ISO/IEC JTC 1/SC 32 N 1215

Date: 2004-12-16

REPLACES: --

ISO/IEC JTC 1/SC 32

Data Management and Interchange

Secretariat: United States of America (ANSI)

Administered by Pacific Northwest National Laboratory on behalf of ANSI

DOCUMENT TYPE	Final Text Submitted for COR Publication
TITLE	Technical Corrigenda – ISO/IEC 9075-14 Information technology - Database Languages - SQL - Part 14: SQL/XML (for SQL:200n)
SOURCE	SC 32 Secretariat
PROJECT NUMBER	1.32.03.05.99.00
STATUS	This is sent to ITTF for publication
REFERENCES	
ACTION ID.	ITTF
REQUESTED ACTION	
DUE DATE	
Number of Pages	6
LANGUAGE USED	English
DISTRIBUTION	P & L Members SC Chair WG Conveners and Secretaries

Douglas Mann, Secretariat, ISO/IEC JTC 1/SC 32

Pacific Northwest National Laboratory *, 13667 Legacy Circle Apt H, Herndon, VA, 20171, United States of America

Telephone: +1 202-566-2126; Facsimile; +1 202-566-1639; E-mail: MannD@battelle.org

available from the JTC 1/SC 32 WebSite http://www.jtc1sc32.org/

*Pacific Northwest National Laboratory (PNL) administers the ISO/IEC JTC 1/SC 32 Secretariat on behalf of ANSI

ISO/IEC JTC 1/SC 32

Date: 2004-12-10

COR ISO/IEC 9075-14:2004 (E)

ISO/IEC JTC 1/SC 32/WG 3

Nederlands Normalisatie Instituut (NNI)

Information technology — Database languages — SQL —

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

TECHNICAL CORRIGENDUM 1

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

Partie 14: Specifications à XML (SQL/XML)

RECTIFICATIE TECHNIQUE 1

Document type: Corridenga
Document subtype: Technical Corrigendum (COR)
Document stage: (5) IS Publication
Document language: English

Statement of purpose for rationale:

A statement indicating the rational for each change to ISO/IEC 9075 is included. This is to inform the users of that standard as to the reason why it was judged necessary to change the original wording. In many cases, the reason is editorial or to clarify the wording; in some cases, it is to correct an error or an omission in the original wording.

Notes on numbering:

Where this Corrigendum introduces new Syntax, Access, General, and Conformance Rules, the new rules have been numbered as follows:

> Rules inserted between, for example, Rules 7) and 8) are numbered 7.1), 7.2), etc. [or 7)a.1), 7)a.2), etc.]. Those inserted before Rule 1) are numbered 0.1), 0.2, etc.

Where this Corrigendum introduces new Subclauses, the new Subclauses have been numbered as follows:

> Subclauses inserted between, for example, Subclause 4.3.2 and Subclause 4.3.3 are numbered 4.3.2a, 4.3.2b, etc. Those inserted before, for example, 4.3.1 are numbered 4.3.0, 4.3.0a, etc.

Contents

Page

Tables

Information technology — Database languages — SQL —

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

TECHNICAL CORRIGENDUM 1

Foreword

1. *Rationale: Remove incorrect reference to obsolete part.*

   In the 6th paragraph, delete the 5th bullet.

2 Normative references

2.1 JTC1 standards

1. *Rationale: Editorial: Correct date in reference.*

   Replace the 1st three references with:

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

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

   [PSM] ISO/IEC 9075-4:2003, *Information technology — Database languages — SQL — Part 4: Persistent Stored Modules (SQL/PSM)*

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

4 Concepts

4.2 XML

4.2.3 Operations involving XML values

1. *Rationale: Editorial.*

   Replace the 1st paragraph with:

   <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 content> can be any value that has a mapping to an XML value.

4.3 Data analysis operations (involving tables)

4.3.1 Aggregate functions

1. *Rationale: XMLAGG concatenates all kinds of XML information items, not just elements.*

   Replace the bulleted item with:

   Add to the bulleted list following the 7th paragraph

   — If XMLAGG is specified, then an XML value formed by concatenating the [children] property of the XML root information item from the <XML value expression> evaluated for each row that qualifies.

4.8 Overview of mappings

4.8.3 Mapping SQL data types to XML

1. *Rationale: Add an allowance for distinct types, and correct an improper characterization of **`XMLT`** as "XML Schema built-in data type".*

   Replace the 1st sentence of the 2nd paragraph with:

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*.

4.8.4 Mapping SQL data types to XML

1. *Rationale: Add an exception for structured and reference types to the description of the supported value space mapping.*

   Replace the 1st sentence of the 1st paragraph with:

   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.

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

1. *Rationale: Correct the definition of XML visible column.*

   Replace the last sentence of the 2nd paragraph with:

   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.

4.8.6 Mapping an SQL table to XML

1. *Rationale: Clarify the requirements for arguments to be supplied to the invocation of a Subclause.*

   Replace Note 4 with:

   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. 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.

2. *Rationale: Use correct XML Schema terminology.*

   Replace the 4th paragraph with:

   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.

4.8.7 Mapping an SQL schema to XML

1. *Rationale: Clarify the requirements for arguments to be supplied to the invocation of a Subclause.*

   Replace Note 5 with:

   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. 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.

2. *Rationale: Use correct XML Schema terminology.*

   Replace the 4th paragraph with:

   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.

4.8.8 Mapping an SQL catalog to XML

1. *Rationale: Clarify the requirements for arguments to be supplied to the invocation of a Subclause.*

   Replace Note 6 with:

   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. 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.

2. *Rationale: Use correct XML Schema terminology.*

   Replace the 4th paragraph with:

   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.

4.8.10 Mapping XML Names to SQL

1. *Rationale: Clarify the requirements for arguments to be supplied to the invocation of a Subclause.*

   Replace Note 7 with:

   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. 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.

6 Scalar expressions

6.9 <XML element>

1. *Rationale: The value of an attribute may not be given by a distinct type whose source type is XML.*

   Replace Syntax Rule 3) c) with:

   3) ...

      c) The declared type of AV_i shall be either a distinct type whose source type is not XML, or a predefined type other than XML.

2. *Rationale: A collection type of an unmappable type is not mappable.*

   Replace Syntax Rule 7) with:

   7) For each <XML element content> *XEC*, the declared type of *XEC* shall be a predefined type, a distinct type, or a collection type that is not based on an XML unmappable data type.

3. *Rationale: Invoke Subclause 9.16, "Mapping values of SQL data types to values of XML Schema data types", with the correct number of parameters.*

   Replace General Rule 3) a) with:

   3) ...

      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, "absent" as *NULLS*, and *False* as *CHARMAPPING*, resulting in a character string CAV_i of Unicode characters.

4. *Rationale: Clarify what the value of the [normalized value] property is.*

   Replace General Rule 3) b) iv) with:

   3) ...

      b) ...

         iv) The [normalized value] is CAV_i.

5. *Rationale: Partially escaped mapping of SQL identifiers is not applicable for explicitly specified <XML attribute name>s.*

   Replace Syntax Rule 3) d) i) with:

   3) ...

      d) ...

         i) If A_i contains an <XML attribute name> AN_i, then let ANC_i be AN_i.

6. *Rationale: Editorial. Remove double "not"*

   Replace Syntax Rule 3) f) with:

   3) ...

      f) ANC_i shall not be equivalent to '**`xmlns`**', and ANC_i shall not have an XML QName prefix that is equivalent to '**`xmlns`**'.

7. *Rationale: Invoke Subclause 9.16, "Mapping values of SQL data types to values of XML Schema data types", with the correct number of parameters.*

   Replace General Rule 4) a) ii) with:

   4) ...

      a) ...

         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, "absent" as *NULLS*, and <u>*True*</u> as the *CHARMAPPING*. $XMLV_j$ is a character string of Unicode characters. Let CEC_j be the result of

         ```
         XMLPARSE ( CONTENT XMLVj PRESERVE WHITESPACE )
         ```

8. *Rationale: Partially escaped mapping of SQL identifiers is not applicable for explicitly specified <XML element name>s.*

   Replace Syntax Rule 5) with:

   5) Let *EN* be the character representation of <XML element name>. *EN* shall be an XML QName.

9. *Rationale: Correct the definition of the [element whitespace content] property.*

   Replace General Rule 6) with:

   6) For each XML character information item *CI* that is included in the [children] property of *EI*, the [element content whitespace] property is set as follows:

      Case:

a) If the [character code] property of *CI* is a white space character, as defined in [XML] section 2.10, “White space handling”, then *Unknown*.

b) Otherwise, *False*.

6.10 <XML forest>

1. *Rationale: A collection type of an unmappable type is not mappable.*

Replace Syntax Rule 2) b) with:

2) ...

b) Let FV_i be the <forest element value> immediately contained in F_i. The declared type of FV_i shall neither be an XML unmappable type, nor be based on an XML unmappable type.

2. *Rationale: Invoke Subclause 9.16, “Mapping values of SQL data types to values of XML Schema data types”, with the correct number of parameters.*

Replace General Rule 2) b) ii) 2) B) with:

2) ...

b) ...

ii) ...

2) ...

B) Otherwise, let CS_i be result of applying the General Rules of Subclause 9.16, “Mapping values of SQL data types to values of XML Schema data types”, to V_i, “absent” as *NULLS*, and *True* as the *CHARMAPPING*. Let C_i be the result of

```
XMLPARSE ( CONTENT CSi PRESERVE WHITESPACE )
```

3. *Rationale: Correct the definition of the [element whitespace content] property.*

Replace General Rule 2) b) ii) 5) with:

2) ...

b) ...

ii) ...

5) For each XML character information item *CI* that is included in the [children] property of E_i, the [element content whitespace] property is set as follows:

Case:

A) If the [character code] property of *CI* is a white space character, as defined in [XML] section 2.10, "White space handling", then *Unknown*.

B) Otherwise, *False*.

4. *Rationale: Partially escaped mapping of SQL identifiers is not applicable for explicitly specified <forest element name>s.*

Replace Syntax Rule 2) c) i) with:

2) ...

c) ...

i) If F_i contains a <forest element name> FEN_i, then let FNC_i be FEN_i.

6.11 <XML parse>

1. *Rationale: Specify the result if the input is null.*

Replace General Rule 4) with:

4) Case:

a) If *V* is the null value, then the result of <XML parse> is the null value.

b) Otherwise, the The result of <XML parse> is determined by applying the General Rules of Subclause 10.14, "Parsing a character string as an XML value", with *DC* as *SYNTAX*, *V* as *TEXT*, and *WO* as *OPTION*.

6.12 <XML root>

1. *Rationale: Specify the result if the input is null.*

Replace General Rule 1) with:

1) If the value of the <XML value expression> is the null value, then the result of <XML root> is the null value.

1.1) Otherwise, let *V* be a variable whose value is identical to the value of the <XML value expression>.

2. *Rationale: Specify the result if the input is null.*

Replace General Rule 2) b) with:

2) ...

b) If the value of the <string value expression> is not the null value, then the value of the <string value expression>.

NOTE 8.1 — If the value of the <string value expression> is the null value, then the value of the [version] property is not changed.

8 Predicates

8.2 <XML document predicate>

1. *Rationale: Correct the rules for how <XML document predicate>'s value is determined.*

Replace General Rule 2) with:

2) Case:

a) If *V* is the null value, then the result of

```
V IS DOCUMENT
```

is *Unknown*.

b) Otherwise, let *C* be the [children] property of the XML root information item of *V*.

Case:

i) If the number of XML element information items contained in *C* is one and the number of XML character information items contained in *C* is zero, then the result of

```
V IS DOCUMENT
```

is *True*.

ii) Otherwise, the result of

```
V IS DOCUMENT
```

is *False*.

9 Mappings

9.4 Mapping an SQL schema to an XML document and an XML Schema document

1. *Rationale: Report the completion condition warning — column cannot be mapped to XML whether or not METADATA is <u>True</u>.*

Insert new General Rule:

2.1) 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 13.1 — "visible column" is defined in Subclause 4.8.5, "Visibility of columns, tables, and schemas in mappings from SQL to XML".

2. *Rationale: Define CT in terms of XML visible columns.*

Replace General Rules 3) a) with:

3) ...

a) Let *CT* be the XML visible columns of the viewed and base tables contained in *S* for *U*.

9.5 Mapping an SQL catalog to an XML document and an XML Schema document

1. *Rationale: Report the completion condition warning — column cannot be mapped to XML whether or not METADATA is <u>True</u>.*

Insert new General Rule:

2.1) 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 14.1 — "visible column" is defined in Subclause 4.8.5, "Visibility of columns, tables, and schemas in mappings from SQL to XML".

2. *Rationale: Define CT in terms of XML visible columns.*

Replace General Rules 3) a) with:

3) ...

a) Let *CT* be the XML visible columns of the viewed and base tables contained in *C* for *U*.

9.6 Mapping an SQL table to XML Schema data types

1. *Rationale: Editorial.*

   Replace General rule 1) with:

   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 an XML forest of 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.

9.7 Mapping an SQL schema to XML Schema data types

1. *Rationale: Editorial.*

   Replace General rule 1) with:

   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 an XML forest of 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.

9.12 Mapping an SQL table to an XML element or an XML forest

1. *Rationale: Invoke Subclause 9.16, "Mapping values of SQL data types to values of XML Schema data types", with the correct number of parameters.*

   Replace General Rule 6) b) v) 3) A) with:

   6) ...

      b) ...

         v) ...

            3) ...

               A) Let $XMLV_j$ be the result of applying the mapping defined in Subclause 9.16, "Mapping values of SQL data types to values of XML Schema data types", using V_j as the SQL data value *DV*, *NULLS* as the choice of whether to map null values to absent elements (absent) or to elements that are marked with **`xsi:nil="true"`** (nil), and *True* as *CHARMAPPING*.

9.15 Mapping SQL data types to XML Schema data types

1. *Rationale: Correct the General Rules associated with mapping SQL's INTEGER, SMALLINT, and BIGINT types to their corresponding XML Schema data types.*

Replace General Rule 7) d) iv) with:

7) ...

d) ...

iv) It is implementation-defined whether *REST* is

```
<xsd:restriction base="xsd:integer">
  FACETMAX
  FACETMIN
</xsd:restriction>
```

or determined by

1) Case:

A) If there is no row in Table 1.1, “Constraining facets of XML Schema integer types”, such that *MAX* is less than or equal to the value in the column “maxInclusive” and *MIN* is greater than or equal to the value in the column “minInclusive” then:

I) Let *TYPE* be xsd:integer.

II) Let *FMAX* be ***FACETMAX***.

III) Let *FMIN* be ***FACETMIN***.

B) Otherwise:

I) Let *TYPE* be the contents of the column “Type” in Table 1.1, “Constraining facets of XML Schema integer types”, taken from the first row from the top of the table for which *MAX* is less than or equal to the value in the column “maxInclusive” and *MIN* is greater than or equal to the value in the column “minInclusive”.

II) If *MAX* is equal to the value of the column “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 column “minInclusive” in the selected row of the table, then let *FMIN* be the zero-length string; otherwise, let *FMIN* be ***FACETMIN***.

C) Let *REST* be

```
<xsd:restriction base="TYPE">
  FMAX
  FMIN
</xsd:restriction>
```

Table 1.1 — Constraining facets of XML Schema integer types

Type	minInclusive	maxInclusive
xsd:unsignedByte	0	255
xsd:byte	-128	127
xsd:unsignedShort	0	2^{16}-1 (65,535)
xsd:short	-2^{15} (-32,768)	2^{15}-1 (32,767)
xsd:unsignedInt	0	2^{32}-1 (4,294,967,295)
xsd:int	-2^{31} (-2,147,483,648)	2^{31}-1 (2,147,483,647)
xsd:unsignedLong	0	2^{64}-1 (18,446,744,073,709,551,615)
xsd:long	-2^{63} (-9,223,372,036,854,775,808)	2^{63}-1 (9,223,372,036,854,775,807)

2. *Rationale: Remove the erroneous close of <xsd:restriction>.*

Replace General Rule 7) d) v) with:

7) ...

d) ...

v) ***XMLT*** is the XML Schema type defined by:

```
<xsd:simpleType>
  ANN
  REST
</xsd:simpleType>
```

9.16 Mapping values of SQL data types to values of XML Schema data types

1. *Rationale: Control the entitization of certain characters with an explicit parameter.*

Insert the following General Rule:

2.1) 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*).

2. *Rationale: Limit application of the CAST to only necessary, and supported, types.*

   Replace General Rules 5) with:

   5) 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) )
   ```

3. *Rationale: Control the entitization of certain characters with an explicit parameter.*

   Replace General Rule 7) a) ii) with:

   7) ...

      a) ...

         ii) Case:

            1) 1) If *CHARMAPPING* is True, then let *XMLV* be *XMLVRAW*, with each instance of “&” (U+0026) replaced by “&amp;”, each instance of “<” (U+003C) replaced by “&lt;”, each instance of “>” (U+003E) replaced by “&gt;”, and each instance of Carriage Return (U+000D) replaced by “&#x0d;”.

            2) Otherwise, let *XMLV* be *XMLVRAW*.

9.17 Mapping XML Names to SQL <identifier>s

1. *Rationale: Editorial.*

   Replace General Rule 2) a) i) with:

   2) ...

      a) ...

         i) If the Unicode code point U+$\boldsymbol{X_{i+2}X_{i+3}X_{i+4}X_{i+5}}$ 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.

2. *Rationale: Editorial.*

   Replace General Rule 2) b) i) with:

   2) ...

      b) ...

i) If the Unicode code point U+$\mathtt{X}_{i+2}\mathtt{X}_{i+3}\,\mathtt{X}_{i+4}\mathtt{X}_{i+5}\,\mathtt{X}_{i+6}\mathtt{X}_{i+7}$ 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.

10 Additional common rules

10.3 Determination of identical values

1. *Rationale: Correct placement of inserted General Rule.*

   Replace the modification instruction for General Rule 1) with:

   1) $\boxed{\text{Insert after GR 2) c)}}$

2. *Rationale: Editorial.*

   Replace the lead text of General Rule 1) b) with:

   1) ...

      a) A property *P* of an SQL/XML information item *XII1* is identical to the same property *P* of another XML information item *XII2*:

10.8 Determination of [namespace name] property

1. *Rationale: Cater for the implicitly defined namespace prefix 'xml'.*

   Replace Syntax Rules 2) a) ii), 2) a) iii), 2) a) iv), and 2) a) v) with:

   2) ...

      a) ...

         ii) Case:

            1) If *P* is equivalent to `'xml'`, then let *NSURI* be `http://www.w3.org/XML/1998/namespace`.

            2) Otherwise:

               A) *B* shall be within the scope of one or more <XML namespace declaration>s that contain an <XML namespace prefix> equivalent to *P*.

               B) Let *XND* be the <XML namespace declaration> that contains an <XML namespace prefix> equivalent to *P* with innermost scope that includes *B*.

C) Let *XNDI* be the <XML namespace declaration item> of *XND* that contains an <XML namespace prefix> equivalent to *P*.

D) Let *NSURI* be the <XML namespace URI> contained in *XNDI*.

10.9 Determination of [namespace attributes] property

1. *Rationale: Replace reference to undefined term with reference to "[namespace name]".*

Replace General Rule 5) d) with:

5) ...

d) The [normalized value] property of $NAII_i$ is

Case:

i) If M_i is the zero-length string, then the [namespace name] property of *EII*.

ii) Otherwise,

Case:

1) If M_i is equivalent to *P*, then the [namespace name] property of *EII*.

2) Otherwise, the [namespace name] property of any XML attribute information item contained in the [attributes] property of *EII* whose [prefix] property is equivalent to M_i.

10.10 Determination of the [in-scope namespaces] property

1. *Rationale: Correct the algorithm for determining the [in-scope namespaces] property to produce correct XML namespace information items.*

Replace General Rule 6) c) i) 1) with:

6) ...

c) ...

i) ...

1) If LN_i is **'xmlns'**, then the [prefix] property of *NSI* has no value.

10.11 Determination of [whitespace element content] property

1. *Rationale: Delete unneeded subclause.*

   Delete the entire Subclause

10.13 Serialization of an XML value

1. *Rationale: Correct xml syntax.*

   Replace General Rule 2) b) ii) 1) E) with:

   2) ...

      b) ...

         ii) ...

            2) ...

               E) Case:

                  I) If *VA* is not the zero-length string, then let *CV1* be

                  CAST ('<?xml *VA CEC SA* ?>' AS *DT*)

                  II) Otherwise, it is implementation-defined whether *CV1* is the zero-length string or *CV1* is

                  CAST ('<?xml version="1.0" encoding="UTF-*x*" ?>' AS *DT*)

                  where *x* is either 8 or 16, depending on whether *CS* is UTF8 or UTF16, respectively.

2. *Rationale: Cater for the possibility that the <data type> specified in <XML serialize> is not capable of holding the serialized representation of the <XML value expression> also specified in <XML serialize>.*

   Replace General Rule 2) b) v) with:

   2) ...

      b) ...

         v) The serialization of *V* is the result of the assignment of

         CV1* || *CV2

         to a target of type *DT* according to the rules of Subclause 10.16, “Store assignment”.

10.14 Parsing a character string as an XML value

1. *Rationale: Align the definition of a textual XML content with corresponding definitions in [XML].*

   Replace General Rule 3) with:

   3) A character string is called a *textual XML content* if any of the following is true:

      a) The character string is a textual XML document.

      b) The character string conforms to the definition of a well-formed external parsed entity as defined in [XML], as modified by [Namespaces].

2. *Rationale: Correctly parse a character string in the presence of the <XML whitespace option>.*

   Replace the lead text of General Rule 5) a) with:

   5) ...

      a) Instead of an XML document information item, an XML root information item *XRII* is produced, as follows:

3. *Rationale: Correctly parse a character string in the presence of the <XML whitespace option>.*

   Replace General Rule 6) with:

   6) 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) [local name] property is “space”.

            2) [namespace name] property is `http://www.w3.org/XML/1998/namespace`.

            3) [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) [local name] property is “space”.

            2) [namespace name] property is `http://www.w3.org/XML/1998/namespace`.

            3) [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:

1) [local name] property is “space”.

2) [namespace name] property is `http://www.w3.org/XML/1998/namespace`.

3) [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 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 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, 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 27.1 — Thus the list XII_j, ... , XII_k is the maximal sublist of XII_1, ... , XII_N containing 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, XII_q is an XML character information item whose [character code] property is a whitespace character, then XII_q is marked for removal from *C*.

iii) For every i between 1 (one) and N, if XII_i is marked for removal from *C*, then XII_i is removed from *C*.

c) Let *N* be the cardinality of the [children] property of *XRII*. Let 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 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, 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.

3) If for all q between j and k, XII_q is an XML character information item whose [character code] property is a whitespace character, then XII_q is marked for removal from *C*.

ii) For every i between 1 (one) and N, if XII_i is marked for removal from *C*, then XII_i is removed from *C*.

10.15 Retrieval assignment

1. *Rationale: Add missing Subclause.*

   Insert the following Subclase:

10.15 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 values to host parameters or host variables).

Syntax Rules

1) Replace SR 2) If *TD* is binary string, numeric, boolean, datetime, interval, XML, or a user-defined type, then either *SD* shall be assignable to *TD* or there shall exist an appropriate user-defined cast function *UDCF* from *SD* to *TD*.

Access Rules

No additional Access Rules.

General Rules

1) Augment GR 6) If the declared type of *T* is the XML type, then *T* is set to *V*.

Conformance Rules

No additional Conformance Rules.

10.16 Store assignment

1. *Rationale: Add missing Subclause.*

   Insert the following Subclase:

10.16 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) Replace SR 2) If *TD* is character string, binary string, numeric, boolean, datetime, interval, XML, or a user-defined type, then either *SD* shall be assignable to *TD* or there shall exist an appropriate user-defined cast function *UDCF* from *SD* to *TD*.

Access Rules

No additional Access Rules.

General Rules

1) Augment GR 2)b) If the declared type of *T* is the XML type, then *T* is set to *V*.

Conformance Rules

No additional Conformance Rules.

11 Additional common elements

11.3 <XML namespace declaration>

1. *Rationale: “Undeclaring” a namespace prefix by assigning an empty string to it is not supported by [Namespaces]. Thus, an <XML namespace declaration> should not allow it either.*

Replace the BNF production of <XML namespace declaration item> with:

```
<XML namespace declaration item> ::=
    <XML regular namespace declaration item>
  | <XML default namespace declaration item>

<XML regular namespace declaration item> ::=
    <XML namespace URI> AS <XML namespace prefix>
```

2. *Rationale: Clarify that an <XML namespace prefix> contained in an <XML namespace declaration> cannot be equivalent to certain reserved namespace prefixes. Clarify that an <XML namespace URI> contained in an <XML namespace declaration> cannot be identical to certain reserved namespace URIs.*

   Insert the following Syntax Rules:

   3.1) No <XML namespace prefix> shall be equivalent to `'xml'` or `'xmlns'`.

   3.2) No <XML namespace URI> shall be identical as defined in [Namespaces] to `http://www.w3.org/2000/xmlns/` or `http://www.w3.org/XML/1998/namespace`.

3. *Rationale: "Undeclaring" a namespace prefix by assigning an empty string to it is not supported by [Namespaces]. Thus, an <XML namespace declaration> should not allow it either.*

   Insert the following Syntax Rule:

   3.3) The value of an <XML namespace URI> contained in an <XML regular namespace declaration item> shall not be a zero-length string.

18 Embedded SQL

18.1 <embedded SQL host program>

1. *Rationale: Editorial.*

   Replace Syntax Rule 6) b) with:

   6) ...

   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.

2. *Rationale: Use defined symbol.*

   Replace Syntax Rule 6) e) vii) 1) with:

   6) ...

   e) ...

   vii) ...

   1) If $XCSIO_l$ is UTF16, then let $IOBOM_l$ be `U&'\FEFF'`.

3. *Rationale: Replace incorrect semicolons with full-stops.*

Replace Syntax Rule 7) with:

7) Insert before SR 21)l)i)7)B) If HV_i identifies an XML VARCHAR host variable or an XML CLOB host variable, then

Case:

a) If P_i is an input host parameter, then let $PXNI_j$, 1(one) ≤ j ≤ 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) ≤ k ≤ 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) ≤ l ≤ 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 $XVIO_l$.

Annex B

(informative)

Implementation-defined elements

1. *Rationale: Use correct Unicode terminology.*

Replace list item 15 with:

15) Subclause 9.17, “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.

2. *Rationale: It is implementation-defined whether an XML declaration is produced when it is optional in [XML].*

Insert the following subitem:

16) ...

b) If the character set is either UTF8 or UTF16 and the [version] property is “1.0” and the [standalone] property is “no value”, then it is implementation-defined whether the output contains an XML declaration of the form **`'<?xml version="1.0" encoding="UTF-`*`x`*`"`**

`?>'` , where *x* is either 8 or 16, depending on whether the character set is UTF8 or UTF16, respectively.