A Conceptual Schema Based XML Schema with Integrity Constraints Checking

Oviliani Yenty Yuliana[1], Suphamit Chittayasothorn[2]
[1]*Department of Informatics Engineering, Petra Christian University, Surabaya, Indonesia*
[2]*Faculty of Engineering, King Mongkut's Institute of Technology, Bangkok, Thailand*
ovi@peter.petra.ac.id, suphamit@kmitl.ac.th

Abstract

The more popular XML for exchanging and representing information on Web, the more important Flat XML (XML) and intelligent editors become. For data exchanging, an XML Data with an XML Schema and integrity constraints are preferred. We employ an Object-Role Modeling (ORM) for enriching the XML Schema constraints and providing better validation the XML Data. An XML conceptual schema is presented using the ORM conceptual model. Editor Meta Tables are generated from the conceptual schema diagram and are populated. A User XML Schema base on the information in the Editor Meta Tables is generated. However, W3C XML Schema language does not support all of the ORM constraints. Therefore, we propose an Editor XML Schema and an Editor XML Data to cover unsupported the ORM constraints. We propose the algorithms for defining constraint in the User XML Schema and extending validity constraint checking. Finally, XQuery is used for extending validity checking.

1. Introduction

XML is gradually accepted as a standard for representing, accessing, and exchanging data in internet applications. It poses many new challenges to XML storages or XML repositories [1]. Therefore, it increases the needs for storing data efficiently in a Flat XML format which is validated by an XML Schema. Additionally, intelligent editors become increasingly important. In order to enrich XML Schema constraints, we employ the ORM as the conceptual schema in our research. There are several reasons [2]: (1) the ORM models and the ORM queries are more stable, (2) the ORM models may be conveniently populated with multiple instances, and (3) the ORM is more uniform.

There are several XML Schema researches with ORM approach [3-5]. Mapping the ORM conceptual schema into XML Schema was proposed by [3, 4]. However, they did not cover up all of the ORM constraints explicitly. One of the reasons is the W3C schema language is not sufficient for defining all of the ORM constraints. The XML Schema enclosed over all the ORM constraints gives a credit to the normalized XML Schemas techniques that developed by [5]. Other researchers [6-8] also concerned with XML constraints. XML constraints with a relational schema approach for a DTD is studied by [7, 8]. Additionally, [4] transformed the ORM to Object Database Schema and [7] captured XML constraints with SQL schema. Furthermore, [9] explained XQuery and XML Schema can serve as an excellent vehicle for data and metadata integration. So far, mapping the fully ORM constraints into the XML Schema constraints and using the XQuery for checking validity constraint are still not addressed.

This paper is organized as follows. Section 2 overviews the ORM constraints. In section 3, we present the XML Schema constraints and the XQuery features. Section 4 discusses a mapping of the ORM conceptual schema into the XML Schema and demonstrates an extending validity constraints checking. Finally, section 5 concludes the paper.

2. ORM constraints

The ORM is a primarily method for modeling and querying information system at a conceptual level [2]. We modified the university case study conceptual schema that proposed by [2] as shown in Figure 1. The conceptual schema specifies the information structure of the application: stored fact types, constraints, and derivation rules [10]. This paper concerns with constraints and derivation rules. Constraints are also known as validation rules or integrity rules. A database is said to have integrity when it is consistent with the universe of discourse being modeled. Although most relevant constraints can be neatly represented on conceptual schema diagrams, some constrains (e.g. dynamic constraints) need to be represented in other ways (e.g. by logical formula or program code). Derivation rules provides a list of functions, operators and rules that may be used to drive information not explicitly stored in the database. These may involve mathematical calculation or logical inference. There are seven constraints in the ORM, i.e. uniqueness, mandatory role, value, comparison, subtype; occurrence frequency and ring.

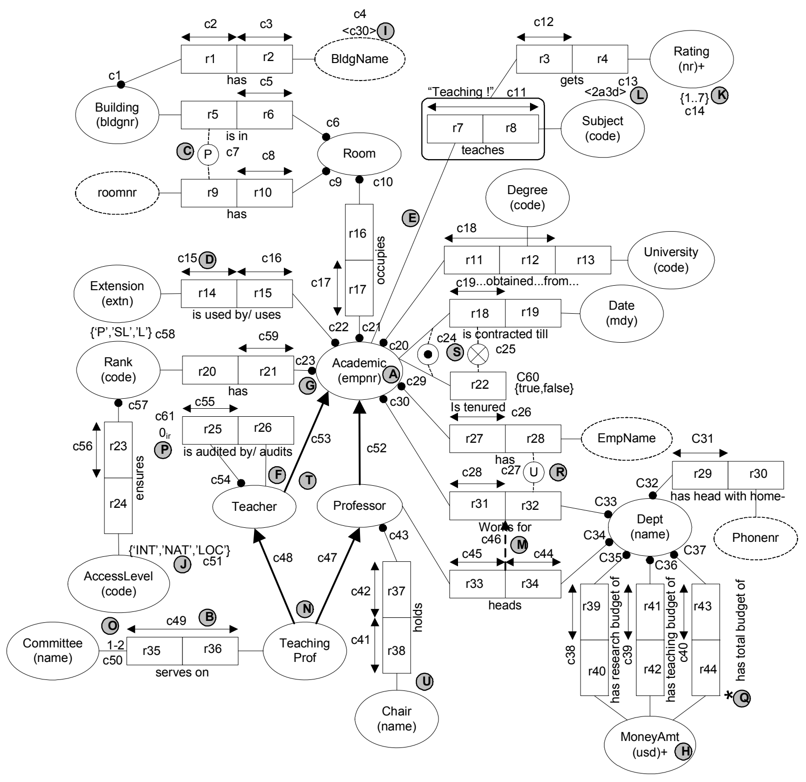

Figure. 1. The Modified ORM Conceptual Schema Diagram for University Case Study

Uniqueness constraint (UC) is used for restricting repetition in a role or a role sequence spanned by a constraint. There are two kinds UC, i.e.: an internal (UCi) and an external (UCe). UCi applies to one or more roles of a single predicate (one-to-one, many-to-one, one-to-many, or many-to-many). Ⓓ and Ⓑ in Figure 1 respectively represent the one-to-one and the many-to-many UCi. UCe is shown by a connecting two or more roles from different predicated. There are two kinds UCe, i.e.: uniqueness (UCeU) and uniqueness constraint primary (UCeP). For instance, Ⓡ and Ⓒ in Figure 1 respectively represent the UCeU and the UCeP. Another constraint that relates with UC is referential integrity constraint. It restricts foreign key to match the value of some primary key.

A role r is mandatory (MR) for an object type A iff, each member of pop(A) is known to play r; otherwise the role is optional. A mandatory role is indicated by a large dot where the role connects to the object type. For example, see a mandatory (Ⓖ) and an optional (Ⓕ) role constraints in Figure 1. Some entity can be a member of two entity types or two different types have the same unit or dimension, therefore combine the entity types into one. Furthermore, if a fact type is arithmetically derivable from others, so provide a derivation rule. If the fact type leaves on the diagram, it mask with "*". For example, see the MoneyAmt combined entity type (Ⓗ) in Figure 1.

Value constraints specify the members of a value type. It may provide a full listing (FL) or an enumeration of all the value, e.g. {'INT', 'NAT', 'LOC'} (Ⓙ). It may specify a subrange definition (SD), e.g. {1..7} (Ⓚ). Or it may indicate a format pattern (FP), e.g. <C30> (Ⓘ) allows any string of up to 30 characters, and <aaddd> or <2a3d> (Ⓛ) requires two letters follows by three digits.

Comparison constraints restrict a way of population a role, a role sequence, or a population of another. Let rs1 and rs2 be role sequence (of one or more roles) played by compatible object type. A subset constraint (SC) from rs1 to rs2 is denoted by a dotted arrow (Ⓜ), indicating pop(rs1) $\subseteq$ pop(rs2). An equality constraint (EqC) is equivalent to SC in both directions, and is shown by a dotted arrow with two heads, demanding that pop(rs1) = pop(rs2). An exclusion constraint (ExC) among two or more role sequences is shown by connecting them to $\otimes$ (Ⓢ) with dotted lines: this means their populations must be disjoint, indicating pop(rs1) $\cap$ pop(rs2) = {} and no a plays both r1 and r2.

An object type A is a subtype (SO) of B iff (A $\neq$ B and) for each database state, pop(A) $\subseteq$ pop(B). The SO is shown by a solid arrow from A to B (Ⓝ) in Figure 1. If A is a SO B, and B is a SO of C then it is transitively implied that A is a SO of C; such indirect SO links should not be displayed.

An occurrence frequency constraint (OF) indicates that an entry in a column (or column combination) must occur there exactly n times (n), at most n times (1-n), at least n times ($\geq$n), or at least n and at most m times (n-m). For instance, see Ⓞ in Figure 1.

A ring constraint may apply only to a pair of roles played by the same (or a compatible) object type. The role pair may form a binary predicate or be embedded in a longer predicate. Let R be the relation type comprising the role pair. R is reflexive (over its population) iff for all x playing either role, xRx. R is symmetric (Rsym) iff for all x, y, xRy $\rightarrow$ yRx. R is transitive iff for all x, y, z, xRy and yRz $\rightarrow$ xRz. These positive properties tend to be used for derivation rather than as constraints. The following negative properties may be marked as ring constraints next to the role pair (or role connector in embedded cases). R is irreflexive (^{o}ir, Rir) iff for all x, ~xRx. For example, see Rir (Ⓟ) in Figure 1. R is asymmetric (^{o}as, Ras) iff for all x, y, xRy $\rightarrow$ ~yRx. R is antisymmetric (^{o}ans, Rans) iff for all x, y, x $\neq$ y & xRy $\rightarrow$ ~yRx. R is intransitive (^{o}it, Rit) iff for all x, y, z, xRy & yRz $\rightarrow$ ~xRz. Ras and Rit each imply Rir. The exclusion implies Ras (and Rir). A recursive ring constraint which may be difficult to enforce is: acyclicity (^{o}ac, Rac).

3. XML schema constraints and xquery

3.1. XML schema constraints

W3C [11-13] define XML Schema constraints language for identities, occurrences, global elements for data types, and facets. **The XML Schema identity constraints** are defined by unique and/or key elements. The unique constraints definition asserts uniqueness, with respect to the content identified by {selector} of the tuples resulting from the evaluation of the {fields} XPath expression(s). The key constraint definition asserts uniqueness as for unique. Furthermore, the key asserts that all selected content actually has such tuples. To enforce the constraints use key and keyref elements. The keyref constraint definition asserts a correspondence, with respect to the content identified by {selector}, of the tuples resulting from evaluation of the {fields} XPath expression(s), with those of the {referenced key}.

The XML Schema occurrence constraints are declared by minOccurs and maxOccurs attributes. If minOccurs and maxOccurs attributes are omitted, the element must appear exactly one. For a Flat XML Schema, maxOccurs is one and minOccurs can be 0 or 1. MinOccurs=”0” and minOccurs=”1” are used to represent optional and mandatory occurrence respectively. In addition, nillable=”true” can be used for representing optional occurrence.

The XML Schema facet constraints are a value space defining, such as length, minLength, maxLength, pattern, enumeration. Length is the number of units of length. MinLength is the minimum number of units of length. MaxLength is the maximum number of units of length. The value of length, minLength, and maxLength must be a nonNegativeInteger. Pattern is a constraint on the value space of a datatype which is achieved by constraining the lexical space to literals which match a specific pattern. The value of pattern must be a regular expression. Enumeration constrains the value space to a specified set of values.

3.2. XQuery

XQuery is a declarative language, currently being developed by W3C [14-16]. It has been designed to query and transform XML data. The XQuery main module comprises a prolog and a query expression. The prolog can consist of several statements, such as namespace declarations, schema import statements, and/or function declarations, that determine the context in which the query expression is to be evaluated. A function may be either user-defined, with its implementation provided by the user in the form of an XQuery expression, or externally-defined, in which case the function’s implementation is provided by some implementation-defined external mechanism. XQuery also defines a notion of reusable library modules that consist of a prolog preceded by a module declaration.

4. The mapping of an ORM conceptual schema to an XML schema

4.1. Conceptual framework

We propose a conceptual framework for creating the Editor Meta Tables from the User Conceptual Schema, mapping the Editor Meta Tables into the User XML Schema and the Editor XML Schema, and checking validity constraints in the User XML Data base on the information in Editor XML Data. The framework is shown in Figure 2 and the Editor Meta Tables relational schema is shown Figure 3.

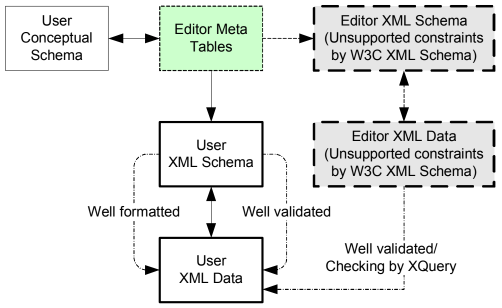
User Conceptual Schema
Editor Meta Tables
Editor XML Schema (Unsupported constraints by W3C XML Schema)
User XML Schema
Editor XML Data (Unsupported constraints by W3C XML Schema)
Well formatted
Well validated
User XML Data
Well validated/ Checking by XQuery

Figure 2. Mapping and checking constraints conceptual framework

Constraint (<u>ConstraintNr</u>, ConstKindCode)
ConstraintValue (<u>ConstraintNr</u>, Value)
ObjectPredicate (<u>ObjectTypeName</u>, <u>RoleNr</u>)
ObjectType (<u>ObjectTypeName</u>, OTKindName, RefModeName)
Role (<u>RoleNo</u>, ObjectTypeName, PredicateName, PositionNr)
RoleConstraint (<u>RoleNr</u>, <u>ConstraintNr</u>)
RoleType (<u>RoleNo</u>, <u>RoleType</u>)
SubtypingObject (<u>SubType</u>, <u>SuperType</u>)

Figure 3. Relational schema editor meta tables

The algorithm for creating, mapping, and checking:
Step 1: Editor Meta Tables is populated by User Conceptual Schema (e.g. see Figure 1).
Step 2: Uniqueness (UCi, UCeU, and UCeP), mandatory role (MR and DR), and value (FL, SD, FP) ORM constraints in Editor Meta Tables are mapping into identity, occurrence, and facet User XML Schema constraints respectively. For further discussion see section 4.2.
Step 3: W3C does not support schema languages for defining comparison (SC, EqC, and ExC), subtype (SO); occurrence frequency (OF), and ring (Rsym, Rir, Ras, Rans, Rac, and Rit) ORM constraints. Therefore, map the Relational Schema Editor Meta Tables into the Editor XML Schema and create the Editor XML Data base on the Editor XML Schema. Furthermore, populate the Editor XML Data with data form Editor Meta Tables which constraint kind code (ConstKindCode) are SC, EqC, ExC, SO, OF, Rsym, Rir, Ras, Rans, Rac, and Rit.
Step 4: The User XML Schema is used as the structure for creating and modifying the User XML Data. Moreover, the User XML Schema is paid for checking well formatted and well validated the User XML Data.
Step 5: The Editor XML Schema is used for modifying and validating the Editor XML Data.
Step 6: Because of not all of the ORM conceptual schemas are mapping into User XML Schemas, the Editor XML Data is used for fully validate User XML Data. For further discussion see section 4.3. As an alternative, we proposed XQuery for validity checking the User XML Data.

4.2. The defining constraints in user XML schema

The mapping from the ORM conceptual schema into the XML Schema i.e.: generating a type definition for each ORM object type, building a complex type definition for each major fact type grouping, and creating a root element for the whole schema have already studied by [3, 4]. For that reason, in this section we only concern for defining XML Schema constraint base on the Meta Tables contents. The algorithm for defining constraints in the User XML Schema:
Step 1: Define a key element for every entity type which OTKindName is Entity and ConstKindCode is UCi. The reference mode (RefModeName) of the OTKindName becomes a field xpath attribute. Additionally, define a key element for every a compound UCi. RefModeName that play on roles in the predicate become field xpath attributes. Finally, define a key element for every UCeP. The RefModeName or value types (ObjectTypeName) that play in the roles which are related with UCeP becomes field xpath attributes.
Step 2: Define a unique element for each UCi ConstKindCode (one-to-one). RefModeName or ObjectTypeName which plays in UCi optional role becomes field xpath attribute. Define a unique element for each UCeU as well. RefModeName or ObjectTypeName which play in the roles that are connected by UCeU becomes field xpath attributes.
Step 3: Define a keyref element for each a referential integrity constraint.
Step 4: Declare attribute minOccurs = ”0” or nillable = ”true” for each optional role constraints.
Step 5: Define a facet for every value constraint. Define enumeration for every ConstKindCode is FL. Moreover, define a minInclusive, a maxInclusive, a minExclusive, or a maxExclusive facet for each

ConstKindCode is SD. Furthermore, define a maxLength, a pattern, or a Length facet for every ConstKindCode is FP.

4.3. The extending validity constraints checking

The Editor XML Data can be exploited for extending validity constraints checking that are not supported by W3C schema language. The Editor XML Data instances are used for validity checking the constraints of: SC, EqC, ExC, SO, OF, Rir, Rit, Rac, Ras, Rans, and Rsym. The algorithm for checking validity constraints:

Step 1: If "SC" is found at ConstKindCode element in Constraint complexType then check subset constraint validity. By the content of ConstraintNr elements find all of the rule number in RoleConstraint complexType. Along with the rule number find all of ObjectTypeName in Role complexType and the role types in RoleType complexType instances, i.e. subset and superset. The instances in subset predicate must be a subset of the instances in superset predicate.

Step 2: If "EqC" exists at ConstKindCode element in Constraint complexType then check equality constraint validity. The process is similar with the subset constraint but without role type checking in RoleType complexType because the subset constraints in both directions. It means the instances in the related predicate are equal.

Step 3: If "ExC" presents at ConstKindCode element in Constraint complexType then check exclusion constraint validity. Base on the content of ConstraintNr element. All of the role numbers can be found in RoleConstraint complexType. Through the role numbers, all of the ObjectTypeName elements can be retrieved in Role complexType. Only one of the ObjectTypeName elements must have an instance in User XML Data.

Step 4: If "SO" is found at ConstKindCode elements in Constraint complexType then check subtype constraints validity. Base on constraint numbers from Contraint complexType, find all of the role number instance in RoleConstraint complexType. Than by the role numbers find all of object type name in Role complexType. Furthermore, to know the type of object, i.e. subtype or supertype, retrieve the object type name. Every subtype become an option element in the supertype complexType and need additional element to decide what element should have content instance. The constraint checks the option element instance.

Step 5: If "OF" exist at ConstKindCode elements in Constraint complexType then check occurrence frequency validity. By the content of ConstraintNr element, the constraint value can be found in ConstraintValue complexType and the role number can be found at RoleNr in RoleConstraint complexType. The constraint value can be used for restricting the occurrence. It could be n, 1-n, ≥n or n-m for exactly n times, at most n times, at least n times or at least n and at most m times respectively. In addition, the role number can be used to getting the object type name which instances occurrence will be checked in User XML Data.

Step 6: If "Rir", "Rit", "Rac", "Ras", Rans", or "Rsym" is found at ConstKindCode elements in Constraint complexType then check ring constraints validity. By the constraint number find the role number in RoleConstraint complexType to know the object type name in Role complexType. The ring constraints are implemented by defining another element from the element as a key element. The validities checking depend on the ring type. Rir checks for iff for all x, ~xRx validity. Rit checks validity iff for all x, y, z, xRy & yRz → ~xRz. Ras checks validity iff for all x, y, xRy → ~yRx. Rans checks validity iff for all x, y, x ≠ y & xRy → ~yRx.

As an alternative, this paper demonstrates XQuery for checking validity constraints. For instance, the first XQuery for checking invalid exclusion constraint is shown in Figure 4. The second XQuery for checking invalid irreflexive ring constraint is shown in Figure 5. The third XQuery for checking the invalid occurrence frequency constraint is shown in Figure 6. The fourth XQuery for validity chair is shown in Figure 7. The last XQuery for validity subset checking is shown in Figure 8.

```
for $a in doc("XMLFile ORM.xml")//Academic
where ($a/tenured) and
    (fn:year-from-dateTime($a/enddate)>0)
return
<InvalidEndDate>
        {$a/empnr}
        {$a/tenured}
        {$a/enddate}
</InvalidEndDate>
```

Figure 4. Exclusion constraint checking XQuery

```
for $e in doc("XMLFile ORM.xml")/ORM/Academic
where $e/empnr=$e/auditor
return
    <InvalidIrreflexive>
        {$e/empnr}
        {$e/empname}
        {$e/auditor}
     </InvalidIrreflexive>
```

Figure 5. Ring constraint checking XQuery

```
for $j in fn:distinct-values(
    fn:doc("XMLFile ORM.xml")/ORM/CteeMember/committee)
let $p := fn:doc("XMLFile ORM.xml")/ORM/CteeMember
   [committee = $j]
```

```
return
  if (fn:exists($p) and count($p/committee)>2) then
    <InvalidMaxOccurece>
      <committee>{$j}</committee>
  <Coutcommittee>{fn:count($p/committee)}</Coutcommittee>
    </InvalidMaxOccurece>
  else ()
```

Figure 6. Occurrence frequency constraint checking XQuery

```
for $d in doc("XMLFile ORM.xml")/ORM/Academic
where not ($d/rank="P") and not ($d/chair="")
return
    <Invalidchair>
        {$d/rank}
        {$d/chair}
     </ Invalidchair >
```

Figure 7. Valid chair checking XQuery

```
for $d in doc("XMLFile ORM.xml")//Department,
    $a in doc("XMLFile ORM.xml")//Academic
where ($d/deptname=$a/deptname)
    and ($d/headempnr=$a/empnr)
    and (($a/tenured=false) or (not($a/rank="P")))
order by $d/extn
return
    <InvalidSubSet>
        {$d/deptname}
        {$d/headempnr}
        {$a/empname}
     </InvalidSubSet>
```

Figure 8. Invalid subset checking XQuery

5. Conclusion

The ORM constraints that could be mapped to the XML Schema constraints are uniqueness, mandatory-optional, and value constraints. However, the other ORM constraints i.e. subset, equality, exclusion, subtyping, occurrence frequencies, and ring are not covered by W3C XML Schema languages. To support all of the ORM constraints we generate Editor Meta Table. The User XML Schema and the Editor XML Schema are created base on information in Editor Meta Tables. The editors use the Editor XML Data and the User XML Schema for validity checking the User XML Data. In our work, we also demonstrate the capability of XQuery as alternative for checking validity constraints. As result, the XML Schema formalism and XQuery can be used for better well-formed and well-validated Flat XML.

6. References

[1] Ferreira R., Jaão M., Ramiro M., and Marta P., "XML Based Metadata Repository for Information System", Proc IEEE Artificial Intelligence, Portuguese, 2005, pp. 205-213.

[2] Halpin T., *Object-Role Modelling (ORM/NIAM), Handbook on Architectures of Information System*, Springer-Verlag, Berlin, 1998.

[3] Bird L., Andrew G., and Terry H., "Object Role Modelling and XML-Schema", Proc. 19th International Conf. on Conceptual Modelling, USA, 2000, pp. 1-14.

[4] Chankuang N. and Suphamit C., "An Object and XML Database Schemas Design Tool", Proc. IEEE Conf. on ITCC, USA, 2004, pp. 421-424.

[5] Yuliana O. and Suphamit C., "XML Schema Re-Engineering Using a Conceptual Schema Approach", Proc. IEEE Conf. on ITCC, USA, 2005, pp. 255-260.

[6] Fan W., "XML Constraints: Specification, Analysis, and Applications", Proc. 16th IEEE Workshop on DESA.

[7] Liu Y., Hao Z., and Yi W., "Capturing XML Constraints with Relational Schema", Proc. 4th IEEE Conf. on CIT, 2004, pp. 309-314.

[8] Liu Y., Hao Z., and Yi W., "XML Constraints Preservation in Relational Schema", Proc. IEEE Conf. on CEC, 2004, pp. 188-195.

[9] Reveliotis P. and Michael C, "Your Enterprise on XQuery and XML Schema: XML-based Data and Metadata Integration", Proc. 22nd IEEE Conf. on DEW.

[10] Halpin T., *Conceptual Schema And Relational Database Design* Prentice Hall, Australia, 1995.

[11] W3C, XML Schema Part 0: Primer Second Edition, http://www.w3.org/TR/xmlschema-0/, 2004.

[12] W3C, XML Schema Part 1: Structures Second Edition, http://www.w3.org/TR/xmlschema-1/, 2004.

[13] W3C, XML Schema Part 2: Datatypes Second Edition, http://www.w3.org/TR/xmlschema-2/, 2004.

[14] W3C, XML Query Use Cases, http://www.w3.org/TR/xquery-use-cases/, 2007.

[15] W3C, XQuery 1.0 and XPath 2.0 Functions and Operators, http://www.w3.org/TR/xpath-functions/, 2005.

[16] W3C, XQuery 1.0: An XML Query Language, http://www.w3.org/TR/xquery/, 2007.