

Australian Standard™

**Information technology—Database
languages—SQL**

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



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

The following are represented on Committee IT-027:

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

Keeping Standards up-to-date

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 Standards may also be withdrawn.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

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

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

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

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

This Standard was issued in draft form for comment as DR 05151.

Australian Standard™

Information technology—Database languages—SQL

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

First published as AS ISO/IEC 9075.14—2005.

COPYRIGHT

© Standards Australia

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

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

ISBN 0 7337 6772 9

PREFACE

This Standard was prepared by the Standards Australia Committee IT-027, Data Management and Interchange.

This Standard is identical with, and has been reproduced from ISO/IEC 9075-14:2003, *Information technology—Database languages—SQL—Part 14: XML-Related Specifications (SQL/XML)*.

The objective of this Standard is to provide database designers, administrators and developers with definitions of the ways in which database language SQL can be used with XML.

This Standard is Part 14 of AS 9075, *Information technology—Database languages—SQL*, which is published in parts as follows:

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

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

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

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

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

Only referenced documents that have been adopted as Australian or Australian/New Zealand Standards have been listed.

CONTENTS

Page

1	Scope.....	1
2	Normative references.....	3
2.1	JTC1 standards.....	3
2.2	Other international standards.....	3
3	Definitions, notations and conventions.....	5
3.1	Definitions.....	5
3.1.1	Definitions taken from XML.....	5
3.1.2	Definitions taken from XML Schema.....	5
3.1.3	Definitions provided in Part 14.....	5
3.2	Notation.....	7
4	Concepts.....	9
4.1	Data types.....	9
4.1.1	Naming of predefined types.....	9
4.1.2	Comparison and ordering.....	9
4.2	XML.....	9
4.2.1	Characteristics of XML values.....	9
4.2.2	XML comparison and assignment.....	10
4.2.3	Operations involving XML values.....	10
4.3	Data analysis operations (involving tables).....	11
4.3.1	Aggregate functions.....	11
4.4	SQL-invoked routines.....	11
4.4.1	Routine descriptors.....	11
4.5	SQL-statements.....	12
4.5.1	SQL-statements classified by function.....	12
4.5.1.1	SQL-session statements.....	12
4.6	SQL-sessions.....	12
4.6.1	SQL-session properties.....	12
4.7	XML namespaces.....	13
4.8	Overview of mappings.....	13
4.8.1	Mapping SQL character sets to Unicode.....	14
4.8.2	Mapping SQL <identifier>s to XML.....	14
4.8.3	Mapping SQL data types to XML.....	14
4.8.4	Mapping values of SQL data types to XML.....	16
4.8.5	Visibility of columns, tables, and schemas in mappings from SQL to XML.....	16

4.8.6	Mapping an SQL table to XML.	17
4.8.7	Mapping an SQL schema to XML.	17
4.8.8	Mapping an SQL catalog to XML.	18
4.8.9	Mapping Unicode to SQL character sets.	19
4.8.10	Mapping XML Names to SQL.	19
5	Lexical elements.	21
5.1	<token> and <separator>.	21
6	Scalar expressions.	23
6.1	<data type>.	23
6.2	<field definition>.	25
6.3	<cast specification>.	26
6.4	<value expression>.	28
6.5	<string value function>.	29
6.6	<XML value expression>.	31
6.7	<XML value function>.	32
6.8	<XML concatenation>.	33
6.9	<XML element>.	34
6.10	<XML forest>.	38
6.11	<XML parse>.	41
6.12	<XML root>.	42
7	Query expressions.	45
7.1	<query expression>.	45
8	Predicates.	47
8.1	<predicate>.	47
8.2	<XML document predicate>.	48
9	Mappings.	51
9.1	Mapping SQL <identifier>s to XML Names.	51
9.2	Mapping a multi-part SQL name to an XML Name.	54
9.3	Mapping an SQL table to XML and an XML Schema document.	55
9.4	Mapping an SQL schema to an XML document and an XML Schema document.	60
9.5	Mapping an SQL catalog to an XML document and an XML Schema document.	65
9.6	Mapping an SQL table to XML Schema data types.	70
9.7	Mapping an SQL schema to XML Schema data types.	74
9.8	Mapping an SQL catalog to XML Schema data types.	77
9.9	Mapping an SQL data type to an XML Name.	79
9.10	Mapping a collection of SQL data types to XML Schema data types.	84
9.11	Mapping an SQL data type to a named XML Schema data type.	86
9.12	Mapping an SQL table to an XML element or an XML forest.	88
9.13	Mapping an SQL schema to an XML element.	91
9.14	Mapping an SQL catalog to an XML element.	93
9.15	Mapping SQL data types to XML Schema data types.	95

9.16	Mapping values of SQL data types to values of XML Schema data types.	114
9.17	Mapping XML Names to SQL <identifier>s.	119
10	Additional common rules.	121
10.1	Type precedence list determination.	121
10.2	Type name determination.	122
10.3	Determination of identical values.	123
10.4	Equality operations.	126
10.5	Grouping operations.	127
10.6	Multiset element grouping operations.	128
10.7	Ordering operations.	129
10.8	Determination of [namespace name] property.	130
10.9	Determination of [namespace attributes] property.	132
10.10	Determination of the [in-scope namespaces] property.	134
10.11	Determination of [whitespace element content] property.	136
10.12	Concatenation of two XML values.	137
10.13	Serialization of an XML value.	139
10.14	Parsing a character string as an XML value.	141
11	Additional common elements.	143
11.1	<routine invocation>.	143
11.2	<aggregate function>.	145
11.3	<XML namespace declaration>.	147
12	Schema definition and manipulation.	149
12.1	<column definition>.	149
12.2	<check constraint definition>.	150
12.3	<view definition>.	151
12.4	<assertion definition>.	152
12.5	<user-defined type definition>.	153
12.6	<attribute definition>.	154
12.7	<SQL-invoked routine>.	155
13	SQL-client modules.	159
13.1	Calls to an <externally-invoked procedure>.	159
13.2	<SQL-procedure statement>.	161
13.3	Data type correspondences.	162
14	Data manipulation.	165
14.1	<delete statement: searched>.	165
14.2	<insert statement>.	166
14.3	<merge statement>.	167
14.4	<update statement: positioned>.	168
14.5	<update statement: searched>.	169
15	Control statements.	171
15.1	<compound statement>.	171

16	Session management.....	173
16.1	<set XML option statement>.....	173
17	Dynamic SQL.....	175
17.1	Description of SQL descriptor areas.....	175
17.2	<input using clause>.....	177
17.3	<output using clause>.....	179
18	Embedded SQL.....	181
18.1	<embedded SQL host program>.....	181
18.2	<embedded SQL Ada program>.....	186
18.3	<embedded SQL C program>.....	188
18.4	<embedded SQL COBOL program>.....	191
18.5	<embedded SQL Fortran program>.....	193
18.6	<embedded SQL MUMPS program>.....	195
18.7	<embedded SQL Pascal program>.....	197
18.8	<embedded SQL PL/I program>.....	199
19	Diagnostics management.....	203
19.1	<get diagnostics statement>.....	203
20	Information Schema.....	205
20.1	PARAMETERS view.....	205
20.2	ROUTINES view.....	207
20.3	Short name views.....	210
21	Definition Schema.....	213
21.1	DATA_TYPE_DESCRIPTOR base table.....	213
21.2	PARAMETERS base table.....	215
21.3	ROUTINES base table.....	216
22	The SQL/XML XML Schema.....	217
22.1	The SQL/XML XML Schema.....	217
23	Status codes.....	221
23.1	SQLSTATE.....	221
24	Conformance.....	223
24.1	Claims of conformance to SQL/XML.....	223
24.2	Additional conformance requirements for SQL/XML.....	223
24.3	Implied feature relationships of SQL/XML.....	224
Annex A	SQL Conformance Summary.....	227
Annex B	Implementation-defined elements.....	239
Annex C	Implementation-dependent elements.....	243
Annex D	Incompatibilities with ISO/IEC 9075:1999.....	245
Annex E	SQL feature taxonomy.....	247

Index.....	251
-------------------	------------

TABLES

Page

1	XML namespace prefixes and their URIs.	13
2	SQL/XML Information Item Properties.	124
3	Data type correspondences for Ada.	162
4	Data type correspondences for C.	162
5	Data type correspondences for COBOL.	162
6	Data type correspondences for Fortran.	163
7	Data type correspondences for M.	163
8	Data type correspondences for Pascal.	163
9	Data type correspondences for PL/I.	164
10	Codes used for SQL data types in Dynamic SQL.	175
11	SQL-statement codes.	203
12	SQLSTATE class and subclass values.	221
13	Implied feature relationships of SQL/XML.	224
14	Feature taxonomy for optional features.	247

INTRODUCTION

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

- 1) Clause 1, “Scope”, specifies the scope of this part of ISO/IEC 9075.
- 2) Clause 2, “Normative references”, identifies additional standards that, through reference in this part of ISO/IEC 9075, constitute provisions of this part of ISO/IEC 9075.
- 3) Clause 3, “Definitions, notations and conventions”, defines the notations and conventions used in this part of ISO/IEC 9075.
- 4) Clause 4, “Concepts”, presents concepts related to this part of ISO/IEC 9075.
- 5) Clause 5, “Lexical elements”, defines the lexical elements of the language.
- 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.
- 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 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 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, “Incompatibilities with ISO/IEC 9075:1999”](#), is an informative Annex. It lists incompatibilities with the previous version of ISO/IEC 9075.
- 29) [Annex E, “SQL feature taxonomy”](#), is an informative Annex. It identifies features and packages of the SQL language specified in this part of ISO/IEC 9075 by an identifier and a short descriptive name. This taxonomy is used to specify conformance to the packages specified in this part of ISO/IEC 9075. The feature taxonomy may be used to develop profiles involving the SQL language.

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

AUSTRALIAN STANDARD

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.

This page intentionally left blank.

2 Normative references

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

2.1 JTC1 standards

[Framework] ISO/IEC 9075-1:1999, *Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework)*.

[Foundation] ISO/IEC 9075-2:1999, *Information technology — Database languages — SQL — Part 2: Foundation (SQL/Foundation)*.

[Schemata] ISO/IEC 9075-11:1999, *Information technology — Database languages — SQL — Part 11: Information and Definition Schemas (SQL/Schemata)*.

[UCS] ISO/IEC 10646-1:2000, *Information technology — Universal Multi-Octet Coded Character Set (UCS) — Part 1: Architecture and Basic Multilingual Plane*.

[UCSupp] ISO/IEC 10646-2:2000, *Information technology — Universal Multi-Octet Coded Character Set (UCS) — Part 2: Supplementary Planes*.

2.2 Other international standards

[Unicode 3.0] The Unicode Consortium, *The Unicode Standard, Version 3.0*, Reading, MA, Addison-Wesley Developers Press, 2000. ISBN 0-201-61633-5.

[Unicode 3.1] The Unicode Consortium, *The Unicode Standard, Version 3.1.0, Unicode Standard Annex #27: Unicode 3.1, (which amends The Unicode Standard, Version 3.0)*, 2001-03-23
<http://www.unicode.org/unicode/reports/tr27>

[Unicode15] Davis, Mark and Dürst, Martin, *Unicode Standard Annex #15: Unicode Normalization Forms, Version 22.0*, 2002-03-26, The Unicode Consortium
<http://www.unicode.org/unicode/reports/tr15-22.html>

[Unicode19] Davis, Mark, *Unicode Standard Annex #19: UTF-32, Version 9.0*, 2002-03-27, The Unicode Consortium
<http://www.unicode.org/unicode/reports/tr19-9.html>

[XML] *Extensible Markup Language (XML) Version 1.0 (second edition)*, 2 October, 2000
<http://www.w3.org/TR/REC-xml>

[XPath] *XML Path Language (XPath) Version 1.0*, 16 November, 1999

<http://www.w3.org/TR/xpath>

[Namespaces] *Namespaces in XML*, 14 January, 1999
<http://www.w3.org/TR/REC-xml-names>

[Schema1] (Recommendation) *XML Schema Part 1: Structures*, 2 May, 2001
<http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/>

[Schema2] (Recommendation) *XML Schema Part 2: Datatypes*, 2 May, 2001
<http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/>

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

[Infoset] (Recommendation) *XML Information Set*, 24 October, 2001
<http://www.w3.org/TR/2001/REC-xml-infoset-20011024>

[UniXML] (Note) *Unicode in XML and Other Markup Languages*, 15 December, 2000
<http://www.w3.org/TR/unicode-xml/>

[RFC2396] RFC 2396, *Uniform Resource Identifiers (URS): Generic Syntax*, T. Berners-Lee, R. Fielding, L. Masinter, August, 1998
<http://www.ietf.org/rfc/rfc2396.txt>

[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>

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 part of ISO/IEC 9075, 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 well-formedness constraint

3.1.1.4 XML declaration

3.1.1.5 XML document

3.1.2 Definitions taken from XML Schema

For the purposes of this part of ISO/IEC 9075, 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

3.1.3 Definitions provided in Part 14

For the purposes of this part of ISO/IEC 9075, 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].

- 3.1.3.2 canonical XML Schema literal:** A canonical lexical representation for an XML Schema type ***XST***, as defined in [Schema2]. There is a unique canonical XML Schema literal for each value in the value space of ***XST***.
- 3.1.3.3 metadata:** Data about data. In this International Standard, metadata is included in table descriptors, column descriptors, and so forth, as defined in ISO/IEC 9075-2 and other parts of this International Standard.
- 3.1.3.4 sequence content model:** The content model of an XML Schema complex type described with ***xs:sequence*** as defined in [Schema1].
- 3.1.3.5 URI:** Uniform Resource Identifier as defined in [RFC2396], as updated by [RFC2732].
- 3.1.3.6 valid XML character:** A legal character as defined in [XML].
- 3.1.3.7 XML attribute:** An attribute as defined by [XML].
- 3.1.3.8 XML attribute information item:** An attribute information item, as defined in [Infoset].
- 3.1.3.9 XML character information item:** A character information item, as defined in [Infoset].
- 3.1.3.10 XML comment information item:** A comment information item, as defined in [Infoset].
- 3.1.3.11 XML document information item:** A document information item, as defined in [Infoset].
- 3.1.3.12 XML document type declaration information item:** A document type declaration information item, as defined in [Infoset].
- 3.1.3.13 XML element:** An element as defined by [XML].
- 3.1.3.14 XML element information item:** An element information item, as defined in [Infoset].
- 3.1.3.15 XML information item:** An information item, as defined in [Infoset].
- 3.1.3.16 XML Name:** A Name as defined by [XML].
- 3.1.3.17 XML NameChar:** A NameChar as defined by [XML].
- 3.1.3.18 XML namespace:** An XML namespace as defined by [Namespaces].
- 3.1.3.19 XML namespace prefix:** A namespace prefix as defined by [Namespaces].
- 3.1.3.20 XML namespace information item:** A namespace information item, as defined in [Infoset].
- 3.1.3.21 XML NCName:** An NCName, as defined by rule [4] in [Namespaces].
- 3.1.3.22 XML notation information item:** A notation information item, as defined in [Infoset].
- 3.1.3.23 XML processing instruction information item:** A processing instruction information item, as defined in [Infoset].
- 3.1.3.24 XML QName:** A QName, as defined by rule [6] of [Namespaces].
- 3.1.3.25 XML QName prefix:** A Prefix, as defined by rule [7] of [Namespaces].
- 3.1.3.26 XML QName local part:** A LocalPart, as defined by rule [8] of [Namespaces].
- 3.1.3.27 XML Schema:** A Schema (or schema) as defined by [Schema1].

- 3.1.3.28 XML Schema built-in data type:** A built-in datatype, as defined in [Schema2].
- 3.1.3.29 XML Schema complex type:** A complex type defined by a complex type definition, as defined in [Schema1].
- 3.1.3.30 XML Schema data type:** A datatype, as defined in [Schema2].
- 3.1.3.31 XML Schema document:** A schema document, as defined in [Schema1].
- 3.1.3.32 XML Schema simple type:** A simple type defined by a simple type definition, as defined in [Schema2].
- 3.1.3.33 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.
- 3.1.3.34 XML target namespace:** A target namespace as defined by [Schema1].
- 3.1.3.35 XML target namespace URI:** The URI of an XML target namespace.
- 3.1.3.36 XML text:** A character string that is a substring of a textual XML content, as defined in [Subclause 10.14](#), “Parsing a character string as an XML value”.
- 3.1.3.37 XML unexpanded entity reference information item:** An unexpanded entity reference information item, as defined in [Infoset].
- 3.1.3.38 XML unparsed entity information item:** An unparsed entity information item, as defined in [Infoset].

3.2 Notation

This Subclause modifies [Subclause 3.2](#), “Notation”, in ISO/IEC 9075-2.

Insert this paragraph XML text, when represented in a conventional English-language paragraph, including Rules, is indicated using bold monospace font, for example, **<xsd:element>**. However, XML text that is presented on a separate line, as opposed to being incorporated in an English-language paragraph, and labeled as being XML text in an accompanying paragraph is written in monospace font (but not in boldface). For example:

```
<xsd:element>
```

Insert this paragraph Similarly, when a textual variable in a Rule denotes XML text, then the textual variable is written in italicized bold monospace font, for example, ***FACETP***, and when the same textual variable appears in a separate line that is clearly marked as XML text, the textual variable is italicized but not bold.

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.

This page intentionally left blank.

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

Augment the 2nd paragraph

— The data type XML is referred to as the *XML type*. Values of the XML type are called *XML values*.

4.1.2 Comparison and ordering

This Subclause modifies Subclause 4.1.4, “Comparison and ordering”, in ISO/IEC 9075-2.

Augment the 4th paragraph

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

4.2 XML

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

— The name of the data type (XML).

4.2.1 Characteristics of XML values

An *XML root information item* is a data item defined in the manner of [Infoset] as follows:

An XML root information item has the following properties:

- [children] An ordered list of child information items. The list may contain zero or one XML document type declaration information item, zero or more XML element information item, zero or more XML character information item, zero or more processing instruction information items, and zero or more comment information items. The XML document type declaration information item, if present, is the first information item in the list. There are no constraints on the order of other child information items; in particular, child information items of the various sorts may be intermingled.
- [notations] An unordered set of XML notation information items. Support for this property is implementation-defined.
- [unparsed entities] An unordered set of XML unparsed entity information items. Support for this property is implementation-defined.
- [standalone] An indication of the standalone status of the XML value, either “yes” or “no” or “no value”.
- [version] A string representing the XML version of the XML value represented by the XML root information item, or the special value “no value”.

NOTE 1 — An XML root information item is similar to an XML document information item, with the following modifications: an XML root information item does not have a [document element] property, a [base URI] property, a [character encoding scheme] property, or an [all declarations processed] property; and the [children] property permits more than one XML element information item.

An *SQL/XML information item* is either an XML root information item or one of the following (defined in Subclause 3.1.3, “Definitions provided in Part 14”): an XML attribute information item, an XML character information item, an XML comment information item, an XML document type declaration information item, an XML element information item, an XML namespace information item, an XML notation information item, an XML processing instruction information item, an XML unexpanded entity reference information item, or an XML unparsed entity information item.

An *XML value* is either the null value, or a collection of SQL/XML information items, consisting of exactly one XML root information item, plus any other SQL/XML information items that can be reached recursively by traversing the properties of the SQL/XML information items.

NOTE 2 — A future version of this part of ISO/IEC 9075 may use a data model based on the work of the XML Query Working Group of the World Wide Web Consortium.

4.2.2 XML comparison and assignment

Values of XML type are assignable to sites of XML type.

XML values are not comparable.

4.2.3 Operations involving XML values

<XML element> is an operator that returns an XML value given an XML element name, an optional list of XML attributes, and an optional list of values as the content of the new element. The value of <XML element> can be any value that has a mapping to an XML value.

<XML forest> is an operator that returns an XML value given a list of <forest element>s. An XML element 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.

<XML concatenation> is an operator that returns an XML value by concatenating a list of XML values, as defined in Subclause 6.8, “<XML concatenation>”.

4.3 Data analysis operations (involving tables)

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

4.3.1 Aggregate functions

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

Add to the bulleted list following the 7th paragraph

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

4.4 SQL-invoked routines

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

4.4.1 Routine descriptors

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

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.

4.5 SQL-statements

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

4.5.1 SQL-statements classified by function

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

4.5.1.1 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>

4.6 SQL-sessions

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

4.6.1 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>.

Insert at the end of dash list in 12nd paragraph

— The current XML option.

4.7 XML namespaces

This standard 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 1, “XML namespace prefixes and their URIs”.

Table 1 — XML namespace prefixes and their URIs

XML namespace prefix	XML namespace URI
xsd	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 **xsd**, **xsi**, or **sqlxml** to reference these XML namespaces, but whatever XML namespace prefix it uses shall be associated with the proper URI.

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 specification.

4.8 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.

4.8.1 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. 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 3 — The entity references `<`, `&`, `>`, `'`, and `"`, as defined by [XML] are regarded as each representing a single character in XML, and do not pose an obstacle to defining homomorphic mappings.

4.8.2 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 **xm** 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 **xm** and **Xm** to `_x0078_m` and `_x0058_M`, respectively, whereas the partially escaped does not.

4.8.3 Mapping SQL data types to XML

For each SQL type or domain, with the exception of structured types, reference types, and the interval type, there is a corresponding XML Schema type. The mapping is fully specified in [Subclause 9.15, “Mapping SQL data types to XML Schema data types”](#). The following is a conceptual description of this mapping.

In general, each SQL predefined type or domain *SQLT* is mapped to the XML Schema built-in data 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 **xsd: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 **xsd:length** is used. Otherwise (*i.e.*, CHARACTER when the mapping is not homomorphic, as well as CHARACTER VARYING and CHARACTER LARGE OBJECT), the facet **xsd: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 **xsd:hexBinary** or the XML Schema type **xsd:base64Binary**. It is implementation-defined which of these types is used to map the SQL binary string types. The **xsd: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.

The exact numeric SQL types NUMERIC and DECIMAL are mapped to the XML Schema type **xsd:decimal** using the facets **xsd:precision** and **xsd:scale**. It is implementation-defined whether the SQL types INTEGER, SMALLINT, and BIGINT are mapped to the XML Schema type **xsd:integer** using the facets **xsd:maxInclusive** and **xsd:minInclusive** or to the closest XML Schema type that is a subtype of **xsd:integer**, using the facets **xsd:maxInclusive** and **xsd: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 **xsd: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 **xsd: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 **xsd:boolean**. Optionally, an annotation indicates the SQL type (BOOLEAN).

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

The SQL types TIME WITHOUT TIME ZONE and TIME WITH TIME ZONE are mapped to the XML Schema type **xsd:time**. The **xsd: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 `xsd:dateTime`. The `xsd: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.

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 distinct type is mapped to an XML Schema simple type by mapping the source type of the distinct 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.

4.8.4 Mapping values of SQL data types to XML

For each SQL type or domain $SQLT$, 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 9.16, “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"`.

4.8.5 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, and the interval type. 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 6.12, “<cast specification>”](#), in ISO/IEC 9075-2. A column C of table T is an *XML visible column* of T for authorization identifier U if 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 .

4.8.6 Mapping an SQL table to XML

Subclause 9.3, “Mapping an SQL table to XML and an XML Schema document”, defines a mapping of an SQL table to an XML Schema document that describes the structure of the mapped XML and either an XML document or an XML forest of elements. Only base tables and viewed tables may be the source of this mapping.

NOTE 4 — This part of this International Standard specifies no syntax for invoking the mapping specified in Subclause 9.3, “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.

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:

- Whether to map the table to an XML forest of 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"**.

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

4.8.7 Mapping an SQL schema to XML

Subclause 9.4, “Mapping an SQL schema to an XML document and an XML Schema document”, defines a mapping between the tables of an SQL schema and two documents, an XML document that represents the data in these tables, and an XML Schema document that describes the first document.

NOTE 5 — This part of this International Standard specifies no syntax for invoking the mapping specified in Subclause 9.4, “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.

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 allows the invoker to specify:

- Whether to map the table to an XML forest of 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"**.

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

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.

4.8.8 Mapping an SQL catalog to XML

Subclause 9.5, “Mapping an SQL catalog to an XML document and an XML Schema document”, defines a mapping between the tables of an SQL catalog and two documents, an XML document that represents the data in these tables, and an XML Schema document that describes the first document.

NOTE 6 — This part of this International Standard specifies no syntax for invoking the mapping specified in Subclause 9.5, “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.

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 the table to an XML forest of 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 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"**.

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

4.8.9 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*.

4.8.10 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 9.17, “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 `_xNNNN` or `_xNNNNNN` where *N* denotes a hexadecimal digit. Such sequences are converted to the character of SQL_TEXT that corresponds to the Unicode code point U+0000NNNN or U+00NNNNNN, respectively.

NOTE 7 — This part of this International Standard specifies no syntax for invoking the mapping specified in [Subclause 9.17, “Mapping XML Names to SQL <identifier>s”](#). This specification is intended to be used by applications and referenced by other standards.

NOTE 8 — The sequence of mappings from SQL <identifier> to XML Name 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>.

This page intentionally left blank.

5 Lexical elements

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

5.1 <token> and <separator>

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

Function

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

Format

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

    | CONTENT
    | DOCUMENT
    | PRESERVE
    | STANDALONE | STRIP
    | VERSION
    | WHITESPACE

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

    | XML | XMLAGG | XMLATTRIBUTES | XMLCONCAT | XMLELEMENT | XMLFOREST
    | XMLNAMESPACES | XMLPARSE | XMLROOT | XMLSERIALIZE
```

Syntax Rules

No additional Syntax Rules.

Access Rules

No additional Access Rules.

General Rules

No additional General Rules.

Conformance Rules

No additional Conformance Rules.

6 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
```

Syntax Rules

- 1) Insert after SR 40 XML specifies the XML data type.

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 name of the data type (XML).

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 the XML type or a distinct type whose source type is the 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 the XML type or a distinct type whose source type is the XML type.

6.2 <field definition>

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 the XML type or a distinct type whose source type is the XML type.

6.3 <cast specification>

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

Function

Specify a data conversion.

Format

No additional Format items.

Syntax Rules

- 1) 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:

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

Where:

EN = Exact Numeric
 AN = Approximate Numeric
 C = Character (Fixed- or Variable-length, or character large object)
 FC = Fixed-length Character
 VC = Variable-length Character
 CL = Character Large Object
 D = Date
 T = Time
 TS = Timestamp

YM = Year-Month Interval
 DT = Day-Time Interval
 BO = Boolean
 UDT = User-Defined Type
 BL = Binary Large Object
 RT = Reference type
 CT = Collection type
 RW = Row type
 DL = Datalink
 XML = XML type

Access Rules

No additional Access Rules.

General Rules

- 1) Insert after GR 2) If *SD* is XML, then *TV* is *SV*.

Conformance Rules

- 1) Insert this CR Without Feature X010, “XML type”, conforming SQL language shall not contain a <cast operand> whose declared type is XML.

6.4 <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)n) An <aggregate function> that specifies <XML aggregate>.

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.

6.5 <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 for ISO/IEC 9075-2
    | <XML serialize>

<XML serialize> ::=
    XMLSERIALIZE <left paren> <document or content> <XML value expression>
    AS <data type> <right paren>

<document or content> ::=
    DOCUMENT
    | CONTENT
```

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>, <fold>, <transcoding>, <character transliteration>, <trim function>, <character overlay function>, <normalize function>, <specific type method>, or <XML serialize>.
- 2) [Insert this SR](#) If <XML serialize> is specified, then <data type> shall be a character string type. The declared type of <XML serialize> is <data type>.

Access Rules

No additional Access Rules.

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>, <fold>, <transcoding>, <character transliteration>, <trim function>, <character overlay function>, <normalize function>, <specific type method>, or <XML serialize>.
- 2) [Insert this GR](#) If <XML serialize> is specified, then let *DC* be the <document or content>, let *XMLV* be the value of the <XML value expression>, and let *DT* be the <data type>. The result of <XML serialize> is the result determined by applying the General Rules of [Subclause 10.13](#), “Serialization of an XML value”, with *DC* as SYNTAX, *XMLV* as VALUE, and *DT* as TYPE.

Conformance Rules

- 1) Without Feature X070, “XMLSerialize: CONTENT option”, in conforming SQL language, <XML serialize> shall not immediately contain a <document or content> that is CONTENT.
- 2) Without Feature X071, “XMLSerialize: DOCUMENT option”, in conforming SQL language, <XML serialize> shall not immediately contain a <document or content> that is DOCUMENT.

6.6 <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 <value expression primary> shall be XML.
- 2) The declared type of <XML value expression> is XML.

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>.

6.7 <XML value function>

Function

Specify a function that yields a value of type XML.

Format

```
<XML value function> ::=
  <XML concatenation>
  | <XML element>
  | <XML forest>
  | <XML parse>
  | <XML root>
```

Syntax Rules

- 1) The declared type of <XML value function> is XML.

Access Rules

None.

General Rules

- 1) The result of an <XML value function> is the XML value of the immediately contained <XML forest>, <XML concatenation>, <XML element>, <XML parse>, or <XML root>.

Conformance Rules

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

6.8 <XML concatenation>

Function

Concatenate a list of XML values.

Format

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

Syntax Rules

None.

Access Rules

None.

General Rules

- 1) Let n be the number of <XML value expression>s immediately contained in <XML concatenation>. Let $XV_1, XV_2, \dots, XV_n$ be the results of these <XML value expression>s.
- 2) Let R_0 be the null value.
- 3) For all i , $1 \text{ (one)} \leq i \leq n$, let R_i be the concatenation of R_{i-1} and XV_i , as determined by applying the General Rules of [Subclause 10.12, “Concatenation of two XML values”](#).
- 4) The result of <XML concatenation> is R_n .

Conformance Rules

- 1) Without Feature X020, “XML concatenation”, conforming SQL language shall not contain an <XML concatenation>.

6.9 <XML element>

Function

Generate an XML value with a single XML element information item as a child of its XML root information item.

Format

```

<XML element> ::=
    XMLELEMENT <left paren> NAME <XML element name>
    [ <comma> <XML namespace declaration> ] [ <comma> <XML attributes> ]
    [ { <comma> <XML element content> }... ] <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>

```

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 .
 - c) The declared type of AV_i shall be a distinct type or a predefined type other than XML.
 - d) Case:
 - i) If A_i contains an <XML attribute name> AN_i , then let ANC_i be the result of the partially escaped mapping from an SQL <identifier> to an XML Name specified in Subclause 9.1, “Mapping SQL <identifier>s to XML Names”, applied to 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 .

- e) ANC_i shall be an XML 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 10.8, “Determination of [namespace name] property”, 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 , $NSURI_i$ equals $NSURI_j$ according to the UCS_BASIC collation, 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> after the partially escaped mapping from an SQL <identifier> to an XML Name specified in Subclause 9.1, “Mapping SQL <identifier>s to XML Names”, has been applied. EN shall be an XML QName.
 - 6) The Syntax Rules of Subclause 10.8, “Determination of [namespace name] property”, 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 distinct type, or a collection type.

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 6.12, “<cast specification>”, in ISO/IEC 9075-2, are applied to:

CAST (VE AS ST)

General Rules

- 1) Let EI be an element information item whose properties are initially set to unknown.
- 2) Let R be an XML root information item whose properties are set as follows:
 - a) The [children] property is a list with one element, EI .
 - b) The [notations] property is the empty set.
 - c) The [unparsed entities] property is the empty set.
 - d) The [version] property is “no value”.
 - e) The [standalone] property is “no value”.
- 3) Let i range from 1 (one) to n . If the value of AV_i is not null, then:

- a) Let CAV_i be the result of applying the General Rules of Subclause 9.16, “Mapping values of SQL data types to values of XML Schema data types”, to AV_i resulting in a character string CAV_i of Unicode characters.
- b) Let AII_i be an XML attribute information item having the following properties:
 - i) The [local name] property is the XML QName local part of ANC_i .
 - ii) The [prefix] property is the XML QName prefix of ANC_i , if any; otherwise, “no value”.
 - iii) The [namespace name] property is $NSURI_i$.
 - iv) The [normalized value] is the result of applying the rules of [XML], section 3.3.3, “Attribute Value Normalization”, to CAV_i .
 - v) The [specified] property indicates that the attribute was specified rather than defaulted.
 - vi) The [attribute type] property is “CDATA”.
 - vii) The [references] property is “no value”.
 - viii) The [owner element] property is EI .

- 4) Let m be the number of occurrences of <XML element content> in <XML element> that are not the null value. Let EC_j , $1 \text{ (one)} \leq j \leq m$, be the j -th non-null <XML element content> contained in <XML element>.

For each j , $1 \text{ (one)} \leq j \leq m$:

- a) Case:
 - i) If the declared type of EC_j is XML, then let CEC_j be a variable whose value is identical to EC_j .
 - ii) Otherwise, let $XMLV_j$ be the result of applying the General Rules of Subclause 9.16, “Mapping values of SQL data types to values of XML Schema data types”, to EC_j . $XMLV_j$ is a character string of Unicode characters. Let CEC_j be the result of

XMLPARSE (CONTENT $XMLV_j$ PRESERVE WHITESPACE)

- b) Let RII_j be the XML root information item of CEC_j . Let $LXII_j$ be the [children] property of RII_j . (Thus, $LXII_j$ is a list of individual XML information items $LXII_{j,k}$, for k between 1 (one) and the number of XML information items in the list, $m(j)$.) The [parent] property of each XML information item $LXII_{j,k}$, for k between 1 (one) and $m(j)$, is set to EI .
 - i) If the [notations] property of RII_j is not empty, then it is implementation-defined whether the [notations] property of RII_j is copied to the [notations] property of R .
 - ii) If the [unparsed entities] property of RII_j is not empty, then it is implementation-defined whether the [unparsed entities] property of RII_j is copied to the [unparsed entities] property of R .

- 5) The properties of EI are set as follows:

- a) The [parent] property is set to R .

- b) The [local name] property is set to the XML QName local part of *EN*.
 - c) The [prefix] property is set to “no value” if *EN* has no XML QName prefix; otherwise, it is set to the XML QName prefix of *EN*.
 - d) The [namespace name] property is set to *ENSURI*.
 - e) The [children] property is the list of XML information items obtained by concatenating the lists *LXII_j*, for *j* from 1 (one) through *m*, in that order.
 - f) The [attributes] property is the set of XML attribute information items *AI_l* for all *l*, $1 \text{ (one)} \leq l \leq n$, such that *AV_l* is not null.
 - g) The [namespace attributes] property is set to the value determined by evaluation of the General Rules of Subclause 10.9, “Determination of [namespace attributes] property”, with *EI* as the *INFOITEM*.
 - h) The [base URI] property is set to an implementation-dependent value.
 - i) The [in-scope namespaces] property, and of all XML element information items that are descendants of *EI*, are the values determined by evaluation of the General Rules of Subclause 10.10, “Determination of the [in-scope namespaces] property”, with *EI* as the *INFOITEM*.
- 6) The General Rules of Subclause 10.11, “Determination of [whitespace element content] property”, are executed, with *EI* as the *INFOITEM*.
 - 7) The result of <XML element> is the XML value that is the collection of SQL/XML information items consisting of *R* together with every SQL/XML information item that can be reached by traversing the properties of *R*.

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>.

6.10 <XML forest>

Function

Generate an XML value with a list of XML element information items as the children of its XML root information item.

Format

```

<XML forest> ::=
    XMLFOREST <left paren> [ <XML namespace declaration> <comma> ]
    <forest element list> <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 distinct type, or a collection type.
 - c) Case:
 - i) If F_i contains a <forest element name> FEN_i , then let FNC_i be the result of the partially escaped mapping from an SQL <identifier> to an XML Name as specified in Subclause 9.1, “Mapping SQL <identifier>s to XML Names”, applied to 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 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) FNC_i shall be an XML QName.
 - e) The Syntax Rules of Subclause 10.8, “Determination of [namespace name] property”, are applied, with FNC_i as $QNAME$ and with the <XML forest> as $BNFTERM$, resulting in an XML namespace URI $NSURI_i$.