CATALOG, XML_SCHEMA_SCHEMA,
                    XML_SCHEMA_NAME, XML_SCHEMA_TARGET_NAMESPACE )
        REFERENCES XML_SCHEMA_NAMESPACES,
    CONSTRAINT XML_SCHEMA_FOREIGN_KEY_XML_SCHEMATA
      FOREIGN KEY ( XML_SCHEMA_CATALOG, XML_SCHEMA_SCHEMA )
        REFERENCES SCHEMATA )
```

Description

1) The value of XML_SCHEMA_TARGET_NAMESPACE is the target namespace URI of the registered XML Schema being described.

2) The value of XML_SCHEMA_LOCATION is the schema location URI of the registered XML Schema being described.

3) The values of XML_SCHEMA_CATALOG and XML_SCHEMA_SCHEMA are the catalog name and the unqualified schema name, respectively, of the registered XML Schema being described.

4) The value of XML_SCHEMA_NAME is the name of the registered XML Schema being described.

5) The values of XML_SCHEMA_IS_DETERMINISTIC have the following meanings:

YES	The registered XML Schema being described is not non-deterministic.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

NO	The registered XML Schema being described is non-deterministic.

NOTE 77 — The concept of a registered XML Schema being non-deterministic is defined in Subclause 4.2.6, "Registered XML Schemas".

6) The values of XML_SCHEMA_IS_PERMANENT have the following meanings:

YES	The registered XML Schema being described is permanently registered.
NO	The registered XML Schema being described is not permanently registered.

NOTE 78 — The concept of a permanently registered XML Schema being non-deterministic is defined in Subclause 4.2.6, "Registered XML Schemas".

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

22 The SQL/XML XML Schema

22.1 The SQL/XML XML Schema

Function

Define the contents of the XML Schema for SQL/XML.

Syntax Rules

1) The contents of the SQL/XML XML Schema are:

```
<?xml version="1.0"?>
<xs:schema
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      targetNamespace="http://standards.iso.org/iso/9075/2003/sqlxml"
      xmlns:sqlxml="http://standards.iso.org/iso/9075/2003/sqlxml">
  <xs:annotation>
    <xs:documentation>
      This document contains definitions of types and
      annotations as specified in ISO/IEC 9075-14.
    </xs:documentation>
  </xs:annotation>
  <xs:simpleType name="kindKeyword">
    <xs:restriction base="xs:string">
      <xs:enumeration value="PREDEFINED"/>
      <xs:enumeration value="DOMAIN"/>
      <xs:enumeration value="ROW"/>
      <xs:enumeration value="DISTINCT"/>
      <xs:enumeration value="ARRAY"/>
      <xs:enumeration value="MULTISET"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="typeKeyword">
    <xs:restriction base="xs:string">
      <xs:enumeration value="CHAR"/>
      <xs:enumeration value="VARCHAR"/>
      <xs:enumeration value="CLOB"/>
      <xs:enumeration value="BLOB"/>
      <xs:enumeration value="NUMERIC"/>
      <xs:enumeration value="DECIMAL"/>
      <xs:enumeration value="INTEGER"/>
      <xs:enumeration value="SMALLINT"/>
      <xs:enumeration value="BIGINT"/>
      <xs:enumeration value="FLOAT"/>
      <xs:enumeration value="REAL"/>
      <xs:enumeration value="DOUBLE PRECISION"/>
      <xs:enumeration value="BOOLEAN"/>
```

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
      <xs:enumeration value="DATE"/>
      <xs:enumeration value="TIME"/>
      <xs:enumeration value="TIME WITH TIME ZONE"/>
      <xs:enumeration value="TIMESTAMP"/>
      <xs:enumeration value="TIMESTAMP WITH TIME ZONE"/>
      <xs:enumeration value="INTERVAL YEAR"/>
      <xs:enumeration value="INTERVAL YEAR TO MONTH"/>
      <xs:enumeration value="INTERVAL MONTH"/>
      <xs:enumeration value="INTERVAL DAY"/>
      <xs:enumeration value="INTERVAL DAY TO HOUR"/>
      <xs:enumeration value="INTERVAL DAY TO MINUTE"/>
      <xs:enumeration value="INTERVAL DAY TO SECOND"/>
      <xs:enumeration value="INTERVAL HOUR"/>
      <xs:enumeration value="INTERVAL HOUR TO MINUTE"/>
      <xs:enumeration value="INTERVAL HOUR TO SECOND"/>
      <xs:enumeration value="INTERVAL MINUTE"/>
      <xs:enumeration value="INTERVAL MINUTE TO SECOND"/>
      <xs:enumeration value="INTERVAL SECOND"/>
      <xs:enumeration value="XML"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="fieldType">
    <xs:attribute name="name" type="xs:string"/>
    <xs:attribute name="mappedType" type="xs:string"/>
  </xs:complexType>
  <xs:element name="sqltype">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="field" type="sqlxml:fieldType"
                    minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="kind"
                    type="sqlxml:kindKeyword"/>
      <xs:attribute name="name"
                    type="sqlxml:typeKeyword" use="optional"/>
      <xs:attribute name="length" type="xs:integer"
                    use="optional"/>
      <xs:attribute name="maxLength" type="xs:integer"
                    use="optional"/>
      <xs:attribute name="characterSetName" type="xs:string"
                    use="optional"/>
      <xs:attribute name="collation" type="xs:string"
                    use="optional"/>
      <xs:attribute name="precision" type="xs:integer"
                    use="optional"/>
      <xs:attribute name="scale" type="xs:integer"
                    use="optional"/>
      <xs:attribute name="maxExponent" type="xs:integer"
                    use="optional"/>
      <xs:attribute name="minExponent" type="xs:integer"
                    use="optional"/>
      <xs:attribute name="userPrecision" type="xs:integer"
                    use="optional"/>
      <xs:attribute name="leadingPrecision" type="xs:integer"
                    use="optional"/>
      <xs:attribute name="maxElements" type="xs:integer"
                    use="optional"/>
```

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

```
      <xs:attribute name="catalogName" type="xs:string"
                    use="optional"/>
      <xs:attribute name="schemaName" type="xs:string"
                    use="optional"/>
      <xs:attribute name="domainName" type="xs:string"
                    use="optional"/>
      <xs:attribute name="typeName" type="xs:string"
                    use="optional"/>
      <xs:attribute name="mappedType" type="xs:string"
                    use="optional"/>
      <xs:attribute name="mappedElementType" type="xs:string"
                    use="optional"/>
      <xs:attribute name="final" type="xs:boolean"
                    use="optional"/>
    </xs:complexType>
  </xs:element>
  <xs:simpleType name="objectType">
     <xs:restriction base="xs:string">
        <xs:enumeration value="CATALOG" />
        <xs:enumeration value="SCHEMA" />
        <xs:enumeration value="BASE TABLE" />
        <xs:enumeration value="VIEWED TABLE" />
        <xs:enumeration value="CHARACTER SET" />
        <xs:enumeration value="COLLATION" />
     </xs:restriction>
  </xs:simpleType>
  <xs:element name="sqlname">
     <xs:complexType>
        <xs:attribute name="type" type="sqlxml:objectType"
                      use="required" />
        <xs:attribute name="catalogName" type="xs:string" />
        <xs:attribute name="schemaName" type="xs:string" />
        <xs:attribute name="localName" type="xs:string" />
     </xs:complexType>
  </xs:element>
</xs:schema>
```

General Rules

None.

Conformance Rules

None.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

(Blank page)

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

23 Status codes

This Clause modifies Clause 24, "Status codes", in ISO/IEC 9075-2.

23.1 SQLSTATE

This Subclause modifies Subclause 24.1, "SQLSTATE", in ISO/IEC 9075-2.

Augment Table 33, "SQLSTATE class and subclass values"

Table 14 — SQLSTATE class and subclass values

Category	Condition	Class	Subcondition	Subclass
X	*data exception*	22	(*no subclass*)	000
			invalid comment	00S
			invalid processing instruction	00T
			invalid XML content	00N
			invalid XML document	00M
			nonidentical notations with the same name	00J
			nonidentical unparsed entities with the same name	00K
			not an XML document	00L
			XML value overflow	00R
			not an XQuery document node	00U
			invalid XQuery context item	00V
			XQuery serialization error	00W
			XQuery sequence cannot be validated	01J

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Category	Condition	Class	Subcondition	Subclass
			XQuery document node cannot be validated	01K
			no XML schema found	01L
			element namespace not declared	01M
			global element not declared	01N
			no XML element with the specified QName	01P
			no XML element with the specified namespace	01Q
			validation failure	01R
X	*SQL/XML mapping error*	0N	(*no subclass*)	000
			unmappable XML Name	001
			invalid XML character	002
X	*XQuery error*	10	(*no subclass*)	000
W	*warning*	01	(*no subclass*)	000
			column cannot be mapped	010

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

24 Conformance

24.1 Claims of conformance to SQL/XML

In addition to the requirements of ISO/IEC 9075-1, Clause 8, "Conformance", a claim of conformance to this part of ISO/IEC 9075 shall:

1) Claim conformance to Feature X010, "XML type".
2) Claim conformance to Feature X016, "Persistent XML values".
3) Claim conformance to Feature X031, "XMLElement".
4) Claim conformance to Feature X032, "XMLForest".
5) Claim conformance to Feature X035, "XMLAgg: ORDER BY option".
6) Claim conformance to Feature X036, "XMLComment".
7) Claim conformance to Feature X037, "XMLPI".
8) Claim conformance to at least one of the following:
   a) Claim conformance to all of the following:
      i) Claim conformance to at least one of the following:
         1) Feature X070, "XMLSerialize: Character string serialization and CONTENT option".
         2) Feature X071, "XMLSerialize: Character string serialization and DOCUMENT option".
      ii) Claim conformance to at least one of the following:
         1) Feature X060, "XMLParse: Character string input and CONTENT option".
         2) Feature X061, "XMLParse: Character string input and DOCUMENT option".
   b) Claim conformance to all of the following:
      i) Claim conformance to at least one of the following:
         1) Feature X073, "XMLSerialize: BLOB serialization and CONTENT option".
         2) Feature X074, "XMLSerialize: BLOB serialization and DOCUMENT option".
      ii) Claim conformance to at least one of the following:
         1) Feature X065, "XMLParse: BLOB input and CONTENT option".
         2) Feature X066, "XMLParse: BLOB input and DOCUMENT option".
   c) Feature X100, "Host language support for XML: CONTENT option", and Feature X110, "Host language support for XML: VARCHAR mapping".

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

d) Feature X100, “Host language support for XML: CONTENT option”, and Feature X111, “Host language support for XML: CLOB mapping”.

e) Feature X101, “Host language support for XML: DOCUMENT option”, and Feature X110, “Host language support for XML: VARCHAR mapping”.

f) Feature X101, “Host language support for XML: DOCUMENT option”, and Feature X111, “Host language support for XML: CLOB mapping”.

9) Claim conformance to at least one of the following:

a) Feature X080, “Namespaces in XML publishing”.

b) Feature X081, “Query-level XML namespace declarations”.

24.2 Additional conformance requirements for SQL/XML

Each claim of conformance to Feature X191, “XML(DOCUMENT(XMLSCHEMA))” type or to Feature X192, “XML(CONTENT(XMLSCHEMA))” shall also state whether conformance to one or more of the following is claimed:

1) Feature X260, “XML type: ELEMENT clause”

2) Feature X261, “XML type: NAMESPACE without ELEMENT clause”

3) Feature X263, “XML type: NO NAMESPACE with ELEMENT clause”

4) Feature X264, “XML type: schema location”

Each claim of conformance to Feature X040, “Basic table mapping”, shall also claim conformance to at least one of:

1) Feature X041, “Basic table mapping: null absent”

2) Feature X042, “Basic table mapping: null as nil”

Each claim of conformance to Feature X040, “Basic table mapping”, shall also claim conformance to at least one of:

1) Feature X043, “Basic table mapping: table as forest”

2) Feature X044, “Basic table mapping: table as element”

Each claim of conformance to Feature X040, “Basic table mapping”, shall also claim conformance to at least one of:

1) Feature X046, “Basic table mapping: data mapping”

2) Feature X047, “Basic table mapping: metadata mapping”

Each claim of conformance to Feature X040, “Basic table mapping”, shall also claim conformance to at least one of:

1) Feature X048, “Basic table mapping: base64 encoding of binary strings”

2) Feature X049, “Basic table mapping: hex encoding of binary strings”

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Each claim of conformance to Feature X050, “Advanced table mapping”, shall also claim conformance to at least one of:

1) Feature X051, “Advanced table mapping: null absent”

2) Feature X052, “Advanced table mapping: null as nil”

Each claim of conformance to Feature X050, “Advanced table mapping”, shall also claim conformance to at least one of:

1) Feature X053, “Advanced table mapping: table as forest”

2) Feature X054, “Advanced table mapping: table as element”

Each claim of conformance to Feature X050, “Advanced table mapping”, shall also claim conformance to at least one of:

1) Feature X056, “Advanced table mapping: data mapping”

2) Feature X057, “Advanced table mapping: metadata mapping”

Each claim of conformance to Feature X050, “Advanced table mapping”, shall also claim conformance to at least one of:

1) Feature X058, “Advanced table mapping: base64 encoding of binary strings”

2) Feature X059, “Advanced table mapping: hex encoding of binary strings”

24.3 Implied feature relationships of SQL/XML

Table 15 — Implied feature relationships of SQL/XML

Feature ID	Feature Name	Implied Feature ID	Implied Feature Name
X011	Arrays of XML type	X010	XML type
X011	Arrays of XML type	S091	Basic array support
X012	Multisets of XML type	X010	XML type
X012	Multisets of XML type	S271	Basic multiset support
X013	Distinct types of XML type	X010	XML type
X014	Attributes of XML type	X010	XML type
X014	Attributes of XML type	S023	Basic structured types
X015	Fields of XML type	X010	XML type

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Feature ID	Feature Name	Implied Feature ID	Implied Feature Name
X015	Fields of XML type	T051	Row types
X016	Persistent XML values	X010	XML type
X020	XMLConcat	X010	XML type
X025	XMLCast	X010	XML type
X026	XMLCast: SQL cast semantics	X025	XMLCast
X030	XMLDocument	X010	XML type
X031	XMLElement	X010	XML type
X032	XMLForest	X010	XML type
X034	XMLAgg	X010	XML type
X035	XMLAgg: ORDER BY option	X034	XMLAgg
X036	XMLComment	X010	XML type
X037	XMLPI	X010	XML type
X038	XMLText	X010	XML type
X041	Basic table mapping: null absent	X040	Basic table mapping
X042	Basic table mapping: null as nil	X040	Basic table mapping
X043	Basic table mapping: table as forest	X040	Basic table mapping
X044	Basic table mapping: table as element	X040	Basic table mapping
X045	Basic table mapping: with target namespace	X040	Basic table mapping
X046	Basic table mapping: data mapping	X040	Basic table mapping
X047	Basic table mapping: metadata mapping	X040	Basic table mapping
X048	Basic table mapping: base64 encoding of binary strings	X040	Basic table mapping
X049	Basic table mapping: hex encoding of binary strings	X040	Basic table mapping

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Feature ID	Feature Name	Implied Feature ID	Implied Feature Name
X051	Advanced table mapping: null absent	X041	Basic table mapping: null absent
X051	Advanced table mapping: null absent	X050	Advanced table mapping
X052	Advanced table mapping: null as nil	X042	Basic table mapping: null as nil
X052	Advanced table mapping: null as nil	X050	Advanced table mapping
X053	Advanced table mapping: table as forest	X043	Basic table mapping: table as forest
X053	Advanced table mapping: table as forest	X050	Advanced table mapping
X054	Advanced table mapping: table as element	X044	Basic table mapping: table as element
X054	Advanced table mapping: table as element	X050	Advanced table mapping
X055	Advanced table mapping: with target namespace	X045	Basic table mapping: with target namespace
X055	Advanced table mapping: with target namespace	X050	Advanced table mapping
X056	Advanced table mapping: data mapping	X046	Basic table mapping: data mapping
X056	Advanced table mapping: data mapping	X050	Advanced table mapping
X057	Advanced table mapping: metadata mapping	X047	Basic table mapping: metadata mapping
X057	Advanced table mapping: metadata mapping	X050	Advanced table mapping
X058	Advanced table mapping: base64 encoding of binary strings	X050	Advanced table mapping
X059	Advanced table mapping: hex encoding of binary strings	X050	Advanced table mapping
X060	XMLParse: Character string input and CONTENT option	X010	XML type

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Feature ID	Feature Name	Implied Feature ID	Implied Feature Name
X061	XMLParse: Character string input and DOCUMENT option	X010	XML type
X065	XMLParse: BLOB input and CONTENT option	X010	XML type
X066	XMLParse: BLOB input and DOCUMENT option	X010	XML type
X068	XMLSerialize: BOM	X010	XML type
X069	XMLSerialize: INDENT	X010	XML type
X070	XMLSerialize: Character string serialization and CONTENT option	X010	XML type
X071	XMLSerialize: Character string serialization and DOCUMENT option	X010	XML type
X073	XMLSerialize: BLOB serialization and CONTENT option	X010	XML type
X074	XMLSerialize: BLOB serialization and DOCUMENT option	X010	XML type
X076	XMLSerialize: VERSION option	X010	XML type
X077	XMLSerialize: Explicit ENCODING option	X075	XMLSerialize: BLOB serialization
X080	Namespaces in XML publishing	X010	XML type
X081	Query-level XML namespace declarations	X010	XML type
X082	XML namespace declarations in DML	X010	XML type
X083	XML namespace declarations in DDL	X010	XML type
X084	XML namespace declarations in compound statements	X010	XML type
X085	Predefined namespace prefixes	X080	Namespaces in XML publishing
X086	XML namespace declarations in XMLTable	X300	XMLTable
X090	XML document predicate	X010	XML type

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Feature ID	Feature Name	Implied Feature ID	Implied Feature Name
X091	XML content predicate	X10	XML type
X096	XMLExists	X010	XML type
X100	Host language support for XML: CONTENT option	X010	XML type
X101	Host language support for XML: DOCUMENT option	X010	XML type
X110	Host language support for XML: VARCHAR mapping	X010	XML type
X111	Host language support for XML: CLOB mapping	X010	XML type
X111	Host language support for XML: CLOB mapping	T041	Basic LOB data type support
X112	Host language support for XML: BLOB mapping	X010	XML type
X112	Host language support for XML: BLOB mapping	T041	Basic LOB data type support
X113	Host language support for XML: STRIP WHITESPACE	X010	XML type
X114	Host language support for XML: PRESERVE WHITESPACE	X010	XML type
X120	XML parameters in SQL routines	X010	XML type
X121	XML parameters in external routines	X010	XML type
X131	Query-level XMLBINARY clause	X010	XML type
X132	XMLBINARY clause in DML	X010	XML type
X133	XMLBINARY clause in DDL	X010	XML type
X134	XMLBINARY clause in compound statements	X010	XML type
X135	XMLBINARY clause in subqueries	X131	Query-level XMLBINARY clause
X141	IS VALID predicate: data-driven case	X145	IS VALID predicate outside check constraints

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Feature ID	Feature Name	Implied Feature ID	Implied Feature Name
X142	IS VALID predicate: ACCORDING TO clause	X010	XML type
X143	IS VALID predicate: ELEMENT clause	X142	IS VALID predicate: ACCORDING TO clause
X144	IS VALID predicate: schema location	X142	IS VALID predicate: ACCORDING TO clause
X145	IS VALID predicate outside check constraints	X010	XML type
X151	IS VALID predicate with DOCUMENT option	X010	XML type
X152	IS VALID predicate with CONTENT option	X010	XML type
X153	IS VALID predicate with SEQUENCE option	X010	XML type
X155	IS VALID predicate: NAMESPACE without ELEMENT clause	X142	IS VALID predicate: ACCORDING TO clause
X157	IS VALID predicate: NO NAMESPACE with ELEMENT clause	X142	IS VALID predicate: ACCORDING TO clause
X160	Basic Information Schema for registered XML Schemas	X010	XML type
X161	Advanced Information Schema for registered XML Schemas	X160	Basic Information Schema for registered XML Schemas
X171	NIL ON NO CONTENT option	X170	XML null handling options
X181	XML(DOCUMENT(UNTYPED)) type	X010	XML type
X182	XML(DOCUMENT(ANY)) type	X010	XML type
X190	XML(SEQUENCE) type	X010	XML type
X191	XML(DOCUMENT(XMLSCHEMA)) type	X010	XML type
X192	XML(CONTENT(XMLSCHEMA)) type	X010	XML type

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Feature ID	Feature Name	Implied Feature ID	Implied Feature Name
X200	XMLQuery	X010	XML type
X201	XMLQuery: RETURNING CONTENT	X200	XMLQuery
X202	XMLQuery: RETURNING SEQUENCE	X200	XMLQuery
X203	XMLQuery: passing a context item	X200	XMLQuery
X204	XMLQuery: initializing an XQuery variable	X200	XMLQuery
X205	XMLQuery: EMPTY ON EMPTY option	X200	XMLQuery
X206	XMLQuery: NULL ON EMPTY option	X200	XMLQuery
X211	XML 1.1 support	X010	XML type
X221	XML passing mechanism BY VALUE	X010	XML type
X222	XML passing mechanism BY REF	X010	XML type
X231	XML(CONTENT(UNTYPED)) type	X010	XML type
X232	XML(CONTENT(ANY)) type	X010	XML type
X241	RETURNING CONTENT in XML publishing	X010	XML type
X242	RETURNING SEQUENCE in XML publishing	X10	XML type
X251	Persistent XML values of XML(DOCUMENT(UNTYPED)) type	X181	XML(DOCUMENT(UNTYPED)) type
X251	Persistent XML values of XML(DOCUMENT(UNTYPED)) type	X016	Persistent XML values
X252	Persistent XML values of XML(DOCUMENT(ANY)) type	X182	XML(DOCUMENT(ANY)) type

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Feature ID	Feature Name	Implied Feature ID	Implied Feature Name
X252	Persistent XML values of XML(DOCUMENT(ANY)) type	X016	Persistent XML values
X253	Persistent XML values of XML(CONTENT(UNTYPED)) type	X231	XML(CONTENT(UNTYPED)) type
X253	Persistent XML values of XML(CONTENT(UNTYPED)) type	X016	Persistent XML values
X254	Persistent XML values of XML(CONTENT(ANY)) type	X232	XML(CONTENT(ANY)) type
X254	Persistent XML values of XML(CONTENT(ANY)) type	X016	Persistent XML values
X255	Persistent XML values of XML(SEQUENCE) type	X190	XML(SEQUENCE) type
X255	Persistent XML values of XML(SEQUENCE) type	X016	Persistent XML values
X256	Persistent XML values of XML(DOCU-MENT(XMLSCHEMA)) type	X191	XML(DOCUMENT(XMLSCHEMA)) type
X256	Persistent XML values of XML(DOCU-MENT(XMLSCHEMA)) type	X016	Persistent XML values
X257	Persistent XML values of XML(CONTENT(XMLSCHEMA)) type	X192	XML(CONTENT(XMLSCHEMA)) type
X257	Persistent XML values of XML(CONTENT(XMLSCHEMA)) type	X016	Persistent XML values
X271	XMLValidate: data driven case	X010	XML type
X272	XMLValidate: ACCORDING TO clause	X010	XML type
X273	XMLValidate: ELEMENT clause	X272	XMLValidate: ACCORDING TO clause
X281	XMLValidate with DOCUMENT option	X010	XML type

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Feature ID	Feature Name	Implied Feature ID	Implied Feature Name
X282	XMLValidate with CONTENT option	X010	XML type
X283	XMLValidate with SEQUENCE option	X010	XML type
X284	XMLValidate: NAMESPACE without ELEMENT clause	X272	XMLValidate: ACCORDING TO clause
X286	XMLValidate: NO NAMESPACE with ELEMENT clause	X272	XMLValidate: ACCORDING TO clause
X287	XMLValidate: schema location	X272	XMLValidate: ACCORDING TO clause
X300	XMLTable	X010	XML type
X301	XMLTable: derived column list option	X300	XMLTable
X302	XMLTable: ordinality column option	X300	XMLTable
X303	XMLTable: column default option	X300	XMLTable
X304	XMLTable: passing a context item	X300	XMLTable
X305	XMLTable: initializing an XQuery variable	X300	XMLTable

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

(Blank page)

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex A
(informative)

SQL Conformance Summary

This Annex modifies Annex A, "SQL Conformance Summary", in ISO/IEC 9075-2.

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

1) Specifications for Feature F251, "Domain support":
   a) Subclause 20.1, "NCNAME domain":
      i) Without Feature F251, "Domain support", conforming SQL language shall not reference INFORMATION_SCHEMA.NCNAME.
   b) Subclause 20.2, "URI domain":
      i) Without Feature F251, "Domain support", conforming SQL language shall not reference INFORMATION_SCHEMA.URI.
2) Specifications for Feature F391, "Long identifiers":
   a) Subclause 20.12, "XML_SCHEMA_ELEMENTS view":
      i) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMATION_SCHEMA.XML_SCHEMA_ELEMENTS.
   b) Subclause 20.13, "XML_SCHEMA_NAMESPACES view":
      i) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMATION_SCHEMA.XML_SCHEMA_NAMESPACES.
   c) Subclause 20.14, "XML_SCHEMAS view":
      i) Without Feature F391, "Long identifiers", conforming SQL language shall not reference INFORMATION_SCHEMA.XML_SCHEMAS.
3) Specifications for Feature F761, "Session management":
   a) Subclause 16.1, "<set XML option statement>":
      i) Without Feature F761, "Session management", conforming SQL language shall not contain a <set XML option statement>.
4) Specifications for Feature X010, "XML type":
   a) Subclause 6.1, "<data type>":
      i) Insert this CR Without Feature X010, "XML type", conforming SQL language shall not contain an <XML type>.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

b) Subclause 6.9, "<XML value expression>":

i) Without Feature X010, "XML type", conforming SQL language shall not contain an <XML value expression>.

c) Subclause 6.10, "<XML value function>":

i) Without Feature X010, "XML type", conforming SQL language shall not contain an <XML value function>.

5) Specifications for Feature X011, "Arrays of XML type":

a) Subclause 6.1, "<data type>":

i) Insert this CR Without Feature X011, "Arrays of XML type", conforming SQL language shall not contain an <array type> that is based on a <data type> that is either an XML type or a distinct type whose source type is an XML type.

6) Specifications for Feature X012, "Multisets of XML type":

a) Subclause 6.1, "<data type>":

i) Insert this CR Without Feature X012, "Multisets of XML type", conforming SQL language shall not contain a <multiset type> that is based on a <data type> that is either an XML type or a distinct type whose source type is an XML type.

7) Specifications for Feature X013, "Distinct types of XML type":

a) Subclause 12.5, "<user-defined type definition>":

i) Insert this CR Without Feature X013, "Distinct types of XML type", conforming SQL language shall not contain a <representation> that is a <predefined type> that is an XML type.

8) Specifications for Feature X014, "Attributes of XML type":

a) Subclause 12.6, "<attribute definition>":

i) Insert this CR Without Feature X014, "Attributes of XML type", conforming SQL language shall not contain an <attribute definition> that contains a <data type> that is based on either an XML type or a distinct type whose source type is an XML type.

9) Specifications for Feature X015, "Fields of XML type":

a) Subclause 6.2, "<field definition>":

i) Insert this CR Without Feature X015, "Fields of XML type", conforming SQL language shall not contain a <field definition> that contains a <data type> that is based on either an XML type or a distinct type whose source type is an XML type.

10) Specifications for Feature X016, "Persistent XML values":

a) Subclause 12.1, "<column definition>":

i) Insert this CR Without Feature X016, "Persistent XML values", conforming SQL language shall not contain a <column definition> whose declared type is based on either an XML type or a distinct type whose source type is an XML type.

b) Subclause 12.3, "<view definition>":

© ISO/IEC 2008 – All rights reserved

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

i) Insert this CR Without Feature X016, “Persistent XML values”, conforming SQL language shall not contain a <view definition> that defines a column whose declared type is based on either an XML type or a distinct type whose source type is an XML type.

11) Specifications for Feature X020, “XMLConcat”:

a) Subclause 6.12, “<XML concatenation>”:

i) Without Feature X020, “XMLConcat”, conforming SQL language shall not contain an <XML concatenation>.

12) Specifications for Feature X025, “XMLCast”:

a) Subclause 6.6, “<XML cast specification>”:

i) Without Feature X025, “XMLCast”, conforming SQL language shall not contain an <XML cast specification>.

13) Specifications for Feature X026, “XMLCast: SQL cast semantics”:

a) Subclause 6.6, “<XML cast specification>”:

i) Without Feature X026, “XMLCast: SQL cast semantics”, *XT* is unconditionally the XML Schema type obtained by applying the General Rules of Subclause 9.5, “Mapping SQL data types to XML Schema data types”, with *SQLT* as *SQLTYPE*, *ENC* as *ENCODING*, and “absent” as *NULLS*.

14) Specifications for Feature X030, “XMLDocument”:

a) Subclause 6.13, “<XML document>”:

i) Without Feature X030, “XMLDocument”, conforming SQL language shall not contain an <XML document>.

15) Specifications for Feature X031, “XMLElement”:

a) Subclause 6.14, “<XML element>”:

i) Without Feature X031, “XMLElement”, conforming SQL language shall not contain an <XML element>.

16) Specifications for Feature X032, “XMLForest”:

a) Subclause 6.15, “<XML forest>”:

i) Without Feature X032, “XMLForest”, conforming SQL language shall not contain an <XML forest>.

17) Specifications for Feature X034, “XMLAgg”:

a) Subclause 11.2, “<aggregate function>”:

i) Insert this CR Without Feature X034, “XMLAgg”, conforming SQL language shall not contain an <XML aggregate>.

18) Specifications for Feature X035, “XMLAgg: ORDER BY option”:

a) Subclause 11.2, “<aggregate function>”:

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

i) Insert this CR Without Feature X035, "XMLAgg: ORDER BY option", conforming SQL language shall not contain an <XML aggregate> that contains a <sort specification list>.

19) Specifications for Feature X036, "XMLComment":

a) Subclause 6.11, "<XML comment>":

i) Without Feature X036, "XMLComment", conforming SQL language shall not contain an <XML comment>.

20) Specifications for Feature X037, "XMLPI":

a) Subclause 6.17, "<XML PI>":

i) Without Feature X037, "XMLPI", conforming SQL language shall not contain an <XML PI>.

21) Specifications for Feature X038, "XMLText":

a) Subclause 6.19, "<XML text>":

i) Without Feature X038, "XMLText", conforming SQL language shall not contain an <XML text>.

22) Specifications for Feature X040, "Basic table mapping":

a) Subclause 9.11, "Mapping an SQL table to XML and an XML Schema document":

i) Without Feature X040, "Basic table mapping", a conforming application shall not invoke this Subclause of this part of this International Standard.

23) Specifications for Feature X041, "Basic table mapping: nulls absent":

a) Subclause 9.11, "Mapping an SQL table to XML and an XML Schema document":

i) Without Feature X041, "Basic table mapping: nulls absent", a conforming application shall not invoke this Subclause of this part of this International Standard with *NULLS* set to indicate that nulls are mapped to elements that are marked to absent elements.

24) Specifications for Feature X042, "Basic table mapping: null as nil":

a) Subclause 9.11, "Mapping an SQL table to XML and an XML Schema document":

i) Without Feature X042, "Basic table mapping: null as nil", a conforming application shall not invoke this Subclause of this part of this International Standard with *NULLS* set to indicate that nulls are mapped to elements that are marked with **`xsi:nil="true"`**.

25) Specifications for Feature X043, "Basic table mapping: table as forest":

a) Subclause 9.11, "Mapping an SQL table to XML and an XML Schema document":

i) Without Feature X043, "Basic table mapping: table as forest", a conforming application shall not invoke this Subclause of this part of this International Standard with *TABLEFOREST* set to *True*.

26) Specifications for Feature X044, "Basic table mapping: table as element":

a) Subclause 9.11, "Mapping an SQL table to XML and an XML Schema document":

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

i) Without Feature X044, "Basic table mapping: table as element", a conforming application shall not invoke this Subclause of this part of this International Standard with *TABLEFOREST* set to *False*.

27) Specifications for Feature X045, "Basic table mapping: with target namespace":

a) Subclause 9.11, "Mapping an SQL table to XML and an XML Schema document":

i) Without Feature X045, "Basic table mapping: with target namespace", a conforming application shall not invoke this Subclause of this part of this International Standard with *TARGETNS* that is not a zero-length string.

28) Specifications for Feature X046, "Basic table mapping: data mapping":

a) Subclause 9.11, "Mapping an SQL table to XML and an XML Schema document":

i) Without Feature X046, "Basic table mapping: data mapping", a conforming application shall not invoke this Subclause of this part of this International Standard with *DATA* set to *True*.

29) Specifications for Feature X047, "Basic table mapping: metadata mapping":

a) Subclause 9.11, "Mapping an SQL table to XML and an XML Schema document":

i) Without Feature X047, "Basic table mapping: metadata mapping", a conforming application shall not invoke this Subclause of this part of this International Standard with *METADATA* set to *True*.

30) Specifications for Feature X048, "Basic table mapping: base64 encoding of binary strings":

a) Subclause 9.11, "Mapping an SQL table to XML and an XML Schema document":

i) Without Feature X048, "Basic table mapping: base64 encoding of binary strings", a conforming application shall not invoke this Subclause of this part of this International Standard with *ENCODING* set to indicate that binary strings are to be encoded using base64.

31) Specifications for Feature X049, "Basic table mapping: hex encoding of binary strings":

a) Subclause 9.11, "Mapping an SQL table to XML and an XML Schema document":

i) Without Feature X049, "Basic table mapping: hex encoding of binary strings", a conforming application shall not invoke this Subclause of this part of this International Standard with *ENCODING* set to indicate that binary strings are to be encoded using hex.

32) Specifications for Feature X050, "Advanced table mapping":

a) Subclause 9.14, "Mapping an SQL schema to an XML document and an XML Schema document":

i) Without Feature X050, "Advanced table mapping", a conforming application shall not invoke this Subclause of this part of this International Standard.

b) Subclause 9.17, "Mapping an SQL catalog to an XML document and an XML Schema document":

i) Without Feature X050, "Advanced table mapping", a conforming application shall not invoke this Subclause of this part of this International Standard.

33) Specifications for Feature X051, "Advanced table mapping: nulls absent":

a) Subclause 9.14, "Mapping an SQL schema to an XML document and an XML Schema document":

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

i) Without Feature X051, “Advanced table mapping: nulls absent”, a conforming application shall not invoke this Subclause of this part of this International Standard with *NULLS* set to indicate that nulls are mapped to elements that are marked to absent elements.

b) Subclause 9.17, “Mapping an SQL catalog to an XML document and an XML Schema document”:

i) Without Feature X051, “Advanced table mapping: nulls absent”, a conforming application shall not invoke this Subclause of this part of this International Standard with *NULLS* set to indicate that nulls are mapped to elements that are marked to absent elements.

34) Specifications for Feature X052, “Advanced table mapping: null as nil”:

a) Subclause 9.14, “Mapping an SQL schema to an XML document and an XML Schema document”:

i) Without Feature X052, “Advanced table mapping: null as nil”, a conforming application shall not invoke this Subclause of this part of this International Standard with *NULLS* set to indicate that nulls are mapped to elements that are marked with **`xsi:nil="true"`**.

b) Subclause 9.17, “Mapping an SQL catalog to an XML document and an XML Schema document”:

i) Without Feature X052, “Advanced table mapping: null as nil”, a conforming application shall not invoke this Subclause of this part of this International Standard with *NULLS* set to indicate that nulls are mapped to elements that are marked with **`xsi:nil="true"`**.

35) Specifications for Feature X053, “Advanced table mapping: table as forest”:

a) Subclause 9.14, “Mapping an SQL schema to an XML document and an XML Schema document”:

i) Without Feature X053, “Advanced table mapping: table as forest”, a conforming application shall not invoke this Subclause of this part of this International Standard with *TABLEFOREST* set to *True*.

b) Subclause 9.17, “Mapping an SQL catalog to an XML document and an XML Schema document”:

i) Without Feature X053, “Advanced table mapping: table as forest”, a conforming application shall not invoke this Subclause of this part of this International Standard with *TABLEFOREST* set to *True*.

36) Specifications for Feature X054, “Advanced table mapping: table as element”:

a) Subclause 9.14, “Mapping an SQL schema to an XML document and an XML Schema document”:

i) Without Feature X054, “Advanced table mapping: table as element”, a conforming application shall not invoke this Subclause of this part of this International Standard with *TABLEFOREST* set to *False*.

b) Subclause 9.17, “Mapping an SQL catalog to an XML document and an XML Schema document”:

i) Without Feature X054, “Advanced table mapping: table as element”, a conforming application shall not invoke this Subclause of this part of this International Standard with *TABLEFOREST* set to *False*.

37) Specifications for Feature X055, “Advanced table mapping: with target namespace”:

a) Subclause 9.14, “Mapping an SQL schema to an XML document and an XML Schema document”:

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

i) Without Feature X055, “Advanced table mapping: with target namespace”, a conforming application shall not invoke this Subclause of this part of this International Standard with *TARGETNS* that is not a zero-length string.

b) Subclause 9.17, “Mapping an SQL catalog to an XML document and an XML Schema document”:

i) Without Feature X055, “Advanced table mapping: with target namespace”, a conforming application shall not invoke this Subclause of this part of this International Standard with *TARGETNS* that is not a zero-length string.

38) Specifications for Feature X056, “Advanced table mapping: data mapping”:

a) Subclause 9.14, “Mapping an SQL schema to an XML document and an XML Schema document”:

i) Without Feature X056, “Advanced table mapping: data mapping”, a conforming application shall not invoke this Subclause of this part of this International Standard with *DATA* set to *True*.

b) Subclause 9.17, “Mapping an SQL catalog to an XML document and an XML Schema document”:

i) Without Feature X056, “Advanced table mapping: data mapping”, a conforming application shall not invoke this Subclause of this part of this International Standard with *DATA* set to *True*.

39) Specifications for Feature X057, “Advanced table mapping: metadata mapping”:

a) Subclause 9.14, “Mapping an SQL schema to an XML document and an XML Schema document”:

i) Without Feature X057, “Advanced table mapping: metadata mapping”, a conforming application shall not invoke this Subclause of this part of this International Standard with *METADATA* set to *True*.

b) Subclause 9.17, “Mapping an SQL catalog to an XML document and an XML Schema document”:

i) Without Feature X057, “Advanced table mapping: metadata mapping”, a conforming application shall not invoke this Subclause of this part of this International Standard with *METADATA* set to *True*.

40) Specifications for Feature X058, “Advanced table mapping: base64 encoding of binary strings”:

a) Subclause 9.14, “Mapping an SQL schema to an XML document and an XML Schema document”:

i) Without Feature X058, “Advanced table mapping: base64 encoding of binary strings”, a conforming application shall not invoke this Subclause of this part of this International Standard with *ENCODING* set to indicate that binary strings are to be encoded using base64.

b) Subclause 9.17, “Mapping an SQL catalog to an XML document and an XML Schema document”:

i) Without Feature X058, “Advanced table mapping: base64 encoding of binary strings”, a conforming application shall not invoke this Subclause of this part of this International Standard with *ENCODING* set to indicate that binary strings are to be encoded using base64.

41) Specifications for Feature X059, “Advanced table mapping: hex encoding of binary strings”:

a) Subclause 9.14, “Mapping an SQL schema to an XML document and an XML Schema document”:

i) Without Feature X059, “Advanced table mapping: hex encoding of binary strings”, a conforming application shall not invoke this Subclause of this part of this International Standard with *ENCODING* set to indicate that binary strings are to be encoded using hex.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

b) Subclause 9.17, "Mapping an SQL catalog to an XML document and an XML Schema document":

i) Without Feature X059, "Advanced table mapping: hex encoding of binary strings", a conforming application shall not invoke this Subclause of this part of this International Standard with *ENCODING* set to indicate that binary strings are to be encoded using hex.

42) Specifications for Feature X060, "XMLParse: Character string input and CONTENT option":

a) Subclause 6.16, "<XML parse>":

i) Without Feature X060, "XMLParse: Character string input and CONTENT option", in conforming SQL language, the declared type of the <string value expression> immediately contained in <XML parse> shall not be a character string type and <XML parse> shall not immediately contain a <document or content> that is CONTENT.

43) Specifications for Feature X061, "XMLParse: Character string input and DOCUMENT option":

a) Subclause 6.16, "<XML parse>":

i) Without Feature X061, "XMLParse: Character string input and DOCUMENT option", in conforming SQL language, the declared type of the <string value expression> immediately contained in <XML parse> shall not be a character string type and <XML parse> shall not immediately contain a <document or content> that is DOCUMENT.

44) Specifications for Feature X065, "XMLParse: BLOB input and CONTENT option":

a) Subclause 6.16, "<XML parse>":

i) Without Feature X065, "XMLParse: BLOB input and CONTENT option", in conforming SQL language, the declared type of the <string value expression> immediately contained in <XML parse> shall not be a binary string type and <XML parse> shall not immediately contain a <document or content> that is CONTENT.

45) Specifications for Feature X066, "XMLParse: BLOB input and DOCUMENT option":

a) Subclause 6.16, "<XML parse>":

i) Without Feature X066, "XMLParse: BLOB input and DOCUMENT option", in conforming SQL language, the declared type of the <string value expression> immediately contained in <XML parse> shall not be a binary string type and <XML parse> shall not immediately contain a <document or content> that is DOCUMENT.

46) Specifications for Feature X068, "XMLSerialize: BOM":

a) Subclause 6.8, "<string value function>":

i) Insert this CR Without Feature X068, "XMLSerialize: BOM", in conforming SQL language, <XML serialize bom> shall not be specified.

47) Specifications for Feature X069, "XMLSerialize: INDENT":

a) Subclause 6.8, "<string value function>":

i) Insert this CR Without Feature X069, "XMLSerialize: INDENT", in conforming SQL language, <XML serialize indent> shall not be specified.

48) Specifications for Feature X070, "XMLSerialize: Character string serialization and CONTENT option":

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

a) Subclause 6.8, "<string value function>":

i) Insert this CR Without Feature X070, "XMLSerialize: Character string serialization and CONTENT option", conforming SQL language shall not contain an <XML character string serialization> that immediately contains a <document or content> that is CONTENT.

49) Specifications for Feature X071, "XMLSerialize: Character string serialization and DOCUMENT option":

a) Subclause 6.8, "<string value function>":

i) Insert this CR Without Feature X071, "XMLSerialize: Character string serialization and DOCUMENT option", conforming SQL language shall not contain an <XML character string serialization> that immediately contains a <document or content> that is DOCUMENT.

50) Specifications for Feature X072, "XMLSerialize: character string serialization":

a) Subclause 6.8, "<string value function>":

i) Insert this CR Without Feature X072, "XMLSerialize: character string serialization", conforming SQL language shall not contain an <XML character string serialization>.

51) Specifications for Feature X073, "XMLSerialize: BLOB serialization and CONTENT option":

a) Subclause 6.8, "<string value function>":

i) Insert this CR Without Feature X073, "XMLSerialize: BLOB serialization and CONTENT option", conforming SQL language shall not contain an <XML binary string serialization> that immediately contains a <document or content> that is CONTENT.

52) Specifications for Feature X074, "XMLSerialize: BLOB serialization and DOCUMENT option":

a) Subclause 6.8, "<string value function>":

i) Insert this CR Without Feature X074, "XMLSerialize: BLOB serialization and DOCUMENT option", conforming SQL language shall not contain an <XML binary string serialization> that immediately contains a <document or content> that is DOCUMENT.

53) Specifications for Feature X075, "XMLSerialize: BLOB serialization":

a) Subclause 6.8, "<string value function>":

i) Insert this CR Without Feature X075, "XMLSerialize: BLOB serialization", conforming SQL language shall not contain an <XML binary string serialization>.

54) Specifications for Feature X076, "XMLSerialize: VERSION":

a) Subclause 6.8, "<string value function>":

i) Insert this CR Without Feature X076, "XMLSerialize: VERSION", in conforming SQL language, <XML character string serialization> shall not contain VERSION.

ii) Insert this CR Without Feature X076, "XMLSerialize: VERSION", in conforming SQL language, <XML binary string serialization> shall not contain VERSION.

55) Specifications for Feature X077, "XMLSerialize: explicit ENCODING option":

a) Subclause 6.8, "<string value function>":

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

i) Insert this CR Without Feature X077, "XMLSerialize: explicit ENCODING option", conforming SQL language shall not contain an <XML binary string serialization> that contains ENCODING.

56) Specifications for Feature X078, "XMLSerialize: explicit XML declaration":

a) Subclause 6.8, "<string value function>":

i) Insert this CR Without Feature X078, "XMLSerialize: explicit XML declaration", in conforming SQL language, <XML character string serialization> shall not contain XMLDECLARATION.

ii) Insert this CR Without Feature X078, "XMLSerialize: explicit XML declaration", in conforming SQL language, <XML binary string serialization> shall not contain XMLDECLARATION.

57) Specifications for Feature X080, "Namespaces in XML publishing":

a) Subclause 6.14, "<XML element>":

i) Without Feature X080, "Namespaces in XML publishing", in conforming SQL language, <XML element> shall not immediately contain <XML namespace declaration>.

b) Subclause 6.15, "<XML forest>":

i) Without Feature X080, "Namespaces in XML publishing", in conforming SQL language, <XML forest> shall not immediately contain <XML namespace declaration>.

58) Specifications for Feature X081, "Query-level XML namespace declarations":

a) Subclause 7.2, "<query expression>":

i) Insert this CR Without Feature X081, "Query-level XML namespace declarations", in conforming SQL language, <with clause> shall not immediately contain an <XML lexically scoped options> that contains an <XML namespace declaration>.

59) Specifications for Feature X082, "XML namespace declarations in DML":

a) Subclause 14.3, "<delete statement: searched>":

i) Insert this CR Without Feature X082, "XML namespace declarations in DML", in conforming SQL language, a <delete statement: searched> shall not immediately contain an <XML lexically scoped options> that contains an <XML namespace declaration>.

b) Subclause 14.4, "<insert statement>":

i) Insert this CR Without Feature X082, "XML namespace declarations in DML", in conforming SQL language, an <insert statement> shall not immediately contain an <XML lexically scoped options> that contains an <XML namespace declaration>.

c) Subclause 14.5, "<merge statement>":

i) Insert this CR Without Feature X082, "XML namespace declarations in DML", in conforming SQL language, a <merge statement> shall not immediately contain an <XML lexically scoped options> that contains an <XML namespace declaration>.

d) Subclause 14.6, "<update statement: positioned>":

i) Insert this CR Without Feature X082, "XML namespace declarations in DML", in conforming SQL language, an <update statement: positioned> shall not immediately contain an <XML lexically scoped options> that contains an <XML namespace declaration>.

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

e) Subclause 14.7, "<update statement: searched>":

i) Insert this CR Without Feature X082, "XML namespace declarations in DML", in conforming SQL language, an <update statement: searched> shall not immediately contain an <XML lexically scoped options> that contains an <XML namespace declaration>.

60) Specifications for Feature X083, "XML namespace declarations in DDL":

a) Subclause 12.1, "<column definition>":

i) Insert this CR Without Feature X083, "XML namespace declarations in DDL", in conforming SQL language, a <generation expression> shall not immediately contain an <XML lexically scoped options> that contains an <XML namespace declaration>.

b) Subclause 12.2, "<check constraint definition>":

i) Insert this CR Without Feature X083, "XML namespace declarations in DDL", in conforming SQL language, a <check constraint definition> shall not immediately contain and <XML lexically scoped options> that contains an <XML namespace declaration>.

c) Subclause 12.4, "<assertion definition>":

i) Insert this CR Without Feature X083, "XML namespace declarations in DDL", in conforming SQL language, an <assertion definition> shall not immediately contain an <XML lexically scoped options> that contains an <XML namespace declaration>.

61) Specifications for Feature X084, "XML namespace declarations in compound statements":

a) Subclause 15.1, "<compound statement>":

i) Insert this CR Without Feature X084, "XML namespace declarations in compound statements", in conforming SQL language, a <compound statement> shall not immediately contain an <XML lexically scoped options> that contains an <XML namespace declaration>.

62) Specifications for Feature X085, "Predefined namespace prefixes":

a) Subclause 6.14, "<XML element>":

i) Without Feature X085, "Predefined namespace prefixes", conforming SQL language shall not contain an <XML element name> *E* that has an XML QName prefix that is not equivalent to an <XML namespace prefix> contained in one or more <XML namespace declaration>s that are the scope of the <XML element> that contains *E*.

ii) Without Feature X085, "Predefined namespace prefixes", conforming SQL language shall not contain an explicit or implicit <XML attribute name> *A* that has an XML QName prefix other than **'xml'** that is not equivalent to an <XML namespace prefix> contained in one or more <XML namespace declaration>s that are the scope of the <XML element> that contains *A*.

b) Subclause 6.15, "<XML forest>":

i) Without Feature X085, "Predefined namespace prefixes", conforming SQL language shall not contain an explicit or implicit <forest element name> *F* that has an XML QName prefix that is not equivalent to an <XML namespace prefix> contained in one or more <XML namespace declaration>s that are the scope of the <XML forest> that contains *F*.

63) Specifications for Feature X086, "XML namespace declarations in XMLTable":

a) Subclause 7.1, "<table reference>":

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

i) Insert this CR Without Feature X086, “XML namespace declarations in XMLTable”, in conforming SQL language, an <XML table> shall not contain an <XML namespace declaration>.

64) Specifications for Feature X090, “XML document predicate”:

a) Subclause 8.3, “<XML document predicate>”:

i) Without Feature X090, “XML document predicate”, conforming SQL language shall not contain <XML document predicate>.

65) Specifications for Feature X091, “XML content predicate”:

a) Subclause 8.2, “<XML content predicate>”:

i) Without Feature X091, “XML content predicate”, conforming SQL language shall not contain <XML content predicate>.

66) Specifications for Feature X096, “XMLExists”:

a) Subclause 8.4, “<XML exists predicate>”:

i) Without Feature X096, “XMLExists”, conforming SQL language shall not contain <XML exists predicate>.

67) Specifications for Feature X100, “Host language support for XML: CONTENT option”:

a) Subclause 12.7, “<SQL-invoked routine>”:

i) Insert this CR Without Feature X100, “Host language support for XML: CONTENT option”, conforming SQL language shall not contain a <document or content> that is CONTENT.

b) Subclause 16.1, “<set XML option statement>”:

i) Without Feature X100, “Host language support for XML: CONTENT option”, conforming SQL language shall not contain a <set XML option statement> that contains CONTENT.

c) Subclause 18.2, “<embedded SQL Ada program>”:

i) Insert this CR Without Feature X100, “Host language support for XML: CONTENT option”, in conforming SQL language, <Ada XML CLOB variable> shall not immediately contain a <document or content> that is CONTENT.

ii) Insert this CR Without Feature X100, “Host language support for XML: CONTENT option”, in conforming SQL language, <Ada XML BLOB variable> shall not immediately contain a <document or content> that is CONTENT.

d) Subclause 18.3, “<embedded SQL C program>”:

i) Insert this CR Without Feature X100, “Host language support for XML: CONTENT option”, in conforming SQL language, neither <C XML VARCHAR variable> nor <C XML CLOB variable> shall immediately contain a <document or content> that is CONTENT.

ii) Insert this CR Without Feature X100, “Host language support for XML: CONTENT option”, in conforming SQL language, <C XML BLOB variable> shall not immediately contain a <document or content> that is CONTENT.

e) Subclause 18.4, “<embedded SQL COBOL program>”:

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

i) Insert this CR Without Feature X100, “Host language support for XML: CONTENT option”, in conforming SQL language, <COBOL XML CLOB variable> shall not immediately contain a <document or content> that is CONTENT.

ii) Insert this CR Without Feature X100, “Host language support for XML: CONTENT option”, in conforming SQL language, <COBOL XML BLOB variable> shall not immediately contain a <document or content> that is CONTENT.

f) Subclause 18.5, “<embedded SQL Fortran program>”:

i) Insert this CR Without Feature X100, “Host language support for XML: CONTENT option”, in conforming SQL language, <Fortran XML CLOB variable> shall not immediately contain a <document or content> that is CONTENT.

ii) Insert this CR Without Feature X100, “Host language support for XML: CONTENT option”, in conforming SQL language, <Fortran XML BLOB variable> shall not immediately contain a <document or content> that is CONTENT.

g) Subclause 18.6, “<embedded SQL MUMPS program>”:

i) Insert this CR Without Feature X100, “Host language support for XML: CONTENT option”, in conforming SQL language, <MUMPS XML CLOB variable> shall not immediately contain a <document or content> that is CONTENT.

ii) Insert this CR Without Feature X100, “Host language support for XML: CONTENT option”, in conforming SQL language, <MUMPS XML BLOB variable> shall not immediately contain a <document or content> that is CONTENT.

h) Subclause 18.7, “<embedded SQL Pascal program>”:

i) Insert this CR Without Feature X100, “Host language support for XML: CONTENT option”, in conforming SQL language, <Pascal XML CLOB variable> shall not immediately contain a <document or content> that is CONTENT.

ii) Insert this CR Without Feature X100, “Host language support for XML: CONTENT option”, in conforming SQL language, <Pascal XML BLOB variable> shall not immediately contain a <document or content> that is CONTENT.

i) Subclause 18.8, “<embedded SQL PL/I program>”:

i) Insert this CR Without Feature X100, “Host language support for XML: CONTENT option”, in conforming SQL language, neither <PL/I XML VARCHAR variable> nor <PL/I XML CLOB variable> shall immediately contain a <document or content> that is CONTENT.

ii) Insert this CR Without Feature X100, “Host language support for XML: CONTENT option”, in conforming SQL language, <PL/I XML BLOB variable> shall not immediately contain a <document or content> that is CONTENT.

68) Specifications for Feature X101, “Host language support for XML: DOCUMENT option”:

a) Subclause 12.7, “<SQL-invoked routine>”:

i) Insert this CR Without Feature X101, “Host language support for XML: DOCUMENT option”, conforming SQL language shall not contain a <document or content> that is DOCUMENT.

b) Subclause 16.1, “<set XML option statement>”:

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

i) Without Feature X101, "Host language support for XML: DOCUMENT option", conforming SQL language shall not contain a <set XML option statement> that contains DOCUMENT.

c) Subclause 18.2, "<embedded SQL Ada program>":

i) Insert this CR Without Feature X101, "Host language support for XML: DOCUMENT option", in conforming SQL language, <Ada XML CLOB variable> shall not immediately contain a <document or content> that is DOCUMENT.

ii) Insert this CR Without Feature X101, "Host language support for XML: DOCUMENT option", in conforming SQL language, <Ada XML BLOB variable> shall not immediately contain a <document or content> that is DOCUMENT.

d) Subclause 18.3, "<embedded SQL C program>":

i) Insert this CR Without Feature X101, "Host language support for XML: DOCUMENT option", in conforming SQL language, neither <C XML VARCHAR variable> nor <C XML CLOB variable> shall immediately contain a <document or content> that is DOCUMENT.

ii) Insert this CR Without Feature X101, "Host language support for XML: DOCUMENT option", in conforming SQL language, <C XML BLOB variable> shall not immediately contain a <document or content> that is DOCUMENT.

e) Subclause 18.4, "<embedded SQL COBOL program>":

i) Insert this CR Without Feature X101, "Host language support for XML: DOCUMENT option", in conforming SQL language, <COBOL XML CLOB variable> shall not immediately contain a <document or content> that is DOCUMENT.

ii) Insert this CR Without Feature X101, "Host language support for XML: DOCUMENT option", in conforming SQL language, <COBOL XML BLOB variable> shall not immediately contain a <document or content> that is DOCUMENT.

f) Subclause 18.5, "<embedded SQL Fortran program>":

i) Insert this CR Without Feature X101, "Host language support for XML: DOCUMENT option", in conforming SQL language, <Fortran XML CLOB variable> shall not immediately contain a <document or content> that is DOCUMENT.

ii) Insert this CR Without Feature X101, "Host language support for XML: DOCUMENT option", in conforming SQL language, <Fortran XML BLOB variable> shall not immediately contain a <document or content> that is DOCUMENT.

g) Subclause 18.6, "<embedded SQL MUMPS program>":

i) Insert this CR Without Feature X101, "Host language support for XML: DOCUMENT option", in conforming SQL language, <MUMPS XML CLOB variable> shall not immediately contain a <document or content> that is DOCUMENT.

ii) Insert this CR Without Feature X101, "Host language support for XML: DOCUMENT option", in conforming SQL language, <MUMPS XML BLOB variable> shall not immediately contain a <document or content> that is DOCUMENT.

h) Subclause 18.7, "<embedded SQL Pascal program>":

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

i) Insert this CR Without Feature X101, “Host language support for XML: DOCUMENT option”, in conforming SQL language, <Pascal XML CLOB variable> shall not immediately contain a <document or content> that is DOCUMENT.

ii) Insert this CR Without Feature X101, “Host language support for XML: DOCUMENT option”, in conforming SQL language, <Pascal XML BLOB variable> shall not immediately contain a <document or content> that is DOCUMENT.

i) Subclause 18.8, “<embedded SQL PL/I program>”:

i) Insert this CR Without Feature X101, “Host language support for XML: DOCUMENT option”, in conforming SQL language, neither <PL/I XML VARCHAR variable> nor <PL/I XML CLOB variable> shall immediately contain a <document or content> that is DOCUMENT.

ii) Insert this CR Without Feature X101, “Host language support for XML: DOCUMENT option”, in conforming SQL language, <PL/I XML BLOB variable> shall not immediately contain a <document or content> that is DOCUMENT.

69) Specifications for Feature X110, “Host language support for XML: VARCHAR mapping”:

a) Subclause 12.7, “<SQL-invoked routine>”:

i) Insert this CR Without Feature X110, “Host language support for XML: VARCHAR mapping”, conforming SQL language shall not contain a <string type option> that contains CHARACTER VARYING, CHAR VARYING, or VARCHAR.

b) Subclause 18.3, “<embedded SQL C program>”:

i) Insert this CR Without Feature X110, “Host language support for XML: VARCHAR mapping”, conforming SQL language shall not contain an <C XML VARCHAR variable>.

c) Subclause 18.8, “<embedded SQL PL/I program>”:

i) Insert this CR Without Feature X110, “Host language support for XML: VARCHAR mapping”, conforming SQL language shall not contain an <PL/I XML VARCHAR variable>.

70) Specifications for Feature X111, “Host language support for XML: CLOB mapping”:

a) Subclause 12.7, “<SQL-invoked routine>”:

i) Insert this CR Without Feature X111, “Host language support for XML: CLOB mapping”, conforming SQL language shall not contain a <string type option> that contains CHARACTER LARGE OBJECT, CHAR LARGE OBJECT, or CLOB.

b) Subclause 18.2, “<embedded SQL Ada program>”:

i) Insert this CR Without Feature X111, “Host language support for XML: CLOB mapping”, conforming SQL language shall not contain an <Ada XML CLOB variable>.

c) Subclause 18.3, “<embedded SQL C program>”:

i) Insert this CR Without Feature X111, “Host language support for XML: CLOB mapping”, conforming SQL language shall not contain an <C XML CLOB variable>.

d) Subclause 18.4, “<embedded SQL COBOL program>”:

i) Insert this CR Without Feature X111, “Host language support for XML: CLOB mapping”, conforming SQL language shall not contain an <COBOL XML CLOB variable>.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

e) Subclause 18.5, "<embedded SQL Fortran program>":

i) Insert this CR Without Feature X111, "Host language support for XML: CLOB mapping", conforming SQL language shall not contain an <Fortran XML CLOB variable>.

f) Subclause 18.6, "<embedded SQL MUMPS program>":

i) Insert this CR Without Feature X111, "Host language support for XML: CLOB mapping", conforming SQL language shall not contain a <MUMPS XML CLOB variable>.

g) Subclause 18.7, "<embedded SQL Pascal program>":

i) Insert this CR Without Feature X111, "Host language support for XML: CLOB mapping", conforming SQL language shall not contain an <Pascal XML CLOB variable>.

h) Subclause 18.8, "<embedded SQL PL/I program>":

i) Insert this CR Without Feature X111, "Host language support for XML: CLOB mapping", conforming SQL language shall not contain an <PL/I XML CLOB variable>.

71) Specifications for Feature X112, "Host language support for XML: BLOB mapping":

a) Subclause 18.2, "<embedded SQL Ada program>":

i) Insert this CR Without Feature X112, "Host language support for XML: BLOB mapping", conforming SQL language shall not contain an <Ada XML BLOB variable>.

b) Subclause 18.3, "<embedded SQL C program>":

i) Insert this CR Without Feature X112, "Host language support for XML: BLOB mapping", conforming SQL language shall not contain a <C XML BLOB variable>.

c) Subclause 18.4, "<embedded SQL COBOL program>":

i) Insert this CR Without Feature X112, "Host language support for XML: BLOB mapping", conforming SQL language shall not contain a <COBOL XML BLOB variable>.

d) Subclause 18.5, "<embedded SQL Fortran program>":

i) Insert this CR Without Feature X112, "Host language support for XML: BLOB mapping", conforming SQL language shall not contain a <Fortran XML BLOB variable>.

e) Subclause 18.6, "<embedded SQL MUMPS program>":

i) Insert this CR Without Feature X112, "Host language support for XML: BLOB mapping", conforming SQL language shall not contain a <MUMPS XML BLOB variable>.

f) Subclause 18.7, "<embedded SQL Pascal program>":

i) Insert this CR Without Feature X112, "Host language support for XML: BLOB mapping", conforming SQL language shall not contain a <Pascal XML BLOB variable>.

g) Subclause 18.8, "<embedded SQL PL/I program>":

i) Insert this CR Without Feature X112, "Host language support for XML: BLOB mapping", conforming SQL language shall not contain a <PL/I XML BLOB variable>.

72) Specifications for Feature X113, "Host language support for XML: STRIP WHITESPACE option":

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

a) Subclause 18.2, "<embedded SQL Ada program>":

   i) Insert this CR Without Feature X113, "Host language support for XML: STRIP WHITESPACE option", in conforming SQL language, <Ada XML CLOB variable> shall not immediately contain an <XML whitespace option> that is STRIP WHITESPACE.

   ii) Insert this CR Without Feature X113, "Host language support for XML: STRIP WHITESPACE option", in conforming SQL language, <Ada XML BLOB variable> shall not immediately contain an <XML whitespace option> that is STRIP WHITESPACE.

b) Subclause 18.3, "<embedded SQL C program>":

   i) Insert this CR Without Feature X113, "Host language support for XML: STRIP WHITESPACE option", in conforming SQL language, <C XML CLOB variable> shall not immediately contain an <XML whitespace option> that is STRIP WHITESPACE.

   ii) Insert this CR Without Feature X113, "Host language support for XML: STRIP WHITESPACE option", in conforming SQL language, <C XML BLOB variable> shall not immediately contain an <XML whitespace option> that is STRIP WHITESPACE.

   iii) Insert this CR Without Feature X113, "Host language support for XML: STRIP WHITESPACE option", in conforming SQL language, <C XML VARCHAR variable> shall not immediately contain an <XML whitespace option> that is STRIP WHITESPACE.

c) Subclause 18.4, "<embedded SQL COBOL program>":

   i) Insert this CR Without Feature X113, "Host language support for XML: STRIP WHITESPACE option", in conforming SQL language, <COBOL XML CLOB variable> shall not immediately contain an <XML whitespace option> that is STRIP WHITESPACE.

   ii) Insert this CR Without Feature X113, "Host language support for XML: STRIP WHITESPACE option", in conforming SQL language, <COBOL XML BLOB variable> shall not immediately contain an <XML whitespace option> that is STRIP WHITESPACE.

d) Subclause 18.5, "<embedded SQL Fortran program>":

   i) Insert this CR Without Feature X113, "Host language support for XML: STRIP WHITESPACE option", in conforming SQL language, <Fortran XML CLOB variable> shall not immediately contain an <XML whitespace option> that is STRIP WHITESPACE.

   ii) Insert this CR Without Feature X113, "Host language support for XML: STRIP WHITESPACE option", in conforming SQL language, <Fortran XML BLOB variable> shall not immediately contain an <XML whitespace option> that is STRIP WHITESPACE.

e) Subclause 18.6, "<embedded SQL MUMPS program>":

   i) Insert this CR Without Feature X113, "Host language support for XML: STRIP WHITESPACE option", in conforming SQL language, <MUMPS XML CLOB variable> shall not immediately contain an <XML whitespace option> that is STRIP WHITESPACE.

   ii) Insert this CR Without Feature X113, "Host language support for XML: STRIP WHITESPACE option", in conforming SQL language, <MUMPS XML BLOB variable> shall not immediately contain an <XML whitespace option> that is STRIP WHITESPACE.

f) Subclause 18.7, "<embedded SQL Pascal program>":

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

i) Insert this CR Without Feature X113, “Host language support for XML: STRIP WHITESPACE option”, in conforming SQL language, <Pascal XML CLOB variable> shall not immediately contain an <XML whitespace option> that is STRIP WHITESPACE.

ii) Insert this CR Without Feature X113, “Host language support for XML: STRIP WHITESPACE option”, in conforming SQL language, <Pascal XML BLOB variable> shall not immediately contain an <XML whitespace option> that is STRIP WHITESPACE.

g) Subclause 18.8, “<embedded SQL PL/I program>”:

i) Insert this CR Without Feature X113, “Host language support for XML: STRIP WHITESPACE option”, in conforming SQL language, <PL/I XML CLOB variable> shall not immediately contain an <XML whitespace option> that is STRIP WHITESPACE.

ii) Insert this CR Without Feature X113, “Host language support for XML: STRIP WHITESPACE option”, in conforming SQL language, <PL/I XML BLOB variable> shall not immediately contain an <XML whitespace option> that is STRIP WHITESPACE.

iii) Insert this CR Without Feature X113, “Host language support for XML: STRIP WHITESPACE option”, in conforming SQL language, <PL/I XML VARCHAR variable> shall not immediately contain an <XML whitespace option> that is STRIP WHITESPACE.

73) Specifications for Feature X114, “Host language support for XML: PRESERVE WHITESPACE option”:

a) Subclause 18.2, “<embedded SQL Ada program>”:

i) Insert this CR Without Feature X114, “Host language support for XML: PRESERVE WHITESPACE option”, in conforming SQL language, <Ada XML CLOB variable> shall not immediately contain an <XML whitespace option> that is PRESERVE WHITESPACE.

ii) Insert this CR Without Feature X114, “Host language support for XML: PRESERVE WHITESPACE option”, in conforming SQL language, <Ada XML BLOB variable> shall not immediately contain an <XML whitespace option> that is PRESERVE WHITESPACE.

b) Subclause 18.3, “<embedded SQL C program>”:

i) Insert this CR Without Feature X114, “Host language support for XML: PRESERVE WHITESPACE option”, in conforming SQL language, <C XML CLOB variable> shall not immediately contain an <XML whitespace option> that is PRESERVE WHITESPACE.

ii) Insert this CR Without Feature X114, “Host language support for XML: PRESERVE WHITESPACE option”, in conforming SQL language, <C XML BLOB variable> shall not immediately contain an <XML whitespace option> that is PRESERVE WHITESPACE.

iii) Insert this CR Without Feature X114, “Host language support for XML: PRESERVE WHITESPACE option”, in conforming SQL language, <C XML VARCHAR variable> shall not immediately contain an <XML whitespace option> that is PRESERVE WHITESPACE.

c) Subclause 18.4, “<embedded SQL COBOL program>”:

i) Insert this CR Without Feature X114, “Host language support for XML: PRESERVE WHITESPACE option”, in conforming SQL language, <COBOL XML CLOB variable> shall not immediately contain an <XML whitespace option> that is PRESERVE WHITESPACE.

ii) Insert this CR Without Feature X114, “Host language support for XML: PRESERVE WHITESPACE option”, in conforming SQL language, <COBOL XML BLOB variable> shall not immediately contain an <XML whitespace option> that is PRESERVE WHITESPACE.

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

d) Subclause 18.5, "<embedded SQL Fortran program>":

i) Insert this CR Without Feature X114, "Host language support for XML: PRESERVE WHITESPACE option", in conforming SQL language, <Fortran XML CLOB variable> shall not immediately contain an <XML whitespace option> that is PRESERVE WHITESPACE.

ii) Insert this CR Without Feature X114, "Host language support for XML: PRESERVE WHITESPACE option", in conforming SQL language, <Fortran XML BLOB variable> shall not immediately contain an <XML whitespace option> that is PRESERVE WHITESPACE.

e) Subclause 18.6, "<embedded SQL MUMPS program>":

i) Insert this CR Without Feature X114, "Host language support for XML: PRESERVE WHITESPACE option", in conforming SQL language, <MUMPS XML CLOB variable> shall not immediately contain an <XML whitespace option> that is PRESERVE WHITESPACE.

ii) Insert this CR Without Feature X114, "Host language support for XML: PRESERVE WHITESPACE option", in conforming SQL language, <MUMPS XML BLOB variable> shall not immediately contain an <XML whitespace option> that is PRESERVE WHITESPACE.

f) Subclause 18.7, "<embedded SQL Pascal program>":

i) Insert this CR Without Feature X114, "Host language support for XML: PRESERVE WHITESPACE option", in conforming SQL language, <Pascal XML CLOB variable> shall not immediately contain an <XML whitespace option> that is PRESERVE WHITESPACE.

ii) Insert this CR Without Feature X114, "Host language support for XML: PRESERVE WHITESPACE option", in conforming SQL language, <Pascal XML BLOB variable> shall not immediately contain an <XML whitespace option> that is PRESERVE WHITESPACE.

g) Subclause 18.8, "<embedded SQL PL/I program>":

i) Insert this CR Without Feature X114, "Host language support for XML: PRESERVE WHITESPACE option", in conforming SQL language, <PL/I XML CLOB variable> shall not immediately contain an <XML whitespace option> that is PRESERVE WHITESPACE.

ii) Insert this CR Without Feature X114, "Host language support for XML: PRESERVE WHITESPACE option", in conforming SQL language, <PL/I XML BLOB variable> shall not immediately contain an <XML whitespace option> that is PRESERVE WHITESPACE.

iii) Insert this CR Without Feature X114, "Host language support for XML: PRESERVE WHITESPACE option", in conforming SQL language, <PL/I XML VARCHAR variable> shall not immediately contain an <XML whitespace option> that is PRESERVE WHITESPACE.

74) Specifications for Feature X120, "XML parameters in SQL routines":

a) Subclause 12.7, "<SQL-invoked routine>":

i) Insert this CR Without Feature X120, "XML parameters in SQL routines", conforming SQL language shall not contain an <SQL-invoked routine> that simply contains a <language clause> that contains SQL and that simply contains a <parameter type> or a <returns data type> that contains a <data type> that is based on either an XML type or a distinct type whose source type is an XML type.

75) Specifications for Feature X121, "XML parameters in external routines":

a) Subclause 12.7, "<SQL-invoked routine>":

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

i) Insert this CR Without Feature X121, “XML parameters in external routines”, conforming SQL language shall not contain an <SQL-invoked routine> that simply contains a <language clause> that contains ADA, C, COBOL, FORTRAN, MUMPS, PASCAL, or PLI and that simply contains a <parameter type> or a <returns data type> that contains a <data type> that is based on either an XML type or a distinct type whose source type is an XML type.

76) Specifications for Feature X131, “Query-level XMLBINARY clause”:

a) Subclause 7.2, “<query expression>”:

i) Insert this CR Without Feature X131, “Query-level XMLBINARY clause”, in conforming SQL language, a <with clause> shall not immediately contain an <XML lexically scoped options> that contains an <XML binary encoding>.

77) Specifications for Feature X132, “XMLBINARY clause in DML”:

a) Subclause 14.3, “<delete statement: searched>”:

i) Insert this CR Without Feature X132, “XMLBINARY clause in DML”, in conforming SQL language, a <delete statement: searched> shall not immediately contain an <XML lexically scoped options> that contains an <XML binary encoding>.

b) Subclause 14.4, “<insert statement>”:

i) Insert this CR Without Feature X132, “XMLBINARY clause in DML”, in conforming SQL language, an <insert statement> shall not immediately contain an <XML lexically scoped options> that contains an <XML binary encoding>.

c) Subclause 14.5, “<merge statement>”:

i) Insert this CR Without Feature X132, “XMLBINARY clause in DML”, in conforming SQL language, a <merge statement> shall not immediately contain an <XML lexically scoped options> that contains an <XML binary encoding>.

d) Subclause 14.6, “<update statement: positioned>”:

i) Insert this CR Without Feature X132, “XMLBINARY clause in DML”, in conforming SQL language, an <update statement: positioned> shall not immediately contain an <XML lexically scoped options> that contains an <XML binary encoding>.

e) Subclause 14.7, “<update statement: searched>”:

i) Insert this CR Without Feature X132, “XMLBINARY clause in DML”, in conforming SQL language, an <update statement: searched> shall not immediately contain an <XML lexically scoped options> that contains an <XML binary encoding>.

78) Specifications for Feature X133, “XMLBINARY clause in DDL”:

a) Subclause 12.1, “<column definition>”:

i) Insert this CR Without Feature X133, “XMLBINARY clause in DDL”, in conforming SQL language, a <generation expression> shall not immediately contain an <XML lexically scoped options> that contains an <XML binary encoding>.

b) Subclause 12.2, “<check constraint definition>”:

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

i) Insert this CR Without Feature X133, “XMLBINARY clause in DDL”, in conforming SQL language, a <check constraint definition> shall not immediately contain an <XML lexically scoped options> that contains an <XML binary encoding>.

c) Subclause 12.4, “<assertion definition>”:

i) Insert this CR Without Feature X133, “XMLBINARY clause in DDL”, in conforming SQL language, an <assertion definition> shall not immediately contain an <XML lexically scoped options> that contains an <XML binary encoding>.

79) Specifications for Feature X134, “XMLBINARY clause in compound statements”:

a) Subclause 15.1, “<compound statement>”:

i) Insert this CR Without Feature X134, “XMLBINARY clause in compound statements”, in conforming SQL language, a <compound statement> shall not immediately contain an <XML lexically scoped options> that contains an <XML binary encoding>.

80) Specifications for Feature X135, “XMLBINARY clause in subqueries”:

a) Subclause 7.2, “<query expression>”:

i) Insert this CR Without Feature X135, “XMLBINARY clause in subqueries”, in conforming SQL language, a <query expression> that is contained in another <query expression> shall not contain an <XML lexically scoped options> that contains an <XML binary encoding>.

81) Specifications for Feature X141, “IS VALID predicate: data-driven case”:

a) Subclause 8.5, “<XML valid predicate>”:

i) Without Feature X141, “IS VALID predicate: data-driven case”, conforming SQL language shall not contain an <XML valid predicate> that does not contain <XML valid according to clause>.

82) Specifications for Feature X142, “IS VALID predicate: ACCORDING TO clause”:

a) Subclause 8.5, “<XML valid predicate>”:

i) Without Feature X142, “IS VALID predicate: ACCORDING TO clause”, conforming SQL language shall not contain an <XML valid predicate> that contains <XML valid according to clause>.

83) Specifications for Feature X143, “IS VALID predicate: ELEMENT clause”:

a) Subclause 8.5, “<XML valid predicate>”:

i) Without Feature X143, “IS VALID predicate: ELEMENT clause”, conforming SQL language shall not contain an <XML valid predicate> that contains <XML valid element clause>.

84) Specifications for Feature X144, “IS VALID predicate: schema location”:

a) Subclause 8.5, “<XML valid predicate>”:

i) Without Feature X144, “IS VALID predicate: schema location”, conforming SQL language shall not contain an <XML valid predicate> that contains <XML valid schema location>.

85) Specifications for Feature X145, “IS VALID predicate outside check constraints”:

a) Subclause 8.5, “<XML valid predicate>”:

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

i) Without Feature X145, “IS VALID predicate outside check constraints”, conforming SQL language shall not contain an <XML valid predicate> that is not directly contained in the <search condition> of a <check constraint definition>.

86) Specifications for Feature X151, “IS VALID predicate: with DOCUMENT option”:

a) Subclause 8.5, “<XML valid predicate>”:

i) Without Feature X151, “IS VALID predicate: with DOCUMENT option”, conforming SQL language shall not contain an <XML valid predicate> that immediately contains a <document or content or sequence> that is DOCUMENT.

87) Specifications for Feature X152, “IS VALID predicate: with CONTENT option”:

a) Subclause 8.5, “<XML valid predicate>”:

i) Without Feature X152, “IS VALID predicate: with CONTENT option”, conforming SQL language shall not contain an <XML valid predicate> that immediately contains a <document or content or sequence> that is CONTENT.

88) Specifications for Feature X153, “IS VALID predicate: with SEQUENCE option”:

a) Subclause 8.5, “<XML valid predicate>”:

i) Without Feature X153, “IS VALID predicate: with SEQUENCE option”, conforming SQL language shall not contain an <XML valid predicate> that immediately contains a <document or content or sequence> that is SEQUENCE.

89) Specifications for Feature X155, “IS VALID predicate: NAMESPACE without ELEMENT clause”:

a) Subclause 8.5, “<XML valid predicate>”:

i) Without Feature X155, “IS VALID predicate: NAMESPACE without ELEMENT clause”, conforming SQL language shall not contain an <XML valid predicate> that contains an <XML valid element clause> that does not contain an <XML valid element name specification>.

90) Specifications for Feature X157, “IS VALID predicate: NO NAMESPACE with ELEMENT clause”:

a) Subclause 8.5, “<XML valid predicate>”:

i) Without Feature X157, “IS VALID predicate: NO NAMESPACE with ELEMENT clause”, conforming SQL language shall not contain an <XML valid predicate> that contains an <XML valid element namespace specification> that contains NO NAMESPACE.

91) Specifications for Feature X160, “Basic Information Schema for registered XML Schemas”:

a) Subclause 20.3, “ATTRIBUTES view”:

i) Insert this CR Without Feature X160, “Basic Information Schema for registered XML Schemas”, conforming SQL language shall not reference INFORMATION_SCHEMA.ATTRIBUTES.XML_SCHEMA_CATALOG, INFORMATION_SCHEMA.ATTRIBUTES.XML_SCHEMA_SCHEMA, or INFORMATION_SCHEMA.ATTRIBUTES.XML_SCHEMA_NAME.

b) Subclause 20.4, “COLUMNS view”:

i) Insert this CR Without Feature X160, “Basic Information Schema for registered XML Schemas”, conforming SQL language shall not reference INFORMA-

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

TION_SCHEMA.COLUMNS.XML_SCHEMA_CATALOG, INFORMATION_SCHEMA.COLUMNS.XML_SCHEMA_SCHEMA, or INFORMATION_SCHEMA.COLUMNS.XML_SCHEMA_NAME.

c) Subclause 20.5, "DOMAINS view":

i) Insert this CR Without Feature X160, "Basic Information Schema for registered XML Schemas", conforming SQL language shall not reference INFORMATION_SCHEMA.DOMAINS.XML_SCHEMA_CATALOG, INFORMATION_SCHEMA.DOMAINS.XML_SCHEMA_SCHEMA, or INFORMATION_SCHEMA.DOMAINS.XML_SCHEMA_NAME.

d) Subclause 20.6, "ELEMENT_TYPES view":

i) Insert this CR Without Feature X160, "Basic Information Schema for registered XML Schemas", conforming SQL language shall not reference INFORMATION_SCHEMA.ELEMENT_TYPES.XML_SCHEMA_CATALOG, INFORMATION_SCHEMA.ELEMENT_TYPES.XML_SCHEMA_SCHEMA, or INFORMATION_SCHEMA.ELEMENT_TYPES.XML_SCHEMA_NAME.

e) Subclause 20.7, "FIELDS view":

i) Insert this CR Without Feature X160, "Basic Information Schema for registered XML Schemas", conforming SQL language shall not reference INFORMATION_SCHEMA.FIELDS.XML_SCHEMA_CATALOG, INFORMATION_SCHEMA.FIELDS.XML_SCHEMA_SCHEMA, or INFORMATION_SCHEMA.FIELDS.XML_SCHEMA_NAME.

f) Subclause 20.8, "METHOD_SPECIFICATION_PARAMETERS view":

i) Insert this CR Without Feature X160, "Basic Information Schema for registered XML Schemas", conforming SQL language shall not reference INFORMATION_SCHEMA.METHOD_SPECIFICATION_PARAMETERS.XML_SCHEMA_CATALOG, INFORMATION_SCHEMA.METHOD_SPECIFICATION_PARAMETERS.XML_SCHEMA_SCHEMA, or INFORMATION_SCHEMA.METHOD_SPECIFICATION_PARAMETERS.XML_SCHEMA_NAME.

g) Subclause 20.9, "METHOD_SPECIFICATIONS view":

i) Insert this CR Without Feature X160, "Basic Information Schema for registered XML Schemas", conforming SQL language shall not reference INFORMATION_SCHEMA.METHOD_SPECIFICATIONS.XML_SCHEMA_CATALOG, INFORMATION_SCHEMA.METHOD_SPECIFICATIONS.XML_SCHEMA_SCHEMA, INFORMATION_SCHEMA.METHOD_SPECIFICATIONS.XML_SCHEMA_NAME, INFORMATION_SCHEMA.METHOD_SPECIFICATION_PARAMETERS.RESULT_CAST_XML_SCHEMA_CATALOG, INFORMATION_SCHEMA.METHOD_SPECIFICATION_PARAMETERS.RESULT_CAST_XML_SCHEMA_SCHEMA, or INFORMATION_SCHEMA.METHOD_SPECIFICATION_PARAMETERS.RESULT_CAST_XML_SCHEMA_NAME.

h) Subclause 20.10, "PARAMETERS view":

i) Insert this CR Without Feature X160, "Basic Information Schema for registered XML Schemas", conforming SQL language shall not reference reference INFORMATION_SCHEMA.PARAMETERS.XML_SCHEMA_CATALOG, INFORMATION_SCHEMA.PARAME-

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

TERS.XML_SCHEMA_SCHEMA, or INFORMATION_SCHEMA.PARAMETERS.XML_SCHEMA_NAME.

i) Subclause 20.11, "ROUTINES view":

i) Insert this CR Without Feature X160, "Basic Information Schema for registered XML Schemas", conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINES.XML_SCHEMA_CATALOG, INFORMATION_SCHEMA.ROUTINES.XML_SCHEMA_SCHEMA, INFORMATION_SCHEMA.ROUTINES.XML_SCHEMA_NAME, NFORMATION_SCHEMA.ROUTINES.RESULT_CAST_XML_SCHEMA_CATALOG, INFORMATION_SCHEMA.ROUTINES.RESULT_CAST_XML_SCHEMA_SCHEMA, or INFORMATION_SCHEMA.ROUTINES.RESULT_CAST_XML_SCHEMA_NAME.

j) Subclause 20.14, "XML_SCHEMAS view":

i) Without Feature X160, "Basic Information Schema for registered XML Schemas", conforming SQL language shall not reference INFORMATION_SCHEMA.XML_SCHEMAS.

k) Subclause 20.15, "Short name views":

i) Insert this CR Without Feature X160, "Basic Information Schema for registered XML Schemas", conforming SQL language shall not reference INFORMATION_SCHEMA.XML_SCHEMAS_S.

92) Specifications for Feature X161, "Advanced Information Schema for registered XML Schemas":

a) Subclause 20.3, "ATTRIBUTES view":

i) Insert this CR Without Feature X161, "Advanced Information Schema for registered XML Schemas", conforming SQL language shall not reference INFORMATION_SCHEMA.ATTRIBUTES.XML_SCHEMA_NAMESPACE or INFORMATION_SCHEMA.ATTRIBUTES.XML_SCHEMA_ELEMENT.

b) Subclause 20.4, "COLUMNS view":

i) Insert this CR Without Feature X161, "Advanced Information Schema for registered XML Schemas", conforming SQL language shall not reference INFORMATION_SCHEMA.COLUMNS.XML_SCHEMA_NAMESPACE or INFORMATION_SCHEMA.COLUMNS.XML_SCHEMA_ELEMENT.

c) Subclause 20.5, "DOMAINS view":

i) Insert this CR Without Feature X161, "Advanced Information Schema for registered XML Schemas", conforming SQL language shall not reference INFORMATION_SCHEMA.DOMAINS.XML_SCHEMA_NAMESPACE or INFORMATION_SCHEMA.DOMAINS.XML_SCHEMA_ELEMENT.

d) Subclause 20.6, "ELEMENT_TYPES view":

i) Insert this CR Without Feature X161, "Advanced Information Schema for registered XML Schemas", conforming SQL language shall not reference INFORMATION_SCHEMA.ELEMENT_TYPES.XML_SCHEMA_NAMESPACE or INFORMATION_SCHEMA.ELEMENT_TYPES.XML_SCHEMA_ELEMENT.

e) Subclause 20.7, "FIELDS view":

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

i) Insert this CR Without Feature X161, “Advanced Information Schema for registered XML Schemas”, conforming SQL language shall not reference INFORMATION_SCHEMA.FIELDS.XML_SCHEMA_NAMESPACE or INFORMATION_SCHEMA.FIELDS.XML_SCHEMA_ELEMENT.

f) Subclause 20.8, “METHOD_SPECIFICATION_PARAMETERS view”:

i) Insert this CR Without Feature X161, “Advanced Information Schema for registered XML Schemas”, conforming SQL language shall not reference INFORMATION_SCHEMA.METHOD_SPECIFICATION_PARAMETERS.XML_SCHEMA_NAMESPACE or INFORMATION_SCHEMA.METHOD_SPECIFICATION_PARAMETERS.XML_SCHEMA_ELEMENT.

g) Subclause 20.9, “METHOD_SPECIFICATIONS view”:

i) Insert this CR Without Feature X161, “Advanced Information Schema for registered XML Schemas”, conforming SQL language shall not reference INFORMATION_SCHEMA.METHOD_SPECIFICATIONS.XML_SCHEMA_NAMESPACE, INFORMATION_SCHEMA.METHOD_SPECIFICATIONS.XML_SCHEMA_ELEMENT, INFORMATION_SCHEMA.METHOD_SPECIFICATIONS.RESULT_CAST_XML_SCHEMA_NAMESPACE, or INFORMATION_SCHEMA.METHOD_SPECIFICATIONS.RESULT_CAST_XML_SCHEMA_ELEMENT.

h) Subclause 20.10, “PARAMETERS view”:

i) Insert this CR Without Feature X161, “Advanced Information Schema for registered XML Schemas”, conforming SQL language shall not reference reference INFORMATION_SCHEMA.PARAMETERS.XML_SCHEMA_NAMESPACE or INFORMATION_SCHEMA.PARAMETERS.XML_SCHEMA_ELEMENT.

i) Subclause 20.11, “ROUTINES view”:

i) Insert this CR Without Feature X161, “Advanced Information Schema for registered XML Schemas”, conforming SQL language shall not reference INFORMATION_SCHEMA.ROUTINES.XML_SCHEMA_NAMESPACE, INFORMATION_SCHEMA.ROUTINES.XML_SCHEMA_ELEMENT, INFORMATION_SCHEMA.ROUTINES.RESULT_CAST_XML_SCHEMA_NAMESPACE, or INFORMATION_SCHEMA.ROUTINES.RESULT_CAST_XML_SCHEMA_ELEMENT.

j) Subclause 20.12, “XML_SCHEMA_ELEMENTS view”:

i) Without Feature X161, “Advanced Information Schema for registered XML Schemas”, conforming SQL language shall not reference INFORMATION_SCHEMA.XML_SCHEMA_ELEMENTS.

k) Subclause 20.13, “XML_SCHEMA_NAMESPACES view”:

i) Without Feature X161, “Advanced Information Schema for registered XML Schemas”, conforming SQL language shall not reference INFORMATION_SCHEMA.XML_SCHEMA_NAMESPACES.

l) Subclause 20.14, “XML_SCHEMAS view”:

--`,,```,,,,````-`-`,,`,,`,`,,`---

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

i) Without Feature X161, “Advanced Information Schema for registered XML Schemas”, conforming SQL language shall not reference INFORMATION_SCHEMA.XML_SCHEMAS.SCHEMA_IS_DETERMINISTIC.

m) Subclause 20.15, “Short name views”:

i) Insert this CR Without Feature X161, “Advanced Information Schema for registered XML Schemas”, conforming SQL language shall not reference INFORMATION_SCHEMA.XML_SCH_ELEMENTS_S.

ii) Insert this CR Without Feature X161, “Advanced Information Schema for registered XML Schemas”, conforming SQL language shall not reference INFORMATION_SCHEMA.XML_SCH_NSPACES_S.

iii) Insert this CR Without Feature X161, “Advanced Information Schema for registered XML Schemas”, conforming SQL language shall not reference INFORMATION_SCHEMA.XML_SCHEMAS_S.XML_SCHEMA_IS_DET.

93) Specifications for Feature X170, “XML null handling options”:

a) Subclause 6.14, “<XML element>”:

i) Without Feature X170, “XML null handling options”, conforming SQL language shall not specify <XML content option>.

94) Specifications for Feature X171, “NIL ON NO CONTENT option”:

a) Subclause 6.14, “<XML element>”:

i) Without Feature X171, “NIL ON NO CONTENT option”, conforming SQL language shall not specify NIL ON NO CONTENT.

95) Specifications for Feature X181, “XML(DOCUMENT(UNTYPED)) type”:

a) Subclause 6.1, “<data type>”:

i) Insert this CR Without Feature X181, “XML(DOCUMENT(UNTYPED)) type”, conforming SQL language shall not contain an <XML type> whose <primary XML type modifier> is DOCUMENT and <secondary XML type modifier> is UNTYPED.

96) Specifications for Feature X182, “XML(DOCUMENT(ANY)) type”:

a) Subclause 6.1, “<data type>”:

i) Insert this CR Without Feature X182, “XML(DOCUMENT(ANY)) type”, conforming SQL language shall not contain an <XML type> whose <primary XML type modifier> is DOCUMENT and <secondary XML type modifier> is ANY.

97) Specifications for Feature X190, “XML(SEQUENCE) type”:

a) Subclause 6.1, “<data type>”:

i) Insert this CR Without Feature X190, “XML(SEQUENCE) type”, conforming SQL language shall not contain an <XML type> whose <XML type modifier> is SEQUENCE.

98) Specifications for Feature X191, “XML(DOCUMENT(XMLSCHEMA)) type”:

a) Subclause 6.1, “<data type>”:

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

i) Insert this CR Without Feature X191, "XML(DOCUMENT(XMLSCHEMA)) type", conforming SQL language shall not contain an <XML type> whose <primary XML type modifier> is DOCUMENT and <secondary XML type modifier> specifies XMLSCHEMA.

99) Specifications for Feature X192, "XML(CONTENT(XMLSCHEMA)) type":

a) Subclause 6.1, "<data type>":

i) Insert this CR Without Feature X192, "XML(CONTENT(XMLSCHEMA)) type", conforming SQL language shall not contain an <XML type> whose <primary XML type modifier> is CONTENT and <secondary XML type modifier> specifies XMLSCHEMA.

100) Specifications for Feature X200, "XMLQuery":

a) Subclause 6.18, "<XML query>":

i) Without Feature X200, "XMLQuery", conforming SQL language shall not contain an <XML query>.

101) Specifications for Feature X201, "XMLQuery: RETURNING CONTENT":

a) Subclause 6.18, "<XML query>":

i) Without Feature X201, "XMLQuery: RETURNING CONTENT", conforming SQL language shall not contain an <XML query> that contains RETURNING CONTENT.

102) Specifications for Feature X202, "XMLQuery: RETURNING SEQUENCE":

a) Subclause 6.18, "<XML query>":

i) Without Feature X202, "XMLQuery: RETURNING SEQUENCE", conforming SQL language shall not contain an <XML query> that contains RETURNING SEQUENCE.

103) Specifications for Feature X203, "XMLQuery: passing a context item":

a) Subclause 6.18, "<XML query>":

i) Without Feature X203, "XMLQuery: passing a context item", in conforming SQL language, an <XML query> shall not contain an <XML query context item>.

104) Specifications for Feature X204, "XMLQuery: initializing an XQuery variable":

a) Subclause 6.18, "<XML query>":

i) Without Feature X204, "XMLQuery: initializing an XQuery variable", in conforming SQL language, an <XML query> shall not contain an <XML query variable>.

105) Specifications for Feature X205, "XMLQuery: EMPTY ON EMPTY option":

a) Subclause 6.18, "<XML query>":

i) Without Feature X205, "XMLQuery: EMPTY ON EMPTY option", conforming SQL language shall not contain an <XML query> that contains an <XML query empty handling option> that specifies EMPTY ON EMPTY.

106) Specifications for Feature X206, "XMLQuery: NULL ON EMPTY option":

a) Subclause 6.18, "<XML query>":

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

i) Without Feature X206, “XMLQuery: NULL ON EMPTY option”, conforming SQL language shall not contain an <XML query> that contains an <XML query empty handling option> that specifies NULL ON EMPTY.

107) Specifications for Feature X211, “XML 1.1 support”:

a) Subclause 6.14, “<XML element>”:

i) Without Feature X211, “XML 1.1 support”, in conforming SQL language, an <XML element name> shall be an XML 1.0 QName.

ii) Without Feature X211, “XML 1.1 support”, in conforming SQL language, an <XML attribute name> shall be an XML 1.0 QName.

b) Subclause 6.15, “<XML forest>”:

i) Without Feature X211, “XML 1.1 support”, in conforming SQL language, a <forest element name> shall be an XML 1.0 QName.

c) Subclause 6.17, “<XML PI>”:

i) Without Feature X211, “XML 1.1 support”, in conforming SQL language, an <identifier> contained in an <XML PI target>, when mapped to Unicode, shall be an XML 1.0 NCName.

NOTE 79 — The set of XML 1.0 NCNames is a proper subset of the set of XML 1.1 NCNames. That is, all XML 1.0 NCNames are also XML 1.1 NCNames, but not all XML 1.1 NCNames are also XML 1.0 NCNames.

d) Subclause 6.18, “<XML query>”:

i) Without Feature X211, “XML 1.1 support”, in conforming SQL language, the value of the <XQuery expression> shall be an XQuery expression with XML 1.0 lexical rules.

ii) Without Feature X211, “XML 1.1 support”, in conforming SQL language, the <identifier> contained in an <XML query variable> shall be an XML 1.0 NCName.

e) Subclause 11.3, “<XML lexically scoped options>”:

i) Without Feature X211, “XML 1.1 support”, in conforming SQL language, each <XML namespace prefix> shall be an XML 1.0 NCName.

108) Specifications for Feature X221, “XML passing mechanism BY VALUE”:

a) Subclause 11.5, “<XML passing mechanism>”:

i) Without Feature X221, “XML passing mechanism BY VALUE”, conforming SQL language shall not contain an <XML passing mechanism> that is BY VALUE.

109) Specifications for Feature X222, “XML passing mechanism BY REF”:

a) Subclause 11.5, “<XML passing mechanism>”:

i) Without Feature X222, “XML passing mechanism BY REF”, conforming SQL language shall not contain an <XML passing mechanism> that is BY REF.

110) Specifications for Feature X231, “XML(CONTENT(UNTYPED)) type”:

a) Subclause 6.1, “<data type>”:

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

i) Insert this CR Without Feature X231, “XML(CONTENT(UNTYPED)) type”, conforming SQL language shall not contain an <XML type> whose <primary XML type modifier> is CONTENT and <secondary XML type modifier> is UNTYPED.

111) Specifications for Feature X232, “XML(CONTENT(ANY)) type”:

a) Subclause 6.1, “<data type>”:

i) Insert this CR Without Feature X232, “XML(CONTENT(ANY)) type”, conforming SQL language shall not contain an <XML type> whose <primary XML type modifier> is CONTENT and <secondary XML type modifier> is ANY.

112) Specifications for Feature X241, “RETURNING CONTENT in XML publishing”:

a) Subclause 6.11, “<XML comment>”:

i) Without Feature X241, “RETURNING CONTENT in XML publishing”, in conforming SQL language, an <XML comment> shall not specify an <XML returning clause> that is RETURNING CONTENT.

b) Subclause 6.12, “<XML concatenation>”:

i) Without Feature X241, “RETURNING CONTENT in XML publishing”, in conforming SQL language, an <XML concatenation> shall not specify an <XML returning clause> that is RETURNING CONTENT.

c) Subclause 6.13, “<XML document>”:

i) Without Feature X241, “RETURNING CONTENT in XML publishing”, in conforming SQL language, an <XML document> shall not specify an <XML returning clause> that is RETURNING CONTENT.

d) Subclause 6.14, “<XML element>”:

i) Without Feature X241, “RETURNING CONTENT in XML publishing”, in conforming SQL language, an <XML element> shall not specify an <XML returning clause> that is RETURNING CONTENT.

e) Subclause 6.15, “<XML forest>”:

i) Without Feature X241, “RETURNING CONTENT in XML publishing”, in conforming SQL language, an <XML forest> shall not specify an <XML returning clause> that is RETURNING CONTENT.

f) Subclause 6.17, “<XML PI>”:

i) Without Feature X241, “RETURNING CONTENT in XML publishing”, in conforming SQL language, an <XML PI> shall not specify an <XML returning clause> that is RETURNING CONTENT.

g) Subclause 6.19, “<XML text>”:

i) Without Feature X241, “RETURNING CONTENT in XML publishing”, in conforming SQL language, an <XML text> shall not specify an <XML returning clause> that is RETURNING CONTENT.

h) Subclause 11.2, “<aggregate function>”:

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

i) Insert this CR Without Feature X241, “RETURNING CONTENT in XML publishing”, in conforming SQL language, an <XML aggregate> shall not specify an <XML returning clause> that is RETURNING CONTENT.

113) Specifications for Feature X242, “RETURNING SEQUENCE in XML publishing”:

a) Subclause 6.11, “<XML comment>”:

i) Without Feature X242, “RETURNING SEQUENCE in XML publishing”, in conforming SQL language, an <XML comment> shall not specify an <XML returning clause> that is RETURNING SEQUENCE.

b) Subclause 6.12, “<XML concatenation>”:

i) Without Feature X242, “RETURNING SEQUENCE in XML publishing”, in conforming SQL language, an <XML concatenation> shall not specify an <XML returning clause> that is RETURNING SEQUENCE.

c) Subclause 6.13, “<XML document>”:

i) Without Feature X242, “RETURNING SEQUENCE in XML publishing”, in conforming SQL language, an <XML document> shall not specify an <XML returning clause> that is RETURNING SEQUENCE.

d) Subclause 6.14, “<XML element>”:

i) Without Feature X242, “RETURNING SEQUENCE in XML publishing”, in conforming SQL language, an <XML element> shall not specify an <XML returning clause> that is RETURNING SEQUENCE.

e) Subclause 6.15, “<XML forest>”:

i) Without Feature X242, “RETURNING SEQUENCE in XML publishing”, in conforming SQL language, an <XML forest> shall not specify an <XML returning clause> that is RETURNING SEQUENCE.

f) Subclause 6.17, “<XML PI>”:

i) Without Feature X242, “RETURNING SEQUENCE in XML publishing”, in conforming SQL language, an <XML PI> shall not specify an <XML returning clause> that is RETURNING SEQUENCE.

g) Subclause 6.19, “<XML text>”:

i) Without Feature X242, “RETURNING SEQUENCE in XML publishing”, in conforming SQL language, an <XML text> shall not specify an <XML returning clause> that is RETURNING SEQUENCE.

h) Subclause 11.2, “<aggregate function>”:

i) Insert this CR Without Feature X242, “RETURNING SEQUENCE in XML publishing”, in conforming SQL language, an <XML aggregate> shall not specify an <XML returning clause> that is RETURNING SEQUENCE.

114) Specifications for Feature X251, “Persistent XML values of XML(DOCUMENT(UNTYPED)) type”:

a) Subclause 12.1, “<column definition>”:

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

i) Insert this CR Without Feature X251, "Persistent XML values of XML(DOCUMENT(UNTYPED)) type", conforming SQL language shall not contain a <column definition> whose declared type is based on either the XML(DOCUMENT(UNTYPED)) type or a distinct type whose source type is the XML(DOCUMENT(UNTYPED)) type.

b) Subclause 12.3, "<view definition>":

i) Insert this CR Without Feature X251, "Persistent XML values of XML(DOCUMENT(UNTYPED)) type", conforming SQL language shall not contain a <view definition> that defines a column whose declared type is based on either the XML(DOCUMENT(UNTYPED)) type or a distinct type whose source type is the XML(DOCUMENT(UNTYPED)) type.

115) Specifications for Feature X252, "Persistent XML values of XML(DOCUMENT(ANY)) type":

a) Subclause 12.1, "<column definition>":

i) Insert this CR Without Feature X252, "Persistent XML values of XML(DOCUMENT(ANY)) type", conforming SQL language shall not contain a <column definition> whose declared type is based on either the XML(DOCUMENT(ANY)) type or a distinct type whose source type is the XML(DOCUMENT(ANY)) type.

b) Subclause 12.3, "<view definition>":

i) Insert this CR Without Feature X252, "Persistent XML values of XML(DOCUMENT(ANY)) type", conforming SQL language shall not contain a <view definition> that defines a column whose declared type is based on either the XML(DOCUMENT(ANY)) type or a distinct type whose source type is the XML(DOCUMENT(ANY)) type.

116) Specifications for Feature X253, "Persistent XML values of XML(CONTENT(UNTYPED)) type":

a) Subclause 12.1, "<column definition>":

i) Insert this CR Without Feature X253, "Persistent XML values of XML(CONTENT(UNTYPED)) type", conforming SQL language shall not contain a <column definition> whose declared type is based on either the XML(CONTENT(UNTYPED)) type or a distinct type whose source type is the XML(CONTENT(UNTYPED)) type.

b) Subclause 12.3, "<view definition>":

i) Insert this CR Without Feature X253, "Persistent XML values of XML(CONTENT(UNTYPED)) type", conforming SQL language shall not contain a <view definition> that defines a column whose declared type is based on either the XML(CONTENT(UNTYPED)) type or a distinct type whose source type is the XML(CONTENT(UNTYPED)) type.

117) Specifications for Feature X254, "Persistent XML values of XML(CONTENT(ANY)) type":

a) Subclause 12.1, "<column definition>":

i) Insert this CR Without Feature X254, "Persistent XML values of XML(CONTENT(ANY)) type", conforming SQL language shall not contain a <column definition> whose declared type is based on either the XML(CONTENT(ANY)) type or a distinct type whose source type is the XML(CONTENT(ANY)) type.

b) Subclause 12.3, "<view definition>":

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

i) Insert this CR Without Feature X254, “Persistent XML values of XML(CONTENT(ANY)) type”, conforming SQL language shall not contain a <view definition> that defines a column whose declared type is based on either the XML(CONTENT(ANY)) type or a distinct type whose source type is the XML(CONTENT(ANY)) type.

118) Specifications for Feature X255, “Persistent XML values of XML(SEQUENCE) type”:

a) Subclause 12.1, “<column definition>”:

i) Insert this CR Without Feature X255, “Persistent XML values of XML(SEQUENCE) type”, conforming SQL language shall not contain a <column definition> whose declared type is based on either the XML(SEQUENCE) type or a distinct type whose source type is the XML(SEQUENCE) type.

b) Subclause 12.3, “<view definition>”:

i) Insert this CR Without Feature X255, “Persistent XML values of XML(SEQUENCE) type”, conforming SQL language shall not contain a <view definition> that defines a column whose declared type is based on either the XML(SEQUENCE) type or a distinct type whose source type is the XML(SEQUENCE) type.

119) Specifications for Feature X256, “Persistent XML values of XML(DOCUMENT(XMLSCHEMA)) type”:

a) Subclause 12.1, “<column definition>”:

i) Insert this CR Without Feature X256, “Persistent XML values of XML(DOCUMENT(XMLSCHEMA)) type”, conforming SQL language shall not contain a <column definition> whose declared type is based on either the XML(DOCUMENT(XMLSCHEMA)) type or a distinct type whose source type is the XML(DOCUMENT(XMLSCHEMA)) type.

b) Subclause 12.3, “<view definition>”:

i) Insert this CR Without Feature X256, “Persistent XML values of XML(DOCUMENT(XMLSCHEMA)) type”, conforming SQL language shall not contain a <view definition> that defines a column whose declared type is based on either the XML(DOCUMENT(XMLSCHEMA)) type or a distinct type whose source type is the XML(DOCUMENT(XMLSCHEMA)) type.

120) Specifications for Feature X257, “Persistent XML values of XML(CONTENT(XMLSCHEMA)) type”:

a) Subclause 12.1, “<column definition>”:

i) Insert this CR Without Feature X257, “Persistent XML values of XML(CONTENT(XMLSCHEMA)) type”, conforming SQL language shall not contain a <column definition> whose declared type is based on either the XML(CONTENT(XMLSCHEMA)) type or a distinct type whose source type is the XML(CONTENT(XMLSCHEMA)) type.

b) Subclause 12.3, “<view definition>”:

i) Insert this CR Without Feature X257, “Persistent XML values of XML(CONTENT(XMLSCHEMA)) type”, conforming SQL language shall not contain a <view definition> that defines a column whose declared type is based on either the XML(CONTENT(XMLSCHEMA)) type or a distinct type whose source type is the XML(CONTENT(XMLSCHEMA)) type.

121) Specifications for Feature X260, “XML type: ELEMENT clause”:

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

a) Subclause 6.1, "<data type>":

i) Insert this CR Without Feature X260, "XML type: ELEMENT clause", conforming SQL language shall not contain an <XML type> that contains <XML valid element clause>.

122) Specifications for Feature X261, "XML type: NAMESPACE without ELEMENT clause":

a) Subclause 6.1, "<data type>":

i) Insert this CR Without Feature X261, "XML type: NAMESPACE without ELEMENT clause", conforming SQL language shall not contain an <XML type> that contains an <XML valid element clause> that does not contain an <XML valid element name specification>.

123) Specifications for Feature X263, "XML type: NO NAMESPACE with ELEMENT clause":

a) Subclause 6.1, "<data type>":

i) Insert this CR Without Feature X263, "XML type: NO NAMESPACE with ELEMENT clause", conforming SQL language shall not contain an <XML type> that contains an <XML valid element namespace specification> that contains NO NAMESPACE.

124) Specifications for Feature X264, "XML type: schema location":

a) Subclause 6.1, "<data type>":

i) Insert this CR Without Feature X264, "XML type: schema location", conforming SQL language shall not contain an <XML type> that contains <XML valid schema location>.

125) Specifications for Feature X271, "XMLValidate: data-driven case":

a) Subclause 6.20, "<XML validate>":

i) Without Feature X271, "XMLValidate: data-driven case", conforming SQL language shall not contain an <XML validate> that does not contain <XML valid according to clause>.

126) Specifications for Feature X272, "XMLValidate: ACCORDING TO clause":

a) Subclause 6.20, "<XML validate>":

i) Without Feature X272, "XMLValidate: ACCORDING TO clause", conforming SQL language shall not contain an <XML validate> that contains <XML valid according to clause>.

127) Specifications for Feature X273, "XMLValidate: ELEMENT clause":

a) Subclause 6.20, "<XML validate>":

i) Without Feature X273, "XMLValidate: ELEMENT clause", conforming SQL language shall not contain an <XML validate> that contains <XML valid element clause>.

128) Specifications for Feature X274, "XMLValidate: schema location":

a) Subclause 6.20, "<XML validate>":

i) Without Feature X274, "XMLValidate: schema location", conforming SQL language shall not contain an <XML validate> that contains <XML valid schema location>.

129) Specifications for Feature X281, "XMLValidate with DOCUMENT option":

a) Subclause 6.20, "<XML validate>":

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

i) Without Feature X281, “XMLValidate with DOCUMENT option”, conforming SQL language shall not contain an <XML validate> that immediately contains a <document or content or sequence> that is DOCUMENT.

130) Specifications for Feature X282, “XMLValidate with CONTENT option”:

a) Subclause 6.20, “<XML validate>”:

i) Without Feature X282, “XMLValidate with CONTENT option”, conforming SQL language shall not contain an <XML validate> that immediately contains a <document or content or sequence> that is CONTENT.

131) Specifications for Feature X283, “XMLValidate with SEQUENCE option”:

a) Subclause 6.20, “<XML validate>”:

i) Without Feature X283, “XMLValidate with SEQUENCE option”, conforming SQL language shall not contain an <XML validate> that immediately contains a <document or content or sequence> that is SEQUENCE.

132) Specifications for Feature X284, “XMLValidate: NAMESPACE without ELEMENT clause”:

a) Subclause 6.20, “<XML validate>”:

i) Without Feature X284, “XMLValidate: NAMESPACE without ELEMENT clause”, conforming SQL language shall not contain an <XML validate> that contains an <XML valid element clause> that does not contain an <XML valid element name specification>.

133) Specifications for Feature X286, “XMLValidate: NO NAMESPACE with ELEMENT clause”:

a) Subclause 6.20, “<XML validate>”:

i) Without Feature X286, “XMLValidate: NO NAMESPACE with ELEMENT clause”, conforming SQL language shall not contain an <XML validate> that contains an <XML valid element namespace specification> that contains NO NAMESPACE.

134) Specifications for Feature X300, “XMLTable”:

a) Subclause 7.1, “<table reference>”:

i) Insert this CR Without Feature X300, “XMLTable”, conforming SQL language shall not contain <XML table>.

135) Specifications for Feature X301, “XMLTable: derived column list option”:

a) Subclause 7.1, “<table reference>”:

i) Insert this CR Without Feature X301, “XMLTable: derived column list option”, in conforming SQL language, a <table primary> that is an <XML table> shall not contain a <derived column list>.

136) Specifications for Feature X302, “XMLTable: ordinality column option”:

a) Subclause 7.1, “<table reference>”:

i) Insert this CR Without Feature X302, “XMLTable: ordinality column option”, in conforming SQL language, an <XML table> shall not contain an <XML table ordinality column definition>.

137) Specifications for Feature X303, “XMLTable: column default option”:

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

a) Subclause 7.1, "<table reference>":

i) Insert this CR Without Feature X303, "XMLTable: column default option", in conforming SQL language, an <XML table regular column definition> shall not contain a <default clause>.

138) Specifications for Feature X304, "XMLTable: passing a context item":

a) Subclause 7.1, "<table reference>":

i) Insert this CR Without Feature X304, "XMLTable: passing a context item", in conforming SQL language, an <XML table argument list> shall not contain an <XML query context item>.

139) Specifications for Feature X305, "XMLTable: initializing an XQuery variable":

a) Subclause 7.1, "<table reference>":

i) Insert this CR Without Feature X305, "XMLTable: initializing an XQuery variable", in conforming SQL language, an <XML table argument list> shall not contain an <XML query variable>.

140) Specifications for Feature X400, "Name and identifier mapping":

a) Subclause 9.3, "Mapping XML Names to SQL <identifier>s":

i) Without Feature X400, "Name and identifier mapping", a conforming application shall not invoke this Subclause of this part of this International Standard.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

(Blank page)

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex B
(informative)

Implementation-defined elements

This Annex modifies Annex B, "Implementation-defined elements", in ISO/IEC 9075-2.

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

1) Subclause 4.2.6, "Registered XML Schemas":

   a) XML Schemas are registered through implementation-defined means.

   b) The <registered XML Schema name>s and schema location URIs of the built-in XML Schemas are implementation-defined.

2) Subclause 4.7.1, "Privileges":

   a) USAGE privileges on registered XML Schemas are granted or revoked by implementation-defined means.

3) Subclause 4.8.1, "SQL-session properties":

   a) The XML option is initially set to an implementation-defined value, but can subsequently be changed by the successful execution of a <set XML option statement>.

4) Subclause 4.10.1, "Mapping SQL character sets to Unicode":

   a) The mapping of an SQL character set to Unicode is implementation-defined.

   b) The choice of base64 or hex as the default encoding of binary strings is imlementation-defined.

   c) For <XML element> and <XML forest>, the choice of whether the default mapping is to **`xs:hexBinary`** or **`xs:base64Binary`** is implementation-defined.

5) Subclause 4.10.2, "Mapping Unicode to SQL character sets":

   a) The mapping of Unicode to a character set in the SQL-environment is implementation-defined.

6) Subclause 4.10.5, "Mapping SQL data types to XML":

   a) The mapping of numeric SQL types INTEGER, SMALLINT, and BIGINT to XML Schema types is implementation-defined.

7) Subclause 4.10.6, "Mapping values of SQL data types to XML":

   a) The maximum length of variable-length character strings implementation-defined.

   b) The mapping of the default of CHARACTER VARYING strings to Unicode is implementation-defined.

8) Subclause 4.10.7, "Mapping XQuery atomic values to SQL values":

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

a) The mapping of a value of XML Schema type **`xs:dateTime`** that denotes a value of UTC in a minute with exactly 59 seconds, and that has a SECOND field that is greater than or equal to 59, is implementation-defined.

9) Subclause 4.10.9, "Mapping an SQL table to XML":

a) It is implementation-defined whether annotations are generated to represent the SQL metadata that is not directly relevant to XML.

10) Subclause 4.10.10, "Mapping an SQL schema to XML":

a) It is implementation-defined whether annotations are generated to represent the SQL metadata that is not directly relevant to XML.

11) Subclause 4.10.11, "Mapping an SQL catalog to XML":

a) It is implementation-defined whether annotations are generated to represent the SQL metadata that is not directly relevant to XML.

12) Subclause 6.1, "<data type>":

a) Whether XML defaults to XML(SEQUENCE)), XML(CONTENT(UNTYPED)), or XML(CONTENT(ANY)) is implementation-defined.

b) Whether XML(CONTENT) defaults to XML(CONTENT(UNTYPED)) or XML(CONTENT(ANY)) is implementation-defined.

c) Whether XML(DOCUMENT) defaults to XML(DOCUMENT(UNTYPED)) or XML(DOCUMENT(ANY)) is implementation-defined.

13) Subclause 6.5, "<cast specification>":

a) If <XML passing mechanism> is not specified, then the implicit <XML passing mechanism> is implementation-defined.

14) Subclause 6.6, "<XML cast specification>":

a) The default encoding of binary strings (either base64 or hex) is implementation-defined.

b) If <XML passing mechanism> is not specified, then the implicit <XML passing mechanism> is implementation-defined.

15) Subclause 6.7, "<value expression>":

a) Implementation-defined rules that allow an SQL-implementation to determine whether an <XML query> is not possibly non-deterministic are permissible.

b) Implementation-defined rules that allow an SQL-implementation to determine whether an <XML exists predicate> is not possibly non-deterministic are permissible.

16) Subclause 6.8, "<string value function>":

a) The default <XML serialize version> for an <XML character string serialization> is implementation-defined.

b) The <XML encoding name>s supported for <XML binary string serialization> are implementation-defined.

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

c) If an <XML encoding specification> is not specified, then the implicit <XML encoding name> is implementation-defined.

d) The default <XML serialize version> for an <XML binary string serialization> is implementation-defined.

17) Subclause 6.11, "<XML comment>":

a) The function used to convert the content of a comment to Unicode is implementation-defined.

b) If <XML returning clause> is not specified, then the implicit <XML returning clause> is implementation-defined.

18) Subclause 6.12, "<XML concatenation>":

a) If <XML returning clause> is not specified, then the implicit <XML returning clause> is implementation-defined.

19) Subclause 6.13, "<XML document>":

a) If <XML returning clause> is not specified, then it is implementation-defined whether RETURNING CONTENT or RETURNING SEQUENCE is implicit.

b) If RETURNING CONTENT is specified or implied, then it is implementation-defined whether the declared type of <XML document> is XML(CONTENT(UNTYPED)) or XML(CONTENT(ANY)).

c) The **base-uri** property of the generated XQuery document node is implementation-defined.

20) Subclause 6.14, "<XML element>":

a) If <XML returning clause> is not specified, then the implicit <XML returning clause> is implementation-defined.

b) The choice of base64 or hex as the default encoding of binary strings is implementation-defined.

21) Subclause 6.15, "<XML forest>":

a) If <XML returning clause> is not specified, then the implicit <XML returning clause> is implementation-defined.

b) The choice of base64 or hex as the default encoding of binary strings is implementation-defined.

22) Subclause 6.16, "<XML parse>"

a) If <document or content> is DOCUMENT, then it is implementation-defined whether the declared type of <XML parse> is XML(DOCUMENT(UNTYPED)) or XML(DOCUMENT(ANY)).

b) If <document or content> is CONTENT, then it is implementation-defined whether the declared type of <XML parse> is XML(CONTENT(UNTYPED)) or XML(CONTENT(ANY)).

23) Subclause 6.17, "<XML PI>":

a) The function used to convert the content of a processing instruction to Unicode is implementation-defined.

b) The **base-uri** property of the generated XQuery processing instruction node is implementation-defined.

c) If <XML returning clause> is not specified, then the implicit <XML returning clause> is implementation-defined.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

24) Subclause 6.18, "<XML query>":

a) The subset of normative rules of XQuery expressions that are found in [XQuery], [XQueryFO], [XQueryFS], and [XQueryUpdate] to which an <XQuery expression> should conform is implementation-defined.

b) If <XML returning clause> is not specified, then the implicit <XML returning clause> is implementation-defined.

25) Subclause 6.19, "<XML text>":

a) If <XML returning clause> is not specified, then it is implementation-defined whether RETURNING CONTENT or RETURNING SEQUENCE is implicit.

b) The function used to convert the text to Unicode is implementation-defined.

26) Subclause 6.20, "<XML validate>"

a) If the <XML validate> does not specify <XML valid according to clause>, then the registered XML Schema used to assess validity of a top-level element *E* of the XML value is chosen from among those registered XML Schemas for which the user has USAGE privilege and that have a global element declaration schema component whose namespace is the namespace of *E*, using a deterministic implementation-defined algorithm that is repeatable, in the sense that if the algorithm is re-evaluated with the same collection of registered XML Schemas that are accessible to the user and the same element *E*, then the same registered XML Schema will be chosen.

27) Subclause 7.1, "<table reference>":

a) Implementation-defined rules that allow an SQL-implementation to determine whether an <XML table> is not possibly non-deterministic are permissible.

28) Subclause 8.5, "<XML valid predicate>":

a) If the <XML valid predicate> does not specify <XML valid according to clause>, then the registered XML Schema used to assess validity of a top-level element *E* of the XML value is chosen from among those registered XML Schemas for which the user has USAGE privilege and that have a global element declaration schema component whose namespace is the namespace of *E*, using a deterministic implementation-defined algorithm that is repeatable, in the sense that if the algorithm is re-evaluated with the same collection of registered XML Schemas that are accessible to the user and the same element *E*, then the same registered XML Schema will be chosen.

29) Subclause 9.1, "Mapping SQL <identifier>s to XML Names":

a) If *S* is a character in an SQL <identifier> *SQLI* and *S* has no mapping to Unicode, then the mapping of *S* to create an XML Name corresponding to *SQLI* is implementation-defined.

30) Subclause 9.3, "Mapping XML Names to SQL <identifier>s":

a) The treatment of an escape sequence of the form _x*NNNN*_ or _x*NNNNNN*_ whose corresponding Unicode code point U+*NNNN* or U+*NNNNNN* is not a Unicode assigned character is implementation-defined.

31) Subclause 9.5, "Mapping SQL data types to XML Schema data types":

a) The mapping of numeric SQL types INTEGER, SMALLINT, and BIGINT to XML Schema types is implementation-defined.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

32) Subclause 9.8, "Mapping values of SQL data types to values of XML Schema data types":

a) The mapping of a TIME or TIMESTAMP value whose SECOND field is greater than or equal to 60 is implementation-defined.

b) It is implementation-defined whether the specified annotations exist or are zero-length strings.

33) Subclause 9.9, "Mapping an SQL table to XML Schema data types":

a) It is implementation-defined whether the specified annotations exist or are zero-length strings.

34) Subclause 9.11, "Mapping an SQL table to XML and an XML Schema document":

a) If the SQL-implementation supports Feature X211, "XML 1.1 support", then the version of XML that is generated may be implementation-defined.

b) The encoding of the results is implementation-defined.

c) The presence and value of the **schemaLocation** hint of the **xs:import** in the generated XML Schema document is implementation-defined.

35) Subclause 9.12, "Mapping an SQL schema to XML Schema data types":

a) It is implementation-defined whether the specified annotations exist or are zero-length strings.

36) Subclause 9.14, "Mapping an SQL schema to an XML document and an XML Schema document":

a) If the SQL-implementation supports Feature X211, "XML 1.1 support", then the version of XML that is generated may be implementation-defined.

b) The encoding of the results is implementation-defined.

c) The presence and value of the **schemaLocation** hint of the **xs:import** in the generated XML Schema document is implementation-defined.

37) Subclause 9.15, "Mapping an SQL catalog to XML Schema data types":

a) It is implementation-defined whether the specified annotations exist or are zero-length strings.

38) Subclause 9.17, "Mapping an SQL catalog to an XML document and an XML Schema document":

a) If the SQL-implementation supports Feature X211, "XML 1.1 support", then the version of XML that is generated may be implementation-defined.

b) The encoding of the results is implementation-defined.

c) The presence and value of the **schemaLocation** hint of the **xs:import** in the generated XML Schema document is implementation-defined.

39) Subclause 10.13, "Construction of an XML element":

a) The validation mode is set to an implementation-defined value.

40) Subclause 10.15, "Serialization of an XML value":

a) If *DT* is a binary string type, *CS* is not UTF16, and *B* is *Unknown*, then it is implementation-defined whether ***BOM*** is **yes** or **no**.

b) It is implementation-defined whether ***SA*** is **yes**, **no**, or **none**.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

c) If *DECL* is *Unknown*, *DC* is DOCUMENT, *ENCODING* is either UTF8 or UTF16, *VERSION* is "1.0", and ***SA*** is **none**, then it is implementation-defined whether ***OXD*** is **yes** or **no**.

d) If *VP* is not 1.0, then it is implementation-defined whether ***UN*** is **yes** or **no**.

41) Subclause 10.16, "Parsing a string as an XML value":

a) Support for the **[notations]** property and the **[unparsed entities]** property of the XML document information item is implementation-defined.

b) Support for external DTDs is implementation-defined.

42) Subclause 10.20, "Creation of an XQuery expression context":

a) The XQuery static and dynamic context may be initialized with implementation-defined values, as permitted by [XQuery].

43) Subclause 10.21, "Determination of an XQuery formal type notation":

a) The XQuery formal type notation of an XQuery variable may denote an implementation-defined subtype of the type specified in this Subclause.

b) The SQL-implementation may use an implementation-defined rule to deduce that the value of a <value expression> cannot be null.

44) Subclause 11.1, "<routine invocation>":

a) It is implementation-defined whether XDO_i is INCLUDING XMLDECLARATION or EXCLUDING XMLDECLARATION.

b) It is implementation-defined whether *XWO* is STRIP WHITESPACE or PRESERVE WHITESPACE.

c) It is implementation-defined whether XWO_i is STRIP WHITESPACE or PRESERVE WHITESPACE.

45) Subclause 11.2, "<aggregate function>":

a) If <XML returning clause> is not specified, then the implicit <XML returning clause> is implementation-defined.

46) Subclause 11.6, "<XML valid according to clause>":

a) It is implementation-defined whether the indicated registered XML Schema shall have the indicated XML namespace among its unordered collection of namespaces.

b) If <XML valid element name> is specified, then it is implementation-defined whether the indicated registered XML Schema shall have a global element declaration schema component whose namespace is the indicated XML namespace and whose XML NCName is the <XML valid element name>.

c) If <XML valid according to URI> is specified and there exist more than one registered XML Schema whose target namespace URIs and possibly schema location URIs match that of the specified <XML valid target namespace URI> and <XML valid schema location URI>, if any, then a <registered XML Schema name> is chosen by a deterministic implementation-defined algorithm that is repeatable in the sense that if the algorithm is reevaluated with the same collection of registered XML Schemas for which the user has USAGE privilege, then the same <registered XML Schema name> will be chosen.

47) Subclause 12.7, "<SQL-invoked routine>":

 © ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

a) If *R* is an SQL-invoked routine, the declared type of an <SQL parameter declaration> is an XML type or a distinct type whose source type is an XML type, and the <SQL parameter declaration> does not contain an <XML passing mechanism>, then it is implementation-defined whether BY REF or BY VALUE is implicit.

b) If *R* is an SQL-invoked routine that is an SQL-invoked function, the declared type of the <returns type> is an XML type or a distinct type whose source type is an XML type, and the <returns clause> does not contain an <XML passing mechanism>, then it is implementation-defined whether BY REF or BY VALUE is implicit.

48) Subclause 14.1, "<fetch statement>":

a) If <XML passing mechanism> is not specified, then the implicit <XML passing mechanism> is implementation-defined.

49) Subclause 14.2, "<select statement: single row>":

a) If <XML passing mechanism> is not specified, then the implicit <XML passing mechanism> is implementation-defined.

50) Subclause 15.2, "<assignment statement>":

a) If <XML passing mechanism> is not specified, then the implicit <XML passing mechanism> is implementation-defined.

51) Subclause 17.2, "<input using clause>":

a) It is implementation-defined whether *XWO* is STRIP WHITESPACE or PRESERVE WHITESPACE.

52) Subclause 17.3, "<output using clause>":

a) It is implementation-defined whether *XDO* is INCLUDING XMLDECLARATION or EXCLUDING XMLDECLARATION.

53) Subclause 17.4, "<prepare statement>":

a) The inferred type *DT* of an <XML value expression> simply contained in <XML character string serialization> , <XML binary string serialization>, <XML concatenation>, <XML document>, <XML validate>, <XML content predicate>, <XML document predicate>, or <XML valid predicate> is an implementation-defined XML type.

54) Subclause 18.1, "<embedded SQL host program>":

a) It is implementation-defined whether $XDOI_j$ is INCLUDING XMLDECLARATION or EXCLUDING XMLDECLARATION.

b) It is implementation-defined whether $XDOO_k$ is STRIP WHITESPACE or PRESERVE WHITESPACE.

c) It is implementation-defined whether $XDOIO_l$ is STRIP WHITESPACE or PRESERVE WHITESPACE.

55) Subclause 18.2, "<embedded SQL Ada program>":

a) If *XO* is the zero-length string, then it is implementation-defined whether DOCUMENT or CONTENT is the *XML option of XML CLOB host variable*.

b) If *XWO* is the zero-length string, then it is implementation-defined whether STRIP WHITESPACE or PRESERVE WHITESPACE is the *XML whitespace option of the XML CLOB host variable*.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

c) If *XO* is the zero-length string, then it is implementation-defined whether DOCUMENT or CONTENT is the *XML option of XML BLOB host variable*.

d) If *XWO* is the zero-length string, then it is implementation-defined whether STRIP WHITESPACE or PRESERVE WHITESPACE is the *XML whitespace option of the XML BLOB host variable*.

56) Subclause 18.3, "<embedded SQL C program>":

a) The implicit character set in a <C XML VARCHAR variable>, or a <C XML CLOB variable> is implementation-defined.

b) If *XO* is the zero-length string, then it is implementation-defined whether DOCUMENT or CONTENT is the *XML option of XML VARCHAR host variable*.

c) If *XWO* is the zero-length string, then it is implementation-defined whether STRIP WHITESPACE or PRESERVE WHITESPACE is the *XML whitespace option of the XML VARCHAR host variable*.

d) If *XO* is the zero-length string, then it is implementation-defined whether DOCUMENT or CONTENT is the *XML option of XML CLOB host variable*.

e) If *XWO* is the zero-length string, then it is implementation-defined whether STRIP WHITESPACE or PRESERVE WHITESPACE is the *XML whitespace option of the XML CLOB host variable*.

f) If *XO* is the zero-length string, then it is implementation-defined whether DOCUMENT or CONTENT is the *XML option of XML BLOB host variable*.

g) If *XWO* is the zero-length string, then it is implementation-defined whether STRIP WHITESPACE or PRESERVE WHITESPACE is the *XML whitespace option of the XML BLOB host variable*.

57) Subclause 18.4, "<embedded SQL COBOL program>":

a) The implicit character set in a <COBOL XML CLOB variable> is implementation-defined.

b) If *XO* is the zero-length string, then it is implementation-defined whether DOCUMENT or CONTENT is the *XML option of XML CLOB host variable*.

c) If *XWO* is the zero-length string, then it is implementation-defined whether STRIP WHITESPACE or PRESERVE WHITESPACE is the *XML whitespace option of the XML CLOB host variable*.

d) If *XO* is the zero-length string, then it is implementation-defined whether DOCUMENT or CONTENT is the *XML option of XML BLOB host variable*.

e) If *XWO* is the zero-length string, then it is implementation-defined whether STRIP WHITESPACE or PRESERVE WHITESPACE is the *XML whitespace option of the XML BLOB host variable*.

58) Subclause 18.5, "<embedded SQL Fortran program>":

a) The implicit character set in a <Fortran XML CLOB variable> is implementation-defined.

b) If *XO* is the zero-length string, then it is implementation-defined whether DOCUMENT or CONTENT is the *XML option of XML CLOB host variable*.

c) If *XWO* is the zero-length string, then it is implementation-defined whether STRIP WHITESPACE or PRESERVE WHITESPACE is the *XML whitespace option of the XML CLOB host variable*.

d) If *XO* is the zero-length string, then it is implementation-defined whether DOCUMENT or CONTENT is the *XML option of XML BLOB host variable*.

--`,,```,,,,````-`-`,,`,,`,`,,`---
© ISO/IEC 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

e) If *XWO* is the zero-length string, then it is implementation-defined whether STRIP WHITESPACE or PRESERVE WHITESPACE is the *XML whitespace option of the XML BLOB host variable*.

59) Subclause 18.6, "<embedded SQL MUMPS program>":

a) If *XO* is the zero-length string, then it is implementation-defined whether DOCUMENT or CONTENT is the *XML option of XML CLOB host variable*.

b) If *XWO* is the zero-length string, then it is implementation-defined whether STRIP WHITESPACE or PRESERVE WHITESPACE is the *XML whitespace option of the XML CLOB host variable*.

c) If *XO* is the zero-length string, then it is implementation-defined whether DOCUMENT or CONTENT is the *XML option of XML BLOB host variable*.

d) If *XWO* is the zero-length string, then it is implementation-defined whether STRIP WHITESPACE or PRESERVE WHITESPACE is the *XML whitespace option of the XML BLOB host variable*.

60) Subclause 18.7, "<embedded SQL Pascal program>":

a) If *XO* is the zero-length string, then it is implementation-defined whether DOCUMENT or CONTENT is the *XML option of XML CLOB host variable*.

b) If *XWO* is the zero-length string, then it is implementation-defined whether STRIP WHITESPACE or PRESERVE WHITESPACE is the *XML whitespace option of the XML CLOB host variable*.

c) If *XO* is the zero-length string, then it is implementation-defined whether DOCUMENT or CONTENT is the *XML option of XML BLOB host variable*.

d) If *XWO* is the zero-length string, then it is implementation-defined whether STRIP WHITESPACE or PRESERVE WHITESPACE is the *XML whitespace option of the XML BLOB host variable*.

61) Subclause 18.8, "<embedded SQL PL/I program>":

a) The implicit character set in a <PL/I XML VARCHAR variable>, or a <PL/I XML CLOB variable> is implementation-defined.

b) If *XO* is the zero-length string, then it is implementation-defined whether DOCUMENT or CONTENT is the *XML option of XML VARCHAR host variable*.

c) If *XWO* is the zero-length string, then it is implementation-defined whether STRIP WHITESPACE or PRESERVE WHITESPACE is the *XML whitespace option of the XML VARCHAR host variable*.

d) If *XO* is the zero-length string, then it is implementation-defined whether DOCUMENT or CONTENT is the *XML option of XML CLOB host variable*.

e) If *XWO* is the zero-length string, then it is implementation-defined whether STRIP WHITESPACE or PRESERVE WHITESPACE is the *XML whitespace option of the XML CLOB host variable*.

f) If *XO* is the zero-length string, then it is implementation-defined whether DOCUMENT or CONTENT is the *XML option of XML BLOB host variable*.

g) If *XWO* is the zero-length string, then it is implementation-defined whether STRIP WHITESPACE or PRESERVE WHITESPACE is the *XML whitespace option of the XML BLOB host variable*.

62) Subclause 20.1, "NCNAME domain":

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

a) It is implementation-defined whether the NCNAME domain specifies all variable-length character string values that conform to the rules for formation and representation of an XML 1.0 or XML 1.1 NCName.

63) Subclause 20.2, “URI domain”:

a) The maximum length of an <XML URI> is implementation-defined.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex C
(informative)

Implementation-dependent elements

This Annex modifies Annex C, "Implementation-dependent elements", in ISO/IEC 9075-2.

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

1) Subclause 4.10.10, "Mapping an SQL schema to XML":

   a) The repeatable ordering of tables in an SQL-schema is implementation-dependent.

2) Subclause 9.4, "Mapping an SQL data type to an XML Name":

   a) It is implementation-dependent whether the types of two sites of row type, having the same number of fields, and having corresponding fields of the same name and declared type, are mapped to the same XML Name.

3) Subclause 9.8, "Mapping values of SQL data types to values of XML Schema data types":

   a) The ordering of elements of a multiset is implementation-dependent.

4) Subclause 9.16, "Mapping an SQL catalog to an XML element":

   a) The repeatable ordering of XML visible schemas within an SQL catalog is implementation-dependent.

5) Subclause 10.15, "Serialization of an XML value":

   a) The values of ***CSE***, ***DP***, ***DS***, ***EUA***, ***ICT***, ***MT***, ***NF***, and ***UCM*** are implementation-dependent, but shall be values that are permitted values for the cdata-section-elements, doctype-public, doctype-system, escape-uri-attributes, include-content-type, media-type, normalization-form, and use-character-maps serialization parameters, respectively, as defined in Section 3, "Serialization Parameters", of [Serialization].

   b) The result is implementation-dependent, but shall be determined according to Section 5, "XML output method", of [Serialization].

   c) The effects on insignificant whitespace of <XML serialize indent> are implementation-dependent.

6) Subclause 10.16, "Parsing a string as an XML value":

   a) The **[base URI]** property of all XML information items is implementation-dependent.

7) Subclause 11.2, "<aggregate function>":

   a) The order in which items are concatenated in the result of XMLAGG is implementation-dependent if no user-specified ordering is specified or if the user-specified ordering is not a total ordering.

8) Subclause 18.1, "<embedded SQL host program>":

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

a) The variable names generated for use in the specification of the transformation from embedded SQL to SQL procedures are implementation-dependent.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex D
(informative)

Deprecated features

This Annex modifies Annex D, "Deprecated features", in ISO/IEC 9075-2.

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

1) The semantics of XMLCAST to unconditionally generate an XML Schema type when casting an XML value as an SQL value will be removed; that is, the current semantics of Feature X026, "XMLCast: SQL cast semantics", will become the default semantics.

© ISO/IEC 2008 – All rights reserved
--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

(Blank page)

© ISO/IEC 2008 – All rights reserved

--`,,```,,,,````-`-`,,`,,`,`,,`---
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex E
(informative)

Incompatibilities with ISO/IEC 9075-14:2006

This Annex modifies Annex E, "Incompatibilities with ISO/IEC 9075:2008", in ISO/IEC 9075-2.

This edition of this part of ISO/IEC 9075 introduces some incompatibilities with the earlier version of Database Language SQL as specified in ISO/IEC 9075-14:2006.

Except as specified in this Annex, features and capabilities of Database Language SQL are compatible with ISO/IEC 9075-14:2006.

None.

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

(Blank page)

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Annex F
(informative)

SQL feature taxonomy

This Annex modifies Annex F, "SQL feature taxonomy", in ISO/IEC 9075-2.

This Annex describes a taxonomy of features defined in this part of ISO/IEC 9075.

Table 16, "Feature taxonomy for optional features", contains a taxonomy of the optional features of the SQL language that are specified in this part of ISO/IEC 9075. In this table, the first column contains a counter that may be used to quickly locate rows of the table; these values otherwise have no use and are not stable — that is, they are subject to change in future editions of or even Technical Corrigenda to ISO/IEC 9075 without notice.

The "Feature ID" column of this table specifies the formal identification of each feature and each subfeature contained in the table.

The "Feature Name" column of this table contains a brief description of the feature or subfeature associated with the Feature ID value.

Table 16, "Feature taxonomy for optional features", does not provide definitions of the features; the definition of those features is found in the Conformance Rules that are further summarized in Annex A, "SQL Conformance Summary".

Table 16 — Feature taxonomy for optional features

	Feature ID	Feature Name
1	X010	XML type
2	X011	Arrays of XML type
3	X012	Multisets of XML type
4	X013	Distinct types of XML type
5	X014	Attributes of XML type
6	X015	Fields of XML type
7	X016	Persistent XML values
8	X020	XMLConcat

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

	Feature ID	Feature Name
9	X025	XMLCast
10	X026	XMLCast: SQL cast semantics
11	X030	XMLDocument
12	X031	XMLElement
13	X032	XMLForest
14	X034	XMLAgg
15	X035	XMLAgg: ORDER BY option
16	X036	XMLComment
17	X037	XMLPI
18	X038	XMLText
19	X040	Basic table mapping
20	X041	Basic table mapping: null absent
21	X042	Basic table mapping: null as nil
22	X043	Basic table mapping: table as forest
23	X044	Basic table mapping: table as element
24	X045	Basic table mapping: with target namespace
25	X046	Basic table mapping: data mapping
26	X047	Basic table mapping: metadata mapping
27	X048	Basic table mapping: base64 encoding of binary strings
28	X049	Basic table mapping: hex encoding of binary strings
29	X050	Advanced table mapping
30	X051	Advanced table mapping: null absent
31	X052	Advanced table mapping: null as nil
32	X053	Advanced table mapping: table as forest
33	X054	Advanced table mapping: table as element

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

	Feature ID	Feature Name
34	X055	Advanced table mapping: with target namespace
35	X056	Advanced table mapping: data mapping
36	X057	Advanced table mapping: metadata mapping
37	X058	Advanced table mapping: base64 encoding of binary strings
38	X059	Advanced table mapping: hex encoding of binary strings
39	X060	XMLParse: Character string input and CONTENT option
40	X061	XMLParse: Character string input and DOCUMENT option
41	X065	XMLParse: BLOB input and CONTENT option
42	X066	XMLParse: BLOB input and DOCUMENT option
43	X068	XMLSerialize: BOM
44	X069	XMLSerialize: INDENT
45	X070	XMLSerialize: Character string serialization and CONTENT option
46	X071	XMLSerialize: Character string serialization and DOCUMENT option
47	X072	XMLSerialize: Character string serialization
48	X073	XMLSerialize: BLOB serialization and CONTENT option
49	X074	XMLSerialize: BLOB serialization and DOCUMENT option
50	X075	XMLSerialize: BLOB serialization
51	X076	XMLSerialize: VERSION
52	X077	XMLSerialize: explicit ENCODING option
53	X078	XMLSerialize: explicit XML declaration
54	X080	Namespaces in XML publishing
55	X081	Query-level XML namespace declarations
56	X082	XML namespace declarations in DML
57	X083	XML namespace declarations in DDL
58	X084	XML namespace declarations in compound statements

© ISO/IEC 2008 – All rights reserved

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

	Feature ID	Feature Name
59	X085	Predefined namespace prefixes
60	X086	XML namespace declarations in XMLTable
61	X090	XML document predicate
62	X091	XML content predicate
63	X096	XMLExists
64	X100	Host language support for XML: CONTENT option
65	X101	Host language support for XML: DOCUMENT option
66	X110	Host language support for XML: VARCHAR mapping
67	X111	Host language support for XML: CLOB mapping
68	X112	Host language support for XML: BLOB mapping
69	X113	Host language support for XML: STRIP WHITESPACE option
70	X114	Host language support for XML: PRESERVE WHITESPACE option
71	X120	XML parameters in SQL routines
72	X121	XML parameters in external routines
73	X131	Query-level XMLBINARY clause
74	X132	XMLBINARY clause in DML
75	X133	XMLBINARY clause in DDL
76	X134	XMLBINARY clause in compound statements
77	X135	XMLBINARY clause in subqueries
78	X141	IS VALID predicate: data-driven case
79	X142	IS VALID predicate: ACCORDING TO clause
80	X143	IS VALID predicate: ELEMENT clause
81	X144	IS VALID predicate: schema location
82	X145	IS VALID predicate outside check constraints
83	X151	IS VALID predicate with DOCUMENT option

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

	Feature ID	Feature Name
84	**X152**	**IS VALID predicate with CONTENT option**
85	**X153**	**IS VALID predicate with SEQUENCE option**
86	**X155**	**IS VALID predicate: NAMESPACE without ELEMENT clause**
87	**X157**	**IS VALID predicate: NO NAMESPACE with ELEMENT clause**
88	**X160**	**Basic Information Schema for registered XML Schemas**
89	**X161**	**Advanced Information Schema for registered XML Schemas**
90	**X170**	**XML null handling options**
91	**X171**	**NIL ON NO CONTENT option**
92	**X181**	**XML(DOCUMENT(UNTYPED)) type**
93	**X182**	**XML(DOCUMENT(ANY)) type**
94	**X190**	**XML(SEQUENCE) type**
95	**X191**	**XML(DOCUMENT(XMLSCHEMA)) type**
96	**X192**	**XML(CONTENT(XMLSCHEMA)) type**
97	**X200**	**XMLQuery**
98	**X201**	**XMLQuery: RETURNING CONTENT**
99	**X202**	**XMLQuery: RETURNING SEQUENCE**
100	**X203**	**XMLQuery: passing a context item**
101	**X204**	**XMLQuery: initializing an XQuery variable**
102	**X205**	XMLQuery: EMPTY ON EMPTY option
103	**X206**	XMLQuery: NULL ON EMPTY option
104	**X211**	**XML 1.1 support**
105	**X221**	**XML passing mechanism BY VALUE**
106	**X222**	**XML passing mechanism BY REF**
107	**X231**	**XML(CONTENT(UNTYPED)) type**
108	**X232**	**XML(CONTENT(ANY)) type**

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

	Feature ID	Feature Name
109	X241	RETURNING CONTENT in XML publishing
110	X242	RETURNING SEQUENCE in XML publishing
111	X251	Persistent XML values of XML(DOCUMENT(UNTYPED)) type
112	X252	Persistent XML values of XML(DOCUMENT(ANY)) type
113	X253	Persistent XML values of XML(CONTENT(UNTYPED)) type
114	X254	Persistent XML values of XML(CONTENT(ANY)) type
115	X255	Persistent XML values of XML(SEQUENCE) type
116	X256	Persistent XML values of XML(DOCUMENT(XMLSCHEMA)) type
117	X257	Persistent XML values of XML(CONTENT(XMLSCHEMA)) type
118	X260	XML type: ELEMENT clause
119	X261	XML type: NAMESPACE without ELEMENT clause
120	X263	XML type: NO NAMESPACE with ELEMENT clause
121	X264	XML type: schema location
122	X271	XMLValidate: data-driven case
123	X272	XMLValidate: ACCORDING TO clause
124	X273	XMLValidate: ELEMENT clause
125	X274	XMLValidate: schema location
126	X281	XMLValidate: with DOCUMENT option
127	X282	XMLValidate with CONTENT option
128	X283	XMLValidate with SEQUENCE option
129	X284	XMLValidate NAMESPACE without ELEMENT clause
130	X286	XMLValidate: NO NAMESPACE with ELEMENT clause
131	X300	XMLTable
132	X301	XMLTable: derived column list option
133	X302	XMLTable: ordinality column option

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

	Feature ID	Feature Name
134	X303	XMLTable: column default option
135	X304	XMLTable: passing a context item
136	X305	XMLTable: initializing an XQuery variable
137	X400	Name and identifier mapping

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

(Blank page)

© ISO/IEC 2008 – All rights reserved

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS

Not for Resale

Annex G
(informative)

Defect reports not addressed in this edition of this part of ISO/IEC 9075

Each entry in this Annex describes a reported defect in the previous edition of this part of ISO/IEC 9075 that remains in this edition.

None.

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS Not for Resale

(Blank page)

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Index

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

— A —

ABS • 146
ABSENT • *31*, *66*, 200
ACCORDING • *31*, 43, *231*, 354
ADA • 245, 378
<Ada derived type specification> • **283**
<Ada XML BLOB variable> • **283**, 284, 285, 370, 372, 374, 375, 376
<Ada XML CLOB variable> • **283**, 284, 285, 370, 372, 373, 375, 376
<aggregate function> • 51, **224**, 405
all group content model • **7**
AND • 107, 321, 322, 323, 329, 330
annotation • **7**
ANY • 13, 14, 15, 16, 20, *35*, 36, 37, 42, 43, 44, 47, 52, 63, 64, 73, 86, 87, 182, 184, 186, 216, 217, 224, 236, 238, 239, 384, 387, 389, 390, 396, 397
ARRAY • 138, 300, 341
AS • *41*, *44*, 45, 46, 50, *53*, 63, *66*, 68, *70*, 71, *78*, 81, 82, *91*, 93, 94, 144, 146, 149, 160, 161, 168, 169, 177, 178, 222, 225, *227*, *243*, *261*, 273, 281, *283*, 284, *286*, 287, 288, *291*, 292, *294*, 295, *297*, 298, *300*, 301, *303*, 304, 309, 310, 317, 318, 319, 321, 322, 323, 327
ASSERTION • *240*
<assertion definition> • 51, **240**, 369, 379
associated string type • 21, *243*, *244*, 245, 319, 332, 334
associated XML option • 21, 222, *244*, 245, 332, 334
ATOMIC • *265*, 281

— B —

BASE64 • *31*, 44, 46, 71, *227*
BEGIN • *265*, 281
BIGINT • 25, 46, 119, 125, 126, 341, 395, 398
BINARY • 25, 118
BLOB • 123, 277, 278, 279, 280, *283*, 284, *286*, 288, *291*, 292, *294*, 295, *297*, 298, *300*, 301, *303*, 304, 341
<blob value function> • **53**, 54, 55
BOM • *31*, *53*, 55, 56
BOOLEAN • 25, 28, 50, 119, 129, 145, 341
BY • 16, 41, 42, 43, 45, 51, 52, 79, 81, 82, 92, 93, 94, 96, 184, *224*, 225, *230*, 245, 255, 257, 267, 332, 333, 334, 335, 386, 401

— C —

C • 245, 378
<C derived variable> • **286**
<C XML BLOB variable> • **286**, 288, 289, 290, 370, 372, 374, 375, 376
<C XML CLOB variable> • **286**, 287, 288, 289, 370, 372, 373, 375, 376, 402
<C XML VARCHAR variable> • **286**, 287, 288, 289, 290, 370, 372, 373, 375, 376, 402
canonical XML Schema literal • **8**
CASE • 93
CAST • *41*, 45, 46, 50, 63, 68, 71, 144, 146, 225, 281
<cast specification> • 20, 26, 39, **41**, 51, 396
CATALOG • 172, 343
CHAR • 122, 246, 300, 341, 373
CHARACTER • 25, 117, 118, 122, 140, 144, 146, 151, 246, 275, 277, 278, 279, 280, *283*, *286*, 287, 288, *291*, 292, *294*, 295, *297*, *300*, *303*, 304, 305, 309, 310, 343, 373, 395
character set of XML BLOB host variable • *288*, *292*, *295*, *298*, *301*, *305*
character set of XML CLOB host variable • *284*, *287*, *291*, *294*, *297*, *301*, *304*
character set of XML VARCHAR host variable • *287*, *304*
<character value function> • **53**, 54, 55
CHARACTER_SET_CATALOG • 319
CHARACTER_SET_NAME • 319
CHARACTER_SET_SCHEMA • 319
CHARACTERS • 298
CHECK • *237*, *240*, 330, 332, 334

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

<check constraint definition> • 51, 109, **237**, 369, 378, 379, 380
CLOB • 122, 149, 160, 161, 168, 169, 177, 178, 246, 277, 278, 279, 280, *283*, *286*, 287, 288, *291*, *294*, *297*, *300*, *303*, 304, 341, 373
COBOL • 245, 378
<COBOL derived type specification> • **291**
<COBOL XML BLOB variable> • **291**, 292, 293, 371, 372, 374, 375, 376
<COBOL XML CLOB variable> • **291**, 292, 293, 371, 372, 373, 375, 376, 402
COLLATION • 151, 343
COLLATION_CATALOG • 319
COLLATION_NAME • 319
COLLATION_SCHEMA • 319
column cannot be mapped • 346
column cannot be mapped to XML • 157, 166, 175
COLUMNS • *31*, *91*
<common value expression> • **51**, 52, 100, 102, 105
<compound statement> • **265**, 266, 369, 379
CONSTRAINT • 330, 332, 334, 337, 338, 339
CONTENT • 13, 14, 15, 16, 20, *31*, *35*, 36, 37, 42, 43, 44, 47, 52, *53*, 54, 56, 57, 60, 61, 62, 63, 64, 65, *66*, 67, 68, 69, 71, 72, 73, 74, 75, 77, 78, 79, 80, 82, 84, 85, 87, 90, 93, *100*, 106, 109, 149, 161, 181, 182, 183, 184, 185, 186, 201, 204, 208, 209, 213, 216, 217, 224, 225, 226, *229*, 236, 239, 246, 269, 284, 285, 287, 288, 289, 292, 293, 295, 296, 298, 299, 301, 302, 304, 305, 306, 332, 334, 366, 367, 370, 371, 380, 384, 385, 387, 388, 389, 390, 392, 396, 397, 398, 401, 402, 403
CREATE • *240*, 309, 310, 321, 322, 323, 326, 327, 337, 338
CURRENT • *262*
CURRENT_USER • 321, 322

— D —

DATA • 295
data exception • 27, 43, 49, 50, 60, 81, 87, 88, 89, 90, 146, 182, 184, 204, 206, 207, 208, 209, 225, 233, 234, 345
data value • 76
DATE • 25, 28, 46, 49, 120, 129, 145, 342
datetime field overflow • 27, 146
DAY • 134, 147, 342
DECIMAL • 25, 46, 118, 124, 341
DECLARE • *265*, 281
DEFAULT • 214, 215, *227*
DELETE • *259*
<delete statement: searched> • **259**, 368, 378
DISTINCT • 137, 341
DOCUMENT • 13, 14, 15, 16, *31*, *35*, 36, 37, 42, 43, 44, *53*, 56, 57, 62, 73, 74, 86, 87, 89, 90, *102*, 105, 107, 109, 160, 161, 168, 169, 177, 178, 181, 182, 183, 184, 185, 186, 204, 205, 208, 209, 216, 217, 224, 236, 238, 246, 269, 284, 285, 287, 288, 289, 292, 293, 295, 296, 298, 299, 301, 302, 304, 305, 306, 332, 334, 366, 367, 371, 372, 373, 380, 384, 385, 389, 390, 392, 396, 397, 400, 401, 402, 403
<document or content> • 22, **53**, 54, 55, 56, 57, *73*, 74, *86*, *243*, 244, 246, *269*, *283*, 284, 285, *286*, 287, 288, 289, *291*, 292, 293, *294*, 295, 296, *297*, 298, 299, *300*, 301, 302, *303*, 304, 305, 306, 366, 367, 370, 371, 372, 373, 397
<document or content or sequence> • **86**, 90, *105*, 109, 380, 392
DOMAIN • 136, 309, 310, 341
DOUBLE • 25, 119, 128, 341
DTD • **7**
dynamic SQL error • 272, 273

— E —

ELEMENT • *231*, 354
element namespace not declared • 233, 346
ELSE • 94
EMPTY • *31*, *66*, 67, *78*, 82, 83, 93, 94, 104, 385, 386
empty XQuery sequence • **8**
ENCODING • *31*, *53*, 57, 368
END • 94, *265*, 281, 283, 284
equivalent • 42, 45, 60, 62, 63, 67, 68, 69, 71, 75, 79, 80, 82, 84, 88, 89, 92, 94, 100, 102, 105, 107, 108, 143, 181, 183, 185, 186, 190, *191*, 193, 198, 202, 206, 212, 213, 225, 227, 280, 295, 369
error • 65
EXCLUDING • *53*, 55, 56, 222, 273, 279, 280, 400, 401
EXECUTE • 28

— F —

Feature F251, “Domain support” • 309, 310, 359
Feature F391, “Long identifiers” • 321, 322, 323, 324, 359
Feature F761, “Session management” • 269, 359
facet • **7**
<fetch target list> • **255**
FLOAT • 25, 119, 128, 341
FOR • *92*, 145, 146, 147, 148
FOREIGN • 337, 338, 339
<forest element> • 16, **70**, 71
<forest element list> • **70**, 71
<forest element name> • 16, **70**, 72, 369, 386
<forest element value> • 16, **70**, 71
FORTRAN • 245, 378
<Fortran derived type specification> • **294**

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

<Fortran XML BLOB variable> • **294**, 295, 296, 371, 372, 374, 375, 377

<Fortran XML CLOB variable> • **294**, 295, 296, 371, 372, 374, 375, 377, 402

FROM • 94, 145, 146, 147, 148, *259*, 319, 321, 322, 323, 327, 330, 331, 332, 334, 336

fully escaped • *24*, *111*, 112, 114, 137, 148, 150, 152, 155

— G —

<generation expression> • **235**, 236, 369, 378

global element declaration schema component • **8**

global element not declared • 234, 346

GRANT • 309, 310, 321, 322, 323, 327

— H —

HEX • *31*, *227*

homomorphic • *23*, 25, 117, 118, 122, 140

HOUR • 134, 135, 147, 342

— I —

ID • *31*, *231*

identical • 15, 16, *18*, 23, 42, 62, 67, 88, 89, 106, 107, 108, 181, 183, 185, 186, 190, 191, 192, 193, 200, 227, 232, 233, 234

IN • 321, 322, 323, 329, 330, 332, 334

INCLUDING • *53*, 55, 56, 222, 273, 279, 280, 400, 401

INDENT • *31*, *53*

indicated global element declaration schema component • 14, 36, 86, 105, *233*

indicated registered XML schema • *233*

indicated XML namespace • 14, 36, 86, 105, *233*, 400

INSERT • *260*

<insert statement> • **260**, 368, 378

INTEGER • 19, 25, 46, 119, 125, 295, 341, 395, 398

INTERVAL • 133, 134, 135, 146, 147, 148, 342

INTO • *260*, *261*

invalid comment • 60, 345

invalid datetime format • 49, 50

invalid processing instruction • 76, 345

invalid XML character • 145, 346

invalid XML content • 87, 208, 209, 345

invalid XML document • 87, 208, 209, 345

invalid XQuery context item • 81, 345

INVOKER • 46, 233

IS • *100*, *102*, *105*, *283*, 284, *286*, 287, 288, *291*, 292, *294*, 295, *297*, 298, *300*, 301, *303*, 304, 329, 330, 354

— J —

JOIN • 318, 319, 321, 322

— K —

KEY • 337, 338, 339

— L —

LARGE • 25, 117, 118, 122, 246, 288, 292, 295, 305, 373

LATERAL • 94

LEFT • 318, 319

LENGTH • 295

length of XML BLOB host variable • *284*, *288*, *292*, *295*, *298*, *301*, *305*

length of XML CLOB host variable • *284*, *287*, *291*, *294*, *297*, *301*, *304*

length of XML VARCHAR host variable • *287*, *304*

LOCATION • *31*, *231*

LOCATOR • 288

LOWER • 145

— M —

MERGE • *261*

<merge statement> • **261**, 368, 378

metadata • **8**

MINUTE • 134, 135, 147, 148, 342

MONTH • 133, 134, 146, 147, 342

MULTISET • 138, 341

MUMPS • 245, 378

<MUMPS derived type specification> • **297**

<MUMPS XML BLOB variable> • **297**, 298, 299, 371, 372, 374, 375, 377

<MUMPS XML CLOB variable> • **297**, 298, 299, 371, 372, 374, 375, 377

— N —

NAME • *66*, 68, *75*

NAMESPACE • *31*, 37, 90, 109, *231*, 232, 233, 380, 391, 392

NCHAR • 288

NCLOB • 288

NIL • *31*, *66*, 69, 201, 384

NO • 37, *53*, 55, 56, *66*, 69, 90, 109, 201, 215, *227*, *231*, 232, 233, 380, 384, 391, 392

no subclass • 345, 346

no XML element with the specified namespace • 88, 89, 346

no XML element with the specified QName • 88, 89, 346

no XML schema found • 88, 89, 346

non-deterministic • *18*, 51, 92, 337, 338, 339, 340, 396, 398

<non-reserved word> • **31**

nonidentical notations with the same name • 345

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

nonidentical unparsed entities with the same name • 345
<nonparenthesized value expression primary> • **39**
NOT • *100*, *102*, *105*, *265*, 329, 330, 337, 338, 339
not an XML document • 43, 182, 184, 204, 345
not an XQuery document node • 43, 182, 184, 345
NULL • 44, *66*, 67, 71, *78*, 82, 83, 93, 200, 201, 329, 330, 337, 338, 339, 386
NUMERIC • 25, 46, 118, 124, 341
numeric value out of range • 49

— O —

OBJECT • 25, 117, 118, 122, 246, 288, 292, 295, 305, 373
OF • *262*, 300
ON • *66*, 67, 69, 71, *78*, 82, 83, 93, 94, 104, 200, 201, *261*, 309, 310, 318, 319, 321, 322, 323, 327, 384, 385, 386
OPTION • *66*, *70*, *269*, 309, 310, 321, 322, 323, 327
OR • 321, 322, 329, 330
ORDER • *224*, 225
ORDINALITY • *92*

— P —

<parameter type> • **243**, 244, 245, 377, 378
partially escaped • *24*, *111*
PASCAL • 245, 378
<Pascal derived type specification> • **300**
<Pascal XML BLOB variable> • **300**, 301, 302, 371, 373, 374, 376, 377
<Pascal XML CLOB variable> • **300**, 302, 371, 373, 374, 376, 377
PASSING • *31*, *78*, *91*, 92, 93
PATH • *31*, *92*
<PL/I derived type specification> • **303**
<PL/I XML BLOB variable> • **303**, 306, 371, 373, 374, 376, 377
<PL/I XML CLOB variable> • **303**, 304, 305, 306, 371, 373, 374, 376, 377, 403
<PL/I XML VARCHAR variable> • **303**, 305, 306, 371, 373, 376, 377, 403
PLI • 245, 378
potentially whitespace-strippable • *209*, 210
PRECISION • 25, 119, 128, 341
<predefined type> • **35**, 241, 360
<predicate> • **99**
PRESERVE • *31*, 47, *73*, 160, 161, 168, 169, 177, 178, 201, 208, 222, 272, 284, 285, 287, 288, 289, 290, 292, 293, 295, 296, 298, 299, 301, 302, 304, 305, 306, 376, 377, 400, 401, 402, 403
PRIMARY • 337, 338, 339
<primary XML type modifier> • 14, **35**, 36, 37, 384, 385, 387
PUBLIC • 309, 310, 321, 322, 323, 327

— R —

REAL • 25, 119, 127, 341
RECURSIVE • *96*
REF • 16, 41, 42, 45, 81, 82, 92, 93, 94, 96, *230*, 245, 255, 257, 267, 288, 332, 334, 386, 401
REFERENCES • 337, 338, 339
registered XML Schema • 11, 14, 15, 17, *18*, 19, 36, 42, 51, 62, 63, 86, 87, 89, 105, 107, 108, 181, 183, 185, 186, 224, 231, 232, 233, 331, 336, 337, 338, 339, 340, 398, 400
<registered XML Schema name> • 18, 19, **33**, *231*, 232, 233, 395, 400
repeatable ordering • *30*, 154, 162, 164, 173, 405
<reserved word> • **32**
restricted data type attribute violation • 272, 273
RESULT • *243*
RETURNING • *31*, 51, 60, 61, 62, 63, 64, 65, 67, 68, 69, 71, 72, 75, 77, 78, 79, 80, 82, 84, 85, 93, 94, 104, 224, 225, 226, *229*, 385, 387, 388, 397, 398
RETURNS • *243*
<returns clause> • 21, 222, **243**, 244, 245, 334, 335, 401
<returns data type> • **243**, 245, 320, 377, 378
ROUTINE • 318, 319, 332, 334
ROW • 137, 341

— S —

SCHEMA • 163, 343
SECOND • 27, 133, 134, 135, 146, 147, 148, 342, 396, 399
<secondary XML type modifier> • 14, **35**, 36, 37, 43, 384, 385, 387
SECURITY • 46, 233
SELECT • 28, 94, 321, 322, 323, 327, 330, 332, 336
<select target list> • **257**
SEQUENCE • 13, 14, 15, 16, 20, 28, *31*, *35*, 36, 37, 43, 47, 51, 60, 61, 62, 63, 64, 65, 67, 68, 69, 71, 72, 75, 77, 78, 79, 80, 81, 82, 84, 85, *86*, 87, 90, 92, 93, 94, 104, 106, 109, 139, 185, 201, 216, 217, 224, 225, 226, 229, 236, 239, 380, 384, 385, 388, 390, 392, 396, 397, 398
sequence content model • **8**
SET • 151, *262*, *263*, *267*, *269*, 277, 278, 279, 280, 281, *283*, *286*, 287, 288, *291*, *294*, *297*, *300*, *303*, 304, 309, 310, 343
<set XML option statement> • 21, 22, *251*, **269**, 307, 359, 370, 371, 372, 395
<singleton variable assignment> • **267**
SMALLINT • 25, 46, 119, 125, 126, 341, 395, 398

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

SPECIFIC_NAME • 319, 332, 334
SQL • 46, 233, *283*, *286*, *291*, *294*, *297*, *300*, *303*
<SQL parameter declaration> • **243**, 244, 245, 401
<SQL session statement> • **251**
SQL value space • **8**
SQL/XML mapping error • 116, 145, 346
STANDALONE • *31*
<string type option> • **243**, 244, 246, 373
STRIP • *31*, *73*, 208, 209, 222, 272, 284, 285, 287, 288, 289, 292, 293, 295, 296, 298, 299, 301, 302, 304, 305, 306, 375, 376, 400, 401, 402, 403
SUBSTRING • 145, 146, 147, 148

— T —

TABLE • 321, 322, 323, 327, 337, 338, 343
<table primary> • **91**, 92, 94, 392
<target specification 1> • **255**, *257*
textual XML 1.0 content • 9, 17, *208*, 209
textual XML 1.0 document • *208*, 209
textual XML 1.1 content • 9, 17, *208*
textual XML 1.1 document • *208*
THEN • 93
TIME • 26, 27, 46, 48, 49, 50, 119, 120, 129, 130, 131, 132, 145, 146, 342, 399
TIMESTAMP • 26, 27, 46, 48, 49, 50, 120, 131, 132, 146, 342, 399
TO • 43, 133, 134, 135, 146, 147, 148, *231*, 309, 310, 321, 322, 323, 327, 342, 354
TYPE • *283*, 284, *286*, 287, 288, *291*, 292, *294*, 295, *297*, 298, *300*, 301, *303*, 304

— U —

UNION • 336
unmappable XML Name • 116, 346
UNTYPED • 13, 14, 15, 20, *31*, *35*, 36, 37, 42, 43, 44, 47, 52, 62, 64, 65, 73, 86, 87, 181, 182, 183, 184, 185, 213, 216, 217, 224, 236, 238, 239, 384, 387, 389, 396, 397
UPDATE • *262*, *263*
<update statement: positioned> • **262**, 368, 378
<update statement: searched> • **263**, 369, 378
<uppercase hexit> • **111**, 112
URI • **8**, *31*, *231*
USAGE • 22, 46, 47, 88, 89, 108, 215, 233, *291*, 309, 310, 398, 400
USING • *227*, *261*, 321, 322

— V —

VALID • *31*, *105*, 354
valid XML 1.0 character • **8**
valid XML 1.1 character • **8**
valid XML character • **8**
validation failure • 89, 90, 346
VALUE • 16, 41, 43, 45, 51, 52, 79, 93, 184, *230*, 245, 255, 257, 267, 332, 333, 334, 335, 386, 401
value space • **7**
VARCHAR • 122, 246, 277, 278, 279, 280, *286*, 288, *303*, 305, 341, 373
VARYING • 25, 117, 118, 122, 144, 146, 246, 275, 288, 309, 310, 373, 395
VERSION • *31*, *53*, 57, 367
VIEW • 321, 322, 323, 326, 327
visible column • *28*, 150, 154
visible schema • *28*, 171, 173
visible table • *28*, 162, 164

— W —

warning • 157, 166, 175, 346
well-formed • **7**
WHEN • 93
<when operand> • **40**
WHERE • *259*, *262*, *263*, 321, 322, 323
WHITESPACE • *31*, 47, *73*, 160, 161, 168, 169, 177, 178, 201, 208, 209, 222, 272, 284, 285, 287, 288, 289, 290, 292, 293, 295, 296, 298, 299, 301, 302, 304, 305, 306, 375, 376, 377, 400, 401, 402, 403
wildcard schema component • **7**
WITH • 26, 27, 46, 49, 50, *53*, 55, 56, 92, *96*, 119, 120, 130, 132, 145, 146, *235*, *237*, *240*, *259*, *260*, *261*, *262*, *263*, 309, 310, 321, 322, 323, 327, 342
<with clause> • **96**, 368, 378
WITHOUT • 26, 27, 46, 48, 49, 50, 119, 120, 129, 131

— X —

Feature X010, “XML type” • 36, 58, 59, 347, 359, 360
Feature X011, “Arrays of XML type” • 36, 360
Feature X012, “Multisets of XML type” • 36, 360
Feature X013, “Distinct types of XML type” • 241, 360
Feature X014, “Attributes of XML type” • 242, 360
Feature X015, “Fields of XML type” • 38, 360
Feature X016, “Persistent XML values” • 236, 238, 347, 360, 361
Feature X020, “XMLConcat” • 63, 361
Feature X025, “XMLCast” • 50, 361
Feature X026, “XMLCast: SQL cast semantics” • 50, 361, 407
Feature X030, “XMLDocument” • 65, 361
Feature X031, “XMLElement” • 69, 347, 361
Feature X032, “XMLForest” • 72, 347, 361
Feature X034, “XMLAgg” • 226, 361

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Feature X035, “XMLAgg: ORDER BY option” • 226, 347, 361, 362
Feature X036, “XMLComment” • 61, 347, 362
Feature X037, “XMLPI” • 76, 347, 362
Feature X038, “XMLText” • 85, 362
Feature X040, “Basic table mapping” • 161, 348, 362
Feature X041, “Basic table mapping: nulls absent” • 161, 348, 362
Feature X042, “Basic table mapping: null as nil” • 161, 348, 362
Feature X043, “Basic table mapping: table as forest” • 161, 348, 362
Feature X044, “Basic table mapping: table as element” • 161, 348, 362, 363
Feature X045, “Basic table mapping: with target namespace” • 161, 363
Feature X046, “Basic table mapping: data mapping” • 161, 348, 363
Feature X047, “Basic table mapping: metadata mapping” • 161, 348, 363
Feature X048, “Basic table mapping: base64 encoding of binary strings” • 161, 348, 363
Feature X049, “Basic table mapping: hex encoding of binary strings” • 161, 348, 363
Feature X050, “Advanced table mapping” • 169, 178, 349, 363
Feature X051, “Advanced table mapping: nulls absent” • 169, 178, 349, 363, 364
Feature X052, “Advanced table mapping: null as nil” • 169, 178, 349, 364
Feature X053, “Advanced table mapping: table as forest” • 169, 178, 349, 364
Feature X054, “Advanced table mapping: table as element” • 169, 178, 349, 364
Feature X055, “Advanced table mapping: with target namespace” • 170, 178, 364, 365
Feature X056, “Advanced table mapping: data mapping” • 170, 179, 349, 365
Feature X057, “Advanced table mapping: metadata mapping” • 170, 179, 349, 365
Feature X058, “Advanced table mapping: base64 encoding of binary strings” • 170, 179, 349, 365
Feature X059, “Advanced table mapping: hex encoding of binary strings” • 170, 179, 349, 365, 366
Feature X060, “XMLParse: Character string input and CONTENT option” • 73, 347, 366
Feature X061, “XMLParse: Character string input and DOCUMENT option” • 74, 347, 366
Feature X065, “XMLParse: BLOB input and CONTENT option” • 74, 347, 366
Feature X066, “XMLParse: BLOB input and DOCUMENT option” • 74, 347, 366
Feature X068, “XMLSerialize: BOM” • 57, 366
Feature X069, “XMLSerialize: INDENT” • 57, 366
Feature X070, “XMLSerialize: Character string serialization and CONTENT option” • 56, 347, 366, 367
Feature X071, “XMLSerialize: Character string serialization and DOCUMENT option” • 56, 347, 367
Feature X072, “XMLSerialize: character string serialization” • 56, 367
Feature X073, “XMLSerialize: BLOB serialization and CONTENT option” • 57, 347, 367
Feature X074, “XMLSerialize: BLOB serialization and DOCUMENT option” • 57, 347, 367
Feature X075, “XMLSerialize: BLOB serialization” • 57, 367
Feature X076, “XMLSerialize: VERSION” • 57, 367
Feature X077, “XMLSerialize: explicit ENCODING option” • 57, 367, 368
Feature X078, “XMLSerialize: explicit XML declaration” • 57, 368
Feature X080, “Namespaces in XML publishing” • 69, 72, 348, 368
Feature X081, “Query-level XML namespace declarations” • 96, 348, 368
Feature X082, “XML namespace declarations in DML” • 259, 260, 261, 262, 263, 368, 369
Feature X083, “XML namespace declarations in DDL” • 235, 237, 240, 369
Feature X084, “XML namespace declarations in compound statements” • 266, 369
Feature X085, “Predefined namespace prefixes” • 69, 72, 369
Feature X086, “XML namespace declarations in XMLTable” • 95, 369, 370
Feature X090, “XML document predicate” • 103, 370
Feature X091, “XML content predicate” • 101, 370
Feature X096, “XMLExists” • 104, 370
Feature X100, “Host language support for XML: CONTENT option” • 246, 269, 285, 289, 293, 296, 298, 299, 302, 305, 306, 347, 348, 370, 371
Feature X101, “Host language support for XML: DOCUMENT option” • 246, 269, 285, 289, 293, 296, 299, 302, 305, 306, 348, 371, 372, 373
Feature X110, “Host language support for XML: VARCHAR mapping” • 246, 289, 305, 347, 348, 373
Feature X111, “Host language support for XML: CLOB mapping” • 246, 284, 289, 292, 295, 298, 302, 305, 348, 373, 374
Feature X112, “Host language support for XML: BLOB mapping” • 285, 289, 293, 296, 299, 302, 306, 374
Feature X113, “Host language support for XML: STRIP WHITESPACE option” • 285, 289, 293, 296, 299, 302, 306, 374, 375, 376

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Feature X114, “Host language support for XML: PRESERVE WHITESPACE option” • 285, 289, 290, 293, 296, 299, 302, 306, 376, 377
Feature X120, “XML parameters in SQL routines” • 245, 377
Feature X121, “XML parameters in external routines” • 245, 377, 378
Feature X131, “Query-level XMLBINARY clause” • 96, 378
Feature X132, “XMLBINARY clause in DML” • 259, 260, 261, 262, 263, 378
Feature X133, “XMLBINARY clause in DDL” • 236, 237, 240, 378, 379
Feature X134, “XMLBINARY clause in compound statements” • 266, 379
Feature X135, “XMLBINARY clause in subqueries” • 97, 379
Feature X141, “IS VALID predicate: data-driven case” • 109, 379
Feature X142, “IS VALID predicate: ACCORDING TO clause” • 109, 379
Feature X143, “IS VALID predicate: ELEMENT clause” • 109, 379
Feature X144, “IS VALID predicate: schema location” • 109, 379
Feature X145, “IS VALID predicate outside check constraints” • 109, 379, 380
Feature X151, “IS VALID predicate: with DOCUMENT option” • 109, 380
Feature X152, “IS VALID predicate: with CONTENT option” • 109, 380
Feature X153, “IS VALID predicate: with SEQUENCE option” • 109, 380
Feature X155, “IS VALID predicate: NAMESPACE without ELEMENT clause” • 109, 380
Feature X157, “IS VALID predicate: NO NAMESPACE with ELEMENT clause” • 109, 380
Feature X160, “Basic Information Schema for registered XML Schemas” • 311, 312, 313, 314, 315, 316, 317, 318, 320, 323, 327, 380, 381, 382
Feature X161, “Advanced Information Schema for registered XML Schemas” • 18, 311, 312, 313, 314, 315, 316, 317, 318, 320, 321, 322, 323, 327, 382, 383, 384
Feature X170, “XML null handling options” • 69, 384
Feature X171, “NIL ON NO CONTENT option” • 69, 384
Feature X181, “XML(DOCUMENT(UNTYPED)) type” • 36, 384
Feature X182, “XML(DOCUMENT(ANY)) type” • 36, 384
Feature X190, “XML(SEQUENCE) type” • 37, 384
Feature X191, “XML(DOCUMENT(XMLSCHEMA)) type” • 37, 348, 384, 385
Feature X192, “XML(CONTENT(XMLSCHEMA)) type” • 37, 348, 385
Feature X200, “XMLQuery” • 82, 385
Feature X201, “XMLQuery: RETURNING CONTENT” • 82, 385
Feature X202, “XMLQuery: RETURNING SEQUENCE” • 82, 385
Feature X203, “XMLQuery: passing a context item” • 82, 385
Feature X204, “XMLQuery: initializing an XQuery variable” • 82, 385
Feature X205, “XMLQuery: EMPTY ON EMPTY option” • 83, 385
Feature X206, “XMLQuery: NULL ON EMPTY option” • 83, 385, 386
Feature X211, “XML 1.1 support” • 8, 47, 65, 69, 72, 76, 77, 82, 83, 112, 145, 202, 208, 219, 228, 386, 399
Feature X221, “XML passing mechanism BY VALUE” • 230, 386
Feature X222, “XML passing mechanism BY REF” • 230, 386
Feature X231, “XML(CONTENT(UNTYPED)) type” • 37, 386, 387
Feature X232, “XML(CONTENT(ANY)) type” • 37, 387
Feature X241, “RETURNING CONTENT in XML publishing” • 61, 63, 65, 69, 72, 77, 85, 226, 387, 388
Feature X242, “RETURNING SEQUENCE in XML publishing” • 61, 63, 65, 69, 72, 77, 85, 226, 388
Feature X251, “Persistent XML values of XML(DOCUMENT(UNTYPED)) type” • 236, 238, 388, 389
Feature X252, “Persistent XML values of XML(DOCUMENT(ANY)) type” • 236, 238, 389
Feature X253, “Persistent XML values of XML(CONTENT(UNTYPED)) type” • 236, 238, 389
Feature X254, “Persistent XML values of XML(CONTENT(ANY)) type” • 236, 239, 389, 390
Feature X255, “Persistent XML values of XML(SEQUENCE) type” • 236, 239, 390
Feature X256, “Persistent XML values of XML(DOCUMENT(XMLSCHEMA)) type” • 236, 238, 390
Feature X257, “Persistent XML values of XML(CONTENT(XMLSCHEMA)) type” • 236, 239, 390
Feature X260, “XML type: ELEMENT clause” • 37, 348, 390, 391
Feature X261, “XML type: NAMESPACE without ELEMENT clause” • 37, 348, 391
Feature X263, “XML type: NO NAMESPACE with ELEMENT clause” • 37, 348, 391
Feature X264, “XML type: schema location” • 37, 348, 391
Feature X271, “XMLValidate: data-driven case” • 90, 391
Feature X272, “XMLValidate: ACCORDING TO clause” • 90, 391
Feature X273, “XMLValidate: ELEMENT clause” • 90, 391

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

Feature X274, “XMLValidate: schema location” • 90, 391
Feature X281, “XMLValidate with DOCUMENT option” • 90, 391, 392
Feature X282, “XMLValidate with CONTENT option” • 90, 392
Feature X283, “XMLValidate with SEQUENCE option” • 90, 392
Feature X284, “XMLValidate: NAMESPACE without ELEMENT clause” • 90, 392
Feature X286, “XMLValidate: NO NAMESPACE with ELEMENT clause” • 90, 392
Feature X300, “XMLTable” • 94, 392
Feature X301, “XMLTable: derived column list option” • 94, 392
Feature X302, “XMLTable: ordinality column option” • 95, 392
Feature X303, “XMLTable: column default option” • 95, 392, 393
Feature X304, “XMLTable: passing a context item” • 95, 393
Feature X305, “XMLTable: initializing an XQuery variable” • 95, 393
Feature X400, “Name and identifier mapping” • 116, 393
XML • 13, 14, 15, 16, 20, 28, *32*, *35*, 36, 42, 43, 44, 47, 52, 60, 62, 63, 64, 65, 67, 68, 71, 73, 75, 79, 80, 81, 82, 84, 86, 87, 92, 93, 139, 181, 182, 183, 184, 185, 186, 188, 189, 213, 216, 217, 221, 222, 224, 229, 236, 238, 239, *269*, 273, 281, *283*, 284, *286*, 287, 288, *291*, 292, *294*, 295, *297*, 298, *300*, 301, *303*, 304, 307, 329, 389, 390, 396, 397
XML 1.0 Name • **8**
XML 1.0 NameChar • **8**
XML 1.0 NCName • **8**
XML 1.0 QName • **9**
XML 1.0 QName local part • **9**
XML 1.0 QName prefix • **9**
XML 1.1 Name • **8**
XML 1.1 NameChar • **8**
XML 1.1 NameStartChar • **8**
XML 1.1 NCName • **8**
XML 1.1 QName • **9**
XML 1.1 QName local part • **9**
XML 1.1 QName prefix • **9**
<XML aggregate> • 51, **224**, 225, 226, 361, 362, 388
<XML attribute> • **8**, **66**, 67
XML attribute information item • **8**
<XML attribute list> • **66**
<XML attribute name> • **66**, 67, 69, 199, 369, 386
<XML attribute value> • **66**, 68
<XML attributes> • **66**, 68
<XML binary encoding> • 25, 44, 68, 71, 84, 96, 97, **227**, 235, 236, 237, 240, 259, 260, 261, 262, 263, 265, 266, 378, 379
<XML binary string serialization> • 17, 52, **53**, 54, 55, 56, 57, 275, 367, 368, 396, 397, 401
XML BLOB host variable • *284*, *288*, *292*, *295*, *298*, *301*, *305*
<XML cast operand> • **44**, 45, 47, 52
<XML cast specification> • 19, *39*, **44**, 45, 46, 47, 48, 50, 52, 81, 361, 396
<XML cast target> • **44**, 45, 46, 52
XML character information item • **8**
<XML character string serialization> • 17, 52, **53**, 54, 55, 56, 57, 273, 275, 367, 368, 396, 401
XML CLOB host variable • *284*, *287*, *291*, 292, *294*, 295, *297*, 298, *301*, *304*
<XML comment> • 16, *59*, **60**, 61, 275, 362, 387, 388, 397
<XML concatenation> • 16, 51, *59*, **62**, 63, 275, 361, 387, 388, 397, 401
<XML content option> • **66**, 67, 68, 69, *70*, 71, 72, 384
<XML content predicate> • 17, *99*, **100**, 101, 275, 370, 401
<XML content predicate part 2> • *40*, **100**
XML declaration • **8**
<XML declaration option> • **53**, 55, 56
<XML default namespace declaration item> • 199, 214, 215, **227**
<XML document> • **7**, 16, *59*, **64**, 65, 275, 361, 387, 388, 397, 401
XML document information item • **8**
<XML document predicate> • 17, *99*, **102**, 103, 275, 370, 401
<XML document predicate part 2> • *40*, **102**
<XML element> • **8**, 16, 25, *59*, **66**, 67, 68, 69, 201, 361, 368, 369, 387, 388, 395, 397
<XML element content> • 16, **66**, 67, 68, 69
XML element information item • **8**
<XML element name> • **66**, 67, 69, 199, 369, 386
<XML encoding name> • **54**, 56, 396, 397
<XML encoding specification> • **53**, 54, 397
<XML exists predicate> • 17, 52, *99*, **104**, 370, 396
<XML forest> • 16, 25, *59*, **70**, 71, 72, 201, 361, 368, 369, 387, 388, 395, 397
XML information item • **8**
<XML iterate> • **91**, 92, 94
<XML lexically scoped option> • 96, **227**, 237
<XML lexically scoped options> • *96*, 97, **227**, *235*, 236, *237*, *240*, *259*, *260*, *261*, *262*, *263*, *265*, 266, 368, 369, 378, 379, 386
XML namespace • **8**

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

<XML namespace declaration> • *66*, 67, 69, *70*, 71, 72, *91*, 92, 95, 96, 198, 199, 214, **227**, 235, 237, 240, 259, 260, 261, 262, 263, 266, 368, 369, 370
<XML namespace declaration item> • 198, **227**, 235, 237, 240, 259, 260, 261, 262, 263, 265
<XML namespace prefix> • **8**, 69, 72, 198, 214, **227**, 228, 369, 386
<XML namespace URI> • 198, 199, 214, **227**, 228
XML option of XML BLOB host variable • *284*, *288*, *292*, *295*, *298*, *301*, *305*, *402*, *403*
XML option of XML CLOB host variable • *284*, *287*, *292*, *295*, *298*, *301*, *304*, *401*, *402*, *403*
XML option of XML VARCHAR host variable • *287*, *304*, *402*, *403*
<XML parse> • 17, 20, *59*, **73**, 74, 272, 366, 397
<XML passing mechanism> • 21, *41*, *44*, 45, 51, 52, *78*, 79, *91*, *92*, 93, 221, 222, 223, **230**, *243*, 244, 245, *255*, 256, 257, 258, *267*, 268, 332, 333, 334, 335, 386, 396, 401
<XML PI> • 16, *59*, **75**, 76, 77, 275, 362, 387, 388, 397
<XML PI target> • **75**, 77, 386
<XML primary> • **58**
<XML query> • 16, 51, *59*, **78**, 79, 80, 81, 82, 83, 385, 386, 396, 398
<XML query argument> • **78**, *91*, 92
<XML query argument list> • **78**, 79, 80, *104*
<XML query context item> • **78**, 79, 81, 82, 95, 385, 393
<XML query default passing mechanism> • **78**
<XML query empty handling option> • **78**, 80, 82, 83, 385, 386
<XML query returning mechanism> • **78**, 79
<XML query variable> • **78**, 79, 81, 82, 83, 95, 385, 386, 393
<XML regular namespace declaration item> • 214, **227**, 228
<XML returning clause> • *60*, 61, *62*, 63, *64*, 65, *66*, 67, 69, *70*, 71, 72, *75*, 77, *78*, 79, *84*, 85, *224*, 226, **229**, 387, 388, 397, 398, 400
XML SCHEMA • 336
XML Schema • **7**
XML Schema built-in data type • **9**
XML Schema complex type • **9**
XML Schema data type • **9**
XML Schema document • **9**
XML Schema primitive type • **9**
XML Schema simple type • **9**
XML Schema type • **9**
<XML serialize bom> • **53**, 55, 56, 57, 366
<XML serialize indent> • **53**, 55, 56, 57, 366, 405
<XML serialize version> • **53**, 54, 55, 56, 396, 397
<XML table> • 17, **91**, 92, 94, 95, 370, 392, 398
<XML table argument list> • 17, **91**, 92, 93, 95, 393
<XML table argument passing mechanism> • **91**, 92
<XML table column definition> • 17, **91**, 92
<XML table column definitions> • 17, **91**
<XML table column pattern> • 17, **92**, 93
<XML table ordinality column definition> • *91*, **92**, 93, 95, 392
<XML table regular column definition> • **92**, 95, 393
<XML table row pattern> • 17, **91**, 92
XML target namespace • **9**
XML target namespace URI • **9**
<XML text> • **9**, 16, *59*, **84**, 85, 275, 362, 387, 388, 398
<XML type> • **35**, 36, 37, 359, 384, 385, 387, 391
<XML type modifier> • **35**, 37, 384
XML types • *13*, 19, 41, 42, 45
XML unmappable column • *28*, 166, 175
XML unmappable data type • *28*
<XML URI> • **231**, 310, 404
<XML valid according to clause> • 51, *86*, 87, 88, 89, 90, *105*, 108, 109, **231**, 233, 379, 391, 398, 400
<XML valid according to identifier> • **231**, 232
<XML valid according to URI> • **231**, 232, 400
<XML valid according to what> • 19, *35*, **231**
<XML valid element clause> • *35*, 37, 51, 88, 89, 90, 107, 108, 109, **231**, 233, 379, 380, 391, 392
<XML valid element name> • **231**, 233, 400
<XML valid element name specification> • 37, 51, 86, 87, 88, 89, 90, 107, 108, 109, **231**, 233, 234, 380, 391, 392
<XML valid element namespace specification> • 37, 51, 86, 87, 88, 89, 90, 107, 108, 109, **231**, 233, 380, 391, 392
<XML valid element namespace URI> • **231**, 233
<XML valid predicate> • 17, 19, 51, *99*, **105**, 106, 107, 108, 109, 275, 379, 380, 398, 401
<XML valid predicate part 2> • *40*, **105**
<XML valid schema location> • 37, 90, 109, **231**, 232, 379, 391
<XML valid schema location URI> • **231**, 232, 400
<XML valid target namespace URI> • **231**, 232, 400
<XML validate> • 17, 19, *59*, **86**, 87, 90, 275, 391, 392, 398, 401
XML value • *15*, 16, 17, 20, 58, 60, 64, 66, 70, 73, 75, 84, 86, 100, 102, 105, 149, 200, 204, 208, 210, 212, 230, 398, 407
<XML value expression> • 14, 20, *51*, 52, *53*, 55, 56, **58**, *62*, *64*, *86*, *91*, 94, 100, 102, 105, *224*, 225, 275, 360, 401
<XML value function> • 51, *58*, **59**, 229, 360
XML value overflow • 225, 345
XML value space • **9**

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

XML values • *13*, 16, 62, 191, 203, 236, 238, 239, 350, 355, 356, 388, 389, 390, 416
XML VARCHAR host variable • *287*, *304*
XML visible column • *28*
XML visible schema • *28*
XML visible table • *28*
<XML whitespace option> • **73**, *283*, 284, 285, *286*, 287, 288, 289, 290, *291*, 292, 293, *294*, 295, 296, *297*, 298, 299, *300*, 301, 302, *303*, 304, 305, 306, 375, 376, 377
XML whitespace option of the XML BLOB host variable • *284*, *288*, *292*, *295*, *298*, *301*, *305*, *402*, *403*
XML whitespace option of the XML CLOB host variable • *284*, *287*, *292*, *295*, *298*, *301*, *304*, *401*, *402*, *403*
XML whitespace option of the XML VARCHAR host variable • *287*, *304*, *402*, *403*
XML-ordered • *13*, 196
XMLAGG • 20, *32*, *224*, 225, 405
XMLATTRIBUTES • *32*, *66*
XMLBINARY • *32*, *227*
XMLCAST • *32*, *44*, 81, 82, 93, 407
XMLCOMMENT • *32*, *60*
XMLCONCAT • *32*, *62*, 63
XMLDECLARATION • *31*, *53*, 55, 56, 57, 222, 273, 279, 280, 368, 400, 401
XMLDOCUMENT • *32*, 60, 63, *64*, 68, 71, 75, 80, 84, 225
XMLELEMENT • *32*, *66*, 68
XMLEXISTS • *32*, 93, *104*
XMLFOREST • *32*, *70*, 71
XMLITERATE • *32*, *91*, 94
XMLNAMESPACES • *32*, *227*
XMLPARSE • 22, *32*, 47, *73*, 160, 161, 168, 169, 177, 178, 201, 222, 272, 278, 281
XMLPI • *32*, *75*
XMLQUERY • *32*, *78*, 80, 93, 94, 104
XMLSCHEMA • 13, 14, 15, *31*, *35*, 36, 37, 42, 43, 44, 62, 63, 86, 87, 181, 182, 183, 184, 185, 186, 216, 217, 224, *231*, 236, 238, 239, 385, 390
XMLSERIALIZE • 22, *32*, *53*, 149, 160, 161, 168, 169, 177, 178, 222, 273, 281, 288, 292, 295, 305
XMLTABLE • *32*, *91*
XMLTEXT • *32*, *84*
XMLVALIDATE • *32*, 43, *86*
XQuery accessor • **9**
XQuery atomic type • **9**
XQuery atomic value • **9**
XQuery attribute node • **9**
XQuery comment node • **9**
XQuery datetime normalized value • **9**
XQuery datetime timezone component • **9**
XQuery document node • **10**
XQuery document node cannot be validated • 88, 346
XQuery dynamic context • **10**
XQuery element node • **10**
XQuery error • **10**, 48, 49, 65, 82, 202, 346
XQuery evaluation with XML 1.0 lexical rules • **10**
XQuery evaluation with XML 1.1 lexical rules • **10**
<XQuery expression> • **10**, **78**, 80, 82, 83, *104*, 386, 398
XQuery expression context • **10**
XQuery expression with XML 1.0 lexical rules • **10**
XQuery expression with XML 1.1 lexical rules • **10**
XQuery formal type notation • **10**
XQuery item • **10**
XQuery namespace node • **10**
XQuery node • **10**
XQuery node identity • **10**
XQuery node property • **10**
XQuery processing instruction node • **10**
XQuery sequence • **10**
XQuery sequence cannot be validated • 87, 345
XQuery serialization error • 206, 207, 345
XQuery serialization normalization • **10**
XQuery static context • **10**
XQuery text node • **10**
XQuery tree • **10**
XQuery variable • **10**

— Y —

YEAR • 133, 146, 342

— Z —

ZONE • 26, 27, 46, 48, 49, 50, 119, 120, 129, 130, 131, 132, 145, 146, 342

© ISO/IEC 2008 – All rights reserved

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

--`,,```,,,,````-`-`,,`,,`,`,,`---

Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale

ICS 35.060

Price based on 411 pages

© ISO/IEC 2008 – All rights reserved
Copyright International Organization for Standardization
Provided by IHS under license with ISO
No reproduction or networking permitted without license from IHS
Not for Resale